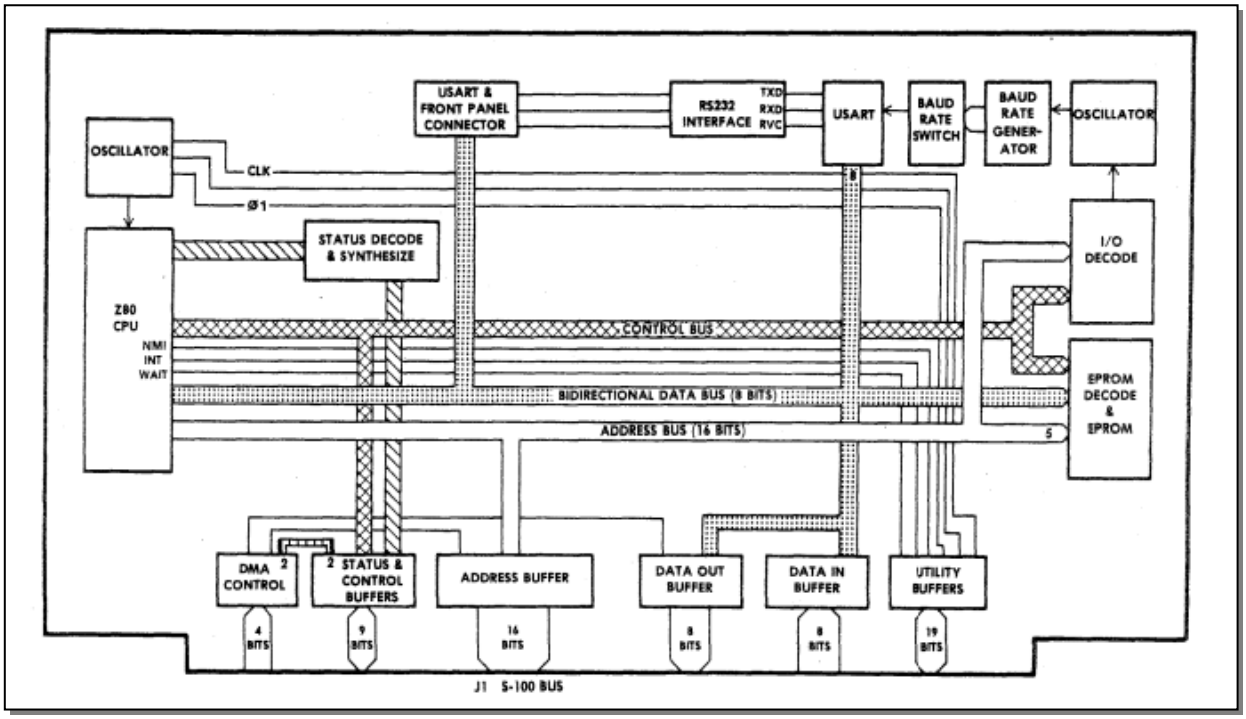
Jade Computer Products

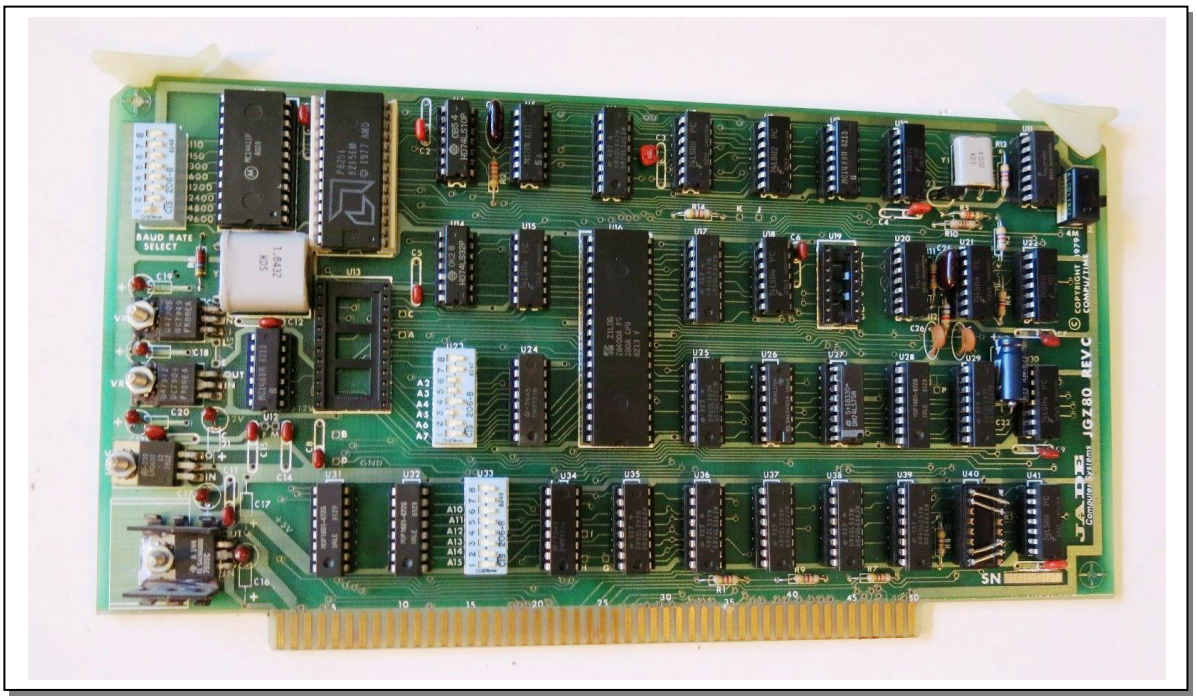
The Big Z Revision C

The Jade Big Z CPU board is a Z80 Main processor card for the S-100 computer bus system developed around 1980. The board was developed before the IEEE-696 standards were put into effect so there are a few complications when using this card with more modern S-100 systems.

The card was known as “The Big Z” or the “JGZ80” board and had revisions A through C with the C version being the final version produced. The board offers the following features…

- Zilog Z-80 CPU
- EPROM onboard accessed on 1K, 2K or 4K boundaries (2708, 27C16 or 27C32)
- POJ power on jump to EPROM at boot
- one M1 wait state
- 8251 USART onboard for RS-232 communication to another Host or Terminal/Console
- CPU speed selectable between 2Mhz and 4Mhz
- Front Panel DIP connector to enable using a front panel (IMSAI)
- Fully buffered S-100 address and data lines
- Voltage regulators for all onboard voltages





The Jade Big-Z revision “C”

The Jade board was engineered before the IEEE specifications were finalized and therefore some incompatibilities exist.

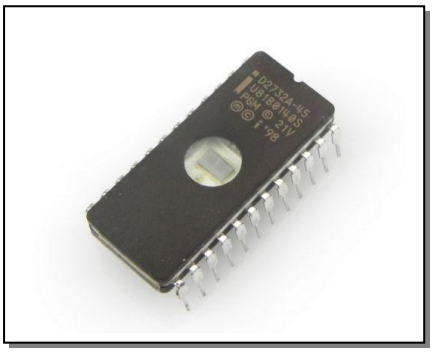
- sXTRQ* – Pin#58 is not implemented on the Big Z (16 bit wide I/O request)
- pSTVAL*- Pin#25 is not implemented (signals when address and status lines are valid)
- SIXTN* - Pin#60 is not implemented (this is an acknowledgement to the sXTRQ* signal)
- RFSH* - Pin#66 is implemented by the Big Z on undefined S-100 pin#66 (Memory Refresh)
- MRQ – Pin#65 is implemented by the Big Z on undefined S-100 pin#65 (Memory Request)
- pWAIT- Pin#27 is implemented by the Big Z on undefined S-100 pin#27 (Wait signal)
- DMA0* – Pin#55 is not implemented (temporary bus master signal)
- DMA1* – Pin#56 is not implemented (temporary bus master signal)
- DMA2* – Pin#57 is not implemented (temporary bus master signal)
- DMA3* – Pin#14 is not implemented (temporary bus master signal)
- GND – Pin#20 is not implemented
- GND – Pin#53 is not implemented
- GND – Pin#70 is not implemented
- CLOCK1 or PHI 1 – Pin#25 is implemented and in **violation of IEEE standards**
  The Pin#9 on U29 has been bent out to disconnect this line from the S-100 Bus
  S-100 Pin#25 should be pSTVAL* and is not implemented, could cause problems.

NOTE: If the "S100Computers.com" System Monitor Board V2 or later is used, U29 should remain intact. The SMB requires pSTVAL* to operate but an inverted PHI 1 on S-100 Bus Pin#25 will enable the SMB Address Display to function (pSTVAL* hack for slower boards).

- SSWDSB*- Pin#53 is implemented and in **violation of IEEE standards** (Sense switch disable) The S-100 bus Pin#53 is a GND line. On the Jade Big Z, the S-100 bus pin#53 has been cut at pull-up resistor pack U31 pin#8.

It has been noted that the Jade Big Z does not work with all memory cards. Most reliable operation of the Jade Big Z will be accomplished using Static RAM memory cards with 8 Data lines and 16 Address lines. Dynamic RAM boards work, but may require rebooting a few times before stable (ie…Jade Memory Bank). So, memory boards that have been used and verified working with the Jade Big Z are the "S100Computers.com" 4MB memory card, Jade Memory Bank, Static Memory Systems "The Last Memory Board" and the Compupro RAM-20. There should be many more boards that work but these are the ones available for testing. The Jade can access 64K of RAM but has no provision for memory paging or extended addressing (A16-A23).

<u>EPROM INTERFACE:</u>



- Use only single voltage EPROM or EEPROM
- Voltage +5V DC
- Cut trace L to E on Big Z to isolate (-5V)
  Note: **Remove C12** as it interferes with A11
- Cut trace F to M on Big Z to isolate (+12V)
- Cut trace G to H on Big Z to isolate (A11)

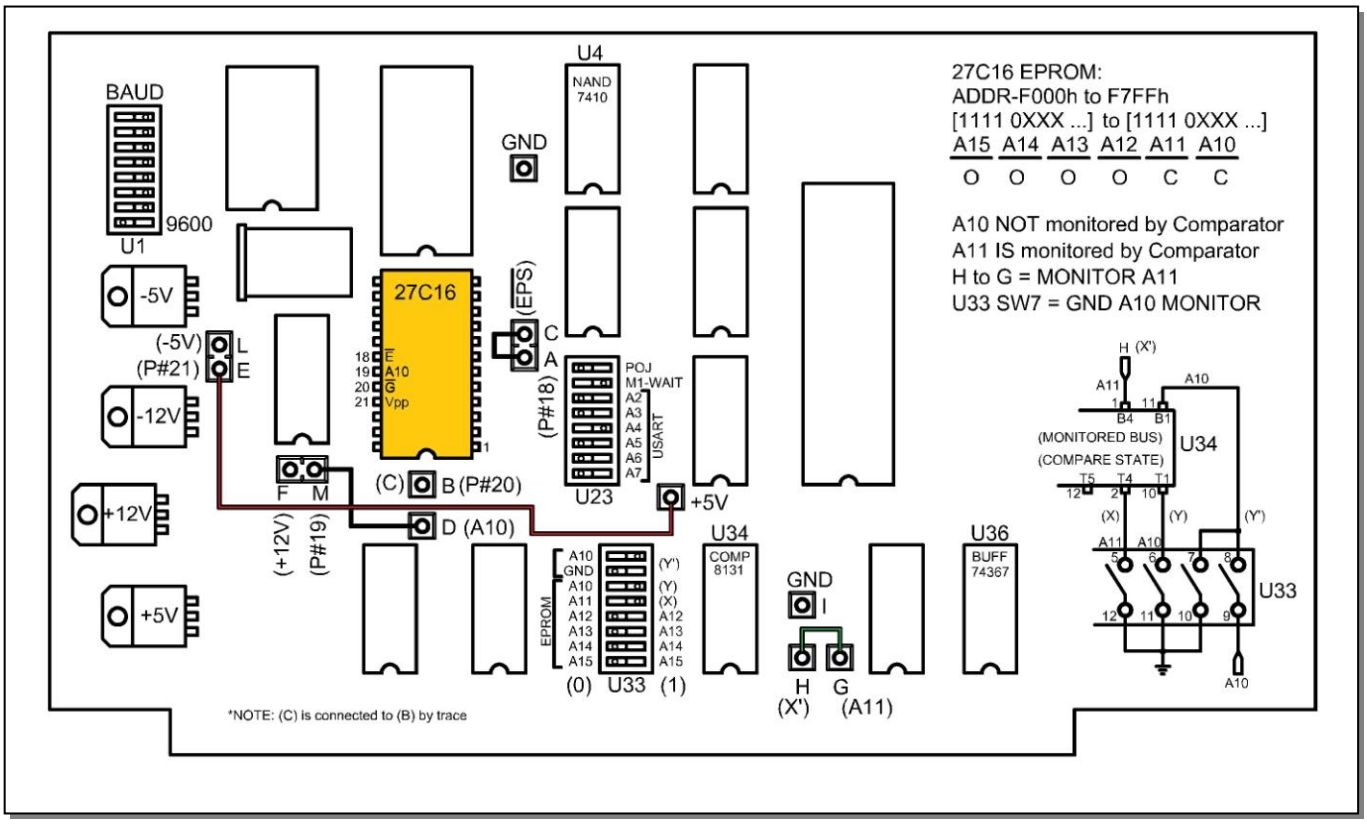
The Jade Big Z will accommodate three types of EPROM's (2708, 27C16 or 27C32). The board is originally configured for the 2708 EPROM but due to it's small 1K size and limited supply, it was never used. There are jumpers on the board to configure the card for the other two EPROM types. In fact, there are just two wires to connect for Pin#21 on the EPROMS (either Vpp or A11). All other connections remain the same for the 27C16 and the 27C32. EEPROMS such as the Atmel AT28C16 can also be used in place of a 27C16 EEPROM. There is no counterpart for the 27C32 though.

The EPROM is first accessed at power on (if the POJ option is enabled) by starting at address 0000H and moving upward in address space until the address space set for the EPROM is reached. When this happens, the EPS* signal becomes active low, and the EPROM is enabled.

Code in the EPROM is then read and acted upon by the Z80 CPU. Typically this is where the Monitor Program for the Jade Board would reside in EPROM at a high address range such as E800H, F000H or F800H (more on this later).

The DIP Switch U33 is used to select the address for the EPROM. The Big Z is capable of utilizing "Shadow EPROM" mode that will enable the EPROM on boot-up, but thereafter will not be seen by the system. This might be useful for a boot to Disk System (refer to the users manual for details).

Wiring up a 27C16 EPROM is as follows…

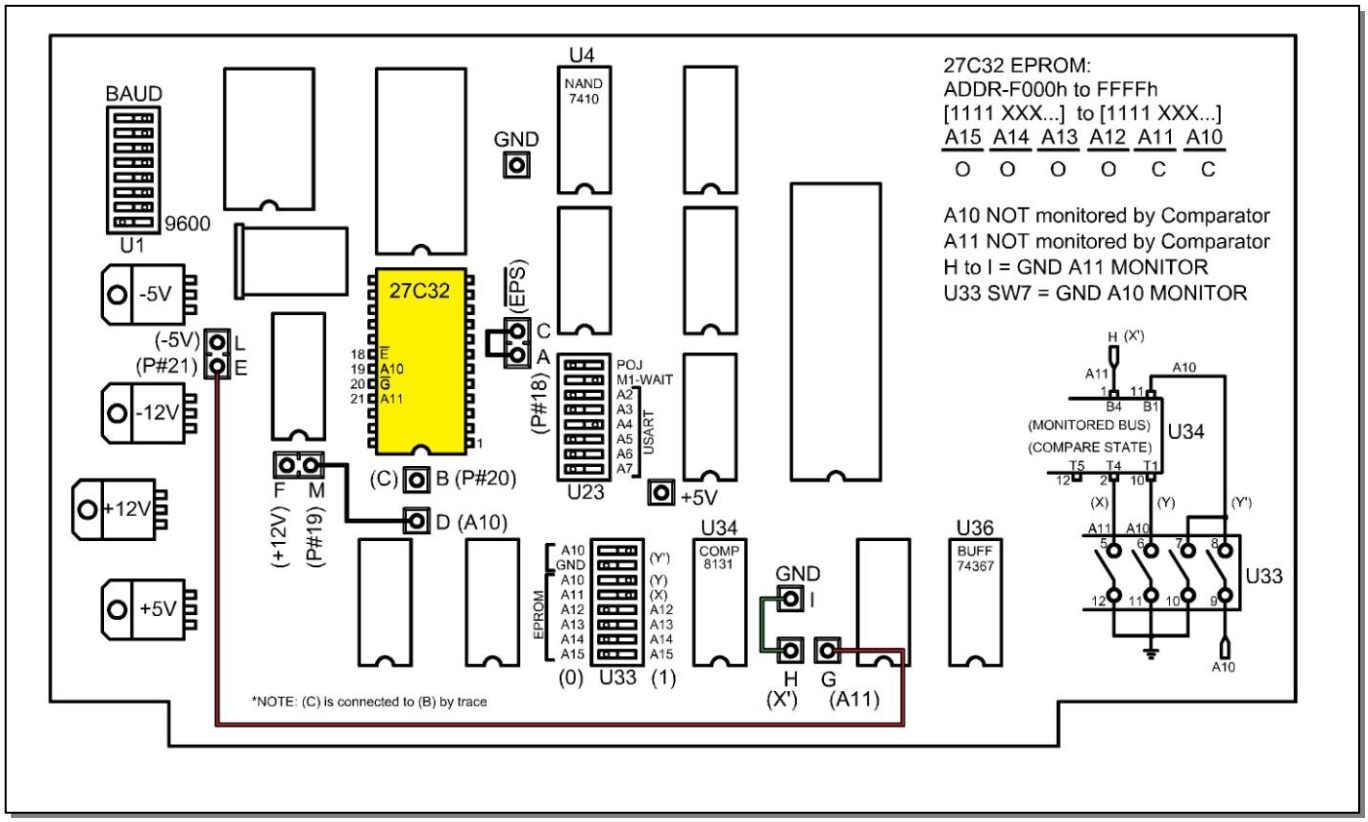


Wiring the 27C16 only requires running a wire-wrap wire from Point (E) to Point (+5V), this routes +5V to Pin#21 on the EPROM which is Vpp that needs to be High to operate.
Then run a wire-wrap wire from Point (H) to Point (G) thereby routing the A11 comparator bus sense input to A11. That's it as all other wires are already attached.

- Wire (E) to (+5V)
- Wire (H) to (G)
- Wire (C) to (A)

The DIP Switch address for the EPROM has to be set using the U33 DIP Switch. See the section below on how to do this.

Wiring up a 27C32 EPROM is as follows…



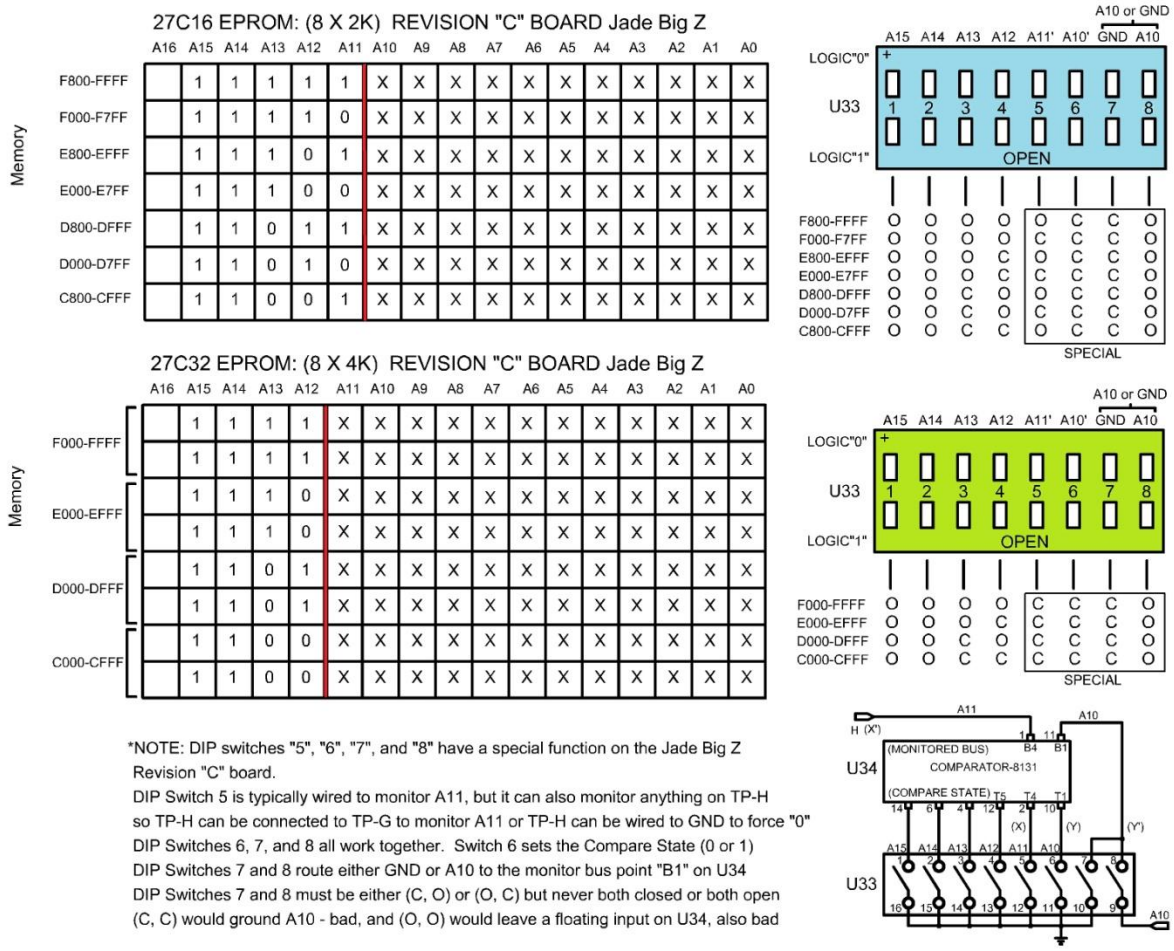
Wiring the 27C32 only requires running a wire-wrap wire from Point (E) to Point (G), this routes address line A11 to Pin#21 on the EPROM which is A11. Then run a wire-wrap wire from Point (H) to Point (I) thereby grounding the A11 comparator bus sense input. That's it as all other wires are already attached.

- Wire (E) to (G)
- Wire (H) to (I)
- Wire (C) to (A)

The DIP Switch address for the EPROM has to be set using the U33 DIP Switch. See the section below on how to do this.

EPROM ADDRESS SELECTION:

The EPROM address is selected by using U33 to enter the required address to activate the EPS* signal for the EPROM enable. The chart below illustrates some possible locations in High Memory for the EPROM to reside (27C16 or 27C32). Since the 27C16 is a 2Kx8 device there are smaller memory blocks allocated to it. The larger 27C32 is a 4Kx8 device and therefore fewer choices. The bits to the left of the red line are bits that identify memory blocks. These can be entered into the DIP switch by using an “open switch” as a logic 1 and a “closed switch” as a logic 0. A11 is special and A10 is not used unless a lower memory address is used. By “special”, this means the Jade Big Z has added a complicated way to represent these two bits. This is to allow more versatility for address selection to the board.



*NOTE: DIP switches "5", "6", "7", and "8" have a special function on the Jade Big Z Revision "C" board.
DIP Switch 5 is typically wired to monitor A11, but it can also monitor anything on TP-H so TP-H can be connected to TP-G to monitor A11 or TP-H can be wired to GND to force "0"
DIP Switches 6, 7, and 8 all work together. Switch 6 sets the Compare State (0 or 1)
DIP Switches 7 and 8 route either GND or A10 to the monitor bus point "B1" on U34
DIP Switches 7 and 8 must be either (C, O) or (O, C) but never both closed or both open (C, C) would ground A10 - bad, and (O, O) would leave a floating input on U34, also bad

Note the drawing on the lower right of the DM8131 comparator and DIP Switch. For a 27C32 the A11 line is routed to the EPROM via point (G) and the S5 switch is simply set O or C as shown in the charts. A10 on the other-hand is hard-wired to the DM8131 and it’s selection is accomplished by using three switches S6, S7 and S8.

The A10 logic state is selected by DIP S6 and will select the logic level to be compared as described above. S7 and S8 will route either A11 or GND (logic 0) to the bus comparator circuit side of the DM8131. S7 is the GND signal line and S8 is the A10 signal line. Either of these signals can be routed to the DM8131 but not both. Refer to the diagram above to clarify.

In the example chart shown above, the 27C32 does not use the A11 or A10 to enable the EPROM.

- Point (H) is tied to (GND) physically on the board with wire-wrap thereby setting the bus comparator sense input to "0" taking it out of the picture (not used).
- A11 DIP S5 is closed or "0" thereby making it match the unchanging bus comparator input and taking it out of the picture (not used).
- DIP S7 is closed "0" and DIP S8 is open not allowing A10 to reach the comparator input and taking it out of the picture (not used)
- A10 DIP S6 is closed or "0" thereby making it match the unchanging bus comparator input and taking it out of the picture (not used).

The same process is used for the 27C16 except A11 is used and only A10 has to be adjusted using DIP S6, S7 and S8 to get it out of the picture.

- DIP S7 is closed "0" and DIP S8 is open not allowing A10 to reach the comparator input and taking it out of the picture (not used)
- A10 DIP S6 is closed or "0" thereby making it match the unchanging bus comparator input and taking it out of the picture (not used).

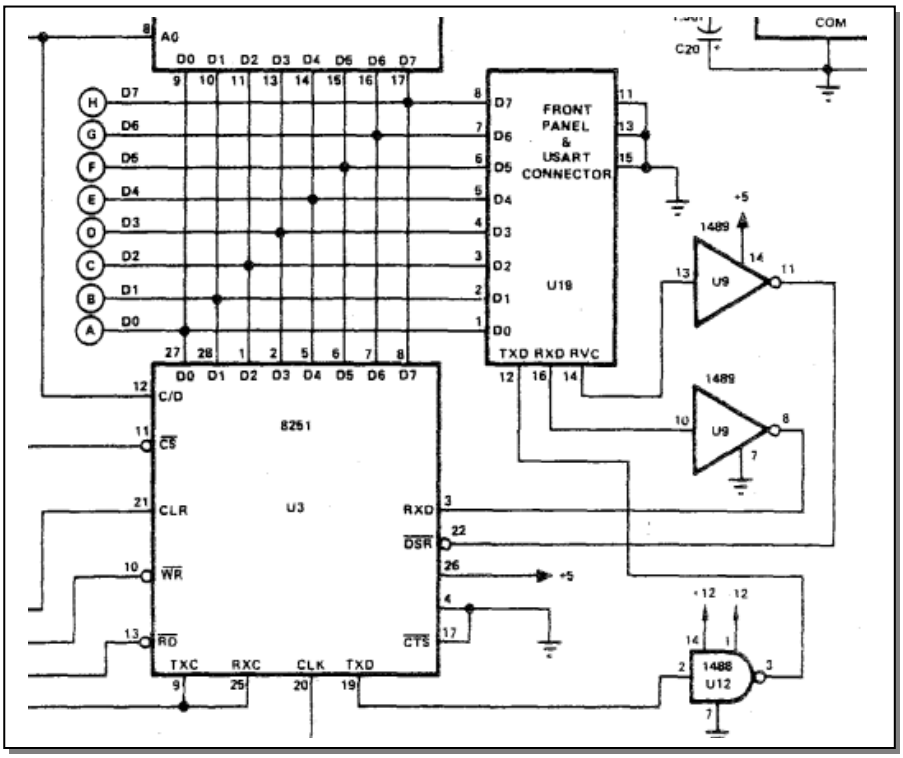
This is how the address is set for the EPROM. There are charts in the Jade Big Z manual that just give the "O" or "C" positions of the U33 DIP switch, but there are errors in the chart depending upon which revision of the board is being used. The above information describes how the DIP switch is set for any address and applies to the revision "C" version of the board.

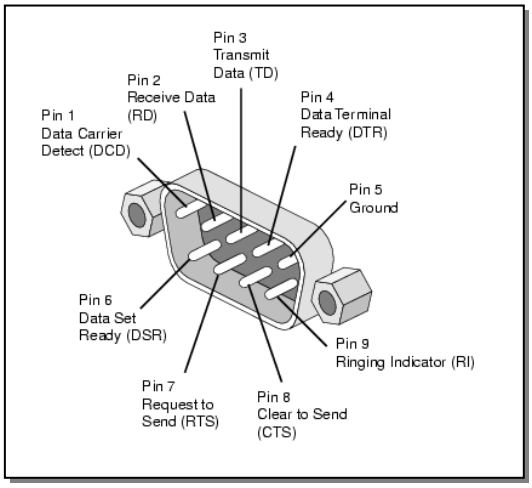
Remember that capacitor C12 is installed for filtering the surge current when using the -5 volt supply on a 2708 EPROM. This capacitor appears to attenuate or distort the A11 signal to a 2732 EPROM making the EPROM incapable of being accessed. This may be due to bad capacitors used on the Jade Board as some of the .1mfd monolithic capacitors have been found to be shorted out. In any case, if not using the 2708 EPROM it is a good idea to remove the C12 capacitor located directly below the 1.8432 MHz crystal.

8251 USART:

The Jade Big Z has an onboard 8251 USART to be primarily used as a console input/output allowing the Jade to communicate to the outside world. The USART could also be used as an RS-232 serial port but there are better dedicated cards that perform this function so this limits the USART to console I/O.

The connector for the USART is the DIP socket U19 on the Jade board. One side of this socket is used for the RS-232 communication and the other side can be used for a front panel connection. **NOTE PIN#1 LOCATION ON U19! DON'T INSERT PLUG BACKWARDS!**





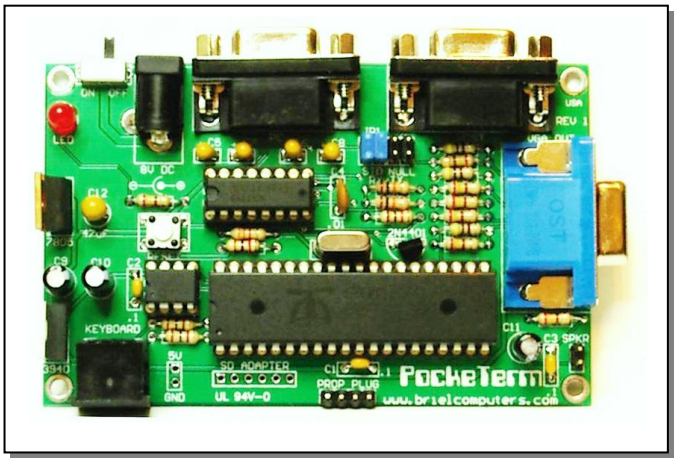
A DB-9 (RS-232) cable can be constructed by wiring the following pins…

- TXD (Pin12) to RD (Pin2)
- RXD (Pin16) to TD (Pin3)
- GND (Pin15) to Ground (Pin5)

That's it, no handshaking required

There is a “Reverse Channel” signal line provided by the Big Z on Pin#14 of U19. RVC can be used as a “Busy” or “Data Not Ready” signal from the Host Equipment to the Jade Big Z USART. This would be accomplished by wiring U19 Pin#14 (RVC) to Pin#6 (DSR).

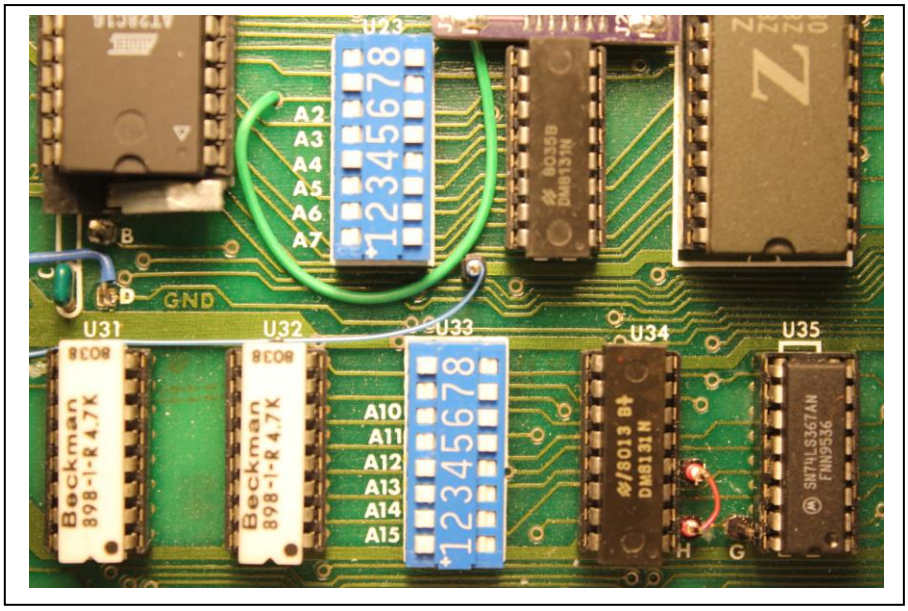
In practice, for console I/O, this signal was not needed as the Big Z controlled all communication and would be fast enough to loop waiting for a keyboard input from the USART and then sending output back at such a slow data rate (9600 baud) that both Host and Jade had no problem keeping up. The other end of the cable should be connected to a RS-232 (VT-100) capable terminal (+/- 12VDC). A PC running emulation software. A laptop or any other terminal device can be used. One option is to use the Propeller driven “Pocket Term” by Briel Computing.



A VGA Monitor is connected to the “blue” connector in the picture above. The Big Z RS-232 cable is connected to the “HOST” connector (one of two on back) and an IBM keyboard is connected to the PS/2 connector on the front of the board. The (2) jumpers in the middle may have to be switched (they act as gender changers). When working, the following is displayed…

Address for the 8251 USART:

The USART is accessed as one of 255 Ports available to the Z80 CPU. The address of the USART Port is set by using DIP Switch U23 (S1-S6). The USART appears to the Z80 as two consecutive port I/O address. U24 on the Big Z decodes a group of four consecutive addresses and the two lower addresses are used for USART communication. An "ODD" address will select the "Status Port" and an "EVEN" address will select the "Data Port".



U23 is used to set the Port Address for the USART. In this example, the USART has been assigned Port 10H and Port 11H as Data Port and Status Port.

10H = [0001 0000]B
11H = [0001 0001]B

A7	A6	A5	A4	A3	A2
0	0	0	1	0	0

OR

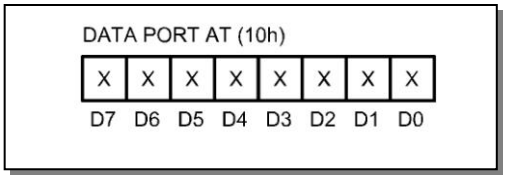
A7	A6	A5	A4	A3	A2
C	C	C	O	C	C

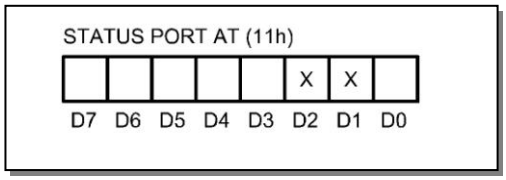
So this is what is entered into the U23 DIP Switch (S1-S6).

As to the 8251 USART itself, different commands can be entered to the USART and different status bits can be used to indicate conditions within the USART itself such as “BUSY”, “READY TO SEND” ect…

DATA PORT & STATUS PORT:

As mention above, the two Ports chosen for the USART communication are described as Data Port and Status Port. These are 8-bit words used to transmit data to and from the Big Z.





In it’s most simple form, the Status Port provides hand shaking control while the Data Port actually transmits and receives the 8-bit data word. A code snippet to do this is shown below…

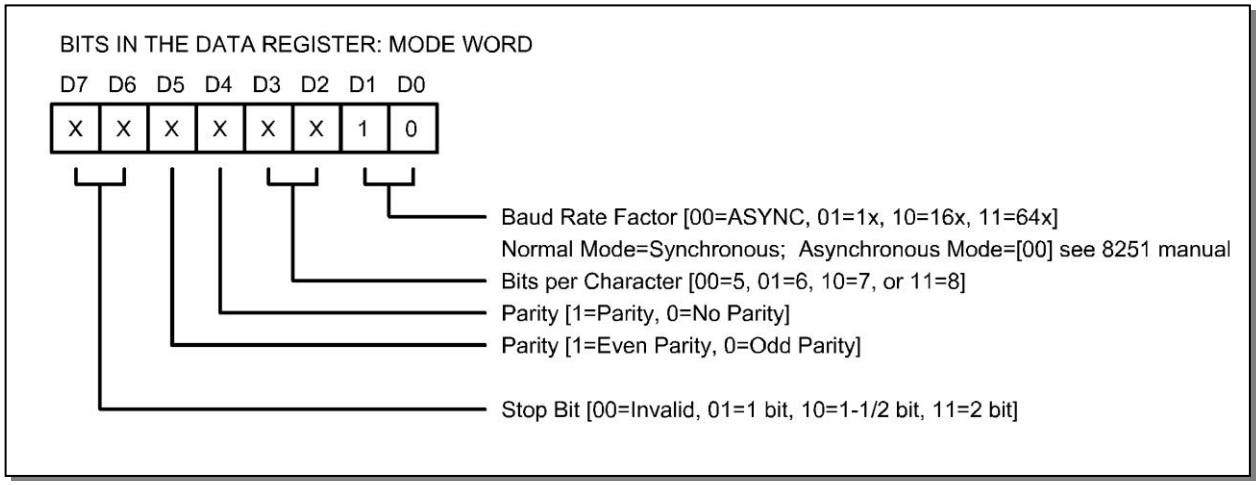
```
INPUT: IN    A,(11H)     ;Read keyboard status [0000 00X0]
       AND   02H         ; 02H = [0000 00X0] evaluate the “X”
       JP    Z,INPUT     ;Loop if not ready,  [0000 0010]=RDY [0000 0000]=NOT RDY
       IN    A,(10H)     ;Get keyboard data
```

```
OUTPUT:  IN    A,(11H)     ;Read keyboard status [0000 0X00]
         AND   04H          ; 04H = [0000 0X00] evaluate the “X”
         JP    Z,OUTPUT    ;Loop if busy,  [0000 0000]=BUSY [0000 0100]=NOT BUSY
         LD    A,C
         OUT   A,(10H)     ;Output character to console
```

This is how data gets into and out of the Jade Big Z; but before this can happen, the 8251 USART must be initialized via software. This is quite complex but offers great versatility without hard wiring the USART.

PROGRAMMING THE 8251 USART:

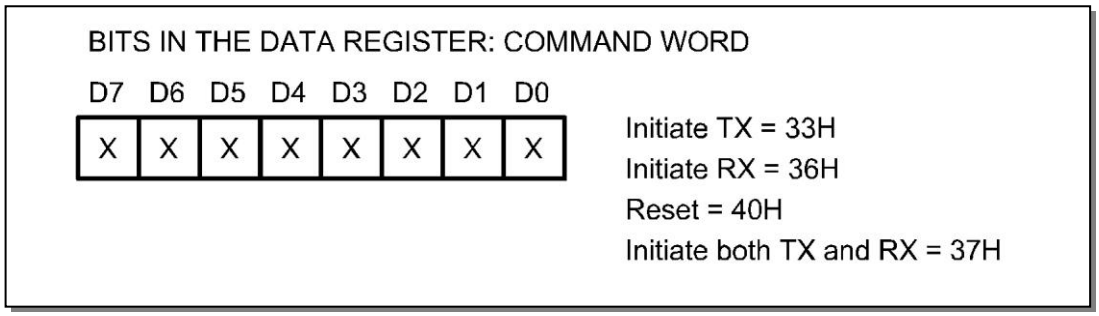
Before the 8251 USART can be used for anything, it must be initialized with word length, stop bits, parity and parity type. This is done by sending a "**MODE WORD**" to the 8251 prior to communicating with it. A MODE WORD is described as follows…



So, to initialize the 8251 USART for **8 data bits, no parity, odd parity, and 1 stop bit**, the following "MODE WORD" would be sent to the 8251 USART…

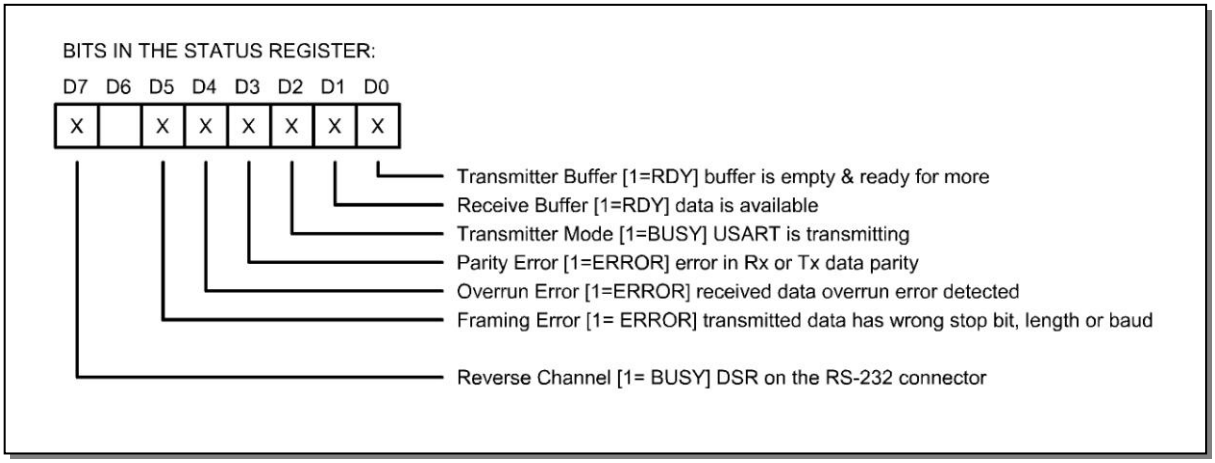
[0100 1110]B or **[4E]H** is the **MODE WORD** sent to the USART

After writing the Mode Word to the 8251, there should be a slight delay and then the "**COMMAND WORD**" would be sent. The delay is accomplished by using a "LD A" instruction followed by the Command Word. The Command Word is sent to control the transmit or receive function of the USART.



The Mode and Command Word are sent only once after a power-on sequence or reset is performed.

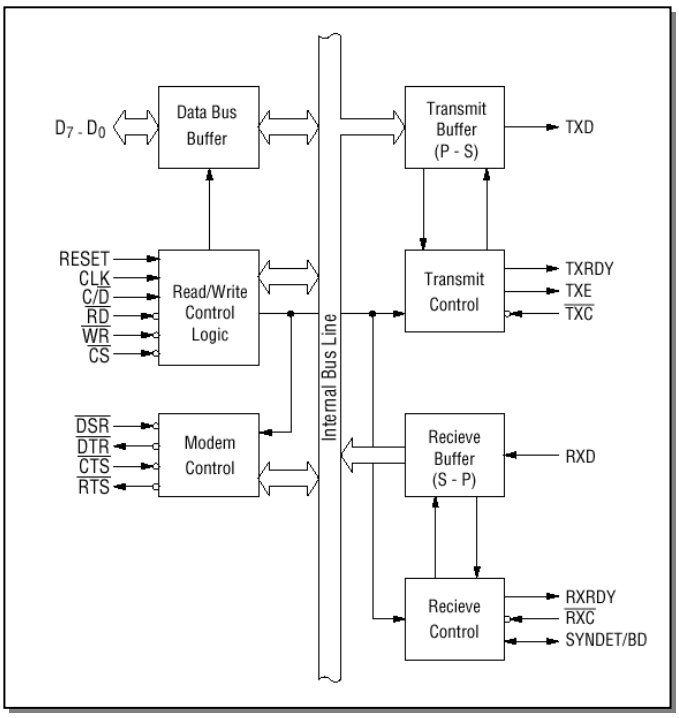
The **Status Register** of the USART is used to determine operating conditions within the 8251 USART as follows…



If the 8251 USART is used solely for console I/O, the main Status Register bits to be concerned with are D1 and D2.

INPUT PORT [00]H = 02H then **DATA RDY** from keyboard; if the Port is = 00H then **DATA NOT RDY**
OUTPUT PORT [00]H = 04H then the **HOST NOT BUSY**; if the Port = 00H then **HOST BUSY**.

The other Status Register bits are useful if more advanced RS-232 data operations are being used.



Block Diagram of the

8251 USART

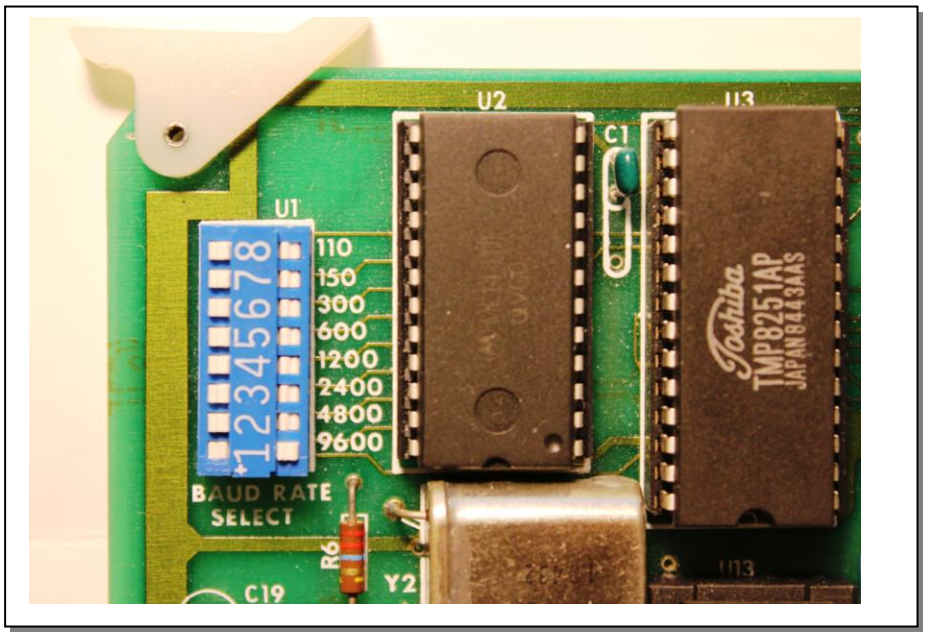
The following code snippet will initialize and set up the 8251 USART for operation…

(Note Foxit PDF reader ver 4.3.0.110 can convert this listing into ASCII text for use in a Z80 compiler)

```
; ************************************************************
;  BIG Z SAMPLE MONITOR USART I/O VERSION 1.0
; ************************************************************
;                                          Asynchronous Communication Mode
;                                          Reset
;                                          Mode Instruction (Asych or Synch, Baud Rate, Word Length, Stop Bits, Parity)
;                                          Mode Word (Write): D7+D6=Stop Bits,D5=E/O Parity,D4=Parity Enable,D3+D2=Char Length,D1+D0=Baud
;                                          *Note: a different Mode Word for Asynchronous Communication is used
;                                          Command Instruction (DTR, RTS, Hunt Mode, Xmt Enable, Rxv Enable)
;                                          Cmd Word (Write): D7=Hunt,D6=Int Rst,D5=RTS,D4=Err Rst,D3=Snd Brk,D3=Rx Enable,D1=DTR,D0=Tx Enable
;                                          Status Word (Read): D7=DSR,D6=Syn Det,D5=Frame Err,D4=Overrun Err,D3=Parity Err,D2=Tx Empty,D1=Rx Rdy,D0=Tx Rdy
            ORG         0E000H
;
SSTAT       EQU         11H             ;8251 Status port
SDATA       EQU         10H             ;8251 Data port
TXRDY       EQU         01H             ;TRANSMIT READY = (0000 0001) or (01)H
;
;                                        initialize USART send 00H three times to guarantee device in "Command Word" mode
;
INIT:       LD          A,00H           ;initialize USART
            OUT         (SSTAT),A
            LD          A,00H           ;initialize USART
            OUT         (SSTAT),A
            LD          A,00H           ;initialize USART
            OUT         (SSTAT),A
            LD          A,40H           ;Send reset "Command Word “  (0100 0000) or 40H and ready 8251 to receive a "Mode Word"
            OUT         (SSTAT),A
;                                        Mode word:(01)-1 stop bits (00)-parity disabled (11)-char length 8 bit (10)-baud 16X
;                                        Mode word: (01001110) or (4E)Hex...1xBaud = 153,600 1/16xBaud = 9,600 1/64xBaud = 2,400
            LD          A,4EH           ;Mode register 8,1,n,9600 or 4EH
            OUT         (SSTAT),A
;                                        Command word:(0)-disable hunt mode (0)-do not return to mode word (1)-reset output 0
;                                        Command word:(1)-reset all error flags (0)-normal operations (1)-receive enable
;                                        Command word:(1)-DTR will output "0" (1)-transmit enable
;                                        Command word: (0011 0111)Binary or (37)Hex
            LD          A,37H           ;Command register 37H essentially enables both transmit & recieve modes
            OUT         (SSTAT),A
;
TEST:       IN          A,(SSTAT)
            AND         TXRDY           ;is transmitter buffer ready (0&0=0,0&1=0,1&0=0,1&1=1)...if SSTAT=1 AND TXRDY=1 the loop exits
            JP          Z,TEST          ;loop until it's empty
;
;                                        Output to say we reached this point "U"
            LD          A,56H
            OUT         (SDATA),A
;
            JP          TEST
            END
```

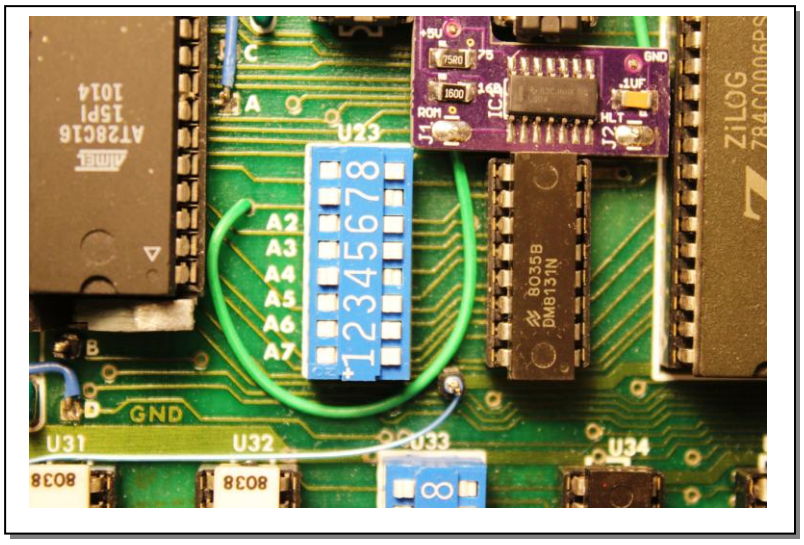
8251 USART BAUD RATE:

The Baud Rate for the 8251 is selected by using DIP Switch U1. The Baud Rates are clearly labeled on the circuit board. Rates go from 110 baud to 9600 baud. Only one switch on U1 can be closed at a time or the baud rate generator will not function.



REVISITING U23 DIP SWITCH:

The remaining two switches (S7 & S8) are option switches controlling the M1 Wait State and the (POJ) Power-On Jump to EPROM functions.

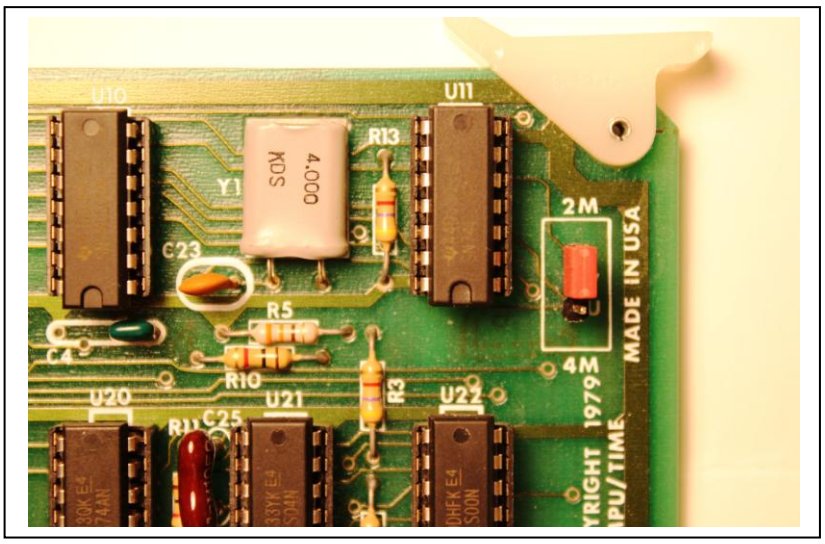


DIP Switch U23

- S7- OPEN=Wait Off
- S7- CLOSE=M1 Wait On
  Jumper **R** to **F**
- S8- OPEN=POJ Off
- S8-CLOSE=POJ On

BIG Z SPEED OPTION:

The Big Z CPU can operate at 2MHz or 4MHz depending upon the position of the T, V, U jumper.



Operation at 4MHz has been successful with the Jade DD Controller Card and a static RAM board but this depends on many factors and is not easy to get working. The other problem with the 4MHz operation may be due to the 8251 not being a 4MHz part. Substitution of a faster USART may resolve the issue or a USART wait state could be implemented as described in the Big Z Manual and Engineering Update #104.

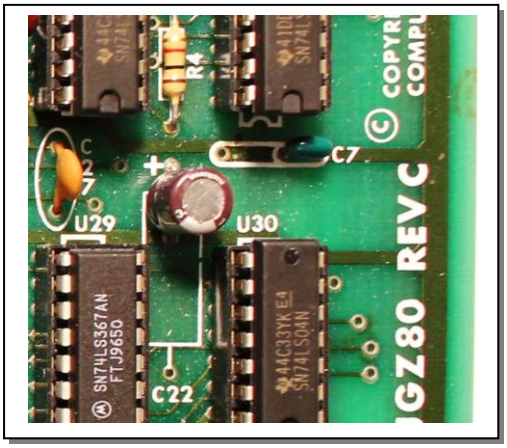
ENGINEERING UPDATES:

Engineering updates or "ECN" are listed in the back of the revision C Big Z user's manual. Most of the ECN's deal with errors in the User's Manual due to board revisions, EPROM tables, EPROM connections and errors in the program listings included within the User's Manual. In particular, the Jade Monitor listing in the manual does not work.

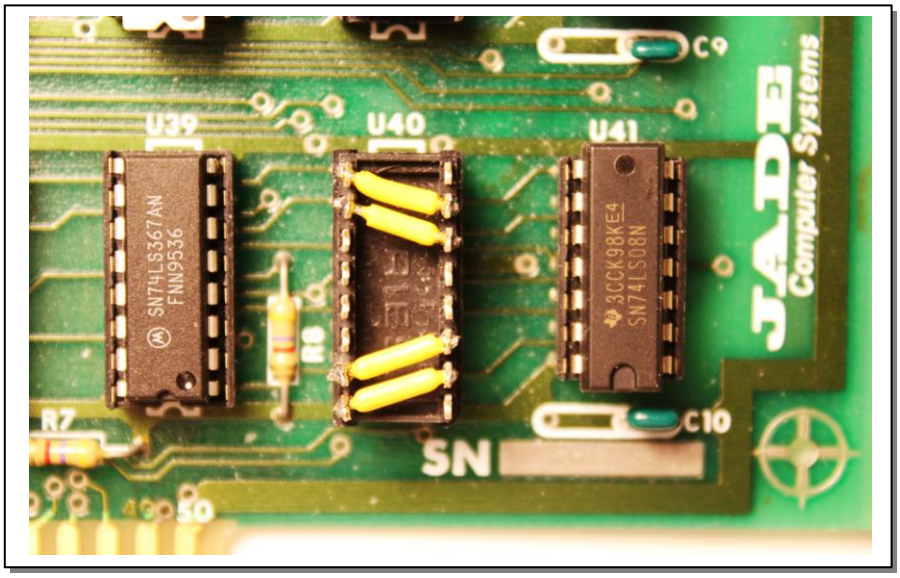
Two of the ECN's have been performed on the Jade Big Z revision C board…

- Erratic Reset ECN#101
- Status Delay Signal ECN#102

Erratic Reset Operation is caused by excessive time constant on RC network on input to U21 Pin#1. This time constant was chosen for operation with front panel systems and resulted in a delay of 470 ms after the Reset was activated. This may be too long of a delay for non-front panel systems. Capacitor C22, a 100mfd capacitor was removed and a smaller 10mfd capacitor was installed in it's place. This may have to be fine-tuned up to around 22mfd before acceptable operation is observed.

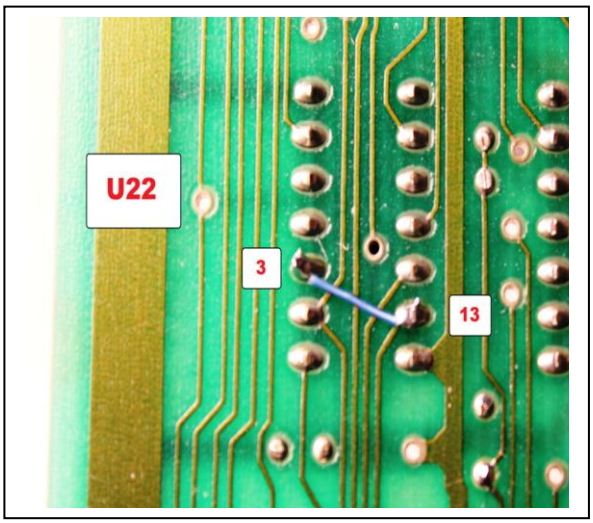


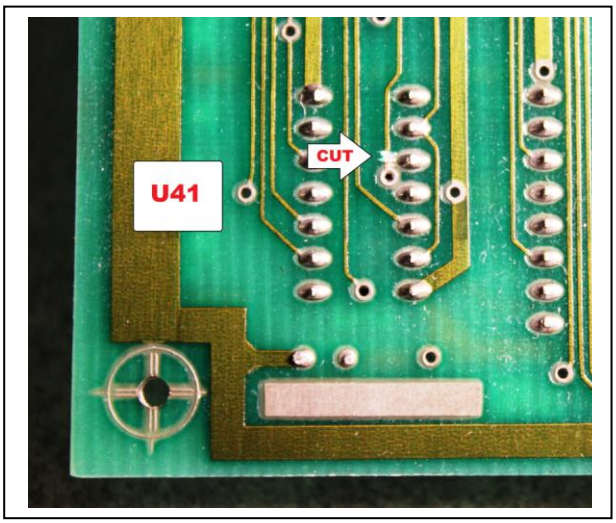
The **Status Signals** from a front panel display are latched by U40 to provide for a stable display operation of the front panel. Unfortunately, passing the status signals through U40 slows them down enough to become non-compliant with some dynamic memory boards operating at 4MHz. To correct this, U40 is removed and a jumper DIP is installed in it's place. This removes the pSYNC delay introduced by the original circuit.



OTHER MODIFICATIONS:

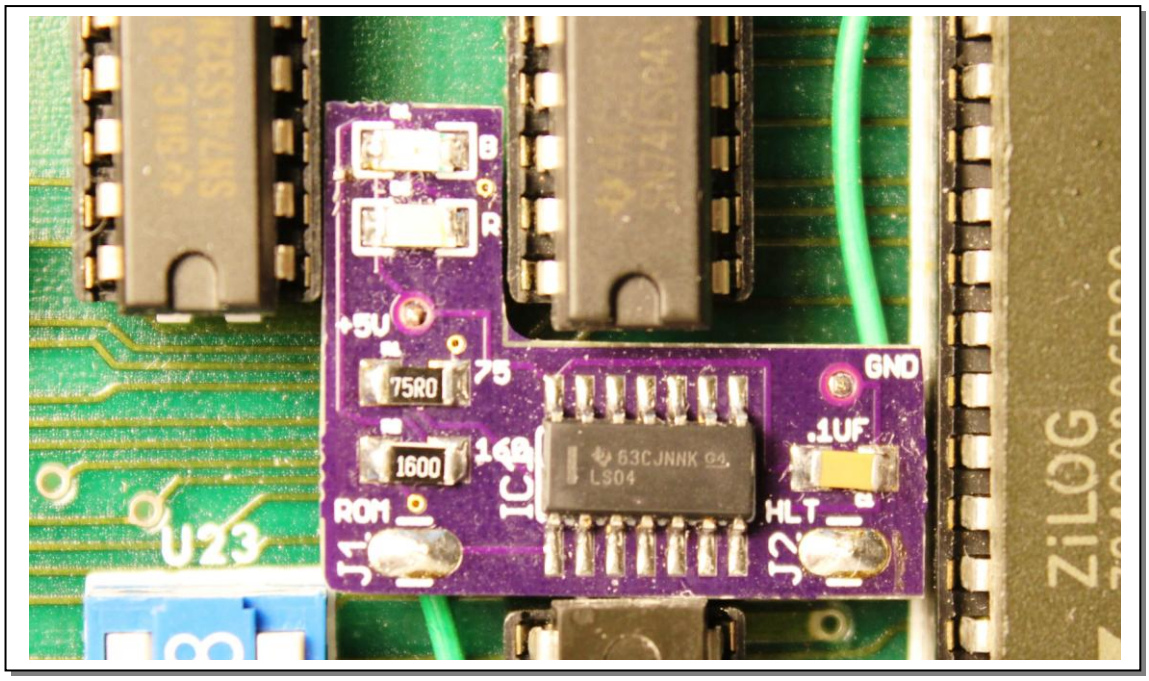
If the Jade Double D Disk Controller board is used in the system, one of the Engineering Notice Bulletins #4 was for erratic operation between the Jade Big Z and the Jade DD (from the Jade DD manual) was to modify U22. Cut the trace going to pin#13 of U22 and jump pin#13 to pin#3. If the Jade DD is not used this modification is not needed.





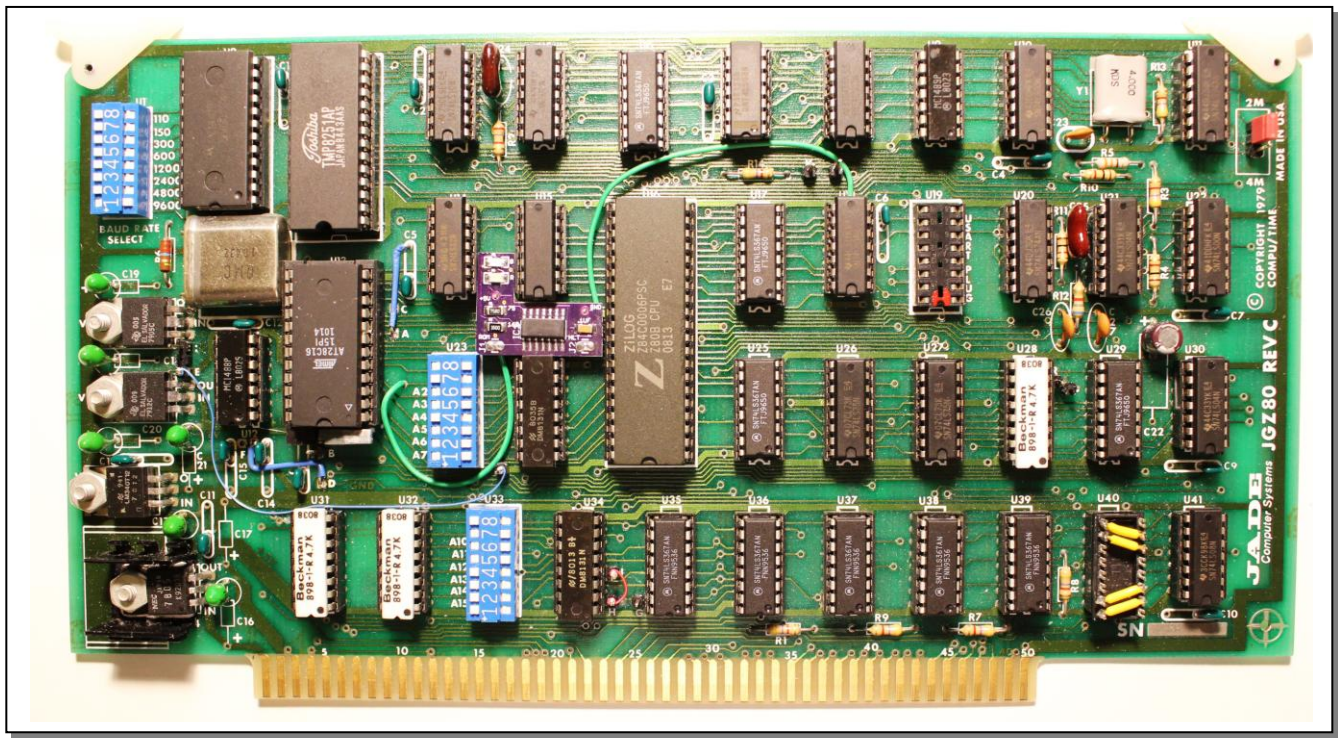
EPS & HALT:

One other modification done to the Big Z CPU board was the addition of two signal LED's to indicate when the EPROM address space is being accessed, and an LED to indicate when a software HALT instruction has been accessed by the CPU. These indicators are useful in determining if the EPROM is set up correctly and if the Big Z board is working by installing an EPROM filled with HALT instructions (76H) that will cause the processor to HALT and turn on the LED indicator.

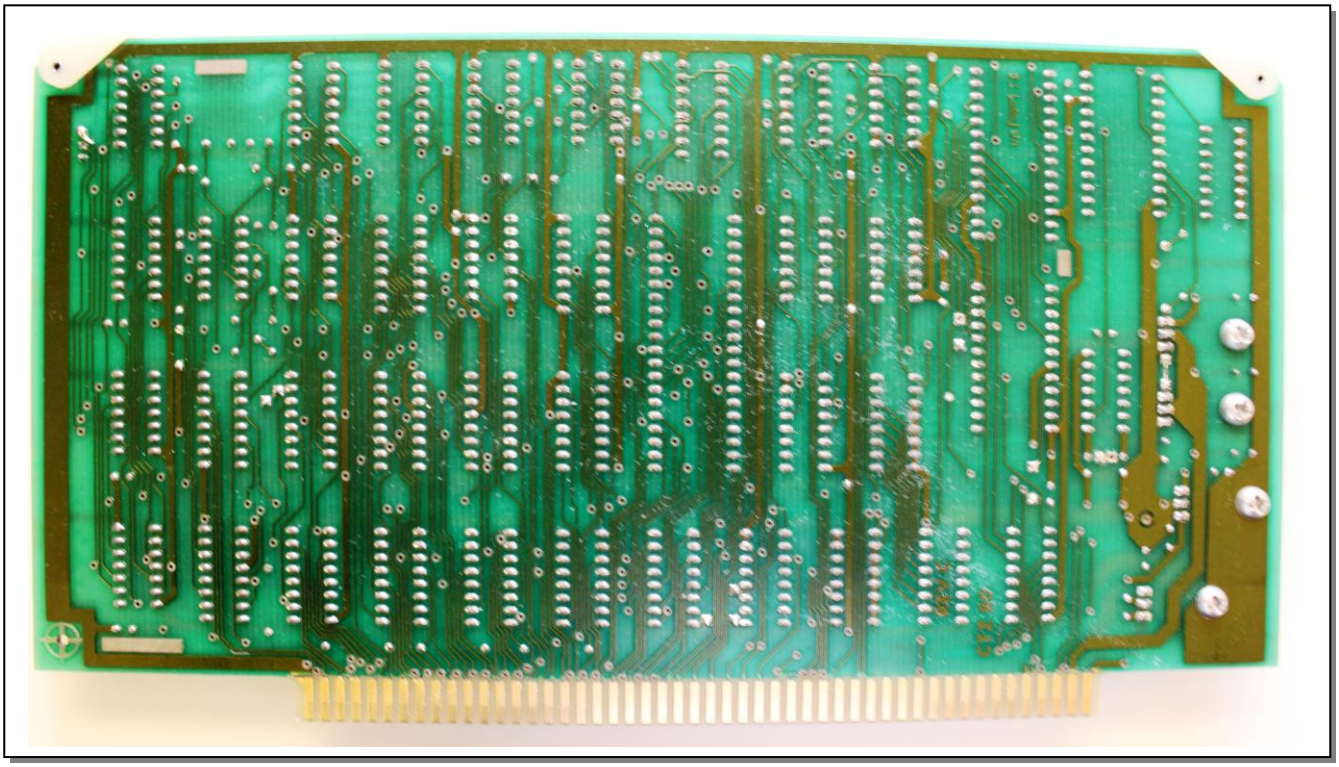


- BLUE LED: ON=EPROM being accessed **EPS** (within the U33 DIP switch address range)
- RED LED: ON=HALT instruction has been read by the CPU and stopped

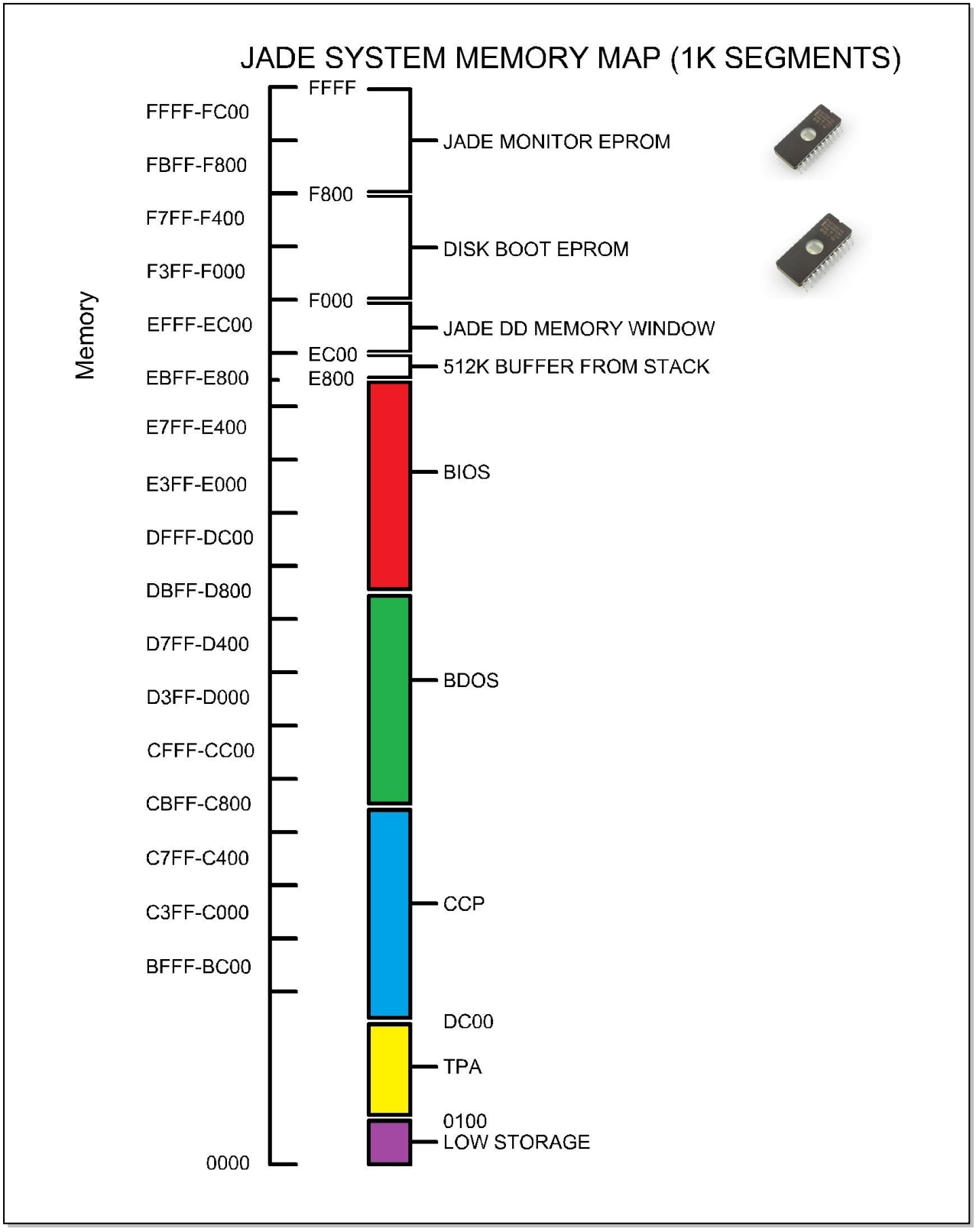
Jade Big Z Front:



Jade Big Z Back:



Jade Memory Map for Disk Based System:



Troubleshooting and Observations:

Hot +5V Regulator Heatsink: The +5V Regulator at VR4 gets rather hot during normal operation. Other (larger heatsinks) have been tried but still, this regulator runs hot. I believe this is normal operation for this board and should not cause problems but it is undesirable. For this reason, a new switching regulator was installed that does not heat up at all. This regulator is manufactured by (EzSBC.COM) and is rated at +5V @ 3 Amps.

This is a 5V 2.5A switch-mode voltage regulator. It is a high-efficiency replacement for popular three-terminal LM323T linear regulators and it is pin-to-pin compatible with the common and now obsolete LM323T linear regulators. The mechanical design allows the PSU5a to fit anywhere where an LM323T or an LM7805 was used. The maximum continuous output current is 3A and at room temperature the PSU5a does not need a heatsink to maintain this current indefinitely. All the required capacitors are included on the module, no external capacitors are required and additional input capacitors do no harm. The output voltage guaranteed to be within +/-1% as the load varies. The original LM323T had a rather loosely specified output voltage and it could vary by as much as 250mV without load and at room temperature. The PSU5a is accurate to winthin +/-2%. The module has thermal shutdown and current limit protection. The absolute maximum input voltage is 20V.

- *Drop-in replacement of the obsolete LM323T or equivalent linear voltage regulator.*
- *Guaranteed 3A output current.*
- *Input voltage range of 7.2V to 20V*
- *Suitable for use in Pinball machines and video game consoles*
- *High efficiency switching regulator design reduces power dissipation with superior voltage regulation compare to the LM323T.*
- *Thermal shutdown and current limit protection*
- *All components are mounted on one side of the PCB*
- *Highest component is the inductor at 5mm above the PCB.*
- *Available with or without pins.*
- *Gold plated pins and PCB to withstand harsh environments over the long term.*
- *Can drive inductive loads such as solenoids and DC motors.*
- *500kHz Switching Frequency*
- *Made in the USA*

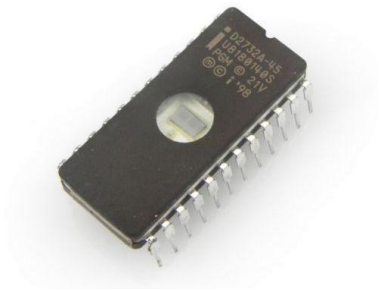
PSU5a 5V 3A Regulator in TO-220 form Factor

As a side note, the previous +5V regulator was changed out with another regulator with a higher Amp rating prior to trying the EzSBC switching regulator. This regulator worked but provided an output of 4.98 volts. This caused strange behavior in the Big Z CPU board. Random crashes, random HALT's and weird operation of the front panel displays (flickering of status LED's). I can't state emphatically that the lower 4.98 volts caused this problem, but after replacing the regulator with the EzSBC switching regulator, along with the two filter capacitors C16 & C17, all problems stopped occurring. The EzSBC output voltage was 5.01 volts. Food for thought…in the future, check the voltage output of the VR4 regulator to insure it is operating at +5.00 volts.

8251 USART Port: Problems with the serial port can be hard to diagnose. The RS-232 connection from the Terminal Equipment to the Jade Big Z is through the 16-Pin DIP socket on the board U19. If the connector plug is installed backwards, +/- 12 volts is applied to the CPU data bus directly and can cause damage. Make sure the plug inserted into U19 is correctly oriented with Pin #1 closest to the Gold Fingers on the bottom of the card. Verify Pin #1! If the serial port becomes unresponsive, replace the MC1488 and MC1489 chips first as they are the interface between the RS-232 and TTL logic. If the computer is turned off, the Big Z plugged in with serial connection on, and there is a small voltage bleeding through on the +12V or -12V system rails; this may indicate failure of the 1488 and/or 1499.

System Monitor known working … JADEV3F.BIN/JADEV3F.ASM 2K 2716 EPROM
Origin= E800-EFFF 8251 Data Port=10H & Status/CMD Port=11H

SWITCH	1	2	3	4	5	6	7	8
U1=	C	O	O	O	O	O	O	O
U23=	C	C	C	O	C	C	O	C
U33=	O	O	O	C	O	C	C	O



JADE Firmware: (EPROM and/or Disk)

Big Z Monitor "A" SFX-58001020E $29.95
Monitor program on 2708 EPROM for JADE Big Z CPU,
original JADE parallel-serial I/O board, serial terminal,
and Versafloppy I or Tarbell disk controller.

Big Z Monitor "B" SFX-58001025E $29.95
Similar to version A, but uses Tarbell cassette for tape I/O.

Big Z Monitor "C"/5-1/4" SFX-58001030E $49.95
Combination monitor and CP/M BIOS for Big Z,
serial terminal, Versafloppy I, and 5-1/4" drives (2716).

Big Z Monitor "C"/8" SFX-58001040E $49.95
Same as above, for use with 8" drives.

Big Z Monitor "D"/5-1/4" SFX-58001050E $49.95
Combination monitor and CP/M BIOS for Big Z, serial terminal,
JADE Double-D disk controller, and 5-1/4" drives.

Big Z Monitor "D"/8" SFX-58001060E $49.95
Same as above, for use with 8" drives.

Double-D Boot SFC-58001200E $20.00
Standard bootstrap routine for JADE Double-D
disk controller (2708).

(Note Foxit PDF reader ver 4.3.0.1110 can convert this listing into ASCII text for use in a Z80 compiler)

Monitor listing based on Big Z Monitor “A” and Big Z Monitor “B”
Note: Cassette functions are untested on actual hardware 07/15/24 so may not work all other functions should work

```
;***************************************************************
;           BIG Z MONITOR (2K VERSION 3.0) 9/10/79 AB
;***************************************************************
;           VERSION: JADEV3FC.Z80   JUNE 25,2020            BY AD
;           TAPE FUNCTIONS INCLUDED NOW...VERSION C
;           BACKGROUND-THIS MONITOR CODE IS FROM THE JADE BIG Z REVISION C MANUAL 1K ROM MONITOR
;           THE VERSION WAS 2.0 A/B FOR CASSETTE STORAGE A=JADE 2S1P/AND B=TARBELL
;           FROM THE ENGINEERING NOTES, THIS VERSION WAS KNOWN NOT TO WORK AND WAS POORLY COMMENTED
;           THE CODE WAS MODIFIED WITH THE GOAL OF KEEPING THE BASIC MONITOR FUNCTIONS INTACT AND
;           ADDING TO THE MONITOR WITH IMPROVEMENTS AND EXTENSIVE COMMENTS WHERE POSSIBLE
;           BETTER MENUS & PROMPTS, DR DOBBS MEMORY MAP, PORT IDENTIFER (S100.COM).  TAPE FUNCTIONS
;           THAT LOAD & SAVE DATA FOR KCTAPE ARE JADE ORIGINALS W/CHECK SUMS (BUT THERE IS NO
;           STANDARD FOR THIS SO EXAMINE THE CODE).  THE TARBELL TAPE ROUTINES ARE BASED ON THE TARBELL
;           MANUAL AND ARE NOT JADE ORIGINAL ROUTINES.
;           THE MONITOR ROM WENT FROM A 1K 2708 TO A 2K 2716 EPROM.
;
;           ASSUMPTIONS:
;           8251 SERIAL PORT ON BIGZ IS SET TO PORTS 10 AND 11H
;           OR S100.COM PROPELLER CONSOLE BOARD AT PORTS 00H AND 01H
;           (s100.COM PROP MUST ONLY ADDRESS THE 256 PORTS FOR A PRE-IEEE696 MACHINE)
;           TARBELL TAPE USING STANDARD TARBELL PORTS
;           OR KC STANDARD VIA JADE SERIAL/PARALLEL CARD 2S1P
;           WITH AY51013 UART SET TO PORTS 00 & 80 HEX
;           NO MEMORY SIZE IS ASSUMED
;           ASSUME A VT-100 SERIAL TERMINAL CONNECTED TO JGZ80 8251 USART USING VT-100 'ESC' COMMANDS
;
;           PROGRAMMING THE JGZ80 8251 UART
;           Asynchronous Communication Mode
;           Mode Instruction MSB(Asych or Synch, Baud Rate, Word Length, Stop Bits, Parity)LSB
;           MODE WORD (Write): D7+D6=Stop Bits,D5=Even Parity,D4=Parity Enable,D3+D2=Char Length,D1+D0=Baud
;           *Note: a different Mode Word for Synchronous Communication is used but not used here
;           Command Instruction MSB(DTR, RTS, Hunt Mode, Xmt Enable, Rxv Enable)LSB
;           COMMAND WORD (Write): D7=Hunt,D6=Int Rst,D5=RTS,D4=Err Rst,D3=Snd Brk,D3=Rx Enable,D1=DTR,D0=Tx Enable
;           STATUS WORD (Read): D7=DSR,D6=Syn Det,D5=Frame Err,D4=Overrun Err,D3=Parity Err,D2=Tx Empty,D1=Rx Rdy,D0=Tx Rdy
;
;           PROGRAMMING THE JADE 2S1P CASSETTE PORT A AY51013/TRI602 UART
;           (note: This board does not actually have a programmable UART; but JADE used 74LS125 & 74LS97 to emulate one)
;           DATA Ports: "FOR A" can be either (00H,04H,08H,0CH,10H,14H,18H or 1CH) => HOLDS THE DATA WORD FOR I/O
;           CONTROL Ports: (DATA Port) + (80H) => WRITE ONLY  MSB(NP,TSB,NB2,NB1,EPS,XX,XX,XX)LSB
;           NP=1(NO PARITY),TBS=1(2 STOP BITS),NB2+NB1(00=5,01=6,10=7,11=8 BITS/CHAR),EPS=1(EVEN PARITY)
;           STATUS Ports: (DATA Port) + (80H) => READ ONLY  MSB(TBMT,PE,FE,DAV,XX,XX,XX,XX)LSB
;           TMBT=1(TX BUFF EMPTY),PE=1(PARITY IS BAD),FE=1(STOP BIT IS BAD),DAV=1(RX IS READY TO READ)
;
;           PROGRAMMING THE TARBELL- THE TARBELL CASSETTE CARD ONLY HAS TTL LOGIC WITH CONTROL/STATUS BITS STORED IN A 74LS75 LATCH
;           CORRESPOUNDING TO DATA BITS (D7,D6,D5,D4) OF AN 8 BIT WORD.  DATA PORT IS AT "6FH" AND THE CONTROL/STATUS PORT AT "6EH"
;           THE PORT ADDRESS BIT "A0" DIFFERENTIATES BETWEEN THE TWO.
;           DATA WORD= I/O (6FH)..............MSB[D7,D6,D5,D4,D3,D2,D1,D0]LSB
;           CONTROL/STATUS WORD= I/O (6EH)....MSB[X,X,TXRDY,RXRDY,  X,X,X,X]LSB RXRDY=10H,TXRDY=20H
;           RXRDY=00H TARBEL READY TO RECEIVE,TXRDY=00H TARBEL READY TO TRANSMIT
;           GP OUTPUT PORT (J1)- PORT 6EH MSB[X,X,X,X, D3,D2,D1,D0]LSB  WRITING A "1" TURNS BIT ON AND "0" TURNS IT OFF
;           GP INPUT PORT (J1)- PORT 6EH MSB[X,X,X,X, D3,D2,D1,D0]LSB  READING THESE 4 BITS FROM PORT 6EH WITH CARE
;           COULD BE USED AS INPUT CONTROLS
;           THE GP OUTPUT PORT HAS FOUR BITS (40mA) AND "D0" IS USED TO CONTROL CASSETTE TAPE "MOTOR-ON" FUNCTION VIA RELAY
;           TARBELL DEFINES: [3CH]=START BYTE, [E6H]=SYNC BYTE, [FFH]=LEADER BYTE NOTHING ELSE IS SPECIFICALLY DEFINED
;
;           SLR SYSTEMS ASSEMBLER USED TO GENERATE HEX CODE FOR USE-(Z80ASM.COM) EXAMPLE: "Z80ASM JADEV3FC.Z80 FH"
;           JADE MONITOR ROM HAS BEEN COMPILED TO RESIDE AT (F800H-FFFFH). THIS WAS TO ALLOW FOR A DISK BOOT ROM TO RESIDE
;           AT (F000H-F7FFH) WITHOUT WASTING MEMORY SPACE.
;
;           COMPILER: COMPILED CODE SHOULD FIT INTO A 2716 EPROM WITH 2048 BYTES AVAILABLE
;           1999 BYTES - PROP/TARB
;           2023 BYTES - SERIAL/TARB
;           1889 BYTES - SERIAL/JADE
;           1865 BYTES - PROP/JADE
;
;           MENU COMMANDS:
;           A(MMOD)-MODIFY A MEMORY LOCATION
;           D(MDUMP)-DUMP A RANGE OF MEMORY TO THE CONSOLE
;           G(RUN)-GOTO AND RUN A PROGRAM AT THAT ADDRESS
;           K(MENU)-REFRESH THE MENU SELECTIONS/SCREEN
;           C(MMOVE)-THIS ACTUALLY COPIES A BLOCK OF MEMORY TO A NEW LOCATION
;           T(MTEST)-SIMPLE NON-DESTRUCTIVE MEMORY TEST THAT DISPLAYS MEMORY BITS THAT CAN'T BE CHANGED AS "11111111"
;           F(MFILL)-WILL FILL A BLOCK OF MEMORY WITH A SELECTED HEX CHARACTER
;           M(MMAP)-DR DOBBS MEMORY MAPPER THAT DISPLAYS RAM,ROM, AND MISSING MEMORY
;           L(MWRT)-ROUTINE TO ENTER SHORT PROGRAMS INTO CONSECUTIVE MEMORY LOCATIONS (MACHINE LANGUAGE PROGRAMS)
;           P(PORTS)-WILL SCAN THE 0-255 I/O PORTS AND DISPLAY VALUES THAT IT FINDS
;           S(CSAVE)-SAVE A BLOCK OF MEMORY TO CASSETTE TAPE
;           R(CLOAD)-LOAD FROM CASSETTE TAPE TO A SPECIFIED STARTING LOCATION IN MEMORY
;           V(MVER)-VERIFY A COPIED BLOCK OF MEMORY BY SPECIFIED STARTING, ENDING, AND NEW LOCATION ADDRESS
;           X(CSYNC)-GENERATE A SYNC STREAM TAPE USED TO ADJUST THE CASSETTE TAPE VOLUME VIA THE (CADJ) ROUTINE
;           Y(CASJ)-ROUTINE TO ADJUST THE CASSETTE PLAYER VOLUME CONTROL USING A SYNC STREAM TAPE
;           B(TARB)-TARBELL FLOPPY DISK BOOT ROUTINE WRITTEN TO MEMORY AND THEN RUN; WORKS ON A FD-1771; NO BOOT ROM REQUIRED
;           E(VERSA)-ROUTINE TO JUMP TO A VERSAFLOPPY BOOT ROM AT F000H AND LOAD A DISK SYSTEM FROM THERE
;           U(XPORT)-CHANGE AN I/O PORT VALUE 0-255 BY ENTERING PORT HEX NUMBER, THEN NEW HEX VALUE FOR THAT PORT
;
;
;           CONDITIONAL ASSEMBLY PARAMETERS
;
;           DEFINE VALUES OF TRUE/FALSE
TRUE:       EQU         0FFFFH
FALSE:      EQU         0
;
;                                           ########## Note: Choose either 8251 UART I/O or PROP I/O ############
;
;           DEFINE CONSOLE I/O
UART:       EQU         FALSE                   ; USE BIG Z ONBOARD 8251 FOR CONSOLE I/O
PROP:       EQU         TRUE                    ; USE S100.COM PROPELLER CONSOLE BOARD FOR I/O
;
;           DEFINE CASSETTE TAPE SYSTEM
TARBEL:     EQU         TRUE                    ; USE THE DON TARBELL CASSETTE BOARD
KCTAPE:     EQU         FALSE                   ; USE THE JADE SERIAL/PARALLEL BOARD
;
;           SYSTEM EQUATES
;
MON:        ORG         0F800H                  ; LOCATION OF JADE MONITOR ROM
;
;
                                                  ASSUME JADE BIG Z MONITOR IS AT (F800)H - (FFFF)H 2KROM
                                                  ASSUME VERSAFLOPPY II BIOS ROM AT (F000)H - (F7FF)H OR OTHER FLOPPY BIOS
;           IF          KCTAPE                  ;                                                   =========(f)IF
TAPE:       EQU         00H                     ; JADE 2S1P BOARD SELECT PORT B KC CASSETTE 'CURRENTLY SET TO PORT 0H'
TAPST:      EQU         80H                     ; JADE 2S1P BOARD ADDRESS UART B I/O (80H + SELECT PORT) 'CURRENTLY PORT 0 & PORT 80'
            ENDIF                               ; JADE 2S1P: TO PROGRAM THE UART OPERATION MODE LOAD THE SELECT PORT + 80  =========(f)ENDIF
;                                                 AS THE I/O ADDRESS THEN OUTPUT THE CONTROL WORD TO THAT ADDRESS...BAUD,PARITY,ETC
;                                                 THE INPUT & OUTPUT ADDRESS IS THE SAME.  THE CONTROL WORD IS AN OUTPUT WHILE THE
;                                                 STATUS SENSE IN AN INPUT.
;                                                 JADE 2S1P CNTL WORD (10110000)B = B0H => NOP,1STOP,8DATA
;                                                 JADE 2S1P STATUS SENSE (1xxxxxxx)B = 80H => TRANSMITTER BUFFER IS EMPTY (TBMT)=1
;                                                 JADE 2S1P STATUS SENSE (xxx1xxxx)B = 10H => CHARACTER READY TO TRANSMIT (DAV)=1
;
```

```
WAIT:       EQU         0FCH
SECT:       EQU         0FAH
DCOM:       EQU         0F8H
DDATA:      EQU         0FBH
DSTAT:      EQU         0F8H
SBOOT:      EQU         007DH
TARBL       EQU         06EH                    ; TARBELL CONTROL/STATUS PORT
;
            IF          UART                    ;                                                       ===============(e)IF
KBDST:      EQU         11H                     ; 8251 Command port
KBDDT:      EQU         10H                     ; 8251 Data port
KBDIN:      EQU         02H                     ; 8251 RECEIVE READY = (0000 0002)B or (02)H
KBDOT:      EQU         01H                     ; 8251 TRANSMIT READY = (0000 0001)B or (01)H
            ENDIF                               ;                                                       ===============(e)ENDIF
;
            IF          PROP                    ;                                                       ===============(g)IF
KBDST:      EQU         00H                     ; Status port is PORT (00)H
KBDDT:      EQU         01H                     ; Data port is PORT (01)H
KBDIN:      EQU         02H                     ; RECEIVE READY = (0000 0002)B or (02)H
KBDOT:      EQU         04H                     ; TRANSMIT READY = (0000 0100)B or (04)H
            ENDIF                               ;                                                       ===============(g)ENDIF
;
;                                                 THE FOLLOWING ARE JUMP SUBROUTINES THAT CAN BE ACCESSED BY OTHER PROGRAMS
;                                                 FOR THIS REASON THEY ARE LISTED HERE; BUT PROGRAM FLOW ONLY USES "INIT"

            JP          INIT
            JP          EXEC
            JP          CONIN
            JP          CONOUT
            JP          HEXIN
            JP          HEXOUT
            JP          DHXOT
            JP          CRLF
            JP          SPACE
            JP          TREAD
            JP          TWRIT
;
;                                   + + + +     DEFINE MESSAGES HERE + + + +
;
                                                ; MESSAGE PRINT ROUTINE [DO NOT USE COMMA AS PUNCTUATION!!!]
;
            IF          TARBEL                  ;                                                   ==================(h)IF
MSG1:       DEFM        'JADE COMPUTER SYSTEMS BIG Z MONITOR 3.0B'
            DEFB        03H                     ; 03H=END OF TEXT
            ENDIF                               ;                                                   ==================(h)ENDIF
;
            IF          KCTAPE                  ;                                                   ==================(i)IF
MSG1:       DEFM        'JADE COMPUTER SYSTEMS BIG Z MONITOR 3.0A'
            DEFB        03H                     ; 03H=END OF TEXT
            ENDIF                               ;                                                   ==================(i)ENDIF
;
MSG2:       DEFM        'TOP OF RAM:'
            DEFB        03H                     ; 03H=END OF TEXT
;
MSG3:       DEFM        '(A)MMOD  (D)MDUMP (L)MWRT (F)MFILL (C)MMOVE (M)MMAP (V)MVER (T)MTEST (G)RUN'
            DEFB        0DH,0AH                 ; 0DH=CARRIAGE RTN 0AH=LINE FEED
            DEFB        03H                     ; 03H=END OF TEXT
;
MSG4:       DEFM        '(S)CSAVE (R)CLOAD (P)PORT (U)XPORT (X)CSYNC (Y)CADJ (B)TARB (E)VERSA (K)MENU'
            DEFB        0DH,0AH                 ; 0DH=CARRIAGE RTN 0AH=LINE FEED
            DEFB        03H                     ; 03H=END OF TEXT
;
MSG10:      DEFM        ' xxxx <CR> <BS> <xx.> or </>exit'
            DEFB        0DH,0AH,03H ; CR LF EOT
;
MSG11:      DEFM        'BAD:'
            DEFB        03H                     ; END OF TEXT '03H'
;
MSG12:      DEFM        ' xxxx,xxxx <CR>'
            DEFB        0DH,0AH,03H ; C/R L/F EOT
;
MSG13:      DEFM        ' xxxx <CR> or </>exit'
            DEFB        0DH,0AH,03H ; C/R L/F EOT
;
MSG14:      DEFM        ' xxxx,xxxx,xx <CR>'
            DEFB        0DH,0AH,03H ; C/R L/F EOT
;
MSG15:      DEFM        ' xxxx,xxxx,xxxx <CR>'
            DEFB        0DH,0AH,03H ; C/R L/F EOT
;
MSG16:      DEFM        ' ADJ VOL GOOD(+)/BAD($)'
            DEFB        0DH,0AH,03H ; C/R L/F EOT
;
MSG17:      DEFM        ' SYNC STREAM TAPE:'
            DEFB        0DH,0AH,03H ; C/R L/F EOT
;
MSG18:      DEFM        ' xxxx,xxxx,xxxx <CR> MEMORY: <STRT>,<END>,<NEW>'
            DEFB        0DH,0AH,03H ; C/R L/F EOT
;
MSG19:      DEFM        'END'
            DEFB        03H                     ; END OF TEXT '03H'
;
MSG20:      DEFM        ' xxxx <CR>'
            DEFB        0DH,0AH,03H ; C/R L/F EOT
;
MSG21:      DEFM        ' XXXX <CR> PORT VALUE'
            DEFB        0DH,0AH,03H ; C/R L/F EOT
;
MSG22:      DEFM        ' DISK BOOTSTRAP LOADER'
            DEFB        0DH,0AH,03H ; C/R L/F EOT
;
MSG23:      DEFM        ' DISK BOOT ROM AT F000H'
            DEFB        0DH,0AH,03H ; C/R L/F EOT
;
INIT:                                           ; SET UP THE UART AND THEN INITIALIZE THE STACK
;
            IF          UART                    ;                                                                   ===============(j)IF
            LD          A,00H                   ; Initialize USART send 00H three times to guarantee device in "Command Instruction"
            OUT         (KBDST),A
            LD          A,00H                   ; INITIALIZE USART
            OUT         (KBDST),A
            LD          A,00H                   ; INITIALIZE USART
            OUT         (KBDST),A
            LD          A,40H                   ; Send internal reset "Command Instruction" (0100 0000) or 40H and ready 8251 to recieve a
"Mode Instruction"
            OUT         (KBDST),A
;                                                 Mode word:(01)-1 stop bits (00)-parity disabled (11)-char length 8 bit (10)-baud 16X
;                                                 Mode word: (01001110) or (4E)Hex...1xBaud = 153,600 1/16xBaud = 9,600 1/64xBaud = 2,400
            LD          A,4EH                   ; Mode register 8,1,n,9600 or 4EH
            OUT         (KBDST),A
;                                                 Command word:(0)-disable hunt mode (0)-do not return to mode word (1)-reset output 0
;                                                 Command word:(1)-reset all error flags (0)-normal operations (1)-receive enable
;                                                 Command word:(1)-DTR will output "0" (1)-transmit enable
;                                                 Command word: (0011 0111)Binary or (37)Hex
            LD          A,37H                   ; Command register 37H essentially enables both transmit & recieve modes
            OUT         (KBDST),A               ; INITIALIZE THE ONBOARD UART
            ENDIF                               ;                                                                   ===============(j)ENDIF
;
;                                                 TOP OF MEMORY ROUTINE AND SETUP STACK
```

```
FTOP:       LD          B,1                 ; SET POINTER TO "1"
            LD          HL,TRUE             ; PRELOAD MEMORY ADDRESS WITH "FFFF"
FTOP1:      INC         HL                  ; ADD 1 TO MEMORY POINTER HL (START AT "0000" GOING TO "FFFF")
            LD          A,(HL)              ; LOAD "A" WITH (HL) CONTENTS
            CPL                             ; MODIFY THE MEMORY CONTENTS
            LD          (HL),A              ; LOAD MEMORY LOCATION (HL) WITH MODIFIED CONTENT
            CP          (HL)
            CPL                             ; SEE IF MEMORY CONTENT COULD BE CHANGED
            LD          (HL),A              ; IF CHANGED=RAM, IF NOT CHANGED=TOP OF RAM
            JP          NZ,FTOP2            ; IF CHANGED=TOP OF RAM, JUMP OUT; OTHERWISE REPEAT
            LD          B,0                 ; SET POINTER TO "0", FOUND SOME MEMORY!
            JR          FTOP1
FTOP2:      LD          A,B                 ; LOAD "A" WITH "MEMORY POINTER"
            OR          A                   ;
            JP          NZ,FTOP1            ; IF POINTER IS "1", THERE IS NO MEMORY AT THIS LOCATION, TRY AGAIN
            DEC         HL                  ; SUBTRACT BY 1
            DEC         HL                  ; SUBTRACT BY 1
            LD          SP,HL               ; LOAD THE (SP) STACK POINTER WITH HL
            PUSH        HL
            POP         IY                  ; SAVE STACK ADDRESS IN IY
;
            CALL        CLRSCN              ; CLEAR SCREEN AND MOVE CURSOR TO UPPER LEFT CORNER
            LD          HL,MSG2             ; TOP OF MEMORY MESSAGE - ONE TIME ONLY AT BOOT
            CALL        MARQ                ; MESSAGE MARQUEE ROUTINE
            LD          HL,1
            ADD         HL,SP
            CALL        DHXOT               ; DISPLAY THE TOP OF MEMORY - HEX OUT TO CONSOLE
            CALL        CRLF
;
INIT1:      LD          HL,MSG1             ; DISPLAY THE JADE SIGN-ON MESSAGE
            CALL        MARQ                ; MESSAGE MARQUEE ROUTINE
            CALL        CRLF
;
EXEC:
            IF          TARBEL              ; NOT SURE OF THIS ROUTINE - CHECK IF USING TARBEL   ======================(d)IF
            CALL        CRLF                ; CRLF TO CONSOLE
            LD          SP,IY               ; LOAD SP FROM IY....IY CONTAINS THE SP ??
            SUB         A                   ; A=A=0
            OUT         (TARBL),A           ; OUTPUT TO TARBEL STATUS PORT MSB[X,X,TXRDY,RXRDY,  X,X,X,X]LSB
            ENDIF                           ;                                                    ======================(d)ENDIF
;
            LD          HL,MSG3             ; ROUTINE TO PRINT THE TWO LINES OF MENU COMMANDS
            CALL        MARQ                ; MESSAGE MARQUEE ROUTINE
            LD          HL,MSG4
            CALL        MARQ                ; MESSAGE MARQUEE ROUTINE
;
EXEC3:                                      ; DISPLAY MONITOR PROMPT
            LD          A,'#'               ; MONITOR PROMPT: # SIGN
            CALL        CRLF
            CALL        CONOUT
;                                             + + + MENU TABLE ENTRIES + + +
;
EXEC4:      CALL        CONIN               ; GET CONSOLE INPUT IN REGISTER 'A'
            CP          21H                 ;
            JP          M,EXEC4             ; LOOP ON CONTROL CHARACTERS ASCII(00H-20H)
            CP          'A'
            JP          Z,ALTER             ; MODIFY MEMORY ROUTINE = A
            CP          'D'
            JP          Z,DUMP              ; DUMP MEMORY ROUTINE = D
            CP          'G'
            JP          Z,GO                ; JUMP TO ADDRESS AND RUN = G
            CP          'K'
            JP          Z,KMENU             ; PRINT THE MENU CHOICES = K
            CP          'C'
            JP          Z,COPY              ; MOVE MEMORY ROUTINE = C
            CP          'T'
            JP          Z,MTEST             ; TEST MEMORY ROUTINE = T
            CP          'F'
            JP          Z,FILL              ; FILL MEMORY ROUTINE = F
            CP          'M'
            JP          Z,MEMMAP            ; MAP RAM AREAS = M
            CP          'L'
            JP          Z,MLOAD             ; WRITE DIRECT INTO MEMORY = L
            CP          'P'
            JP          Z,PORTS             ; DISPLAY AVAILABLE PORTS
            CP          'S'
            JP          Z,TSAVE             ; SAVE MEMORY ON CASSETTE = S
            CP          'R'
            JP          Z,TLOAD             ; LOAD MEMORY FROM CASSETTE = R
            CP          'V'
            JP          Z,VERIFY            ; VERIFY MEMORY BLOCK COPY/MOVE = V
            CP          'X'
            JP          Z,STRM              ; DO A SYNC STREAM OUTPUT = X
            CP          'Y'
            JP          Z,TUNE              ; ADJUST CASSETTE VOLUME ROUTINE = Y
            CP          'B'
            JP          Z,BOOT              ; TARBELL BOOT ROUTINE = B
            CP          'E'
            JP          Z,BIOS              ; VERSAFLOPPY II FLOPPY BIOS ROM = E
            CP          'U'
            JP          Z,QUERY             ; CHANGE PORT VALUE
;
            JP          EXEC4               ; IF INCORRECT OR NO SELECTION IS MADE, TRY AGAIN
;
BIOS:       LD          HL,MSG23            ; OUTPUT BRIEF INSTRUCTION
            CALL        MARQ                ; OUTPUT TO CONSOLE
            JP          0F000H              ; JUMP TO FLOPPY ROM AT F000H
;
GO:                                         ; JUMP TO A MEMORY LOCATION AND RUN A PROGRAM THERE
            LD          HL,MSG20            ; OUTPUT BRIEF INSTRUCTION
            CALL        MARQ                ; OUTPUT TO CONSOLE
            CALL        SPHIN
            CALL        CRLF
            JP          (HL)                ; EXECUTE A PROGRAM AT '(HL)' NO RETURN NEED RE-BOOT
;
ALTER:                                      ; MODIFY OR EXAMINE MEMORY ROUTINE
;                                             # A _ __ _    <enter>   # A 0100 <CR>
;                                             0100  00  _ _   <memory location 0100 displayed>
;                                             0100  00  FF.   <memory location 0100 changed to FF and PC INC>
;                                             0101  00  _ _   <memory location 0101 displayed>
;                                             0101  00  /     <memory unchanged exit routine>
;                                             0101  00 <CR>   <memory unchanged and PC DEC>
;                                             0101  00 <BS>   <memory unchanged and PC INC>
;                                             0100  FF  __ __
            LD          HL,MSG10            ; OUTPUT BRIEF INSTRUCTION
            CALL        MARQ                ; OUTPUT TO CONSOLE
            CALL        SPHIN
ALT1:       CALL        CRLF
            CALL        DHXOT
            CALL        SPACE
            LD          A,(HL)
            CALL        HEXOUT
            PUSH        HL
            CALL        SPHIN
            LD          E,L
            POP         HL
            CP          0DH                 ; ODH=CR MEANS DON'T CHANGE BUT DECREMENT TO THE NEXT LOCATION
            JP          Z,ALT3
            CP          '/'                 ; THE '/' IS THE EXIT CHARACTER W/O CHANGE
```

```
            JP          Z,EXEC3
            CP          '.'                 ; THE '.' IS THE MODIFY CHARACTER AS IN '0100 00 12.'
            JP          NZ,ALT2             ; IF THE '.' IS DETECTED
            LD          (HL),E              ; LOAD THE VALUE IN 'E' TO MEMORY LOCATION '(HL)'
ALT2:       INC         HL                  ; INCREMENT THE MEMORY LOCATION '(HL)'
            JR          ALT1                ; START AGAIN AT ALT1
ALT3:       DEC         HL                  ; DECREMENT THE MEMORY LOCATION AND START AGAIN AT ALT1
            JR          ALT1
;
DUMP:                                       ; DUMP MEMORY ROUTINE
;                                             # D _ _ _ _ , _ _ _ _    <enter>  # D 0100 <CR>
;                                             0100  D3                 <memory location 0100 displayed>
;                                             # D _ _ _ _ , _ _ _ _    <enter>  # D 0100,0110 <CR>
;                                             0100  D3 D3 D3 D3 D3... <memory locations displayed>
;                                             0110  D3 D3 D3 D3 D3... <memory locations displayed>
            LD          HL,MSG12
            CALL        MARQ
            CALL        DHXIN
DUMP1:      CALL        CRLF
            CALL        DHXOT
            LD          B,16
DUMP2:      CALL        SPACE
            LD          A,(HL)
            CALL        HEXOUT
            CALL        CMPDH
            JP          C,EXEC3             ; EXIT THE ROUTINE
            INC         HL                  ; INCREMENT THE PC 'HL'
            DEC         B
            JP          NZ,DUMP2
            JR          DUMP1
;
MEMMAP:                                     ; MEMORY MAP PROGRAM CF.DR.DOBBS VOL 31 P40 AND JOHN MONAHAN S-100.COM
            LD          HL,0                ; IT WILL SHOW ON CONSOL TOTAL MEMORY SUMMARY OF RAM, PROM, AND NO MEMORY
            LD          B,1
MAP1A:      LD          E,'R'               ; PRINT R FOR RAM
            LD          A,(HL)
            CPL
            LD          (HL),A
            CP          (HL)
            CPL
            LD          (HL),A
            JP          NZ,MAP2A
            CP          (HL)
            JP          Z,PRINTA
MAP2A:      LD          E,'p'
MAP3A:      LD          A,0FFH
            CP          (HL)
            JP          NZ,PRINTA
            INC         L
            XOR         A
            CP          L
            JP          NZ,MAP3A
            LD          E,'.'
PRINTA:     LD          L,0
            DEC         B
            JP          NZ,NLINEA
            LD          B,16
            CALL        CRLF
            CALL        HXOT4               ; 16 HEX CONSOL OUT ROUTINE
NLINEA:     LD          A,20H               ; LOAD REG A WITH 'SPACE' (20H)
            CALL        OTA                 ; SEND TO CONSOL WHAT'S IN REG A
            LD          A,E
            CALL        OTA                 ; SEND TO CONSOL WHAT'S IN REG A
            INC         H
            JP          NZ,MAP1A
            CALL        CRLF
            JP          EXEC3
;
MLOAD:                                      ; WRITE DIRECTLY INTO MEMORY ROUTINE:
;                                             THIS ROUTINE WILL ACCEPT A STARTING ADDRESS FOLLOWED BY HEX DATA THAT IS
;                                             PLACED CONSECUTIVELY INTO MEMORY.  ENTERING AN '/' WILL EXIT THE ROUTINE.
;                                             THIS COULD BE USED TO ENTER SHORT PROGRAMS INTO MEMORY TO BE RUN WITH THE
;                                             'GO' COMMAND.
;
;                                             ENTER HEX VALUES STARTING AT SPECIFIC MEMORY LOCATION
;                                             # L _ _ _ _     <enter>   # L 0100 <CR>
;                                             0100  _ _       <memory location 0100 displayed>
;                                             0100  AF <CR>   <memory location 0100 changed to AF and PC INC>
;                                             0101  _ _       <memory location 0101 displayed>
;                                             0101  _ _ /     <memory unchanged exit routine>
;
            LD          HL,MSG13
            CALL        MARQ
ML1:        CALL        SPHIN               ; GET A MEMORY ADDRESS TO PLACE THE CODE
            CALL        CRLF                ; GOTO A NEW LINE
ML2:        CALL        DHXOT               ; DISPLAY THE MEMORY ADDRESS
            CALL        SPACE               ; ADD A SPACE
            PUSH        HL                  ; REG PAIR 'HL' CONTAINS ADDRESS OF DATA LOCATION IS PUSHED ONTO STACK
            CALL        SPHIN               ; ENTER SOME HEX DATA, A <CR> OR A '/'
            LD          E,L
            POP         HL
            CP          '/'                 ; THE '/' IS THE EXIT CHARACTER W/O CHANGE
            JP          Z,ML4               ; EXIT TO EXEC
            CP          0DH                 ; <CR> = 0DH
            JP          NZ,ML3              ; IF THE '.' IS DETECTED
            LD          (HL),E              ; LOAD THE VALUE IN 'E' TO MEMORY LOCATION '(HL)'
ML3:        INC         HL                  ; INCREMENT THE MEMORY LOCATION '(HL)'
            CALL        CRLF
            JR          ML2                 ; START AGAIN AT ML2
ML4:        CALL        CRLF
            CALL        CRLF
            LD          A,'D'
            CALL        CONOUT
            LD          A,'O'
            CALL        CONOUT
            LD          A,'N'
            CALL        CONOUT
            LD          A,'E'
            CALL        CONOUT
            JP          EXEC3
;
KMENU:                                      ; ROUTINE TO CLR SCREEN & HOME POSITION
            CALL        CLRSCN              ; FILL SCREEN WITH BLANKS AND HOME CURSOR
            JP          INIT1               ; JUMP TO INIT1 - DISPLAY JADE SIGN-ON MESSAGE
;
;
FILL:                                       ; MEMORY FILL ROUTINE
;                                             # F _ _ _ _ , _ _ _ _ , _ _ <enter>
;                                             # F 0100,0150,DE <enter>
;                                             This will write "DE" into all memory locations
;                                             between 0100 - 0150
;                                             Only one byte "_ _" entered is valid as the fill
            LD          HL,MSG14
            CALL        MARQ
            CALL        DHXIN
            SUB         0DH
            JP          Z,FILL0
            PUSH        HL
            CALL        HEXIN
```

```
            LD          A,L
            POP         HL
FILL0:      DEC         HL
FILL1:      INC         HL
            LD          (HL),A
            CALL        CMPDH
            JP          NC,FILL1
            JP          EXEC3
;
COPY:                                   ; COPY MEMORY FROM ONE LOCATION TO ANOTHER LOCATION
;                                         # C _ _ _ _ , _ _ _ _ , _ _ _ _ <enter>
;                                         # C 0100,0150,0200 <enter>
;                                         Copies Memory Block between 0100 - 0150 to Memory
;                                         starting at 0200  This will over-write what is
;                                         at Memory Location 0200 leaving 0100-0150 unchanged
;
            LD          HL,MSG15
            CALL        MARQ
            CALL        TRPIN               ; TRPIN: INPUT 3 WORDS...PLACE IN [HL],[DE],[BC]...ex: (hhll),(ddee),(bbcc) <CR>
;                                             [HL]=END  [DE]=BEG  [BC]=DEST...WANT TO INC DE AND BC UNTIL DE=HL
            CALL        CRLF
COPY0:      LD          A,(DE)
            LD          (BC),A
            CALL        CMPDH               ; COMPARES [DE] TO [HL] IF EQUAL [CARRY FLAG] = 1
            JP          C,EXEC3             ; EXIT THE ROUTINE IF [CARRY FLAG] = 1
            INC         DE
            INC         BC
            JP          COPY0
;
;                                           MEMORY TEST ROUTINE: (64K OR LESS)
;                                           # T _ _ _ _ , _ _ _ _ <enter>
;                                           # T 0100,0200 <enter>
;                                           - WILL TEST MEMORY FROM 0100H TO 0200H
;                                           - WILL RETURN NOTHING IF MEMORY IS OK OR
;                                           - ADDRESS OF BAD MEMORY WITH BAD BITS
;
;                                           THIS IS A 'QUICKIE' MEMORY TEST TO SPOT
;                                           HARD MEMORY FAILURES, OR ACCIDENTLY
;                                           PROTECTED MEMORY LOCATIONS.  IT IS NOT
;                                           MEANT TO BE THE DEFINITIVE MEMORY DIAGNOSTIC.
;                                           IT IS, HOWEVER, NON-DESTRUCTIVE.  ERRORS ARE
;                                           PRINTED ON THE CONSOLE AS FOLLOWS-
;                                           <ADDR> 00000100 WHERE <1> IS THE BAD BIT.
;                                           BIT LOCATION OF THE FAILURE IS EASILY
;                                           DETERMINED.  NON-R/W MEMORY WILL RETURN
;                                           WITH- 11111111
;
MTEST:      LD          HL,MSG12
            CALL        MARQ                ; INPUT TWO HEX NUMBERS IN REGISTER PAIR 'HL' AND 'DE' (XXh XXh) AND (XXh XXh)
            LD          HL,0                ; INITIALIZE HL TO ZERO
            CALL        DHXIN               ; GET TWO PARAMETERS FROM THE CONSOL HL=(START ADDR) DE=(END ADDR)
            CALL        CRLF
MT1:        LD          A,(HL)              ; LOAD 'REG A' WITH BYTE (HL)
            LD          B,A                 ; STORE ORIGINAL BYTE (HL) TO 'REG B'
            CPL                             ; COMPLEMENT A...(INVERT A)...if 'REG A'=[11110001] => 'REG A'=[00001110]
            LD          (HL),A              ; LOAD (COMPLEMENT'd) A TO (HL)
            XOR         (HL)                ; XOR: (both bits same, then make '0')...IF NOT '0' THEN BITS ARE STUCK, BAD OR ROM
            JP          Z,MT2               ; IF JUMP TO MT2, NO BAD MEMORY FOUND
;                                             BAD MEMORY FOUND, DISPLAY IT ON CONSOL
            PUSH        DE                  ; SAVE END POINTER 'DE' ON STACK SO 'REG D' AND 'REG E' ARE AVAILABLE FOR USE
            LD          D,B                 ; LOAD 'REG D' WITH ORIGINAL BYTE (HL) THAT IS BAD
            LD          E,A                 ; LOAD 'REG E' WITH CMP AND XOR BYTE (HL) ???
            CALL        BAD                 ; PRINT 'BAD: ' TO THE CONSOL (AF AND HL ARE UNCHANGED IN ROUTINE 'BAD')
            LD          A,E                 ; MAKE SURE 'REG A' IS UNCHANGED AFTER MSG
            CALL        HLSP                ; PRINT 'HL' TO CONSOL FOLLOWED BY A SPACE
            LD          A,' '               ; SEND ' ' BLANK TO CONSOL OUT
            CALL        CONOUT
            LD          A,E
            CALL        BITS1               ; CONVERT HEX TO BINARY BITS AND DISPLAY
            CALL        CRLF
            LD          B,D
            POP         DE
MT2:        LD          (HL),B              ; RESTORE SAVED BYTE (HL)
                                            ; NOW SEE IF WE REACHED THE END
MND1:       LD          A,H                 ; COMPARE H[XXXX]L TO D[XXXX]E
            LD          C,D
            CP          C                   ; DOES H=D
            JP          Z,MND2              ; IF H=D GOTO MND2
            JR          INCR1               ; H<>D GOTO INCR1
MND2:       LD          A,L                 ; COMPARE H[XXXX]L TO D[XXXX]E
            LD          C,E                 ; DOES L=E
            CP          C                   ; IF L=E WE ARE ALL DONE SO GOTO MND3
            JP          Z,MND3              ; IF L<>E CONTINUE TO INCR1
;
INCR1:      INC         HL                  ; INC HL, THIS MOVES HL (START) CLOSER TO DE (END)
            JR          MT1                 ; NOT THERE YET SO GO BACK FOR MORE
;
MND3:       CALL        CRLF                ; REACHED THE END
            LD          HL,MSG19            ; LOAD MESSAGE "END "
            CALL        MARQ                ; DISPLAY MESSAGE
            JP          EXEC3
;
PORTS:                                  ; PORTS ROUTINE TO DISPLAY DETECTED PORTS - FROM JOHN MONAHAN S100.COM
;
            CALL        CRLF
            LD          B,0                 ; LOOP THROUGH ALL PORTS (0-FF)
            LD          D,6                 ;Display 6 ports across
            LD          E,0FFH              ;Will contain port number
;
LOOPIO:     LD          C,E                 ; LOAD 'REG C' WITH 0FFH
            LD          A,E                 ; LOAD 'REG A' WITH 0FFH
;
            IN          A,(C)               ; Remember  [ZASMB does not work with this opcode,SLR is OK]
            CP          0FFH                ;
            JR          Z,SKIP              ; IF DATA PORT C CONTAINS (FF)H, SKIP IT
            LD          H,A                 ; DATA PORT C HAS SOMETHING SO STORE IN 'H' FOR BELOW
            LD          A,E                 ; LOAD CURRENT PORT NUMBER TO 'REG A'
            CALL        LBYTE               ; PRINT PORT NUMBER
            LD          A,'-'
            CALL        CONOUT
            LD          A,'>'
            CALL        CONOUT
            LD          A,H                 ; GET PORT DATA STORED IN 'H'
            CALL        LBYTE               ; PRINT TO CONSOL
            LD          A,09H               ; 09H = TAB
            CALL        CONOUT
            DEC         D                   ; 6 PORTS PER LINE
            JR          NZ,SKIP
            LD          D,6
            CALL        CRLF
SKIP:       DEC         E                   ; NEXT PORT
            DJNZ        LOOPIO
            CALL        CRLF
            JP          EXEC3
;
                                            ; OUTPUT VALUE TO A PORT
QUERY:      LD          HL,MSG21            ; GET INPUT PROMPT "XXXX <CR> PORT VALUE" EX: PORT(10)=>(FF).."10FF <CR>" SET (PORT 10) TO (FF)
            CALL        MARQ                ; DISPLAY MESSAGE TO CONSOLE
```

```
            CALL        DHXIN                   ; GET TWO HEX VALUES [HL]; H=PORT,L=VALUE
            LD          C,H                     ; LOAD REG "C" WITH THE HARDWARE PORT
            LD          A,L                     ; LOAD REG "A" WITH THE NEW VALUE
            OUT         (C),A                   ; WRITE PORT (C) WITH VALUE "A"
;
            JP          EXEC3
;
;******************************** CASSETTE TAPE ROUTINES ******************************
;
;
            IF          KCTAPE                  ; KANSAS CITY TAPE            ================================================(a)IF
;
                                                ; ROUTINE TO LOAD TAPE DATA TO MEMORY
                                                ; PREPARE CASSETTE PLAYER, ENTER <LOAD ADDR START>, <LOAD ADDR END>,<CR>, START PLAYER,
                                                ; LEADER LOADS, DATA STARTS,"$" DISPLAYED, DATA ENDS,"*" DISPLAYED IF BAD CHKSUNM
TLOAD:      LD          HL,MSG20                ; LOAD KANSAS CITY TAPE DATA AT SPECIFIED MEMORY LOCATION
                                                ; GET STARTING MEMORY ADDRESS...XXYY <CR>  HL=(XX), DE=(YY)
            CALL        MARQ                    ; DISPLAY MESSAGE
            CALL        DHXIN                   ; GET TWO HEX VALUES FROM KEYBOARD
            CALL        TREAD                   ;
            JP          Z,EXEC3                 ; TAPE LOAD WAS SUCCESSFUL
            CALL        SPACE                   ; TAPE LOAD ERROR; SEND SPACE TO CONSOLE
            LD          A,'*'
            CALL        CONOUT                  ; SEND "*" TO CONSOLE INDICATING A CHKSUM ERROR
            JP          EXEC3                   ;
                                                ;
                                                ; TAPE FORMAT: [FF][FF][FF][FF][E6][DD][DD][DD][DD][CHKSUM]
TREAD:      LD          A,0B0H                  ; SET UP 2S1P AY51013 UART ON CASSETTE PORT A
            OUT         (TAPST),A               ; WRITE B0H TO "CONTROL WORD PORT" MSB(NP,TSB,NB2,NB1,EPS,X,X,X)LSB
                                                ; B0H=(1,0,1,1, 0,0,0,0)=(NO PARITY,1 STOP BIT,8 DATA BITS,ODD PARITY)
            LD          HL,TRUE                 ; HL IS A DELAY SEED; TRUE = 0FFFFH
            CALL        DELAY
TRDA:       LD          B,4                     ; SET COUNTER B=4; LATER B BECOMES CHKSUM
TRDB:       CALL        CIN                     ; GET CASSETTE DATA
            CP          0FFH                    ; IF DATA IS NOT "FFH" THEN LEADER HASN'T STARTED YET NZ=FALSE
            JR          NZ,TRDA                 ; IF DATA IS "FFH", FOUND LEADER SO KEEP READING CASSETTE DATA, IF "E6H"
            DEC         B                       ; DECREMENT THE COUNTER B
            JR          NZ,TRDB                 ; GET NEXT (4) BYTES FROM CASSETTE [FF][FF][FF][FF]
TRDC:       CALL        CIN                     ; READ DATA FROM THE TAPE
            CP          0FFH                    ; IS DATA LEADER "FFH"
            JR          Z,TRDA                  ; IF IT IS LEADER; START ALL OVER READING THE NEXT (4) BYTES
            CP          0E6H                    ; IS BYTE "E6H"...START OF DATA WITH "SYNC BYTE"
            JR          NZ,TRDA                 ; IF IT IS NOT "E6H" SOMETHING IS WRONG SO START AGAIN
            LD          B,0                     ; FOUND THE "SYNC BYTE" (E6); SO SET CHECKSUM=B=0
            LD          A,'$'
            CALL        CONOUT                  ; SEND "$" TO CONSOLE INDICATING START OF MEMORY LOAD FROM TAPE
            DEC         HL
TRD1:       INC         HL                      ; INCREMENT LOAD ADDRESS HL
            CALL        CIN                     ; LOAD TAPE DATA IN "A"
            LD          (HL),A                  ; WRITE "A" TO MEMORY LOCATION (HL)
            ADD         A,B                     ; ADD CHKSUM TO A
            CALL        CMPDH                   ; COMPARE "D" TO "H"...START=END NC=FALSE
            JR          NC,TRD1
            CALL        CIN                     ; LOAD TAPE DATA [CHKSUM]
            CP          B                       ; IF TAPE CHKSUM = B; THEN Z=TRUE
            RET
;
CIN:        IN          A,(TAPST)               ; READ "STATUS SENSE PORT" FOR 2S1P AY51013 UART ON CASSETTE PORT B
            AND         10H                     ; MSB(TBMT,PE,FE,DAV,X,X,X,X)LSB 10H=(0,0,0,1, 0,0,0,0)=> DAV=0 (RX IS RDY)
            JR          Z,CIN
            IN          A,(TAPE)                ; READ DATA INTO "DATA WORD PORT"
            RET
                                                ; ROUTINE TO ALLOW FOR ADJUSTMENT OF TAPE PLAYER VOLUME CONTROL-NEED SYNC STREAM TAPE AS INPUT
TUNE:       LD          A,0B0H                  ; SET UP 2S1P AY51013 UART ON CASSETTE PORT B
            OUT         (TAPST),A               ; WRITE B0H TO "CONTROL WORD PORT" MSB(NP,TSB,NB2,NB1,EPS,X,X,X)LSB
            CALL        CLRSCN                  ; B0H=(1,0,1,1, 0,0,0,0)=(NO PARITY,1 STOP BIT,8 DATA BITS,ODD PARITY)
TUN0:       LD          B,30                    ; B IS A LINE COUNTER TO PRINT 30 LINES TO CONSOLE
            CALL        HOME                    ; MOVE CURSOR TO UPPER TOP LEFT OF SCREEN
            LD          HL,MSG16                ;
            CALL        MARQ                    ; LOAD A "SYNC STREAM TAPE" AND OBSERVE DISPLAY FOR "$" OR "+"
            CALL        SPACE                   ; ADJUST THE TAPE PLAYER VOLUME TO ONLY DISPLAY "+"
TUN1:       CALL        CRLF                    ; SEN CRLF TO CONSOLE
            LD          H,40                    ; LOAD COUNTER TO 40; PRINTS 40 "+" OR "$" TO CONSOLE = 1 LINE
TUN2:       CALL        CIN                     ; INPUT TAPE DATA
            CP          0FFH                    ; IS DATA [FF]; IF IT IS, Z=TRUE
            JR          Z,TUN2                  ; DATA IS LEADER, TRY AGAIN...OHERWISE CONTINUE BELOW
            LD          L,'+'                   ; L="+"
            CP          0E6H                    ; IS DATA [E6] THE SYNC BYTE...IF SO, Z=TRUE
            JR          Z,TUN3                  ; IF FOUND [E6] GOTO TUN3...(L="+" GOOD/L="$"BAD)
            LD          L,'$'                   ; L="$"
TUN3:       LD          A,L
            CALL        CONOUT                  ; SEND L TO CONSOLE
            DEC         H                       ; DECREMENT COUNTER H
            JR          NZ,TUN2                 ; IF H=0, START ALL OVER AGAIN
            DEC         B
            JR          NZ,TUN1
            JP          TUN0
;
                                                ; TSAVE ROUTINE-USED TO SAVE A BLOCK OF MEMORY TO CASSETTE TAPE
TSAVE:      LD          HL,MSG12                ; CONSOLE PROMPT FOR BEGINNING AND ENDING MEMORY ADDRESS
                                                ; TSAVE XXXX,YYYY <CR> STARTS THE TAPE SAVE
            CALL        MARQ                    ; FORMAT: [FF][FF]--16--[FF][E6][DD][DD][DD][DD][CHKSUM][CHKSUM][CHKSUM]
            CALL        DHXIN                   ; GET STARTING MEMORY ADDRESS...XXYY <CR>  HL=(XX), DE=(YY)
            CALL        TWRIT
            JP          EXEC3
TWRIT:      LD          A,0B0H                  ; SET UP 2S1P AY51013 UART ON CASSETTE PORT A
            OUT         (TAPST),A               ; WRITE B0H TO "CONTROL WORD PORT" MSB(NP,TSB,NB2,NB1,EPS,X,X,X)LSB
                                                ; B0H=(1,0,1,1, 0,0,0,0)=(NO PARITY,1 STOP BIT,8 DATA BITS,ODD PARITY)
            LD          B,16                    ; COUNTER B=16
TWRT0:      LD          A,0FFH                  ; LOAD A=[FF]
            CALL        COUT                    ; WRITE [FF] TO TAPE
            DEC         B                       ; DECREMENT COUNTER B
            JR          NZ,TWRT0                ; KEEP GOING UNTIL 16 [FF]'S HAVE BEEN WRITTEN TO TAPE
            LD          A,0E6H                  ; LOAD A WITH [E6] THE SYNC BYTE
            CALL        COUT                    ; WRITE SYNC BYTE TO TAPE
            DEC         HL
            LD          B,0                     ; LOAD CHKSUM B=0
TWRT1:      INC         HL                      ; INCREMENT HL
            LD          A,(HL)                  ; LOAD DATA FROM (HL) TO A
            CALL        COUT                    ; WRITE DATA TO TAPE
            ADD         A,B                     ; ADD CHKSUM TO A
            LD          B,A                     ; SAVE SUM BACK TO CHKSUM
            CALL        CMPDH                   ; COMPARE D TO H
            JR          NZ,TWRT1                ; KEEP WRITING DATA UNTIL NZ=FALSE
            LD          A,B                     ; LOAD THE CHKSUM TO A
            CALL        COUT                    ; WRITE CHECKSUM ONCE
            CALL        COUT                    ; WRITE CHCKSUM TWICE
            JR          COUT                    ; WRITE CHKSUM A THIRD TIME
COUT:       PUSH        AF                      ; PUSH DATA TO BE WRITTEN ONTO AF
            IN          A,(TAPST)               ; READ STATUS PORT MSB(TBMT,PE,FE,DAV,XX,XX,XX,XX)LSB
            AND         80H                     ; 80H=[1,0,0,0, 0,0,0,0]  IF TBMT=1, TRANSMIT BUFFER NOT RDY
            JR          Z,COUT+1                ; IF TBMT <> 0 KEEP CHECKING
            POP         AF                      ; POP DATA TO BE WRITTEN TO TAPE
            OUT         (TAPE),A                ; WRITE DATA TO TAPE
            RET
;
STRM:       LD          HL,MSG17                ; STREAM ROUTINE-GENERATES A SERIES OF "FFH" AND "E6H"...[FF] [E6] [FF] [E6] [FF]
            CALL        MARQ                    ; START THE ROUTINE AND RECORD THE OUTPUT TO CASSETTE TAPE
                                                ; THIS WILL CREATE A "SYNC STREAM TAPE" USED TO CALIBRATE THE TAPE VOLUME
            LD          A,0B0H                  ; SET UP 2S1P AY51013 UART ON CASSETTE PORT A
```

```
            OUT         (TAPST),A           ; WRITE B0H TO "CONTROL WORD PORT" MSB(NP,TSB,NB2,NB1,EPS,X,X,X)LSB
STRM1:      LD          A,0FFH              ; B0H=(1,0,1,1, 0,0,0,0)=(NO PARITY,1 STOP BIT,8 DATA BITS,ODD PARITY)
            CALL        COUT                ; WRITE [FF] TO TAPE
            LD          A,0E6H
            CALL        COUT                ; WRITE [E6] TO TAPE
            JR          STRM1
;
            ENDIF                           ; KANSAS CITY TAPE          ================================================(a)ENDIF
;
            IF          TARBEL              ; TARBELL TAPE              ================================================(b)IF
;
                                            ; TAPE FORMAT: [FF][FF][FF][3C][E6][DD][DD][DD][DD][1A][FF][CHKSUM][FF][FF][FF][FF]
                                            ; [FF][FF][FF]= TAPE LEADER; [3C]=START BYTE; [E6]=SYNC BYTE; [DD]=DATA BYTES;
                                            ; [1A]+[FF]=STOP BYTES; [CHKSUM]=CHECKSUM...SIMPLE SUM OF DATA BYTES
                                            ; END OF RECORD- [1A][FF][CHKSUM][FF][FF][FF][FF]
                                            ; NOTE: ROUTINES ARE FROM THE DON TARBELL MANUAL WHERE POSSIBLE AS THEY SEEM MUCH BETTER
                                            ; THAN THE JADE TARBEL ROUTINES...REFERENCE "WRITING PROGRAMS FOR THE CASSETTE INTERFACE",
                                            ; "CASSETTE INTERFACE INPUT ROUTINE", "CASSETTE INTERFACE OUTPUT ROUTINE"
;
                                            ; LOAD MEMORY FROM TAPE ROUTINE-CASSETTE INTERFACE INPUT ROUTINE
                                            ; GET STARTING MEMORY ADDRESS...XXYY <CR>  HL=(XX), DE=(YY)
TLOAD:      LD          HL,MSG20            ; LOAD MESSAGE PROMPT
            CALL        MARQ                ; DISPLAY MESSAGE
            CALL        DHXIN               ; GET STARTING MEMORY ADDRESS...XXYY <CR>  HL=(XX), DE=(YY)
            DEC         HL                  ;
            INC         HL                  ;
            CALL        TREAD               ; TARBELL OUTPUT PORT J1 BIT D0=CASSETTE MOTOR CONTROL ON/OFF
            JP          Z,EXEC3             ; IF THE CHKSUM MATCHED (FLAG Z IS TRUE), THEN EXIT
            CALL        CRLF                ; IF THE CHECKSUM DID NOT MATCH...
            LD          A,C                 ; LOAD TAPE CHKSUM
            CALL        CONOUT              ;
            CALL        SPACE               ;
            LD          A,B                 ; LOAD THE CALCULATED CHKSUM
            CALL        CONOUT              ;
            CALL        SPACE               ;
            LD          HL,MSG11            ; "BAD" MESSAGE
            CALL        MARQ                ;
            JP          EXEC3               ;
;
TREAD:      LD          A,11H               ; LOAD 11H TO "A" BECAUSE WE WANT TO RESET THE INTERFACE & START THE CASSETTE MOTOR
            OUT         (TARBL),A           ; WRITE 11H TO THE STATUS PORT (6EH) MSB[X,X,TXRDY,RXRDY, 0,0,0,D0]LSB
                                            ; MSB[0,0,0,1, 0,0,0,1]LSB = 11H
            LD          B,0                 ; B=CHKSUM BYTE, SET TO "00"
            LD          C,0                 ; REG "C" IS USED AS PATTERN IDENTIFER=(1)[FF];(2)[3C];(3)[E6];(4)[1A];(5)[FF] following [1A]
TRD0:       LD          A,C                 ;
            CP          3                   ;
            JR          Z,TRD1              ; IF Z=TRUE; [FF][3C][E6] PATTERN FOUND SO GO GET DATA [DD]
            CALL        CIN                 ; GET CASSETTE DATA IN REG "A"
            CP          0FFH                ; LEADER [FF]
            JR          Z,IDB1              ;
            CP          03CH                ; START BYTE [3C]
            JR          Z,IDB2              ;
            CP          0E6H                ; SYNC BYTE [E6]
            JR          Z,IDB3              ;
            LD          C,0                 ; RESET "C" TO 0
            JR          TRD0                ; TRY AGAIN TO FIND PATTERN
;
IDB1:       LD          C,1                 ;
            JP          TRD0                ;
IDB2:       LD          C,2                 ;
            JP          TRD0                ;
IDB3:       LD          C,3                 ;
            JP          TRD0                ;
;
                                            ; TO GET HERE, THE PATTERN [FF][3C][E6] WAS FOUND..."C"=3
TRD1:       LD          C,0                 ; RESET "C" TO 0
            CALL        CIN                 ;
            PUSH        HL                  ; PUSH [DATA] ONTO HL
            CP          01AH                ; FOUND [1A], COULD BE A STOP BYTE...
            JR          Z,IDB4              ;
            CP          0FFH                ; FOUND [FF], COULD BE "END OF RECORD"...
            JR          Z,IDB5              ;
            JP          TRD2                ;
;
IDB4:       LD          C,4                 ; [1A] FLAG ?
                                            ; [DATA]=[DATA1] AND IS IN HL
            JP          (TRD1+1)            ; GET NEXT DATA BYTE
;
IDB5:       LD          A,C                 ; [1A][FF] FLAG ?
            CP          4                   ;
            JP          NZ,TRD2             ;
            LD          C,5                 ; IF C=4; THEN [1A][FF] IS TRUE AND C=5
;
TRD2:       LD          A,C                 ; GET C
            CP          4                   ; IF C=4; [1A] FLAG SET
            JR          Z,TRD4              ;
            CP          5                   ; IF C=5; [1A][FF] FLAG SET
            JR          Z,TRD5              ;
;
TRD3:       POP         HL                  ; POP [DATA] FROM HL INTO "A"
            CALL        DATIN               ; [DATA] INSERTED TO MEMORY (HL); HL INCREMENTED; CHKSUM UPDATED
            JP          TRD1                ; GO GET MORE DATA
;
TRD4:       POP         HL                  ; TO GET HERE, END NOT FOUND AND [DATA1]=HL, [DATA2]=HL
            LD          C,A                 ; "C" IS REUSED AS VARIABLE STORAGE; [DATA2] STORED IN "C"
            POP         HL                  ; POP [DATA1] FROM HL INTO "A"
            CALL        DATIN               ; [DATA1] INSERTED TO MEMORY (HL); HL INCREMENTED; CHKSUM UPDATED
            LD          A,C                 ; RESTORE [DATA2] TO A
            CALL        DATIN               ; [DATA2] INSERTED TO MEMORY (HL); HL INCREMENTED; CHKSUM UPDATED
            JP          TRD1                ; GO GET MORE DATA
;
TRD5:       POP         HL                  ; TO GET HERE, "END OF RECORD" FOUND AND [DATA1]=1A, [DATA2]=FF
            POP         HL                  ; STOP BYTE AND FOLLOWING FF ARE DISCARDED
            CALL        CIN                 ; GET TAPE [CHKSUM]
            LD          C,A                 ; REGISTER "C" IS REUSED; STORE TAPE [CHKSUM] = C
            LD          A,B                 ; GET CAKCULATED [CHKSUM] = A
            CP          C                   ; COMPARE TAPE TO CALCULATED CHKSUM
            JR          Z,TRD6              ; CHKSUM MATCHED...SHUTDOWN AND RETURN
            RET                             ; CHKSUM DID NOT MATCH, RETURN WITH CP=NZ
;
TRD6:       LD          A,10H               ; LOAD 10H TO "A" BECAUSE WE WANT TO RESET THE INTERFACE & STOP THE CASSETTE MOTOR
            OUT         (TARBL),A           ; WRITE 10H TO THE STATUS PORT (6EH) MSB[X,X,TXRDY,RXRDY, 0,0,0,D0]LSB
            CP          A,10H               ; SET Z FLAG
            RET
;
CIN:        IN          A,(TARBL)           ; READ STATUS PORT (6EH) MSB[0,0,0,1, 0,0,0,1]LSB RDY=MSB[0,0,0,0, 0,0,0,1]LSB =01H
            AND         01H                 ; 01H=[0,0,0,0, 0,0,0,1]...IF RXRDY=0 AND D0=0 CASSETTE MTR=ON, RXRDY=0=RDY TO RECEIVE
            JR          NZ,CIN              ; LOOP WAITING FOR RXRDY=0
            IN          A,(TARBL+1)         ; LOAD DATA WORD FROM DATA PORT (6FH)
            RET
;
DATIN:      INC         HL                  ; LOAD DATA INTO MEMORY, CHKSUM, AND INCREMENT HL
            LD          (HL),A              ;
            ADD         A,B                 ;
            LD          B,A                 ;
            RET
                                            ; ROUTINE TO ALLOW FOR ADJUSTMENT OF TAPE PLAYER VOLUME CONTROL-NEED SYNC STREAM TAPE AS INPUT
TUNE:       LD          A,11H               ; LOAD 11H TO "A" BECAUSE WE WANT TO RESET THE INTERFACE & START THE CASSETTE MOTOR
            OUT         (TARBL),A           ; WRITE 11H TO THE STATUS PORT (6EH) MSB[X,X,TXRDY,RXRDY, 0,0,0,D0]LSB
```

```
                                        ; MSB[0,0,0,1, 0,0,0,1]LSB = 11H
            CALL        CLRSCN          ; CLEAR THE CONSOLE
TUN0:       LD          B,30            ; B IS A LINE COUNTER TO PRINT 30 LINES TO THE CONSOLE
            CALL        HOME            ; MOVE CURSOR TO UPPER TOP LEFT OF SCREEN
            LD          HL,MSG16        ; B IS A LINE COUNTER TO PRINT 30 LINES TO CONSOLE
            CALL        MARQ            ; LOAD A "SYNC STREAM TAPE" AND OBSERVE DISPLAY FOR "$" OR "+"
            CALL        SPACE           ; ADJUST THE TAPE PLAYER VOLUME TO ONLY DISPLAY "+"
TUN1:       CALL        CRLF            ; SEND CRLF TO CONSOLE
            LD          H,40            ; LOAD COUNTER TO 40; PRINTS 40 "+" OR "$" TO CONSOLE = 1 LINE
TUN2:       CALL        CIN             ; INPUT TAPE DATA
            CP          0FFH            ; IS DATA [FF]; IF IT IS, Z=TRUE
            JR          Z,TUN2          ; DATA IS LEADER, TRY AGAIN...OHERWISE CONTINUE BELOW
            LD          L,'+'           ; L="+"
            CP          0E6H            ; IS DATA [E6] THE SYNC BYTE...IF SO, Z=TRUE
            JR          Z,TUN3          ; IF FOUND [E6] GOTO TUN3...(L="+" GOOD/L="$"BAD)
            LD          L,'$'           ; L="$"
TUN3:       LD          A,L
            CALL        CONOUT          ; SEND L TO CONSOLE
            DEC         H               ; DECREMENT COUNTER H
            JR          NZ,TUN2         ; IF H=0, START ALL OVER AGAIN
            DEC         B
            JR          NZ,TUN1
            JP          TUN0
;
                                        ; ROUTINE TO SAVE A BLOCK OF MEMORY TO THE TARBELL TAPE
TSAVE:      LD          HL,MSG12        ; FORMAT: [FF][FF][FF][3C][E6][DD][DD][DD][DD][1A][FF][CHKSUM][FF][FF][FF]
            CALL        MARQ            ; MESSAGE PROMPT FOR START/END MEMORY ADDRESS INPUT
            CALL        DHXIN           ; GET STARTING MEMORY ADDRESS...XXXX,YYYY <CR>  HL=(START ADDR), DE=(END ADDR)
            CALL        TWRIT           ; THE "TAPE WRITE" ROUTINE
            JP          EXEC3
;
TWRIT:      LD          A,21H           ; MSB[X,X,TXRDY,RXRDY,  X,X,X,D0]LSB 21H=[X,X,1,0, 0,0,0,1]...SET TXRDY=1 AND D0=1...MTR ON
            OUT         (TARBL),A       ; WRITE TO STATUS PORT (6EH), THIS SHOULD RESET THE TARBELL TXRDY TO NOT RDY AND START THE MTR
            PUSH        HL              ; STORE HL ON THE STACK (START ADDR)
            LD          HL,0FFFFH       ; LOAD DELAY SEED
            CALL        DELAY           ; WAIT A BIT, THIS ASSUMES WITH NO DIRECT OUTPUT, THE TARBELL WRITES [FF][FF][FF] TO TAPE AS
THE LEADER
            POP         HL              ; LEADER IS FINISHED, RECOVER HL
            SUB         A               ; AFTER "DELAY", REG "A" SHOULD CONTAIN (00)..."A" SUB "A" = 0
            LD          B,A             ; LOAD THE CHCKSUM "B" WITH (00)
            LD          A,03CH          ; LOAD "A" WITH THE START BYTE (3CH)
            CALL        COUT            ; WRITE [3C] TO THE TAPE
            LD          A,0E6H          ; LOAD "A" WITH THE SYNC BYTE (E6H)
            CALL        COUT            ; WRITE [E6] TO THE TAPE
            DEC         HL
TWRT1:      INC         HL              ; DECREMENT THE STARTING BLOCK OF MEMORY ADDRESS STORED IN HL
            LD          A,(HL)          ; LOAD REG "A" WITH THE BYTE OF MEMORY IN LOCATION (HL)
            CALL        COUT            ; WRITE [(HL)] TO TAPE
            ADD         A,B             ; ADD "A" + CHKSUM
            LD          B,A             ; STORE RESULT TO CHKSUM
            CALL        CMPDH           ; COMPARE START ADDRESS TO END ADDRESS
            JR          NC,TWRT1        ; IF NC=TRUE, KEEP SENDING MEMORY BYTES TO TAPE
            LD          A,01AH          ; LOAD "A" WITH FIRST PART OF STOP BYTE (1AH)
            CALL        COUT            ; WRITE [1A] TO TAPE
            LD          A,0FFH          ; LOAD "A" WITH SECOND PART OF STOP BYTE (FFH)
            CALL        COUT            ; WRITE [FF] TO TAPE
            LD          A,B             ; LOAD REG "A" WITH THE CHECKSUM VALUE
            CALL        COUT            ; WRITE [CHKSUM] TO TAPE
            LD          A,21H           ; AGAIN SET TXRDY=1 AND D0=1
            OUT         (TARBL),A       ; WRITE TO STATUS PORT (6EH), THIS SHOULD RESET THE TARBELL TXRDY TO NOT RDY AND KEEP MTR
RUNNING
            LD          HL,07FFFH   ; LOAD DELAY SEED...SHORTER THAN STARTING LEADER
            CALL        DELAY           ; WAIT A BIT, THIS ASSUMES WITH NO DIRECT OUTPUT, THE TARBELL WRITES [FF][FF][FF] TO TAPE AS
THE LEADER
            LD          A,00H           ; MSB[X,X,TXRDY,RXRDY,  X,X,X,D0]LSB 00H=[X,X,0,0, 0,0,0,0]...SET TXRDY=0 AND D0=0...MTR OFF
            OUT         (TARBL),A       ; WRITE [0,0,0,0, 0,0,0,0] TO STATUS PORT; (TURN OFF CASSETTE MTR) AND SET RXRDY=TXRDY=0
            CALL        CRLF
            LD          HL,MSG19        ; LOAD MESSAGE "END "
            CALL        MARQ            ; DISPLAY MESSAGE
            RET
;
COUT:       PUSH        AF              ; STORE BYTE TO BE WRITTEN TO TAPE IN AF
            IN          A,(TARBL)       ; READ STATUS PORT (6EH)
            AND         20H             ; MSB[X,X,TXRDY,RXRDY,  X,X,X,X]LSB 20H=[X,X,1,0, 0,0,0,0]...IF TXRDY=1 NOT RDY TO TRANSMIT
            JR          NZ,COUT+1
            POP         AF              ; TARBEL RDY TO TRANSMIT; POP DATA BYTE FROM AF
            OUT         (TARBL+1),A     ; WRITE [DATA] TO TAPE
            RET
;
STRM:       LD          HL,MSG17        ; STREAM ROUTINE-GENERATES A SERIES OF "FFH" & "E6H"...[FF][FF][FF][E6][E6][E6][E6][E6][E6]...
            CALL        MARQ            ; START THE ROUTINE AND RECORD THE OUTPUT TO CASSETTE TAPE
            LD          B,1EH           ; THIS WILL CREATE A "SYNC STREAM TAPE" USED TO CALIBRATE THE TAPE VOLUME
            LD          A,11H           ; MSB[X,X,TXRDY,RXRDY,  X,X,X,X]LSB 11H=[X,X,1,0, 0,0,0,1]...IF TXRDY=1 NOT RDY TO TRANSMIT
            OUT         (TARBL),A       ; SEND TO STATUS PORT (6EH) TO RESET TARBELL AND START TAPE RECORDER MOTOR
STRM1:      LD          A,0FFH          ;
            CALL        COUT            ; WRITE [FF] TO TAPE
            DEC         B               ; DECREMENT COUNTER B
            JR          NZ,STRM1        ; IF NOT ZERO, CONTINUE WRITING LEADER
STRM2:      LD          A,0E6H
            CALL        COUT            ; WRITE [E6] TO TAPE
            JR          STRM2           ; CONTINUE UNTIL TAPE RECORDER IS SHUT OFF
;
            ENDIF                       ; TARBELL TAPE                      ============================================(b)ENDIF
;
;********************************* SUBROUTINES BELOW ***************************************
;
VERIFY:     LD          HL,MSG18        ; VERIFY ORIGINAL MEMORY BLOCK TO COPIED/MOVED MEMORY BLOCK
            CALL        MARQ            ; IF MISMATCH, PRINT...<ORIG ADDR> <ORIG DATA> <COPY/MOVE DATA> TO CONSOLE
            CALL        TRPIN           ; CALL TRIPLE INPUT...X,Y,Z WHERE X=START ORG(HL), Y=END ORG(DE), Z=START MOVE/COPY BLOCK(BC)
            EX          DE,HL
            DEC         HL
            DEC         BC
VRFY1:      INC         HL              ; INCREMENT ORIGINAL STARTING ADDRESS HL
            INC         BC              ; INCREMENT COPIED/MOVED STARTING ADDRESS BC
            LD          A,(BC)          ; LOAD "A" WITH COPY MEMORY DATA (BC)
            CP          (HL)            ; CP (BC) TO (HL); IF (BC)=(HL) Z=TRUE
            JR          Z,VRFY2         ; (BC) AND (HL) MATCH, ALL IS GOOD SO CONTINUE CHECKING
            CALL        CRLF            ; (BC) <> (HL), ERROR EXISTS; PRINT CRLF TO CONSOLE
            CALL        DHXOT           ; SEND CURRENT HL VALUE TO CONSOLE...ORIG MEMORY ADDRESS
            CALL        SPACE           ; SEND SPACE TO CONSOLE
            LD          A,(HL)          ; LOAD "A" WITH (HL) LOCATION CONTENTS
            CALL        HEXOUT          ; SEND HEX VALUE (HL) TO CONSOLE
            CALL        SPACE           ; SEND SPACE TO CONSOLE
            LD          A,(BC)          ; LOAD "A" WITH (BC) LOCATION CONTENTS
            CALL        HEXOUT          ; SEND HEX VALUE (BC) TO CONSOLE
VRFY2:      CALL        CMPDH           ; COMPARE START HL TO END ADDRESS DE
            JR          NC,VRFY1        ; IF START <> END KEEP CHECKING
            JP          EXEC3           ; ALL DONE
;
                                        ; DELAY USES SEED STORED IN HL AS A BASIS FOR TIME DELAY
DELAY:      EX          (SP),HL         ; WASTE TIME FOR DELAY
            EX          (SP),HL         ; WASTE TIME FOR DELAY
            DEC         HL              ; DECREMENT SEED IN HL....example 2000O=HL=[4E][20]...DEC HL=[4E][1F]
            LD          A,L             ; REGISTER "L" AS IN HL...example L=[1F]
            OR          H               ; REGISTER "H" AS IN HL...example H=[4E]
            JR          NZ,DELAY        ; KEEP DECREMENTING UNTIL HL=[00][00]
            RET
;
CMPDH:      PUSH        AF              ; COMPARE 'D' TO 'H'...H[XX XX]L AND D[XX XX]E
```

```
            LD          A,D
            CP          H
            JP          NZ,CMP1                 ; IF D <> H THEN GOTO CMP1, [CLEAR CARRY] AND RETURN
            LD          A,E
            CP          L
            JP          NZ,CMP1                 ; IF E <> H THEN GOTO CMP1, [CLEAR CARRY] AND RETURN
            POP         AF
            SCF                                 ; H=D AND L=E [SET CARRY] AND RETURN
            RET
CMP1:       POP         AF                      ; POP WHAT WAS SAVED IN AF
            SCF                                 ; SET CARRY FLAG
            CCF                                 ; CLEAR CARRY FLAG
            RET
;
DHXIN:      CALL        SPHIN                   ; OUTPUT A SPACE AND GET SOME CONSOL INPUT
            PUSH        HL                      ; PUSH 'HL' ONTO THE STACK
            CP          0DH                     ; COMPARE...subtract <REG A> - <0D> = NZ...IS IT '0DH' A <CARRIAGE RTN>
            CALL        NZ,HEXIN                ; IF 'NOT ZERO' ... NOT <CR>... GO GET MORE
            EX          DE,HL                   ; <CR> DETECTED SO EXCHANGE THE 'DE' AND 'HL' REGISTERS
            POP         HL
            RET
;
;                                               THIS IS THE MAIN "PARAMETER-GETTING" ROUTINE.
;                                               THIS ROUTINE WILL ABORT ON A NON-HEX CHARACTER.
;                                               IT TAKES THE MOST RECENTELY TYPED FOUR VALID
;                                               HEX CHARACTERS, AND PLACES THEM UP ON THE STACK.
;                                               (AS ONE 16 BIT VALUE, CONTAINED IN TWO
;                                               8-BIT BYTES.)  IF A CARRIAGE RETURN IS ENTERED,
;                                               IT WILL PLACE THE VALUE OF "0000" IN THE STACK.
;
SPHIN:      CALL        SPACE                   ; LOAD 'A' WITH 20H 'SPACE' AND SEND TO CONSOLE OUTPUT
            CALL        SPACE
            CALL        SPACE
HEXIN:      LD          HL,0                    ; LOAD HL WITH '0'...THIS ROUTINE IS FOR HEX INPUT, SO REJECTS ANYTHING ELSE
HXIN1:      CALL        CONIN                   ; GET CONSOL INPUT AND RETURN IN 'REG A' CAN BE '00H TO 7FH'
            CP          '0'                     ; COMPARE...subtract <REG A> - <0> = M...IF NOT '0' OR LARGER RETURN
            RET         M                       ; IF 'SIGN NEG' RETURN
            CP          'F'+1                   ; COMPARE...subtract <REG A> - <F'+1> = P...IF IT IS LARGER THAN 'F' IE NOT HEX RETURN
            RET         P                       ; IF 'SIGN POS'
            CP          '9'+1                   ; COMPARE...subtract <REG A> - <9 + 1> = M...IF IT IS LARGER THAN '9'
            JP          M,HXIN2                 ; IF 'SIGN NEG' GOTO HXIN2
            CP          'A'                     ; COMPARE...subtract<REG A> - <A> = M...IF IT IS NOT 'A' THEN RETURN
            RET         M                       ; IF 'SIGN NEG'
            ADC         A,9                     ; ADD "A" + 9 + "CARRY FLAG 0 OR 1"
HXIN2:      AND         0FH
            ADD         HL,HL                   ; MULTIPLY BY 16
            ADD         HL,HL
            ADD         HL,HL
            ADD         HL,HL
            OR          L                       ; 'OR' IN THE SINGLE NIBBLE
            LD          L,A
            JR          HXIN1                   ; GET SOME MORE FROM CONSOL IN
;
TRPIN:      CALL        SPHIN                   ; TRPIN: INPUT 3 BYTES...PLACE IN [HL],[DE],[BC]...ex: (hhll),(ddee),(bbcc) <CR>
            EX          DE,HL                   ; EXCHANGE THE DE AND HL REGISTERS - LOAD ALL THREE PAIRS WITH HEX
            CALL        HEXIN                   ; ENTRIES
            PUSH        HL
            CALL        HEXIN
            PUSH        HL
            POP         BC
            POP         HL
            RET
;
DHXOT:      LD          A,H                     ; DISPLAY CURRENT HL VALUE
            CALL        HEXOUT
            LD          A,L
;
HEXOUT:     PUSH        AF
            RRCA
            RRCA
            RRCA
            RRCA
            CALL        HXOT1
            POP         AF
HXOT1:      AND         0FH
            ADD         A,30H
            CP          '9'+1
            JP          M,CONOUT
            ADD         A,7
            JP          CONOUT
;
CRLF:       PUSH        AF
            LD          A,0DH
            CALL        CONOUT
            LD          A,0AH
            CALL        CONOUT
            POP         AF
            RET
;
SPACE:      PUSH        AF                      ; PUSH CONTENTS OF AF ONTO THE STACK
            LD          A,20H                   ; A IS LOADED WITH 20H = 'SPACE'
            CALL        CONOUT                  ; A 'SPACE' IS SENT TO THE CONSOL OUTPUT
            POP         AF                      ; POP FROM THE STACK BACK TO AF
            RET
;
CLRSCN:                                         ; ROUTINE TO CLEAR THE CONSOLE SCREEN AND HOME TO UPPER LEFT CORNER
;                                               VT-100 'ESC' COMMANDS...CLR_SCRN='ESC'[2J  HOME='ESC'[H
;
CLEAR:      DEFB        1BH,5BH,32H,4AH         ; ESC=1BH [=5BH 2=32H J=4AH
            DEFB        03H                     ; 03H=END OF TEXT
            LD          HL,CLEAR
CALL        MARQ
;
HOME:       DEFB        1BH,5BH,48H ; ESC=1BH [=5BH H=48H
            DEFB        03H                     ; 03H=END OF TEXT
            LD          HL,HOME
CALL        MARQ
            RET
;
MARQ:                                           ; MARQUEE ROUTINE TO SEND MESSAGE IN (HL) TO CONSOLE SCREEN
MAR2:       LD          A,(HL)
            CALL        CONOUT
            INC         HL
            CP          03H                     ; LOOKING FOR "03H" TO INDICATE END OF MESSAGE
            JP          NZ,MAR2
            RET
;
CONOUT:                                         ; ROUTINE FOR CONSOLE OUTPUT TO DISPLAY
            PUSH        AF                      ; STORE CONTENTS OF 'A' REGISTER "OUTPUT CHARACTER" IN 'AF' REGISTER PAIR
CONO1:      IN          A,(KBDST)               ; LOAD STATUS PORT INTO 'A'
;                                               STATUS PORT=MSB[0][0][0][0][0][0][0][0]LSB MEANS DISPLAY BUSY
;                                               STATUS PORT=MSB[0][0][0][0][0][1][0][0]LSB MEANS DISPLAY READY
            AND         A,(KBDOT)               ; 'AND' REGISTER 'A' WITH 04H=MSB[0][0][0][0][0][1][0][0]LSB
            JR          NZ,CONO2                ; IF RESULT IS NOT-ZERO, JUMP OUT OF LOOP TO CONO2
            JR          CONO1
CONO2:      POP         AF                      ; RESTORE "OUTPUT CHARACTER" TO REGISTER 'A'
            OUT         (KBDDT),A               ; SEND "OUTPUT" TO (KBDDT)
            RET
;
```

```
CONIN:                                          ; ROUTINE FOR CONSOLE INPUT FROM KEYBOARD
            IN          A,(KBDST)               ; STATUS PORT=MSB[0][0][0][0][0][0][0][0]LSB MEANS NO CHARACTER WAITING
;                                                 STATUS PORT=MSB[0][0][0][0][0][0][0][1]LSB MEANS CHARACTER IS WAITING
            AND         A,(KBDIN)               ; 'AND' REGISTER 'A' WITH 02H=MSB[0][0][0][0][0][0][1][0]LSB
            JP          NZ,CONI1                ; IF RESULT IS NOT-ZERO, JUMP OUT OF LOOP TO CONI1
            JP          CONIN                   ; RESULT IS ZERO, SO GO BACK AND CHECK STATUS PORT AGAIN
CONI1:      IN          A,(KBDDT)               ; INPUT FROM DATA PORT 01H FOR PROPELLER CONSOLE
;                                               ; CODE BELOW FILTERS INPUT AND RETURNS...
            AND         7FH                     ; ASCII RANGE 00H-7FH COVERS ALL 128 CHARACTERS; PASSES ALL ASCII UP TO 7FH, THEN 'A'= 0,1,2...

            CP          61H                     ; 61H-7FH ARE LOWER CASE LETTERS; ASCII (00H-61H)=C;ASCII (62H-7FH)=NC;ASCII (80H-FFH)=C BUT
NOT ON KEYBOARD
            JP          C,ECHO                  ; SENDS ASCII(00H-60H) TO OUTPUT CONSOLE DISPLAY; NOTE-CARRY SET IF NEG OR GREATER THAN FF
            CP          7CH                     ; ONLY ASCII(61H-7FH) REMAIN; FILTER OUT ASCII < 7CH
            JP          NC,ECHO                 ; SENDS ASCII(7DH-7FH) TO OUTPUT CONSOLE DISPLAY ONLY ASCII(61H-7CH) REMAIN
            SUB         20H                     ; ONLY ASCII(41H-5CH) REMAIN...IE...(A TO \) WHICH GO TO CONSOLE DISPLAY
ECHO:       CP          18H                     ; 18H='CAN' OR CANCEL
            JP          Z,EXEC4                 ; IF 'A' IS 'CAN' RETURN WITH NO ACTION
            JR          CONOUT                  ; SEND 'A' TO CONSOLE OUTPUT DISPLAY
;
PTXT:       LD          A,(HL)
            CP          03H                     ; PRINT A MESSAGE
            RET         Z                       ; IF [03H], THEN END AND RETURN
            CALL        CONOUT
            INC         HL
            JR          PTXT
;
HXOT4:      LD          C,H                     ; 16 HEX OUTPUT ROUTINE
            CALL        HXO2
            LD          C,L
HXO2:       LD          A,C
            RRA
            RRA
            RRA
            RRA
            CALL        HXO3
            LD          A,C
HXO3:       AND         0FH
            CP          10
            JP          C,HADJ
            ADD         A,7
HADJ:       ADD         A,30H
OTA:        PUSH        BC
            LD          C,A
            CALL        CONOUT                  ; SEND TO CONSOL WHAT'S IN REGISTER A
            POP         BC
            RET
;
;                                                 PRINT [HL] AND A SPACE
HLSP:       PUSH        HL
            PUSH        BC
            CALL        LADR
            LD          A,20H
            CALL        CONOUT
            POP         BC
            POP         HL
            RET
;
;                                                 PRINT [HL] ON CONSOL
LADR:       LD          A,H
            CALL        LBYTE
            LD          A,L
LBYTE:      PUSH        AF                      ; STORE "H" IN "AF"
            RRCA                                ; MSB[X000 000]LSB = MSB[0X00 0000]LSB
            RRCA                                ; MSB[0X00 000]LSB = MSB[00X0 0000]LSB
            RRCA                                ; MSB[00X0 000]LSB = MSB[000X 0000]LSB
            RRCA                                ; MSB[000X 000]LSB = MSB[0000 X000]LSB
            CALL        SF598
            POP         AF                      ; RESTORE "H" FROM "AF"
SF598:      CALL        CONV
            LD          A,C
            JP          CONOUT
;
;                                                 CONVERT HEX TO ASCII
CONV:       AND         0FH
            ADD         A,90H
            DAA
            ADC         A,40H
            DAA
            LD          C,A
            RET
;
BITS1:      PUSH        DE                      ;  DISPLAY 8 BITS OF [A]
            PUSH        BC
            LD          E,A
            CALL        BITS
            POP         BC
            POP         DE
            RET
;
BITS:       LD          B,08H                   ;  DISPLAY 8 BITS OF [E]
            CALL        SPACE
SF76E:      SLA         E
            LD          A,18H
            ADC         A,A
            LD          C,A
            CALL        CONOUT
            DJNZ        SF76E
            RET
;
BAD:        PUSH        AF                      ; PRINT BAD
            PUSH        HL                      ; SAVE ORIGINAL 'AF' AND 'HL'
            LD          HL,MSG11                ; PRINT 'BAD: ' TO CONSOL
            CALL        MARQ                    ; PRINT MARQUEE ROUTINE
            POP         HL                      ; RESTORE HL
            POP         AF                      ; RESTORE AF
            RET
;
BOOT:       LD          HL,MSG22                ; OUTPUT MESSAGE
            CALL        MARQ                    ; OUTPUT TO CONSOLE
;
                                                ; THIS ROUTINE CAN ACCESS THE FLOPPY DISK CONTROLLER'S SIX I/O PORTS STARTING AT F8H
                                                ; IT CAN BE USED TO CREATE CBOOT CODE IN MEMORY AT LOCATION (007D)H
                                                ; THIS CODE SEEMS TO WORK ON WESTERN DIGITAL FD-1771 CONTROLLERS BUT MAYBE NOT LATER ONES
                                                ; AS THE PORT VALUES MAY HAVE BEEN CHANGED.
                                                ;
                                                ; (0F8H)-OUT             DISK COMMAND PORT
                                                ; (0F8H)-IN DISK STATUS PORT
                                                ; (0F9H)-I/O             TRACK REGISTER PORT
                                                ; (0FAH)-I/O             SECTOR REGISTER PORT
                                                ; (0FBH)-I/O             DATA PORT
                                                ; (0FCH)-IN DISK WAIT PORT (XRDY/PRDY)
                                                ; (0FCH)-OUT             DISK EXTENDED COMMAND PORT
                                                ;
                                                ; THE CODE APPEARS VERY SIMILAR TO THE BOOT LOADER FOR TARBELL 1011D (FD-1771) CONTROLLER
                                                ; IT'S PURPOSE IS TO READ THE FIRST SECTOR OF TRACK 0 INTO MEMORY AT 0000H, AND THEN EXECUTE IT
;
            IN          A,(WAIT)                ; WAIT FOR HOME
            XOR         A                       ; COMPLETE
```

```
            LD          L,A                 ; SET L=0
            LD          H,A                 ; H=0; L=0
            INC         A                   ; SET A=1
            OUT         (SECT),A            ; SECTOR=1
            LD          A,8CH               ; READ SECTOR
            OUT         (DCOM),A            ;
RLOOP:      IN          A,(WAIT)            ; WAIT FOR DRQ OR INTRQ
            OR          A                   ; SET FLAGS
            JP          P,RDONE             ; DONE IF INTRQ
            IN          A,(DDATA)           ; READ A BYTE OF DATA
            LD          (HL),A              ; LOAD IT INTO MEMORY
            INC         HL                  ; INCREMENT MEMORY POINTER
            JP          RLOOP               ; DO IT AGAIN
RDONE:      IN          A,(DSTAT)           ; READ DISK STATUS
            OR          A                   ; SET FLAGS
            JP          Z,SBOOT             ; IF ZERO, GO TO SBOOT AT 007DH
            HALT                            ; DISK ERROR, SO HALT
;
            END
```