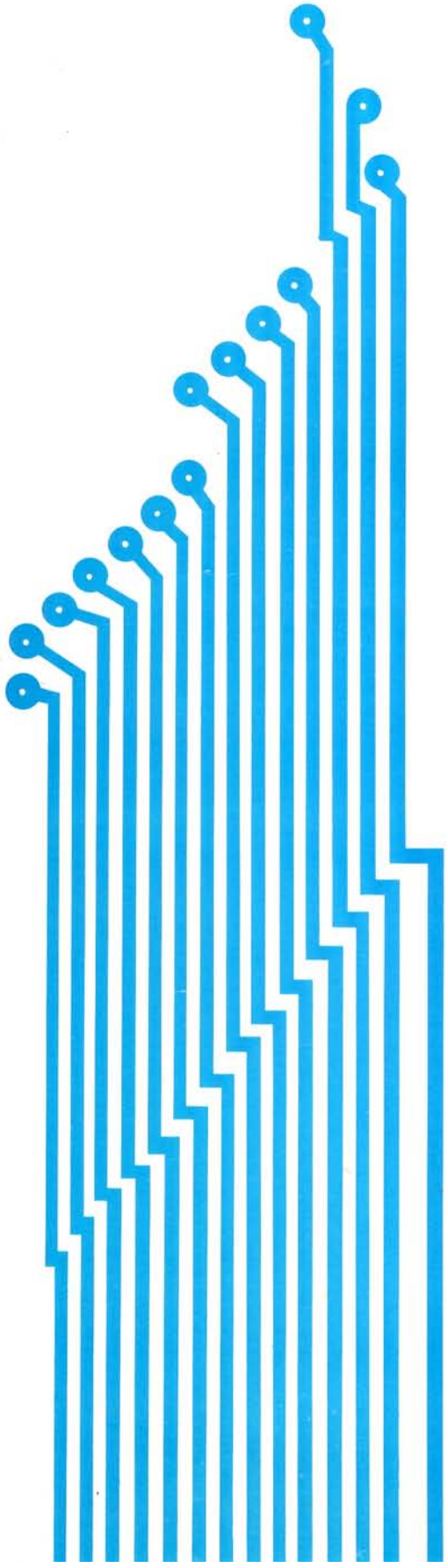


SSZ-80
PROCESSOR MANUAL

DESIGN Ltd. RD 2 FREWSBURG, NEW YORK 14738 716/569-6060

SSZ-80 Processor Board Manual

SSZ-80 Board Description

The Design Ltd. SSZ-80 is a processor board which brings full Z-80 microprocessor power to the SS-50 BUSS. In addition to the Z-80 microprocessor, the board comes equipped with a 2K RAM monitor, 1K of read/write RAM, full 65K memory addressing and full vectored interrupt capability. The address, data, and control lines are buffered and the board supports an additional 14K of ROM or EPROM. The SSZ-80 may be used alone or in conjunction with the SWTPC MP-A or MP-A2 processor board. Thus software that has already been developed or purchased for the 6800 processor will not have to be discarded.

The SSZ-80 has a resident 2K ROM monitor. This is a version of the TDL ZAPPLE® monitor which has been modified for use with the SS-50 hardware configuration. The Zapple monitor is one of the most versatile monitors available, giving the operator a very powerful debug and executive control system. This monitor provides 23 commands plus up to 29 additional user defined commands. All I/O routines and assignment of I/O devices are handled through the monitor. Since all software that we supply or software purchased from TDL uses this monitor for its I/O handling, this gives the most powerful I/O handling and operational capability for a microprocessor. Further information regarding capability and use of the monitor may be found in the monitor software section of this manual.

In addition to the 2K ROM monitor, which resides at F000 to F7FF HEX, the SSZ-80 board will hold 14 K of ROM or PROM for a total 16K of read only memory on the board. This is divided into two sections. The lower segment is addressed from hexadecimal B000 to DFFF. This 12K segment is available for user EPROM (2716's) or for future ROM programs to be offered by Design Ltd. The remaining 2K segment is at F800 to FFFF HEX, and is intended for monitor extensions such as the disk and cassette operating systems which will also be offered by Design Ltd. in the future. This segment could also be used for user EPROM if desired.

There is 1K of read/write RAM on the board at E000 to E3FF HEX. This memory is intended primarily for user I/O routines and user commands and vectors from the monitor. Certain addresses within this 1K block are reserved for monitor functions, so the monitor RAM usage chart should be checked before changing data within, or writing programs utilizing, this RAM area.

All addresses from 0000 to AFFF HEX are open and available for user read/write RAM or ROM located on SS-50 BUSS memory boards.

There is a ten pin interrupt vector input port connector located at the top of the SSZ-80 board. This port is used to allow an interrupting device to specify the address of the interrupt service routine when the Z-80 processor is operating in interrupt mode 0 or 2. Using this vector input port with the powerful Z-80 interrupt structure allows the interrupt service routine to be located anywhere in memory.

Due to the complexity of the Z-80 interrupt structure, the Z-80 CPU technical manual should be studied carefully before attempting to use interrupts with the SSZ-80. It should be noted, however, that for interrupt Mode 1, non-maskable interrupt, and non-interrupting modes of operation no connections to the interrupt vector port are required.

The actual interrupt inputs to the SSZ-80 processor are indentical to those of the SS-50 6800 boards. The IRQ (interrupt request) and NMI (non-maskable interrupt) lines on the SS-50 mother board thus perform the same function in interrupting the processor whether the Z-80 or the 6800 processor is active. Existing interrupt strapping on periphial interface boards will therefore be the same with the SSZ-80 as they are with the 6800 system.

Compatibility with SWTPC Processor Boards

The SSZ-80 is designed to implement full Z-80 processor capability while still allowing continued use of the SWTPC MP-A or MP-A2 processor boards if desired. This is accomplished via a miniature toggle switch mounted on the front panel of the computer. When this switch is in the 6800 (down) position, all SSZ-80 board outputs to the address, data, and control busses are put in the tri-state mode. This allows the 6800 board to take control of the system. When the switch is in the Z-80 (up) position the situation is reversed and the SSZ-80 board has control.

When the 6800 processor board is enabled system operation will be exactly as with the 6800 processor only, even though the Z-80 board may be plugged into the computer.

The only lines which do not have tri-state control are the 5 baud rate output lines, pins 1 through 5 on the 50 pin edge connector. The SSZ-80 board is supplied with sockets to implement these baud rate outputs should the SSZ-80 be operated alone in the system. In this case, Option B may be purchased and installed on the SSZ-80 board. Alternatively, the MC14411 baud rate generator IC, the baud crystal, and the SN7404 baud rate drivers may be removed from the 6800 CPU board and be installed on the SSZ-80 board. If both CPU boards are to be operated together in the system then the baud rate components must be installed on only one of the CPU boards.

SSZ-80 Input/Output Requirements

The MC6800 processor has no independent I/O structure for perhipial devices, but rather uses memory addresses to select the different I/O devices. This is called memory-mapped I/O. In the case of the SWTPC computer, the memory block from 8000 through 8FFF HEX is reserved for I/O transfers. One undesireable effect of this is to limit contiguous memory in the system to a maximum of 32K bytes.

The Z-80 processor was designed to eliminate this undesireable waste of memory space by creating a separate 256 byte address block specifically for I/O transfers. These 256 locations are referenced by the input and output instructions of the Z-80 command set. (See the Z-80 technical manual for information regarding the use of these instruction.) Each of the 256 I/O locations is called an I/O port and they are addressed from 00 to FF HEX. In order to utilize the superior I/O structure of the Z-80 processor it was necessary to make a slight modification to the SS-50 mother board. With this modification, the same switch that selects the desired processor board also switches the periphial side of the mother board from memory address space to port address space when the SSZ-80 board is selected.

In SS-50 nomenclature, ports 0 through 7 represent card slots in the mother board. Each is selected by four consecutive addresses starting at 8000 HEX. Port 0 uses addresses 8000 - 8003, port 1 uses addresses 8004 - 8007, etc. As mentioned above however, each Z-80 I/O location is called a port, numbered 00 through FF. When using the Z-80 then, card slot 0 on the periphial side of the mother board is addressed by four Z-80 ports, numbered 0 through 3. Similarly the old port 1 is now selected by Z-80 ports 4 through 7, etc.

It is important to keep these nomenclature differences in mind because when we are talking about Z-80 software and refer to a port or port number, we will be referring to a single I/O location and not a group of four locations as in the case of the SS-50 BUSS nomenclature.

System Preparation for Initial Startup

The amount of preparation to effect initial startup is dependent upon the system configuration that the SSZ-80 will be operated in. The simplest system is one in which the SSZ-80 is the only CPU board ever to be used in the computer. In this case it is only necessary to perform Steps 1 through 3 of the SS-50 motherboard modification instructions, and to plug the SSZ-80 board into the SS-50 BUSS. However, the baud rate generator components will have to be installed on the SSZ-80 board in this configuration. This is done by soldering in the baud rate crystal at XTAL 1, plugging a MC14411 IC into IC socket U11, and plugging a SN7404 IC into IC socket U24 on the SSZ-80 board. These components may be purchased from Design Ltd. as Option B, or may be removed from the 6800 processor board.

Note that if the baud rate components are removed from the 6800 CPU board, it will still be possible to operate the 6800 board if the SSZ-80 is plugged into the SS-50 BUSS at the same time. All of the steps for mother board modification and the steps for modification of the SWTPC MP-A or MP-A2 CPU board will also have to be performed in this case.

If it is desired to operate both the 6800 and the SSZ-80 CPU boards in the system then all of the steps for modification of the motherboard should be performed first. Then the SWTPC MP-A or MP-A2 CPU board should be inserted into the computer and "powered up" with the Z-80/6800 selection switch in the 6800 (down) position. The 6800 board should work normally at this time. If not, go over the motherboard modification instructions again as an error there is the most likely problem.

Next the modifications to the 6800 CPU board should be done. Make sure that you use the proper instruction set for the version CPU board that you have. Again test the 6800 board for proper operation to be sure that no errors have occurred. If all is well, you may now plug in the SSZ-80 CPU board. (After removing power from the system.) With the CPU selection switch still in the 6800 position the 6800 CPU board should continue to operate normally.

SSZ-80 Options

There are two jumper locations on the SSZ-80 board. The board is normally supplied with these jumpers not installed. Jumper 1 is used to connect the buss acknowledge output from the SSZ-80 board to the SS-50 BA line. This line would normally be used when doing DMA to let the device requesting DMA know that the buss was clear for use. Since the 6800 CPU board also has a buss acknowledge output, jumper 1 should not be install on the SSZ-80 board unless the 6800 BA output has been disabled or the 6800 board is not installed in the system.

Jumper 2 may be used to connect the wait input to the Z-80 to the "UD2" line on the motherboard. This would allow the use of slow memory or peripheral devices with the system. The SS-50 memory board which has slow devices on it should generate a low on the wait line every time it is addressed. After the necessary time for the memory to stabilize has elapsed the wait line should be brought back high. The processor will then read or write data to the memory addressed. Processor wait states may also be generated for slow I/O or interrupting devices. Additional information on this can be found in the Z-80 technical manual.

SSZ-80 Initial Startup

Once the necessary preparation has been completed you are ready for initial startup of the SSZ-80 CPU. The most important factor for successful startup of the SSZ-80 is,the proper setting of the sense switches. These switches, located at the top edge of the SSZ-80 board, are used to tell the monitor software what the initial hardware interface conditions are. If the switches are in the wrong positions for your hardware configuration the SSZ-80 will not sign on. If you have a standard SWTPC system with your console device connected to a control interface which is plugged into periphial card Slot 1 (SS-50 Port 1), then the proper initial sense switch condition is for all 8 switches to be in the down position. It does not matter if the control interface is a parallel port as used with MIKBUG (R) or a serial port as may be used with SWTBUG (R); the SSZ-80 will work with either one. If you do not have a standard SWTPC system then you will have to refer to the I/O handling section of the monitor software section of this manual in order to determine what is required for initial startup.

When the above hardware considerations have been determined and met, plug the SSZ-80 board into the computer and apply power. If the CPU selection switch has been installed, place it in the Z-80 (up) position. Push the reset switch on the front of the computer and the ZAPPLE monitor will sign on. You are now ready to use your Z-80 computer system.

Z-80 Software Considerations

As mentioned earlier, the ZAPPLE monitor supplied with the SSZ-80 is one of the most powerful, versatile monitors available. The interaction that it provides between the operator and the computer goes a long way towards making it easy for you to become proficient in Z-80 software and computer operation. You should study the monitor software section of this manual thoroughly and become familiar with all monitor commands and functions, and how to use them. Merely reading about each command is not enough. "Play" with them on the computer until you are sure you know what they can do for you.

If you intend to do much assembly language programming, it is highly recommended that you purchase the TDL ZAPPLE® text editor and relocating MACRO assembler. This inexpensive software package will give you the most powerful assembly language software development system available for microprocessor use. A good way to start familarizing yourself with the Z-80 software is to study the Zapple monitor software source listing in this manual. It contains a wealth of practical Z-80 programming information. The Z-80 technical manual is an essential to understanding the Z-80 software. It provides an exact explanation of each of the Z-80 software instructions.

System Preparation for Initial Startup

The amount of preparation to effect initial startup is dependent upon the system configuration that the SSZ-80 will be operated in. The simplest system is one in which the SSZ-80 is the only CPU board ever to be used in the computer. In this case it is only necessary to perform Steps 1 through 3 of the SS-50 motherboard modification instructions, and to plug the SSZ-80 board into the SS-50 BUSS. However, the baud rate generator components will have to be installed on the SSZ-80 board in this configuration. This is done by soldering in the baud rate crystal at XTAL 1, plugging a MC14411 IC into IC socket U11, and plugging a SN7404 IC into IC socket U24 on the SSZ-80 board. These components may be purchased from Design Ltd. as Option B, or may be removed from the 6800 processor board.

Note that if the baud rate components are removed from the 6800 CPU board, it will still be possible to operate the 6800 board if the SSZ-80 is plugged into the SS-50 BUSS at the same time. All of the steps for mother board modification and the steps for modification of the SWTPC MP-A or MP-A2 CPU board will also have to be performed in this case.

If it is desired to operate both the 6800 and the SSZ-80 CPU boards in the system then all of the steps for modification of the motherboard should be performed first. Then the SWTPC MP-A or MP-A2 CPU board should be inserted into the computer and "powered up" with the Z-80/6800 selection switch in the 6800 (down) position. The 6800 board should work normally at this time. If not, go over the motherboard modification instructions again as an error there is the most likely problem.

Next the modifications to the 6800 CPU board should be done. Make sure that you use the proper instruction set for the version CPU board that you have. Again test the 6800 board for proper operation to be sure that no errors have occurred. If all is well, you may now plug in the SSZ-80 CPU board. (After removing power from the system.) With the CPU selection switch still in the 6800 position the 6800 CPU board should continue to operate normally.

SSZ-80 Options

There are two jumper locations on the SSZ-80 board. The board is normally supplied with these jumpers not installed. Jumper 1 is used to connect the buss acknowledge output from the SSZ-80 board to the SS-50 BA line. This line would normally be used when doing DMA to let the device requesting DMA know that the buss was clear for use. Since the 6800 CPU board also has a buss acknowledge output, jumper 1 should not be install on the SSZ-80 board unless the 6800 BA output has been disabled or the 6800 board is not installed in the system.

Jumper 2 may be used to connect the wait input to the Z-80 to the "UD2" line on the motherboard. This would allow the use of slow memory or periphial devices with the system. The SS-50 memory board which has slow devices on it should generate a low on the wait line every time it is addressed. After the necessary time for the memory to stabilize has elapsed the wait line should be brought back high. The processor will then read or write data to the memory addressed. Processor wait states may also be generated for slow I/O or interrupting devices. Additional information on this can be found in the Z-80 technical manual.

SSZ-80 Initial Startup

Once the necessary preparation has been completed you are ready for initial startup of the SSZ-80 CPU. The most important factor for successful startup of the SSZ-80 is,the proper setting of the sense switches. These switches, located at the top edge of the SSZ-80 board, are used to tell the monitor software what the initial hardware interface conditions are. If the switches are in the wrong positions for your hardware configuration the SSZ-80 will not sign on. If you have a standard SWTPC system with your console device connected to a control interface which is plugged into periphial card Slot 1 (SS-50 Port 1), then the proper initial sense switch condition is for all 8 switches to be in the down position. It does not matter if the control interface is a parallel port as used with MIKBUG (R) or a serial port as may be used with SWTBUG (R); the SSZ-80 will work with either one. If you do not have a standard SWTPC system then you will have to refer to the I/O handling section of the monitor software section of this manual in order to determine what is required for initial startup.

When the above hardware considerations have been determined and met, plug the SSZ-80 board into the computer and apply power. If the CPU selection switch has been installed, place it in the Z-80 (up) position. Push the reset switch on the front of the computer and the ZAPPLE monitor will sign on. You are now ready to use your Z-80 computer system.

Z-80 Software Considerations

As mentioned earlier, the ZAPPLE monitor supplied with the SSZ-80 is one of the most powerful, versatile monitors available. The interaction that it provides between the operator and the computer goes a long way towards making it easy for you to become proficient in Z-80 software and computer operation. You should study the monitor software section of this manual thoroughly and become familiar with all monitor commands and functions, and how to use them. Merely reading about each command is not enough. "Play" with them on the computer until you are sure you know what they can do for you.

If you intend to do much assembly language programming, it is highly recommended that you purchase the TDL ZAPPLE® text editor and relocating MACRO assembler. This inexpensive software package will give you the most powerful assembly language software development system available for microprocessor use. A good way to start familarizing yourself with the Z-80 software is to study the Zapple monitor software source listing in this manual. It contains a wealth of practical Z-80 programming information. The Z-80 technical manual is an essential to understanding the Z-80 software. It provides an exact explanation of each of the Z-80 software instructions.

Modifications to Early SWTPC CPU Boards

If it is desired to operate the SSZ-80 processor board in the SWTPC computer while the MP-A processor board is installed, there are several minor modifications which must be preformed. The purpose of these modifications is to allow the 6800 CPU to totally release all address, data, and control lines on the mother board whenever the 6800 is in the "halt" mode. This is necessary so that the Z-80 processor will have full control when it is enabled.

The halt line on the mother board of the SWTPC computer controls which CPU card will be active at any given time. When the halt line is high, the 6800 processor is enabled and all Z-80 outputs are floating. When the halt line is low the situation is reversed and the 6800 board outputs are floating. The halt line is called "BUSS REQUEST" on the SSZ-80 board.

1. Locate IC-12 and IC-16 on the 6800 board.
2. Cut the trace on the solder side of the board between IC-12 Pins 4 and 5.
3. Cut the trace on the component side of the board between IC-16 Pins 10 and 13.
4. On the solder side of the board, connect a short jumper wire (approximately 1" long) to IC-16, Pin 13.
5. On the solder side of the board, connect another short jumper (1" long) to IC-12, Pin 4.
6. Connect the free ends of both of the jumper wires installed in Steps 4 and 5 to IC-12, Pins 1 and 2.
7. Remove Pins 14 and 15 of IC-7 from the circuit board or IC socket. Bend these pins away from the body of the IC for use in Steps 8 and 9.
8. On the component side of the board, connect a wire from Pin 15 of IC-7 to the wide trace going from Pin 1 of IC 6 to Pin 15 of IC-5.
9. On the component side of the board, connect a wire from Pin 14 of IC-7 to Pin 40 of IC-1, the MC6800. (Second trace from top of board, component side.)
10. Cut the trace at Pin 9 of the 50 pin edge connector from the rest of the board.
11. On the solder side of the board, connect a wire from IC-7, Pin 13 to Pin 9 of the 50 pin edge connector. (Pin 9 is the reset line.)
12. Solder a 4.7K resistor from Pin 14 of IC-20 at C-14, to the trace coming from Pin 40 of IC-1, the MC6800. Do this on the component side of the board.

NOTE: This modification will not allow direct reset of the 6800 processor board from the reset line of the mother board during a "WAI" instruction. Pin 40 of the MC6800 must be grounded directly in order to reset the CPU in such instances.

Modifications to Later SWTPC CPU Boards

If it is desired to operate the SSZ-80 processor board in the SWTPC computer while the MP-A2 processor board is installed, there are several minor modifications which must be preformed. The purpose of these modifications is to allow the 6800 CPU to totally release all address, data, and control lines on the mother board whenever the 6800 is in the "halt" mode. This is necessary so that the Z-80 processor will have full control when it is enabled.

The halt line on the mother board of the SWTPC computer controls which CPU card will be active at any given time. When the halt line is high, the 6800 processor is enabled and all Z-80 outputs are floating. When the halt line is low the situation is reversed and the 6800 board outputs are floating. The halt line is called "BUSS REQUEST" on the SSZ-80 board.

1. Locate IC-10 and IC-11 and cut the trace on the component side of the board between these two IC's. This trace runs from Pin 15 of IC-11 to the ground plain located under the body of IC-10.
2. Solder a short wire jumper from Pin 15 of IC-11 to Pin 1 of IC-11. Do this on the solder side of the board.
3. On the solder side of IC-5, cut the ground plane from Pins 12 and 14. Do this so that Pins 12 and 14 are disconnected from each other and ground. (Refer to PC overlay for ground plane.)
4. Locate IC-11 Pin 3 on the solder side of the board. Cut the trace (on solder side) that runs from IC-11 Pin 3 to IC-14 Pin 5.
5. Solder a jumper wire (approximately 4" long) from IC-11 Pin 3 to IC-5 Pin 12. Do this on the solder side of the board.
6. Solder a jumper wire (approximately 4" long) to the trace going from IC-14 Pin 5 (cut in Step 4) to IC-5 Pin 11. Do this on the solder side of the board.
7. Cut the trace from IC-1 Pin 40 to IC-6 Pin 14 on the solder side of the board.
8. Solder a jumper wire from IC-6 Pin 14 to IC-5 Pin 14 on the solder side of the board.
9. Solder a jumper wire from IC-5 Pin 13 to IC-1 Pin 40 on the solder side of the board.

NOTE: This modification will not allow direct reset of the 6800 processor board from the reset line of the mother board during a "WAI" instruction. Pin 40 of the MC6800 must be grounded directly in order to reset the CPU in such instances.

Modifications to SWTPC Mother Boards

The following modifications must be preformed on the MP-B mother board in order to enable it to work with the Z-80 and with the MP-A processors.

1. Remove the MP-B mother board from the computer.
2. Locate IC-6, Pin 5 on the foil side of the board and cut the trace coming to it from the rest of the board.
3. Install a jumper wire from IC-6, Pin 5 to Pin 13 (UD1) of the nearest 50 Pin edge connector. Do this on the solder side of the board.
4. Solder four six inch lengths of wire to the front edge of the MP-B mother board as follows:

   A) One wire to Pin 6, the "halt" line.
   B) One wire to Pin 11, the "VMA" line.
   C) One wire to Pin 13, the "UD1" line.
   D) One wire to Pin 25, the ground buss.
5. Connect the wire from ground of the MP-B board to the center pin of one side of the double pole double throw miniature toggle switch supplied.
6. Connect the wire from the "halt" line to the bottom contact of the same side of the switch used in Step 5.
7. Connect the wire coming from "VMA" to the center and connect the wire from "UD1" to the upper contact of the other side of the switch.
8. Re-install the MP-B mother board in the computer. Then mount the switch at a convenient location on the front panel of the computer, keeping the "bottom" and "top" of the switch in mind. Dress any extra wire along the front corner of the computer cabinet.

This completes the modification of the SWTPC 6800 computer for operation with both the Z-80 and the 6800 CPU boards.

NOTE: If the SSZ-80 processor board is the only board to be used with the computer then Steps 4 through 8 of the above instructions may be omitted.

```
                        ;       <<< ZAPPLE 2-K MONITOR SYSTEM >>>

                        ;                   BY

                        ;       TECHNICAL DESIGN LABS, INC.
                        ;       RESEARCH PARK
                        ;       PRINCETON, NEW JERSEY 08540

                        ;       COPYRIGHT APRIL 1978 TDL INC.

                        ;       ASSEMBLED BY ROGER AMIDON
                        ;       MODIFIED BY RUSSELL PILLSBURY

                        .PHEX
                        .PABS
                        .XLINK

F000                    BASE    =\"STARTING ADDRESS?"

                        ;       <I/O DEVICES>

                        ; TELEPRINTER
0005                    TTI     = 5     ;DATA IN PORT
0007                    TTO     = 7     ;DATA OUT PORT
0004                    TTOP    = 4     ;PIA DATA PORT
0006                    TTS     = 6     ;STATUS PORT (IN)
0001                    TTYDA   = 1     ;DATA AVAILABLE MASK BIT
0002                    TTYBE   = 2     ;XMTR BUFFER EMPTY MASK

                        ; MAGTAPE SYSTEM
F800                    CASS    = 0F800H        ;TABLE VECTOR TO MAGTAPE
F803                    POUSR   = 0F803H        ;MAGTAPE PUNCH SUBROUTINE
F806                    RIUSR   = 0F806H        ;MAGTAPE READ SUBROUTINE
0012                    T.ON    = 12H           ;TAPE WRITE ON
0014                    T.OFF   = 14H           ;TAPE WRITE OFF
0011                    X.ON    = 11H           ;READER ON
0013                    X.OFF   = 13H           ;READER OFF

                        ;       <CONSTANTS>

E000                    USER    = 0E000H
E000                    IOBYT   = USER
00FF                    SENSE   = 0FFH          ;INPUT PORT FOR INITIAL I/O
0038                    RST7    = 38H
0000                    FALSE   = 0
FFFF                    TRUE    = # FALSE
000D                    CR      = 0DH           ;ASCII CARRIAGE RETURN
000A                    LF      = 0AH           ;ASCII LINE FEED
0008                    BS      = 8             ;BACK SPACE
0000                    FIL     = 00            ;FILL CHARACTERS AFTER CRLF
0007                    MAX     = 7             ;NUMBER OF QUES IN EOF

                        ;       <I/O CONFIGURATION MASKS>
```

```
00FC                    CMSK    = 11111100B     ;CONSOLE DEVICE
00F3                    RMSK    = 11110011B     ;STORAGE DEVICE (IN)
00CF                    PMSK    = 11001111B     ;STORAGE DEVICE (OUT)
003F                    LMSK    = 00111111B     ;LIST DEVICE


                        ; CONSOLE CONFIGURATION
0000                    CTTY    = 0     ;TELEPRINTER
0001                    CCRT    = 1     ;C.R.T.
0002                    BATCH   = 2     ;READER FOR INPUT, LIST FOR OUTPUT
0003                    CUSE    = 3     ;USER DEFINED

                        ; STORAGE INPUT CONFIGURATION
0000                    RTTY    = 0     ;TELEPRINTER READER
0004                    RPTR    = 4     ;HIGH-SPEED RDR (EXTERNAL ROUTINE)
0008                    RCAS    = 8     ;CASSETTE (EXTERNAL ROUTINE)
000C                    RUSER   = 0CH   ;USER DEFINED

                        ; STORAGE OUTPUT CONFIGURATION
0000                    PTTY    = 0     ;TELEPRINTER PUNCH
0010                    PPTP    = 10H   ;HIGH-SPEED PUNCH (EXTERNAL ROUTINE)
0020                    PCAS    = 20H   ;CASSETTE (EXTERNAL ROUTINE)
0030                    PUSER   = 30H   ;USER DEFINED

                        ; LIST DEVICE CONFIGURATION
0000                    LTTY    = 0     ;TELEPRINTER PRINTER
0040                    LCRT    = 40H   ;C.R.T. SCREEN (EXTERNAL ROUTINE)
0080                    LINE    = 80H   ;LINE PRINTER (EXTERNAL ROUTINE)
00C0                    LUSER   = 0C0H  ;USER DEFINED

                        ;       VECTORS FOR USER DEFINED ROUTINES

E020                    .LOC    USER+20H
E020                    CILOC:  .BLKB 3 ;CONSOLE INPUT
E023                    COLOC:  .BLKB 3 ;CONSOLE OUTPUT
E026                    CRTIN:  .BLKB 3 ;CRT KEYBOARD RTN.
E029                    CRTOUT: .BLKB 3 ;CRT OUTPUT ROUTINE
E02C                    CRTST:  .BLKB 3 ;CRT STATUS CHECK RTN.
E02F                    LPNTR:  .BLKB 3 ;LINE PRINTER OUTPUT ROUTINE
E032                    RPTPL:  .BLKB 3 ;PAPER TAPE READER ROUTINE
E035                    RULOC:  .BLKB 3 ;USER DEFINED STORAGE (INPUT)
E038                    PTPL:   .BLKB 3 ;HIGH-SPEED PUNCH
E03B                    PULOC:  .BLKB 3 ;USER DEFINED STORAGE (OUTPUT)
E03E                    LULOC:  .BLKB 3 ;USER DEFINED PRINTER
E041                    CSLOC:  .BLKB 3 ;CONSOLE INPUT STATUS ROUTINE
E044                    J =.

                        ;       PROGRAM CODE BEGINS HERE

F000                    .LOC    BASE
F000    C3 F0E4                 JMP BEGIN       ;GO AROUND VECTORS
```

```
                        .REMARK

                                <VECTORS FOR CALLING PROGRAMS>

                        THESE VECTORS MAY BE USED BY USER WRITTEN
                        PROGRAMS TO SIMPLIFY THE HANDLING OF I/O
                        FROM SYSTEM TO SYSTEM.  WHATEVER THE CURRENT
                        ASSIGNED DEVICE, THESE VECTORS WILL PERFORM
                        THE REQUIRED I/O OPERATION AND RETURN TO THE
                        CALLING PROGRAM.

                        REGISTER CONVENTIONS:

                        ANY INPUT OR OUTPUT DEVICE -
                                CHARACTER TO BE OUTPUT IN 'C' REGISTER.
                                CHARACTER WILL BE IN 'A' REGISTER UPON
                                RETURNING FROM AN INPUT OR OUTPUT.
                        CSTS -
                                RETURNS TRUE (0FFH IN 'A' REG.) IF THERE IS
                                A CONSOLE INPUT CHARACTER WAITING, AND FALSE
                                (0 IN 'A' REG.) IF NOT.
                        IOCHK -
                                RETURNS WITH THE CURRENT I/O CONFIGURATION
                                BYTE IN THE 'A' REGISTER.
                        IOSET -
                                ALLOWS A PROGRAM TO DYNAMICALLY ALTER THE
                                I/O CONFIGURATION.  REQUIRES THE NEW BYTE
                                IN THE 'C' REGISTER.
                        MEMCK -
                                RETURNS WITH THE HIGHEST ALLOWED USER
                                MEMORY ADDRESS.  'B'=HIGH BYTE, 'A'=LOW BYTE.
                        TRAP -
                                THIS IS THE BREAKPOINT ENTRY POINT, BUT MAY
                                BE CALLED.  IT WILL SAVE THE MACHINE STATE.
                                RETURN IS MADE WITH A 'G[CR]' ON THE CONSOLE.

F003    C3 F6D3                 JMP CI          ;CONSOLE INPUT
F006    C3 F70C                 JMP RI          ;READER INPUT
F009    C3 F54C                 JMP CO          ;CONSOLE OUTPUT
F00C    C3 F423                 JMP PO          ;PUNCH OUTPUT
F00F    C3 F584                 JMP LO          ;LIST OUTPUT
F012    C3 F652                 JMP CSTS        ;CONSOLE STATUS
F015    C3 F1DB                 JMP IOCHK       ;I/O CHECK
F018    C3 F1D6                 JMP IOSET       ;I/O SET
F01B    C3 F0C2                 JMP MEMCK       ;MEMORY LIMIT CHECK

F01E    F5              TRAP:   PUSH PSW        ;SAVE USER'S ACCUMULATOR
F01F    ED57                    LDAI            ;GET INTERUPT CONDITION
F021    F3                      DI              ;SHUT OFF INTERUPTS
F022    E3                      XTHL            ;HL=USER ACC, SP=USER HL
F023    D5                      PUSH D          ;PUSH ALL REGISTERS
F024    C5                      PUSH B
```

```
F025   E5                    PUSH H          ;USER'S ACC.
F026   F5                    PUSH PSW        ;GET INTERUPT CONDITION
F027   C1                    POP B           ;IN C REG.
F028   ED5F                  LDAR            ;GET REFRESH REG.
F02A   E67F                  ANI 7FH         ;CLEAR BIT 7
F02C   CB51                  BIT 2, C        ;TEST PARITY FLAG
F02E   2802                  JRZ .+4         ;INTERUPTS WERE OFF
F030   F680                  ORI 80H         ;ELSE SET BIT 7
F032   ED4F                  STAR            ;AND SAVE IN "R" REG.
F034   11 F039               LXI D, ..TR0    ;SET-UP A RETURN
F037   186B                  JMPR MEMSIZ+1   ;& GET MONITOR'S STACK AREA
F039   21 000A       ..TR0:  LXI H, 10       ;GO UP 10 BYTES IN STACK
F03C   39                    DAD SP
F03D   0604                  MVI B, 4        ;PICK OFF REG.
F03F   EB                    XCHG            ;HL=MONITOR STACK, DE=USER'S+1
F040   2B             ..TR1: DCX H
F041   72                    MOV M, D        ;SAVE IN WORKAREA
F042   2B                    DCX H
F043   73                    MOV M, E
F044   D1                    POP D
F045   10F9                  DJNZ ..TR1
F047   C1                    POP B
F048   0B                    DCX B           ;ADJUST P.C. VALUE
F049   F9                    SPHL            ;SET MONITOR STACK
F04A   CD F5FE               CALL INTCK      ;SEE IF INTERRUPTS ALLOWED
F04D   21 0025               LXI H, TLOCX
F050   39                    DAD SP
F051   CD F09C               CALL ..TR6      ;TEST FOR A SET TRAP
F054   23                    INX H
F055   23                    INX H
F056   C4 F09C               CNZ ..TR6       ;TEST FOR 2ND TRAP
F059   2801                  JRZ ..TR2
F05B   03                    INX B           ;NO TRAPS SET, RE-ADJUST P.C.
F05C   21 001F       ..TR2:  LXI H, LLOCX
F05F   39                    DAD SP
F060   73                    MOV M, E        ;STORE USER H&L
F061   23                    INX H
F062   72                    MOV M, D
F063   23                    INX H
F064   23                    INX H
F065   23                    INX H
F066   71                    MOV M, C        ;AND USER P.C.
F067   23                    INX H
F068   70                    MOV M, B
F069   21 0025               LXI H, TLOCX
F06C   39                    DAD SP
F06D   C5                    PUSH B
F06E   01 0200               LXI B, 200H
F071   5E             ..TR3: MOV E, M        ;REPLACE BYTES TAKEN FOR TRAP
F072   71                    MOV M, C        ;ZERO OUT STORAGE AREA
F073   23                    INX H
F074   56                    MOV D, M
F075   71                    MOV M, C
F076   23                    INX H
F077   7B                    MOV A, E
```

```
F078    B2                      ORA D           ;DO NOTHING IF ZERO
F079    2802                    JRZ ..TR4
F07B    7E                      MOV A,M
F07C    12                      STAX D          ;STORE BYTE
F07D    23              ..TR4:  INX H           ;SAME THING
F07E    10F1                    DJNZ ..TR3      ;  FOR OTHER BREAKPOINT
F080    08                      EXAF            ;GET ALTERNATE SET OF REG.'S
F081    D9                      EXX
F082    E3                      XTHL            ;AND STORE IN WORKSPACE
F083    D5                      PUSH D
F084    C5                      PUSH B
F085    F5                      PUSH PSW
F086    DDE5                    PUSH X
F088    FDE5                    PUSH Y
F08A    ED5F                    LDAR            ;GET REFRESH BYTE
F08C    4F                      MOV C,A
F08D    ED57                    LDAI            ;GET INTERUPT VECTOR BYTE
F08F    47                      MOV B,A
F090    C5                      PUSH B          ;SAVE
F091    0E40                    MVI C, '@'      ;DISPLAY BREAK ADDRESS.
F093    CD F54C                 CALL CO
F096    CD F4CA                 CALL LADR
F099    C3 F12A                 JMP START       ;BACK TO START

F09C    7E              ..TR6:  MOV A,M
F09D    91                      SUB C           ;LOOK FOR A TRAP/MATCH
F09E    23                      INX H
F09F    C0                      RNZ
F0A0    7E                      MOV A,M
F0A1    90                      SUB B
F0A2    C9                      RET

F0A3    D1              MEMSIZ: POP D           ;KEEP RETURN IN DE
F0A4    21 FFFF                 LXI H, -1       ;RAM SEARCH STARTING PT.
F0A7    F3                      DI              ;DISABLE INTERUPTS
F0A8    24              ..M0:   INR H           ;FIRST FIND R/W MEMORY
F0A9    7E                      MOV A,M
F0AA    2F                      CMA
F0AB    77                      MOV M,A
F0AC    BE                      CMP M
F0AD    2F                      CMA
F0AE    77                      MOV M,A
F0AF    20F7                    JRNZ ..M0
F0B1    24              ..M1:   INR H           ;R/W FOUND, NOW FIND END
F0B2    7E                      MOV A,M
F0B3    41                      MOV B,C         ;KEEP PREVIOUS 'M' IN B
F0B4    4F                      MOV C,A
F0B5    2F                      CMA
F0B6    77                      MOV M,A
F0B7    BE                      CMP M
F0B8    71                      MOV M,C         ;REPLACE THE BYTE
F0B9    28F6                    JRZ ..M1        ;NOT THERE YET
F0BB    78                      MOV A,B         ;GET PREVIOUS 'M'
F0BC    01 FEDD         ..M2:   LXI B, (EXIT-ENDX)-256
F0BF    09                      DAD B
```

```
F0C0    EB                      XCHG            ;VALUE IN DE
F0C1    E9                      PCHL            ;RETURN

F0C2    E5              MEMCK:  PUSH H          ;SAVE HL
F0C3    D5                      PUSH D          ;SAVE DE
F0C4    C5                      PUSH B          ;SAVE C
F0C5    ED5F                    LDAR            ;TEST INTERUPT CONDITION
F0C7    F5                      PUSH PSW        ;SAVE IT
F0C8    CD F0A3                 CALL MEMSIZ
F0CB    F1                      POP PSW         ;RECOVER INTERUPT CONDITION
F0CC    C1                      POP B
F0CD    42                      MOV B,D         ;USER'S HIGH BYTE
F0CE    D1                      POP D
F0CF    E1                      POP H
F0D0    3EA0                    MVI A, 0A0H     ;USER'S LOW BYTE
F0D2    E0                      RPO             ;INTERUPTS NOT IN USE
F0D3    FB                      EI              ;ELSE ENABLE INTERUPTS
F0D4    C9                      RET



F0D5    0D0A0000        MSG:    .BYTE   CR,LF,FIL,FIL
F0D9    5A4150504C45            .ASCII  'ZAPPLE V2.R'
000F                    MSGL    =.-MSG
```

```
F0E4    DBFF        BEGIN:  IN SENSE        ;INITIALIZE I/O CONFIGURATION
F0E6    32 E000             STA IOBYT
F0E9    31 E3FF             LXI SP, USER+3FFH       ;SET TEMP STACK
F0EC    3E07                MVI A, 7
F0EE    D304                OUT TTOP        ;SET DDRA
F0F0    D305                OUT TTOP+1      ;CLOSE DDR
F0F2    D304                OUT TTOP        ;SET MARK
F0F4    D306                OUT TTS         ;SET DDRB
F0F6    3E3C                MVI A, 3CH
F0F8    D307                OUT TTS+1       ;CLOSE DDRB, ECHO OFF
F0FA    CD F730             CALL PORA       ;IS IT THE PIA ?
F0FD    2008                JRNZ STSTK      ;  IF YES
F0FF    3E03                MVI A, 3        ;  ELSE INIT ACIA
F101    D306                OUT TTS         ;RESET
F103    3E51                MVI A, 51H
F105    D306                OUT TTS         ;INIT ACIA
F107    CD F0A3     STSTK:  CALL MEMSIZ     ;GET MONITOR'S STACK AREA
F10A    EB                  XCHG
F10B    F9                  SPHL            ;SET TRUE STACK
F10C    CD F5FE             CALL INTCK      ;SEE IF INTERUPTS ALLOWED
F10F    EB                  XCHG
F110    01 0023             LXI B, ENDX-EXIT
F113    21 F7AB             LXI H, EXIT
F116    EDB0                LDIR            ;MOVE TO RAM
F118    21 FFA1             LXI H, -5FH     ;SET UP A USER'S STACK VALUE
F11B    19                  DAD D
F11C    E5                  PUSH H          ;PRE-LOAD STACK VALUE
F11D    21 0000             LXI H, 0        ;INITIALIZE OTHER REGISTERS
F120    060A                MVI B, 10       ;  (16 OF THEM)
F122    E5          ..B2:   PUSH H          ;  TO ZERO
F123    10FD                DJNZ ..B2
F125    060F                MVI B, MSGL     ;SAY HELLO TO THE FOLKS
F127    CD F5BE             CALL TOM        ;OUTPUT SIGN-ON MSG
F12A    11 F12A     START:  LXI D, START    ;MAIN 'WORK' LOOP
F12D    D5                  PUSH D          ;SET UP A RETURN TO HERE
F12E    CD F5F4             CALL CRLF
F131    0E2E                MVI C, '.'
F133    CD F54C             CALL CO
F136    21 F14D             LXI H, TBL      ;POINT TO INTERNAL TABLE
F139    CD F785     STAR0:  CALL TI         ;GET A CONSOLE CHARACTER
F13C    28FB                JRZ STAR0       ;GET ANOTHER IF ZERO
F13E    D641                SUI 'A'         ;QUALIFY THE CHARACTER
F140    D8                  RC              ;<A
F141    FE1A                CPI 'Z'-'A'+1
F143    30F4                JRNC STAR0      ;>Z
F145    87                  ADD A           ;A*2
F146    85                  ADD L
F147    6F                  MOV L,A         ;POINT TO PLACE ON TABLE
F148    7E                  MOV A,M
F149    23                  INX H
F14A    66                  MOV H,M
F14B    6F                  MOV L,A
F14C    E9                  PCHL            ;GO EXECUTE COMMAND
```

```
                        ;       <COMMAND BRANCH TABLE>


F14D                    TBL:

F14D    F199            .WORD   ASSIGN  ;A - ASSIGN I/O
F14F    F18D            .WORD   BRANCH  ;B - BRANCH TO USER TABLE "B.[A-Z]"
F151    F800            .WORD   CASS    ;C - MAGTAPE OPERATING SYSTEM
F153    F1DF            .WORD   DISP    ;D - DISPLAY MEMORY ON CONS.  IN HEX
F155    F3E9            .WORD   EOF     ;E - END OF FILE TAG FOR HEX DUMPS
F157    F181            .WORD   FILL    ;F - FILL MEMORY WITH A CONSTANT
F159    F1F5            .WORD   GOTO    ;G - GOTO [ADDR]<, >BREAKPOINTS (2)
F15B    F4BE            .WORD   HEXN    ;H - HEX MATH.  <SUM>, <D)FFERENCE>
F15D    E044            .WORD   J       ;I * USER DEFINED
E047                            J=J+3
F15F    F24C            .WORD   TEST    ;J - NON-DESTRUCTIVE MEMORY TEST
F161    E047            .WORD   J       ;K * USER DEFINED
E04A                            J=J+3
F163    F4DD            .WORD   LOAD    ;L - LOAD A BINARY FORMAT FILE
F165    F265            .WORD   MOVE    ;M - MOVE BLOCKS OF MEMORY
F167    F6D0            .WORD   NULL    ;N - PUNCH NULLS ON PUNCH DEVICE
F169    F5B7            .WORD   POUT    ;O - O<N)=OUTPUT TO PORT N
F16B    F517            .WORD   PUTA    ;P - 'PUT' ASCII INTO MEMORY.
F16D    F59F            .WORD   QUERY   ;Q - Q<N)=DISPLAY PORT N
F16F    F270            .WORD   READ    ;R - READ A HEX FILE <W/CHECKSUMS)
F171    F345            .WORD   SUBS    ;S - SUBSTITUTE &/OR EXAMINE MEMORY
F173    F36C            .WORD   TYPE    ;T - TYPE MEMORY IN ASCII
F175    F6BB            .WORD   UNLD    ;U - MEMORY TO PUNCH <BINARY FORMAT)
F177    F38C            .WORD   VERIFY  ;V - COMPARE MEMORY AGAINST MEMORY
F179    F39D            .WORD   WRITE   ;W - MEMORY TO PUNCH <HEX FORMAT)
F17B    F438            .WORD   XAM     ;X - EXAMINE & MODIFY CPU REGISTERS
F17D    F746            .WORD   WHERE   ;Y - FIND SEQUENCE OF BYTES IN MEM.
F17F    F53A            .WORD   SIZE    ;Z - ADDRESS OF LAST R/W LOCATION

E080                    UTAB    = USER+80H
```

```
F181    CD F642     FILL:   CALL EXP3       ;GET 3 PARAMETERS
F184    71          ..F:    MOV M,C         ;STORE THE BYTE
F185    CD F677             CALL HILO
F188    30FA                JRNC ..F
F18A    D1                  POP D           ;RESTORE STACK
F18B    189D                JMPR START      ; IN CASE OF ACCIDENTS

F18D    CD F785     BRANCH: CALL TI         ;GET A "."
F190    FE2E                CPI "."
F192    2037                JRNZ ERRX
F194    21 E080             LXI H, UTAB     ;POINT TO USER TABLE
F197    18A0                JMPR STAR0

F199    CD F785     ASSIGN: CALL TI         ;GET DEVICE NAME
F19C    21 F796             LXI H, LTBL-1   ;POINT TO DEVICE TABLE
F19F    11 0004             LXI D, 4        ;4 DEV.
F1A2    CD F1C1             CALL ..A3       ;GET DEVICE
F1A5    C5                  PUSH B
F1A6    CD F785     ..A1:   CALL TI         ;SCAN PAST '='
F1A9    D63D                SUI '='
F1AB    20F9                JRNZ ..A1
F1AD    5F                  MOV E,A         ;CLEAR E
F1AE    CD F785             CALL TI         ;GET NEW ASSIGNMENT
F1B1    CD F1C1             CALL ..A3       ;GET IT
F1B4    D1                  POP D           ;E=DEVICE
F1B5    2603                MVI H, 3        ;SET UP A MASK
F1B7    69                  MOV L,C         ;L=ASSIGNMENT
F1B8    7B                  MOV A,E         ;GET DEVICE
F1B9    3D          ..A2:   DCR A           ;DEVICE IN A
F1BA    FA F1CE             JM ASET         ;GOT IT
F1BD    29                  DAD H           ;DOUBLE LEFT SHIFT
F1BE    29                  DAD H           ; MASK & ASSIGNMENT
F1BF    18F8                JMPR ..A2
F1C1    01 0400     ..A3:   LXI B, 400H     ;4 DEVICES TO LOOK FOR
F1C4    23          ..A4:   INX H           ;POINT TO ASSIGNMENT NAME
F1C5    BE                  CMP M           ;LOOK FOR PROPER MATCH
F1C6    C8                  RZ              ;MATCH FOUND
F1C7    19                  DAD D           ;POINT TO NEXT
F1C8    0C                  INR C           ;KEEP TRACK OF ASSIGNMENT NMBR
F1C9    10F9                DJNZ ..A4
F1CB    C3 F5D2     ERRX:   JMP ERROR       ;NO MATCH, ERROR
F1CE    AC          ASET:   XRA H           ;INVERT FOR AND'ING
F1CF    67                  MOV H,A         ;SAVE IN H
F1D0    CD F1DB             CALL IOCHK      ;GET PRESENT CONFIGURATION
F1D3    A4                  ANA H           ;MODIFY ONLY SELECTED DEVICE
F1D4    B5                  ORA L           ;'OR' IN NEW BIT PATTERN
F1D5    4F                  MOV C,A         ;NEW CONFIGURATION

F1D6    79          IOSET:  MOV A,C         ;NEW I/O BYTE PASSED IN C REG
F1D7    32 E000             STA IOBYT       ;SAVE IN MEMORY
F1DA    C9                  RET

F1DB    3A E000     IOCHK:  LDA IOBYT       ;GET SAVED BYTE
F1DE    C9                  RET
```

```
F1DF    3E10        DISP:   MVI A, 16       ;SET TO DEFAULT
F1E1    CD F623             CALL EXPC       ;GET DISPLAY RANGE
F1E4    CD F535     ..D0:   CALL LFADRX     ;CRLF & PRINT ADDR.
F1E7    CD F54A     ..D1:   CALL BLK        ;SPACE OVER
F1EA    7E                  MOV A,M
F1EB    CD F4CF             CALL LBYTE
F1EE    CD F671             CALL HILOX      ;RANGE CHECK
F1F1    10F4                DJNZ ..D1
F1F3    18EF                JMPR ..D0       ;TIME TO CRLF

F1F5    CD F6AC     GOTO:   CALL PCHK       ;GET A POSSIBLE ADDRESS
F1F8    280B                JRZ ..G0        ;DELIMETER ENTERED
F1FA    CD F606             CALL EXF        ;GET ONE EXPRESSION
F1FD    D1                  POP D
F1FE    21 0034             LXI H, PLOC     ;PLACE ADDRESS IN 'P' LOCATION
F201    39                  DAD SP
F202    72                  MOV M,D         ;HIGH BYTE
F203    2B                  DCX     H
F204    73                  MOV M,E         ;LOW BYTE
F205    FE0D        ..G0:   CPI CR          ;SEE IF LAST CHARACTER WAS CR
F207    282C                JRZ ..G4        ;YES, LEAVE
F209    1602                MVI D, 2        ;TWO BREAKPOINTS MAX
F20B    21 0035             LXI H, TLOC     ;POINT TO TRAP STORAGE
F20E    39                  DAD SP
F20F    E5          ..G1:   PUSH H          ;SAVE STORAGE POINTER
F210    CD F603             CALL EXPR       ;GET A TRAP ADDRESS
F213    C1                  POP B           ;TRAP ADDR.
F214    E1                  POP H           ;STORAGE
F215    F5                  PUSH PSW        ;SAVE DELIMETER
F216    78                  MOV A,B         ;LOOK AT TRAP ADDR
F217    B1                  ORA C
F218    280A                JRZ ..G2        ;DON'T SET A TRAP AT 0
F21A    71                  MOV M,C         ;SAVE BKPT ADDR
F21B    23                  INX H
F21C    70                  MOV M,B
F21D    23                  INX H
F21E    0A                  LDAX    B       ;PICK UP INST. BYTE
F21F    77                  MOV M,A         ;SAVE THAT TOO
F220    23                  INX H
F221    3EFF                MVI A, 0FFH     ;RST 7
F223    02                  STAX B          ;SOFTWARE INTERUPT
F224    F1          ..G2:   POP PSW         ;LOOK AT DELIMITER
F225    3803                JRC ..G3
F227    15                  DCR D           ;COUNT BKPTS
F228    20E5                JRNZ ..G1       ;GET ONE MORE
F22A    3EC3        ..G3:   MVI A, JMP      ;SET UP JMP INSTRUCTION
F22C    32 0038             STA RST7        ;  AT RESTART TRAP LOC.
F22F    21 F01E             LXI H, TRAP     ;  TO MONITOR VECTOR
F232    22 0039             SHLD    RST7+1
F235    CD F5F4     ..G4:   CALL CRLF
F238    D1                  POP D           ;CLEAR SYSTEM RETURN
F239    D1                  POP D           ;GET 'R' REG
F23A    D5                  PUSH D          ;IN E
F23B    CB7B                BIT 7, E        ;SEE IF 'EI' OR 'DI'
```

```
F23D    21 002F             LXI H, INTLOC
F240    39                  DAD SP          ;POINT TO OPCODE
F241    36F3                MVI M, (DI)     ;INITIALIZE TO DI
F243    2802                JRZ ..G5
F245    36FB                MVI M, (EI)     ;ENABLE INTERUPTS
F247    21 0016     ..G5:   LXI H, 16H      ;FIND 'EXIT' ROUTINE
F24A    39                  DAD SP          ;UP IN STACK
F24B    E9                  PCHL            ;GO SOMEPLACE

F24C    CD F623     TEST:   CALL EXPC       ;GET TWO PARAMS
F24F    7E          ..T1:   MOV A,M         ;READ A BYTE
F250    47                  MOV B,A         ;SAVE IN B REG.
F251    2F                  CMA
F252    77                  MOV M,A         ;READ/COMPLIMENT/WRITE
F253    AE                  XRA M           ;  & COMPARE
F254    70                  MOV M,B         ;REPLACE BYTE
F255    2809                JRZ ..T2        ;SKIP IF ZERO (OK)
F257    D5                  PUSH D          ;SAVE END POINTER
F258    5F                  MOV E,A         ;SET-UP TO DISPLAY
F259    CD F547             CALL HLSP       ;PRINT BAD ADDR
F25C    CD F5A5             CALL BITS       ;PRINT BAD BIT LOC.
F25F    D1                  POP D           ;RESTORE DE
F260    CD F671     ..T2:   CALL HILOX      ;RANGE TEST
F263    18EA                JMPR ..T1

F265    CD F642     MOVE:   CALL EXP3       ;GET 3 PARAMETERS
F268    7E          ..M:    MOV A,M         ;PICK UP
F269    02                  STAX B          ;PUT DOWN
F26A    03                  INX B           ;MOVE UP
F26B    CD F671             CALL HILOX      ;CHECK IF DONE
F26E    18F8                JMPR ..M

F270    CD F603     READ:   CALL EXPR       ;GET BIAS BASE [0]
F273    D1                  POP D           ;DE=BIAS
F274    21 0000             LXI H, 0        ;SET UP A DEFAULT BASE[1]
F277    E5                  PUSH H          ;SAVE DEFAULT BASE[1]
F278    380C                JRC ..R0        ;CR ENTERED
F27A    CD F603             CALL EXPR       ;GET RELOCATION BASE[1]
F27D    E1                  POP H           ;ACTUAL RELOCATION VALUE
F27E    3806                JRC ..R0        ;CR ENTERED
F280    E3                  XTHL            ;REPLACE BASE[1]
F281    CD F603             CALL EXPR       ;GET BASE[2]
F284    E1                  POP H           ;ACTUAL BASE[2]
F285    E3                  XTHL            ;SP=BASE[2], HL=BASE[1]
F286    0E11        ..R0:   MVI C, X.ON     ;TURN ON READER
F288    CD F54C             CALL CO         ;THROUGH CONSOLE
F28B    C1                  POP B           ;BC=BASE[2]
F28C    D9                  EXX             ;IN PRIME SET.
F28D    CD F5F4             CALL CRLF       ;BEGIN READING FILE
F290    CD F725     ..R1:   CALL RIX        ;GET A CHARACTER
F293    D63A                SUI ':'         ;ABSOLUTE FILE CUE?
F295    47                  MOV B,A         ;SAVE CUE CLUE
F296    E6FE                ANI 0FEH        ;KILL BIT 0
F298    20F6                JRNZ ..R1       ;  NO, KEEP LOOKING
F29A    57                  MOV D,A         ;ZERO CHECKSUM
```

```
F29B    CD F332             CALL . .BYTE     ;GET FILE LENGTH
F29E    5F                  MOV E,A          ;SAVE IN E REG.
F29F    CD F332             CALL . .BYTE     ;GET LOAD MSB
F2A2    F5                  PUSH PSW         ;SAVE IT
F2A3    CD F332             CALL . .BYTE     ;GET LOAD LSB
F2A6    E1                  POP H            ;H=LOAD MSB
F2A7    6F                  MOV L,A          ;L=LOAD LSB
F2A8    CD F332             CALL . .BYTE     ;GET LOAD BASE
F2AB    B7                  ORA A            ;TEST IT
F2AC    F5                  PUSH PSW         ;SAVE LOAD BASE IN SP
F2AD    C4 F318             CNZ . .FLIP      ;0=ABS. LOAD
F2B0    F1                  POP PSW          ;GET LOAD BASE BACK
F2B1    2001                JRNZ . .R1A
F2B3    3C                  INR A            ;FORCE BASE[1]
F2B4    1C          . .R1A: INR E            ;TEST FILE LENGTH
F2B5    1D                  DCR E
F2B6    2856                JRZ . .DONE      ;ZERO FILE LENGTH
F2B8    F5                  PUSH PSW         ;SAVE CURRENT LOAD BASE(SP)
F2B9    AF                  XRA A            ;SET A FOR BIAS
F2BA    CD F318             CALL . .FLIP     ;ADD BIAS TO LOAD
F2BD    78                  MOV A,B          ;TEST CUE
F2BE    3D                  DCR A            ;Z=TDL FILE, NZ=INTEL
F2BF    2034                JRNZ . .R7       ;INTEL TYPE FILE
F2C1    CD F2FF     . .R2:  CALL . .R8       ;GET A DATA BYTE
F2C4    301E                JRNC . .R4       ;NORMAL DATA BYTE
F2C6    08                  EXAF             ;GET A'
F2C7    F1                  POP PSW          ;SET TO CURRENT LOAD BASE
F2C8    F5                  PUSH PSW         ;SAVE AGAIN
F2C9    08                  EXAF             ;SAVE A'
F2CA    E5                  PUSH H           ;SAVE LOAD ADDR.
F2CB    6F                  MOV L,A          ;SAVE AS LOW BYTE
F2CC    CD F2FF             CALL . .R8       ;GET MSB. ?.
F2CF    3008                JRNC . .R3       ;USE LOAD BASE
F2D1    08                  EXAF             ;ELSE UPDATE BASE IDENT
F2D2    7D                  MOV A,L          ;IN L
F2D3    08                  EXAF
F2D4    6F                  MOV L,A          ;NEW LOW BYTE
F2D5    CD F2FF             CALL . .R8       ;GET HIGH BYTE
F2D8    1D                  DCR E            ;ACCOUNT FOR EXTRAS
F2D9    67          . .R3:  MOV H,A          ;HL=.WORD
F2DA    08                  EXAF             ;GET BASE
F2DB    CD F318             CALL . .FLIP     ;RELOCATE IT
F2DE    7D                  MOV A,L          ;NOW WRITE TO MEMORY
F2DF    E3                  XTHL             ;GET LOAD ADDRESS
F2E0    CD F32E             CALL . .SAVE     ;WRITE LSB
F2E3    F1                  POP PSW          ;MSB
F2E4    CD F32E     . .R4:  CALL . .SAVE     ;WRITE IT
F2E7    20D8                JRNZ . .R2       ;MORE TO GO
F2E9    F1          . .R5:  POP PSW          ;REMOVE LOAD BASE
F2EA    CD F332             CALL . .BYTE     ;TEST CHECKSUM
F2ED    28A1                JRZ . .R1        ;OK, CONTINUE
F2EF    CD F544     . .R6:  CALL LFADR       ; ELSE PRINT LOAD ADDR
F2F2    C3 F5D2             JMP ERROR        ; & ABORT
F2F5    CD F332     . .R7:  CALL . .BYTE     ;GET A DATA BYTE
F2F8    CD F32E             CALL . .SAVE     ;STORE IT
```

```
F2FB   20F8              JRNZ ..R7        ;MORE TO GO
F2FD   18EA              JMPR ..R5        ; ELSE TEST CHECKSUM
F2FF   1007      ..R8:   DJNZ ..R9        ;TEST BIT/BYTE COUNT
F301   CD F332           CALL ..BYTE      ;GET NEW RELOC. MAP
F304   1D                DCR E            ;INCLUDE IN BYTE COUNT
F305   4F                MOV C,A          ;SAVE IN C
F306   0608              MVI B, 8         ;RESET B FOR NEXT 8
F308   CD F332   ..R9:   CALL ..BYTE      ;GET DATA BYTE
F30B   CB21              SLAR C           ;TEST FOR RELOC.
F30D   C9                RET

F30E   0E13      ..DONE: MVI C, X.OFF     ;TURN OFF READER
F310   CD F54C           CALL CO          ;THROUGH CONSOLE
F313   7C                MOV A,H          ;DON'T JUMP IF ZERO
F314   B5                ORA L
F315   C8                RZ
F316   E3                XTHL             ;JUMP VECTOR IN STACK
F317   C9                RET

F318   D9        ..FLIP: EXX              ;GET PRIME SET
F319   B7                ORA A            ;TEST BASE REQUEST
F31A   280D              JRZ ..DE         ;BIAS
F31C   3D                DCR A            ;1
F31D   280C              JRZ ..HL         ;BASE[1]
F31F   3D                DCR A            ;2
F320   20CD              JRNZ ..R6        ;BASE >2, ABORT
F322   C5                PUSH B           ;BASE[2]
F323   D9        ..DAD:  EXX              ;GET MAIN REG'S BACK
F324   EB                XCHG             ;PRESERVE DE
F325   E3                XTHL             ;DAD(SP)
F326   19                DAD D            ;.WORD+BASE[N]
F327   D1                POP D            ;RESTORE DE
F328   C9                RET              ;HL=.WORD+BASE

F329   D5        ..DE:   PUSH D           ;BIAS
F32A   FE                .BYTE   (CPI)
F32B   E5        ..HL:   PUSH H           ;BASE[1]
F32C   18F5              JMPR ..DAD       ;CONTINUE

F32E   77        ..SAVE: MOV M,A          ;WRITE TO MEMORY
F32F   23                INX H            ;BUMP POINTERS
F330   1D                DCR E
F331   C9                RET

F332   C5        ..BYTE: PUSH B           ;PRESERVE BC
F333   CD F694           CALL RIBBLE      ;GET A CONVERTED ASCII CHAR.
F336   07                RLC
F337   07                RLC
F338   07                RLC
F339   07                RLC              ;MOVE IT TO HIGH NIBBLE
F33A   4F                MOV C,A          ;SAVE IT
F33B   CD F694           CALL RIBBLE      ;GET OTHER HALF
F33E   B1                ORA C            ;MAKE WHOLE
F33F   4F                MOV C,A          ;SAVE AGAIN IN C
F340   82                ADD D            ;UPDATE CHECKSUM
```

```
F341    57                      MOV D,A         ;NEW CHECKSUM
F342    79                      MOV A,C         ;CONVERTED BYTE
F343    C1                      POP B
F344    C9                      RET

F345    CD F603         SUBS:   CALL EXPR        ;GET STARTING ADDR.
F348    E1                      POP H
F349    D8                      RC
F34A    7E              ..S0:   MOV A,M
F34B    CD F4CF                 CALL LBYTE      ;DISPLAY THE BYTE
F34E    CD F6A7                 CALL COPCK      ;MODIFY?
F351    D8                      RC              ; NO, ALL DONE
F352    2808                    JRZ ..S1        ;DON'T MODIFY
F354    E5                      PUSH H          ;SAVE POINTER
F355    CD F606                 CALL EXF        ;GET NEW VALUE
F358    D1                      POP D           ;VALUE IN E
F359    E1                      POP H
F35A    73                      MOV M,E         ;MODIFY
F35B    D8                      RC              ;DONE
F35C    FE2C            ..S1:   CPI ','         ;BACKUP DELIMITER ?
F35E    2809                    JRZ ..S4
F360    23                      INX H
F361    7D              ..S2:   MOV A,L         ;SEE IF TIME TO CRLF
F362    E607                    ANI 7
F364    CC F544         ..S3:   CZ LFADR        ;TIME TO CRLF
F367    18E1                    JMPR ..S0
F369    2B              ..S4:   DCX H           ;DECREMENT POINTER
F36A    18F8                    JMPR ..S3       ;AND PRINT DATA THERE.

F36C    3E40            TYPE:   MVI A, 64       ;SET DEFAULT
F36E    CD F623                 CALL EXPC       ;GET RANGE
F371    CD F535         ..T0:   CALL LFADRX      ;DISPLAY ADDRESS
F374    7E              ..T1:   MOV A,M
F375    E67F                    ANI 7FH         ;KILL PARITY BIT
F377    FE20                    CPI ' '         ;RANGE TEST
F379    3002                    JRNC ..T3       ;<SPACE
F37B    3E2E            ..T2:   MVI A, '.'      ;REPLACE NON-PRINTING
F37D    FE7D            ..T3:   CPI 07DH        ;ABOVE LOWER CASE Z
F37F    30FA                    JRNC ..T2
F381    4F                      MOV C,A         ;SEND IT
F382    CD F54C                 CALL CO
F385    CD F671                 CALL HILOX      ;MORE TO GO?
F388    10EA                    DJNZ ..T1       ;SEE IF TIME TO CRLF
F38A    18E5                    JMPR ..T0       ; YES.

F38C    CD F642         VERIFY: CALL EXP3        ;GET 3 PARAMETERS
F38F    0A              ..V0:   LDAX B
F390    C5                      PUSH B          ;SAVE POINTER
F391    46                      MOV B,M         ;GET MEMORY
F392    B8                      CMP B           ;MATCH?
F393    C4 F5E5                 CNZ CERR        ;DISPLAY ERRORS
F396    C1              ..V1:   POP B           ;RESTORE ADDR
F397    03                      INX B
F398    CD F671                 CALL HILOX
F39B    18F2                    JMPR ..V0
```

```
F39D    3E18        WRITE:  MVI A, 24       ;SET TO DEFAULT
F39F    CD F623             CALL EXPC       ;GET TWO PARAMETERS
F3A2    CD F686             CALL TPON       ;TURN ON TAPE
F3A5    CD F41C     ..W0:   CALL PEOL       ;CRLF TO PUNCH
F3A8    01 003A             LXI B, ':'      ;START-OF-FILE CUE
F3AB    CD F423             CALL PO         ;PUNCH IT
F3AE    D5                  PUSH D          ;SAVE
F3AF    E5                  PUSH H          ; POINTERS
F3B0    08                  EXAF            ;SET-UP RECORD LENGTH
F3B1    4F                  MOV C,A         ;WITH A'
F3B2    08                  EXAF            ;SAVE FOR LATER
F3B3    04          ..W1:   INR B           ;CALCULATE FILE LENGTH
F3B4    CD F677             CALL HILO
F3B7    380B                JRC ..W2        ;SHORT FILE
F3B9    79                  MOV A,C
F3BA    90                  SUB B           ;ENOUGH YET?
F3BB    20F6                JRNZ ..W1       ; NO.
F3BD    E1                  POP H           ;GET START ADDR BACK.
F3BE    CD F3CB             CALL ..W3       ;SEND THE BLOCK
F3C1    D1                  POP D           ;RESTORE END OF FILE POINTER
F3C2    18E1                JMPR ..W0       ;KEEP GOING
F3C4    E1          ..W2:   POP H           ;CLEAR STACK
F3C5    D1                  POP D           ; OF POINTERS
F3C6    CD F3CB             CALL ..W3       ;PUNCH LAST BLOCK
F3C9    183A                JMPR TOFF       ;TAPE OFF & RETURN
F3CB    78          ..W3:   MOV A,B         ;FILE LENGTH
F3CC    CD F40E             CALL PBYTE      ;PUNCH IT
F3CF    CD F409             CALL PADR       ;PUNCH ADDRESS
F3D2    78                  MOV A,B         ;SET-UP CHECKSUM
F3D3    84                  ADD H
F3D4    85                  ADD L
F3D5    57                  MOV D,A         ;SAVE IT IN D
F3D6    AF                  XRA A           ;FILE TYPE=0
F3D7    CD F40E             CALL PBYTE      ;PUNCH IT
F3DA    4E          ..W4:   MOV C,M         ;GET A DATA BYTE
F3DB    7A                  MOV A,D         ;UPDATE CHECKSUM
F3DC    81                  ADD C
F3DD    57                  MOV D,A         ;NEW CHECKSUM
F3DE    79                  MOV A,C         ;BYTE TO PUNCH
F3DF    CD F40E             CALL PBYTE      ;PUNCH IT
F3E2    23                  INX H           ;POINT TO NEXT BYTE
F3E3    10F5                DJNZ ..W4       ;DECREMENT FILE COUNT
F3E5    AF                  XRA A
F3E6    92                  SUB D           ;CALCULATE CHECKSUM
F3E7    1825                JMPR PBYTE      ;PUNCH IT, RETURN

F3E9    CD F603     EOF:    CALL EXPR       ;GET OPTIONAL ADDR.
F3EC    CD F686             CALL TPON       ;TAPE ON
F3EF    CD F41C             CALL PEOL       ;CRLF TO PUNCH
F3F2    0E3A                MVI C, ':'      ;FILE MARKER CUE
F3F4    CD F423             CALL PO
F3F7    AF                  XRA A           ;ZERO LENGTH
F3F8    CD F40E             CALL PBYTE
F3FB    E1                  POP H
```

```
F3FC    CD F409             CALL PADR       ;PUNCH OPTIONAL ADDR.
F3FF    21 0000             LXI H, 0        ;FILE TYPE & CHECKSUM
F402    CD F409             CALL PADR
F405    0E14        TOFF:   MVI C, T.OFF    ;TURN OFF TAPE
F407    181A                JMPR PO         ;PUNCH IT & RETURN

F409    7C          PADR:   MOV A,H
F40A    CD F40E             CALL PBYTE
F40D    7D                  MOV A,L

F40E    F5          PBYTE:  PUSH PSW        ;NIBBLE AT A TIME
F40F    0F                  RRC
F410    0F                  RRC
F411    0F                  RRC
F412    0F                  RRC
F413    CD F417             CALL ..2
F416    F1                  POP PSW         ;NEXT NIBBLE
F417    CD F648     ..2:    CALL CONV
F41A    1807                JMPR PO

F41C    0E8D        PEOL:   MVI C, CR!80H   ;TO SPOT FILE BREAKS
F41E    CD F423             CALL PO
F421    0E0A                MVI C, LF

F423    CD F1DB     PO:     CALL IOCHK
F426    E630                ANI # PMSK
F428    CA F553             JZ TTYOUT       ;PUNCH=TELEPRINTER
F42B    FE20                CPI PCAS        ;CASSETTE?
F42D    CA F803             JZ POUSR
F430    FE10                CPI PPTP
F432    CA E038             JZ PTPL         ;EXTERNAL VECTOR
F435    C3 E03B             JMP PULOC       ;USER VECTOR

F438    CD F6AC     XAM:    CALL PCHK
F43B    21 F7CE             LXI H, ACTBL
F43E    383C                JRC ..X6        ;FULL REG. DISPLAY
F440    FE27                CPI "'"         ;SEE IF PRIMES WANTED
F442    2008                JRNZ ..X0
F444    21 F7E7             LXI H, PRMTB
F447    CD F6AC             CALL PCHK
F44A    3830                JRC ..X6        ;FULL REG. DISPLAY
F44C    BE          ..X0:   CMP M           ;TEST FOR REGISTER NAME
F44D    2809                JRZ ..X1
F44F    CB7E                BIT 7, M        ;SEE IF END OF TABLE
F451    C2 F5D2             JNZ ERROR
F454    23                  INX H
F455    23                  INX H
F456    18F4                JMPR ..X0
F458    CD F54A     ..X1:   CALL BLK
F45B    CD F49D     ..X2:   CALL ..X8       ;PRINT REGISTER VALUE
F45E    CD F6A7     ..X3:   CALL COPCK      ;MODIFY?
F461    2813                JRZ ..X5        ;SKIP TO NEXT REG.
F463    E5                  PUSH H
F464    C5                  PUSH B
F465    CD F606             CALL EXF        ;GET NEW VALUE
```

```
F468    E1                      POP H
F469    C1                      POP B           ;OLD B
F46A    F5                      PUSH PSW        ;SAVE DELIMITER
F46B    7D                      MOV A,L
F46C    12                      STAX D
F46D    CB78                    BIT 7, B        ;SEE IF 8 BIT OR 16 BIT REG.
F46F    2803                    JRZ ..X4        ;8 BIT
F471    13                      INX D
F472    7C                      MOV A,H         ;HIGH BYTE OF 16 BIT REG.
F473    12                      STAX D
F474    F1              ..X4:   POP PSW         ;GET DELIMITER
F475    E1                      POP H           ;RESTORE TABLE POINTER
F476    D8              ..X5:   RC              ;CR ENTERED, ALL DONE
F477    CB7E                    BIT 7, M        ;SEE IF END OF TABLE
F479    C0                      RNZ             ;RETURN IF SO
F47A    18DF                    JMPR ..X2
F47C    CD F5F4         ..X6:   CALL CRLF
F47F    CD F54A         ..X7:   CALL BLK
F482    4E                      MOV C,M
F483    CD F54C                 CALL CO
F486    0E3D                    MVI C, '='
F488    CD F54C                 CALL CO
F48B    CD F49D                 CALL ..X8       ;PRINT REG. VALUE
F48E    CB7E                    BIT 7, M        ;END OF TABLE?
F490    28ED                    JRZ ..X7        ; NO
F492    CB76                    BIT 6, M        ;SEE IF EI/DI
F494    C0                      RNZ             ;ALL DONE
F495    1B                      DCX D           ;GET "R" REG. VALUE
F496    1A                      LDAX D
F497    17                      RAL             ;TEST BIT 7
F498    D0                      RNC             ;INTERUPTS DISABLED
F499    0E2A                    MVI C, '*'      ;ELSE ENABLED
F49B    183E                    JMPR COX        ;PRINT '*' AND RETURN
F49D    23              ..X8:   INX H           ;POINT TO REG. DISPLACEMENT
F49E    7E                      MOV A,M         ;GET IT
F49F    23                      INX H           ;POINT TO NEXT IN TABLE
F4A0    EB                      XCHG            ;SAVE IN DE
F4A1    47                      MOV B,A         ;SAVE FOR FLAGS
F4A2    E63F                    ANI 3FH         ;CLEAN UP FOR OFFSET
F4A4    6F                      MOV L,A
F4A5    2600                    MVI H, 0
F4A7    39                      DAD SP
F4A8    23                      INX H           ;ADJUST FOR THE
F4A9    23                      INX H           ; RETURN ON STACK
F4AA    CB70                    BIT 6, B        ;TEST FOR "M"
F4AC    2804                    JRZ ..X9        ; NO
F4AE    7E                      MOV A,M         ;GET "M" POINTER
F4AF    2B                      DCX H
F4B0    6E                      MOV L,M
F4B1    67                      MOV H,A         ;GOT IT
F4B2    7E              ..X9:   MOV A,M         ;GET A BYTE
F4B3    CD F4CF                 CALL LBYTE      ;PRINT REG. VALUE
F4B6    EB                      XCHG            ;RESTORE TABLE POINTER
F4B7    CB78                    BIT 7, B        ;SINGLE OR DOUBLE?
F4B9    C8                      RZ              ;SINGLE
```

```
F4BA    1B                  DCX D
F4BB    1A                  LDAX D
F4BC    1811                JMPR LBYTE      ;PRINT IT & RETURN

F4BE    CD F623     HEXN:   CALL EXPC       ;GET TWO PARAMS
F4C1    E5                  PUSH H          ;SAVE HL FOR LATER
F4C2    19                  DAD D           ;GET SUM
F4C3    CD F547             CALL HLSP       ;PRINT IT
F4C6    E1                  POP H           ;THIS IS LATER
F4C7    B7                  ORA A           ;CLEAR CARRY
F4C8    ED52                DSBC D          ;GET DIFFERENCE & PRINT IT

F4CA    7C          LADR:   MOV A,H
F4CB    CD F4CF             CALL LBYTE
F4CE    7D                  MOV A,L
F4CF    F5          LBYTE:  PUSH PSW
F4D0    0F                  RRC
F4D1    0F                  RRC
F4D2    0F                  RRC
F4D3    0F                  RRC
F4D4    CD F4D8             CALL ..2
F4D7    F1                  POP PSW
F4D8    CD F648     ..2:    CALL CONV
F4DB    186F        COX:    JMPR CO

F4DD    CD F603     LOAD:   CALL EXPR       ;INITIAL LOAD ADDRESS
F4E0    CD F5F4             CALL CRLF
F4E3    E1                  POP H
F4E4    16FF                MVI D, 0FFH     ;START-OF-FILE TAG
F4E6    01 0407     ..L0:   LXI B, 407H     ;FIND FOUR 0FFH'S, C=BELL
F4E9    CD F598     ..L1:   CALL RIFF
F4EC    20F8                JRNZ ..L0
F4EE    10F9                DJNZ ..L1
F4F0    CD F598     ..L2:   CALL RIFF       ;4 FOUND, NOW WAIT FOR NON-0FFH
F4F3    28FB                JRZ ..L2
F4F5    77                  MOV M,A         ;FIRST REAL DATA BYTE
F4F6    CD F54C             CALL CO                 ;TELL CONSOLE
F4F9    23          ..L3:   INX H
F4FA    CD F598             CALL RIFF
F4FD    2803                JRZ ..L5        ;POSSIBLE END OF FILE
F4FF    77          ..L4:   MOV M,A
F500    18F7                JMPR ..L3
F502    0601        ..L5:   MVI B, 1        ;INITIALIZE
F504    CD F598     ..L6:   CALL RIFF
F507    2008                JRNZ ..L7
F509    04                  INR B           ;COUNT QUES
F50A    3E07                MVI A, MAX      ;LOOK FOR EOF
F50C    B8                  CMP B           ;FOUND MAX?
F50D    20F5                JRNZ ..L6       ;NOPE
F50F    1833                JMPR LFADR      ;YEP, PRINT END ADDR
F511    72          ..L7:   MOV M,D
F512    23                  INX H
F513    10FC                DJNZ ..L7
F515    18E8                JMPR ..L4
```

```
F517  CD F603     PUTA:   CALL EXPR        ;GET THE STARTING ADDRESS
F51A  CD F5F4             CALL CRLF
F51D  E1                  POP H
F51E  CD F72A     ..P1:   CALL KI          ;GET A CHARACTER
F521  FE04                CPI 04           ;CONTROL-D (EOT)
F523  281F                JRZ LFADR        ;STOP & PRINT ADDRESS
F525  FE08                CPI BS           ;ERASE MISTAKE?
F527  2808                JRZ ..P3         ;  YES.
F529  77                  MOV M,A          ;ELSE STORE IT
F52A  4F                  MOV C,A
F52B  23                  INX H            ;POINT TO NEXT MEMORY LOCATION
F52C  CD F54C     ..P2:   CALL    CO       ;ECHO IT ON CONSOLE
F52F  18ED                JMPR    ..P1     ;AND CONTINUE
F531  2B          ..P3:   DCX H            ;BACK UP POINTER
F532  4E                  MOV C,M          ;PICK UP OLD CHAR.
F533  18F7                JMPR ..P2        ;AND DISPLAY IT

F535  08          LFADRX: EXAF
F536  47                  MOV B,A          ;RESET B=A'
F537  08                  EXAF
F538  180A                JMPR LFADR       ;PRINT ADDR & CONT.

F53A  CD F0A3     SIZE:   CALL MEMSIZ      ;GET THE VALUE
F53D  CD F5FE             CALL INTCK       ;SEE IF INTERUPTS ALLOWED
F540  21 0023             LXI H, (ENDX-EXIT)
F543  19                  DAD D            ;ADJUST IT

F544  CD F5F4     LFADR:  CALL CRLF
F547  CD F4CA     HLSP:   CALL LADR
F54A  0E20        BLK:    MVI C, ' '

F54C  CD F1DB     CO:     CALL IOCHK
F54F  E603                ANI # CMSK
F551  2029                JRNZ CO0
F553  CD F730     TTYOUT: CALL PORA        ;PIA OR ACIA ?
F556  200A                JRNZ ..TT1       ;  PIA.
F558  DB06        ..TT0:  IN TTS
F55A  E602                ANI TTYBE
F55C  28FA                JRZ ..TT0
F55E  79                  MOV A,C
F55F  D307                OUT TTO
F561  C9                  RET
F562  C5          ..TT1:  PUSH B           ;PIA OUTPUT RTN.
F563  060B                MVI B, 11        ;# OF BITS TO SEND
F565  AF                  XRA A            ;START BIT IN CARRY
F566  17          ..TT2:  RAL              ;CARRY IS BIT TO SEND
F567  D304                OUT TTOP         ;PIA DATA PORT
F569  3E05                MVI A, 5         ;RESET TIMER
F56B  D306                OUT TTS
F56D  3D                  DCR A            ;ENABLE TIMER
F56E  D306                OUT TTS
F570  DB06        ..TT3:  IN TTS           ;GET TIMER STATUS
F572  17                  RAL              ;TIMED OUT ?
F573  30FB                JRNC ..TT3
F575  CB19                RARR C           ;SHIFT NEXT BIT INTO CARRY
```

```
F577    10ED              DJNZ ..TT2      ;DO ALL THE BITS
F579    C1                POP B
F57A    79                MOV A,C         ;RETURN WITH BYTE IN A
F57B    C9                RET
F57C    3D          CO0:  DCR A           ;CCRT ?
F57D    CA E029           JZ CRTOUT       ; YES.
F580    3D                DCR A           ;BATCH ?
F581    C2 E023           JNZ COLOC       ; NO, MUST BE USER

F584    CD F1DB     LO:   CALL IOCHK
F587    E6C0              ANI # LMSK
F589    28C8              JRZ TTYOUT
F58B    FE40              CPI LCRT
F58D    CA E029           JZ CRTOUT
F590    FE80              CPI LINE
F592    CA E02F           JZ LPNTR        ;EXT LINE PRINTER RTN.
F595    C3 E03E           JMP LULOC       ;USER VECTOR

F598    CD F70C     RIFF: CALL RI         ;GET READER CHARACTER
F59B    3835              JRC ERROR       ;ABORT ON CARRY
F59D    BA                CMP D           ;TEST D
F59E    C9                RET

F59F    CD F603     QUERY: CALL EXPR
F5A2    C1                POP B
F5A3    ED58              INP E
F5A5    0608        BITS: MVI B, 8        ;DISPLAY 8 BITS
F5A7    CD F54A           CALL BLK
F5AA    CB23        ..Q2: SLAR E
F5AC    3E18              MVI A, '0' >1
F5AE    8F                ADC A           ;MAKE "0" OR "1"
F5AF    4F                MOV C,A
F5B0    CD F54C           CALL CO
F5B3    10F5              DJNZ ..Q2
F5B5    183D              JMPR CRLF       ;CRLF & RETURN

F5B7    CD F623     POUT: CALL EXPC       ;OUTPUT TO PORT RTN.
F5BA    4D                MOV C,L         ;L=PORT
F5BB    ED59              OUTP E          ;E=DATA
F5BD    C9                RET

F5BE    21 F0D5     TOM:  LXI H, MSG
F5C1    4E          TOM1: MOV C,M         ;GET A CHARACTER
F5C2    23                INX H
F5C3    CD F54C           CALL CO         ;OUTPUT IT
F5C6    10F9              DJNZ TOM1
F5C8    CD F652           CALL CSTS       ;SEE IF AN ABORT REQUEST
F5CB    3C                INR A           ; WAITING
F5CC    CC F72A           CZ KI
F5CF    FE03              CPI 3           ;CONTROL-C?
F5D1    C0                RNZ

F5D2    CD F0A3     ERROR: CALL MEMSIZ
F5D5    21 FFEA           LXI H, -22      ;STACK POINTER OFFSET
F5D8    19                DAD D
```

```
F5D9    F9                      SPHL            ;RESET STACK
F5DA    CD F5FE                 CALL INTCK      ;SEE IF INTERUPTS ALLOWED
F5DD    0E2A                    MVI C, '*'      ;ANNOUNCE ERROR
F5DF    CD F54C                 CALL CO
F5E2    C3 F12A                 JMP START       ;BACK TO WORK

F5E5    08              CERR:   EXAF            ;SAVE ACC.
F5E6    CD F547                 CALL HLSP       ;DISPLAY H&L
F5E9    78                      MOV A,B
F5EA    CD F4CF                 CALL LBYTE      ;PRINT 'M'
F5ED    CD F54A                 CALL BLK        ;SPACE OVER
F5F0    08                      EXAF
F5F1    CD F4CF                 CALL LBYTE      ;PRINT ACC.

F5F4    E5              CRLF:   PUSH H          ;SAVE HL
F5F5    C5                      PUSH B
F5F6    0604                    MVI B, 4        ;CRLF LENGTH
F5F8    CD F5BE                 CALL TOM        ;SEND CRLF
F5FB    C1                      POP B
F5FC    E1                      POP H
F5FD    C9                      RET

F5FE    FEAA            INTCK:  CPI 0AAH        ;SEE IF INTERUPTS DESIRED
F600    C0                      RNZ             ;NO
F601    FB                      EI              ;ELSE ENABLE THEM
F602    C9                      RET

F603    CD F785         EXPR:   CALL TI         ;GET SOMETHING FROM CONSOLE
F606    21 0000         EXF:    LXI H, 0        ;INITIALIZE HL TO ZERO
F609    47              ..EX1:  MOV B,A         ;SAVE IT
F60A    CD F697                 CALL NIBBLE     ;CONVERT ASCII TO HEX.
F60D    380B                    JRC ..EX2       ;ILLEGAL CHARACTER DETECTED
F60F    29                      DAD H           ;MULTIPLY BY 16
F610    29                      DAD H
F611    29                      DAD H
F612    29                      DAD H
F613    B5                      ORA L           ;OR IN THE SINGLE NIBBLE
F614    6F                      MOV L,A
F615    CD F785                 CALL TI
F618    18EF                    JMPR ..EX1      ;GET SOME MORE
F61A    E3              ..EX2:  XTHL            ;SAVE UP IN STACK
F61B    E5                      PUSH H          ;REPLACE THE RETURN
F61C    78                      MOV A,B         ;TEST THE DELIMITER
F61D    CD F6AF                 CALL QCHK
F620    20B0                    JRNZ ERROR      ;SOMETHING WRONG
F622    C9                      RET             ;ELSE RETURN

F623    08              EXPC:   EXAF            ;SAVE ANY DEFAULT A'
F624    CD F603                 CALL EXPR       ;GET 1ST. PARAMETER
F627    38A9                    JRC ERROR       ;CR TOO SOON
F629    CD F603                 CALL EXPR
F62C    D1                      POP D           ;GET 2ND.
F62D    E1                      POP H           ;GET 1ST.
F62E    380C                    JRC ..P         ;CARRY SET=2 PARAMETERS
F630    E5                      PUSH H          ;SAVE 1ST.
```

```
F631    CD F603             CALL EXPR       ;GET ONE MORE
F634    C1                  POP B           ;GET 3RD.
F635    E1                  POP H           ;GET 1ST.
F636    79                  MOV A,C         ;TEST 3RD. LSB
F637    B7                  ORA A           ;ZERO?
F638    2802                JRZ ..P         ;USE DEFAULT (A')
F63A    08                  EXAF            ;ELSE SWITCH ACC.
F63B    B7                  ORA A           ;CARRY CLEAR=3 PARAMETERS
F63C    F5          ..P:    PUSH PSW        ;SAVE FLAGS
F63D    CD F5F4             CALL CRLF       ;DO CRLF
F640    F1                  POP PSW
F641    C9                  RET

F642    CD F623     EXP3:   CALL EXPC       ;GET 3
F645    388B                JRC ERROR       ;I SAID 3
F647    C9                  RET

                    ; CONVERT HEX TO ASCII

F648    E60F        CONV:   ANI 0FH         ;LOW NIBBLE ONLY
F64A    C690                ADI 90H
F64C    27                  DAA
F64D    CE40                ACI 40H
F64F    27                  DAA
F650    4F                  MOV C,A
F651    C9                  RET

F652    CD F1DB     CSTS:   CALL IOCHK
F655    E603                ANI # CMSK
F657    200F                JRNZ ..CS1
F659    CD F730             CALL PORA       ;PIA OR ACIA ?
F65C    2802                JRZ ..CS0       ;  IF ACIA
F65E    AF                  XRA A           ;PIA IS ALWAYS FALSE
F65F    C9                  RET
F660    DB06        ..CS0:  IN TTS
F662    1F                  RAR
F663    3EFF                MVI A, TRUE     ;MODIFY TO SUIT
F665    D8                  RC
F666    2F                  CMA
F667    C9                  RET
F668    3D          ..CS1:  DCR A           ;CCRT
F669    CA E02C             JZ CRTST        ;GET CRT KBD STATUS
F66C    3D                  DCR A           ;BATCH ?
F66D    C8                  RZ              ;BATCH IS ALWAYS FALSE
F66E    C3 E041             JMP CSLOC       ;USER

F671    CD F677     HILOX:  CALL HILO
F674    D0                  RNC             ;OK
F675    D1                  POP D           ;RETURN ONE LEVEL BACK
F676    C9                  RET

F677    23          HILO:   INX H           ;INCREMENT HL
F678    7C                  MOV A,H         ;TEST FOR CROSSING 64K BORDER
F679    B5                  ORA L
F67A    37                  STC             ;CARRY SET=STOP
```

```
F67B    C8                      RZ              ;YES, BORDER CROSSED
F67C    7B                      MOV A,E         ;NON, TEST HL VS. DE
F67D    95                      SUB L
F67E    7A                      MOV A,D
F67F    9C                      SBB H
F680    C9                      RET             ;IF CARRY WAS SET, THEN STOP

F681    01 08FF         MARK:   LXI B, 08FFH    ;SET-UP B&C
F684    1808                    JMPR LE0

F686    0E12            TPON:   MVI C, T.ON     ;TURN ON TAPE
F688    CD F423                 CALL PO
F68B    01 3C00         LEAD:   LXI B, 3C00H    ;PRESET FOR SOME NULLS
F68E    CD F423         LE0:    CALL PO
F691    10FB                    DJNZ LE0
F693    C9                      RET

F694    CD F725         RIBBLE: CALL RIX
F697    D630            NIBBLE: SUI '0'         ;QUALIFY & CONVERT
F699    D8                      RC              ;<0
F69A    FE17                    CPI 'G'-'0'     ;>F?
F69C    3F                      CMC             ;PERVERT CARRY
F69D    D8                      RC
F69E    FE0A                    CPI 10          ;NMBR?
F6A0    3F                      CMC             ;PERVERT AGAIN
F6A1    D0                      RNC             ;RETURN CLEAN
F6A2    D607                    SUI 'A'-'9'-1   ;ADJUST
F6A4    FE0A                    CPI 0AH         ;FILTER ":" THRU "@"
F6A6    C9                      RET

F6A7    0E2D            COPCK:  MVI C, '-'
F6A9    CD F54C                 CALL CO

F6AC    CD F785         PCHK:   CALL TI

F6AF    FE20            QCHK:   CPI ' '         ;RETURN ZERO IF DELIMITER
F6B1    C8                      RZ
F6B2    FE2C                    CPI ','
F6B4    C8                      RZ
F6B5    FE0D                    CPI CR          ;RETURN W/CARRY SET IF CR
F6B7    37                      STC
F6B8    C8                      RZ
F6B9    3F                      CMC             ;ELSE NON-ZERO, NO CARRY
F6BA    C9                      RET

F6BB    CD F623         UNLD:   CALL EXPC       ;GET TWO PARAMETERS
F6BE    CD F68B                 CALL LEAD       ;PUNCH LEADER
F6C1    CD F681                 CALL MARK       ;PUNCH FILE MARKER
F6C4    4E              . . U:  MOV C,M         ;GET MEMORY BYTE
F6C5    CD F423                 CALL PO         ;PUNCH IT
F6C8    CD F677                 CALL HILO       ;SEE IF DONE
F6CB    30F7                    JRNC . . U
F6CD    CD F681                 CALL MARK       ;PUNCH END FILE MARKER

F6D0    CD F68B         NULL:   CALL LEAD       ;PUNCH NULLS
```

```
F6D3    CD F1DB     CI:     CALL IOCHK
F6D6    E603                ANI # CMSK
F6D8    202A                JRNZ CI1
F6DA    CD F730     TTYIN:  CALL PORA       ;PIA OR ACIA ?
F6DD    2008                JRNZ ..TT1      ; PIA.
F6DF    DB06        ..TT0:  IN TTS
F6E1    1F                  RAR
F6E2    30FB                JRNC ..TT0
F6E4    DB05                IN TTI
F6E6    C9                  RET
F6E7    C5          ..TT1:  PUSH B          ;PIA INPUT ROUTINE
F6E8    DB04        ..TT2:  IN TTOP         ;GET START BIT
F6EA    17                  RAL
F6EB    38FB                JRC ..TT2
F6ED    3E01                MVI A, 1        ;RESET TIMER VALUE
F6EF    CD F73B             CALL RDEL+2     ;WAIT HALF BIT TIME
F6F2    0608                MVI B, 8        ;# OF BITS TO GET
F6F4    CD F739     ..TT3:  CALL RDEL       ;WAIT BIT TIME
F6F7    DB04                IN TTOP         ;GET BIT
F6F9    17                  RAL             ;BIT TO CARRY
F6FA    CB19                RARR C          ;SHIFT BITS IN FROM LEFT
F6FC    10F6                DJNZ ..TT3      ;GET THEM ALL
F6FE    CD F739             CALL RDEL       ;WAIT FOR STOP
F701    79                  MOV A,C         ;BYTE IN A
F702    C1                  POP B
F703    C9                  RET
F704    3D          CI1:    DCR A           ;CONSOLE=CRT?
F705    CA E026             JZ CRTIN        ;EXT.
F708    3D          CI2:    DCR A           ;BATCH?
F709    C2 E020             JNZ CILOC       ;NO, MUST BE USER DEFINED

F70C    CD F1DB     RI:     CALL IOCHK
F70F    E60C                ANI # RMSK
F711    2005                JRNZ ..RI0      ;NOT TTY
F713    CD F6DA             CALL TTYIN
F716    B7                  ORA A           ;CLEAR CARRY
F717    C9                  RET
F718    FE08        ..RI0:  CPI RCAS        ;CASSETTE ?
F71A    CA F806             JZ RIUSR
F71D    FE04                CPI RPTR        ;FAST RDR ?
F71F    CA E032             JZ RPTPL        ;EXT RTN.
F722    C3 E035             JMP RULOC       ;USER

F725    CD F598     RIX:    CALL RIFF
F728    1803                JMPR KPTY

F72A    CD F6D3     KI:     CALL CI         ;GET CHARACTER FROM CONSOLE
F72D    E67F        KPTY:   ANI 7FH         ;CLEAR PARITY BIT
F72F    C9                  RET

F730    C5          PORA:   PUSH B          ;ROUTINE TO FIND ACIA OR PIA
F731    DB04                IN TTOP
F733    47                  MOV B,A
F734    DB06                IN TTS
```

```
F736    B8                      CMP B
F737    C1                      POP B
F738    C9                      RET

F739    3E05        RDEL:       MVI A, 5            ;RESET TIMER/FULL BIT INIT.
F73B    D306                    OUT TTS
F73D    3D                      DCR A               ;ENABLE TIMER
F73E    D306                    OUT TTS
F740    DB06        ..RD0:      IN TTS              ;GET STATUS
F742    17                      RAL                 ;TIMED OUT ?
F743    30FB                    JRNC ..RD0
F745    C9                      RET

F746    21 0000     WHERE:      LXI H, 0            ;COUNT SEARCH BYTES
F749    4D                      MOV C,L             ;IN C
F74A    39                      DAD SP              ;GET CURRENT SP
F74B    2B                      DCX H               ;-1
F74C    EB                      XCHG                ;SAVE IN DE
F74D    CD F603     ..Y0:       CALL EXPR           ;GET A MATCH BYTE
F750    E1                      POP H               ;IN L
F751    65                      MOV H,L             ;STICK IN HIGH BYTE
F752    E5                      PUSH H              ;PUT IT IN STACK
F753    33                      INX SP              ;ADJUST STACK
F754    0C                      INR C               ;COUNT UP
F755    30F6                    JRNC ..Y0           ;MORE TO GO
F757    79                      MOV A,C             ;GET BYTE COUNT IN A
F758    F5                      PUSH PSW            ;SAVE COUNT IN SP
F759    D5                      PUSH D              ;MATCH STRING POINTER
F75A    01 0000                 LXI B, 0            ;STARTING ADDRESS
F75D    C5                      PUSH B
F75E    E1          ..Y2:       POP H               ;HL=SEARCH ADDR
F75F    D1                      POP D               ;D=SEARCH BYTE POINTER
F760    1A                      LDAX D              ;GET FIRST MATCH VALUE
F761    EDB1                    CCIR                ;COMPARE, INCR., & REPEAT
F763    E2 F781                 JPO ..DONE          ;ODD PARITY=DONE
F766    F1                      POP PSW             ;RESET COUNT
F767    F5                      PUSH PSW            ;SAVE IT AGAIN
F768    D5                      PUSH D              ;SAVE POINTERS
F769    E5                      PUSH H
F76A    3D          ..Y3:       DCR A
F76B    280A                    JRZ ..Y5            ;FOUND ALL
F76D    08          ..Y4:       EXAF                ;SAVE THE COUNT
F76E    1B                      DCX D               ;LOOK AT NEXT MATCH
F76F    1A                      LDAX D
F770    BE                      CMP M               ;TEST NEXT
F771    20EB                    JRNZ ..Y2           ;NO MATCH
F773    23                      INX H               ;BUMP POINTERS
F774    08                      EXAF                ;GET COUNT AGAIN
F775    18F3                    JMPR ..Y3           ;TEST NEXT MATCH
F777    E1          ..Y5:       POP H
F778    E5                      PUSH H
F779    2B                      DCX H
F77A    C5                      PUSH B              ;SAVE SEARCH COUNT LIMIT
F77B    CD F544                 CALL LFADR          ;TELL CONSOLE
F77E    C1                      POP B               ;RESTORE
```

```
F77F    18DD                    JMPR ..Y2       ;DO IT AGAIN
F781    EB              ..DONE: XCHG            ;GET STACK BACK
F782    23                      INX H
F783    F9                      SPHL            ;STACK RESTORED
F784    C9                      RET

F785    CD F72A         TI:     CALL KI
F788    C8                      RZ
F789    FE7F                    CPI 7FH         ;IGNORE RUB-OUTS
F78B    C8                      RZ
F78C    FE0D                    CPI CR          ;IGNORE CR'S
F78E    C8                      RZ
F78F    C5                      PUSH B
F790    4F                      MOV C,A
F791    CD F54C                 CALL CO
F794    B7                      ORA A
F795    C1                      POP B
F796    C9                      RET

                        ; <SYSTEM I/O LOOK-UP TABLE>

                        ; THE FIRST CHARACTER IS THE DEVICE NAME
                        ; (ONE LETTER) AND THE NEXT FOUR ARE THE
                        ; NAMES OF THE FOUR POSSIBLE DRIVERS TO BE
                        ; ASSIGNED.

F797                    LTBL:
F797    43              .BYTE   'C'     ;CONSOLE ASSIGNMENTS
F798    54              .BYTE   'T'     ;CTTY   T=TELEPRINTER
F799    56              .BYTE   'V'     ;CCRT   V=CRT (VIDEO MONITOR)
F79A    42              .BYTE   'B'     ;BATCH= COMMANDS FROM READER
F79B    55              .BYTE   'U'     ;CUSE   USER

F79C    52              .BYTE   'R'     ;READER ASSIGNMENTS
F79D    54              .BYTE   'T'     ;RTTY
F79E    50              .BYTE   'P'     ;RPTR   P=PAPER TAPE
F79F    43              .BYTE   'C'     ;RCAS   C=CASSETTE
F7A0    55              .BYTE   'U'     ;RUSER  USER

F7A1    50              .BYTE   'P'     ;PUNCH ASSIGNMENTS
F7A2    54              .BYTE   'T'     ;PTTY
F7A3    50              .BYTE   'P'     ;PPTP
F7A4    43              .BYTE   'C'     ;PCAS   C=CASSETTE
F7A5    55              .BYTE   'U'     ;PUSER  USER

F7A6    4C              .BYTE   'L'     ;LIST ASSIGNMENTS
F7A7    54              .BYTE   'T'     ;LTTY   LIST=TELEPRINTER
F7A8    56              .BYTE   'V'     ;LCRT   LIST=CRT
F7A9    4C              .BYTE   'L'     ;LINE PRINTER
F7AA    55              .BYTE   'U'     ;LUSER  USER


                        ; THIS IS A SHORT PROGRAM, EXECUTED
                        ; UPON EXECUTING A "GO" COMMAND.  IT
                        ; IS PLACED IN THE WORK AREA WHEN
```

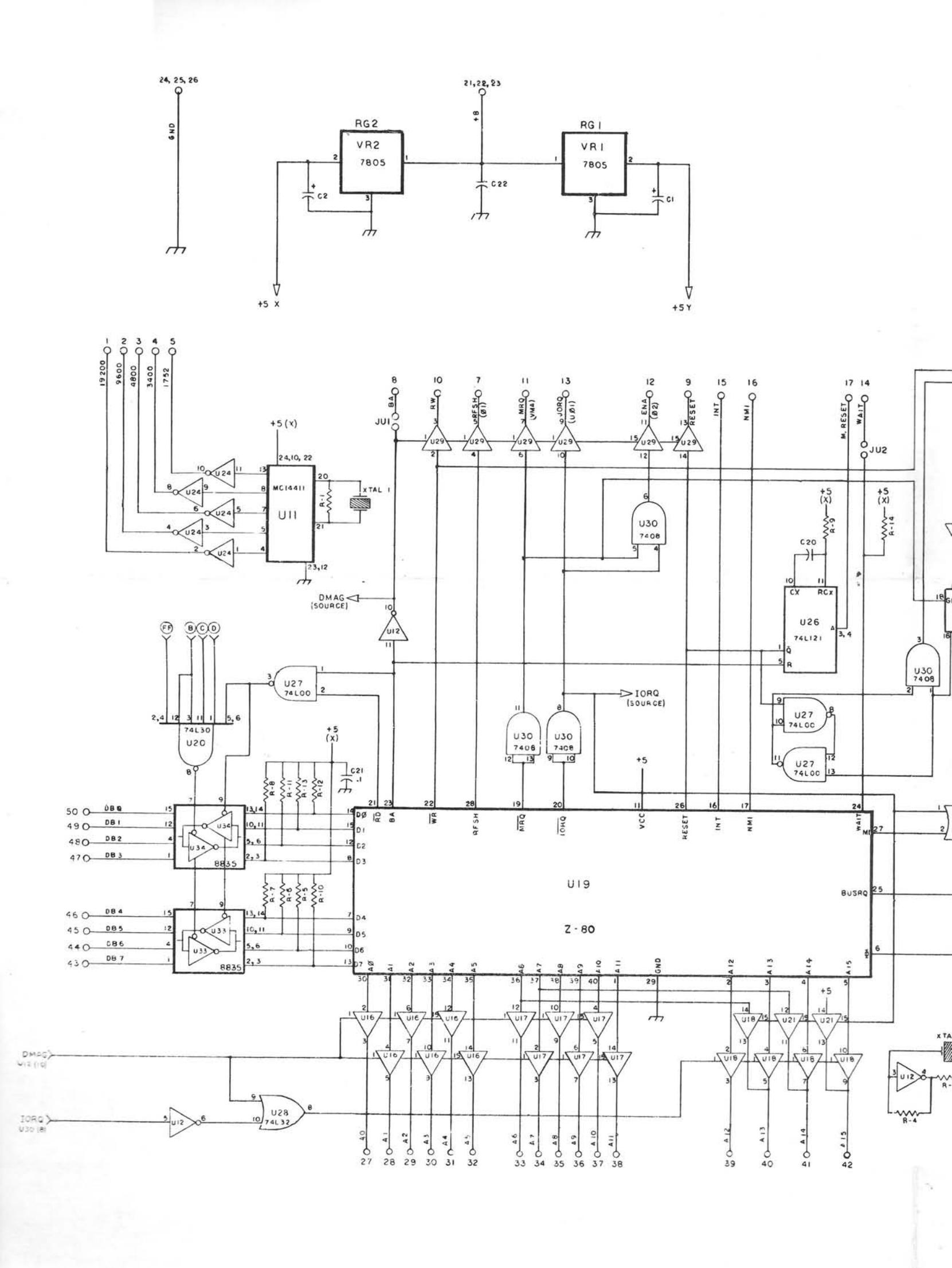
```
                        ; THE MONITOR IS INITIALIZED, AS IT
                        ; REQUIRES RAM FOR PROPER OPERATION.

F7AB                    EXIT:           ;EXIT ROUTINE (LOADS ALL REGISTERS)
F7AB    C1                      POP B
F7AC    79                      MOV A,C
F7AD    ED4F                    STAR
F7AF    78                      MOV A,B
F7B0    ED47                    STAI
F7B2    FDE1                    POP Y
F7B4    DDE1                    POP X
F7B6    F1                      POP PSW
F7B7    C1                      POP B
F7B8    D1                      POP D
F7B9    E1                      POP H
F7BA    08                      EXAF
F7BB    D9                      EXX
F7BC    D1                      POP D
F7BD    C1                      POP B
F7BE    F1                      POP PSW
F7BF    E1                      POP H
F7C0    F9                      SPHL
F7C1    21 0000                 LXI H, 0
F7C4    00                      NOP             ;RESERVED FOR ENABLE INTERUPTS
F7C5    C3 0000                 JMP 0

F7C8    0000                    .WORD   0       ;STORAGE AREA FOR TRAP DATA
F7CA    00                      .BYTE   0
F7CB    0000                    .WORD   0
F7CD    00                      .BYTE   0

                        ; DISPLACEMENTS OF REGISTER
                        ; STORAGE FROM NORMAL STACK
                        ; LOCATION.

F7CE                    ENDX:

0015                            ALOC    = 15H
0013                            BLOC    = 13H
0012                            CLOC    = 12H
0011                            DLOC    = 11H
0010                            ELOC    = 10H
0014                            FLOC    = 14H
0030                            HLOC    = 30H
002F                            LLOC    = 2FH
002F                            INTLOC  = 2FH
0034                            PLOC    = 34H
0017                            SLOC    = 17H
0035                            TLOC    = 35H
0025                            TLOCX   = 25H
001F                            LLOCX   = 1FH

0009                            APLOC   = 09H
000B                            BPLOC   = 0BH
000A                            CPLOC   = 0AH
```

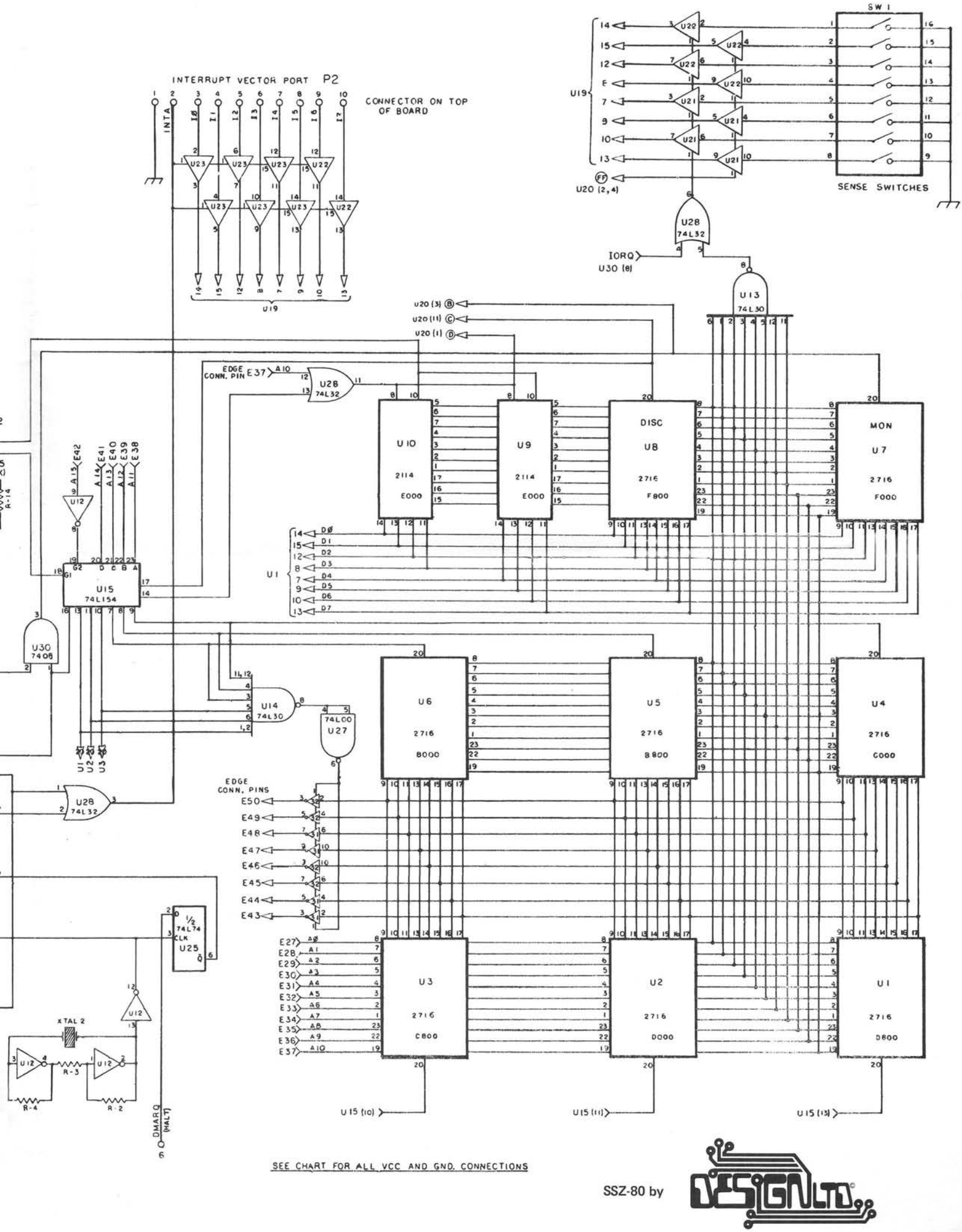
```
000D                            DPLOC   = 0DH
000C                            EPLOC   = 0CH
0008                            FPLOC   = 08H
000F                            HPLOC   = 0FH
000E                            LPLOC   = 0EH
0007                            XLOC    = 07
0005                            YLOC    = 05
0002                            RLOC    = 02
0003                            ILOC    = 03


                        ; THIS IS THE TABLE USED TO DETERMINE
                        ; A VALID REGISTER IDENTIFIER, AND IT'S
                        ; DISPLACEMENT FROM THE STACK POINTER.

                        ; POSITION ONE= REGISTER NAME, WITH BIT 7 INDICATING
                        ; END OF TABLE.

                        ; POSITION TWO= BIAS FROM CURRENT STACK LEVEL OR'ED
                        ; WITH A TWO-BIT FLAG:
                        ;                       00XXXXXX=NORMAL REG. BYTE
                        ;                       01XXXXXX=SPECIAL FOR "M" REG.
                        ;                       10XXXXXX=WORD

F7CE                    ACTBL:          ;NORMAL SET OF REGISTERS (8080)
                        ;               ;PLUS THE INTERUPT REGISTER ("I")

F7CE    4115                    .BYTE   'A',    ALOC    !0
F7D0    4213                    .BYTE   'B',    BLOC    !0
F7D2    4312                    .BYTE   'C',    CLOC    !0
F7D4    4411                    .BYTE   'D',    DLOC    !0
F7D6    4510                    .BYTE   'E',    ELOC    !0
F7D8    4614                    .BYTE   'F',    FLOC    !0
F7DA    4830                    .BYTE   'H',    HLOC    !0
F7DC    4C2F                    .BYTE   'L',    LLOC    !0
F7DE    4D70                    .BYTE   'M',    HLOC    !040H
F7E0    50B4                    .BYTE   'P',    PLOC    !080H
F7E2    5397                    .BYTE   'S',    SLOC    !080H
F7E4    4903                    .BYTE   'I',    ILOC    !0

F7E6    80                      .BYTE   80H

F7E7                    PRMTB:          ;ADDITIONAL SET OF REGISTERS (Z-80)

F7E7    4109                    .BYTE   'A',    APLOC   !0
F7E9    420B                    .BYTE   'B',    BPLOC   !0
F7EB    430A                    .BYTE   'C',    CPLOC   !0
F7ED    440D                    .BYTE   'D',    DPLOC   !0
F7EF    450C                    .BYTE   'E',    EPLOC   !0
F7F1    4608                    .BYTE   'F',    FPLOC   !0
F7F3    480F                    .BYTE   'H',    HPLOC   !0
F7F5    4C0E                    .BYTE   'L',    LPLOC   !0
F7F7    4D4F                    .BYTE   'M',    HPLOC   !040H
F7F9    5887                    .BYTE   'X',    XLOC    !080H
```

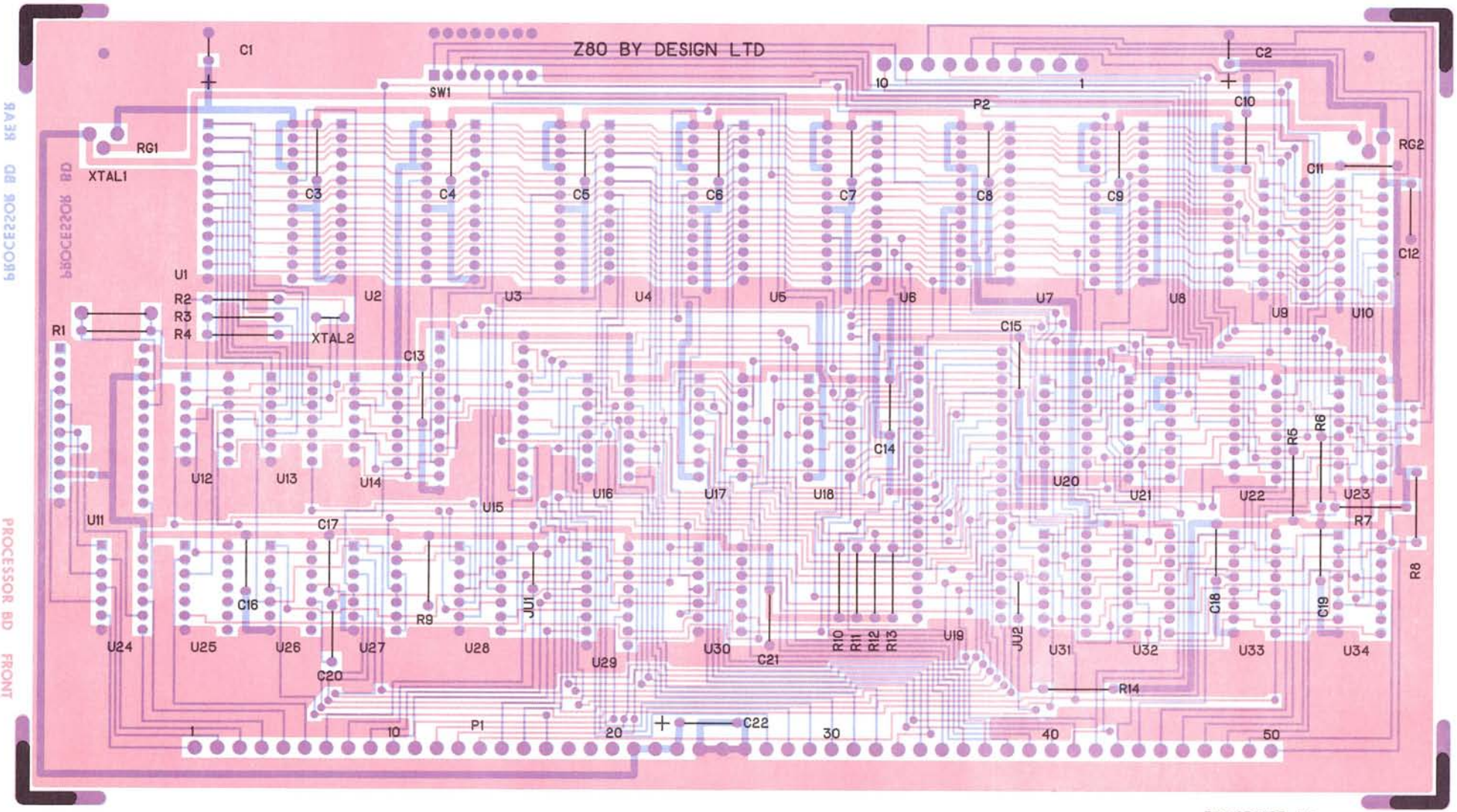
```
F7FB    5985                .BYTE   'Y',    YLOC    !080H
F7FD    5202                .BYTE   'R',    RLOC    !0
F7FF    C0                  .BYTE   0C0H

F800                Z:                              ;END OF PROGRAM


F000                .END    BASE
```

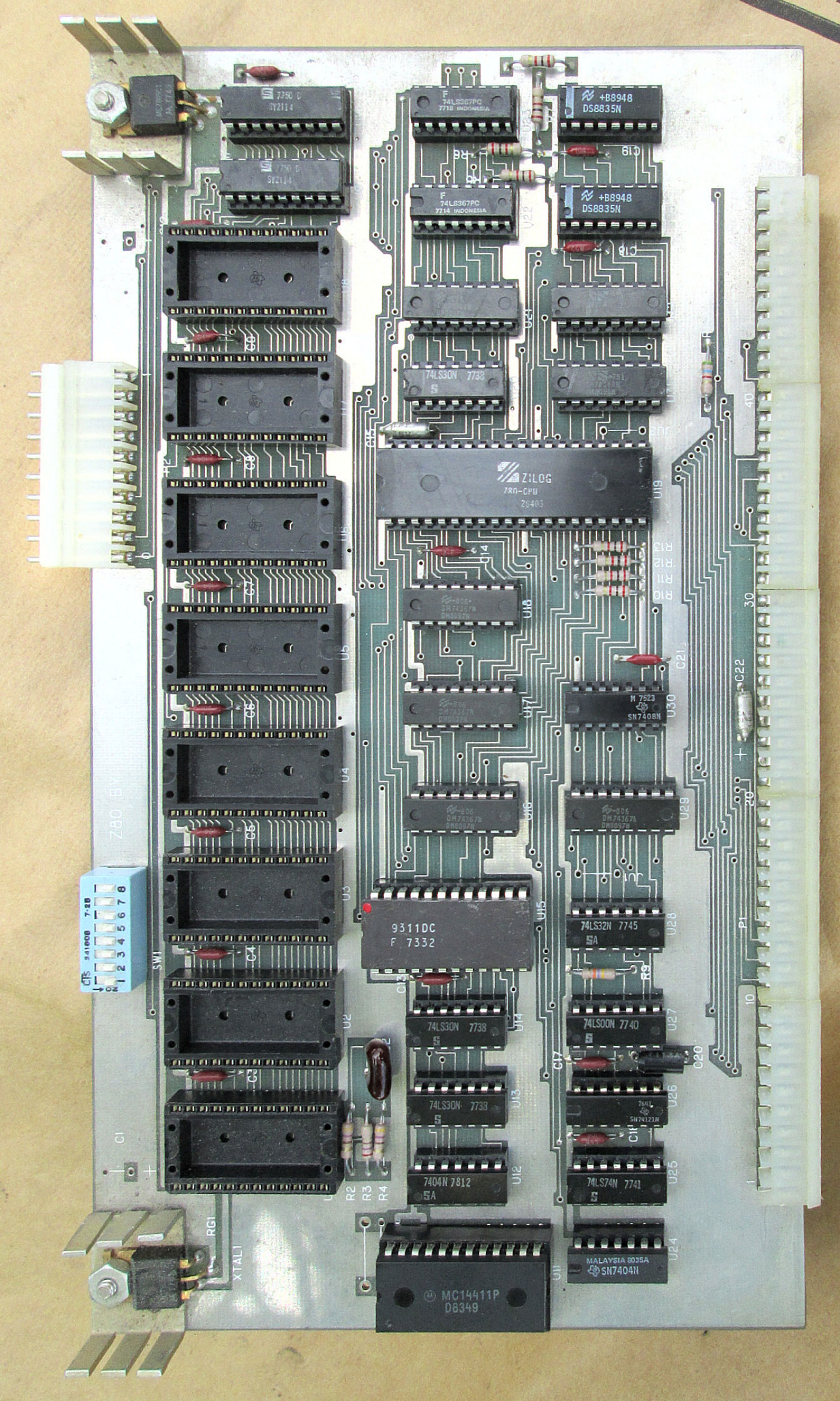
+++++ SYMBOL TABLE +++++

ACTBL	F7CE	ALOC	0015	APLOC	0009	ASET	F1CE
ASSIGN	F199	BASE	F000	BATCH	0002	BEGIN	F0E4
BITS	F5A5	BLK	F54A	BLOC	0013	BPLOC	0008
BRANCH	F18D	BS	0008	CASS	F800	CCRT	0001
CERR	F5E5	CI	F6D3	CI1	F704	CI2	F708
CILOC	E020	CLOC	0012	CMSK	00FC	CO	F54C
CO0	F57C	COLOC	E023	CONV	F648	COPCK	F6A7
COX	F4DB	CPLOC	000A	CR	000D	CRLF	F5F4
CRTIN	E026	CRTOUT	E029	CRTST	E02C	CSLOC	E041
CSTS	F652	CTTY	0000	CUSE	0003	DISP	F1DF
DLOC	0011	DPLOC	000D	ELOC	0010	ENDX	F7CE
EOF	F3E9	EPLOC	000C	ERROR	F5D2	ERRX	F1CB
EXF	F606	EXIT	F7AB	EXP3	F642	EXPC	F623
EXPR	F603	FALSE	0000	FIL	0000	FILL	F181
FLOC	0014	FPLOC	0008	GOTO	F1F5	HEXN	F4BE
HILO	F677	HILOX	F671	HLOC	0030	HLSP	F547
HPLOC	000F	ILOC	0003	INTCK	F5FE	INTLOC	002F
IOBYT	E000	IOCHK	F1DB	IOSET	F1D6	J	E04A
KI	F72A	KPTY	F72D	LADR	F4CA	LBYTE	F4CF
LCRT	0040	LE0	F68E	LEAD	F68B	LF	000A
LFADR	F544	LFADRX	F535	LINE	0080	LLOC	002F
LLOCX	001F	LMSK	003F	LO	F584	LOAD	F4DD
LPLOC	000E	LPNTR	E02F	LTBL	F797	LTTY	0000
LULOC	E03E	LUSER	00C0	MARK	F681	MAX	0007
MEMCK	F0C2	MEMSIZ	F0A3	MOVE	F265	MSG	F0D5
MSGL	000F	NIBBLE	F697	NULL	F6D0	PADR	F409
PBYTE	F40E	PCAS	0020	PCHK	F6AC	PEOL	F41C
PLOC	0034	PMSK	00CF	PO	F423	PORA	F730
POUSR	F803	POUT	F5B7	PPTP	0010	PRMTB	F7E7
PTPL	E038	PTTY	0000	PULOC	E03B	PUSER	0030
PUTA	F517	QCHK	F6AF	QUERY	F59F	RCAS	0008
RDEL	F739	READ	F270	RI	F70C	RIBBLE	F694
RIFF	F598	RIUSR	F806	RIX	F725	RLOC	0002
RMSK	00F3	RPTPL	E032	RPTR	0004	RST7	0038
RTTY	0000	RULOC	E035	RUSER	000C	SENSE	00FF
SIZE	F53A	SLOC	0017	STAR0	F139	START	F12A
STSTK	F107	SUBS	F345	TBL	F14D	TEST	F24C
TI	F785	TLOC	0035	TLOCX	0025	TOFF	F405
TOM	F5BE	TOM1	F5C1	TPON	F686	TRAP	F01E
TRUE	FFFF	TTI	0005	TTO	0007	TTOP	0004
TTS	0006	TTYBE	0002	TTYDA	0001	TTYIN	F6DA
TTYOUT	F553	TYPE	F36C	T.OFF	0014	T.ON	0012
UNLD	F6BB	USER	E000	UTAB	E080	VERIFY	F38C
WHERE	F746	WRITE	F39D	XAM	F438	XLOC	0007
X.OFF	0013	X.ON	0011	YLOC	0005	Z	F800


24, 25, 26
GND
21, 22, 23
+8
RG2
VR2
7805
RG1
VR1
7805
C2
C22
C1
+5 X
+5Y
19200
9600
4800
3400
1752
+5 (X)
24, 10, 22
MC14411
U11
XTAL 1
R-1
23, 12
JU1
BA
RW
RFSH (Ø1)
MRQ (VM4)
IORQ (UØ1)
ENA (Ø2)
RESET
INT
NMI
M. RESET
WAIT
JU2
U29
U24
U30 7408
U26 74L121
C20
R-9
R-14
DMAG (SOURCE)
U12
U27 74L00
U20 74L30
IORQ (SOURCE)
C21 .1
R-8
R-11
R-13
R-12
R-7
R-6
R-5
R-10
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
U34
U33
8835
U19
Z-80
BUSRQ
U16
U17
U18
U21
U28 74L32
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
GND
R-4


INTERRUPT VECTOR PORT P2
CONNECTOR ON TOP OF BOARD
INTA
I0
I1
I2
I3
I4
I5
I6
I7
U19
SW 1
SENSE SWITCHES
U20 (2,4)
U28 74L32
IORQ
U30 (8)
U13 74L30
U20 (3) B
U20 (11) C
U20 (1) D
EDGE CONN. PIN E37
A10
U10 2114 E000
U9 2114 E000
DISC U8 2716 F800
MON U7 2716 F000
U12
U15 74L154
U30 7408
U14 74L30
74LS00 U27
U6 2716 8000
U5 2716 8800
U4 2716 C000
EDGE CONN. PINS
E50
E49
E48
E47
E46
E45
E44
E43
U3 2716 C800
U2 2716 D000
U1 2716 D800
U15 (10)
U15 (11)
U15 (13)
1/2 74L74 CLK U25
XTAL 2
R-3
R-4
R-2
DMAQ (HALT)
SEE CHART FOR ALL VCC AND GND. CONNECTIONS
SSZ-80 by
DESIGN LTD.
DESIGN Ltd.• RD 2 FREWSBURG, NEW YORK 14738


Z80 BY DESIGN LTD
C1
C2
SW1
P2
C10
RG1
RG2
XTAL1
C3
C4
C5
C6
C7
C8
C9
C11
C12
U1
R2
R3
R4
U2
U3
U4
U5
U6
U7
U8
U9
U10
R1
XTAL2
C13
C15
C14
R5
R6
U12
U13
U14
U15
U16
U17
U18
U20
U21
U22
U23
R7
U11
C17
C16
R9
JU1
C18
C19
R8
U24
U25
U26
U27
U28
U29
U30
C21
R10
R11
R12
R13
U19
JU2
U31
U32
U33
U34
C20
R14
1
10
P1
20
C22
30
40
50
PROCESSOR BD
PROCESSOR BD FRONT


ZILOG
Z80-CPU
Z6403
MC14411P
D8349
9311DC
F 7332
74LS30N 7738
74LS32N 7745
74LS00N 7740
74LS74N 7741
7404N 7812
SN7408N
SN7404N
MALAYSIA 8035A
+B8948
DS8835N
74LS367PC
7714 INDONESIA
Z80 BY
SW1
XTAL1
RG1