Sol Terminal Computer

Sol Terminal Computer
Sol Terminal Computer

Sol Systems Manual

Processor Technology Corporation

Sol SYSTEMS MANUAL

Processor Technology

6200 Hollis Street
Emeryville, CA. 94608
Phone: (415) 652-8080

Third Printing, June, 1977
Manual No. 106000

PREFACE

This new edition of the Sol Systems Manual contains many revisions and additions. Its release coincides with the release of a new "2708" Personality Module, and the Revision E version of the main circuit board: Sol-PC. The new "Sol-PC Rev E" has several improvements: resistors have been added which increase the reliability of the cassette motor relays, jumper options have been added, and traces moved to improve performance. Many improvements which had been accumulating as update information have been integrated into the text. Section VII, Operating Procedures, and Appendix 5, IC Pin Configurations, are now included. A subsection, Modification for 625 Line Video, has been added. If your copy is missing Section VIII, Theory of Operation, it will be available soon. New divider pages with plastic-coated tabs are included to make it easier to flip to frequently referenced sections.

Much effort has gone towards making this manual complete and accurate. The process of updating and revision always continues, however, and we invite your input. If you should find an error, or have suggestions for improving any of our documentation, please submit your suggestions in writing to our Technical Documentation Department, and they will be given thorough consideration.

The three-ring binder you are holding, is an "easel" binder. The cover is hinged from side to side, as well as down the binding, so that it may form its own "easel" stand. To use this feature, lay the manual open on a table. Bend the full width of the manual along the creased hinge, until a resistance to further bending is felt. Then set the manual up on the table, with the bottom of the pages down against the table, and the top inclining away from you. It is supported from falling by the portion of the binder you have bent back. In this position your hands are free for building, making measurements, or troubleshooting.

The first part of this manual you should read is at the very end: the Updates section. Integrate this information into your manual before you begin.

CONTENTS OUTLINE

Detailed contents precede each section.

LIST OF ILLUSTRATIONS

1.1 INTRODUCTION

This manual supplies the information needed to assemble, test and use the Sol-PC Single Board Terminal Computer. We suggest that you first scan the entire manual before starting assembly. Then make sure you have all the parts and components listed in the "Parts List" (Table 3-1) in Section III. When assembling the module, follow the instructions in the order given.

Should you encounter any problem during assembly, call on us for help if necessary. If your completed module does not work properly, recheck your assembly step by step. Most problems stem from poor soldering, backward installed components, and/or installing the wrong component. Once you are satisfied that the module is correctly assembled, feel free to ask for our help.

1.2 GENERAL INFORMATION

1.2.1 Sol-PC Description

The Sol-PC is a single board microcomputer/terminal built around an 8080 microprocessor. Support circuitry permits full implementation of every 8080 function.

It features both parallel and serial communications interfaces, a keyboard interface, an audio cassette interface, a video display generator, 1024 8-bit words of system RAM (random access memory), 1024 8-bit words of display RAM, and a plug-in personality module with up to 2048 bytes of ROM (read only memory) stored program, and bus compatibility with all Processor Technology hardware and firmware products. Power requirements for the Sol-PC are +5 V dc ±5% at 2.5 A, +12 V dc ±5% at 150 mA and -12 V dc ±5% at 200 mA.

Parallel interfacing is eight bits each for input and output plus control handshaking signals, and the output bus is tristated TTL for bidirectional interfaces. The serial interface circuit includes both asynchronous RS-232 and 20 mA current loop provisions, 75 to 9600 baud (switch selectable).

Seven-level ASCII encoded, TTL keyboard interfacing requires a 2 to 10 usec strobe pulse after data is stable. The dual rate, 300 or 1200 bps (bits per second), audio cassette interface is program controlled and self clocking with phase-lock loop. It includes automatic level control in both the record and playback modes. Recording is CUTS/Byte standard compatible, asynchronously Manchester coded at 1200/2400 Hz or 600/1200 Hz.

The video display circuitry generates sixteen 64 character lines from data stored in an on-card 1024 8-bit word display RAM. Alphanumeric and control characters (the full 128 upper and lower case plus control ASCII character set) are displayed black on white or reverse (switch selectable). Solid video inversion cursors, with switch selectable blink, are programmable. The display output is standard EIA, 1.0 to 2.5 V p-p with composite negative sync, with a nominal bandwidth of 7 MHz. It can consequently be used to drive any standard video monitor. (A monochrome TV, converted for video input, can also be used. See Appendix VI.)

Included on the card are 1024 words of static, low power system RAM capable of full speed operation and a plug-in personality module which contains the software control program. Three personality modules are available for Sol:

> CONSOL^{TM}--allows simple terminal operations plus direct control of the basic computer functions for entering or examining data in any memory location, or executing a program stored at a known location in memory.
>
> SOLED^{TM}--allows advanced terminal operations with CONSOL plus screen, file and cassette tape editing/transmission operations.
>
> SOLOS^{TM}--allows full stand-alone terminal-computer operation.

1.2.2 Receiving Inspection

When your kit arrives, examine the shipping container for signs of possible damage to the contents during transit. Then inspect the contents for damage. (We suggest you save the shipping materials for use in returning the kit to Processor Technology should it become necessary to do so.) If your Sol-PC kit is damaged, please write us at once describing the condition so that we can take appropriate action.

1.2.3 Warranty Information

In brief, parts which fail because of defects in materials or workmanship are replaced at no charge for 3 months for kits, and one year for assembled products, following the date of purchase. Also, products assembled by the buyer are warranted for a period of 3 months after the date of purchase; factory assembled units carry a one year warranty. Refer to Appendix I for the complete "Statement of Warranty".

1.2.4 Replacement Parts

Order replacement parts by component nomenclature (DM8131 IC or 1N2222 diode, for example) and/or a complete description (680 ohm, ¼ watt, 5% carbon resistor, for example).

1.2.5 Factory Service

In addition to in-warranty service, Processor Technology also provides factory repair service on out-of-warranty Processor Technology products. Before returning the unit to us, first obtain our authorization to do so by writing us a letter describing the problem. After you receive our authorization to return the unit, proceed as follows:

1. Write a description of the problem.

2. Pack the unit with the description in a container suitable to the method of shipment.

3. Ship <u>prepaid</u> to Processor Technology Corporation, 6200 Hollis Street, Emeryville, CA 94608.

Your unit will be repaired as soon as possible after receipt and return shipped to you prepaid. (Factory service charges will not exceed $20.00 without prior notification and your approval.)

II Sol POWER SUPPLY ASSEMBLY and TEST

2.1 INTRODUCTION

The Sol power supply consists of a regulator board plus additional chassis-mounted components. This section covers assembly and test of the complete power supply.

2.2 PARTS AND COMPONENTS

2.2.1 Sol Regulator (Sol-REG)

Check all parts and components against the appropriate "Parts List", Tables 2-1, 2-2 and 2-3. If you have difficulty in identifying any parts by sight, refer to Figure 3-1 on Page III-5 in Section III of this manual.

2.2.2 Power Supply Subchassis and Components

In addition to the Sol-REG, you will need the following parts and components supplied with the Sol Cabinet-Chassis Kit. Check these parts against the appropriate "Parts List(s)", Tables 6-1 and 6-2, in Section VI and separate them from the other cabinet-chassis parts.

Fan Closure Plate
Power Supply Subchassis (L-shaped)

4 each	4-40 x 3/16 Machine Screw
4 each	4-40 x 5/16 Machine Screw
4 each	4-40 Hex Nut
10 each	#4 Lockwasher
14 each	6-32 x ½ Machine Screw
14 each	6-32 Hex Nut
16 each	#6 Lockwasher
3 each	8-32 x ½ Machine Screw
3 each	8-32 Hex Nut
3 each	#8 Lockwasher
11 each	#6 x ¼ Sheet Metal Screw
1 each	#6 x 5/16 Sheet Metal Screw
2 each	#4 Solder Lug
2 each	¼" Spacer, 4-40 Tapped

Table 2-1. Sol Regulator Parts List.

INTEGRATED CIRCUITS**

1	1458 (U2)
1	7812 (U1)
1	7912 (U3)

TRANSISTORS

2	2N2222 (Q2 & 3)
1	TIP41 (Q1)

DIODES and RECTIFIERS

1	MDA101A (FWB2)
1	MDA970-1 (FWB1)
1	IR106B2 or MCR106-2 (SCR1)
2	1N4001 (D3 & 4)
1	1N4148 (D2)
1	1N5231B (D1)

RESISTORS

1	0.1	ohm, 3 watt, 5% or 5 watt, 5%
1	68	ohm, ¼ watt, 5%
1	330	ohm, ¼ watt, 5%
2	1 K	ohm, ¼ watt, 5%
4	10 K	ohm, ¼ watt, 5%
1	56 K	ohm, ¼ watt, 5%
1	1690	ohm, ¼ watt, 5%
1	4020	ohm, ¼ watt, 5%

CAPACITORS

2	.1	ufd, disc
3	15	ufd, tantalum dipped
2	2500	ufd, tubular electrolytic
1	*18,000	ufd, electrolytic

CABLE ASSEMBLIES

1	*Single wire, 3" (Fuse Holder to Power Switch)
1	*Single wire, 3¼" (Power Switch to Commoning Block)
1	Two wire, 10" (C8 to Regulator Board)

*Chassis-mounted component

**When identifying IC's, you can ignore prefix and suffix characters in the IC nomenclature since these vary with the manufacturer. For example a 1458CP, 1458CPI and MC1458N are all 1458 IC's. This applies to all Parts Lists in this manual.

Table 2-1. Sol Regulator Parts List (Continued).

MISCELLANEOUS	
1	Sol REG Circuit Board
1	Heat Sink, 690-220-P
1	Heat Sink, 203-AP
1	Heat Sink, aluminum
1	Package Heat Sink Compound
2	Coax Connector, female* (Video Output)
1	Coax Connector, male (Video Output Cable)
1	Coax Connector Adapter Sleeve (Video Output Cable)
1	*AC Receptacle, female
1	*Fuse Holder
1	*SPST Power Switch, pushbutton (S5)
1	AC Power Cord
2	*Commoning Blocks
1	*Clamp for C8, 1½"
4	Tie Wraps
3	Mica Insulators
1	4-40 x 7/16 screw
1	4-40 x 5/8 screw
2	4-40 Hex Nut
1	6-32 x ½ screw, metal
2	6-32 x ½ screw, Nylon
3	6-32 Hex Nut
5	#4 Lockwasher, internal tooth
1	Length Solder

*Chassis-mounted component

Table 2-2. Sol-10 Power Supply Parts List.

The Sol-10 Power Supply Kit includes all Sol-REG parts listed in Table 2-1 plus the following components:
1 *Power Transformer, T1 1 *Fuse, 3 amp Slo-Blo (F1)

*Chassis-mounted component

Table 2-3. Sol-20 Power Supply Parts List.

The Sol-20 Power Supply Kit includes all Sol-REG parts listed in Table 2-1 plus the following components:	
<u>RESISTORS</u> 1 *39 ohm, 2 watt, 5%	<u>CAPACITORS</u> 1 *54,000 ufd, electrolytic
<u>RECTIFIERS</u> 1 *MDA980-1 (FWB3)	<u>TRANSFORMERS</u> 1 *Power Transformer, T2
<u>MISCELLANEOUS</u>	
1 *Fan	1 5-wire Cable Assembly
1 *Fan Guard	1 *Clamp for C9, 2½"
1 *Fuse, 3 amp Slo-Blo	2 *#10 solder lug, internal tooth

*Chassis-mounted component

2.3 ASSEMBLY TIPS

2.3.1 Electrical

For the most part the assembly tips given in Paragraph 3.2 of Section III (Page III-1) apply to assembling the Sol regulator board and power supply.

In addition, scan Section II completely before you start to assemble the power supply.

2.3.2 Mechanical

1. If you do not have the proper screwdrivers (see Paragraph 2.5), we recommend that you buy them rather than using a knife point, a blade screwdriver on a Phillips screw, and other makeshift means. Proper screwdrivers minimize the chances of stripping threads, disfiguring screw heads and marring decorative surfaces.

2. To assure a correct fit and tight assembly, be sure you use the screws specified in the instructions.

3. Lockwashers are widely used in the power supply assembly so that screws will not loosen when subjected to stress or vibration. When a lockwasher is specified, do not omit it and make sure you install it correctly.

4. Some instructions call for prethreading holes. This is done to make assembly easier by giving you maximum working space for installing relatively hard-to-drive sheet metal screws. If you bypass prethreading instructions you will only make subsequent cabinet-chassis assembly more difficult.

To prethread a hole, insert specified screw in the hole and position it as straight as possible. While holding the screw in this position, drive it into the metal with the proper screwdriver. If started straight the screw will continue to go straight into the metal so that the head and sheet metal surfaces are in full contact.

5. The diameter of the shank (threaded portion) of a screw increases in relation to its number. For example, a 6-32 screw is larger in diameter than a 4-40 screw. Also, a #8 lockwasher is larger than a #4 lockwasher.

6. Heat sink compound is supplied with this kit in a small clear plastic package. It is a thick white substance which improves heat transfer between components and their heat sinks. To use the compound, pierce a small hole near the edge of the top surface of the plastic package, using a pin or sharp knife point. Squeezing the package will cause a small amount of the compound to ooze out

out of the hole, which may then be applied with a toothpick or small screwdriver blade. Spread a thin film of the compound on the mating surfaces of both the heat-generating component and the heat sink surface which it will contact. Then assemble as directed.

2.4 ASSEMBLY PRECAUTIONS

The precautions concerning soldering and the installation and removal of integrated circuits given in Paragraph 3.3 of Section III (Page III-6) also apply to assembling the Sol regulator board.

2.5 REQUIRED TOOLS, EQUIPMENT AND MATERIALS

The following tools, equipment and materials are recommended for assembling the Sol regulator board:

1. Needle nose pliers
2. Diagonal cutters
3. Sharp knife
4. Screwdriver, thin ¼" blade
5. Screwdriver, #2 Phillips
6. Controlled heat soldering iron, 25 watt
7. 60-40 rosin-core solder (supplied)
8. Volt-ohm meter
9. Ruler

2.6 ORIENTATION

2.6.1 Sol-REG PC Board

Location C5 (2500 ufd capacitor) will be located in the lower right-hand corner of the circuit board when locations SCR1, Q1 and FWB1 are positioned along the top of the board. In this position the component (front) side of the board is facing up and the horizontal legends will read from left to right; the other legends will read from bottom to top. Subsequent position references related to the Sol-REG board assume this orientation.

2.6.2 Fan Closure Plate

The large circular cutout will be located in the upper right quadrant of the plate when the heavy guage doubler plate is facing up. In this position the rectangular cutouts are on the left, the front side of the plate is facing down, the back side is facing up, and the small circular cutout is at the bottom. We suggest you label the two sides.

2.7 ASSEMBLY-TEST

NOTE: Instructions that apply only to the Sol-20 are preceded by an asterisk. Skip these instructions if you are assembling a Sol-10.

2.7.1 Fan Closure Plate Assembly

Refer to Assembly Drawings on Pages X-1 and 2 in Section X. (Figure 2-1 shows a completed fan closure plate assembly.)

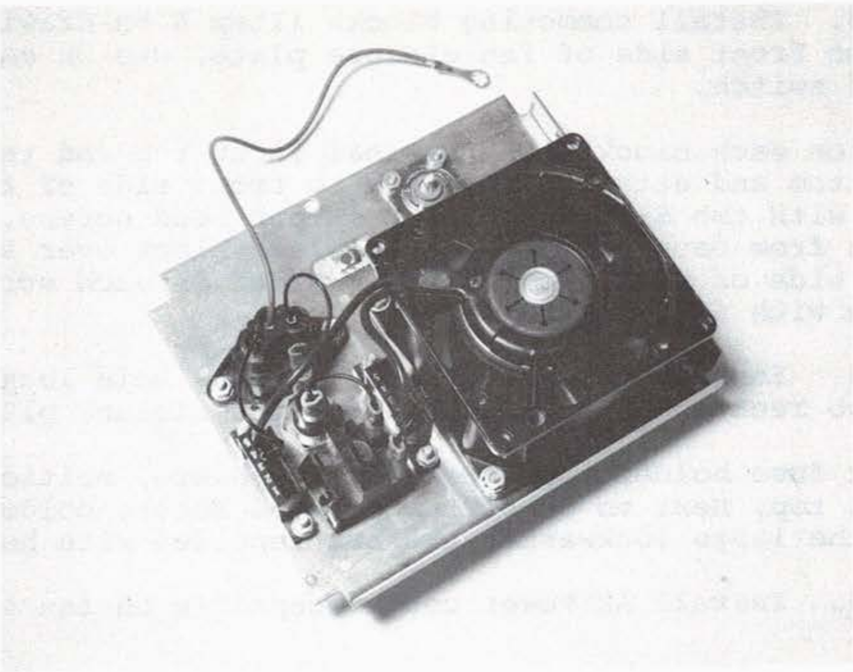


Figure 2-1. Sol-20 fan closure plate assembly. (Top of plate in foreground.)

*( ) Step 1. Mount cooling fan and guard to fan closure plate.

Insert four 6-32 x ½" binder or pan head screws from back side of fan closure plate. (Use the holes positioned in each quandrant of the large circular cutout.) Slip fan guard over screws on front side of plate. Position fan so that air flow will be from front to back side of plate and with its leads next to the rectangular cutouts in the place. Place #6 lockwasher on each screw and secure with 6-32 hex nut.

WARNING

FAILURE TO INSTALL FAN GUARD MAY RESULT IN DAMAGE TO THE Sol AND/OR PERSONAL INJURY.

( ) Step 2. Install power on-off switch in upper rectangular cutout in fan closure plate.

(Step 2 continued on Page II-8.)

Bend four retainer tabs on switch in and position switch with terminals facing front side of fan closure plate. Push switch unit from back side of plate through mounting hole and bend retainer tabs outward if needed to hold switch in place.

( ) Step 3. Install commoning blocks (Item 6 on drawing on Page X-1) on front side of fan closure plate, one on each side of on-off switch.

Position each block with terminal #1 at top and terminal #5 at bottom and attach each block to front side of fan closure plate with two 6-32 X ½ binder or pan head screws. Insert screws from back side of plate, place block over screws, on front side of plate, put #6 lockwasher on each screw and secure with 6-32 hex nut.

( ) Step 4. Install fuse holder in mounting hole located between the two rectangular cutouts in the fan closure plate.

Insert fuse holder from back side of plate, poition large tab at top, next to on-off switch, and secure holder to plate with the large lockwasher and nut supplied with holder.

( ) Step 5. Install AC Power cord receptacle on fan closure plate.

Position receptacle on front side of fan closure plate over the rectangular cutout below fuse holder. Orient receptacle with green lead at the botton and align the receptacle and closure plate mounting holes. Insert two 6-32 x ½ binder or pan head screws from back side of plate through each mounting hole, put #6 lockwasher on each screw and secure with 6-32 hex nut. Be sure receptacle is properly seated in cutout before tightening to avoid damage.

( ) Step 6. Install female coaxial connector on fan closure plate.

Insert connector from front side of plate so that the threaded end projects through to the back side. Then insert four 4-40 x 5/16 binder or pan head screws from back side of plate through the four connector and plate mounting holes. Place #4 lockwasher on each screw except the upper one which is closest to the AC receptacle. Secure with 4-40 hex nuts. (Leave upper nut closest to receptacle loose.)

( ) Step 7. Prepare RG59/U coaxial cable.

Cut a 13" piece of coaxial cable from that supplied with the Sol-PC kit. Strip away one inch of the outer insulation at both ends to expose shield. Unbraid shield at one end and twist it into a single lead. Do the same thing at the other end. Tin shield lead at each end and solder a #4 lug to each lead. Then remove ½" of the inner conductor insulation at both ends. (See Figure 2-2.)

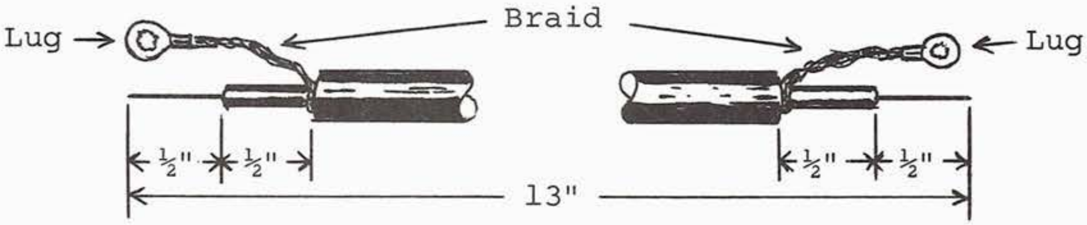


Figure 2-2. Coaxial cable preparation.

( ) Step 8. Connect coaxial cable to coaxial connector installed in Step 6.

Solder inner conductor on one end to the pin of the connector. Remove hex nut on upper connector mounting screw closest to AC receptacle, place lug (coaxial shield) on screw and reinstall hex nut.

( ) Step 9. Connect fan closure plate wiring.

( ) Install the 3" power switch-to-commoning block cable supplied with your Sol-REG kit. Connect the female spade lug end to the upper terminal of the on-off switch and the commoning block lug end to the #1 terminal of the commoning block closest to the fan. NOTE: To install commoning block lugs, position lug with its open side facing away from the terminal numbers on the block. Then gently push lug into appropriate terminal receptacle until it is fully seated.

( ) Install the 3¼" fuse holder-to-power switch cable supplied with your Sol-REG kit. (This cable has female spade lugs at both ends.) Connect one end to the bottom terminal of the on-off switch and the other to the longer male spade lug on the fuse holder.

( ) Connect the AC receptacle wire closest to the fan to the other fuse holder lug. NOTE: The green AC receptacle wire will be connected later.

( ) Connect other AC receptacle wire to terminal #4 on the commoning block furthest away from the fan.

*( ) Connect upper wire of fan cord to terminal #3 of the commoning block closest to fan.

*( ) Connect lower wire of fan cord to terminal #5 of commoning block furthest from fan.

( ) Put fan closure assembly aside.

2.7.2 Sol-REG Assembly and Test

Circuit references, values and outlines are printed on the component side of the board to assist in assembly.

( ) Step 10. Visually check Sol-REG board for solder bridges (shorts) between traces, broken traces and similar defects.

If visual inspection reveals any defects, return the board to Processor Technology for replacement. If the board is not defective, proceed to next paragraph.

( ) Step 11. Install the following resistors in the indicated locations. Bend leads to fit distance between mounting holes, insert leads, pull down snug to board, solder and trim.

	LOCATION	VALUE (ohms)	COLOR CODE
( )	R1	.1, 3 watt	none
( )	R2	330 , 5 watt	orange-orange-brown
( )	R3	10 K	brown-black-orange
( )	R4	10 K	" " "
( )	R5	1 K	brown-black-red
( )	R6	68	blue-gray-black
( )	R7	10 K	brown-black-orange
( )	R8	1 K	brown-black-red
( )	R9	56 K	green-blue-orange
( )	R10	10 K	brown-black-orange
( )	R11	1690	bronw-blue-white-brown
( )	R12	4020	yellow-black-red-brown

( ) Step 12. Install U2 (1458) in its location between C2 and C3. U2 is positioned with pin 1 in the lower left-hand corner and soldered into place. See "Loading DIP Devices" in Appendix IV.

( ) Step 13. Install diodes D1 (1N5231B), D2 (1N4148), D3 and D4 (1N4001). Bend leads to fit distance between mounting holes, insert leads, pull down snug to board, solder and trim. BE SURE to position D1 with its cathode (dark band) to the left, D2 and D3 with their cathode at the bottom, and D4 with its cathode at the top.

( ) Step 14. Install the following capacitors in the indicated locations. Take care to observe the proper value, type and orientation, if applicable, for each installation. Bend leads outward on solder (back) side of board, solder and trim.

(See NOTE on Page II-11.

NOTE

Disc capacitor leads are usually coated with wax during the manufacturing process. After inserting leads through mounting holes, remove capacitor and clear the holes of any wax. Reinsert and install.

	LOCATION	VALUE (ufd)	TYPE	ORIENTATION
( )	C1	15	Tantalum	"+" lead bottom right
( )	C2	.1	Disc	None
( )	C3	.1	Disc	None
( )	C6	15	Tantalum	"+" lead right
( )	C7	15	Tantalum	"+" lead left

( ) Step 15. Install 2500 ufd capacitors in locations C4 and C5. Bend leads to fit distance between mounting holes, insert leads, pull down snug to board, solder and trim. Be sure to install C4 with its "+" lead to the right and C5 with its "+" lead to the left.

( ) Step 16. Install Q2 and Q3 (2N2222) in their locations. The emitter lead (closest to tab on can) of Q2 is oriented toward the left and the base lead toward the bottom. The emitter lead of Q3 is oriented toward the bottom and the base lead toward the right.

( ) Step 17. Read assembly tip 6, on page II-5. Apply heat sink compound to the inside of the small black "star-shaped" cooling fin, and install it, with the cylinderical grip down, on Q2 by slipping it down onto the can. Be sure heat sink does not touch any other component on the board.

( ) Step 18. Install bridge rectifier FWB 2 (MDA101A) in its location at the bottom of the board. Apply heat sink compound, per Assembly tip 6 on page II-5. Position FWB2 with its "+" lead at the top and its "-" lead at the bottom, insert leads, solder and trim.

( ) Step 19. Install large heat sink, U1 and U3 in their locations on the bottom left corner of the circuit board.

( ) Position large black heat sink, (flat side to board) over the square foil area in the lower left corner of the PC board. Orient sink so that the two triangular cutouts in the sink are over the two triangles of mounting holes in the board.

( ) Position U1 (7812) on heat sink and observe how leads must be bent to fit mounting holes. Note that the center lead must be bent down approximately 0.2 inches.

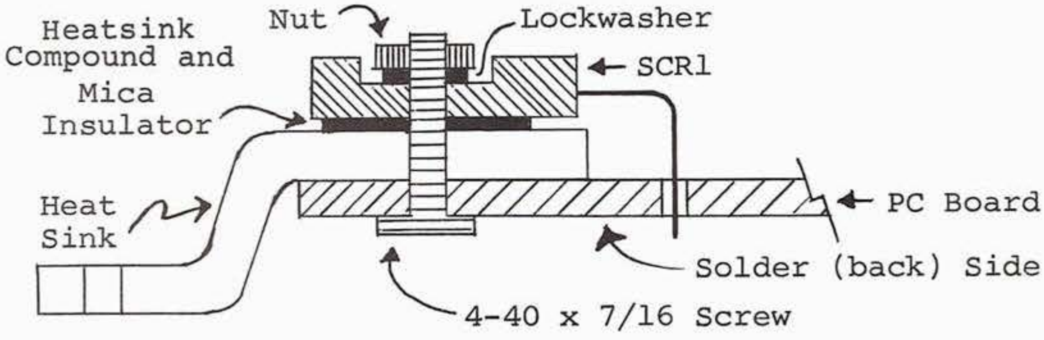
further from the body than the other two leads. Bend leads so that no contact is made with the heat sink when U1 is flat against the sink and its mounting hole is aligned with the holes in the sink and PC board. Apply heat sink compound per Assembly Tip 6, on page II-5. Fasten U1 and sink to board using a 6-32 x ½ metal screw, lockwasher and nut. Insert screw from back (solder) side of board and drive nut finger tight.

( ) Position U3 (7912) on heat sink, determine how leads must be bent as you did for U1, and bend leads. Place a rectangular mica insulator over the leads of U3 so that it fully covers the bottom side of the U3 package. Apply heat sink compound to U3, the heat sink, and both sides of the mica insulator. Bend the two outside leads of U3 slightly in toward the center lead, insert leads in mounting holes as you did for U1, and fasten U3 to heat sink and PC board using a 6-32 x ½ Nylon screw, lockwasher and nut. Insert screw from back (solder) side of board and drive nut finger tight.

( ) Position heat sink, U1 and U3 as needed to obtain correct fit and tighten the U1 and U3 mounting screws. REMEMBER, NO LEADS CAN CONTACT THE SINK. Solder all leads and trim if required.

( ) Step 20. Install aluminum heat sink, SCR1, Q1 and bridge rectifier FWB1.

( ) Position aluminum heat sink (see Figure 2-3) along top of PC board so that the three holes in one side of the sink are aligned with the SCR1, Q1 and FWB1 mounting holes in the PC board.



(Left end, cross-section view)

Figure 2-3. Aluminum heat sink installation.

( ) Position Q1 (TIP41), with component nomenclature up, on heat sink so hole in Q1 package is aligned with the holes in sink and PC board. Observe how the leads of Q1 must be bent down to fit the pads for Q1 and bend them accordingly. Apply heat sink compound to Q1, the heat sink, and both sides of the rectangular mica insulator. Place mica insulator between heat sink and Q1, insert leads (emitter lead to right) and fasten Q1, insulator and heat sink to board with a 6-32 x ½ Nylon screw, lockwasher and nut. Insert screw from back (solder) side of board and drive nut finger tight.

( ) Position FWB1 (MDA970-1), with "+" lead to the right, on heat sink, determine how leads must be bent as you did for Q1, and bend leads. Apply heat sink compound. Insert leads ("+" lead to right) and fasten FWB1 and heat sink to PC board with a 4-40 x 5/8 screw, lockwasher and nut. Insert screw from back (solder) side of board and drive nut finger tight.

( ) Position SCR1 (IR106B2 or MCR106-2) on heat sink with component nomenclature up and prepare it for installation as you did Q1 and FWB1. Apply heat sink compound to SCR1, the heat sink, and both sides of the circular mica insulator. Place the mica insulator between the heat sink and SCR1, insert leads and fasten SCR1, insulator and heat sink to PC board with a 4-40 x 7/16" screw, lockwasher and nut. Insert screw from back (solder) side of board and drive nut finger tight.

( ) Check alignment of heat sink, SCR1, Q1 and FWB2 and tighten the three mounting screws. Solder all leads and trim if required. Wipe off excess heat sink compound, if necessary. <u>NOTE</u>: The heat sink may have to be repositioned when you mount the Sol-REG on the power supply subchassis. This will require that you loosen the mounting screws for SCR1, Q1 and FWB2 and retighten them after repositioning the heat sink.

( ) <u>Step 21</u>. Connect two wire cable assembly (C8 to Regulator Board cable) to regulator. Tin ends without lugs and solder green (+) lead to pad X2 and white (-) lead to pad X3.

( ) <u>Step 22</u>. Test Sol-REG for short circuits. Check for continuity between FWB1 (MDA970-1) mounting screw and the following points: (The resistance should be greater than 20 ohms in all cases.)

X2
T2
T1
Q1, Emitter
Q1, Base
Q1, Collector
D1, right-hand lead
R1, left-hand lead
D3, top lead
D4, top lead
*D3, bottom lead
*D4, bottom lead

*Resistance will be initially low due to C4 and C5, but it should increase to greater than 20 ohms after a few seconds.

( ) Step 23. Set Sol-REG to one side.

2.7.3 Power Supply Subchassis Assembly and Test

( ) Step 24. Mount transformer (T1 for Sol-10, T2 for Sol-20) on power supply subchassis (L-shaped chassis).

Position transformer as shown in drawing on Page X-2 and attach it to the subchassis with three 8-32 x ½ binder or pan head screws, #8 lockwashers and 8-32 hex nuts. Insert screws from bottom and outer side of chassis as shown. Place lockwasher on each screw and secure loosely with hex nuts. Slide transformer as close as possible to the edge of the chassis and tighten nuts.

NOTE

> Only one of the holes in the side wall is used. Use the one that lines up with the transformer mounting tab.

( ) Step 25. Prepare transformer leads.

( ) Twist the two black wires together except for the last two inches at the commoning block lug end.

( ) Twist the two green wires together for their full length.

( ) Twist the two yellow wires together for their full length.

*( ) Twist the two blue wires together for their full length.

( ) Step 26. Connect Sol-PC power cable (4-wire cable which connects to J10 on Sol-PC) to Sol-REG. Tin ends of cable and solder green lead to pad X9, white lead to pad X1, red lead to pad X7 and white-yellow lead to pad X8.

*( ) Step 27. Connect Sol-20 DC power cable (5 wire) to Sol-REG. Tin ends of cable and solder white lead to pad X4 (above R8), red-white lead to pad X5 (between C5 and FWB2) and yellow-white lead to pad X6 (left of C5).

( ) Step 28. Connect transformer leads to Sol-REG.

( ) Solder green leads to pads T1 and T2, white-yellow lead to pad T3 and yellow leads to pads T4 and T5 on Sol-REG circuit board.

( ) Step 29. Prethread the three Sol-REG heat sink mounting holes in the power supply subchassis shown in drawing on page X-2 with #6 x 5/16 sheet metal screws. Remove screws.

( ) Step 30. Place #4 lockwashers on two 4-40 x 3/16 binder or pan head screws. Insert these screws from the bottom side of the power supply subchassis through the two mounting holes located near the middle of the bottom of the power supply subchassis, one on each side. Place another #4 lockwasher on the screws and drive each screw tightly into a 4-40 x ¼ tapped spacer.

( ) Step 31. Position Sol-REG PC board with top edge over the previously installed spacers. Place #4 lockwashers on two 4-40 x 3/16 binder or pan head screws and drive screws t through Sol-REG board into spacers.

( ) Step 32. Attach heat sink on Sol-REG to power supply subchassis as shown in drawing on Page X-2. At this point use only the two side screws which you used in Step 29 to prethread the holes. (The middle screw will be installed later.) Place a #6 lockwasher on each screw before driving it through the sink into the subchassis. Figure 2-4 shows an assembled Sol-10 power supply subchassis.

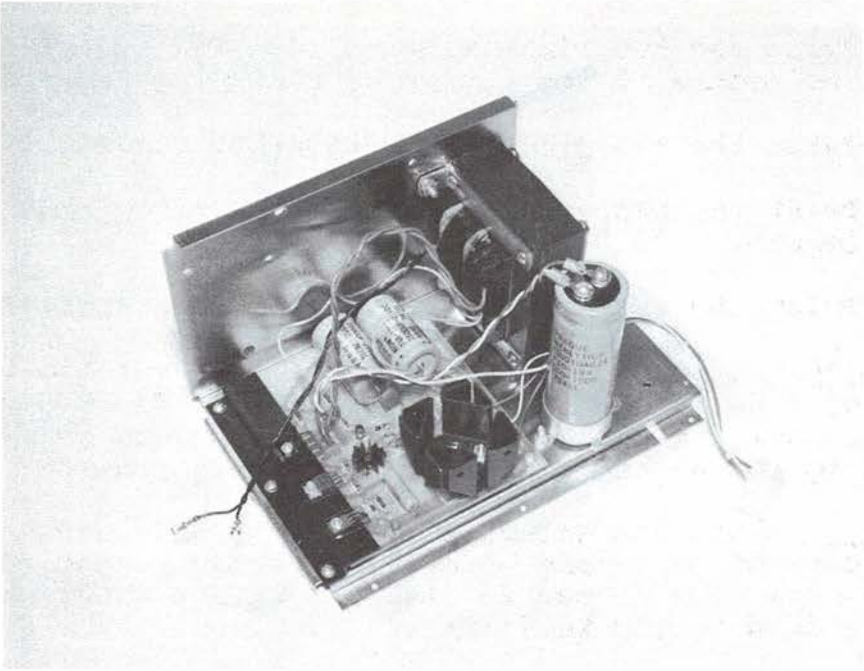


Figure 2-4. Sol-10 power supply subchassis assembly. (Rear of subchassis at left.)

*( ) Step 33. Install bridge rectifier FWB3 on power supply subchassis.

(Step 33 continued on Page II-16.)

Position FWB3 (MDA980-1) on power supply subchassis as shown in drawing on Page X-2. BE SURE NEGATIVE (-) TERMINAL OF FWB3 is next to transformer. Insert a 6-32 x ½ binder or pan head screw from bottom of subchassis, place #6 lockwasher on screw and secure with 6-32 hex nut.

*( ) Step 34. Connect blue transformer wires to unmarked terminals of FWB3.

*( ) Step 35. Install large (2½") mounting ring for C9 (54,000 ufd capacitor) on side wall of power supply subchassis as shown in drawing on Page X-2.

Position ring over the three mounting holes in the side wall of subchassis so the clamping screw faces the bottom of subchassis and so it will be accessible from the Sol-REG end of the subchassis. Insert three 6-32 x ½ binder or pan head screws from outer side of side wall through the mounting holes. Place #6 lockwasher on each screw and secure with 6-32 hex nut. Figure 2-5 shows an assembled Sol-20 power supply subchassis.

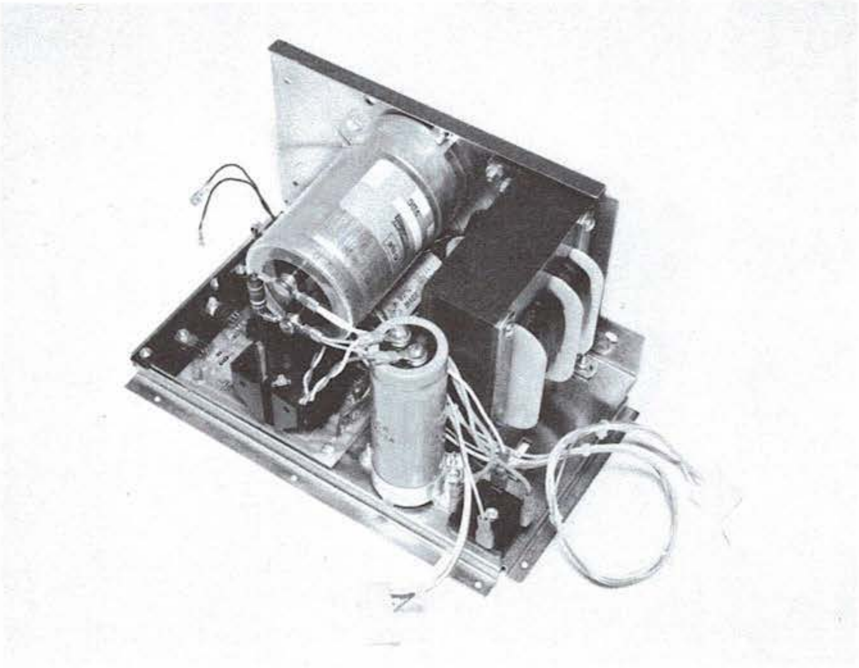


Figure 2-5. Sol-20 power supply subchassis assembly. (Rear of subchassis at left.)

( ) Step 36. Install small (1½") mounting ring for C8 (18,000 ufd capacitor) as shown in drawing on Page X-2.

(Step 36 continued on Page II-17.)

Position ring over the two mounting holes located between FWB3 and the Sol-REG so that the clamping screw is positioned between the transformer and FWB3. Insert two 6-32 x ½ binder or pan head screws from bottom side of chassis through the mounting holes. Place #6 lockwasher on each screw and secure with 6-32 hex nut. (Refer to Figure 2-4.)

( ) Step 37. Route Sol-PC power cable between C8 mounting ring and the transformer, mount C8 in its mounting ring, and tighten clamping screw. (See Figure 2-4.)

( ) Step 38. Connect white wire of C8 cable to negative (-) terminal of C8 and green wire to positive (+) terminal of C8. (This cable was soldered to the Sol-REG when you assembled it.) Remove terminal screws, place #10 lockwasher on each screw, place cable lugs on screws and drive screws tightly into appropriate terminals.

*( ) Step 39. Mount C9 in its mounting ring with its "+" terminal slightly toward C8 and tighten clamping screw. (See Figure 2-5.)

*( ) Step 40. Prepare R13 (39 ohm 2 watt) for installation on C9.

Solder a #10 lug to each lead of R13. Bend leads of R13 to fit the terminals of C9. (R13 should fit on C9 as shown in Figure 2-5.)

*( ) Step 41. Connect Sol-20 DC power cable (5 wire) and R13 to C9. Route cable between C8 and transformer.

Remove terminal screws from C9. Place lockwasher, terminal screw, blue lead of Sol-20 DC cable and one R13 lead on one terminal screw and drive it into the positive (+) terminal on C9. Attach lockwasher, white cable lead and other R13 lead to negative (-) terminal on C9 in the same manner. Tighten both capacitor terminals tightly.

CAUTION

LOOSE CONNECTIONS ON C9 CAN LEAD TO ARCING AND SUBSEQUENT POWER SUPPLY DAMAGE.

*( ) Step 42. Connect blue pigtail of Sol-20 DC cable to positive (+) terminal of FWB3. (This pigtail has a spade lug at its free end and is connected to the lug you just attached to the positive terminal of C9.) Connect white pigtail of Sol-20 DC cable to negative (-) terminal of FWB3. (This pigtail has a spade lug at its free end and is connected to the lug you just attached to the negative terminal of C9.)

( ) Step 43. Connect green lead from AC receptacle (mounted on fan closure plate) to power supply subchassis assembly as shown in drawing on Page X-2. (Use the #6 x ¼ sheet metal screw with which you prethreaded the middle Sol-REG heat sink mounting hole in Step 29.) Place lug on screw and drive screw into the middle Sol-REG heat sink mounting hole.

( ) Step 44. Route black transformer leads along side wall of power supply subchassis out toward the Sol-REG heat sink. (See Figure 2-4.) Attach one lead to pin 2 of the commoning block (mounted on fan closure plate) nearest the fan. Attach other lead to pin 3 of the other commoning block.

( ) Step 45. Install cable tie wraps.

( ) Install one wrap around the wires that connect to Sol-REG pads T1,2,3,X2 and X3 as shown in the Detail A - Wiring portion of the drawing on Page X-2.

*( ) Install another wrap around the leads from C9 as shown in Detail B of drawing on Page X-2.

Two other wraps are supplied with your kit. Use them as appropriate to make your power supply cabling neater.

( ) Step 46. Using a #6 x ¼ sheet metal screw, attach fan closure plate to power supply subchassis as shown in Drawing No. X-2.

( ) Step 47. Push on-off switch in and out to determine the OFF position (switch mechanically out). With switch in OFF position, connect AC power cord to AC receptacle. Then plug power cord into 110 V ac outlet.

( ) Step 48. Test power supply for proper operation.

( ) Make sure on-off switch is in OFF position.

( ) Install fuse in fuse holder. CAUTION: NEVER INSTALL OR REMOVE FUSE WITH POWER ON.

( ) Check connector on Sol-PC power cable (4 wire) to insure it is wired as shown in Figure 2-6.

*( ) Check connector on Sol-20 power cable (5 wire) to insure it is wired as shown in Figure 2-7.

( ) Turn on-off switch ON.

( ) Measure the voltages at the Sol-PC connector at the points indicated in Figure 2-6. The voltages must be as given in Figure 2-6. NOTE: Do not take voltage measurements at any other points in the power supply, even through they may be more accessible. It is important that the indicator voltages be available at the connector.

*( ) Measure the voltages at the Sol-20 connector at the points indicated in Figure 2-7. The voltages must be within the ranges given in Figure 2-7. (See preceding NOTE.)

( ) If the power supply fails any of the preceding tests, locate and correct the cause before proceeding.

If the power supply is operating correctly, turn on-off switch OFF, disconnect power cord, set power supply to one side and go on to Section III.

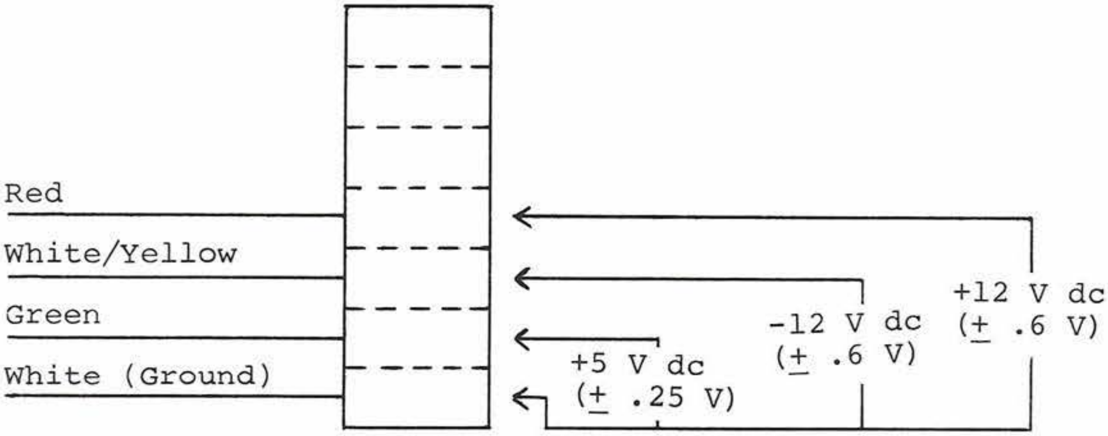


Figure 2-6. Sol-PC power connector and voltage measurements.

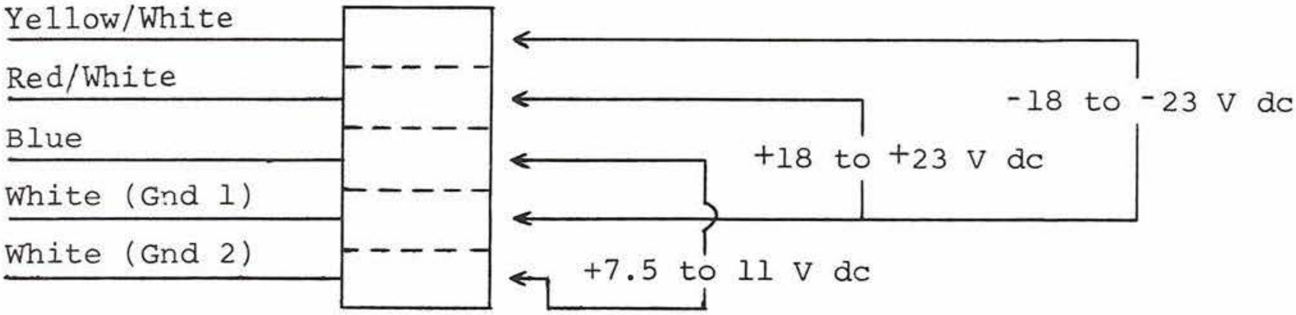


Figure 2-7. Sol-20 power connector and voltage measurements.

III Sol-PC ASSEMBLY and TEST

3.1 PARTS AND COMPONENTS

Check all parts and components against the "Parts List" on Pages III-2 through III-4 (Table 3-1). If you have difficulty in identifying any parts by sight, refer to Figure 3-1 on Page III-5.

3.2 ASSEMBLY TIPS

1. Scan Sections III and IV in their entirety before you start to assemble your Sol-PC kit.

2. In assembling your Sol-PC, you will be following an integrated assembly-test procedure. Such a procedure is designed to progressively insure that individual circuits and sections in the Sol-PC are operating correctly. IT IS IMPORTANT THAT YOU FOLLOW THE STEP-BY-STEP INSTRUCTIONS IN THE ORDER GIVEN.

3. Assembly steps and component installations are preceded by a set of parentheses. Check off each installation and step as you complete them. This will minimize the chances of omitting a step or component.

4. When installing components, make use of the assembly aids that are incorporated on the circuit boards and the assembly drawings. (These aids are designed to assist you in correctly installing the components.)

   a. The circuit reference (R3, C10 and U20, for example) for each component is silk screened on the PC boards near the location of its installation.

   b. Both the circuit reference and value or nomenclature (1.5K and 74H00, for example) for each component are included on the assembly drawings near the location of its installation.

5. To simplify reading resistor values after installation, install resistors so that the color codes or imprints read from left to right and top to bottom as appropriate (boards oriented as defined in Paragraph 3.5 on Page III-7).

6. Unless specified otherwise, install components, especially disc capacitors, as close as possible to the boards.

7. Should you encounter any problem during assembly, call on us for help if needed.

Table 3-1. Sol-PC Parts List.

INTEGRATED CIRCUITS

1	AM0026 or DM0026 (U104)
1	4N26 (U39)
1	8T94 (U58)
5	8T97 (U67,68,77,80,81)
2	1458CP or 1558CP (U56,108)
1	1489A (U38)
2	TMS6011NC (U51,69)
1	MCM6574 or MCM6575 (U25)
1	4001 (U102)
2	4013 (U100,113)
1	4019 (U111)
1	4023 (U98)
1	4024 (U86)
1	4027 (U101)
3	4029 (U1,11,84)
1	4030 (U99)
2	4046 (U85,110)
2	4049 (U88,109)
1	4520 (U112)
1	74H00 (U91)
3	74LS00 (U44,48,55)
2	74LS02 or 9LS02 (U53,60)
4	74LS04 (U24,45,49,54)
1	74S04 (U92)
2	7406 (U57,87)
2	74LS10 (U47,61)
3	74LS20 (U23,59,83)
1	74LS86 (U74)
8	74LS109 (U43,52,63,64,70,72,73,75)
1	74LS136 (U22)
3	74LS138 (U34,35,36)
3	74LS157 (U12,30,32)
4	74LS163 or 25LS163 (U28,31,33,40)
1	74166 (U41)
2	74173 (U95,96)
1	74175 (U97)
9	74LS175 or 25LS175 (U2,13,26,27,42,76,90,93,106)
4	74LS253 (U65,66,78,79)
7	74LS367 (U29,37,50,71,89,94,107)
1	8080, 8080A or 9080A (U105)
1	8836 or 8T380 (U46)
16	91L02APC or 2102L1PC (U3 - 10, U14 - 21)
1	93L16 (U62)

TRANSISTORS

2	2N2222 (Q4 & Q5)
2	2N2907 or 2N3460 (Q1 & Q2)
1	2N4360 (Q3)

DIODES

9	1N4148 or 1N914 (D1,D3 - 10)
1	1N5231B Zener Diode (D11)
4	1N4001 (D2,12,13,14)

CRYSTAL

1	14.318 MHz in HC-18/U Case (XTAL)

RELAYS

2	DIP Reed, Sigma 191-TE1A15S (K1 & K2)

Table 3-1. Sol-PC Parts List (Continued).

RESISTORS		
2	6.8	ohm, ½ watt, 5%
2	47	ohm, ¼ watt, 5%
1	75	ohm, ¼ watt, 5%
1	100	ohm, ¼ watt, 5%
3	100	ohm, ½ watt, 5%
1	200	ohm, ¼ watt, 5%
13	330	ohm, ¼ watt, 5%
1	330	ohm, ½ watt, 5%
3	470	ohm, ¼ watt, 5%
2	470	ohm, ½ watt, 5%
9	680	ohm, ¼ watt, 5%
63	1.5K	ohm, ¼ watt, 5%
1	3.3K	ohm, ¼ watt, 5%
6	5.6K	ohm, ¼ watt, 5%
32	10 K	ohm, ¼ watt, 5%
1	15 K	ohm, ¼ watt, 5%
2	39 K	ohm, ¼ watt, 5%
1	47 K	ohm, ¼ watt, 5%
3	50 K	ohm, Potentiometer
4	100 K	ohm, ¼ watt, 5%
2	150 K	ohm, ¼ watt, 5%
2	1 M	ohm, ¼ watt, 5%
1	2.2M	ohm, ¼ watt, 5%
2	3.3M	ohm, ¼ watt, 5%

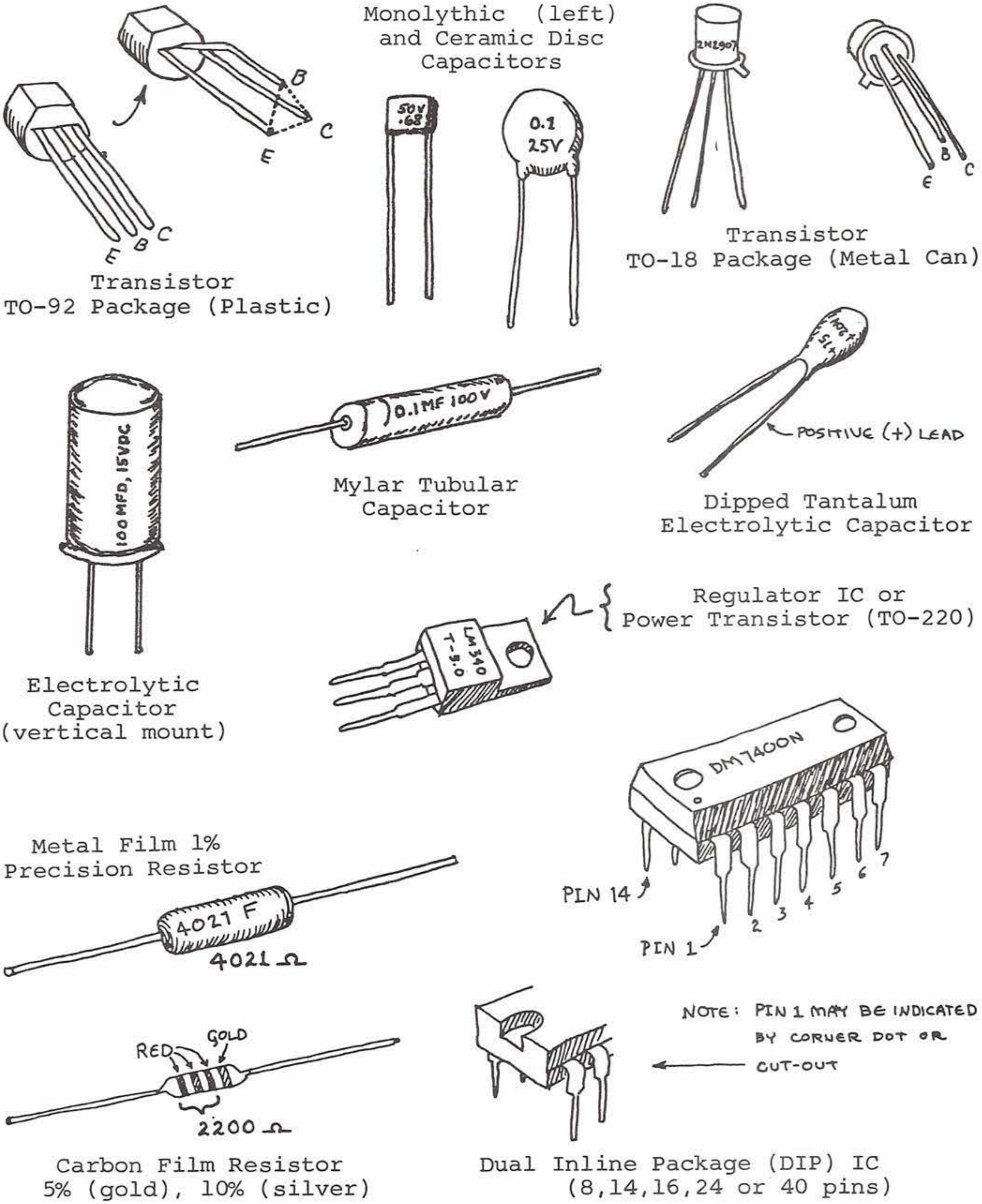
CAPACITORS		
1	10	pfd, disc
1	330	pfd, disc
1	470	pfd, disc
3	680	pfd, monolythic or disc ceramic (labeled 681 and usually blue)
6	.001	ufd, disc
2	.001	ufd, Mylar tubular
2	.01	ufd, Mylar tubular
37	.047	ufd, disc
12	.1	ufd, disc
1	.1	ufd, Mylar tubular
1	.68	ufd, monolythic ceramic
1	1	ufd, tantalum dipped (usually orange or red)
5	15	ufd, tantalum dipped (usually orange or red)
1	100	ufd, aluminum electrolytic

CONNECTORS	
1	25-pin Female, AMP206584-2 (J1)
1	25-pin Male, AMP206604-1 (J2)
2	20-pin Header, 3M3492-2002 (J3 & J4)
1	30-pin Right Angle Edge Connector, VIKING 3KH15/1JKC15 (J5)
2	Miniature Phone Jack (J6 & J7)
2	Subminiature Phone Jack (J8 & J9)
1	7-pin Male Locking Molex Connector (J10)
1	100-pin Edge Connector, TI H322150-0306A (J11)
1	Molex-type DC Power Cable, mates with J10 (prefabricated)

Table 3-1. Sol-PC Parts List (Continued).

MISCELLANEOUS

1	Sol-PCB Circuit Board	length of #24 bare wire
2	8-pin DIP Socket	
29	14-pin DIP Socket	
74	16-pin DIP Socket	
1	24-pin DIP Socket	
3	40-pin DIP Socket	
16	Augat Pins on Carrier	
2	DIP Switch, 6 position (S1 & S4)	
2	DIP Switch, 8 position (S2 & S3)	
1	4-foot Length 72-ohm Coaxial Cable	
1	Tie Wrap for Coaxial Cable	
2	Mounting Bracket, Sol-1040	
2	Card Guide, SAE1250F	
10	#4 Lockwasher, internal tooth	
2	#4 Insulating Washer	
4	4-40 x ¼ Binder Head Screw	
6	4-40 x 7/16 Binder Head Screw	
2	4-40 x 5/8 Binder Head Screw	
10	4-40 Hex Nut	
1	Length Solder	
1	Manual	
1	Personality Module Kit (See Section IV for contents.)	



Monolythic (left) and Ceramic Disc Capacitors

Transistor TO-92 Package (Plastic)

Transistor TO-18 Package (Metal Can)

Mylar Tubular Capacitor

Dipped Tantalum Electrolytic Capacitor

Electrolytic Capacitor (vertical mount)

Regulator IC or Power Transistor (TO-220)

Metal Film 1% Precision Resistor

Carbon Film Resistor 5% (gold), 10% (silver)

Dual Inline Package (DIP) IC (8,14,16,24 or 40 pins)

Figure 3-1. Identification of components.

3.3 ASSEMBLY PRECAUTIONS

3.3.1 Handling MOS Integrated Circuits

Many of the IC's used in the Sol-PC are MOS devices. They can be damaged by static electricity discharge. Always handle MOS IC's so that no discharge will flow through the IC. Also, avoid unnecessary handling and wear cotton--rather than synthetic--clothing when you do handle these IC's.

3.3.2 Soldering **IMPORTANT**

1. Use a fine tip, low-wattage iron, 25 watts maximum.

2. DO NOT use excessive amounts of solder. DO solder neatly and as quickly as possible.

3. Use only 60-40 rosin-core solder. NEVER use acid-core solder or externally applied fluxes.

4. To prevent solder bridges, position iron tip so that it does not touch adjacent pins and/or traces simultaneously.

5. DO NOT press tip of iron on pad or trace. To do so can cause the pad or trace to "lift" off the board and permanently damage the board.

6. The Sol-PC uses circuit boards with plated-through holes. Solder flow through to the component (front) side of the board can produce solder bridges. Check for such bridges after you install each component.

7. The Sol-PC circuit boards have integral solder masks (a lacquer coating) that shield selected areas on the boards. This mask minimizes the chances of creating solder bridges during assembly. DO, however, check all solder joints for possible bridges.

8. Additional pointers on soldering are provided in Appendix IV of this manual.

3.3.3 Power Connection (J10)

NEVER connect the DC power cable to the Sol-PC when power supply is energized. To do so can damage the Sol-PC.

3.3.4 Installing and Removing Integrated Circuits

NEVER install or remove integrated circuits when power is applied to the Sol-PC. To do so can damage the IC.

3.3.5 Installing and Removing Personality Module

NEVER install or remove the plug-in personality module when power is applied to the Sol-PC. To do so can damage the module.

3.3.6 Use of Clip Leads

TAKE CARE when using a clip lead to establish a ground connection when testing the Sol-PCB circuit board. Make sure that the clip makes contact <u>only</u> with the ground bus on the perimeter of the board.

3.4 REQUIRED TOOLS, EQUIPMENT AND MATERIALS

The following tools, equipment and materials are recommended for assembling and testing the Sol-PC:

1. Needle nose pliers
2. Diagonal cutters
3. Screwdriver
4. Sharp knife
5. Controlled heat soldering iron, 25 watt
6. 60-40 rosin-core solder (supplied)
7. Small amount of #24 solid wire
8. Volt-ohm meter
9. Video monitor or monochrome TV converted for video input.
10. IC test clip (optional)
11. Oscilloscope (optional)

3.5 ORIENTATION (Sol-PCB)

Location J5 (personality plug-in module connector) will be located in the upper right-hand area of the circuit board when location J10 (power connector) is positioned at the bottom of the board. In this position the component (front) side of the board is facing up and all IC legends (U1 through U10, U22 through U24, etc.) will read from left to right. Subsequent position references related to the Sol-PCB assume this orientation.

3.6 Sol-PC ASSEMBLY-TEST PROCEDURE

The Sol-PC is assembled and tested in sections and/or circuits. You will first test the Sol-PCB circuit board for shorts (solder bridges) between the power buses and ground. After assembling the personality module (see Section IV), the clock and display control circuits are assembled. The bus, CPU, decoder and memory circuits are then assembled, followed by the parallel and serial input/output (I/O) and audio cassette I/O sections.

CAUTION

THE Sol-PC USES MANY MOS INTEGRATED CIRCUITS. THEY CAN BE DAMAGED BY STATIC ELECTRICITY DISCHARGE. HANDLE THESE IC's SO THAT NO DISCHARGE FLOWS THROUGH THE IC. AVOID UNNECESSARY HANDLING AND WEAR COTTON, RATHER THAN SYNTHETIC, CLOTHING WHEN YOU DO HANDLE MOS IC's. (STATIC CHARGE PROBLEMS ARE MUCH WORSE IN LOW HUMIDITY CONDITIONS.)

3.6.1 Circuit Board Check

( ) Visually check Sol-PCB board for solder bridges (shorts) between traces, broken traces and similar defects.

( ) Check board to insure that the +5-volt-bus, +12 volt-bus and -12-volt bus are not shorted to each other or to ground. Using an ohmmeter, on "OHMS X 1K" or "OHMS X 10K" scale, make the following measurements (refer to Sol-PC Assembly Drawing X-3).

( ) +5-volt Bus Test. Measure between positive and negative mounting pads for C58. There should be no continuity. (Meter reads close to "infinity" ohms.)

( ) +12-volt Bus Test. Measure between positive and negative mounting pads for C59. There should be no continuity.

( ) -12-volt Bus Test. Measure between positive and negative mounting pads for C60. There should be no continuity.

( ) 5/12/(-12) Volt Bus Test. Measure between positive mounting pads for C58 and C59, between positive pad for C58 and negative pad for C60, and between positive pad for C59 and negative pad for C60. You should measure no continuity in any of these measurements.

If visual inspection reveals any defects, or you measure continuity in any of the preceding tests, return the board to Processor Technology for replacement. If the board is not defective, proceed to next paragraph.

3.6.2 Personality Module Assembly

Since the personality module is required for testing the Sol-PC in the later stages of its assembly, we suggest that you assemble the personality module first. In so doing, your Sol-PC assembly will proceed uninterrupted. Assembly instructions for the personality module are provided in Section IV of this manual.

If you wish to wait to assemble the personality module until it is needed, go on to Paragraph 3.6.3.

3.6.3 Sol-PCB Assembly and Test

Refer to Sol-PC assembly drawing X-3.

( ) Step 1. Install DIP sockets. Install each socket in the indicated location with its end notch oriented as shown on the circuit board and assembly drawing. Take care not to create solder bridges between the pins and/or traces. (Refer to footnotes at end of this step before installing U105.)

INSTALLATION TIP

Insert socket pins into mounting pads of appropriate location. On solder (back) side of board, bend pins at opposite corners of socket (e.g., pins 1 and 9 on a 16-pin socket) outward until they are at a 45° angle to the board surface. This secures the socket until it is soldered. Repeat this procedure with each socket until all are secured to the board. Then solder the unbent pins on all sockets. Now straighten the bent pins to their original position and solder.

LOCATION	TYPE SOCKET
( ) U1 through 21	16 pin
( ) U22 through 24	14 pin
( ) U25	24 pin
( ) U26 through 37	16 pin
( ) U38	14 pin
( ) U39	None
( ) U40 through 43	16 pin
( ) U44 through 49	14 pin
( ) U50	16 pin
( ) U51	40 pin
( ) U52	16 pin
( ) U53 through 55	14 pin
( ) U56	8 pin
( ) U57 through 61	14 pin

(Continued on Page III-10.)

LOCATION	TYPE SOCKET
( ) U62 through 68	16 pin
( ) U69	40 pin
( ) U70 through 73	16 pin
( ) U74	14 pin
( ) U75 through 81	16 pin
( ) U82#	None#
( ) U83	14 pin
( ) U84,85	16 pin
( ) U86,87	14 pin
( ) U88 through 90	16 pin
( ) U91,92	14 pin
( ) U93 through 97	16 pin
( ) U98 through 100	14 pin
( ) U101	16 pin
( ) U102	14 pin
( ) U103#	None #
( ) U104	None
( ) U105*	40 pin
( ) U106,107	16 pin
( ) U108	8 pin
( ) U109 through 112	16 pin
( ) U113	14 pin

#Spare locations, not used.
*Note that U105 notch is positioned at the top.

( ) Step 2. Install the following capacitors in the indicated locations. Take care to observe the proper value, type and orientation, if applicable, for each installation. Bend leads outward on solder (back) side of board, solder and trim.

NOTE

Disc capacitor leads are usually coated with wax during the manufacturing process. After inserting leads through mounting holes, remove capacitor and clear the holes of any wax. Reinsert and install.

LOCATION	VALUE (ufd)	TYPE	ORIENTATION
( ) C1	.047	Disc	None
( ) C2	.047	"	"
( ) C3	.047	"	"
( ) C4	.047	"	"
( ) C5	.047	"	"
( ) C6	.047	"	"
( ) C7	.047	"	"
( ) C8	.047	"	"

LOCATION	VALUE (ufd)	TYPE	ORIENTATION
( ) C10	.047	Disc	None
( ) C11	.047	"	"
( ) C13	.047	"	"
( ) C14	.047	"	"
( ) C15	15	Tantalum	"+" lead bottom
( ) C16	.047	Disc	None

( ) Step 3. Check for +5-volt bus to ground shorts. Using an ohmmeter, measure between positive and negative mounting pads for C58. There should be no continuity. If there is, find and correct the problem before proceeding to Step 4.

( ) Step 4. Install the following capacitors in the indicated locations. Take care to observe the proper value, type and orientation, if applicable, for each installation. Bend leads outward on solder (back) side of board, solder and trim. (refer to NOTE in Step 2.)

LOCATION	VALUE (ufd)	TYPE	ORIENTATION
( ) C19	.047	Disc	None
( ) C20	.047	"	"
( ) C21	.047	"	"
( ) C24	.047	"	"
( ) C25	.047	"	"
( ) C26	.047	"	"
( ) C33	.047	"	"
( ) C38	.047	"	"
( ) C40	15	Tantalum	"+" lead bottom
( ) C41	.047	Disc	None
( ) C42	.047	"	"
( ) C45	.047	"	"
( ) C56	.047	"	"
( ) C58	15	Tantalum	"+" lead top
( ) C59	15	Tantalum	"+" lead top
( ) C60	15	Tantalum	"+" lead top
( ) C65	.047	Disc	None

( ) Step 5. Check for +5-volt bus to ground shorts. Using an ohmmeter, measure between the positive and negative leads of C58. You should measure at least 100 ohms. Less than 100 ohms indicates a short. If required, find and correct the problem before proceeding to Step 6. NOTE: In this and subsequent resistance measurements, any value greater than the minimum may normally occur, even much higher, unless otherwise indicated.

( ) Step 6. Install the following capacitors in the indicated locations. Take care to observe the proper value and type for each installation. Bend leads outward on solder (back) side of board, solder and trim. (Refer to NOTE in Step 2.)

(Step 6 continued on Page III-12.)

LOCATION	VALUE (ufd)	TYPE	ORIENTATION
( ) C9	.047	Disc	None
( ) C12	.047	"	"
( ) C17	.047	"	"
( ) C18	.047	"	"
( ) C22	.047	"	"
( ) C23	.047	"	"
( ) C27	.047	"	"
( ) C28	.047	"	"
( ) C46	.047	"	"

( ) Step 7. Check for +5-volt bus to ground shorts. Using an ohmmeter, measure between the positive and negative leads of C58. You should measure some resistance. Zero resistance indicates a short. If required, find and correct the problem before proceeding to Step 8.

( ) Step 8. Install diodes D8 (1N4148 or 1N914), D11 (1N5231B) and D12 (1N4001) in their locations (in the area below U90 through U92). Position D8 with its dark band (cathode) to the right, D11 with its band at the bottom, and D12 with its band at the top.

NOTE

The leads of D12 and its mounting holes are a snug fit. Take care when installing this diode.

( ) Step 9. Install the following resistors in the indicated locations. Bend leads to fit distance between mounting holes, insert leads, pull down snug to board, solder and trim.

LOCATION	VALUE (ohms)	COLOR CODE
( ) R104	10 K	brown-black-orange
( ) R105	1.5K	brown-green-red
( ) R106	1.5K	" " "
( ) R130	100, ½ watt	brown-black-brown
( ) R131	100, ½ watt	" " "
( ) R132	100, ½ watt	" " "
( ) R133	330	orange-orange-brown
( ) R134	330	" " "
( ) R135 & 136	10 K	brown-black-orange
( ) R137 & 138	47	yellow-violet-black

( ) Step 10. Install the following capacitors in the indicated locations. Take care to observe the proper value and type for each installation. Bend leads outward on solder (back) side of board, solder and trim. (Refer to NOTE in Step 2.)

LOCATION	VALUE	TYPE
( ) C39	.1 ufd	Disc
( ) C43	680 pfd	Monolythic or Disc
( ) C44	680 pfd	Monolythic or Disc
( ) C61	.001 ufd	Disc
( ) C62	.68 ufd	Monolythic
( ) C63	.1 ufd	Disc
( ) C64	10 pfd	Disc

( ) Step 11. Install 14.318 MHz crystal in its location just above C61. Insert leads and pull down until the case is 1/16" above the front surface of the board. Solder quickly and trim.

( ) Step 12. Install male Molex connector in location J10. Position connector so the locking clip is facing the crystal (XTAL), insert shorter pins in mounting holes and solder.

( ) Step 13. In the jumper area labeled CLK on the assembly drawing (between U90 and U91), install Augat pins in mounting holes A,B,C,D and E. (Refer to "Installing Augat Pins" in Appendix IV.) Using #24 bare wire, install a jumper between the A and B pins and another jumper between the D and E pins.

( ) Step 14. Install the following IC's in the indicated locations. Pay careful attention to the proper orientation. DO NOT SUBSTITUTE FOR ANY OF THESE IC's.

NOTE

Dots on the assembly drawing and PC board indicate the location of pin 1 of each IC.

IC NO.	TYPE
( ) U77	8T97
( ) U90	74LS175 or 25LS175
( ) U91	74H00
( ) U92	74S04
( ) U104*	AM0026 or DM0026*

*Solder this IC in its location. See "Loading DIP Devices" in Appendix IV.

( ) Step 15. Connect power to power connector J10. Power and interconnection requirements are as follows:

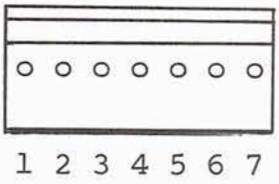
(Step 15 continued on Page III-14.)

CAUTION 1

NEVER CONNECT POWER CABLE TO J10 WITH POWER SUPPLY ENERGIZED.

CAUTION 2

MAKE SURE POWER CABLE CONNECTOR MATES EXACTLY WITH J10; THAT IS, PIN 1 TO PIN 1, PIN 2 TO PIN 2, ETC. ANY OTHER MATING RELATIONSHIP WILL "BLOW" THE IC's.



(J10, Top View)

J10 PIN NO.	POWER
1	Ground
2 and 6	+5 V dc ±5%, 2 A max
3 and 5	-12 V dc ±5%, 300 mA max
4	+12 V dc ±5%, 100 mA max
7	Ground

NOTE

Though not labeled on the connector, J10 pins are designated 1 through 7, reading from left to right.

( ) Step 16. Check clock circuits. If you have an oscilloscope, use part A of this step. If you do not, use part B.

A. Oscilloscope Check

( ) Using an oscilloscope, check for the waveforms given in Figure 3-2 on Page III-15 at the indicated observation points and in the order given. The waveforms shown in Figure 3-2 approximate actual waveforms. If any waveforms are incorrect, determine and correct the cause before proceeding with assembly.

NOTE

Irregularities up to 1 volt are acceptable on positive portions of waveforms. Negative portions, however, should be relatively flat.

B. Volt-ohm Meter Check

( ) Using the test probe shown in Figure 3-3 on Page III-16, set meter to DC volts and make the following measurements:

(Volt-ohm Meter Check continued on Page III-16.)

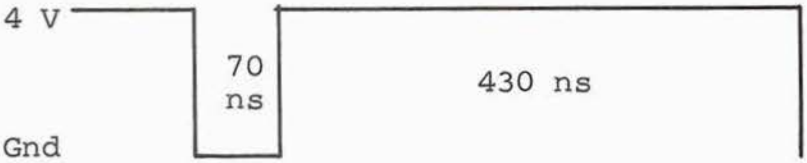
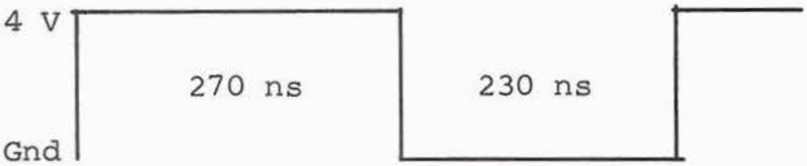
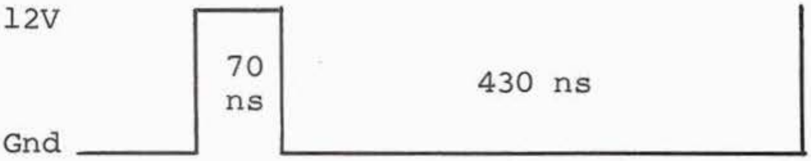
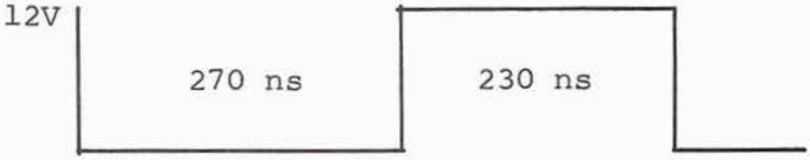
	CHECK POINT	SIGNAL	WAVEFORM
( )	U77, Pin 7	Oscillator Output	14.3 MHz square wave. (This is not a perfect square wave. It in fact more resembles a poor sine wave.)
( )	U91, Pin 6	Clock Divider Output	
( )	U91, Pin 11	Clock Divider Output	
( )	U104, Pin 7	CPU Clock Ø1	
( )	U104, Pin 5	CPU Clock Ø2	

Figure 3-2. Clock circuit waveforms.

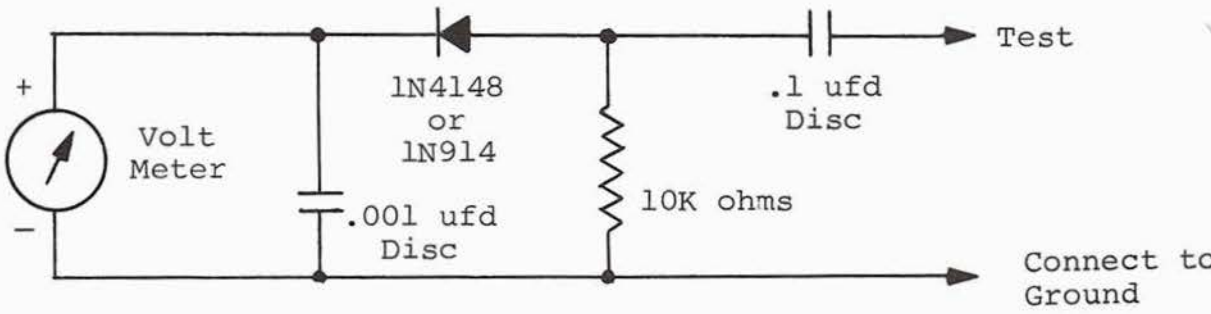


Figure 3-3. Test probe for Steps 16B and 25B.

NOTE 1

The probe shown in Figure 3-3 can be made using parts supplied with your Sol-PC kit. Since these parts will be used later in the Sol-PC assembly, DO NOT shorten the leads or otherwise alter the components. Assemble the probe using tack soldering technique.

NOTE 2

Make sure you have a good ground connection between the meter, probe and Sol-PCB.

( ) At pin 7 of U77 you should measure 1.5 V dc or higher. (A significantly lower reading indicates a faulty oscillator circuit.)

( ) At pin 6 of U91 you should measure 0.25 V dc or higher. (A significantly lower reading indicates a faulty clock divider, U90.)

( ) At pin 11 of U91 you should measure 1.25 V dc or higher. (A significantly lower reading indicates a faulty clock divider, U90.)

( ) At pin 5 of U104 you should measure 4 V dc or higher. (A significantly lower reading indicates a problem with U104.)

( ) At pin 7 of U104 you should measure 8 V dc or higher. (A significantly lower reading indicates a problem with U104.)

( ) If any voltages are incorrect, correct the problem before proceeding; if correct, turn off the power supply and disconnect the power cable.

( ) Step 17. Install the following resistors in the indicated locations. Bend leads to fit distance between mounting holes, insert leads, pull down snug to board, solder and trim.

LOCATION	VALUE (ohms)	COLOR CODE
( ) R1	1.5K	brown-green-red
( ) R2	1.5K	" " "
( ) R3	1.5K	" " "
( ) R4	1.5K	" " "
( ) R5	1.5K	" " "
( ) R6	1.5K	" " "
( ) R7	1.5K	" " "
( ) R8	1.5K	" " "
( ) R9	1.5K	" " "
( ) R10	1.5K	" " "
( ) R11	1.5K	" " "
( ) R16	1.5K	" " "
( ) R17	1.5K	" " "
( ) R19	1.5K	" " "
( ) R30	1.5K	" " "
( ) R80*	330, ½ watt	orange-orange-brown
( ) R81	75	violet-green-black
( ) R82	200	red-black-brown
( ) R83	1.5K	brown-green-red
( ) R84	3.3M	orange-orange-green
( ) R85	1.5K	brown-green-red
( ) R86	1.5K	" " "
( ) R87	330	orange-orange-brown
( ) R88	680	blue-gray-brown
( ) R89	1.5K	brown-green-red
( ) R90	1.5K	" " "
( ) R96	1.5K	" " "
( ) R97	1.5K	" " "
( ) R98	10 K	brown-black-orange
( ) R99	1.5K	brown-green-red
( ) R100	10 K	brown-black-orange
( ) R101	1.5K	brown-green-red
( ) R102	3.3M	orange-orange-green
( ) R103	1.5K	brown-green-red
( ) R120	100 K	brown-black-yellow
( ) R121	10 K	brown-black-orange
( ) R122	10 K	" " "
( ) R123	39 K	orange-white-orange
( ) R124	1.5K	brown-green-red
( ) R125	1.5K	" " "
( ) R126	39 K	orange-white-orange
( ) R127	10 K	brown-black-orange
( ) R128	3.3K	orange-orange-red
( ) R129	10 K	brown-black-orange
( ) VR1 & VR2	50 K	Potentiometer

*The leads of R80 and its mounting holes form a snug fit. Take care when installing this resistor.

( ) Step 18. Install the following capacitors in the indicated locations. Take care to observe the proper value and type for each installation. Bend leads outward on solder (back) side of board, solder and trim. (Refer to NOTE in Step 2.)

CAUTION

REFER TO FOOTNOTE AT END OF THIS STEP BEFORE INSTALLING C31.

	LOCATION	VALUE		TYPE
( )	C31*	100	ufd	Aluminum Electrolytic
( )	C32	.1	ufd	Disc
( )	C34	680	pfd	Monolythic or Disc
( )	C35	.1	ufd	Mylar Tubular
( )	C36	.1	ufd	Disc
( )	C37	.1	ufd	Disc
( )	C52	.001	ufd	Mylar Tubular
( )	C53	.01	ufd	Mylar Tubular
( )	C54	.001	ufd	Disc
( )	C55	.001	ufd	Disc
( )	C57	.1	ufd	Disc

*Install C31 with "+" lead at the top.

( ) Step 19. Install Q2 (2N2907 or 2N3460) in its location below and to the right of U88. The emitter lead (closest to tab on can) is oriented toward the left of the board and the base is oriented toward the bottom. Push straight down on transistor until it is stopped by the leads. Solder and trim.

( ) Step 20. Install diodes D9 and D10 (1N4148 or 1N914) in their locations below U88. Position D9 with its dark band (cathode) to the left and D10 with its band to the right.

( ) Step 21. Install coaxial cable, composite video output. (See Figure 3-4 for details on how to prepare cable.)

( ) Strip away about 1¼" of the outer insulation to expose shield. Unbraid shield, gather and twist into a single lead. Then strip away the inner conductor insulation, leaving about ¼" at the shield end.

CAUTION

WHEN PREPARING AND INSTALLING SHIELD, BE SURE BITS OF BRAID DO NOT FALL ONTO BOARD. SUCH DEBRIS CAN CREATE HARD-TO-FIND SHORT CIRCUITS.

( ) Insert inner conductor in mounting hole P1 (left side of board), solder and trim.

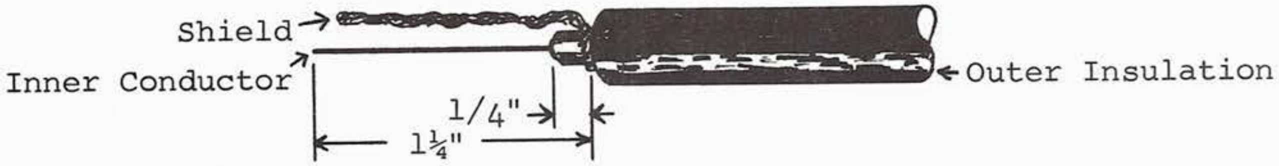


Figure 3-4. Coaxial cable preparation.

( ) Insert twisted shield in mounting hole P2, solder and trim. Using the two large holes to the right of VR1 and VR2, tie cable to board with tie wrap (see CAUTION below).

CAUTION

AFTER INSTALLATION, FINE BITS OF THE BRAID FROM THE SHIELD MAY WORK LOOSE AND FALL ONTO THE BOARD AND CREATE HARD-TO-FIND SHORT CIRCUITS. TO PREVENT THIS, COAT ALL EXPOSED BRAID WITH AN ADHESIVE AFTER SOLDERING AND TIEING. USE AN ADHESIVE SUCH AS SILICONE, CONTACT CEMENT OR FINGERNAIL POLISH. DO NOT USE WATER BASE ADHESIVES.

( ) Step 22. Install 6-position DIP switch in location S1 on left end of board. Position Switch No. 1 at the bottom.

( ) Step 23. Install 20-pin header in location J4 (video expansion connector) between U28 and U29. Position header so pin 1 is in the lower right corner. (An arrow on the connector points to pin 1.)

( ) Step 24. Install the following IC's in the indicated locations. Pay careful attention to the proper orientation.

NOTE

Dots on the assembly drawing and PC board indicate the location of pin 1 of each IC.

IC NO.	TYPE
( ) U28	74LS163 or 25LS163
( ) U31	74LS163 or 25LS163
( ) U33	74LS163 or 25LS163
( ) U40	74LS163 or 25LS163
( ) U43	74LS109
( ) U47	74LS10
( ) U49	74LS04

(Step 24 continued on Page III-20.)

IC NO.	TYPE
( ) U59	74LS20
( ) U60	74LS02 or 9LS02
( ) U62	93L16
( ) U74	74LS86
( ) U75	74LS109
( ) U87	7406
( ) U88*	4049*
( ) U102*	4001*

*MOS device. Refer to CAUTION on Page III-8.

( ) Step 25. Apply power to Sol-PC and check display section timing chain operation. If you have an oscilloscope, use part A of this step. If you do not, use part B.

A. Oscilloscope Check

( ) Using an oscilloscope, check for the waveforms given in Figure 3-5 at the indicated observation points and in the order given. The waveforms shown in Figure 3-5 approximate actual waveforms. If any waveforms are incorrect, determine and correct the cause before proceeding with assembly.

NOTE

Irregularities up to 1 volt are acceptable on positive portions of waveforms. Negative portions, however, should be relatively flat.

B. Volt-ohm Meter Check

( ) Using the test probe made in Step 16B, measure the voltage at pin 12 of U28. You should measure approximately 1 V dc. If you get a significantly lower reading, find and correct the cause before you proceed with assembly.

( ) Turn off power supply and disconnect power connector.

( ) Step 26. Check synchronization circuits.

( ) Set all S1 switches to OFF.

( ) Connect Sol-PC video output cable to video monitor.

SEE CAUTION ON PAGE III-22 BEFORE CONNECTING MONITOR.

(Step 26 continued on Page III-22.)

CHECK POINT WAVEFORM

( ) U28, Pin 12

( ) U47, Pin 8

( ) U59, Pin 8

( ) U43, Pin 9

( ) U88, Pin 10

( ) U88, Pin 4

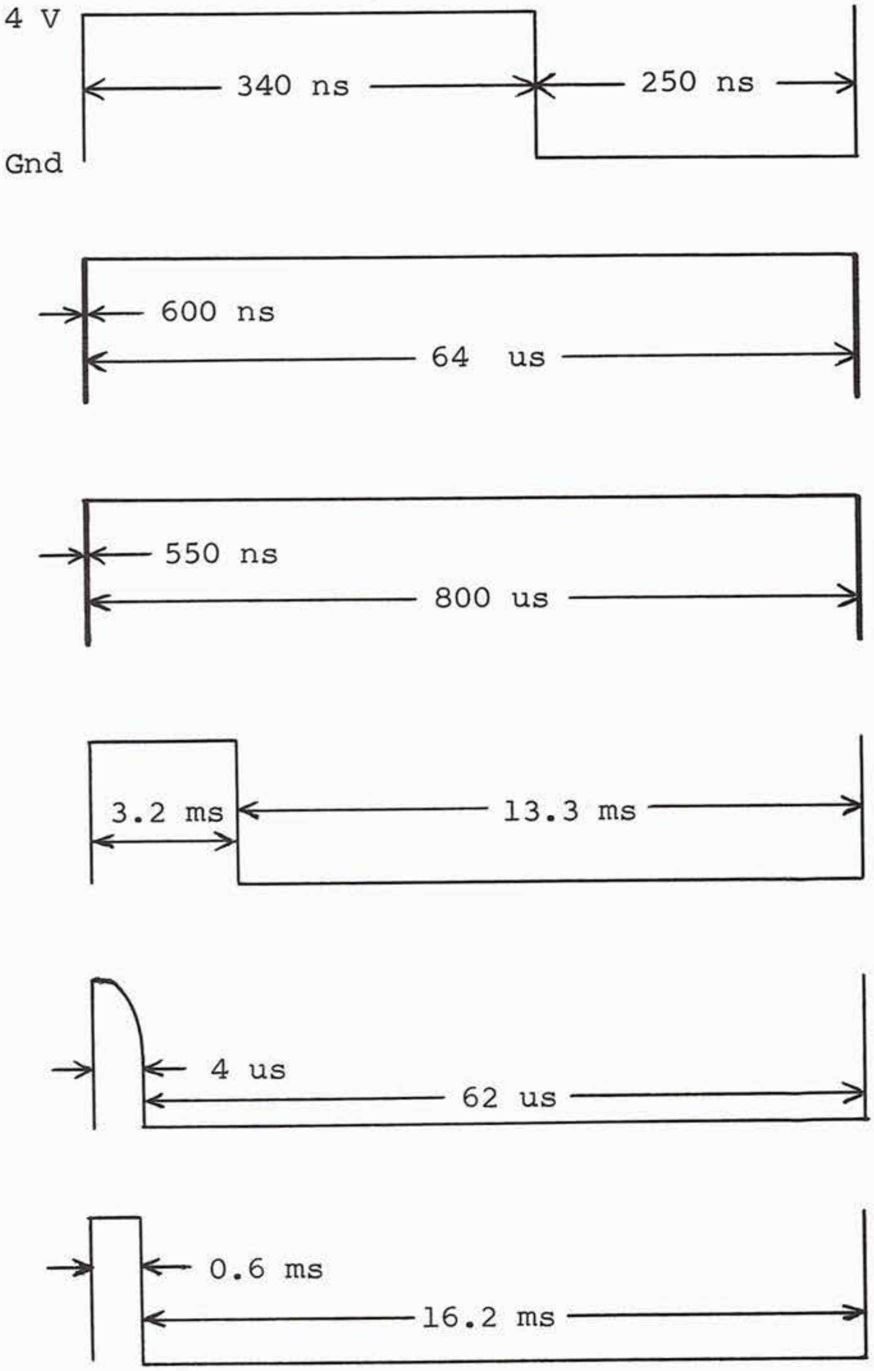


Figure 3-5. Display section timing waveforms.

CAUTION

DO NOT CONNECT THE Sol-PC VIDEO OUTPUT TO A MONITOR OR TV RECEIVER THAT IS NOT EQUIPPED WITH AN ISOLATION TRANSFORMER. (SEE PAGE AVI-7 IN APPENDIX VI.)

( ) Set VR2 (VERT) and VR1 (HORIZ) on the Sol-PC to their mid-range settings. Turn monitor on and apply power to the Sol-PC.

( ) The display raster will be pulled in. Using the monitor Vertical Hold, you should be able to obtain a slow roll (black horizontal bar moves slowly down the screen) and a stationary raster. Using the monitor Horizontal Hold, you should be able to adjust for an out of sync raster (numerous black lines cutting across the raster) and a stable raster. If you cannot obtain these conditions, locate and correct the cause before proceeding.

NOTE

For a stable presentation, a few monitors (especially modified TV sets) may require a higher sync amplitude than that supplied by the Sol-PC. In such cases, increase sync amplitude by reducing the value of R80. DO NOT DECREASE R80 BELOW 225 OHMS.

( ) If the synchronization circuits are operating correctly, turn monitor and power supply off, disconnect the power cable and go on to Step 27.

( ) Step 27. Install the following IC's in the indicated locations. Pay careful attention to the proper orientation.

NOTE

Dots on the assembly drawing and PC board indicate the location of pin 1 of each IC.

(Step 27 continued on Page III-23.)

IC NO.	TYPE
( ) U1*	4029*
( ) U2	74LS175 or 25LS175
( ) U11*	4029*
( ) U12	74LS157
( ) U13	74LS175 or 25LS175
( ) U25*	MCM6574 or MCM6575*
( ) U26	74LS175 or 25LS175
( ) U27	74LS175 or 25LS175
( ) U29	74LS367
( ) U30	74LS157
( ) U32	74LS157 or 25LS157
( ) U41	74166
( ) U42	74LS175 or 25LS175
( ) U44	74LS00
( ) U61	74LS10
( ) U89	74LS367

*MOS device. Refer to CAUTION on Page III-8.

( ) Step 28. Check display circuits.

( ) Set S1 switches as follows:

No. 1 through 5: OFF

No. 6: ON

( ) Remove U42 and bend pin 6 out 45° to its normal position. (See Figure 3-6.) Re-install U42 with pin 6 out of the socket.

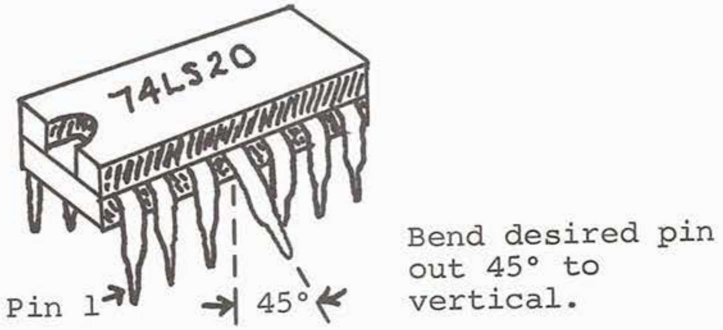


Figure 3-6. Bending selected pins on U42, 59 and 75 (U59 shown).

( ) Remove U59 and bend pin 4 in same manner as U42. Re-install U59 with pin 4 out of the socket.

(Step 28 continued on Page III-24.)

( ) Remove U75 and bend pin 5 in same manner as U42. Reinstall U75 with pin 5 out of the socket.

( ) Using #24 wire, install the following TEMPORARY jumpers in the sockets for U14 through U21. Double check jumpers after installing for correctness. (See Figure 3-7.)

IC SOCKET	JUMPER
( ) U14	Pin 12 to 6
( ) U15	Pin 12 to 5
( ) U16	Pin 12 to 4
( ) U17	Pin 12 to 8
( ) U18	Pin 12 to 2
( ) U19	Pin 12 to 7
( ) U20	Pin 12 to 1
( ) U21	Pin 12 to 16

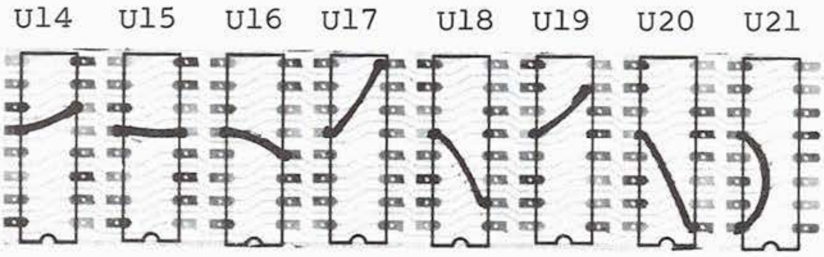


Figure 3-7. U14 through U21 socket jumpers.

( ) Turn monitor on and apply power to Sol-PC.

( ) Momentarily ground pin 1 of U2 and pin 5 of U75. The display shown in Figure 3-8 should appear on the monitor screen.

( ) If the display circuits do not pass this test, determine and correct the cause before proceeding with assembly.

( ) If the display circuits are operating correctly:

- ( ) Turn monitor and power supply off and disconnect the power cable.
- ( ) Remove jumpers from U14 through U21 sockets.
- ( ) Bend pin 6 on U42, pin 4 on U49 and pin 5 on U75 back to their normal position and re-install these three IC's in their appropriate sockets.

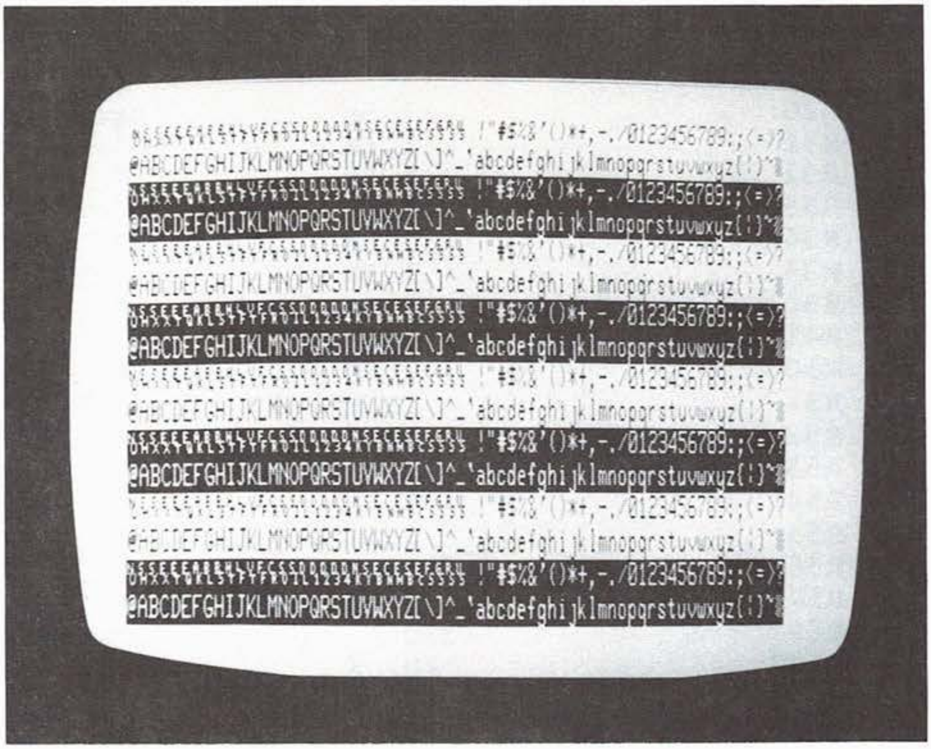


Figure 3-8. Display circuits test pattern with 6575 character generator as U25. 6574 is the same except graphic control characters are displayed.

( ) Step 29. Install 91L02APC or 2102L1PC IC's in locations U14 through U21. Dots on the assembly drawing and PC board legend indicate the location of pin 1 of each IC.

CAUTION

IC's U14 THROUGH U21 ARE MOS DEVICES. REFER TO CAUTION ON PAGE III-8 BEFORE YOU INSTALL THESE IC's.

( ) Step 30. Install the following resistors in the indicated locations. Bend leads to fit distance between mounting holes, insert leads, pull down snug to board, solder and trim.

LOCATION	VALUE (ohms)	COLOR CODE
( ) R12	1.5K	brown-green-red
( ) R18	10 K	brown-black-orange

(Step 30 continued on Page III-26.)

LOCATION	VALUE (ohms)	COLOR CODE
( ) R20	1.5K	brown-green-red
( ) R31	1.5K	" " "
( ) R32	1.5K	" " "
( ) R33	1.5K	" " "
( ) R34	1.5K	" " "
( ) R35	1.5K	" " "
( ) R36	1.5K	" " "
( ) R41	1.5K	" " "
( ) R50	1.5K	" " "
( ) R51	1.5K	" " "
( ) R52	1.5K	" " "
( ) R53	1.5K	" " "
( ) R54	1.5K	" " "
( ) R55	1.5K	" " "
( ) R56	1.5K	" " "
( ) R57	1.5K	" " "
( ) R58	330	orange-orange-brown
( ) R107	10 K	brown-black-orange
( ) R108	10 K	" " "
( ) R109	10 K	" " "
( ) R110	10 K	" " "
( ) R111	10 K	" " "
( ) R112	10 K	" " "
( ) R113	10 K	" " "
( ) R114	10 K	" " "
( ) R115	1.5K	brown-green-red

( ) Step 31. Install diode D7 (1N4148 or 1N914) in its location between U46 and U47. Position D7 with its dark band (cathode) at the bottom.

( ) Step 32. Install 20-pin header in location J3 (keyboard interconnect) between U64 and U65. Position header so pin 1 is in the upper left corner. (An arrow on the connector points to pin 1.)

( ) Step 33. In the jumper area labeled $\overline{\text{PHTM}}$ on the assembly drawing (below U64), install Augat pins in mounting holes F and G. (Refer to "Installing Augat Pins" in Appendix IV.) Using #24 bare wire, install a jumper between pins F and G.

( ) Step 34. In the jumper area labeled RST on the assembly drawing (between U76 and U77), install Augat pins in mounting holes N and P. (Refer to "Installing Augat Pins" in Appendix IV.) Using #24 bare wire, install a jumper between pins N and P.

( ) Step 35. Install the following IC's in the indicated locations. Pay careful attention to the proper orientation.

NOTE

Dots on the assembly drawing and PC board indicate the location of pin 1 of each IC.

	IC NO.	TYPE
( )	U45	74LS04
( )	U46	8T380 or 8836
( )	U48	74LS00
( )	U50	74LS367
( )	U54	74LS04
( )	U63	74LS109
( )	U64	74LS109
( )	U67	8T97
( )	U68	8T97
( )	U76	74LS175
( )	U94	74LS367
( )	U107	74LS367

( ) Step 36. Apply power to Sol-PC and make the following voltage measurements:

MEASUREMENT POINT	VOLTAGE*
Pin 11 of U105 Socket	-5 V dc ± .25 V
Pin 20 of U105 Socket	+5 V dc ± .25 V
Pin 28 of U105 Socket	+12 V dc ± .6 V
Pin 1 of U51 Socket	+5 V dc ± .25 V
Pin 2 of U51 Socket	-12 V dc ± .6 V

*All voltages referenced to ground.

( ) If any voltages are incorrect, locate and correct the cause before going on to Step 37.

( ) If voltages are correct, turn power supply off, disconnect power cable and go on to Step 37.

( ) Step 37. Install the following IC's in the indicated locations. Pay careful attention to the proper orientation.

NOTE

Dots on the assembly drawing and PC board indicate the location of pin 1 of each IC.

(Step 37 continued on Page III-28.)

IC NO.	TYPE
( ) U51*	TMS6011NC*
( ) U105* #	8080,8080A or 9080A* #

*MOS device. Refer to CAUTION on Page III-8.

#Note that pin 1 of this IC is in the upper left corner.

( ) Step 38. Perform Functional Test No. 1 of CPU circuits.

( ) Set S1 switches as follows:

No. 1 through 5: OFF

No. 6: ON

( ) Turn monitor on and apply power to Sol-PC.

( ) Momentarily ground pin 1 of U2. You should see a full display (64 characters x 16 lines) on the monitor.

( ) Momentarily ground pin 2 of U75. The display should blank while pin 2 of U75 is grounded. When you remove the ground, the display shown in Figure 3-9 on Page III-29 should appear.

NOTE

The pattern shown in Figure 3-9 (delete characters) results from all bits of the DIO Bus being high. If you do not see the delete characters, one or more bits of the DIO bus are low. Consult the MCM6575 or MCM6574 pattern, as appropriate, in Section VIII of this manual to determine which bits are low.

( ) If the test fails, determine and correct the cause before proceeding with assembly.

( ) If the Sol-PC passes this test, turn monitor and power supply off, disconnect power cable and proceed to Step 39.

( ) Step 39. Install the following IC's in the indicated locations. Pay careful attention to the proper orientation.

(Step 39 continued on Page III-29.)

NOTE

Dots on the assembly drawing and PC board indicate the location of pin 1 of each IC.

IC NO.	TYPE
( ) U80	8T97#
( ) U81	8T97#

#DO NOT substitute.

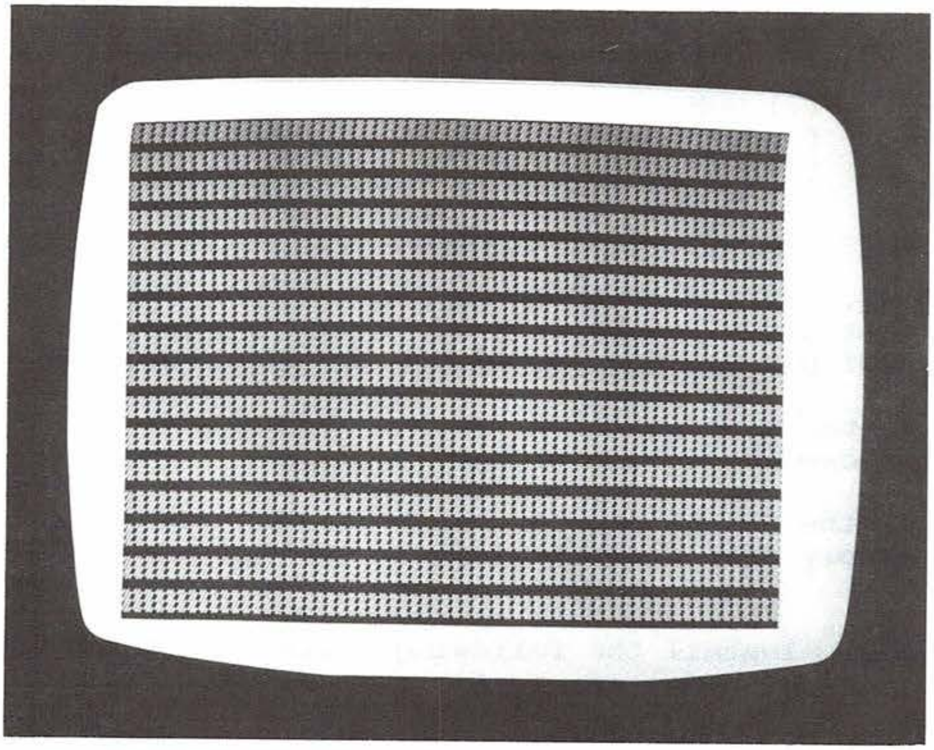


Figure 3-9. CPU Functional Test No. 1 display, 6574 or 6575 character generator (U25).

( ) Step 40. Perform Functional Test No. 2 of CPU circuits.

( ) Check that S1 switches are set as specified in Step 38.

( ) Turn monitor on and apply power to Sol-PC.

( ) Momentarily ground pin 1 of U2 and pin 2 of U75. The display shown in Figure 3-10 on Page III-31 should appear on the monitor.

( ) If the test fails, determine and correct the cause before proceeding with assembly.

( ) If the Sol-PC passes this test, turn monitor and power supply off, disconnect power cable and proceed to Step 41.

( ) Step 41. Install the following IC's in the indicated locations. Pay careful attention to the proper orientation.

NOTE

Dots on the assembly drawing and PC board indicate the location of pin 1 of each IC.

IC NO.	TYPE
( ) U65	74LS253
( ) U66	74LS253
( ) U78	74LS253
( ) U79	74LS253
( ) U93	74LS175
( ) U106	74LS175
( ) U70	74LS109

( ) Step 42. Turn monitor on, apply power to Sol-PC and perform the test described in Step 40, except ground pin 5 of U75 instead of pin 2. You should get the same results.

( ) If the test fails, determine and correct the cause before proceeding with assembly.

( ) If the Sol-PC passes this test, turn monitor and power supply off, disconnect power cable and proceed to Step 43.

( ) Step 43. Install the following resistors in the indicated locations. Bend leads to fit distance between mounting holes, insert leads, pull down snug to board, solder and trim.

LOCATION	VALUE (ohms)	COLOR CODE
( ) R13	1.5K	brown-green-red
( ) R14	1.5K	" " "
( ) R15	1.5K	" " "
( ) R60	1.5K	" " "

( ) Step 44. Using two 4-40 x 5/8 binder head screws, two #4 insulating washers, two lockwashers and hex nuts, install 30-pin right angle edge connector in location J5. Insert screws from back (solder) side of board and place an insulating washer on each screw on front (component) side of board. Position connector with socket side facing right, place over screws and seat pins in mounting holes. Then place lockwasher on each screw, start nuts and tighten. Solder pins to board.

( ) Step 45. Using four 4-40 x ¼ binder head screws, lockwashers and hex nuts, install two brackets (Sol-1040) for personality module in area to right of J5. Position brackets over the mounting holes as shown in Figure 3-11. Insert screws from front (component) side of board, place lockwasher on each screw on back (solder) side of board, start nuts and tighten.

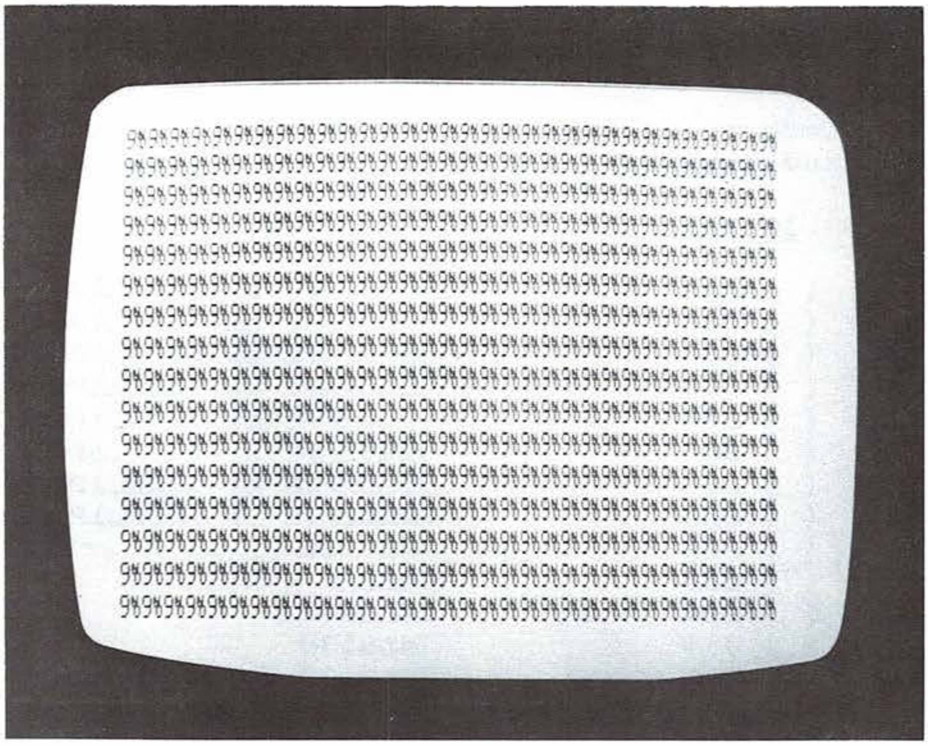


Figure 3-10. CPU Functional Test No. 2 display, 6575 character generator (U25). 6574 displays: 9 □ 9 □ 9 □ etc.

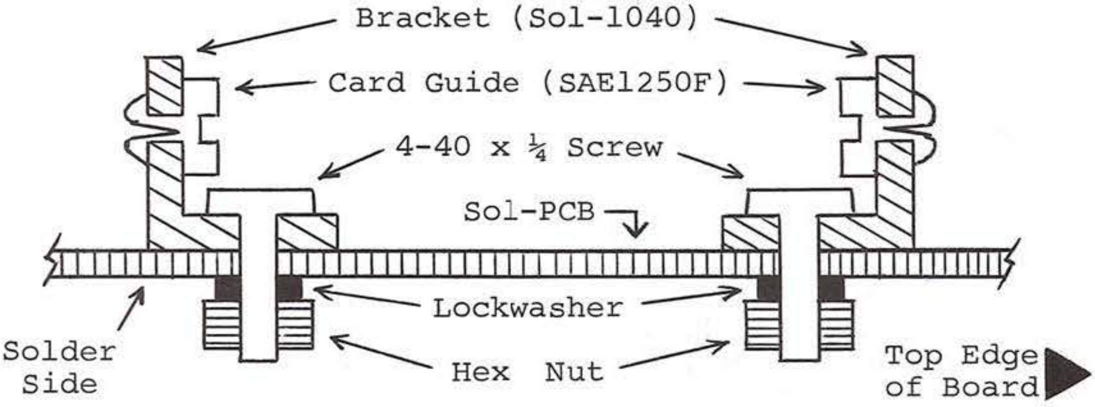


Figure 3-11. Personality module bracket/guide installation (Viewed from right end of Sol-PCB).

( ) Step 46. Attach plastic card guide (SAE1250F) to each of the brackets installed in Step 45. (See Figure 3-11.) Insert posts on guides into bracket holes and push in until they snap into place.

( ) Step 47. Install the following IC's in the indicated locations. Pay careful attention to the proper orientation.

NOTE

Dots on the assembly drawing and PC board indicate the location of pin 1 of each IC.

IC NO.	TYPE
( ) U3*	91L02APC or 2102L1PC*
( ) U4*	91L02APC or 2102L1PC*
( ) U5*	91L02APC or 2102L1PC*
( ) U6*	91L02APC or 2102L1PC*
( ) U7*	91L02APC or 2102L1PC*
( ) U8*	91L02APC or 2102L1PC*
( ) U9*	91L02APC or 2102L1PC*
( ) U10*	91L02APC or 2102L1PC*
( ) U22	74LS136
( ) U23	74LS20
( ) U24	74LS04
( ) U34	74LS138
( ) U35	74LS138
( ) U36	74LS138
( ) U53	74LS02 or 9LS02
( ) U71	74LS367
( ) U83	74LS20

*MOS device. Refer to CAUTION on Page III-8.

( ) Step 48. Test memory and decoder circuits.

( ) Set S1 switches as specified in Step 38.

( ) Turn monitor on and apply power to Sol-PC.

( ) Ground pin 1 of U2. You should see the same display as shown in Figure 3-10 on Page III-31. In this case, however, there should be a vertical "flickering" movement with an apparent flicker rate of approximately three times per second.

( ) Turn Switch No. 1 of S1 to ON. The flicker should stop.

(Step 48 continued on Page III-33.)

( ) If the test fails, determine and correct the cause before proceeding with assembly.

( ) If the Sol-PC passes this test, turn monitor and power supply off, disconnect power cable, set Switch No. 1 of S1 to OFF and go on to Step 49.

( ) Step 49. Assemble personality module if you have not yet done so. (See Section IV.) If you have, go to Step 9 in Section IV and complete the personality module assembly.

( ) Step 50. Install the following resistors in the indicated locations. Bend leads to fit distance between mounting holes, insert leads, pull down snug to board, solder and trim.

	LOCATION	VALUE (ohms)	COLOR CODE
( )	R21	470	yellow-violet-brown
( )	R22	470, ½ watt	" " "
( )	R23	470, ½ watt	" " "
( )	R24	1.5K	brown-green-red
( )	R25	10 K	brown-black-orange
( )	R26	10 K	" " "
( )	R27	470	yellow-violet-brown
( )	R28	10 K	brown-black-orange
( )	R29	10 K	" " "
( )	R37	1.5K	brown-green-red
( )	R38	1.5K	" " "
( )	R39	5.6K	green-blue-red
( )	R40	1.5K	brown-green-red
( )	R42	1.5K	" " "
( )	R43	1.5K	" " "
( )	R44	1.5K	" " "
( )	R45	330	orange-orange-brown
( )	R46	5.6K	green-blue-red
( )	R47	10 K	brown-black-orange
( )	R48	10 K	" " "
( )	R49	1.5K	brown-green-red
( )	R59	1.5K	" " "
( )	R61	1.5K	" " "
( )	R62	5.6K	green-blue-red
( )	R63	5.6K	" " "
( )	R64	330	orange-orange-brown
( )	R65	330	" " "
( )	R66	330	" " "
( )	R67	330	" " "
( )	R68	330	" " "
( )	R69	330	" " "
( )	R70	330	" " "
( )	R71	330	" " "

(Step 50 continued on Page III-34.)

	LOCATION	VALUE (ohms)	COLOR CODE
( )	R72	680	blue-gray-brown
( )	R73	680	" " "
( )	R74	680	" " "
( )	R75	680	" " "
( )	R76	680	" " "
( )	R77	680	" " "
( )	R78	680	" " "
( )	R79	680	" " "
( )	R92	5.6K	green-blue-red
( )	R93	1.5K	brown-green-red
( )	R94	10 K	brown-black-orange
( )	R95	15 K	brown-green-orange
( )	R116	1.5K	brown-green-red

( ) Step 51. Install the following capacitors in the indicated locations. Take care to observe the proper value and type for each installation. Bend leads outward on solder (back) side of board, solder and trim. (Refer to NOTE in Step 2.)

	LOCATION	VALUE	TYPE
( )	C29	.1 ufd	Disc
( )	C30	330 pfd	Disc

( ) Step 52. Install diodes D1 (1N4148 or 1N914), D2 (1N4001) and D3 through D6 (1N4148 or 1N914) in their locations in the area of U39. Position all diodes with their dark band (cathode) to the right.

( ) Step 53. Install the following DIP switches in the indicated locations. Take care to observe proper orientation.

	LOCATION	TYPE	ORIENTATION
( )	S2	8-position	Switch No. 1 at top
( )	S3	8-position	Switch No. 1 at top
( )	S4	6-position	Switch No. 1 at top

( ) Step 54. Install Q1 (2N2907 or 2N3460) in its location between U55 and U56. The emitter lead (closest to tab on can) is oriented toward the bottom and the base lead toward the right. Push straight down on transistor until it is stopped by the leads. Solder and trim.

( ) Step 55. Using two 4-40 x 7/16 binder head screws, hex nuts and lockwashers, install 25-pin female connector in location J1 (serial I/O interface). Position connector with socket side facing right and insert pins into their holes in the circuit board. Insert screws from back (solder) side of board, place lockwasher on each screw, start nuts and tighten. Then solder connector pins to board.

( ) Step 56. Using two 4-40 x 7/16 binder head screws, hex nuts and lockwashers, install 25-pin male connector in location J2 (parallel I/O interface). Install J2 in the same manner as you did J1.

( ) Step 57. Install Augat pins in mounting holes K, L and M. (Refer to "Installing Augat Pins" in Appendix IV.) These holes are located between U85 and U86. No jumper will be installed.

( ) Step 58. Install the following IC's in the indicated locations. Pay careful attention to the proper orientation.

NOTE

Dots on the assembly drawing and PC board indicate the location of pin 1 of each IC.

IC NO.	TYPE
( ) U37	74LS367
( ) U38*	1489A*
( ) U39#	4N26#
( ) U52	74LS109
( ) U55	74LS00
( ) U56	1458CP or 1558CP
( ) U57	7406
( ) U58	8T94
( ) U72	74LS109
( ) U73	74LS109
( ) U84*	4029*
( ) U85*	4046*
( ) U86*	4024*
( ) U95	74173
( ) U96	74173
( ) U97	74175

*MOS device. Refer to CAUTION on Page III-8.

#Solder this IC in its location. See "Loading DIP Devices" in Appendix IV.

( ) Step 59. Check input/output (I/O) circuits.

NOTE

The parallel I/O interface should be tested with the device you will be using. Refer to "I/O Interfacing" in Section VII.

(Step 59 continued on Page III-36.)

To check the serial I/O circuits, proceed as follows:

( ) Set S1 as in previous test,
Set S2 switches all OFF,
Set S3 switches all OFF, except S3-1 ON,
Set S4 switches all OFF

( ) Set <u>all</u> S4 switches to OFF.

( ) Connect Sol-PC video output cable to monitor, turn monitor on and apply power to Sol-PC.

( ) Set Sol-PC to local by depressing LOCAL key on keyboard to turn keyboard indicator light on.

( ) Data entered from the keyboard should appear on the monitor.

( ) If the Sol-PC fails this test, locate and correct the cause before proceeding.

( ) If the Sol-PC passes this test, turn monitor and power supply off, disconnect power cable and video output cable and go on to Step 60.

( ) <u>Step 60</u>. Install the following resistors in the indicated locations. Bend leads to fit distance between mounting holes, insert leads, pull down snug to board, solder and trim.

	<u>LOCATION</u>	<u>VALUE (ohms)</u>	<u>COLOR CODE</u>
( )	R117	10 K	brown-black-orange
( )	R118	10 K	" " "
( )	R119	10 K	" " "
( )	R139	1.0M	brown-black-green
( )	R140	10 K	brown-black-orange
( )	R141	150 K	brown-green-yellow
( )	R142	10 K	brown-black-orange
( )	R143	1 M	brown-black-green
( )	R144	47 K	yellow-violet-orange
( )	R145	10 K	brown-black-orange
( )	R146	10 K	" " "
( )	R147	2.2M	red-red-green
( )	R148	100 K	brown-black-yellow
( )	R149	100	brown-black-brown
( )	R150	470	yellow-violet-brown
( )	R151	5.6K	green-blue-red
( )	R152	150 K	brown-green-yellow
( )	R153	100 K	brown-black-yellow
( )	R154	100 K	" " "
( )	R155	6.8, ½ watt	blue-grey-gold
( )	R156	6.8, ½ watt	blue-grey-gold
( )	VR3	50 K	Potentiometer

( ) Step 61. Install the following capacitors in the indicated locations. Take care to observe the proper value and type for each installation. Bend leads outward on solder (back) side of board, solder and trim. (Refer to NOTE in Step 2.)

CAUTION

REFER TO FOOTNOTE AT END OF THIS STEP BEFORE INSTALLING C67.

	LOCATION	VALUE (ufd)	TYPE
( )	C47	.001	Disc
( )	C48	.047	"
( )	C49	.001	"
( )	C50	.01	Mylar Tubular
( )	C51	.1	Disc
( )	C66	.1	"
( )	C67*	1	Tantalum
( )	C68	.1	Disc
( )	C69	.1	"
( )	C70	.1	"
( )	C71	.001	"
( )	C72	.001	Mylar Tubular
( )	C73	.047	Disc
( )	C74	470 pfd	"

*Install C67 with "+" lead at top right.

( ) Step 62. Install miniature phone jacks in locations J6 and J7 located to the right of U101. Position J6 and J7 with jack facing right, insert pins in mounting holes and solder.

( ) Step 63. Install subminiature phone jacks in locations J8 and J9 in lower right corner of board. Install J8 and J9 as you did J6 and J7.

( ) Step 64. Install Q3 (2N4360) in its location to the left of C67. Install Q3 with its flat "side" at the bottom. Push straight down on transistor until it is stopped by the leads, solder and trim.

CAUTION

THE 2N4360 IS STATIC SENSITIVE. REFER TO CAUTION ON PAGE III-8.

( ) Step 65. Install Q4 and Q5 (2N2222) in their locations above and to the left of U108. For both transistors, the emitter lead (closest to tab on can) is oriented toward the left and the base lead toward the right. Push straight down on transistor until it is stopped by the leads, solder and trim.

( ) Step 66. Install diodes D13 and D14 (1N4001) in their locations in the lower right corner of the board. Position both diodes with their dark band (cathode) at the bottom.

( ) Step 67. Install DIP reed relays in locations K1 and K2 to the right of U113. Be sure to install K1 and K2 with their end notch at the bottom)pin 1 in lower right corner). These relays are soldered to the board. (Refer to "Loading DIP Devices" in Appendix IV.)

( ) Step 68. Install the following IC's in the indicated locations. Pay careful attention to the proper orientation.

NOTE

Dots on the assembly drawing and PC board indicate the location of pin 1 of each IC.

	IC NO.	TYPE
( )	U69*	TMS6011NC*
( )	U98*	4023*
( )	U99*	4030*
( )	U100*	4013*
( )	U101*	4027*
( )	U108	1458CP or 1558CP
( )	U109*	4049*
( )	U110*	4046*
( )	U111*	4019*
( )	U112*	4520*
( )	U113*	4013*

*MOS device. Refer to CAUTION on Page III-8.

( ) Step 69. Install Augat pins in mounting holes H, I and J (located to left of C70). (Refer to "Installing Augat Pins" in Appendix IV.) Using #24 bare wire, install a jumper between pins I and J.

( ) Step 70. Adjust VR3.

( ) Using a cable with a male phono jack on both ends, connect ACI audio output (J6) to ACI audio input (J7).

( ) Apply power to Sol-PC.

( ) Set VR3 fully clockwise (CW).

( ) Measure the DC voltage at pin 13 of U110 and write the measured voltage down. (Call this Voltage A.)

( ) Set VR3 fully counterclockwise (CCW).

(Step 70 continued on Page III-39.)

( ) Measure the DC voltage at pin 13 of U110 and write the measured voltage down. (Call this Voltage B.)

( ) Add Voltages A and B and divide the sum by 2. (Call the result Voltage C.) An example follows:

Voltage A (VR3 full CW):	3.45 V dc
Voltage B (VR3 full CCW):	1.80 V dc
A + B =	5.25 V dc

Voltage C = 5.25 V dc / 2 = 2.63 V dc

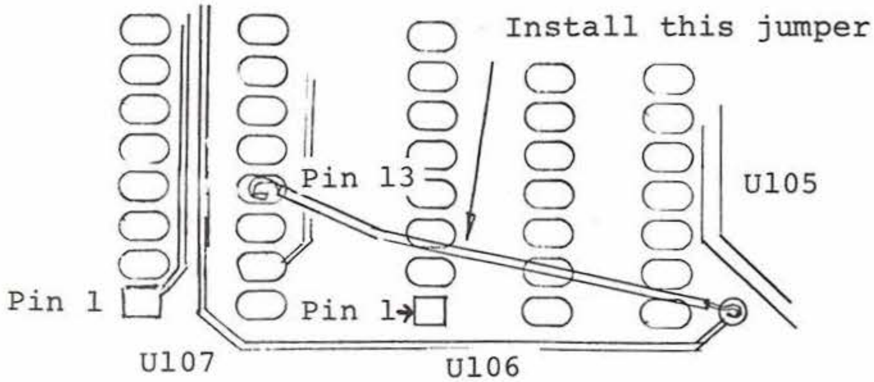
( ) Adjust VR3 so that the voltage at pin 13 of U110 equals Voltage C. (In the preceding example this would be 2.63 V dc.)

( ) Step 71. If your recorder has only a microphone input, remove the I-to-J jumper you installed in Step 69 and install a jumper (#24 bare wire is recommended) between the I and H pins.

Otherwise, leave the I-to-J jumper in and go on to Step 72.

( ) Step 72. Install 100-pin edge connector, J11. Using two 4-40 x 7/16 binder head screws, install 100-pin edge connector in location J11 (center of PC board). Seat the pins in the mounting holes. Then thread screws from front (component) side of board into the threaded inserts that are pre-installed in the J11 mounting holes. Tighten screws and solder pins to board.

( ) Step 73. Look on the rear of the board, on the component side, where the Personality Module plugs in, for a mark "Rev E". If your board is marked this way, complete this step, otherwise ignore this step. Connect a jumper of #24 a.w.g. insulated wire between pin 13 of U107 and the feedthrough pad adjacent to pin 21 of U105. Solder, check for solder bridges, and trim excess wire strands if needed. The installed jumper is shown below.



Solder side of board shown

3.6.4 Modification for 625 Line Video

The European televisions standard defines a raster of 625 lines at a field rate of 50 Hz. The horizontal rate of the U.S. standard, 15,750 Hz., is maintained. Only the number of scan lines on the screen is increased.

The Video Display Generator section may be modified for the 50 Hz. standard by following the additional steps below. The effect of the modification is to increase the modulus of the counter U62 to eight during $\overline{\text{VDISP}}$. This results in four extra character lines (52 scan lines) between the bottom and top of the display area, for a total of 312 scan lines per field and 624 scan lines per frame.

The field rate should be close enough to 50 Hz. to reduce any swim effects to less than 0.1 Hz. Some difficulty may be encountered in obtaining centering of the display within the frame. This is because the stand-off time to VSYNC from the bottom of the display is unchanged from the 60 Hz. standard. If objectionable, increase the value of resistor R100 which is in series with the VPOS control.

To convert for 50 Hz., perform these additional steps:

( ) Locate U62 on the component side legend. Find pin 5 of this IC on the component (front) side of the board. Cut the "V"-shaped trace connecting pin 5 to the nearby pad designated "AF", using a sharp exacto blade or scribe, so that there is no continuity between these pads.

( ) Bend a small piece of bare wire, such as a resistor clipping, into a loop to form a jumper between pad "AF", and the adjacent pad "AG". Insert the jumper, pull close to the board, solder, and trim the leads.

If this modification is made, change the schematic, X-18, to show that pin 5 of U62 now connects to pin 4 (ground), instead of pin 6 as shown.

4.1 PARTS AND COMPONENTS

When ordering your Sol, you selected one of two types of Personality Modules: CONSOL Or SOLOS. The outer carton of your kit is stamped with the Personality Module type. Both use the same PC board marked 2708, assembly #107000, and differ only in the type of ROM's and their programming. (An alternative PC board marked 5204 and designed for type 5204 EPROM's is also available but not supplied with this kit. Schematic diagram X-4 and assembly drawing X2Ø refer to this alternative board.) Check all parts against Table 4-1 below. If you have difficulty identifying any parts, refer to Figure 3-1 on page III-5. One of two kits, using the same PC board: 2708-0 or 2708-1 may be supplied. The 2708-0 version uses one 9216 masked ROM which has no window on top of the IC package. The 2708-1 version uses two 2708 EPROM's which have windows.

Table 4-1. PM2708 Personality Module Parts List.

1	PM2708 PC Board	1 or 4*	1-ufd Capacitor, Tantalum Dipped
1 or 2*	9216 ROM or 2708 EPROM's with Personality program	1 or 2*	24-pin DIP Socket
1	74LSØ8	1	14-pin DIP Socket
0 or 2*	1N5231B Zener Diode	1	Handle Bracket (Sol-1045)
3 or 4*	10K ohm, ¼ watt, 5% Res.	2	2-56X1/8 Binder Head Screw
0 or 2*	100 ohm, ½ watt, 5% Res.		
1	.047-ufd Disc Ceramic		

* These are the quantities of parts used in the 2708-1 version.

4.2 ASSEMBLY TIPS

For the most part the assembly tips given in Paragraph 3.2 of Section III (Page III-1) apply to assembling the personality module.

4.3 ASSEMBLY PRECAUTIONS

For the most part the assembly precautions given in Paragraph 3.3 in Section III (Page III-6) apply.

4.4 REQUIRED TOOLS, EQUIPMENT AND MATERIALS

The following tools, equipment and materials are recommended for assembling the personality module.

1. Needle nose pliers
2. Diagonal cutters
3. Screwdriver
4. Soldering iron, 25 watt
5. 60-40 rosin-core solder (supplied)
6. Small amount of #24 solid wire

4.5 ORIENTATION

Capacitor location C2 will be located in the upper left hand corner of the board when the edge connector is positioned at the left end of the board. In this position the component (front) side of the board is facing up. Subsequent position references related to the personality module circuit board assume this orientation.

4.6 ASSEMBLY-TEST

4.6.1 Circuit Board Check

( ) Visually check circuit board for broken traces, shorts (solder bridges) between traces and similar defects.

( ) Check circuit board to insure that the +5-volt bus, +12 volt bus and -12-volt bus are not shorted to each other or to ground. Using an ohmmenter, make the following measurements (refer to personality module assembly drawing in Section X):

( ) +5 volt Bus Test. On U1, measure between pin 12, (ground) and pin 24 (+5 volts). There should be no continuity.

( ) -5 volt Bus Test. On U1 and U2, measure between pin 12 (ground) and pin 21 (-5 volts). There should be no continuity.

( ) +12 volt Bus Test. Also on U1, measure between pin 12 (ground) and the bottom edge connector pin on the component side of the board marked A1.

( ) Inter-bus Test. On U1, measure between pins 12 and 21, then between edge connector pin A1 and pins 21, then 12. There should be no continuity in any of these measurements.

If visual inspection reveals any defect, or you measure continuity in any of the preceding tests, return the board to Processor Technology for replacement. If the board is not defective, proceed to next paragraph.

4.6.2 Assembly-Test Procedure

Refer to personality module assembly drawing X-6.

CAUTION

THE MEMORY IC'S USED ON THE PERSONALITY MODULE ARE MOS DEVICES. THEY CAN BE

(CAUTION continued on Page IV-3)

DAMAGED BY STATIC ELECTRICITY DISCHARGE. HANDLE THESE IC's SO THAT NO DISCHARGE FLOWS THROUGH THE IC. AVOID UNNECESSARY HANDLING AND WEAR COTTON, RATHER THAN SYNTHETIC, CLOTHING WHEN HANDLING MOS IC's. (STATIC DISCHARGE PROBLEMS ARE MUCH WORSE IN LOW HUMIDITY CONDITIONS.)

( ) Step 1. Install DIP sockets. Install each socket in the indicated location with its end notch oriented as shown on the circuit board and assembly drawing. Take care not to create solder bridges between the pins and/or traces.

INSTALLATION TIP

Insert socket pins into mounting pads of appropriate location. On back (solder) side of board, bend pins at opposite corners of socket (e.g. pins 1 and 9 on a 16-pin socket) outward until they are at a 45° angle to the board surface. This secures the socket until it is soldered. Repeat this procedure with each socket until all are secured to the board. Then solder the pins on all sockets.

LOCATION	TYPE SOCKET
( ) U1	24 pin
( ) U2*	24 pin*
( ) U3	14 pin

*Used on 2708-1 version only.

( ) Step 2. Install the following resistors in the indicated locations. Install these resistors parallel with the board. Bend leads by using needle nose pliers to grip the resistor lead right next to the resistor body, and bend the portion of the lead on the other side of the pliers with your finger. The bend must be the right distance from the resistor body for the resistor to fit easily into its two holes. Insert the leads into the two holes, and from the opposite side of the board pull the leads to bring the resistor body down to touch the board. Bend the leads outward on the solder (back) side of the board so the resistors do not slip out of position.

LOCATION	VALUE	COLOR CODE
( ) R1*	100 ohms	brown-black-brown
( ) R2*	100 ohms	brown-black-brown
( ) R3	10K	brown-black-orange
( ) R4*	10K	brown-black-orange
( ) R5	10K	brown-black-orange
( ) R6	10K	brown-black-orange

*not used on 2708-0 version

( ) Step 3. Install 1N5231B Zener Diodes in locations Z1, and Z2 if you have the 2708-1 version. Form the leads as in Step 2. Insert the diodes so that the white band on the diode is in the position indicated by the legend. Bend the leads outward to retain the diodes, then solder and trim the leads.

( ) Step 4. Install the following capacitors in the indicated locations. Take care to observe the proper value, type and orientation for each installation. On the dipped tantalum capacitors, the "+" lead is the one which is closest to the "+" marking on the body of the capacitor. Insert this lead in the hole marked "+" on the PC board legend. After inserting C5, remove it from the board before soldering to clear wax from the leads and holes. After inserting all capacitors, pull them close to the board and bend the leads outward to secure them. Solder and trim all leads.

LOCATION	VALUE (ufd)	TYPE
( ) C1*	1	Dipped Tantalum
( ) C2	1	Dipped Tantalum
( ) C3*	1	Dipped Tantalum
( ) C4*	1	Dipped Tantalum
( ) C5	.047	Disc Ceramic

*not used on 2708-0 version

( ) Step 5. Check for +5, +12, and -12 volt bus-to-ground shorts. Using an ohmmeter on OHMS times 1K or OHMS times 10K scale, make the following measurements. A typical reading is 1 Megohm. A reading less than 10K indicates a short.

( ) Measure between edge connector pins A2 and A15.

( ) Measure between edge connector pins A14 and A15.

( ) Measure between edge connector pins A1 and A15.

( ) If any measurement indicates a short, find and correct the problem before proceeding.

( ) Step 6. Using two 2-56 x 1/8" binder head screws, install handle bracket (Sol-1045). Position bracket on front (component) side of board at the right end as shown in Figure 4-2. Align bracket holes with mounting holes in board, insert screws from back (solder) side of board and drive into bracket. No nuts are needed since the bracket holes are tapped.

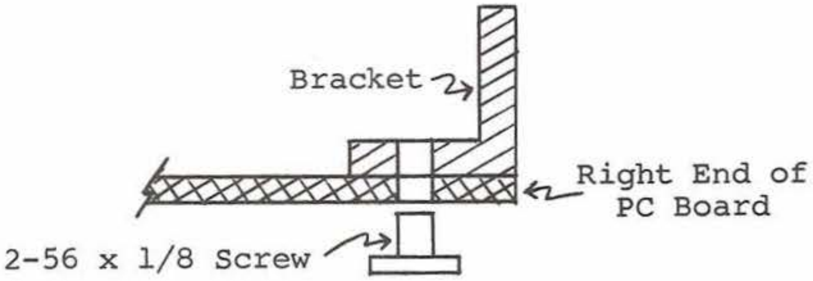


Figure 4-2. Handle bracket (Sol-1045) installation.

( ) Step 7. If you have a 2708-0 version with the 9216 ROM (windowless), omit this step. If you have the 2708-1 version, find the area above the U1 socket where the legend reads "-5V 21 CO 19 +12V." This legend designates five PC pads in a row directly underneath. On the back (solder) side of the board, there is a small trace which connects the "CO" and "21" pad. Cut this trace with a sharp knife or scribe point so there is no longer continuity between these pads. Form the clipping from a resistor lead, or other small bare wire into a loop and insert this jumper between the "-5V" pad and the "21" pad. Solder and trim the leads. Next find the two pads between C2 and R6, with legend "-16" under the right pad of the pair. On the back (solder) side of the board, cut the trace which connects these pads.

( ) Step 8. Stop assembly at this point and proceed with Sol-PC assembly and test up through Step 48. (See Section III.) Then go on to Step 9 of this procedure.

( ) Step 9. Plug personality module into J5 on Sol-PC, apply power to Sol-PC and make the following voltage measurements on the personality module, with respect to chassis ground:

MEASUREMENT POINT	VOLTAGE
Pin 24 of U1, U2	+5 V dc ± 5%
Pin 14 of U3	+5 V dc ± 5%
Pin 21*of U1, U2	-5 V dc ± 5%
Pin 12 of U1, U2	Ground
Pin 7 of U3	Ground

*For 2708-1 version only

( ) Measure between edge connector pin B14 and pin B15. You should measure more than 1M ohms. A reading less than 10K ohms indicates a short.

( ) If any voltages are incorrect, locate and correct the cause before proceeding to Step 10.

( ) If the voltages are correct, turn power off, disconnect power cable, unplug personality module and go on to Step 10.

( ) Step 10. Install IC's in the sockets numbered U1 through U3. Make sure the dot or notch indicating pin 1 on the IC package is in the correct position as indicated on the PC board component legend and the assembly drawing X-6. Socket U2 is left empty on 2708-0 versions (9216 ROM with no window). As shown in the table, the 2708 EPROM's have paper labels with the designation shown, while 9216 ROM's have the designation printed on the IC package itself.

			IC LABEL	
	IC NO.	TYPE	CONSOL	SOLOS
2708-0 version	( ) U1*	2708	C	S4
	( ) U2*	2708	Empty	S5
	( ) U3	74LSØ8	--	--
2708-1 version	( ) U1*	9216	--	SOLOS
	( ) U2	Empty	--	--
	( ) U3	74LSØ8	--	--

*MOS devices. See CAUTION on pages IV-2, 3.

( ) Step 11. Plug personality module into J5 on Sol-PC and connect Sol-PC video output cable to video monitor. (Refer to CAUTION on Page III-22 in Section III.)

( ) Set S1 switches as follows:

No. 1 through 4: OFF

No. 5: ON

No. 6: OFF

( ) Turn monitor on and apply power to Sol-PC

( ) With both the CONSOL and SOLOS modules, you should see the cursor, preceded by a prompt character, like this: >█

( ) If you do not see a cursor, locate and correct the problem before proceeding.

( ) If a blinking cursor is present, the ENter and DUmp commands should operate as described in Section IX of this manual.

( ) If the ENter and DUmp commands do not operate correctly, locate and correct the problem before proceeding.

( ) If the personality module is operating correctly, turn monitor and power off, disconnect power cable and video output cable and go on to Step 50 in Section III. (The personality module can be left plugged in.)

V KEYBOARD ASSEMBLY and TEST

5.1 PARTS AND COMPONENTS

Check all parts and components against the "Parts List", Table 5-1. If you have difficulty in identifying any parts by sight, refer to Figure 3-1 on Page III-5 in Section III of this manual.

5.2 ASSEMBLY TIPS

For the most part the assembly tips given in Paragraph 3.2 of Section III (Page III-1) apply to assembling the Sol keyboard.

In addition, be sure your hands are clean before handling the circuit board, especially the area containing the keyboard switch pads.

5.3 ASSEMBLY PRECAUTIONS

For the most part the assembly precautions given in Paragraph 3.3 in Section III (Page III-6) apply to assembling the Sol keyboard.

5.4 REQUIRED TOOLS, EQUIPMENT AND MATERIALS

The following tools, equipment and materials are recommended for assembling the personality module:

1. Needle nose pliers

2. Diagonal cutters

3. Screwdriver (thin blade)

4. Controlled heat soldering iron, 25 watt

5. 60-40 rosin-core solder (supplied)

5.5 ORIENTATION

Light emitting diode location LED3 will be located in the lower left-hand corner of the board when locations J1 and U4 through U16 are at the top of the board. In this position the component (front) side of the board is facing up and all horizontal reading legends will read from left to right. Subsequent position references related to the keyboard circuit board assume this orientation.

Table 5-1. Sol Keyboard Parts List.

INTEGRATED CIRCUITS

1	555 (U3)
1	2101 or 9101 (U20)
2	4051A (U19 & 22)
4	74LS00 (U4,10,14,16)
1	74LS04 (U23)
1	7406 (U24)
2	74LS10 (U13 & 27)
1	74LS30 (U25)
2	7442 (U17 & 21)
5	74LS74 (U8,9,11,15,26)
2	7493 (U6,U5)
1	74LS132 (U7)
2	74LS175 (U1,U2)
1	8334, 9334 or 83L34 (U12)
1	8574, 74S287, or 82S129 (U18)

TRANSISTORS

6	2N3640
3	2N4274

DIODES (ZENER)

1	1N5221B (D1)

DIODES (LIGHT EMITTING)

3	MV5752 (LED1,2,3)

RESISTORS

1	10	ohm, ¼ watt, 5%
3	150	ohm, ¼ watt, 5%
1	390	ohm, ¼ watt, 5%
1	680	ohm, ¼ watt, 5%
7	1 K	ohm, ¼ watt, 5%
10	1.5K	ohm, ¼ watt, 5%
1	2.2K	ohm, ¼ watt, 5%
5	3 K	ohm, ¼ watt, 5%
2	33 K	ohm, ¼ watt, 5%
2	68 K	ohm, ¼ watt, 5%
2	2.2K	ohm resistor network
2	33 K	ohm resistor network

CAPACITORS

2	220	pfd, disc
1	470	pfd, disc
1	.0022	ufd, disc
2	.01	ufd, disc
5	.047	ufd, disc
1	.1	ufd, Mylar tubular
2	15	ufd, tantalum dipped

Table 5-1. Sol Keyboard Parts List (Continued).

MISCELLANEOUS

1	Sol-KBD Printed Circuit Board
1	8-pin DIP Socket
17	14-pin DIP Socket
8	16-pin DIP Socket
1	22-pin DIP Socket
1	20-pin Header, 3M3492-2002
1	9-3/4" 20-conductor Rainbow Cable Assembly
1	70-key (Sol-10) or 85-key (Sol-20) Keyboard Assembly
1	Plastic Insert (Sol-10) for Key Pad
18	Torx Screw (Similar to #4 by 3/8" sheet metal screws.)
3	Fiber Spacer
1	Length Solder

5.6 ASSEMBLY-TEST

5.6.1 Circuit Board Check

( ) Visually inspect circuit board for obvious flaws. (The design of the board includes numerous unconnected traces and traces that are shorted to each other.)

( ) Check circuit board to insure that the +5-volt bus is not shorted to ground. Using an ohmmeter, measure between the GND and +5V pads located in the upper left corner of the board. There should be no continuity.

If no visual inspection reveals any defect, or you measure continuity between the GND and +5V pads, return the board to Processor Technology for replacement. If the board is not defective, proceed to next paragraph.

5.6.2 Assembly-Test Procedure

Refer to keyboard assembly drawing X-7.

CAUTION

SOME MOS INTEGRATED CIRCUITS ARE USED ON THE Sol KEYBOARD. THEY CAN BE DAMAGED BY STATIC ELECTRICITY DISCHARGE. HANDLE MOS IC's SO THAT NO DISCHARGE FLOWS THROUGH THE IC. AVOID UNNECESSARY HANDLING AND WEAR COTTON, RATHER THAN SYNTHETIC, CLOTHING WHEN YOU DO HANDLE MOS IC's. (STATIC CHARGE PROBLEMS ARE MUCH WORSE IN LOW HUMIDITY CONDITIONS.)

( ) Step 1. Install DIP sockets. Install each socket in the indicated location with its end notch oriented as shown on the circuit board and assembly drawing. Take care not to create solder bridges between the pins and/or traces. (Refer to "Installation Tip" on Page III-9 in Section III.)

LOCATION	TYPE SOCKET
( ) U1 and 2	16 pin
( ) U3	8 pin
( ) U4 through U11	14 pin
( ) U12	16 pin
( ) U13 through U16	14 pin
( ) U17 through U19	16 pin
( ) U20	22 pin
( ) U21 and 22	16 pin
( ) U23 through U27	14 pin

( ) Step 2. Install the following capacitors in the indicated locations. Take care to observe the proper value, type and orientation (if applicable) for each installation. Insert leads, pull down snug to board, bend leads outward on solder (back) side of board, solder and trim.

NOTE

Disc capacitor leads are usually coated with wax during the manufacturing process. After inserting leads through mounting holes, remove capacitor and clear the holes of any wax. Reinsert and install.

LOCATION	VALUE	TYPE	ORIENTATION
( ) C1	15 ufd	Tantalum	"+" lead top
( ) C2	.047 ufd	Disc	None
( ) C3	.1 ufd	Mylar	"

(Continued on Page V-5.)

LOCATION	VALUE	TYPE	ORIENTATION
( ) C4	.01 ufd	Disc	None
( ) C5	.047 ufd	"	"
( ) C6	.047 ufd	"	"
( ) C7	.0022 ufd	"	"
( ) C8	470 pfd	"	"
( ) C9	220 pfd	"	"
( ) C10	220 pfd	"	"
( ) C11	.01 ufd	"	"
( ) C12	.047 ufd	"	"
( ) C13	.047 ufd	"	"
( ) C14	15 ufd	Tantalum	"+" lead top

( ) Step 3. Install the following resistors in the indicated locations. Bend leads to fit distance between mounting holes, insert leads, pull down snug to board, solder and trim.

LOCATION	VALUE (ohms)	COLOR CODE
( ) R1	150	brown-green-brown
( ) R2	150	" " "
( ) R3	150	" " "
( ) R4	68 K	blue-gray-orange
( ) R5	560 K	green-blue-yellow
( ) R6	33 K	orange-orange-orange
( ) R7	1 K	brown-black-red
( ) R8	1.5K	brown-green-red
( ) R9	3 K	orange-black-red
( ) R10	3 K	" " "
( ) R11	3 K	" " "
( ) R12	3 K	" " "
( ) R13	1.5K	brown-green-red
( ) R14	1.5K	" " "
( ) R15	1.5K	" " "
( ) R16	1 K	brown-black-red
( ) R17	390	orange-white-brown
( ) R18	1 K	brown-black-red
( ) R19	10	brown-black-black
( ) R20	1 K	brown-black-red
( ) R21	1 K	" " "
( ) R22	3 K	orange-black-red
( ) R23	1 K	brown-black-red
( ) R24	1 K	" " "
( ) R25	1.5K	brown-green-red
( ) R26	680	blue-gray-brown
( ) R27	33 K	orange-orange-orange
( ) R28	1.5K	brown-green-red
( ) R29	1.5K	" " "

(Continued on Page V-6.)

LOCATION	VALUE (ohms)	COLOR CODE
( ) R30	1.5K	brown-green-red
( ) R31	1.5K	" " "
( ) R32	68 K	blue-gray-orange
( ) R33	1.5K	brown-green-red
( ) R34	2.2K	red-red-red

( ) Step 4. Install Zener diode D1 (1N5221B) in its location to the left of R17. Position D1 with its dark band (cathode) at the bottom.

( ) Step 5. Install Q1, Q2 and Q9 (2N4274) and Q3 through Q8 (2N3640) in their respective locations at the top center of the board. The emitter lead (closest to flat side of case) is oriented toward the right of the board and the base is oriented toward the top. Insert leads until transistor is approximately 3/16" above surface of circuit board, solder and trim.

( ) Step 6. Install resistor networks RX1 and RX3 (2.2K ohms) and RX2 and RX4 (33K ohms) in their respective locations just above the keyboard pads. Install each network so that the dot on its package is positioned next to the foil square on the circuit board. Recheck values before soldering.

CAUTION

THESE RESISTOR NETWORKS ARE DELICATE.
HANDLE WITH CARE.

( ) Step 7. Install light emitting diodes LED1, 2 and 3 (MV5752) in their respective locations in the lower left corner of the circuit board. Insert leads through fiber spacer, position each diode with its cathode lead (longer lead and/or the lead next to flat edge of LED package) at the bottom, insert leads into mounting holes in circuit board, pull down so that spacer and LED are snug to board, solder and trim. (If fiber spacers are not supplied with your kit, install LED's so they are approximately 3/16" above surface of circuit board.)

( ) Step 8. Install 20-pin header in location J1 (upper left corner of board). Position header so pin 1 is in the lower left corner. (An arrow on the header points to pin 1.) Solder.

( ) Step 9. Using an ohmmeter, measure between GND and +5V pads in upper left corner of the board. You should measure some resistance. Zero resistance indicates a short. If required, find and correct the problem before proceeding to Step 10.

( ) Step 10. Install the following IC's in the indicated locations. Pay careful attention to the proper orientation.

NOTE

Dots on the assembly drawing and PC board indicate the location of pin 1 of each IC.

IC NO.	TYPE
( ) U1	74LS175
( ) U2	74LS175
( ) U3	555
( ) U4	74LS00
( ) U5	7493
( ) U6	7493
( ) U7	74LS132
( ) U8	74LS74
( ) U9	74LS74
( ) U10	74LS00
( ) U11	74LS74
( ) U12	8334,9334 or 83L34
( ) U13	74LS10
( ) U14	74LS00
( ) U15	74LS74
( ) U16	74LS00
( ) U17	7442
( ) U18	8574, 74S287, or 82S129
( ) U19*	4051A*
( ) U20*	2101 or 9101*
( ) U21	7442
( ) U22*	4051A*
( ) U23	74LS04
( ) U24	7406
( ) U25	74LS30
( ) U26	74LS74
( ) U27	74LS10

*MOS device. Refer to CAUTION on Page V-4.

( ) Step 11. Connect 20-conductor ribbon cable between J1 on keyboard to J3 on Sol-PC so that cable goes left from J3.

( ) Step 12. Check keyboard operation.

( ) Set S1 switches on Sol-PC as follows:

No. 1 through 4: OFF

No. 5: ON

No. 6: OFF

( ) Connect TV monitor to Sol-PC.

( ) With personality module installed, apply power to Sol-PC.

( ) Using a CLEAN finger, touch key pad #62 (MODE SELECT).

( ) You should get a carriage return and line feed and see a "greater than" sign (>) on the screen above the cursor.

NOTE

You may have to touch pad #62 several times to obtain the specified display.

( ) If you are unable to obtain the specified display, locate and correct the problem before proceeding.

( ) If the keyboard is operating correctly, turn monitor and Sol-PC power off, disconnect 20-conductor ribbon cable at J1 on the keyboard and go on to Step 13.

( ) Step 13. Place keyboard assembly carefully over key pads on PC board. Be sure the three LED's fit in the holes in the sheet metal. Carefully align holes in PC board, 18 in all, with threaded mounting holes on bottom of keyboard assembly. Insert Torx screws from solder (back) side of board and, using a thin-blade screwdriver, drive into keyboard assembly mounting holes. Drive screws evenly and tighten just enough to hold keyboard assembly in place.

CAUTION: DO NOT OVERTIGHTEN THESE SCREWS.

( ) Step 14. Reconnect 20-conductor ribbon cable to J1 on keyboard.

( ) Step 15. Test keyboard for proper operation.

( ) Apply power to monitor and Sol-PC.

( ) Strike MODE SELECT key.

( ) Strike UPPER CASE key. Indicator light should come on.

( ) Strike UPPER CASE key again. Indicator light should go off.

( ) Strike LOCAL key. Indicator light should come on.

( ) Strike LOCAL key again. Indicator light should go off.

(Step 15 continued.)

( ) Strike SHIFT LOCK key. Indicator light should come on.

( ) Strike either SHIFT key. Indicator light should go off.

( ) Verify operation of all alphanumeric keys. (As you strike each key you should observe the corresponding character on the monitor.)

( ) Should the keyboard fail any of the preceding checks, locate and correct the problem before proceeding.

( ) If the keyboard passes all of the preceding tests, congratulations on a job well done.

At this point you have successfully assembled the Sol keyboard and tested it for proper operation. It is now ready for use with the Sol-PC Single Board Terminal Computer^{TM}.

Having completed the Sol keyboard, power supply, Sol-PC and personality module, you are now ready to assemble the Sol cabinet-chassis. Cabinet-chassis assembly instructions are provided in Section VI.

VI Sol CABINET-CHASSIS ASSEMBLY

6.1 INTRODUCTION

This section covers assembly of the Sol-10 and Sol-20 chassis and cabinet. The instructions contained herein assume that you have already assembled the power supply and Sol-PC Single Board terminal Computer^{TM}...including the personality module and the Sol keyboard (Sol-KBD).

6.2 PARTS AND COMPONENTS

Check all parts and components against the appropriate "Parts List(s)", Table 6-1 and 6-2. If you have any difficulty in identifying any parts by sight, refer to Figures 6-1 and 6-2 on Pages VI-4 and VI-4.

6.3 ASSEMBLY TIPS

6.3.1 General

1. Scan Section VI in its entirety before you start to assemble your Sol cabinet-chassis.

2. IT IS IMPORTANT that you follow the step-by-step instructions <u>in the order given</u> when assembling the Sol cabinet-chassis if your assembly is to be done correctly and with minimum effort.

3. Assembly steps and component installations are preceded by a set of parentheses. Check off each installation and step as you complete them. This will minimize the chances of omitting a step or component.

4. Should you encounter any problem during assembly, call on us for help if needed.

6.3.2 Electrical

1. Use a low-wattage soldering iron, 25 watts maximum, for all soldering.

2. Solder neatly and as quickly as possible.

3. Use only 60-40 rosin-core solder. NEVER use acid-core solder or externally applied fluxes.

4. DO NOT press the tip of the soldering iron on pads or traces when installing components and/or attaching leads to a PC board. To do so can cause the pad or trace to "lift" off the board and permanently damage it.

Table 6-1. Sol-10 Cabinet-Chassis Parts List.

CHASSIS and SUBCHASSIS

1	Main Chassis
1	Expansion Chassis
1	Power Supply Subchassis
1	Fan Closure Plate

BRACKETS

1	Power Supply Subchassis Bracket
2	Keyboard Bracket

CABINET

1	Left Side Piece, Walnut
1	Left Side Piece, Masonite
1	Right Side Piece, Walnut
1	Right Side Piece, Masonite
1	Keyboard Cover
1	Top Cover

MISCELLANEOUS

2	Finger Well Label, Black *
1	Printed Trim Plate, Paper
1	Plexiglass Strip
1	Serial Number Label
1	Connector Identification Label

HARDWARE

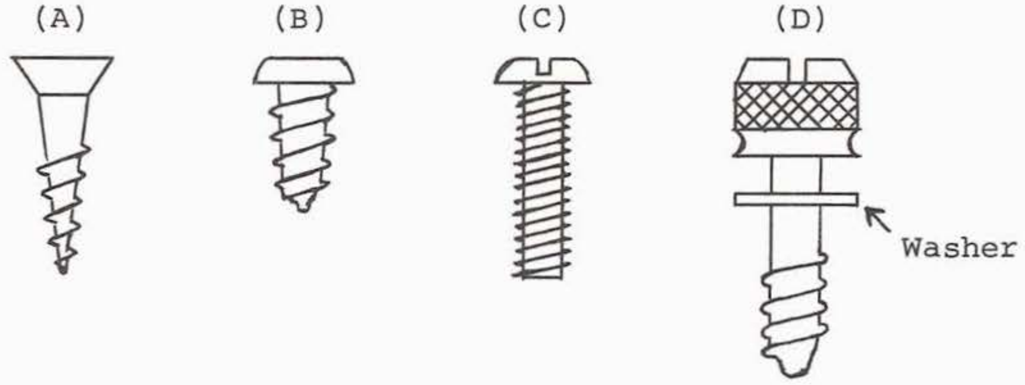
4	4-40 x 3/16 Screw, Machine
16	4-40 x 5/16 Screw, Machine
8	4-40 Hex Nut
22	#4 Lockwasher, Internal Tooth
16	6-32 x ½ Screw, Machine
8	6-32 Hex Nut
16	#6 Lockwasher, Internal Tooth
11	8-32 x ½ Screw, Machine
2	8-32 x 1 Screw, Machine
3	8-32 Hex Nut
3	#8 Lockwasher, Internal Tooth
2	10-24 Thumb Screw
21	#6 x ¼ Screw, Sheet Metal
4	#6 x 5/16 Screw, Sheet Metal
12	5/8 Screw, Wood
2	#4 Solder Lug
10	Tinnerman Plastic Inserts, Tapped
2	¼" Spacer, 4-40 Tapped
4	Self-stick Protective Pads

* May be packaged under the plexiglass strip.

Table 6-2. Sol-20 Cabinet-Chassis Parts List.

The Sol-20 Cabinet-Chassis Kit includes all Sol-10 parts listed in Table 6-1 plus the following items:

1	Sol-BPB Circuit Board (Backplane Board)
1	3" 5-wire Cable with Molex Connector
1	100-pin Edge Connector, VIKING 3KH50/90V5
5	100-pin Edge Connector, other brand
2	Right Angle Backplane Bracket
1	Gusset Bracket, Left
1	Gusset Bracket, Right
10	Plastic Card Guide
6	4-40 x 5/16 Screw, Machine
6	4-40 x 5/8 Screw, Machine
12	4-40 Hex Nut
12	#4 Lockwasher, Internal Tooth
8	6-32 x ½ Screw, Machine
8	6-32 Hex Nut
9	#6 Lockwasher, Internal Tooth
12	#6 x ¼ Screw, Sheet Metal



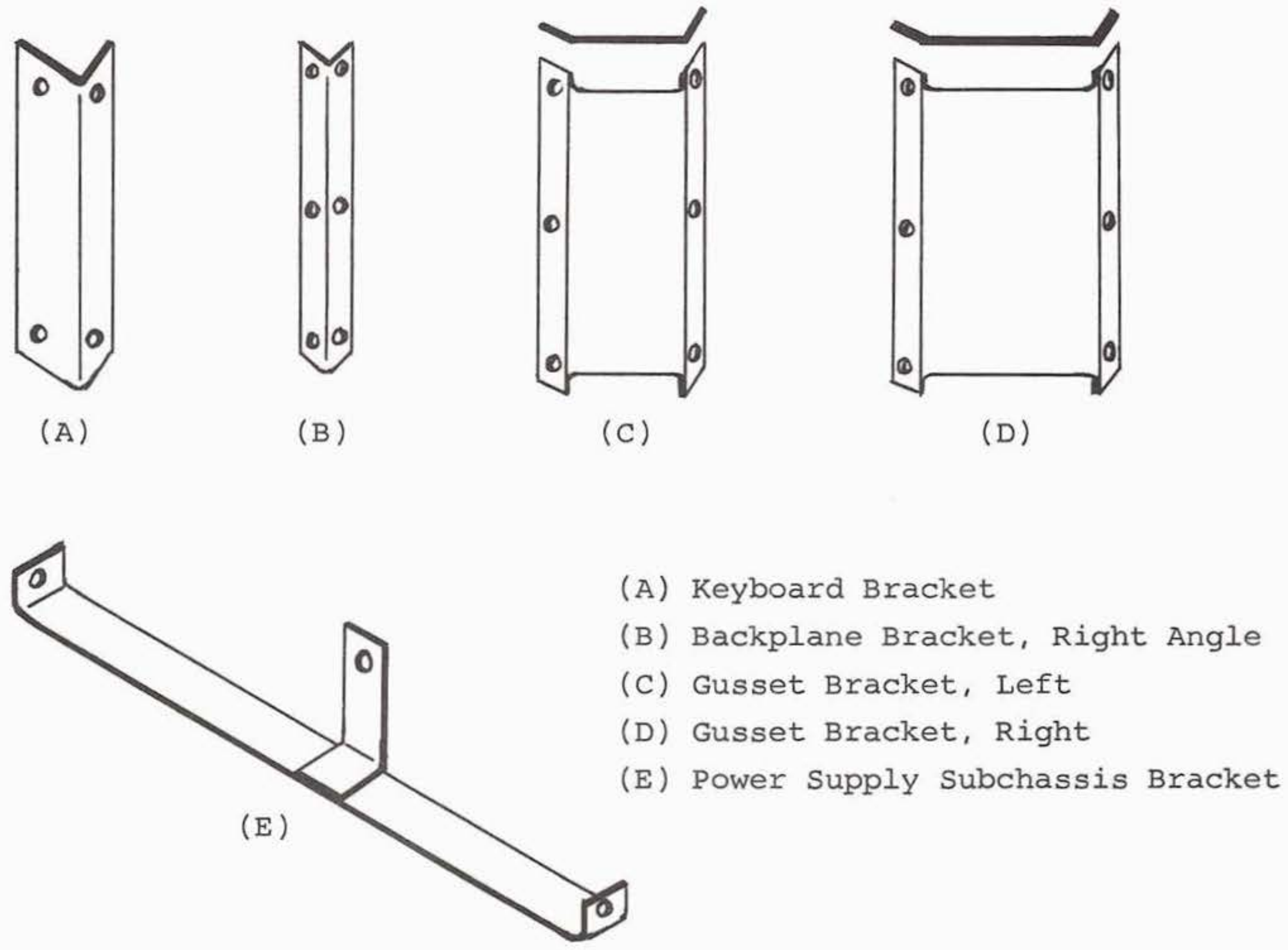
(A) Flat Head Wood Screw

(B) Sheet Metal Screw

(C) Binder or Pan Head Screw

(D) Thumb Screw

Figure 6-1. Types of screws used in Sol cabinet-chassis assembly.



(A) Keyboard Bracket

(B) Backplane Bracket, Right Angle

(C) Gusset Bracket, Left

(D) Gusset Bracket, Right

(E) Power Supply Subchassis Bracket

Figure 6-2. Brackets used in Sol cabinet-chassis assembly.

6.3.2 Electrical (continued)

5. (Sol-20 only.) The Backplane PC board (Sol-BPB) has plated-through holes. Solder flow through to the component side of the board can produce solder bridges (shorts). Check for such bridges after you install each component or wire.

6. (Sol-20 only.) The Backplane PC board (Sol-BPB) has an integral solder mask (a lacquer coating) that shields selected areas on the board. This mask minimizes the chances of creating solder bridges during assembly.

6.3.3 Mechanical

1. If you do not have the proper screwdrivers (see Paragraph 6.5), we recommend that you buy them rather than using a knife point, a blade screwdriver on a Phillips screw, and other makeshift means. Proper screwdrivers minimize the chances of stripping threads, disfiguring screw heads and marring decorative surfaces.

2. To assure a correct fit and tight assembly, be sure you use the screws specified in the instructions.

3. Lockwashers are widely used in the Sol cabinet-chassis assembly so that screws will not loosen when subjected to stress or vibration. When a lockwasher is specified, do not omit it and make sure you install it correctly.

4. Some instructions call for prethreading holes. This is done to make assembly easier by giving you maximum working space for installing relatively hard-to-drive sheet metal screws. If you bypass prethreading instructions you will only make your cabinet-chassis assembly more difficult.

To prethread a hole, insert specified screw in the hole and position it as straight as possible. While holding the screw in this position, drive it into the metal with the proper screwdriver. If started straight the screw will continue to go straight into the metal so that the head and sheet metal surfaces are in full contact.

5. The diameter of the shank (threaded portion) of a screw increases in relation to its number. For example, a 6-32 screw is larger in diameter than a 4-40 screw. Also, a #8 lockwasher is larger than a #4 lockwasher.

6.4 REQUIRED TOOLS, EQUIPMENT AND MATERIALS

The following tools, equipment and materials are recommended for assembling the Sol cabinet-chassis. (Unless indicated otherwise, none of the following items are supplied with your kit.)

1. Needle nose pliers
2. Diagonal cutters
3. Screwdriver, thin ¼" blade
4. Screwdriver, #2 Phillips
5. Controlled heat soldering iron, 25 watt
6. 60-40 rosin-core solder (supplied)
7. Silicon grit abrasive paper, 220 & 400 grit
8. Boiled linseed oil
9. Turpentine or mineral spirits
10. Masking tape
11. Transparent tape
12. Rubber mallet or small hammer

6.5 ORIENTATION

6.5.1 Sol Backplane Board, Sol-BPB (Sol-20 Only)

The PC board identification (Sol-BPB) and revision level will be located in the upper left-hand corner of the board when the edge connector (gold contacts) is positioned at the bottom of the board. In this position, the <u>component</u> (<u>front</u>) side of the board is facing up. Subsequent position references related to the Sol-BPB assume this orientation.

6.5.2 Sol Cabinet-chassis

Unless specified otherwise, all position references (e.g., left, right, front, back, bottom and top) in the cabinet-chassis assembly instructions assume the Sol cabinet is viewed from the front (keyboard) when it is sitting in its normal position (keyboard up).

6.6 ASSEMBLY-TEST

NOTE

> Instructions that apply only to the Sol-20 are preceded by an asterisk. Skip these instructions if you are assembling a Sol-10.

*6.6.1 Backplane Board (Sol-BPB) Assembly

Refer to assembly drawing, page X-11.

*( ) Step 1. Visually inspect Sol-BPB PC board for obvious flaws such as solder bridges (shorts) between traces, broken traces and similar defects.

If visual inspection reveals any defects, return the board to Processor Technology for replacement. If the board passes inspection, go on to Step 2.

*( ) Step 2. Install VIKING 3KH50/9VC5 100-pin edge connector on top edge of PC board. (This edge has silver (not gold) contacts.)

NOTE

> This connector is supplied as a trouble-shooting aid. It is not critical to normal operation of the Sol-20.

Position connector on PC board so that its #1 trace is aligned with the #1 trace on the board, and push connector fully onto board. Bend the connector pins slightly so that both rows of pins are in light contact with the traces on the board. DO NO' CLOSE CONNECTOR PINS SO MUCH THAT YOU WILL DAMAGE THE TRACES WHEN PLACING THE CONNECTOR OVER THE EDGE OF THE BOARD. While holding the connector and board together, place board solder side down on a book, or other flat surface that is higher than your work surface, so the connector extends fully over the edge. That is, the connector should not rest on the book. Reposition connector if needed to align the pins and traces. On the component (front) side of board, solder a pair of traces. On the component (front) side of board, solder a pair of pins at each end of the connector to their respective traces on the

(Step 2 continued on Page VI-8.)

board. Then solder the remaining 46 pins on the component side to traces.

The connector must be perpendicular to the edge of the board. If it is not, bend the pins you just soldered to obtain the required alignment. Then solder the other 50 pins to the traces.

*( ) Step 3. Install the other five 100-pin edge connectors. Position connector on front side of board and insert pins. On solder (back) side of board, solder pins at opposite corners of the connector to hold it in place. Then solder remaining 98 pins. (Refer to Paragraph 6.6.1 on Page VI-6 for definition of front side of board.)

*( ) Step 4. Connect 3" 5-wire cable to circuit board to uppermost pads in top right corner: Insert wires from solder (back) side of board and solder on component (front) side side of board. If a wire is too large for the mounting hole, snip off as many individual strands as needed to obtain a fit. Connect cable leads as follows:

CABLE LEAD	PAD
White	Ground (fifth hole from right)
White	Ground (fourth hole from right)
Blue	+8 V dc (third hole from right)
Red-White	+16 V dc (second hole from right)
Yellow-White	-16 V dc (first hole from right)

NOTE

Pad orientations given above are as viewed from component (front) side of circuit board.

*( ) Step 5. Fill all exposed (not covered with lacquer) feethrough holes on right-hand side of board with solder.

The backplane board is now assembled. Set it to one side for later installation in the cabinet-chassis.

6.6.2 Wooden-masonite Parts

Refer to assembly drawings, pages X8 and X9.

( ) Step 6. Finish walnut side panels.

The sides of the Sol cabinet are solid black walnut which have been sanded to a smooth surface. If there should be any blemishes, remove them with 220 grit abrasive paper. SAND WITH THE GRAIN...NEVER ACROSS THE GRAIN.

We recommend an oil finish be applied to the walnut since such a finish lies "in" the wood, not on "top" of it. Also, no wax is necessary with an oil finish.

You may obtain a good finish by using a half-and-half mixture of boiled linseed oil and turpentine. Apply mixture with rag, soaking all surfaces. (End grain may require more oil than face grain.) Let stand for one-half hour, recoating any dry spots, and wipe dry with a clean cloth. Repeat as often as needed to obtain a lustrous finish. (It may take several days.)

You may also use a commercially available penetrating oil such as Watco Danish Oil or Tung Oil. Follow directions on the container if you use such an oil. For a more durable finish when using a penetrating oil:

1. Sand between applications with 220 grit silicon carbide abrasive paper. (Wipe clean after 15 minutes to avoid build-up.)

2. Repeat the following day using 400 grit paper between applications.

3. Repeat as often as desired, using a still finer grit paper between applications. DO NOT sand after final application, but wipe the surfaces clean and let dry for one day. Then coat with previously mentioned linseed oil-turpentine mixture and wipe dry.

( ) Step 7. Using a black broad tipped felt pen, darken all edges of the two masonite parts.

( ) Step 8. Mate the left walnut and masonite side pieces. (Refer to assembly drawing on page X-8.)

NOTE

> When the walnut and masonite side pieces are correctly mated, the countersink side of the six countersunk (funnel-shaped enlargement) holes in the masonite will be next to the main chassis.

Insert five Tinnerman plastic inserts in the holes indicated on Drawing X-8. Insert these from the side that mates with the walnut. These inserts may be seated by gently tapping them with a hammer until fully seated.

( ) Step 9. Insert remaining five Tinnerman inserts in right masonite side piece as you did the left side piece. (Refer to Drawing on page X-9.

( ) Step 10. Attach left masonite side piece to left walnut side piece with six 5/8" flat head wood screws. Drive these screws through the countersunk holes in the masonite into the walnut. (Refer to Drawing No. X-8.)

NOTE

Lead holes have been predrilled in the walnut to make driving these screws easier.

( ) Step 11. Attach right masonite side piece to right walnut side piece as you did the left side pieces. (Refer to Drawing No X-9.)

( ) Step 12. Set both side piece assemblies to one side.

6.6.3 Sol Assembly

Refer to Drawing No. X-10 in Section X. Figure 6-3 and 6-4 show complete Sol assemblies without covers.

( ) Step 13. Mount keyboard support bracket (heavy gauge right angle brackets) to each side of the main chassis as shown in Drawing No. X-10 . These are mounted with the narrower side of the bracket at the top.

Attach each bracket to main chassis with two 6-32 x ½ binder or pan head screws and #6 lockwashers. Place lockwasher on screw, insert screw from outer surface of main chassis side wall and drive into the threaded bracket mounting holes.

( ) Step 14. Attach power supply subchassis bracket (short leg "T" shaped bracket) to top front of power supply subchassis as shown in Drawing No. X-10. (Note that leg of "T" is closer to side wall of subchassis. This leg is for mounting a "power on" indicator light--not supplied.) Insert #6 x ¼ sheet metal screw from right side of side wall and drive into bracket.

( ) Step 15. To gain access to the rear area of the power supply subchassis side wall, remove the #6 x ¼ sheet metal screw that attaches the fan closure plate to the subchassis. You should not have to disconnect the transformer (black wires) or AC receptacle ground (green wire) leads since they have sufficient slack to permit moving the closure plate out of the way. (Set screw to one side for use in re-installing the fan closure plate.)

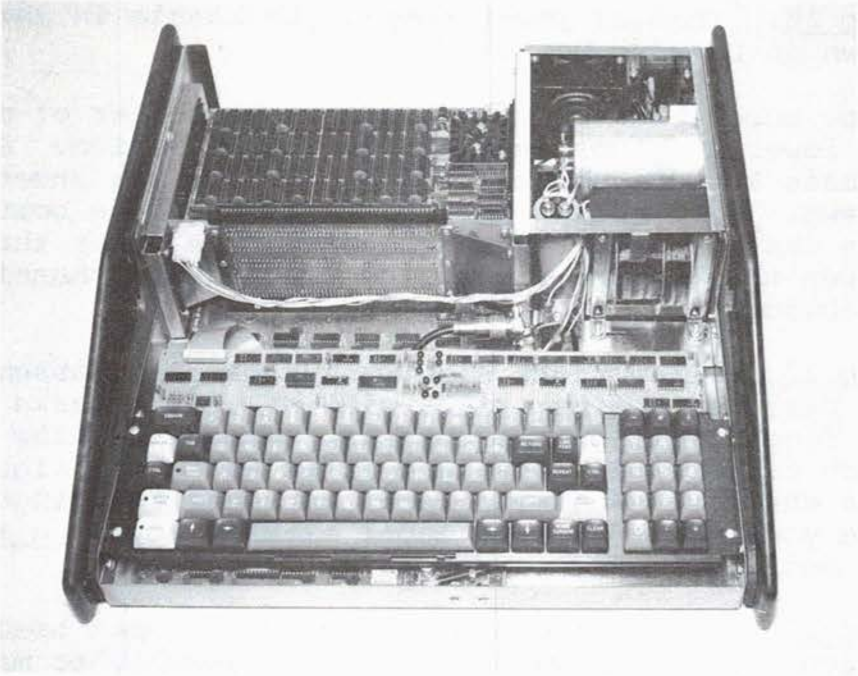


Figure 6-3. Sol-20 with covers removed. Front (or keyboard) is in foreground, power supply is in right rear corner, expansion chassis (with 8KRA Memory installed) is to left of power supply. The vertical board just behind white connector on left is the backplane board.

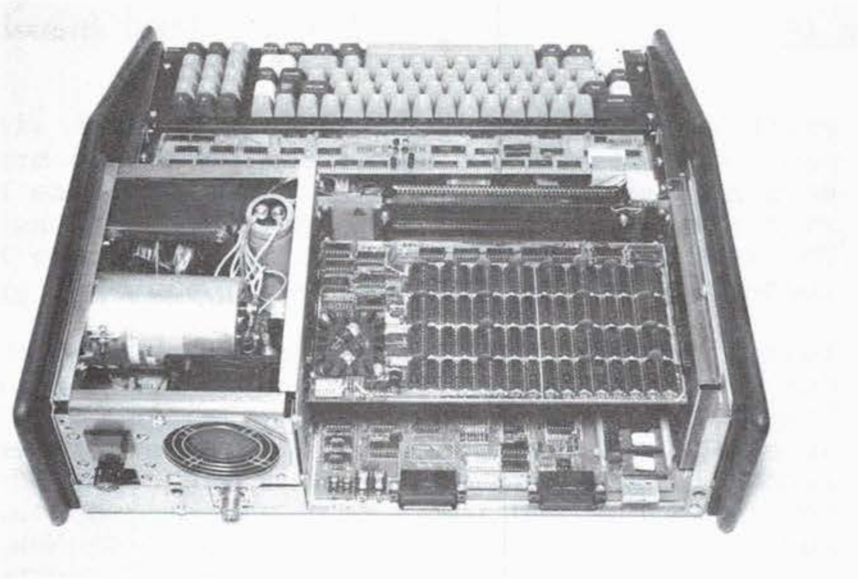


Figure 6-4. Sol-20 with covers removed. Rear side of assembly is in foreground and Sol-PC is just visible at lower right rear of assembly. 8KRA Memory is installed in expansion chassis above Sol-PC.

( ) Step 16. Install power supply subchassis in main chassis as shown in Drawing No. X-10.

Place subchassis over the rear right corner of main chassis and lower it almost vertically into position. Attach subchassis to main chassis using eight #6 x ¼ sheet metal screws. Five screws are driven through the bottom of the main chassis into the subchassis. The other three are driven through the right side of the main chassis into the subchassis.

( ) Step 17. Place right side walnut-masonite assembly in proper position against right side of main chassis and outline the finger well on the chassis. Remove backing from one black finger well label and affix it to the right side of main chassis. Position label to cover the finger well outline you made. Be sure label extends beyond all edges of the outline.

( ) Step 18. Using five 8-32 x ½ binder or pan head screws, attach right side walnut-masonite assembly to main chassis and power supply subchassis as shown in Drawing No. X-10. Insert screws from inside surface of chassis and drive into the plastic inserts you installed in Step 9. Note that the two front screws are driven through the main chassis, the two lower rear screws are driven through both the power supply subchassis, and the upper rear screw is driven through the power supply subchassis.

( ) Step 19. Assemble expansion chassis ("U" shaped chassis).

*( ) Prethread 12 mounting holes (six on each side) on expansion chassis side walls for backplane brackets with #6 x ¼ sheet metal screws. Three of these holes on each side are located near the front edge of the main chassis. The remaining three holes on each side are about 1½ to 2 inches behind the front three. Leave screws installed.

( ) Install female coaxial connector on the tab that extends out from the lower right front of the expansion chassis. Insert connector through tab so threaded end faces left as shown in Drawing No. X-10. Insert three 4-40 x 5/16 binder or pan head screws from left side of tab through the two front and lower rear mounting holes. Place #4 lockwahser on each and secure with 4-40 hex nuts. Insert another 4-40 x 5/16 binder or pan head screw through upper rear mounting hole and install 4-40 hex nut. (Leave this nut loose.)

*( ) Install 10 plastic card guides (five on each side) on inside surface of both side walls of the expansion chassis.

These are installed over the ventillation grilles with the gripper fingers pointing towards the backplane board. To install, simply insert posts on guide into appropriate mounting holes and push in until they snap into place.

( ) Step 20. Install expansion chassis on main chassis as shown in Drawing No. X-10.

Position expansion chassis with coaxial connector at the front (near FWB3 on power supply subchassis) over left rear area of main chassis and lower into place. Attach expansion chassis to main chassis using eight #6 x ¼ sheet metal screws. Four screws are driven through the bottom of the main chassis into the expansion chassis, three are driven through the left side of the main chassis into the expansion chassis, and one is driven through the lower left corner of the back side of the main chassis into the expansion chassis.

( ) Step 21. Attach left end of power supply subchassis bracket to expansion chassis as shown in Drawing No. X-10. Drive one 6 x ¼ sheet metal screw through expansion chassis into bracket.

( ) Step 22. Route coaxial cable from connector on fan closure plate along left side of power supply subchassis to connector on expansion chassis.

( ) Step 23. Using the #6 x ¼ sheet metal screw you removed in Step 15, re-attach fan closure plate to power supply subchassis. (Make sure side lip on plate is on right side of expansion chassis side wall.

( ) Step 24. Attach fan closure plate to expansion chassis with two #6 x ¼ sheet metal screws. Drive screws through expansion chassis into fan closure plate.

NOTE

If lip on fan closure plate and expansion chassis are not in contact, insert one or two ½" flat washers as needed between the two surfaces. Place washers so screws pass through them.

( ) Step 25. Connect free end of coaxial cable from connector on fan closure plate to connector on expansion chassis. Solder inner conductor to pin of connector. Remove hex nut on upper rear connector mounting screw, place lockwasher lug (coaxial shield) on screw and secure with nut.

( ) Step 26. Install male coaxial connector on free end of coaxial cable that is connected to Sol-PC (the composite video output cable). Install connector as follows (refer to Figure 6-5):

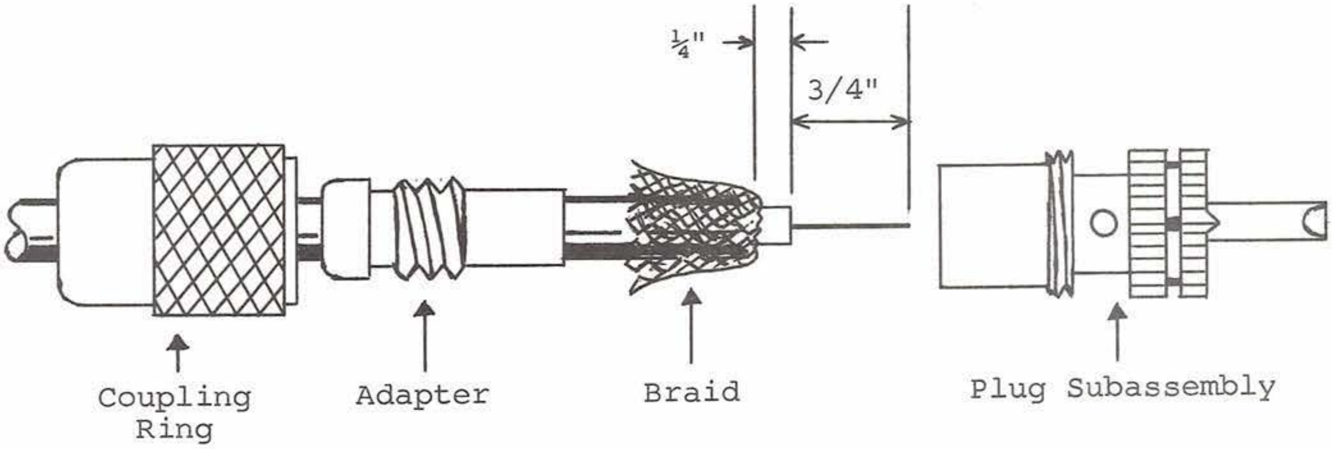


Figure 6-5. Sol-PC coaxial cable connector assembly.

( ) Slide coupling ring and adapter on cable in that order and cut end of cable even.

( ) Remove one inch of outer insulation.

( ) Fan braid slightly and fold back over outer insulation as shown.

( ) Slide adapter fully up under braid and press braid down over adapter body.

( ) Trim braid so that it does not interfer with adapter threads.

( ) Remove 3/4" of inner conductor insulation and tin exposed conductor.

( ) Slide cable fully into plug subassembly and screw subassembly on adapter.

( ) Solder braid to plug subassembly shell through solder holes. (Use enough heat to create a good bond between braid and shell.)

( ) Solder center conductor to plug contact by filling contact with solder. Cut off excess conductor.

( ) Slide coupling ring over plug subassembly and screw it onto plug.

( ) Step 27. Install Sol-PC in expansion chassis.

Position Sol-PC on bottom of expansion chassis with J1, J2 and J6 through J9 at the rear. Align threaded standoffs on bottom of Sol-PC with the oblong holes in the bottom of the main chassis.

Attach Sol-PC to chassis with eight 4-40 x 5/16 binder or pan head screws and #4 lockwashers. Place washer on screw and drive screw loosely into standoff from bottom of main chassis. Leave all eight screws loose.

( ) Step 28. Connect Sol-PC composite video output cable to expansion chassis coaxial connector.

( ) Step 29. Affix black finger well label to left side of main chassis in same manner as you did the right side. (See Step 17.) MAKE SURE LABEL DOES NOT OBSTRUCT ANY OF THE COOLING VENTS.

( ) Step 30. Using three 8-32 x ½ and two 8-32 x 1 binder or pan head screws, attach left side walnut-masonite assembly to main chassis as shown in Drawing No. 101000. Insert screws from inside surface of chassis and drive into the plastic inserts you installed in Step 8. Note that the two front screws (8-32 x ½) are driven through the main chassis, the uppermost screw (8-32 x ½) is driven through the expansion chassis, and the two lower rear screws (8-32 x 1) are driven through both the expansion chassis and main chassis.

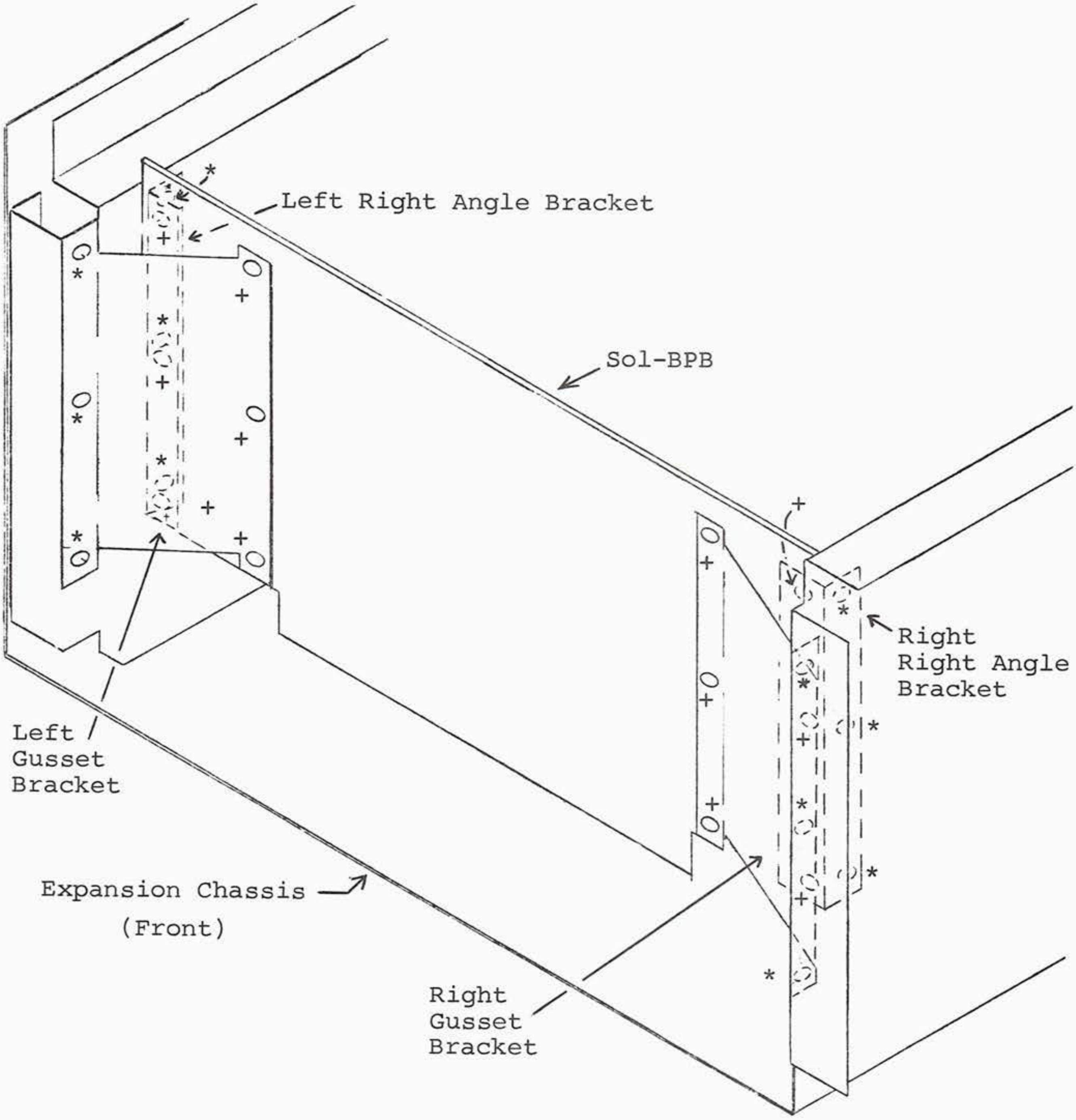
*( ) Step 31. Install left and right backplane right angle brackets (light gauge brackets) on expansion chassis side walls. Refer to Figure 6-6 on Page VI-16.) These two brackets are installed just to the front of the card guides and should be positioned as shown in Figure 6-6. Attach each bracket to the chassis with three #6 x ¼ sheet metal screws. USE THE SCREWS YOU USED IN STEP 19 TO PRETHREAD THE HOLES.

*( ) Step 32. Install backplane circuit board (Sol-BPB). The photograph in Figure 6-7 on Page VI-17 shows the backplane board installed.

*( ) Position Sol-BPB with 100-pin male edge connector down and the five female edge connectors facing the card guides. The board should rest against the front face of the right angle brackets as shown in Figure 6-6. Adjust position of Sol-PC as needed so that you can plug the Sol-BPB edge connector into J11 on the Sol-PC.

*( ) Align holes on left and right ends of Sol-BPB with those in right angle brackets.

(Step 32 continued on Page VI-17)



*#6 x ¼ sheet metal screw

+4-40 x 5/8 binder or pan head screw,
#4 lockwasher and 4-40 hex nut

Figure 6-6. Backplane board (Sol-BPB) installation.

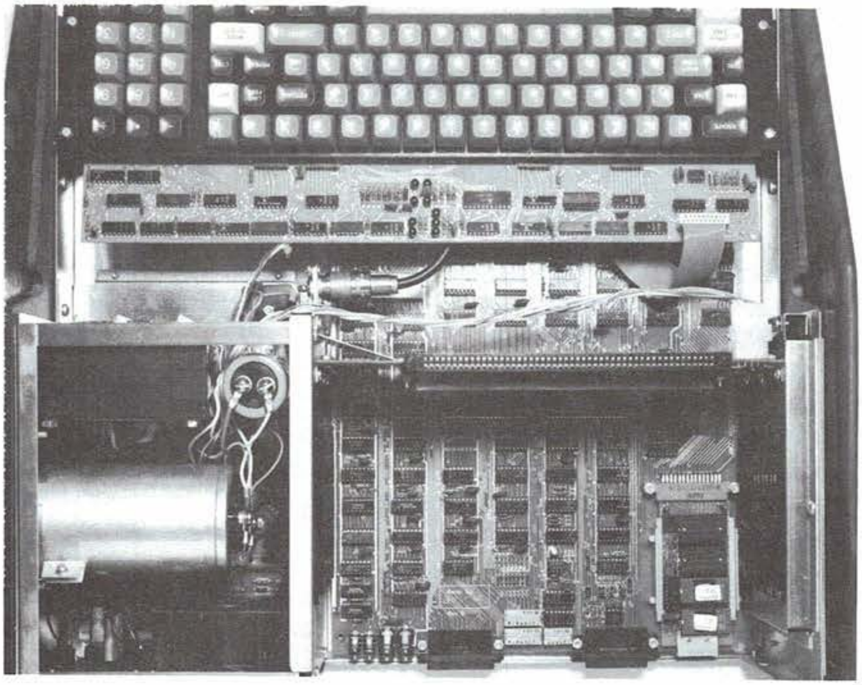


Figure 6-7. Backplane board (Sol-BPB) installation. Rear of Sol is at bottom and Sol-BPB is to right of power supply subchassis in line with C8 and transformer.

(Step 32 continued)

*( ) Attach Sol-BPB to brackets with three 4-40 x 5/16 binder or pan head screws, #4 lockwashers and 4-40 hex nuts on each side. Insert screws from the <u>back side</u> of bracket through Sol-BPB, place lockwasher on each screw and secure each with nut.

*( ) <u>Step 33</u>. Install left and right gusset brackets as shown in Figure 6-6 on Page VI-16.

*( ) Fit narrower gusset bracket on left side so that its flanges are flat against the expansion chassis side wall and the backplane board. (You may have to bend the flange slightly to obtain a proper fit.)

*( ) Attach bracket to expansion chassis side wall with the three #6 x ¼ sheet metal screws you used in Step 19 to prethread the holes.

See WARNING on Page VI-18.

(Step 33 continued on Page VI-18.)

(Step 33 continued.)

WARNING

IT IS QUITE EASY TO SCRATCH OR CUT YOUR HAND ON THE SOLDER SIDE OF THE BACKPLANE BOARD WHEN DRIVING THESE SCREWS. PLACE A SUITABLE PROTECTIVE BARRIER, SUCH AS CARDBOARD, AGAINST SOLDER SIDE OF BACKPLANE BOARD DURING INSTALLATION TO PREVENT SUCH INJURY.

*( ) Attach bracket to backplane board with three 4-40 x 5/8 binder or pan head screws, #4 lockwashers and 4-40 hex nuts. Insert screws from front side of bracket through Sol-BPB, place lockwasher on each screw and secure each with nut.

*( ) Install wider gusset bracket on right side in the same manner as you did the left bracket. THE PRECEDING WARNING ALSO APPLIES TO INSTALLING THIS BRACKET.

*( ) Step 34. Connect Sol-20 DC power cable from power supply subchassis to the Sol-BPB power cable you installed in Step 4.

( ) Step 35. Check that Sol-PC is in optimum position and tighten the eight screws holding the Sol-PC to the expansion-main chassis assembly. (See Step 27.)

( ) Step 36. Connect Sol-PC power cable (4-wire) to J10 on Sol-PC. CAUTION: Make sure cable connector mates exactly with J10; that is, pin 1 to pin 1, pin 2 to pin 2, etc. Any other mating relationship will damage the IC's on the Sol-PC. (Refer to Step 15 in Section III.)

( ) Step 37. Position keyboard (Sol-KBD) near its mounting brackets and connect 20-conductor ribbon cable supplied with Sol keyboard kit between J1 on keyboard and J3 on Sol-PC. With the cable connected properly, the cable will run away from the keys from J1 on the keyboard, and towards the keys from J3 on Sol-PC.

( ) Step 38. Attach keyboard to keyboard brackets with two 6-32 x ½ binder or pan head screws and #6 lockwashers on each side. Place washer on each screw and drive screws loosely into threaded holes in brackets.

( ) Step 39. If your kit does not include the 15-key numeric pad, install the plastic insert supplied with your Sol keyboard kit to the keyboard cover. Attach it on the right end and to the bottom of the cover with masking tape.

( ) Step 40. Remove protective cover from one side of Plexiglass strip and attach "Sol Terminal Computer" trim plate to Plexiglass with small pieces of transparent tape. Place trim plate with printed side against Plexiglass.

( ) Step 41. Remove protective cover from other side of Plexiglass and slide it into the channel above the keyboard cutout.

NOTE

A hole is provided in the sheet metal behind the trim plate. This may be used for a "power on" indicator light if desired.

( ) Step 42. Install keyboard cover. Hook front of cover under front edge of main chassis and lower it over the keybaord. (A slight adjustment of the keyboard position may be needed to obtain a proper fit.)

Position keyboard within cutout in cover if needed and tighten keyboard mounting screws.

( ) Step 43. Install top cover.

( ) Be sure power cord is not plugged into 110 V ac outlet and disconnect cord from fan closure plate receptacle.

( ) Remove fuse holder cap and fuse.

CAUTION

NEVER REMOVE OR INSTALL FUSE WITH POWER ON.

( ) Hook top cover over back edge of keyboard cover and lower it down into place over the rear of the main chassis. Install the two thumb screws (one at the lower left corner and the other to the right of the fan closure plate coaxial connector) to attach cover to rear of main chassis.

( ) Step 44. Re-install fuse and plug power cord into receptacle. BE SURE POWER CORD IS NOT PLUGGED INTO 110 V ac OUTLET.

See CAUTION on Page VI-20.

(Step 44 continued.)

CAUTION

NEVER REMOVE OR INSTALL FUSE WITH POWER ON.

( ) Step 45. Remove backing from connector identification label and affix it to rear of top cover. Position label just above Sol-PC connector opening in cover so that "J9" is aligned with left most (as viewed from rear of Sol) subminiature phone jack and "J1" is aligned with right most 25-pin female connector.

( ) Step 46. Remove backing from serial number label and affix it to rear of top cover. Position label to right (as viewed from rear of Sol) of fan opening in cover.

( ) Step 47. Affix self-stick protective pads to bottom of Sol as shown in Figure 6-8.

You have now completed assembly of your Sol Terminal Computer^{TM}. It is ready for use as a stand-alone computer or CRT terminal. Congratulations on a job well done.

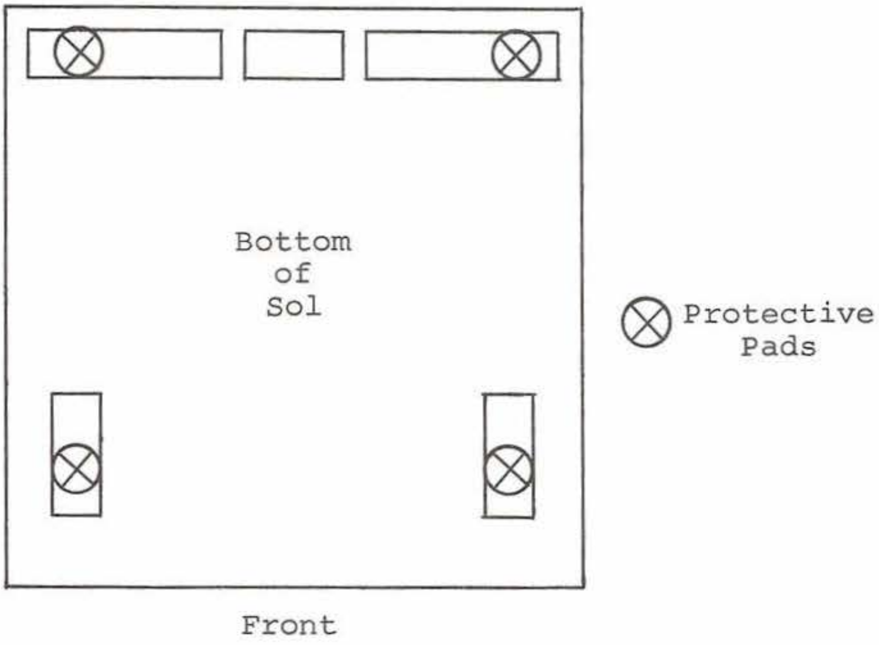


Figure 6-8. Protective foot pad installation.

PARAGRAPH TITLE PAGE

(Continued on next page.)

Sol OPERATING PROCEDURES

7.1 INTRODUCTION

Information in this section will help you to become familiar with the operation of your Sol Terminal Computer™. Following brief explanations of the operating controls and the two basic operating modes, you will put your Sol through some simple operations. This should sufficiently acquaint you with the keyboard and control switches so that you will feel at ease with your Sol. In addition, you will have performed functional tests of all Sol sections except the parallel data interface.

Detailed descriptions of the control switches are also provided to allow you to gain greater proficiency in their use. For the same reason, individual keyboard key descriptions are also given. They are intended to be used along with the BASIC/5 and SOLOS Users' Manuals (or if applicable the CONSOL description in Section IX of this manual).

The balance of this section supplies instructions for 1) connecting typical peripheral devices to the serial and parallel data interfaces (J1 and J2), 2) using audio cassette recorders, and 3) changing the fuse.

7.2 THE OPERATING CONTROLS

Sol operating controls are identified and their functions briefly defined in Table 7-1 on Page VII-2. Unless noted otherwise, the location of each control is shown on the Sol-PC assembly drawing in Section X, Page X-3.

7.3 BASIC OPERATING MODES

7.3.1 Command Mode

In this mode Sol operates as a stand alone computer under control of the program (software) contained in the personality module and additional software that is stored in the Sol, stored either in a read only memory (ROM) that is plugged into the computer or the Sol random access memory (RAM). (For a description of the CONSOL and SOLOS Personality Modules, refer to Section IX in this manual and the SOLOS Users' Manual respectively.)

With the SOLOS Personality Module installed, the computer is in the command mode when power is applied to the Sol. Command mode is a sort of "home base" from which excursions may be made into other programs. An analysis of three levels of programs will make the concept of command mode more understandable.

At the lowest level of software are the instructions which the 8080 CPU (central processing unit), the brains of the computer,

Table 7-1. Sol Operating Controls and Their Functions.

CONTROL	FUNCTION
ON-OFF Switch (See Figure 7-1)	Connects and disconnects primary power to Sol.
RST (Restart) Switch, S1-1	Permits manual restart of Sol without turning power off. (Useful for test purposes.)
BLANK Switch, S1-3	Determines if control characters are displayed or not.
POLARITY Switch, S1-4	Selects normal (white characters on black background) or reverse video display.
BLINK-SOLID Switches, S1-5 & 6	Selects blinking, nonblinking or no cursor.
SSWØ - 7 S2-1 through 8	Permits direct data entry to processor.
BAUD RATE Switches, S3-1 through 8	Sets operating speed of serial data interface (SDI).
PS & PI Switches S4-1 & 5	Selects no parity, even parity or odd parity for SDI.
WLS-1 & 2 Switches, S4-2 & 3	Selects number of data bits in transmitted word for SDI.
SBS Switch, S4-4	Determines number of stop bits in transmitted word for SDI.
F/$\overline{H}$ Switch, S4-6	Selects half or full duplex operation for SDI.
Keyboard (See Figure 7-4)	Data entry, mode selection, command input and cursor control.

can understand and run. All programs must ultimately be reduced to this basic level to be operated on by the computer. In the case of the 8080 microprocessor , the program is in an "object code" or "machine language", since the "machine" or 8080 CPU understands it. The SOLOS program contained in the personality module is stored in this machine language form, and the computer can therefore run directly from this program. Since the SOLOS program is contained in permanent ROM which is plugged directly into the computer, the SOLOS program is always available, and is automatically selected whenever the power switch of the Sol is turned on. There is also provision for returning at all times to the command mode of SOLOS. From the command mode other programs may be brought in for various operations or stored on cassette tape. The contents of the computer's memory may be displayed or changed. The command mode also performs "housekeeping" functions such as setting the rate at which data is read from tape, or the rate at which characters are displayed on the video monitor.

The command mode allows the introduction of the second level of software. This level includes higher-level language programs such as BASIC/5 or FOCAL in which complex application programs may be more easily written. These are called higher level languages because they permit the user to write programs in a form much closer to human languages such as English. However, programs written in these languages must be translated into the more basic machine language before they can be run. Besides higher level languages, this second level of software includes programs such as the TREK 80 and GAMEPAC video games and the ALS-8 program (a software package used for developing programs), all of which are offered by Processor Technology Corporation. Through the facilities of the command mode, these second level programs are transferred (loaded) into memory from cassette tape or other storage media, and then "executed" (used). These programs may also exist in ROM or EPROM (erasable programmable ROM) memory which is plugged into the computer to make them instantly available like the SOLOS program. All first and second level programs are stored in the computer as binary object code.

Let us illustrate the concept of the second level of programs with an example, BASIC/5. Using the "XEQ" command available in the SOLOS command mode, we load the BASIC/5 program into the computer's memory from cassette tape. With this command BASIC/5 is ready for use as soon as the tape has stopped moving. The control of the computer is now taken over by the BASIC/5 program now in memory, and SOLOS is no longer in command. All the features of BASIC/5 language are now available to us, with a new set of commands and rules. Since the CPU of the computer only understands the machine language of the first level of software, the BASIC/5 program must translate the commands and data we enter to this lower level. BASIC/5 does this as we go. While we are using BASIC/5, we still have access to some of the commands and features of SOLOS, although they may have a modified form while we are in BASIC/5. We will load and use BASIC/5 later in this section.

The third level of software consists of programs written using the higher order languages of the second level programs. A program written in BASIC/5 is on this third level. This program only makes sense to the computer while the computer has BASIC/5 in memory and control has been transferred to the BASIC/5 program. Third level programs written in any high level language are often called "applications programs" since they are usually written in order to fit a specific application need.

The ALS-8 Program Development System is another second level program. A program to be developed within ALS-8 would then be a third-level application program. The ALS-8 also includes an Assembler which takes a program written on the third level in "assembly" language, and translates it to object code which the computer can run. The object code version then resides in memory and can be run in another operation. For a further discussion of types of software see the article "Your Personal Genie" in Appendix VIII of this manual.

7.3.2 Terminal Mode

Sol operates as a CRT terminal in this mode, capable of sending keyboard data to an output port and displaying data received at the serial input port on an external video monitor via the Sol video display circuitry. When Sol is "hard-wired" to another computer or connected to a modem, the terminal mode is used for data entry, data retrieval, inquiry/response and monitoring and control applications.

Capabilities in the terminal mode depend on the personality module used. Both CONSOL and SOLOS Personality Modules permit operation as a CRT terminal. CONSOL 1) initializes Sol in the terminal mode whenever you turn the power on or initiate a system reset, 2) sends keyboard data to the serial data interface (SDI) only, and 3) provides simple stand-alone computer capabilities. SOLOS, on the other hand, 1) enters the terminal mode when given the "TERM" (terminal) command, 2) sends keyboard data to any output port available with the "SET O" (set out) command, and 3) duplicates CONSOL functions while providing additional capabilities.

7.4 GETTING ACQUAINTED WITH Sol

One of the best ways to get acquainted with your Sol is to use it. After connecting a cassette recorder and video monitor to your Sol, you will operate the system in the terminal mode to become familiar with the keyboard and the functions of the video display switches. You will then switch to the command mode and perform some of the basic computer operations.

7.4.1 Monitor and Cassette Recorder Connections

The basic Sol system consists of the Sol, a video monitor for display (e.g., the Processor Technology PT-872 TV-Video Monitor by Panasonic) and a cassette recorder for external storage (e.g., the Panasonic Model RQ-413S).

To connect these three system components, you will need the following cables:

> Audio In & Out Cables--two cables of shielded wire fitted with miniature phone plugs at both ends.
>
> Motor 1 Cable--one cable pair, such as speaker wire, fitted with subminiature phone plugs at both ends. (An identical cable for Motor 2 is needed if you use two recorders.)
>
> Video Cable--one RG59/U coaxial cable fitted with a PL259 UHF male connector on one end and a monitor-compatible connector on the other.

Connect the basic Sol system as follows (refer to Figure 7-1 on Page VII-6):

( ) <u>Step 1</u>. Remove top and keyboard covers from Sol.

( ) <u>Step 2</u>. Plug one end of Audio In Cable into Audio IN jack (J7) on Sol rear panel, and plug other end into MONITOR or EARPHONE jack on recorder.

( ) <u>Step 3</u>. Plug one end of Audio Out Cable into Audio OUT jack (J6) on Sol rear panel, and plug other end into AUXILIARY or MICROPHONE jack on recorder. (The AUXILIARY input is preferred and recommended over the MICROPHONE input.)

<u>NOTE</u>

If your recorder has only a microphone jack, remove the I-to-J jumper installed in Step 69 in Section III and install a jumper between I and H.

( ) <u>Step 4</u>. Plug one end of Motor 1 Cable into Motor 1 jack (J8) on Sol rear panel, and plug other end into REMOTE jack on recorder.

( ) <u>Step 5</u>. Connect PL259 UHF connector on Video Cable to video output connector on Sol rear panel, and connect other end to video monitor input connector.

( ) <u>Step 6</u>. Make sure monitor, recorder and Sol power switches are in their OFF position. Then connect AC power cord to AC receptacle on Sol rear panel and connect Sol, monitor and recorder to appropriate power source.

7.4.2 Terminal Mode Operation

The following procedure assumes your Sol is equipped with a SOLOS personality module.

( ) <u>Step 7</u>. Set Sol control switches as follows (see Figure 7-2 on Page VII-7):

RST Switch (S1-1): OFF

S1-2 (spare): OFF

BLANK Switch (S1-3): OFF (display control characters)

POLARITY Switch (S1-4): OFF (reverse video display)

BLINK Switch (S1-5): OFF (solid cursor)

SOLID Switch (S1-6): ON (solid cursor)

(Step 7 continued on Page VII-7.)

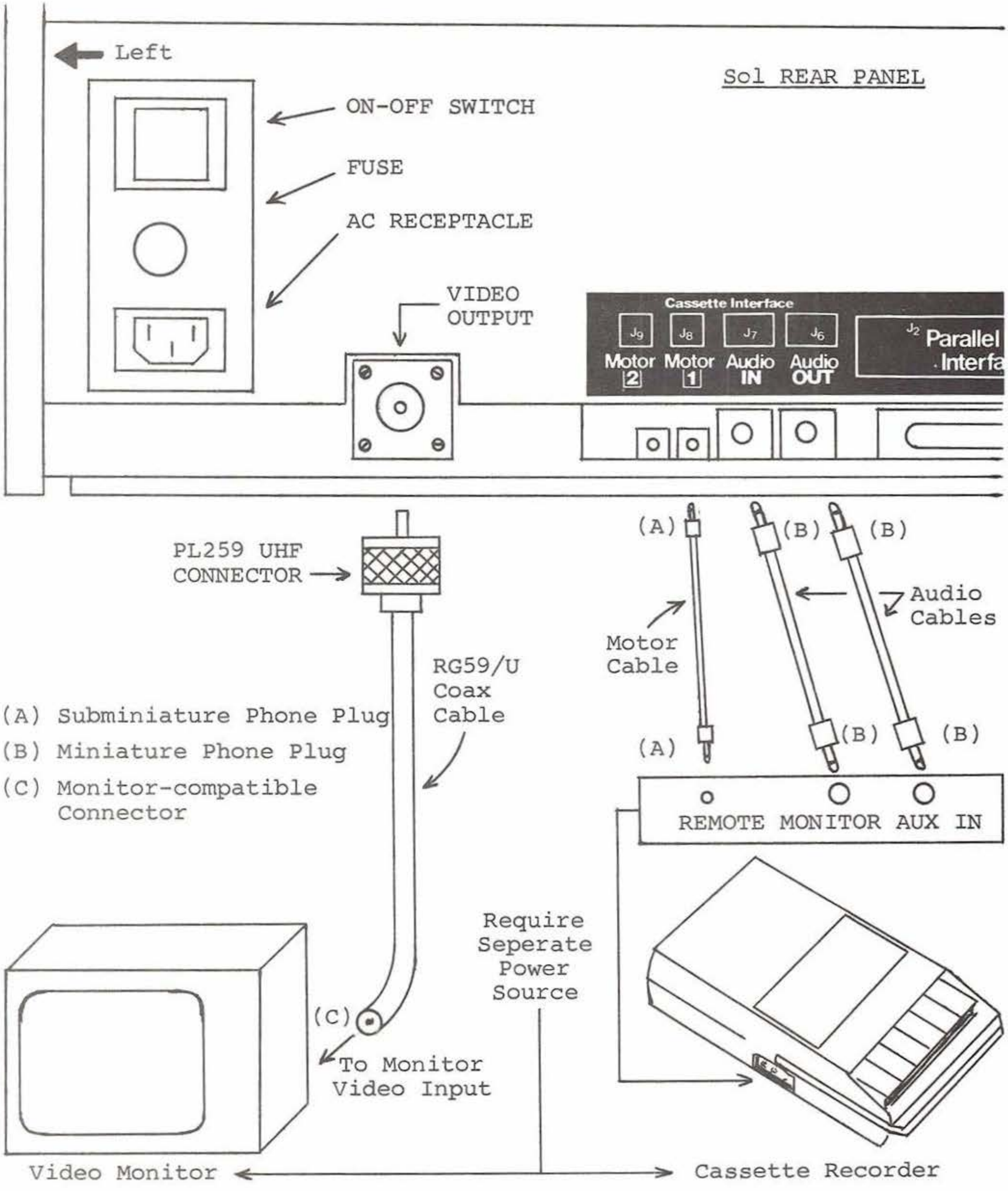


Figure 7-1. Connecting the basic Sol system.

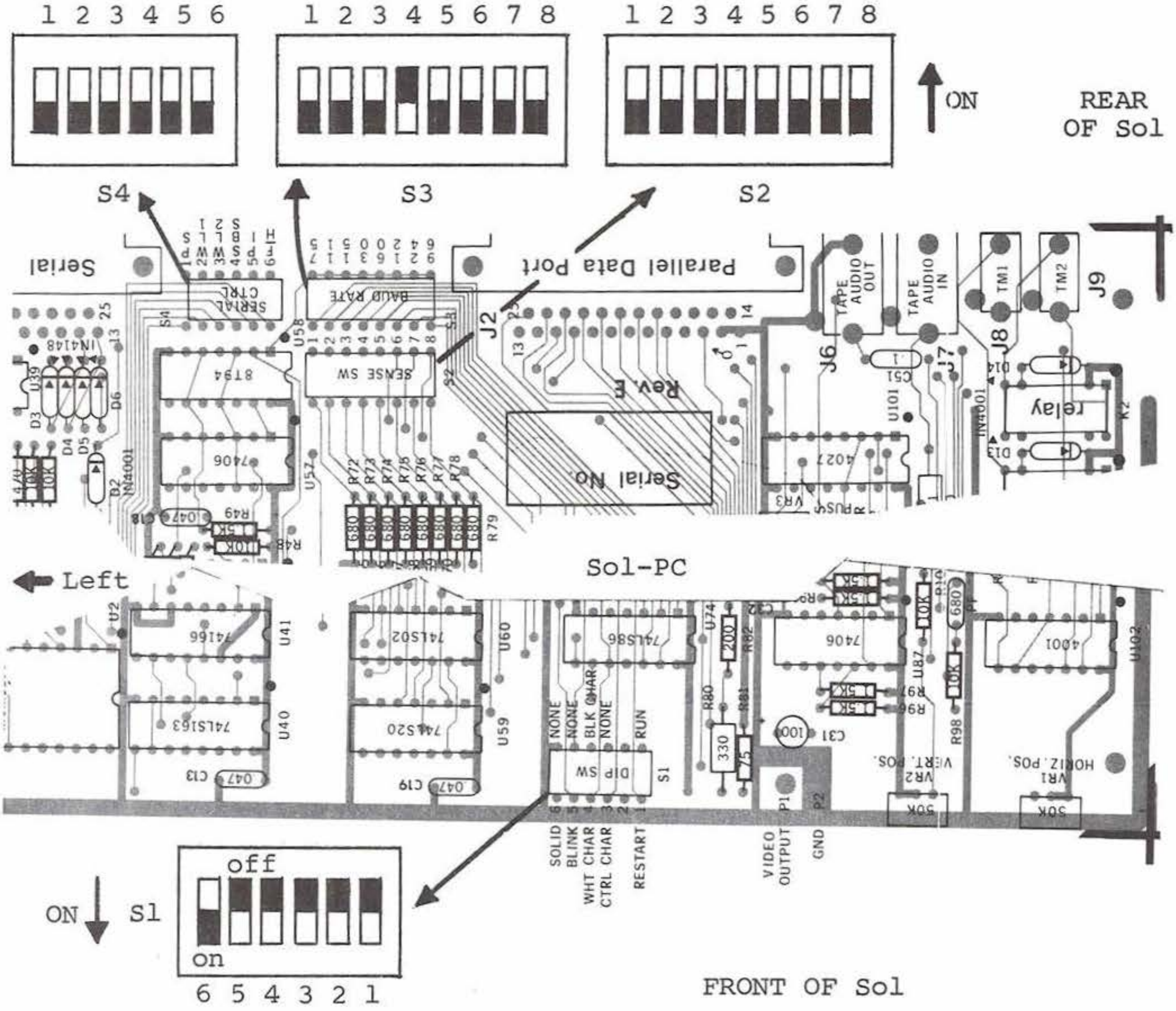


Figure 7-2. Sol control switch settings for terminal mode.

(Step 7 continued.)

SSW Switches (S2-1 - 8): OFF

BAUD RATE Switches (S3-1 - 8): S3-4 ON, all others OFF (300 Baud)

SDI Switches (S4-1 - 6): OFF (selects full duplex operation, 8 data bits, 2 stop bits and no parity)

( ) Step 8. Turn Sol and monitor on.

( ) Step 9. If the monitor display raster is out of sync (black horizontal bar moves slowly down screen, numerous black lines cut across raster, or both), adjust monitor vertical and horizontal hold controls for a stable raster.

( ) Step 10. You should see a prompt character followed by the cursor (>▌) in the upper left corner of the screen. If you don't, adjust VR1 and VR2 (see Figure 7-3) to move the prompt character and cursor onto the screen. (With CONSOL, only the cursor will appear on the screen.)

NOTE

Use VR1 (horizontal position) and VR2 (vertical position) to center the display page (16 lines, 64 characters/line) on the screen.

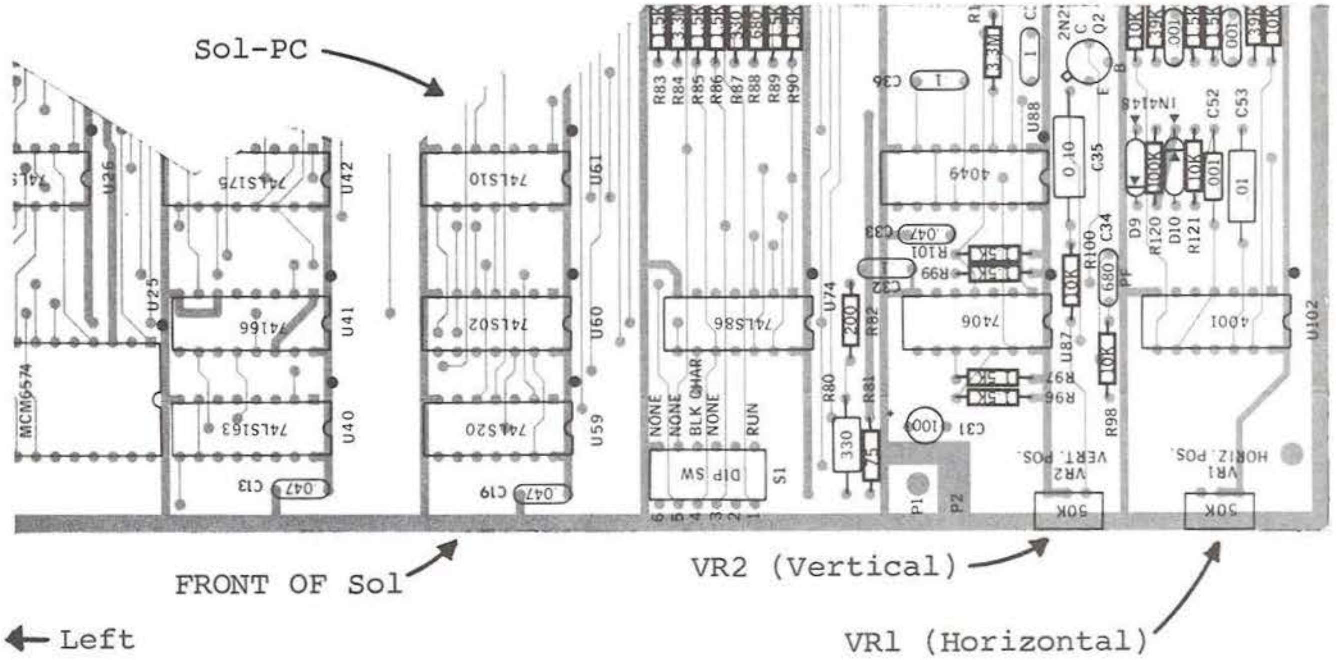


Figure 7-3. Location of positioning adjustments, VR1 and VR2.

( ) Step 11. Enter terminal mode by 1) pressing UPPER CASE key to turn the indicator light on (Alphabetic characters are now entered as upper case, regardless of SHIFT key status, but dual character keys do respond to SHIFT key.), 2) typing TERM and 3) pressing RETURN key. (If your Sol is equipped with CONSOL, it entered terminal mode when you turned the Sol on.) "TERM" will appear on the screen as you type, and the cursor will disappear when you press the RETURN key.

NOTE: All commands must be given in upper case characters in order to be recognized, and the RETURN key must be pressed after a command so that SOLOS can execute the command (MODE SELECT excepted).

( ) Step 12. Set for local operation by pressing LOCAL key to turn indicator light on. Set for lower case operation by pressing UPPER CASE again (indicator light out).

( ) Step 13. Press each of the alphanumeric, punctuation and symbol keys. As each is pressed, the lower case character in the UNSHIFTED column of Table 7-4 should appear on the screen. Read Section 7.7 on page VII-17 to become familiar with Table 7-4.

NOTE: If the MODE SELECT key is pressed, SOLOS will return to the command mode and display a prompt character followed by the cursor. In this case return to terminal mode by typing "TERM" in upper case letters, followed by a carriage return.

( ) Step 14. Press SHIFT LOCK key to return keyboard to shifted operation (indicator light will go out) and repeat Step 13. Each corresponding upper case character should appear from the SHIFTED column of Table 7-4.

( ) Step 15. Use the control sequences given in Table 7-4 on Page VII-18 to generate the indicated control characters. Control characters are generated by pressing the CTRL (control) key and, while holding it depressed, pressing the desired key given in the first column of the table. As the table shows in the last two columns, the symbol generated by a control sequence depends on whether a 6574 or 6575 character generator (U25) is installed in your Sol. Two examples follow:

CONTROL SEQUENCE	6574 SYMBOL	6575 SYMBOL
CTRL and I	→	H_T
CTRL and 5 or %	⊠	E_Q

( ) Step 16. Change video display polarity by setting POLARITY Switch (S1-4) to ON and observe the effect on the display. It should change from black characters on a white background to white characters on a black background.

( ) Step 17. Switch from non-blinking cursor to a blinking cursor by setting SOLID Switch (S1-6) to OFF and BLINK Switch (S1-5) to ON *in that order*. You should see a rectangular solid cursor that blinks on and off approximately two times per second. *Never put S1-5 and S1-6 ON at the same time*.

( ) Step 18. Blank control characters by setting BLANK Switch (S1-3) to ON. Any control characters generated (refer to Step 15) should not appear on the screen.

Up to this point, keyboard data has been processed by the CPU, transmitted out through the serial channel output, looped back to the serial channel input and then displayed on the video monitor. You have consequently just "tested" the CPU, serial channel and display section functions in your Sol.

7.4.3 Command Mode Operation

The following operations assume your Sol is equipped with a SOLOS personality module.

Using the Cassette Recorder. The following procedure for loading a program from cassette tape into Sol memory provides a good example of how to use an audio cassette recorder with Sol. In this example you will use the BASIC/5 cassette supplied with your Sol.

( ) Step 19. Set POLARITY (S1-4) and BLANK (S1-3) Switches as desired.

( ) Step 20. Replace top and keyboard covers.

( ) Step 21. Load BASIC/5 cassette in recorder. If required, fully rewind tape. (This can be done by disconnecting the REMOTE plug from the recorder and using the REWIND control on the recorder.) After rewinding, reconnect REMOTE plug.

( ) Step 22. Set the following recorder controls and indicator, if so equipped, as indicated:

Transport: press STOP control

Volume: midrange

Tone: top of range (maximum treble)

Tape Counter: Ø

( ) Step 23. Press PLAY control on recorder. The tape should not move. If it does, there is a malfunction in the remote control circuitry or cabling. (With the Sol off, there should be no continuity between the REMOTE plug contacts.)

NOTE

The tape head must be clean to reliably read a tape or write on tape.

( ) Step 24. If needed, press MODE SELECT key on Sol to enter command mode. (Remember SOLOS initializes in the command mode while CONSOL initializes in the terminal mode whenever Sol is turned on.) You should see a prompt character followed by the cursor (>▮) on the left of the screen.

( ) Step 25. Type the XEQ command as follows:

```
XEQ BASIC
```

( ) Step 26. Press the RETURN key on Sol. The cursor should disappear and the tape should advance. The display should not change otherwise. NOTE: With certain cassette recorders or cassettes there may be a misreading of the tape when the splice joining the leader to the tape passes the tape head. In this case an ERROR message will appear and the tape will stop. To resume tape "loading", retype the XEQ BASIC command. If further difficulty is encountered, try different cassette recorder volume settings until a reliable setting is found.

( ) Step 27. If the tape has loaded successfully, in approximately two minutes BASIC/5 will display five lines of text ending with:

```
SOL BASIC 5

READY
```

( ) Step 28. BASIC/5 is now ready for use. Refer to your BASIC/5 User's Manual. Become familiar with both BASIC/5 and the Sol keyboard. Try some exercises in BASIC/5.

Dump Operation. The dump operation displays memory data in hexadecimal on the video monitor. It can also be used with the appropriate SET command to output memory data to a hard-copy device (e.g., a printer). As an example, dump the first part of the SOLOS personality module (CØØØ through CØEØ) as follows:

( ) Step 29. Set UPPER CASE key so that the indicator is on. If you are still in BASIC/5, type the BASIC/5 command "BYE" at the beginning of a command line to re-enter SOLOS command mode. BASIC/5 remains in memory and may be returned to by typing a command line: "EXEC Ø".

( ) Step 30. Type the DUMP command as follows:

```
DUMP CØØØ CØEØ
```

( ) Step 31. Press RETURN key. Lines of 16 bytes of hexadecimal data will scroll (move) rapidly up the screen until the last address (CØEØ) is displayed. At this point the display will stop scrolling.

Enter Operation. The enter operation is used to enter hexadecimal data from the keyboard into available Sol memory. As an example, enter 16 bytes of data, starting at address C9ØØ and ending at address C9ØF, as follows:

( ) Step 32. Type the ENTER command as follows:

```
ENTER C9ØØ
```

( ) Step 33. Press RETURN key. The monitor should display a colon (:) prompt character at the start of the next line.

( ) Step 34. Type the following data:

```
11 22 33 44 55 66 77 88 99 ØØ AA BB CC DD EE FF/
```

NOTE

The slash (/) terminates the enter function.

( ) Step 35. If you made a mistake in typing the above line of data, refer to Paragraph 7.8.3 on Page VII-25. If you made no mistakes, press RETURN key.

The data entered in Step 34 now resides in locations C9ØØ through C9ØF in the Sol memory.

( ) Step 36. To verify that the data did indeed enter Sol memory, simply give your Sol this DUMP command:

```
DUMP C9ØØ C9ØF
```

Then press RETURN key. The line of data you entered in Step 35 should be displayed on the monitor screen, preceded by the starting address.

( ) Step 37. Using your SOLOS User's Manual, experiment with the other commands until you feel at home with your Sol.

The preceding command mode operations used the CPU, personality module, audio cassette interface (ACI) and the Sol RAM. You have consequently just tested the functions of these sections.

7.5 OPERATING CONTROLS IN DEPTH

Unless indicated otherwise, the location of the controls described in this paragraph are shown on the Sol-PC assembly drawing in Section X, Page X-3.

7.5.1 ON-OFF Switch (See Figure 7-1 on page VII-6.)

Push this switch in to turn your Sol on. In the ON position the switch remains locked in its "in" position. To turn your Sol off, push the switch again. This releases the locking mechanism, and the switch will return to its OFF ("out") position.

7.5.2 Restart (RST) Switch, S1-1

This switch permits you to restart your Sol without turning the power off. You should normally leave it in its OFF, or run, position. Set it to ON and then OFF to initialize the Sol circuitry and reset the CPU program counter to zero. (A manual restart with this switch performs the same function as turning the power on or pressing a keyboard generated restart: UPPER CASE key with REPEAT key.)

7.5.3 Control Character Blanking (BLANK) Switch, S1-3

Set this switch to its ON position if you do not want control characters (see Table 7-4 on Page VII-18) to be displayed on the screen. In the OFF position, control characters are displayed.

7.5.4 Video Display (POLARITY) Switch, S1-4

If you want a normal video display (white characters on a black background), set this switch to its ON position. In the OFF position, black characters will be displayed on a white background (reverse video display).

7.5.5 Cursor Selection (BLINK, SOLID) Switches, S1-5 & 6

CAUTION

> DO NOT SET S1-5 AND S1-6 TO THEIR ON POSITIONS AT THE SAME TIME. TO DO SO MAY DAMAGE YOUR Sol.

If you want the cursor to blink, set S1-6 to OFF and S1-5 to ON. The cursor will blink on and off about two times per second.

Set S1-5 to OFF and S1-6 to ON if you want a non-blinking (solid) cursor.

With both S1-5 and S1-6 in their OFF positions, there will be no cursor display.

7.5.6 Sense (SSWØ - 7) Switches, S2-1 through S2-8

These eight switches are normally left in the OFF position. They are used to manually enter data into the CPU. (They serve the same function as the front panel sense switches on the Altair 8800 and IMSAI 8080.)

S2-1 is the least significant data bit (DIOØ) and S2-8 is the most significant data bit (DIO7). To pull a DIO bit low (when the program tests SSWØ - 7), set the switch associated with the bit to ON. An open (OFF) switch pulls the associated DIO bit high when the program tests SSWØ - 7.

NOTE

> The configuration of SSWØ - 7 is tested by the CPU only when it executes an input port FF instruction. Otherwise, the Sense Switches have no bearing on Sol operation.

7.5.7 Baud Rate Switches, S3-1 through S3-8

The setting of the Baud Rate Switches determines the operating speed of the Serial Data Interface (SDI). Assuming you have not installed any of the K, L and M jumper options, you can select any one of eight Baud rates. Table 7-2 on page VII-15 defines Baud rate as a function of S3-1 through S3-8.

CAUTION

> DO NOT SET MORE THAN ONE S3 SWITCH TO THE ON POSITION AT THE SAME TIME. TO DO SO CAN DAMAGE YOUR Sol.

7.5.8 Parity (PS, PI) Switches, S4-1 & 5

With these two switches you can select no parity, parity, even parity or odd parity for data handled through the SDI (J1).

Set S4-5 (PI) to its ON position if you want a parity bit. When OFF, there will be no parity bit. (A stop bit immediately follows the data if no parity bit is selected.)

S4-1 (PS) selects even or odd parity if S4-5 is ON. It otherwise has no affect. For even parity, set S4-1 to ON. Set S4-1 OFF for odd parity.

7.5.9 Data Word Length (WLS-1 & 2) Switches, S4-2 & 3

Use these two switches to select the number of bits, excluding parity, in the transmitted word for the SDI. You have a choice of 5, 6, 7 or 8 bits. Table 7-3 defines word length as a function of S4-2 and S4-3.

7.5.10 Stop Bit Selection (SBS) Switch, S4-4

Set this switch to ON if you want one stop bit transmitted out of the SDI. In the OFF position, two stop bits are transmitted unless you have selected a five bit word length. In that case 1.5 stop bits are transmitted.

Table 7-2. Baud Rate Selection With Switch S3.

BAUD RATE	SWITCH S3 CONFIGURATION*
75	S3-1 ON, all others OFF
110**	S3-2 ON, all others OFF
150	S3-3 ON, all others OFF
300	S3-4 ON, all others OFF
600	S3-5 ON, all others OFF
1200	S3-6 ON, all others OFF
2400	S3-7 ON, all others OFF
4800***	S3-8 ON, all others OFF

*Set no more than one switch to ON at the same time.

**Rate required by standard 8-level TTY's (Teletype machine).

***Assumes K-to-M jumper on Sol-PC is not installed. With K-M jumper in and L-M trace on back side of Sol-PC cut, SDI operates at 9600 Baud when S3-8 is ON and all others OFF.

<u>NOTE FOR REV D Sol-PC BOARDS</u>: With S3-7 ON and all others OFF, Baud rate is either 2400 (K-to-M jumper not installed) or 4800 (K-M jumper in and L-M trace on back side of Sol-PC cut). With S3-8 ON and all others OFF, Baud rate is 9600.

Table 7-3. Word Length Selection With S4-2 & 3.

WORD LENGTH	SWITCH SETTINGS	
(Number of Bits)	S4-2	S4-3
5	ON	ON
6	ON	OFF
7	OFF	ON
8	OFF	OFF

7.5.11 Full/Half Duplex (F/$\overline{H}$) Switch, S4-6

Set this switch to ON if you want half duplex operation in the terminal mode. In half duplex operation, data transmitted out the SDI (J1) is "looped back" and received by the SDI for subsequent display on the monitor. Use this type of operation when your Sol works with an external computer that does not "echo" data back to the Sol.

For full duplex operation in the terminal mode, set S4-6 to OFF. Only received data is displayed in full duplex operation. Use full duplex when Sol's transmitted data need not be displayed. (Note that transmitted data from the Sol, if echoed back, is displayed as received data.)

NOTE

If no Baud rate is selected, data will not be transmitted out of the SDI.

7.5.12 Keyboard

The keyboard is an output device that produces ASCII (American Standard Code for Information Interchange) encoded data. It is hard-wired to an input port on the Sol and is used for data entry. ASCII data is interpreted by the Sol as data and/or commands as determined by the current system monitor program. The monitor program may be in the personality module, ALS-8, Sol RAM memory or some memory.

7.6 THE KEYBOARD, GENERAL DESCRIPTION

The Sol Terminal Computer has ASCII 96-character keyboard. Its key arrangment conforms with the QWERTY (standard typewriter) format. As shown in the photo on page X-26, there are also 12 control keys (including five basic cursor controls) and seven special function keys. A 15-key arithmetic pad, available as an option on the Sol-10, is provided as standard equipment on the Sol-20.

7.6.1 Operating Features

The Sol keyboard features N-key rollover. That is, several keys can be pressed at the same time without loss of characters or commands. Key entries, however, are in the order of actual key closures. (The keyboard circuitry includes a scanning circuit that prevents simultaneous key operation.)

7.6.2 Keyboard Indicators

Three keys (SHIFT LOCK, UPPER CASE and LOCAL) have indicator lights to indicate keyboard/terminal status. When any of these keys is pressed to turn an indicator light on, the light remains on after the key is released to show that the status persists. Pressing the key again turns the light out to indicate the change in status.

7.7 INDIVIDUAL KEY DESCRIPTIONS

The exact function of most keys on the Sol keyboard is determined by the software used (e.g., the personality module). Others have predefined functions that are common to the CONSOL and SOLOS Personality Modules. (Note that any key that generates a code can be redefined by a program to perform a specific funstion.) The code generated by each key on the keyboard and the corresponding character, or symbol, produced by the Sol's character generator (U25) are given in Table 7-4 on Pages VII-18 through VII-21.

Table 7-4 has two main headings: 1) KEY which identifies the keys on the Sol keyboard and 2) HEXADECIMAL CODE/CHARACTER GENERATION which specifies for each key the hexadecimal code generated by the keyboard and the symbol produced by the Sol's character generator. The second heading is divided into three major categories: UNSHIFTED, SHIFTED and CONTROL. UNSHIFTED defines the results when operating the keys unshifted (lower case), SHIFTED provides the same information when they are operated shifted (upper case), and CONTROL defines the results of control sequences (refer to Paragraph 7.7.7 on Page VII-22). Within each of these three categories you will find the hexadecimal code generated and the symbol displayed in response to that code by either of the two possible character generators that can be supplied with your Sol, the 6574 and 6575. Some keys move the cursor without displaying a new character.

Looking at the "W" entry on Page VII-18 and reading across the table, we see that:

1. Pressing "W" unshifted would generate the code 77 and either character generator (6574 or 6575) produces a lower case "W" (w). Do not actually press the keys at this point.

2. Pressing "W" shifted would generate the code 57 and either character generator would produce an upper case "W" (W).

3. Pressing CTRL (control) and "W", whether shifted or unshifted, generates the code 17 which causes the 6574 to produce the graphic symbol ⊣ for the ASCII "end of transmission block" control character and the 6575 to produce a two-character mnemonic (E_B) for that same control character.

In the following paragraphs, each key function is described in terms of its role in the terminal mode only and assumes the control character display option is enabled and the LOCAL indicator light is on. Many key functions differ from these descriptions in SOLOS command modes BASIC/5, ALS-8, etc. As an aid to learning each key location, we suggest that you keep the keyboard photo, X-26, in view as you study these functions.

7.7.1 Alphanumeric-Punctuation-Symbol Keys

These keys enter the applicable character into the Sol.

Table 7-4. Sol Keyboard Assignments.

KEY#	HEXADECIMAL CODE/CHARACTER GENERATION								
	UNSHIFTED			SHIFTED			CONTROL		
	Hex. Code	Symbol Displayed*		Hex. Code	Symbol Displayed*		Hex. Code	Symbol Displayed*	
		6574	6575		6574	6575		6574	6575
STANDARD KEYS									
ESCAPE	1B	None	None	1B	None	E_C	1B	None	None
1 | !	31	1	1	21	!	!	01	┌	S_H
2 | "	32	2	2	22	"	"	02	⊥	S_X
3 | #	33	3	3	23	#	#	03	┘	E_X
4 | $	34	4	4	24	$	$	04	↯	E_T
5 | %	35	5	5	25	%	%	05	☒	E_Q
6 | &	36	6	6	26	&	&	06	✓	A_K
7 | '	37	7	7	27	'	'	07	⍾	B_L
8 | (	38	8	8	28	(	(	08	↶	B_S
9 |)	39	9	9	29	)	)	09	→	H_T
Ø |	30	Ø	Ø	20	None	None	00	None	None
— | =	2D	—	—	3D	=	=	0D	Return	Return
^ | ~	5E	^	^	7E	~	~	1E	◲	R_S
[| {	5B	[	[	7B	{	{	1B	None	None
\ | ¦	5C	\	\	7C	¦		1C	◳	F_S
] | }	5D	]	]	7D	}	}	1D	◱	G_S
BREAK	None	None	None	None	None	None	None	None	None
TAB	09	→	H_T	09	→	H_T	09	→	H_T
Q	71	q	q	51	Q	Q	11	◷	D_1
W	77	w	w	57	W	W	17	⊣	E_B
E	65	e	e	45	E	E	15	☒	E_Q
R	72	r	r	52	R	R	12	◶	D_2
T	74	t	t	54	T	T	14	◵	D_4
Y	79	y	y	59	Y	Y	19	⫯	E_M
U	75	u	u	55	U	U	15	⍻	N_K
I	69	i	i	49	I	I	09	→	H_T

Table 7-4. Sol Keyboard Assignments. (Continued)

KEY#	HEXADECIMAL CODE/CHARACTER GENERATION								
	UNSHIFTED			SHIFTED			CONTROL		
	Hex. Code	Symbol Displayed*		Hex. Code	Symbol Displayed*		Hex. Code	Symbol Displayed*	
		6574	6575		6574	6575		6574	6575
STANDARD KEYS (Continued)									
O	6F	o	o	4F	O	O	0F	⊙	S_I
P	70	p	p	50	P	P	10	⊟	D_L
@ │ `	40	@	@	60	`	`	00	None	None
RETURN	0D	←	C_R	0D	←	C_R	0D	Return	Return
LINE FEED	0A	Line Feed	Line Feed	0A	Line Feed	Line Feed	0A	Line Feed	Line Feed
CTRL	None	None	None	None	None	None	None	None	None
SHIFT LOCK	None	None	None	None	None	None	None	None	None
A	61	a	a	41	A	A	01	┌	S_H
S	73	s	s	53	S	S	13	⊖	D_3
D	64	d	d	44	D	D	04	↯	E_T
F	66	f	f	46	F	F	06	✓	A_K
G	67	g	g	47	G	G	07	🔔	B_L
H	68	h	h	48	H	H	08	↶	B_S
J	6A	j	j	4A	J	J	0A	Line Feed	Line Feed
K	6B	k	k	4B	K	K	0B	↓	V_T
L	6C	l	l	4C	L	L	0C	↡	F_F
; │ +	3B	;	;	2B	+	+	0B	↓	V_T
: │ *	3A	:	:	2A	*	*	0A	Line Feed	Line Feed
DEL │ _	7F	None	None	5F	Delete	Delete	1F	◳	U_S
REPEAT	None	None	None	None	None	None	None	None	None
CTRL	None	None	None	None	None	None	None	None	None
UPPER CASE	None	None	None	None	None	None	None	None	None
SHIFT	None	None	None	None	None	None	None	None	None
Z	7A	z	z	5A	Z	Z	1A	$	S_B
X	78	x	x	58	X	X	18	⧗	C_N
C	63	c	c	43	C	C	03	┘	E_X

*See notes at end of this table, Page VII-21.

Table 7-4. Sol Keyboard Assignments. (Continued)

KEY#	HEXADECIMAL CODE/CHARACTER GENERATION								
	UNSHIFTED			SHIFTED			CONTROL		
	Hex. Code	Symbol Displayed*		Hex. Code	Symbol Displayed*		Hex. Code	Symbol Displayed*	
		6574	6575		6574	6575		6574	6575
STANDARD KEYS (Continued)									
V	76	v	v	56	V	V	16	⎍	S_Y
B	62	b	b	42	B	B	02	⊥	S_X
N	6E	n	n	4E	N	N	0E	⊗	S_O
M	60	m	m	40	M	M	0D	Return	Return
, <	2C	,	,	3C	<	<	0C	↓	F_F
. >	2E	.	.	3E	>	>	0E	⊗	S_O
/ ?	2F	/	/	3F	?	?	0F	⊙	S_I
SHIFT	None	None	None	None	None	None	None	None	None
LOCAL	None	None	None	None	None	None	None	None	None
Space Bar	20	None	None	20	None	None	20	None	None
ARITHMETIC PAD KEYS									
—	2D	—	—	2D	—	—	2D	—	—
*	2A	*	*	2A	*	*	2A	*	*
÷	2F	/	/	2F	/	/	2F	/	/
7	37	7	7	37	7	7	37	7	7
8	38	8	8	38	8	8	38	8	8
9	39	9	9	39	9	9	39	9	9
4	34	4	4	34	4	4	34	4	4
5	35	5	5	35	5	5	35	5	5
6	36	6	6	36	6	6	36	6	6
1	31	1	1	31	1	1	31	1	1
2	32	2	2	32	2	2	32	2	2
3	33	3	3	33	3	3	33	3	3
Ø	30	Ø	Ø	30	Ø	Ø	30	Ø	Ø
.	2E	.	.	2E	.	.	2E	.	.
+	2B	+	+	2B	+	+	2B	+	+

Table 7-4. Sol Keyboard Assignments. (Continued)

KEY#	HEXADECIMAL CODE/CHARACTER GENERATION								
	UNSHIFTED			SHIFTED			CONTROL		
	Hex. Code	Symbol Displayed*		Hex. Code	Symbol Displayed*		Hex. Code	Symbol Displayed*	
		6574	6575		6574	6575		6574	6575
SPECIAL KEYS									
LOAD	8C	None	F_F	8C	None	F_F	8C	None	F_F
MODE SELECT	80	None	None	80	None	None	80	None	None
↑	97	None	None	97	None	None	97	None	None
←	81	None	None	81	None	None	81	None	None
→	93	None	None	93	None	None	93	None	None
↓	9A	None	None	9A	None	None	9A	None	None
HOME CURSOR	8E	None	None	8E	None	None	8E	None	None
CLEAR	8B	None	None	8B	None	None	8B	None	None

#Vertical line between characters indicates dual character key. *Character generated is displayable and transmittable. "None" means no code is generated or no symbol is displayed. Return is defined in Section 7.7.11, and line feed in Section 7.7.12, on page VII-24.

7.7.2 Space Bar

Pressing the Space Bar, shifted or unshifted, generates the ASCII space code (2Ø) and moves the cursor one space to the right.

7.7.3 Arithmetic Pad Keys

Except for the division symbol key (÷), these keys enter the applicable character into the Sol. The division symbol key enters a forward slash (/) character. SHIFT does not affect these keys.

The arithmetic pad is useful for entering large amounts of numerical data. Each key in the pad duplicates its corresponding numeric, period (decimal point), dash (minus), plus (addition), asterisk (multiplication) and forward slash (division) key in the "typewriter" group of keys. That is, pressing one of the pad keys does the same thing as pressing its corresponding key in the "typewriter" group.

7.7.4 ESCAPE Key

Pressing ESCAPE, shifted or unshifted, generates the ASCII escape character (1B). The character is displayed.

7.7.5 BREAK Key

Pressing BREAK, shifted or unshifted, forces the SDI output line to a space level for as long as the key is depressed. No character is displayed. (Some communications systems use this feature.)

7.7.6 TAB Key

Pressing TAB, shifted or unshifted, generates the ASCII horizontal tab character (Ø9). The character is displayed.

7.7.7 Control (CTRL) Key

CTRL, shifted or unshifted, is used with alphanumeric, punctuation and symbol keys to initiate functions or generate the characters defined in Table 7-4. Table 7-5 defines the ASCII control characters. The characters in Table 7-5 are not always displayed on the video monitor.

A control sequence (e.g., CTRL plus J, which produces ASCII line feed) requires that CTRL be pressed first and held down while the other key or keys are pressed in sequence.

7.7.8 SHIFT Key and SHIFT LOCK Key/Indicator

The SHIFT key generates no code and is thus not displayed. It is interpreted as a direct internal operation, and when pressed specifically shifts the keyboard from lower case to upper case and from the lower to upper character on dual character keys as on a typewriter. The keyboard remains in upper case as long as SHIFT is held down.

Pressing SHIFT LOCK to turn the indicator light on electronically locks the SHIFT key in the upper case position. Again, no code is generated and no character is displayed. Pressing SHIFT returns the keyboard to lower case and causes the SHIFT LOCK indicator light to go out.

7.7.9 UPPER CASE Key/Indicator

Pressing this key, shifted or unshifted, to turn the indicator light on activates the upper case keyboard function so that all alphabetic characters entered from the keyboard, regardless of SHIFT key status, are transmitted as upper case characters. (Dual character keys, however, do respond to the SHIFT key.) With the indicator light on, the Sol keyboard essentially simulates a teletype (TTY) keyboard.

Pressing UPPER CASE to turn the indicator light off returns the keyboard to normal SHIFT key operation.

Table 7-5. Control Character Symbols and Definitions.

HEXADECIMAL CODE	SYMBOL GENERATED BY		DEFINITION
	6574 Generator	6575 Generator	
06	✓	AK	Acknowledge
07	⍾	BL	Bell
08	↶	BS	Backspace
18	⧖	CN	Cancel
0D	←	CR	Carriage Return
11	◷	D1	Device Control 1
12	◶	D2	Device Control 2
13	◵	D3	Device Control 3
14	◴	D4	Device Control 4
7F	▨	▨	Delete
10	⊟	DL	Data Link Escape
17	⊣	EB	End of Transmission Block
1B	⊖	EC	Escape
19	⍿	EM	End of Medium
05	⌧	EQ	Enquiry
04	⌁	ET	End of Transmission
03	⌋	EX	End of Text
0C	↡	FF	Form Feed
1C	◰	FS	File Separator
1D	◱	GS	Group Separator
09	→	HT	Horizontal Tab
0A	≡	LF	Line Feed
15	⍻	NK	Negative Acknowledge
00	□	NU	Null
1E	◲	RS	Record Separator
1A	§	SB	Substitute
01	┌	SH	Start of Heading
0F	⊙	SI	Shift In
0E	⊗	SO	Shift Out
02	⊥	SX	Start of Text
16	⎍	SY	Synchronous Idle
1F	·◳	US	Unit Separator
0B	↓	VT	Vertical Tab

7.7.10 LOCAL Key/Indicator

The LOCAL key internally connects the SDI output to the SDI input and disables serial transmission. No character is displayed. Pressing LOCAL, shifted or unshifted, to turn the indicator light on sets Sol for local operation. Keyboard entries are not transmitted, but they are "looped back" to the SDI input for display. That is, Sol is not on "line". Pressing LOCAL to turn the light off ends local operation. This corresponds to the local/line operation of a TTY.

7.7.11 RETURN Key

Pressing RETURN, shifted or unshifted, generates the ASCII carriage return character (ØD), which is not displayed, and moves the cursor to the start of the line on which it resided prior to RETURN being depressed. (This is the same action as a TTY carriage return.) RETURN also erases all data in the line to the right of the cursor.

7.7.12 LINE FEED Key

Pressing LINE FEED, shifted or unshifted, generates the ASCII line feed character (ØA), which is not displayed, and moves the cursor vertically downward one line. (This is the same action as a TTY line feed.) Line feed action does not erase any data in the line to the right of the cursor.

7.7.13 LOAD Key

The LOAD key character is displayed, but the key is non-functional with CONSOL and SOLOS. The code generated by this key is 8C, and it may be used by a program to meet a specific need.

7.7.14 REPEAT Key

The REPEAT key generates no character and is consequently not displayed. Pressing REPEAT, shifted or unshifted, and another key at the same time causes the other key to repeat at an approximate rate of 15 times per second as long as both keys are held down. Pressing REPEAT at the same time as UPPER CASE performs a restart. See Section 7.5.2 on page VII-13.

7.7.15 MODE SELECT Key

Pressing this key, shifted or unshifted, generates the code 8Ø and causes Sol to enter the command mode.

7.7.16 CLEAR Key

Pressing CLEAR, shifted or unshifted, erases the entire screen and moves the cursor to its "home" position (upper left corner of the screen).

7.7.17 Cursor Control (HOME CURSOR and Arrows) Keys

Five keys control basic cursor movement. They are HOME CURSOR and the four keys with arrows. None are affected by SHIFT status, and none are displayed or transmitted.

Pressing HOME CURSOR moves the cursor to its home position--the first character position in the upper left corner of the screen.

To move the cursor up, down, left or right, press the applicable "arrow" key. Each time you press a key the cursor moves one unit in the direction you wish--one space horizontally or one line vertically. These keys may be used with REPEAT. The cursor will not move across any margin of the screen with these four keys.

7.8 BASIC OPERATIONS

7.8.1 Switching From Terminal To Command Mode

To switch from terminal to command mode, simply press the MODE SELECT key. Sol enters the command mode, issues a prompt character (>) and waits for a command input.

7.8.2 Switching From Command To Terminal Mode

To switch from command to terminal mode, press UPPER CASE, TERM and RETURN in that order. Sol enters the terminal mode and all keyboard data will be sent to the SDI output and all data received (including "looped back" data) will appear on the screen.

7.8.3 Entering Commands In The Command Mode

The various commands for CONSOL and SOLOS are described in Section IX of this manual and the SOLOS Users' Manual respectively.

You can place more than one command on the screen. For each command, use the arrowed cursor control keys to position the cursor at the start of a new line and begin the new command line with a prompt character (>).

A command is executed when you press the RETURN key, and all characters on the line to the left of the cursor are interpreted as the command. This means that if more than one command line is on the screen, you can execute any one of them as follows: position the cursor with the arrowed cursor control keys to the right of the desired command and press RETURN.

Should you make a mistake when entering a command, there are two ways to correct it:

(Paragraph 7.8.3 continued on Page VII-26.)

1. If you see the error immediately (the error is to the immediate left of the cursor), press the DEL key (unshifted) to erase the mistake. Then make the correction.

2. If the error is more than one character position to the left of the cursor, use the arrowed cursor control keys to position the cursor over the mistake. Then make the correction

7.8.4 Keyboard Restart

To perform a keyboard restart, press the UPPER CASE and REPEAT keys at the same time. This key combination performs the same function as a power on initialization or setting the RST switch to ON. Use the keyboard restart to return to SOLOS/CONSOL from 1) a program which does not recognize the MODE SELECT key or 2) a program that is stuck in an endless loop.

7.9 Sol-PERIPHERAL INTERFACING

7.9.1 Audio Cassette Recorders

Your Sol is capable of controlling one or two recorders. The interconnect requirements for one recorder were previously covered in Paragraph 7.4.1 in this section.

Since the Sol has only one audio input and one audio output jack, however, the interconnect requirements for two recorders are somewhat different than for one.

You will need two "Y" adapters, one to feed the single Sol audio output to the AUXILIARY input of two recorders and the other to feed the MONITOR output of two recorders to the single Sol audio input. (If you intend to use the Audio In and Out cables described in Paragraph 7.4.1 in this section, miniature phone jack-to-two miniature phone plug adapters are required.) Since the recorder outputs are most likely unbalanced, we also suggest that you incorporate 1000 ohm resistors in the MONITOR adapter as shown in Figure 7-5 on Page VII-29. Figure 7-5 also illustrates, in schematic form, how to connect two recorders to your Sol.

When using two recorders you may read or write to both under program control as well as read one tape while writing on the other.

If you intend to read one tape while writing on the other, however, you may have to disconnect the MONITOR plug from the write unit, with the need for disconnect being determined by the recorder design. The MONITOR disconnect must be made if the recorder has a "monitor" output in the record mode. (Panasonic RQ-413S and RQ-309DS do, for example.)

NOTE 1

> Recorders on which the "monitor" jack is labeled MONITOR usually provide a monitor output in the record mode. If the jack is labeled EAR or EARPHONE, the recorder usually does not provide a monitor output in the record mode.

NOTE 2

> To determine if your recorder provides a monitor output in the record mode, install a blank tape, plug earphone into "monitor jack and microphone into MICROPHONE jack, set recorder controls to record, and speak into microphone while listening with the earphone. If you hear yourself through the earphone, your recorder does provide a monitor output in the record mode.

Write Operations. Other than placing the recorder(s) in the record mode, loading the cassette(s) and making sure that the head(s) is on tape (not leader), no manual operations are needed to write on tape.

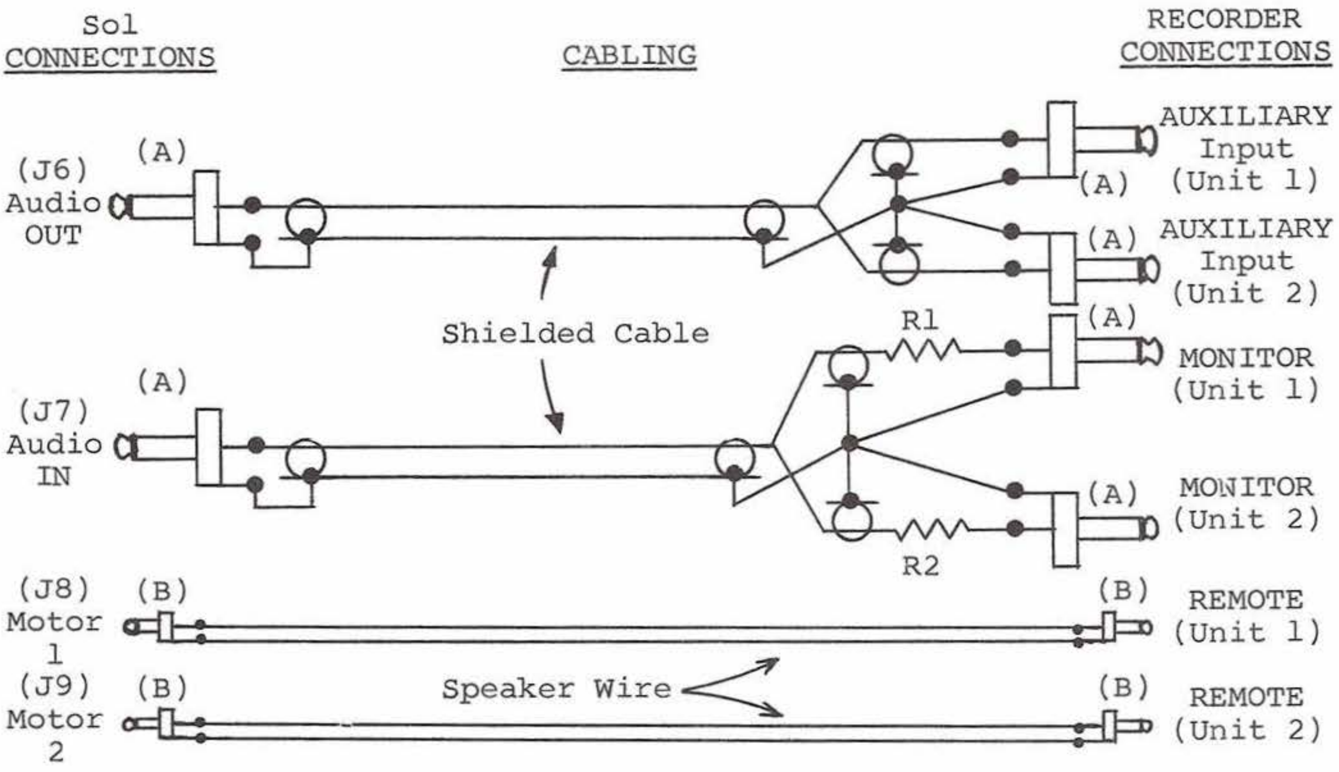
In the case of two recorders, however, Unit 1 and 2 must be specified in the SAVE command in order to select the desired recorder. A default selects Unit 1. Refer to your SOLOS Users' Manual for instructions on how to use tape commands.

Read Operations. In order to read a specific file on tape, you must start the tape at least two seconds ahead of that file. This delay allows the Sol audio cassette interface circuitry and the recorder playback electronics to stabilize after power is turned on. Since all file searches are in the forward direction, the simplest approach is to fully rewind the cassette(s) before a read operation unless you know that the file of interest is advanced at least two seconds. (See Paragraph 7.4.3, Step 21 for instructions on how to rewind the tape.)

For a read operation, proceed as follows:

1. Load cassette(s) as just described.

2. If only one recorder is used, set its volume control at midrange. With two recorders, set both volume controls at their high end.



(A) Miniature Phone Plug
(B) Subminiature Phone Plug

R1 = R2 = 1000 ohms, ¼ watt

Figure 7-5. Connecting Sol to two cassette recorders.

3. Set recorder(s) tone control(s) at the top of the range (maximum treble).

4. Set PLAY control(s) for playback mode.

5. Give Sol the GET or "GET, then Execute" command as appropriate. (Refer to your SOLOS Users' Manual for instructions on how to use tape commands.)

7.9.2 Serial Data Interface (SDI)

The Sol Serial Data Interface (J1) is capable of driving an RS-232 device, such as a modem, or a current loop device, such as the ASR33 TTY.

S3 (Baud Rate) and S4 (Parity, Word Length, Stop Bits and Full/Half Duplex) are used to select the various serial interface options as described in Paragraphs 7.5.7 through 7.5.11 in this section.

Set S3 switches to select the Baud rate required by the modem or current loop device. (Standard 8-level TTY's operate at 110 Baud, S3-2 ON and all other S3 switches OFF.) For standard 8-level TTY's and most modems, set all S4 switches OFF. (This selects eight data bits, two stop bits, no parity bit and full duplex operation for the SDI.

Figures 7-6 and 7-7 show examples of current loop and modem interconnections to the Sol SDI connector (J1). The ASR33 TTY is used to illustrate a current loop interconnect, and the Bell 103 modem is used to illustrate a modem interconnect.

When operating in the terminal mode and full duplex, Sol keyboard data is transmitted out on Pin 2 of J1 and date received on Pin 3 of J1 is displayed on the video monitor. In the command mode, SOLOS set in and out commands can be used to channel output data and input data through the SDI. (Refer to your SOLOS Users' Manual for instructions on how to use the set commands.)

In either mode, the LOCAL key directly controls the SDI. With the LOCAL indicator light on, received data is ignored and keyboard data is not transmitted. It is, however, looped back for display on the video monitor. With the LOCAL light off, received data is displayed and keyboard data is transmitted but not displayed unless it is echoed back.

7.9.3 Parallel Data Interface (PDI)

The Sol Parallel Data Interface (J2) is used to drive parallel devices such as paper tape readers/punches and line printers. It provides eight output data lines, eight input data lines, four handshaking signals and three control signals. The latter allow up to four devices to share the PDI connector. (See Appendix VII for J2 pinouts.)

The port address for parallel input and output data is FD (hexadecimal), and the control port address for the PDI is FA (hexadecimal). PXDR is available at bit 2 of port FA. When this bit is set to Ø, the external device is ready to receive a byte of data. PDR is available at bit 1 of port FA, with Ø indicating the external device is ready to send a byte of data. Parallel Unit Select (PUS) is controlled by bit 4 of port FA. The input and output enable lines are available for tri-stating an external two-way data bus.

Use of the three control signals is optional and is unnecessary when only one device is connected to the PDI connector.

(Paragraph 7.9.3 continued on Page 31.)

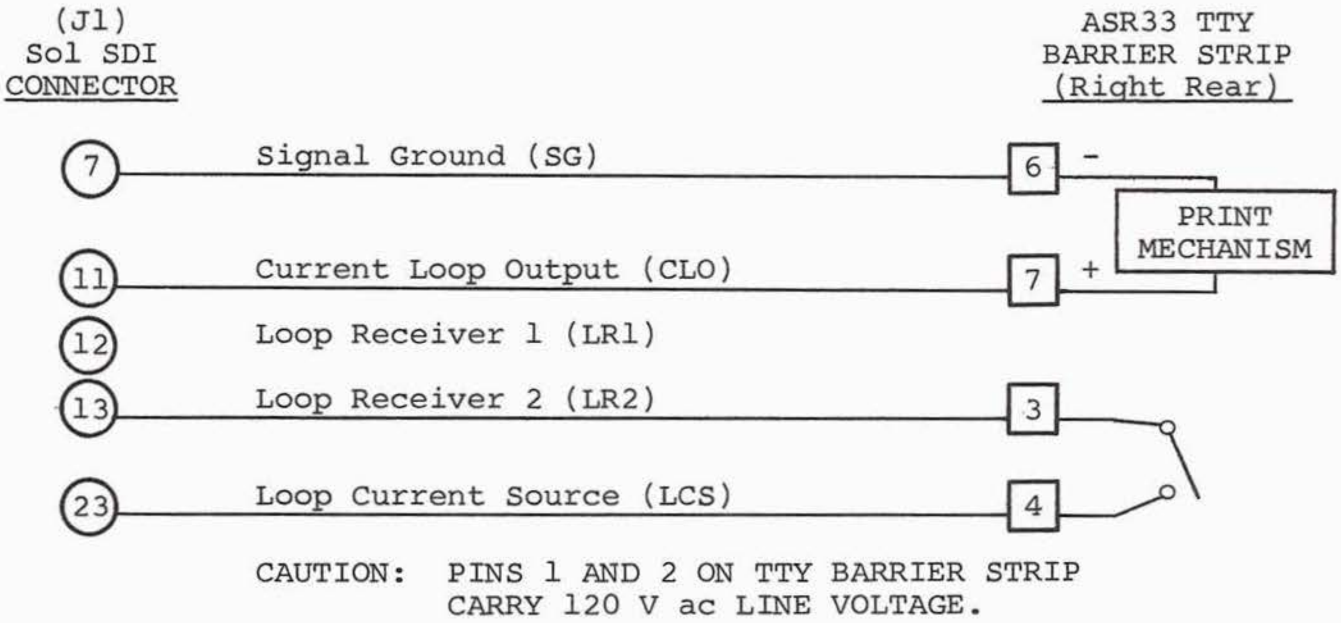
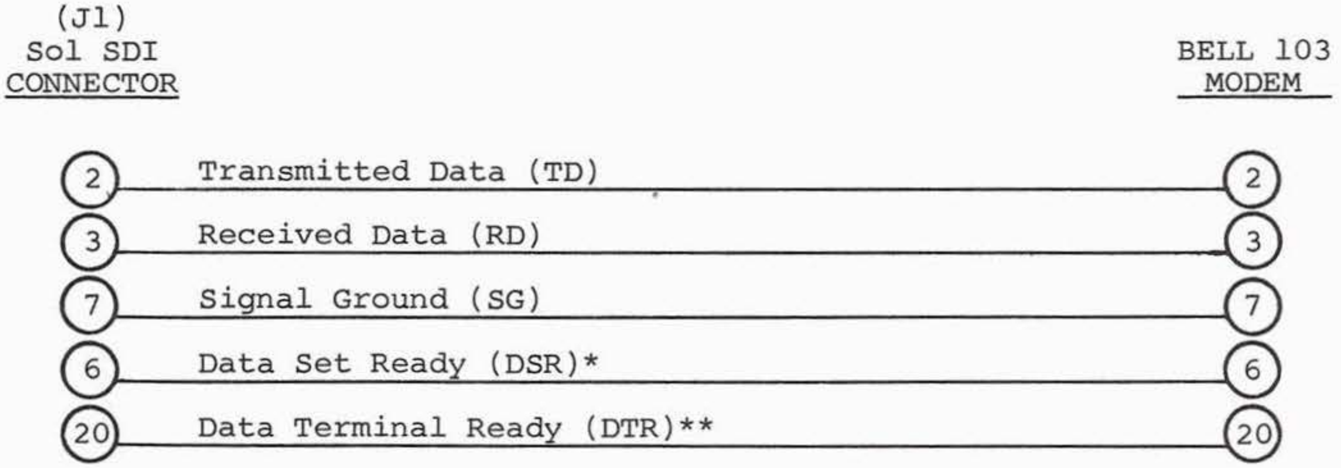


Figure 7-6. Connecting Sol SDI to current loop device such as TTY.



*Available at bit 1 of port F8. Terminal mode software (SOLOS et al) does not use this signal and transmits data whether or not the modem is ready.

**Sol is wired so that DTR indicates a ready condition whenever power is on.

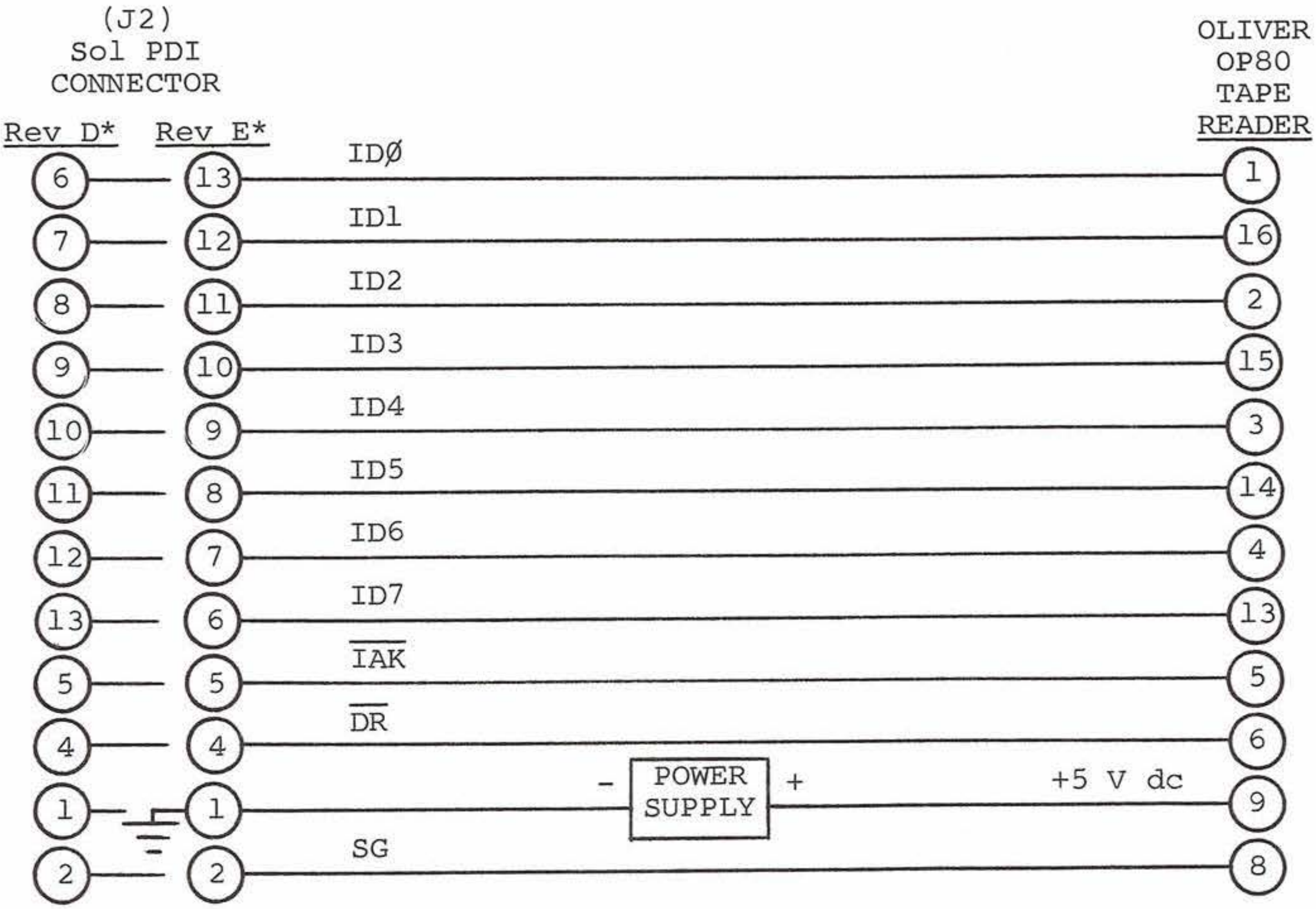
Figure 7-7. Connecting Sol SDI to communications modem.

In Figure 7-8, the Oliver OP80 Manual Paper Tape Reader is used to illustrate a typical PDI interconnect.

7.10 CHANGING THE FUSE

Sol is protected with a 3.0 amp Slo-Blo fuse housed on the rear panel (see Figure 7-1 on Page VII-6). To remove the fuse, turn Sol off, disconnect power cord, turn fuse post cap one quarter turn counterclockwise, pull straight out and remove fuse from cap.

To install a fuse, insert fuse in cap, push in and turn one-quarter turn clockwise.



NOTE: +5 V dc is not available at J2. The use of an external +5 V dc power supply with its ground connected to Pin 1 of J2 (Sol chassis ground) is recommended.

*Sol-PC Board

Figure 7-8. Connecting Sol PDI to parallel device.

VIII THEORY OF OPERATION

TABLES AND ILLUSTRATIONS

8.1 INTRODUCTION

This section concerns itself with the hardware aspects of the Sol Terminal Computer T.M.. It specifically deals with the operation of the power supply and the logic associated with the Sol-PC and keyboard. Descriptions of software and the operation of the circuitry contained in the multitude of integrated circuits (IC's) used in the Sol fall outside the scope of this section. In some cases, references to other publications or sections in this manual are provided when it is felt that additional information will contribute to a better understanding of how Sol operates. Should the reader wish to delve further into the operation of a specific IC, we suggest that he study the appropriate data sheet for that IC.

The section begins with an overview of the Sol design. A block diagram analysis then provides the reader with an understanding of the relationship between the functional elements of the Sol-PC. This analysis sets the stage for detailed descriptions of the circuitry that makes up these elements. The section concludes with a block diagram analysis and circuit description of the keyboard.

8.2 OVERVIEW

The Sol Terminal Computer T.M., as the name implies, is both a terminal and computer. It is designed around the S-100 bus structure used in other 8080 microprocessor-based computers and incorporates all of the circuitry needed to perform either function. In essence, Sol combines a central processor unit (CPU) with several S-100 peripheral modules--memory, keyboard input interface (including the keyboard), video display output interface plus audio cassette tape, parallel, and serial input/output (I/O) interfaces. Sol-20 also includes a five-slot backplane board for adding other memory and I/O modules that are compatible with the S-100 bus.

An 8080 microprocessor (the CPU) is the "brain" of the Sol. It controls the functions performed by the other system components, obtains (fetches) instructions stored in memory (the program), accepts (inputs) data, manipulates (processes) data according to the instructions and communicates (outputs) the results to the outside world through an output port. (For information on 8080 operation, refer to the "Intel® 8080 Microcomputer Systems User's Manual.")

As shown in the Sol Simplified Block Diagram on Page X-24 in Section X, data and control signals travel between the CPU and the rest of the Sol over three buses: 1) a 16-line Address Bus, 2) an eight-line Bidirectional Data Bus, and 3) a 28-line Control Bus which is interfaced to the CPU with support logic circuitry. (Note that the use of a bidirectional data bus permits eight lines to do the work of 16, eight input and eight output.) These three buses account for the bulk of the S-100 Bus which connects the Sol to expansion memory and I/O modules.

In the Sol-20, the S-100 Bus structure takes the form of a five-slot backplane board. It consists of a printed circuit board with 100 lines (50 on each side) and five edge connectors on which like-numbered pins are connected from one connector to another. Functionally, the Sol version of the S-100 Bus is comprised of:

1. Sixteen output address lines from the CPU which are input to all external memory and I/O circuitry. (Direct memory access (DMA) devices must generate addresses on these lines for DMA transfers.)

2. Eight data input/output lines that transfer data between external memory and I/O devices and the CPU or DMA devices. (These eight lines are paralleled with eight other bus lines.)

3. Eight status output lines from the CPU support logic: Memory and I/O devices use status signals to obtain information concerning the nature of the CPU cycle. (DMA devices must generate these signals for DMA transfers.)

4. Nine processor command and control lines: Six of these are output signals from the CPU support logic; three of them are input signals to the CPU support logic from memory and I/O devices. (In a DMA transfer, the DMA device assumes control of these lines.)

5. Five disable lines: Four of these are supplied by a DMA device to disable the tri-state drivers on the CPU outputs during DMA transfers. The fifth is a derivative of the DBIN output from the CPU, and it is used to disable any memory addressed in Page Ø. Use of this disable is optional with a jumper.

6. Two input lines to the CPU support logic which are used for requesting a wait period. One is used by memory and I/O devices and the other by external devices.

7. Six power supply lines which supply power to expansion modules.

8. Three clock lines.

9. Four special purpose signal lines.

10. Thirty-one unused lines.

Definitions for each S-100 Bus line, as used in the Sol, are provided on Pages AVII-3 through AVII-6 in Appendix VII.

In addition to the S-100 Bus structure, Sol also uses an eight-line keyboard input port, an eight-line parallel input port, an eight-line parallel output port, an eight-line sense switch logic input port, and a unidirectional eight-line internal data bus.

The use of a unidirectional (input) data bus accommodates Sol's internal low-drive memory and I/O devices that do not meet the heavy drive requirement of the bidirectional data bus. The low-drive requirement of the internal bus also allows using the tri-state capabilities of the UART's (Universal Asynchronous Receiver/Transmitter) in the serial and audio cassette I/O circuits without additional drivers.

All CPU data and address lines are buffered through tri-state drivers to support a larger array of memory and I/O devices than would otherwise be possible with the 8080 output drive capability. Data input to the CPU is selected by a four-input multiplexer from the Keyboard Port, Parallel Port, Bidirectional Data Bus and Internal Data Bus. The Internal Data Bus is the source of all data input to the CPU from Sol's internal memory, the serial interface and the cassette interface. The Bidirectional Data Bus is the source of all data fed to memory and I/O, both internal and external. It is also the source of data input to the CPU from eight internal sense switches as well as from external memory and I/O.

8.3 BLOCK DIAGRAM ANALYSIS, Sol-PC

8.3.1 Functional Elements And Their Relationships

As can be seen in the Sol block diagram on Page X-24 in Section X, timing signals for Sol are derived from a crystal controlled oscillator that produces a "dot clock" frequency of 14.31818 MHz. (This frequency, four times that of the NTSC color burst, provides compatibility with color graphics devices.) The dot clock is applied directly to the Video Display Generator circuit and divided in the Clock Generator to provide Ø1, Ø2 and CLOCK. CLOCK synchronizes all control inputs to the 8080; Ø1 and Ø2 are the nonoverlapping, two phase clocks required by the 8080.

Memory internal to the Sol is divided between 2K of ROM (Read Only Memory), 1K of System RAM (Random Access Read/Write Memory) and 1K of Display RAM. The ROM permanently stores the instructions that direct the CPU's activities. (To enhance Sol's versatility, this particular memory is on a plug-in "personality module". Thus, Sol can be easily optimized for a particular application by plugging in a personality module that contains a software control program designed for the task. The CONSOL and SOLOS programs, which are described in Section IX, are examples of such personality modules.) Display RAM stores data for display on a video monitor, and the System RAM provides temporary storage for programs and data. All memories are addressed on the Address Bus (ADRØ-15) and, except for the Display RAM, input data to the CPU on the Internal Data Bus (INTØ-7). Data entry into both RAM's is done on the Bidirectional Data Bus (DIOØ-7).

As can be seen, Sol's internal memory consists of four contiguous 1024-byte pages. There are two pages (CØ and C4, hexadecimal or hex) of ROM, with Page CØ at hex addresses CØØØ through C3FF and Page C4 at hex addresses C4ØØ through C7FF. System RAM (Page C8) is at hex addresses C8ØØ through CBFF, and Display RAM (Page CC) is at hex addresses CCØØ through CFFF.

The six high order bits of the address are decoded in the Address Page and I/O Port Decoder to supply the required four memory page selection signals. The I/O Port Decoder portion of this circuit decodes the eight high order address bits to provide outputs that control Data Input Multiplexer switching, Data Bus Driver enablement and I/O port selection.

The video display section consists of the Video Display Generator and Display RAM. The RAM is a two-port memory, with the CPU having the higher priority. Screen refresh circuitry in the Video Display Generator controls the second port to call up data as needed for conversion by a character generator ROM into video output signals. Other circuitry generates horizontal and vertical sync and blanking signals as well as cursor and video polarity options.

A 1200 Hz signal, extracted from dot clock by a divider in the Video Display Generator, drives the Baud Rate Generator. This generator supplies the receive and transmit clocks for the serial data interface (SDI/UART) and provides all frequencies required for Baud rates between 75 and 9600. It also supplies clock signals to the Cassette Data Interface (CDI).

A UART controls data flow through the Serial Data Interface (SDI/UART) and provides for compatibility between the Sol and a data communications system, be it RS-232 standard or a 20 ma current loop device. In the transmit mode, parallel data on the Bidirectional Data Bus is converted into serial form for transmission. Received serial data is converted in the receive mode into parallel form for entry into the CPU on the Internal Data Bus. SDI/UART status is also reported to the CPU on the Internal Data Bus. The SDI/UART channel is enabled by the port strobe from the Address Page and I/O Port Decoder.

Circuitry within the CDI derives timing signals from clocks supplied by the Baud Rate Generator. The Cassette Data UART functions to 1) convert parallel data on the Bidirectional Data Bus into serial audio signals for recording on cassette tape, and 2) convert serial audio signals from a cassette recorder into parallel data for entry into the CPU from the Internal Data Bus. Note that Cassette Data UART status is also reported to the CPU on the Internal Data Bus. Again, a UART performs the necessary parallel-to-serial and serial-to-parallel conversions. Other CDI circuitry performs the needed digital-to-audio and audio-to-digital conversions and provides the signals that allow motor control for two recorders. As with the SDI/UART, the Cassette Data UART is enabled by a port strobe from the Address Page and I/O Port Decoder.

Output data from the CPU that is channeled through the Parallel Port (PP) is latched from the Bidirectional Data Bus by the parallel strobe from the Address Page and I/O Port Decoder. This data is made available at P2, the PP connector. Parallel input data (PIDØ-7) on P2, however, is fed directly to the Data Input Multiplexer for entry into the CPU.

As can be seen, keyboard data (KBDØ-7) from J3 is also fed directly to the Data Input Multiplexer. The keyboard data ready flag, though, is input to the CPU on the internal data bus.

The remaining internal source of data input to the CPU is the Sense Switch Logic, with the data being input on the Bidirectional Data Bus. This is an eight-switch Dual Inline Package (DIP) array that lets the CPU read an eight-bit word when it issues the sense switch strobe via the Address Page and I/O Port Decoder. The sense switch data source is available to interact with the user's software.

CPU Support Logic accepts six control outputs from the CPU, status information from the CPU's data bus and control signals from the Control Bus. It controls traffic on the data buses by generating signals to 1) select the type of internal or external device (memory or I/O) that will have bus access and 2) assure that the device properly transfers data with the CPU.

8.3.2 Typical System Operation

Basic Sol system operation is as follows: The CPU fetches an instruction and in accordance with that instruction issues an activity command on the Control Bus, outputs a binary code on the Address Bus to identify the memory location or I/O device that is to be involved in the activity, sends or receives data on the data bus with the selected memory location or I/O device, and upon completion of the activity issues the next activity command.

Let's now look at some typical operating sequences.

Keyboard Data Entry and Display. Assume the "A" and SHIFT keys on the keyboard are pressed. The keyboard circuitry converts the key closures into the 7-bit ASCII (American Standard Code for Information Interchange) code for an "A" (1ØØØØØ1) and sends a keyboard-data-ready status signal to the CPU on the Internal Data Bus. The monitor program in ROM repetitively "looks" for the status signal. When it finds this signal the program enters its keyboard routine and enables the transfer by switching the Data Input Multiplexer to the keyboard bus via the Address Page and I/O Port Decoder.

Following program instructions, the CPU addresses the Display RAM on the Address Bus to determine where the next character is to appear on the screen. It then stores the ASCII code for the "A" at the appropriate location in the Display RAM and adds one to the cursor position in readiness for the next character. (Addressing is done over the Address Bus; cursor position and the "A" enter the Display RAM on the Bidirectional Data Bus.) The CPU is now finished with the transfer, and will issue the next activity command.

When the refresh control circuitry calls up (addresses) the "A" from the Display RAM, the character generator ROM decodes the ASCII-coded "A" that is input from the Display RAM and generates the "A" dot pattern (see Figure 8-5 and 6) in parallel form. The ROM output is serialized into a video signal and combined with a composite sync signal to provide an Electronic Industries Association (EIA) composite video signal for display on an external video monitor.

SDI/UART Transfer and Display. A data transfer through the SDI/UART is similar to a keyboard entry, but data can be transferred in either direction.

Assume the SDI/UART wants to transfer an "A" from a modem to the CPU for display on a video monitor. The ASCII code for the "A", received in serial form from the modem on the serial data input of the SDI connector (J1), is fed to the SDI/UART. In the receiver section of the UART the serial data is converted into parallel form and placed in the UART's output register. The UART also sends a "received data ready" status signal to the CPU on the Internal Data Bus. When the program in ROM checks and finds the status signal, the program enters the SDI routine, and enables the transfer by switching the Data Input Multiplexer to the Internal Data Bus. The "A" enters the CPU on the Internal Data Bus and is sent to the Display RAM on the Bidirectional Data Bus. Operations involved in displaying the "A" are identical to a keyboard entry.

Now assume the CPU wants to send an "A" to the SDI/UART for transmission. The CPU, under program control, sends the SDI/UART status input port strobe via the Address Page and I/O Port Decoder to the UART. In turn, the UART responds with its status on the Internal Data Bus. Assuming the UART is ready to transmit, the CPU places the ASCII code for the "A" on the Bidirectional Data Bus and sends the SDI/UART data output port strobe which loads the Bidirectional Data Bus content into the UART's transmitter section. The "A" is serialized by the UART and sent out the transmitted data pin of J1.

8.4 POWER SUPPLY CIRCUIT DESCRIPTION

Refer to the Sol-REG and Sol-10 or Sol-20 Power Supply Schematics in Section X, Pages X-12, 13 and 14.

The Sol power supply consists of the Sol-REG regulator and either the Sol-10 or Sol-20 power supply components. An 8 V dc unregulated supply in the Sol-20 is the only difference between the two. We will, therefore, describe the complete Sol-10 supply followed by the unregulated 8 V dc supply in the Sol-20.

Fused primary power is applied through S5 to Tl (T2 in the Sol-20). FWBl, a full-wave bridge rectifier, is connected across the 8-volt secondary (green leads). The rectified output is filtered by C8 and applied to the collector of Ql. Ql, a pass transistor, is driven by Q2, with the two connected as a Darlington pair. The output of Ql is connected to Rl which serves as an overload current sensor.

An overload current (approximately 4 amps) increases the voltage drop across Rl. The difference is amplified in one-half of U2 (an operational amplifier) and the output on pin 7 turns Q3 on. Q3 in turn "steals" current from Ql-Q2 and diverts current from the output on pin 1 of U2. This in effect turns the supply off to reduce the current and voltage. Note that the circuit is not a constant current regulator since the current is "folded back" by R6 and R8. The current is reduced to about 1 amp as the output voltage falls to zero.

Divider network Rll and Rl2, which is returned to -12 volts, senses changes in the output voltage. If the output voltage is 5 volts, the input on pin 2 of U2 is at zero volts. U2 provides a positive output on pin 1 if pin 3 is more positive than pin 2 and a negative output for the opposite condition.

When the output voltage falls below 5 volts, pin 2 of U2 goes more negative than pin 3. This means pin 1 of U2 goes positive to supply more current to the base of Ql. The resulting increase in current to the load causes the output voltage to rise until it stabilizes at 5 volts. Should the output voltage rise above 5 volts, the circuit operates in a reverse manner to lower the voltage.

Protection against a serious over-voltage condition (more than 6 volts) is provided by SCRl, Dl, R2, Rl3, Rl4 and C8. Zener diode, (Dl), with a 5.1 zener voltage, is connected in series with Rl3 and R2. When the output voltage exceeds about 6 volts, the resulting voltage drop across R2 triggers SCRl to short the foldback current to ground. Since the overload current circuit is also working, the current through SCRl is about 1 amp. Once the current is removed, this circuit restores itself to its normal condition; that is, SCRl turns off. Rl3, Rl4 and C8 serve to slightly desensitize the circuit so that it will not respond to small transient voltage spikes.

Bridge rectifier FWB2, connected across the other Tl secondary, supplies +12 and -12 V dc. The positive output of FWB2 is filtered by C5 and regulated by IC regulator Ul. The negative output is filtered by C4 and regulated by U3. Shunt diodes D3 and D4 protect Ul and U3 against discharge of C6 and C7 when power is turned off. (Note that should the -12 volt supply short to ground, the +5 volt supply turns off by the action of U2.

Unregulated -16 and +16 V dc, at 1 amp, from the filtered outputs of FWB2 are made available on terminals X6 and X5. These are not used in the Sol-10, but they are supplied to the backplane board in the Sol-20 to drive S-100 Bus modules.

In the case of the Sol-20, the power transformer (T2) has an additional 8-volt secondary winding and a third bridge rectifier (FWB3) to supply +8 V dc at 8 amps. The output of FWB3 is filtered by C9 and controlled by bleeder resistor R13. Again, this voltage is supplied to the backplane board in the Sol-20.

Sol-20 also includes a cooling fan powered by the AC line voltage.

8.5 Sol-PC CIRCUIT DESCRIPTIONS

8.5.1 CPU and Bus

Refer to the CPU and Bus Schematic in Section X, Page X-15.

A crystal, two inverter sections in U92 and four D flip-flops (U90) and associated logic make up the Clock Generator.

The two U92 sections function as a free-running oscillator that runs at the crystal frequency of 14.31818 MHz. R133 and R134 drive these two sections of U92 into their linear regions, and C61 and 64 provide the required feedback loop through the crystal. U77, a permanently enabled tri-state non-inverting buffer/amplifier, furnishes a high drive capability.

This fundamental clock frequency is fed directly to the Video Display Generator and to the clock inputs of U90. U90 is a four-stage register connected as a ring counter that is reset to zero when power is applied to the Sol. This reset is accomplished with D8, R104 and C39.

The bits contained in the ring counter shift one to the right with every positive-going clock transition, but the output of the last stage is inverted or "flipped" before being fed back to the input. In a simple four-stage "flip-tail" ring counter, the contents would progress from left to right as follows: 1ØØØ, 11ØØ, 111Ø, 1111, Ø111, ØØ11, ØØØ1, ØØØØ--on the first through eighth clocks respectively. The hypothetical counter would go through eight states, dividing the clock by eight.

The Sol counter, however, is a modified flip-tail ring counter that can be configured to divide by one of three divisors--5, 6 or 7. This is made possible by using a two-input NAND gate (U91) in the feedback path and three jumper options (no jumper, D-to-C and D-to-E) to alter the feedback path. Let's see how it works.

Sol is normally configured with the D-to-E jumper installed to meet the clock requirements of the 8080A CPU. With this jumper installed, the outputs of the third and fourth U90 stages are applied to pins 9 and 10 of U91. Assuming U90 is reset to zero, pin 8 of U91 is high, and on the first clock pulse the counter contents change to 1ØØØ. (Refer to 2.045 MHz Clocks portion of Figure 8-1 on Page VIII-11.) Pin 8 of U91 cannot change until the fourth state (1111), at which time it goes to zero. On the fifth clock pulse the counter changes to Ø111. Again, pin 8 of U91 cannot change from zero until one of its inputs changes. As shown in Figure 8-1, the third U90 stage (C) changes on the seventh clock. The counter now stands at ØØØ1, and on the eighth clock the counter flips to 1ØØØ and the count cycle repeats. The pattern is thus 1ØØØ, 11ØØ, 111Ø, 1111, Ø111, ØØ11, ØØØ1. U90 consequently goes through seven states. We have a 3.5-stage counter that divides DOT CLOCK by seven to supply a 2.045 MHz output.

With no jumper installed, pin 10 of U91 is pulled high by R105, and U91 operates as a simple inverter for feeding back the output of the third U90 stage. In effect we have a three-stage counter that operates in a similar manner to that described in the preceding paragraph. It goes through six states (1ØØ, 11Ø, 111, Ø11, ØØ1, ØØØ) to divide DOT CLOCK by six which produces a 2.386 MHz output. The timing for this option is also shown in Figure 8-1.

Let's now put the D-to-C jumper in. The feedback in this case is the NAND combination of the outputs from the second (B) and third (C) U90 stages. This gives us a 2.5-stage counter that divides DOT CLOCK by five. As can be determined from the 2.863 MHz portion of Figure 8-1, the counter has five states with this option, and the count pattern is: 1ØØ, 11Ø, 111, Ø11, ØØ1.

Outputs from U90 are applied to the logic comprised of the remaining three sections in U91. This logic and the A-to-B jumper option permits extracting clock pulses of varying widths and relationships to each other from various points within the counter. We extract two clock signals: Ø1 on pin 6 of U91 and Ø2 on pin 11 of U91. (The ability to select the frequency and pulse width for Ø1 and Ø2 permits the use of either the 8080A, 8080A-1 or 8080A-2 CPU for U105. The "A" version is the slowest speed unit, the "A-2" has an intermediate speed, and the "A-1" is the fastest.) Let's now see how the pulse width of Ø1 and Ø2 are determined.

Ø1 on pin 6 of NAND gate U91 is low only when its two inputs are high, and this happens only when there is a 1 in the second and fourth stages of U90. This occurs during the time between the fourth and sixth fundamental clocks for 2.04 MHz operation--the fourth and fifth clocks for 2.38 MHz and 2.86 MHz. Keeping in mind that the fundamental clock period is 70 nsec, it is readily seen that the low frequency pulse train on pin 6 of U91 has a pulse width of 140 nsec and the two higher frequency pulse trains have a pulse width of 70 nsec. (Refer to Figure 8-1 on Page VIII-11.)

The A-to-B jumper is installed when the 8080A or 8080A-1 CPU is used in the Sol. Note that the output (Ø2) on pin 11 of NAND gate U91 is low only when the output on pin 3 of NOR gate U91 is high. (This section in U91 is actually a two-input NAND gate which is functionally the same as a two-input NOR gate.) Pin 3 of U91, with the A-to-B jumper in, is high when either the second (B) or third (C) U90 stage is at zero. As shown in Figure 8-1, this occurs between the sixth and tenth DOT CLOCKS, or 280 nsec (4 x 70 nsec), for 2.04 MHz operation. For 2.863 MHz, it occurs between the fifth and eighth DOT CLOCKS for 210 nsec. The section of NAND gate U91 with its output on pin 11 inverts the output on pin 3 of U91 and introduces a slight delay to insure there is no overlap between Ø1 and Ø2.

With the A-to-B jumper out, pin 11 of U91 is low only when the second stage (B) of U90 is at zero. At 2.386 MHz, this occurs between the fifth and eighth DOT CLOCKS for 210 nsec. This configuration is used for the 8080A-2 CPU.

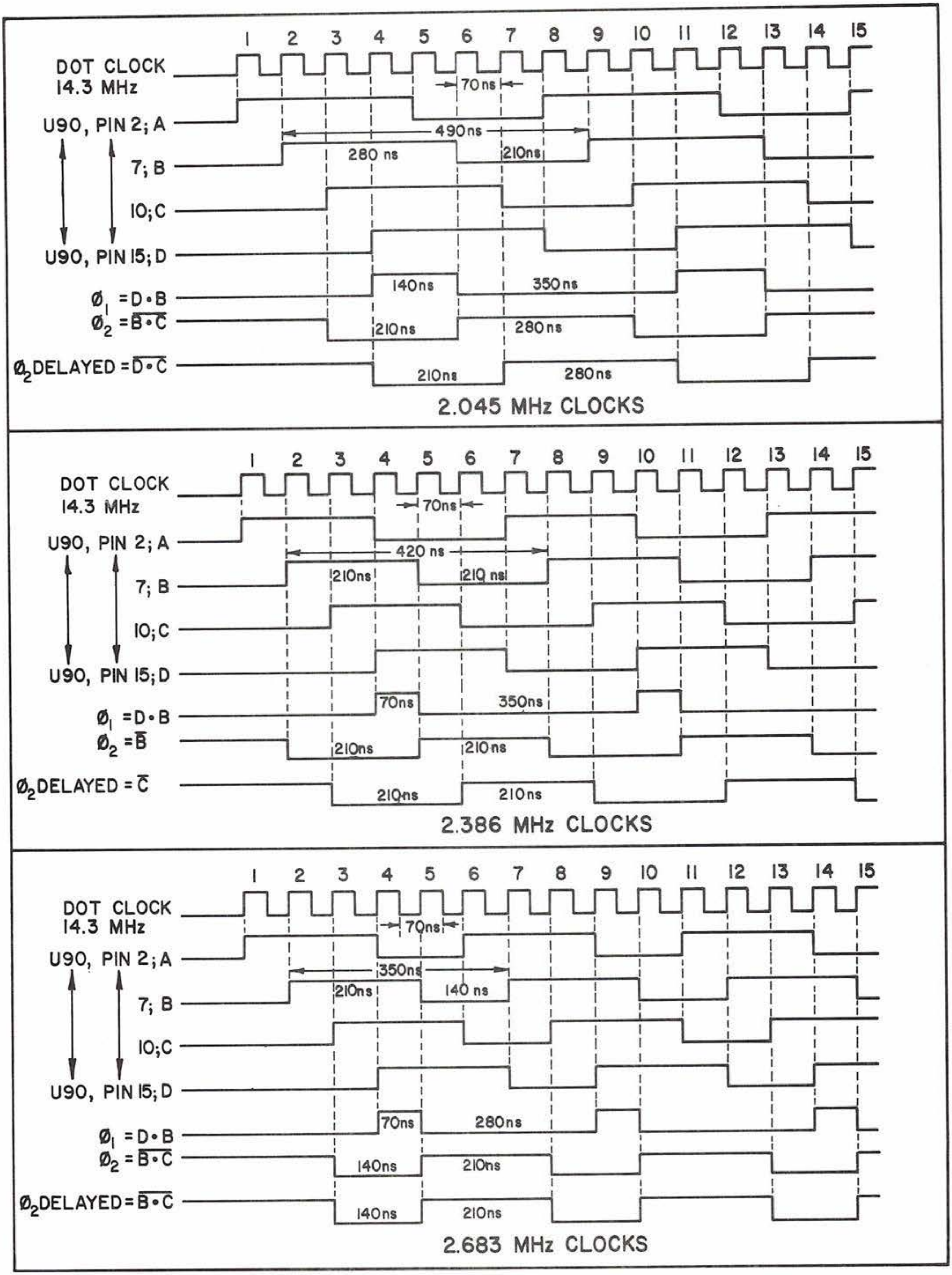
In summary, we have two non-overlapping pulse trains which represent the Ø1 and Ø2 clocks required by the 8080 CPU, and the pulse widths of these two clocks vary with frequency as follows:

FREQUENCY	Ø1 PULSE WIDTH	Ø2 PULSE WIDTH	CPU
2.045 MHz	140 nsec	280 nsec	8080A
2.386 MHz	70 nsec	210 nsec	8080A-2
2.863 MHz	70 nsec	210 nsec	8080A-1

Ø1 and Ø2 are applied to S-100 Bus pins 25 and 24 respectively through inverters (U92) and bus drivers (U77). They are also capacitively coupled to pins 2 and 4 respectively of driver U104, the phase clock conditioner.

An additional clock, called CLOCK, is taken from pin 8 of NAND gate U91. It occurs 70 nsec after Ø2. It is used on the Sol-PC and is also made available on S-100 Bus pin 49 as a general 2.04, 2.38 or 2.86 MHz clock signal.

Three J-K flip-flops (U63 and 64) are used to synchronize the READY, RESET and HOLD inputs to the CPU. All three are connected as D-type flip-flops so that their outputs follow their inputs on the low-to-high transition of the clock. The READY flip-flop input on pins 2 and 3 of one section in U63 is either PRDY or XRDY from the S-100 Bus; these are normally pulled high by R34 and R12 respectively. S-100 Bus signal $\overline{\text{PRESET}}$, which is normally pulled high by R55, inputs the RESET flip-flop, the other section of U63. The HOLD flip-flop (U64) input is $\overline{\text{P HOLD}}$, normally pulled high by R56, from the S-100 Bus. Pull up resistors R51, R50 and R53 insure that the high states of these three flip-flops are adequate for the CPU.



2.045 MHz CLOCKS

2.386 MHz CLOCKS

2.683 MHz CLOCKS

Figure 8-1. CLOCK GENERATOR TIMING

Diode D7, C15 and R18 make up the POC (power on clear) circuit. When power is applied, C15 starts to charge slowly until it reaches the threshold on pin 6 of U46, a Schmitt trigger. (By this time the logic and 5 volt supply have stabilized.) When the threshold is reached, pin 1 of U46 suddenly goes low. The resulting output on pin 8 of inverter U92 is initially low and then rapidly goes high. This signal is passed through a section of U77, a permanently enabled noninverting tri-state driver, as $\overline{\text{POC}}$ to S-100 Bus pin 99. It is also inverted in a section of U45 to become POC.

The output on pin 8 of U92 is also connected to pin 15 of U63. Thus, pin 9 (RESET) of U63 is high to start the CPU in the reset condition when the Sol is initially turned on.

When $\overline{\text{POC}}$ goes high, the RESET flip-flop section of U63 is free to clock. Assuming $\overline{\text{PRESET}}$ is not active, it will change state on the first CLOCK transition. The resulting high on pins 10 and 5 of U63 cause pin 7 (READY) of U63 to go low to place the CPU in the not ready or wait state. This state is subsequently removed on the CLOCK transition following the transition which removed the low from pin 5 of U63. This helps prevent the CPU from starting in a crash condition.

The HOLD flip-flop (U64), however, is not affected by the POC circuit, and was clocked to a low on pin 7 well before the RESET and READY signals became active.

Operation of the POC circuit can also be initiated, without turning the power off, by a keyboard restart signal on pin 13 of J3 or by closing S1-1 if the N-P jumper is in. In either case, C15 is discharged through R58 and then allowed to recharge after $\overline{\text{KBD RESTART}}$ is removed or S1-1 is opened.

$\overline{\text{POC}}$ also resets all stages of D flip-flop U76 (the phantom start-up circuit) to zero. On initial start-up, the CPU performs four fetch machine cycles (refer to Intel® 8080 Microcomputer Systems User's Manual) in accordance with program instructions. For each fetch, the CPU outputs a DBIN on pin 17. U76, connected as a four-stage shift register, is clocked by the inverted DBIN signal on pin 3 of NOR gate U46. Thus, $\overline{\text{PHANTOM}}$, on S-100 Bus pin 67, is active low (assuming the F-to-G jumper is in) for the first four fetches or machine cycles. After the fourth DBIN, $\overline{\text{PHANTOM}}$ goes high. $\overline{\text{PHANTOM}}$ is used to 1) disable any memory addressed in Page Ø that has Processor Technology's exclusive "Phantom Disable" feature and 2) cause the Sol program memory (ROM), which normally responds to Page CØ (hex) to respond to Page ØØ (hex). The second function is discussed in Paragraph 8.5.2.

The inverted DBIN on pin 3 of U46 is also applied to pin 12 of NOR gate U46 and inverted to appear as PDBIN on S-100 Bus pin 78. This section of U46 also allows $\overline{\text{DIGI}}$ (bus pin 57) to override DBIN. ($\overline{\text{DIGI}}$ is used when an external DMA device replaces the CPU in terms of writing into and reading from memory.) The other CPU control signals (SYNC, INTE, HLDA, WR and WAIT) are also fed to the S-100 Bus pins as indicated. These, as well as DBIN or $\overline{\text{DIGI}}$, are placed on the bus through tri-state drivers which are enabled by C/C DSB on S-100 Bus pin 19. Note that this signal is normally pulled high by R20.

The data lines of the CPU (DØ-7) are bidirectional and are used for several functions. One of these is to output status at the start of each cycle which is marked by the SYNC output of the CPU. Status on DØ-7 is latched in U93 and U106 (each of which contains four D flip-flops) when pin 8 of inverter U45 goes high. Status information, as identified on the schematic, is then buffered through tri-state drivers U94 and U107 to the S-100 Bus. The status latch strobe on pin 8 of U45 is extracted in the middle of the SYNC pulse by gating PSYNC and $\overline{\text{Ø2}}$ in NAND gate U44. $\overline{\text{STAT DSB}}$ on S-100 Bus pin 18 is used to disable the U94 and U107 buffers when a DMA device or another processor assumes control of the S-100 Bus.

A second function of DØ-7 is to output data from the CPU to the Bidirectional Data Bus. Data out of the CPU is placed on this bus through tri-state drivers (U80 and U81). Note that these drivers are normally enabled unless this bus is in the input mode or an external device has control of the bus. In the latter case, $\overline{\text{DO DSB}}$ on S-100 Bus pin 23 would be pulled low to make pin 8 of NOR gate U48 high. In the input mode pin 8 of U48 is high because $\overline{\text{OUT DSB}}$ is low. This signal is generated by decoding $\overline{\text{PAGE CC}}$, MEM SEL, $\overline{\text{PORT IN FC}}$, PORT IN FD, INT SEL to produce MPX ADR A and MPX ADR B on pins 3 and 11 respectively of two NOR gates in U48. MPX ADR A and MPX ADR B are decoded with $\overline{\text{DBIN}}$ on pin 5 of NAND gate U47.

The DØ-7 bus lines are also used to input data to the CPU. Data input to the CPU is multiplexed from four data buses with four 4-to-1 line multiplexers (U65, 66, 70 and 79). These four buses are the: 1) Keyboard Data Bus, KDBØ-7, 2) Parallel Input Data Bus, PIDØ-7, 3) Internal Data Bus, INTØ-7, and 4) Bidirectional Data Bus, DIOØ-7.

These data multiplexers are tri-state devices, with their outputs pulled up by R107 through R114 to a level that satisfies the input requirements of the CPU. Their outputs are active only when both their E1 and E2 (pins 1 and 15) are low. As can be seen, this occurs only when $\overline{\text{DBIN}}$ on pin 3 of NOR gate U46 is low; that is, when the DBIN output of the CPU is active to indicate its data bus is in the input mode.

Input selection to the multiplexers is done with the A and B inputs to U65, 66, 78 and 79. These two inputs are driven by MPX ADR A on pin 3 of NOR gate U48 and MPX ADR B on pin 11 of NOR gate U48. There are four possible states for the combination of MPX ADR A and B, and their relation to input selection is as follows:

1. If both are active (high), the multiplexers select the Bidirectional Data Bus.

2. When the keyboard is called up by the CPU, only $\overline{\text{PORT IN FC}}$ is active (low) to make MPX ADR A low. This selects the Keyboard Data Bus.

3. When the parallel port is called up by the CPU, only $\overline{\text{PORT IN FD}}$ is active (low) to make MPX ADR B low. This selects the Parallel Input Data Bus.

4. When the CPU selects any I/O port that uses the Internal Data Bus, only $\overline{\text{INT SEL}}$ (pin 2 of U47 and U61) is active. Thus, both MPX ADR A and B are low to select the Internal Data Bus.

Two other conditions, defined by $\overline{\text{PAGE CC}}$ on pin 2 and MEM SEL on pin 1 of NAND gate U44, are possible. When any of the four memory pages in the Sol are accessed, MEM SEL goes high and an inversion in U44 ($\overline{\text{PAGE CC}}$ is normally high) appears as a low MPX ADR A and B to select the Internal Data Bus. Should Page CC (the Display RAM) be addressed, $\overline{\text{PAGE CC}}$ also goes active (low) to override MEM SEL. MPX ADR A and B are consequently high to select the Bidirectional Data Bus. These two conditions are required since the ROM and System RAM use the Internal Data Bus and the Display RAM uses the Bidirectional Data Bus.

The address outputs of the CPU (AØ-15) are placed on the Address Bus via tri-state drivers (U67, 68 and 81). These drivers are normally enabled since pin 3 of inverter U49 is pulled high by R36. $\overline{\text{ADD DSB}}$ on S-100 Bus pin 22 is used to disable the address drivers when a DMA device or another CPU takes over the bus.

A 5.1 volt zener diode, D11, and a divider network composed of R130, 131 and 132 derive -5 V dc from the -12 V dc supply for use by the CPU. Diode D12 and the same divider supply -12 V dc to pin 3 of U104, the phase clock conditioner.

8.5.2 Memory and Decoder

Refer to the Memory and Decoder Schematic in Section X, Page X-16.

The System RAM consists of eight 1K by 1 bit static memory chips, U3 through U10, and it is assigned addresses C8ØØ-CBFF (hex). When the CPU wants to write data into memory, it addresses the System RAM on ADRØ-15. ADRØ-4 select the row inside the RAM chips, ADR5-9 select the column, and ADR1Ø-15 select the page (in this case Page C8, hex). Page selection enables the eight RAM chips on pin 13. For a read operation, MWRITE on S-100 Bus pin 68 is low, and the resulting high on pin 3 (WE) of the RAM chips keeps them in the read mode. Thus, data on the Bidirectional Data Bus is read into the RAM's on their DI (pin 11) inputs. MWRITE is high, however, during the time the CPU wants to write data into memory. In this case, pin 3 of the RAM's is low to enable them to accept data from the Bidirectional Data Bus.

The ROM is also addressed on ADRØ-15 as is the System RAM. Since there can be two pages, however, two enable lines (one for Page CØ, hex, and the other for C4, hex) are provided. The CØ and C4 enables are connected to pins A6 and A5 respectively of J5, the Personality Module connector. Unlike the RAM, the ROM can only read data into the CPU, so the previously discussed MWRITE signal is not needed. Data out of the ROM is output on the Internal Data Bus on pins A3, A4 and B5-1Ø of J5.

ADR1Ø-15 are input to the Address Page and Port Decoder (U34, 35, 36 and their associated logic). U34 (Address Page), U35 (Output Port) and U36 (Input Port) are 3-to-8 line decoders which have three enable inputs (G1, G2A and G2B). G1 must be high and both G2A and B must be low in order to obtain an active output.

Let's look at the Address Page Decoder, U34, first. It must be able to decode four pages: CØ and C4 (ROM), C8 (System RAM) and CC (Display RAM). (Note that these are the hexadecimal digits of the six high order address bits, ADR1Ø-15).

The high order four bits (ADR12-15) must be 11ØØ (C, hex) in all cases by virtue of the U22 exclusive OR logic. If they are not, the G1 enable on U34 is low to disable that decoder. Bits ADR1Ø and 11 (The A and B inputs to U34) are the high order bits of the second hexadecimal digit which must be ØØ (Ø, hex), Ø1 (4, hex), 1Ø (8, hex) or 11 (12, hex) if U34 is to have an active output. For CØ, pin 11 of U34 is active (low); for C4, pin 1Ø is active; for C8 pin 9 is active; and for CC pin 7 is active. These outputs are applied to the appropriate memories and also provide the MEM SEL signal on pin 6 of one section in U23. (This section is actually a 4-input NAND gate which is functionally the same as a 4-input NOR gate.)

Note that the U22 logic input with ADR14 and 15 is also connected to PHANTOM. When this signal is active (low), the output on pins 3 and 11 will be low to disable U34 when ADR12-15 represent a C. If Page Ø is addressed, however, pins 3 and 11 of U22 are high, and this, coupled with lows on ADR1Ø-13, are decoded by U34 as an active output on pin 11. The ROM will consequently respond to addresses in Page Ø and CØ (hex) as long as PHANTOM is active.

The other two enables on U34 (G2A and G2B) are connected to SINP and SOUT. These two status signals indicate an input or output operation during the CPU cycle. U34 is therefore disabled during these operations.

SINP and SOUT are also fed to pins 5 and 6 of NOR gate U53 which detects an input or output operation. Its output is inverted by U54 and applied to pin 9 of another U53 NOR gate. The other input (pin 8) to U53 is MEM SEL. So during a memory reference, input operation or output operation, pin 10 of U53 is active to enable the PRDY driver, U71. The low on pin 10 of U53 is also clocked by Ø2 as a high to pin 7 of U70, a J-$\overline{\text{K}}$ flip-flop that is connected as a D flip-flop. Note that the PSYNC ● $\overline{\text{Ø2}}$ signal on pin 5 of U70 forces U70 to set during the middle of PSYNC (refer to CPU and Bus discussion). U70 cannot clock until pin 5 is released, and this occurs simultaneously with the low-to-high transition of Ø2. PRDY is thus low immediately after pin 10 of U53 goes low and remains in that state from the middle of PSYNC to the first positive-going Ø2 after PSYNC. This is the time the CPU tests the status of the ready lines (PRDY and XRDY). If either is low, the CPU enters a WAIT state. U53, 70 and 71 thus guarantees that the CPU enters one WAIT state during cycles in which an input, output or memory reference is made.

U35 and 36, the Output and Input Port Decoders respectively, decode the higher order eight address bits (ADR8-15).

All Sol ports have a hexadecimal F (1111) in their high order four bits (ADR12-15 are 1's). The second hexadecimal digit is also never less than eight. This means that ADR11 is always 1 for a port address. These five address bits are thus NAND gated in U23 to provide one of the enables on U35 and 36. Note that the ADR14-15 combination is derived from the output on pins 3 and 11 of the U22 exclusive OR logic. This is permissible since no I/O operations are performed during the first four start-up cycles of the CPU.

The A, B, and C inputs to U35 and 36 (ADR8, 9 and 10 respectively) specify the second hexadecimal digit in the port address and are decoded to supply the indicated outputs. These outputs and their functions are defined in Table 8-1. U36 is enabled to decode when PDBIN and SINP are active; that is, during an input operation. U35 is enabled when SOUT and $\overline{\text{PWR}}$ are active; that is, during an output operation.

INT SEL on pin 8 of inverter U83 is the remaining signal generated by the Input Port Decoder circuit. This signal is active when either input port F8, F9, FA or FB is decoded by U36.

Both the address page and input/output decoders can be disabled by SINTA (S-100 Bus pin 96) when the AE-to-AC and AB-to-AD jumpers are installed. SINTA is active (high) when the CPU is responding to an interrupt. Should an external device issue addresses during this time, any memory response would interfere with the

Table 8-1. Port Decoder (U35 & U36) Outputs and Their Functions.

PORT DECODER OUTPUT	FUNCTION
$\overline{\text{PORT OUT FE}}$	Loads starting row address and first display line position information from Bidirectional Data Bus into Video Display scroll circuit.
$\overline{\text{PORT OUT FD}}$	Clocks data from Bidirectional Data Bus to output data pins of PP connector.
$\overline{\text{PORT OUT FB}}$	Loads data from Bidirectional Data Bus into Cassette Data UART.
$\overline{\text{PORT OUT FA}}$	Clocks PP and CDI control bits from Bidirectional Data Bus.
$\overline{\text{PORT OUT F9}}$	Loads data from Bidirectional Data Bus into SDI UART.
$\overline{\text{PORT OUT F8}}$	Clocks RTS (request to send) from bit 4 of Bidirectional Data Bus to pin 4 of SDI connector.
$\overline{\text{PORT IN FF}}$	Permits CPU to read data byte entered from Sense Switches.
$\overline{\text{PORT IN FE}}$	Places Video Display scroll timer and screen position status on bits Ø and 1 of Bidirectional Data Bus.
$\overline{\text{PORT IN FD}}$	Switches Data Input Multiplexer to input data pins of PP connector and resets PP at end of a transfer to ready it for another.
$\overline{\text{PORT IN FC}}$	Switches Data Input Multiplexer to Keyboard Data Bus.
$\overline{\text{PORT IN FB}}$	Strobes received data in CDI UART to Internal Data Bus.
$\overline{\text{PORT IN FA}}$	Places PP, keyboard and CDI UART status on Internal Data Bus.
$\overline{\text{PORT IN F9}}$	Strobes received data in SDI UART to Internal Data Bus.
$\overline{\text{PORT IN F8}}$	Places SDI UART status on Internal Data Bus.

interrupt operation. To prevent this, SINTA is inverted in U58 to 1) disable U34 on pin 6 and 2) force pin 8 of NAND gate U23 high to disable U35 and U36 on pin 5. (This feature is provided to enable future versions of Sol to operate with a vectored interrupt system.)

8.5.3 Input/Output

Refer to the Input/Output Schematic In Section X, Page X-17.

This section in the Sol has five functional circuits: 1) Parallel I/O Logic, 2) Sense Switch Logic, 3) Keyboard Flag Logic, 4) SDI/UART and 5) Baud Rate Generator.

The PP uses U95 and 96 (4-bit D-type registers) and their related logic. Data output to the PP connector (J2) is latched from DIOØ-7 by U95 and U96. Data is strobed into these registers on the leading edge of an inverted active $\overline{\text{PORT OUT FD}}$ signal on pin 4 of inverter U54. This strobe is also applied to pin 2 of U73 which functions as a J-K flip-flop that is clocked by $\overline{\text{Ø2}}$. When the $\overline{\text{Ø2}}$ goes from low to high 200 to 300 nsec after $\overline{\text{PORT OUT FD}}$, pin 7 of U73 goes low to become $\overline{\text{POL}}$ on pin 17 of J2. (This delay allows U95 and 96 to stabilize.) U73 is reset in the middle of the following PSYNC which means $\overline{\text{POL}}$ is active for the balance of the cycle.

The outputs of U95 and 96 are tri-state outputs that are enabled by a low on pin 2. In the absence of POE at pin 15 of J2, pin 2 of U95 and 96 are low by virtue of the output on pin 8 of inverter U55. Note that the input to U55 is normally pulled up through R63. The POE provision permits tri-stating an external bidirectional data bus.

As discussed in Paragraph 8.5.1, parallel input data on J2 is fed directly to the Data Input Multiplexer (see Page X-15). The strobe that indicates the presence of input data, $\overline{\text{PDR}}$ on pin 4 of J2, is applied to pins 2 and 3 of one section in U72, a J-$\overline{\text{K}}$ flip-flop which is connected as a D flip-flop. When $\overline{\text{PDR}}$ goes active (low), pin 7 of U72 will go high on the next low-to-high transition of Ø2 to toggle the following U72 stage. At this point pins 9 and 10 of the second section in U72 go high and low respectively. Pin 9 supplies PIAK on pin 5 of J2. When high, PIAK signals the external device that Sol has yet to complete acceptance of the data. The state of pin 10 of U72 is transmitted to INT1 of the Internal Data Bus through a U71 tri-state noninverting buffer. U71 is enabled only for the duration of $\overline{\text{PORT IN FA}}$ (auxiliary status). During the time U71 is enabled, the CPU reads the Internal Data Bus. A high INT1 indicates the parallel input data is not ready; a low indicates the data is ready.

The second U72 flip-flop is preset by $\overline{\text{PORT IN FD}}$ or POC. $\overline{\text{PORT IN FD}}$ is active to read data in from the PP; POC occurs only when Sol is restarted or power is turned on. Thus the PP is reset and ready for another transfer at the end of a transfer or when POC is active.

PXDR on pin 16 of J2 is supplied by the external device. It indicates the device is ready to receive data. $\overline{\text{PXDR}}$ is buffered to INT2 and will effect the transfer of data to the Internal Data Bus during the status input to the CPU. $\overline{\text{PXDR}}$ is analogous to the previously discussed PIAK signal.

Sense Switches S2-1 through 8 are driven by $\overline{\text{PORT IN FF}}$ when it is low. Thus, the DIO lines connected to closed switches are driven low, and those connected to open switches are pulled high.

U97 (a 4-bit D-type register) and one section of U52 (a J-$\overline{\text{K}}$ flip-flop connected as a D flip-flop) latch five bits of data on DIO3-7 when $\overline{\text{PORT OUT FA}}$ goes active. These bits, which supply the indicated outputs, control conditions in both the PP and CDI. With respect to the PP, PIE enables parallel input, and PUS selects the parallel device for the transfer. The data in these two latches remains until either a new word is read out or POC goes active.

Also during $\overline{\text{PORT OUT FA}}$, the keyboard flag is reported. $\overline{\text{KEYBOARD DATA READY}}$ on pin 3 of J3 is a low going pulse 1 to 10 usec in duration. It is applied to pin 13 of J-$\overline{\text{K}}$ flip-flop U70. Some time after pin 13 of U70 goes low, but before 500 nsec, U70 is set by Ø2 and pin 10 goes low. This low is buffered through U71 to INTØ to indicate the keyboard is ready to send data. Reset of U70 occurs with a POC or by $\overline{\text{PORT IN FC}}$. The latter occurs when data is accepted from the keyboard.

The other half of flip-flop U52, with its output on pin 6, latches one bit of status, DIO4, when $\overline{\text{PORT OUT F8}}$ is active. Its output is applied to pin 5 of one operational amplifier section in U56 to become the SRTS (request to send) signal on pin 4 of J1, the SDI connector.

The SDI/UART centers around a UART, U51. The UART transmission conditions (parity, word length and stop bits) are determined by the settings of S4-1 through 5. (Refer to Paragraphs 7.5.8 through 7.5.10 in Section VII for descriptions of the switch settings and their effect on transmission.

Data destined to leave Sol through the SDI/UART enters the UART on its TI1-6 inputs from the Bidirectional Data Bus when TBRL (pin 23) is low; that is, when $\overline{\text{PORT OUT F9}}$ goes active. Circuitry within the UART serializes the input data, which is in parallel form, and outputs it on pin 25 at a rate determined by the clock on pin 40. The binary states at pin 25 are low for a zero and high for a one. Assuming Sol is not in local operation ("off line"), the output on pin 25 of the UART is applied to pins 2 and 11 of J1 via two gates in U55 and the other half of U56.

Data that enters Sol through the SDI/UART on pins 3, 12 or 13 of J1 is input to the SDI UART on pin 20 by way of U38, an inverting level converter that converts data levels of up to ±25 volts to TTL levels. (Note that current loop data on pin 12 or 13 of J1 is first rectified before it is applied to U38.) The UART converts this serial data into parallel form and outputs it on RO1 through RO8 (pins 12 through 5 respectively) to the Internal Data Bus when ROD (pin 4) is low; that is, when $\overline{\text{PORT IN F9}}$ goes active.

The receive-transmit clock for the SDI UART is supplied by the Baud Rate Generator (U84, U85, U86 and their associated circuitry). U85 is a phase locked loop, U86 is a 7-stage binary counter and U84 is connected as a divide-by-11 counter. The 1200 Hz reference signal applied to pin 14 of U85 is supplied from the Video Display Generator. A phase comparator in U85 compares this signal to the output of a voltage controlled oscillator (VCO) in U85. By feeding an output from U86 (in this case the 1200 Hz output on pin 3) back to the compare input (pin 3) of U85, the circuit acts as a frequency multiplier. The output (pin 4) of U85 remains locked, therefore, to a multiple of its input on pin 14. In this case we have a 128X multiplier to generate 153.6 KHz which is counted down in U86. Since U86 is a 7-stage binary counter, the first stage output (pin 12) is 76.8 KHz (one-half of 153.6 KHz, the clock for U86), the second stage output (pin 11) is 38.4 KHz (one-fourth of 153.6 KHz), the third stage output (pin 9) is 19.2 KHz (one-eighth of 153.6 KHz), and so on to the seventh stage output (pin 3) which is 1.2 KHz (1/128 of 153.6 KHz).

With the exception of outputs on pins 12 and 9, the outputs of U86 are connected to S3, the Baud Rate Switch. The 19.2 KHz output on pin 9 is divided by 11 in U84 to supply 1745 Hz to S3-2. The 38.4 KHz on pin 12 can be connected to S3-8 instead of the 153.6 Hz clock by cutting the L-M connection and installing a jumper between K and M.

Let's now translate the frequencies input to S3 into Baud rates. The Baud rate of a UART is 1/16 of its clock rate. Thus, a 1200 Hz clock equates to a 75 Baud transmission rate, a 1745 Hz clock equates to a 109.1 (110) Baud rate, etc. It is now readily seen that the Baud rate available with S3-8 is 9600 assuming the L-M connection is made (153.6 KHz ÷ 16 = 9600). (The L-M connection is default wired on the Sol-PC; that is, there is a trace between L and M on the circuit board.) If the L-M trace is cut and a jumper is installed between K and M, the Baud rate with S3-8 is 4800 (76.8 KHz ÷ 16 = 4800).

We can thus select any one of eight clock frequencies for the SDI UART with S3, with the highest being determined by the K, L and M jumper arrangement. The selected clock is applied to both the receive and transmit clock inputs (pins 17 and 40 respectively) of the UART. This means, of course, that the UART always receives and transmits at the same Baud Rate.

Returning to the SDI UART, we see that its transmitter output on pin 25 is applied to pin 5 of U55, a two-input NAND gate that is functionally a NOR gate. It is normally enabled on pin 4 by pull-up resistor R44. A low on pin 5 represents a binary Ø; a high represents a binary 1. The inverted output on pin 6 of U55 is again inverted (assuming Sol is not operating in Local) by the following U55 NAND gate. One-half of operational amplifier U56, operating open loop, converts TTL levels to RS-232 levels (5 to 15 volts). Pin 3 of U56 is held at +2.5 V dc by the R47 and R48 divider network. When pin 2 is more negative than pin 3, the output on pin 1 of U56, which is fed to pin 2 of J1, is at approximately +10 volts. For the opposite condition, pin 2 of J1 is about -10 volts. Thus, U56 also inverts, and a high or low on pin 2 of J1 represent a binary 1 and Ø respectively.

Two conditions can override transmitted data: a keyboard break ($\overline{\text{BRK}}$) or local ($\overline{\text{KBD LOC}}$) command. For a break command, $\overline{\text{BRK}}$ on pin 4 of J3 and pin 4 of NOR gate U55, is low to hold pin 6 of U55 high for the duration of the $\overline{\text{BRK}}$ signal. This appears as a "space", or high level, on pin 2 of J1. (A space, or break, condition requires that the space level exist for a period longer than the normal length of a character.) In the case of a $\overline{\text{KBD LOC}}$ command from the keyboard, pins 1 and 13 of the other two U55 sections are low. Thus, data cannot be transmitted to pin 3 of NAND gate U55, and pin 11 of NOR gate U55 is held high to enable tri-state driver U37 at pin 15. Data on pin 6 of U55 is consequently looped back by way of U37 and R21 to pin 12 of U38. Data on pin 12 of U38 overrides any data arriving at pin 13 of U38. In local operation, therefore, data from pin 25 of the UART does not appear at pin 2 of J1, but it is looped back to the receiver input (pin 20) of the UART via U37, R21 and U38.

Notice that data on pin 25 of the UART will also be looped back if S4-6 is closed (half duplex operation). But in this case, data from the UART is also fed to pin 2 of J1.

Serial data from the UART that appears at pin 1 of U56 also drives transistor Q1 by way of R45 and R46 to supply the serial current loop output (SCLO) on pin 11 of J1. Q1 supplies 20 ma. (max.) current for a binary 1 and no current for a binary Ø.

Pin 23 of J1 (connected through R23 to +12 V dc) is the serial loop current source (SLCS). It can supply up to 20 ma of current to ground and is used when the external current loop device has no current source.

Data received from a current loop device enters Sol on pins 12 and 13 of J1 in the form of no current for a Ø and 20 ma of current for a 1. This input is rectified by bridge rectifier D3-D6 and applied to a light emitting diode (LED) in optical isolator U39. As its name implies, U39 electrically isolates the current loop circuit from the rest of the Sol. (This isolation permits a high offset voltage on pins 12 and 13 of J1.) For a 1, the LED is energized, and the light is optically coupled to the base of a photo transistor in U39 to cause the transistor to conduct. Conduction translates to a low, or mark, level at the input (pin 13) of U38. Since both the current loop and RS-232 received data (SLR1/SLR2 and SRD respectively) share the input to U38, both should not be used simultaneously.

There are five external control signals in the RS-232 section of the SDI/UART: two are sent to the external device (SRTS and SDTR), and three are received from the device (SCTS, SCD and SDSR).

SRTS on pin 4 of J1 was discussed earlier. SDTR (serial data terminal ready) is simply tied to +12 V dc through R24. This indicates to the external device that Sol is connected to it.

SCTS (serial clear to send), SCD (serial carrier detect) and SDSR (serial data set ready) indicate status of the external device. They enter Sol on pins 5, 8 and 6 of J1 respectively, and all three are active high. Following level conversion and inversion in line receivers U38, data on these lines is gated through noninverting tristate buffers U37 to the Internal Data Bus when $\overline{\text{PORT IN F8}}$ is active.

$\overline{\text{PORT IN F8}}$ also enables five bits of UART status to be reported over the Internal Data Bus. These are PE, FE, OE, DR and TBRE on pins 13, 14, 15, 19 and 22 respectively of the UART. They are defined as follows:

PE: Parity Error--received parity does not compare to that programmed. (Bit INT2)

FE: Framing Error--valid stop bit not received when expected. (Bit INT3)

OE: Overrun Error--CPU did not accept data before it was replaced with additional data. (Bit INT4)

DR: Data Ready--data received by UART is available when requested. (Bit INT6)

TBRE: Transmitter Buffer Register Empty--UART is ready to accept another word from the Bidirectional Data Bus. (Bit INT7)

8.5.4 Display Section

An understanding of how characters are formed on the video monitor will help you follow operation of the display section.

The monitor screen can be thought of as a large matrix of small light elements, or dots, that can be turned on and off. In this context the overall video presentation consists of light and dark dots.

In the Sol, the display format is 64 characters maximum per character row, with a maximum of 16 rows per frame (page). Thus, up to 1024 characters can be displayed per page.

A 9 x 13 (columns by lines) dot area, or character position, is alloted on the monitor screen for each displayed character (see Figures 8-2 and 8-3 on Page VIII-24). Consequently, each character row consisting of sixty-four 9 x 13 dot areas requires 13 horizontal scan lines. To provide spacing between both characters and rows, only 12 dot lines and seven dot columns within the 9 x 13 matrix are used for character display. Only nine of the available 12 dot lines, however, are used for any given character.

Let's take a closer look at how the 9 x 13 dot matrix is used. The first seven dot columns are available for all character displays; the last two are used to provide a space between characters. The first dot line in a character row is always blank to provide a space between character rows. As shown in Figure 8-2, the second through tenth dot lines are available for all upper case (capital) and control characters, all symbol and punctuation marks (except the comma and semicolon), and all lower case characters (except the g, j, p, q and y). As shown in Figure 8-3, dot lines five through 13 are available to display characters that normally extend below the base line--lower case g, j, p, q and y plus the comma and semicolon.

Now that we have a feeling for how characters are formed on the video monitor screen, we will move on to the circuit description.

Refer to Display Section Schematic in Section X, Page X-18.

The 14.31818 MHz DOT CLOCK, which defines the period of one dot (69.8 nsec) in a character display matrix, controls all timing in the Video Display Generator. DOT CLOCK is applied to pin 2 of U28, a four-bit binary counter that is preset to count from seven through 15 to divide DOT CLOCK by nine. Two 1.591 MHz outputs are supplied by U28: LOAD CLOCK on pin 11 and $\overline{\text{CHARACTER CLOCK}}$ on pin 12. Pin 11 is a low-active pulse of one DOT CLOCK duration. Pin 12 is high for five and low for four DOT CLOCK periods. Both the LOAD and $\overline{\text{CHARACTER CLOCK}}$ low-to-high transitions occur synchronously on the same DOT CLOCK.

$\overline{\text{CHARACTER CLOCK}}$, which defines the period of one character position (628 nsec), is inverted in U49 to become CHARACTER CLOCK. It performs most of the clocking functions in the Video Display Generator and is made available on pin 4 of J4 for use by external graphic display devices.

CHARACTER CLOCK is in turn divided in U31 and U33, both of which are presettable four-bit binary counters. Both start at count 3 when pin 8 of NAND gate U47 is low, and together they count 102 CHARACTER CLOCKS to define horizontal timing at 64 usec (102 x 628 nsec = 64 usec).

CHARACTER ADDRESS*	LINE ADDRESS	SCAN LINE NO.	COLUMN NO.									VIDEO INFORMATION BITS
			1	2	3	4	5	6	7	8	9	
1001001	1111	1	○	○	○	○	○	○	○	○	○	000000000 (blank)
	0000	2	○	●	●	●	●	●	○	○	○	011111000
	0001	3	○	○	○	●	○	○	○	○	○	000100000
	0010	4	○	○	○	●	○	○	○	○	○	000100000
	0011	5	○	○	○	●	○	○	○	○	○	000100000
	0100	6	○	○	○	●	○	○	○	○	○	000100000
	0101	7	○	○	○	●	○	○	○	○	○	000100000
	0110	8	○	○	○	●	○	○	○	○	○	000100000
	0111	9	○	○	○	●	○	○	○	○	○	000100000
	1000	10	○	●	●	●	●	●	○	○	○	011111000
	1001	11	○	○	○	○	○	○	○	○	○	000000000 (blank)
	1010	12	○	○	○	○	○	○	○	○	○	000000000 (blank)
1001001	1011	13	○	○	○	○	○	○	○	○	○	000000000 (blank)

*7-bit ASCII code for I

● = illuminated dot

Figure 8-2. Example of uppercase character (I) display.

CHARACTER ADDRESS*	LINE ADDRESS	SCAN LINE NO.	COLUMN NO.									VIDEO INFORMATION BITS
			1	2	3	4	5	6	7	8	9	
1110000	1111	1	○	○	○	○	○	○	○	○	○	000000000 (blank)
	0000	2	○	○	○	○	○	○	○	○	○	000000000 (blank)
	0001	3	○	○	○	○	○	○	○	○	○	000000000 (blank)
	0010	4	○	○	○	○	○	○	○	○	○	000000000 (blank)
	0011	5	●	○	●	●	●	○	○	○	○	101110000
	0100	6	●	●	○	○	○	●	○	○	○	110001000
	0101	7	●	○	○	○	○	●	○	○	○	100001000
	0110	8	●	○	○	○	○	●	○	○	○	100001000
	0111	9	●	●	○	○	○	●	○	○	○	110001000
	1000	10	●	○	●	●	●	○	○	○	○	101110000
	1001	11	●	○	○	○	○	○	○	○	○	100000000
	1010	12	●	○	○	○	○	○	○	○	○	100000000
1110000	1011	13	●	○	○	○	○	○	○	○	○	100000000

*7-bit ASCII code for p

● = illuminated dot

Figure 8-3. Example of lowercase character (p) display.

As indicated in Figure 8-4 on Page VIII-27, Subgroup Counter U31 and Group Counter U33 are preset to a count of 3 at the start of each horizontal scan line. U31 counts from 3 through 15 (13 character positions) and enables U33 for one count. U31 then counts Ø through 15 and enables U33 for the second count. The sequence continues through four more groups of 16 character positions, and at this point U33 is at its sixth count (a binary 9). Thus, pins 11 and 14 are high at pins 10 and 11 of U47. U31 continues to count from Ø, and on the ninth count (a binary 8) pin 9 of U47 goes high. The resulting low on output pin 8 of U47 loads three into U31 and U33, and the cycle repeats. The U31-U33 cycle, from preset, is then 13, 16, 16, 16, 16, 16 and 9 character position counts for a total of 102.

The QD output on pin 11 of U33 is SCAN ADV, and the QC output on pin 12 is HDISP. SCAN ADV is used to generate horizontal synchronization signals, and HDISP defines the start of the display portion of the horizontal scan line.

Four outputs from U31 and the two low order outputs of U33 (pins 13 and 14) are input to the Character Address Multiplexer, U30 and U32, which supplies the low order six address bits to the Display RAM (U14 through U21). The second address source for the Display RAM is the Address Bus, bits ADRØ-5. Address source selection is controlled by the output on pin 7 of D flip-flop U75. Pin 7 of U75 goes high when $\overline{\text{PAGE CC}}$ (the Display RAM) is active and PSYNC ● $\overline{\text{Ø2}}$ goes high (which it does in the middle of PSYNC). Pin 7 of U75 remains high for the rest of the memory access cycle.

The preset signal (pin 8 of U47) to U31 and U33 is applied to the Scan Counter (U40) via inverter U87. U40 counts the horizontal scan lines that make up a row of characters and supplies the line number to U25, the Character Generator ROM. (This ROM is discussed later.) U40 is preset to a count of 15 for the first scan line in the character row. It then counts from Ø through 11. On count 11, SCAN ENABLE on pin 8 of U47 is inverted in U87 to disable the Scan Counter. A decoder, comprised of NAND gates U59 and U60, decodes the 13th count (count 11) in U40 and SCAN ENABLE to supply a load pulse to pin 9 of U40. This resets U40 to a count of 15, and the cycle repeats. (Presetting the Scan Counter to a count of 15 permits the Character Generator ROM to provide a blank spacer line between character rows since line 15 in the ROM is always blank.)

The output on pin 8 of NAND gate U59, after inversion in U87, becomes the OVERFLOW LINE signal. This signal occurs after each character row and appears at pins 7 and 10 of Text Counter U62 to enable it to count. Thus, the Text Counter counts character rows. It resets itself with its carry output (pin 15) through another inverter in U87, with the reset count being determined by the state on pin 10 (VDISP) of J-$\overline{\text{K}}$ flip-flop U43. If VDISP is low, the Text Counter resets to a count of Ø; if VDISP is high, it resets to a count of 12.

Assume VDISP is active (low), which it is during the vertical display portion of the displayable area on the screen. (Refer to Figure 8-4.) U62 is then preset to a count of Ø and will count from Ø through 15 (16 character rows). The resulting carry output on count 15 of the Text Counter causes the U43 VDISP flip-flop to toggle. It also appears as a low on the load input of the Text Counter. The Text Counter is also enabled to reset by virtue of the OVERFLOW LINE going low after the reset of the Scan Counter. Since VDISP is now high, the Text Counter is reset to a count of 12 and will count 12 through 15 (four character rows). The carry output from the Text Counter then causes the U43 VDISP flip-flop to toggle, and the Text Counter is reset to a count of Ø. We can now see that the Text Counter counts 16 character rows when the display is active (VDISP is low) and four character rows when the display is blanked (VDISP is high). The total of 20 character rows represents a full display of 260 scan lines for 60 Hz operation (13 scan lines/row x 20 rows = 260 scan lines per page).

Horizontal and vertical synchronization signals are generated by two one-shot multivibrators consisting of three two-input NOR gates in U102. Horizontal sync is triggered by SCAN ADVANCE and vertical sync by $\overline{\text{VDISP}}$. Both circuits generate fixed-length sync pulses with adjustable starting times. C52 determines the length of the horizontal sync pulse and C53 the length of the vertical sync pulse. The starting times, with respect to triggering, are variable with variable resistors VR1 (HORIZ) and VR2 (VERT) to provide continuous adjustment of the display position on the screen. An exclusive OR gate in U74 combines the two sync pulses into a composite sync (COMP SYNC) signal. Note that the use of the exclusive OR inverts the horizontal sync pulses when the vertical sync pulse appears. Since vertical sync information is extracted in a monitor by an integrating, or averaging, process, this technique maintains horizontal synchronization during the vertical sync period.

Two types of blanking are available: control character blanking and video blanking. The first blanks control characters and causes cursor information to be displayed in their place. Video blanking forces portions of the video display to a white or black level, depending on whether normal or reverse video is selected with S1-4.

Control character blanking, switch selectable with S1-3, is accomplished with one NAND gate in U60 and one NAND gate in U61. When a control character is present in the Data Latch (U26 and U27), pins 3 and 15 of U26 are high. Assuming the blanking option is selected (S1-3 closed), the output of U60 ($\overline{\text{LOAD CLOCK}}$) is gated with the control character bits by U61 to clear the video parallel-to-serial converter, U41. U41 then loads all zeros instead of the character.

Video blanking is initiated by the PRE BLANK or COMP BLANK (pin 14 of Blank Latch U42) inputs to U59, a three-input NOR gate. The third input, the video output on pin 6 of exclusive OR gate U74, is blanked when any of the two blanking inputs is active.

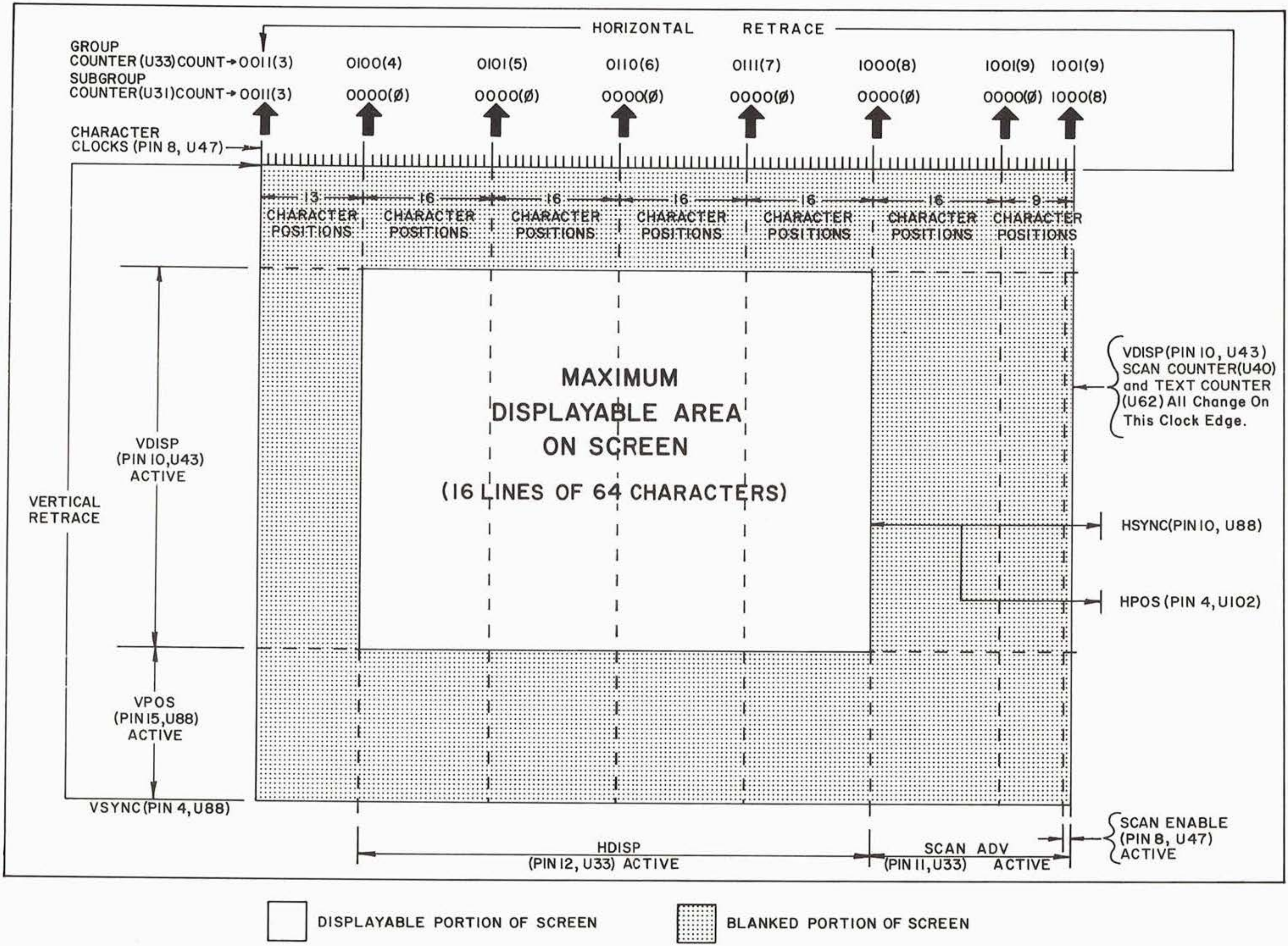

HORIZONTAL RETRACE
GROUP COUNTER (U33) COUNT → 0011(3)
SUBGROUP COUNTER(U31) COUNT → 0011(3)
0100(4)
0000(Ø)
0101(5)
0000(Ø)
0110(6)
0000(Ø)
0111(7)
0000(Ø)
1000(8)
0000(Ø)
1001(9)
0000(Ø)
1001(9)
1000(8)
CHARACTER CLOCKS (PIN 8, U47)
13 CHARACTER POSITIONS
16 CHARACTER POSITIONS
16 CHARACTER POSITIONS
16 CHARACTER POSITIONS
16 CHARACTER POSITIONS
16 CHARACTER POSITIONS
9 CHARACTER POSITIONS
MAXIMUM DISPLAYABLE AREA ON SCREEN
(16 LINES OF 64 CHARACTERS)
VDISP (PIN 10,U43) ACTIVE
VERTICAL RETRACE
VPOS (PIN 15,U88) ACTIVE
VSYNC (PIN 4, U88)
VDISP (PIN 10, U43) SCAN COUNTER(U40) and TEXT COUNTER (U62) All Change On This Clock Edge.
HSYNC(PIN 10, U88)
HPOS (PIN 4, U102)
SCAN ENABLE (PIN 8, U47) ACTIVE
HDISP (PIN 12, U33) ACTIVE
SCAN ADV (PIN 11, U33) ACTIVE
DISPLAYABLE PORTION OF SCREEN
BLANKED PORTION OF SCREEN

Figure 8-4. VIDEO DISPLAY TIMING

The PRE BLANK input provides "window shade" blanking which is analogous to pulling a window shade down from the top of the display. PRE BLANK is generated in one half of J-$\overline{K}$ flip-flop U43. U43 is reset by the TC output of First Screen Position Counter, Ull, and set by VDISP. The output on pin 7 of Ull is generated by the scrolling circuitry (to be discussed later) and defines the character row for which the "window shade" ends. It may begin with any character row from zero through 14.

The remaining video blanking function concerns the output on pin 14 of D flip-flop U42. This signal, COMP BLANK, is a composite of HDISP and VDISP.

Since there is a two character time delay between Display RAM addressing and the corresponding video output on pin 6 of exclusive OR gate U74, the horizontal and vertical blanking signals must be delayed an equal amount. U42, connected as a two-stage shift register, functions to shift the blanking into synchronization with the video. Since U42 is clocked by LOAD CLOCK (which has a period equal to one character time), COMP BLANK is delayed two character times from the input on pin 4 of U42. COMP BLANK is active low during nondisplayable portions of the video scan to override any video input data on pins 1 and 2 of NOR gate U59. The display is thus blanked.

The Display RAM consists of eight 1K x 1 bit RAM (random access memory) chips, U14 through U28. All chips are held permanently enabled by connecting their CE (pin 13) inputs to ground. Memory addressing is provided through two-to-one multiplexers (U30, U32 and U12) which select one of two display address sources: 1) an external address on Address Bus bits ADRØ-9 and 2) an internal address supplied by the Subgroup Counter (U31), Group Counter (U33) and the Beginning Address Counter (U1). The function of the address bits associated with each address source is as follows:

1. External address bits ADRØ-5 specify the character position (one of 64) in the character row.

2. External address bits ADR6-9 specify the character row position (one of 16) on the display screen.

3. Internal address bits, a total of six outputs from U31 and U33, specify the character position (one of 64) in the character row.

4. Internal address bits, the four outputs from U1, specify the character row position (one of 16) on the display screen.

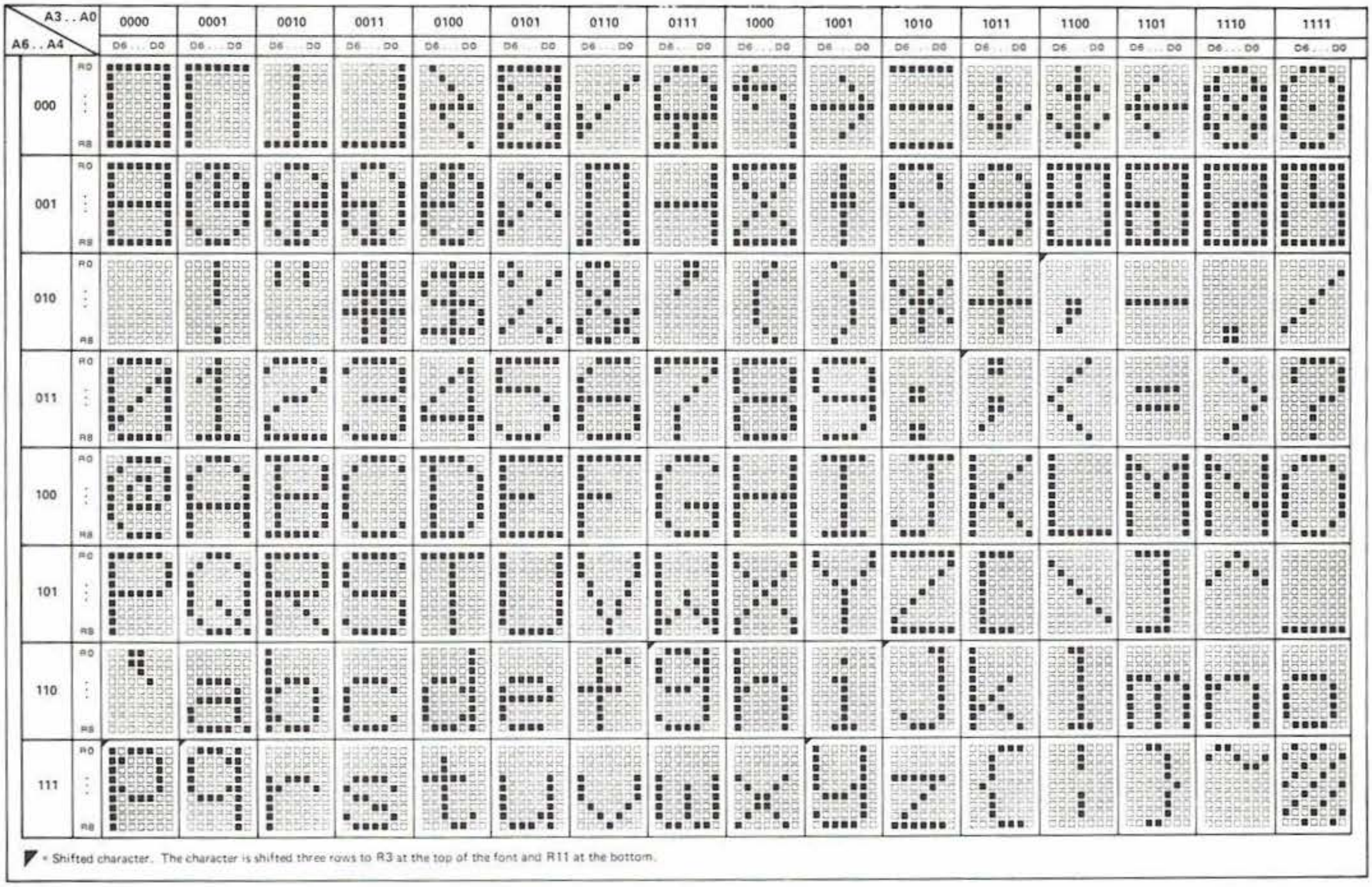
Normally the internal display address is multiplexed to the Display RAM. When the CPU or a DMA device requests access ($\overline{\text{PAGE CC}}$ active), the multiplexers switch to the external address lines, ADRØ-9.

Seven-bit ASCII-coded data is written into RAM chips U14 through U20 from bits DIOØ-6 of the Bidirectional Data Bus, and the cursor bit (DIO7) is written into RAM chip U21. This writing occurs when the write enable (WE) input to the RAM chips is low. This occurs when the Display RAM is addressed ($\overline{\text{PAGE CC}}$ active low) and MWRITE on S-100 Bus pin 68 is high. The enable is supplied on output pin 8 of NAND gate U44. Data is read out of the Display RAM when pin 8 of U44 is high. Data out of the Display RAM is placed on the Bidirectional Data Bus via tri-state drivers U29 and U89 when $\overline{\text{PAGE CC}}$ and PDBIN (S-100 Bus pin 78) are active. U29 and U89 are enabled by a low output on pin 11 of another U44 NAND gate.

Data out of the Display RAM is also strobed into Data Latches U26 and U27 by LOAD CLOCK. Seven outputs from these latches are used to address the Character Generator ROM, U25. Note that the output from RAM chip U19 is inverted in exclusive OR gate U74 before being applied to the C input (pin 13) of U26, and the complement (pin 14) of the QC output of U26 is used in addressing U25. This is done so that the Data latches will output the space code (Ø1ØØØØØ) to the Character Generator ROM when the latches are reset. These latches are reset each time $\overline{\text{PAGE CC}}$ is active by way of U75, a J-$\overline{\text{K}}$ flip-flop connected as a D flip-flop, and D flip-flop U42 ($\overline{\text{Q}}$ output pin 6). By outputting the space code on reset, the Data Latches insure a blank character position on the screen.

The Character Generator ROM, U25, has seven character address inputs (A1 through A7), four scan line inputs (RS1 through RS4) and seven data outputs (B1 through B7). It is programmed to generate seven bits (dots) of character information for the selected scan line of the character row. U25 also automatically blanks scan lines that are not a part of the character and shifts the g, j, p, q, y, comma and semicolon to the fifth through 13th scan lines in the dot matrix (refer to Figures 8-2 and 8-3 on Page VIII-24). Complete patterns for the 6574 and 6575 Character Generator ROM's are provided in Figures 8-5 and 8-6 respectively. Note that the address bits AØ through A6 in Figures 8-4 and 8-5 correspond to the A1 through A7 inputs to U25 on the schematic, scan lines RØ through R8 are specified by the RS1 through RS4 inputs to U25 on the schematic, and the data output bits DØ through D6 correspond to the B1 through B7 outputs from U25 on the schematic.

Let's see how the Character Generator ROM produces a character using an uppercase "C" and "T" as an example. In this example, these two characters are to be displayed in the first and second character positions respectively on the third character row of the display screen. Remember that the character position and row parameters are contained in the Display RAM since the 7-bit ASCII-coded



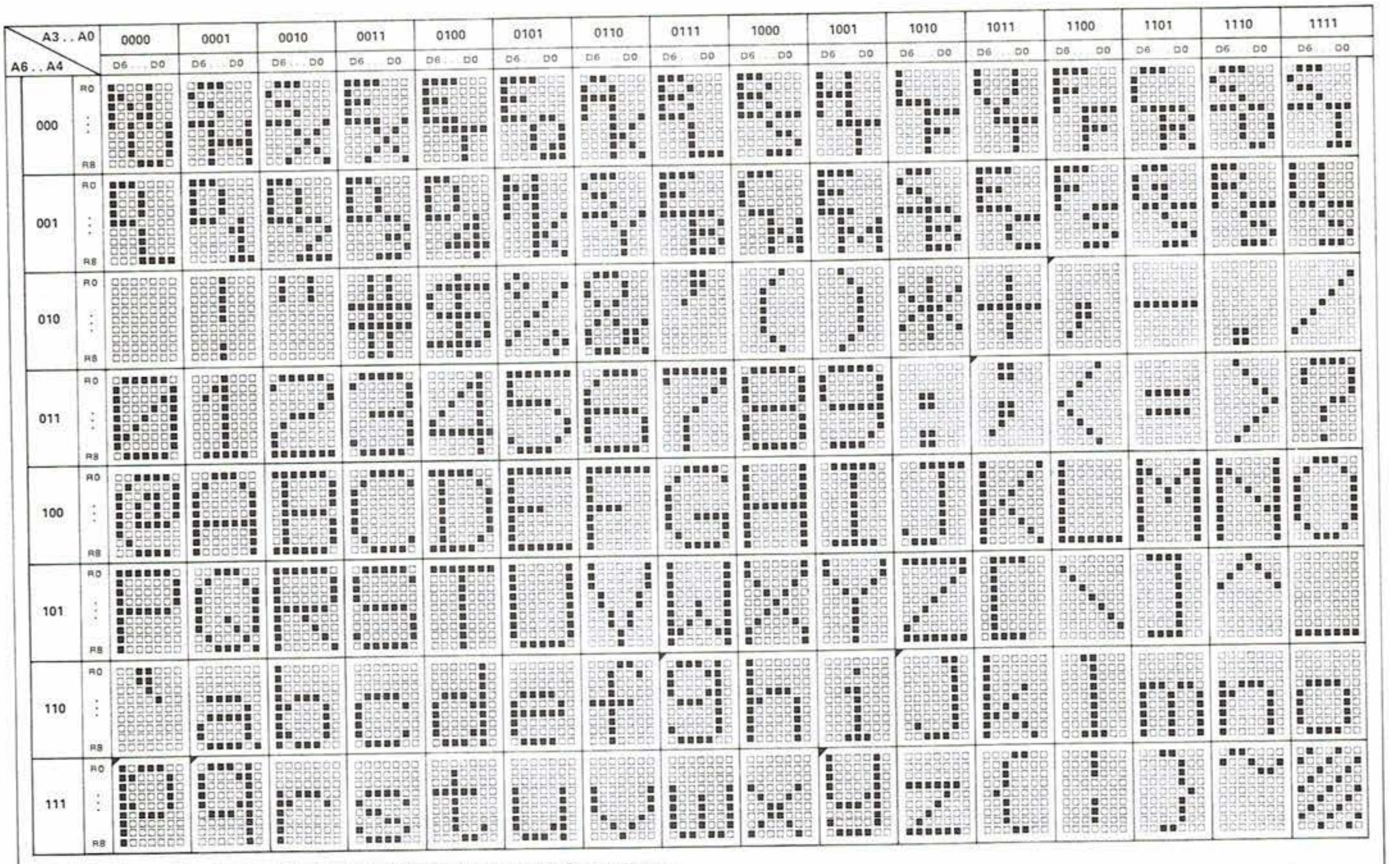
= Shifted character. The character is shifted three rows to R3 at the top of the font and R11 at the bottom.

Figure 8-5. 6574 Character Generator ROM pattern.

"C" and "T" were stored in the RAM in the proper character positions in the third character row.

After the first two character rows have been displayed, the Scan Counter (U40) is reset to a binary count of 15 (1111) and the Character and Line Address Multiplexers (U30, U32 and U12) call up the "C" in the Display RAM. The Scan Counter output specifies line 15 in the Character Generator ROM on RS1 through RS4. As previously mentioned, this line in the ROM is blank. Thus, the first scan line of the third character row is blank.

The 7-bit ASCII code for the "C" (1ØØØØ11) is input from the Display RAM to address the Character Generator ROM by way of the Data Latches (U26 and U27). This address is applied to ROM inputs A7 through A1 (A6 through AØ in Figures 8-5 and 8-6). The Scan Counter changes to a count of zero which specifies scan line RØ in the Character Generator ROM. As shown in Figures 8-5 and 8-6, the ROM in turn outputs a 7-bit word, ØØ1111Ø, on D6 through DØ respectively (B7 through B1 on the schematic).



= Shifted character. The character is shifted three rows to R3 at the top of the font and R11 at the bottom.

Figure 8-6. 6575 Character Generator ROM pattern.

For the second character position the Character and Line Address Multiplexers call up the "T" in the Display RAM. The resulting ASCII code for a "T" (1Ø1Ø1ØØ) ultimately appears on the address inputs to the Character Generator ROM. Since the Scan Counter is still at a count of zero, the ROM outputs 1111111. This process continues for the balance of the displayable portion of the video scan line.

At the end of the horizontal scan line, the Scan Counter changes to a binary count of ØØØ1 which specifies scan line R1 in the Character Generator ROM. The "C" and "T" are again called up from the Display RAM for the first and second character position respectively. The ROM consequently outputs Ø1ØØØØ1 and then ØØØ1ØØØ. This sequence continues through scan line R8 when the Scan Counter is at a count of 8 (1ØØØ) to produce the "C" and "T".

As discussed earlier, the Scan Counter cycles through 13 counts or scan lines. For the "C" and "T" in our example, the Scan Counter has counted ten lines (15, Ø, 1, 2, 3, 4, 5, 6, 7 and 8). The remaining three scan lines are not used in forming the "C" or "T", so on counts 9, 10 and 11 of the Scan Counter the Character

Generator ROM automatically outputs all zeros for these two character positions. After the last scan line in the third character row, the Scan Counter is reset to a count of 15 to start the fourth character row.

The Character Generator ROM output is converted from parallel to serial form in an 8-bit shift register (U41) that is clocked by DOT CLOCK. For each high bit on the input, the serial output (QH, pin 13) of U41 is high for one DOT CLOCK period. For each low bit, QH is low for one DOT CLOCK period. Note that parallel input bit PH (pin 14) is tied to ground. This effectively adds a low bit (or dot) following the data and provides one of the spacer dots between characters. The second spacer dot is generated by connecting the serial input (pin 1) to ground and applying LOAD CLOCK to the load (LD, pin 15) input to U41. When LOAD CLOCK goes low, which it does every ninth DOT CLOCK, U41 shifts in one zero.

A blink oscillator (two inverter sections in U88), a latch (one section in U42) and their associated components comprise the cursor circuit. The blink oscillator runs continuously at a rate set by R84 and C36. Its output has a nominal 0.5 sec period. If the blink option is selected with S1-5, the blink signal is applied to one input of a gate in U60. The other input to this gate is provided by the blink latch, one section in U41. If the cursor bit QA out of Data Latch U26 is high, D flip-flop U42 sets for the time the ROM is active on the character and remains set during the period when video data is shifted out of U41. The output of U42 is gated high through NAND gate U60 when BLINK (pin 6 of U88) is low. BLINK is held low when the blink option is not selected. The output of U60 is in turn gated with the video output of U41 in U74, an exclusive OR gate. U74 thus inverts the video if the output of U60 is high, and no inversion takes place if the output of U60 is low.

The video signal including the cursor, is gated to pin 9 of another U74 exclusive OR gate in the absence of any blanking signals at the other two inputs to NOR gate U59. If S1-4 is open, U74 inverts the video signal to produce a reverse (black on white) display. Raw video on pin 8 of U74 is supplied to pin 15 of J4. Video out on pin 6 of inverter U87 is combined with COMP SYNC on pin 8 of another U87 inverter in a resistive mixer, R80-R82, to meet EIA composite video signal standards, and coupled to P1 for use by a video monitor. This mixer has a 61-ohm output impedance.

Both Beginning Address Counter U1 and First Screen Position Counter U11 are enabled to advance their counts when pin 9 of J-$\overline{K}$ flip-flop U75 is low, which it is for about 600 nsec following $\overline{\text{OVERFLOW LINE}}$; that is, after the Scan Counter (U40) is loaded. This, of course, occurs at the end of every scan line in the character row.

The scroll circuit consists of U1, U11, Scoll Control Latch U2 and Screen Position Control Latch U13 and their associated circuitry. U1 and U11 are up and down counters respectively that are preset to the outputs of latches U2 and 13. U2 latches the starting row address from DIOØ-3 and U13 latches the data on DIO4-7, with $\overline{\text{PORT OUT FE}}$ being the strobe. Data on DIO4-7 specifies where the first line will be displayed. Thus, the number loaded into U1 is the address of the first displayable scan line, and the number loaded into U11 defines the character row (Ø through 15).

U11 is preset by $\overline{\text{VDISP}}$ from pin 9 of J-$\overline{\text{K}}$ flip-flop U43. This means U11 is forced to its preset condition from the end of the displayed text to the top of the next character row. During this time, pin 6 of another U43 J-$\overline{\text{K}}$ flip-flop is set high to preset U1. If U11 is preset to Ø, its TC output on pin 7 is low and pin 6 of U43 is reset to a low. This allows U1 to count with each horizontal scan line.

If U11 is preset to any number other than Ø, pin 6 of U43 cannot be reset low until U11 reaches zero. Assume U11 is preset to two. It must count down two character rows before U1 starts counting. During this time, pin 7 of U43 (PRE BLANK) is low, and as previously discussed, the display is blanked.

We can now see that the PRE BLANK time, often called "window shade", is variable with the number loaded into U11. Therefore, scrolling is performed by changing the numbers in U2 and U13 without the need to reposition the text within the Display RAM.

The remaining circuit in the Display Section consists of transistor Q2, one section of U87, 89 and 102. U88 and U102 are connected as a one-shot 250 msec timer that is triggered when $\overline{\text{PORT OUT FE}}$ goes active (pin 1 of inverter U87 goes high). Thus, when data is loaded into U2 and U13, this timer starts. Tri-state driver U89, which is enabled by $\overline{\text{PORT IN FE}}$, transmits the state of this timer to DIOØ on the Bidirectional Data Bus. The CPU can consequently test the timer status by looking for a high on DIOØ. This timing allows a 250 msec scroll rate without the need for complex timing routines in the CPU. Q2, R102 and C37 serve to speed up timer reset.

8.5.5 Audio Tape I/O

Refer to Audio Tape I/O Schematic in Section X, Page X-19.

Timing for the Audio Tape I/O is derived from the 1200, 2400, 4800, 19,200 and 38,400 Hz signals received from the Baud Rate Generator in the Input/Output section of Sol. The first two are used by the write data synchronizer (U100) and the digital-to-audio converter (U101).

The remaining three signals are fed to two sections of U111, a quad multiplexer or select gate. All four sections of U111 are used to select clocks for low speed or high speed operation according to the select inputs, pins 9 (A) and 14 (B). The states of these two select inputs must be complementary to each other in order to select the high or low speed clocks. Specifically, A must be high and B low to select high speed clocks; the converse condition selects low speed clocks. The select inputs are supplied by TAPE HI SPEED and TAPE HI SPEED.

The output of the second section on pin 11 of U111 is BYTE WRITE CLOCK, 4800 Hz on low speed and 19.2 KHz on high speed. The third section outputs a 19.2 KHz (high speed) or 38.4 KHz (low speed) timing signal to input pin 10 of binary up counter (U112).

RECOVER CLOCK is produced by a phase locked loop (U110), another U112 binary up counter and the first and fourth sections of U111. The signal input (pin 14) to U110 is supplied from output pin 1 of D flip-flop U113. It is a constant frequency, regardless of whether one or two transitions are detected in the read data during the count out time (12 counts) of the U112 counter with outputs on pins 13 and 14. A phase comparator in U110 compares the signal input to the output of a voltage controlled oscillator (VCO) in U110 (pin 4). By feeding the VCO output through a counter (the other half of U112) before feeding the counter output back to the compare input (pin 3) of U110, the circuit acts as a frequency multiplier. The output of this circuit remains locked, therefore, to a multiple of the signal input on pin 14 of U110.

The output of U110 is nominally 19.2 KHz. The actual output is determined by the signal input which in turn is a function of tape speed. In other words, the phase lock loop circuit tracks input frequency variations. And it will track such variations within its locking range which is determined by the setting of variable resistor VR3 (connected to pin 12 of U110).

For high speed, the divide-by-four output of U112 (pin 4) is selected as RECOVER CLOCK. For low speed, the VCO output of U110 is selected for RECOVER CLOCK. This clock serves as read clock for the CDI UART, U69.

CDI control involves PORT IN FA, PORT IN FB, PORT OUT FB, TAPE CONTROL 1 and 2, POC (power on clear), TAPE HIGH SPEED and TAPE HI SPEED. The last two were previously explained in the discussion of U111. PORT IN FA strobes the CDI UART status (DR, TBRE, OE and FE--refer to Page VIII-22 for definitions) to the Internal Data Bus, INT3-7. PORT IN FB strobes received data on pins 5-12 of U69 to the Internal Data Bus, INTØ-7. PORT OUT FB loads data from the Bidirectional Data Bus (DIOØ-7) into U69. POC simply resets U69 whenever power is applied to the Sol.

TAPE CONTROL 1 and 2 are used to turn one or two recorder motors on and off. An active low TAPE CONTROL 1 energizes K1 to close its contacts and turn recorder #1 on; a high de-energizes K1 to turn the recorder off. TAPE CONTROL 2 does the same thing with K2 to control another recorder. Diodes D13 and 14, which shunt K1 and K2 respectively, prevent damage to the logic circuitry in the Input/ Output section due to inductive kickback. R155 and 156 are current limiters that keep the relay contacts from "welding" together.

When the CDI is in the write mode, data is input to the UART (U69) under control of $\overline{\text{PORT OUT FB}}$. Upon completion of this strobe, the transmit sequence is initiated within the UART, with the transmission rate being governed by BYTE WRITE CLOCK.

The transmission sequence begins with a start bit, a low (data zero) on the UART's TO output. It is followed by eight data bits and two stop bits (high on the UART's TO output), with the number of bits being fixed by the connections to pins 34 through 39 of U69.

The data from U69 is applied to the D input of D flip-flop U100 which is clocked at 1200 Hz. Consequently, the output on pin 1 of U100 follows the input data on pin 5 after the rising edge of the 1200 Hz clock. This output is connected to the reset (pin 4) of U101, so when the data out of the UART is high, the first section in U101 is forced to a reset condition. In this condition the J and K inputs to the second stage of U101 are held high which allows the flip-flop to change state on the rising edge of the clock.

The clock for U101 (OUTPUT CLOCK) is 2400 Hz in the high speed mode or 4800 Hz in the low speed mode. This clock is derived from 2400 Hz in conjunction with the low speed select signal in NAND gate U98 and exclusive-OR gate U99.

In the high speed mode, pins 12 and 13 of U98 are held low, thus holding pin 10 of U98 high. As a result the 2400 Hz signal is inverted in U99 to become the clock for U101.

Pins 12 and 13 of U98 are held high, however, in the low speed mode to enable U98. In this case R117 and C47 provide a delay in the U98 gate. When the 2400 Hz signal on pin 2 of U99 changes state, so does pin 3 of U99. Also, C47 charges through R117 for several usec, at which point pin 10 of U98 is brought to the opposite polarity. The output from U99 then goes high. A series of positive pulses, with a pulse width approximately equal to the R117, C47 time constant (10 usec) and occuring at every transition of the 2400 Hz signal, appears on pin 3 of U99. This circuit thus operates as a frequency doubler in the low speed mode to provide a 4800 Hz clock for U101.

The 2400 Hz signal from which the U101 clocks are derived also produces the 1200 Hz clock signal for U100. As a result the 1200 Hz signal changes state following a propagation delay after the 2400 Hz signal falls.

As previously stated, the second stage of U101 is allowed to change state on the positive going transitions of the OUTPUT CLOCK as long as the data out of the synchronizer is a "1". The end result is an output on pin 14 of U101 that is one-half the clock frequency (1200 Hz and 2400 Hz in the high and low speed modes respectively).

Assume the data stream out of the UART goes low ("Ø"). On the next rising edge of the 1200 Hz signal, U100 will reset with Q low and $\overline{Q}$ high. A low reset on pin 4 of U101 enables the first U101 stage to toggle on the next rising edge of the OUTPUT CLOCK which occurs 1/2400 second after the synchronizer output falls. Remember that OUTPUT CLOCK moves from a low to a high shortly before the 1200 Hz signal did. The reset on pin 4 of U101 is thus removed slightly after the OUTPUT CLOCK occurred. With the J and K inputs to the first U101 stage high, its output will change state on each succeeding low to high transition of OUTPUT CLOCK. The second U101 stage in turn can only toggle on the positive going transition of OUTPUT CLOCK when its J and K inputs are high. Since the inputs are high at one-half the clock rate, by virtue of the first U101 stage, the second U101 stage toggles at one-fourth the OUTPUT CLOCK rate.

The two sections of U101, therefore, operate as a frequency divider, dividing the OUTPUT CLOCK by two when the write data is a "1" and by four when the data is a "Ø". Thus, in the low speed mode, four cycles of the 1200 Hz represent a "Ø" and eight cycles of 2400 Hz represent a "1". In the high speed mode, one cycle of 1200 Hz represents a "1" and one-half cycle of 600 represents a "Ø".

The output on pin 14 of U101 is applied to one section in U109 which provides sufficient current drive for the divider network. This divider and a jumper arrangement allow selecting one of three outputs to be fed to the audio output jack J6. The I-to-J jumper selects a 500 mv signal for the auxiliary input to an audio recorder; the I-to-H jumper selects a 50 mv signal for the microphone input to an audio recorder.

When the CDI is in the read mode, data from the recorders enters on J7. This input is fed to the negative input (pin 6) of operational amplifier U108.

The first section of U108 is a high gain amplifier, with its gain (approximately 100) being determined by R142 and R143. The output from this amplifier is coupled to input pin 2 of the following U108 stage and the base of a Darlington pair (Q4 and Q5) which provides high current gain.

Current into the base of transistor Q5 causes C67 to discharge. (C67 charges through R39 to 5 V dc.) The voltage on C67 in turn controls the gate of field effect transistor (FET) Q3. Q3 functions as a variable resistor which can be changed by its gate voltage. Since Q3 is connected between ground and the input network to the first U108 stage, it serves as a variable shunt. A low gate voltage on Q3 decreases the shunt resistance and the input to U108. In a like manner, a high voltage on C67 results in an increased input to U108. Q3, Q4 and Q5 with their associated circuitry, therefore, serve as an automatic gain control (AGC) circuit which limits the input to the second U108 stage to approximately a positive 2 volt peak signal.

The second stage of U108 is a comparator with hysteresis that performs the needed audio to digital conversion. Feedback resistor R147, in conjunction with R145, establishes the level on the positive input (pin 3) of U108. This level, be it positive or negative, is the threshold voltage, ±50 mv, which the negative input (pin 2) must exceed in order for the output of U108 to switch levels, positive to negative and the converse. Since the feedback loop is regenerative, U108 switches at its maximum rate, and U108 switches on each transition of the audio signal input. It is in this manner that U108 performs the audio to digital conversion.

The digital output of U108 is inverted in one section of inverter U109 and applied to pin 9 of exclusive OR gate U99 which is connected as a buffer without inversion. If the output of U109 is low, the output on pin 10 of U99 is also low and the output on pin 4 of another U99 exclusive OR gate is high. The voltage across C49 under this condition is minimal. When the output of U109 goes high, C49 starts to charge through R118 until pin 9 of U99 crosses the threshold of that gate. At this point pin 10 of U99 goes high, and since the two inputs to the second exclusive-OR gate are both high, pin 4 of U99 goes low. C49 now discharges because pins 9 and 10 of U99 are at the same level so that the circuit can repeat the operation on the next high to low transition at pin 4 of U109. R118, C49 and U99 consequently serve as a transition detector that produces a pulse less than one microsecond long for each transition of the output on pin 4 of U109, regardless of the polarity of the transition.

Transition pulses from U99 clock both D flip-flops in U113. A transition pulse clocks the top U113 at pin 3 which sets Q (pin 1) high and Q (pin 2) low to enable up binary counter U112 on pin 15. Pin 1 is applied to the D input (pin 9) of the lower U113 and the circuit remains in this state until one of two things occurs: 1) a second transition pulse arrives before U112 reaches count 12 or 2) U112 reaches count 12.

If a second transition pulse arrives before count 12, the bottom U113 stage is set and presents a "1" to the D input (pin 9) of flip-flop U100. This is clocked by the $\overline{Q}$ output on pin 2 of U113 as a low to pin 12 of U100.

If a transition pulse does not arrive before count 12, the bottom U113 stage outputs a "Ø" to input pin 9 of U100. On count 12, the C and D outputs of U112 go high to reset U113 by way of U98 at pin 4. As a result the U100 clock goes high, as does pin 12 of

U100. The output on pin 12 of U100 is inverted by U109 and applied to the receive input (pin 20) of the UART.

The Q output on pin 1 of U113, which occurs at the actual bit rate of the incoming data, is also used by the receive clock circuitry to reconstruct the receive clock from the data signal.

Received data undergoes serial-to-parallel conversion in the UART and is placed on the RO1-8 data outputs of the UART when ROD (pin 4 of the UART) is low ($\overline{\text{PORT IN FB}}$ active) and onto INTØ-7.

Four status outputs from the UART can also be enabled when SFD (pin 16) is low. These four bits are FE (framing error), OE (overrun error), DR (data ready) and TBRE (transmitter buffer register empty).

8.6 KEYBOARD

8.6.1 Block Diagram Analysis

A simplified block diagram of the keyboard is provided on Page X-25 in Section X.

The Clock Oscillator produces the basic timing signals for the keyboard, and they are distributed as indicated.

At the heart of the keyboard is a Key Switch Capacitive Matrix which can be viewed as a 16 x 16 X-Y matrix, with X being the column and Y the row. Conceptually, a key depression increases the capacitance between the X and Y coordinates that uniquely define that key.

The Column Scanner supplies a pulse train to the X lines in the matrix, with only one line being pulsed at any given point in time. When a key is depressed to increase the capacitance between the Column Scanner output and a Row Scanner input, the X-Y coordinates for that key are detected to provide an input to the Sense Circuit.

The Sense Circuit in turn generates an input to the Sequence Detector when a key closure occurs. This detector basically detects key closures and count cycles of the Row Scanner to discriminate against false key signals and insure that valid closures are serviced in order.

In the absence of key closures, the Sequence Detector feeds PKD to the Sense Circuit to increase its threshold. This action serves to increase the circuit's noise immunity. On valid key closures, the PKD input is such as to decrease the Sense Circuit's threshold. When valid key closures exist, the Sequence Detector strobes data into the Output Latch. The low order four bits to this latch are supplied by the Row Scanner; the high order four bits are supplied by the Encoding ROM, with the data being determined by inputs from the Column Scanner and Function Latch Decoder. This strobe (Data Out) also enables the Strobe Generator to output $\overline{\text{STROBE}}$, a 6 usec pulse that signals the Sol CPU that the Keyboard is ready to send data.

Eight bits of keyboard data (KBDØ through KBD7) are stored in the Output Latch. KBDØ through KBD6 represent the ASCII code for the character associated with the key closure, or closures, that initiated the Data Out strobe from the Sequence Detector. KBD7 is used only for special control characters (e.g. MODE SELECT, CLEAR and cursor movement) that are recognized by the Sol program. The data on KBDØ-7 is input to the Sol CPU when it issues $\overline{\text{PORT IN FC}}$ (refer to Paragraph 8.5.2 on Page VIII-14).

The Repeat Counter is enabled when the REPEAT key and a character key in the Key Switch Capacitive Matrix are pressed at the same time. When this occurs, Key Out (initiated by the character key closure) is active, and the Repeat Counter generates a periodic Repeat Strobe. This strobe disables the Sequence Detector and causes the Strobe Generator to output repetitive $\overline{\text{STROBE}}$ pulses. Column 3Ø also prevents the Sequence Detector from strobing additional data into the Output Latch.

The Function Latch and Decoder latches and decodes the Low Order Count from the Row Scanner when the "function key" column in the Switch Matrix is selected by the Column Scanner. It then outputs, as appropriate, $\overline{\text{LOCAL}}$, $\overline{\text{RST}}$ and $\overline{\text{BRK}}$ to J1 and SHIFT/SHIFT LOCK, UPPER CASE and CONTROL bits to the Encoding ROM. The latter three supply three of the seven address bits to the ROM which specify the high order four KBD bits (KBD4-7).

All keyboard outputs are supplied to J1 which is connected to J3 on the Sol-PC.

8.6.2 Circuit Description

Refer to the Keyboard schematic in Section X, Page X-23.

Keyboard operation is controlled by a 3 usec clock circuit consisting of NAND gate U7, R7 and C7. U7 is connected as a Schmitt trigger inverter with negative feedback through R7 and C7. The output on pin 11 of U7, a square wave with a 3 usec period, is inverted in U4 (a NAND gate connected as a simple inverter) and applied to the clock input (pin 11) of U8. U8 operates in a toggle mode by virtue of feeding its $\overline{\text{Q}}$ output on pin 8 to the D input on pin 12. Thus, its output state changes on each clock to produce a 6 usec and an inverted 6 usec clock on pins 9 and 8 respectively.

Each of these outputs is connected to a section of U7 where each is AND'ed with the 3 usec clock. This generates two negative going clocks at pins 8 and 6 of U7. These outputs are called $\overline{\text{ØI}}$ and

$\overline{\text{Ø2}}$ respectively. This circuit thus generates a symmetrical two phase clock, with each phase having a 6 usec period with a 1.5 usec negative going pulse.

$\overline{\text{Ø1}}$ advances the cascaded ripple counter, U5 and 6, in the Column Scanner circuit (U5, U6, NAND gates U4 and decoders U17 and U21). U6 divides $\overline{\text{Ø1}}$ by two on each advance. The output on pin 12 is consequently a square wave with a 12 usec period, the output on pin 9 is a square wave with a 24 usec period, and so on to pin 11 which has a 96 usec period. The output on pin 11 is then divided by two in U5 to provide 192, 384, 768 and 1536 usec periods. We will call these Clock 1 for the 12 usec period, Clock 2 for the 24 usec period, Clock 4 for the 48 usec period, and so on from Clock 8, 16, 32, 64 and 128.

Clocks 16, 32 and 64 are applied to the A, B and C inputs of binary-to-decimal decoders U17 and U21. In order for these decoders to yield outputs, their D inputs (pin 12) must be low. U4 is used to enable one or the other of these inputs, with Clock 128 being the determining factor. When Clock 128 is low, U17 is selected through U4 when $\overline{\text{Ø1}}$ is high at pin 4 of U4. U21 is selected when Clock 128 is high and $\overline{\text{Ø1}}$ is high at pin 13 of U4. By AND'ing $\overline{\text{Ø1}}$ and Clock 128, neither decoder is selected when $\overline{\text{Ø1}}$ is low, the time U5 and U6 count. During this time false binary signals can appear on the outputs of U5 and 6.

The net effect is that only one of the 15 outputs from U17 and 21 will be low, and this low advances on each count advance. The low outputs of U17 and 21 drive the column lines in the key switch matrix.

Clocks 1 through 8 are connected to analog multiplexers U19 and U22. Only one channel from input to output is connected at one time. Note that Clock 8 and $\overline{\text{Clock 8}}$ from U6 enable U19 and U22 respectively. U19 and U22 (the Row Scanner) thus scan through the 16 rows in the sequence indicated by the numbers contained within the "boxes" of the key switch matrix. An entire scan of the rows is made before the next column is selected by U17 and 21.

We now have U17 and U21 driving the column lines and U19 and U22 testing each row line by connecting it to an input to the Capacitance Keyswitch (KTC) Detector. These two inputs are normally high at 5 volts. Within the switch matrix there is a small capacitance connected between each column and row line; that is, there is a capacitance associated with each key on the keyboard. When a key is depressed on the keyboard, the capacitance associated with that key increases. When the column and row lines associated with that key are selected, there is a significant voltage difference between the two and the capacitance charges to produce a small negative going spike at the input to the Capacitance Keyswitch Detector.

This detector circuit consists of three transistors, Q7, Q8, and Q9 (connected as a linear amplifier with negative feedback) followed by Q4 and Q2. Q4 and Q2 are large signal amplifiers biased in their cut-off region. The input to the detector is selectively connected to +5 V dc by way of the analog multiplexers (U19 and U22), the row matrix wires, and the 33K resistors. A key depression causes a negative current pulse through R16 to the base of the input amplifier transistor, Q8, which is biased near cut-off. The pulse is then amplified by Q8 with inversion to appear as a positive pulse at the input of Q7. Q7 is an emitter follower circuit which gives a positive pulse at its output, across R18, at a low impedance. This signal is coupled back to the input through transistor Q9, a common base amplifier which has its base clamped to 2.5 V dc by zener diode CR4. When the positive pulse appears at the emitter of Q9, it is amplified without inversion and applied to the input of Q8. Since the original input was a negative pulse, the positive pulse constitutes negative feedback. The output across R18, a positive pulse, is further amplified by pulse amplifier transistor Q4, a common base amplifier that is normally biased off. The output stage Q2 is biased in the cut-off region also, but a sufficient positive pulse from Q4 will cause Q2 to conduct to give a negative pulse output across R12.

Transistors Q1, Q6, Q5 and Q3, represent a second pulse amplifier circuit that is analogous to transistors Q9, Q8, Q7 and Q4 respectively. The output of this second amplifier, which appears at the collector of Q3, is also connected to the base of the output transistor Q2. An input pulse from either U19 or U22 will therefore supply an amplified negative pulse to pin 13 of NOR gate U14.

The $\overline{\text{PKD}}$ signal through R24 helps to set the threshold at the base of Q4 and Q3. This threshold is normally high when $\overline{\text{PKD}}$ is high, so the output from Q7 and Q5 has to overcome a higher threshold at the emitter of Q4 and Q3 in order to cause conduction of Q4 and Q3. On the second such pulse on the same count address, $\overline{\text{PKD}}$ goes low to reduce the threshold at the bases of Q4 and Q3. This sensitizes the circuit, acting as a positive feedback path, and gives an output. Thus two consecutive detections of a key stroke are necessary to give an output. This feature provides noise immunity since a single noise pulse will not pass through the amplifier. The complete key switch matrix is scanned at a very high rate compared to the time it takes to physically press and release a key. Thus a key closure will be detected, even though the key is not held down for any appreciable time.

Two sections of NOR gate U14 are connected as a cross-coupled flip-flop. A low on pin 13 of U14 sets output pin 11 of U14 high, providing that the low is longer than 1.5 usec (which it is when a valid key closure is detected). That is because $\overline{\text{Ø1}}$ is applied to pin 9 of U14. $\overline{\text{Ø1}}$ effectively prevents switching noise, which is short in duration, from being interpreted as a key closure. The high, let's call it KEY, on pin 11 of U14 will remain until $\overline{\text{Ø1}}$ again goes low about 4.5 usec later.

KEY is fed to pin 5 of 8-input NAND gate U25, pin 9 of ROM U20 and pin 1 of NAND gate U27. Let's examine the other inputs to U25.

KEY, as mentioned, is fed to pin 9 of U20 which is a 256 x 4 bit static ROM. Only two bits are used. For each possible row-column combination, there is one storage location in U20. DI1 and DO1 (pins 9 and 11) are the input and output respectively of one bit location; DI2 and DO2 serve the same functions for the other bit location. The row count is applied to AØ-4 and the column count is applied to A5-7 to address U20.

When a key closure is detected, the counts are presented to U20 continuously. When the counts change shortly after the falling edge of $\overline{\text{ØI}}$, U20 outputs the status of the address that is already stored in the ROM about 1 usec later on pin 10. On the rising edge of Ø1 after the address change, the status on pin 10 is latched in one-half of D flip-flop U26 and presented at output pins 9 and 8. About 1.5 usec later the R/W signal on pin 20 of U20 goes low, and the KEY signal on pin 9 enters the specified location in U20. Note that this KEY is related with the new count address. The key stored in U26 represents the preceding address. We consequently call the KEY in U26 "KEY minus 1", and it is applied to pin 11 of U25.

The remaining inputs to U25 are 1) Ø2 (an inverted $\overline{\text{Ø2}}$) on pin 12, 2) a repeat strobe signal on pin 4 (supplied by pin 11 of NAND gate U16 which is high without a repeat command), 3) PKD minus 1 on pin 6 (supplied on pin 3 of U26 which is low if three or more count cycles have occurred since one key closure), and 4) the column output on pin 4 of U17 which is applied to pins 1, 2 and 3. The last signal drives the column associated with the special function keys on the keyboard (SHIFT, SHIFT LOCK, LOCAL, BREAK, UPPER CASE, REPEAT and CONTROL).

In order for U25 to output a low on pin 8, therefore, we need a current KEY, a KEY from the preceding count cycle, no repeat function, no drive on pin 4 (column 3Ø, hexadecimal), and we must be on the second count cycle during the current key depression.

With these conditions satisfied output pin 8 of U25 goes low. It is inverted by U10 to a high on pin 11. This signal then clocks the output latches, U1 and 2. On this signal, the data present on the inputs are latched into U1 and 2, and it remains latched until the next output on pin 8 of U25 occurs.

A low on pin 8 of U25 also resets one-half of D flip-flop U11 at pin 13 which causes output pin 9 to go low. On the rising edge of the inverted 6 usec clock from U8, the second U11 stage sets and output pin 5 goes low to clear the first stage. The high on output pin 6 is inverted by NAND gate U10 to supply a low active $\overline{\text{STROBE}}$ on pin 3 of J1. (Note that J1 on the keyboard connects to J3 on the Sol-PC.) The next inverted 6 usec clock resets the second U11 stage. We thus have a 6 usec strobe pulse following the latching of data into U1 and U2.

The complement of KEY minus 1 on output pin 8 of U26 is fed to input pin 10 of NAND gate U16 and is translated to a high on pin 8. The other input on pin 9 is high at this time since it is driven by the signal which indicates the third count cycle. A three-input NAND gate, U27, thus has a high on pin 2. A second input on pin 1 is KEY which is active (high) from the first count cycle of the key closure. The remaining input on pin 13 is supplied by pin 11 of U16, and it is low only when the repeat function is operating. U27 is consequently satisfied and outputs a low on pin 12.

This low appears at pin 5 of NOR gate U16. Pin 4 of U5 is high at this point by virtue of a low on pin 1 of U16 which indicates the third count. Thus, the high on pin 6 of U16 will be stored in the second bit location U20 when $\overline{\text{Ø2}}$ goes low at pin 20 of U20. When this happens DO2 (pin 12) of U20 goes high to indicate the new status of this bit.

The DO2 output is inverted in U10 and applied to input pin 2 of another U26 D flip-flop and to the Capacitance Keyswitch Detector as PKD. PKD serves to lower the detector threshold; that is, the detector offers less "resistance" to its input. This is positive feedback that allows the detector to discriminate between noise and a key closure. Note that two key closures are required before the detector threshold is lowered.

The inverted DO2 output from U20 also appears at the D input (pin 2) of U26. Since this flip flop is clocked by $\overline{\text{Ø1}}$, the prior status of $\overline{\text{PKD}}$, called "$\overline{\text{PKD}}$ minus 1", is already present in this latch on output pin 5. If we are on the second count cycle of a key closure, pin 5 is high. If we are on the third count or more, it is low to inhibit U25. As previously mentioned, $\overline{\text{PKD}}$ minus 1 is also connected to the NOR gate (U16) used to feed data to pin 11 of U20 from KEY minus 1.

When the current KEY signal is released, pin 12 of NAND gate U27 and pin 5 of NAND gate U16 go high. The U16 NAND gate that inputs to pin 4 of U16 looks at KEY minus 1 on pin 2 and the complement of $\overline{\text{PKD}}$ minus 1 on pin 1. Thus, pin 1 is high for the first one and a half counts and pin 2 is high for the first count. Upon release of KEY, therefore, pin 3 of U16 is low for the first count. On the second count, KEY minus 1 goes low--as do pin 6 of U16 and pin 12 of U20. On the next $\overline{\text{Ø2}}$ clock, the data is read into U20. The output on pin 12 of U20 changes to remove $\overline{\text{PKD}}$ which increases the Capacitance Keyswitch Detector threshold for greater noise immunity. It also sets $\overline{\text{PKD}}$ minus 1 on pin 5 of U26 on the third count cycle following release of KEY. On the third cycle the circuit reverts to its original state.

This circuit, comprised of U20, U26, U16 and U17 serves two functions. By requiring two events during two consecutive count cycles before generating a KEY, it discriminates against false key closures. It also insures that multiple key strokes are serviced in order. (This is the n-key rollover feature.) That is because the row-column addresses are continuously presented to U20 and this circuit's cycle can occur for each possible key closure. U20 can thus contain data for all possible key closures, and the data will enter U1 and U2 on the KEY generated for each closure as the row-column count progresses.

The previously mentioned column 3Ø output on pin 4 of U17 drives the keyboard control key "switches". Data for these key closures, present on pins 1, 2 and 3 of addressable latch U12 is latched in U12 during Clock 8 and Ø2 when column 3Ø is driven. Pin 13 of U12 is connected to the complement of PKD minus 1. Thus, the data (active low) is strobed into U12 on the first count cycle. During the third count it will be strobed again and a high is read in. When the key is released, a low is strobed in again. As a result, a high active pulse appears on the output line related to the key that was closed for the duration of the key closure.

SHIFT and SHIFT LOCK, on pins 11 and 10 respectively, are applied through U23 inverter stages to NOR gates U13 and U14. These are connected as a cross-coupled flip-flop. An active SHIFT sets this flip-flop at pin 5 of U13 to make output pin 6 of U13 and output pin 3 of U14 high. The latter is connected to pin 3 of U18, a 512 x 4 bit ROM. U18 is programmed to output the high-order four bits of the data to U1 according to the states of pins 1, 2 or 3.

The U13-14 flip-flop is set to a high on pin 6 if SHIFT LOCK is active. As can be seen, the shift bit to U18 is high by virtue of the low on pin 6 of U13 and it will remain so until SHIFT again causes U13-14 to change state. When output pin 6 of U14 is high, pin 12 of U24 is low to turn light emitting diode LED1 on. This LED is located in the SHIFT LOCK key and indicates the keyboard is in a locked shift condition.

When UPPER CASE is active, pin 7 of U12 goes high to clock D flip-flop U15 on pin 3. This flip-flop is connected to operate in a toggle mode. On the UPPER CASE "clock", pin 5 of U15 goes to make pin 2 of U18 low. The high on pin 6 of U15 is inverted by U24 to turn on LED2. LED2 is located in the UPPER CASE key. A second closure of this key toggles U15 to the opposite condition.

Now assume the LOCAL key is depressed, the output on pin 5 of U12 goes active high to clock the other D flip-flop U15 stage at pin 11. This stage also operates as a toggle, and output pin 9 goes low to become $\overline{\text{LOCAL}}$ on pin 14 of J1. Again, the high on output pin 8 causes LED3, the LOCAL light, to turn on. A second closure of the LOCAL key toggles this section of U15 to the opposite condition. Note that $\overline{\text{LOCAL}}$ has no affect on keyboard data.

The other outputs from U12 are BREAK (pin 12), CONTROL (pin 6) and REPEAT (pin 9). BREAK is inverted in U23 to become $\overline{\text{BRK}}$ on pin 4 of J1. CONTROL is applied directly to input pin 1 of U18 so that the control character related to the low order bits enters U1 and U2.

REPEAT is applied to pins 10 and 11 of NAND gate U27 and pin 13 of NAND gate U16. The input to U27 is gated with UPPER CASE to generate $\overline{\text{RST}}$ at pin 13 of J1. This means, of course, that REPEAT and UPPER CASE must be depressed at the same time to generate $\overline{\text{RST}}$.

On pin 13 of U16, REPEAT enables that gate so that U16 transmits the output on pin 9 of U9. U9 is connected as a two-stage shift register whose input (pin 2) is ground. It is clocked by clock 128 from U5.

U9 is initially set with output pins 5 and 9 high during the third count cycle by PKD minus 1. This is also the time when U12 outputs data. If the key is released, U9 clears to a low on pin 9 five count cycles following KEY. If the key is held down, U9 cannot shift since PKD minus 1 remains on preset input pins 4 and 10.

When REPEAT exists at pin 13 of U16, pin 11 of U16 is low to inhibit U25 and U27 at pin 13. This prevents further KEY signals and disables the n-key rollover circuitry. The low on pin 11 of U16 is also inverted by open collector inverter U24 to enable the repeat oscillator (timer U3, R4, R5 and C3). U3 generates a square wave on pin 3 with a period determined by the RC network.

This clocks the first stage of D flip-flop U11, the $\overline{\text{STROBE}}$ generator, and U11 produces the previously discussed 6 usec $\overline{\text{STROBE}}$. U11 continues to generate $\overline{\text{STROBE}}$ at the repeat oscillator rate until either the REPEAT or character key is released. And with each $\overline{\text{STROBE}}$, of course, the data associated with the character key is latched into U1 and U2.

Eight ASCII-coded data bits are output by U1 and U2 to J1 as indicated. Seven bits (Ø-6) are used for ASCII characters, and the eighth bit (7) is set only for certain control characters that are recognized by the Sol program. These are used for control functions such as MODE SELECT and cursor movement.

The remaining circuit, R32 and C14, initializes the keyboard when power is applied. That is, it resets the output latches and the SHIFT/SHIFT LOCK, UPPER CASE and LOCAL flip-flops. It also inhibits $\overline{\text{STROBE}}$ at pin 1 of NAND gate U10.

IX SOFTWARE

9.1 CONSOL

If you have SOLOS refer to the SOLOS/CUTERS USERS MANUAL, and ignore Section 9.1 and 9.2. CONSOL is a 1024 byte program designed to allow the Sol TERMINAL/COMPUTER to operate as a standard CRT terminal and to provide access to the essential computer capabilities of the Sol. This in addition to providing verification of correct system operation helps in finding errors in case of a malfunction.

In addition, CONSOL contains standardized entry points for all normal I/O operations. These routines are common with all Sol System Software allowing each personality module in the Sol line to interface with external programs in an almost identical manner.

A cassette read routine is also resident in the CONSOL module allowing Sol Software to be loaded and run in a system with additional memory. Sol System Software as of November 1976 includes BASIC, FOCAL, a Scientific Calculator and numerous "game" packages including a 8K assembly language version of STARTREK called TREK8Ø.

When power is applied to the Sol unit, CONSOL initializes the system RAM area, clears the screen, and enters the terminal mode.

In this mode the Sol System acts as a standard CRT terminal sending keyboard data to an output port and displaying received data on the screen. The COMMAND KEYS of the keyboard are not transmitted to the output port but are interpreted as direct internal operation keys. CURSOR MOVEMENT, HOME and CLEAR SCREEN all operate in this manner, while MODE SELECT causes an immediate change in the operation of the unit.

When the MODE key is depressed CONSOL issues a prompt character (>) and waits for a command line to be input. The Sol is now operating as a computer and is ready to accept one of the following commands:

DUmp	Dump memory locations to screen
ENter	Enter data to memory
EXecute	Execute a program in external memory
BAsic	Execute a program located at address zero
TErminal	Return to terminal mode
TLoad	Load program or data from cassette tape
MODE	Press MODE SELECT key to start new command line

Try using the commands as described in the following pages.

9.1.1 DUmp (addr) (addr)

The DUmp command displays memory data on the screen in a Hexidecimal representation. As with all Sol commands the command is recognized by the first two characters and up to ten additional characters can be input without an error being forced.

Thus, DU; DUST; DUMP; DUMPTHESE would all be recognized as being a DUmp command.

At least one address must follow the command or a error displaned on the screen. If two addresses are input then all values from the first address to the last will be displayed.

```
DUMP Ø          EF
```

Up to ten blanks may be inserted between each parameter without forcing an error condition. Errors are indicated by a question mark (?) replacing the character where the error occurred. For example if the DU command were given without an address the question mark would appear ten spaces to the right of the "U".

9.1.2 ENter addr

The ENter command places sequential bytes into memory beginning at the specified address. Data, represented as hexidecimal values, are input from the keyboard for entry to memory. All CONSOL commands except MODE SELECT are executed when the RETURN key is pressed. After the ENTER, (address), RETURN sequence the Sol Displays a colon (:) prompt character. Values are then input one line at a time with each line terminated by a carriage return or linefeed. The ENter function itself is terminated with a slash (/) and the Sol goes back to the command mode when the slash is encountered.

With all command functions of CONSOL, input lines are terminated with a carriage return or line feed. If the terminator is a C/R, CONSOL will erase all characters from the current cursor location to the end of the screen line. In this case, all valid input should be to the left of the cursor. If an error occurred during input the cursor may be moved to the left using the "cursor-left" key and the erroneous characters changed. A linefeed would then be used as a terminator since LF does not erase the line prior to processing the characters. This is particularly useful when using the ENter command since the input line can be visually scanned and errors corrected prior to the actual entry of input data to memory.

9.1.3 TLoad (speed)

Included within CONSOL are routines to read standardized cassette tape Software which is recorded with a sixteen byte header that includes NAME, LOAD INFORMATION, FILE TYPE and execute address. CONSOL, because of space limitations, is unable to search for a program or file by name. After receiving the TLoad command, CONSOL turns on the cassette player and waits for the next header, then uses the header information and loads the file to memory. The cassette recorder must be in play mode and properly connected before executing the TLoad command.

After loading the data, CONSOL returns to the command mode where the EXEC command can be used to execute the just loaded program. Also, a return can normally be made to the command mode by pressing the MODE SELECT key. Space limitations again limited escape during the header search, so if the system locks up in this routine the standard Sol restart must be used. To restart the Sol press UPPER CASE and REPEAT keys simultaneously.

The CUTS cassette interface electronics within the Sol will record or receive data at either of two standard speeds. TLoad will accept a parameter to select this speed, Ø being high speed and 1 being low. (1200 and 300 bits per second respectively). If no parameter is given CONSOL will default to high speed operation as all standard Processor Technology Sol-System Software is recorded at this speed.

9.1.4 EXecute addr

The execute command is used to run programs located in external memory. CONSOL branches to the external routine in a manner similar to an 8080 CALL instruction so the program can return to the command mode using a standard 8080 RET instruction if normal stack operations are used.

9.1.5 BAsic

The BAsic command is provided for executing programs whose starting address is Ø, such as Sol-BASIC5.

9.2 STANDARD I/O ROUTINES

All Sol System personality modules contain similar I/O code for input/output operations. CONSOL, using 1K of memory, has routines for KEYBOARD and SERIAL PORT input as well as Serial Communications Channel and VIDEO DISPLAY OUTPUT. Although the same code for SOLOS and SOLED contains expanded functions, the I/O operations appear almost identical when used with external software.

Sol-BASIC5, for example, performs all I/O using the jump table of the personality modules. Thus, without altering BASIC the user may output to either the serial port or to the display screen. Provision is also made within BASIC to programatically change to any of the four available Input or Output options. CONSOL is of course limited to the two provided.

M0100

SOLOS$^{(tm)}$/CUTER$^{(tm)}$

USER'S MANUAL

PROCESSOR TECHNOLOGY CORP.
6200 Hollis Street
Emeryville, CA 94608

(415) 652-8080

SOFTWARE TECHNOLOGY CORP.
P. O. Box 5260
San Mateo, CA 94402

(415) 349-8080

IMPORTANT NOTICE

SOFTWARE WARRANTY

Software Technology-Corporation warrants this Software Product to be free from defects in material and workmanship for a period of three months from the date of original purchase.

This warranty is made in lieu of any other warranty expressed or implied and is limited to repair or replacement, at the option of Software Technology Corporation, transportation and handling charges excluded.

To obtain service under the terms of this warranty, the defective part must be returned, along with a copy of the original bill of sale, to Software Technology Corporation within the warranty period.

The warranty herein extends only to the original purchaser and is not assignable or transferable and shall not apply to any software product which has been repaired by anyone other than Software Technology Corporation or which may have been subject to alterations, misuse, negligence, or accident, or any unit which may have had the name altered, defaced or removed.

PREFACE

This manual describes the use and operation of either SOLOS[(tm)] or CUTER[(tm)]. SOLOS is a program designed to be a personality module in a Sol[(tm)]. CUTER is a program designed to provide much of the power of SOLOS for the non-Sol user. Because SOLOS and CUTER have been designed to be compatible operating systems, this manual will refer to SOLOS meaning the SOLOS/CUTER operating system. The few differences between SOLOS and CUTER will be stated explicitly.

[(tm)] SOLOS, CUTER and Sol are trademarks of Processor Technology Corporation.

SOLOS/CUTER User's Manual

TABLE OF CONTENTS

TABLE OF CONTENTS (cont.)

I. INTRODUCTION

SOLOS is a 2048 byte program that configures the Sol-20 and one or two cassette tape recorders into a powerful, stand-alone computing system. SOLOS takes advantage of the Sol-20's built-in hardware peripherals and the 8080 instruction set to optimize the convenience and power of the inherent computer capabilities of the Sol.

Outstanding features of SOLOS include...

- **STANDARDIZED I/O SOFTWARE PROTOCOL** which makes all Sol-20 I/O (keyboard, display, serial, parallel and cassette) accessible to external programs from one entry point--a standard feature in all future Sol system software products that will require less memory than would normally be used for I/O routines.
- **SOFTWARE INTERFACE** permits user defined routines for custom applications.
- **"INDUSTRY STANDARD-SETTING" CASSETTE I/0 CONTROL** includes methods for loading and saving programs and commands that execute programs after automatic loading.
- **EXCLUSIVE CASSETTE I/0 ROUTINES** allow cassette files to be accessed on a byte-by-byte basis as though each file were a byte-by-byte device. Thus, data transfer to and from cassettes appears as normal I/O--and two cassettes can be used simultaneously to assemble and edit programs.
- **NEW DISPLAY CONTROL** features found only in expensive video terminals--including ESCAPE sequences for cursor positioning and character speed control.
- **19 COMMANDS** to access the basic requirements of the Sol-20 control cassette tape recorders and set up special conditions in SOLOS. (See the "Quick Command Reference List".)

Definition of Terms

In this manual:

addr means word address hexadecimal characters, (0-FFFF) range

data means hexadecimal characters, (0-FF) range

file means a collection of data

name means any one to five character identification for a file

port means a SOLOS pseudoport from 0 to 3

unit means a number of 1 or 2 corresponding to the appropriate tape recorder

( ) means optional parameters

INTRODUCTION (cont.)

Only the first two letters of the command expressions must be typed when entering a command expression. (The underscored letters in the following Quick Command Reference List.)

Quick Command Reference List

Console

COMMAND	FUNCTION
EXEC addr	Begin program execution at 'addr'
ENTR addr	Enter data into memory starting at 'addr'
DUMP addr1 (addr2)	Dump memory data, 'addr1' to 'addr2'
TERM (portin (portout))	Enter Terminal Mode
CUST name (addr)	Insert or remove a custom command

Tape

COMMAND	FUNCTION
GET (name(/unit) (addr))	Get a tape file into memory
SAVE name (/unit) addrl addr2 (addr3)	Save a file from memory to tape
XEQ (name(/unit) (addr))	Get then execute a tape file
CAT (/unit)	Catalog tape files

Set

COMMAND	FUNCTION
SET S=data	Screen character rate
SET I=port	Input port to SOLOS
SET O=port	Output port to SOLOS
SET N=data	Number of NULLS following CRLF
SET XEQ addr	Auto-execute addr
SET TAPE 0 or 1	0=1200 baud, 1=300 baud
SET TYPE data	Type 'byte' header
SET COUT addr	Custom output addr
SET CIN addr	Custom input addr
SET CRC data	Allows ignoring of tape CRC Read Errors

With a Sol, or CUTER on a Processor Technology GPM board, a power-on performs a reset which causes a SOLOS system reset. The Sol user may initiate this system reset anytime by simultaneously pressing the upper case and repeat keys.

A SOLOS system reset enters SOLOS into COMMAND mode. When in COMMAND mode, SOLOS will do a Carriage Return-Line Feed (CRLF) followed by a prompt (>). SOLOS then awaits the entry of a COMMAND. A COMMAND is processed upon receipt of a Carriage Return (CR). Pressing the MODE (or Control-@) key while awaiting a COMMAND causes the current COMMAND input line to be ignored and return to COMMAND mode. CUTER also resets the current I/O pseudo port selections to the system default.

The MODE (or Control-@) key is also used to abort the execution of most commands. This use of the MODE (or Control-@) key turns off both tape machines (if on) and returns to COMMAND mode.

II. CONSOLE COMMANDS

Console Commands in Brief

SOLOS has five console commands. They are:

Command	Function
EXEC addr	Begin program execution at 'addr'.
ENTR addr	Enter data into memory starting at 'addr'.
DUMP addrl (addr2)	Dump memory data, 'addr1' to 'addr2'.
TERM (portin (portout))	Enter Terminal Mode (available under SOLOS only)
CUST name (addr)	Insert or remove a custom command.

Console Commands in Detail

Execute Command EXEC addr

This command begins program execution at memory location specified by (addr).

Example: EXEC 200

Enter Command ENTR addr

Example: ENTR 500

```
: C3 00 01 1000: 05/
```

Result: Beginning at memory location 500, the following data was entered: C3 00 01. The new memory location of 1000: was selected to enter the data 51. The slash (/) terminated the ENTR command and returned to command mode.

Dump Command DUMP addr1 (addr2)

This command displays sequential memory data on the screen starting at location (addrl) and ending with (addr2).

Example: DUMP C02E C037

Result:

```
C02E E1 DB FA 2F E6 01 C8 DB FC C9
```

Dumped the SOLOS keyboard input routine. (See listing.) Starting at memory location C02E and ending at memory location C037.

Terminal Command TERM (port-I (port-O)) (Available under SOLOS only)

This command causes the Sol system to become a video terminal for connection to an external computer or modem. This command begins by automatically setting the I/O pseudo ports to the specified values. An omitted port parameter will be set to 1. Execution then proceeds by sending all Sol keyboard entries (except cursor control) to the specified Output pseudo port. Any input available from the Input pseudo port will be processed by the SOLOS display driver.

```
          Example:  TERM

          Result:   Keyboard data will be sent to the serial
                    port and all data from the serial port will
                    appear on the display screen.
```

Custom Command   CUST name (addr)   definition/removal

When a non-SOLOS command is entered, a separate table of custom commands (in RAM) will be searched. The CUST command is used to enter and remove up to six custom command names from the custom command table. (Only the first two letters of the name are significant.) When the name (2 to 5 letters) specified by the CUST command is not already in the custom command table, a new custom command will be entered into the table having an execute address as specified. When the addr is not specified, the beginning address of SOLOS will be used.

When the name specified on the CUST command already exists in the custom command table, this table entry will be replaced with an 'end-of-table' indicator. Therefore, not only will the specified name be removed, but any other custom command names following in the table will also be removed.

```
          Example:  CUST BASIC 0
                    CUST ALS8  E060

          Result:   Two new custom commands are now known.
                    ALS8 at location E060, and
                    BASIC at location 0.
```

III. TAPE COMMANDS

Tape commands are used to control the tape cassette recorders. In these commands, unit selection is optional, with a default selecting unit 1. When a unit is specified, however, it must be separated from the file identification name with a slash (/) and without spaces in between: e.g., TARGT/2.

Tape Header

At the start of each tape file is header information. This information includes the following data:

name: name of file, 5 ASCII characters or less

type: number is specified by user at time file is created

addr: starting address of file

size: number of data bytes in file

XEQ addr: auto-execute address word (See Set Commands - Section IV)

Error Messages

Cassette error messages are printed in this format:

"ERROR (name) (type) (addr) (size)"

Reasons for an error message are:

1. bad read of file (tape error or CRC ERROR)

2. MODE (or Control-@) key used for escaping while reading a tape file

3. XEQ command given to a non-executable file.

Tape Commands in Brief

SOLOS has four tape commands. They are:

GET (name (/unit) (addr))	Get a file from tape to memory
SAVE name (/unit) addr1 addr2 (addr3)	Save file
XEQ (name (/unit) (addr))	Get, then execute, a file
CAT (/unit)	Catalog of tape files

III. TAPE COMMANDS (cont.)

Tape Commands in Detail

Get a file from tape   GET (name(/unit) (addr))

This command transfers the specified or next tape file into memory. If a (name/unit) is given, this command will search forward on the cassette until that file is found. The (addr) parameter, if given, specifies the memory location at which the file will be loaded. If the addr is omitted, the file will be loaded as specified in the header.

Example: GET TARGT/2

Result: Gets the program WARM from tape unit #2 into memory as specified by the tape file header information. Returns to SOLOS command mode.

Get, then Execute   XEQ (name(/unit) (addr))

This command is an extension of the GET command which gets a tape file and executes as specified by the header information. The (/unit) and (addr) are optional and operate the same as with the GET command.

Example: XEQ FOCAL

Result: Gets, then executes, a program named "FOCAL" from tape unit 1.

Save a file   SAVE name (/unit) addr1 addr2 (addr3)

This command transfers program or data onto a tape cassette file name (name) starting at (addr1) and ending at (addr2). The name of the file becomes part of the tape's header information. SET TYPE and SET XEQ commands affect the header information on the tape file. The optional addr3 specifies the address (if different than addr1) to be entered in the tape header.

Example: SAVE CHASE/2 0 1FF

Result: Saves onto tape unit 2 a program named "CHASE" starting at location 0000 and ending at location 1FF.

Catalog of files   CAT (/unit)

This command will start the tape unit specified and list each tape file header information.

Example: CAT /2

Result:

```
SLOPE 0500 0200
HUM   0500 0B00
```

Note: A very useful feature of the CAT command is to apply power to the tape units when needed to rewind tape. Depressing the MODE (or Control-@) key will remove power from tape unit and return to COMMAND mode.

IV. SET COMMANDS

SOLOS has 10 set commands. They are:

SET S=data	Screen character rate
SET I=port	Input port to SOLOS
SET O=port	Output port to SOLOS
SET N=data	Number of NULLS following CRLF
SET XEQ addr	Auto-execute addr
SET TAPE 0 or 1	0=1200 baud, 1=300 baud
SET TYPE data	Type 'byte' header
SET COUT addr	Custom output addr
SET CIN addr	Custom input addr
SET CRC data	Allows ignoring of tape CRC Read errors

Set Commands In Detail

Set Speed of Display   SET S=0-FF

This command determines character display rate to the screen:

data = 0 – Fastest

data = FF – Slowest

Input/Output Command Parameters

The next two SET commands affect SOLOS input and output command parameters.

Set Out Command   SET O=port

This command selects the output driver routine to which SOLOS routes data. Under SOLOS, COMMAND mode text is always sent to the display screen. Under CUTER, all output goes to the current Output pseudo port. In all cases, the output from each command is sent to the current output pseudo port.

The Output Pseudo ports command parameter values are:

0 = Video Display

1 = Serial Output Port

2 = Parallel Output Port

3 = User Defined by SET COUT command

Example: SET O=1
DUMP 0 2F

Result: Select serial output port. 'Dump 0 2F' would be displayed, but the data would go to the serial output port.

Set In Command SET I=port

This command selects the input driver routine to SOLOS. All future input commands would come from the new selected input pseudo port.

The Input Pseudo port parameter values are:

0 = Keyboard

1 = Serial Input Port

2 = Parallel Input Port

3 = User defined by SET CIN command

Example: SET I=1

Result: SOLOS would expect the next command to come from the serial port input routine. The Sol keyboard would have no affect except to simultaneously hit repeat and upper case keys to reset the computer.

Cassette Tape Parameter Commands

The Following SET commands affect the cassette tape parameters:

Set Tape Command SET TAPE 0 or 1

This command selects one of two standard speeds.

0 = 1200 baud high speed

1 = 300 baud low speed

Normally set to 0.

IV. SET COMMANDS (cont.)

Set Type Command SET TYPE data

This command sets (data) values into the 'type' byte in the tape header information when used in conjunction with the SAVE command. The 'type' byte data is entered as a hexadecimal value, but it will appear on the screen as an ASCII character when displayed by the GET or CAT command. Only displayable characters should be used for type values (data). The most significant bit of the type value determines if the tape file can be executed automatically by an XEQ command. (0 = Auto-execute, 1 = Not executable.) Typing of tape files can be very useful in grouping common files.

Example: SET TYPE 47

47 = 'G' character for GAME FILES
Sign Bit = 0, auto-execute

SET TYPE 50

50 = 'P' character for PROGRAM FILES
Sign Bit = 0, auto-execute

SET TYPE C4
C4 = 'D' character for DATA FILES
Sign Bit = 1, non-execute

Set Execute Command SET XEQ addr

This command sets the auto-execute address (addr) word into the tape header information when used in conjunction with the SAVE command. This address word is used by the XEQ command after loading a tape file to begin program execution at location specified by tape header information (addr). Note that the 'TYPE' byte determines if the file is of the auto-execute type.

Example: SET XEQ 200

Result: The auto-execute address of 200 Hex will be written onto the tape header when the next SAVE command is issued.

Custom Input/Output Commands

The next SET commands set address pointers to custom input and output driver routines when 'SET I=3' and/or 'SET O=3' are used. These custom I/O drivers must meet the SOLOS I/O drivers requirements. See the SOLOS software listing for model input routine.

Set Custom Output Command SET COUT addr

This command informs SOLOS software where the user defined output routine specified by 'addr' is located.

The Custom Output driver requirements are:

1. The 'addr' (address) word in the SET COUT command will equal the starting address of the output routine.

2. It is the user's responsibility to save registers prior to any modification of the register.

3. The "B" register will contain the data passed from SOLOS for output routine.

4. The output routine will end with a 'RET' instruction or equivalent.

Set Custom Input Command SET CIN addr

This command informs SOLOS software where the user defined input routine specified by 'addr' is located.

The Custom Input driver requirements are:

1. The 'addr' address word in the SET CIN command will equal the starting address of the input routine.

2. It is the user's responsibility to save registers prior to any modification of the register.

3. The input routine combines actually inputting the character along with STATUS. The routine returns either a zero flag indicating no character is available or the character in Register "A" with a non-zero flag. The calling program can then take appropriate action based on a zero or non-zero condition.

Set CRC Error Checking SET CRC data

This command is used to specify whether or not the standard CRC error checking routines are to be used. When a value of FF is specified, all further tape reads will ignore CRC errors. Any value other than FF indicates standard error checking is to be in effect. This command is very useful to allow a tape to be read in which would otherwise not be readable. When CRC errors are being ignored, it must be remembered that the data read in may not be valid.

Example: SET CRC FF

Result: CRC error checking will be set to ignore all CRC errors.

Set Number of NULLS SET N=data

This command sets the number of nulls (binary zeroes) to be output following a carriage return-linefeed (CRLF) sequence. The value is initialized to zero but may be set to any number up to FF (hex). This command is useful when using output devices requiring a delay following a carriage return.

Example: SET N=3

Result: Every CRLF issued by SOLOS will be followed by three nulls.

V. SUBROUTINES

A. Introduction to the SOLOS Machine Language Interface

The Machine Language Interface with SOLOS is based on:

1. A predefined set of 'pseudo' I/O ports allowing software compatibility as well as providing an easy means of supporting any I/O device.

2. A system defined register usage when interfacing with SOLOS.

3. A system jump table of entry points.

First are the pseudo ports. Built into SOLOS are four input and four output pseudo ports. I/O requests made to a pseudo port are converted internally to a request either to a specific device, a built-in routine, or a user written routine. All non-tape I/O requests made to SOLOS are made with reference to one of the following pseudo ports.

PSEUDO PORTS FOR SOLOS

Pseudo Port	Input	Output
0	Keyboard	VDM driver
1	Serial port	Serial port
2	Parallel Port	Parallel Port
3	User written routine	User written routine

PSEUDO PORTS FOR CUTER

Pseudo Port	Input	Output
0	Keyboard data from parallel port 3,not KDR status, on port 0; bit 0.	VDM driver
1	Serial port 1, RDA status on port 0, bit 6.	Serial port 1, TBE status on port 0, bit 7.
2	Parallel port 2 with not-PDR status on port 0, bit 2.	Parallel port 2 with not-PXDR status on port 0, bit 1.
3	User written routine.	User written routine.

Second are the defined register usages when interfacing at the machine language level with SOLOS.

Whenever a machine program is executed by SOLOS (via the EXEC or XEQ command, or via a custom command), the stack pointer and HL registers are predefined by SOLOS **>**. The stack pointer is set such that the user may perform stacking operations which will use the SOLOS stack. The SOLOS stack begins at the end of the SOLOS RAM area and works its way down from there. Excessive use of this stack can destroy data maintained by SOLOS within its RAM area. The stack is also prepared so that the user may issue a standard RET instruction to return control to SOLOS command mode processor.

The HL register pair is initialized to point to the very beginning of SOLOS. It is at this point that the SOLOS jump table begins. The user program may then use the address presented in the HL register pair as the beginning of the jump table.

This address is provided for two reasons:

1. CUTER may be located at any address in memory, providing the means for programs to function with CUTER located at any address, and

2. the first byte of the jump table for SOLOS is different from the first byte for CUTER, providing an easy means of distinguishing between SOLOS and CUTER.

Third is the SOLOS jump table (see next page). All requests to SOLOS should be made based on this jump table and not to the actual routine addresses as scattered throughout SOLOS. By using only this jump table, the user can be assured of maintaining compatibility between SOLOS and CUTER.

JUMP TABLE

Address	Label	Length	Function
C000	START	1	This byte allows power-on reset of SOLOS. It is 00 for SOLOS and 7F for CUTER, providing an easy means of differentiating the exact operating system in use.
C001	INIT	3	This is a "JMP" to the power-on reset.
C004	RETRN	3	Enter at this point to return control to SOLOS command mode processor.
C007	FOPEN	3	Enter here to open a tape file.
C00A	FCLOS	3	Enter here to close a tape file.
C00D	RDBYT	3	Enter here to read a byte from an open tape file.
C010	WRBYT	3	Enter here to write a byte to an open tape file.
C013	RDBLK	3	Enter here to read one tape block into memory based on a header.
C016	WRBLK	3	Enter here to write one tape block from memory based on a header.
C019	SOUT	3	Enter here to output the character in register "B" to the current system output pseudo port. This is always an "LDA" pointing to the byte containing the current system output pseudo port value.
C01C	AOUT	3	Enter here to output the character in register "B" to the pseudo port specified in register "A".
C01F	SINP	3	Enter here to obtain status/character from the current system input pseudo port into register "A". This is always an "LDA" to the byte containing the current system input pseudo port value.
C022	AINP	3	Enter here to obtain status/character from the input pseudo port specified in the "A" register. On return, register "A" will contain the character with the flags set to indicate whether a character is present or not.

V. SUBROUTINES (cont.)

B. System Entry Points

There are actually only two system entry points within the SOLOS jump table. Entry at these points does not require that any register be initialized. The first (at either label "START" or "INIT") is used to perform a complete power-on system reset. As a part of the system reset, the system RAM area data used by SOLOS will be cleared. The only reason for entering via "START" or "INIT" is that the power-on circuitry requires a one byte instruction to allow various circuits to stabilize. The other use of the byte labeled "START" is to determine if a user program is being executed under SOLOS or is CUTER controlled. When under SOLOS, this byte will be zero. When under CUTER, this byte will be non-zero.

The other system entry point ("RETRN") is used to return to SOLOS command mode. This entry point does not perform a system reset.

C. SOLOS Input Entry Points

SINP   entry point address C01F

This entry point will set register "A" to the current system input pseudo port. The current system input pseudo port is changed by the "SET I=" command. After setting register "A", this command proceeds by executing an "AINP". (See below.)

AINP   entry point address C022

This entry point is used to input one character or status from any pseudo port. Register "A" on entry indicates the desired input pseudo port from 0 to 3. Because this entry point is a combination status/get-character routine, it is the user's responsibility to interpret return flags properly. When a character is not available, the zero flag will be reset and the character will be placed into register "A". What this means is that, if the user wants to wait for a character to be entered, simply follow the CALL AINP (or SINP) with a "JZ" jump-if-zero instruction back to the call. A combined status/get-character routine is very important when allowing user written input routines.

D. SOLOS Output Entry Points

SOUT   entry point address C019

This entry point will set register "A" to the current system out-put pseudo port. The current system output pseudo port is changed by using the "SET O=" command. After setting register "A", this command proceeds by executing an "AOUT". (See next definition.)

AOUT   entry point address C01C

This entry point is used to output one character to any pseudo port. Register "A" is assumed to be a binary value from 0 to 3 indicating the desired output pseudo port. Register "B" will contain the character to be output. On return, the PSW and Register "A" are undefined. All other registers are as they were on entry.

E. SOLOS VDM Display Driver

Because the VDM is much more powerful than a standard hardcopy device, the built-in VDM driver supports many expanded functions. The following characters, when sent to the VDM driver (output pseudo port 0), cause special functions to be performed:

Hex	Character		Function
01	Control-A	(SOH)	Move cursor left (wrap mode) one position.
0B	Control-K	(VT)	Clear screen; position cursor at home.
0D	Control-M	(CR)	Clear remainder of line; then move cursor to beginning of same line.
13	Control-S	(DC3)	Move cursor right (wrap mode) one position.
17	Control-W	(ETB)	Move cursor up (wrap mode) one line.
1A	Control-Z	(SUB)	Move cursor down (wrap mode) one line.

The escape key (hex code 1B) is also a special character to the VDM driver. It initiates what is known as an escape sequence. The escape character is always followed by one or two hexadecimal values (bytes) which indicate what expanded function is to be performed. The following lists the escape sequences and corresponding results. Where a third byte must follow the escape, this will be represented by (##), indicating that this third byte actually contains a value being passed to the VDM driver.

Escape sequence	Function
1B 01 ##	Place the cursor onto position (##) of the current display line. (##) is in the range 00 - 3F.
1B 02 ##	Place the cursor onto line number (##) of the display screen. (##) is in the range 00 - 0F, with the topmost line being line 00.
1B 03	Pass back the current cursor line/character position in Registers BC. Register "B" is set to the character position (00-3F), and Register "C" is set to the line position (00-0F).
1B 04	Pass back the memory address of the current cursor location into Registers "BC".

more escape sequences . . .

Escape sequence	Function
1B 05 ## 1B 06 ## 1B 07 ##	The third byte is output to the VDM at the current cursor position exactly as is, regardless of this byte's value. No check is made of this character (##). Being a control character, it is only placed into the VDM memory as-is, and the cursor is advanced one position.
1B 08 ##	The display speed is set to the value (##) specified. The speed ranges from 00 (fastest) to FF (slowest).
1B 09 ##	This functions the same as escape sequence 01. The cursor is positioned to character position ## of the current display line.

F. Cassette Tape Entry Points to SOLOS

SOLOS contains subroutines to handle data transfer to and from two cassette units. Both block-by-block and byte-by-byte access are available. While performing any tape read, the user can return to the present calling software program by pressing the MODE (or Control-@) key.

In block transfers, each request results in tape movement and a transfer of an information block to or from a location in memory. SOLOS uses block-by-block access to provide the tape commands.

In byte transfers, on the other hand, SOLOS buffers the data into 256 byte blocks, doing cassette operations only once per 256 transfers. BASIC uses byte-by-byte access for data files. Other programs--such as editors, assemblers or special user-written programs--can also call the byte-by-byte routines if a few specific conventions and calling sequences are followed.

File Header

The file header for SOLOS provides specific attributes to a file. These attributes consist of a five ASCII character name and a file type.

File name serves two functions:

1. It permits easy human identification of the file, and

2. It provides the identification for which SOLOS searches to find the correct file.

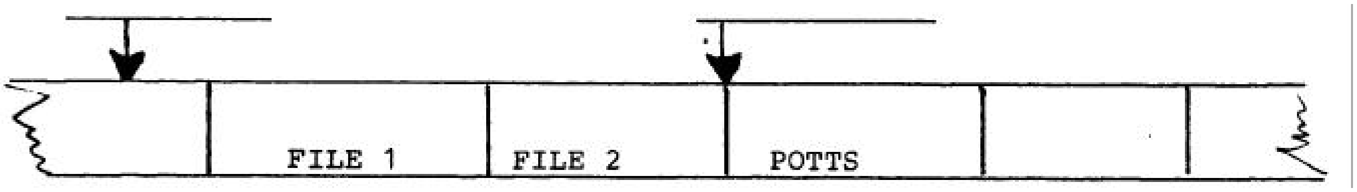
File type is used in SOLOS to prevent certain operations, such as automatic XEQ, if the file is not of the proper type.

When calling open the register, pair "HL" should point to a memory location that contains the header. Following is the layout of a SOLOS file header:

```
NAME ASC  '12345'   A five character name with trailing binary zeroes.

     DB   0         Should always be zero.

TYPE DB   'B'+80H   File type. If Bit 7=1, then this is a data file
                    (not executable).

SIZE DW   LENGTH    Length of file in number of bytes.

ADDR DW   FROM      Address at which file is to be read to or from
                    which it is to be written.

XEQ  DW   EXEC      Auto execute address (ignored for data files).

     DS   3         Space - not currently used by SOLOS.
```

As previously mentioned, SOLOS uses the name to find the correct data for the file operations. Assume you were about to read data from a file named POTTS, for example, and you had correctly opened the file with a header pointing to that name. SOLOS, when you first requested a data transfer, would read past File 1 and File 2 (as shown below) and then read data from the POTTS file.

Beginning position of tape (current position)   Beginning of file to be read



Block Access

The Block Access method invokes no management by the system. Each 'call' to the 'Read' or 'Write' routines performs a complete cassette operation. Read and Write routines are used by SOLOS for GET and SAVE commands and serve as examples of the calling conventions for RDBLK and WRBLK routines.

Read Tape Block Routine   RDBLK

The entry point for RDBLK is C013.

On entry: Register A contains Unit and Speed data with bit 5 (speed) 0 for 1200 baud (or 1 for 3$0 baud); bit 7=1 for Tape 1; bit 6=1 for Tape 2; and all other bits=0.

Registers H & L contain the address of file header information.

Registers D & E contain the address-of where the file is to be loaded into memory. (If set to 0, this information is taken from file header information on tape.)

On exit: Normal return: Carry Flag is cleared, and data has been transferred into memory.

Error return: On errors, or user pressing MODE (or Control-@) from keyboard, the Carry Flag is set.

Write Tape Block Routine WRBLK

The entry point for WRBLK is C016.

On entry: Register A contains unit and speed with the same bit values as specified for RDBLK.

Registers H & L contain file header address. The file header information will be written onto the specified tape unit followed by the data.

On exit: Normal return: Carry Flag is-cleared, and data has been transferred to tape.
There are no error returns.

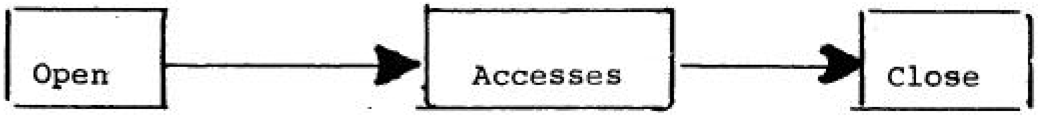
Byte Access

Data stored on, or about to be stored on, a tape should be considered a file. In a SOLOS file, data is stored one byte at a time as a string of bytes along the tape with no assumed meaning or structure. It is simply a collection of bytes that can be accessed by someone with responsibility for the intelligence of the data.

When writing to tape, SOLOS records the data in a form that allows the data to be read from the tape later. When reading from tape, SOLOS provides the management to access each byte sequentially.

SOLOS also provides start and stop control of two units. File operations view unit 1 as File 1 and Unit 2 as File 2. Thus, data in Unit 1 is associated with File 1, and data in Unit 2 is associated with File 2.

When using Byte Access, two important user management operations are necessary. As shown in Figure below, the first is to open a file to tell SOLOS you want to access the file. The second is to close a file to inform SOLOS you are finished with it.



SOLOS provides entry points to Open, Read, Write and Close tape files. Each of these routines requires that certain conventions be followed to ensure accurate data transfers.

File Open Routine FOPEN

The Open routine sets up certain internal parameters to keep track of data requests. This operation should be called only once prior to the first access of the file. The File Header information is the same format as in the Block Access mode and is used in both reading and writing of files. If the Byte Accesses are of the Read type, SOLOS will search the tape file until the correct file 'name' is found as specified by the File Header information. On the next Read access, SOLOS will transfer the first data byte of the file. If the Byte Accesses are of the Write type, the File Header information will be transferred onto the file.

The entry point for FOPEN is C007.

On entry: Register A contains File # (1 or 2) same as tape unit (1 or 2).

Registers H & L contain address of the File Header information.

On exit: Normal return: All registers are altered and file is ready for accesses.

Error return: The Carry Flag is set. Reason for error: file already open.

Write Byte Routine WRBYT

The Write Byte routine writes a single byte of data into a buffer file. SOLOS stores this data until it contains 256 bytes. It then writes this block onto the tape, followed by a CRC character (error checking character). SOLOS then resets the buffer file for the next 256 bytes of data.

The entry point for WRBYT is C010.

On entry: Register A contains File # (1 or 2).

Register B contains the byte of data to be transferred onto tape.

On exit: Normal return: Carry Flag cleared.
Error return: Carry Flag set - errors caused by:

1. file NOT open, or
2. file previously used for reading.

Read Byte Routine RDBYT

The Read Byte routine reads a single byte of data from a buffer file. SOLOS fills this buffer as needed per read request. Each time SOLOS fills the file buffer (reads a block), the CRC character is checked for data accuracy.

The entry point for RDBYT is C00D.

On entry: Register contains file # (1 or 2)

On exit: Normal return: Register A contains data byte. Carry and Minus Flags set mean 'end of file'.

Error return: Carry Flag set. Errors are caused by:

1. file NOT open
2. file previously used for writing
3. CRC character error
4. pressing MODE (or Control-@) while actually reading from the tape.

Close File Routine FCLOS

The Close file routine closes the current file and resets the internal parameters for the next open operation. It is very important to close the file after all data transfers are completed. Failure to do so could result in lost data and prevent further open operations.

The entry point for FCLOS is C00A.

On entry: Register A contains File # (1 or 2) to be closed.

On exit: Normal return: Carry Flag cleared.

Error return: Carry Flag set. (Error is caused by file NOT open.)

VI. LOADING & EXECUTING CUTER (Applicable to CUTER only)

CUTER is available (1) on cassette tape with its own loader which can be loaded at any memory address from 0200 through F400, or (2) in ROM at the address C000. In order to load CUTER from cassette tape, perform the following steps. When CUTER is being used in ROM, the procedure is much simpler: make sure the sense switches are set according to H below prior to executing location C000.

A. Verify that the hardware is connected and functioning properly.

B. Enter the following bootstrap routine into memory beginning at location 0. The following is presented in a format similar to that produced by a "DUMP" command with an address shown every 10 (hex) bytes:

```
0000: 21 40 00 F9  45 4D 3E 80  D3 FA E7 05  C2 0A 00 E7
0010: 3D C2 0F 00  E7 02 03 FE  DD C2 14 00  E9 00 00 00
0020: DB FA A5 CA  20 00 DB FB  C9
```

C. Verify that the above bootstrap is in memory exactly as presented.

D. Set the sense switches to the address at which CUTER is to be loaded. The sense switches will be the hi-order byte of the memory address, with the lo-order byte zero. As an example: Sense switches set to 34 hex will cause CUTER to be loaded into memory beginning at location 3400 hex. For convenience, a memory address should be selected that also specifies the default I/O pseudo ports (see "H" below). The address specified must be between 0200 and F400. Remember, however, that CUTER occupies 2K of memory and uses 1K of RAM beyond that.

E. Make sure that the CUTER tape is rewound and placed into the proper cassette machine. The CUTER bootstrap will activate the motor control for tape unit one. If your cassette machine motor control is attached as tape unit one, you may now place the machine into "PLAY" mode.

F. Execute location zero (the bootstrap). This can be accomplished by allowing a "Reset" to specify an address of zero. At this time, be certain that the cassette machine is in "PLAY" mode and is activated:

G. When completed, the CUTER loader program will "HALT". This is not an error condition. When completed, the motor control will also be turned off.

VI. LOADING & EXECUTING CUTER (cont.) (Applicable to CUTER only)

H. Via sense switches, select the default I/O pseudo ports as follows:

	X X X X	I I O O
Bit	7 6 5 4	3 2 1 0

Where: X X X X doesn't matter

I I which pseudo port from 0 - 3 (00-11 binary) is to be the default input pseudo port.

O O which pseudo port from 0 - 3 (00-11 binary) is to be the default Output pseudo port.

NOTE: Whenever CUTER does a full system reset (begins execution at its beginning memory address), the sense switches will be accessed to determine the default I/O pseudo ports.

I. If either Input or Output default is to be pseudo port 3 (user written routine), verify the following:

(i) The appropriate user written routine is in memory.

(ii) The address of the appropriate I/O routine is entered into the CUTER system RAM area. The system RAM area begins exactly 2K (800 hex) after the beginning of CUTER. The first word of this area is used to contain the address for the user Input routine. The second word will contain the address of the user Output routine. Addresses are entered in lo-hi order.

J. Execute location ZERO. The CUTER loader will have properly prepared this location to either transfer control to the CUTER just loaded or to indicate an error while loading CUTER. If there was no error, CUTER will now be in control.

Remember to turn off the cassette machine and remove the CUTER tape.

K. IF your computer halts again, this means one of the following errors has occurred. Display memory location ONE to determine the error code. The error code will be one of the following:

Error Code in Hex	Meaning
00	The specified load address was not within the range 0200-F400, or the tape file loaded was not CUTER.
01	A tape read error was detected.
02	There was no tape read error, but the CRC (error checking) character was invalid.
40	The file was loaded, but it was not CUTER.

M0101

BASIC/5

User's Manual

For Use with

SOLOS, CUTER and CONSOL

PROCESSOR TECHNOLOGY CORP.
6200 Hollis Street
Emeryville, CA 94608

(415) 652-8080

SOFTWARE TECHNOLOGY CORP.
P. O. Box 5260
San Mateo, CA 94402

(415) 349-8080

IMPORTANT NOTICE

This copyrighted software product is distributed on an individual sale basis for the personal use of the original purchaser only. No license is granted herein to copy, duplicate, sell or otherwise distribute to any other person, firm or entity. This software product is copyrighted and all rights are reserved.

SOFTWARE WARRANTY

Software Technology Corporation warrants this software product to be free from defects in material and workmanship for a period of three months from the date of original purchase.

This warranty is made in lieu of any other warranty expressed or implied and is limited to repair or replacement, at the option of Software Technology Corporation, transportation and handling charges excluded.

To obtain service under the terms of this warranty, the defective part must be returned, along with a copy of the original bill of sale, to Software Technology Corporation within the warranty period.

The warranty herein extends only to the original purchaser and is not assignable or transferable and shall not apply to any software product which has been repaired by anyone other than Software Technology Corporation or which may have been subject to alterations, misuse, negligence, or accident, or any unit which may have had the name altered, defaced or removed.

Sol BASIC/5 USER'S MANUAL

Table of Contents

Table of Contents (cont.)

APPENDICES

I. INTRODUCTION

BASIC (Beginner's All-purpose Symbolic Instruction Code) is a computer programming language characterized by versatility and ease of use. Its resemblance to standard mathematical notation and simple English statements enables novices and professionals to program solutions to a variety of problems in the shortest possible time.

BASIC is a conversational language which permits a user to sit down at his computer or terminal device and engage in a dialog with it. The results may be either immediate answers to a mathematical problem or a working computer program which may be used in the future to process new data.

There are many good books available to instruct the user in how to program in BASIC; therefore, no attempt has been made to teach BASIC in this manual. Appendix F lists several references that may be of interest.

Here, we present only Processor Technology's Sol BASIC programming language, its features and restrictions. One of the best ways to learn Sol BASIC is to experiment with your system.

Sol BASIC features include...

- Full 8-digit precision
- Multiple statement entry on one line
- BCD (Binary Coded Decimal) arithmetic for maximum accuracy in all mathematical operations
- User formatting of output data
- Program storage on, and retrieval from, cassette tape
- Data files for processing and saving numeric data
- Implementation of many function subprograms
- Execution of most program statements in direct mode for immediate calculations and enhanced debugging
- Only 8K byte memory needed to run many programs
- ARG and CALL functions facilitate linkage to 8080 machine language program segments
- Unique line editor using video display and SOLOS
- Fully formatted listings with automatic FOR-NEXT indentation.

I. INTRODUCTION (cont.)

Definition of Terms

In this manual--

exp	means	mathematical expression
num	means	any number
rel exp	means	relational expression
statement n	means	statement number
"string"	means	uninterrupted series of literal alphanumeric characters enclosed with quotation marks
var	means	variable name
( )	means	optional

Sol BASIC is a program for use with SOLOS, CONSOL and CUTER with at least 8K bytes of memory available for use by Sol BASIC. Because SOLOS and CUTER operating systems are compatible, this manual will refer to SOLOS meaning either SOLOS or CUTER. Some features described herein require that the current system output pseudo port be zero (the VDM driver of SOLOS/CUTER).

II. PROGRAM STRUCTURE

A Sol BASIC program is comprised of statements. Every statement begins with a statement number, followed by the statement body, and terminated by a CR (carriage return), line feed, or a colon in the case of multiple statements.

There are four types of statements in BASIC: declarations, assignments, control and input/output. These statement types are described in corresponding sections of this manual.

A. Statements

General statement requirements for the Sol BASIC program are as follows:

1. Every statement must have a statement number ranging between 1 and 65000. Statement numbers are used by Sol BASIC to order the program statements sequentially.

2. In any program, a statement number can be used only once.

3. Statements need not be entered in numerical order because Sol BASIC will automatically arrange them in ascending order.

4. A statement may contain no more than 80 characters including blanks.

5. Blanks, unless within a character string and enclosed by quotation marks, are not processed by Sol BASIC. BASIC removes all excess blanks as the line is processed into the file so that minimum memory area is used. When listing or editing, it automatically inserts blanks to make the line more readable.

Example:

```
G O T O  5 0 0
```

is exactly the same as

```
GOTO500
```

6. More than one statement can be input on a line if separated by a colon, but only one statement number is allowed.

Example:

```
520  LET A=1: B=3.2: C=5E2
```

B. Data Format

The range of numbers that can be represented in this version of Sol BASIC is: ±.99999999E±127.

There are eight digits of significance in this version of Sol BASIC. Numbers are internally truncated on entry to fit this precision.

Numbers may be entered and displayed in three formats: integer, decimal, and exponential.

Example: 153,   34.52,   136E-2

C. Variable Names

Variables may be named any single alphabetic character or any alphabetic character followed by a single numerical digit, e.g., A, B5, X, D1.

D. REM Statement

The REM, or remark statement, is a non-executable statement which has been provided for the purpose of documenting program listings. By generous use of REM statements, a complex program may be more easily understood. REM statements are only reproduced on the program listing. They are not executed. If control is given to a REM statement, it will perform no operation. (It does, however, take a finite amount of time to process the REM statement.)

CAUTION: A REM statement cannot be terminated by a colon with statements following on the same line.

Example: 150 REM NOW HOW: LET R1=3.5E2.1

The assignment statement ("LET" above) will never be executed. The entire line is considered to be a non-executable comment.

III. PROGRAM PREPARATION

After Sol BASIC is loaded into your system, it may be started at memory address 0. (Refer to Appendix A for a complete description.) Sol BASIC is initialized to support only 8K of memory. The amount of memory to be supported may be increased by using the "SET" command.

A return to SOLOS or CONSOL can be made by giving the BYE command. Sol BASIC can then be re-entered, leaving the existing program intact, by executing location zero.

The system is then ready to accept commands or statements. The user might enter the following program, for example:

```
    150  REM PROGRAM TO DEMO
    160  PRINT"ENTER SOME DATA",
    170  INPUT B5
    180  LET P7=B5+3/2
    185  PRINT
    190  PRINT B5,P7
    200  END
```

A. Inserting a Statement

If the user wishes to insert a statement between two others, he needs only to type a statement number that falls between the other two. For example:

```
    181  REM NOW FOLLOWS THE LET STATEMENT
```

B. Replacing a Statement

If it is desired to replace a statement, a new statement is typed that has the same number as the one to be replaced. For example:

```
    180  LET P7=SIN(B5)
```

replaces the previous LET statement.

C. Terminating a Line

Each line entered is terminated by a carriage return or line feed, Sol BASIC positions the print unit to the correct position on the next line.

D. Editing

The MODE key may be used to erase a character or a line that was typed in error. Also, all editing functions--as outlined in the EDIT command description--are fully functional during normal input.

IV. COMMANDS

It is possible to communicate with Sol BASIC by typing direct commands at the terminal device. Also, certain other statements can be directly executed when they are given without statement numbers. See Calculator Mode in the next Section for more information.

Commands have the effect of causing Sol BASIC to take immediate action. A Sol BASIC language program, by contrast, is first entered into the memory and then executed later when the RUN command is given.

When Sol BASIC is first ready to receive a command, the word READY is displayed on the terminal device.

Commands are typed without statement numbers. After a command has been executed, the user will either be prompted for more information, or READY will again be displayed indicating that BASIC is ready for more input--either another command or additional program statements.

CLEAR COMMAND - CLEAR

Sets all variables to zero, resets the READ pointer and initializes the program so that it may be run. CLEAR may be used as a statement in programs that exit FOR TO loops or GOSUB in a non-standard fashion. The RUN command produces an automatic clear.

LIST   COMMAND - LIST (statement n)

Causes all the statements of the current program to be displayed on the user's terminal. The lines are listed in increasing numerical order by statement number. The display will begin with statement n, if given.

RUN COMMAND - RUN

Causes the current program to begin execution at the first statement number. RUN always begins at the lowest statement number. RUN resets the DATA pointer and performs a CLEAR.

New Program COMMAND - NEW

The NEW command causes working storage and all variables and pointers to be reset. The effect of this command is to erase all traces of the program from memory and to start over.

HALT   COMMAND - MODE (or CTL-@ Key)

This key on the terminal console will cause BASIC to halt its current operation and to respond with a READY. BASIC will then accept further commands. This command is often used to stop a LIST command before it has completed or to halt the execution of a program.

LINE CLEAR COMMAND - MODE (or CTL-@) Key

Clear the current line buffer. If the user types a line at the terminal and decides that the line is in error and should be deleted, depression of the MODE (or CTL-@) key will clear the line.

CHARACTER ERASE COMMAND - DEL Key

Single character erase. If a character is determined to have been typed in error, it may be erased by striking the "DEL" key and then entering the correct character. See EDIT command for further explanation.

MULTIPLE STATEMENT PER LINE COMMAND - :

The use of colons provides the ability to enter more than one statement on a line. Each statement must be separated by a colon, and the total number of characters may not exceed the line length of 80 characters. There may be only one statement number on a line. Therefore, one cannot transfer control to any of the appended statements except by the natural program flow.

```
    Example:  150 LET A=A+A:B=2*A: IF A=6 THEN PRINT B
```

SET COMMANDS

The SET command is used to specify various system options. The command is always followed by two operands. The first operand specifies the option being selected, and the second is the new value to be associated with that option. For example, SET S=1 means that the speed is to be set to the value of "one". The following operands are allowed via a SET command. (Note that some operands are allowable as both direct execution statements and statements of a program.)

SET I=exp   Direct or Program Statement

This command sets the current system input to be the indicated Sol pseudo port (of the range 0 - 3). Once processed, all further system input will come from the specified device and not from the system keyboard unless the keyboard (pseudo port 0) is specified.

SET O=exp   Direct or Program Statement

This command sets the current system output to be the indicated Sol pseudo port (of the range 0 - 3). Once processed, all further system output will be directed to the specified device and not to the display unless the display (pseudo port 0) is specified.

SET S=exp   Direct or Program Statement   *SOLOS only*

This command specifies the speed to be used by the Sol display driver. This is the speed at which the driver will display lines on the screen. The value of the expression must be from the range 0 (the fastest rate) through 255 (the slowest rate).

SET N=exp   Direct or Program Statement

This command specifies the number of nulls to follow every line-feed. The number of nulls required is unique to each output device. The Sol display driver does not require any nulls, while some printers may require as many as 30.

SET M=exp   Direct Only

This command allows the user to specify a memory size larger than the default of 8192 (8K). The expression is evaluated as an integer number of bytes indicating the maximum memory to be used.

RETURN TO SOLOS/CONSOL COMMAND - BYE Direct or Program Statement

This command returns control to SOLOS or CONSOL, whichever is installed.

STORE PROGRAM COMMAND - SAVE name Direct Only *SOLOS Only*

This command, when used with SOLOS, records the current BASIC program onto cassette tape under the name specified. Only unit 1 is used for program storage. The tape speed is set to "high". CONSOL does not support this function.

READ PROGRAM COMMAND - GET name   Direct Only   *SOLOS Only*

This command when used with SOLOS will read from the specified CUTS tape and find the named program which must be the name assigned when the program was recorded. Once the program has been found, the entire program as recorded will be read into memory and become the current BASIC program. If the program is larger than memory defined by "SET M=", the entire program will be loaded followed by an "SO" error message.

AUTO RUN COMMAND - XEQ name   Direct Only *SOLOS Only*

This command is similar to GET, but once the program has been read into memory and becomes the current BASIC program, an automatic RUN will be performed.

EDIT COMMAND - EDIT line-number   *SOLOS Only*

This command allows the user to edit the specified line of the current BASIC Program. The Sol BASIC editor allows a line of program to be edited without having to re-enter the entire line. To accomplish this, various special keys found on the Sol keyboard are used to direct this editing process in conjunction with the display driver of SOLOS. They are:

DEL or Rubout — This key causes the current character (under the cursor) to be deleted and the remainder of this line to be shifted to the left.

← or CTL-A — The left arrow functions as a cursor control by moving the cursor to the left one character. This is used to position the cursor prior to making a change in the line.

→ or CTL-S — The right arrow moves the cursor to the right one character.

↑ or CTL-W — The up-arrow activates the insert mode. When enter ing characters to the editor, there are two possible modes: insert and non-insert. Non-insert mode is the standard mode specifying that any character entered replaces the character at the cursor location. In insert mode, every character of the line from the cursor to the end of the line is shifted to the right to make room for the character being entered. Insert mode provides an easy way to enter a letter or word at any point on a line.

↓ or CTL-X — The down-arrow deactivates the insert mode.

RETURN — The carriage return terminates the editing of this line by clearing the line from the cursor to the end of the line.

LINE FEED — The line feed also terminates the editing of this line, but leaves the line exactly as on the screen.

The use of the REPEAT key facilitates rapid movement of the cursor through the line.

All of the edit functions are available at all times during input to Sol BASIC.

Pausing the Display

Sol BASIC offers a feature that is quite useful whether outputting to the display screen or any other device. At every carriage return, Sol BASIC looks to see if the space bar of the keyboard has been pressed. If the space bar has been pressed, Sol BASIC will pause until any other key on the keyboard has been pressed.
What this means, for example, is that pressing the space bar while listing a program causes the listing to stop to keyboard control.

V. DIRECT EXECUTION - CALCULATOR MODE

Sol BASIC facilitates computer utilization for the immediate solution of problems--generally of a mathematical nature--which do not require iterative program procedures. To clarify: Sol BASIC may be used as a sophisticated electronic calculator by means of its "Direct" statement execution capability.

While BASIC is in the command mode, *some* BASIC statements may be entered *without* statement numbers. BASIC will *immediately* execute such statements. This is called the direct mode of execution:

```
    Example:  A=1.5
              PRINT A
```

Statements that are entered *with* statement numbers are considered to be program statements which will be executed later.

In the following sections of this manual, all Sol BASIC statements are described. Only those statements which are flagged with the word "Direct" may be used in the direct mode.

Another use for direct execution is as an aid in program development and debugging. Through use of direct statements, program variables can be altered or read, and program flow may be directly controlled.

VI. DECLARATION STATEMENTS

NUMERICAL ARRAY DIMENSION STATEMENT - DIM var (exp) Direct

Allocates memory space for an array. In Sol BASIC, only single dimension arrays are allowed. Maximum array size is determined by available memory. All array elements are set to zero by the DIM statement.

If an array is not explicitly defined by a DIM statement, it is assumed to be defined as an array of 10 elements upon the first reference to it in a program.

CAUTION: An array can be dimensioned only once in a program--dynamically or statically.

```
DATA STATEMENT      - DATA num (,num...,num)
READ STATEMENT      - READ var (,var...,var)
RESTORE STATEMENT   - RESTORE
```

The DATA and READ statements are used in conjunction with each other as one of the methods to assign values to variables.

Every time a DATA statement is encountered, the values in the argument field are assigned sequentially to the next available positions of a data buffer. All DATA statements, no matter where they occur in a program, cause data to be combined into one data list.

READ statements cause values in the data buffer to be accessed sequentially and assigned to the variables named in the READ statement.

```
    Example:  110 DATA 1,2,3.5
              120 DATA 4.5,7,70
              130 DATA 80,81
              140 READ B2,B3,D5,Z6

              is the equivalent of:

              10 LET B2=1
              20 LET B3=2
              30 LET D5=3.5
              40 LET Z6=4.5
```

The RESTORE statement causes the data buffer pointer, which is advanced by the execution of READ statements, to be reset to point to the first position in the data buffer.

```
    Example:  110 DATA 1,2,3.5
              120 DATA 4.5,7,70
              130 DATA 8,81
              140 READ B2,B3
              150 RESTORE
              160 READ D5,D6
```

In this example, the variables would be assigned values equal to:

```
    100  LET B2=1:B3=2:D5=1:D6=2
```

VII. ASSIGNMENT STATEMENTS

LET STATEMENT - LET var=exp Direct

The LET statement is used to assign a value to a variable. The use of the word LET is optional.

```
    Example:  100 LET B=827
              110 LET B5=87E2
              120 R=(X*Y)/2*A
```

The equal sign does not mean equivalence as in ordinary mathematics. It is the replacement operator. It says, replace the value of the variable named on the left with the value of the expression on the right. The expression on the right can be a simple numerical value or an expression composed of numerical values, variables, mathematical operations, and functions.

MATHEMATICAL OPERATORS

The mathematical operators used to form expressions are:

- (unary) ..	Negate (requires only one operand)
*	Multiplication
/	Division
+	Addition
-	Subtraction

The unary minus (negate) may appear in sequence with any other mathematical operator (e.g., A#-B). No other mathematical operators may appear in sequence, and no operator is ever assumed: A++B and (A+2) (B-3) are not valid.

An arithmetic expression is evaluated in a particular order of precedence: negation is performed first, then multiplication and division, and last, addition and subtraction.

In cases of equal precedence, the evaluation is performed from left to right.

The order of evaluation can be controlled explicitly through use of pairs of parentheses. The expression inside the innermost pair is evaluated first; the outermost last.

```
    Example:  150 LET R=A+B-C/2*3

                    is evaluated as:

              Temp1= A + B - Temp2
              R = A + B - Temp2

    Example:  137 LET R= ((A+B)-C)/(2*3)

                    is evaluated as:

              Temp1= A+B   Temp2=Temp1 - C
              Temp3 = 2*3   R=Temp2/Temp3
```

VIII. CONTROL STATEMENTS

Control statements are used to control the natural sequential progression of program statement execution. They can be used to transfer control to another part of a program, terminate execution, or control iterative processes (loops).

```
FOR AND NEXT STATEMENTS - FOR var=exp1 TO exp2 (STEP exp3)
                            :
                            :
                           NEXT (var)
```

The FOR and NEXT statements are used together for setting up program loops. A loop causes the execution of one or more statements for a specified number of times. The variable in the FOR-TO statement is initially set to the value of the first expression (expl). Subsequently, the statements following the FOR are executed. When the NEXT statement is encountered, the named variable is added to the value indicated by the STEP expression in the FOR-TO statement, and execution is resumed at the statement following the FOR-TO. If the addition of the STEP value develops a sum that is greater than the TO expression (exp2), the next instruction executed will be the one following the NEXT statement. If no STEP is specified, a value of one will be assigned. If the TO value is initially less than the initial value, the FOR-NEXT loop will still be executed once.

```
    Example:  110 FOR I=1 TO 10
              120   INPUT X
              130   PRINT I,X,X/5.6
              140 NEXT I
```

Although expressions are permitted for the initial, final, and STEP values in the FOR statement, they will be evaluated only once--first time the loop is entered.

If the variable in the NEXT statement is not given by name, Sol BASIC will properly add the STEP value to the variable in the last FOR statement,

```
    Example:  110 FOR K=1 TO 350
                   :
                   :
              120 FOR L=1 TO 80
                   :
                   :
              130 NEXT
              135 NEXT
                   :
                   :
```

In the preceding example, the NEXT at statement number 130 will STEP the FOR loop beginning at statement 120. The NEXT at 135 will STEP the FOR loop beginning at 110.

It is not possible to use the same variable in two loops if they are nested. In the preceding example, the variable in line 120 could not be K.

When the statement after the NEXT statement is executed, the variable is equal to the value that caused the loop to terminate, not the TO value itself. In the first example, I would be equal to 11 when the loop terminates.

Sol BASIC lists FOR .. NEXT loops indented such that the nesting of loops produces a graphic demonstration of this nesting. This form of indention of loops has proved useful both in problem solving (debugging) and in structured programming.

STOP STATEMENT - STOP

The STOP statement causes the program to stop executing. Sol BASIC returns to the command mode. The STOP statement differs from the END statement in that it causes Sol BASIC to display the statement number where the program halted, and the program can be restarted by a GOTO. The message displayed is:

```
       "STOP IN LINE XXXX"
```

END STATEMENT - END

The END statement causes the program to stop executing. Sol BASIC returns to the command mode. In Sol BASIC, END may appear more than once and need not appear at all. When END is encountered, the message "Sol BASIC" will be displayed. When a program terminates without an END, then only the message "READY" will be displayed.

GOTO STATEMENT - GOTO statement n   Direct

The GOTO statement directs Sol BASIC to execute the specified statement unconditionally. Program flow continues from the new statement. The GOTO must be the last or only statement on a line. Any additional characters following the GOTO will be considered remarks.

```
    Example:  150 GOTO 270
```

IF STATEMENT   Direct

```
    IF relational exp THEN statement n

    IF relational exp THEN ONE OR MORE BASIC statements
```

VIII. CONTROL STATEMENTS (cont.)

The IF statement is used to control the sequence of program statements to be executed, depending on specific conditions. If the relational expression given in the IF is "true", then control is given to the statement after the THEN, If the relational expression is "false", program execution continues at the line following the line with the IF statement.

It is also possible to provide a BASIC statement after, the THEN in the IF statement. If this is done and the relational expression is true, the BASIC statement will be executed and the program will continue at the statement following the expression. When the IF is false, program execution continues at the line following the line with the IF statement.

When evaluating relational expressions, arithmetic operations take precedence in their usual order, and the relational operators are given equal weight and are evaluated last.

```
    Example:  5+6*5 > 15*2      evaluates to be true
```

Relational expressions will have a value of -1 if they are evaluated to be "true", and a value of zero if they evaluate to "false".

```
    Example:  (12 > 10)=-1    or   (A <> A)=0
```

The relational operators are:

=	means	Equal
<>	means	Not equal
<	means	Less than
>	means	Greater than
<=	means	Less than or equal
>=	means	Greater than or equal

```
    Examples:  110 IF A > B+3 THEN 160

               180 IF A=B+3 THEN PRINT "VALUE A ",A: GOTO 140

               190 IF A < B THEN T1=B
```

IX. INPUT/OUTPUT STATEMENTS

INPUT STATEMENTS <u>Direct or Program Statement</u>

```
INPUT ("string"(,))var(,var..var)(,)
```

The INPUT statement allows the user to enter data from the current system input device, usually the keyboard.

```
    Example:  110 INPUT A,B,C
              120 INPUT ""V(1),R,V(2)
```

When the program comes to an INPUT statement, a question mark will be displayed to the current output device, The user then enters the requested data separated by commas and followed by a carriage return. If no data is entered, or if insufficient data is given, the system will prompt the user with a double question mark: "??". Constants are the only data which may be entered in response to an INPUT. If the last variable is followed by a comma, the carriage return-linefeed will be suppressed if the input is also terminated with a linefeed.

If the optional preceding string expression is given, it causes the carriage return/line feed and the "?" prompt to be suppressed, When the optional "string" is given, the "?" prompt will be replaced with the user supplied "string" as the prompt, A null string ("") may also be used to suppress the prompt.

```
PRINT STATEMENTS - PRINT var
                   PRINT exp
                   PRINT %(Z)(E)(N)% (var,exp)
```

The PRINT statement directs Sol BASIC to output the specified values to the output device.

The value of expressions, literal values, simple variables, or text strings may be printed out, the various forms may be combined in the print list by separating each with a comma. If the entire list is terminated by a comma, the terminating carriage return/line feed will be suppressed.

Sol BASIC prints data in fixed width fields, Each datum begins at the leftmost position within each field, The number of digits, trailing zeroes, etc., are specified by the format specification, if any. If a (;) is used as a separator between elements of a print list instead of a comma, the next field will begin after the last character of the preceding field regardless of field widths.

```
    Example:  X=0:Y=0:Z=1: PRINT X;Y;Z

                   prints:

              0 1 1
```

If the next position to be printed is greater than or equal to position 56, then a carriage return/line feed is given before the next value is printed.

PRINT given with no arguments causes one line to be skipped.

All PRINT strings have a special feature--an easy means to output control characters. Whenever an ampersand (&) is encountered in a PRINT string it will not be printed but will cause the very next character of the string to be output as the control of that character.

```
Example:  "&A"=CTL-A;
          "&C"=CTL-C;
          "&&"=&.
```

The TAB Function

The TAB function is used in the PRINT statement to cause data to be printed in exact locations. TAB tells Sol BASIC which position to begin printing the next value in the print list. The argument of TAB may be an expression.

```
Example:  110 PRINT TAB(2),B,TAB(2*R),C
```

Note: The print positions are numbered zero to 71.

Formatted Print

Sol BASIC enables the user to control the format of the printed output by specifying: free format, exponential format, trailing zeroes, and the number of places of accuracy to the right of the decimal point.

If no specification is made, Sol BASIC will print eight places of precision with the low order digit rounded and trailing zeroes suppressed. It will also automatically select between the decimal, integer, and exponential formats, depending on the magnitude of the value to be printed.

It is possible for the user to override Sol BASIC's automatic formatting by including a format specification in the output list, A format specification is two percent signs with interposed code characters.

Format Specification %(Z) (E) (F) (N)%

F = Free Format (BASIC selects format)
Z = Print Trailing Zeroes
E = Print in Exponential Format
N = Print N (N=1-8) places to right of decimal point

All parameters are optional, but once a format specification is given, it will continue to be used until a new format specification is given. To force Sol BASIC to return to its usual default format, a format specification of %% must be given.

```
    Example:  110 PRINT %5E%

              245 PRINT %Z2%,A,B: PRINT %Z3%,CD,%%

    Example:  LIST

               5 FOR I = 1 TO 150 STEP 7.5
               6       B=I: GOSUB 50
               7       PRINT %Z2%;TAB(9);"$";TAB(M);B
               8       B=I*15/2: GOSUB 50
               9    PRINT %Z3%; TAB(M+10) ;B
              10 NEXT
              20 END
              50 M=13: IF B < 1 THEN RETURN
              55 M=12: IF B < 10 THEN RETURN
              60 M=11: IF B < 100 THEN RETURN
              65 M=10: IF B < 1000 THEN RETURN
              70 M=9: RETURN
              READY

              RUN

                  $   1.00      7.500
                  $   8.50     63.750
                  $  16.00    130.000
                  $  23.50    176.250
                  $  31.00    232.500
                  $  38.50    288.750


                                       etc. ...
```

Try running this program yourself.

X. SUBPROGRAMS

A subprogram is a sequence of instructions which perform some task that would have utility in more than one place in a Sol BASIC program. To use such a sequence from more than one place, Sol BASIC provides subroutines and functions.

A subroutine is a program unit that receives control upon execution of a GOSUB statement, Upon completion of the subroutine, control is returned to the statement following the GOSUB by execution of a RETURN statement.

A function is a program unit to which control is passed by a reference to the function name in an expression. A value is computed for the function name, and control is returned to the statement that invoked the function.

```
GOSUB STATEMENT - GOSUB statement n
                    :
                    :
                    .
                  statement n
                    :
                    .
                  RETURN
```

The GOSUB statement causes control to be passed to the given statement number. It is assumed that the given statement number is an entry point of a subroutine. The subroutine returns control to the statement following the GOSUB statement with a RETURN statement. A RETURN must be the last or only statement on a line. Any additional characters following the RETURN will be considered remarks.

Subroutine example:

```
100 X=1
110 GOSUB 200
120 PRINT X
125 X=5.1
130 GOSUB 200
140 PRINT X
150 STOP
200 X=(X+3)*5.32E3
210 RETURN
211 END
```

Subroutines may be nested; that is, subroutines can use GOSUB to call another subroutine which in turn can call another. A subroutine cannot call itself. Subroutine nesting is limited to six levels.

BASIC FUNCTIONS

ABS (exp) Gives the absolute value of the expression.

INT (exp) Gives the largest integer less than or equal to its argument.

RND (exp) Generates pseudo-random numbers ranging between 0.0 and 1.0. The argument is required for syntax but does not alter the function. The random number generator is reset by the CLEAR command.

SGN (exp) Gives a value of +1, if argument is greater than 0. Gives a value of -1 if argument is negative. Gives a value of 0 if argument is 0.

SQR (exp) Gives the square root of the argument.

SIN (exp) Gives the sine of the argument, when the argument is given in radians.

COS (arg) Gives the cosine of the argument, when the argument is given in radians.

TAN (exp) Gives the tangent of the argument, when the argument is given in radians.

TAB (exp) See PRINT statement. Used to position output characters.

ARG (exp)
CALL (exp) ARG and CALL are used together to link to assembly language program segments. Both may be used in the direct mode.

ARGUMENT AND CALL FUNCTIONS - ARG and CALL

When the ARG function appears in some Sol BASIC statement such as B=ARG(V1), the argument will be evaluated as a sixteen bit integer and temporarily stored in the BASIC monitor. Should linkage be made to an assembly language (8080) program segment via the CALL function, the previously stored sixteen bits will be passed to the assembly language code in the B,C register pair.

When the CALL function is invoked by coding it into some BASIC statement such as X6=CALL(5.2*A4); the argument of the CALL function will be evaluated as a sixteen bit address. Sol BASIC will transfer control to that address using an 8080 "CALL" instruction.

Your machine language code loads registers B,C with any desired information. This information is then passed back into the Sol BASIC program as the value of the CALL.

```
    Example:  110 REM LINK TO ASSY LANG PROG
              120 LET X=12: R3=3192
              130 B=ARG(X/5)
              140 LET M=CALL(R3)
              150 PRINT M
              160 END
```

In this example, B is assigned the value of the ARG argument, linkage is made to assembly language program at address 3192, and M is set to whatever was returned in B,C.

To let back into ALS8 the user can use B=CALL(57440).

XI. FILES

Sol BASIC, when used with SOLOS, has provision for writing data to and reading data from cassette data files. Two cassette recorders are supported, so one file may be read from one unit, the data processed, and the processed data written to the other unit. When using files, the cassette operations--other than correctly placing the cassettes--are under control of Sol BASIC running with SOLOS.

Prior to using the FILE OPERATIONS, a few concepts are important, The term FILE, for example, refers to a collection of data, and no assumptions are made by Sol BASIC as to the structure of the FILE. It only provides convenient access to the file information while you, or your program, can define any structure,

For example, the following program:

```
4 FILE #1
5 FOR I=1 TO 10
6 INPUT "NEXT NUMBER"A
7 PRINT #1, I,A,SQR(I+A)
8 NEXT I
9 CLOSE #1
```

would produce a file with the following definable structure:

1. The FILE is 30 "ELEMENTS" long, (3 elements "printed" to the file 10 times.)

2. Every third ELEMENT, starting with the first (1,4,7...) will be a number one larger than the preceding element. (1,20 the value of "I".)

3. Every third element, starting with the second, will be a number as input by the user at line 6.

4. Every third element, starting with the third, will be a number determined by the square root of the sum of the preceding two elements.

5. The END OF FILE follows the 30th element.

Using this type of "structuring" the file "characteristics" are as follows:

Element - one data unit

Record - A series of elements determined by the user. (Three elements per record in the preceding example.)

Length - The number of elements in a file or the number of records. (30 elements, 10 records in the preceding example.)

EOF - The end of the file which is important if the length of the file is unknown or may vary.

The following example illustrates another type of file which produces a usable structure:

```
    5  FILE #1
   10  INPUT "PART NUMBER?"  A
   15  IF A=0 THEN 100
   20  PRINT
   25  INPUT "QUANTITY ON HAND?"  B
   30  PRINT
   35  INPUT "MINIMUM QUANTITY?"  C
   40  PRINT
   45  INPUT "COST PER UNIT?"  D
   50  PRINT
   55  PRINT #1,A,B,C,D
   60  GOTO 10
  100  CLOSE #1
  105  END
```

A. File Operations

FILE #1, FILE #2

In order for the file to be accessed, Sol BASIC must first set up a number of internal parameters. This is known as an OPEN operation, and it must be performed once--and only once--prior to any attempt to access a file. If an OPEN is attempted on an already open file, the program will abort giving a DM error message.

CLOSE #1, CLOSE #2

This operation informs Sol BASIC that no further accesses are going to be made to the file. The operation must be performed each time you are through with a file.

PRINT #1, PRINT #2

This operation writes data to a file. Any number of expressions or variables may follow the comma.

READ #1, READ #2

This operation reads data from a previously written file. Any number of variables can follow the comma.

B. END OF FILE - EOF

When Sol BASIC detects an END of file, it makes a special return to your program. After each successful READ, Sol BASIC returns to the line following the READ statement. Notice this is the next valid line preceded by a line number.

When the END of file is encountered, Sol BASIC searches for the first colon following the READ #1, and executes the statement found there. The following statements--

```
10  FOR I=1 to 1000000
15  READ #1,A:PRINT "END OF FILE";:GOTO 30
20  PRINT I,A
25  NEXT I
30  CLOSE #1
35  END
```

read elements of a file, printing each one until the EOF is reached (assuming there are fewer than one million elements!!). Upon encountering the end, Sol BASIC would print "END OF FILE", close the file and then stop.

C. Let's Use the File Operations

1. Put a cassette in Unit 1. Be sure the ON-OFF control is properly connected, and set the unit to the RECORD mode. Note the counter position so you can rewind the tape to the starting point later.

2. INPUT and RUN the following program:

```
 5  FILE #1
 6  FOR I=1 TO 250
 7  PRINT #1;I,SIN(I), SQR(1)
 8  NEXT I
10  CLOSE #1
```

3. You have now written a file to the cassette tape. Take the recorder out of the RECORD mode and rewind the tape to the start of the file.

4. Place the recorder in the PLAY mode.

5. INPUT and RUN the following program:

```
 5  FILE #1
10  FOR I=1 TO 300
15  READ #1,A,B,C:PRINT "END OF FILE";:GOTO 30
20  PRINT A,B,C
25  NEXT I
30  CLOSE #1
35  END
```

APPENDICES

Appendix A

Loading Sol BASIC

Sol BASIC is distributed on a single cassette. Side one of this tape is 1200 Baud CUTS format while Side two is 300 Baud Kansas City format. CONSOL, SOLOS and CUTER provide the commands necessary to read in this tape.

Sol BASIC may be read into memory and automatically executed by entering "XEQ BASIC". If "GET BASIC" is entered, Sol BASIC will be read into memory but will not automatically be executed.

Sol BASIC is always executed beginning at location zero (EXEC 0). The first execution of Sol BASIC will perform initialization procedures. Once initialized, Sol BASIC may be re-entered at location zero which will leave the BASIC program intact.

A very special feature of Sol BASIC is the capability of specifying the end-of-statement character (usually colon ":") and the print-concatenate character (usually semi-colon ";"). As part of the initialization procedures, Sol BASIC will display the characters currently being used for the end-of-statement and print-concatenate as well as the memory locations which the user may alter to specify any other character to be used.

This capability provides the means to make Sol BASIC appear compatible with most other BASIC's. For example, some BASIC's use a back slash (\) as an end-of-statement character. By changing one location in memory, Sol BASIC will accept and list programs with a back-slash instead of a colon.

The characters altered in this manner by the user should not be used for any other purpose. Should the same character be specified for both end-of-statement and print-concatenate, this character will then be used solely for end-of-statement.

Appendix B

ABBREVIATIONS

Sol BASIC offers the capability of entering abbreviations for each of the reserved words. This capability will greatly reduce the number of keystrokes improving efficiency. Once entered, all abbreviations will be converted to the complete reserved word when listed. Abbreviations are indicated by a character string terminated with a period which first matches an entry in the table of reserved words.

Reserved Word	Shortest Way To Enter
ABS	ABS
ARG	ARG
BYE	B.
CALL	CA.
CLEAR	CLE.
CLOSE	CLO.
COS	COS
DATA	D.
DIM	DIM
EDIT	ED.
END	E.
FILE	FI.
FOR	F.
GET	GET
GOSUB	GOS.
GOTO	G.
IF	IF
INPUT	I.
INT	INT
LET	LET
LIST	L.
NEXT	N.
NEW	NEW
PRINT	P.
READ	READ
REM	REM
RESTORE	RES.
RETURN	R.
RND	RND
RUN	RUN
SAVE	SA.
SET	SET
SGN	SGN
SIN	SIN
SQR	SQR
STEP	STEP
STOP	S.
TAB	TAB
TAN	TAN
THEN	TH.
TO	TO
XEQ	X.

Appendix C

LINE EDITING

The line editing capability of Sol BASIC makes use of the enhanced display driver available with SOLOS. This driver is always monitoring the character sequences being displayed. Whenever an "escape" character is to be displayed, the driver treats this as the beginning of a command and not to be displayed. An escape is followed by one or two bytes which further indicate the nature of the command as well as various options. For example, an escape followed by a byte containing a binary eight (8) allows the display speed to be varied. Table C-1 defines the various escape sequences supported by the driver contained within SOLOS. In each case the bytes) following the escape character is a binary value.

Table C-1. Escape Sequences Supported by SOLOS Display Driver.

Escape Code	Meaning
01	The next byte indicates the position within the current line at which to position the cursor.
02	The next byte indicates the line number at which to position the cursor. The top most line is known as Line 1, and the bottom line is Line 16.
03	This command is useful only to a machine language program. The current line number of the cursor is returned in Register "B", and the current position within that line is returned to Register "C",
04	This command is useful only to a machine language program. The absolute memory address of the cursor within the display RAM is returned in Register Pair "BC".
05 06 07	These three commands cause whatever character is in Register "B" to be output directly to the current cursor location of the display. This is useful both to display the escape character itself and to display characters in inverse video.
08	The next byte following indicates the speed which the display is to occur. A binary zero (0) is the fastest and a binary 255 is the slowest.
09	This escape command functions the same as does the 01 above.

Although these escape sequences appear to be of value only to the machine language programmer, let your imagination run wild. With these sequences, you could write a BASIC program which might move the cursor around the screen to produce various patterns . . . or who knows what?

Remember that these escape sequences are formed by outputting this data to the display driver. Instead of displaying the escape, and following byte(s), this data is treated like an internal command to the driver.

Appendix D

ERROR MESSAGES

ERRORS EXPLANATION

BA Bad argument. A command has been given an illegal argument.

SN Bad syntax.

CS Control stack error. For example, FOR has no corresponding NEXT, illegal FOR-NEXT, GOSUB-RETURN nesting, or control stack too deep.

DI Direct input error. User has tried to give BASIC a command which it cannot process in the direct mode.

DM Dimension error. Attempt to dimension (DIM) array more than once in program, attempt to open an already opened file used for reads.

FP Floating point arithmetic error. User has attempted to divide by zero, or a calculation has resulted in a number too large to be represented in BASIC's number format.

Note: Underflow will result in zero with no error indication.

IN Input error. User has given a number in incorrect format in response to an INPUT statement.

LL Line too long. User has attempted to input a line of more than 72 characters.

LN Line number error. Line number specified in a GOTO, GOSUB, or IF statement was not found.

NA Negative argument for square root function.

OB Out of bounds. An array index, TAB value or other integer has exceeded its permissible limit. Also an attempt to load a BASIC program above available memory.

RD Read error. No more data in data buffer, the number of READ statements has exceeded the number of DATA values given, or an error during cassette read operations.

SO Storage overflow. Working memory has insufficient room for text, symbol table, array space, or program is too large.

Appendix E

THE BASIC CHARACTER SET

I	II	III	IV	V
←	@	:	+	%
↑	?	9-0	*	$
]	>	/	)	#
\	=	.	(	"
[	<	-	'	!
Z-A	;	,	&	

Appendix F

Sol BASIC Statement-Summary

CLOSE #n	Terminates processing of a CUTS tape file where n-1 or 2. This statement is required for an output tape.
DATA num(,num...,num)	Supplies data for READ state ment.
DIM var(exp)	Used to dimension numerical arrays containing a subscript greater than 10.
END	Halts program execution.
FILE #n	Prepares BASIC for later INPUT's or PRINT's to a CUTS tape file where n=1 or 2.
FOR var=exp TO exp(STEPexp) . NEXT (var)	Loop control statements; var must be the same in both statements (if used).
GOSUB statement n . statement n . RETURN	Transfers control to the subroutine beginning at statement n, and then returns control to the statement following GOSUB.
GOTO statement n	Branches to statement n.

IF relational exp THEN statement n
IF rel. exp THEN statement n
	If the relational expression is true, branches to statement n, or executes statement n. The next program line will be executed if false.

INPUT ("string"(,))var(,var…var)(,) READ #n,var(,var...,var);	A statement processed at end of file. Requests numerical data at program execution time.
(LET) var=exp	Assigns value of expression to variable.
PRINT var PRINT "string" PRINT exp PRINT #n;var or exp	Outputs variable or literal values. Forms may be combined (except as noted). An "&" within a string outputs the next character in control mode (e.g., &X means CTL-X).
READ var(,var...,var)	Reads numerical values from DATA statements.

REM anything	Comment statement.
RESTORE	Resets READ pointer to beginning of first DATA statement.
SET I=exp	Select the system input device.
SET O=exp	Select the system output device.
SET S=exp	Select the display speed.
SET N=exp	Select number of nulls.
BYE	Return control to the CONSOL or SOLOS personality module.
STOP	Terminates program.

Appendix G

REFERENCES

Entering BASIC, J. Sack and J. Meadows, Science Research Associates, 1973.

BASIC: A Computer Programming Language, C. Pegels, Holden-Day, Inc., 1973.

BASIC Programming, J. Kemeny and T. Kurtz, Peoples Computer Co., 1010 Doyle (P.O. Box 310), Menlo Park, CA 94025, 1967.

BASIC, Albrecht, Finkle and Brown, Peoples Computer Co., 1010 Doyle (P.O. Box 310), Menlo Park, CA 94025, 1973.

A Guided Tour of Computer Programming in BASIC, T. Dwyer, Houghton Mifflin Co., 1973.

Programming Time Shared Computer in BASIC, Eugene H. Barnett, Wiley-Interscience, L/C 72-175789 ($12.00).

Programming Language #2, Digital Equipment Corp., Maynard, MA 01754.

101 BASIC Computer Games, Software Distribution Center, Digital Equipment Corp., Maynard, MA 01754 ($7.50).

What to Do After You Hit Return, Peoples Computer Co., 1010 Doyle (P.O. Box 310), Menlo Park, CA 94025 ($6.95).

SOLOS^{TM} Monitor Program Source Listing

```
                    9999         COPY   SOLOS1/1                              1 OF 3 ******
                    0002 *
                    0003 *
                    0004 *
                    0005 *
                    0006 *                              *****           *
                    0007 *                             *                *
                    0008 *                             * * *     **     *
                    0009 *                                  *   *  *    *
                    0010 *                             *****     **     *
                    0011 *
                    0012 *                                SYSTEM SOLFTWARE
                    0013 *
                    0014 *
                    0015 *
                    0016 *    NOTE:   CONSOL, SOLOS AND SOLED ARE REGISTERED
                    0017 *            TRADEMARKS OF:
                    0018 *
                    0019 *                         PROCESSOR TECHNOLOGY CORP.
                    0020 *                            EMERYVILLE, CALIF
                    0021 *
                    0022 *
                    0023 *
                    0024 *                         =-=-=-     SOLOS     -=-=-=
                    0025 *
                    0026 *
                    0027 *     VERSION  1.3
                    0028 *     RELEASE  77-03-27
                    0029 *
                    0030 *
                    0031 *     THIS 2048 BYTE PROGRAM IS THE STANDARD Sol STAND
                    0032 *  ALONE OPERATING SYSTEM.  IT IS CONFIGURED TO OPTIMIZE
                    0033 *  THE CONVENIENCE AND POWER OF THE Sol-20 AND ONE OR TWO
                    0034 *  CASSETTE RECORDERS IN STAND ALONE COMPUTER APPLICATIONS.
                    0035 *
                    0036 *
                    0037 *
                    0038 *   AUTO-STARTUP CODE
                    0039 *
C000 00             0040 START  DB     0     FOUR PHASE WONDER
C001 C3 AF C1       0041 INIT   JMP    STRTA  SYSTEM RESTART ENTRY POINT
                    0042 *
                    0043 *
                    0044 *                    ENTRY POINTS
                    0045 *
                    0046 *     THESE JUMP POINTS ARE PROVIDED TO ALLOW COMMON ENTRY
                    0047 *  LOCATIONS FOR ALL VERSIONS OF SOLOS.  THEY ARE USED
                    0048 *  EXTENSIVLY BY Sol SYSTEM PROGRAMS AND IT IS RECOMMENDED
                    0049 *  THAT USER ROUTINES ACCESS SOLOS THROUGH THESE POINTS.
                    0050 *
C004 C3 C9 C1       0051 RETRN  JMP    COMND  RETURN TO SYSTEM ENTRY POINT
```

```
C007 C3 E0 C5       0052 FOPEN  JMP    BOPEN  FILE OPEN ENTRY
C00A C3 03 C6       0053 FCLOS  JMP    PCLOS  FILE CLOSE ENTRY
C00D C3 46 C6       0054 RDBYT  JMP    RTBYT  CASSETTE READ BYTE ENTRY
C010 C3 83 C6       0055 WRBYT  JMP    WTBYT  CASSETTE WRITE BYTE ENTRY
C013 C3 CB C6       0056 RDBLK  JMP    RTAPE  CASSETTE READ BLOCK ENTRY
C016 C3 7F C7       0057 WRBLK  JMP    WTAPE  CASSETTE WRITE BLOCK ENTRY
                    0058 *
                    0059 *
                    0060 *       SYSTEM I/O ENTRY POINTS
                    0061 *
                    0062 *  THESE ROUTINES PERFORM SYSTEM I/O
                    0063 *  THERE ARE TWO ENTRY TYPES:
                    0064 *   SINP/SOUT       REG "A" WILL BE SET TO THE STANDARD
                    0065 *                   SYSTEM PSEUDO PORT.
                    0066 *   AINP/AOUT       REG "A" MUST BE SET BY THE USER AND WILL
                    0067 *                   SPECIFY THE DESIRED PSEUDO PORT.
                    0068 *
                    0069 *  THE FOLLOWING ARE THE PSEUDO PORTS:
                    0070 *    PORT    DESCRIPTION
                    0071 *    ----    ----------------------------------------
                    0072 *     0      KEYBOARD WHEN INPUT, AND VDM WHEN OUTPUT
                    0073 *     1      SERIAL
                    0074 *     2      PARALLEL
                    0075 *     3      USER DEFINED
                    0076 *
C019 3A 07 C8       0077 SOUT   LDA    OPORT  SOUT ENTRY POINT
C01C C3 3B C0       0078 AOUT   JMP    OUTPR  AOUT ENTRY POINT
C01F 3A 06 C8       0079 SINP   LDA    IPORT  SINP ENTRY POINT
     C022           0080 AINP   EQU    $      AINP ENTRY POINT
                    0081 * * * * * * * * * * END OF SYSTEM ENTRY POINTS * * * *
C022 E5             0082        PUSH   H      THIS IS ACTUALLY AINP
C023 21 9A C2       0083        LXI    H,ITAB
                    0084 *
                    0085 *
                    0086 *     THIS ROUTINE PROCESSES THE I/O REQUESTS BY DISPATCHING
                    0087 *  TO THE DRIVER REQUESTED IN REGISTER "A".  ON ENTRY HL
                    0088 *  HAVE THE PROPER DISPATCH TABLE.
                    0089 *
C026 E6 03          0090 IOPRC  ANI    3      KEEP REGISTER "A" TO FOUR VALUES
C028 07             0091        RLC    .      COMPUTE ENTRY ADDRESS
C029 85             0092        ADD    L
C02A 6F             0093        MOV    L,A    WE HAVE ADDRESS
C02B C3 27 C2       0094        JMP    DISPT  DISPATCH TO IT
                    0095 *
                    0096 *
                    0097 *
                    0098 *
                    0099 *
                    0100 *                ----=  Sol SYSTEM I/O ROUTINES =----
                    0101 *
                    0102 *
                    0103 *     THIS ROUTINE IS A MODEL OF ALL INPUT ROUTINES WITHIN
                    0104 *  SOLOS.  EACH ROUTINE FIRST TESTS THE STATUS INPUT FOR
```

```
               *  DATA AVAILABLE.  IF NO CHARACTER HAS BEEN RECEIVED THE
               *  ROUTINE RETURNS WITH THE ZERO FLAG SET.  OTHERWISE THE
               *  CHARACTER IS INPUT AND A RETURN MADE WITH THE CHARACTER
               *  IN THE ACCUMULATOR AND THE ZERO FLAG RESET.
               *
               *
               *          KEYBOARD INPUT DRIVER
               *
C02E DB FA     KSTAT  IN     STAPT  GET STATUS WORD
C030 2F               CMA    .      INVERT IT FOR PROPER RETURN
C031 E6 01            ANI    KDR    TEST KEYBOARD BIT
C033 C8               RZ     .      ZERO IF NO CHARACTER RECEIVED
               *
C034 DB FC            IN     KDATA  GET CHARACTER
C036 C9               RET    .      GO BACK WITH IT
               *
               *
               *    THIS JUMP IS PART OF THE AUTO START UP CODE
               *
C037 00               DB     0      VERIFY ADDR=C037 SO NEXT INSTRUCTION IS AT C038
C038 C3 01 C0         JMP    INIT
               *
               *
               *             JUMP TABLE OUTPUT ROUTINES
               *
               *    THIS ROUTINE SETS UP THE DISPATCH TABLE FOR OUTPUT
               * ROUTINES.  THE CHARACTER FOR OUTPUT IS IN REGISTER "B".
               * OUTPUT IS MADE TO THE DRIVER POINTED TO BY THE REGISTER
               * "A".  THE DEVICE DRIVERS ARE DEFINED AS FOLLOWS:
               *
               *
               *       0 - DISPLAY SCREEN
               *       1 - SERIAL OUTPUT PORT
               *       2 - PARALLEL OUTPUT PORT
               *       3 - USER DEFINED OR ERROR FLAG
               *
               * ENTRY AT:  SOUT SELECTS CURRENT OUTPUT DEVICE
               *            AOUT SELECTS DEVICE IN REGISTER 'A'
               *
C03B E5        OUTPR  PUSH   H
C03C 21 92 C2         LXI    H,OTAB  POINT TO OUTPUT TABLE
C03F C3 26 C0         JMP    IOPRC  AND DISPATCH TO OUTPUT ROUTINE
               *
               *
               *
               *
               *          SERIAL INPUT DRIVER
               *
C042 DB F8     SSTAT  IN     SERST  GET SERIAL STATUS WORD
C044 E6 40            ANI    SDR    TEST FOR SERIAL DATA READY
C046 C8               RZ     .      FLAGS ARE SET .
               *
C047 DB F9            IN     SDATA  GET DATA BYTE
```

```
C049 C9               RET    .      WE HAVE IT
               *
               *
               *  SERIAL DATA OUTPUT
               *
C04A DB F8     SDROT  IN     SERST  GET PORT STATUS
C04C 17               RAL    .      PUT HIGH BIT IN CARRY
C04D D2 4A C0         JNC    SDROT  LOOP UNTIL TRANSMITTER BUFFER IS EMPTY
C050 78               MOV    A,B    GET THE CHARACTER BACK
C051 D3 F9            OUT    SDATA  SEND IT OUT
C053 C9               RET    .      AND WE'RE DONE
               *
               *
               *
               *
               *
               *                    VIDEO DISPLAY ROUTINES
               *
               *
               *  THESE ROUTINES ALLOW FOR STANDARD VIDEO TERMINAL
               *  OPERATIONS.  ON ENTRY, THE CHARACTER FOR OUTPUT IS IN
               *  REGISTER B AND ALL REGISTERS EXCEPT "A" AND FLAGS ARE
               *  UNALTERED ON RETURN.
               *
               *
               *
C054 E5        VDMOT  PUSH   H      SAVE MOST REGISTERS
C055 D5               PUSH   D
C056 C5               PUSH   B
               *
               *  TEST IF ESC SEQUENCE HAS BEEN STARTED
               *
C057 3A 0C C8         LDA    ESCFL  GET ESCAPE FLAG
C05A B7               ORA    A
C05B C2 5F C1         JNZ    ESCS   IF NON ZERO GO PROCESS THE REST OF THE SEQUENC
               *
               *
C05E 78        CHPCK  MOV    A,B    SAVE IN B...STRIP PARITY BEFORE SCREEN!
C05F E6 7F            ANI    7FH    CLR PARITY TO LOCATE IN TBL
C061 47               MOV    B,A    KEEP IT W/OUT PARITY IN B TOO
C062 CA 7C C0         JZ     GOBK   DO A QUICK EXIT IF A NULL
C065 21 73 C2         LXI    H,TBL  POINT TO SPECIAL CHARACTER TABLE
C068 CD 82 C0         CALL   TSRCH  GO PROCESS
               *
C06B CD 1C C1  GOBACK CALL   VDADD  GET SCREEN ADDRESS
C06E 7E               MOV    A,M    GET PRESENT CURSOR CHARACTER
C06F F6 80            ORI    80H
C071 77               MOV    M,A    CURSOR IS BACK ON
C072 2A 0A C8         LHLD   SPEED-1  GET DELAY SPEED
C075 2C               INR    L      MAKE SURE IT IS NON-ZERO
C076 AF               XRA    A      DELAY WILL END WHEN H=0
C077 2B        TIMER  DCX    H      TIMER DELAYS HERE
C078 BC               CMP    H      DONE WITH DELAY YET
```

```
C079 C2 77 C0             JNZ    TIMER  KEEP DELAYING
C07C C1            GOBK   POP    B
C07D D1                   POP    D      RESTORE REGISTERS
C07E E1                   POP    H
C07F C9                   RET    .      EXIT FROM VDMOT
                   *
C080 23            NEXT   INX    H
C081 23                   INX    H
                   *
                   *
                   *  THIS ROUTINE SEARCHES THROUGH A SINGLE CHARACTER
                   * TABLE FOR A MATCH TO THE CHARACTER IN "B".  IF FOUND
                   * A DISPATCH IS MADE TO THE ADDRESS FOLLOWING THE MATCHED
                   * CHARACTER.  IF NOT FOUND THE CHARACTER IS DISPLAYED ON
                   * THE MONITOP.
                   *
C082 7E            TSRCH  MOV    A,M    GET CHR FROM TABLE
C083 B7                   ORA    A
C084 CA 94 C0             JZ     CHAR   ZERO IS THE LAST
C087 B8                   CMP    B      TEST THE CHR
C088 23                   INX    H      POINT FORWARD
C089 C2 80 C0             JNZ    NEXT
C08C E5                   PUSH   H      FOUND ONE...SAVE ADDRESS
C08D CD 36 C1             CALL   CREM   REMOVE CURSOR
C090 E3                   XTHL   .      GET DISPATCH ADDRESS TO HL
C091 C3 27 C2             JMP    DISPT  DISPATCH NOW
                   *
                   *      PUT CHARACTER TO SCREEN
                   *
C094 78            CHAR   MOV    A,B    GET CHARACTER
C095 FE 7F                CPI    7FH    IS IT A DEL?
C097 C8                   RZ     .      GO BACK IF SO
                   *
                   *
                   *
   C098            OCHAR  EQU    $      ACTUALLY PUT CHAR TO SCREEN NOW
C098 CD 1C C1             CALL   VDADD  GET SCREEN ADDRESS
C09B 70                   MOV    M,B    PUT CHR ON SCREEN
                   *
C09C 3A 08 C8             LDA    NCHAR  GET CHARACTER POSITION
C09F FE 3F                CPI    63     END OF LINE?
C0A1 DA C1 C0             JC     OK
C0A4 3A 09 C8             LDA    LINE
C0A7 FE 0F                CPI    15     END OF SCREEN?
C0A9 C2 C1 C0             JNZ    OK
                   *
                   *   END OF SCREEN...ROLL UP ONE LINE
                   *
C0AC AF            SCROLL XRA    A
C0AD 32 08 C8             STA    NCHAR  BACK TO FIRST CHAR POSITION
C0B0 4F            SROL   MOV    C,A
C0B1 CD 23 C1             CALL   VDAD   CALCULATE LINE TO BE BLANKED
C0B4 AF                   XRA    A
```

```
C0B5 CD FA C0             CALL   CLIN1  CLEAR IT
C0B8 3A 0A C8             LDA    BOT
C0BB 3C                   INR    A
C0BC E6 0F                ANI    0FH
C0BE C3 EE C0             JMP    ERAS3
                   *
                   *   INCREMENT LINE COUNTER IF NECESSARY
                   *
C0C1 3A 08 C8      OK     LDA    NCHAR  GET CHR POSITION
C0C4 3C                   INR    A
C0C5 E6 3F                ANI    3FH    MOD 64 AND WRAP
C0C7 32 08 C8             STA    NCHAR
C0CA C0                   RNZ    .      DIDN'T HIT END OF LINE, OK
   C0CB            PDOWN  EQU    $      CURSOR DOWN ONE LINE HERE
C0CB 3A 09 C8             LDA    LINE   GET THE LINE COUNT
C0CE 3C                   INR    A
C0CF E6 0F         CURSC  ANI    0FH    MOD 15 INCREMENT
C0D1 32 09 C8      CUR    STA    LINE   STORE THE NEW
C0D4 C9                   RET
                   *
                   *    ERASE SCREEN
                   *
C0D5 21 00 CC      PERSE  LXI    H,VDMEM  POINT TO SCREEN
C0D8 36 A0                MVI    M,80H+' '  THIS IS THE CURSOR
                   *
C0DA 23                   INX    H      BUMP 1ST
   C0DB            ERAS1  EQU    $      LOOPS HERE TO ERASE SCREEN
C0DB 36 20                MVI    M,' '  BLANK IT OUT
C0DD 23                   INX    H      NEXT
C0DE 7C                   MOV    A,H    SEE IF END OF SCREEN YET
C0DF FE D0                CPI    0D0H   ?
C0E1 DA DB C0             JC     ERAS1  NO--KEEP BLANKING
C0E4 37                   STC    .      CARRY WILL SAY COMPLETE ERASE
                   *
C0E5 3E 00         PHOME  MVI    A,0    RESET CURSOR--CARRY=ERASE, ELSE HOME
C0E7 32 09 C8             STA    LINE   ZERO LINE
C0EA 32 08 C8             STA    NCHAR  LEFT SIDE OF SCREEN
C0ED D0                   RNC    .      IF NO CARRY, WE ARE DONE WITH HOME
                   *
C0EE D3 FE         ERAS3  OUT    DSTAT  RESET SCROLL PARAMETERS
C0F0 32 0A C8             STA    BOT    BEGINNING OF TEXT OFFSET
C0F3 C9                   RET
                   *
                   *
C0F4 CD 1C C1      CLINE  CALL   VDADD  GET CURRENT SCREEN ADDRESS
C0F7 3A 08 C8             LDA    NCHAR  CURRENT CURSOR POSITION
C0FA FE 40         CLIN1  CPI    64     NO MORE THAN 63
C0FC D0                   RNC    .      ALL DONE
C0FD 36 20                MVI    M,' '  ALL SPACED OUT
C0FF 23                   INX    H
C100 3C                   INR    A
C101 C3 FA C0             JMP    CLIN1  LOOP TO END OF LINE
                   *
```

```
                    *
                    *  ROUTINE TO MOVE THE CURSOR UP ONE LINE
                    *
C104 3A 09 C8       PUP    LDA    LINE   GET LINE COUNT
C107 3D                    DCR    A
C108 C3 CF C0              JMP    CURSC  MERGE TO HANDLE CURSOR
                    *
                    *  MOVE CURSOR LEFT ONE POSITION
                    *
C10B 3A 08 C8       PLEFT  LDA    NCHAR
C10E 3D                    DCR    A
     C10F           PCUR   EQU    $      CURSOR ON SAME LINE
C10F E6 3F                 ANI    3FH    LET CURSOR WRAP
C111 32 08 C8              STA    NCHAR  UPDATED CURSOR
C114 C9                    RET
                    *
                    *     CURSOR RIGHT ONE POSITION
                    *
C115 3A 08 C8       PRIT   LDA    NCHAR
C118 3C                    INR    A
C119 C3 0F C1              JMP    PCUR
                    *
                    *   ROUTINE TO CALCULATE SCREEN ADDRESS
                    *
                    *   ENTRY AT:     RETURNS:
                    *
                    *        VDADD   CURRENT SCREEN ADDRESS
                    *        VDAD2   ADDRESS OF CURRENT LINE, CHAR 'C'
                    *        VDAD    LINE 'A', CHARACTER POSITION 'C'
                    *
C11C 3A 08 C8       VDADD  LDA    NCHAR  GET CHARACTER POSITION
C11F 4F                    MOV    C,A    'C' KEEPS IT
C120 3A 09 C8       VDAD2  LDA    LINE   LINE POSITION
C123 6F             VDAD   MOV    L,A    INTO 'L'
C124 3A 0A C8              LDA    BOT    GET TEXT OFFSET
C127 85                    ADD    L      ADD IT TO THE LINE POSITION
C128 0F                    RRC    .      TIMES TWO
C129 0F                    RRC    .      MAKES FOUR
C12A 6F                    MOV    L,A    L HAS IT
C12B E6 03                 ANI    3      MOD THREE FOR LATER
C12D C6 CC                 ADI    <VDMEM  LOW SCREEN OFFSET
C12F 67                    MOV    H,A    NOW H IS DONE
C130 7D                    MOV    A,L    TWIST L'S ARM
C131 E6 C0                 ANI    0C0H
C133 81                    ADD    C
C134 6F                    MOV    L,A
C135 C9                    RET    .      H & L ARE NOW PERVERTED
                    *
                    *    ROUTINE TO REMOVE CURSOR
                    *
C136 CD 1C C1       CREM   CALL   VDADD  GET CURRENT SCREEN ADDRESS
C139 7E                    MOV    A,M
C13A E6 7F                 ANI    7FH    STRIP OFF THE CURSOR
```

```
C13C 77                    MOV    M,A
C13D C9                    RET
                    *
                    *     ROUTINE TO BACKSPACE
                    *
C13E CD 0B C1       PBACK  CALL   PLEFT
C141 CD 1C C1              CALL   VDADD  GET SCREEN ADDRESS
C144 36 20                 MVI    M,' '  PUT A BLANK THERE
C146 C9                    RET
                    *
                    *     ROUTINE TO PROCESS A CARRIAGE RETURN
                    *
C147 CD F4 C0       PCR    CALL   CLINE  CLEAR FROM CURRENT CURSOR TO END OF LINE
                    *   REG A COMES BACK W/64 AND WILL BE CLEARED BY PCUR
C14A C3 0F C1              JMP    PCUR   AND STORE THE NEW VALUE
                    *
                    *   ROUTINE TO PROCESS LINEFEED
                    *
C14D 3A 09 C8       PLF    LDA    LINE   GET LINE COUNT
C150 3C                    INR    A
C151 E6 0F                 ANI    15     SEE IF IT WRAPPED AROUND
C153 C2 D1 C0              JNZ    CUR    NO--NO NEED TO SCROLL
C156 C3 B0 C0              JMP    SROL   YES--THEN SCROLL
                    *
                    *     SET ESCAPE PROCESS FLAG
                    *
C159 3E FF          PESC   MVI    A,-1
C15B 32 0C C8              STA    ESCFL  SET FLAG
C15E C9                    RET
                    *
                    *       PROCESS ESCAPE SEQUENCE
                    *
C15F CD 36 C1       ESCS   CALL   CREM   REMOVE CURSOR
C162 CD 68 C1              CALL   ESCSP  PROCESS THE NEXT PART OF SEQUENCE
C165 C3 6B C0              JMP    GOBACK
                    *
C168 3A 0C C8       ESCSP  LDA    ESCFL  GET ESCAPE FLAG
C16B FE FF                 CPI    -1     TEST FLAG
C16D CA 90 C1              JZ     SECOND
                    *
                    *  PROCESS THIRD CHR OF ESC SEQUENCE
                    *
C170 21 0C C8              LXI    H,ESCFL
C173 36 00                 MVI    M,0    NO MORE PARTS TO THE SEQUENCE
C175 FE 02                 CPI    2
C177 DA 88 C1              JC     SETX   SET X  IF IS ONE
C17A CA 8C C1              JZ     SETY   SET Y  IF IS TWO
C17D FE 08                 CPI    8
C17F CA 98 C5              JZ     STSPD  SET NEW DISPLAY SPEED IF "8"
C182 FE 09                 CPI    9
C184 DA 98 C0              JC     OCHAR  PUT IT ON THE SCREEN
C187 C0                    RNZ
                    *
```

```
                        0423 *  TAB ABSOLUTE TO VALUE IN REG B
                        0424 *
C188 78                 0425 SETX   MOV   A,B    GET CHARACTER
C189 C3 0F C1           0426        JMP   PCUR
                        0427 *
                        0428 *  SET CURSOR TO LINE "B"
                        0429 *
C18C 78                 0430 SETY   MOV   A,B
C18D C3 CF C0           0431        JMP   CURSC
                        0432 *
                        0433 *
                        0434 *   PROCESS SECOND CHR OF ESC SEQUENCE
                        0435 *
C190 78                 0436 SECOND MOV   A,B    GET WHICH
C191 FE 03              0437        CPI   3
C193 CA A6 C1           0438        JZ    CURET  RETURN CURSOR PARAMETERS
C196 FE 04              0439        CPI   4
C198 C2 A2 C1           0440        JNZ   ARET2
                        0441 *
                        0442 *   ESC <4>  RETURN ABSOLUTE SCREEN ADDRESS
                        0443 *
C19B 44                 0444 ARET   MOV   B,H
C19C 4D                 0445        MOV   C,L    PRESENT SCREEN ADDRESS TO BC FOR RETURN
                        0446 *
C19D E1                 0447 ARET1  POP   H      RETURN ADDRESS
C19E D1                 0448        POP   D      OLD B
C19F C5                 0449        PUSH  B
C1A0 E5                 0450        PUSH  H
C1A1 AF                 0451        XRA   A
C1A2 32 0C C8           0452 ARET2  STA   ESCFL
C1A5 C9                 0453        RET
                        0454 *
                        0455 *
                        0456 *     RETURN PRESENT SCREEN PARAMETERS IN BC
                        0457 *
C1A6 21 08 C8           0458 CURET  LXI   H,NCHAR
C1A9 46                 0459        MOV   B,M    CHARACTER POSITION
C1AA 23                 0460        INX   H
C1AB 4E                 0461        MOV   C,M    LINE POSITION
C1AC C3 9D C1           0462        JMP   ARET1
                        0463 *
                        0464 *
                        0465 *                START UP SYSTEM
                        0466 *
                        0467 *   CLEAR SCREEN AND THE FIRST 256 BYTES OF GLOBAL RAM
                        0468 *  THEN ENTER THE COMMAND MODE.
                        0469 *
C1AF AF                 0470 STRTA  XRA   A
C1B0 4F                 0471        MOV   C,A
C1B1 21 00 C8           0472        LXI   H,SYSRAM  CLEAR THE FIRST PAGE
                        0473 *
C1B4 77                 0474 CLERA  MOV   M,A
C1B5 23                 0475        INX   H
```

```
C1B6 0C                 0476        INR   C
C1B7 C2 B4 C1           0477        JNZ   CLERA
                        0478 *
C1BA 31 FF CB           0479        LXI   SP,SYSTP  SET UP THE STACK FOR CALL
C1BD CD D5 C0           0480        CALL  PERSE
C1C0 AF                 0481 COMN1  XRA   A
C1C1 D3 FA              0482        OUT   STAPT  BE SURE TAPES ARE OFF
C1C3 32 07 C8           0483        STA   OPORT
C1C6 32 06 C8           0484        STA   IPORT
                        0485 *
                        0486 *
                        0487 *
                        0488 *            =--  COMMAND MODE  --=
                        0489 *
                        0490 *
                        0491 *   THIS ROUTINE GETS AND PROCESSES COMMANDS
                        0492 *
C1C9 31 FF CB           0493 COMND  LXI   SP,SYSTP  SET STACK POINTER
C1CC 3A 07 C8           0494        LDA   OPORT  GET PORT
C1CF F5                 0495        PUSH  PSW
C1D0 AF                 0496        XRA   A
C1D1 32 07 C8           0497        STA   OPORT  FORCE SCREEN OPERATIONS
C1D4 CD F1 C2           0498        CALL  PROMPT  PUT PROMPT ON SCREEN
C1D7 CD E4 C1           0499        CALL  GCLIN  GET COMMAND LINE
C1DA F1                 0500        POP   PSW
C1DB 32 07 C8           0501        STA   OPORT  RESTORE DEFAULT PORT
C1DE CD 05 C2           0502        CALL  COPRC  PROCESS THE LINE
C1E1 C3 C9 C1           0503        JMP   COMND  OVER AND OVER
                        0504 *
                        0505 *
                        0506 *
                        0507 *   THIS ROUTINE READS A COMMAND LINE FROM THE SYSTEM
                        0508 *  KEYBOARD
                        0509 *
                        0510 *  C/R   TERMINATES THE SEQUENCE ERASING ALL CHARS TO THE
                        0511 *        RIGHT OF THE CURSOR
                        0512 *  L/F   TERMINATES THE SEQUENCE
                        0513 *  MODE  RESTARTS THE COMMAND LINE.
                        0514 *
C1E4 CD 1F C0           0515 GCLIN  CALL  SINP   READ INPUT DEVICE
C1E7 CA E4 C1           0516        JZ    GCLIN
C1EA E6 7F              0517        ANI   7FH    CLEAR PARITY BIT
C1EC CA C0 C1           0518        JZ    COMN1  THIS WAS A MODE (OR EVEN CTL-@)
C1EF 47                 0519        MOV   B,A
C1F0 FE 0D              0520        CPI   CR     CARRIAGE RETURN
C1F2 CA F4 C0           0521        JZ    CLINE  YES--DONE WITH LINE
C1F5 FE 0A              0522        CPI   LF     LINE FEED
C1F7 C8                 0523        RZ    .      YES--DONE WITH LINE, LEAVE AS IS
C1F8 FE 7F              0524        CPI   7FH    DELETE CHR?
C1FA C2 FF C1           0525        JNZ   CONT
C1FD 06 5F              0526        MVI   B,BACKS  REPLACE IT
                        0527 *
C1FF CD 19 C0           0528 CONT   CALL  SOUT
```

```
C202 C3 E4 C1          0529           JMP    GCLIN
                       0530 *
                       0531 *
                       0532 *         FIND AND PROCESS COMMAND
                       0533 *
C205 CD 35 C1          0534 COPRC  CALL   CREM    REMOVE THE CURSOR
C208 0E 01             0535           MVI    C,1     SET FOR CHARACTER POSITION
C20A CD 20 C1          0536           CALL   VDAD2   GET SCREEN ADDRESS
C20D EB                0537           XCHG
C20E 21 00 C0          0538           LXI    H,START  MAKE SURE HL PT TO SOLOS START
C211 E5                0539           PUSH   H       SAVE IT FOR LATER DISPT
C212 CD 2E C3          0540           CALL   SCHR    SCAN PAST BLANKS
C215 CA 80 C4          0541           JZ     ERR1    NO COMMAND?
C218 EB                0542           XCHG   ..      HL HAS FIRST CHR
                       0543 *
C219 11 4A C2          0544           LXI    D,COMTAB  POINT TO COMMAND TABLE
C21C CD 31 C2          0545           CALL   FDCOM   SEE IF IN PRIMARY COMMAND TABLE
C21F CC 2E C2          0546           CZ     FDCOU   IF NOT, TRY CUSTOM TABLE NEXT
     C222              0547 DISPO  EQU    $       HERE TO SEE IF ERROR OR DISP
C222 CA 81 C4          0548           JZ     ERR2    NOT VALID, ERROR
C225 13                0549           INX    D       BUMP TO PTR OF RTN
C226 EB                0550           XCHG   .       HL PT TO RTN ADDR
                       0551 *
                       0552 *
                       0553 *  THIS IS THE DISPATCH ROUTINE.
                       0554 *  HL PT TO RTN ADDRESS, HL WILL BE RESTORED FM STACK
                       0555 *  SO THAT HL ARE RESTORED BEFORE DISPATCH.
                       0556 *
     C227              0557 DISPT  EQU    $       OFF TO A ROUTINE
C227 7E                0558           MOV    A,M     LO ADDR
C228 23                0559           INX    H
C229 66                0560           MOV    H,M     HI ADDR
C22A 6F                0561           MOV    L,A     HL NOW COMPLETE
     C22B              0562 DISP1  EQU    $       HERE TO GO OFF TO HL
C22B E3                0563           XTHL   .       XCHG HL W/HL ON STACK
C22C 7D                0564           MOV    A,L     ALSO COPY HERE FOR SETS
C22D C9                0565           RET    .       AND GO OFF TO THE RTN
                       0566 *
                       0567 *
                       0568 *   THIS ROUTINE SEARCHES THROUGH A TABLE, POINTED TO
                       0569 *  BY 'DE', FOR A DOUBLE CHARACTER MATCH OF THE 'HL'
                       0570 *  MEMORY CONTENT.  IF NO MATCH IS FOUND THE SCAN ENDS
                       0571 *  WITH HL POINTING TO ORIGINAL VALUE AND ZERO FLAG SET.
                       0572 *
C22E 11 3C C8          0573 FDCOU  LXI    D,CUTAB  HERE TO SCAN CUSTOM TBL ONLY
                       0574 *
C231 1A                0575 FDCOM  LDAX   D
C232 B7                0576           ORA    A       TEST FOR TABLE END
C233 C8                0577           RZ     .       NOT FOUND..COMMAND ERROR
C234 E5                0578           PUSH   H       SAVE START OF SCAN ADDRESS
C235 BE                0579           CMP    M       TEST FIRST CHR
C236 13                0580           INX    D
C237 C2 43 C2          0581           JNZ    NCOM
```

```
                       0582 *
C23A 23                0583           INX    H
C23B 1A                0584           LDAX   D
C23C BE                0585           CMP    M       NOW SECOND CHARACTER
C23D C2 43 C2          0586           JNZ    NCOM    GOODNESS
                       0587 *
C240 E1                0588           POP    H       RESTORE ORIGINAL SCAN ADDR
C241 B7                0589           ORA    A       SET NON-ZERO FLAG SAYING FOUND
C242 C9                0590           RET    .       WITH NON-ZERO SET
                       0591 *
                       0592 *
C243 13                0593 NCOM   INX    D       GO TO NEXT ENTRY
C244 13                0594           INX    D
C245 13                0595           INX    D
C246 E1                0596           POP    H       GET BACK ORIGINAL ADDRESS
C247 C3 31 C2          0597           JMP    FDCOM   CONTINUE SEARCH
                       0598 *
                       0599 *
                       0600 *                    COMMAND TABLE
                       0601 *
                       0602 *  THIS TABLE DESCRIBES THE VALID COMMANDS FOR SOLOS
                       0603 *
C24A 54 45             0604 COMTAB ASC    'TE'    TERMINAL MODE
C24C 67 C3             0605           DW     TERM
C24E 44 55             0606           ASC    'DU'    DUMP
C250 BF C3             0607           DW     DUMP
C252 45 4E             0608           ASC    'EN'    ENTR
C254 23 C4             0609           DW     ENTER
C256 45 58             0610           ASC    'EX'    EXEC
C258 5E C4             0611           DW     EXEC
C25A 47 45             0612           ASC    'GE'    GET A FILE
C25C A7 C4             0613           DW     TLOAD
C25E 53 41             0614           ASC    'SA'    SAVE A FILE
C260 E6 C4             0615           DW     TSAVE
C262 58 45             0616           ASC    'XE'    XEQ (EXECUTE) A FILE
C264 A6 C4             0617           DW     TXEQ
C266 43 41             0618           ASC    'CA'    CATALOG OF FILES
C268 2B C5             0619           DW     TLIST
C26A 53 45             0620           ASC    'SE'    SET COMMAND
C26C 7A C5             0621           DW     SET
C26E 43 55             0622           ASC    'CU'    CUSTOM COMMAND
C270 BD C5             0623           DW     CUSET
C272 00                0624           DB     0       END OF TABLE MARK
                       0625 *
                       0626 *
                       0627 *                DISPLAY DRIVER COMMAND TABLE
                       0628 *
                       0629 *     THIS TABLE DEFINES THE CHARACTERS FOR SPECIAL
                       0630 *  PROCESSING.  IF THE CHARACTER IS NOT IN THE TABLE IT
                       0631 *  GOES TO THE SCREEN.
                       0632 *
C273 0B                0633 TBL    DB     CLEAR-80H  SCREEN
C274 D5 C0             0634           DW     PERSE
```

```
C276 17                   DB      UP-80H  CURSOR
C277 04 C1                DW      PUP
C279 1A                   DB      DOWN-80H
C27A CB C0                DW      PDOWN
C27C 01                   DB      LEFT-80H
C27D 0B C1                DW      PLEFT
C27F 13                   DB      RIGHT-80H
C280 15 C1                DW      PRIT
C282 0E                   DB      HOME-80H
C283 E5 C0                DW      PHOME
C285 0D                   DB      CR      CARRIAGE RETURN
C286 47 C1                DW      PCR
C288 0A                   DB      LF      LINE FEED
C289 4D C1                DW      PLF
C28B 5F                   DB      BACKS   BACK SPACE
C28C 3E C1                DW      PBACK
C28E 1B                   DB      ESC     ESCAPE KEY
C28F 59 C1                DW      PESC
C291 00                   DB      0       END OF TABLE
                *
                *
                *     OUTPUT DEVICE TABLE
                *
C292 54 C0      OTAB      DW      VDMOT   VDM DRIVER
C294 4A C0                DW      SDROT   SERIAL OUTPUT
C296 E6 C2                DW      PROUT   PARALLEL OUTPUT
C298 D2 C2                DW      ERROT   ERROR OR USER DRIVER HANDLER
                *
                *
                *     INPUT DEVICE TABLE
                *
C29A 2E C0      ITAB      DW      KSTAT   KEYBOARD INPUT
C29C 42 C0                DW      SSTAT   SERIAL INPUT
C29E DD C2                DW      PASTAT   PARALLEL INPUT
C2A0 CB C2                DW      ERRIT   ERROR OR USER DRIVER HANDLER
                *
                *
                *   SECONDARY COMMAND TABLE FOR SET COMMAND
                *
C2A2 54 41      SETAB     ASC     'TA'    SET TAPE SPEED
C2A4 8E C5                DW      TASPD
C2A6 53 3D                ASC     'S='    SET DISPLAY SPEED
C2A8 99 C5                DW      DISPD
C2AA 49 3D                ASC     'I='    SET INPUT PORT
C2AC 9D C5                DW      SETIN
C2AE 4F 3D                ASC     'O='    SET OUTPUT PORT
C2B0 A1 C5                DW      SETOT
C2B2 4E 3D                ASC     'N='    NULLS
C2B4 B5 C5                DW      SETNU
C2B6 43 49                ASC     'CI'    SET CUSTOM DRIVER ADDRESS
C2B8 A5 C5                DW      SETCI
C2BA 43 4F                ASC     'CO'    SET CUSTOM OUTPUT DRIVER ADDRESS
C2BC A9 C5                DW      SETCO
```

```
C2BE 58 45                ASC    'XE'   SET HEADER XEQ ADDRESS
C2C0 B1 C5                DW     SETXQ
C2C2 54 59                ASC    'TY'   SET HEADER TYPE
C2C4 AD C5                DW     SETTY
C2C6 43 52                ASC    'CR'   SET CRC TO ALLOW IGNORING OF CRC ERRORS
C2C8 B9 C5                DW     SETCR
C2CA 00                   DB     0      END OF TABLE MARK
                *
                *
                *  SOLOS PORT ERROR HANDLER
                *
C2CB E5         ERRIT     PUSH   H      SAVE HL ONCE AGAIN
C2CC 2A 00 C8             LHLD   UIPRT  GET USER INPUT PORT ADDRESS
C2CF C3 D6 C2             JMP    ERRO1  AND GO PROCESS
                *
C2D2 E5         ERROT     PUSH   H
C2D3 2A 02 C8             LHLD   UOPRT  GET USER OUTPUT PORT ADDRESS
C2D6 7D         ERRO1     MOV    A,L    TEST HL FOR ZERO
C2D7 B4                   ORA    H
C2D8 CA C0 C1             JZ     COMN1  IF ZERO RETURN TO COMMAND MODE
C2DB E3                   XTHL   .      ADDRESS TO STACK...OLD HL TO HL
C2DC C9                   RET    .      GO TO THE DRIVER
                * -*-
                          COPY   SOLOS2/1                            2 OF 3 ******
                *
                *   THIS ROUTINE IS THE PARALLEL DEVICE HANDLER
                *  NO PROVISION IS MADE FOR CONTROLLING THE PORT
                *  CONTROL BIT.
                *
                *
                *     PARALLEL INPUT DRIVER
                *
C2DD DB FA     PASTAT     IN     STAPT
C2DF 2F                   CMA    .      INVERT STATUS FLAGS
C2E0 E6 02                ANI    PDR  * TEST BIT
C2E2 C8                   RZ     .      WITH FLAG SET
C2E3 DB FD                IN     PDATA  GET DATA
C2E5 C9                   RET
                *
                *    PARALLEL OUTPUT HANDLER
                *
C2E6 DB FA      PROUT     IN     STAPT  GET STATUS
C2E8 E6 04                ANI    PXDR   TEST IF DEVICE IS READY
C2EA C2 E6 C2             JNZ    PROUT  LOOP UNTIL SO
C2ED 78                   MOV    A,B
C2EE D3 FD                OUT    PDATA
C2F0 C9                   RET
                *
                *
                *      OUTPUT A CRLF FOLLOWED BY A PROMPT
                *
C2F1 CD F9 C2   PROMPT    CALL   CRLF
C2F4 06 3E                MVI    B,'>'  THE PROMPT
```

```
C2F6 C3 19 C0         0740         JMP    SOUT   PUT IT ON THE SCREEN
                      0741 *
                      0742 *
C2F9 06 0A            0743 CRLF    MVI    B,LF   LINE FEED
C2FB CD 19 C0         0744         CALL   SOUT
C2FE 06 0D            0745         MVI    B,CR   CARRIAGE RETURN
C300 CD 19 C0         0746         CALL   SOUT
                      0747 *  NOW OUTPUT THE NULLS
C303 3A 10 C8         0748         LDA    NUCNT  GET DESIRED COUNT
C306 4F               0749         MOV    C,A    STORE IN C
C307 0D               0750 NULOT   DCR    C
C308 F8               0751         RM     .      RETURN WHEN PAST ZERO
C309 AF               0752         XRA    A      GET A NULL
C30A CD 1F C4         0753         CALL   OUTH
C30D C3 07 C3         0754         JMP    NULOT
                      0755 *
                      0756 *
                      0757 *   SCAN OFF OPTIONAL PARAMETER.  IF PRESENT RETURN WITH
                      0758 * VALUE IN HL AND COPY OF "L" IN "A".  IF NOT PRESENT
                      0759 * RETURN WITH A "1" IN "A" AND HL UNTOUCHED.
                      0760 *
C310 CD 1B C3         0761 PSCAN   CALL   SBLK
C313 3E 01            0762         MVI    A,1    DEFAULT VALUE
C315 C8               0763         RZ     .      IF NONE
C316 CD 40 C3         0764         CALL   SHEX   CONVERT VALUE
C319 7D               0765         MOV    A,L    GET LOWER HALF
C31A C9               0766         RET
                      0767 *
                      0768 *
                      0769 *  SCAN OVER UP TO 12 CHARACTERS LOOKING FOR A BLANK
                      0770 *
C31B 0E 0C            0771 SBLK    MVI    C,12   MAXIMUM COMMAND STRING
C31D 1A               0772 SBLK1   LDAX   D
C31E FE 20            0773         CPI    BLANK
C320 CA 2E C3         0774         JZ     SCHR   GOT A BLANK NOW SCAN PAST IT
C323 13               0775         INX    D
C324 FE 3D            0776         CPI    '='    ALSO ALLOW AN EQUAL TO STOP US
C326 CA 2E C3         0777         JZ     SCHR   IF SO, PTR AT CHAR FOLLOWING
C329 0D               0778         DCR    C      NO MORE THAN TWELVE
C32A C2 1D C3         0779         JNZ    SBLK1
C32D C9               0780         RET    .      GO BACK WITH ZERO FLAG SET
                      0781 *
                      0782 *
                      0783 *    SCAN PAST UP TO 10 BLANK POSITIONS LOOKING FOR
                      0784 *  A NON BLANK CHARACTER.
                      0785 *
C32E 0E 0A            0786 SCHR    MVI    C,10   SCAN TO FIRST NON BLANK CHR WITHIN 10
C330 1A               0787 SCHR1   LDAX   D      GET NEXT CHARACTER
C331 FE 20            0788         CPI    SPACE
C333 C0               0789         RNZ    .      WE'RE PAST THEM
C334 13               0790         INX    D      NEXT SCAN ADDRESS
C335 0D               0791         DCR    C
C336 C8               0792         RZ     .      COMMAND ERROR
```

```
C337 C3 30 C3         0793         JMP    SCHR1  KEEP LOOPING
                      0794 *
                      0795 *
                      0796 *    THIS ROUTINE SCANS OVER CHARACTERS, PAST BLANKS AND
                      0797 * CONVERTS THE FOLLOWING VALUE TO HEX.  ERRORS RETURN TO
                      0798 * THE ERROR HANDLER.
                      0799 *
C33A CD 1B C3         0800 SCONV   CALL   SBLK   FIND IF VALUE IS PRESENT
C33D CA 80 C4         0801         JZ     ERR1   ABORT TO ERROR IF NONE
                      0802 *
                      0803 *
                      0804 *  THIS ROUTINE CONVERTS ASCII DIGITS INTO BINARY FOLLOWING
                      0805 * A STANDARD HEX CONVERSION.  THE SCAN STOPS WHEN AN ASCII
                      0806 * SPACE IS ENCOUNTERED.  PARAMETER ERRORS REPLACE THE ERROR
                      0807 * CHARACTER ON THE SCREEN WITH A QUESTION MARK.
                      0808 *
C340 21 00 00         0809 SHEX    LXI    H,0    CLEAR H & L
C343 1A               0810 SHE1    LDAX   D      GET CHARACTER
C344 FE 20            0811         CPI    20H    IS IT A SPACE?
C346 C8               0812         RZ     .      IF SO
C347 FE 2F            0813         CPI    '/'    SLASH IS ALSO LEGAL
C349 C8               0814         RZ
C34A FE 3A            0815         CPI    ':'    EVEN THE COLON IS ALLOWED
C34C C8               0816         RZ
                      0817 *
C34D 29               0818 HCONV   DAD    H      MAKE ROOM FOR THE NEW ONE
C34E 29               0819         DAD    H
C34F 29               0820         DAD    H
C350 29               0821         DAD    H
C351 CD 5D C3         0822         CALL   HCOV1  DO THE CONVERSION
C354 D2 80 C4         0823         JNC    ERR1   NOT VALID HEXIDECIMAL VALUE
C357 85               0824         ADD    L
C358 6F               0825         MOV    L,A    MOVE IT IN
C359 13               0826         INX    D      BUMP THE POINTER
C35A C3 43 C3         0827         JMP    SHE1
                      0828 *
C35D D6 30            0829 HCOV1   SUI    48     REMOVE ASCII BIAS
C35F FE 0A            0830         CPI    10
C361 D8               0831         RC     .      IF LESS THAN 9
C362 D6 07            0832         SUI    7      IT'S A LETTER
C364 FE 10            0833         CPI    10H
C366 C9               0834         RET    .      WITH TEST IN HAND
                      0835 *
                      0836 *
                      0837 *                    TERM COMMAND
                      0838 *
                      0839 *   THIS ROUTINE GETS CHARACTERS FROM THE SYSTEM KEYBOARD
                      0840 *  AND OUTPUTS THEM TO THE SELECTED OUTPUT PORT.  IT IS
                      0841 *  INTENDED TO CONFIGURE THE Sol AS A STANDARD VIDEO
                      0842 *  TERMINAL.  COMMAND KEYS ARE NOT OUTPUT TO THE OUTPUT
                      0843 *  PORT BUT ARE INTERPRETED AS DIRECT Sol COMMANDS.
                      0844 *  THE MODE COMMAND, RECEIVED BY THE KEYBOARD, PUTS THE
                      0845 *  Sol IN THE COMMAND MODE.
```

```
                    0846 *
                    0847 *
                    0848 *
C367 CD 10 C3       0849 TERM   CALL   PSCAN   FIND IF INPUT PARAMETER IS PRESENT
C36A 32 06 C8       0850        STA    IPORT   SINP WILL USE THIS DRIVER (DEFAULT IS 1)
C36D CD 10 C3       0851        CALL   PSCAN   NOW FOR THE OUTPUT DRIVER
C370 32 07 C8       0852        STA    OPORT
                    0853 *
C373 CD 2B C0       0854 TERM1  CALL   KSTAT   IS THERE ONE WAITING?
C376 CA 8B C3       0855        JZ     TIN     IF NOT
C379 47             0856        MOV    B,A     SAVE IT IN B
C37A FE 80          0857        CPI    MODE    IS IT MODE
C37C CA C0 C1       0858        JZ     COMN1   YES--RESET AND QUIT TERM
C37F DA 88 C3       0859        JC     TOUT    NON-CURSOR KEY---SEND TO TERM PORT
C382 CD 54 C0       0860        CALL   VDMOT   PROCESS IT
C385 C3 8B C3       0861        JMP    TIN
                    0862 *
C388 CD 19 C0       0863 TOUT   CALL   SOUT    OUTPUT IT TO THE SERIAL PORT
C38B CD 1F C0       0864 TIN    CALL   SINP    GET INPUT STATUS
C38E CA 73 C3       0865        JZ     TERM1   LOOP IF NOT
C391 E6 7F          0866        ANI    7FH     NO HIGH BITS FROM HERE
C393 CA 73 C3       0867        JZ     TERM1   A NULL IS IGNORED
C396 47             0868        MOV    B,A     IT'S OUTPUT FROM 'B'
C397 FE 1B          0869        CPI    1BH     IS IT A CONTROL CHAR TO BE IGNORED
C399 D2 B9 C3       0870        JNC    TERM2   NO--TO VDM AS IS THEN
C39C FE 0D          0871        CPI    CR      CR OR LF ARE SPECIAL CASES THOUGH
C39E CA B9 C3       0872        JZ     TERM2   AND MUST BE PASSED STD MODE TO VDM DRIVER
C3A1 FE 0A          0873        CPI    LF
C3A3 CA B9 C3       0874        JZ     TERM2
C3A6 3A 0C C8       0875        LDA    ESCFL   A CTL CHAR---ARE WE W/IN AN ESC SEQUENCE?
C3A9 B7             0876        ORA    A       IF YES, THEN OUTPUT CTL CHAR DIRECTLY TO VDM
C3AA C2 B9 C3       0877        JNZ    TERM2   WE SURE ARE, LET VDM DRIVER HANDLE IT
C3AD C5             0878        PUSH   B       SAVE THE CHAR
C3AE 06 1B          0879        MVI    B,ESC   CTL CHAR TO VDM VIA ESC SEQUENCE
C3B0 CD 54 C0       0880        CALL   VDMOT
C3B3 06 07          0881        MVI    B,7     SAY TO PUT OUT NEXT CHAR AS IS
C3B5 CD 54 C0       0882        CALL   VDMOT   ALMOST READY
C3B8 C1             0883        POP    B       RESTORE CHAR
     C3B9           0884 TERM2  EQU    $       ALL READY TO OUTPUT THE CHAR
C3B9 CD 54 C0       0885        CALL   VDMOT   PUT IT ON THE SCREEN
C3BC C3 73 C3       0886        JMP    TERM1   LOOP OVER AND OVER
                    0887 *
                    0888 *
                    0889 *
                    0890 *                   DUMP COMMAND
                    0891 *
                    0892 *      THIS ROUTINE DUMPS CHARACTERS FROM MEMORY TO THE
                    0893 *   CURRENT OUTPUT DEVICE.  ALL VALUES ARE DISPLAYED AS
                    0894 *   ASCII HEX.
                    0895 *
                    0896 *   THE COMMAND FORM IS AS FOLLOWS:
                    0897 *
                    0898 *                 DUmp  addr1  addr2
```

```
                    0899 *
                    0900 *     THE VALUES FROM ADDR1 TO ADDR2 ARE THEN OUTPUT TO THE
                    0901 *  OUTPUT DEVICE.  IF ONLY ADDR1 IS SPECIFIED THEN THE
                    0902 *  VALUE AT THAT ADDRESS IS OUTPUT.
                    0903 *
C3BF CD 3A C3       0904 DUMP   CALL   SCONV   SCAN TO FIRST ADDRESS AND CONVERT IT
C3C2 E5             0905        PUSH   H       SAVE THE VALUE
C3C3 CD 10 C3       0906        CALL   PSCAN   SEE IF SECOND WAS GIVEN
C3C6 D1             0907        POP    D       GET BACK START
C3C7 EB             0908        XCHG   .       HL HAS START, DE HAS END
                    0909 *
C3C8 CD F9 C2       0910 DLOOP  CALL   CRLF
C3CB CD E8 C3       0911        CALL   ADOUT   OUTPUT ADDRESS
C3CE CD 06 C4       0912        CALL   BOUT    ANOTHER SPACE TO KEEP IT PRETTY
C3D1 0E 10          0913        MVI    C,16    VALUES PER LINE
                    0914 *
C3D3 7E             0915 DLP1   MOV    A,M     GET THE CHR
C3D4 C5             0916        PUSH   B       SAVE VALUE COUNT
C3D5 CD ED C3       0917        CALL   HBOUT   SEND IT OUT WITH A BLANK
C3D8 7D             0918        MOV    A,L     COMPARE DE & HL
C3D9 93             0919        SUB    E
C3DA 7C             0920        MOV    A,H
C3DB 9A             0921        SBB    D
C3DC D2 C9 C1       0922        JNC    COMND   ALL DONE
C3DF C1             0923        POP    B       VALUES PER LINE
C3E0 23             0924        INX    H
C3E1 0D             0925        DCR    C       BUMP THE LINE COUNT
C3E2 C2 D3 C3       0926        JNZ    DLP1    NOT ZERO IF MORE FOR THIS LINE
C3E5 C3 C8 C3       0927        JMP    DLOOP   DO A LFCR BEFORE THE NEXT
                    0928 *
                    0929 *
                    0930 *     OUTPUT HL AS HEX 16 BIT VALUE
                    0931 *
C3E8 7C             0932 ADOUT  MOV    A,H     H FIRST
C3E9 CD 0B C4       0933        CALL   HEOUT
C3EC 7D             0934        MOV    A,L     THEN "L" FOLLOWED BY A SPACE
                    0935 *
C3ED CD 0B C4       0936 HBOUT  CALL   HEOUT
C3F0 CD 1F C0       0937        CALL   SINP    SEE IF A CHAR WAITING
C3F3 CA 06 C4       0938        JZ     BOUT    NO
C3F6 E6 7F          0939        ANI    7FH     CLR PARITY 1ST THO
C3F8 CA C9 C1       0940        JZ     COMND   EITHER MODE OR CTL-@
C3FB FE 20          0941        CPI    ' '     IS IT A SPACE
C3FD C2 06 C4       0942        JNZ    BOUT    NO--IGN THE CHAR
C400 CD 1F C0       0943 WTLP1  CALL   SINP    IF SPACE, WAIT UNTIL ANY OTHER KEY HIT
C403 CA 00 C4       0944        JZ     WTLP1   THIS ALLOWS LOOKING AT THE DISPLAY
C406 06 20          0945 BOUT   MVI    B,' '
C408 C3 19 C0       0946        JMP    SOUT    PUT IT OUT
                    0947 *
C40B 4F             0948 HEOUT  MOV    C,A     GET THE CHARACTER
C40C 0F             0949        RRC
C40D 0F             0950        RRC            MOVE THE HIGH FOUR DOWN
C40E 0F             0951        RRC
```

```
C40F 0F                          RRC
C410 CD 14 C4                    CALL   HEOU1  PUT THEM OUT
C413 79                          MOV    A,C    THIS TIME THE LOW FOUR
                    *
C414 E6 0F              HEOU1    ANI    0FH    FOUR ON THE FLOOR
C416 C6 30                       ADI    48     WE WORK WITH ASCII HERE
C418 FE 3A                       CPI    58     0-9?
C41A DA 1F C4                    JC     OUTH   YUP!
C41D C6 07                       ADI    7      MAKE IT A LETTER
C41F 47                 OUTH     MOV    B,A    OUTPUT IT FROM REGISTER 'B'
C420 C3 19 C0                    JMP    SOUT
                    *
                    *
                    *                          ENTR COMMAND
                    *
                    *   THIS ROUTINE GETS VALUES FROM THE KEYBOARD AND ENTERS
                    * THEM INTO MEMORY.  THE INPUT VALUES ARE SCANNED FOLLOWING
                    * A STANDARD 'GCLIN' INPUT SO ON SCREEN EDITING MAY TAKE
                    * PLACE PRIOR TO THE LINE TERMINATOR.  A BACK SLASH '/'
                    * ENDS THE ROUTINE AND RETURNS CONTROL TO THE COMMAND MODE.
                    * A COLON ':' SETS THE PREVIOUS VALUE AS A NEW ADDRESS FOR
                    * ENTER.
                    *
C423 CD 3A C3           ENTER    CALL   SCONV  SCAN OVER CHARS AND GET ADDRESS
C426 E5                          PUSH   H      SAVE ADDRESS
C427 AF                          XRA    A
C428 32 07 C8                    STA    OPORT  ENTER VALUES TO SCREEN BUFFER
                    *
C42B CD F9 C2           ENLOP    CALL   CRLF
C42E 06 3A                       MVI    B,':'
C430 CD FF C1                    CALL   CONT   GET LINE OF INPUT
C433 CD 36 C1                    CALL   CREM   REMOVE THE CURSOR
C436 0E 01                       MVI    C,1    START SCAN
C438 CD 20 C1                    CALL   VDAD2  GET ADDRESS
C43B EB                          XCHG   .      ....TO DE
                    *
                    *
C43C 0E 03              ENLO1    MVI    C,3    NO MORE THAN THREE SPACES BETWEEN VALUES
C43E CD 30 C3                    CALL   SCHR1  SCAN TO NEXT VALUE
C441 CA 2B C4                    JZ     ENLOP  LAST ENTRY FOUND START NEW LINE
                    *
C444 FE 2F                       CPI    '/'    COMMAND TERMINATOR?
C446 CA C0 C1                    JZ     COMN1  IF SO...RETURN TO STANDARD INPUT
C449 CD 40 C3                    CALL   SHEX   CONVERT VALUE
C44C FE 3A                       CPI    ':'    ADDRESS TERMINATOR?
C44E CA 59 C4                    JZ     ENLO3  GO PROCESS IF SO
C451 7D                          MOV    A,L    GET LOW PART AS CONVERTED
C452 E1                          POP    H      GET MEMORY ADDRESS
C453 77                          MOV    M,A    PUT IN THE VALUE
C454 23                          INX    H
C455 E5                          PUSH   H      BACK GOES THE ADDRESS
C456 C3 3C C4                    JMP    ENLO1  CONTINUE THE SCAN
                    *
```

```
C459 E3                 ENLO3    XTHL   .      PUT NEW ADDRESS ON STACK
C45A 13                          INX    D      MOVE SCAN PAST TERMINATOR
C45B C3 3C C4                    JMP    ENLO1
                    *
                    *
                    *                          EXECUTE COMMAND
                    *
                    *    THIS ROUTINE GETS THE FOLLOWING PARAMETER AND DOES A
                    *  PROGRAM JUMP TO THE LOCATION GIVEN BY IT.  IF PROPER
                    *  STACK OPERATIONS ARE USED WITHIN THE EXTERNAL PROGRAM
                    *  IT CAN DO A STANDARD 'RET'URN TO THE SOLOS COMMAND MODE.
                    *  THE STARTING ADDRESS OF SOLOS IS PASSED TO THE PROGRAM
                    *  IN REGISTER PAIR HL SO IT CAN ADJUST INTERNAL PARAMETERS
                    *  FOR SOLOS OPERATION.
                    *
                    *
C45E CD 3A C3           EXEC     CALL   SCONV  SCAN PAST BLANKS AND GET PARAMETER
C461 E5                 EXEC1    PUSH   H      PUT GO ADDRESS ON STACK
C462 21 00 C0                    LXI    H,START  TELL THE PROGRAM WHERE WE CAME FROM
C465 C9                          RET    .      AND DISPATCH TO IT
                    *
                    *
                    *   THIS ROUTINE GETS A NAME OF UP TO 5 CHARACTERS
                    *  FROM THE INPUT STRING.  IF THE TERMINATOR IS A
                    *  SLASH (/) THEN THE CHARACTER FOLLOWING IS TAKEN
                    *  AS THE CASSETTE UNIT SPECIFICATION.
                    *
                    *
C466 21 1C C8           NAMES    LXI    H,THEAD  POINT TO INTERNAL HEADER
C469 CD 1B C3           NAME     CALL   SBLK   SCAN OVER TO FIRST CHRS
C46C 06 06                       MVI    B,6    UP TO SIX ARE ACCEPTED
                    *
C46E 1A                 NAME1    LDAX   D      GET CHARACTER
C46F FE 20                       CPI    ' '    NO UNIT DELIMITER
C471 CA 86 C4                    JZ     NFIL
C474 FE 2F                       CPI    '/'    UNIT DELIMITER
C476 CA 86 C4                    JZ     NFIL
C479 77                          MOV    M,A
C47A 13                          INX    D      BUMP THE SCAN POINTER
C47B 23                          INX    H
C47C 05                          DCR    B
C47D C2 6E C4                    JNZ    NAME1  FALL THROUGH TO ERR1 IF TOO MANY CHRS IN NA
                    *
                    *
                    *    SOLOS ERROR HANDLER
                    *
C480 EB                 ERR1     XCHG   .      GET SCAN ADDRESS TO HL
C481 36 3F              ERR2     MVI    M,'?'  PUT QUESTION MARK ON SCREEN
C483 C3 C0 C1                    JMP    COMN1  AND RETURN TO COMMAND MODE
                    *
                    *
                    *  HERE WE HAVE SCANNED OFF THE NAME.  ZERO FILL FOR
                    *  NAMES LESS THAN FIVE CHARACTERS.
```

```
                          1058 *
C486 36 00                1059 NFIL   MVI    M,0    PUT IN AT LEAST ONE ZERO
C488 23                   1060        INX    H
C489 05                   1061        DCR    B
C48A C2 86 C4             1062        JNZ    NFIL   LOOP UNTIL B IS ZERO
                          1063 *
C48D FE 2F                1064        CPI    '/'    IS THERE A UNIT SPECIFICATION?
C48F 3E 01                1065        MVI    A,1    PRETEND NOT
C491 C2 9A C4             1066        JNZ    DEFLT
C494 13                   1067        INX    D      MOVE PAST THE TERMINATOR
C495 CD 2E C3             1068        CALL   SCHR   GO GET UNIT SPEC
C498 D6 30                1069        SUI    '0'    REMOVE ASCII BIAS
                          1070 *
     C49A                 1071 DEFLT  EQU    $      MOVE OVER TO INTERNAL REPRESENTATION
C49A E6 01                1072        ANI    1      JUST BIT ZERO
C49C 3E 80                1073        MVI    A,TAPE1  ASSUME TAPE ONE
C49E C2 A2 C4             1074        JNZ    STUNT  IF NON-ZERO, ITS ONE
C4A1 1F                   1075        RAR
C4A2 32 54 C8             1076 STUNT  STA    FNUMF  SET IT IN
C4A5 C9                   1077        RET
                          1078 *
                          1079 *
                          1080 *
                          1081 *   THIS ROUTINE PROCESSES THE XEO AND GET COMMANDS
                          1082 *
                          1083 *
C4A6 3E                   1084 TXEQ   DB     3EH    THIS BEGINS "MVI A,0AFH"
C4A7 AF                   1085 TLOAD  XRA    A      A=0 MEANS TLOAD, ELSE TXEQ
C4A8 F5                   1086        PUSH   PSW    SAVE FLAG FOR LATER
C4A9 21 2C C8             1087        LXI    H,DHEAD  PLACE DUMMY HEADER HERE
C4AC CD 69 C4             1088        CALL   NAME   SET IN NAME AND UNIT
C4AF 21 00 00             1089        LXI    H,0    PRETEND NO SECOND VALUE
C4B2 CD 10 C3             1090        CALL   PSCAN  GO GET THE ADDRESS (IF PRESENT)
                          1091 *
C4B5 EB                   1092 TLOA2  XCHG   .      PUT ADDRESS IN DE
C4B6 21 2C C8             1093        LXI    H,DHEAD  PT TO DUMMY HEADER W/ NAME TO LOAD
C4B9 7E                   1094        MOV    A,M    SEE IF A NAME WAS ENTERED
C4BA B7                   1095        ORA    A      IS THERE A NAME?
C4BB C2 C1 C4             1096        JNZ    TLOA3  YES--SEARCH FOR IT
C4BE 21 1C C8             1097        LXI    H,THEAD  NO NAME, LOAD 1ST FILE
C4C1 E5                   1098 TLOA3  PUSH   H      SAVE PTR TO NAME TO LOAD
C4C2 CD 48 C5             1099        CALL   ALOAD  GET UNIT AND SPEED
C4C5 E1                   1100        POP    H      RESTORE PTR TO HDR TO LOAD
C4C6 CD CB C6             1101        CALL   RTAPE  READ IN THE TAPE
C4C9 DA 14 C5             1102        JC     TAERR  TAPE ERROR?
                          1103 *
C4CC CD 50 C5             1104        CALL   NAOUT  PUT OUT THE HEADER PARAMETERS
C4CF F1                   1105        POP    PSW    RESTORE FLAG FROM ORIGINAL ENTRY
C4D0 B7                   1106        ORA    A
C4D1 C8                   1107        RZ     .      AUTO XEQ NOT WANTED
C4D2 3A 22 C8             1108        LDA    HTYPE  CHECK TYPE
C4D5 B7                   1109        ORA    A      SET FLAGS
C4D6 FA 14 C5             1110        JM     TAERR  TYPE IS NON XEQ
```

```
C4D9 3A 21 C8             1111        LDA    THEAD+5  GET CHARACTER PAST NAME
C4DC B7                   1112        ORA    A
C4DD C2 14 C5             1113        JNZ    TAERR  THE BYTE MUST BE ZERO FOR AUTO XEO
C4E0 2A 27 C8             1114        LHLD   XEQAD  GET THE TAPE ADDRESS
C4E3 C3 61 C4             1115        JMP    EXEC1  AND GO TO IT
                          1116 *
                          1117 *
                          1118 *                 =- GET -=
                          1119 *
                          1120 *  THIS ROUTINE IS USED TO SAVE PROGRAMS AND DATA ON
                          1121 *  THE CASSETTE UNIT.
                          1122 *
                          1123 *
C4E6 CD 66 C4             1124 TSAVE  CALL   NAMES  GET NAME AND UNIT
C4E9 CD 3A C3             1125        CALL   SCONV  GET START ADDRESS
C4EC E5                   1126        PUSH   H      USE THE STACK AS A REGISTER
C4ED CD 3A C3             1127        CALL   SCONV  GET END ADDRESS
C4F0 E3                   1128        XTHL   .      PUT END ON STACK, GET BACK START
C4F1 E5                   1129        PUSH   H      SAVE START ON TOP OF STACK
C4F2 CD 10 C3             1130        CALL   PSCAN  SEE IF OPTIONAL HEADER ADDRESS WAS GIVEN
C4F5 22 25 C8             1131        SHLD   LOADR  PUT HEADER ADDRESS IN PLACE
                          1132 *
C4F8 E1                   1133        POP    H      "FROM" ADDRESS TO HL
C4F9 D1                   1134        POP    D      GET BACK "END" ADDRESS
C4FA E5                   1135        PUSH   H      SAVE FROM AGAIN FOR LATER
C4FB 7B                   1136        MOV    A,E    NOW CALCULATE SIZE
C4FC 95                   1137        SUB    L      SIZE=END-START+1
C4FD 6F                   1138        MOV    L,A
C4FE 7A                   1139        MOV    A,D
C4FF 9C                   1140        SBB    H
C500 67                   1141        MOV    H,A
C501 23                   1142        INX    H
C502 22 23 C8             1143        SHLD   BLOCK  STORE THE SIZE
C505 E5                   1144        PUSH   H      SAVE IT FOR THE READ ALSO
                          1145 *
C506 CD 48 C5             1146        CALL   ALOAD  GET UNIT AND SPEED
C509 21 1C C8             1147        LXI    H,THEAD  POINT TO HEADER
C50C CD AF C7             1148        CALL   WHEAD  AND WRITE IT OUT
                          1149 *  NOW WRITE OUT THE DATA
C50F D1                   1150        POP    D      GET SIZE TO DE
C510 E1                   1151        POP    H      GET BACK "FROM" ADDRESS
C511 C3 90 C7             1152        JMP    WRLO1  WRITE OUT THE DATA AND RETURN
                          1153 *
                          1154 *
                          1155 *  OUTPUT ERROR AND HEADER
                          1156 *
C514 CD F9 C2             1157 TAERR  CALL   CRLF
C517 16 06                1158        MVI    D,6
C519 21 25 C5             1159        LXI    H,ERRM  POINT TO ERROR MESSAGE
C51C CD 6A C5             1160        CALL   NLOOP  OUTPUT ERROR
C51F CD 50 C5             1161        CALL   NAOUT  THEN THE HEADER
C522 C3 C0 C1             1162        JMP    COMN1  AND BE SURE THE TAPE UNITS ARE OFF
                          1163 *
```

```
C525 45 52 52 4F      1164 ERRM   ASC    !ERROR !
     52 20
                      1165 *
                      1166 *
                      1167 *   THIS ROUTINE READS HEADERS FROM THE TAPE AND OUTPUTS
                      1168 *   THEM TO THE OUTPUT DEVICE.  IT CONTINUES UNTIL THE
                      1169 *   MODE KEY IS DEPRESSED.
                      1170 *
C52B CD 66 C4         1171 TLIST  CALL   NAMES  SET UP UNIT IF GIVEN
C52E CD F9 C2         1172        CALL   CRLF
                      1173 *
                      1174 *
C531 CD 48 C5         1175 LLIST  CALL   ALOAD
C534 06 01            1176        MVI    B,1
C536 CD EF C7         1177        CALL   TON    TURN ON THE TAPE
                      1178 *
C539 CD 23 C7         1179 LIST1  CALL   RHEAD
C53C DA CC C1         1180        JC     COMN1  TURN OFF THE TAPE UNIT
C53F C2 39 C5         1181        JNZ    LIST1
C542 CD 50 C5         1182        CALL   NAOUT  OUTPUT THE HEADER
C545 C3 39 C5         1183        JMP    LIST1  LOOP UNTIL MODE IS DEPRESSED
                      1184 *
                      1185 *
                      1186 *   THIS ROUTINE GETS THE CASSETTE UNIT NUMBER AND
                      1187 *   SPEED TO REGISTER "A" FOR THE TAPE CALLS
                      1188 *
C548 21 54 C8         1189 ALOAD  LXI    H,FNUMF  POINT TO THE UNIT SPECIFICATION
C54B 3A 0D C8         1190        LDA    TSPD   GET THE TAPE SPEED
C54E B6               1191        ORA    M      PUT THEM TOGETHER
C54F C9               1192        RET    .      AND GO BACK
                      1193 *
                      1194 *
                      1195 *   THIS ROUTINE OUTPUTS THE NAME AND PARAMETERS OF
                      1196 *   THEAD TO THE OUTPUT DEVICE.
                      1197 *
                      1198 *
C550 16 08            1199 NAOUT  MVI    D,8
C552 21 1B C8         1200        LXI    H,THEAD-1  POINT TO THE HEADER
C555 CD 6A C5         1201        CALL   NLOOP  OUTPUT THE HEADER
C558 CD 06 C4         1202        CALL   BOUT   ANOTHER BLANK
C55B 2A 25 C8         1203        LHLD   LOADR  NOW THE LOAD ADDRESS
C55E CD E8 C3         1204        CALL   ADOUT  PUT IT OUT
C561 2A 23 C8         1205        LHLD   BLOCK  AND THE BLOCK SIZE
C564 CD E8 C3         1206        CALL   ADOUT
C567 C3 F9 C2         1207        JMP    CRLF   DO THE CRLF AND RETURN
                      1208 *
                      1209 *
C56A 7E               1210 NLOOP  MOV    A,M    GET CHARACTER
C56B B7               1211        ORA    A
C56C C2 71 C5         1212        JNZ    CHRLI  IF IT ISN'T A ZERO
C56F 3E 20            1213        MVI    A,' '
C571 CD 1F C4         1214 CHRLI  CALL   OUTH   OUTPUT CHAR NOW
C574 23               1215        INX    H
```

```
C575 15               1216        DCR    D
C576 C2 6A C5         1217        JNZ    NLOOP
C579 C9               1218        RET
                      1219 *
                      1220 *
                      1221 *
                      1222 *
                      1223 *      "SET" COMMAND
                      1224 *
                      1225 *   THIS ROUTINE GETS THE ASSOCIATED PARAMETER AND
                      1226 *   DISPATCHES TO THE PROPER ROUTINE FOR SETTING
                      1227 *   GLOBAL VALUES.
                      1228 *
     C57A             1229 SET    EQU    $      THIS IS THE SET COMMAND
C57A CD 1B C3         1230        CALL   SBLK   LOOK FOR SET NAME
C57D CA 80 C4         1231        JZ     ERR1   MUST HAVE AT LEAST SOMETHING!!
C580 D5               1232        PUSH   D      SAVE SCAN ADDRESS
C581 CD 3A C3         1233        CALL   SCONV  CONVERT FOLLOWING VALUE
C584 E3               1234        XTHL   .      GET SCAN ADDRESS BACK..SAVE VALUE ON STACK
C585 11 A2 C2         1235        LXI    D,SETAB  SECONDARY COMMAND TABLE
C588 CD 31 C2         1236        CALL   FDCOM  SEE IF IN TABLE
C58B C3 22 C2         1237        JMP    DISPO  AND EITHER ERR OR OFF TO IT
                      1238 *
                      1239 *
                      1240 *  THIS ROUTINE SETS THE TAPE SPEED
                      1241 *
C58E B7               1242 TASPD  ORA    A      IS IT ZERO?
C58F CA 94 C5         1243        JZ     SETSP  YES--THAT'S A VALID SPEED
C592 3E 20            1244        MVI    A,32   SET TO SLOW IF NON-ZERO
C594 32 0D C8         1245 SETSP  STA    TSPD   SPEED IS STORED HERE
C597 C9               1246        RET
                      1247 *
                      1248 *
C598 78               1249 STSPD  MOV    A,B    ESCAPE COMES HERE TO SET SPEED
C599 32 0B C8         1250 DISPD  STA    SPEED  SET DISPLAY SPEED
C59C C9               1251        RET    .
                      1252 *
                      1253 *  SET INPUT DRIVER
                      1254 *
     C59D             1255 SETIN  EQU    $
C59D 32 06 C8         1256        STA    IPORT
C5A0 C9               1257        RET
                      1258 *
                      1259 *  SET OUTPUT DRIVER
                      1260 *
     C5A1             1261 SETOT  EQU    $
C5A1 32 07 C8         1262        STA    OPORT
C5A4 C9               1263        RET
                      1264 *
                      1265 *   SET USERS CUSTOM INPUT DRIVER ADDRESS
                      1266 *
C5A5 22 00 C8         1267 SETCI  SHLD   UIPRT
C5A8 C9               1268        RET
```

```
                  1269 *
                  1270 *  SET USERS CUSTOM OUTPUT DRIVER ADDRESS
                  1271 *
C5A9 22 02 C8     1272 SETCO  SHLD   UOPRT
C5AC C9           1273        RET
                  1274 *
                  1275 *  SET TYPE BYTE INTO HEADER
                  1276 *
C5AD 32 22 C8     1277 SETTY  STA    HTYPE
C5B0 C9           1278        RET
                  1279 *
                  1280 *   SET EXECUTE ADDRESS INTO HEADER
                  1281 *
C5B1 22 27 C8     1282 SETXQ  SHLD   XEQAD
C5B4 C9           1283        RET
                  1284 *
                  1285 *
C5B5 32 10 C8     1286 SETNU  STA    NUCNT  SET THE NULL COUNT
C5B8 C9           1287        RET    .      THAT'S DONE
                  1288 *
                  1289 *
   C5B9           1290 SETCR  EQU    $      SET TO IGNORE CRC ERRORS
C5B9 32 11 C8     1291        STA    IGNCR  FF=IGNORE ERRORS, ELSE=NORMAL
C5BC C9           1292        RET    .
                  1293 *
                  1294 *
                  1295 *
                  1296 *   CUSTOM COMMAND NAME AND ADDRESS INTO CUSTOM COMMAND
                  1297 *
C5BD CD 66 C4     1298 CUSET  CALL   NAMES  CUSTOM COMMAND ENTRY/REMOVAL
C5C0 21 C9 C1     1299        LXI    H,COMND  DEFAULT ADDR IF NONE GIVEN
C5C3 CD 10 C3     1300        CALL   PSCAN  GET RTN ADDR
C5C6 E5           1301        PUSH   H      SAVE RTN ADDR
C5C7 21 1C C8     1302        LXI    H,THEAD  PT AT NAME TO SEARCH
C5CA CD 2E C2     1303        CALL   FDCOU  SEARCH IT IN CUSTOM TABLE
C5CD CA D3 C5     1304        JZ     CUSE2  NOT IN TABLE--ENTER IT
C5D0 1B           1305        DCX    D      IN TABLE, REMOVE IT
C5D1 36 00        1306        MVI    M,0    CHANGE NEW NAME TO BE ZERO
C5D3 7E           1307 CUSE2  MOV    A,M    GET 1ST CHAR OF NAME
C5D4 12           1308        STAX   D      ENTER IT INTO TABLE
C5D5 13           1309        INX    D      AND THE 2ND NAME
C5D6 23           1310        INX    H
C5D7 7E           1311        MOV    A,M
C5D8 12           1312        STAX   D      NAME NOW ENTERED
C5D9 13           1313        INX    D      GET SET TO ENTER ADDRESS
C5DA E1           1314        POP    H      RESTORE RTN ADDR
C5DB EB           1315        XCHG   .
C5DC 73           1316        MOV    M,E    SET ADDR IN NOW
C5DD 23           1317        INX    H      AND HI BYTE OF ADDR
C5DE 72           1318        MOV    M,D
C5DF C9           1319        RET    .      NAME IS NOW ENTERED OR CLEARED
                  1320 *
                  1321 * -*-
```

```
                  9999       COPY   SOLOS3/1                         3 OF 3 ****
                  1322 *
                  1323 *
                  1324 *
                  1325 *
                  1326 *  THE FOLLOWING ROUTINES PROVIDE "BYTE BY BYTE" ACCESS
                  1327 * TO THE CASSETTE TAPES ON EITHER A READ OR WRITE BASIS.
                  1328 *
                  1329 * THE TAPE IS READ ONE BLOCK AT A TIME AND INDIVIDUAL
                  1330 * TRANSFERS OF DATA HANDLED BY MANAGING A BUFFER AREA.
                  1331 *
                  1332 * THE BUFFER AREA IS CONTROLLED BY A FILE CONTROL BLOCK
                  1333 * (FCB) WHOSE STRUCTURE IS:
                  1334 *
                  1335 *
                  1336 *    7 BYTES FOR EACH OF THE TWO FILES STRUCTURED AS
                  1337 *  FOLLOWS:
                  1338 *
                  1339 *         1 BYTE -  ACCESS CONTROL   00  IF CLOSED
                  1340 *                                    FF  IF READING
                  1341 *                                    FE  IF WRITING
                  1342 *         1 BYTE -  READ COUNTER
                  1343 *         1 BYTE -  BUFFER POSITION POINTER
                  1344 *         2 BYTE -  CONTROL HEADER ADDRESS
                  1345 *         2 BYTE -  BUFFER LOCATION ADDRESS
                  1346 *
                  1347 *
                  1348 *
                  1349 *       THIS ROUTINE "OPENS" THE CASSETTE UNIT FOR ACCESS
                  1350 *
                  1351 *  ON ENTRY:  A - HAS THE TAPE UNIT NUMBER (1 OR 2)
                  1352 *            HL - HAS USER SUPPLIED HEADER FOR TAPE FILE
                  1353 *
                  1354 *
                  1355 *  NORMAL RETURN:    ALL REGISTERS ARE ALTERED
                  1356 *                    BLOCK IS READY FOR ACCESS
                  1357 *
                  1358 *  ERROR RETURN:     CARRY BIT IS SET
                  1359 *
                  1360 *  ERRORS:  BLOCK ALREADY OPEN
                  1361 *
                  1362 *
C5E0 E5           1363 BOPEN  PUSH   H      SAVE HEADER ADDRESS
C5E1 CD 33 C6     1364        CALL   LFCB   GET ADDRESS OF FILE CONTROL
C5E4 C2 FA C5     1365        JNZ    TERE2  FILE WAS ALREADY OPEN
C5E7 36 01        1366        MVI    M,1    NOW IT IS
C5E9 23           1367        INX    H      POINT TO READ COUNT
C5EA 77           1368        MOV    M,A    ZERO
C5EB 23           1369        INX    H      POINT TO BUFFER CURSOR
C5EC 77           1370        MOV    M,A    PUT IN THE ZERO COUNT
                  1371 *
                  1372 *  ALLOCATE THE BUFFER
                  1373 *
```

```
C5ED 11 63 C8           LXI     D,FBUF1  POINT TO BUFFER AREA
C5F0 3A 54 C8           LDA     FNUMF  GET WHICH ONE WE ARE GOING TO USE
C5F3 82                 ADD     D
C5F4 57                 MOV     D,A     256 BIT ADD
                 *
C5F5 C1          UBUF   POP     B       HEADER ADDRESS
C5F6 B7                 ORA     A       CLEAR CARRY AND RETURN AFTER STORING PARAMS
C5F7 C3 B6 C6           JMP     PSTOR   STORE THE VALUES
                 *
                 *   GENERAL ERROR RETURN POINTS FOR STACK CONTROL
                 *
C5FA E1          TERE2  POP     H
C5FB D1          TERE1  POP     D
C5FC AF          TERE0  XRA     A       CLEAR ALL FLAGS
C5FD 37                 STC     .       SET ERROR
C5FE C9                 RET
                 *
                 *
C5FF 3D          EOFER  DCR     A       SET MINUS FLAGS
C600 37                 STC     .       AND CARRY
C601 D1                 POP     D       CLEAR THE STACK
C602 C9                 RET     .       THE FLAGS TELL ALL
                 *
                 *
                 *
                 *
                 *  THIS ROUTINE CLOSES THE FILE BUFFER TO ALLOW ACCESS
                 *  FOR A DIFFERENT CASSETTE OR PROGRAM.  IF THE FILE
                 *  OPERATIONS WERE "WRITE" THEN THE LAST BLOCK IS WRITTED
                 *  OUT AND AN "END OF FILE" WRITTEN TO THE TAPE.  IF
                 *  THE OPERATIONS WERE "READS" THEN THE FILE IS JUST
                 *  MADE READY FOR NEW USE.
                 *
                 *  ON ENTRY:  A - HAS WHICH UNIT (1 OR 2)
                 *
                 *  ERROR RETURNS:  FILE WASN'T OPEN
                 *
                 *
C603 CD 33 C6    PCLOS  CALL    LFCB    GET CONTROL BLOCK ADDRESS
C606 C8                 RZ      .       WASN'T OPEN, CARRY IS SET FROM LFCB
C607 B7                 ORA     A       CLEAR CARRY
C608 3C                 INR     A       SET CONDITION FLAGS
C609 36 00              MVI     M,0     CLOSE THE CONTROL BYTE
C60B C8                 RZ      .       WE WERE READING...NOTHING MORE TO DO
                 *
                 *   THE FILE OPERATIONS WERE "WRITES"
                 *
                 * PUT THE CURRENT BLOCK ON THE TAPE
                 * (EVEN IF ONLY ONE BYTE!!)
                 * THEN WRITE AN END OF FILE TO THE TAPE
                 *
                 *
C60C 23                 INX     H
```

```
C60D 23                 INX     H
C60E 7E                 MOV     A,M     GET CURSOR POSITION
C60F 7E
C610 CD BF C6           CALL    PLOAD   BC GET HEADER ADDRESS, DE BUFFER ADDRESS
C613 C5                 PUSH    B       HEADER TO STACK
C614 21 07 00           LXI     H,BLKOF  OFFSET TO BLOCK SIZE
C617 09                 DAD     B
C618 B7                 ORA     A       TEST COUNT
C619 CA 2B C6           JZ      EOFW    NO BYTES...JUST WRITE EOF
                 *
                 *   WRITE LAST BLOCK
                 *
C61C E5                 PUSH    H       SAVE BLOCK SIZE POINTER FOR EOF
C61D 77                 MOV     M,A     PUT IN COUNT
C61E 23                 INX     H
C61F 36 00              MVI     M,0     ZERO THE HIGHER BYTE
C621 23                 INX     H
C622 73                 MOV     M,E     BUFFER ADDRESS
C623 23                 INX     H
C624 72                 MOV     M,D
C625 60                 MOV     H,B
C626 69                 MOV     L,C     PUT HEADER ADDRESS IN HL
C627 CD 7C C7           CALL    WFBLK   GO WRITE IT OUT
C62A E1                 POP     H       BLOCK SIZE POINTER
                 *
                 *  NOW WRITE END OF FILE TO CASSETTE
                 *
C62B AF          EOFW   XRA     A       PUT IN ZEROS FOR SIZE:  EOF MARK IS ZERO BYTE
C62C 77                 MOV     M,A
C62D 23                 INX     H
C62E 77                 MOV     M,A
C62F E1                 POP     H       HEADER ADDRESS
C630 C3 7C C7           JMP     WFBLK   WRITE IT OUT AND RETURN
                 *
                 *
                 *
                 *
                 *  THIS ROUTINE LOCATES THE FILE CONTROL BLOCK POINTED TO
                 *  BY REGISTER "A".  ON RETURN HL POINT TO THE CONTROL BYT
                 *  AND REGISTER "A" HAS THE CONTROL WORD WITH THE FLAGS
                 *  SET FOR IMMEDIATE CONDITION DECISIONS.
                 *
                 *
C633 21 55 C8    LFCB   LXI     H,FCBAS  POINT TO THE BASE OF IT
C636 1F                 RAR     .       MOVE THE 1 & 2 TO 0 & 1 LIKE COMPUTERS LIKE
C637 E6 01              ANI     1       SMALL NUMBERS ARE THE RULE
C639 32 54 C8           STA     FNUMF   CURRENT ACCESS FILE NUMBER
C63C CA 42 C6           JZ      LFCB1   UNIT ONE (VALUE OF ZERO)
C63F 21 5C C8           LXI     H,FCBA2  UNIT TWO--PT TO ITS FCB
     C642        LFCB1  EQU     $       HL PT TO PROPER FCB
C642 7E                 MOV     A,M     PICK UP FLAGS FM FCB
C643 B7                 ORA     A       SET FLAGS BASED ON CONTROL WORD
C644 37                 STC     .       SET CARRY IN CASE OF IMMEDIATE ERROR RETURN
```

```
C645 C9                     RET
                 *
                 *
                 *
                 *
                 *    READ TAPE BYTE ROUTINE
                 *
                 *    ENTRY:        -  A -  HAS FILE NUMBER
                 *    EXIT: NORMAL -  A -  HAS BYTE
                 *            ERROR
                 *              CARRY SET     - IF FILE NOT OPEN OR
                 *                              PREVIOUS OPERATIONS WERE WRITE
                 *              CARRY & MINUS - END OF FILE ENCOUNTERED
                 *
                 *
                 *
                 *
C646 CD 33 C6    RTBYT  CALL   LFCB   LOCATE THE FILE CONTROL BLOCK
C649 C8                 RZ     .      FILE NOT OPEN
C64A 3C                 INR    A      TEST IF FF
C64B FA FC C5           JM     TEREO  ERROR WAS WRITING
C64E 36 FF              MVI    M,-1   SET IT AS READ  (IN CASE IT WAS JUST OPENED)
C650 23                 INX    H
C651 7E                 MOV    A,M    GET READ COUNT
C652 E5                 PUSH   H      SAVE COUNT ADDRESS
C653 23                 INX    H
C654 CD BF C6           CALL   PLOAD  GET THE OTHER PARAMETERS
C657 E1                 POP    H
C658 B7                 ORA    A
C659 C2 75 C6           JNZ    GTBYT  IF NOT EMPTY GO GET BYTE
                 *
                 *  CURSOR POSITION WAS ZERO...READ A NEW BLOCK INTO
                 *  THE BUFFER.
                 *
C65C D5          RDNBLK PUSH   D      BUFFER POINTER
C65D E5                 PUSH   H      TABLE ADDRESS
C65E 23                 INX    H
C65F CD A6 C6           CALL   PHEAD  PREPARE THE HEADER FOR READ
C662 CD C8 C6           CALL   RFBLK  READ IN THE BLOCK
C665 DA FA C5           JC     TERE2  ERROR POP OFF STACK BEFORE RETURN
C668 E1                 POP    H
C669 7B                 MOV    A,E    LOW BYTE OF COUNT (WILL BE ZERO IF 256)
C66A B2                 ORA    D      SEE IF BOTH ARE ZERO
C66B CA FF C5           JZ     EOFER  BYTE COUNT WAS ZERO....END OF FILE
C66E 73                 MOV    M,E    NEW COUNT ( ZERO IS 256 AT THIS POINT)
C66F 23                 INX    H      BUFFER LOCATION POINTER
C670 36 00              MVI    M,0
C672 2B                 DCX    H
C673 7B                 MOV    A,E    COUNT TO A
C674 D1                 POP    D      GET BACK BUFFER ADDRESS
                 *
                 *
                 *
```

```
                 *   THIS ROUTINE GETS ONE BYTE FROM THE BUFFER
                 *  AND RETURNS IT IN REGISTER "A".  IF THE END
                 *  OF THE BUFFER IS REACHED IT MOVES THE POINTER
                 *  TO THE BEGINNING OF THE BUFFER FOR THE NEXT
                 *  LOAD.
                 *
C675 3D          GTBYT  DCR    A      BUMP THE COUNT
C676 77                 MOV    M,A    RESTORE IT
C677 23                 INX    H
C678 7E                 MOV    A,M    GET BUFFER POSITION
C679 34                 INR    M      BUMP IT
                 *
C67A 83                 ADD    E
C67B 5F                 MOV    E,A    DE NOW POINT TO CORRECT BUFFER POSITION
C67C D2 80 C6           JNC    RT1
C67F 14                 INR    D
C680 1A          RT1    LDAX   D      GET CHARACTER FROM BUFFER
C681 B7                 ORA    A      CLEAR CARRY
C682 C9                 RET    .      ALL DONE
                 *
                 *
                 *
                 *
                 *      THIS ROUTINE IS USED TO WRITE A BYTE TO THE FILE
                 *
                 *      ON ENTRY:   A -  HAS FILE NUMBER
                 *                  B -  HAS DATA BYTE
                 *
                 *
C683 CD 33 C6    WTBYT  CALL   LFCB   GET CONTROL BLOCK
C686 C8                 RZ     .      FILE WASN'T OPEN
C687 3C                 INR    A
C688 C8                 RZ     .      FILE WAS READ
C689 36 FE              MVI    M,OFEH  SET IT TO WRITE
C68B 23                 INX    H
C68C 23                 INX    H
C68D 78                 MOV    A,B    GET CHARACTER
C68E F5                 PUSH   PSW
C68F E5                 PUSH   H      SAVE CONTROL ADDRESS+2
                 *
                 *   NOW DO THE WRITE
                 *
C690 CD BF C6           CALL   PLOAD  BC GETS HEADER ADDR, DE BUFFER ADDRESS
C693 E1                 POP    H
C694 7E                 MOV    A,M    COUNT BYTE
C695 83                 ADD    E
C696 5F                 MOV    E,A
C697 D2 9B C6           JNC    WT1
C69A 14                 INR    D
C69B F1          WT1    POP    PSW    CHARACTER
C69C 12                 STAX   D      PUT CHR IN BUFFER
C69D B7                 ORA    A      CLEAR FLAGS
C69E 34                 INR    M      INCREMENT THE COUNT
```

```
C69F C0                     RNZ    .      RETURN IF COUNT DIDN'T ROLL OVER
                      *
                      *  THE BUFFER IS FULL.  WRITE IT TO TAPE AND RESET
                      * CONTROL BLOCK.
                      *
C6A0 CD A6 C6               CALL   PHEAD  PREPARE THE HEADER
C6A3 C3 7C C7               JMP    WFBLK  WRITE IT OUT AND RETURN
                      *
                      *
                      *
                      *
                      * THIS ROUTINE PUTS THE BLOCK SIZE (256) AND BUFFER
                      * ADDRESS IN THE FILE HEADER.
                      *
C6A6 CD BF C6         PHEAD CALL   PLOAD  GET HEADER AND BUFFER ADDRESSES
C6A9 C5                     PUSH   B      HEADER ADDRESS
C6AA 21 06 00               LXI    H,BLKOF-1  PSTOR DOES AN INCREMENT
C6AD 09                     DAD    B      HL POINT TO BLOCKSIZE ENTRY
C6AE 01 00 01               LXI    B,256
C6B1 CD B6 C6               CALL   PSTOR
C6B4 E1                     POP    H      HL RETURN WITH HEADER ADDRESS
C6B5 C9                     RET
                      *
                      *
C6B6 23               PSTOR INX    H
C6B7 71                     MOV    M,C
C6B8 23                     INX    H
C6B9 70                     MOV    M,B
C6BA 23                     INX    H
C6BB 73                     MOV    M,E
C6BC 23                     INX    H
C6BD 72                     MOV    M,D
C6BE C9                     RET
                      *
                      *
C6BF 23               PLOAD INX    H
C6C0 4E                     MOV    C,M
C6C1 23                     INX    H
C6C2 46                     MOV    B,M
C6C3 23                     INX    H
C6C4 5E                     MOV    E,M
C6C5 23                     INX    H
C6C6 56                     MOV    D,M
C6C7 C9                     RET
                      *
                      *
                      *
                      *
                      *
                      *   THIS ROUTINE SETS THE CORRECT UNIT FOR SYSTEM READS
C6C8 CD DE C7         RFBLK CALL   GTUNT  SET UP A=UNIT WITH SPEED
                      *
                      *
```

```
                      *
                      *
                      *                TAPE READ ROUTINES
                      *
                      *     ON ENTRY:      A   HAS UNIT AND SPEED
                      *                    HL  POINT TO HEADER BLOCK
                      *                    DE  HAVE OPTIONAL PUT ADDRESS
                      *
                      *     ON EXIT:       CARRY IS SET IF ERROR OCCURED
                      *                    TAPE UNITS ARE OFF
                      *
                      *
C6CB D5               RTAPE PUSH   D      SAVE OPTIONAL ADDRESS
C6CC 06 03                  MVI    B,3    SHORT DELAY
C6CE CD EF C7               CALL   TON
C6D1 DB FB                  IN     TDATA  CLEAR THE UART FLAGS
                      *
C6D3 E5               PTAP1 PUSH   H      HEADER ADDRESS
C6D4 CD 23 C7               CALL   RHEAD  GO READ HEADER
C6D7 E1                     POP    H
C6D8 DA 06 C7               JC     TERR   IF AN ERROR OR ESC WAS RECEIVED
C6DB C2 D3 C6               JNZ    PTAP1  IF VALID HEADER NOT FOUND
                      *
                      * FOUND A VALID HEADER NOW DO COMPARE
                      *
C6DE E5                     PUSH   H      GET BACK AND RESAVE ADDRESS
C6DF 11 1C C8               LXI    D,THEAD
C6E2 CD D2 C7               CALL   DHCMP  COMPARE DE-HL HEADERS
C6E5 E1                     POP    H
C6E6 C2 D3 C6               JNZ    PTAP1
                      *
                      *
C6E9 D1                     POP    D      OPTIONAL "PUT" ADDRESS
C6EA 7A                     MOV    A,D
C6EB B3                     ORA    E      SEE IF DE IS ZERO
C6EC 2A 23 C8               LHLD   BLOCK  GET BLOCK SIZE
C6EF EB                     XCHG   .      ...TO DE
                      * DE HAS HBLOCK....HL HAS USER OPTION
C6F0 C2 F6 C6               JNZ    RTAP   IF DE WAS ZERO GET TAPE LOAD ADDRESS
C6F3 2A 25 C8               LHLD   LOADR  GET TAPE LOAD ADDRESS
                      *
                      *
                      *    THIS ROUTINE READS "DE" BYTES FROM THE TAPE
                      *    TO ADDRESS HL.  THE BYTES MUST BE FROM ONE
                      *    CONTIGUOUS PHYSICAL BLOCK ON THE TAPE.
                      *
                      *        HL HAS "PUT" ADDRESS
                      *        DE HAS SIZE OF TAPE BLOCK
                      *
C6F6 D5               RTAP  PUSH   D      SAVE SIZE FOR RETURN TO CALLING PROGRAM
                      *
    C6F7              RTAP2 EQU    $      HERE TO LOOP RDING BLKS
C6F7 CD 15 C7               CALL   DCRCT  DROP COUNT, B=LEN THIS BLK
```

```
C6FA CA 10 C7              JZ     RTOFF  ZERO=ALL DONE
                    *
C6FD CD 44 C7              CALL   RHED1  READ THAT MANY BYTES
C700 DA 06 C7              JC     TERR   IF ERROR OR ESC
C703 CA F7 C6              JZ     RTAP2  RD OK--READ SOME MORE
                    *
                    *  ERROR RETURN
                    *
C706 AF             TERR   XRA    A
C707 37                    STC    .      SET ERROR FLAGS
C708 C3 11 C7              JMP    RTOF1
                    *
                    *
C70B 06 01          TOFF   MVI    B,1
C70D CD F1 C7              CALL   DELAY
C710 AF             RTOFF  XRA    A
C711 D3 FA          RTOF1  OUT    TAPPT
C713 D1                    POP    D      RETURN BYTE COUNT
C714 C9                    RET
                    *
                    *
    C715            DCRCT  EQU    $      COMMON RTN TO COUNT DOWN BLK LENGTHS
C715 AF                    XRA    A      CLR FOR LATER TESTS
C716 47                    MOV    B,A    SET THIS BLK LEN=256
C717 B2                    ORA    D      IS AMNT LEFT < 256
C718 C2 20 C7              JNZ    DCRC2  NO--REDUCE AMNT BY 256
C71B B3                    ORA    E      IS ENTIRE COUNT ZERO
C71C C8                    RZ     ALL DONE--ZERO=THIS CONDITION
C71D 43                    MOV    B,E    SET THIS BLK LEN TO AMNT REMAINING
C71E 5A                    MOV    E,D    MAKE ENTIRE COUNT ZERO NOW
C71F C9                    RET    .      ALL DONE (NON-ZERO FLAG)
    C720            DCRC2  EQU    $      REDUCE COUNT BY 256
C720 15                    DCR    D      DROP BY 256
C721 B7                    ORA    A      FORCE NON-ZERO FLAG
C722 C9                    RET    .      NON-ZERO=NOT DONE YET (BLK LEN=256)
                    *
                    *
                    *   READ THE HEADER
                    *
C723 06 0A          RHEAD  MVI    B,10   FIND 10 NULLS
C725 CD 5D C7       RHEA1  CALL   STAT
C728 D8                    RC     .      IF ESCAPE
C729 DB FB                 IN     TDATA  IGNORE ERROR CONDITIONS
C72B B7                    ORA    A      ZERO?
C72C C2 23 C7              JNZ    RHEAD
C72F 05                    DCR    B
C730 C2 25 C7              JNZ    RHEA1  LOOP UNTIL 10 IN A ROW
                    *
                    *   WAIT FOR THE START CHARACTER
                    *
C733 CD 6F C7       SOHL   CALL   TAPIN
C736 D8                    RC     .      ERROR OR ESCAPE
C737 FE 01                 CPI    1      AT LEAST 10 NULLS IMMEDIATELY FOLLOWED BY AN 01
```

```
C739 DA 33 C7              JC     SOHL   STILL A NULL, KEEP WAITING
C73C C2 23 C7              JNZ    RHEAD  NON-ZERO, START SEQUENCE OVER AGAIN
                    *
                    *   NOW GET THE HEADER
                    *
C73F 21 1C C8              LXI    H,THEAD  POINT TO BUFFER
C742 06 10                 MVI    B,HLEN  LENGTH TO READ
                    *
    C744            RHED1  EQU    $      RD A BLOCK INTO HL FOR B BYTES
C744 0E 00                 MVI    C,0    INIT THE CRC
    C746            RHED2  EQU    $      LOOP HERE
C746 CD 6F C7              CALL   TAPIN  GET A BYTE
C749 D8                    RC
C74A 77                    MOV    M,A    STORE IT
C74B 23                    INX    H      INCREMENT ADDRESS
C74C CD A8 C7              CALL   DOCRC  GO COMPUTE THE CRC
C74F 05                    DCR    B      WHOLE HEADER YET?
C750 C2 46 C7              JNZ    RHED2  DO ALL THE BYTES
                    *
                    *  THIS ROUTINE GETS THE NEXT BYTE AND COMPARES IT
                    * TO THE VALUE IN REGISTER C.  THE FLAGS ARE SET ON
                    * RETURN.
                    *
C753 CD 6F C7              CALL   TAPIN  GET CRC BYTE
C756 A9                    XRA    C      CLR CARRY AND SET ZERO IF MATCH, ELSE NON-ZER
C757 C8                    RZ     .      CRC WAS FINE
C758 3A 11 C8              LDA    IGNCR  GET POSSIBLE OVERRIDE CRC ERROR FLAG
C75B 3C                    INR    A      FF=IGNORE CRC ERRORS, ELSE PROCESS CRC ERROR
C75C C9                    RET
                    *
                    *   THIS ROUTINE GETS THE NEXT AVAILABLE BYTE FROM THE
                    *  TAPE.  WHILE WAITING FOR THE BYTE THE KEYBOARD IS TESTED
                    *  FOR AN ESC COMMAND.  IF RECEIVED THE TAPE LOAD IS
                    *  TERMINATED AND A RETURN TO THE COMMAND MODE IS MADE.
                    *
C75D DB FA          STAT   IN     TAPPT  TAPE STATUS PORT
C75F E6 40                 ANI    TDR
C761 C0                    RNZ
C762 CD 1F C0              CALL   SINP   CHECK INPUT
C765 CA 5D C7              JZ     STAT   NOTHING THERE YET
C768 E6 7F                 ANI    7FH    CLR PARITY 1ST
C76A C2 5D C7              JNZ    STAT   NOT A MODE (OR EVEN CTL-@)
C76D 37                    STC    .      SET ERROR FLAG
C76E C9                    RET    .      AND RETURN
                    *
                    *
                    *
C76F CD 5D C7       TAPIN  CALL   STAT   WAIT UNTIL A CHARACTER IS AVAILABLE
C772 D8                    RC
                    *
C773 DB FA          TREDY  IN     TAPPT  TAPE STATUS
C775 E6 18                 ANI    TFE+TOE  DATA ERROR?
C777 DB FB                 IN     TDATA  GET THE DATA
```

```
C779 C8                  RZ     .      IF NO ERRORS
C77A 37                  STC    .      SET ERROR FLAG
C77B C9                  RET
                  *
                  *
                  *  THIS ROUTINE GETS THE CORRECT UNIT FOR SYSTEM WRITES
C77C CD DE C7     WFBLK  CALL   GTUNT  SET UP A WITH UNIT AND SPEED
                  *
                  *
                  *
                  *      WRITE TAPE BLOCK ROUTINE
                  *
                  *   ON ENTRY:   A   HAS UNIT AND SPEED
                  *              HL   HAS POINTER TO HEADER
                  *
                  *
    C77F          WTAPE  EQU    $      HERE TO WRITE TAPE
C77F E5                  PUSH   H      SAVE HEADER ADDRESS
C780 CD AF C7            CALL   WHEAD  TURN ON, THEN WRITE HDR
C783 E1                  POP    H
C784 11 07 00            LXI    D,BLKOF  OFFSET TO BLOCK SIZE IN HEADER
C787 19                  DAD    D      HL POINT TO BLOCK SIZE
C788 5E                  MOV    E,M
C789 23                  INX    H
C78A 56                  MOV    D,M    DE HAVE SIZE
C78B 23                  INX    H
C78C 7E                  MOV    A,M
C78D 23                  INX    H
C78E 66                  MOV    H,M
C78F 6F                  MOV    L,A    HL HAVE STARTING ADDRESS
                  *
                  *    THIS ROUTINE WRITES ONE PHYSICAL BLOCK ON THE
                  *  TAPE "DE" BYTES LONG FROM ADDRESS "HL".
                  *
                  *
    C790          WRLO1  EQU    $      HERE FOR THE EXTRA PUSH
C790 E5                  PUSH   H      A DUMMY PUSH FOR LATER EXIT
    C791          WTAP2  EQU    $      LOOP HERE UNTIL ENTIRE AMOUNT READ
C791 CD 15 C7            CALL   DCRCT  DROP COUNT IN DE AND SET UP B W/LEN THIS BLK
C794 CA 0B C7            JZ     TOFF   RETURNS ZERO IF ALL DONE
C797 CD C3 C7            CALL   WTBL   WRITE BLOCK FOR BYTES IN B (256)
C79A C3 91 C7            JMP    WTAP2  LOOP UNTIL ALL DONE
                  *
                  *
C79D F5           WRTAP  PUSH   PSW
C79E DB FA        WRWAT  IN     TAPPT  TAPE STATUS
C7A0 E6 80               ANI    TTBE   IS TAPE READY FOR A CHAR YET
C7A2 CA 9E C7            JZ     WRWAT  NO--WAIT
C7A5 F1                  POP    PSW    YES--RESTORE CHAR TO OUTPUT
C7A6 D3 FB               OUT    TDATA  SEND CHAR TO TAPE
                  *
    C7A8          DOCRC  EQU    $      A COMMON CRC COMPUTATION ROUTINE
C7A8 91                  SUB    C
```

```
C7A9 4F                  MOV    C,A
C7AA A9                  XRA    C
C7AB 2F                  CMA
C7AC 91                  SUB    C
C7AD 4F                  MOV    C,A
C7AE C9                  RET    .      ONE  BYTE NOW WRITTEN
                  *
                  *
                  *  THIS ROUTINE WRITES THE HEADER POINTED TO BY
                  *  HL TO THE TAPE.
                  *
    C7AF          WHEAD  EQU    $      HERE TO 1ST TURN ON THE TAPE
C7AF CD ED C7            CALL   WTON   TURN IT ON, THEN WRITE HEADER
C7B2 16 32               MVI    D,50   WRITE 50 ZEROS
C7B4 AF           NULOP  XRA    A
C7B5 CD 9D C7            CALL   WRTAP
C7B8 15                  DCR    D
C7B9 C2 B4 C7            JNZ    NULOP
                  *
C7BC 3E 01               MVI    A,1
C7BE CD 9D C7            CALL   WRTAP
C7C1 06 10               MVI    B,HLEN  LENGTH TO WRITE OUT
                  *
C7C3 0E 00        WTBL   MVI    C,0    RESET CRC BYTE
C7C5 7E           WLOOP  MOV    A,M    GET CHARACTER
C7C6 CD 9D C7            CALL   WRTAP  WRITE IT TO THE TAPE
C7C9 05                  DCR    B
C7CA 23                  INX    H
C7CB C2 C5 C7            JNZ    WLOOP
C7CE 79                  MOV    A,C    GET CRC
C7CF C3 9D C7            JMP    WRTAP  PUT IT ON THE TAPE AND RETURN
                  *
                  *
                  *  THIS ROUTINE COMPARES THE HEADER IN THEAD TO
                  *  THE USER SUPPLIED HEADER IN ADDRESS HL.
                  *  ON RETURN IF ZERO IS SET THE TWO NAMES COMPARED
                  *
C7D2 06 05        DHCMP  MVI    B,5
C7D4 1A           DHLOP  LDAX   D
C7D5 BE                  CMP    M
C7D6 C0                  RNZ
C7D7 05                  DCR    B
C7D8 C8                  RZ     .      IF ALL FIVE COMPARED
C7D9 23                  INX    H
C7DA 13                  INX    D
C7DB C3 D4 C7            JMP    DHLOP
                  *
    C7DE          GTUNT  EQU    $      SET A=SPEED + UNIT
C7DE 3A 54 C8            LDA    FNUMF  GET UNIT
C7E1 B7                  ORA    A      SEE WHICH UNIT
C7E2 3A 0D C8            LDA    TSPD   BUT 1ST GET SPEED
C7E5 C2 EA C7            JNZ    GTUN2  MAKE IT UNIT TWO
C7E8 C6 40               ADI    TAPE2  THIS ONCE=UNIT 2, TWICE=UNIT 1
```

```
C7EA C6 40        GTUN2  ADI    TAPE2  UNIT AND SPEED NOW SET IN A
C7EC C9                  RET    .      ALL DONE
                  *
C7ED 06 04        WTON   MVI    B,4    SET LOOP DELAY  (BIT LONGER ON A WRITE)
     C7EF         TON    EQU    $      HERE TO TURN A TAPE ON THEN DELAY
C7EF D3 FA               OUT    TAPPT  GET TAPE MOVING, THEN DELAY
                  *
C7F1 11 00 00     DELAY  LXI    D,0
C7F4 1B           DLOP1  DCX    D
C7F5 7A                  MOV    A,D
C7F6 B3                  ORA    E
C7F7 C2 F4 C7            JNZ    DLOP1
C7FA 05                  DCR    B
C7FB C2 F1 C7            JNZ    DELAY
C7FE C9                  RET
                  *
                  *
                  ***** -- END OF PROGRAM--
                  *
                  *
                  *
                  *
                  *    S Y S T E M    E Q U A T E S
                  *
                  *
                  *           VDM PARAMETERS
                  *
     CC00         VDMEM  EQU    0CC00H  VDM SCREEN MEMORY
                  *
                  *
                  *           KEYBOARD SPECIAL KEY ASSIGNMENTS
                  *
                  *  THESE DEFINITIONS ARE DESIGNED TO ALLOW
                  *  COMPATABILITY WITH CUTER(TM).  THESE ARE THE
                  *  SAME KEYS WITH BIT 7 (X'80') STRIPPED OFF.
                  *
     009A         DOWN   EQU    9AH    CTL-Z
     0097         UP     EQU    97H    CTL-W
     0081         LEFT   EQU    81H    CTL-A
     0093         RIGHT  EQU    93H    CTL-S
     008B         CLEAR  EQU    8BH    CTL-K
     008E         HOME   EQU    8EH    CTL-N
     0080         MODE   EQU    80H    CTL-@
     005F         BACKS  EQU    5FH    BACKSPACE
     000A         LF     EQU    10
     000D         CR     EQU    13
     0020         BLANK  EQU    ' '
     0020         SPACE  EQU    BLANK
     0018         CX     EQU    'X'-40H
     001B         ESC    EQU    1BH
                  *
                  *           PORT ASSIGNMENTS
                  *
```

```
     00FA         STAPT  EQU    0FAH   STATUS PORT GENERAL
     00F8         SERST  EQU    0F8H   SERIAL STATUS PORT
     00F9         SDATA  EQU    0F9H   SERIAL DATA
     00FD         PDATA  EQU    0FDH   PARALLEL DATA
     00FC         KDATA  EQU    0FCH   KEYBOARD DATA
     00FE         DSTAT  EQU    0FEH   VDM CONTROL PORT
     00FA         TAPPT  EQU    0FAH   TAPE STATUS PORT
     00FB         TDATA  EQU    0FBH   TAPE DATA PORT
     00FF         SENSE  EQU    0FFH   SENSE SWITCHES
                  *
                  *
                  *
                  *        BIT ASSIGNMENT MASKS
                  *
     0001         SCD    EQU    1      SERIAL CARRIER DETECT
     0002         SDSR   EQU    2      SERIAL DATA SET READY
     0004         SPE    EQU    4      SERIAL PARITY ERROR
     0008         SFE    EQU    8      SERIAL FRAMING ERROR
     0010         SOE    EQU    16     SERIAL OVERRUN ERROR
     0020         SCTS   EQU    32     SERIAL CLEAR TO SEND
     0040         SDR    EQU    64     SERIAL DATA READY
     0080         STBE   EQU    128    SERIAL TRANSMITTER BUFFER EMPTY
                  *
     0001         KDR    EQU    1      KEYBOARD DATA READY
     0002         PDR    EQU    2      PARALLEL DATA READY
     0004         PXDR   EQU    4      PARALLEL DEVICE READY
     0008         TFE    EQU    8      TAPE FRAMING ERROR
     0010         TOE    EQU    16     TAPE OVERFLOW ERROR
     0040         TDR    EQU    64     TAPE DATA READY
     0080         TTBE   EQU    128    TAPE TRANSMITTER BUFFER EMPTY
                  *
     0001         SOK    EQU    1      SCROLL OK FLAG
                  *
     0080         TAPE1  EQU    80H    1=TURN TAPE ONE ON
     0040         TAPE2  EQU    40H    1=TURN TAPE TWO ON
                  *
                  *
                  *
                  *
                  *     S Y S T E M   G L O B A L    A R E A
                  *
C800                     ORG    START+0800H   RAM STARTS JUST AFTER ROM
                  *
     C800         SYSRAM EQU    $      START OF SYSTEM RAM
     CBFF         SYSTP  EQU    SYSRAM+3FFH  STACK WORKS FM TOP DOWN
                  *
                  *
                  *   PARAMETERS STORED IN RAM
                  *
C800              UIPRT  DS     2      USER DEFINED INPUT RTN IF NON ZERO
C802              UOPRT  DS     2      USER DEFINED OUTPUT RTN IF NON ZERO
C804              DFLTS  DS     2      DEFAULT PSUEDO I/O PORTS (ALWAYS ZERO IN SOLOS
C806              IPORT  DS     1      CRNT INPUT PSUEDO PORT
```

```
C807                      2010 OPORT  DS     1      CRNT OUTPUT PSUEDO PORT
C808                      2011 NCHAR  DS     1      CURRENT CHARACTER POSITION
C809                      2012 LINE   DS     1      CURRENT LINE POSITION
C80A                      2013 BOT    DS     1      BEGINNING OF TEXT DISPLACEMENT
C80B                      2014 SPEED  DS     1      SPEED CONTROL BYTE
C80C                      2015 ESCFL  DS     1      ESCAPE FLAG CONTROL BYTE
C80D                      2016 TSPD   DS     1      CURRENT TAPE SPEED
C80E                      2017 INPTR  DS     2      FOR COMPATABILITY W/ CUTER
C810                      2018 NUCNT  DS     1      NUMBER OF NULLS AFTER CRLF
C811                      2019 IGNCR  DS     1      FF=IGNORE CRC ERRORS, ELSE NORMAL
                          2020 *
C812                      2021        DS     10     ROOM FOR FUTURE EXPANSION
                          2022 *
                          2023 * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                          2024 *    T H I S   I S   T H E   H E A D E R   L A Y O U T   *
                          2025 * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                          2026 *
C81C                      2027 THEAD  DS     5      NAME
C821                      2028        DS     1      THIS BYTE MUST BE ZERO
C822                      2029 HTYPE  DS     1      TYPE
C823                      2030 BLOCK  DS     2      BLOCK SIZE
C825                      2031 LOADR  DS     2      LOAD ADDRESS
C827                      2032 XEQAD  DS     2      AUTO EXECUTE ADDRESS
C829                      2033 HSPR   DS     3      SPARES
                          2034 *
    0010                  2035 HLEN   EQU    $-THEAD  LENGTH OF HEADER
    0007                  2036 BLKOF  EQU    BLOCK-THEAD  OFFSET TO BLOCK SIZE
C82C                      2037 DHEAD  DS     HLEN   A DUMMY HDR FOR COMPARES WHILE RD'ING
                          2038 *
                          2039 *
C83C                      2040 CUTAB  DS     6*4    ROOM FOR UP TO 6 CUSTOM USER COMMANDS
                          2041 *
                          2042 *
C854                      2043 FNUMF  DS     1      FOR CURRENT FILE OPERATIONS
C855                      2044 FCBAS  DS     7      1ST FILE CONTROL BLOCK
C85C                      2045 FCBA2  DS     7      2ND FILE CONTROL BLOCK
C863                      2046 FBUF1  DS     2*256  SYSTEM FILE BUFFER BASE
CA63                      2047        DS     81     THIS IS AN AREA USED BY CUTER
    CAB4                  2048 USARE  EQU    $      START OF USER AREA
                          2049 *   REMEMBER THAT THE STACK WORKS ITS WAY DOWN FROM
                          2050 *   THE END OF THIS 1K RAM AREA.
                          2051 *
                          2052 * -*-

ADOUT   C3E8     AINP    C022     ALOAD   C548     AOUT    C01C
ARET    C19B     ARET1   C19D     ARET2   C1A2     BACKS   005F
BLANK   0020     BLKOF   0007     BLOCK   C823     BOPEN   C5E0
BOT     C80A     BOUT    C406     CHAR    C094     CHPCK   C05E
CHRLI   C571     CLEAR   008B     CLERA   C1B4     CLIN1   C0FA
CLINE   C0F4     COMN1   C1C0     COMND   C1C9     COMTA   C24A
CONT    C0FF     COPRC   C205     CR      000D     CREM    C136
CRLF    C0F9     CUR     C0D1     CURET   C1A6     CURSC   C0CF
CUSE2   C5D3     CUSET   C5BD     CUTAB   C83C     CX      0018
```

```
DCRC2   C720     DCRCT   C715     DEFLT   C49A     DELAY   C7F1
DFLTS   C804     DHCMP   C7D2     DHEAD   C82C     DHLOP   C7D4
DISP0   C222     DISP1   C22B     DISPD   C599     DISPT   C227
DLOOP   C3C8     DLOP1   C7F4     DLP1    C3D3     DOCRC   C7A8
DOWN    009A     DSTAT   00FE     DUMP    C3BF     ENLO1   C43C
ENLO3   C459     ENLOP   C42B     ENTER   C423     EOFER   C5FF
EOFW    C62B     ERAS1   C0DB     ERAS3   C0EE     ERR1    C480
ERR2    C481     ERRIT   C2CB     ERRM    C525     ERRO1   C2D6
ERROT   C2D2     ESC     001B     ESCFL   C80C     ESCS    C15F
ESCSP   C168     EXEC    C45E     EXEC1   C461     FBUF1   C863
FCBA2   C85C     FCBAS   C855     FCLOS   C00A     FDCOM   C231
FDCOU   C22E     FNUMF   C854     FOPEN   C007     GCLIN   C1E4
GOBAC   C06B     GOBK    C07C     GTBYT   C675     GTUN2   C7EA
GTUNT   C7DE     HBOUT   C3ED     HCONV   C34D     HCOV1   C35D
HEOU1   C414     HEOUT   C40B     HLEN    0010     HOME    008E
HSPR    C829     HTYPE   C822     IGNCR   C811     INIT    C001
INPTR   C80E     IOPRC   C026     IPORT   C806     ITAB    C29A
KDATA   00FC     KDR     0001     KSTAT   C02E     LEFT    0081
LF      000A     LFCB    C633     LFCB1   C642     LINE    C809
LIST1   C539     LLIST   C531     LOADR   C825     MODE    0080
NAME    C469     NAME1   C46E     NAMES   C466     NAOUT   C550
NCHAR   C808     NCOM    C243     NEXT    C080     NFIL    C486
NLOOP   C56A     NUCNT   C810     NULOP   C7B4     NULOT   C307
OCHAR   C098     OK      C0C1     OPORT   C807     OTAB    C292
OUTH    C41F     OUTPR   C03B     PASTA   C2DD     PBACK   C13E
PCLOS   C603     PCR     C147     PCUR    C10F     PDATA   00FD
PDOWN   C0CB     PDR     0002     PERSE   C0D5     PESC    C159
PHEAD   C6A6     PHOME   C0E5     PLEFT   C10B     PLF     C14D
PLOAD   C6BF     PRIT    C115     PROMP   C2F1     PROUT   C2E6
PSCAN   C310     PSTOR   C6B6     PTAP1   C6D3     PUP     C104
PXDR    0004     RDBLK   C013     RDBYT   C00D     RDNBL   C65C
RETRN   C004     RFBLK   C6C8     RHEA1   C725     RHEAD   C723
RHED1   C744     RHED2   C746     RIGHT   0093     RT1     C680
RTAP    C6F6     RTAP2   C6F7     RTAPE   C6CB     RTBYT   C646
RTOF1   C711     RTOFF   C710     SBLK    C31B     SBLK1   C31D
SCD     0001     SCHR    C32E     SCHR1   C330     SCONV   C33A
SCROL   C0AC     SCTS    0020     SDATA   00F9     SDR     0040
SDROT   C04A     SDSR    0002     SECON   C190     SENSE   00FF
SERST   00F8     SET     C57A     SETAB   C2A2     SETCI   C5A5
SETCO   C5A9     SETCR   C5B9     SETIN   C59D     SETNU   C5B5
SETOT   C5A1     SETSP   C594     SETTY   C5AD     SETX    C188
SETXQ   C5B1     SETY    C18C     SFE     0008     SHE1    C343
SHEX    C340     SINP    C01F     SOE     0010     SOHL    C733
SOK     0001     SOUT    C019     SPACE   0020     SPE     0004
SPEED   C80B     SROL    C0B0     SSTAT   C042     STAPT   00FA
START   C000     STAT    C75D     STBE    0080     STRTA   C1AF
STSPD   C598     STUNT   C4A2     SYSRA   C800     SYSTP   CBFF
TAERR   C514     TAPE1   0080     TAPE2   0040     TAPIN   C76F
TAPPT   00FA     TASPD   C58E     TBL     C273     TDATA   00FB
TDR     0040     TEREO   C5FC     TERE1   C5FB     TERE2   C5FA
TERM    C367     TERM1   C373     TERM2   C3B9     TERR    C706
TFE     0008     THEAD   C81C     TIMER   C077     TIN     C38B
TLIST   C52B     TLOA2   C4B5     TLOA3   C4C1     TLOAD   C4A7
```

```
TOE     0010     TOFF    C70B     TON     C7EF     TOUT    C388
TREDY   C773     TSAVE   C4E6     TSPD    C80D     TSRCH   C082
TTBE    0080     TXEQ    C4A6     UBUF    C5F5     UIPRT   C800
UOPRT   C802     UP      0097     USARE   CAB4     VDAD    C123
VDAD2   C120     VDADD   C11C     VDMEM   CC00     VDMOT   C054
WFBLK   C77C     WHEAD   C7AF     WLOOP   C7C5     WRBLK   C016
WRBYT   C010     WRLO1   C790     WRTAP   C79D     WRWAT   C79E
WT1     C69B     WTAP2   C791     WTAPE   C77F     WTBL    C7C3
WTBYT   C683     WTLP1   C400     WTON    C7ED     XEQAD   C827
```

CONSOL^{TM} Monitor Program Source Listing

```
                      9999        COPY   CONSOL1/1                                1 OF 3 ******
                      0002 *
                      0003 *
                      0004 *
                      0005 *          CONSOL (TM)
                      0006 *
                      0007 *                     COPYRIGHT (C) 1976, 1977
                      0008 *                     PROCESSOR TECHNOLOGY CORP.
                      0009 *
                      0010 *    A L L   R I G H T S   R E S E R V E D ! ! !
                      0011 *
                      0012 *
                      0013 *                       *****             *
                      0014 *                      *                  *
                      0015 *                      * * *      **      *
                      0016 *                          *     *  *     *
                      0017 *                      *****      **      *
                      0018 *
                      0019 *                          SYSTEM SOLFTWARE
                      0020 *
                      0021 *
                      0022 *
                      0023 *   NOTE:   CONSOL, SOLOS AND SOLED ARE REGISTERED
                      0024 *           TRADEMARKS OF:
                      0025 *
                      0026 *                      PROCESSOR TECHNOLOGY CORP.
                      0027 *                          EMERYVILLE, CALIF
                      0028 *
                      0029 *
                      0030 *
                      0031 *                      =-=-=-     CONSOL     -=-=-=
                      0032 *
                      0033 *
                      0034 *    VERSION  1.3
                      0035 *    RELEASE  77-04-23
                      0036 *
                      0037 *
                      0038 *    THIS 2048 BYTE PROGRAM IS THE MINIMUM Sol STAND
                      0039 *  ALONE OPERATING SYSTEM.  IT IS CONFIGURED TO OPTIMIZE
                      0040 *  THE CONVENIENCE AND POWER OF THE Sol-20 AND ONE OR TWO
                      0041 *  CASSETTE RECORDERS IN STAND ALONE COMPUTER APPLICATIONS.
                      0042 *
                      0043 *
                      0044 *
                      0045 *   AUTO-STARTUP CODE
                      0046 *
C000 00               0047 START  DB     0      FOUR PHASE WONDER
C001 C3 4F C1         0048 INIT   JMP    STRTA  SYSTEM RESTART ENTRY POINT
                      0049 *
                      0050 *
                      0051 *                   ENTRY POINTS
```

```
                      0052 *
                      0053 *     THESE JUMP POINTS ARE PROVIDED TO ALLOW COMMON ENTRY
                      0054 *  LOCATIONS FOR ALL VERSIONS OF SOLOS.  THEY ARE USED
                      0055 *  EXTENSIVLY BY Sol SYSTEM PROGRAMS AND IT IS RECOMMENDED
                      0056 *  THAT USER ROUTINES ACCESS SOLOS THROUGH THESE POINTS.
                      0057 *  THIS JUMP TABLE IS PRESENT IN CONSOL TO PROVIDE
                      0058 *  A COMPATIBLE SYSTEM BETWEEN CONSOL///SOLOS///CUTER.
                      0059 *
C004 C3 69 C1         0060 RETRN  JMP    COMND  RETURN TO SYSTEM ENTRY POINT
C007 C3 69 C1         0061 FOPEN  JMP    COMND  ---NOT SUPPORTED---
C00A C3 69 C1         0062 FCLOS  JMP    COMND  ---NOT SUPPORTED---
C00D C3 69 C1         0063 RDBYT  JMP    COMND  ---NOT SUPPORTED---
C010 C3 69 C1         0064 WRBYT  JMP    COMND  ---NOT SUPPORTED---
C013 C3 69 C1         0065 RDBLK  JMP    COMND  ---NOT SUPPORTED---
C016 C3 69 C1         0066 WRBLK  JMP    COMND  ---NOT SUPPORTED---
                      0067 *
                      0068 *
                      0069 *       SYSTEM I/O ENTRY POINTS
                      0070 *
                      0071 *  THESE ROUTINES PERFORM SYSTEM I/O
                      0072 *  THERE ARE TWO ENTRY TYPES:
                      0073 *   SINP/SOUT      REG "A" WILL BE SET TO THE STANDARD
                      0074 *                  SYSTEM PSEUDO PORT.
                      0075 *   AINP/AOUT      REG "A" MUST BE SET BY THE USER AND WILL
                      0076 *                  SPECIFY THE DESIRED PSEUDO PORT.
                      0077 *
                      0078 *  THE FOLLOWING ARE THE PSEUDO PORTS:
                      0079 *    PORT    DESCRIPTION
                      0080 *    ----    ----------------------------------------
                      0081 *     0      KEYBOARD WHEN INPUT, AND VDM WHEN OUTPUT
                      0082 *     1      SERIAL
                      0083 *     2      PARALLEL
                      0084 *     3      USER DEFINED
                      0085 *
C019 3A 07 C8         0086 SOUT   LDA    OPORT  SOUT ENTRY POINT
C01C C3 3B C0         0087 AOUT   JMP    OUTPR  AOUT ENTRY POINT
C01F 3A 06 C8         0088 SINP   LDA    IPORT  SINP ENTRY POINT
     C022             0089 AINP   EQU    $      AINP ENTRY POINT
                      0090 * * * * * * * * * * END OF SYSTEM ENTRY POINTS * * * *
C022 E5               0091        PUSH   H      THIS IS ACTUALLY AINP
C023 21 1B C2         0092        LXI    H,ITAB
                      0093 *
                      0094 *
                      0095 *     THIS ROUTINE PROCESSES THE I/O REQUESTS BY DISPATCHING
                      0096 *  TO THE DRIVER REQUESTED IN REGISTER "A".  ON ENTRY HL
                      0097 *  HAVE THE PROPER DISPATCH TABLE.
                      0098 *
C026 E6 03            0099 IOPRC  ANI    3      KEEP REGISTER "A" TO FOUR VALUES
C028 07               0100        RLC    .      COMPUTE ENTRY ADDRESS
C029 85               0101        ADD    L
C02A 6F               0102        MOV    L,A    WE HAVE ADDRESS
C02B C3 B8 C1         0103        JMP    DISPT  DISPATCH TO IT
                      0104 *
```

```
                     *
                     *
                     *
                     *
                     *               ----=  Sol SYSTEM I/O ROUTINES =----
                     *
                     *
                     *   THIS ROUTINE IS A MODEL OF ALL INPUT ROUTINES WITHIN
                     * SOLOS.  EACH ROUTINE FIRST TESTS THE STATUS INPUT FOR
                     * DATA AVAILABLE.  IF NO CHARACTER HAS BEEN RECEIVED THE
                     * ROUTINE RETURNS WITH THE ZERO FLAG SET.  OTHERWISE THE
                     * CHARACTER IS INPUT AND A RETURN MADE WITH THE CHARACTER
                     * IN THE ACCUMULATOR AND THE ZERO FLAG RESET.
                     *
                     *
                     *        KEYBOARD INPUT DRIVER
                     *
C02E DB FA           KSTAT  IN     STAPT  GET STATUS WORD
C030 2F                     CMA    .      INVERT IT FOR PROPER RETURN
C031 E6 01                  ANI    KDR    TEST KEYBOARD BIT
C033 C8                     RZ     .      ZERO IF NO CHARACTER RECEIVED
                     *
C034 DB FC                  IN     KDATA  GET CHARACTER
C036 C9                     RET    .      GO BACK WITH IT
                     *
                     *
                     *   THIS JUMP IS PART OF THE AUTO START UP CODE
                     *
C037 00                     DB     0      ****VERIFY ADDR=C037****
C038 C3 01 C0               JMP    INIT
                     *
                     *
                     *             JUMP TABLE OUTPUT ROUTINES
                     *
                     *    THIS ROUTINE SETS UP THE DISPATCH TABLE FOR OUTPUT
                     * ROUTINES.  THE CHARACTER FOR OUTPUT IS IN REGISTER "B".
                     * OUTPUT IS MADE TO THE DRIVER POINTED TO BY THE REGISTER
                     * "A".  THE DEVICE DRIVERS ARE DEFINED AS FOLLOWS:
                     *
                     *
                     *       0 - DISPLAY SCREEN
                     *       1 - SERIAL OUTPUT PORT
                     *       2 - PARALLEL OUTPUT PORT
                     *       3 - USER DEFINED OR ERROR FLAG
                     *
                     * ENTRY AT:  SOUT SELECTS CURRENT OUTPUT DEVICE
                     *            AOUT SELECTS DEVICE IN REGISTER 'A'
                     *
C03B E5             OUTPR  PUSH   H
C03C 21 13 C2               LXI    H,OTAB  POINT TO OUTPUT TABLE
C03F C3 26 C0               JMP    IOPRC  AND DISPATCH TO OUTPUT ROUTINE
                     *
                     *
```

```
                     *
                     *
                     *          SERIAL INPUT DRIVER
                     *
C042 DB F8           SSTAT  IN     SERST  GET SERIAL STATUS WORD
C044 E6 40                  ANI    SDR    TEST FOR SERIAL DATA READY
C046 C8                     RZ     .      FLAGS ARE SET
                     *
C047 DB F9                  IN     SDATA  GET DATA BYTE
C049 C9                     RET    .      WE HAVE IT
                     *
                     *
                     *  SERIAL DATA OUTPUT
                     *
C04A DB F8           SDROT  IN     SERST  GET PORT STATUS
C04C 17                     RAL    .      PUT HIGH BIT IN CARRY
C04D D2 4A C0               JNC    SDROT  LOOP UNTIL TRANSMITTER BUFFER IS EMPTY
C050 78                     MOV    A,B    GET THE CHARACTER BACK
C051 D3 F9                  OUT    SDATA  SEND IT OUT
C053 C9                     RET    .      AND WE'RE DONE
                     *
                     *
                     *
                     *
                     *
                     *                   VIDEO DISPLAY ROUTINES
                     *
                     *
                     * THESE ROUTINES ALLOW FOR STANDARD VIDEO TERMINAL
                     * OPERATIONS.  ON ENTRY, THE CHARACTER FOR OUTPUT IS IN
                     * REGISTER B AND ALL REGISTERS EXCEPT "A" AND FLAGS ARE
                     * UNALTERED ON RETURN.
                     *
   C054              VDMOX  EQU    $      SPECIAL ENTRY POINT TO IGN CTL CHARS FM USER
C054 78                     MOV    A,B    GET THE CHAR
C055 FE 0A                  CPI    LF     IS IT A CTL CHAR TO BE IGNORED???
C057 D8                     RC     .      YES--IGNORE EM
C058 FE 1B                  CPI    ESC    ALSO THIS ONE TO BE IGNORED
C05A C8                     RZ     .      YUP, IGNORE IT
                     *
                     *
C05B E5              VDMOT  PUSH   H      SAVE MOST REGISTERS
C05C D5                     PUSH   D
C05D C5                     PUSH   B
                     *
                     *
C05E 78              CHPCK  MOV    A,B    SAVE IN B...STRIP PARITY BEFORE SCREEN!
C05F E6 7F                  ANI    7FH    CLR PARITY TO LOCATE IN TBL
C061 47                     MOV    B,A    KEEP IT W/OUT PARITY IN B TOO
C062 CA 72 C0               JZ     GOBK   DO A QUICK EXIT IF A NULL
C065 21 F7 C1               LXI    H,TBL  POINT TO SPECIAL CHARACTER TABLE
C068 CD 78 C0               CALL   TSRCH  GO PROCESS
                     *
```

```
C06B CD 12 C1      GOBACK CALL   VDADD  GET SCREEN ADDRESS
C06E 7E                   MOV    A,M    GET PRESENT CURSOR CHARACTER
C06F F6 80                ORI    80H
C071 77                   MOV    M,A    CURSOR IS BACK ON
C072 C1            GOBK   POP    B
C073 D1                   POP    D      RESTORE REGISTERS
C074 E1                   POP    H
C075 C9                   RET    .      EXIT FROM VDMOT
                   *
C076 23            NEXT   INX    H
C077 23                   INX    H
                   *
                   *
                   *   THIS ROUTINE SEARCHES THROUGH A SINGLE CHARACTER
                   * TABLE FOR A MATCH TO THE CHARACTER IN "B".  IF FOUND
                   * A DISPATCH IS MADE TO THE ADDRESS FOLLOWING THE MATCHED
                   * CHARACTER.  IF NOT FOUND THE CHAR/ :TER IS DISPLAYED ON
                   * THE MONITOR.
                   *
C078 7E            TSRCH  MOV    A,M    GET CHR FROM TABLE
C079 B7                   ORA    A
C07A CA 8A C0             JZ     CHAR   ZERO IS THE LAST
C07D B8                   CMP    B      TEST THE CHR
C07E 23                   INX    H      POINT FORWARD
C07F C2 76 C0             JNZ    NEXT
C082 E5                   PUSH   H      FOUND ONE...SAVE ADDRESS
C083 CD 2C C1             CALL   CREM   REMOVE CURSOR
C086 E3                   XTHL   .      GET DISPATCH ADDRESS TO HL
C087 C3 B8 C1             JMP    DISPT  DISPATCH NOW
                   *
                   *      PUT CHARACTER TO SCREEN
                   *
C08A 78            CHAR   MOV    A,B    GET CHARACTER
C08B FE 7F                CPI    7FH    IS IT A DEL?
C08D C8                   RZ     .      GO BACK IF SO
                   *
                   *
                   *
    C08E           OCHAR  EQU    $      ACTUALLY PUT CHAR TO SCREEN NOW
C08E CD 12 C1             CALL   VDADD  GET SCREEN ADDRESS
C091 70                   MOV    M,B    PUT CHR ON SCREEN
                   *
C092 3A 08 C8             LDA    NCHAR  GET CHARACTER POSITION
C095 FE 3F                CPI    63     END OF LINE?
C097 DA B7 C0             JC     OK
C09A 3A 09 C8             LDA    LINE
C09D FE 0F                CPI    15     END OF SCREEN?
C09F C2 B7 C0             JNZ    OK
                   *
                   *   END OF SCREEN...ROLL UP ONE LINE
                   *
C0A2 AF            SCROLL XRA    A
C0A3 32 08 C8             STA    NCHAR  BACK TO FIRST CHAR POSITION
```

```
C0A6 4F            SROL   MOV    C,A
C0A7 CD 19 C1             CALL   VDAD   CALCULATE LINE TO BE BLANKED
C0AA AF                   XRA    A
C0AB CD F0 C0             CALL   CLIN1  CLEAR IT
C0AE 3A 0A C8             LDA    BOT
C0B1 3C                   INR    A
C0B2 E6 0F                ANI    0FH
C0B4 C3 E4 C0             JMP    ERAS3
                   *
                   *   INCREMENT LINE COUNTER IF NECESSARY
                   *
C0B7 3A 08 C8      OK     LDA    NCHAR  GET CHR POSITION
C0BA 3C                   INR    A
C0BB E6 3F                ANI    3FH    MOD 64 AND WRAP
C0BD 32 08 C8             STA    NCHAR
C0C0 C0                   RNZ    .      DIDN'T HIT END OF LINE, OK
    C0C1           PDOWN  EQU    $      CURSOR DOWN ONE LINE HERE
C0C1 3A 09 C8             LDA    LINE   GET THE LINE COUNT
C0C4 3C                   INR    A
C0C5 E6 0F         CURSC  ANI    0FH    MOD 15 INCREMENT
C0C7 32 09 C8      CUR    STA    LINE   STORE THE NEW
C0CA C9                   RET
                   *
                   *    ERASE SCREEN
                   *
C0CB 21 00 CC      PERSE  LXI    H,VDMEM  POINT TO SCREEN
C0CE 36 A0                MVI    M,80H+' '  THIS IS THE CURSOR
                   *
C0D0 23                   INX    H      BUMP 1ST
    C0D1           ERAS1  EQU    $      LOOPS HERE TO ERASE SCREEN
C0D1 36 20                MVI    M,' '  BLANK IT OUT
C0D3 23                   INX    H      NEXT
C0D4 7C                   MOV    A,H    SEE IF END OF SCREEN YET
C0D5 FE D0                CPI    0D0H   ?
C0D7 DA D1 C0             JC     ERAS1  NO--KEEP BLANKING
C0DA 37                   STC    .      CARRY WILL SAY COMPLETE ERASE
                   *
C0DB 3E 00         PHOME  MVI    A,0    RESET CURSOR--CARRY=ERASE, ELSE HOME
C0DD 32 09 C8             STA    LINE   ZERO LINE
C0E0 32 08 C8             STA    NCHAR  LEFT SIDE OF SCREEN
C0E3 D0                   RNC    .      IF NO CARRY, WE ARE DONE WITH HOME
                   *
C0E4 D3 FE         ERAS3  OUT    DSTAT  RESET SCROLL PARAMETERS
C0E6 32 0A C8             STA    BOT    BEGINNING OF TEXT OFFSET
C0E9 C9                   RET
                   *
                   *
C0EA CD 12 C1      CLINE  CALL   VDADD  GET CURRENT SCREEN ADDRESS
C0ED 3A 08 C8             LDA    NCHAR  CURRENT CURSOR POSITION
C0F0 FE 40         CLIN1  CPI    64     NO MORE THAN 63
C0F2 D0                   RNC    .      ALL DONE
C0F3 36 20                MVI    M,' '  ALL SPACED OUT
C0F5 23                   INX    H
```

```
C0F6 3C                 0317         INR    A
C0F7 C3 F0 C0           0318         JMP    CLIN1  LOOP TO END OF LINE
                        0319 *
                        0320 *
                        0321 *  ROUTINE TO MOVE THE CURSOR UP ONE LINE
                        0322 *
C0FA 3A 09 C8           0323 PUP     LDA    LINE   GET LINE COUNT
C0FD 3D                 0324         DCR    A
C0FE C3 C5 C0           0325         JMP    CURSC  MERGE TO HANDLE CURSOR
                        0326 *
                        0327 *  MOVE CURSOR LEFT ONE POSITION
                        0328 *
C101 3A 08 C8           0329 PLEFT   LDA    NCHAR
C104 3D                 0330         DCR    A
     C105               0331 PCUR    EQU    $      CURSOR ON SAME LINE
C105 E6 3F              0332         ANI    3FH    LET CURSOR WRAP
C107 32 08 C8           0333         STA    NCHAR  UPDATED CURSOR
C10A C9                 0334         RET
                        0335 *
                        0336 *     CURSOR RIGHT ONE POSITION
                        0337 *
C10B 3A 08 C8           0338 PRIT    LDA    NCHAR
C10E 3C                 0339         INR    A
C10F C3 05 C1           0340         JMP    PCUR
                        0341 *
                        0342 *   ROUTINE TO CALCULATE SCREEN ADDRESS
                        0343 *
                        0344 *   ENTRY AT:    RETURNS:
                        0345 *
                        0346 *        VDADD   CURRENT SCREEN ADDRESS
                        0347 *        VDAD2   ADDRESS OF CURRENT LINE, CHAR 'C'
                        0348 *        VDAD    LINE 'A', CHARACTER POSITION 'C'
                        0349 *
C112 3A 08 C8           0350 VDADD   LDA    NCHAR  GET CHARACTER POSITION
C115 4F                 0351         MOV    C,A    'C' KEEPS IT
C116 3A 09 C8           0352 VDAD2   LDA    LINE   LINE POSITION
C119 6F                 0353 VDAD    MOV    L,A    INTO 'L'
C11A 3A 0A C8           0354         LDA    BOT    GET TEXT OFFSET
C11D 85                 0355         ADD    L      ADD IT TO THE LINE POSITION
C11E 0F                 0356         RRC    .      TIMES TWO
C11F 0F                 0357         RRC    .      MAKES FOUR
C120 6F                 0358         MOV    L,A    L HAS IT
C121 E6 03              0359         ANI    3      MOD THREE FOR LATER
C123 C6 CC              0360         ADI    <VDMEM  LOW SCREEN OFFSET
C125 67                 0361         MOV    H,A    NOW H IS DONE
C126 7D                 0362         MOV    A,L    TWIST L'S ARM
C127 E6 C0              0363         ANI    0C0H
C129 81                 0364         ADD    C
C12A 6F                 0365         MOV    L.A
C12B C9                 0366         RET    .      H & L ARE NOW PERVERTED
                        0367 *
                        0368 *    ROUTINE TO REMOVE CURSOR
                        0369 *
```

```
C12C CD 12 C1           0370 CREM    CALL   VDADD  GET CURRENT SCREEN ADDRESS
C12F 7E                 0371         MOV    A,M
C130 E6 7F              0372         ANI    7FH    STRIP OFF THE CURSOR
C132 77                 0373         MOV    M,A
C133 C9                 0374         RET
                        0375 *
                        0376 *     ROUTINE TO BACKSPACE
                        0377 *
C134 CD 01 C1           0378 PBACK   CALL   PLEFT
C137 CD 12 C1           0379         CALL   VDADD  GET SCREEN ADDRESS
C13A 36 20              0380         MVI    M,' '  PUT A BLANK THERE
C13C C9                 0381         RET
                        0382 *
                        0383 *     ROUTINE TO PROCESS A CARRIAGE RETURN
                        0384 *
C13D CD EA C0           0385 PCR     CALL   CLINE  CLEAR FROM CURRENT CURSOR TO END OF LINE
                        0386 *  NOTE THAT "A" COMES BACK=64 AND IS CLEARED BY PCUR
C140 C3 05 C1           0387         JMP    PCUR   AND STORE THE NEW VALUE
                        0388 *
                        0389 *  ROUTINE TO PROCESS LINEFEED
                        0390 *
C143 3A 09 C8           0391 PLF     LDA    LINE   GET LINE COUNT
C146 3C                 0392         INR    A
C147 E6 0F              0393         ANI    15     SEE IF IT WRAPPED AROUND
C149 C2 C7 C0           0394         JNZ    CUR    NO--NO NEED TO SCROLL
C14C C3 A6 C0           0395         JMP    SROL
                        0396 *
                        0397 *
                        0398 *               START UP SYSTEM
                        0399 *
                        0400 *  CLEAR SCREEN AND THE FIRST 256 BYTES OF GLOBAL RAM
                        0401 * THEN ENTER THE COMMAND MODE.
                        0402 *
C14F AF                 0403 STRTA   XRA    A
C150 4F                 0404         MOV    C,A
C151 21 00 C8           0405         LXI    H,SYSRAM  CLEAR THE FIRST PAGE
                        0406 *
C154 77                 0407 CLERA   MOV    M,A
C155 23                 0408         INX    H
C156 0C                 0409         INR    C
C157 C2 54 C1           0410         JNZ    CLERA
                        0411 *
C15A 31 FF CB           0412         LXI    SP,SYSTP  SET UP THE STACK FOR CALL
C15D CD CB C0           0413         CALL   PERSE
     C160               0414 ERROT   EQU    $      INVALID I/O PORTS COME HERE FOR CONSOL
C160 AF                 0415 COMN1   XRA    A
C161 D3 FA              0416         OUT    STAPT  BE SURE TAPES ARE OFF
C163 32 07 C8           0417         STA    OPORT
C166 32 06 C8           0418         STA    IPORT
                        0419 *
                        0420 *
                        0421 *
                        0422 *             =--  COMMAND MODE  --=
                    0423 *
                    0424 *
                    0425 *   THIS ROUTINE GETS AND PROCESSES COMMANDS
                    0426 *
C169 31 FF CB       0427 COMND  LXI    SP,SYSTP  SET STACK POINTER
C16C CD 23 C2       0428        CALL   PROMPT  PUT PROMPT ON SCREEN
C16F CD 78 C1       0429        CALL   GCLIN   GET COMMAND LINE
C172 CD 99 C1       0430        CALL   COPRC   PROCESS THE LINE
C175 C3 69 C1       0431        JMP    COMND   OVER AND OVER
                    0432 *
                    0433 *
                    0434 *
                    0435 *   THIS ROUTINE READS A COMMAND LINE FROM THE SYSTEM
                    0436 *  KEYBOARD
                    0437 *
                    0438 *  C/R   TERMINATES THE SEQUENCE ERASING ALL CHARS TO THE
                    0439 *        RIGHT OF THE CURSOR
                    0440 *  L/F   TERMINATES THE SEQUENCE
                    0441 *  MODE  RESTARTS THE COMMAND LINE.
                    0442 *
C178 CD 2E C0       0443 GCLIN  CALL   KSTAT   GET A CHAR FM SOL KEYBOARD
C17B CA 78 C1       0444        JZ     GCLIN
C17E E6 7F          0445        ANI    7FH     CLEAR PARITY BIT
C180 CA 60 C1       0446        JZ     COMN1   THIS WAS A MODE (OR EVEN CTL-@)
C183 47             0447        MOV    B,A
C184 FE 0D          0448        CPI    CR      CARRIAGE RETURN
C186 CA EA C0       0449        JZ     CLINE   YES--DONE WITH LINE
C189 FE 0A          0450        CPI    LF      LINE FEED
C18B C8             0451        RZ     .       YES--DONE WITH LINE, LEAVE AS IS
C18C FE 7F          0452        CPI    7FH     DELETE CHR?
C18E C2 93 C1       0453        JNZ    CONT
C191 06 5F          0454        MVI    B,BACKS  REPLACE IT
                    0455 *
C193 CD 5B C0       0456 CONT   CALL   VDMOT   OUTPUT TO VDM ALWAYS
C196 C3 78 C1       0457        JMP    GCLIN
                    0458 *
                    0459 *
                    0460 *      FIND AND PROCESS COMMAND
                    0461 *
C199 CD 2C C1       0462 COPRC  CALL   CREM    REMOVE THE CURSOR
C19C 0E 01          0463        MVI    C,1     SET FOR CHARACTER POSITION
C19E CD 16 C1       0464        CALL   VDAD2   GET SCREEN ADDRESS
C1A1 EB             0465        XCHG
C1A2 21 00 C0       0466        LXI    H,START  MAKE SURE HL PT TO SOLOS START
C1A5 E5             0467        PUSH   H       SAVE IT FOR LATER DISPT
C1A6 CD 4B C2       0468        CALL   SCHR    SCAN PAST BLANKS
C1A9 CA 3C C3       0469        JZ     ERR1    NO COMMAND?
C1AC EB             0470        XCHG   .       HL HAS FIRST CHR
                    0471 *
C1AD 11 DE C1       0472        LXI    D,COMTAB  POINT TO COMMAND TABLE
C1B0 CD C5 C1       0473        CALL   FDCOM   SEE IF IN PRIMARY COMMAND TABLE
C1B3 CA 3D C3       0474        JZ     ERR2    NOT VALID, ERROR
C1B6 13             0475        INX    D       BUMP TO PTR OF RTN
```

```
C1B7 EB             0476        XCHG   .       HL PT TO RTN ADDR
                    0477 *
                    0478 *
                    0479 *  THIS IS THE DISPATCH ROUTINE.
                    0480 *  HL PT TO RTN ADDRESS, HL WILL BE RESTORED FM STACK
                    0481 *  SO THAT HL ARE RESTORED BEFORE DISPATCH.
                    0482 *
   C1B8             0483 DISPT  EQU    $       OFF TO A ROUTINE
C1B8 7E             0484        MOV    A,M     LO ADDR
C1B9 23             0485        INX    H
C1BA 66             0486        MOV    H,M     HI ADDR
C1BB 6F             0487        MOV    L,A     HL NOW COMPLETE
   C1BC             0488 DISP1  EQU    $       HERE TO GO OFF TO HL
C1BC E3             0489        XTHL   .       XCHG HL W/HL ON STACK
C1BD 7D             0490        MOV    A,L     ALSO COPY HERE FOR SETS
C1BE C9             0491        RET    .       AND GO OFF TO THE RTN
                    0492 *
                    0493 *
C1BF 21 00 00       0494 GOBAS  LXI    H,0     THIS EXECUTES SOL BASIC LOCATION ZERO
C1C2 C3 37 C3       0495        JMP    EXEC1   BECAUSE HL MUST BE PASSED PROPERLY
                    0496 *
                    0497 *   THIS ROUTINE SEARCHES THROUGH A TABLE, POINTED TO
                    0498 *  BY 'DE', FOR A DOUBLE CHARACTER MATCH OF THE 'HL'
                    0499 *  MEMORY CONTENT.  IF NO MATCH IS FOUND THE SCAN ENDS
                    0500 *  WITH HL POINTING TO ORIGINAL VALUE AND ZERO FLAG SET.
                    0501 *
                    0502 *
C1C5 1A             0503 FDCOM  LDAX   D
C1C6 B7             0504        ORA    A       TEST FOR TABLE END
C1C7 C8             0505        RZ     .       NOT FOUND..COMMAND ERROR
C1C8 E5             0506        PUSH   H       SAVE START OF SCAN ADDRESS
C1C9 BE             0507        CMP    M       TEST FIRST CHR
C1CA 13             0508        INX    D
C1CB C2 D7 C1       0509        JNZ    NCOM
                    0510 *
C1CE 23             0511        INX    H
C1CF 1A             0512        LDAX   D
C1D0 BE             0513        CMP    M       NOW SECOND CHARACTER
C1D1 C2 D7 C1       0514        JNZ    NCOM    GOODNESS
                    0515 *
C1D4 E1             0516        POP    H       RESTORE ORIGINAL SCAN ADDR
C1D5 B7             0517        ORA    A       SET NON-ZERO FLAG SAYING FOUND
C1D6 C9             0518        RET    .       WITH NON-ZERO SET
                    0519 *
                    0520 *
C1D7 13             0521 NCOM   INX    D       GO TO NEXT ENTRY
C1D8 13             0522        INX    D
C1D9 13             0523        INX    D
C1DA E1             0524        POP    H       GET BACK ORIGINAL ADDRESS
C1DB C3 C5 C1       0525        JMP    FDCOM   CONTINUE SEARCH
                    0526 *
                    0527 *
                    0528 *                 COMMAND TABLE
                    0529 *
                    0530 *  THIS TABLE DESCRIBES THE VALID COMMANDS FOR SOLOS
                    0531 *
C1DE 54 45          0532 COMTAB ASC    'TE'   TERMINAL MODE
C1E0 B4 C2          0533        DW     TERM
C1E2 44 55          0534        ASC    'DU'   DUMP
C1E4 3E C2          0535        DW     DUMP
C1E6 45 4E          0536        ASC    'EN'   ENTR
C1E8 C7 C3          0537        DW     ENTER
C1EA 45 58          0538        ASC    'EX'   EXEC
C1EC 34 C3          0539        DW     EXEC
C1EE 47 45          0540        ASC    'GE'   GET A FILE
C1F0 42 C3          0541        DW     TLOAD
C1F2 42 41          0542        ASC    'BA'   BASIC
C1F4 EF C1          0543        DW     GOBAS
C1F6 00             0544        DB     0      END OF TABLE MARK
                    0545 *
                    0546 *
                    0547 *                 DISPLAY DRIVER COMMAND TABLE
                    0548 *
                    0549 *    THIS TABLE DEFINES THE CHARACTERS FOR SPECIAL
                    0550 *  PROCESSING.  IF THE CHARACTER IS NOT IN THE TABLE IT
                    0551 *  GOES TO THE SCREEN.
                    0552 *
C1F7 0B             0553 TBL    DB     CLEAR-80H  SCREEN
C1F8 C3 C0          0554        DW     PERSE
C1FA 17             0555        DB     UP-80H  CURSOR
C1FB F1 C0          0556        DW     PUP
C1FD 1A             0557        DB     DOWN-80H
C1FE C1 C0          0558        DW     PDOWN
C200 01             0559        DB     LEFT-80H
C201 01 C1          0560        DW     PLEFT
C203 13             0561        DB     RIGHT-80H
C204 03 C1          0562        DW     PRIT
C206 0E             0563        DB     HOME-80H
C207 D5 C0          0564        DW     PHOME
C209 0D             0565        DB     CR     CARRIAGE RETURN
C20A 31 C1          0566        DW     PCR
C20C 0A             0567        DB     LF     LINE FEED
C20D 41 C1          0568        DW     PLF
C20F 5F             0569        DB     BACKS  BACK SPACE
C210 34 C1          0570        DW     PBACK
C212 00             0571        DB     0      END OF TABLE
                    0572 *
                    0573 *
                    0574 *            OUTPUT DEVICE TABLE
                    0575 *
C213 54 C0          0576 OTAB   DW     VDMOX  SPECIAL VDM ENTRY POINT TO IGN CTL CHARS
C215 4A C0          0577        DW     SDROT  SERIAL OUTPUT
C217 60 C1          0578        DW     ERROT  PARALLEL NOT SUPPORTED BY CONSOL
C219 60 C1          0579        DW     ERROT  USER ROUTINE NOT SUPPORTED BY CONSOL
                    0580 *
                    0581 *
```

```
                    0582 *          INPUT DEVICE TABLE
                    0583 *
C21B 2E C0          0584 ITAB   DW     KSTAT  KEYBOARD INPUT
C21D 42 C0          0585        DW     SSTAT  SERIAL INPUT
C21F 60 C1          0586        DW     ERROT  PARALLEL NOT SUPPORTED BY CONSOL
C221 60 C1          0587        DW     ERROT  USER WOUTINE NOT SUPPORTED BY CONSOL
                    0588 *
                    0589 *
                    0590 *
                    0591 * -*-
                    9999        COPY   CONSOL2/1                              2 OF 3 ******
                    0592 *
                    0593 *
                    0594 *      OUTPUT A CRLF FOLLOWED BY A PROMPT
                    0595 *
C223 CD 2B C2       0596 PROMPT CALL   CRLF
C226 06 3E          0597        MVI    B,'>'  THE PROMPT
C228 C3 19 C0       0598        JMP    SOUT   PUT IT ON THE SCREEN
                    0599 *
                    0600 *
C22B 06 0A          0601 CRLF   MVI    B,LF   LINE FEED
C22D CD 5B C0       0602        CALL   VDMOT
C230 06 0D          0603        MVI    B,CR   CARRIAGE RETURN
C232 C3 5B C0       0604        JMP    VDMOT
                    0605 *
                    0606 *
                    0607 *    SCAN OFF OPTIONAL PARAMETER.  IF PRESENT RETURN WITH
                    0608 * VALUE IN HL AND COPY OF "L" IN "A".  IF NOT PRESENT
                    0609 * RETURN WITH A "1" IN "A" AND HL UNTOUCHED.
                    0610 *
C235 CD 3D C2       0611 PSCAN  CALL   SBLK
C238 C8             0612        RZ     .      IF NONE
C239 CD 5D C2       0613        CALL   SHEX   CONVERT VALUE
C23C C9             0614        RET
                    0615 *
                    0616 *
                    0617 *  SCAN OVER UP TO 12 CHARACTERS LOOKING FOR A BLANK
                    0618 *
C23D 0E 0C          0619 SBLK   MVI    C,12   MAXIMUM COMMAND STRING
C23F 1A             0620 SBLK1  LDAX   D
C240 FE 20          0621        CPI    BLANK
C242 CA 4B C2       0622        JZ     SCHR   GOT A BLANK NOW SCAN PAST IT
C245 13             0623        INX    D
C246 0D             0624        DCR    C      NO MORE THAN TWELVE
C247 C2 3F C2       0625        JNZ    SBLK1
C24A C9             0626        RET    .      GO BACK WITH ZERO FLAG SET
                    0627 *
                    0628 *
                    0629 *    SCAN PAST UP TO 10 BLANK POSITIONS LOOKING FOR
                    0630 *  A NON BLANK CHARACTER.
                    0631 *
C24B 0E 0A          0632 SCHR   MVI    C,10   SCAN TO FIRST NON BLANK CHR WITHIN 10
C24D 1A             0633 SCHR1  LDAX   D      GET NEXT CHARACTER
```

```
C24E FE 20                      CPI    SPACE
C250 C0                         RNZ    .      WE'RE PAST THEM
C251 13                         INX    D      NEXT SCAN ADDRESS
C252 0D                         DCR    C
C253 C8                         RZ     .      COMMAND ERROR
C254 C3 4D C2                   JMP    SCHR1  KEEP LOOPING
                         *
                         *
                         *    THIS ROUTINE SCANS OVER CHARACTERS, PAST BLANKS AND
                         * CONVERTS THE FOLLOWING VALUE TO HEX.  ERRORS RETURN TO
                         * THE ERROR HANDLER.
                         *
C257 CD 3D C2            SCONV  CALL   SBLK   FIND IF VALUE IS PRESENT
C25A CA 3C C3                   JZ     ERR1   ABORT TO ERROR IF NONE
                         *
                         *
                         *  THIS ROUTINE CONVERTS ASCII DIGITS INTO BINARY FOLLOWING
                         * A STANDARD HEX CONVERSION.  THE SCAN STOPS WHEN AN ASCII
                         * SPACE IS ENCOUNTERED.  PARAMETER ERRORS REPLACE THE ERROR
                         * CHARACTER ON THE SCREEN WITH A QUESTION MARK.
                         *
C25D 21 00 00            SHEX   LXI    H,0    CLEAR H & L
C260 1A                  SHE1   LDAX   D      GET CHARACTER
C261 FE 20                      CPI    20H    IS IT A SPACE?
C263 C8                         RZ     .      IF SO
C264 FE 2F                      CPI    '/'    SLASH IS ALSO LEGAL
C266 C8                         RZ
C267 FE 3A                      CPI    ':'    EVEN THE COLON IS ALLOWED
C269 C8                         RZ
                         *
C26A 29                  HCONV  DAD    H      MAKE ROOM FOR THE NEW ONE
C26B 29                         DAD    H
C26C 29                         DAD    H
C26D 29                         DAD    H
C26E CD 7A C2                   CALL   HCOV1  DO THE CONVERSION
C271 D2 3C C3                   JNC    ERR1   NOT VALID HEXIDECIMAL VALUE
C274 85                         ADD    L
C275 6F                         MOV    L,A    MOVE IT IN
C276 13                         INX    D      BUMP THE POINTER
C277 C3 60 C2                   JMP    SHE1
                         *
C27A D6 30               HCOV1  SUI    48     REMOVE ASCII BIAS
C27C FE 0A                      CPI    10
C27E D8                         RC     .      IF LESS THAN 9
C27F D6 07                      SUI    7      IT'S A LETTER
C281 FE 10                      CPI    10H
C283 C9                         RET    .      WITH TEST IN HAND
                         *
                         *
                         *                  TERM COMMAND
                         *
                         *    THIS ROUTINE GETS CHARACTERS FROM THE SYSTEM KEYBOARD
                         *  AND OUTPUTS THEM TO THE SERIAL PORT.  IT IS
```

```
                         *  INTENDED TO CONFIGURE THE Sol AS A STANDARD VIDEO
                         *  TERMINAL.  COMMAND KEYS ARE NOT OUTPUT TO THE OUTPUT
                         *  PORT BUT ARE INTERPRETED AS DIRECT Sol COMMANDS.
                         *  THE MODE COMMAND, RECEIVED BY THE KEYBOARD, PUTS THE
                         *  Sol IN THE COMMAND MODE.
                         *
                         *
                         *
    C284                 TERM   EQU    $      TERM COMMAND VIA PORT 1 ONLY
                         *
C284 CD 2E C0            TERM1  CALL   KSTAT  IS THERE ONE WAITING?
C287 CA 9C C2                   JZ     TIN    IF NOT
C28A 47                         MOV    B,A    SAVE IT IN B
C28B FE 80                      CPI    MODE   IS IT MODE
C28D CA 60 C1                   JZ     COMN1  YES--RESET AND QUIT TERM
C290 DA 99 C2                   JC     TOUT   NON-CURSOR KEY---SEND TO TERM PORT
C293 CD 5B C0                   CALL   VDMOT  PROCESS IT
C296 C3 9C C2                   JMP    TIN
                         *
C299 CD 4A C0            TOUT   CALL   SDROT  OUTPUT IT TO THE SERIAL PORT
C29C CD 42 C0            TIN    CALL   SSTAT  GET INPUT STATUS
C29F CA 84 C2                   JZ     TERM1  LOOP IF NOT
C2A2 E6 7F                      ANI    7FH    NO HIGH BITS FROM HERE
C2A4 CA 84 C2                   JZ     TERM1  A NULL IS IGNORED
C2A7 47                         MOV    B,A    IT'S OUTPUT FROM 'B'
C2A8 CD 5B C0                   CALL   VDMOT  PUT IT ON THE SCREEN
C2AB C3 84 C2                   JMP    TERM1  LOOP OVER AND OVER
                         *
                         *
                         *
                         *                  DUMP COMMAND
                         *
                         *    THIS ROUTINE DUMPS CHARACTERS FROM MEMORY TO THE
                         *  CURRENT OUTPUT DEVICE.  ALL VALUES ARE DISPLAYED AS
                         *  ASCII HEX.
                         *
                         *  THE COMMAND FORM IS AS FOLLOWS:
                         *
                         *              DUmp  addr1  addr2
                         *
                         *    THE VALUES FROM ADDR1 TO ADDR2 ARE THEN OUTPUT TO THE
                         *  OUTPUT DEVICE.  IF ONLY ADDR1 IS SPECIFIED THEN THE
                         *  VALUE AT THAT ADDRESS IS OUTPUT.
                         *
C2AE CD 57 C2            DUMP   CALL   SCONV  SCAN TO FIRST ADDRESS AND CONVERT IT
C2B1 E5                         PUSH   H      SAVE THE VALUE
C2B2 CD 35 C2                   CALL   PSCAN  SEE IF SECOND WAS GIVEN
C2B5 D1                         POP    D      GET BACK START
C2B6 EB                         XCHG   .      HL HAS START, DE HAS END
                         *
C2B7 CD 2B C2            DLOOP  CALL   CRLF
C2BA CD D7 C2                   CALL   ADOUT  OUTPUT ADDRESS
C2BD CD EA C2                   CALL   BOUT   ANOTHER SPACE TO KEEP IT PRETTY
```

```
C2C0 0E 10              0740           MVI    C,16    VALUES PER LINE
                        0741 *
C2C2 7E                 0742 DLP1      MOV    A,M     GET THE CHR
C2C3 C5                 0743           PUSH   B       SAVE VALUE COUNT
C2C4 CD DC C2           0744           CALL   HBOUT   SEND IT OUT WITH A BLANK
C2C7 7D                 0745           MOV    A,L     COMPARE DE & HL
C2C8 93                 0746           SUB    E
C2C9 7C                 0747           MOV    A,H
C2CA 9A                 0748           SBB    D
C2CB D2 60 C1           0749           JNC    COMN1   ALL DONE
C2CE C1                 0750           POP    B       VALUES PER LINE
C2CF 23                 0751           INX    H
C2D0 0D                 0752           DCR    C       BUMP THE LINE COUNT
C2D1 C2 C2 C2           0753           JNZ    DLP1    NOT ZERO IF MORE FOR THIS LINE
C2D4 C3 B7 C2           0754           JMP    DLOOP   DO A LFCR BEFORE THE NEXT
                        0755 *
                        0756 *
                        0757 *    OUTPUT HL AS HEX 16 BIT VALUE
                        0758 *
C2D7 7C                 0759 ADOUT     MOV    A,H     H FIRST
C2D8 CD EF C2           0760           CALL   HEOUT
C2DB 7D                 0761           MOV    A,L     THEN "L" FOLLOWED BY A SPACE
                        0762 *
C2DC CD EF C2           0763 HBOUT     CALL   HEOUT
C2DF CD 2E C0           0764           CALL   KSTAT   SEE IF A CHAR WAITING
C2E2 CA EA C2           0765           JZ     BOUT    NO
C2E5 E6 7F              0766           ANI    7FH     SEE IF MODE OR CTL-@
C2E7 CA 60 C1           0767           JZ     COMN1   YES--GET OUT
C2EA 06 20              0768 BOUT      MVI    B,' '
C2EC C3 5B C0           0769           JMP    VDMOT   PUT IT OUT
                        0770 *
C2EF 4F                 0771 HEOUT     MOV    C,A     GET THE CHARACTER
C2F0 0F                 0772           RRC
C2F1 0F                 0773           RRC    .       MOVE THE HIGH FOUR DOWN
C2F2 0F                 0774           RRC
C2F3 0F                 0775           RRC
C2F4 CD F8 C2           0776           CALL   HEOU1   PUT THEM OUT
C2F7 79                 0777           MOV    A,C     THIS TIME THE LOW FOUR
                        0778 *
C2F8 E6 0F              0779 HEOU1     ANI    0FH     FOUR ON THE FLOOR
C2FA C6 30              0780           ADI    48      WE WORK WITH ASCII HERE
C2FC FE 3A              0781           CPI    58      0-9?
C2FE DA 03 C3           0782           JC     OUTH    YUP!
C301 C6 07              0783           ADI    7       MAKE IT A LETTER
C303 47                 0784 OUTH      MOV    B,A     OUTPUT IT FROM REGISTER 'B'
C304 C3 5B C0           0785           JMP    VDMOT
                        0786 *
                        0787 *
                        0788 *                 ENTER COMMAND
                        0789 *
                        0790 *    THIS ROUTINE GETS VALUES FROM THE KEYBOARD AND ENTERS
                        0791 * THEM INTO MEMORY.  THE INPUT VALUES ARE SCANNED FOLLOWING
                        0792 * A STANDARD 'GCLIN' INPUT SO ON SCREEN EDITING MAY TAKE
```

```
                        0793 * PLACE PRIOR TO THE LINE TERMINATOR.  A BACK SLASH '/'
                        0794 * ENDS THE ROUTINE AND RETURNS CONTROL TO THE COMMAND MODE.
                        0795 *
C307 CD 57 C2           0796 ENTER     CALL   SCONV   SCAN OVER CHARS AND GET ADDRESS
C30A E5                 0797           PUSH   H       SAVE ADDRESS
                        0798 *
C30B CD 2B C2           0799 ENLOP     CALL   CRLF
C30E 06 3A              0800           MVI    B,':'
C310 CD 93 C1           0801           CALL   CONT    GET LINE OF INPUT
C313 CD 2C C1           0802           CALL   CREM    REMOVE THE CURSOR
C316 0E 01              0803           MVI    C,1     START SCAN
C318 CD 16 C1           0804           CALL   VDAD2   GET ADDRESS
C31B EB                 0805           XCHG   .       ....TO DE
                        0806 *
                        0807 *
C31C 0E 03              0808 ENLO1     MVI    C,3     NO MORE THAN THREE SPACES BETWEEN VALUES
C31E CD 4D C2           0809           CALL   SCHR1   SCAN TO NEXT VALUE
C321 CA 0B C3           0810           JZ     ENLOP   LAST ENTRY FOUND START NEW LINE
                        0811 *
C324 FE 2F              0812           CPI    '/'     COMMAND TERMINATOR?
C326 CA 60 C1           0813           JZ     COMN1   IF SO...RETURN TO STANDARD INPUT
C329 CD 5D C2           0814           CALL   SHEX    CONVERT VALUE
C32C 7D                 0815           MOV    A.L     GET LOW PART AS CONVERTED
C32D E1                 0816           POP    H       GET MEMORY ADDRESS
C32E 77                 0817           MOV    M,A     PUT IN THE VALUE
C32F 23                 0818           INX    H
C330 E5                 0819           PUSH   H       BACK GOES THE ADDRESS
C331 C3 1C C3           0820           JMP    ENLO1   CONTINUE THE SCAN
                        0821 *
                        0822 *
                        0823 *
                        0824 *                 EXECUTE COMMAND
                        0825 *
                        0826 *    THIS ROUTINE GETS THE FOLLOWING PARAMETER AND DOES A
                        0827 * PROGRAM JUMP TO THE LOCATION GIVEN BY IT.  IF PROPER
                        0828 * STACK OPERATIONS ARE USED WITHIN THE EXTERNAL PROGRAM
                        0829 * IT CAN DO A STANDARD 'RET'URN TO THE SOLOS COMMAND MODE.
                        0830 * THE STARTING ADDRESS OF SOLOS IS PASSED TO THE PROGRAM
                        0831 * IN REGISTER PAIR HL SO IT CAN ADJUST INTERNAL PARAMETERS
                        0832 * FOR SOLOS OPERATION.
                        0833 *
                        0834 *
C334 CD 57 C2           0835 EXEC      CALL   SCONV   SCAN PAST BLANKS AND GET PARAMETER
C337 E5                 0836 EXEC1     PUSH   H       PUT GO ADDRESS ON STACK
C338 21 00 C0           0837           LXI    H,START  TELL THE PROGRAM WHERE WE CAME FROM
C33B C9                 0838           RET    .       AND DISPATCH TO IT
                        0839 *
                        0840 *
                        0841 *
                        0842 *   SOLOS ERROR HANDLER
                        0843 *
C33C EB                 0844 ERR1      XCHG   .       GET SCAN ADDRESS TO HL
C33D 36 3F              0845 ERR2      MVI    M,'?'   PUT QUESTION MARK ON SCREEN
```

```
C33F C3 60 C1         0846        JMP    COMN1  AND RETURN TO COMMAND MODE
                      0847 *
                      0848 *
                      0849 *
                      0850 * -*-
                      9999        COPY   CONSOL3/1                                    3 OF 3 ******
                      0851 *
                      0852 *
                      0853 *  THIS ROUTINE PROCESSES THE "GET" COMMAND
                      0854 *
     C342             0855 TLOAD  EQU    $      PREP TO GET THE VERY NEXT FILE
C342 3A 0D C8         0856        LDA    TSPD   GET SPEED FM SYSTEM RAM AREA
C345 F6 C0            0857        ORI    TAPE1+TAPE2  TURN BOTH MACHINES ON
                      0858 *  DROP THRU TO READ IN THIS FILE
                      0859 *
                      0860 *               TAPE READ ROUTINE
                      0861 *
                      0862 *     ON ENTRY:     A   HAS UNIT AND SPEED
                      0863 *
                      0864 *     ON EXIT:      IF CARRY WAS SET---BELL TO SCREEN
                      0865 *                   OTHERWISE--NORMAL--BACK TO COMN1
                      0866 *
                      0867 *
     C347             0868 RTAPE  EQU    $      READ VERY NEXT TAPE FILE
C347 06 03            0869        MVI    B,3    SHORT DELAY
C349 CD EE C3         0870        CALL   TON
C34C DB FB            0871        IN     TDATA  CLEAR THE UART FLAGS
                      0872 *
     C34E             0873 PTAP1  EQU    $      HERE TO RD THE HDR
C34E CD 8E C3         0874        CALL   RHEAD  GO READ HEADER
C351 DA 78 C3         0875        JC     TERR   IF AN ERROR OR ESC WAS RECEIVED
C354 C2 4E C3         0876        JNZ    PTAP1  IF VALID HEADER NOT FOUND
                      0877 *
                      0878 *  FOUND A VALID HEADER NOW JUST READ THE FILE IN
                      0879 *
C357 2A 23 C8         0880        LHLD   BLOCK  GET BLOCK SIZE
C35A EB               0881        XCHG   .      ...TO DE
C35B 2A 25 C8         0882        LHLD   LOADR  GET TAPE LOAD ADDRESS
                      0883 *
                      0884 *
                      0885 *     THIS ROUTINE READS "DE" BYTES FROM THE TAPE
                      0886 *     TO ADDRESS HL.  THE BYTES MUST BE FROM ONE
                      0887 *     CONTIGUOUS PHYSICAL BLOCK ON THE TAPE.
                      0888 *
                      0889 *          HL HAS "PUT" ADDRESS
                      0890 *          DE HAS SIZE OF TAPE BLOCK
                      0891 *
     C35E             0892 RTAP   EQU    $      ACTUALLY RD IT NOW
                      0893 *
     C35E             0894 RTAP2  EQU    $      HERE TO LOOP RDING BLKS
C35E CD 80 C3         0895        CALL   DCRCT  DROP COUNT, B=LEN THIS BLK
C361 CA 75 C3         0896        JZ     RTOFF  ZERO=ALL DONE
                      0897 *
```

```
C364 CD AF C3         0898        CALL   RHED1  READ THAT MANY BYTES
C367 DA 78 C3         0899        JC     TERR   IF ERROR OR MODE STOPPED US
C36A CA 5E C3         0900        JZ     RTAP2  RD OK--READ SOME MORE
C36D C3 78 C3         0901        JMP    TERR   CRC ERROR--TAKE ERROR EXIT
                      0902 *
                      0903 *
                      0904 *
C370 06 01            0905 TOFF   MVI    B,1
C372 CD F0 C3         0906        CALL   DELAY
     C375             0907 RTOFF  EQU    $      TURN OFF TAPES AND EXIT
C375 D2 60 C1         0908        JNC    COMN1  WENT AOK,WE'RE DONE
     C378             0909 TERR   EQU    $      HERE WHEN AN ERROR IS TO STOP US
C378 06 07            0910        MVI    B,'G'-40H  WE HAD AN ERROR--PUT BELL ON SCREEN
C37A CD 5B C0         0911        CALL   VDMOT
C37D C3 60 C1         0912        JMP    COMN1  DONE AND TURN TAPES OFF
                      0913 *
                      0914 *
     C380             0915 DCRCT  EQU    $      COMMON RTN TO COUNT DOWN BLK LENGTHS
C380 AF               0916        XRA    A      CLR FOR LATER TESTS
C381 47               0917        MOV    B,A    SET THIS BLK LEN=256
C382 B2               0918        ORA    D      IS AMNT LEFT < 256
C383 C2 8B C3         0919        JNZ    DCRC2  NO--REDUCE AMNT BY 256
C386 B3               0920        ORA    E      IS ENTIRE COUNT ZERO
C387 C8               0921        RZ     ALL DONE--ZERO=THIS CONDITION
C388 43               0922        MOV    B,E    SET THIS BLK LEN TO AMNT REMAINING
C389 5A               0923        MOV    E,D    MAKE ENTIRE COUNT ZERO NOW
C38A C9               0924        RET    .      ALL DONE (NON-ZERO FLAG)
     C38B             0925 DCRC2  EQU    $      REDUCE COUNT BY 256
C38B 15               0926        DCR    D      DROP BY 256
C38C B7               0927        ORA    A      FORCE NON-ZERO FLAG
C38D C9               0928        RET    .      NON-ZERO=NOT DONE YET (BLK LEN=256)
                      0929 *
                      0930 *
                      0931 *   READ THE HEADER
                      0932 *
C38E 06 0A            0933 RHEAD  MVI    B,10   FIND 10 NULLS
C390 CD C8 C3         0934 RHEA1  CALL   STAT
C393 D8               0935        RC     .      IF ESCAPE
C394 DB FB            0936        IN     TDATA  IGNORE ERROR CONDITIONS
C396 B7               0937        ORA    A      ZERO?
C397 C2 8E C3         0938        JNZ    RHEAD
C39A 05               0939        DCR    B
C39B C2 90 C3         0940        JNZ    RHEA1  LOOP UNTIL 10 IN A ROW
                      0941 *
                      0942 *    WAIT FOR THE START CHARACTER
                      0943 *
C39E CD DA C3         0944 SOHL   CALL   TAPIN
C3A1 D8               0945        RC     .      ERROR OR ESCAPE
C3A2 FE 01            0946        CPI    1      LOOK FOR SOH (HEX 01)
C3A4 DA 9E C3         0947        JC     SOHL   ZERO--STILL WAITING FOR A ONE
C3A7 C2 8E C3         0948        JNZ    RHEAD  NOT ZERO OR ONE--LK FOR ANOTHER 10 NULLS
                      0949 *
                      0950 *    NOW GET THE HEADER
```

```
                    0951 *
C3AA 21 1C C8       0952        LXI    H,THEAD  POINT TO BUFFER
C3AD 06 10          0953        MVI    B,HLEN  LENGTH TO READ
                    0954 *
    C3AF            0955 RHED1  EQU    $      RD A BLOCK INTO HL FOR B BYTES
C3AF 0E 00          0956        MVI    C,0    INIT THE CRC
    C3B1            0957 RHED2  EQU    $      LOOP HERE
C3B1 CD DA C3       0958        CALL   TAPIN  GET A BYTE
C3B4 D8             0959        RC
C3B5 77             0960        MOV    M,A    STORE IT
C3B6 23             0961        INX    H      INCREMENT ADDRESS
C3B7 CD E7 C3       0962        CALL   DOCRC  GO COMPUTE THE CRC
C3BA 05             0963        DCR    B      WHOLE HEADER YET?
C3BB C2 B1 C3       0964        JNZ    RHED2  DO ALL THE BYTES
                    0965 *
                    0966 *   THIS ROUTINE GETS THE NEXT BYTE AND COMPARES IT
                    0967 * TO THE VALUE IN REGISTER C.  THE FLAGS ARE SET ON
                    0968 * RETURN.
                    0969 *
C3BE CD DA C3       0970        CALL   TAPIN  GET CRC BYTE
C3C1 A9             0971        XRA    C      CLR CARRY AND SET ZERO IF MATCH, ELSE NON-ZERO
C3C2 C8             0972        RZ     .      CRC OK
C3C3 3A 11 C8       0973        LDA    IGNCR  GET CRC IGNORE FLAG
C3C6 3C             0974        INR    A      SET FLAGS SO THAT CRC ERRORS CAN BE IGNORED IF FF
C3C7 C9             0975        RET
                    0976 *
                    0977 *    THIS ROUTINE GETS THE NEXT AVAILABLE BYTE FROM THE
                    0978 *  TAPE.  WHILE WAITING FOR THE BYTE THE KEYBOARD IS TESTED
                    0979 *  FOR AN ESC COMMAND.  IF RECEIVED THE TAPE LOAD IS
                    0980 *  TERMINATED AND A RETURN TO THE COMMAND MODE IS MADE.
                    0981 *
C3C8 DB FA          0982 STAT   IN     TAPPT  TAPE STATUS PORT
C3CA E6 40          0983        ANI    TDR
C3CC C0             0984        RNZ
C3CD CD 2E C0       0985        CALL   KSTAT  CHECK SOL KBD
C3D0 CA C8 C3       0986        JZ     STAT   NOTHING THERE YET
C3D3 E6 7F          0987        ANI    7FH    CLR PARITY 1ST
C3D5 C2 C8 C3       0988        JNZ    STAT   NOT A MODE (OR EVEN CTL-@)
C3D8 37             0989        STC    .      SET ERROR FLAG
C3D9 C9             0990        RET    .      AND RETURN
                    0991 *
                    0992 *
                    0993 *
C3DA CD C8 C3       0994 TAPIN  CALL   STAT   WAIT UNTIL A CHARACTER IS AVAILABLE
C3DD D8             0995        RC
                    0996 *
C3DE DB FA          0997 TREDY  IN     TAPPT  TAPE STATUS
C3E0 E6 18          0998        ANI    TFE+TOE  DATA ERROR?
C3E2 DB FB          0999        IN     TDATA  GET THE DATA
C3E4 C8             1000        RZ     .       IF NO ERRORS
C3E5 37             1001        STC    .      SET ERROR FLAG
C3E6 C9             1002        RET
                    1003 *
```

```
                    1004 *
    C3E7            1005 DOCRC  EQU    $      A COMMON CRC COMPUTATION ROUTINE
C3E7 91             1006        SUB    C
C3E8 4F             1007        MOV    C,A
C3E9 A9             1008        XRA    C
C3EA 2F             1009        CMA
C3EB 91             1010        SUB    C
C3EC 4F             1011        MOV    C,A
C3ED C9             1012        RET    .      ONE  BYTE NOW WRITTEN
                    1013 *
                    1014 *
    C3EE            1015 TON    EQU    $      HERE TO TURN A TAPE ON THEN DELAY
C3EE D3 FA          1016        OUT    TAPPT  GET TAPE MOVING, THEN DELAY
                    1017 *
C3F0 11 00 00       1018 DELAY  LXI    D,0
C3F3 1B             1019 DLOP1  DCX    D
C3F4 7A             1020        MOV    A,D
C3F5 B3             1021        ORA    E
C3F6 C2 F3 C3       1022        JNZ    DLOP1
C3F9 05             1023        DCR    B
C3FA C2 F0 C3       1024        JNZ    DELAY
C3FD C9             1025        RET
                    1026 *
                    1027 *
                    1028 * **************END OF PROGRAM**************
                    1029 *
                    1030 *
                    1031 *
                    1032 *
                    1033 *    S Y S T E M    E Q U A T E S
                    1034 *
                    1035 *
                    1036 *          VDM PARAMETERS
                    1037 *
    CC00            1038 VDMEM  EQU    0CC00H  VDM SCREEN MEMORY
                    1039 *
                    1040 *
                    1041 *          KEYBOARD SPECIAL KEY ASSIGNMENTS
                    1042 *
                    1043 *  THESE DEFINITIONS ARE DESIGNED TO ALLOW
                    1044 *  COMPATABILITY WITH CUTER(TM).  THESE ARE THE
                    1045 *  SAME KEYS AS BELOW, EXCEPT THAT THE X'80' BIT
                    1046 *  (BIT 7) IS STRIPPED OFF.
                    1047 *
    009A            1048 DOWN   EQU    9AH    (CTL-Z)
    0097            1049 UP     EQU    97H    (CTL-W)
    0081            1050 LEFT   EQU    81H    (CTL-A)
    0093            1051 RIGHT  EQU    93H    (CTL-S)
    008B            1052 CLEAR  EQU    8BH    (CTL-K)
    008E            1053 HOME   EQU    8EH    (CTL-N)
    0080            1054 MODE   EQU    80H    (CTL-@)
    005F            1055 BACKS  EQU    5FH    BACKSPACE
    000A            1056 LF     EQU    10
    000D          1057 CR     EQU    13
    0020          1058 BLANK  EQU    ' '
    0020          1059 SPACE  EQU    BLANK
    0018          1060 CX     EQU    'X'-40H
    001B          1061 ESC    EQU    1BH
                  1062 *
                  1063 *            PORT ASSIGNMENTS
                  1064 *
    00FA          1065 STAPT  EQU    0FAH   STATUS PORT GENERAL
    00F8          1066 SERST  EQU    0F8H   SERIAL STATUS PORT
    00F9          1067 SDATA  EQU    0F9H   SERIAL DATA
    00FD          1068 PDATA  EQU    0FDH   PARALLEL DATA
    00FC          1069 KDATA  EQU    0FCH   KEYBOARD DATA
    00FE          1070 DSTAT  EQU    0FEH   VDM CONTROL PORT
    00FA          1071 TAPPT  EQU    0FAH   TAPE STATUS PORT
    00FB          1072 TDATA  EQU    0FBH   TAPE DATA PORT
    00FF          1073 SENSE  EQU    0FFH   SENSE SWITCHES
                  1074 *
                  1075 *
                  1076 *
                  1077 *            BIT ASSIGNMENT MASKS
                  1078 *
    0001          1079 SCD    EQU    1      SERIAL CARRIER DETECT
    0002          1080 SDSR   EQU    2      SERIAL DATA SET READY
    0004          1081 SPE    EQU    4      SERIAL PARITY ERROR
    0008          1082 SFE    EQU    8      SERIAL FRAMING ERROR
    0010          1083 SOE    EQU    16     SERIAL OVERRUN ERROR
    0020          1084 SCTS   EQU    32     SERIAL CLEAR TO SEND
    0040          1085 SDR    EQU    64     SERIAL DATA READY
    0080          1086 STBE   EQU    128    SERIAL TRANSMITTER BUFFER EMPTY
                  1087 *
    0001          1088 KDR    EQU    1      KEYBOARD DATA READY
    0002          1089 PDR    EQU    2      PARALLEL DATA READY
    0004          1090 PXDR   EQU    4      PARALLEL DEVICE READY
    0008          1091 TFE    EQU    8      TAPE FRAMING ERROR
    0010          1092 TOE    EQU    16     TAPE OVERFLOW ERROR
    0040          1093 TDR    EQU    64     TAPE DATA READY
    0080          1094 TTBE   EQU    128    TAPE TRANSMITTER BUFFER EMPTY
                  1095 *
    0001          1096 SOK    EQU    1      SCROLL OK FLAG
                  1097 *
    0080          1098 TAPE1  EQU    80H    1=TURN TAPE ONE ON
    0040          1099 TAPE2  EQU    40H    1=TURN TAPE TWO ON
                  1100 *
                  1101 *
                  1102 *
                  1103 *
                  1104 *          S Y S T E M   G L O B A L   A R E A
                  1105 *
C800              1106        ORG    START+0800H   RAM STARTS JUST AFTER ROM
                  1107 *
    C800          1108 SYSRAM EQU    $      START OF SYSTEM RAM
    CBFF          1109 SYSTP  EQU    SYSRAM+3FFH  STACK WORKS FM TOP DOWN
```

```
                  1110 *
                  1111 *
                  1112 *  PARAMETERS STORED IN RAM
                  1113 *
                  1114 * THE FOLLOWING RAM AREA IS DEFINED AS FOLLOWS
                  1115 * PRIMARILY TO MAINTAIN COMPATABILITY BETWEEN
                  1116 * CONSOL///SOLOS///CUTER.
                  1117 *
                  1118 *
C800              1119 UIPRT  DS     2      USER DEFINED INPUT RTN IF NON ZERO
C802              1120 UOPRT  DS     2      USER DEFINED OUTPUT RTN IF NON ZERO
C804              1121 DFLTS  DS     2      DEFAULT PSUEDO I/O PORTS (ALWAYS ZERO IN SOLOS)
C806              1122 IPORT  DS     1      CRNT INPUT PSUEDO PORT
C807              1123 OPORT  DS     1      CRNT OUTPUT PSUEDO PORT
C808              1124 NCHAR  DS     1      CURRENT CHARACTER POSITION
C809              1125 LINE   DS     1      CURRENT LINE POSITION
C80A              1126 BOT    DS     1      BEGINNING OF TEXT DISPLACEMENT
C80B              1127 SPEED  DS     1      SPEED CONTROL BYTE
C80C              1128 ESCFL  DS     1      FOR COMPATABILITY W/ SOLOS/CUTER
C80D              1129 TSPD   DS     1      CURRENT TAPE SPEED---00=FAST, 20H=SLOW
C80E              1130 INPTR  DS     2      FOR COMPATABILITY W/ CUTER
C810              1131 NUCNT  DS     1      FOR COMPATABILITY W/ SOLOS/CUTER
C811              1132 IGNCR  DS     1      FF=IGNORE CRC ERRORS, ELSE=NORMAL
                  1133 *
C812              1134        DS     10     ROOM FOR FUTURE EXPANSION
                  1135 *
                  1136 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                  1137 *    T H I S   I S   T H E   H E A D E R   L A Y O U T    *
                  1138 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                  1139 *
C81C              1140 THEAD  DS     5      NAME
C821              1141        DS     1      THIS BYTE MUST BE ZERO
C822              1142 HTYPE  DS     1      TYPE
C823              1143 BLOCK  DS     2      BLOCK SIZE
C825              1144 LOADR  DS     2      LOAD ADDRESS
C827              1145 XEQAD  DS     2      AUTO EXECUTE ADDRESS
C829              1146 HSPR   DS     3      SPARES
                  1147 *
    0010          1148 HLEN   EQU    $-THEAD  LENGTH OF HEADER
    0007          1149 BLKOF  EQU    BLOCK-THEAD  OFFSET TO BLOCK SIZE
C82C              1150 DHEAD  DS     HLEN   A DUMMY HDR FOR COMPARES WHILE RD'ING
                  1151 *
                  1152 *
C83C              1153 CUTAB  DS     6*4    FOR COMPATABILITY W/ SOLOS/CUTER
                  1154 *
                  1155 *
C854              1156 FNUMF  DS     1      FOR CURRENT FILE OPERATIONS
C855              1157 FCBAS  DS     7      1ST FILE CONTROL BLOCK
C85C              1158 FCBA2  DS     7      2ND FILE CONTROL BLOCK
C863              1159 FBUF1  DS     2*256  SYSTEM FILE BUFFER BASE
CA63              1160        DS     81     THIS IS AN AREA USED BY CUTER
    CAB4          1161 USARE  EQU    $      START OF USER AREA
                  1162 *
```

```
 *   REMEMBER THAT THE STACK WORKS ITS WAY DOWN FROM
 *   THE END OF THIS 1K RAM AREA.
 *
 *
 * -*-

ADOUT   C2D7     AINP    C022      AOUT    C01C      BACKS   005F
BLANK   0020     BLKOF   0007      BLOCK   C823      BOT     C80A
BOUT    C2EA     CHAR    C08A      CHPCK   C05E      CLEAR   008B
CLERA   C154     CLIN1   C0F0      CLINE   C0EA      COMN1   C160
COMND   C169     COMTA   C1DE      CONT    C193      COPRC   C199
CR      000D     CREM    C12C      CRLF    C22B      CUR     C0C7
CURSC   C0C5     CUTAB   C83C      CX      0018      DCRC2   C38B
DCRCT   C380     DELAY   C3F0      DFLTS   C804      DHEAD   C82C
DISP1   C1BC     DISPT   C1B8      DLOOP   C2B7      DLOP1   C3F3
DLP1    C2C2     DOCRC   C3E7      DOWN    009A      DSTAT   00FE
DUMP    C2AE     ENLO1   C31C      ENLOP   C30B      ENTER   C307
ERAS1   C0D1     ERAS3   C0E4      ERR1    C33C      ERR2    C33D
ERROT   C160     ESC     001B      ESCFL   C80C      EXEC    C334
EXEC1   C337     FBUF1   C863      FCBA2   C85C      FCBAS   C855
FCLOS   C00A     FDCOM   C1C5      FNUMF   C854      FOPEN   C007
GCLIN   C178     GOBAC   C06B      GOBAS   C1BF      GOBK    C072
HBOUT   C2DC     HCONV   C26A      HCOV1   C27A      HEOU1   C2F8
HEOUT   C2EF     HLEN    0010      HOME    008E      HSPR    C829
HTYPE   C822     IGNCR   C811      INIT    C001      INPTR   C80E
IOPRC   C026     IPORT   C806      ITAB    C21B      KDATA   00FC
KDR     0001     KSTAT   C02E      LEFT    0081      LF      000A
LINE    C809     LOADR   C825      MODE    0080      NCHAR   C808
NCOM    C1D7     NEXT    C076      NUCNT   C810      OCHAR   C08E
OK      C0B7     OPORT   C807      OTAB    C213      OUTH    C303
OUTPR   C03B     PBACK   C134      PCR     C13D      PCUR    C105
PDATA   00FD     PDOWN   C0C1      PDR     0002      PERSE   C0CB
PHOME   C0DB     PLEFT   C101      PLF     C143      PRIT    C10B
PROMP   C223     PSCAN   C235      PTAP1   C34E      PUP     C0FA
PXDR    0004     RDBLK   C013      RDBYT   C00D      RETRN   C004
RHEA1   C390     RHEAD   C38E      RHED1   C3AF      RHED2   C3B1
RIGHT   0093     RTAP    C35E      RTAP2   C35E      RTAPE   C347
RTOFF   C375     SBLK    C23D      SBLK1   C23F      SCD     0001
SCHR    C24B     SCHR1   C24D      SCONV   C257      SCROL   C0A2
SCTS    0020     SDATA   00F9      SDR     0040      SDROT   C04A
SDSR    0002     SENSE   00FF      SERST   00F8      SFE     0008
SHE1    C260     SHEX    C25D      SINP    C01F      SOE     0010
SOHL    C39E     SOK     0001      SOUT    C019      SPACE   0020
SPE     0004     SPEED   C80B      SROL    C0A6      SSTAT   C042
STAPT   00FA     START   C000      STAT    C3C8      STBE    0080
STRTA   C14F     SYSRA   C800      SYSTP   CBFF      TAPE1   0080
TAPE2   0040     TAPIN   C3DA      TAPPT   00FA      TBL     C1F7
TDATA   00FB     TDR     0040      TERM    C284      TERM1   C284
TERR    C378     TFE     0008      THEAD   C81C      TIN     C29C
TLOAD   C342     TOE     0010      TOFF    C370      TON     C3EE
TOUT    C299     TREDY   C3DE      TSPD    C80D      TSRCH   C078
TTBE    0080     UIPRT   C800      UOPRT   C802      UP      0097
USARE   CAB4     VDAD    C119      VDAD2   C116      VDADD   C112
```

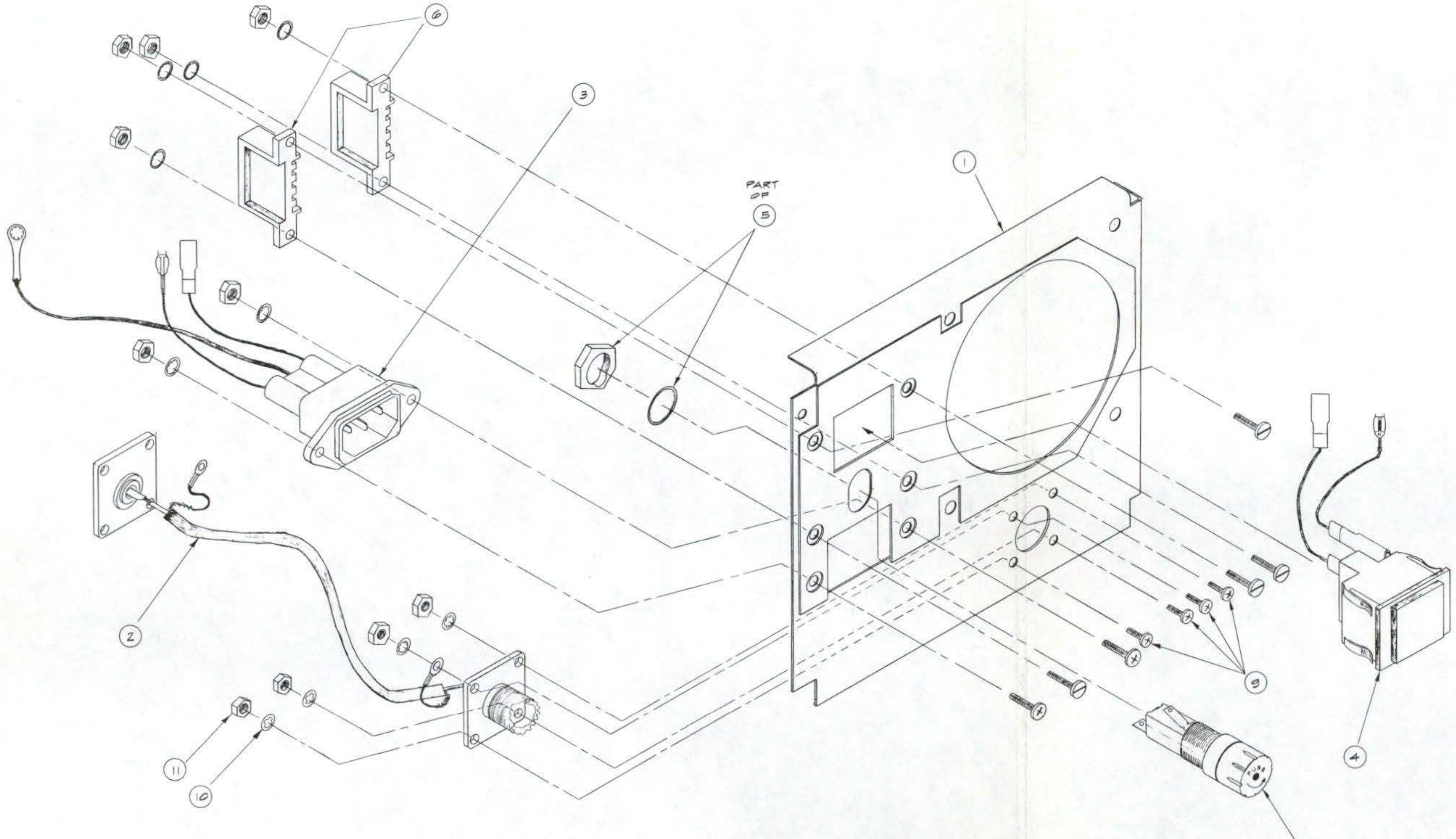
```
VDMEM   CC00     VDMOT   C05B      VDMOX   C054      WRBLK   C016
WRBYT   C010     XEQAD   C827
```

X DRAWINGS

NOTES:

UNLESS OTHERWISE SPECIFIED:

1. ALL SCREWS PART NO. ⑦
2. ALL LOCKWASHERS PART NO. ⑫
3. ALL NUTS PART NO. ⑧



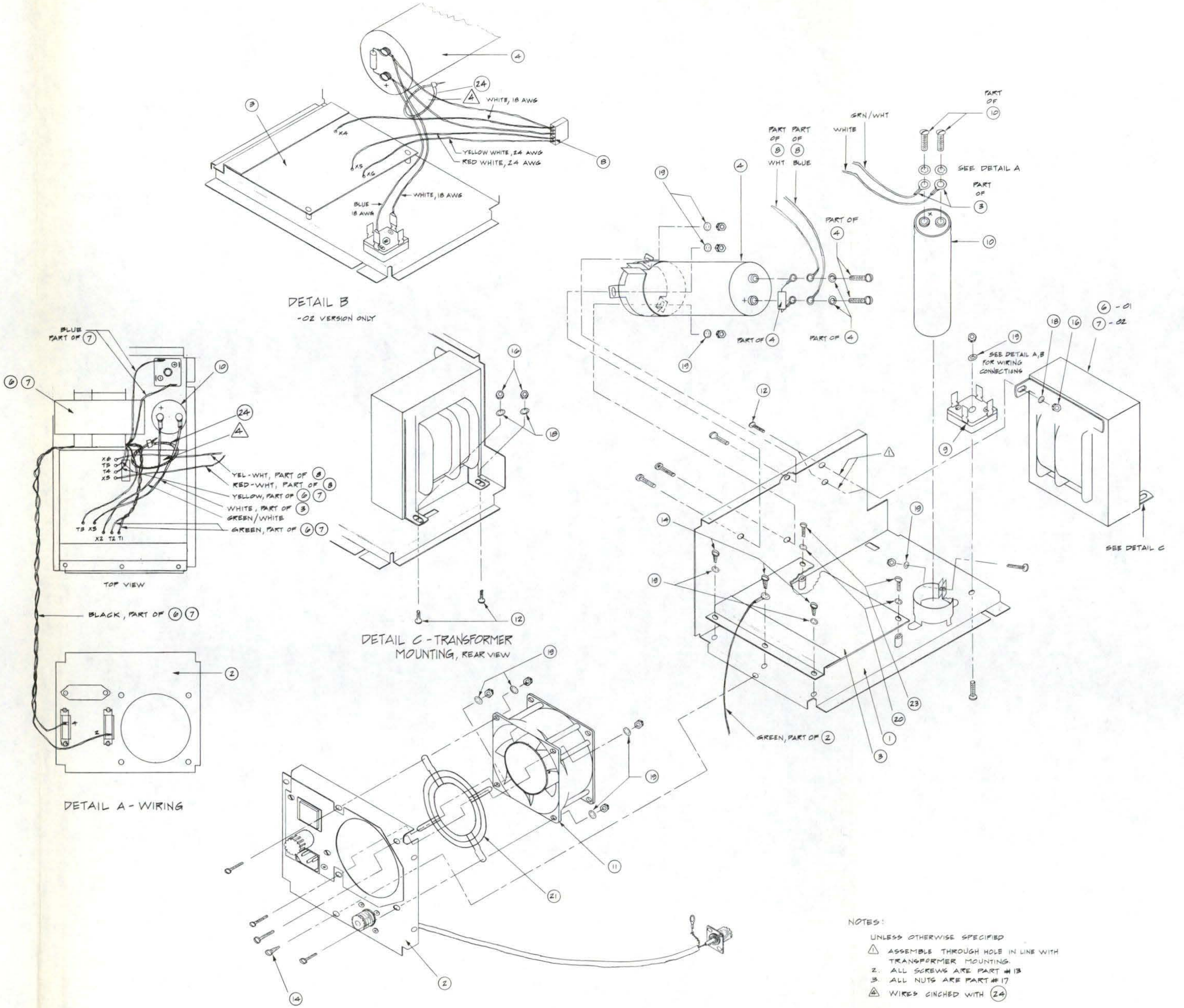
REF. DES.	PART OR DWG. NO.	PART DESCRIPTION	-01		
12		LOCKWASHER, INT. TOOTH #6	6		
11		NUT, HEX, 4-40	4		
10		LOCKWASHER, INT. TOOTH #4	3		
9		SCREW, MACHINE, 4-40 × 5/16" PAN HEAD	4		
8		NUT, HEX, 6-32	6		
7		SCREW, MACHINE, 6-32 × 1/2", PAN HEAD	6		
6		COMMONING BLOCKS	2		
5		FUSE POST, W/ NUT & LOCKWASHER	1		
4	105022	ASSEMBLY, AC SWITCH	1		
3	105018	ASSEMBLY, AC CONNECTOR	1		
2	105016	ASSEMBLY, VIDEO CABLE	1		
1	105031	SHEET METAL, FAN CLOSURE	1		

ASSEMBLY, FAN CLOSURE

SCALE: —	APPROVED BY: R. MARSH	DRAWN BY LITO
DATE: 01-5-77		REVISED

PROCESSOR TECHNOLOGY

DRAWING NUMBER
105014



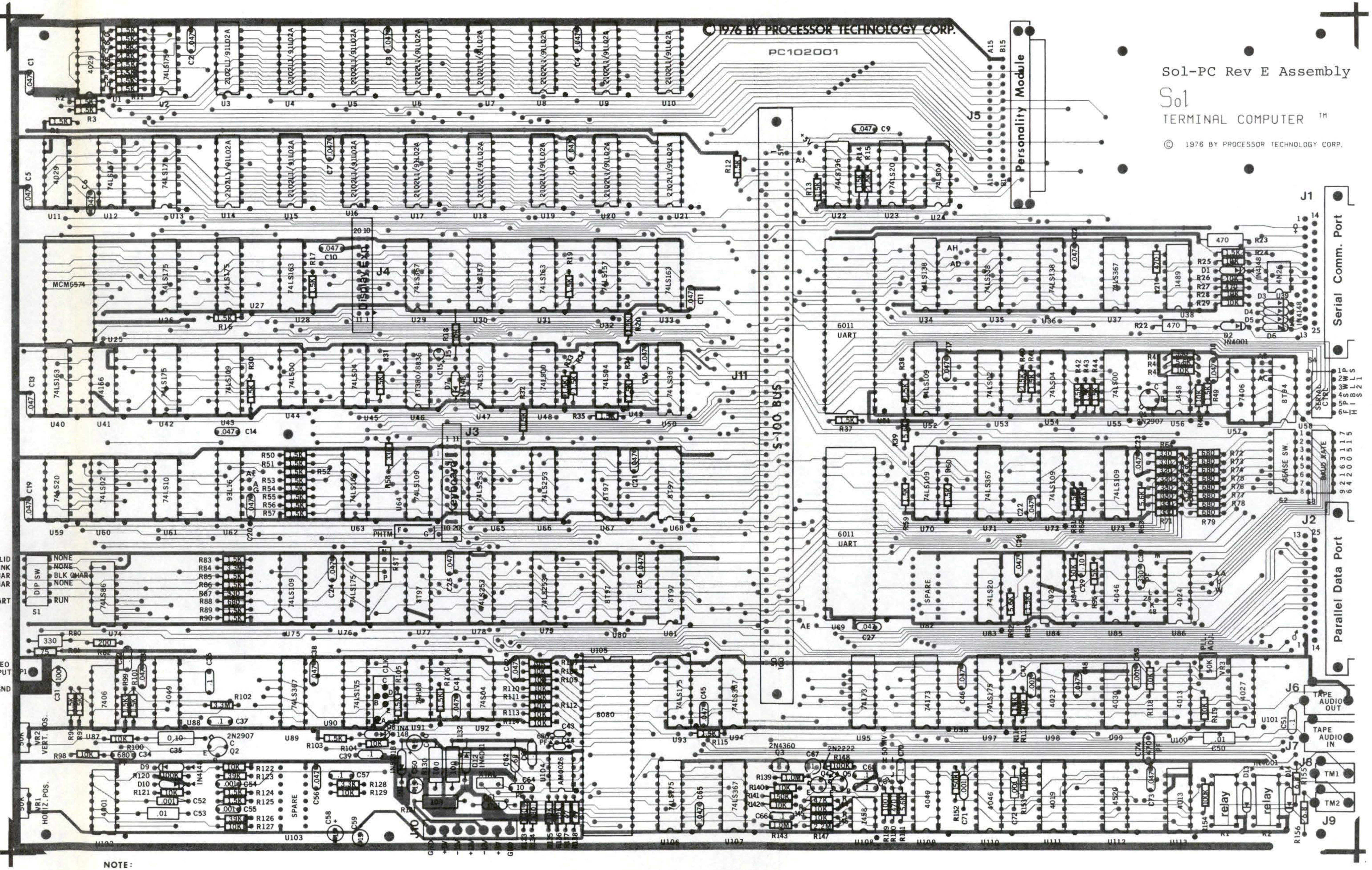
REF DES.	PART OR DWG. NO.	PART DESCRIPTION	-01	-02
24		TIE, CABLE, PLASTIC	1	2
23		SCREW, 4-40 × 3/16 PAN HEAD	2	2
22	105002	SCHEMATIC, Sol-PWR, Sol-20	—	REF
21	105003	SCHEMATIC, Sol-PWR, Sol-10	REF	—
20		LOCKWASHER, #4 INT. TOOTH	2	2
19		LOCKWASHER, #6 INT. TOOTH		
18		LOCKWASHER, #8 INT. TOOTH	3	3
17		NUT, 6-32 HEX		
16		NUT, 8-32 HEX	3	3
15		SCREW, 4-40 × 5/16" PAN HEAD	2	2
14		SCREW, #6 × 5/16 SELF TAPPING PAN HEAD	4	4
13		SCREW, 6-32 × 1/2" PAN HEAD		
12		SCREW, 8-32 × 1/2" PAN HEAD	3	3
11		FAN	0	1
10		CAPACITOR, ALUM. 18,000 μF, 12V	1	1
9		RECTIFIER BRIDGE	0	1
8	105007	CABLE ASSEMBLY, BFD DC PWR	0	1
7	103028	TRANSFORMER, Sol-20	0	1
6	103026	TRANSFORMER, Sol-10	1	0
5		FINGER GUARD, FAN	0	1
4	105005	CAPACITOR ASSEMBLY, 54000 μF	0	1
3	105008	REGULATOR ASSEMBLY	1	1
2	105014	FAN CLOSURE ASSEMBLY	1	1
1	105004	CHASSIS ASSEMBLY	1	1

NOTES:

UNLESS OTHERWISE SPECIFIED

1. ASSEMBLE THROUGH HOLE IN LINE WITH TRANSFORMER MOUNTING.
2. ALL SCREWS ARE PART #13
3. ALL NUTS ARE PART #17
4. WIRES CINCHED WITH (24)

ASSEMBLY, POWER SUPPLY, Sol-PWR		
SCALE: ——	APPROVED BY:	DRAWN BY LITO
DATE: 12-29-76	R. MARSH	REVISED
PROCESSOR TECHNOLOGY CORP.		
(Sol 10 & Sol 20) 105001-01 & 105001-02		DRAWING NUMBER 105001



Sol-PC Rev E Assembly

Sol

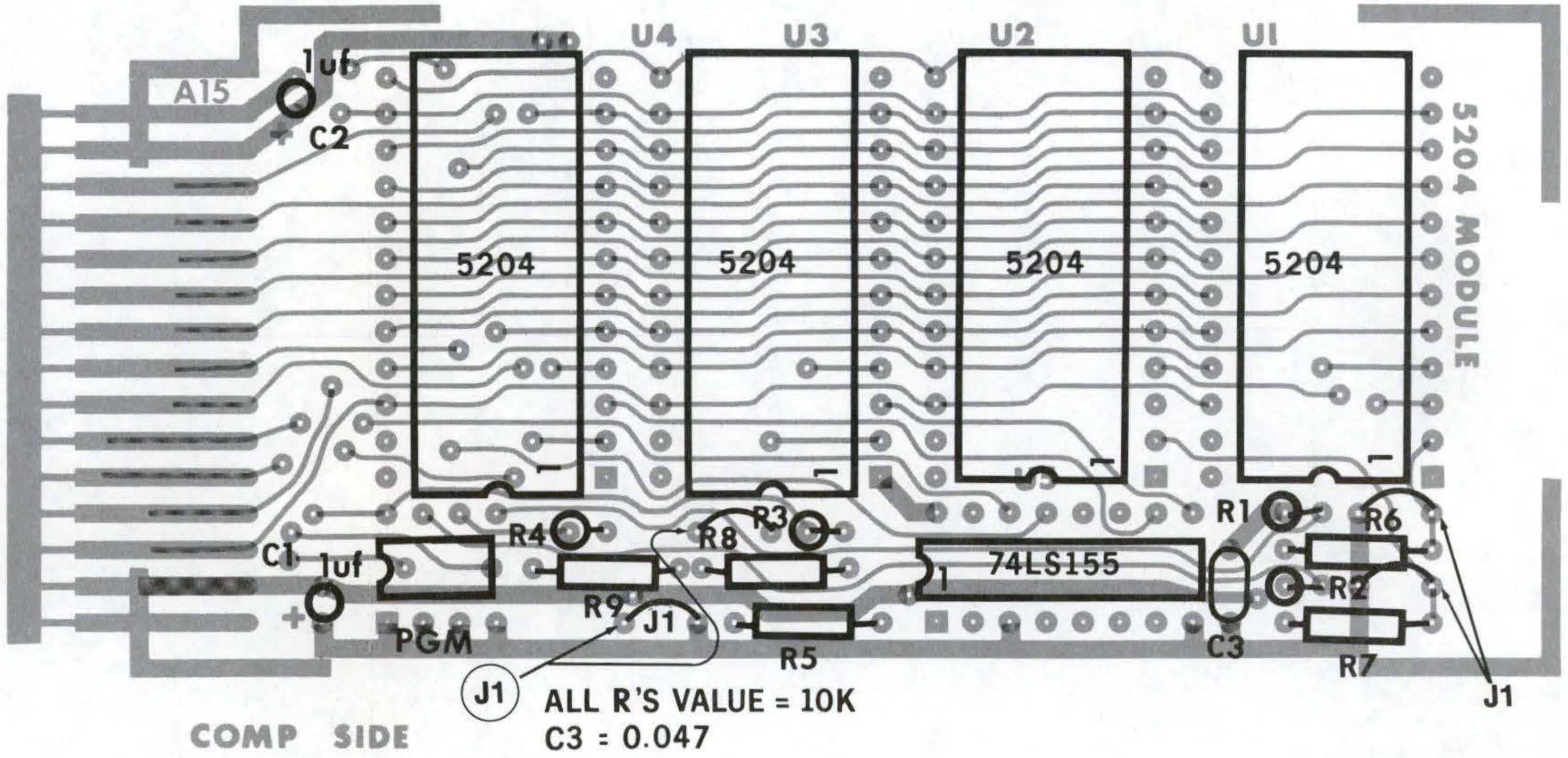
TERMINAL COMPUTER TM

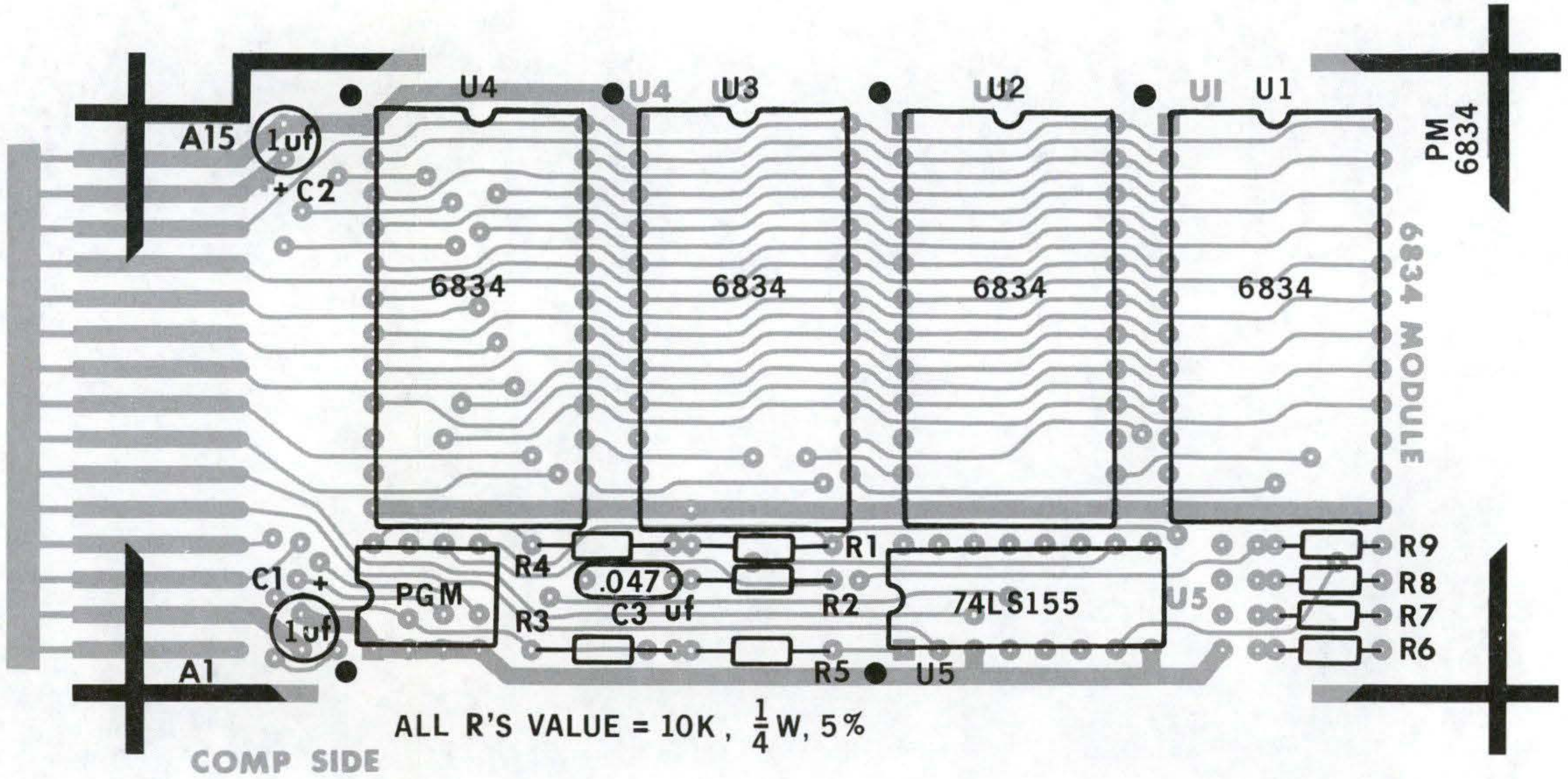
© 1976 BY PROCESSOR TECHNOLOGY CORP.

NOTE:

R91 is not used

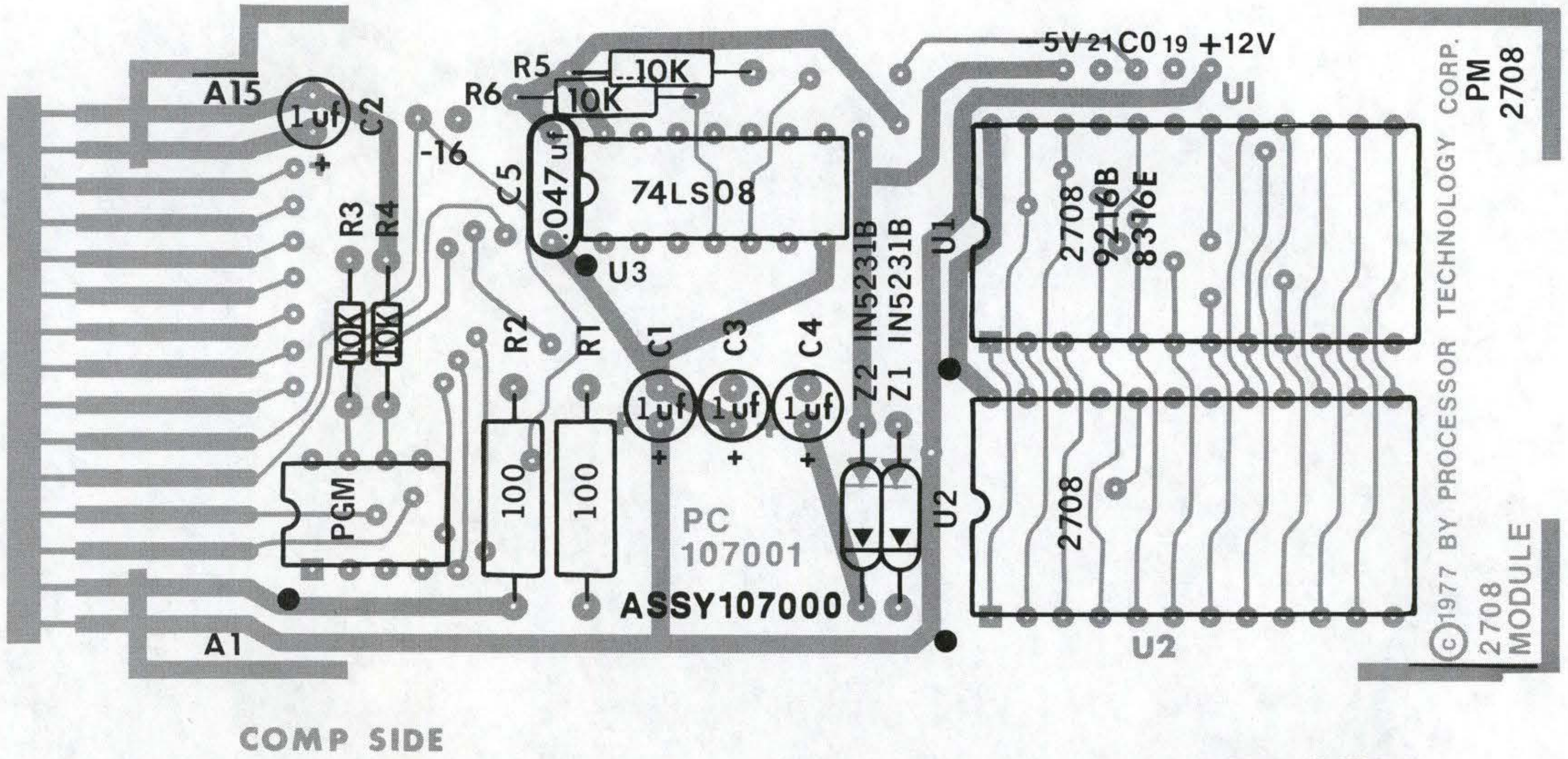
Note: jumpers labelled "J1" must be installed on PM5204 modules using National 5204Q PROMS.





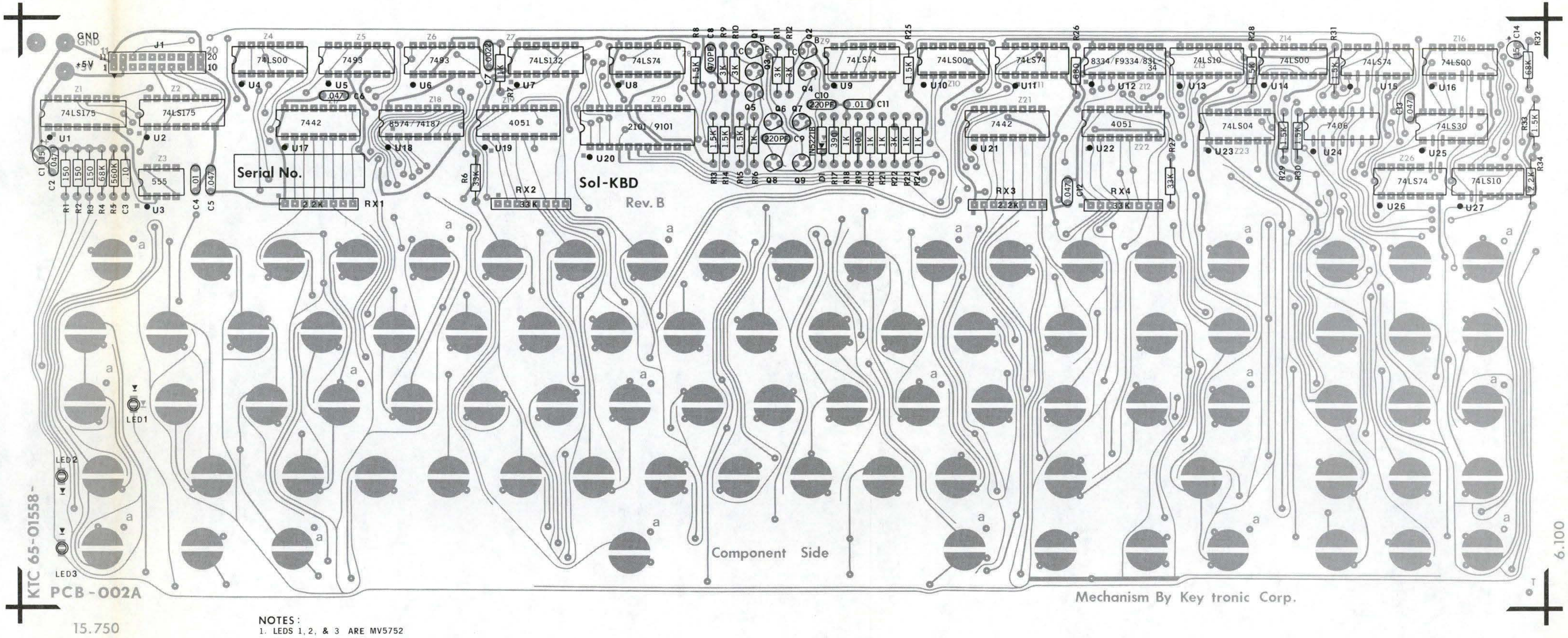
PM 6834

ASSEMBLY



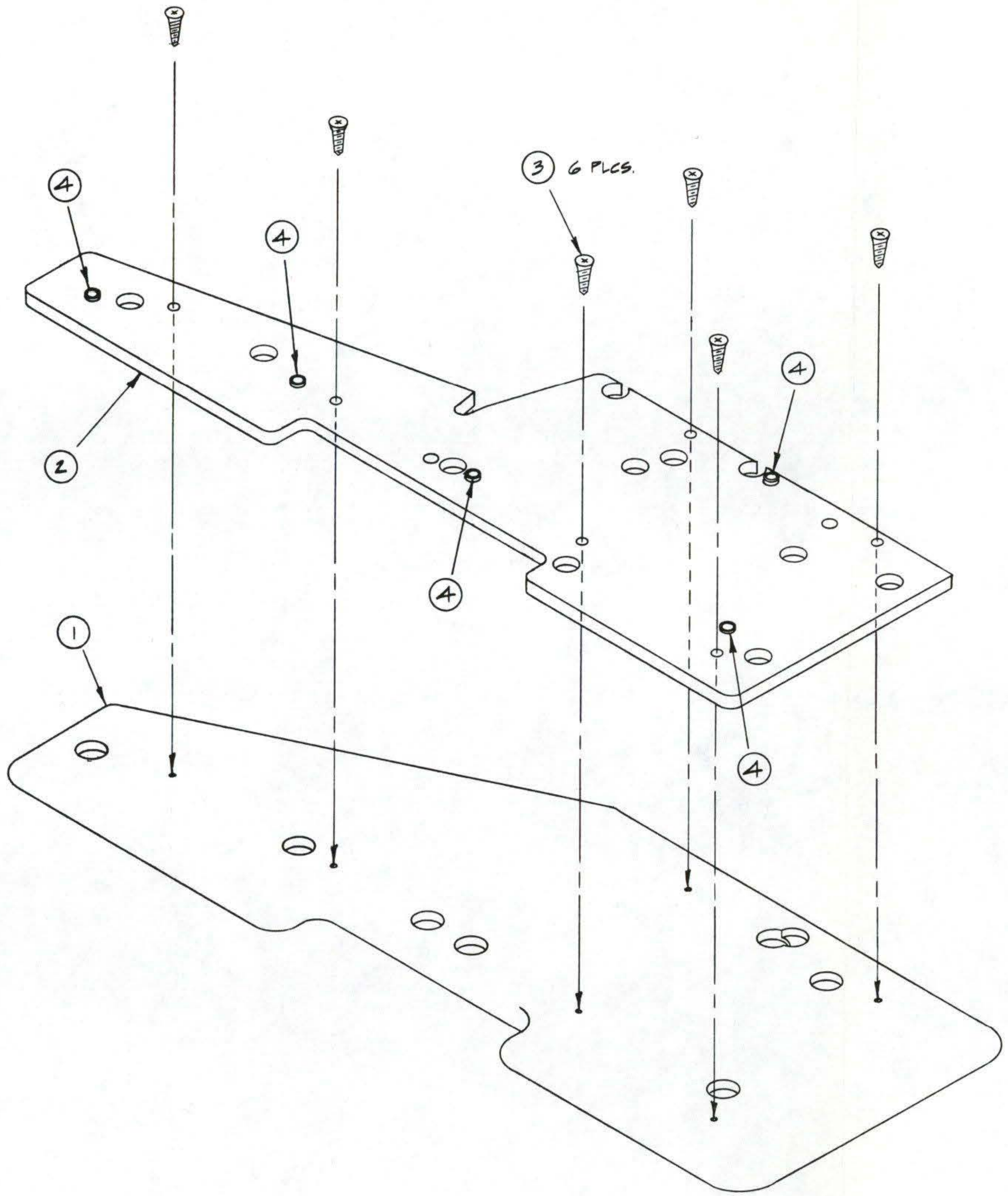
PM 2708

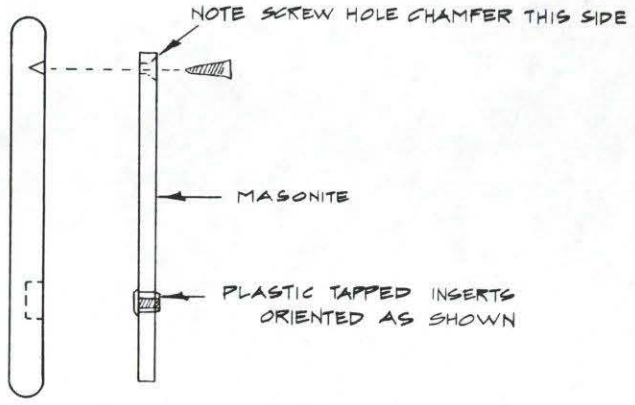
ASSEMBLY



NOTES:
1. LEDS 1, 2, & 3 ARE MV5752
2. Q1, Q2 & Q9 ARE 2N4274, Q3 — Q8 ARE 2N3640

Sol-KEYBOARD
ASSEMBLY





DETAIL - ORIENTATION OF
MASONITE CROSS SECTION

REF. DES.	PART OR DWG. NO.	PART DESCRIPTION	QTY
4		NUT, PLASTIC INSERT, 6-32	5
3		SCREW, WOOD, FLAT HEAD #6 x 5/8"	6
2	101011	SPACER, LEFT SIDE PANEL	1
1	101009	SLAB, LEFT SIDE PANEL	1

NEXT ASSY.	USED ON
101000	501

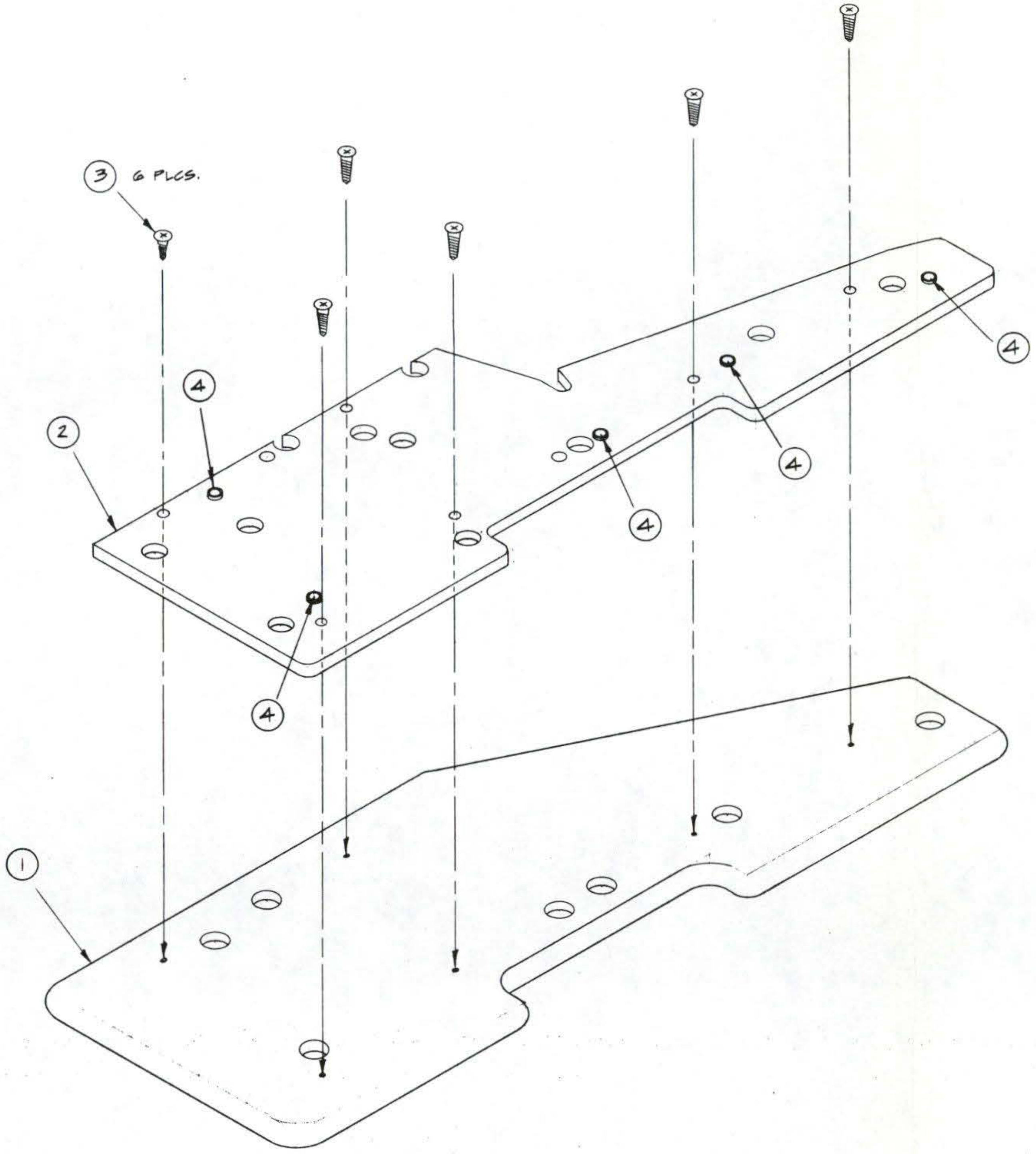
Processor Technology Corporation
6200 Hollis Street
Emeryville, CA 94608

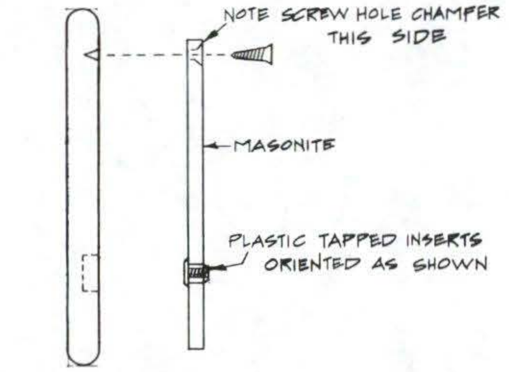
DRAWN: LITO
DATE: 1-4-77
CHECKED:
DATE:
ENGR:
DATE:
RELEASED:
DATE:

ASSEMBLY, SIDE PANEL, LEFT HAND

REV. A
DRAWING NO. 101007
SCALE: NONE
SHEET 1 OF 1
SIZE C

X-9




NOTE SCREW HOLE CHAMFER THIS SIDE
MASONITE
PLASTIC TAPPED INSERTS ORIENTED AS SHOWN

DETAIL - ORIENTATION OF MASONITE CROSS SECTION

REF. DES.	PART OR DWG. NO.	PART DESCRIPTION	QTY
4		NUT, PLASTIC INSERT, 6-32	5
3		SCREW, FLAT HEAD, WOOD, #6 x 5/8"	6
2	101012	SPACER, RIGHT SIDE	1
1	101010	SLAB, RIGHT SIDE	1

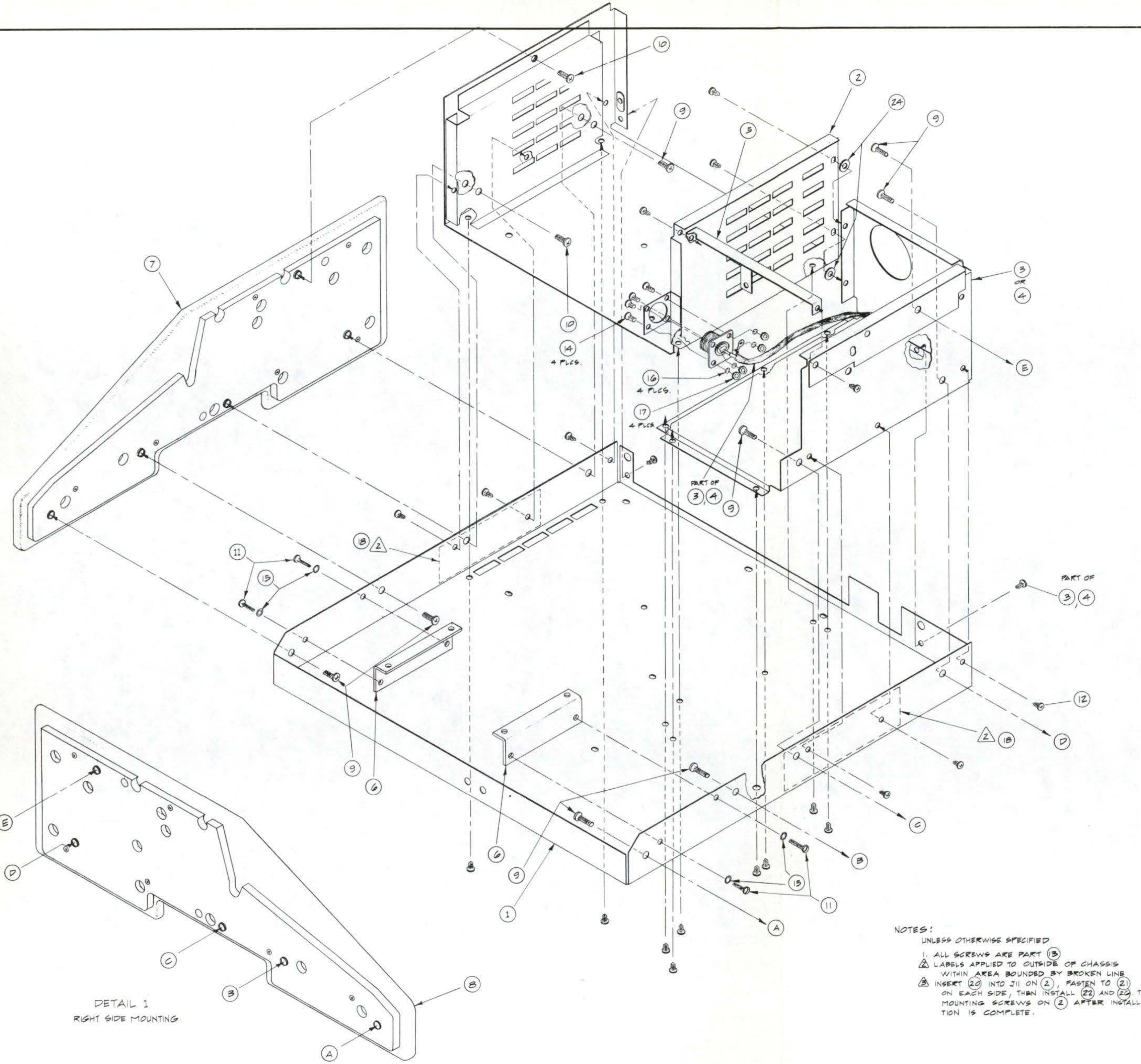
NEXT ASSY.	USED ON
101000	501

Processor Technology Corporation
6200 Hollis Street
Emeryville, CA 94608

DRAWN:	LITO	ENGR:	
DATE:	1-4-77	DATE:	
CHECKED:		RELEASED:	
DATE:		DATE:	

ASSEMBLY, SIDE PANEL, RIGHT HAND

REV.	DRAWING NO.	SCALE:	SHEET	SIZE
A	101008	NONE	1 OF 1	C



DETAIL 1
RIGHT SIDE MOUNTING

REF. DES	PART OR DWG. NO.	PART DESCRIPTION	-10	-20
27		SCREW, MACHINE, 4-40 × 5/16", PAN HEAD	2	2
26	101021	GUSSET, BACKPLANE, LEFT-HAND	0	1
25	102000	Sol PC CIRCUIT CARD	1	1
24		WASHER, FLAT, 1/2" O.D., 3/16 I.D., 1/16" THK	2	2
23		CARD GUIDE	–	10
22	101017	GUSSET, BACKPLANE, RIGHT HAND	–	1
21	101016	BRACKET, BACKPLANE RIGHT ANGLE	–	2
20	103000	BACK PLANE BOARD ASSEMBLY	–	1
19		SCREW, MACHINE, 4-40 × 5/8", PAN HEAD	–	6
18		LABEL, FINGER WELL, BLACK	2	2
17		NUT, #4-40, HEX	4	4
16		LOCKWASHER, #4, INTERNAL TOOTH	3	3
15		LOCKWASHER, #6, INTERNAL TOOTH	4	4
14		SCREW, MACHINE, 4-40 × 5/16" PAN HEAD	10	16
13		SCREW, SHEET METAL, #6 × 1/4" PAN HEAD	19	31
12		SCREW, SHEET METAL, #6 × 5/16" PAN HEAD	1	1
11		SCREW, MACHINE, 6-32 × 1/2" PAN HEAD	4	4
10		" " 8-32 × 1" " "	2	2
9		SCREW, MACHINE, 8-32 × 1/2" PAN HEAD	8	8
8	101008	SIDE ASSEMBLY, RIGHT-HAND	1	1
7	101007	SIDE ASSEMBLY, LEFT-HAND	1	1
6	101006	BRACKET, KEYBOARD SUPPORT	2	2
5	101005	BRACKET, CONNECTING	1	1
4	103001-02	POWER SUPPLY, Sol-20	–	1
3	103001-01	POWER SUPPLY, Sol-10	1	–
2	101004	EXPANSION SUBCHASSIS	1	1
1	101003	MAIN CHASSIS	1	1

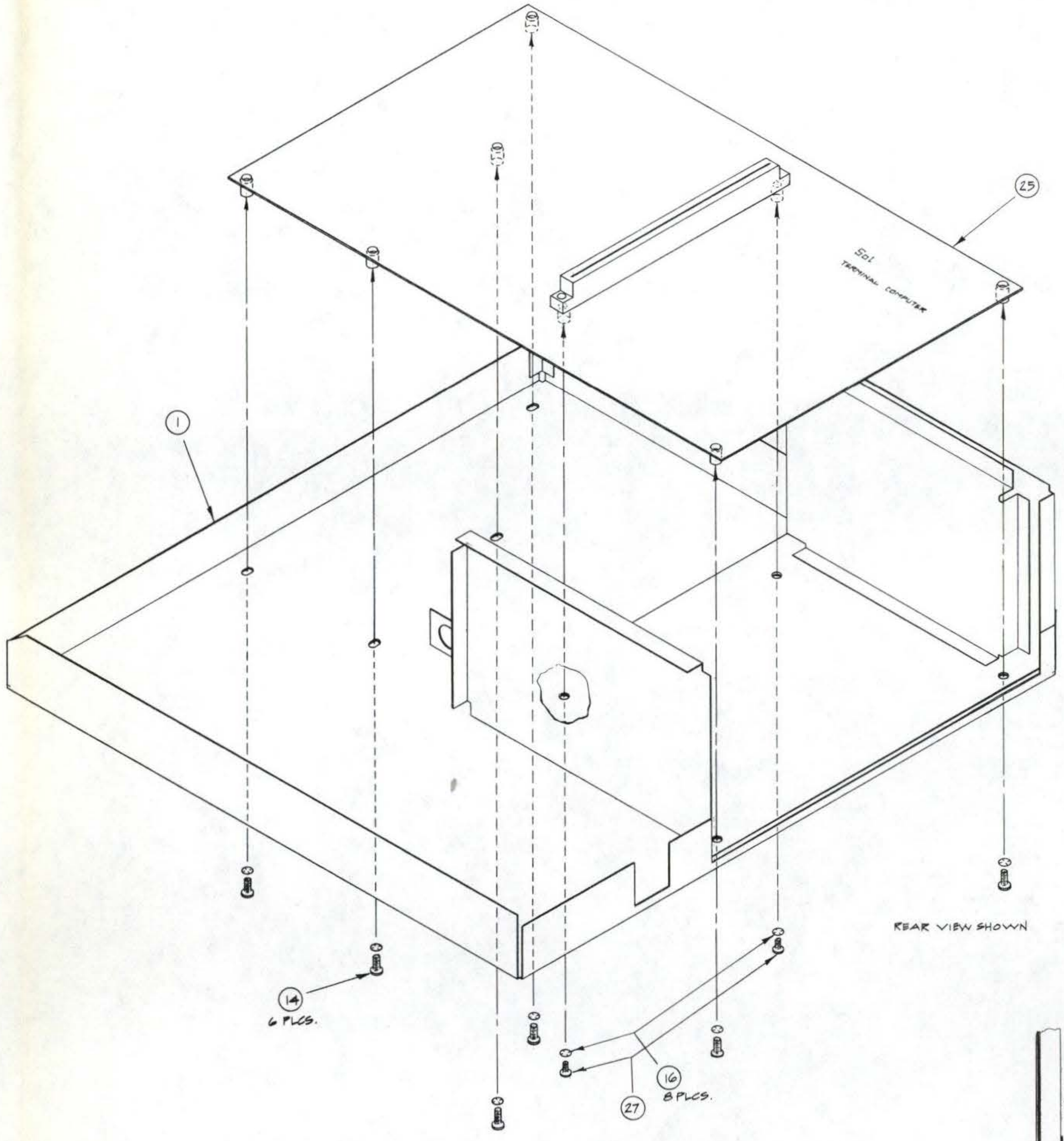
NOTES:

UNLESS OTHERWISE SPECIFIED

1. ALL SCREWS ARE PART (13)
2. LABELS APPLIED TO OUTSIDE OF CHASSIS WITHIN AREA BOUNDED BY BROKEN LINE
3. INSERT (20) INTO J11 ON (2), FASTEN TO (21) ON EACH SIDE, THEN INSTALL (22) AND (26). TIGHTEN MOUNTING SCREWS ON (2) AFTER INSTALLATION IS COMPLETE.

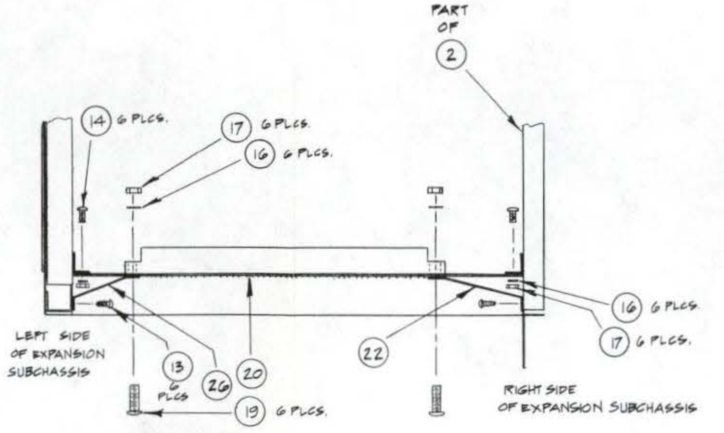
ASSEMBLY, Sol		
SCALE: ———	APPROVED BY: R. MARSH	DRAWN BY LITO
DATE: 01-17-77		REVISED
PROCESSOR TECHNOLOGY CORP.		
		DRAWING NUMBER 101000

X-11



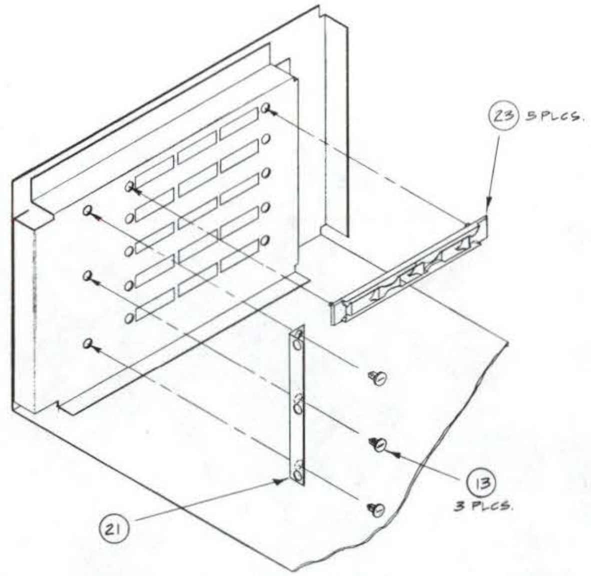
REAR VIEW SHOWN

DETAIL - Sol PC INSTALLATION



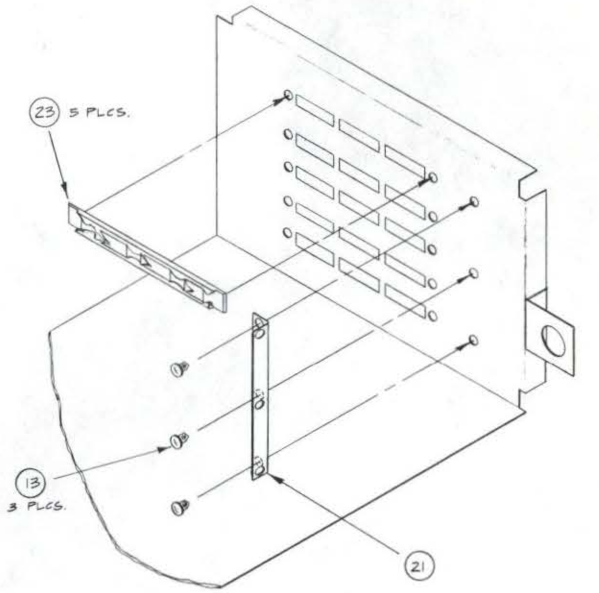
TOP VIEW

DETAIL - MOUNTING OF BACKPLANE BOARD
(Sol-20 ONLY)



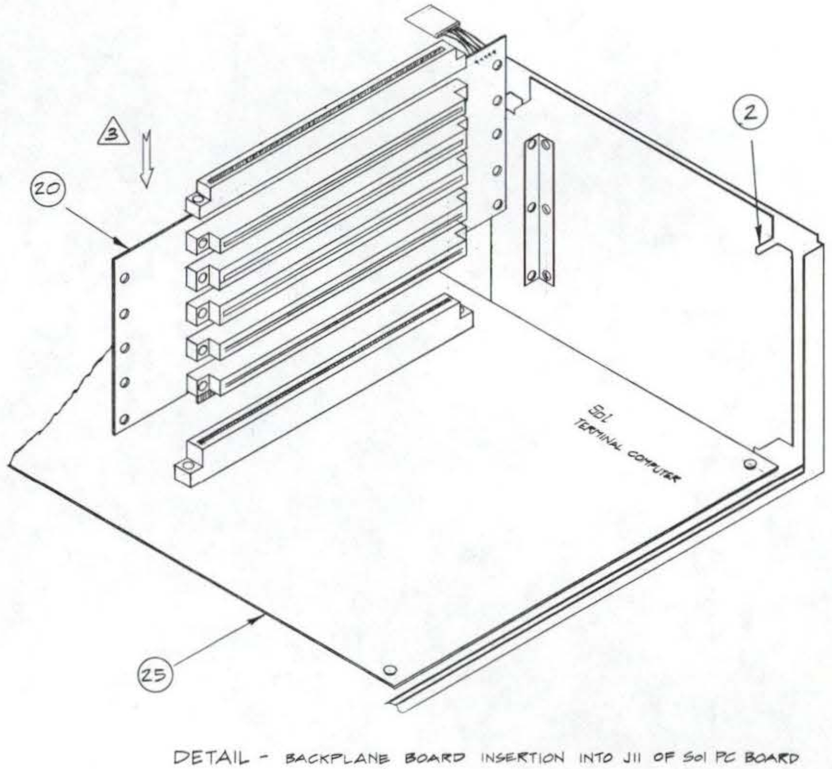
VIEW FROM FRONT RIGHT

DETAIL - LEFT SIDE EXPANSION SUBCHASSIS BRACKET AND CARD GUIDE MOUNTING.
(Sol-20 ONLY)



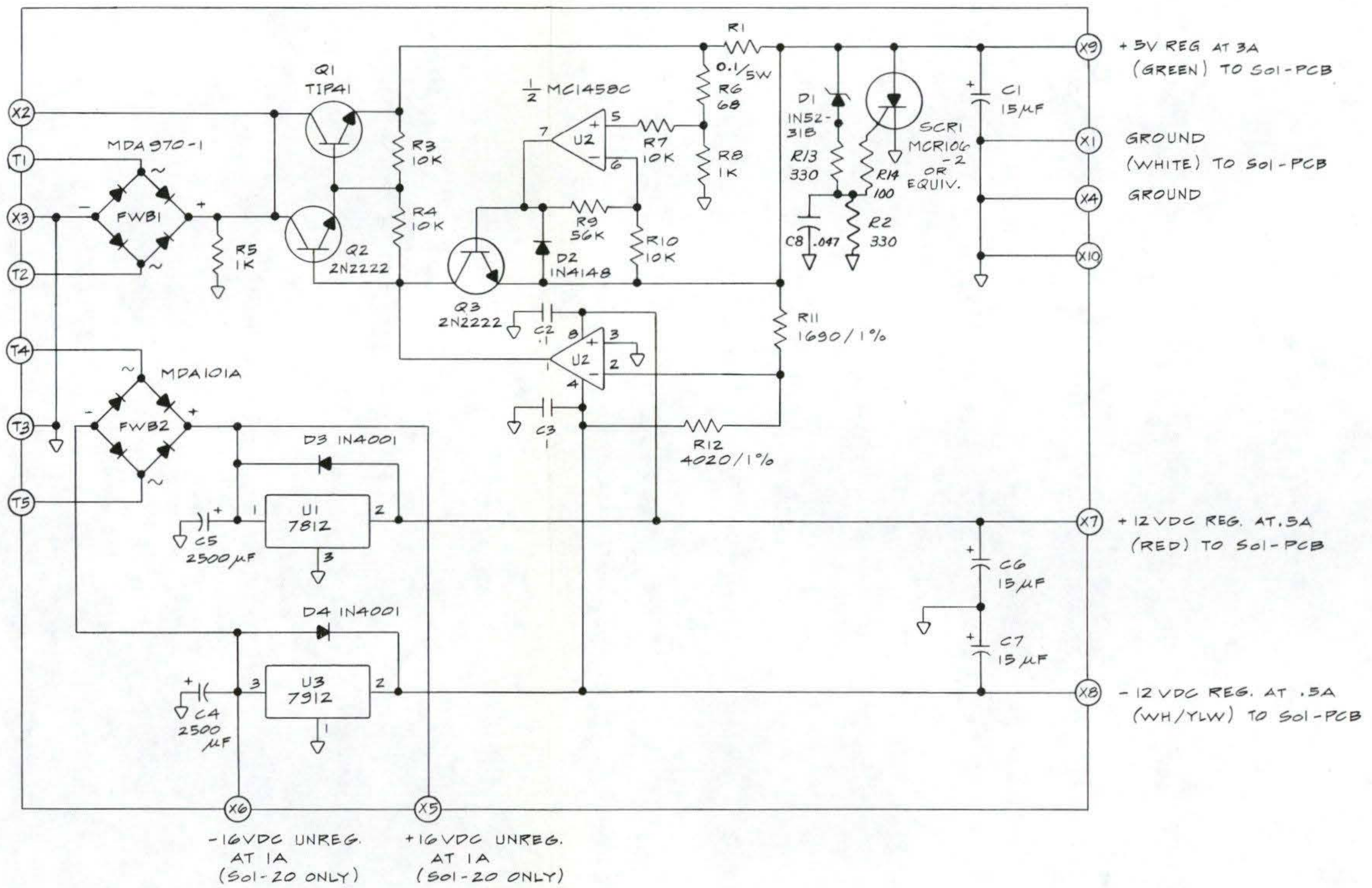
VIEW FROM FRONT LEFT

DETAIL - RIGHT SIDE EXPANSION SUBCHASSIS BRACKET AND CARD GUIDE MOUNTING.
(Sol-20 ONLY)



DETAIL - BACKPLANE BOARD INSERTION INTO J11 OF Sol PC BOARD
(Sol-20 ONLY)

ASSEMBLY, Sol		
SCALE: —	APPROVED BY: R. MARSH	DRAWN BY LITO
DATE: 01-20-77		REVISED
PROCESSOR TECHNOLOGY CORP.		
Rev. A	SHEET 2	DRAWING NUMBER 101000



REF. DES.	PART OR DWG. NO.	PART DESCRIPTION

NEXT ASSY.	USED ON
105008	Sol

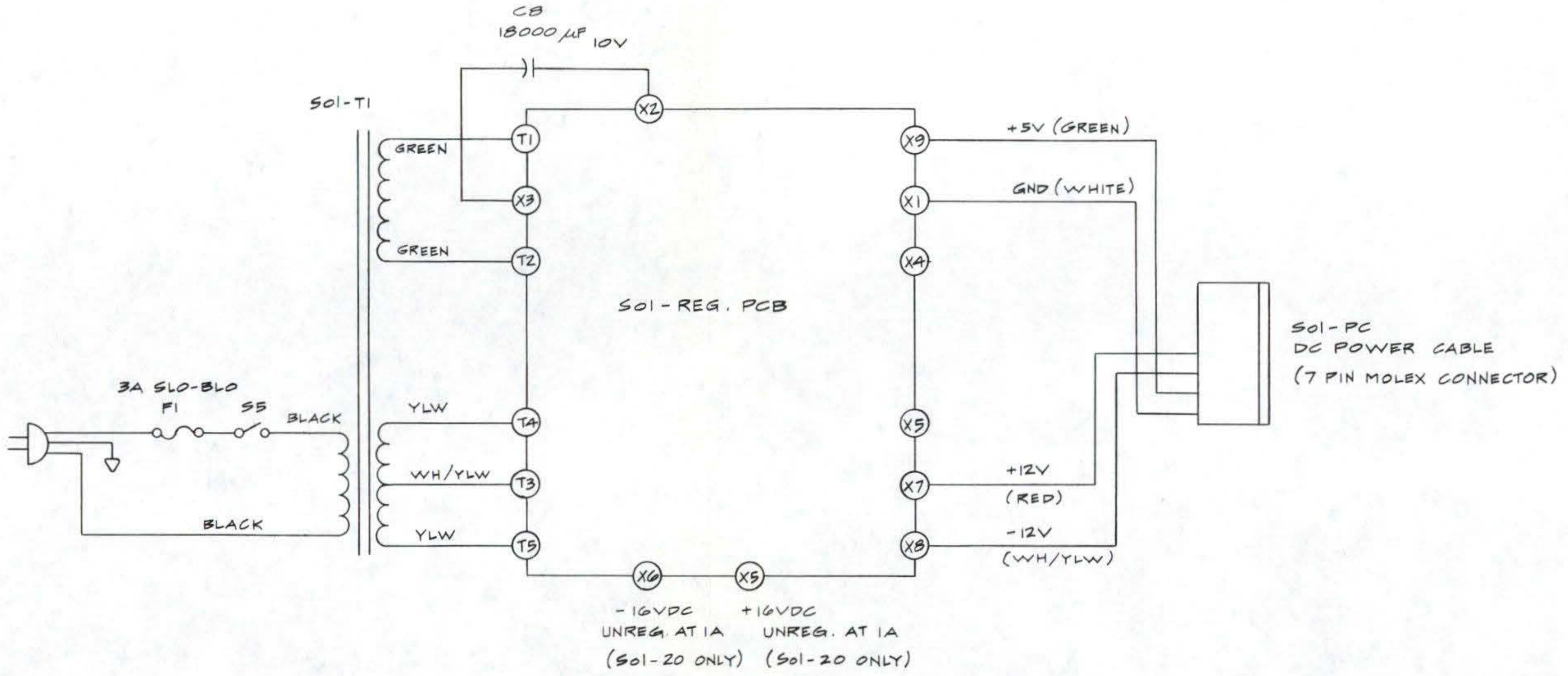
Processor Technology Corporation
6200 Hollis Street
Emeryville, CA 94608

DRAWN:	LITO	ENGR:	L. FELSENSTEIN
DATE:	11-12-76	DATE:	6-1-77
CHECKED:	EJ Lynch	RELEASED:	L. FELSENSTEIN
DATE:	6/1/77	DATE:	6-1-77

SCHEMATIC, Sol-REGULATOR PCB

REV.	DRAWING NO.	SCALE: NONE SHEET 1 OF 1	SIZE
B	105009		C

X-13

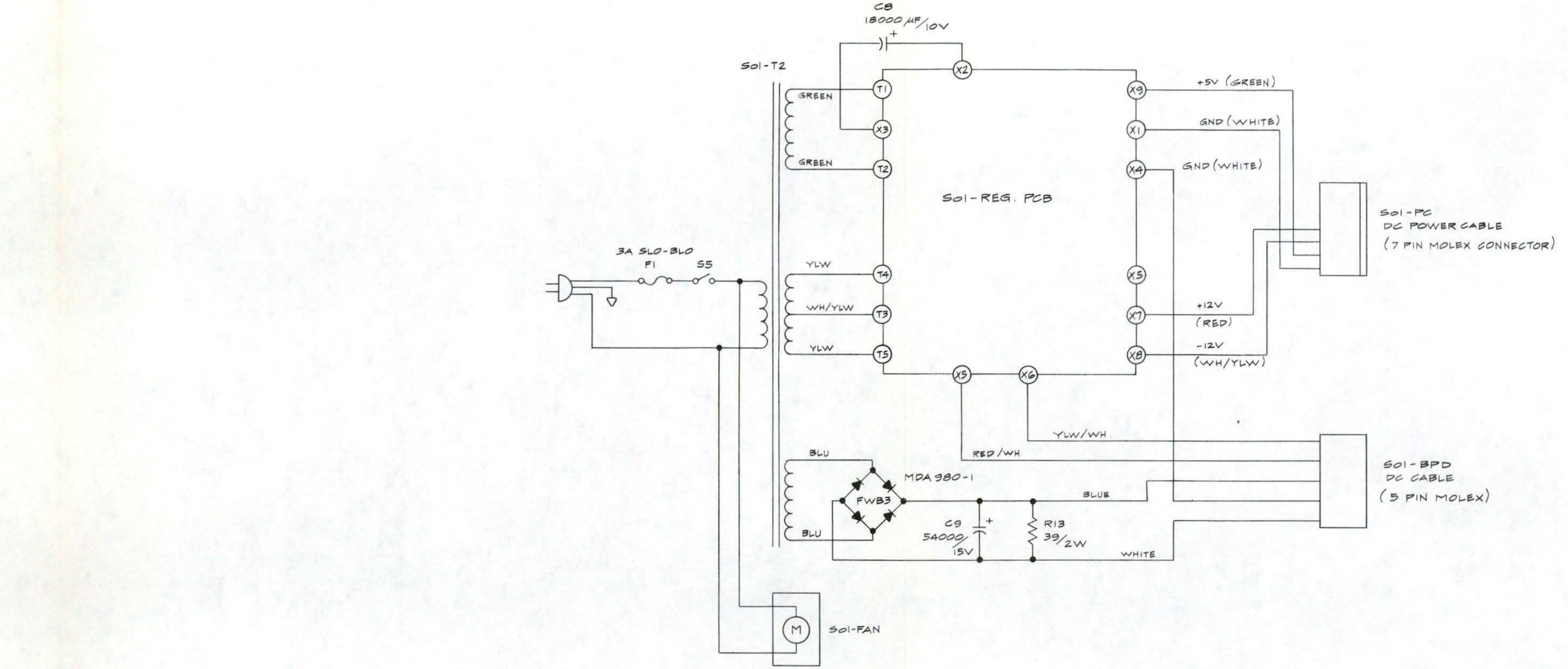


SCHEMATIC, S01-10 POWER SUPPLY

SCALE: ——
DATE: 11-12-76
APPROVED BY: R. MARSH
DRAWN BY: LITO

PROCESSOR TECHNOLOGY CORP.

X-14



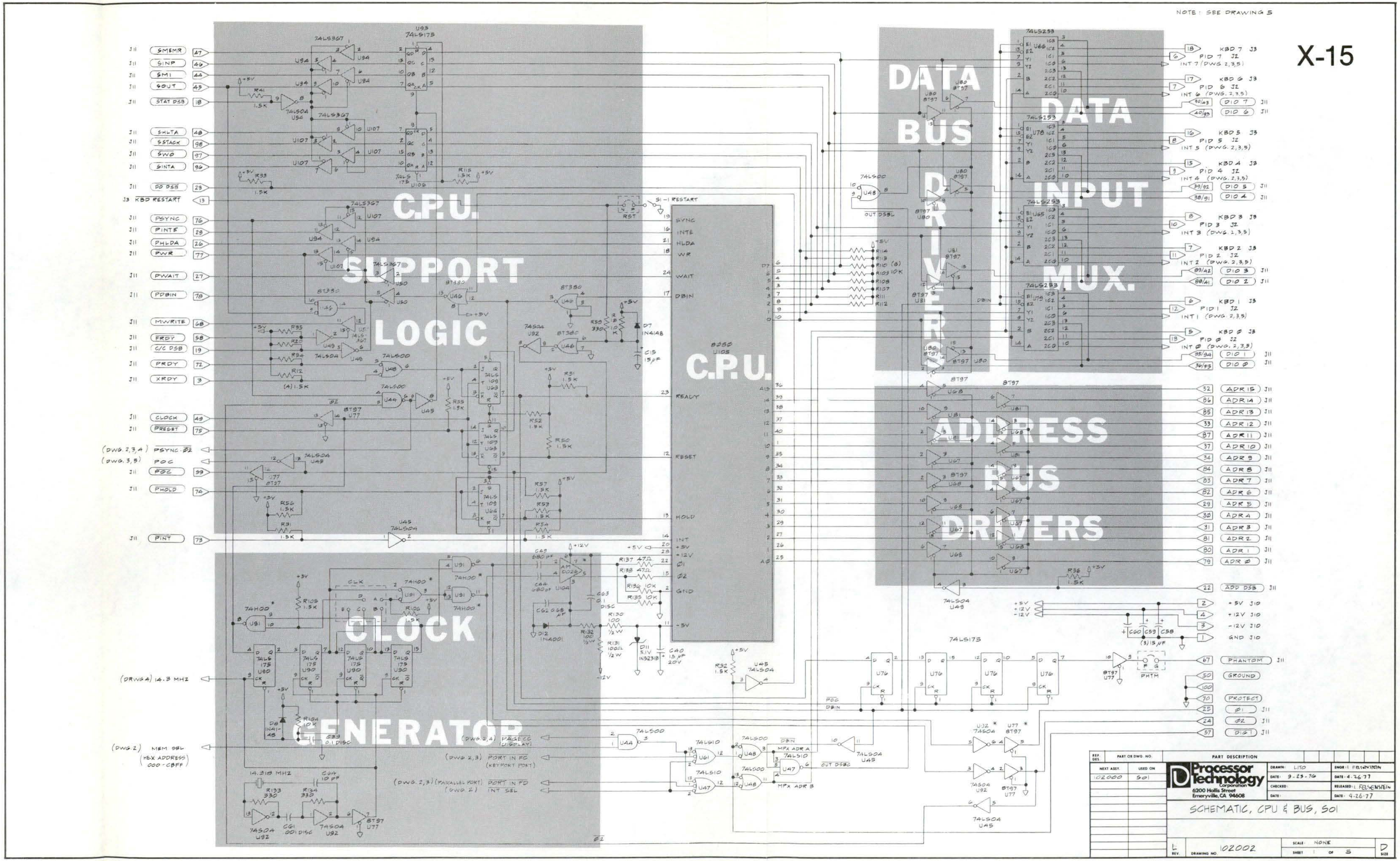
SCHEMATIC, Sol-20 POWER SUPPLY

SCALE: ——
DATE: 11-10-76
APPROVED BY: R. MARSH
DRAWN BY: LITO

PROCESSOR TECHNOLOGY CORP.

Rev. B

X-15



REF. DES.	PART OR DWG. NO.	PART DESCRIPTION

NEXT ASSY.	USED ON
102000	SOI

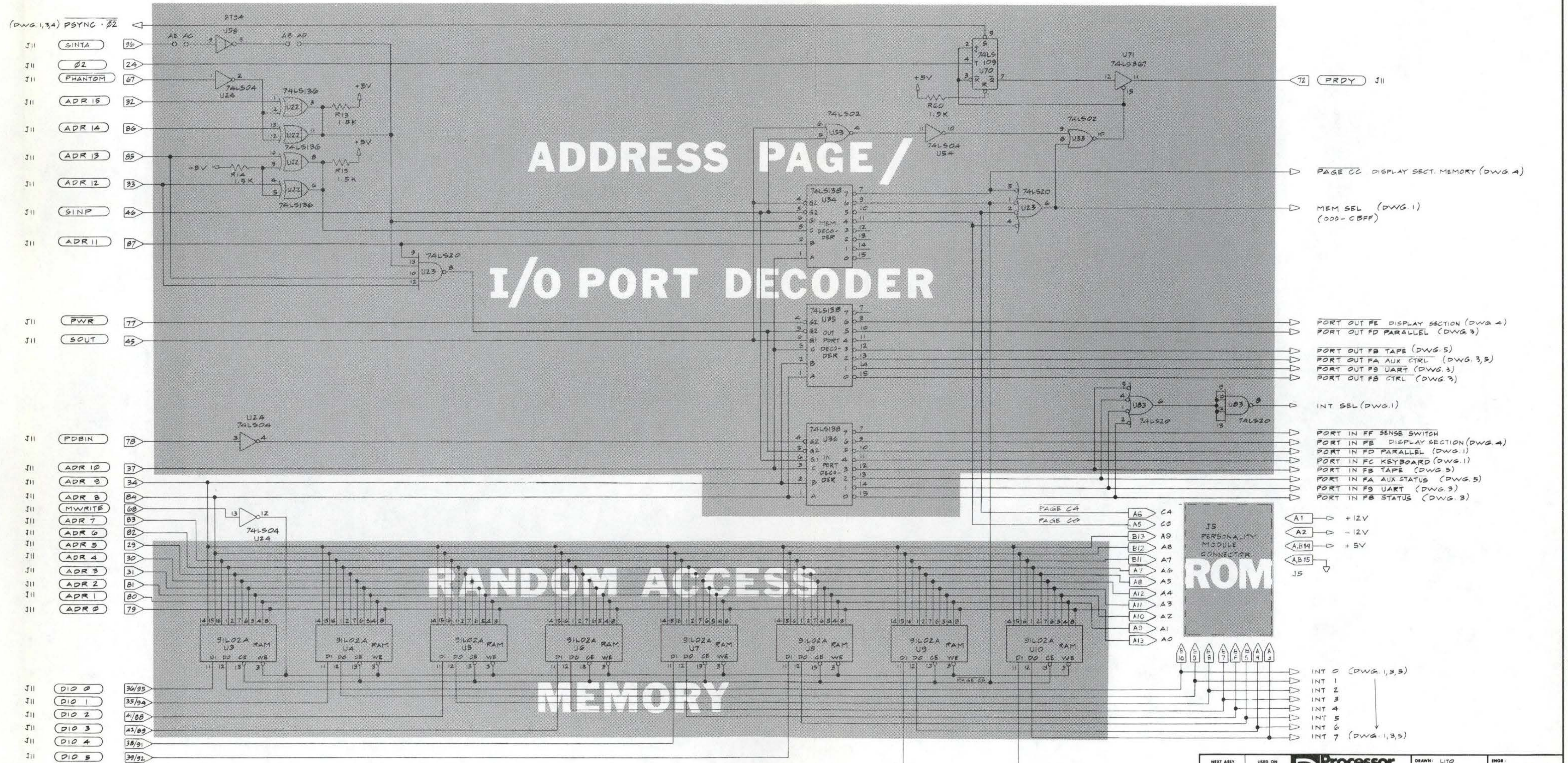
Processor Technology Corporation
6200 Hollis Street
Emeryville, CA 94608

DRAWN:	LITO	ENGR:	L FELSENSTEIN
DATE:	9-23-76	DATE:	4-26-77
CHECKED:		RELEASED:	L FELSENSTEIN
DATE:		DATE:	4-26-77

SCHEMATIC, CPU & BUS, SOI

REV.	DRAWING NO.	SCALE:	SHEET	OF	SIZE
L	102002	NONE	1	5	D

NOTE: SEE DRAWING 5



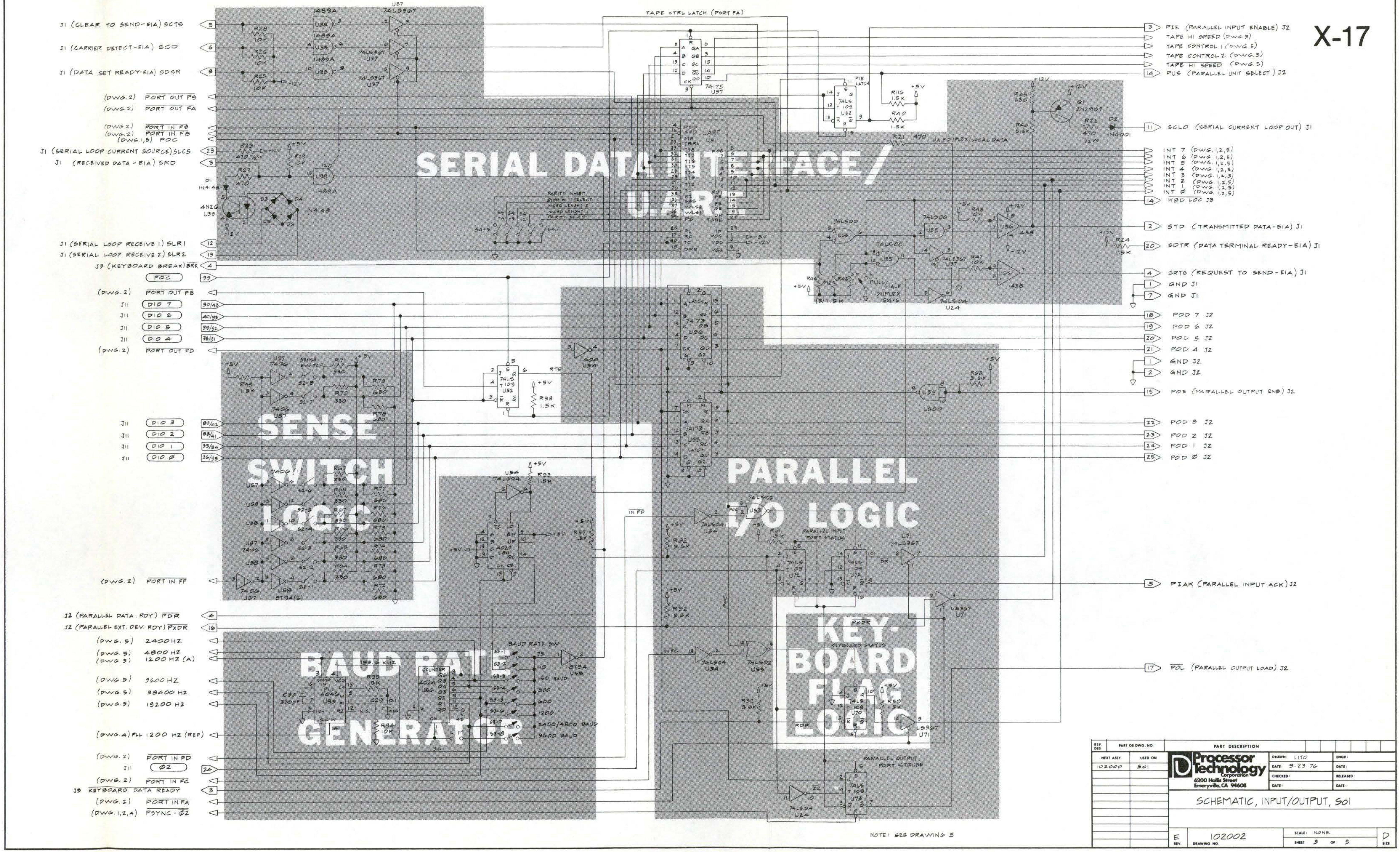
NEXT ASSY.	USED ON
102000	Sol

Processor Technology Corporation
6200 Hollis Street
Emeryville, CA 94608

DRAWN: LITO
DATE: 9-23-76
CHECKED:
DATE:
ENGR:
DATE:
RELEASED:
DATE:

SCHEMATIC, MEMORY & DECODER, Sol

E REV.
DRAWING NO. 102002
SCALE: NONE
SHEET 2 OF 5
D SIZE



NOTE: SEE DRAWING 5

REF. DES.	PART OR DWG. NO.	PART DESCRIPTION
NEXT ASSY.	USED ON	
102000	Sol	

Processor Technology Corporation
6200 Hollis Street
Emeryville, CA 94608

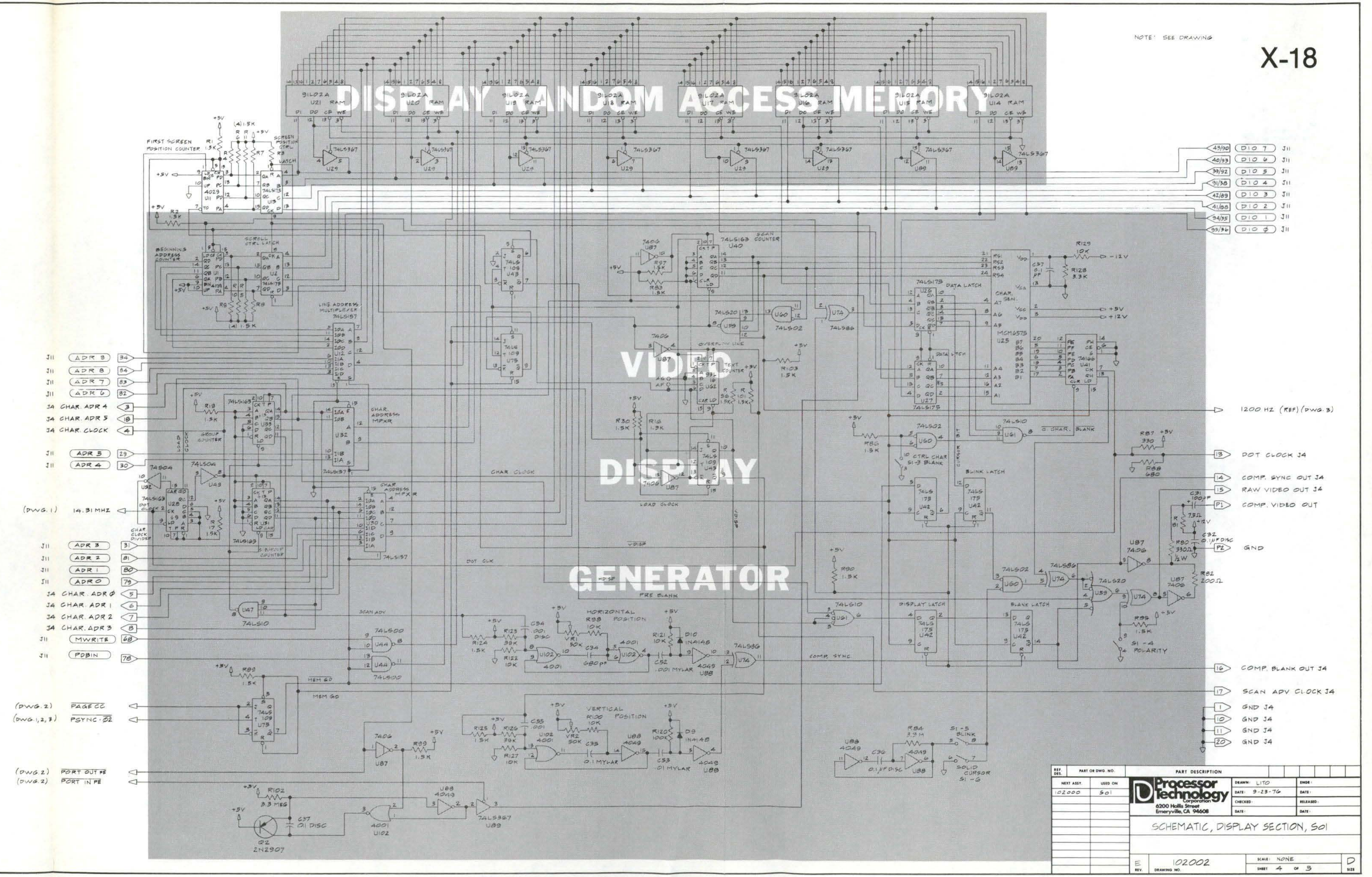
DRAWN: LITO
DATE: 9-23-76
CHECKED:
DATE:
ENGR:
DATE:
RELEASED:
DATE:

SCHEMATIC, INPUT/OUTPUT, Sol

E REV.
102002 DRAWING NO.
SCALE: NONE
SHEET 3 OF 5
D SIZE

X-18

NOTE: SEE DRAWING



REF. DES.	PART OR DWG. NO.	PART DESCRIPTION
NEXT ASSY.	USED ON	
102000	S01	

Processor Technology Corporation
6200 Hollis Street
Emeryville, CA 94608

DRAWN: LITO
DATE: 9-23-76
CHECKED:
DATE:
ENGR:
DATE:
RELEASED:
DATE:

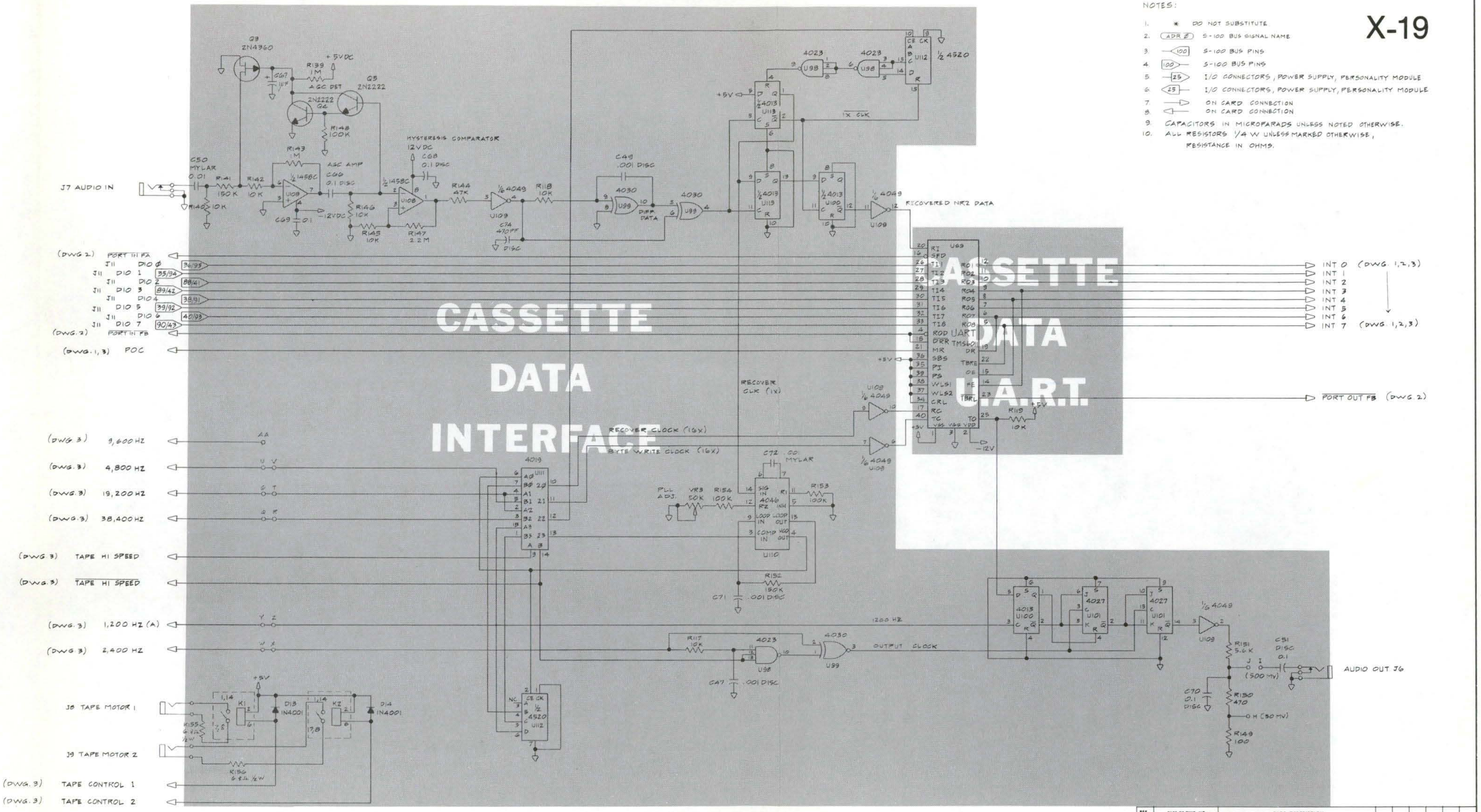
SCHEMATIC, DISPLAY SECTION, S01

E REV.
102002 DRAWING NO.
SCALE: NONE
SHEET 4 OF 5
D SIZE

X-19

NOTES:

1. ✱ DO NOT SUBSTITUTE
2. (ADR Ø) S-100 BUS SIGNAL NAME
3. —<100] S-100 BUS PINS
4. [100>— S-100 BUS PINS
5. —[25> I/O CONNECTORS, POWER SUPPLY, PERSONALITY MODULE
6. <25]— I/O CONNECTORS, POWER SUPPLY, PERSONALITY MODULE
7. ——▷ ON CARD CONNECTION
8. ◁—— ON CARD CONNECTION
9. CAPACITORS IN MICROFARADS UNLESS NOTED OTHERWISE.
10. ALL RESISTORS 1/4 W UNLESS MARKED OTHERWISE, RESISTANCE IN OHMS.



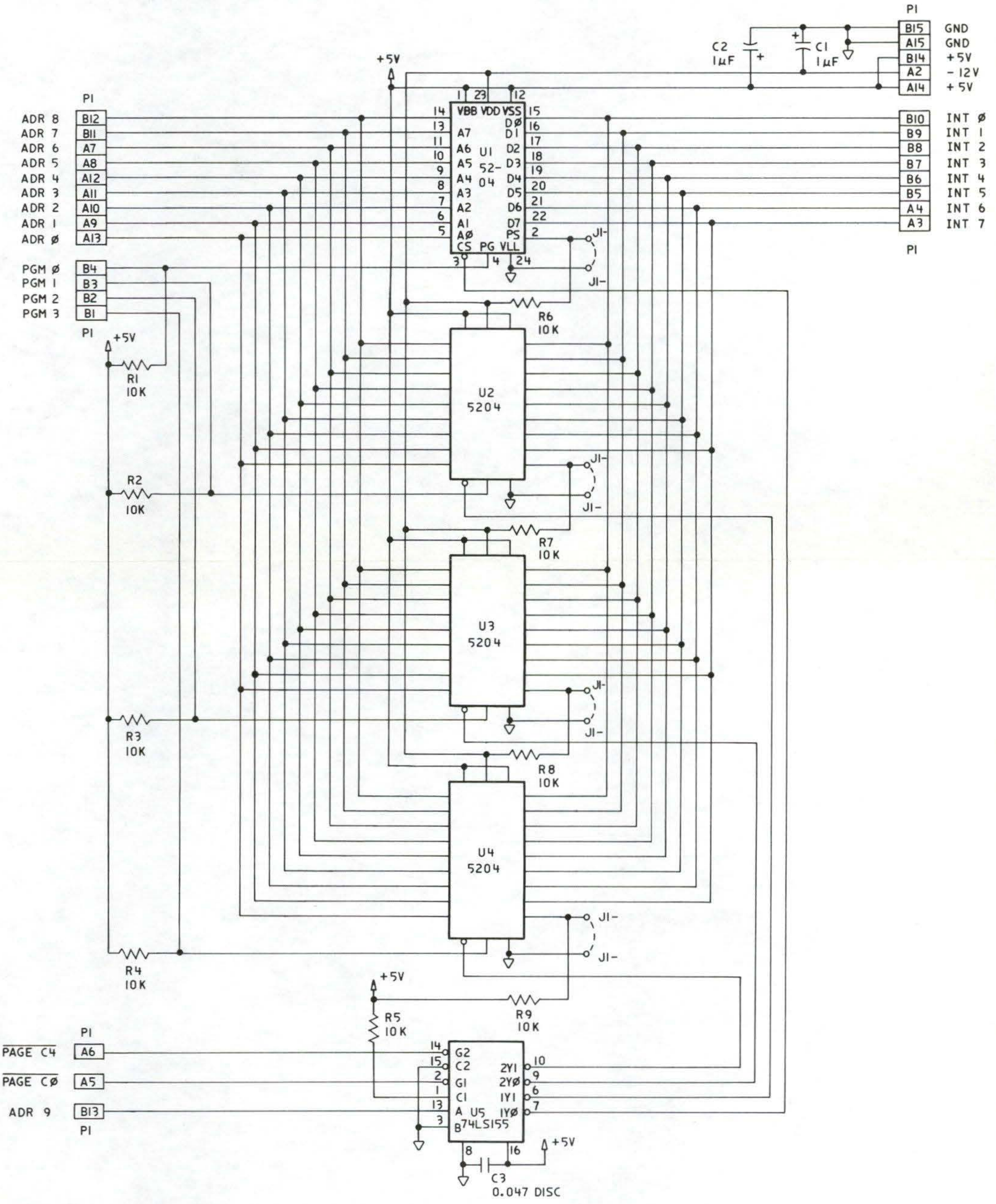
REF. DES.	PART OR DWG. NO.	PART DESCRIPTION
NEXT ASSY.	USED ON	
102000	501	

Processor Technology Corporation
6200 Hollis Street
Emeryville, CA 94608

DRAWN: LITO		ENGR:	
DATE: 6-28-76		DATE:	
CHECKED:		RELEASED:	
DATE:		DATE:	

SCHEMATIC, AUDIO TAPE I/O, SOI

REV.	DRAWING NO.	SCALE:	SHEET	SIZE
E	102002	NONE	5 OF 5	D



REF. DES.	PART OR DWG. NO.	PART DESCRIPTION

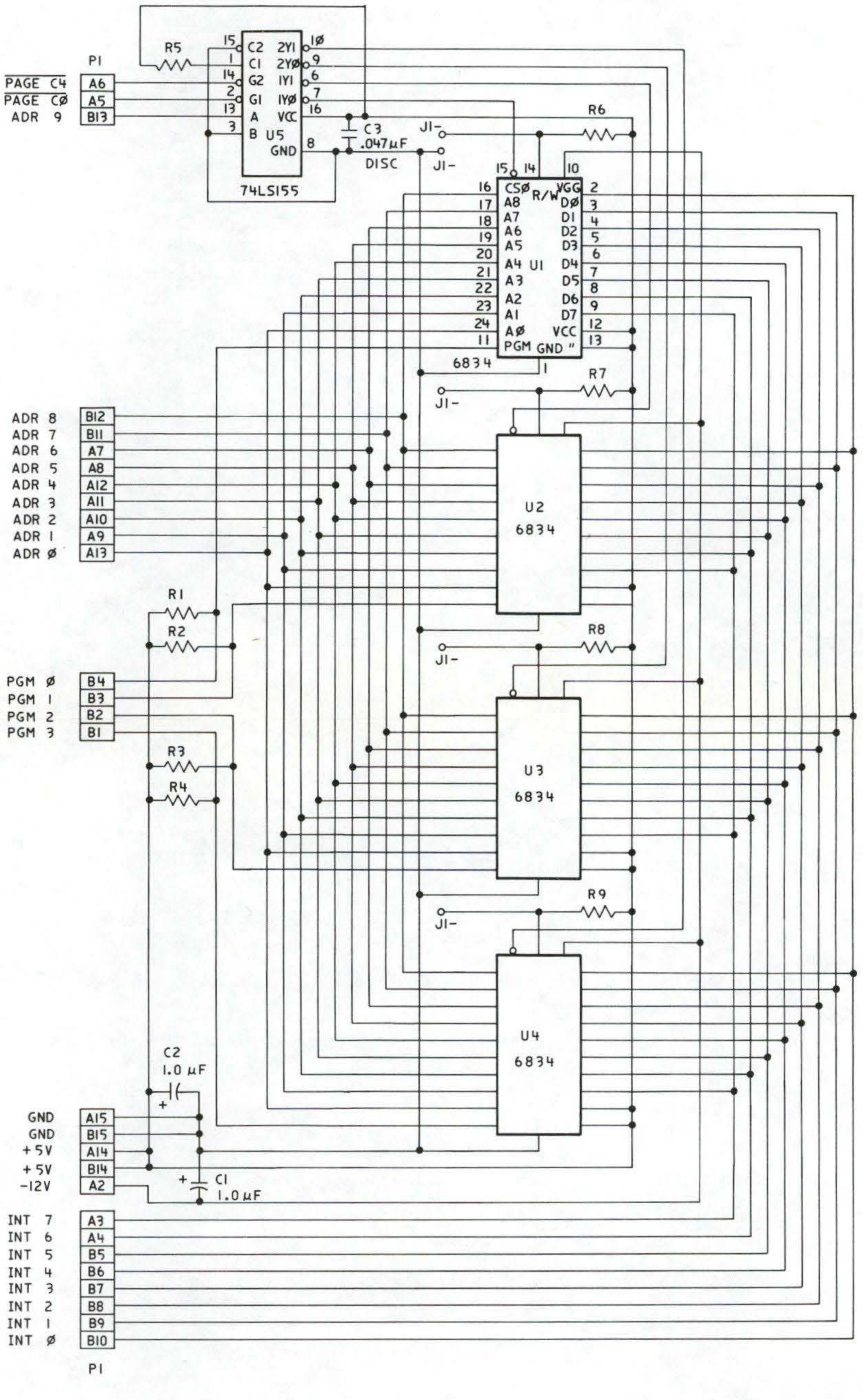
NEXT ASSY.	USED ON
108000	Sol

Processor Technology Corporation
6200 Hollis Street
Emeryville, CA 94608

DRAWN: LITO
DATE: 10-19-76
CHECKED:
DATE:
ENGR:
DATE:
RELEASED:
DATE:

SCHEMATIC, PERSONALITY MODULE 5204

REV.
DRAWING NO. 108002
SCALE: NONE
SHEET 1 OF 1
SIZE C



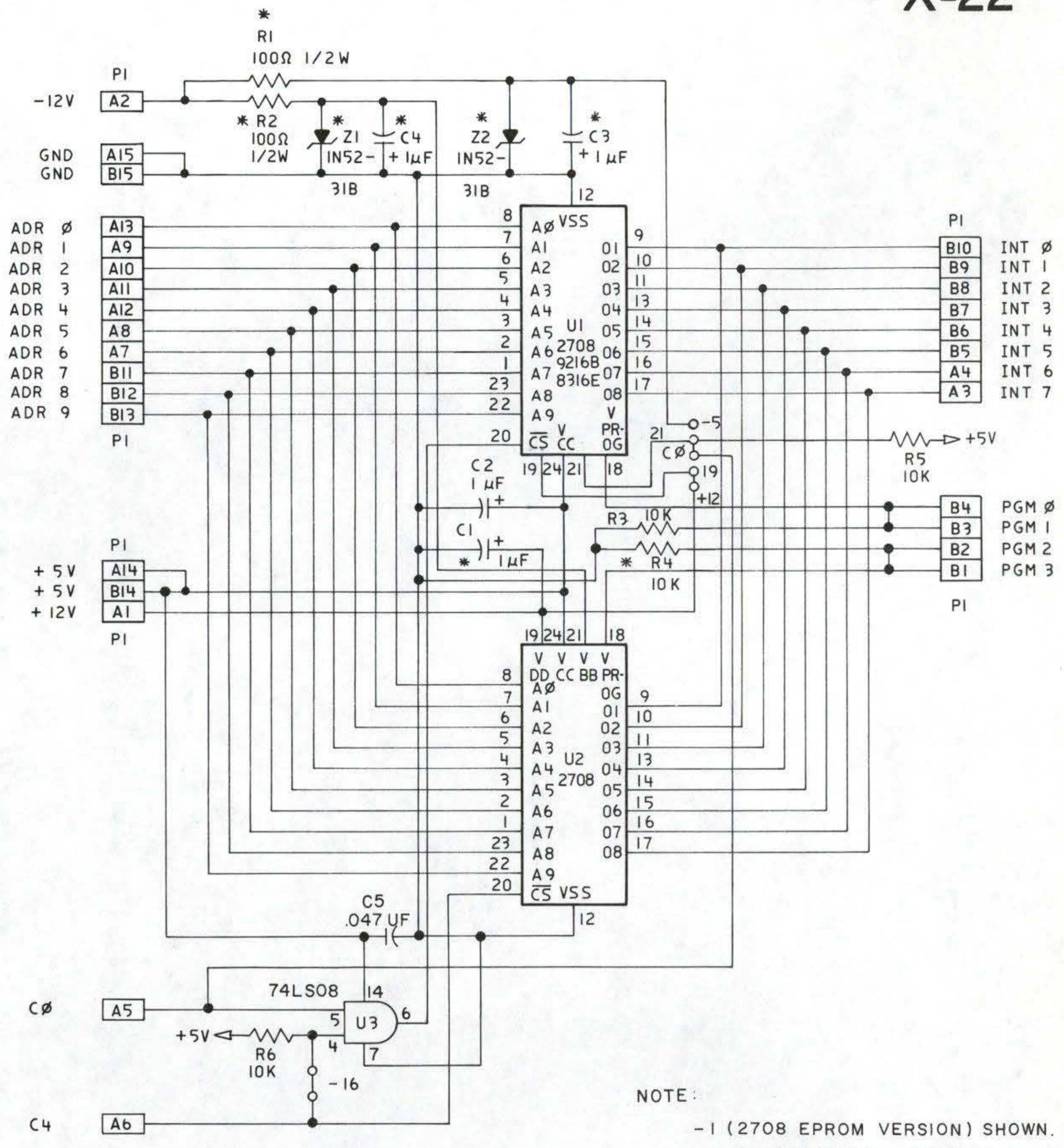
NOTE: ALL RESISTORS 10K, 1/4W, 5%

SCHEMATIC, Personality Module 6834

SCALE: —— APPROVED BY: ROBERT M. MARSH DRAWN BY: LITO
DATE: 10-20-76 CHECKED BY: LEE

PROCESSOR TECHNOLOGY CORP.

Sol



NOTE:

-1 (2708 EPROM VERSION) SHOWN.

* = PART WHICH IS MISSING FROM REV. 0 (9216 MASKED ROM VERSION)

REF. DES.	PART OR DWG. NO.	PART DESCRIPTION

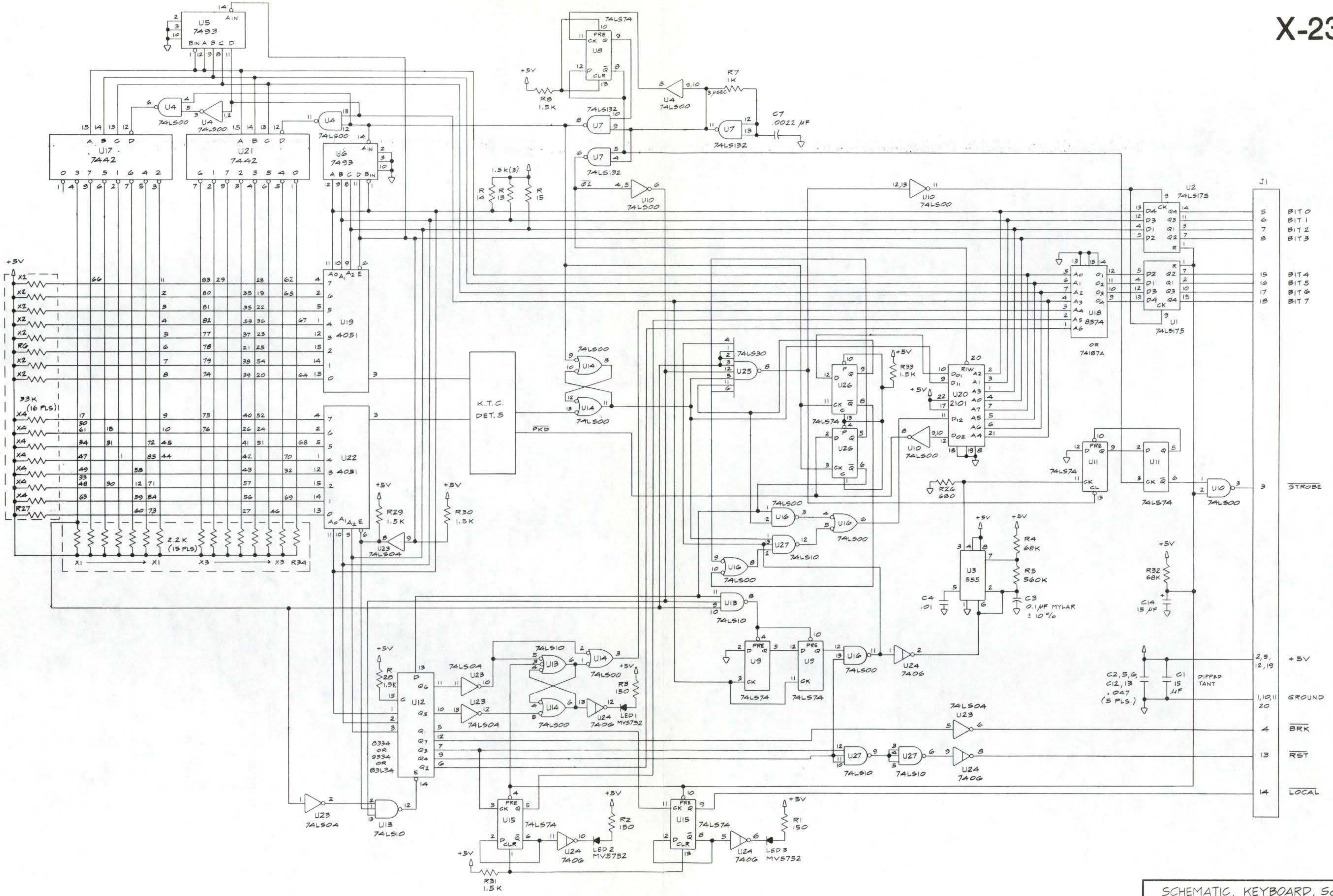
NEXT ASSY.	USED ON
107000	501

Processor Technology Corporation
6200 Hollis Street
Emeryville, CA 94608

DRAWN: LITO	ENGR:
DATE: 10-20-76	DATE:
CHECKED:	RELEASED:
DATE:	DATE:

SCHEMATIC, PERSONALITY MODULE 2708

REV.	DRAWING NO.		SIZE
B	07002	SCALE: NONE SHEET 1 OF 1	B



SCHEMATIC, KEYBOARD, Sol

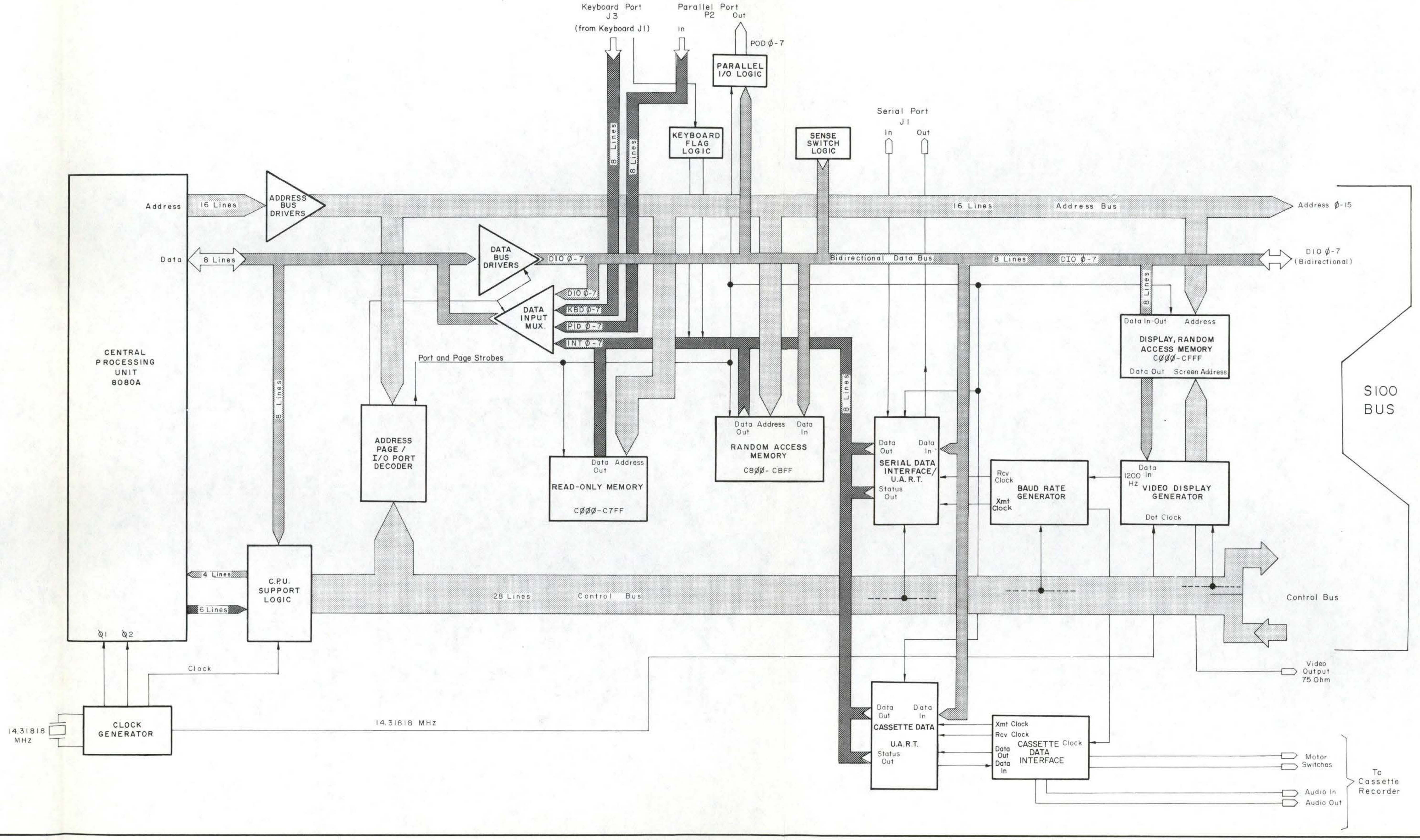
SCALE: ———
DATE: 11-8-76
APPROVED BY: R. MARSH
DRAWN BY LITO
REVISED

PROCESSOR TECHNOLOGY CORP.

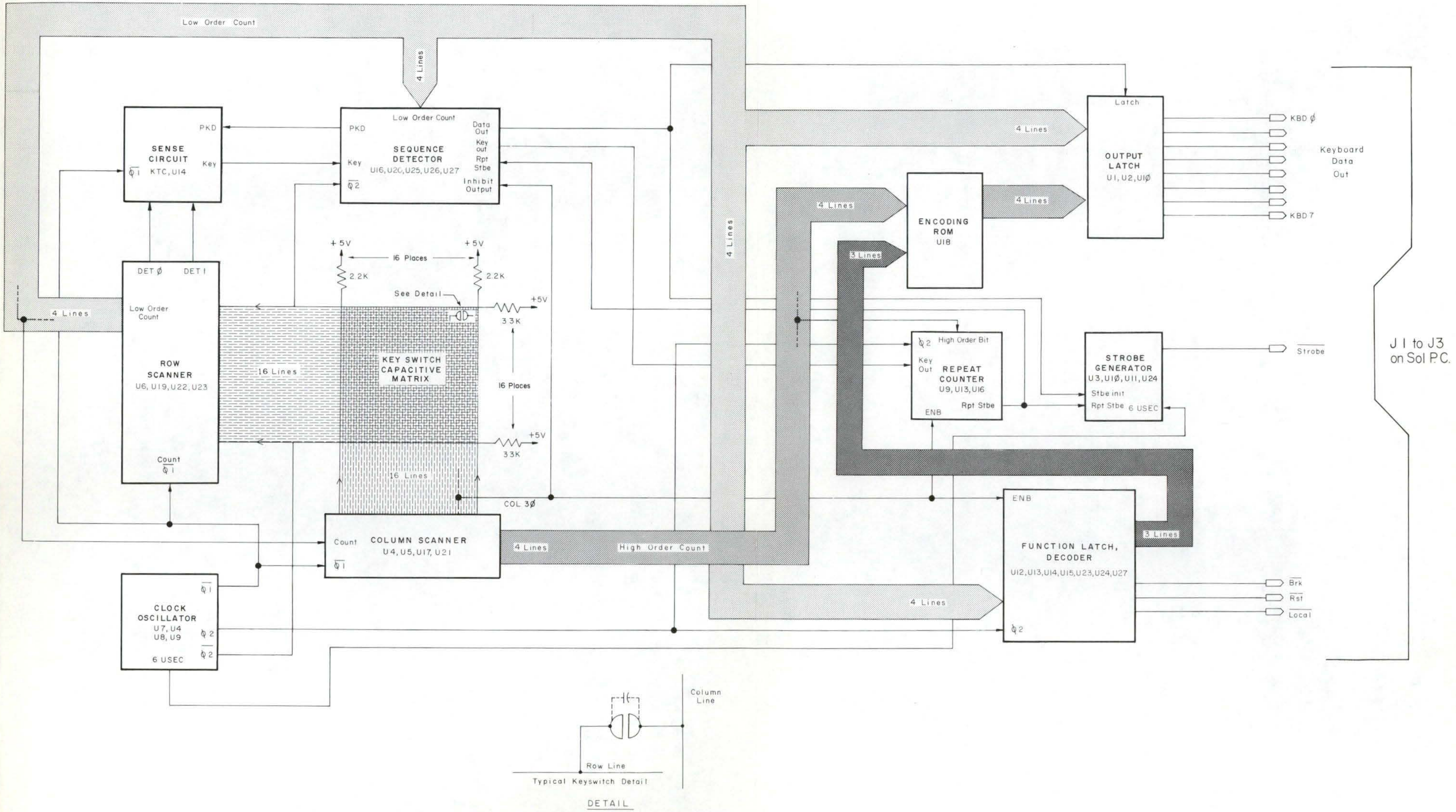
DRAWING NUMBER

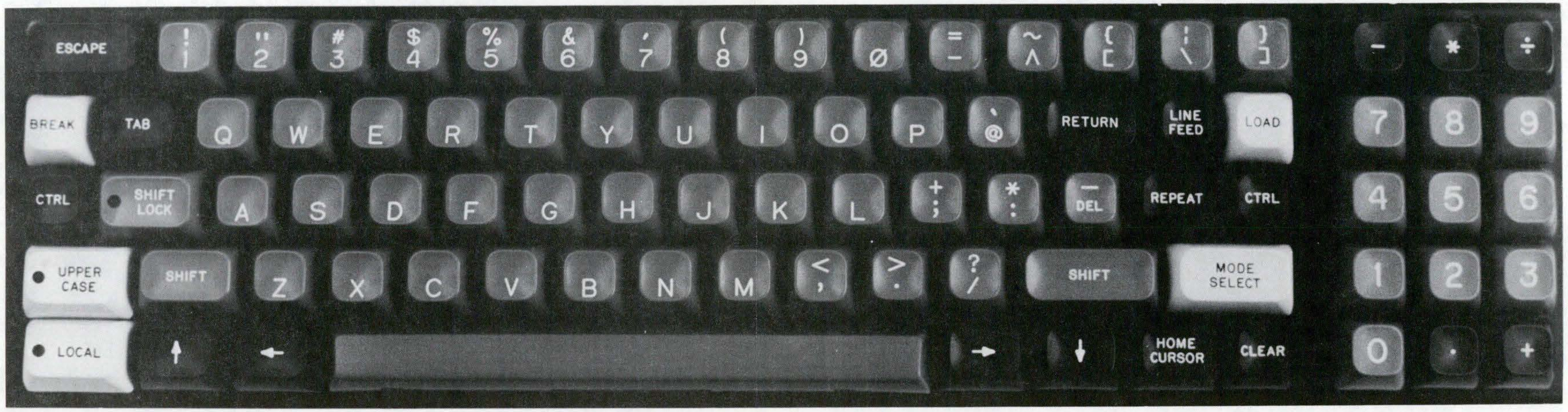
BLOCK FUNCTIONS (in alphabetic order)	SCHEMATIC PAGE	USES IC NO's
1. ADDRESSES BUS DRIVERS	X15	U67,68,81
2. ADDRESS PAGE/I/O PORT DECODER	X15,X16	U22,23,24,34,35, 36,44,47,48,53, 61,83
3. BAUD RATE GENERATOR	X17	U84,85,86
4. CASSETTE DATA INTERFACE	X19	U99,100,101,108, 109,110,113,U69, U98,U110,U111,U112
5. CASSETTE DATA U.A.R.T.	X19	U69
6. CENTRAL PROCESSING UNIT	X15	U105
7. CLOCK GENERATOR	X15	U77,90,91,92,104
8. C.P.U. SUPPORT LOGIC	X15	U44,45,46,48,49, 50,76,77,93,106,107
9. DATA BUS DRIVERS	X15	U80,81
10. DATA INPUT MULTIPLEXER	X15	U65,66,77,78
11. DISPLAY RANDOM ACCESS MEMORY	X18	U14-21,U29,U89,U44, U75
12. KEYBOARD FLAG LOGIC	X17	U53,54,70,71
13. PARALLEL I/O LOGIC	X17	U53,54,71,72,73, 95,96
14. RANDOM ACCESS MEMORY (RAM)	X16	U3,4,5,6,7,8,9, 10,24
15. READ-ONLY MEMORY (ROM)	X16	(SEE SEC. IV)
16. SENSE SWITCH LOGIC	X17	U57,58
17. SERIAL DATA INTERFACE U.A.R.T.	X17	U37,38,39,24,51, 55,56
18. VIDEO DISPLAY GENERATOR	X18	U1,2,11,12,13,25, 26,27,28,30,31,32, 33,40,41,42,43,47, 59,60,61,62,74,75, 87,88,89,U92,102

Sol-P.C. BLOCK DIAGRAM- Rev. D, E



Sol-KEYBOARD BLOCK DIAGRAM – Rev. B, C





Sol - KEYBOARD PHOTO

APPENDICES

Warranty

PROCESSOR TECHNOLOGY CORPORATION, in recognition of its responsibility to provide quality components and adequate instruction for their proper assembly, warrants its products as follows:

All components sold by **Processor Technology Corporation** are purchased through normal factory distribution and any part which fails because of defects in workmanship or material will be replaced at no charge for a period of 3 months for kits, and one year for assembled modules, following the date of purchase. The defective part must be returned postpaid to **Processor Technology Corporation** within the warranty period.

Any malfunctioning module, purchased as a kit directly from **Processor Technology** and returned to the factory within the three-month warranty period, which in the judgement of **PTC** has been assembled with care and not subjected to electrical or mechanical abuse, will be restored to proper operating condition and returned, regardless of cause of malfunction, without charge. Kits purchased from authorized **PTC** dealers should be returned to the selling dealer for the same warranty service.

Any modules purchased as a kit and returned to **PTC**, which in the judgement of **PTC** are not covered by the above conditions, will be repaired and returned at a cost commensurate with the work required. In any case, this charge will not exceed $20.00 without prior notification and approval of the owner.

Any modules, purchased as assembled units are guaranteed to meet specifications in effect at the time of manufacture for a period of at least one year following purchase. These modules are additionally guaranteed against defects in materials or workmanship for the same one year period. All warranted factory assembled units returned to **PTCO** postpaid will be repaired and returned without charge.

This warranty is made in lieu of all other warranties expressed or implied and is limited in any case to the repair or replacement of the module involved.

JUMP

C3	JMP	Adr
C2	JNZ	
CA	JZ	
D2	JNC	
DA	JC	
E2	JPO	
EA	JPE	
F2	JP	
FA	JM	
E9	PCHL	

MOVE IMMEDIATE

06	MVI	B.	D8
0E	MVI	C.	
16	MVI	D.	
1E	MVI	E.	
26	MVI	H.	
2E	MVI	L.	
36	MVI	M.	
3E	MVI	A.	

INCREMENT**

04	INR	B
0C	INR	C
14	INR	D
1C	INR	E
24	INR	H
2C	INR	L
34	NR	M
3C	INR	A
03	NX	B
13	NX	D
23	NX	H
33	NX	SP

CALL

CD	CALL	Adr
C4	CNZ	
CC	CZ	
D4	CNC	
DC	CC	
E4	CPO	
EC	CPE	
F4	CP	
FC	CM	

Acc IMMEDIATE*

C6	ADI	D8
CE	ACI	
D6	SUI	
DE	SBI	
E6	ANI	
EE	XRI	
F6	ORI	
FE	CPI	

DECREMENT**

05	DCR	B
0D	DCR	C
15	DCR	D
1D	DCR	E
25	DCR	H
2D	DCR	L
35	DCR	M
3D	DCR	A
0B	DCX	B
1B	DCX	D
2B	DCX	H
3B	DCX	SP

RETURN

C9	RET
C0	RNZ
C8	RZ
D0	RNC
D8	RC
E0	RPO
E8	RPE
F0	RP
F8	RM

LOAD IMMEDIATE

01	LXI	B.	D16
11	LXI	D.	
21	LXI	H.	
31	LXI	SP.	

DOUBLE ADD†

09	DAD	B
19	DAD	D
29	DAD	H
39	DAD	SP

LOAD/STORE

0A	LDAX	B
1A	LDAX	D
2A	LHLD	Adr
3A	LDA	Adr
02	STAX	B
12	STAX	D
22	SHLD	Adr
32	STA	Adr

RESTART

C7	RST	0
CF	RST	1
D7	RST	2
DF	RST	3
E7	RST	4
EF	RST	5
F7	RST	6
FF	RST	7

STACK OPS

C5	PUSH	B
D5	PUSH	D
E5	PUSH	H
F5	PUSH	PSW
C1	POP	B
D1	POP	D
E1	POP	H
F1	POP	PSW*
E3	XTHL	
F9	SPHL	

SPECIALS

EB	XCHG
27	DAA*
2F	CMA
37	STC†
3F	CMC†

INPUT/OUTPUT

D3	OUT	D8
DB	IN	D8

ROTATE†

07	RLC
0F	RRC
17	RAL
1F	RAR

CONTROL

00	NOP
76	HLT
F3	DI
FB	EI

MOVE

40	MOV	B,B
41	MOV	B,C
42	MOV	B,D
43	MOV	B,E
44	MOV	B,H
45	MOV	B,L
46	MOV	B,M
47	MOV	B,A
48	MOV	C,B
49	MOV	C,C
4A	MOV	C,D
4B	MOV	C,E
4C	MOV	C,H
4D	MOV	C,L
4E	MOV	C,M
4F	MOV	C,A
50	MOV	D,B
51	MOV	D,C
52	MOV	D,D
53	MCV	D,E
54	MOV	D,H
55	MOV	D,L
56	MOV	D,M
57	MOV	D,A

MOVE (cont)

58	MOV	E,B
59	MOV	E,C
5A	MOV	E,D
5B	MOV	E,E
5C	MOV	E,H
5D	MOV	E,L
5E	MOV	E,M
5F	MOV	E,A
60	MOV	H,B
61	MOV	H,C
62	MOV	H,D
63	MOV	H,E
64	MOV	H,H
65	MOV	H,L
66	MOV	H,M
67	MOV	H,A
68	MOV	L,B
69	MOV	L,C
6A	MOV	L,D
6B	MOV	L,E
6C	MOV	L,H
6D	MOV	L,L
6E	MOV	L,M
6F	MOV	L,A
70	MOV	M,B
71	MOV	M,C
72	MOV	M,D
73	MOV	M,E
74	MOV	M,H
75	MOV	M,L

77	MOV	M,A
78	MOV	A,B
79	MOV	A,C
7A	MOV	A,D
7B	MOV	A,E
7C	MOV	A,H
7D	MOV	A,L
7E	MOV	A,M
7F	MOV	A,A

ACCUMULATOR*

80	ADD	B
81	ADD	C
82	ADD	D
83	ADD	E
84	ADD	H
85	ADD	L
86	ADD	M
87	ADD	A
88	ADC	B
89	ADC	C
8A	ADC	D
8B	ADC	E
8C	ADC	H
8D	ADC	L
8E	ADC	M
8F	ADC	A
90	SUB	B
91	SUB	C
92	SUB	D
93	SUB	E
94	SUB	H
95	SUB	L
96	SUB	M
97	SUB	A
98	SBB	B
99	SBB	C
9A	SBB	D
9B	SBB	E
9C	SBB	H
9D	SBB	L
9E	SBB	M
9F	SBB	A
A0	ANA	B
A1	ANA	C
A2	ANA	D
A3	ANA	E
A4	ANA	H
A5	ANA	L
A6	ANA	M
A7	ANA	A

A8	XRA	B
A9	XRA	C
AA	XRA	D
AB	XRA	E
AC	XRA	H
AD	XRA	L
AE	XRA	M
AF	XRA	A
B0	ORA	B
B1	ORA	C
B2	ORA	D
B3	ORA	E
B4	ORA	H
B5	ORA	L
B6	ORA	M
B7	ORA	A
B8	CMP	B
B9	CMP	C
BA	CMP	D
BB	CMP	E
BC	CMP	H
BD	CMP	L
BE	CMP	M
BF	CMP	A

PSEUDO INSTRUCTION

ORG	Adr	
END		
EQU	D16	
DS	D16	
DB	D8	[ ]
DW	D16	[ ]

CONSTANT DEFINITION

0BDH 1AH	Hex
105D 105	Decimal
72O 72Q	Octal
11011B 00110B	Binary
'TEST' 'A' 'B'	ASCII

OPERATORS

+ −

STANDARD SETS

A	SET	7
B	SET	0
C	SET	1
D	SET	2
E	SET	3
H	SET	4
L	SET	5
M	SET	6
SP	SET	6
PSW	SET	6

D8 = constant, or logical/arithmetic expression that evaluates to an 8 bit data quantity.

* = all Flags (C,Z,S,P) affected

D16 = constant, or logical/arithmetic expression that evaluates to a 16 bit data quantity.

† = only CARRY affected

Adr = 16 bit address

** = all Flags except CARRY affected; (exception: INX & DCX affect no Flags)

APPENDIX II

00	NOP		28	- - -		50	MOV	D,B	78	MOV	A,B	A0	ANA	B	C8	RZ	
01	LXI	B,D16	29	DAD	H	51	MOV	D,C	79	MOV	A,C	A1	ANA	C	C9	RET	
02	STAX	B	2A	LHLD	Adr	52	MOV	D,D	7A	MOV	A,D	A2	ANA	D	CA	JZ	
03	INX	B	2B	DCX	H	53	MOV	D,E	7B	MOV	A,E	A3	ANA	E	CB	- - -	
04	INR	B	2C	INR	L	54	MOV	D,H	7C	MOV	A,H	A4	ANA	H	CC	CZ	Adr
05	DCR	B	2D	DCR	L	55	MOV	D,L	7D	MOV	A,L	A5	ANA	L	CD	CALL	Adr
06	MVI	B,D8	2E	MVI	L,D8	56	MOV	D,M	7E	MOV	A,M	A6	ANA	M	CE	ACI	D8
07	RLC		2F	CMA		57	MOV	D,A	7F	MOV	A,A	A7	ANA	A	CF	RST	1
08	- - -		30	- - -		58	MOV	E,B	80	ADD	B	A8	XRA	B	D0	RNC	
09	DAD	B	31	LXI	SP,D16	59	MOV	E,C	81	ADD	C	A9	XRA	C	D1	POP	D
0A	LDAX	B	32	STA	Adr	5A	MOV	E,D	82	ADD	D	AA	XRA	D	D2	JNC	Adr
0B	DCX	B	33	INX	SP	5B	MOV	E,E	83	ADD	E	AB	XRA	E	D3	OUT	D8
0C	INR	C	34	INR	M	5C	MOV	E,H	84	ADD	H	AC	XRA	H	D4	CNC	Adr
0D	DCR	C	35	DCR	M	5D	MOV	E,L	85	ADD	L	AD	XRA	L	D5	PUSH	D
0E	MVI	C,D8	36	MVI	M,D8	5E	MOV	E,M	86	ADD	M	AE	XRA	M	D6	SUI	D8
0F	RRC		37	STC		5F	MOV	E,A	87	ADD	A	AF	XRA	A	D7	RST	2
10	- - -		38	- - -		60	MOV	H,B	88	ADC	B	B0	ORA	B	D8	RC	
11	LXI	D,D16	39	DAD	SP	61	MOV	H,C	89	ADC	C	B1	ORA	C	D9	- - -	
12	STAX	D	3A	LDA	Adr	62	MOV	H,D	8A	ADC	D	B2	ORA	D	DA	JC	Adr
13	INX	D	3B	DCX	SP	63	MOV	H,E	8B	ADC	E	B3	ORA	E	DB	IN	D8
14	INR	D	3C	INR	A	64	MOV	H,H	8C	ADC	H	B4	ORA	H	DC	CC	Adr
15	DCR	D	3D	DCR	A	65	MOV	H,L	8D	ADC	L	B5	ORA	L	DD	- - -	
16	MVI	D,D8	3E	MVI	A,D8	66	MOV	H,M	8E	ADC	M	B6	ORA	M	DE	SBI	D8
17	RAL		3F	CMC		67	MOV	H,A	8F	ADC	A	B7	ORA	A	DF	RST	3
18	- - -		40	MOV	B,B	68	MOV	L,B	90	SUB	B	B8	CMP	B	E0	RPO	
19	DAD	D	41	MOV	B,C	69	MOV	L,C	91	SUB	C	B9	CMP	C	E1	POP	H
1A	LDAX	D	42	MOV	B,D	6A	MOV	L,D	92	SUB	D	BA	CMP	D	E2	JPO	Adr
1B	DCX	D	43	MOV	B,E	6B	MOV	L,E	93	SUB	E	BB	CMP	E	E3	XTHL	
1C	INR	E	44	MOV	B,H	6C	MOV	L,H	94	SUB	H	BC	CMP	H	E4	CPO	Adr
1D	DCR	E	45	MOV	B,L	6D	MOV	L,L	95	SUB	L	BD	CMP	L	E5	PUSH	H
1E	MVI	E,D8	46	MOV	B,M	6E	MOV	L,M	96	SUB	M	BE	CMP	M	E6	ANI	D8
1F	RAR		47	MOV	B,A	6F	MOV	L,A	97	SUB	A	BF	CMP	A	E7	RST	4
20	- - -		48	MOV	C,B	70	MOV	M,B	98	SBB	B	C0	RNZ		E8	RPE	
21	LXI	H,D16	49	MOV	C,C	71	MOV	M,C	99	SBB	C	C1	POP	B	E9	PCHL	
22	SHLD	Adr	4A	MOV	C,D	72	MOV	M,D	9A	SBB	D	C2	JNZ	Adr	EA	JPE	Adr
23	INX	H	4B	MOV	C,E	73	MOV	M,E	9B	SBB	E	C3	JMP	Adr	EB	XCHG	
24	INR	H	4C	MOV	C,H	74	MOV	M,H	9C	SBB	H	C4	CNZ	Adr	EC	CPE	Adr
25	DCR	H	4D	MOV	C,L	75	MOV	M,L	9D	SBB	L	C5	PUSH	B	ED	- - -	
26	MVI	H,D8	4E	MOV	C,M	76	HLT		9E	SBB	M	C6	ADI	D8	EE	XRI	D8
27	DAA		4F	MOV	C,A	77	MOV	M,A	9F	SBB	A	C7	RST	0	EF	RST	5

F0	RP	
F1	POP	PSW
F2	JP	Adr
F3	DI	
F4	CP	Adr
F5	PUSH	PSW
F6	ORI	D8
F7	RST	6
F8	RM	
F9	SPHL	
FA	JM	Adr
FB	EI	
FC	CM	Adr
FD	- - -	
FE	CPI	D8
FF	RST	7

HEX-ASCII TABLE

Non-Printing

00	NULL
07	BELL
09	TAB
0A	LF
0B	VT
0C	FORM
0D	CR
11	X-ON
12	TAPE
13	X-OFF
14	
1B	ESC
7D	ALT MODE
7F	RUB OUT

HEX-ASCII TABLE

Printing		Characters		
30	0	40	@	
31	1	20	space	
32	2	21	!	
33	3	22	"	
34	4	23	#	
35	5	24	$	
36	6	25	%	
37	7	26	&	
38	8	27	'	
39	9	28	(	
		29	)	
41	A	2A	*	
42	B	2B	+	
43	C	2C	,	
44	D	2D	–	
45	E	2E	.	
46	F	2F	/	
47	G	3A	:	
48	H	3B	;	
49	I	3C	<	
4A	J	3D	=	
4B	K	3E	>	
4C	L	3F	?	
4D	M	5B	[	
4E	N	5C	\	
4F	O	5D	]	
50	P	5E	↑	(^)
51	Q	5F	←	(-)
52	R			
53	S			
54	T			
55	U			
56	V			
57	W			
58	X			
59	Y			
5A	Z			

D8 = constant, or logical/arithmetic expression that evaluates to an 8 bit data quantity.

D16 = constant, or logical/arithmetic expression that evaluates to a 16 bit data quantity.

Adr = 16 bit address

Processor Technology Corp.

APPENDIX II

STANDARD COLOR CODE FOR RESISTORS AND CAPACITORS

COLOR	SIGNIFICANT FIGURE	DECIMAL MULTIPLIER	TOLERANCE (%)	VOLTAGE RATING*
Black	0	1		--
Brown	1	10		100
Red	2	100		200
Orange	3	1,000		300
Yellow	4	10,000		400
Green	5	100,000		500
Blue	6	1,000,000		600
Violet	7	10,000,000		700
Gray	8	100,000,000		800
White	9	1,000,000,000		900
Gold	-	0.1	5	1000
Silver	-	0.01	10	2000
No Color	-	---	20	500

*Applies to capacitors only.

LOADING DIP (DUAL IN-LINE PACKAGE) DEVICES

Most DIP devices have their leads spread so that they can not be dropped straight into the board. They must be "walked in" using the following procedure:

(1) Orient the device properly. Pin 1 is indicated by a small embossed dot on the top surface of the device at one corner. Pins are numbered counterclockwise from pin 1.

(2) Insert the pins on one side of the device into their holes on the printed circuit card. Do not press the pins all the way in, but stop when they are just starting to emerge from the opposite side of the card.

(3) Exert a sideways pressure on the pins at the other side of the device by pressing against them where they are still wide below the bend. Bring this row of pins into alighment with its holes in the printed circuit card and insert them an equal distance, until they begin to emerge.

(4) Press the device straight down until it seats on the points where the pins widen.

(5) Turn the card over and select two pins at opposite corners of the device. Using a fingernail or a pair of long-nose pliers, push these pins outwards until they are bent at a 45° angle to the surface of the card. This will secure the device until it is soldered.

SOLDERING TIPS

(1) Use a low-wattage iron--25 watts is good. Larger irons run the risk of burning the printed-circuit board. Don't try to use a soldering gun, they are too hot.

(2) Use a small pointed tip and keep it clean. Keep a damp piece of sponge by the iron and wipe the tip on it after each use.

(3) Use 60-40 rosin-core solder ONLY. DO NOT use acid-core solder or externally applied fluxes. Use the smallest diameter solder you can get.

NOTE: DO NOT press the top of the iron on the pad or trace. This will cause the trace to "lift" off of the board which will result in permanent damage.

(4) In soldering, wipe the tip, apply a light coating of new solder to it, and apply the tip to both parts of the joint, that is, both the component lead and the printed-circuit pad. Apply the solder against the lead and pad being heated, but not directly to the tip of the iron. Thus, when the solder

melts the rest of the joint will be hot enough for the solder to "take", (i.e., form a capillary film).

(5) Apply solder for a second or two, then remove the solder and keep the iron tip on the joint. The rosin will bubble out. Allow about three or four bubbles, but don't keep the tip applied for more than ten seconds.

(6) Solder should follow the contours of the original joint. A blob or lump may well be a solder bridge, where enough solder has been built upon one conductor to overflow and "take" on the adjacent conductor. Due to capillary action, these solder bridges look very neat, but they are a constant source of trouble when boards of a high trace density are being soldered. Inspect each integrated circuit and component after soldering for bridges.

(7) To remove solder bridges, it is best to use a vacuum "solder puller" if one is available. If not, the bridge can be reheated with the iron and the excess solder "pulled" with the tip along the printed circuit traces until the lump of solder becomes thin enough to break the bridge. Braid-type solder remover, which causes the solder to "wick up" away from the joint when applied to melted solder, may also be used.

INSTALLING AUGAT PINS

Augat pins are normally supplied on carriers (e.g., 8-pin and 16-pin carriers). In many cases the PC board layout permits Augat pins to be installed while still attached to the carrier or a portion of the carrier. In other cases the pins must be installed singly.

To install two or more pins that are still attached to the carrier, proceed as follows:

NOTE

> It is perfectly alright to appropriately cut a carrier to accommodate the installation. For example, an 8-pin carrier can be cut in half (4 pins each) across the short dimension to fit a 4-pin, 4-corner layout. It may also be cut in half along the long dimension to fit a 4-pin, inline layout.

(1) Insert pins in the mounting holes from the front (component) side of board. (The carrier will hold the pins perpendicular to the board.)

(2) Solder all pins from back (solder) side of board so the solder "wicks up" to the front side.

(3) Check for solder bridges.

(4) Remove carrier.

To install single pins, proceed as follows:

(1) Hold board between two objects so that it stands on edge.

(2) Insert pins in the mounting holes from front (component) side of board.

(3) Solder pins from back (solder) side of board so the solder "wicks up" to the front side. (This will hold the pins firmly in place.)

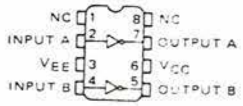
(4) Insert a component lead into one pin and reheat the solder. Using the component lead, adjust pin until it is perpendicular to board. Allow solder to cool while holding the pin as steady as possible. Remove component lead. Repeat this procedure with other pins.

NOTE

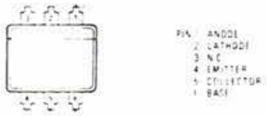
If cooled solder is mottled or crystallized, a "cold joint" is indicated, and the solder should be reheated.

(5) Check each installation for cold joints and solder bridges.

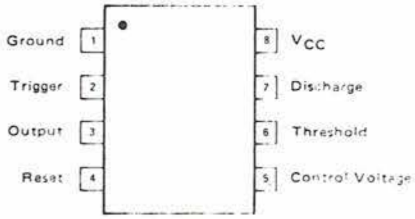
0026



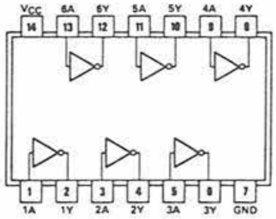
4N26



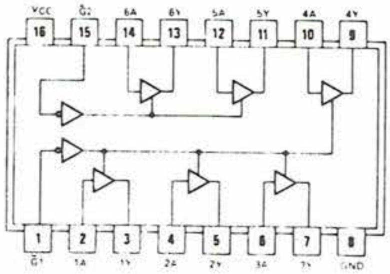
555



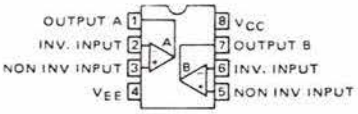
8T94



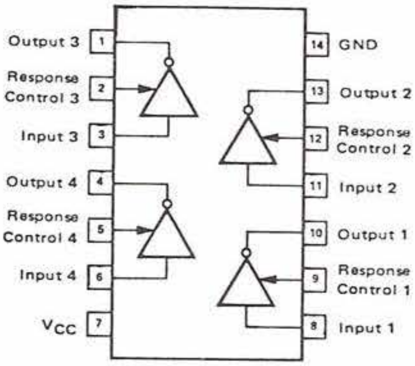
8T97 or 74367



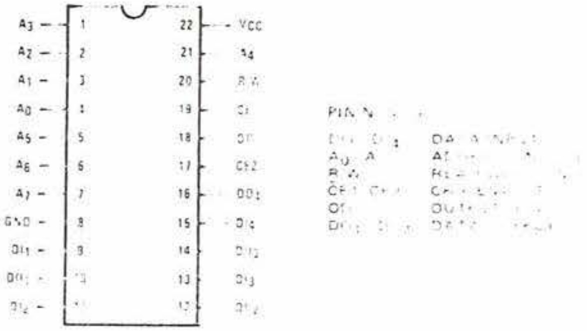
1458



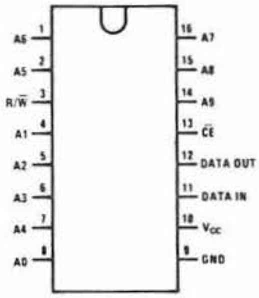
1489



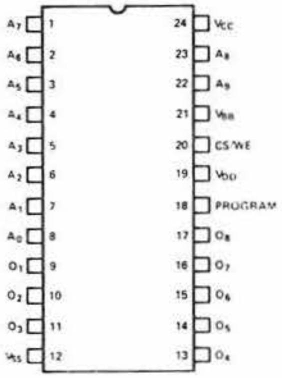
2101



21L02 or 91L02



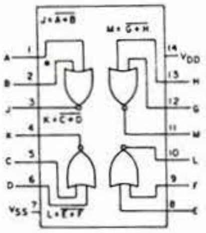
2708



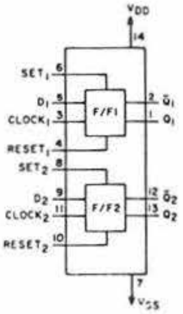
PIN NAMES

A_0-A_9	ADDRESS INPUTS
O_1-O_8	DATA OUTPUTS
CS	CHIP SELECT INPUTS

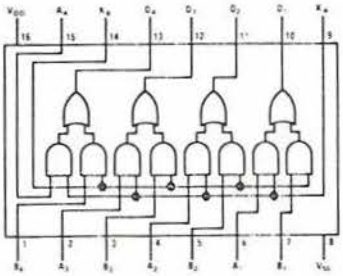
4001



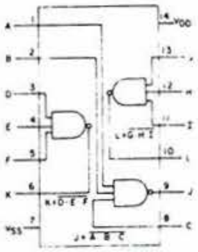
4013



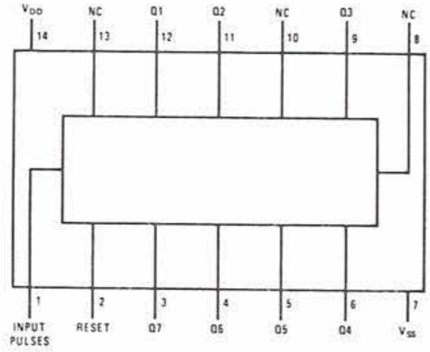
4019



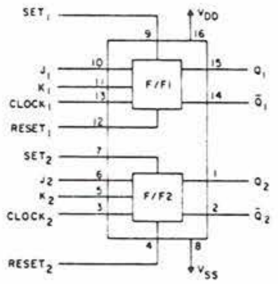
4023



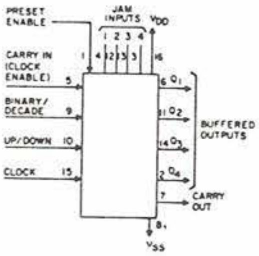
4024



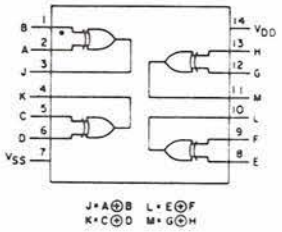
4027



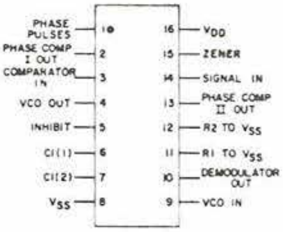
4029



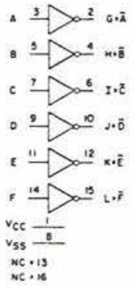
4030



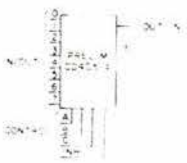
4046



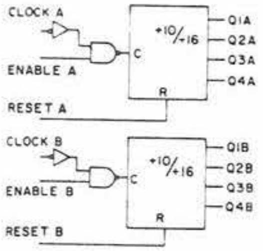
4049



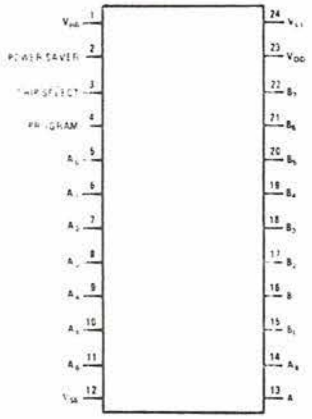
4051



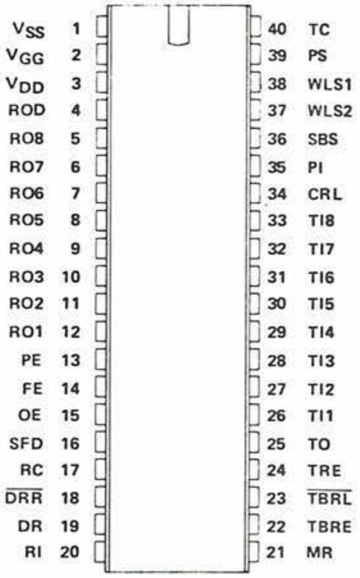
4520



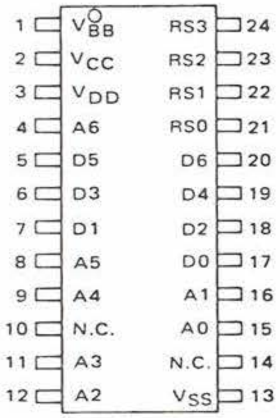
5204



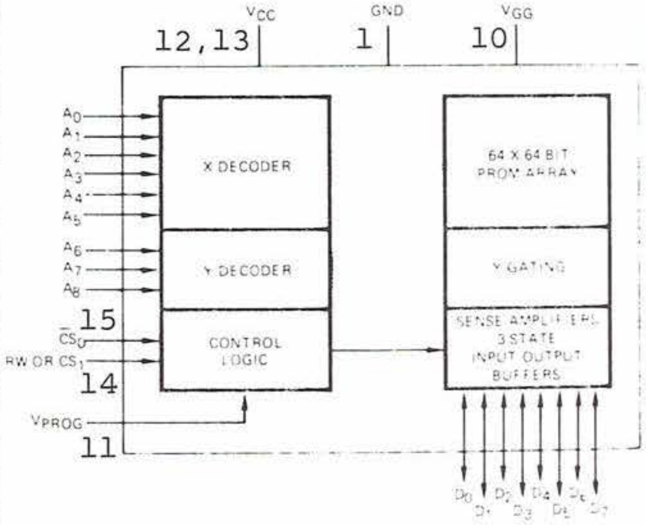
6011



6574 or 6575

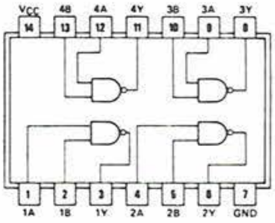


6834

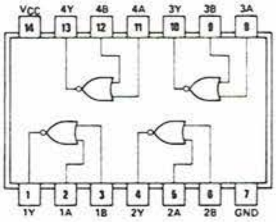


AØ - 24	A5 - 19	Dl - 3	D6 - 8
Al - 23	A6 - 18	D2 - 4	D7 - 9
A2 - 22	A7 - 17	D3 - 5	
A3 - 21	A8 - 16	D4 - 6	
A4 - 20	DØ - 2	D5 - 7	

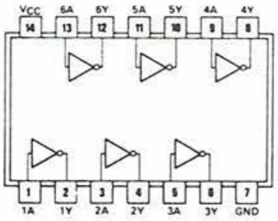
7400



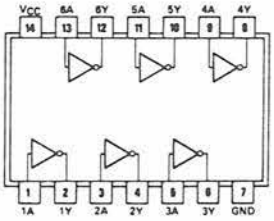
7402



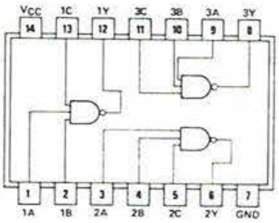
7404



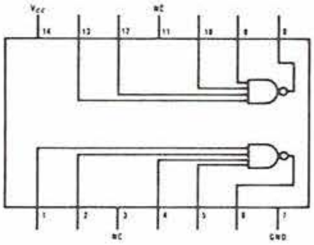
7406



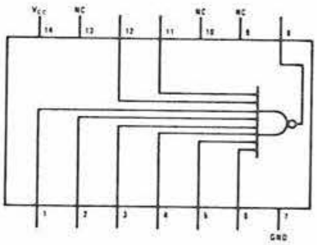
7410



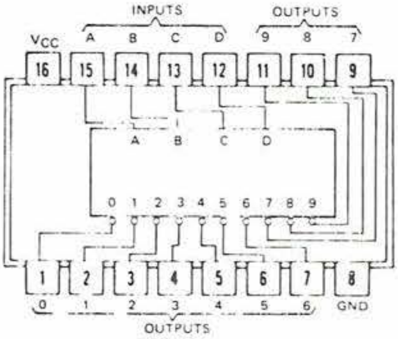
7420



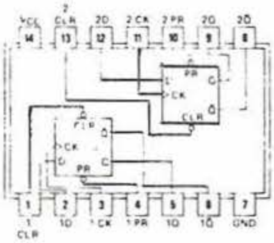
7430



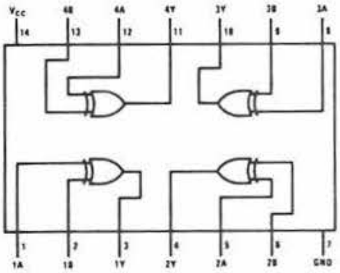
7442



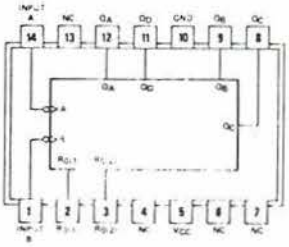
7474



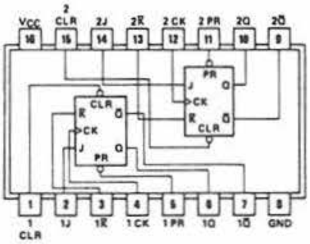
7486



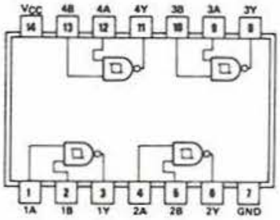
7493



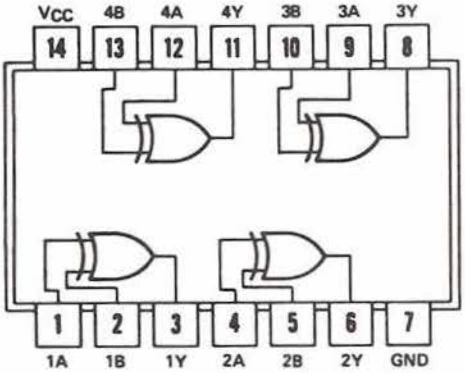
74109



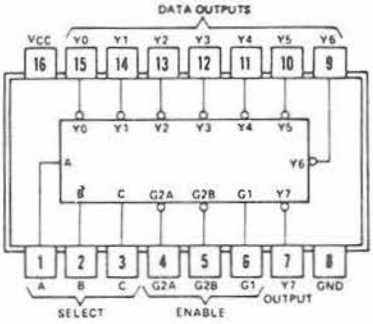
74132



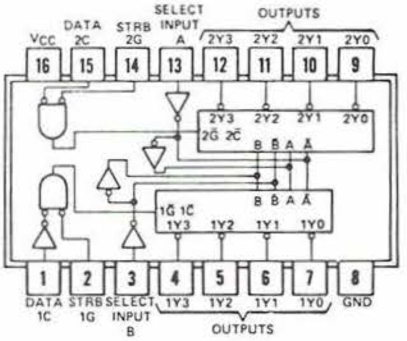
74136



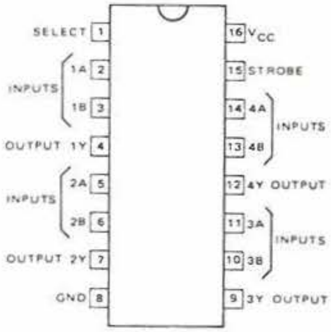
74138



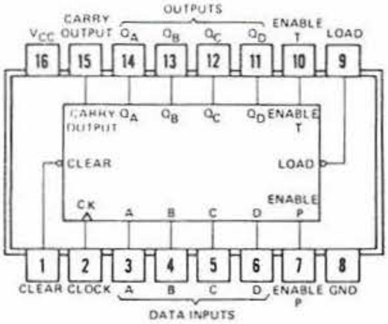
74155



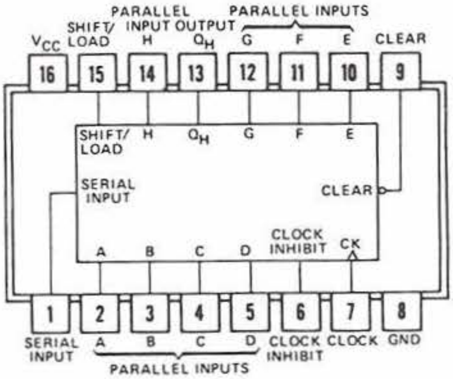
74157



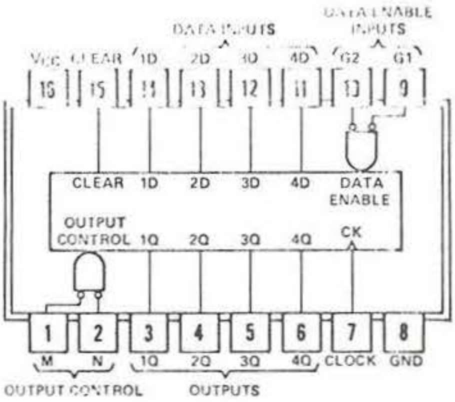
74163 or 93L16



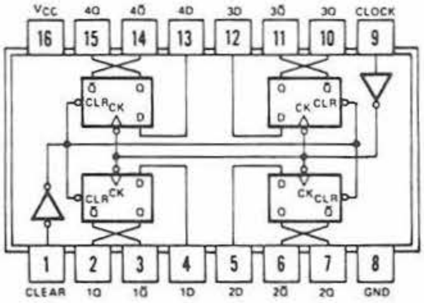
74166



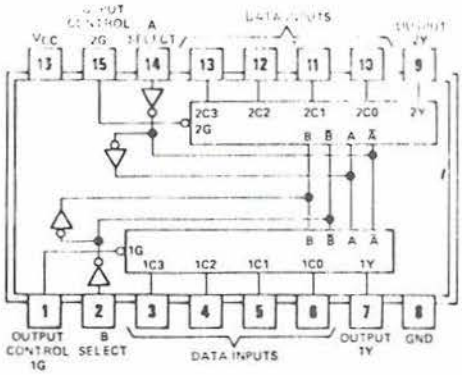
74173



74175



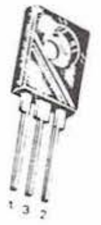
74253



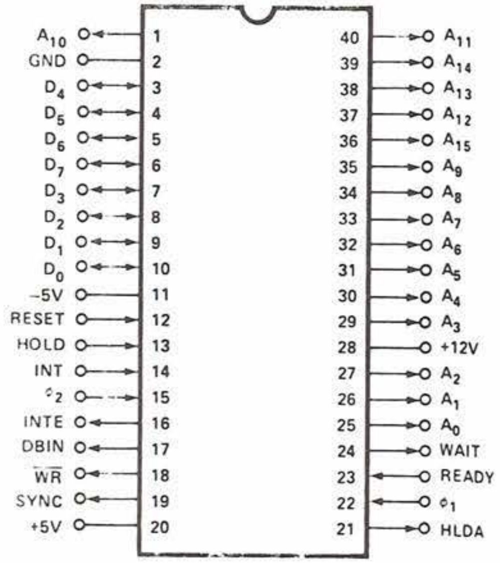
7812 and 7912

Pin 1 Input (Base)
Pin 2 Output (Emitter)
Pin 3 Ground (Collector)

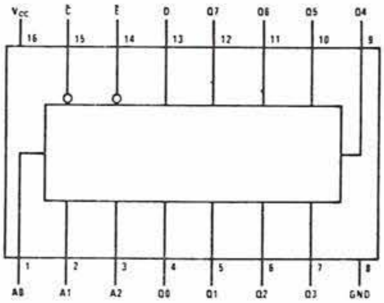
Heat sink surface connected to pin 3



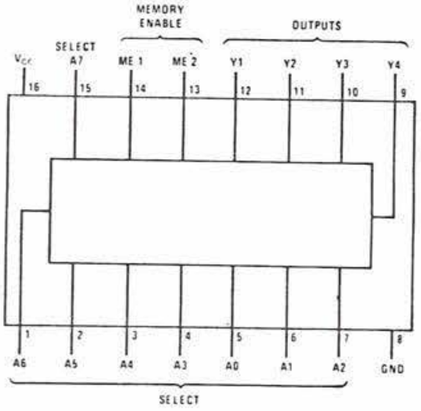
8080



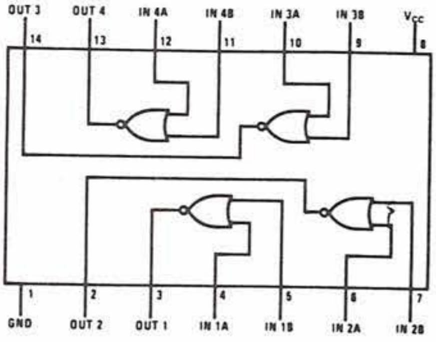
8334



8574



8836



REPRINTED FROM OCTOBER 1975 BYTE WITH PERMISSION

Television Interface

Anyone with a bunch of memory circuits, control logic and a wire wrap gun can whip up a digital video generator with TTL output levels. The problem as I see it is to get that digital video signal into a form 'that the TV set can digest. The care and feeding of digital inputs to the TV set is the subject of Don Lancaster's contribution to BYTE 2 – an excerpt from his forthcoming book, *TV Typewriter Cookbook,* to be published by Howard W. Sams, Indianapolis, Indiana.

. . . CARL

by
Don Lancaster
Box 1112
Parker AZ 85344

We can get between a TV typewriter and a television style display system either by an rf modulator or a direct video method.

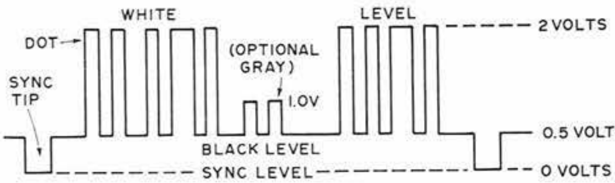
In the rf modulator method, we build a miniature, low power, direct wired TV transmitter that clips onto the antenna terminals of the TV set. This has the big advantage of letting you use any old TV set and ending up with an essentially free display that can be used just about anywhere. No set modifications are needed, and you have the additional advantage of automatic safety isolation and freedom from hot chassis shock problems.

There are two major restrictions to the rf modulator method. The first of these is that transmitters of this type must meet certain exactly spelled out FCC regulations and that system type approval is required. The second limitation is one of bandwidth. The best you can possibly hope for is 3.5 MHz for black and white and only 3 MHz for color, and many economy sets will provide far less. Thus, long character line lengths, sharp characters, and premium (lots of dots) character generators simply aren't compatible with clip-on rf entry.

In the direct video method, we enter the TV set immediately following its video detector but before sync is picked off. A few premium TV sets and all monitors already have a video input directly available, but these are still expensive and rare. Thus, you usually have to modify your TV set, either adding a video input and a selector switch or else dedicating the set to exclusive TV typewriter use. Direct video eliminates the bandwidth restrictions provided by the tuner, i-f strip, and video detector filter. Response can be further extended by removing or shorting the 4.5 MHz sound trap and by other modifications to provide us with longer line lengths and premium characters. No FCC approval is needed, and several sets or monitors are easily driven at once without complicated distribution problems.

There are two limitations to the direct video technique. One is that the set has to be modified to provide direct video entry. A second, and far more severe, restriction, is that many television sets are "hot chassis" or ac-dc sets with one side of their chassis connected to the power line. **These sets introduce a severe shock hazard and cannot be used as TV typewriter video entry displays unless some isolation technique is used with them.** If the TV set has a power transformer, there is usually no hot chassis problem. Transistor television sets and IC sets using no vacuum tubes tend to have power transformers, as do older premium tube type sets. All others (around half the sets around today) do not.

Fig. 1. Standard video interface levels. (Source impedance = 72 or 100 Ohms.)



Direct Video Methods

With either interface approach, we usually start by getting the dot matrix data, blanking, cursor, and sync signals together into one composite video signal whose form is useful to monitors and TV sets. A good set of standards is shown in Fig. 1. The signal is dc coupled and always positive going. Sync tips are grounded and blacker than black. The normal open circuit black level is positive by one-half a volt, and the white level is two volts positive. In most TV camera systems, intermediate levels between the half volt black level and the two volt white level will be some shade of gray, proportionately brighter with increasing positive voltage. With most TV typewriter systems, only the three states of zero volts (sync), half a volt (black), and two volts (white dot) would be used. One possible exception would be an additional one volt dot level for a dim but still visible portion of a message or a single word.

The usual video source impedance is either 72 or 100 Ohms. Regardless of how far we travel with a composite video output, some sort of shielding is absolutely essential.

For short runs from board to board or inside equipment, tightly twisted conductors should be OK, as should properly guarded PC runs. Fully shielded cables should be used for interconnections between the TVT and the monitor or TV set, along with other long runs. As long as the total cable capacitance is less than 500 pF or so (this is around 18 feet of RG178-U miniature coax), the receiving end of the cable need not be terminated in a 72 or 100 Ohm resistor. When terminated cable systems are in use for long line runs or multiple outputs, they should be arranged to deliver the signal levels of Fig. 1 at their output under termination. Generally, terminated cable systems should be avoided as they need extra in the way of drivers and supply power.

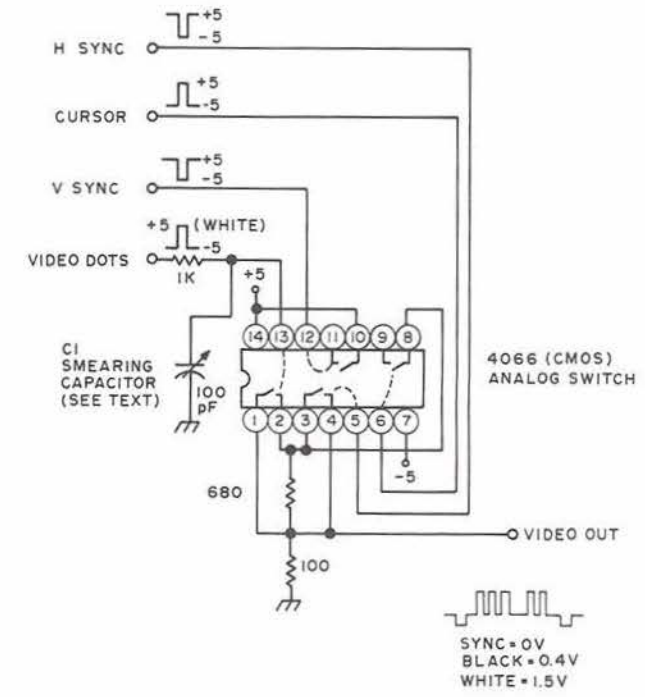
The exact width of the horizontal and vertical sync pulses isn't usually too important, so long as the shape and risetime of these pulses are independent of position control settings and power supply variations. One exception to this is when you're using a color receiver and a color display. Here, the horizontal sync pulse should be held closely to 5.1 microseconds, so the receiver's color burst sampling does in fact intercept a valid color burst. More on this later.

Intentional Smear

Fig. 2 shows us a typical composite video driver using a 4066 quad analog switch. It gives us a 100 Ohm output impedance and the proper signal levels. Capacitor C1 is used to purposely reduce the video rise and fall times. It is called a smearing capacitor.

Why would we want to further reduce the bandwidth and response of a TV system that's already hurting to begin with? In the case of a quality video monitor, we wouldn't. But if we're using an ordinary run-of-the-mill TV set, particularly one using rf entry, this capacitor can very much improve the display legibility and contrast. Why?

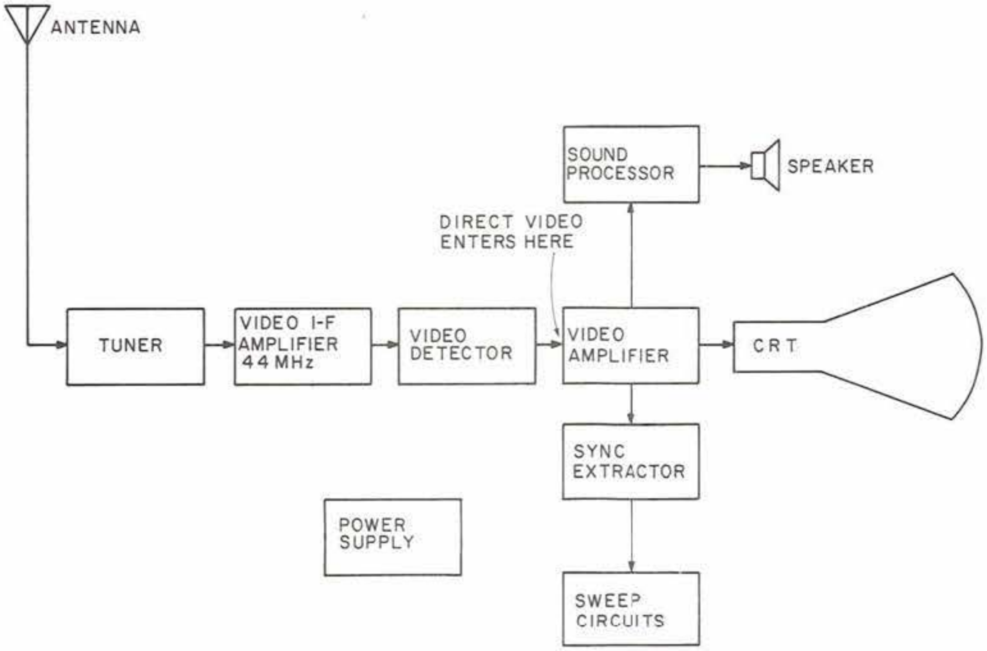
Fig. 2. Analog switch combiner generates composite video.



Because we are interested in getting the most legible character of the highest contrast we can. This is not necessarily the one having the sharpest dot rise and fall times. Many things interact to determine the upper video response of a TV display. These include the tuner settings and the i-f response and alignment, the video detector response, video peaking, the sound trap setting, rf cable reflections, and a host of other responses. Many of these stages are underdamped and will ring if fed too sharp a risetime input, giving us a ghosted, shabby, or washed out character. By reducing the video bandwidth going into the system, we can move the dot matrix energy lower in frequency, resulting in cleaner characters of higher contrast.

For most TV displays, intentional smearing will help the contrast, legibility, and overall appearance. The ultimate limit to this occurs when the dots overlap and become illegible. The

Fig. 3. Block diagram of typical B and W television.



optimum amount of intentional smear is usually the value of capacitance that is needed to just close the inside of a "W" presented to the display.

Adding a Video Input

Video inputs are easy to add to the average television set, provided you follow some reasonable cautions. First and foremost, you must have an accurate and complete schematic of the set to be modified, preferably a Sams Photofact or something similar. The first thing to check is the power supply on the set. If it has a power transformer and has the chassis properly safety isolated from the power line, it's a good choice for a TVT monitor. This is particularly true of recent small screen, solid state portable TV sets. On the other hand, if you have a hot chassis type with one side of the power line connected to the chassis, you should avoid its use if at all possible. If you must use this type of set, be absolutely certain to use one of the safety techniques outlined later in Fig. 8.

A block diagram of a typical TV set appears in Fig. 3. UHF or VHF signals picked up by the tuner are downconverted in frequency to a video i-f frequency of 44 MHz and then filtered and amplified. The output of the video i-f is transformer coupled to a video detector, most often a small signal germanium diode. The video detector output is filtered to remove the carrier and then routed to a video amplifier made up of one or more tubes or transistors.

At some point in the video amplification, the black and white signal is split three ways. First, a reduced bandwidth output routes sync pulses to the sync separator stage to lock the set's horizontal and vertical scanning to the video. A second bandpass output sharply filtered to 4.5 MHz extracts the FM sound subcarrier and routes this to a sound i-f amplifier for further processing. The third output is video, which is strongly amplified and then capacitively coupled to the cathode of the picture tube.

The gain of the video amplifier sets the contrast of the display, while the bias setting on the cathode of the picture tube (with respect to its grounded control grid) sets the display brightness. Somewhere in the video amplifier, further rejection of the 4.5 MHz sound subcarrier is usually picked up to minimize picture interference. This is called a sound trap. Sound traps can be a series resonant circuit to ground, a parallel resonant circuit in the video signal path, or simply part of the transformer that is picking off the sound for more processing.

The video detector output is usually around 2 volts peak to peak and usually subtracts from a white level bias setting. The stronger the signal, the more negative the swing, and the blacker the picture. Sync tips are blacker than black, helping to blank the display during retrace times.

Fig. 4 shows us the typical video circuitry of a transistor black and white television. Our basic circuit consists of a diode detector, a unity gain emitter follower, and a variable gain video output stage that is capacitively coupled to the picture tube. The cathode bias sets the brightness, while the video gain sets the contrast. Amplified signals for sync and sound are removed from the collector of the video driver by way of a 4.5 MHz resonant transformer for the sound and a low pass filter for the sync. A parallel resonant trap set to 4.5 MHz eliminates sound interference. Peaking coils on each stage extend the bandwidth by providing higher impedances and thus higher gain to high frequency video signals.

Note particularly the biasing of the video driver. A bias network provides us with a stable source of 3 volts. In the absence of input video, this 3 volts sets the white level of the display, as well as establishing proper bias for both stages. As an increasing signal appears at the last video output transformer, it is negatively rectified by the video detector, thus lowering the 3 volts proportionately. The stronger the signal, the blacker the picture. Sync will be the strongest of all, giving us a blacker than black bias level of only one volt.

The base of our video driver has the right sensitivity we need for video entry, accepting a maximum of a 2 volt peak to peak signal. It also has the right polarity, for a positive going bias level means a whiter picture. *But, an unmodified set is already biased to the white level, and if we want to enter our own video, this bias must be shifted to the black level.*

We have a choice in any TV of direct or ac coupling of our input video. Direct coupling is almost always better as it eliminates any

Fig. 4. Typical video circuitry of transistor B and W TV set.

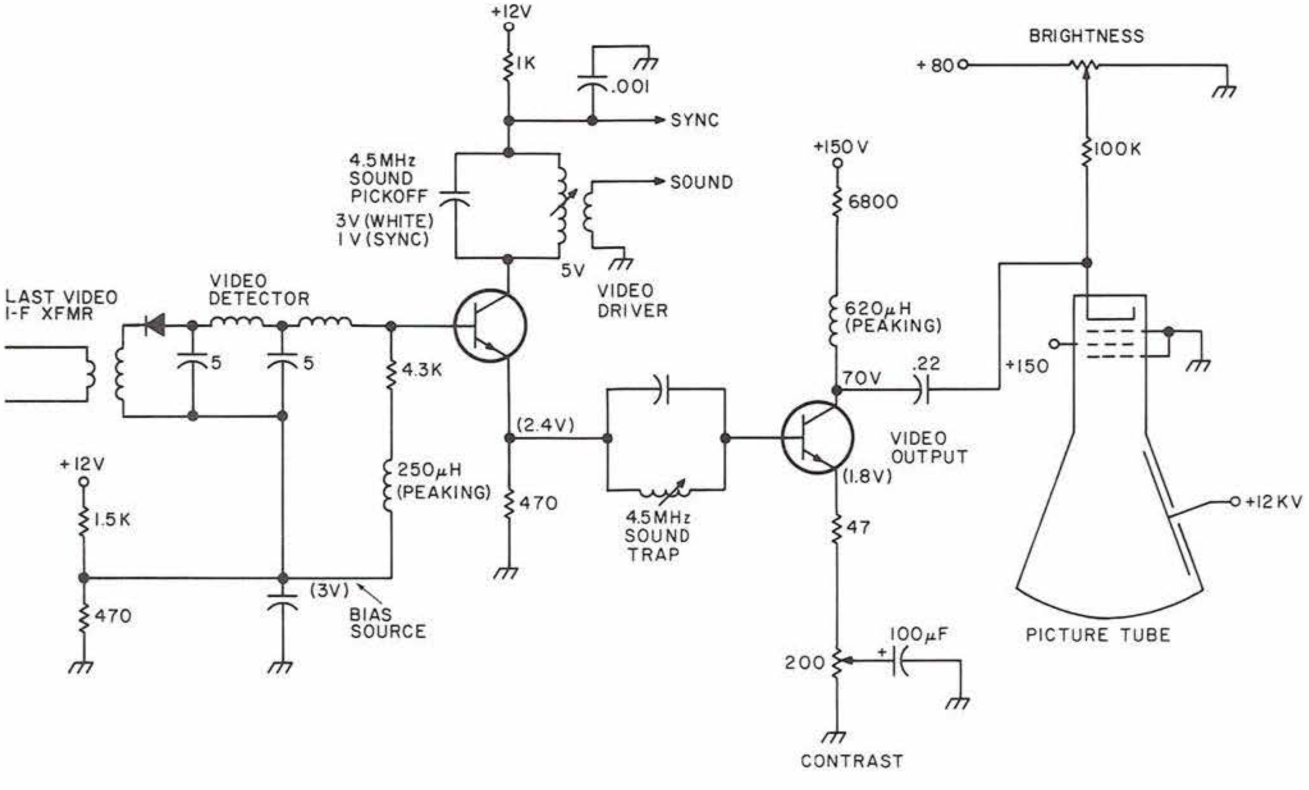
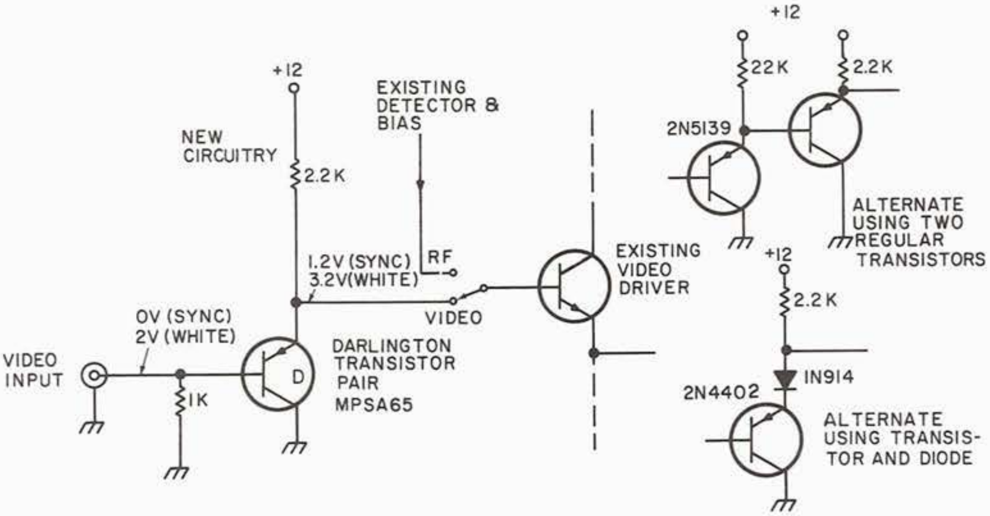


Fig. 5. Direct coupled video uses 1.2 volt offset of Darlington transistor as bias.



shading effects or any change of background level as additional characters are added to the screen. Fig. 5 shows how we can direct couple our video into a transistor black and white set. We provide a video input, usually a BNC or a phono jack, and route this to a PNP Darlington transistor or transistor pair, borrowing around 5 mils from the set's +12 volt supply. This output is routed to the existing video driver stage through a SPDT switch that either picks the video input or the existing video detector and bias network.

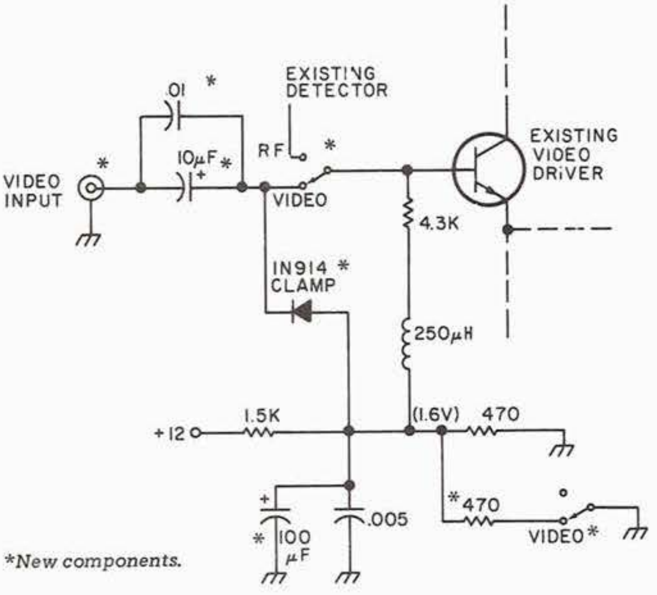
The two base-emitter diode drops in our Darlington transistor add up to a 1.2 volt positive going offset; so, in the absence of a video input or at the base of a sync tip, the video driver is biased to a blacker than black sync level of 1.2 volts. With a white video input of 2 volts, the video driver gets biased to its usual 3.2 volts of white level. Thus, our input transistor provides just the amount of offset we need to match the white and black bias levels of our video driver. Note that the old bias network is on the other side of the switch and does nothing in the video position.

Two other ways to offset our video input are to use two ordinary transistors connected in the Darlington configuration, or to use one transistor and a series diode to pick up the same amount of offset, as shown in Fig. 5. If more or less offset is needed, diodes or transistors can be stacked up further to pick up the right amount of offset.

The important thing is that the video driver ends up with the same level for white bias and for black bias in either position of the switch.

Ac or capacitively coupled video inputs should be avoided. Fig. 6 shows a typical circuit. The TV's existing bias network is lowered in voltage by adding a new parallel resistor to ground to give us a voltage that is 0.6 volts more positive than the blacker than black sync tip voltage. For instance, with a 3 volt white level, and 2 volt peak to peak video, the sync tip voltage would be 1 volt; the optimum bias is then 1.6 volts. Input video is capacitively coupled by a fairly large electrolytic capacitor in parallel with a good high frequency capacitor. This provides for a minimum of screen shading and still couples high frequency signals properly. A clamping diode constantly clamps the sync tips to their bias value, with the 0.6 volt drop of this diode being taken out by the extra 0.6 volts provided for in the bias network. This clamping diode automatically holds the sync tips to their proper value, regardless of the number of white dots in the picture. Additional bypassing of the bias network by a large electrolytic may be needed for proper operation of the clamping diode, as shown in Fig. 6. Note that our bias network is used in both switch positions – its level is shifted as needed for the direct video input.

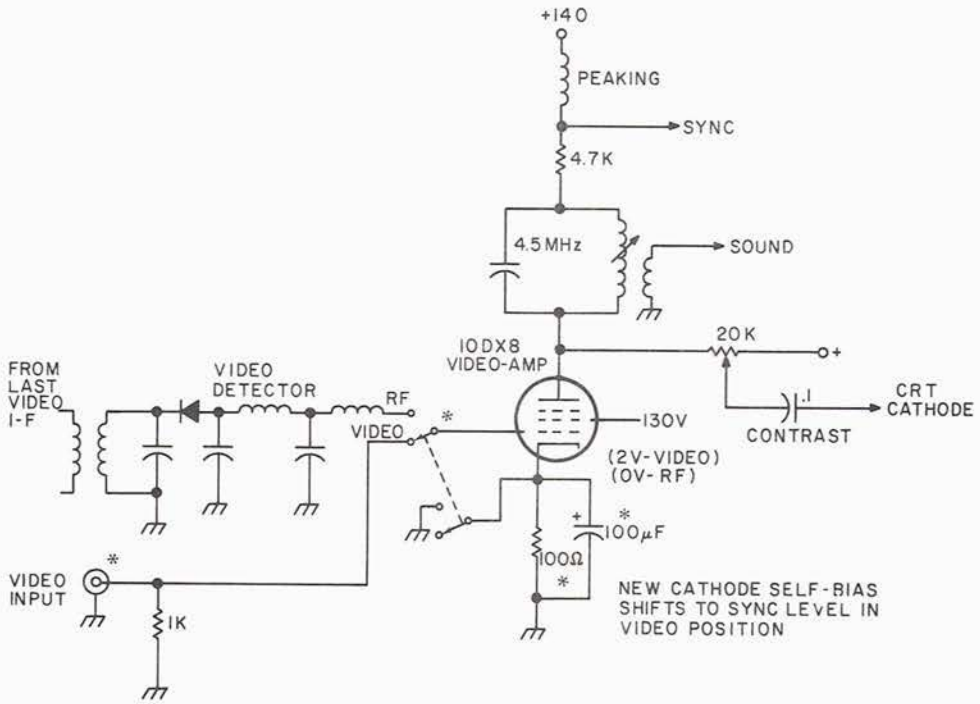
Fig. 6. Ac coupled video needs shift of bias to black level plus a clamping diode.



*New components.

Tube type sets present about the same interface problems as the solid state versions do. Fig. 7 shows a typical direct coupled tube interface. In the unmodified

Fig. 7. Direct coupled video added to tube type B and W television.



**New components.*

circuit, the white level is zero volts and the sync tip black level is minus two volts. If we can find a negative supply (scarce in tube type circuits), we could offset our video in the negative direction by two volts to meet these bias levels.

Instead of this, it is usually possible to self bias the video amplifier to a cathode voltage of +2 volts. This is done by breaking the cathode to ground connection and adding a small resistor (50 to 100 Ohms) between cathode and ground to get a cathode voltage of +2 volts. Once this value is found, a heavy electrolytic bypass of 100 microfarads or more is placed in parallel with the resistor. Switching then grounds the cathode in the normal rf mode and makes it +2 volts in the video entry mode.

In the direct video mode, a sync tip grounded input presents zero volts to the grid, which is self biased minus two volts with respect to the cathode. A white level presents +2 volts to the grid, which equals zero volts grid to cathode.

Should there already be a self bias network on the cathode, it is increased in value as needed to get the black rather than white level bias in the direct video mode.

Hot Chassis Problems

There is usually no shock hazard when we use clip-on rf entry or when we use a direct video jack on a transformer-powered TV. A very severe shock hazard can exist if we use direct video entry with a TV set having one side of the power line connected to the chassis. Depending on which way the line cord is plugged in, there is a 50-50 chance of the hot side of the power line being connected directly to the chassis.

Hot chassis sets, particularly older, power hungry tube versions, should be avoided entirely for direct video entry. If one absolutely must be used, some of the suggestions of Fig. 8 may ease the hazard. These include using an isolation transformer, husky back-to-back filament transformers, three wire power systems, optical coupling of the video input, and total package isolation. Far and away the best route is simply never to attempt direct video entry onto a hot chassis TV.

Making the Conversion

Fig. 9 sums up how we modify a TV for direct video entry. Always have a complete schematic on hand, and use a transformer style TV set if at all possible. Late models, small screen, medium to high quality solid state sets are often the best display choice. Avoid using junk sets, particularly very old ones. Direct coupling of video is far preferable to ac capacitor coupling. Either method has to maintain the black and white bias levels on the first video amplifier stage. A shift of the first stage quiescent bias from normally white to normally black is also a must. Use short, shielded leads between the video input jack and the rest of the circuit. If a changeover switch is used, keep it as close to the rest of the video circuitry as you possibly can.

Extending Video and Display Bandwidth

By using the direct video input route, we eliminate any bandwidth and response restrictions of an rf modulator, the tuner, video i-f strip, and the video detector filter. Direct video entry should bring us to a 3 MHz bandwidth for a color set and perhaps 3.5 MHz for a black and white model, unless we are using an extremely bad set. The resultant 6 to 7 million dot per second rate is adequate for short character lines of 32, 40, and possibly 48 characters per line. But the characters will smear and be illegible if we try to use longer line lengths and premium (lots of dots) character generators on an ordinary TV. Is there anything we can do to the set to extend the video bandwidth and display response for these longer line lengths?

In the case of a color TV, the answer is probably no. The video response of a color set is limited by an essential delay line and an essential 3.58 MHz trap. Even if we were willing to totally separate the chrominance and luminance channels, we'd still be faced with an absolute limit set by the number of holes per horizontal line in the shadow mask of the tube. This explains why video color displays are so expensive and so rare. Later on, we'll look at what's involved in adding color to the shorter line lengths.

With a black and white TV, there is often quite a bit

Fig. 8. Getting Around a Hot Chassis Problem.

Hot chassis problems can be avoided entirely by using only transformer-powered TV circuits or by using clip-on rf entry. If a hot chassis set must be used, here are some possible ways around the problem:

1. Add an isolation transformer.

A 110 volt to 110 volt isolation transformer whose wattage exceeds that of the set may be used. These are usually expensive, but a workable substitute can be made by placing two large surplus filament transformers back to back. For instance, a pair of 24 volt, 4 Amp transformers can handle around 100 Watts of set.

2. Use a three wire system with a solid ground.

Three prong plug wiring, properly polarized, will force the hot chassis connection to the cold side of the power line. This protection is useful only when three wire plugs are used in properly wired outlets. A severe shock hazard is reintroduced if a user elects to use an adaptor or plugs the system into an unknown or improperly wired outlet. The three wire system should **NOT** be used if anyone but yourself is **ever** to use the system.

3. Optically couple the input video.

Light emitting diode-photocell pairs are low in cost and can be used to optically couple direct video, completely isolating the video input from the hot chassis. Most of these optoelectronic couplers do not have enough bandwidth for direct video use; the Litronix IL-100 is one exception. Probably the simplest route is to use two separate opto-isolators, one for video and one for sync, and then recombine the signals inside the TV on the hot side of the circuit.

4. Use a totally packaged and sealed system.

If you are only interested in displaying messages and have no other input/output devices, you can run the entire circuit hot chassis, provided everything is sealed inside one case and has no chassis-to-people access. Interface to teletypes, cassettes, etc., cannot be done without additional isolation, and servicing the circuit presents the same shock hazards that servicing a hot chassis TV does.

we can do to present long lines of characters, depending on what set you start out with and how much you are willing to modify the set.

The best test signal you can use for bandwidth extension is the dot matrix data you actually want to display, for the frequency response, time delay, ringing, and overshoot all get into the act. What we want to end up with is a combination that gives us reasonably legible characters.

A good oscilloscope (15 MHz or better bandwidth) is very useful during bandwidth extension to show where the signal loses its response in the circuit. At any time during the modification process, there is usually one response bottleneck. This, of course, is what should be attacked first. Obviously the better a TV you start with, the easier will be the task. Tube type gutless wonders, particularly older ones, will be much more difficult to work with than with a modern, small screen, quality solid state portable.

Several of the things we can do are watching the control settings, getting rid of the sound trap, minimizing circuit strays, optimizing spot size, controlling peaking, and shifting to higher current operation. Let's take a look at these in turn.

Control Settings

Always run a data display at the lowest possible contrast and using only as much brightness as you really need. In many circuits, low contrast means a lower video amplifier gain, and thus less of a gain-bandwidth restriction.

Eliminate the Sound Trap

The sound trap adds a notch at 4.5 MHz to the video response. If it is eliminated or switched out of the circuit, a wider video bandwidth automatically

Fig. 9. How to Add a Direct Video Input to a TV Set.

1. Get an accurate and complete schematic of the set – either from the manufacturer's service data or a Photofact set. **Do not try adding an input without this schematic!**

2. Check the power supply to see if a power transformer is used. If it is, there will be no shock hazard, and the set is probably a good choice for direct video use. If the set has one side of the power line connected to the chassis, a severe shock hazard exists, and one of the techniques of Fig. 8 should be used. **Avoid the use of hot chassis sets.**

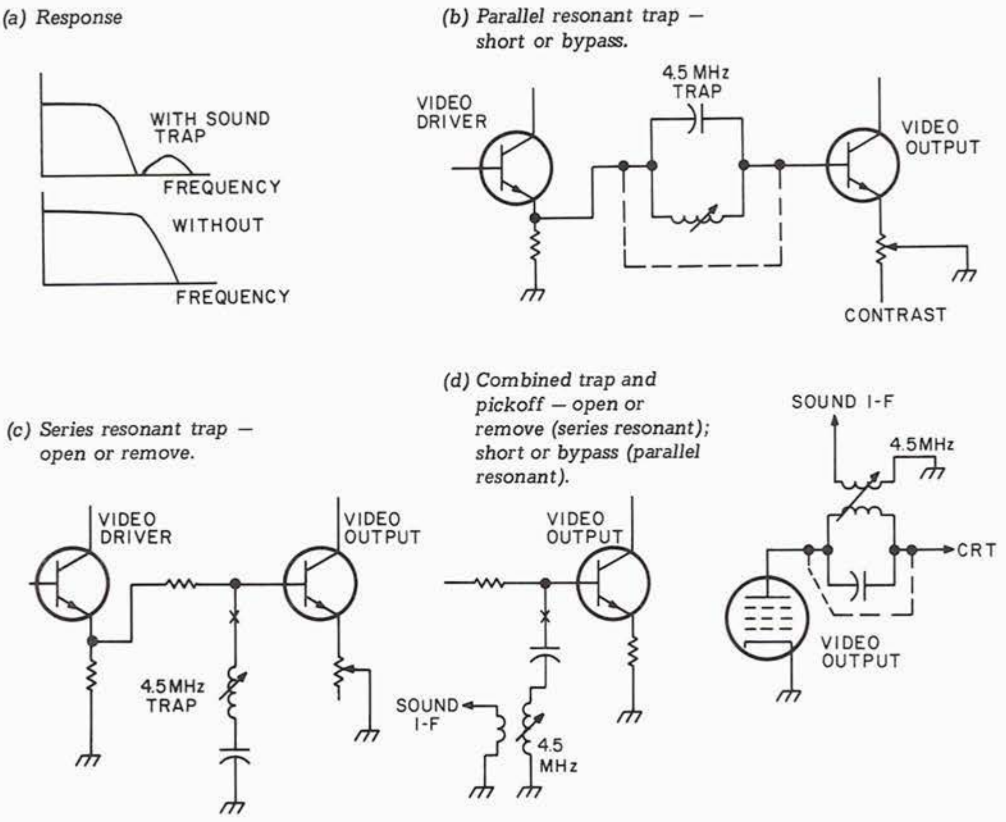
3. Find the input to the first video amplifier stage. Find out what the white level and sync level bias voltages are. The marked or quiescent voltage is usually the white level; sync is usually 2 volts less. A transistor TV will typically have a +3 volt white level and a +1 volt sync level. A tube type TV will typically have a zero volt white level and a -2 volt sync level.

4. Add a changeover switch using minimum possible lead lengths. Add an input connector, either a phono jack or the premium BNC type connector. Use shielded lead for interconnections exceeding three inches in length.

5. Select a circuit that couples the video and biases the first video amplifier stage so that the white and sync levels are preserved. For transistor sets, the direct coupled circuits of Fig. 5 may be used. For tube sets, the circuit of Fig. 7 is recommended. Avoid the use of ac coupled video inputs as they may introduce shading problems and changes of background as the screen is filled.

6. Check the operation. If problems with contrast or sync tearing crop up, recheck and adjust the white and sync input levels to match what the set uses during normal rf operation. Note that the first video stage must be biased to the **white** level during rf operation and to the **sync** level for direct video use. The white level is normally two volts more positive than the sync level.

Fig. 10. Removing the sound trap can extend video bandwidth.



(a) Response

(b) Parallel resonant trap – short or bypass.

(c) Series resonant trap – open or remove.

(d) Combined trap and pickoff – open or remove (series resonant); short or bypass (parallel resonant).

results. Fig. 10 shows us the response changes and the several positions for this trap. Generally, series resonant traps are opened and parallel resonant traps are shorted or bypassed through suitable switching or outright elimination. The trap has to go back into the circuit if the set is ever again used for ordinary program reception. Sometimes simply backing the slug on the trap all the way out will improve things enough to be useful.

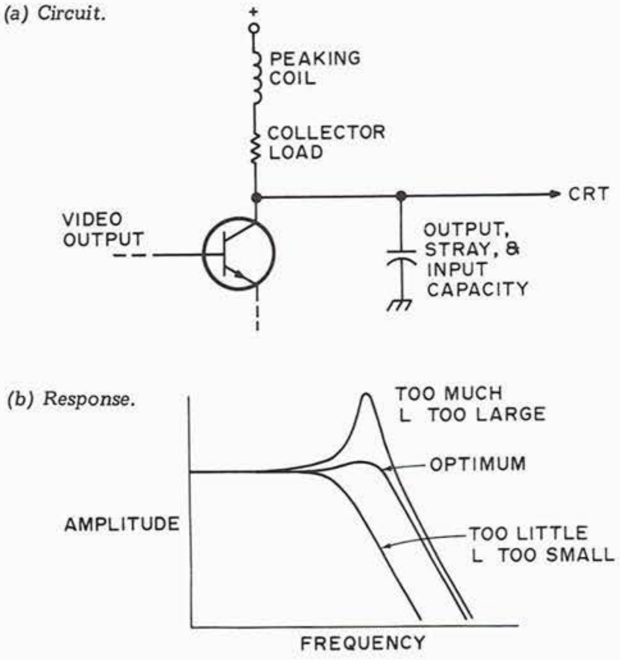
Minimizing Strays

One of the limits of the video bandwidth is the stray capacitance both inside the video output stage and in the external circuitry. If the contrast control is directly in the signal path and if it has long leads going to it, it may be hurting the response. If you are using the TV set exclusively for data display, can you rearrange the control location and simplify and shorten the video output to picture tube interconnections?

Additional Peaking

Most TV sets have two peaking networks. The first of these is at the video detector output and compensates for the vestigial sideband transmission signal that makes sync and other low frequency signals double the amplitude of the higher frequency ones. The second of these goes to the collector or plate of the video output stage and raises the circuit impedance and thus the effective gain for very high frequencies. Sometimes you can alter this second network to favor dot presentations. Fig. 11 shows a typical peaking network and the effects of too little or too much peaking. Note that the stray capacitance also enters into the peaking, along with the video amplifier output capacitance and the picture tube's input capacitance. Generally, too little peaking will give you low contrast dots, while too much will give you sharp dots, but will run dots together and shift the more continuous portions of the characters objectionably. Peaking is changed by increasing or decreasing the series inductor from its design value.

Fig. 11. Adjusting the peaking coil can extend video response.



(a) Circuit.

(b) Response.

Running Hot

Sometimes increasing the operating current of the video output stage can increase the system bandwidth – IF this stage is in fact the limiting response, IF the power supply can handle the extra current, IF the stage isn't already parked at its gain-bandwidth peak, and IF the extra heat can be gotten rid of without burning anything up. Usually, you can try adding a resistor three times the plate or collector load resistor in parallel, and see if it increases bandwidth by 1/3. Generally, the higher the current, the wider the bandwidth, but watch carefully any dissipation limits. Be sure to provide extra ventilation and additional heatsinking, and check the power supply for unhappiness as well. For major changes in operating current, the emitter resistors and other biasing components should also be proportionately reduced in value.

Spot Size

Even with excellent video bandwidth, if you have an out-of-focus, blooming, or changing spot size, it can completely mask character sharpness. Spot size ends up the ultimate limit to resolution, regardless of video bandwidth.

Once again, brightness and contrast settings will have a profound effect, with too much of either blooming the spot. Most sets have a focus jumper in which ground or a positive voltage is selected. You can try intermediate values of voltage for maximum sharpness. Extra power supply filtering can sometimes minimize hum and noise modulation of the spot.

Anything that externally raises display contrast will let you run with a smaller beam current and a sharper spot. Using circularly polarized filters, graticule masks, or simple colored filters can minimize display washout from ambient lighting. Fig. 12 lists several sources of material for contrast improvement. Much of this is rather expensive, with pricing from $10 to $25 per square foot being typical. Simply adding a hood and positioning the display away from room lighting will also help and is obviously much cheaper.

Fig. 12. Contrast Enhancing Filter Materials.

Circularly polarized filters:

Polaroid Corp.
Cambridge MA 02139

Anti-reflection filters:

Panelgraphic Corp.
10 Henderson Dr.
West Caldwell NJ 07006

Light control film:

3M Visual Products Div.
3M Center
St. Paul MN 55101

Acrylic plexiglas filter sheets:

Rohm and Haas
Philadelphia PA 19105

Fig. 13. Standard rf interface levels. Impedance = 300Ω. Carrier frequency per Fig. 14.

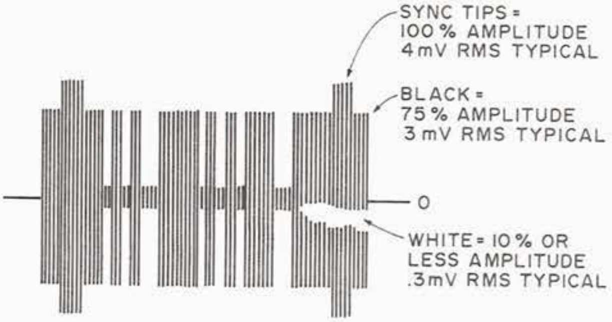


Fig. 14. Television Picture Carrier Frequencies.

Channel 2	**55.25 MHz**
Channel 3	**61.25 MHz**
Channel 4	**67.25 MHz**
Channel 5	**77.25 MHz**
Channel 6	**83.25 MHz**

Direct Rf Entry

If we want the convenience of a "free" display, the freedom from hot chassis problems, and "use it anywhere" ability, direct rf entry is the obvious choice. Its two big limitations are the need for FCC type approval, and a limited video bandwidth that in turn limits the number of characters per line and the number of dots per character.

An rf interface standard is shown in Fig. 13. It consists of an amplitude modulated carrier of one of the standard television channel video frequencies of Fig. 14. Channel 2 is most often used with a 55.250 MHz carrier frequency, except in areas where a local commercial Channel 2 broadcast is intolerably strong. Circuit cost, filtering problems, and stability problems tend to increase with increasing channel number.

The sync tips are the strongest part of the signal, representing 100% modulation, often something around 4 millivolts rms across a 300 Ohm line. The black level is 75% of the sync level, or about 3 millivolts for 4 millivolt sync tips. White level is less than 10% of maximum. Note that the signal is weakest when white and strongest when sync. This is the exact opposite of the video interface of Fig. 1.

Rf modulators suitable for clip-on rf entry TV typewriter use are called Class 1 TV Devices by the FCC. A Class 1 TV device is supposed to meet the rules and regulations summarized in Fig. 15.

Fig. 16 shows us a block diagram of the essential parts of a TV modulator. We start

Fig. 15. FCC Regulations on Class 1 TV Devices. More complete information appears in subpart H of Part 15 and subpart F of Part 2 of the Federal Communications Commission Rules and Regulations. *It is available at many large technical libraries.*

A Class 1 TV device generates a video modulated rf carrier of a standard television channel frequency. It is directly connected to the antenna terminals of the TV set.

The maximum rms rf voltage must be less than 6 millivolts using a 300 Ohm output line.

The maximum rf voltage on any frequency more than 3 MHz away from the operating channel must be more than 30 dB below the peak in-channel output voltage.

An antenna disconnect switch of at least 60 dB attenuation must be provided.

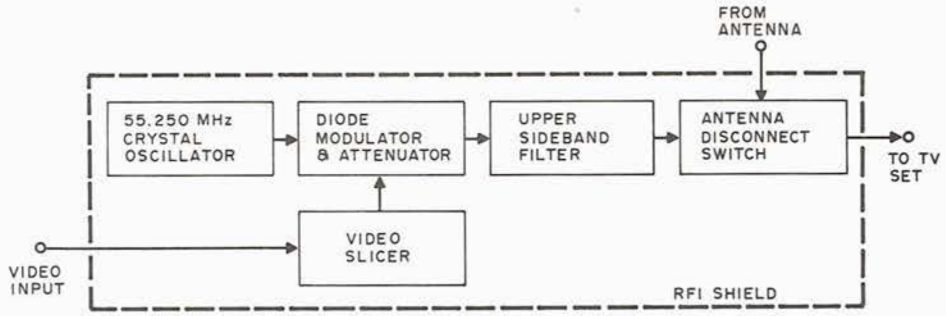
No user adjustments are permitted that would exceed any of the above specifications.

Residual rf radiation from case, leads and cabinet must be less than 15 microvolts per meter.

A Class 1 TV device must not interfere with TV reception.

Type approval of the circuit is required. A filing fee of $50 and an acceptance fee of $250 is involved.

Fig. 16. Block diagram of rf modulator.


FROM ANTENNA
55.250 MHz CRYSTAL OSCILLATOR
DIODE MODULATOR & ATTENUATOR
UPPER SIDEBAND FILTER
ANTENNA DISCONNECT SWITCH
TO TV SET
VIDEO SLICER
VIDEO INPUT
RFI SHIELD

with a stable oscillator tuned to one of the Fig. 14 frequencies. A crystal oscillator is a good choice, and low cost modules are widely available. The output of this oscillator is then amplitude modulated. This can be done by changing the bias current through a silicon small signal diode. One milliampere of bias current makes the diode show an ac and rf impedance of 26 Ohms. Half a mil will look like 52 Ohms, and so on. The diode acts as a variable resistance attenuator in the rf circuit, whose bias is set and changed by the video circuit.

Since diode modulators are non-linear, we can't simply apply a standard video signal to them and get a standard rf signal out. A differential amplifier circuit called a video slicer may be used to compensate for this non-linearity. The video slicer provides three distinct currents to the diode modulator. One of these is almost zero for the white level, while the other two provide the black and sync levels. A contrast control that sets the slicing level lets you adjust the sync tip height with respect to the black level. The video slicer also minimizes rf getting back into the video. An attenuator to reduce the size of the modulated signal usually follows the diode modulator.

An upper side band filter removes most of the lower sideband from the AM modulated output, giving us a vestigial sideband signal that stays inside the channel band limits. This same filter eliminates second harmonic effects and other spurious noise. The filter's output is usually routed to an antenna disconnect switch and the TV's antenna terminals. A special switch is needed to provide enough isolation.

Some of the actual circuitry involved is shown in Fig. 17. The video slicer consists of a pair of high gain, small signal NPN transistors, while the oscillator is a commercially available module.

Rf entry systems always must be direct coupled to the antenna terminals of the set and should never provide any more rf than is needed for a minimum snow-free picture. They should be permanently tuned to a single TV channel. Under no circumstances should an antenna or cable service hookup remain connected to the set during TVT use, nor should radiation rather than a direct rf cable connection ever be used.

Color Techniques

We can add a full color capability to a TV typewriter system fairly easily and cheaply — provided its usual black and white video dot rate is low enough in frequency to be attractively displayed on an ordinary color TV. Color may be used to emphasize portions of a message, to attract attention, as part of an electronic game, or as obvious added value to a graphics display. Color techniques work best on TV typewriter systems having a horizontal frequency very near 15,735 Hertz.

All we basically have to do is generate a subcarrier sine wave to add to the video output. The phase of this subcarrier (or its time delay) is shifted with respect to what the phase was immediately after each horizontal sync pulse to generate the various colors.

Fig. 18 shows us the differences between normal color and black and white operation. Black and white baseband video is some 4 MHz wide and has a narrow 4.5 MHz sound subcarrier. The video is amplitude modulated, while the sound is narrow band frequency

Fig. 17. Channel two oscillator, modulator, video slicer and attenuator. R sets output level.

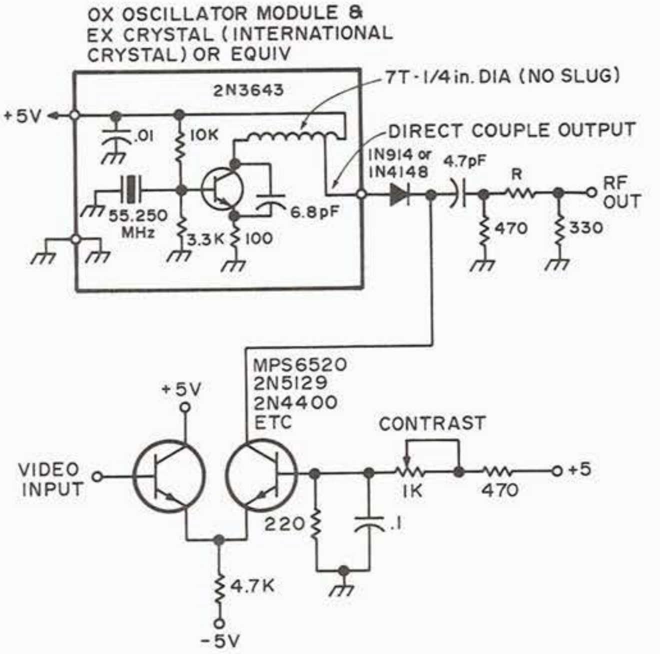
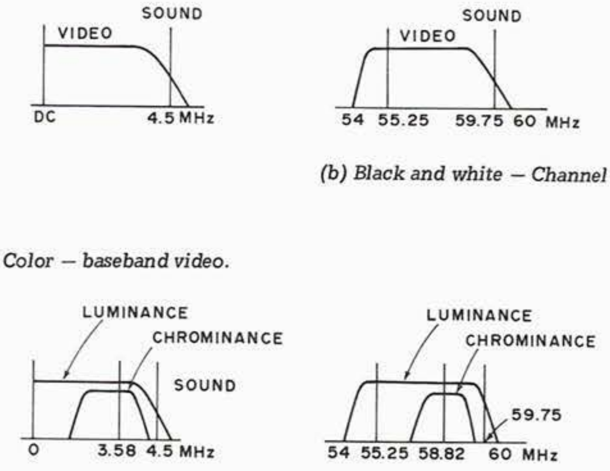


Fig. 18. Differences between color and black and white spectra.



(a) Black and white – baseband video.

(b) Black and white – Channel two rf.

(c) Color – baseband video.

(d) Color – Channel two rf.

modulated. This translates up to a 6 MHz rf channel with a vestigial lower sideband as shown in Fig. 18(b).

To generate color, we add a new pilot or subcarrier at a magic frequency of 3.579545 MHz – see Fig. 18(c). What was the video is now called the luminance, and is the same as the brightness in a black and white system. The new subcarrier and its modulation is called the chrominance signal and determines what color gets displayed and how saturated the color is to be.

Since the black and white information is a sampled data system that is scanned at the vertical and horizontal rates, there are lots of discrete holes in the video spectrum that aren't used. The color subcarrier is designed to stuff itself into these holes (exactly in a NSTC color system, and pretty much in a TVT display). Both chrominance and luminance signals use the same spectral space, with the one being where the other one isn't, overlapping comb style.

The phase or relative delay of the chrominance signal with respect to a reference determines the instantaneous color, while the amplitude of this signal with respect to the luminance sets the saturation of the color. Low amplitudes generate white or pastel shades, while high amplitudes of the chrominance signal produce saturated and deep colors.

At least eight cycles of a reference or burst color phase are transmitted immediately following each horizontal sync pulse as a timing reference, as shown in Fig. 19. The burst is around 25% of maximum amplitude, or about the peak to peak height of a sync pulse.

The TV set has been trained at the factory to sort all this out. After video detection, the set splits out the chrominance channel with a bandpass amplifier and then synchronously demodulates it with respect to an internal 3.58 MHz reference. The phase of this demodulation sets the color and the amplitude sets the saturation by setting the ratios of electron beam currents on the picture tube's red, blue and green guns.

Meanwhile, the luminance channel gets amplified as brightness style video. It is delayed with a delay line to make up for the time delay involved in the narrower band color processing channel. It is then filtered with two traps – the 4.5 MHz sound trap, and a new trap to get rid of any remaining 3.58 MHz color subcarrier that's left. The luminance output sets the overall brightness by modulating the cathodes of all three color guns simultaneously.

Just after each horizontal sync pulse, the set looks for the reference burst and uses this reference in a phase

Fig. 19 Adding a color reference burst to the back porch of the horizontal sync pulses.

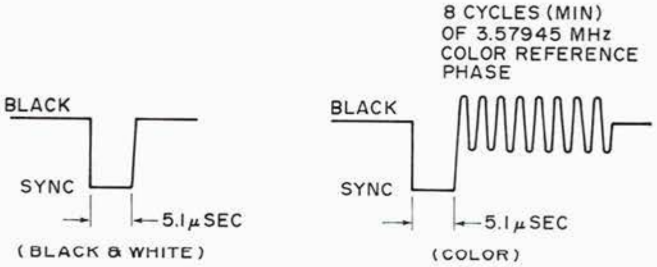


Fig. 20. Colors Are Generated by Delaying or Phase Shifting the Burst Frequency.

Color	Approximate Phase	Approximate Delay
Burst	0°	0
Yellow	15°	12 nanoseconds
Red	75°	58 nanoseconds
Magenta	135°	105 nanoseconds
Blue	195°	151 nanoseconds
Cyan	255°	198 nanoseconds
Green	315°	244 nanoseconds

detector circuit to keep its own 3.58 MHz reference locked to the version being transmitted.

Fig. 20 shows us the phase angles related to each color with respect to the burst phase. It also shows us the equivalent amount of delay we need for a given phase angle. Since we usually want only a few discrete colors, it's far easier to digitally generate colors simply by delaying the reference through gates or buffers, rather than using complex and expensive analog phase shift methods.

Strictly speaking, we should control both the chrominance phase and amplitude to be able to do both pastel and strongly saturated colors. But simply keeping the subcarrier amplitude at the value we used for the burst – around 25% of video amplitude – is far simpler and will usually get us useful results.

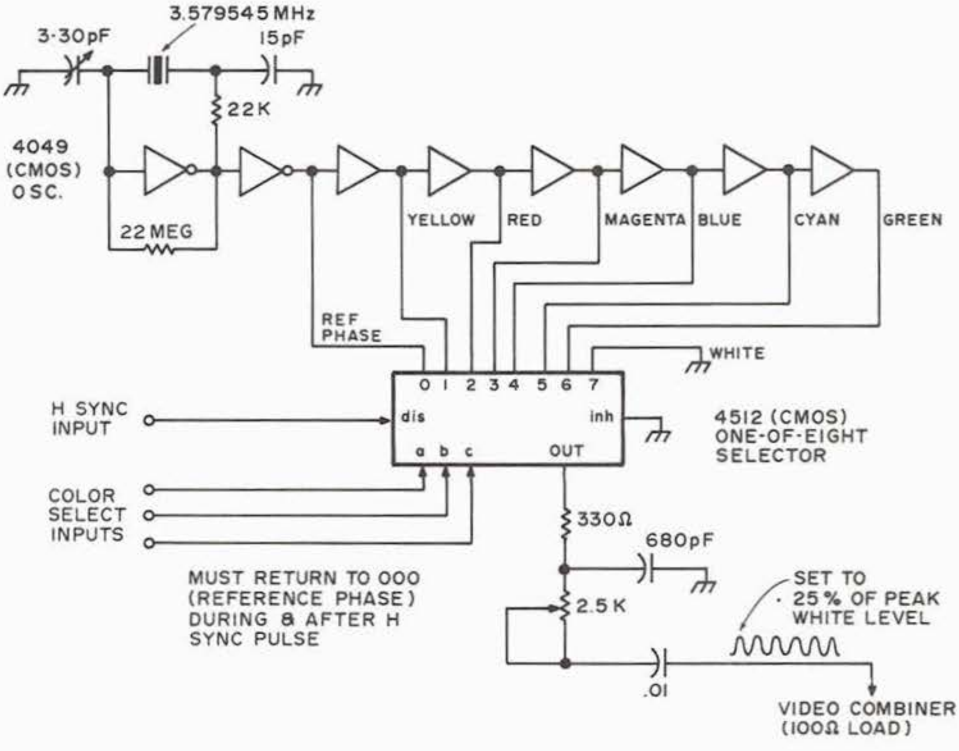
A circuit to add color to a TV typewriter is shown in Fig. 21. A 3.579545 MHz crystal oscillator drives a string of CMOS buffers that make up a digital delay line. The output delays caused by the propagation delay times in each buffer can be used as is, or can be trimmed to specific colors by varying the supply voltage.

The reference phase and the delayed color outputs go to a one-of-eight data selector. The data selector picks either the reference or a selected color in response to a code presented digitally to the three select lines. The logic that is driving this selector must return to the reference phase position (000) immediately before, during and for a minimum of a few microseconds after each horizontal sync pulse. This gives the set a chance to lock and hold onto the reference color burst.

The chrominance output from the data selector should be disabled for the duration of the sync pulses and any time a white screen display is wanted. The output chrominance signal is RC filtered to make it somewhat sinusoidal. It's then cut down in amplitude to around one-quarter the maximum video white level and is capacitively coupled to the 100 Ohm video output of Fig. 2 or otherwise summed into the video or rf modulator circuitry. For truly dramatic color effects, the amplitude and delay of the chrominance signal can be changed in a more complex version of the same circuit.

More information useful in solving television interface appears in the *Television Engineering Handbook*, by Donald Fink, and in various issues of the *IEEE Transactions on Consumer Electronics.*

Fig. 21. Color subcarrier generator. Hex buffer used as delay line. Use supply voltage variation on 4050 to trim colors.



Pinouts: Parallel Data Interface (PDI) as used on Processor Tech. Sol System — Sept. 30, 1976

MASTER UNIT-Male connector

J2 Pin #	Signal mnemonic	Signal name	J2 pin#	Signal mnemonic	Signal name
1	CG	Chassis Ground	14	US	Unit Select
2	SG	Signal Ground	15	OE	Output Enable
3	IE	Input Enable	16	$\overline{\text{XDR}}$	$\overline{\text{eXternal Device Ready}}$
4	$\overline{\text{DR}}$	$\overline{\text{Data Ready}}$	17	$\overline{\text{OL}}$	$\overline{\text{Output Load}}$
5	$\overline{\text{IAK}}$	$\overline{\text{Input Acknowledge}}$	18	OD7	Output Data,bit 7
6	ID7	Input Data, bit 7	19	OD6	Output Data, bit 6
7	ID6	Input Data, bit 6	20	OD5	Output Data, bit 5
8	ID5	Input Data, bit 5	21	OD4	Output Data, bit 4
9	ID4	Input Data, bit 4	22	OD3	Output Data, bit 3
10	ID3	Input Data, bit 3	23	OD2	Output Data, bit 2
11	ID2	Input Data, bit 2	24	OD1	Output Data, bit 1
12	ID1	Input Data, bit 1	25	OD0	Output Data, bit 0
13	ID0	Input Data, bit 0			

Pinouts: Serial Data Interface (SCI) as used on Processor Tech. Sol System

Female connector-DB25S

J1 pin#	Signal mnemonic	Signal name	J1 pin#	Signal mnemonic	Signal name
1	CG	Chassis Ground	8	CD	Carrier Detect
2	TD	Transmit Data	11	CLO	Current Loop Output
3	RD	Receive Data	12	LR1	Loop Receiver 1
4	RTS	Request To Send	13	LR2	Loop Receiver 2
5	CTS	Clear To Send	20	DTR	Data Terminal Ready
6	DSR	Data Set Ready	23	LCS	Loop Current Source
7	SG	Signal Ground			

Note 1: Many pins not specified here are used in EIA RS-232C specification. USE THEM WITH CAUTION.

Note 2: Terminals output on pins 2,4 & 20 and input on pins 3,5 & 6 for EIA type hookups. Modems and computer mainframes output on pins 3,5 & 6 and input on pins 2,4 & 20.

Note 3: Current loop hookups are the same for terminals, modems,mainframes.

J3 Keyboard Connector (between U64 and U65)

pin no.	Signal name	pin no.	Signal name
1	ground	11	ground
2	+5v	12	+5v
3	$\overline{\text{Kbd Data Ready}}$	13	$\overline{\text{Restart}}$
4	$\overline{\text{Break}}$	14	$\overline{\text{Local}}$
5	Kbd Data Ø	15	KBd Data 4
6	Kbd Data 1	16	KBd Data 5
7	Kbd Data 2	17	KBD Data 6
8	Kbd Data 3	18	KBD Data 7
9	+5v	19	+5v
10	ground	20	ground

J4 Display Expansion Connector (between U28, 29)

pin no.	Signal name	pin no.	Signal name
1	ground	11	ground
2	N.C.	12	N.C.
3	Char. addr. 4	13	Dot Clock, 14.318MHz
4	Character clock	14	Composite sync. out
5	Char. addr. Ø	15	TTL Serial Data Out
6	Char. addr. 1	16	Composite blanking out
7	Char. addr. 2	17	Scan advance out
8	Char. addr. 3	18	Char. addr. 5
9	N.C.	19	N.C.
10	ground	20	ground

J5 Personality Module Edge Connector

pin no.		Signal name	pin no.		Signal name
B15	B	Ground	A15		Ground
B14	O	+5VDC	A14		+5VDC
B13	T	Addr. 9	A13	T	Addr. Ø
B12	T	Addr. 8	A12	O	Addr. 4
B11	O	Addr. 7	A11	P	Addr. 3
B10	M	INT Bus Ø	A10		Addr. 2
B9		INT Bus 1	A9	R	Addr. 1
B8		INT Bus 2	A8	O	Addr. 5
B7	R	INT Bus 3	A7	W	Addr. 6
B6	O	INT Bus 4	A6		C4
B5	W	INT Bus 5	A5	P	CØ
B4		Program Ø	A4	I	INT Bus 6
B3	P	Program 1	A3		INT Bus 7
B2	I	Program 2	A2	N	-12VDC
B1	N	Program 3	A1	S	+12VDC
	S				

J6 Audio Out for CUTS Cassette Interface: Mini-phone jack at rear panel

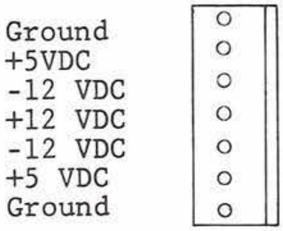
J7 Audio In for CUTS Cassette Interface: Mini-phone jack at rear panel

J8 Tape Motor Control 1: (See output port FA, bit 7) Sub-mini jack at rear panel

J9 Tape Motor Control 2: (See output port FA, bit 6) Sub-mini jack at rear panel

Rev A

J10 DC Power Connector, Sol-PC



S-100 Bus Definitions

PIN NUMBER	SYMBOL	NAME	FUNCTION
1	+8V	+8 Volts	Unregulated voltage on bus, supplied to PC boards and regulated to 5V supplied by Sol-20 supply
2	-16V	-16 Volts	Positive unregulated voltage supplied by Sol-20 power supply
3	XRDY	EXTERNAL READY	External ready input to CPU ready circuitry
4	VI0	Vectored Interrupt Line #0	
5	VI1	Vectored Interrupt Line #1	
6	VI2	Vectored Interrupt Line #2	
7	VI3	Vectored Interrupt Line #3	
8	VI4	Vectored Interrupt Line #4	
9	VI5	Vectored Interrupt Line #5	
10	VI6	Vectored Interrupt Line #6	
11	VI7	Vectored Interrupt Line #7	
12	XRDY2	EXTERNAL READY #2	not used by Sol-PC
13 to 17	TO BE DEFINED		
18	$\overline{\text{STAT DSB}}$	STATUS DISABLE	-Allows the buffers for the 8 status lines to be tri-stated
19	$\overline{\text{C/C DSB}}$	COMMAND/CONTROL DISABLE	-Allows the buffers for the 6 output command/control lines to be tri-stated
20	UNPROT	UNPROTECT	-not used by Sol-PC electronics
21	SS	SINGLE STEP	-not used by Sol-PC
22	$\overline{\text{ADD DSB}}$	ADDRESS DISABLE	-Allows the buffers for the 16 address lines to be tri-stated
23	$\overline{\text{DO DSB}}$	DATA OUT DISABLE	-Allows the buffers for the 8 data output lines to be tri-stated
24	Ø2	PHASE 2 CLOCK	
25	Ø1	PHASE 1 CLOCK	
26	PHLDA	HOLD ACKNOWLEDGE	Processor command/control output signal that appears in response to the HOLD signal; indicates that the data and address bus will go to the high impedance state and processor will enter HOLD state after completion of the current machine cycle.

PIN NUMBER	SYMBOL	NAME	FUNCTION
27	PWAIT	WAIT	-Processor command/control signal that appears in response to the HOLD signal; indicates that the data and address bus will go to the high impedance state and processor will enter HOLD state after completion of the current machine cycle
28	PINTE	INTERRUPT ENABLE	-Processor command/control output signal; indicates interrupts are enabled, as determined by the contents of the CPU internal interrupt flip-flop. When the flip-flop is set (Enable Interrupt instruction), interrupts are accepted by the CPU; when it is reset (Disable Interrupt instruction), interrupts are inhibited.
29	A5	Address Line #5	
30	A4	Address Line #4	
31	A3	Address Line #3	
32	A15	Address Line #15	(MSB)
33	A12	Address Line #12	
34	A9	Address Line #9	
35	DIO1	Data In/Out line #1	same as pin 94
36	DIOØ	Data In/Out line #0	same as pin 95
37	A10	Address Line #10	
38	DIO4	Data In/Out Line #4	same as pin 91
39	DIO5	Data In/Out Line #5	same as pin 92
40	DIO6	Data In/Out Line #6	same as pin 93
41	DIO2	Data In/Out Line #2	same as pin 88
42	DIO3	Data In/Out Line #3	same as pin 89
43	DIO7	Data In/Out Line #7	same as pin 90
44	SM1	MACHINE CYCLE 1	-Status output signal that indicates that the processor is in the fetch cycle for the first byte of an instruction
45	SOUT	OUTPUT	-Status output signal that indicates the address bus contains the address of an output device and the data bus will cohtain the ouput data when PWR is active
46	SINP	INPUT	-Status output signal that indicates the address bus contains the address of an input device and the input data should be placed on the data bus when PDBIN is active
47	SMEMR	MEMORY READ	-Status output signal that indicates the data bus will be used to read memory data
48	SHLTA	HALT ACKNOWLEDGE	- Status output signal that acknowledges a HALT instruction
49	$\overline{\text{CLOCK}}$	$\overline{\text{CLOCK}}$	- Inverted output of the Ø2 CLOCK
50	GND	GROUND	
51	+8V	+8 Volts	Unregulated input to 5 volt regulators supplied by Sol-20 power supply
52	-16V	-16 Volts	Negative unregulated voltage supplied by Sol-20 power supply

PIN NUMBER	SYMBOL	NAME	FUNCTION
53	SSWI	SENSE SWITCH INPUT	not used by Sol
54	$\overline{\text{EXT CLR}}$	EXTERNAL CLEAR	not used by Sol-PC electronics
55	RTC	REAL TIME CLOCK	not used by Sol-PC electronics
56	$\overline{\text{STSTB}}$	STATUS STROBE	not used by Sol
57	$\overline{\text{DIG1}}$	DATA INPUT GATE #1	When low forces PDBINS low and forces CPU input multiplexers to the DIO bus. During CPU DBIN cycle, disables CPU DIO bus drivers
58	FRDY	FRONT PANEL READY	-When low disables MWRITE driver
59 to 64	TO BE DEFINED		
65	$\overline{\text{MREQ}}$	$\overline{\text{MEMORY REQUEST}}$	-Z 80 signal not used by Sol-PC electronics
66	$\overline{\text{REF}}$	$\overline{\text{REFRESH}}$	- Z 80 signal not used by Sol-PC electronics
67	$\overline{\text{PHANTOM}}$	PHANTOM DISABLE	-Output from CPU section used to disable RAM or ROM during power on initialization program execution
68	MWRITE	MEMORY WRITE	-Indicates that the data present on the Data Out Bus is to be written into the memory location currently on the address bus
69	$\overline{\text{PS}}$	$\overline{\text{PROJECT STATUS}}$	-not used by Sol-PC electronics
70	PROT	PROTECT	-not used by Sol-PC electronics
71	RUN	RUN	- not used by Sol-PC electronics
72	PRDY	PROCESSOR READY	- Memory and I/O input to the CPU Board wait circuitry
73	$\overline{\text{PINT}}$	$\overline{\text{INTERRUPT REQUEST}}$	- The processor recognizes an interrupt request on this line at the end of the current instruction or while halted. If the processor is in the HOLD state or the Interrupt Enable flip-flop is reset, it will not honor the request.
74	$\overline{\text{PHOLD}}$	$\overline{\text{HOLD}}$	-Processor command/control input signal that requests the processor enter the HOLD state; allows an external device to gain control of address and data buses as soon as the processor has completed its use of these buses for the current machine cycle
75	$\overline{\text{PRESET}}$	$\overline{\text{RESET}}$	-Processor command/control input; while activated, the content of the program counter is cleared and the instruction register is set to 0
76	PSYNC	SYNC	-Processor command/control output; provides a signal to indicate the beginning of each machine cycle
77	$\overline{\text{PWR}}$	$\overline{\text{WRITE}}$	-Processor command/control output; used for memory write or I/O output control. Data on the data bus is stable while the PWR is active
78	PDBIN	DATA BUS IN	-Processor command/control output; indicates to external circuits that the data bus is in the input mode

S-100 Bus Definitions-continued

PIN NUMBER	SYMBOL	NAME	FUNCTION
79	A0	Address Line #0	(LSB)
80	A1	Address Line #1	
81	A2	Address Line #2	
82	A6	Address Line #6	
83	A7	Address Line #7	
84	A8	Address Line #8	
85	A13	Address Line #13	
86	A14	Address Line #14	
87	A11	Address Line #11	
88	DI02	Data In/Out Line #2	same as pin 41
89	DI03	Data In/Out Line #3	same as pin 42
90	DI07	Data In/Out Line #7	same as pin 43
91	DI04	Data In/Out Line #4	same as pin 38
92	DI05	Data In/Out Line #5	same as pin 39
93	DI06	Data In/Out Line #6	same as pin 40
94	DI01	Data In/Out Line #1	same as pin 35
95	DIO∅	Data In/Out Line #0	same as pin 36
96	SINTA	INTERRUPT ACKNOWLEDGE	-Status output signal; acknowledges signal for INTERRUPT request
97	$\overline{\text{SWO}}$	$\overline{\text{WRITE OUT}}$	-Status output signal; indicates that the operation in the current machine cycle will be a WRITE memory or output function
98	SSTACK	STACK	-Status output signal indicates that the address bus holds the pushdown stack address from the Stack Pointer
99	$\overline{\text{POC}}$	$\overline{\text{POWER-ON CLEAR}}$	
100	GND	GROUND	

SWITCH FUNCTION DEFINITION -- Display Ctrl---Schematic Drawing #4

		Function	
Switch No.	Mnemonic	ON	OFF
S1-1	RST	Restart to Zero	RUN (Dwg. #1)
S1-2	not used		
S1-3	BLANK	Blank Ctrl Characters	Display Ctrl Char.
S1-4	Polarity		
S1-5	BLINK	Blinking cursor	*Solid or NO cursor
S1-6	SOLID	Solid cursor	*Blinking or NO cursor

*NO cursor if S1-5 and S1-6 are off at same time.
Both switches should not be on at the same time.

Drawing #3 -- Sense Switch

		Function		
Switch No.	Mnemonic	ON	OFF	
S2-1	SSW∅	LSB, data bit	∅=LO	HI
S2-2thruS2-7		etc.	LO	HI
S2-8	SSW7	MSB data bit 7	LO	HI

SERIAL I/O BAUD RATE SWITCH -- Schematic Drawing #3

Switch No.	Mnemonic	Function	
		ON	OFF
S3-1	75	75 BAUD	* Do not turn more than
S3-2	11	110 BAUD	one switch on at a time
S3-3	15	150 BAUD	
S3-4	30	300 BAUD	
S3-5	60	600 BAUD	
S3-6	12	1200 BAUD	
S3-7	24/48	2400 or 4800(normally 2400 if not jumpered K to M)	
S3-8	96	9600 BAUD	

SERIAL I/O CONTROL -- Schematic Drawing #3

Switch No.	Mnemonic	ON	OFF
S4-1	PS	Parity even	Parity odd (if S4-5 on)
S4-2	WLS 1	Data word length	8bits 7bits 6bits 5bits
S4-3	WLS 2		Off Off On On
			Off On Off On
S4-4	SBS	1 stop bit	2 stop bits (1.5 if 5bits/word)
S4-5	PI_	Parity	No parity
S4-6	F/$\overline{H}$	Half duplex	Full duplex

MEMORY ALLOCATION: ON CARD

Hexidecimal Address	Function
CØØØ - C7FF	Personality Module ROM or PROM (2048 words)
C8ØØ - CBFF	System RAM (1024 words)
CCØØ - CFFF	Display RAM Memory (1024 characters)

ON CARD INPUT PORT ALLOCATION

Hexidecimal Port Address	Function
F8	Status, Serial Comm. channel
F9	Serial Communication Channel Data
FA	Aux. Status, Cassette tape interface, parallel I/O, keyboard input
FB	Audio Cassette (CUTS) Data
FC	Keyboard Data (from J3)
FD	Parallel Port Data (from J2)
FE	Display Status
FF	Sense Switch (S2-1 thru S2-8)

OUTPUT PORTS

Hex Port Address	Function
F8	Control, Serial Comm. Channel
F9	Data, Serial Comm. Channel
FA	Control, Parallel I/O, CUTS Cassette I/O
FB	Data, CUTS audio cassette Interface
FC	Alarm (optional)
FD	Data, Parallel output Data channel
FE	Scroll control, Display Section
FF	not used in Sol-PC

STATUS PORT INPUT BIT ASSIGNMENTS

PORT F8 (STATUS, SERIAL COMM. CHANNEL)

BIT	SIGNAL NAME	FUNCTION	ACTIVE DIRECTION
Ø	SCD	Serial Carrier Detect (EIA)	1 carrier
1	SDSR	Serial Data Set Ready (EIA)	Ø link ok
2	SPE	Serial Parity Error	1 error
3	SFE	Serial Framing Error	1 error
4	SOE	Serial Overrun Error	1 error
5	SCTS	Serial Clear to Send (EIA)	Ø clear
6	SDR	UART Serial Data Ready	1 ready
7	STBE	UART Serial Transmit Buffer Empty	1 empty

PORT FA (AUX. STATUS, CASSETTE TAPE INTERFACE, PARALLEL I/O, KEYBOARD INPUT)

BIT	SIGNAL NAME	FUNCTION	ACTIVE DIRECTION
Ø	KDR	Keyboard Data Ready	Ø ready
1	PDR	Parallel Data Ready	Ø ready
2	PXDR	Parallel eXternal Device Ready	Ø ready
3	TFE	Tape Framing Error	1 error
4	TOE	Tape Overrun Error	1 error
5	not used		
6	TDR	Tape Data Ready	1 ready
7	TTBE	Tape Transmitter Buffer Empty	1 empty

PORT FE (DISPLAY STATUS)

BIT	SIGNAL NAME	FUNCTION	ACTIVE DIRECTION
Ø	SOK	Scroll OK; ¼ sec timeout after scroll	Ø time complete

CONTROL PORT OUTPUT BIT ASSIGNMENTS

PORT F8 (CONTROL, SERIAL COMM. CHANNEL)

BIT	SIGNAL NAME	FUNCTION	ACTIVE DIRECTION
4	SRTS	Serial Request to Send	1 request

PORT FA (CONTROL, PARALLEL I/O, CUTS CASSETTE I/O)

BIT	SIGNAL NAME	FUNCTION	ACTIVE DIRECTION
3	PIE	Parallel Input Enable	1 pin 3 J2 low
4	PUS	Parallel Unit Select	0 pin 14 J2 low
5	TBR	Tape Baud Rate (300/1200)	0 1200 Baud
6	TT2	Tape Transport 2	0 run tape
7	TT1	Tape Transport 1	0 run tape

PORT FE (SCROLL CONTROL, DISPLAY SECTION)

BIT	SIGNAL NAME	FUNCTION	ACTIVE DIRECTION
Ø - 3	BDLA	Beginning Display Line Absolute address	4-bit data nybble
4 - 7	FDSP	First Displayed Line Screen Position	4-bit data nybble

CONNECTOR DESIGNATION

J1	Serial data	J6	Cassette Tape Audio Out
J2	Parallel Data	J7	Cassette Tape Audio In
J3	Keyboard	J8	Tape Motor 1
J4	Display Expansion	J9	Tape Motor 2
J5	ROM Personality Module	J10	PC Power
		J11	S-100 Bus Expansion

Personal Computing Magazine, May/June 1977

YOUR PERSONAL GENIE

by Tom Munnecke

It helps you with your income tax, then it takes you in the Starship Enterprise on an outer space crusade against the Klingons. It teaches you Boolean logic, then it becomes an opponent at checkers. It draws vivid pictures on your television set, then telephones a distant computer to calculate the value of your personal stock portfolio.

What is this personal genie? How can it take on so many personalities? It is the personal computer, and its personalities are the unique products of its programmer. The computer is capable of nothing more, nothing less than the programmer instructing it. For all the precision and rigidity associated with a computer, the programmer's work is still a uniquely personal reflection of himself.

The fundamental connection between the programmer and the computer is the computer language. The increasing number and sophistication of computer languages bring the power of the personal computer to the non-professional.

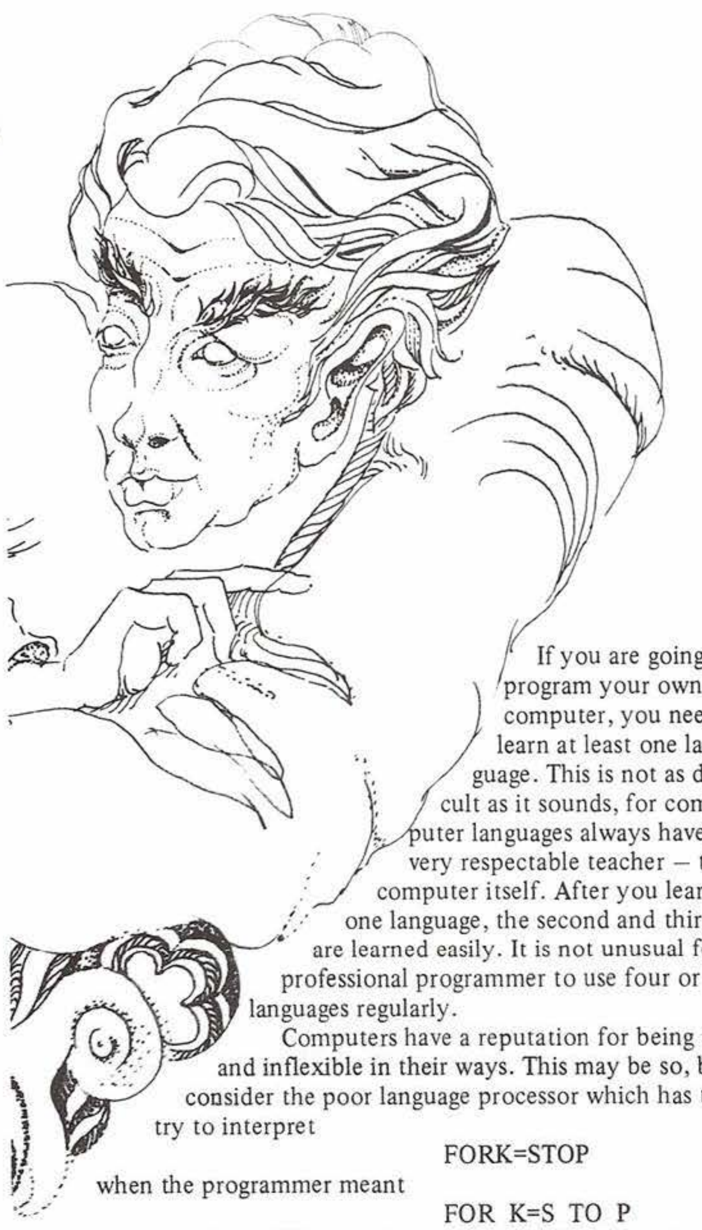
Computers are simple to deal with once certain fundamentals are understood. After that, learning becomes a trial and error experience. A person learning to walk does not need to understand each muscle, joint, and bone; he simply tries to walk and corrects his mistakes. So it is with computer programming. The novice programmer does not need to know the intricacies of the computer. He needs only: to know the fundamentals of the language, to know what his errors are and how to correct them, and to have time enough to try out his ideas.

The personal computer is a tool – the most powerful tool ever put in the hands of the private individual. Its potential is limited only by its owner's capacity to apply it.

This article provides a head start on learning any computer language, discussing the merits and drawbacks of many of the computer languages available to the personal computing enthusiast.

What is a Computer Language?

Computers operate in sequences of primitive decisions made in millionths of seconds. People think in terms of vague concepts derived over days and months. The computer language is the means of linking these vague human concepts to the primitive computer decision.



If you are going to program your own computer, you need to learn at least one language. This is not as difficult as it sounds, for computer languages always have a very respectable teacher – the computer itself. After you learn one language, the second and third are learned easily. It is not unusual for a professional programmer to use four or five languages regularly.

Computers have a reputation for being rigid and inflexible in their ways. This may be so, but consider the poor language processor which has to try to interpret

FORK=STOP

when the programmer meant

FOR K=S TO P

Most of the rigidity of the computer is there for a purpose. If you learn how they interpret things, some apparent inflexibility will fade away.

In order for the computer and the programmer to communicate, they must have some common physical medium for communicating. Usually, this is a keyboard/printer or video display. The programmer enters his programs in whatever language he is using, in his version of the language, known as the source language. He then asks a language processor to prepare it for the computer to process it. There are two types of language processors – translators and interpreters. The translator accepts the source language and translates it to an object language, which is then loaded into the computer to be executed. Translators are further broken down into assemblers and compilers. The assembler is a means of manipulating machine-level operations for a specific computer, while the compiler translates higher-level, or more human-oriented languages. Interpreters execute the source language directly without the intermediate process of translating to an object language.

Languages are classified into two vaguely defined classifications: high-level and low-level. A low-level language is one in which each of the source code instructions corresponds to a machine-level operation. Source code in a high-level language may generate many machine-level instructions.

Assemblers, Compilers, and Interpreters

Each of the types of language processors has its merits and drawbacks – assemblers give the programmer great power but require very detailed instructions; compilers support higher-level languages, but sacrifice machine efficiency; and interpreters are easy to use, but are not as efficient as compilers.

Assemblers

The assembler is the simplest form of computer language. It accepts source code and translates it one-for-one into machine-level instructions or object code. Thus, the programmer has detailed control (and responsibility) of each instruction. For example, the programmer might write a line in assembler such as:

```
NEXT JSR INCHAR ; Jump to subroutine to get a
                          character.
```

'NEXT' is a label for the line. 'JSR' is a mnemonic for the Motorola 6800 instruction 'Jump to subroutine'. 'INCHAR' represents the address of the subroutine to be used. ';Jump . . .' is a comment inserted by the programmer to explain the instruction for documentation.

The assembler (for the 6800) will assemble this instruction into the hexidecimal '8DXXXX' where '8D' is the operation code for branch to subroutine, and 'XXXX' is the address of subroutine INCHAR. See Fig. 1.

Since the assembler may not know where the INCHAR subroutine is to be located when the program is executed, it must be resolved at a later time by the *loader* program.

Compilers

The compiler acts much like the assembler, but works with higher level languages. The compiler understands more

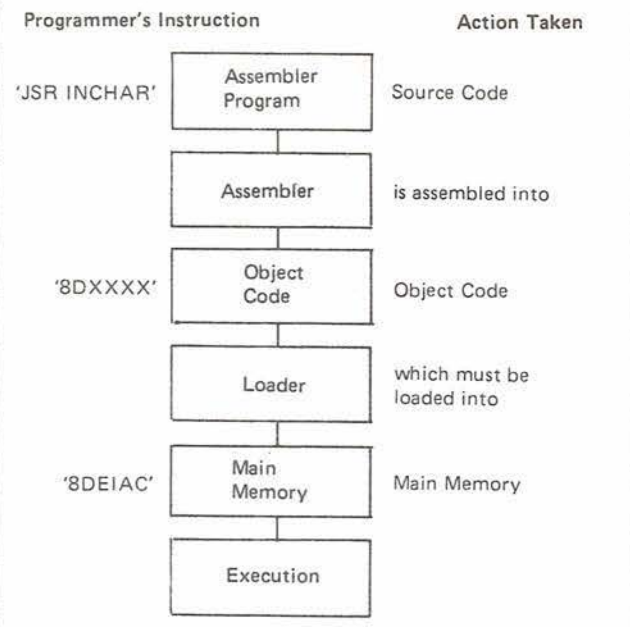


Fig. 1 Operation of an Assembler

complex expressions, and does much more work than the assembler. Figure 2 illustrates a single high level language expression which would require 6 lines to write in assembler.

Compiler	Assembler			
TOT=SUM+NUM	LDA	A	SUM+1	Add the right
	ADD	A	NUM+1	most bytes and
	STA	A	TOT+1	store the result.
	LDA	A	SUM	Add the left
	ADC	A	NUM	most bytes and
	STA	A	TOT	store the result.

Fig. 2 Comparison of high-level expression and its low-level language equivalent.

This is a simple example, but a more complex example, such as:

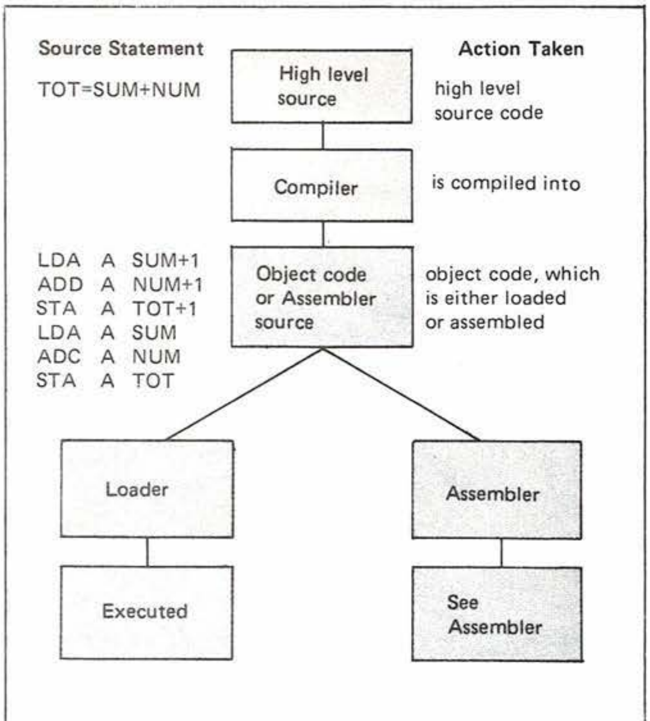
TOT=(SUM + NUM/1.238 * COS (ARC/360)) ** 2/7.32 could give the assembler language programmer a tremendous amount of difficulty.

Typically, a compiler produces assembly language code, which is then passed through the assembler.

Interpreters

The interpreter is a departure from the techniques of the assembler or compiler. While the translators create a program which must be loaded and executed later, the interpreter executes source instructions directly. The source remains in its original form.

Many languages may be either compiled or interpreted, although some features of a language may make compilation difficult, if not impossible. The interpreted language can change its interpretation as it receives new data, while the compiler does not know what data the program will receive until after it has finished its work.



```
TOT=SUM+NUM

LDA  A  SUM+1
ADD  A  NUM+1
STA  A  TOT+1
LDA  A  SUM
ADC  A  NUM
STA  A  TOT
```

Fig. 3 Operation of a Compiler

Comparison of the Methods:

Each of the methods is used in the commercial computing world, indicating that there is sound economic need for each. The methods may intermingle, as in compilers that accept assembler language code, incremental compilers, which are a cross between interpreting and compiling, and compilers which produce interpretive object code. Fig. 5 illustrates many common considerations of the various language processors.

Disadvantages of Assemblers

Because the programmer must detail each operation of the computer, his workload is much greater than with higher level languages. His chances for making an error are much greater than in high-level languages. The programmer can easily become enmeshed in the maze of details he must remember. Modifying an intricate assembler language program may be very difficult, if not impossible. Assemblers are not usually interactive, requiring the entire program to be reassembled when an error is made.

Advantages of Compilers

The compiler is capable of supporting much higher level languages than assembler or macro assembler. The programmer can work faster, make fewer errors, and learn the language faster than he can assembler. The compiler's object code may be executed much faster than an interpreter could execute the program (between 5 and 10 times faster). Programs written in the higher level language may be recompiled on a new type of computer, without modifying the program.

Disadvantages of Compilers

Compilers are usually large, complex programs which require some time to compile a program, in addition to a significant amount of off-line storage. Compilers are not usually interactive, because they require an entire program to be recompiled when a single change must be made.

Due to the internal workings of the compiler, data types must be fixed during compilation. This process, known as binding, reduces the program's ability to adapt to new data as the program is executed. An interpreter, however, does not bind its variables until execution.

Advantages of the Interpreter

Since the interpreter executes its source code directly, the programmer may interact more directly with the computer. Usually, the interpreter provides a direct mode, where the programmer may execute statements directly as he enters them, and an indirect mode, where his commands are stored in a program for later execution. The programmer can usually stop the program, examine variables, and resume execution. Some interpreters (such as APL and MUMPS) provide an EXECUTE command, which allows the program to execute a character string as if it were program text. Conversely, some interpreters (MUMPS) allow a program to treat its own text as data. Interpreters are useful for systems where the language processor needs to be 'built in' to the computer, as in intelligent terminals.

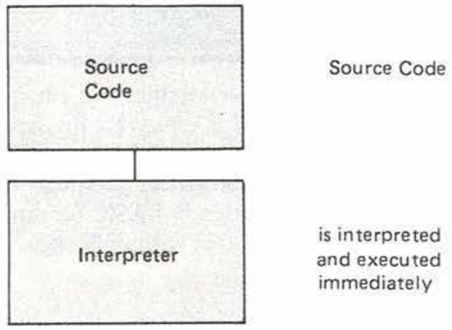


Fig. 4 Operation of an Interpreter

Characteristic	Compiler	Interpreter	Assembler
Binding Time	Compile	Execution	Assembly
Off line Storage	much	little/none	much
CPU Efficiency	medium	low	high
Programmer Efficiency	medium	high	low
Program Size	large	medium	small
Error Detection Language	machine	source	machine
Interactive Debugging	no	yes	no
Language Processing Efficiency	low	medium	high

Fig 5 Comparison of the features of the various types of language processors.

• Binding time — the point when the program's data types are fixed. • Offline Storage — the amount of storage such as floppy disks, cassettes, etc, required for the language processor to work. • CPU Efficiency — of the program being processed. • Programmer Efficiency — of the programmer writing the program to be processed. • Program Size — of the object code, or source code, in the case of the interpreter. • Error Detection Language — the language in which run time errors are detected.

Disadvantages of the Interpreter

Interpreters tend to be slower than compilers, between 5 and 10 times slower, as a rule of thumb. This slowness is due to the interpreter's need to analyze each statement every time it sees it, whereas the compiler need analyze it only once. The interpreter program must remain in memory for even a small program.

A Bit of History

The first computers were large, expensive devices requiring a roomful of air conditioning just to keep them cool. Programming them was very difficult, and they ran quite slow:

> ". . . the machine will then continue in operation hour after hour, completely checking its own results until either the problem is solved, or a breakdown occurs" (A Manual of Operation for the Automated Sequence Controlled Calculator, Harvard University, 1946).

At that time, a computer cost millions of dollars, and a programmer cost a few hundred dollars per month. Today, a computer costs hundreds of dollars, and the programmer costs thousands of dollars per month. To put it in another way, in 1946 a computer cost the equivalent of 250 programmers, today the programmer costs the equivalent of 100 computers.

Everyone agrees that computers should be used 'efficiently'. The problem is that people think of making the CPU efficient, not the person using it. The microcomputer has undermined the conventional wisdom of computer efficiency. The person who spends several month's rent on a personal computer wants to see it do something for him immediately, regardless of whether it uses the CPU 'efficiently'. Chances are he uses the computer only a few hours a day. On the other hand, the professional programmer who works as one of a score of programmers using a large computer must contend with CPU efficiency in order to keep from overloading the computer.

The microcomputer user needs to worry about CPU efficiency only when he reaches some limit — not enough memory response not fast enough, etc. Since no one else is waiting to use his computer, he does not have to worry about inefficiencies which do not force him beyond his limits.

The large computer programmer, however, must constantly worry about sharing the computer with all the other users. Even if a program works fast enough for him, and uses little enough memory, it still must be made 'efficient' for the other users of the system.

As a result of this historical concern for CPU efficiency, people are fixated on "making the computer run efficiently". Language design has been heavily weighted in favor of making the computer efficient, not the programmer.

The personal computing software scene was a completely unforeseen turn of events. None of the language designers ever thought that the programmer would be working alone on his own computer. As a result, the design tradeoffs were heavily slanted in favor of the commerical user.

Which language is Best?

> "I speak Spanish to God, Italian to women, French to men, and German to my horse".
> Charles V of France

What is the best language? BASIC? Assembler? PL/M,

- PROGRAM
  - LINE
    - LINE NUMBER
    - COMMANDS
    - ARGUMENTS

Fig. 6 BASIC Program Elements

- PROGRAM
  - STATEMENTS
    - LABEL
    - OPERATION CODE
    - OPERANDS

Fig. 7 Assembler Program Elements

- PROGRAM
  - BLOCK
    - STATEMENTS
      - EXPRESSIONS

Fig. 8 PL/M Program Elements

- PROGRAM
  - ROUTINE
    - LINE
    - LABEL
    - COMMAND
    - ARGUMENTS

Fig. 9 MUMPS Program Elements

- WORKSPACE
  - FUNCTION
    - LINE
      - OPERATORS
      - LITERALS
      - FUNCTION REFERENCES

Fig. 10 APL Program Elements

- PROGRAM
  - GROUP
    - LINE NUMBER
    - COMMAND
    - ARGUMENTS

Fig. 11 FOCAL Program Elements

MUMPS, APL, PASCAL, FORTRAN, SNOBOL, COBOL, LISP, COMIT, MAD, or any of the hundreds of others? And after the best language is chosen, which dialect is best? Consider the dialects of BASIC: Tiny BASIC, Extended BASIC, BASIC Plus, Business BASIC, ANS BASIC . . .

Perhaps a good analogy could be drawn between computer languages and spoken languages. Which spoken language is best? English? French? Chinese? Italian? It all depends on what you want to do with it. If you are in Paris, French would be a good contender for the 'best' language. Suppose you are in Kansas, and believed Charles' statement above that Italian is best for speaking to women. Romantic pretentions aside, you would probably have better luck with English.

The "best" computer language is not selected on the basis of its syntax or grammar. It is a very pragmatic decision based on what is available, what the programmer knows, whether it can perform the task at hand, and what programs are available to him from other sources.

The selection of a computer language is an important decision to the personal programmer for many reasons beyond the above pragmatic ones. The language a programmer uses profoundly affects the way he sees a problem. As Whorf said. "We dissect nature along lines laid down by our native language". The APL programmer thinks in terms of vectors, the MUMPS programmer thinks in terms of data bases, and the Assembly language programmer thinks in terms of individual bytes of memory.

Therefore, in reviewing each of the languages, the reader must apply them to his own needs. The following list is a sample of some of the languages available (or may be soon) to the micro-computer user.

BASIC – (Beginner's All purpose Symbolic Instruction Code). This is the most common high-level language used on personal computers. It is a very simple, easy to learn language. There is a large library of programs available, since BASIC is used in many universities and schools. Because it is a simple language, it is somewhat limited and difficult to use for some complex problems. BASIC is usually interpreted on microcomputers, although some compilers exist. Programs written in BASIC for one computer can often be run on another with only slight changes.

Assembler – Assembler language is commonly used on personal computers. Since many personal computers have neither the memory or Input/Output capability to run an assembler, the programmer often manually assembles his program and enters it through the switches on the panel. Assembler language is unique to each computer, so program exchange is limited to one particular computer type.

Assembler language is the common denominator of all programs – eventually, all programs are just a sequence of assembler-level instructions. Therefore, any one wishing to really know how his computer works must learn at least a little Assembler. Often, a program is written in a high-level language which calls an Assembler language subroutine for difficult or critical portions of logic. This can be a very economical mix for programs which exceed the limits of a high-level language.

PL/M – (A program name copyrighted by Intel Corp.) is a compiled language derived from IBM's PL/1. Versions exist for the 8080, 6800, and Signetics 2650. Some high speed, mass storage (floppy disk, for example) is required. It is an alternative to assembler, producing slightly less efficient programs in much less programming time. A basic user would find PL/M difficult to use for simple problems, but easier to use for more complex problems. There is no extensive library of programs in PL/M as with BASIC.

MUMPS – (Massachusetts General Hospital Utility Multi-Programming System) is an interpretive language oriented towards interactive data management applications. MUMPS has many characteristics of BASIC, FOCAL, and IBM's PL/1. It differs from all these in that it has built-in data base capabilities for handling data on mass storage devices. Although not widely available on microcomputers now, the National Bureau of Standards published a standard version (NBS Handbook 118) which details how one would write an interpreter for MUMPS.

MUMPS has extensive data handling capabilities, suited for applications such as personal accounting, word processing, and general information systems. Since the development of MUMPS was federally supported, much MUMPS software is in the public domain.

APL – (A Programming Language) is a computer language derived from Iverson's elegant mathematical notation. It is a very powerful mathematical tool, having primitive functions for matrix inversion, inner products, sorting, and many other areas. Although initially developed for large scale computers, it is now available for portable commercial computers. APL is usually interpreted, and therefore well suited for interactive personal computing.

FOCAL – (Formulating On-Line Calculations in Algebraic Language) is a language brought out as an early on-line language for calculations. Its syntax is similar to MUMPS, although its functions are closer to BASIC. FOCAL is available on the 8080 and has a modest programming library.

Learning a Computer Language

Your first task in learning a new language is to build up a basic understanding of the language. This can be gained from the reference manual for the language distributed with the software. Magazines such as Personal Computing carry many articles on the more popular languages. There is a variety of books available in libraries and computer stores, and more advertised in professional data processing magazines.

When studying a language, it is helpful to divide the project into three areas:

SYNTAX – How you say something
SEMANTICS – What you mean
PRAGMATICS – How you make the language do what you want

Syntax. The syntax of the language is usually the quickest part to learn. How does the language distinguish between a number and a variable? Do you need a number before each line? What characters are allowed by the language?

Semantics. The semantic aspects of the language are more difficult to learn, but you do not have to understand everything to use the language. What are arithmetic functions in the language? How do you retrieve data from the terminal? How do you format output?

Pragmatics. This is the most difficult portion to learn, yet it is the skill most easily carried over to other languages. How do you make the language solve your problem? How do you create, change, and delete programs? Can you stop the program while it is executing, examine the state of things, then resume execution?

These three classifications are very useful for comparing languages. For example, BASIC, FOCAL and FORTRAN have similar semantics but different syntaxes. MUMPS and FOCAL have similar syntaxes, but different semantics.

With this background, you should be able to modify a simple program to make it do increasingly complex tasks. Each time you modify the program, use some new aspect of the language, being careful to add one aspect at a time. Then, try the new version to see if it does what you expect.

Each step of the way, you will be informed of your mistakes by your friendly adversary, the computer.

The Importance of Making Errors

> "Nine times out of ten, in the arts as well as life, there is actually no truth to be discovered; there is only error to be exposed."
>
> H.L. Mencken

Making an error in a computer program is a fundamental source of learning. You tried something and the computer told you it didn't work. The programmer who proudly announces "my last program worked the first time without any bugs" is a programmer who probably did not learn anything new writing it.

	BASIC	MUMPS	APL	FOCAL	PL/M	FORTRAN
Integer (16 Bit)	X	X	X	X	X	X
Byte			X		X	
Character String	X	X				
Floating Point	X	X	X	X		X
Logical		X	X			X
Labels		X				

Fig. 12 Cross Index of Data Element Types

	BASIC	MUMPS	APL	FOCAL	PL/M	FORTRAN
Assignment	LET	SET	←	SET	=	=
Read from Console	INPUT	READ	←□	ASK	INPUT	INPUT
Write to Console	PRINT	WRITE	□←	TYPE	OUTPUT	OUTPUT

Fig. 13 Cross Index of Data Movement

	BASIC	MUMPS	APL	FOCAL	PL/M	FORTRAN
Unconditional Branch	GOTO	GOTO	→	GO	GOTO	GOTO
Conditional Branching	IF	IF	→	IF	IF	IF
Involation	GO SUB	DO	NAME	DO	CALL	CALL
Return from Involation	RETURN	QUIT	→0	QUIT	END	RETURN
Looping	FOR/ NEXT	FOR		FOR	DO	DO

Fig. 14 Cross Index of Control of Flow

	BASIC	MUMPS	APL	FOCAL	PL/M	FORTRAN
And		&	∧			
Or		!	∨			
Not		'	~			
Greater Than	>	>	>	*		
Less Than	<	<	<	*		
Equal	=	=	=	*		
Not Equal	<>			*		
Less Than Or Eq.	<=		≤	*		
Greater Than or Equal	=>		≥	*		

Fig. 15 Logical and Arithmetic Comparison Function

*handled by IF statement structure.

	BASIC	MUMPS	APL	FOCAL	PL/M
Addition	+	+	+	+	+
Subtraction	—	—	—	—	—
Divide	/	/	+	/	/
Multiply	*	*	X	*	*
Exponentiation	↑		*	FExP	
Square root	SQR		*.5	FSQT	
Cosine	COS		2○	FCOS	
Tangent	TAN		3○		
SINE	SIN		1○	FSIN	
e^X Exponential	EXP		*	EXP	
Natural log	LOG		⍟	FLOG	
Absolute Val	ABS		|	FABS	
Greatest Integer	INT		⌊	FITR	
Random Number	RND	SR	?	FRAN	
Signum	SGN		X	FSGN	
Modulo		#	|		

Fig. 16 Cross Index of Arithmetic Functions

The absence of an error when writing a program indicates only that a situation new to the programmer did not come up – not that the programmer has learned the language.

There are generally four types of errors: syntax, semantic, pragmatic, and covert.

Syntax errors

The syntax error is the most common error which faces the beginning programmer. A syntax error is a statement that violates the language's basic rule for expression. Typically, they are caused by:

a) typing errors – a finger slips to the wrong key, a zero instead of the letter O, etc.

b) misunderstanding the syntax. The new programmer may not understand that he has to put a comma between variables in a print statement, or put apostrophes around literals.

c) confusing the syntax. The programmer might confuse a colon and the comma, or, he might carry over some syntax from another language he knows.

One thing in common with all these errors is that the computer can detect them. In most interpreters, the programmer may directly enter and execute any questionable statements. The lesson is clear: When in doubt, try it. Let the computer tell you whether it will accept the statement. Many manuals are not reliable enough to trust anyway.

The above advice flies directly in the face of conventional computer programming wisdom. In the past, there was considerable stigma attached to anyone found 'letting the computer do his debugging'. The theory was, that the computer is a valuable resource, and that a programmer should not waste computer time. Instead, he should carefully desk-check his program before each submission. In the microcomputer world, this philosophy is radically altered. It makes no sense for the programmer to check his work on paper when his computer is waiting for him to enter it.

Semantic errors

These errors are also common in the early stages of learning a new language, but continue to plague the programmer throughout the use of the language. These errors are statements which are syntactically correct, but do not perform the function desired by the programmer. Some typical semantic errors are:

a) Mode errors – the programmer tries to add a number to a character string, but the language does not handle the conversion.

b) Binding errors – the programmer names the wrong variable, label or subroutine.

c) Juxtaposition or sequencing errors. An end of a loop is placed too far down in the program, or a variable is used before it is initialized.

Most of the same advice for syntax errors applies to grammatical errors. Sometimes, grammatical errors can slip through and only be detected by erratic program behavior.

Pragmatic errors

The pragmatic error is a statement which is syntactically and semantically correct, but does not do what the programmer wants it to. These cannot be caught by the language processor. Typical pragmatic errors are:

a) wrong function or command – the programmer uses a sine function instead of cosine.

b) an improper formula – the programmer thought that Interest was Principal divided by Rate instead of Principal times Rate.

Pragmatic errors tend to be the last errors in a program to be detected, if only because the programmer will not see them until he cleans up the syntax and semantic errors and the program executes.

Pragmatic errors can be very difficult to detect, particularly in programs which are time dependent or involve much

	BASIC	MUMPS	APL	FOCAL	PL/M
Search		$FIND	?		
Extract	MID$ LEFT$ RIGHT$	$EXTRACT	SUBSCRIPTS		
Concatenation		—			
Convert String to Number		$ASCII			
Convert Number to String		$CHAR			
Length	LEN$	$LENGTH			

Fig. 17 Cross Index String Functions

logic. Pragmatic errors are generally discovered with what the computing world euphemistically calls "testing". "I'll test this program to make sure it won't blow up," is an often heard phrase. Unfortunately after he completes his testing, he all too often says "my program blew up".

Testing can confirm the existence of an error, not that one doesn't exist. Just because 99 combinations of input data were tried does not guarantee that the hundreth combination will not fail.

Covert errors

When a program is tested and declared correct by the programmer, any remaining errors are by definition covert. These are insidious problems that appear only when events combine to form some previously untried condition. Some covert errors are:

a) An angle in a trigonometric equation goes to zero, causing a zero divide error in a later division.

b) Improper data is entered, which the program does not reject as invalid. Recently, a program sent out a letter to the Emmet County Jail, "Dear Emmet C. Jail, you are among a select group of persons . . . " As the saying goes – garbage in, garbage out.

c) The programmer leaves room for only 3 digits of a number, but the number grows past 999.

Covert errors always have and always will exist in computer software. However, a great deal of attention in computer science circles has been given to writing programs which may be "proved" correct. These efforts, named "structured programming", "software engineering", and "composite design" will be covered in a future article. The fundamental principles common to these are:

a) Break the big problem into clusters of independent little problems.

b) Link the clusters together in a hierarchical manner such that each cluster is independently testable.

c) Limit the number of paths the program may take. This is accomplished by limiting the use of the GOTO statement.

The programmer should learn to improve his skills by analyzing the errors he makes. When he meets that benevolent dictator of linguistic purity – the error message – he should treat it as a means of learning a little more about the language.

UPDATES

Sol TERMINAL COMPUTER^{TM}

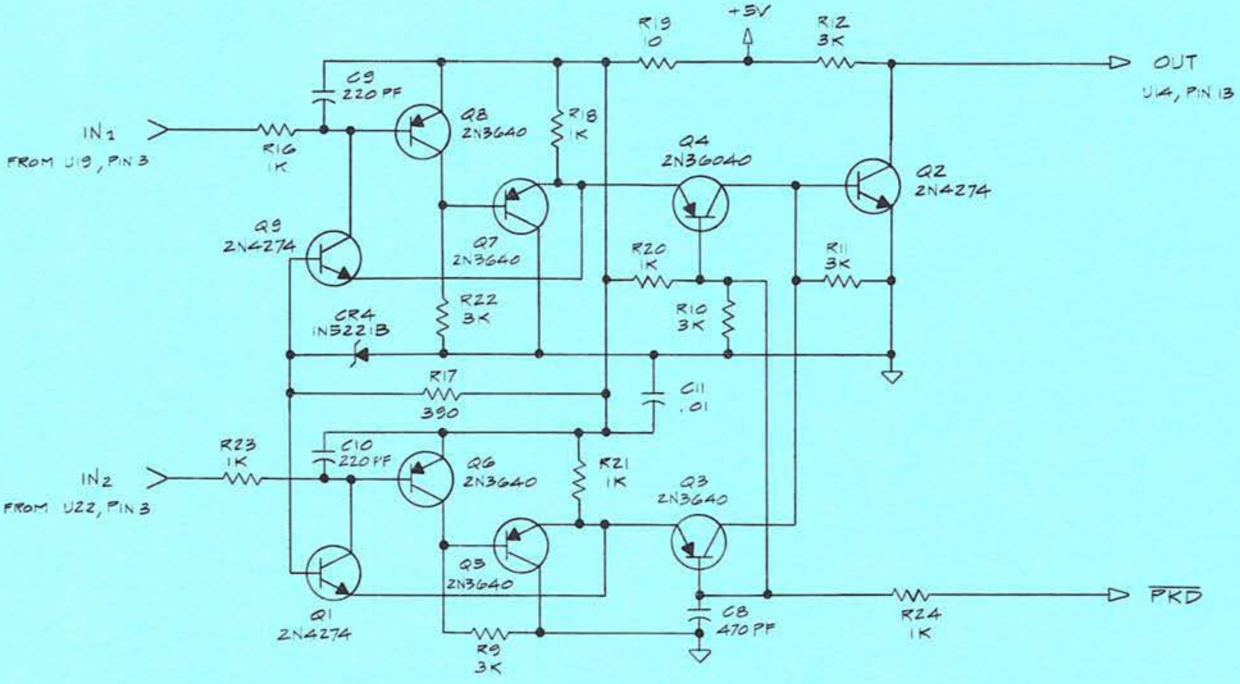
Electronics is a very fast moving field. Development of new products, and improvements in the old products proceedes at an unprecedented rate. The continuing development of the Sol Terminal Computer is no exception. Better parts become available and are included, experience yields circuit improvements, and new circuitry is developed. This process generates changes much more frequently than this manual is reprinted. As a result, we include the improvements as blue update sheets, added to this section as they become available. Be sure to integrate this information into the body of the manual before beginning, by making indicated changes in the text, adding or replacing pages, or making notes referring you to the update page.

If you have a question as to the currency of a particular page of text, look in the lower left-hand corner of the page. The inital version of the page will have this corner blank. When the contents of the page have changed, the new version will have "Rev A" in this corner; a third version will have "Rev B", and so forth. When a whole new page and page number are added, the corner is blank.

Sol MANUAL ADDENDUM #1

Reference Section X, Drawing X-23.

A block function labelled "K.T.C." is shown between U-19 and U22, and U14. This block contains the Capacitive Switch Detector Circuit. The parts constituting this circuit are listed, and the assembly covered in Section V of this manual. The theory of operation is covered in Section VIII. At the time of publication of this manual, operation of this circuit was proprietary information, but has now been released. The schematic is shown below. Note on the schematic X-23 that this detail is shown here on this page.



Sol MANUAL ADDENDUM #3

Processor Technology

Cassette Recorders for use with Sol

Not all audio cassette recorders are suited for data storage use with the Sol. The following models have been tested and approved by Processor Technology:

1. Panasonic RQ-413AS
2. Realistic CTR-21

Some users have reported unsuccessful results with the Panasonic RQ-309 and the J. C. Penny Catalog #851-0018. If you should wish to select a different model, the following features, included on the models above, are necessary:

1. An AUX input. Although the Sol can be jumpered for low level Microphone level input, the procedure is no longer recommended.
2. A digital counter. The counter is necessary in locating programs on the cassette.
3. A tone control. The existence of a tone control is one indication of high quality electronics.

Even though a recorder has the three features, there is no guarantee that it will work properly for the purpose. Recorders vary greatly in the quality of their electronics. If possible, test the recorder with a long file before purchasing it, in both record (SAVE) and playback (GET or XEQ) mode. If the recorder is not working properly, either you will get an error message, or you will find differences between what was recorded and what was played back.

Observe the following pointers for best results:

1. Keep the recorder at least a foot away from the Sol, or other equipment which can generate magnetic fields. The recorder can pick up hum which may generate errors.
2. Keep the tape heads cleaned and demagnetized in accordance with the manufacturer's instructions.
3. Use high quality brand-name tape. Cheap tape can wear down the tape heads and give erratic results.
4. Bulk erase tapes before using.
5. Keep the cassettes in their protective plastic covers, in a cool place, when not in use. Cassettes are vulnerable to dirt, high temperatures, liquids, and physical abuse.
6. Set the tone control at midrange, and set the volume control about 2/3 full volume. The Sol has an automatic gain control circuit which compensates for a wide range of levels, but operation in the middle of this range will

give the most reliable results. Experiment to find the best setting of volume and tone controls.

7. On some cassette recorders, the microphone can be live while recording through the AUX input. Deactivate the mike in accordance with the manufacturer's instructions. In some cases this can be done by inserting a dummy plug into the microphone jack.

8. During recording, some recorders present the signal being recorded at the monitor or earphone output. In a system with two cassette recorders this could cause problems if an attempt was made to read from one recorder while the other was writing. Since both recorders share the same audio lines, the monitor output of the recorder which was recording could interfere with the signal being read from the other recorder.

9. If you record more than one file on a tape side, SAVE a special file, which could be named END, to let you know when you have played past the files of interest. After recording the last file on a side, rewind the tape, set the digital counter to zero, and issue a CATalog command (see SOLOS/CUTER User's Manual). As each file header is displayed, make a note of the reading on the digital counter, the exact name of the file, load address, and file length. Mark the cassette with this information to make file retrieval much easier.

If you experience a read error, use the following procedure to isolate the problem:

1. Check for proper settings, and make sure you have followed the pointers above.

2. Check cables for intermittant connections and shorts.

3. Note the exact reading of the digital counter at the time of the error.

4. Rewind the tape and try to read the same part of the tape again. If the tape reads without errors this time, the error was not recorded on the tape. If there is a read error at the same point, then the error is recorded on the tape.

5. Rewind the tape and record a file on the same part of the tape. Read the file. If the tape reads without errors, then the original read error was generated during the recording process. If there is still a read error at the same point, then the cassette itself is faulty.

Processor Technology

Sol MANUAL ERRATA NOTICE #3

1. Reference Section X, Drawings, Drawing X-17, Input/Output.

   In the Baud Rate Generator section of this schematic, the function of switch S3-7 is incorrectly shown as selecting 2400/4800 Baud, and S3-8 is incorrectly shown as selecting only 9600 Baud. Change the schematic to show that S3-7 selects 2400 Baud only, and that S3-8 selects 9600/4800 Baud. Draw a line connecting points L and M to indicate a jumper.

2. Reference Section VII, page VII-15, Table 7-2.

   In the Baud Rate column of this table, change "4800***" to read "9600***". Also, in the footnote with the triple asterisk, change the phrase "SDI operates at 9600 Baud..." to read "SDI operates at 4800 Baud..."

ES 3 6/77

ASSEMBLY PROCEDURE CHANGE NOTICE #6-2 Rev B

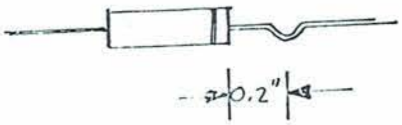
This Change Notice concerns the Sol-REG board and applies only to Revision Level B boards.

A problem was detected in early Sol-REG boards in which the "crowbar" circuit would trigger without adequate cause and short circuit the 5-volt output. A circuit change has been made which will be reflected in Revision Level C and above boards to correct the problem. Revision Level B boards, however, require the following modification to correct the problem. Parts for this modification are supplied with your kit:

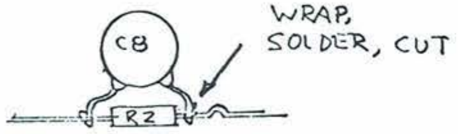
1) R2, 330 ohms, 1/4 watt, color code orange-orange-brown
2) R14, 100 ohms, 1/4 watt, color code brown-black-brown
3) D1, 1N5231B
4) C8, 0.047 uF disc ceramic

Assemble these parts as follows:

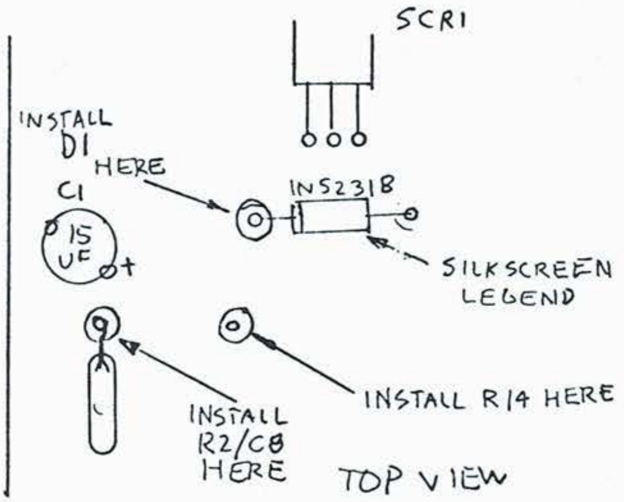
1. Form one lead of R2, R14, and the cathode (banded) lead of D1 for upright P.C. insertion as shown:

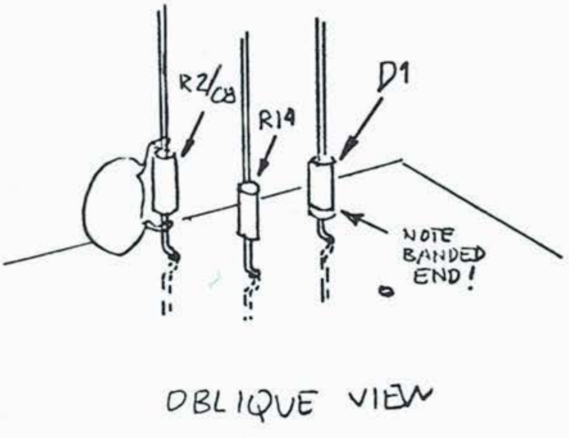


2. Solder C8 in parallel with R2 as shown:

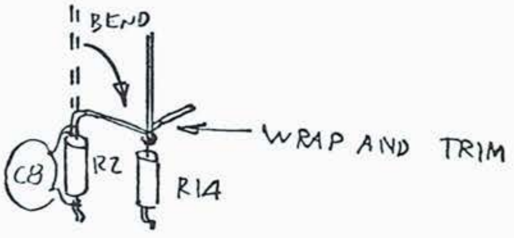


3. Install and solder R2-C8, R14, and D1 as shown below. Install the formed leads into the board with the unformed leads vertical. Position R2-C8 so that the body of C8 is parallel to the board edge and oriented away from C1.

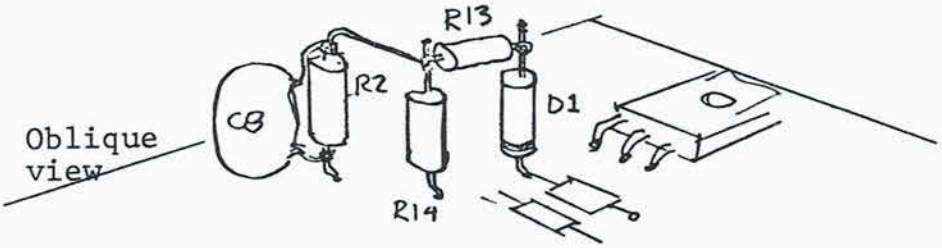




4. Bend the top lead of R2-C8 over towards R14, and bend it around the top lead of R14 one-eigth inch from the body of R14. Solder, and trim the excess lead of R2-C8 only.



5. Install R13 between the top leads of D1 and R14. Wrap R13's leads around R14 and D1 leads. Solder all connections at both points, and trim excess lead lengths. The resulting final configuration is shown below.



Schematic Diagram X-12 of the regulator includes these changes.