TABLE OF CONTENTS

USER GUIDE

A. INTRODUCTION

The ZPU Card, TDL's Altair/IMSAI compatible Z-80 CPU card was designed to allow the Z-80 microprocessor to run, without modification to the mainframe, in either an Altair 8800 or an IMSAI 8080. At the same time, the design was configured to allow maximum versatility to the user, allowing the full potential of the Z-80 to be available to the user.

The ZPU Card is constructed of only the finest materials throughout. All components are first quality prime and obtained from reputable distributors, factories, or their representatives. No surplus material is used anywhere in the design.

In order to complement the Z80, which requires only a regulated +5 Volt supply, no components were used which require any other voltage. The total current drain is typically 750ma.

Separate jacks are provided to accomodate the front panel connectors of both the Altair and the IMSAI, and the ZPU user may at his discretion elect to install either one, or both during assembly.

B. SYSTEM DESCRIPTION

1. The Z-80

The specifications and details of the Z-80 are covered in depth in Zilog's Z-80 CPU Technical Manual which is provided with this kit. A complete understanding of the theory of operation for this board requires a careful study of this manual.

2. Theory of Operation

Proper operation of the Z-80 in the Altair Bus requires the generation of a number of bus signals not generated by the Z-80. The ZPU Card creates these by interaction and gating of the Z-80's status signals and the clock lines. The most important status signals generated by the Z-80 are:

1. Memory Request
2. I/O Request
3. Read
4. Write
5. M1

These 5 signals properly gated are used in conjunction with the clock to generate all of the required control timing.

The Z-80, unlike the 8080, outputs continuous status information wheras the 8080 information is strobed into an 8 bit latch (usually an 8212) during "Sync" time. Consequently, the Z-80 generates no sync pulse. In order to retain the Altair Bus structure, a "psuedo-sync pulse" was created.

Specifically, PSYNC is generated by gating I/O request and Memory Request thru a NAND gate (IC21) whose output goes to the input of a 74LS74 (IC16) which is clocked by the Phase 2 signal. PSYNC is taken off of the $\bar{Q}$ of IC16.

A wait is accomplished by gating the PRDY signal and forcing a low into the wait control line of the Z80. In addition, an extra PRDY line has been made available which may be jumpered to any unused bus line for future applications. When not in use these lines should be jumpered together. (Pins 3 and 5 of IC17) The wait signal is initiated by the coincidence of the clock pulse with the pulling down of any of the 3 ready lines (PRDY,XRDY, LRDY)

The Interrupts Enabled flag is not provided on the Z-80. This has been simulated by the use of an 8 input NAND gate (IC14) and some decode gating (IC17) feeding a set-reset flip flop (IC18) to provide the user with a proper indication when the interrupts are enabled.

The interrupt pin of the Z-80 is handled in exactly the same fashion as that of the 8080, coming to the same bus pin. However, the non-maskable interrupt pin of the Z-80, which represents a significant feature of the Z80 is brought out to a pull-up resistor, and may be jumpered to pin 4 on the bus, VIØ, the highest priority interrupt line. Thus configuring the Z-80 into the Altair Bus does not detract from this Z-80 feature.

The SSTACK status signal of the 8080 is not generated. Instead, the Z-80 REFRESH signal may be jumpered out to this line for use with future dynamic memory designs.

Processor write is generated by the Z-80, however in this application we have added some additional delay in order that the STATUS OUT or MWRITE may be properly decoded.

Handling of the remaining control timing is straight-forward. HALT ACK is generated by the Z-80. The MREAD signal is a function of the Z-80 READ and MREQ signals. STATUS OUTPUT is a function of WRITE in conjunction with an I/O Request. STATUS INPUT is a function of a

READ in conjunction with an I/O Request. PDBIN is a function of the READ signal. The Interrupt Acknowledge signal is a function of a simultaneous M1 and I/O request.

All processor signals with the exceptions of Phase One, Phase Two and Not CLOCK are tri-statable thru the normal Altair Bus Signal.

3. The Clock

The ZPU card features two clocks on-board. The first is fixed at 2Mhz thru crystal control, and the second is variable between less than 1 and greater than 4 Mhz by means of a 20 turn trimpot.

The 2Mhz crystal controlled clock is selected by placing a jumper between the augat pins labeled "C" and "2M".

The variable speed clock is selected by jumpering between "C" and "V". (The pins "C","2M" and "V" are located in area A on the ZPU Card.)

The crystal oscillator is a parallel resonant circuit using a 2Mhz crystal in conjunction with several gates of IC24, a 4049 CMOS oscillator chip. This clock generates Phase One, Phase Two, and system Not CLOCK.

The variable oscillator utilizes the remaining sections of IC24 in a free-running oscillator whose frequency is controlled by a precision RC network, and the frequency may be varied by adjusting R33, a 20K 20 turn trimpot. The variable oscillator presents Phase One and Two to the bus. Not CLOCK is always a function of the crystal oscillator and is always maintained at 2Mhz by that clock so that peripheral cards may be made to operate correctly regardless of processor speed. See the section on High-speed operation for details on this.

Regardless of which clock is selected, if the variable clock is tuned to within 100Khz or so of the crystal, there is a tendency for the 2 clocks to "lock in" to each other, that is to go into a fixed resonance. The operational effect of this is that when the variable

clock is selected in this condition, initial frequency change either up or down will tend to be resisted, until the frequency "jumps" roughly 50khz, at which point smooth frequency adjustment may be made.

Two augat pins (in area "B" and "C" respectively on the board) are provided for observation of the Phase One and Phase Two signals. These points are test points only and not intended for adjustment of clock speed. Clock speed should always be measured at point C in area A.

By removing the jumper choosing either of the two on-board clocks and connecting the common pin (C) to an external frequency source, the ZPU card may be synchronized with another system if the user chooses. This also makes it possible to run the processor at very low speeds (down to DC) which on occassion can be tremendously useful. (For example, individual T-states may be observed on the front panel.)

4. I/O Operation

A visual inspection of the ZPU card reveals more buffers (8T97s or 74367s, ICs 1 - 10) than are usually seen on a CPU card. This additional buffering was necessary to reduce bus loading and to assure normal front panel operation.

The front panels of both the Altair and the IMSAI look at the high order addresses for information about the I/O port number during I/O operations. This was optional with the original designers of the 8080 systems because the I/O port number is output to both the high and low order addresses by the 8080.

The Z-80 outputs I/O port information only to the low order addresses. (Contents of the accumulator are then present on the high order addresses.) So, in order for the sense switches to operate normally 8 additional buffers have been added which transfer the lower 8 bits to the high order address lines during I/O operations.

C. OPERATION

The normal configuration of the ZPU card is that which enables it to operate in an Altair or IMSAI with other peripheral boards.

The kit as supplied and the instructions as given result in a CPU card which may act as a direct replacement for your current 8080 processor. There are however some options which may be exercised by the user which take advantage of several of the Z-80 options. These are:

1. Connecting the REFRESH signal to pin 98 on the bus.
2. Connecting the Non-maskable interrupt to vectored interrupt lines.
3. Altering the processor speed.
4. Use of the duplicate PRDY line.

1. The Refresh Line

Pin 28 of the Z-80 outputs a $\overline{\text{RFSH}}$ signal, which may be used to provide refresh timeing for dynamic memories. This signal may be placed on pin 98 of the bus. Pin 98 is normally occupied by SSTACK on your 8080 system, however, this status indicator is not terribly useful and was ommitted on the Z-80 altogether. So, we chose this line for RFSH.

The $\overline{\text{RFSH}}$ signal may be picked up at Area F, immediatly to the left of the Z-80, and jumpered to the pad in Area G, straight down and slightly to the left from the Z-80. This places the signal on the bus.

When the signal is on the bus, the status light on your front panel, labeled STACK will now stay lit when the processor is running, indicating that the REFRESH signal is on the bus.

For the exact timing information about the $\overline{\text{RFSH}}$ signal, see the Z-80 manual.

2. The Non-maskable interrupt

On the Z-80, pin 17 is $\overline{\text{NMI}}$, the non-maskable interrupt. To quote the Z-80 manual:

"The non maskable interrupt request line has a higher priority than $\overline{\text{INT}}$ and is always recognized at the end of the current instruction, independent of the status of the interrupt enable flip-flop. $\overline{\text{NMI}}$ automatically forces the Z-80 CPU to restart to location 0066_H."

This powerful interrupt capability is made available to the ZPU user.

Pin 17 of the Z-80 and pin 4 of the bus (VIØ) are normally both held high by pullups. Solder pads at location H and location E may be jumpered together, thus making the NMI available

at VIØ, the highest priority vectored interrupt line.

3. Altering the Processor Speed.

The Z-80 has the capability of operating from DC on up to some maximum limit greater than 2.5Mhz due to its static nature. To take full advantage of this capability the ZPU card has been designed with a variable speed clock on-board.

P1, an augat pin pin soldered to a wire represents the phase one and two inputs to the processor. If the pin is placed in J2, the augat pin labeled "V" in area A, then by adjusting the trimpot located above the crystal, the frequency may be varied over a range of approximatly 3 Mhz.

Normally, when one is reducing the speeds, simply turning the speed down is sufficient, and no problems will be encountered. For individuals whose systems may currently be marginal at 2Mhz, reducing the processor speed may well greatly increase reliability of your system.

When speed is increased it is sometimes necessary to readjust the timing of the 74123 for stable operation. This RC network (R36 and C26) effects the Phase one and Phase two relationships, which become more critical as processor speed is increased.

The procedure for speed adjustment is covered in the section on high-speed operation.

4. Use of the duplicate PRDY line.

This line was included in order to facilitate operation with the Altair 8800B, or for any other use the user might dream up.

The extra PRDY line comes off of IC17. Area D, immediatly to the left of the IC has 2 pads which are normally jumpered together. If one wishes to use the extra PRDY line, remove the jumper, and take the PRDY signal off of pin 3, the top of the two pads.

The 8800B requires 2 additional RDY lines. XRDY2 is on bus line 12. If operation with the 8800B is desired, jumper the additional RDY line on the ZPU to this bus pin. The other RDY line is FRDY, which is pin 58 on the bus. The user may use this line as he wishes.

In addition to these options, there are two points about operation which bear mentioning:

1. Single stepping the processor
2. Use of 500ns memory

The Z-80, unlike the 8080 does not necessarily stop on an M1. The processor however must be in an M1 for the front panel to operate normally. Rather than create circuitry to stop the Z-80 on an M1, we felt the simpler approach was simply to single step the processor to an M1, and then operate the front panel.

With the Z80, system integrity is even more important than with the 8080. In part due to its more efficient architecture (remember that the Z-80 executes 8080 software 10% faster at the same 2Mhz clock speed) and also due to slight timing variations, the Z-80 is more demanding of your system at 2Mhz. The primary practical importance of this occurs when 500ns memory is in use.

Memory manufacturers rate their memory speed as chip access time, neglecting to add the select and enable logic time. These add up with chip access time to what may be termed board access time. While the 8080 may not note the difference, the Z-80 may. Therefore, it is recommended that when 500ns memory is in use, if any problems in running programs are encountered, the simple addition of one wait state will resolve it. If the small reduction of execution time is of any importance, switch the Z80 to its variable clock, and increase the processor speed to accomodate for the difference. (It is due to the presence of otherwise minor inefficiencies in current systems that TDL implemented the variable speed clock. It allows you to get the most out of your system.)

The best approach to system integrity when using a faster, more efficient chip such as the Z-80 is to gradually upgrade your system to very high speed memory such as TDL's Z16K.

Other than these mentioned points, operation of your ZPU is rather identical to that of your 8080.

D. HIGH SPEED OPERATION

Among the many features of the Z80 is its ability to operate at clock speeds higher (and lower) than that of the 8080.

The ability to have a Z-80 operate in your system which was originally designed for a 2Mhz processor at clock speeds higher than 2Mhz is determined by 4 primary variables: 1. The Z80 chip itself, 2. The access time of your memory 3. The integrity of your system as a whole, 4. Your own technical knowledge.

The Z-80 chip itself is guaranteed to operate up to speeds of 2.5MHz. In practice we have found that the overwhelming majority of Z-80 chips operate comfortably at 3MHz. A good many operate at 4MHz. It is expected that Zilog will release Z80's tested good for 4MHz, at slightly higher cost. Pending the general availability of these chips, the ZPU is designed to operate at clock speeds up to 4Mhz.

Operation at 4Mhz requires a memory access time of 250ns. 3 Mhz requires 333ns. 2MHz requires 500ns. The quality and rated access time of the memory you possess should give a fair estimate of the maximum processor clock speed without wait states at which you can expect to operate your Z80 successfully. Bear in mind however that most manufacturers rate memory access time as chip access time, not board access time, as was described earlier. The Z-80 is less tolerant of slightly slow memory than is the 8080.

NOTE: Do not try to operate the Z-80 at higher than 2Mhz with unbuffered memory. This create excessive loading.

The integrity of your system is affected by many variables. Bear in mind that the Altair/IMSAI systems were not designed for 2MHz+ operation, and the system itself presents a final limit beyond which increased processor speed will be useless. The problems of noise, inductive and capacitive coupling, impedence matching etc. become increasingly significant as processor speed is increased.

A system of high integrity can operate at 4Mhz. High integrity impiles that the common faults which a home built computer may suffer from are absent. Such factors as sloppy construction, cold solder joints, or out of spec, but not quite "bad" components which might go un-noticed at 2MHz would be likely to present serious problems at 4MHz. Your own technical skill is the only solution to these problems.

Your own ability is the greatest variable you will encounter. A thorough understanding of how your

system operates, and in particular how the ZPU and the Z80 operate is the best guarantee of successful operation at 2MHz+. The better you understand your hardware, the more success you'll have at getting the last bit of speed from your processor.

The simplest procedure for increasing the speed of your processor is as follows:

1. Place P1 (from point C Area A) into the augat pin marked V.
2. Using an insulated adjustment tool, adjust the processor speed to 2MHz as measured at Point C.
3. If no accurate frequency measuring instrument is available, turn the 20 turn trimpot clockwise to maximum resistance which gives you the lowest clock speed, and then proceed.
4. Load a program into the processor which will show if the processor is operating properly. A listing of some sort would be ok. The DISPLAY MEMORY command of the monitor is very good.
5. Increase the processor speed by turning the trimpot counter-clockwise until the processor bombs. Turn the adjustment screw back ½ turn or so.
6. Run various programs on the processor at this speed and test for reliability. If any problems show up, reduce the speed a bit more.

Your processor is now running at the maximum speed which your system, without tune-up or adjustment, and without tweaking the values of any components, is capable of handling. Your own skill, of course, can cause this figure to rise.

It is important to note here that although the system $\overline{\text{CLOCK}}$ line is maintained at 2MHz regardless of processor speed, some boards, particularly I/O boards, use Phase 1 or Phase 2 for their timing, and will not operate correctly when the Z-80 speed is altered. In this case, the fix is very simple - cut the malfunctioning board trace from the phase 1 or 2 and jumper it to the $\overline{\text{CLOCK}}$ line. It will now operate correctly at processor speeds other than 2MHz and will still operate with your 8080 as well.

E. COMPATIBILITY

Due to the pin for pin compatibility which the ZPU shares with the Altair Bus structure, it is highly compatible with existing hardware. Bear in mind that the design of the ZPU was aimed at simulating the bus while not impeding the Z-80 in any way, or confining it by hardware compromises. Generally this has been achieved.

You will find that the front panel of your system will operate in the usual fashion with all switches serving their normal function and all lights (save STACK) indicating their normal signal conditions.

The only hardware "incompatibility" ever noted is that slightly out of spec memory which the 8080 will tolerate at 2Mhz will on rare occassions be found unacceptable to the Z-80. The solution would be to introduce one wait state in the memory, or slowing the processor down by a small fraction.

The Z-80 is 100% machine code compatible with the 8080's 78 instructions. Thus standard 8080 software will run without modification on the Z-80.

However, where the actual execution time (in real time) of each machine cycle is used to create a timing loop, 100% compatibility may not be found. This situation is created by a feature of the Z-80.

The architecture of the Z-80 is more efficient than that of the 8080. In its design, many instructions of the 8080, while having the same machine code, have fewer "T-states" and thus the instruction is executed faster in real time. It should be clear then, that 8080 software timing loops where real time length of execution is controlled in software will have to be readjusted to the higher real time execution speed of the Z-80. Note that this is true when the 8080 and the Z-80 are both run at 2MHz.

While this feature of the Z-80 may require minor software modifications for an occassional user, in general it is a very useful feature. For example, a member of the Amateur Computer Group of NJ has a benchmark program in basic which he has run on a large array of machines, both minis and micros. This program was run using Altair Basic using both the 8080 processor and the ZPU. Only the 8080 instructions of the Z-80 were used,

and both processors were maintained at a 2MHz clock speed. However, the ZPU executed the same program in 10% less time. This is a significant improvement.

As for 8080 languages, TDL procured and tested versions of virtually every language yet written for an 8080 processor. With one exception, they ran without a hitch.

This sole exception is Altair Basic. This basic has as part of its routines several occassions where the Parity Flag is checked as part of the function. In the Z-80, the parity flag indicates OVERFLOW during math routines, not Parity. As a result, Altair basic will not run on the Z-80.

The exact mechanics of this bug may be examined by studying the sections on Flags in the 8080 and Z-80 Technical Manuals.

Since the several routines which cause this bug to occur were written to reduce program space by several bytes, and are not required by the structure of the language, it can be patched by those who wish to do so. Appendix C of this manual describes the patching technique.

For those who do not wish to go to the trouble of generating the patch, it is advised that they procure TDL's 8K basic which is Altair compatible and which has a large number of exclusive and desirable features. It will be available as of Mid-September 1976.

No software incompatibilities other than the above have been encountered. In any applications of existing software, if a problem is found please inform us of the exact details in writing and we will be pleased to advise you on a proper solution.

ASSEMBLY

CAUTION

THE ZPU KIT CONTAINS TWO STATIC SENSITIVE DEVICES. DO NOT REMOVE THESE DEVICES FROM THEIR PROTECTIVE PACKING UNTIL NEEDED IN ASSEMBLY. HANDLE ONLY AS PER THE INSTRUCTIONS IN THIS MANUAL. FAILURE TO HEED THIS PRECAUTION MAY RESULT IN PERMENANT DAMAGE TO THESE DEVICES AND AUTOMATICALLY VOIDS THE WARRANTY.

ALSO, THE Z-80 IS NOT PIN FOR PIN COMPATIBLE WITH THE 8080. ATTEMPTING TO RUN THE Z-80 IN YOUR 8080 CPU CARD WILL DESTROY THE Z-80.

A. General Kit Building

It's a good feeling to construct a kit on your own, plug it in, and have it work the first time up. Two factors are of the utmost importance in this: quality engineering which makes the kit easy to build, and careful construction. We've taken care of the engineering, the construction is up to you. We've listed here some construction tips which are considered standard operation in most commercial shops. Following these procedures in your own construction will increase the likelyhood that your kits will work first time, every time.

1. ALWAYS read all of the instructions before starting construction.
2. Always work in a clean, well-lit area.
3. Use only high quality rosin core solder of a guage similar in size to the leads being soldered.
4. Ensure that you have all the necessary parts for a given stage of construction before starting that stage.
5. Use the lowest power soldering iron that will get the job done. A 25 watt iron is quite adequate for this kit.
6. Use a fine point soldering iron, and keep the tip clean and well tinned.
7. Avoid overheating the PC board and components.
8. Before soldering, check and make sure that the right component is in the right place. Having to remove and resolder a wrongly placed component is difficult, and there is a great likelyhood of damage to the board or component.

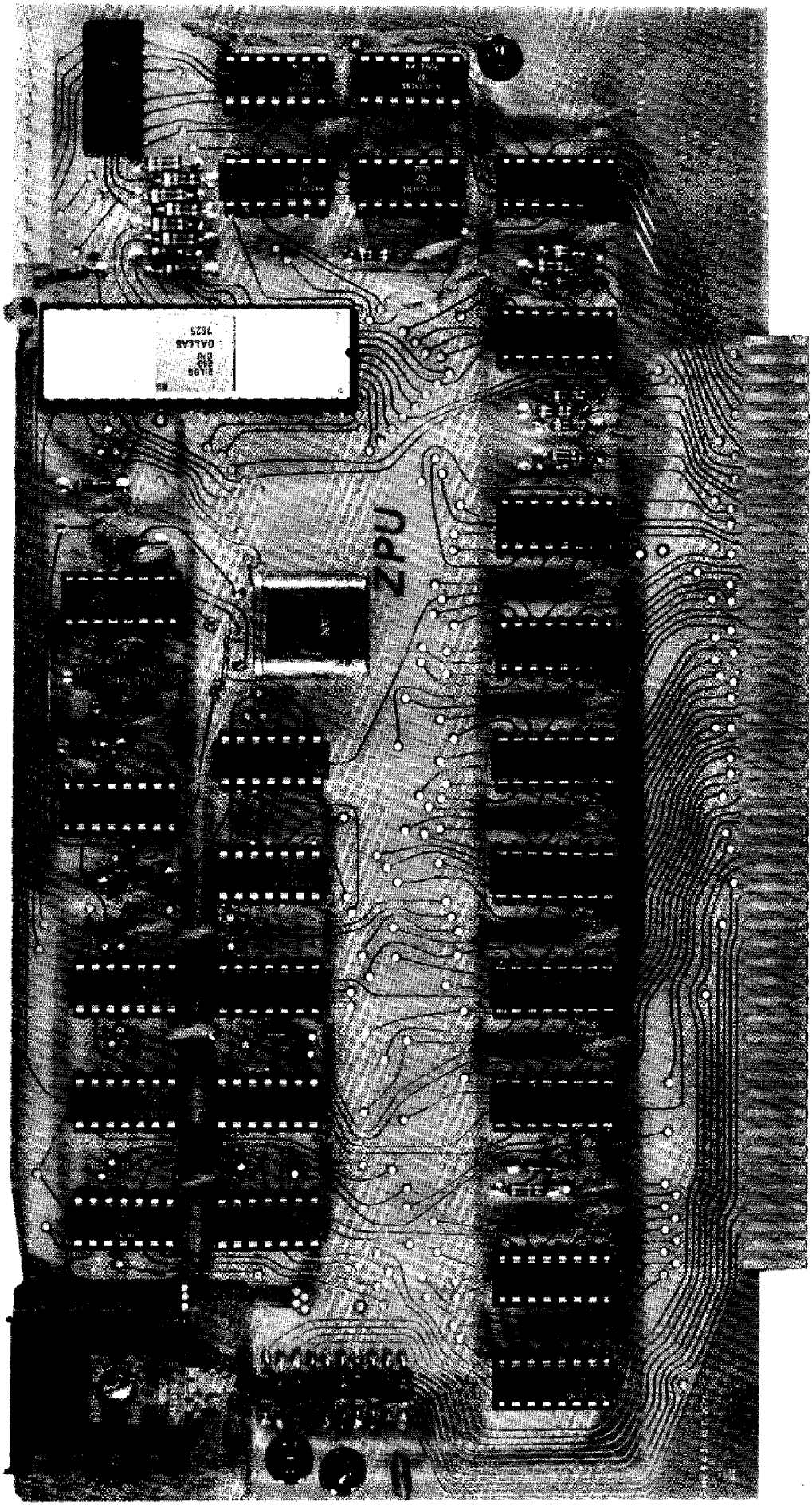
9. Apply the solder to the iron tip, the pad and the component lead at the same time. The solder will melt and flow in a second or two. If it doesn't, stop and find out why before continuing.
10. Use only enough solder to assure electro-mechanical integrity. 1/8th inch or so of the solder supplied with this kit is generally adequate around IC pads.
11. Look carefully at each joint both during and after soldering it. It should have a clean, bright appearance. If the surface is rough or dull it might be a "cold" solder joint. If so, reheat and apply very little or no additional solder.
12. Don't work on construction if you're very tired.
13. Always check the voltages on the appropriate IC pins after soldering and before installing the ICs in their sockets.
14. Never install ICs in sockets when there is voltage on the board.
15. ALWAYS install MOS/CMOS devices LAST, when you're sure that all else is perfect.
16. NEVER insert the board into its socket when power is on the machine.

B. Handling MOS/CMOS Devices

When handled correctly, static damage to these sensitive devices is quite unlikely to occur. The rules for correct handling are simple:

1. Keep everything in contact with everything else. While the IC is still in its case, hold it in your hand, touch both to the table, the PC board etc. This allows any static to discharge.
2. Work on a conductive surface. Bare grounded metal (a cookie tin or piece of aluminum foil will do.) is best. Glass very bad, plastic the worst.
3. Wear cotton clothes instead of synthetics.
4. A medium humidity environment is better than a very dry room.

These Rules are very simple. Remember: the most basic rule is to keep everything in contact with everything else. If you adhere to this rule and the others, plus add some common sense, it's very unlikely that you will ever damage a static-sensitive component.


Z8U

ZPU PARTS LIST

IC 1 to 10	8T97 or 74367
IC 11,12,20	74LS02
IC 13,15,19	74LS04
IC 14	74LS30
IC 16	74LS74
IC 17	74LS10
IC 18,21,22	74LS00
IC 23	74123
IC 24	4049
IC 25	Z-80
IC 26	7805

R 1 to 10	1K, 5%, Brown, Black, Red, Gold
R 12 to 20	1K, 5% " "
R 30	1K, 5% " "
R 11,31	100 ohm, 5%, Brown, Black, Brown, Gold
R 21	47 ohm, 5%, Yellow, Violet, Black, Gold
R 22 to 29	4.7K, 5%, Yellow, Violet, Red, Gold
R 32	4.7K, 5% " "
R 34	3.3K, 5%, Orange, Orange, Red, Gold
R 35	10K, 5%, Brown, Black, Orange, Gold
R 36	12K, 5%, Brown, Red, Orange, Gold
R 33	20K, 20 turn trimpot

C 1,2,14	47Mf, 25V, dipped tantalum electrolytic
C 3 to 13	.1Mf disc ceramic
C 16,18,19	.1Mf " "
C 15,17	.001Mf Disc Ceramic
C 20	33Mf,25V dipped tantalum electrolytic
C 21,24	.1Mf Disc Ceramic
C 27 to 31	.1Mf Disc Ceramic
C 22	10Pf " "
C 23	6 Pf " "
C 25	27 Pf Disc Ceramic
C 26	47 Pf " "

Y1	2Mhz Crystal
J1A	10 pin molex connector
J1B	16 pin high profile DIP socket
J3,4,5,6	Augat pins
P1	Augat Pin

1 Heatsink
1 ea. 6/32 x 5/16" machine screw, lockwasher, nut
1 ZPU PC board

12	14 pin low profile IC sockets
12	16 pin low profile IC sockets
1	40 pin high profile IC socket

Miscellaneous

6"	jumper wire
5'	solder
1	Zilog Z80 CPU Technical Manual
1	ZPU Documentation Manual
1	Paper tape of the ZAP monitor

E. Detailed Assembly Instructions

( ) 1. Read these instructions through once from beginning to end before continuing.

( ) 2. Inventory all parts against the parts list.

( ) 3. Open out the board layout diagram at the end of this section, and place the board so that it is similarly oriented in front of you. Compare the two and familiarize yourself with the layout.

( ) 4. Install the twenty 1K resistors (R 1 to 10, 12 to 20 and 30) in the appropriate locations, and solder.

NOTE: In soldering in large numbers of similar components, it is quickest to perform all similar actions on all of the components at the same time. For the above, you can bend the leads, insert the resistors, pull them close and bend the leads out to hold them in place, and then solder as 4 actions for all the resistors.

( ) 5. Repeat the above for the nine 4.7K resistors. (R 22 to 29, 32)

( ) 6. Install the two 100 ohm resistors (R11,31); the 10K resistor (R35); the 47 ohm resistor (R21); the 12K resistor (R36); the 3.3K resistor (R34)

( ) 7. Insert the twelve 14 pin IC sockets and the twelve 16 pin IC sockets in their respective positions, with all pin 1s toward the top of the board. (IC sockets have a notch or chamfer to indicate pin 1.)

( ) 8. Invert the board and solder all of the pins. Make sure each socket is all the way in before you solder - it's difficult to correct not fully inserted sockets after soldering.

NOTE: If the sockets tend to fall out, either bend two diagonally opposite leads on each, or place a piece of thin stiff cardboard over all the sockets and turn the board over holding them in place with the card board. Then slide the cardboard out.

( ) 9. Insert the 20K trimpot (R33) and solder in place.

( ) 10. Insert the 40 pin high profile socket in place, and solder. (NOTE: PIN ONE OF THE Z-80 GOES DOWN ALL OTHER PIN ONES ARE UP.)

( ) 11. Insert the 16 pin high profile socket (J1B) and solder. Pin one goes to the upper right.

( ) 12. Solder a 1½" piece of jumper wire into the top of one of the Augat pins. Be sure to not get any solder on the tip of the pin.

( ) 13. Trim back ¼" of insulation from the other end of the wire and insert this end into hole C of Area A on the board and solder. The pin is now P1.

( ) 14. Place another Augat pin on the tip of P1. Insert the tip of this second Augat pin into Hole V in Area A. Using the stiffness of the wire to hold the pin in position, solder the pin in Hole V.

( ) 15. Again using P1 as a holder, repeat the above procedure placing Augat pins in hole 2M of Area A and Hole 1 of Area B.

( ) 16. Cut another piece of jumper wire about 2" long. Use it as a holder while soldering the remaining Augat pin into hole 2 in Area C.

( ) 17. Install the 4 small disc ceramic capacitors (C22, 10pf; C23, 6pf; C25, 27pf; C26, 47pf) in their respective positions and solder.

( ) 18. Install the two .001Mf disc ceramic capacitors (C15 and 17) in position and solder.

( ) 19. Install the twenty-one .1Mf disc ceramic capacitors (C 3 to 13, 16, 18-19, 21, 24 and 27 to 31) in their respective positions and solder.

NOTE: The leads of C3 need to be left 3/16ths of an inch or so, and the disc pushed in toward the center of the board in order for the board edge to clear any card guides. Also, to get the discs close to the board, it may be necessary to bend the leads inward from the base slightly. If the insulation on the leads is too far down, grasp the lead in the teeth of long-nose pliers, rotate the lead, and pull the insulation off.

( ) 20. Install the three 47Mf dipped tantalum electrolytics in their positions and solder. Make sure that they are properly oriented for polarity.

( ) 21. Install the 33Mf dipped tantalum electrolytic (C20) and solder. Insure that the polarity is correct.

NOTE: Polarity of tantalums is marked in 3 ways. PLUS is either the lead with the dot next to it, the side of the component with the + stamped on it, or if the unit has one large dot on it, it is the right hand lead when the dot is oriented towards you.

( ) 22. Install the 7805 voltage regulator. Refer to the picture for correct orientation. The shortest distance to the hole in the heatsink goes under the 7805. The longest distance goes toward the top of the board. The leads of the 7805 should be bent down at 90 degree angles to go into the holes. Solder the leads.

NOTE: The screw holding the 7805 is inserted with the nut and lockwasher on the component side of the PC board.

( ) 23. Install a short jumper between the two solder pads to the left of IC17 in Area D. (This is an option - only install if the extra PRDY line is NOT being used.)

( ) 24. If you are using or plan to use the ZPU Board with an Altair 8800, now install the 10 pin molex connector (J1A) at the upper right hand corner of the board.

( ) 25. Install the crystal (Y1) immediatly below the 3 augat pins in area A. Bend the leads approximatly 3/16ths of an inch from the crystal body down 90 degrees in a smooth arc. Solder. (Don't overheat - it can damage the crystal.)

This compleats soldering of the board.

( ) 26. Trim all leads, including IC socket pins down as close to the board as you can using the flat side of diagonal cutters.

( ) 27. Using Acetone, Alcohol, or some other solvent, plus a stiff ½ inch artist's brush and a clean cloth, clean all the residue from the soldering operation off of the board.

NOTE: This is the construction step most often ommitted by the unwise. Cleaning the board will handle 95% of those "solder splashes" that can cause so much trouble, and make finding the remaining few a snap. Start in a corner, apply the solvent liberally by pouring on and "scrubbing" with the bruch. Before all the solvent evaporates, BLOT off the remainder with the cloth. (You can't rub over the sharp cut edges.) Repeat if necessary. Then apply some solvent to the rag and clean the board edges, connector etc. well.

( ) 28. Now examine the board carefully for solder shorts, cold solder joints, unsoldered leads etc. Corrent any problems which you find.

( ) 29. Check once more to be sure that you have all the right components in the correct spot.

Now you are ready to proceed with electrical checkout.

( ) 30. Measure the resistance between pins 1 and 50 on the edge connector. It should measure a fairly high resistance, 20,000 ohms or so. Completly open means the voltage regulator is not connected, or broken. A dead short indicates that either the regulator is blown, or you have a solder short.

( ) 31. If the resistance is OK, now insert the ZPU card (with no ICs other than the 7805) into your motherboard (with no other cards in the slots.) apply voltage, and measure the voltage between the center lead and the right lead of the 7805, pin 7 and pin 14 of the 14 pin IC sockets, and pin 8 and 16 of the 16 pin IC sockets. All should measure within a very small fraction of +5 volts. If they do not, find the problem and correct it before proceeding.

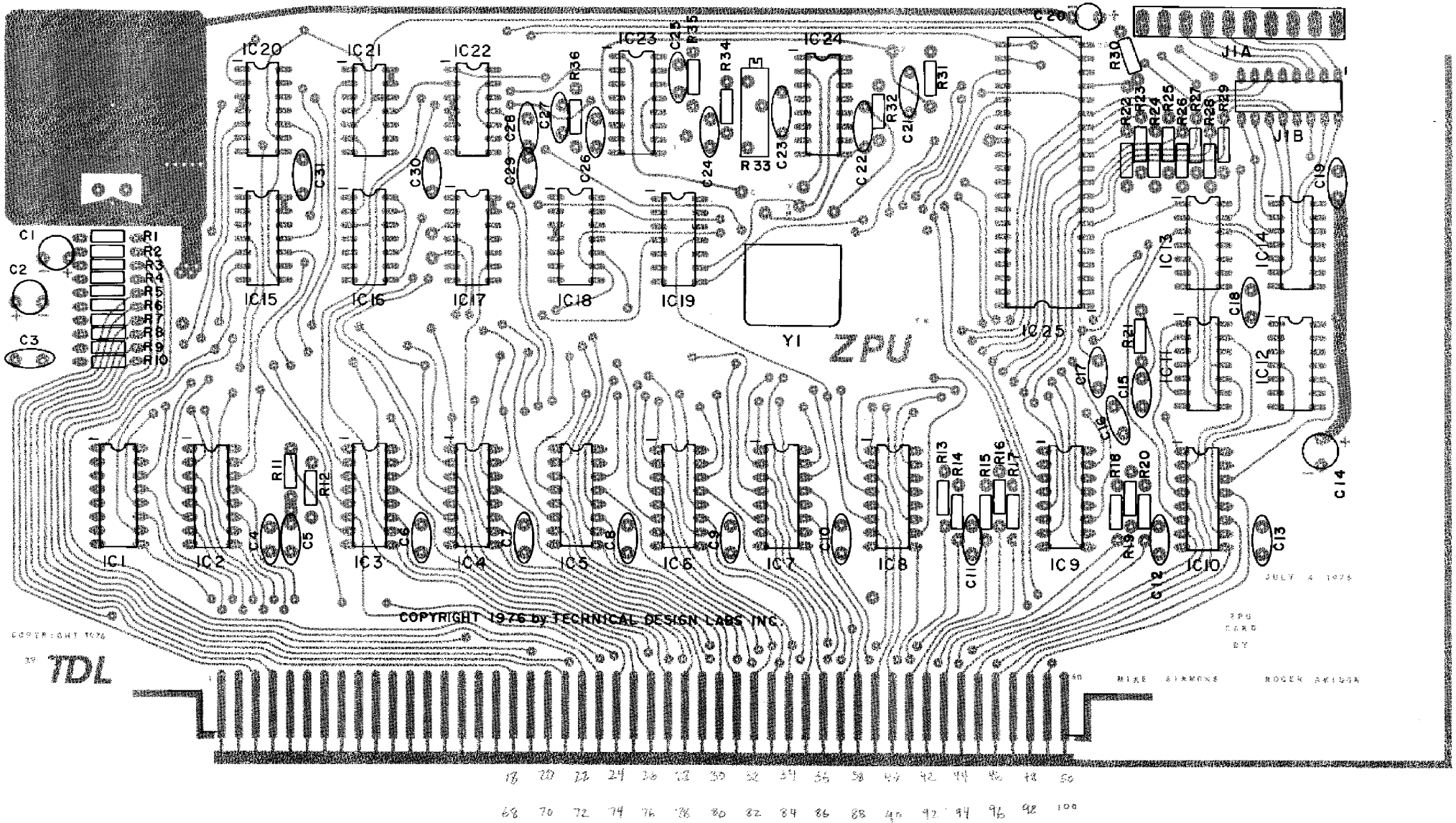
( ) 32. When these voltages are correct, install all the ICs except the 4049 and the Z-80. Put the board back in the motherboard, turn on the power and check the voltages again. They should be the same.

( ) 33. If the voltages are OK, now install the 4049 and then the Z-80. Be sure to adhere strictly to the procedure for handling MOS/CMOS devices outlined at the beginning of this section.

( ) 34. Insert P1 into pin 2M in area A (this places the clock on the 2MHz crystal.

This completes mechanical construction and electrical checkout of the ZPU card. Now go to the ZAP MONITOR section for procedure to checkout the operation of the Z80 board itself.

WARNING: NEVER INSERT ANY IC OR BOARD INTO ITS SOCKET WHEN POWER IS APPLIED. THIS IS LIKELY TO SEVERLY DAMAGE THE BOARD OR COMPONENT.


IC20
IC21
IC22
IC23
IC24
IC15
IC16
IC17
IC18
IC19
IC25
IC1
IC2
IC3
IC4
IC5
IC6
IC7
IC8
IC9
IC10
IC11
IC12
IC13
IC14
Y1
ZPU
J1A
J1B
C1
C2
C3
C4
C5
C6
C7
C8
C9
C10
C11
C12
C13
C14
C15
C16
C17
C18
C19
C20
C21
C22
C23
C24
C25
C26
C27
C28
C29
C30
C31
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
R16
R17
R18
R19
R20
R21
R22
R23
R24
R25
R26
R27
R28
R29
R30
R31
R32
R33
R34
R35
R36
COPYRIGHT 1976 by TECHNICAL DESIGN LABS INC.
TDL
JULY 4 1976
ZPU
CARD
BY
MIKE SIMMONS
ROGER AMIDON

THE ZAP MONITOR

A. FEATURES

The ZAP Monitor is a 1K version of TDL's 2K ZAPPLE Monitor. It is relocatable (can be placed anywhere in memory), expandable ("modules" of additional commands can be tacked on at the end, like cars on a freight train.), and quite powerful as a system executive.

The expandable feature should be of great interest to the user. Since it is designed in a modular fashion, and since the ZAPPLE is its direct parent, this monitor features tremendous expandability - either of routines generated by the user, or by routines provided by Technical Design Labs. Several "modules" which will be of great interest include powerful "breakpoint", "search" and "register display" commands. Paper tapes of these modules will be available from TDL in the early fall. (Contact us for the latest word on availability.)

B. LOADING PROCEDURE

The loading procedure is presented on the following two pages exactly as it was prepared on the computer.

```
.LIST
.REMARK /
   THIS VERSION OF THE TDL BOOT LOADER AND
TDL RELOCATING LOADER SHOULD MAKE IT EASIER
FOR PEOPLE WITH WIDELY DIVERGENT HARDWARE
TO LOAD THE MONITOR.

THE GENERAL MEMORY MAP LOOKS LIKE THIS:
0000 - 00FF  BOOT LOADER
0100 - 01FF  RELOCATING LOADER
0200 - FFFF  WHERE MONITOR MAY BE PLACED

THE BOOT LOADER MEMORY MAP:
0000 - 0019  HARDWARE INITIALIZATION ROUTINE
001A - 001C  LXI SP,200H
001D - 001F  LXI H,01F3H (CHANGED BY UPPER LOADER)
0020 - 0022  CALL READER (CALL CHANGED TO JMP)
0023 - 00FF  BOOT LOADER AND READER ROUTINES

THE THREE INSTRUCTIONS SHOWN IN THE BOOT LOADER
MEMORY MAP ARE FIXED AND MUST BE AS SHOWN,
BECAUSE THE RELOCATING LOADER USES OR MODIFIES
THEM.

THE READER ROUTINE IS EXPECTED TO RETURN AN
8 BIT CHARACTER FROM THE TAPE EACH TIME IT
IS CALLED.

THE BOOT LOADER ROUTINE LOADS THE RELOCATING
LOADER INTO MEMORY STARTING AT 01F3H AND
DOWNWARD TO 0100H.
/
.PAGE
```

UART STYLE BOOT LOADER ROUTINES

```
                        .LIST
                        ;
                        ;
0000  C31A00            ..INIT: JMP     ..LOAD  ;NO INITIALIZATION NEEDED
                        ;
001A                    .LOC 1AH
                        ;
001A  310002            ..LOAD: LXI     SP,200H ;SET STACK
001D  21F301                    LXI     H,01F3H ;LOAD LOADER
0020  CD2B00            ..RDR:  CALL    ..READ  ;GET A CHARACTER
0023  BD                        CMP     L       ;TEST LEADER
0024  28FA                      JRZ     ..RDR   ;WALK OVER LEADER
0026  2D                        DCR     L       ;MOVE POINTER
0027  77                        MOV     M,A     ;SAVE DATA
0028  20F6                      JRNZ    ..RDR   ;GET MORE DATA OR
002A  E9                        PCHL            ; GO TO LOADER
                        ;
                        ;  ALTAIR SIOA REV 1.0 READER ROUTINE
                        ;
002B  DB00              ..READ: IN      0       ;STATUS PORT
002D  E601                      ANI     1       ;DATA AVAILABLE BIT
002F  20FA                      JRNZ    ..READ  ;0=DATA AVAILABLE
0031  DB01                      IN      1       ;DATA PORT
0033  C9                        RET             ;DONE

                        ;
                        .LIST
                        ;
                        ;  PTCO 3P+S READER ROUTINE
                        ;
002B  DB00              ..READ: IN      0       ;STATUS PORT
002D  E640                      ANI     040H    ;DATA AVAILABLE BIT
002F  28FA                      JRZ     ..READ  ;1=DATA AVAILABLE
0031  DB01                      IN      1       ;DATA PORT
0033  C9                        RET             ;DONE
                        ;
                        ;
                        .PAGE
```

```
                        .LIST
                        ;
                        ;
                        ; THIS ROUTINE WOULD BE USED FOR AN I/O BOARD
                        ; THAT USES A MOTOROLA ACIA.
                        ; SUCH AS AN ALTAIR 2SIO.
                        ;
0000  3E03              ..INIT: MVI     A,003H  ;RESET
0002  D320                      OUT     20H
0004  3E11                      MVI     A,011H  ;CLOCK/16, 8 DATA BITS
0006  D320                      OUT     20H     ;NO PARITY
0008  C31A00                    JMP     ..LOAD
                        ;
001A                    .LOC 1AH
                        ;
001A  310002            ..LOAD: LXI     SP,200H ;SET STACK
001D  21F301                    LXI     H,01F3H ;LOAD LOADER
0020  CD2B00            ..RDR:  CALL    ..READ  ;GET A CHARACTER
0023  BD                        CMP     L       ;TEST LEADER
0024  28FA                      JRZ     ..RDR   ;WALK OVER LEADER
0026  2D                        DCR     L       ;MOVE POINTER
0027  77                        MOV     M,A     ;SAVE DATA
0028  20F6                      JRNZ    ..RDR   ;GET MORE DATA OR
002A  E9                        PCHL            ; GO TO LOADER
                        ;
                        ; READER ROUTINE
                        ;
002B  DB20              ..READ: IN      20H     ;STATUS PORT
002D  E601                      ANI     1       ;DATA AVAILABLE BIT
002F  28FA                      JRZ     ..READ  ;1=DATA AVAILABLE
0031  DB21                      IN      21H     ;DATA PORT
0033  C9                        RET             ;DONE
                        ;
                        ;
                        .PAGE
```

INTEL USART BOOT LOADER ROUTINE

```
                        .LIST
                        ;
                        ;
                        ; THIS ROUTINE WOULD BE USED FOR AN I/O BOARD
                        ; THAT USES AN INTEL USART.
                        ; SUCH AS AN IMSAI 2SIO.
                        ;
0000  3ECE              ..INIT: MVI     A,0CEH  ;CLOCK/16, 8 DATA BITS
0002  D303                      OUT     3       ;NO PARITY, 2 STOP BITS
0004  3E17                      MVI     A,017H  ;ENABLE XMIT & REC
0006  D303                      OUT     3       ;RESET ERROR FLAGS
0008  C31A00                    JMP     ..LOAD
                        ;
001A                    .LOC 1AH
                        ;
001A  310002            ..LOAD: LXI     SP,200H ;SET STACK
001D  21F301                    LXI     H,01F3H ;LOAD LOADER
0020  CD2B00            ..RDR:  CALL    ..READ  ;GET A CHARACTER
0023  BD                        CMP     L       ;TEST LEADER
0024  28FA                      JRZ     ..RDR   ;WALK OVER LEADER
0026  2D                        DCR     L       ;MOVE POINTER
0027  77                        MOV     M,A     ;SAVE DATA
0028  20F6                      JRNZ    ..RDR   ;GET MORE DATA OR
002A  E9                        PCHL            ; GO TO LOADER
                        ;
                        ; READER ROUTINE
                        ;
002B  DB03              ..READ: IN      3       ;STATUS PORT
002D  E602                      ANI     2       ;DATA AVAILABLE BIT
002F  28FA                      JRZ     ..READ  ;1=DATA AVAILABLE
0031  DB02                      IN      2       ;DATA PORT
0033  C9                        RET             ;DONE
                        ;
                        ;
                        .PAGE
```

```
                        .LIST
                        ;
                        ; THIS IS AN EXAMPLE OF A ROUTINE THAT
                        ; "MIGHT" BE USED TO CONTROL A PARALLEL
                        ; READER.
                        ;
0000  3E20      ..INIT: MVI     A,20H   ;INITIALIZE THE HARDWARE
0002  D31B              OUT     01BH
0004  3E30              MVI     A,30H
0006  D31B              OUT     01BH
0008  3E28              MVI     A,28H
000A  D31B              OUT     01BH
000C  3E20              MVI     A,20H
000E  D31B              OUT     01BH
0010  C31A00            JMP     ..LOAD
                        ;
001A                    .LOC 1AH
                        ;
001A  310002    ..LOAD: LXI     SP,200H ;SET STACK
001D  21FE01            LXI     H,01FEH ;LOAD LOADER
0020  CD2B00    ..RDR:  CALL    ..READ  ;GET A CHARACTER
0023  BD                CMP     L       ;TEST LEADER
0024  28FA              JRZ     ..RDR   ;WALK OVER LEADER
0026  2D                DCR     L       ;MOVE POINTER
0027  77                MOV     M,A     ;SAVE DATA
0028  20F6              JRNZ    ..RDR   ;GET MORE DATA OR
002A  E9                PCHL            ; GO TO LOADER
                        ;
                        ; READER ROUTINE
                        ;
002B  3E20      ..READ: MVI     A,20H
002D  D31B              OUT     1BH
002F  3E30              MVI     A,30H
0031  D31B              OUT     1BH
0033  DB1B      ..LOOP: IN      1BH     ;STATUS
0035  E601              ANI     1
0037  28FA              JRZ     ..LOOP
0039  DB1A              IN      1AH     ;DATA
003B  2F                CMA             ;UPSIDE DOWN
003C  F5                PUSH    PSW
003D  3E28              MVI     A,28H
003F  D301              OUT     1B
0041  3E20              MVI     A,20H
0043  D31B              OUT     1BH
0045  F1                POP     PSW
0046  C9                RET
                        ;
                        ;
                        .END
```

```
                        ;
                        ;
                        .TITLE   / APPENDIX B.     <*TDL RELOCATING LOADER, VERSION
                         3.2 - DEC. 28, 1976*>/
                        ;
                        ;       STAND-ALONE VERSION, TO BE USED
                        ;       AS A BINARY BOOT-STRAP LOADER.
                        ;
                        .PABS           ;ABSOLUTE ASSEMBLY
                        ;
00FF                    SENSE   = 0FFH  ;ALTAIR/IMSAI/TDL/ETC SENSE SWITCHES
001E                    HLMOD   = 01EH  ;ADDRESS MODIFIED TO A JMP
0020                    USER    = 0020H ;USER WRITTEN I/O ROUTINE
0200                    TOP     = 0200H ;STACK AREA
                        ;
0100                    .LOC    100H    ;LOADER ON PAGE ONE
                        ;
                        ;       SET-UP
                        ;
0100  3EC3              BEGIN:  MVI     A,JMP   ;IN CASE OF TROUBLE
0102  32 001D                   STA     HLMOD-1 ; STORE A JMP TO HERE
0105  21 0100                   LXI     H,BEGIN ; AT BOTTOM
0108  22 001E                   SHLD    HLMOD   ;
                                                ;
010B  32 0020                   STA     USER    ;MODIFY READER CALL
                                                ; TO A JMP
010E  31 0200                   LXI     SP,TOP  ;INSURE A STACK
0111  DBFF                      IN      SENSE   ;SEE WHERE TO LOAD
0113  FE02                      CPI     2       ;CAN'T BE LESS THAN PAGE 2
0115  DA 0159                   JC      ERROR   ;ABORT IF SO
0118  47                      MOV     B,A     ;SAVE RELOCATION
0119  0E00                    MVI     C,0     ;FORCE PAGE BORDER
011B  D9                        EXX             ;SAVE IT IN BC'
                        ;
                        ;       ACTUAL LOADER CODE
                        ;
011C  CD 01BE           LOD0:   CALL    RDR     ;GET A CHARACTER
011F  D63A                      SUI     ':'     ;ABSOLUTE FILE?
0121  47                        MOV     B,A     ;SAVE INFO
0122  E6FE                      ANI     0FEH    ;KILL BIT ZERO
0124  20F6                      JRNZ    LOD0    ;FILE NOT STARTED YET
0126  57                        MOV     D,A     ;ZERO CHECKSUM
0127  CD 01A0                   CALL    SBYTE   ;GET FILE LENGTH
012A  5F                        MOV     E,A     ;SAVE IN E
012B  CD 01A0                   CALL    SBYTE   ;LOAD MSB
012E  F5                        PUSH    PSW     ;SAVE IT
012F  CD 01A0                   CALL    SBYTE   ;LOAD LSB
0132  E1                        POP     H       ;H=MSB
0133  6F                        MOV     L,A     ;L=LSB
0134  E5                        PUSH    H
0135  DDE1                      POP     X       ;INDEX X=LOAD ADDR
0137  D9                        EXX             ;ALTERNATE REG.'S
0138  C5                        PUSH    B       ;BC'=RELOCATION
0139  D9                        EXX
013A  CD 01A0                   CALL    SBYTE   ;GET FILE TYPE
```

```
013D  3D              DCR     A       ;1=REL. 0=ABS.
013E  78              MOV     A,B     ;GET OLD INFO
013F  C1              POP     B       ;RELOCATION FACTOR
0140  2003            JRNZ    ..A     ;MUST BE ABSOLUTE LOAD
0142  DD09            DADX    B       ;ELSE RELOCATE
0144  09              DAD     B       ; BOTH HL & X
0145  1C      ..A:    INR     E       ;TEST LENGTH
0146  1D              DCR     E       ;0=DONE
0147  2822            JRZ     DONE
0149  3D              DCR     A       ;TEST OLD INFO
014A  2824            JRZ     LODR    ;RELATIVE FILE
014C  CD 01A0   ..L1:  CALL    SBYTE   ;NEXT...
014F  CD 01C4          CALL    STORE   ;STORE IT
0152  20F8            JRNZ    ..L1    ;MORE COMING
0154  CD 01A0   LOD4:  CALL    SBYTE   ;GET CHECKSUM
0157  28C3             JRZ     LOD0    ;ALL O.K.
                ;
0159  AF        ERROR: XRA     A       ;FLASH ADDRESS & SENSE LINES
015A  2F               CMA
015B  D3FF             OUT     SENSE
015D  1B        ..SIT1: DCX    D
015E  7A               MOV     A,D
015F  B3               ORA     E
0160  20FB             JRNZ    ..SIT1
0162  D3FF             OUT     SENSE
0164  1B        ..SIT2: DCX    D
0165  7A               MOV     A,D
0166  B3               ORA     E
0167  20FB             JRNZ    ..SIT2
0169  18EE             JMPR    ERROR
                ;
                ;
016B  7C        DONE:  MOV     A,H
016C  B5               ORA     L       ;CAN'T GO TO ZERO
016D  28FE             JRZ     .       ;TIGHT LOOP HERE
016F  E9               PCHL            ;ELSE SIGN ON PROGRAM
                ;
0170  2E01      LODR:  MVI     L,1
0172  CD 0190   ..L1:  CALL    LODCB   ;GET CONTROL BYTE
0175  3807             JRC     ..L3    ;DOUBLE BIT
0177  CD 01C4   ..L5:  CALL    STORE   ;WRITE IT
017A  20F6             JRNZ    ..L1    ;MORE TO GO
017C  18D6             JMPR    LOD4    ;TEST CHECKSUM
                ;
017E  4F        ..L3:  MOV     C,A     ;LOW BYTE
017F  CD 0190          CALL    LODCB   ;NEXT
0182  47               MOV     B,A     ;HIGH BYTE
0183  D9               EXX
0184  C5               PUSH    B       ;GET RELOCATION
0185  D9               EXX
0186  E3               XTHL
0187  09               DAD     B
0188  7D               MOV     A,L     ;RELOCATE LOW BYTE
0189  CD 01C4          CALL    STORE   ;SAVE IT
018C  7C               MOV     A,H     ;RELOCATED HIGH BYTE
```

```
018D  E1              POP     H       ;RESTORE HL
018E  18E7            JMPR    ..L5    ;SAVE HIGH, REPEAT
                ;
0190  2D        LODCB:DCR     L       ;COUNT BITS
0191  2007           JRNZ    ..LC1   ;MORE LEFT
0193  CD 01A0         CALL    SBYTE   ;GET NEXT
0196  1D              DCR     E       ;COUNT BYTES
0197  67              MOV     H,A     ;SAVE THE BITS
0198  2E08            MVI     L,8     ;8 BITS/BYTE
019A  CD 01A0   ..LC1:CALL    SBYTE   ;GET A DATA BYTE
019D  CB24            SLAR    H       ;TEST NEXT BIT
019F  C9              RET
01A0  C5        SBYTE:PUSH    B       ;PRESERVE BC
01A1  CD 01B3         CALL    RIBBLE  ;GET 1/2 BYTE
01A4  07              RLC
01A5  07              RLC
01A6  07              RLC
01A7  07              RLC
01A8  4F              MOV     C,A     ;SAVE LEFT HALF
01A9  CD 01B3         CALL    RIBBLE  ;GET OTHER HALF
01AC  B1              ORA     C       ;MAKE WHOLE
01AD  4F              MOV     C,A     ;IN C
01AE  82              ADD     D       ;UPDATE CHECKSUM
01AF  57              MOV     D,A     ;NEW VALUE
01B0  79              MOV     A,C     ;CONVERTED BYTE
01B1  C1              POP     B
01B2  C9              RET
                ;
01B3  CD 01BE   RIBBLE:CALL   RDR
01B6  D630            SUI     '0'
01B8  FE0A            CPI     10
01BA  D8              RC
01BB  D607            SUI   'A'-'9'-1 ;ADJUST
01BD  C9              RET
                ;
01BE  CD 0020   RDR:  CALL    USER    ;USER WRITTEN ROUTINE AT 10H
01C1  E67F            ANI     7FH
01C3  C9              RET
                ;
01C4  DD7700    STORE:MOV     0(X),A  ;WRITE TO MEMORY
01C7  DDBE00          CMP     0(X)    ;VALID WRITE?
01CA  208D            JRNZ    ERROR   ; NO.
01CC  DD23            INX     X       ;ADVANCE POINTER
01CE  1D              DCR     E       ;DECREMENT COUNT
01CF  C9              RET
                ;
                .END
```

APPENDIX B. <*TDL RELOCATING LOADER, VERSION 3.2 - DEC. 28, 1976*>
+++++ SYMBOL TABLE +++++

BEGIN	0100	DONE	016B	ERROR	0159	HLMOD	001E
LOD0	011C	LOD4	0154	LODCB	0190	LODR	0170
RDR	01BE	RIBBLE	01B3	SBYTE	01A0	SENSE	00FF
STORE	01C4	TOP	0200	USER	0020		

ADDENDUM:

Here is a DUMP of the LOADER, Version 3.2. It may be used to insure proper loading after the boot part of the tape has been read. This should not be required unless you are having trouble loading the monitor.

Remember: The new format requires the monitor be loaded at 0200H minimum. We strongly urge that you load at 0F000H. If you still wish to locate the monitor between 0 and 0200H, first load a temporary copy up higher, and then use THAT one to load it elsewhere. This monitor runs ANYWHERE when loaded by a copy of itself, but when using an initial boot strap, it is forced to a page boundry. Running the monitor on other than a page border sounds a little pointless in any case.

addr	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0100	3E	C3	32	1D	00	21	00	01	22	1E	00	32	20	00	31	00
0110	02	DB	FF	FE	02	DA	59	01	47	0E	00	D9	CD	BE	01	D6
0120	3A	47	E6	FE	20	F6	57	CD	A0	01	5F	CD	A0	01	F5	CD
0130	A0	01	E1	6F	E5	DD	E1	D9	C5	D9	CD	A0	01	3D	78	C1
0140	20	03	DD	09	09	1C	1D	28	22	3D	28	24	CD	A0	01	CD
0150	C4	01	20	F8	CD	A0	01	28	C3	AF	2F	D3	FF	1B	7A	B3
0160	20	FB	D3	FF	1B	7A	B3	20	FB	18	EE	7C	B5	28	FE	E9
0170	2E	01	CD	90	01	38	07	CD	C4	01	20	F6	18	D6	4F	CD
0180	90	01	47	D9	C5	D9	E3	09	7D	CD	C4	01	7C	E1	18	E7
0190	2D	20	07	CD	A0	01	1D	67	2E	08	CD	A0	01	CB	24	C9
01A0	C5	CD	B3	01	07	07	07	07	4F	CD	B3	01	B1	4F	82	57
01B0	79	C1	C9	CD	BE	01	D6	30	FE	0A	D8	D6	07	C9	CD	20
01C0	00	E6	7F	C9	DD	77	00	DD	BE	00	20	8D	DD	23	1D	C9

C. COMMAND SET AND USAGE

The following are the commands and operating symbols of the ZAP Monitor.

COMMAND	DESCRIPTION
D	DISPLAY COMMAND - this command displays the contents of memory in base hex. Memory is displayed 16 bytes per line, with the starting address of the line given as the first information on the line. In use, first the command is given, then the starting address, the ending address and a carriage return. The form is: DØØ,FFF(cr). (This would display memory from ØØ to FFF.)
E	END OF FILE - this command outputs the end of file pattern for the checksum loader. It is used after punching a block of memory with a "W" command. An address parameter for the End of File may be given. For use, when the file being dumped is finished, type: E(cr).
F	FILL - This command fills a block of memory with a specific value. It is handy for initializing a block to a specific value (such as for tests, zeroing memory when starting up, etc.) In use, first the command, then the starting address, ending address, and the value to be entered, followed by a carriage return. The form is F1ØØØ,1FFF,AA(cr). This would fill the block 1ØØØ to 1FFF with AA.
G	GOTO - this command causes the processor to go to the specific address named and start executing. If a Return command is included in the program, the processor may jump back to the monitor after execution of the program. (RETURN is C9 hex). To use, the command is followed by the address chosen to execute from and a carriage return. The form is: G2FD4(cr). The processor will goto address 2FD4 and execute.

J — MEMORY TEST - this is a "hard" memory test which will locate bad bits and represent them in their binary form. It is not meant to be the definitive memory test, but rather serves as an aid. It can also serve to very quickly locate accidentally or mistakenly protected areas of memory. It is non destructive of the memory contained in the area being examined.

In use, the command is followed by starting and ending addresses. A read/complement/write is executed and if any errors are found, the bad address will be printed followed by the binary representation of the bit pattern. The form is: JØØ,FF(cr). If address AA were bad on its fourth bit, the processor will print back AA 00010000, the "1" representing the bad bit found.

L — LOAD A BINARY FILE - This reads a binary file, either from cassette or tape. The form is: LØØØ (cr). This would load a binary file starting at address 000. To use, enter the command and the starting address, type carriage return, and start the reader with nulls on the tape.

M — MOVE COMMAND - this command can move a block of memory from one location to another. This command should be used with some caution as careless placing could "smash" memory locations containing wanted data.

To use, type M followed by the starting address of the memory block to be moved, the ending address of the block to be moved, and the starting address of the new location. The form is: MØØ,AA,CC. This would move the block of memory starting at location ØØ and extending to location AA up to location CC.

N — NULL - this command may be used to print nulls on paper tape as a leader. To use simply type N - and nulls will be punched.

Q OUTPUT OR DISPLAY FROM/TO I/O PORTS - this command instructs the processor where to look for or where to send data to.To use, enter the command, indicating wether the processor is to input or output, name the port, and name the value to be output, if you are outputting. The form is: QOØ,AA or QIØ. The first would output an AA to port Ø, the second would input from port zero.

R READ CHECKSUMMED HEX FILE - this command reads the check-summed hex files for both the normal Intel format and the TDL relocating format. On both files, a "bias" (a shift in the address) may be added which will allow the object code to be placed in a location other than its intended execution location. The bias is added to what would have been the normal loading location and may wrap around. When used with the TDL relocating assembler, it allows generating a program to execute anywhere, and to be stored anywhere else in memory. When loading a relocatable file, an additional parameter may be added which represents the actual execution address desired. This may also be any location in memory.

To use, with a normal file, type R(cr) and start the reader.
With a relocating file, the following examples should clarify the use of bias.

R(cr) = Ø bias, Ø execution address
R1(cr) = 1 bias, Ø execution address
R,1(cr)= Ø bias, 1 execution address
R1,1(cr) = 1 bias, 1 execution address

S SINGLE BYTE INSPECT AND MODIFY - this command allows single bytes of memory to be examined and modified or not as the user desires.

To use, give the command followed by an address and push the space bar - the data at that address will be displayed followed by a "-". If you wish to change the data at that address, simply type in the new data in hex and press the space bar. The old data will be replaced, and then the next byte of data will appear. If you wish to retain the old data,

simply press the space bar and the next byte will appear. Typing a carriage return ends the sequence.

U — BINARY DUMP - this command simply dumps core to the punch device. It may be used with a cassette system as well, with no start-up problems. It does not generate checksum. The format which will be generated is a leader, 8-ØFFH's, and a trailer. The rub-outs are called file ques and are detected and counted to determine the start and end of files.
To use, type the command followed by the starting and ending addresses, start the reader and (cr). The form is: UØØ,FF(start reader - cr). This would generate a binary tape in the above format of the core contained in memory location ØØ to FF.

W — HEX DUMP - this routine dumps memory in the standard Intel-style hex file format. The start and end parameters are required and the End of File should be separately generated with the "E" command.
To use, enter the command, starting address, ending address, start the reader, (cr). When dump finished, type E(cr) to generate end of file. The form is: WØØ,FF(start punch - cr) ----E(cr). (N here is optional).

Z — TOP OF MEMORY - this command locates and names the top byte of RAM in the system. It does not include the space the monitor is occupying. Simply type Z - no (cr) is needed. The top of memory will be displayed in hex.

H — HEXIDECIMAL MATH - this command allows hex addition and subtraction to be executed.
To use, type H, and the two hex figures to be added and subtracted. The form is:H00,11(cr). The computer will print out first the hex sum and then the hex difference, in hex.

This concludes the command set of the ZAP Monitor.

In addition to these commands there are two symbols which you will observe. The first is an *, which is an error message. The second is a > (greater than) which is a prompter basically saying "OK, continue...".

To interrupt a routine such as a D or J command, just type a CONTROL C. This ends the routine.

D. ZPU FINAL CHECKOUT USING MONITOR

Assembly and electrical checkout of the ZPU was conducted elsewhere. However, only operation will show if the ZPU is actually operating correctly. The monitor is the best means of achieving this. Load the monitor as per the preceeding instructions, and experiment with its various commands. The FILL and DISPLAY, plus MOVE and J commands provide good exercise for the processor and if they seem to function normally, all is probably well.

E. SOURCE LISTING

The following pages are an "off the printer" copy of the ZAP Monitor source code. It is provided for your understanding, plus as an invitation to experiment with Z-80 programming which can be quite exciting given 696 opcodes.

```
                        ;       << ZAP 1-K MONITOR SYSTEM >>
                        ;               by
                        ;
                        ;       TECHNICAL DESIGN LABS, INC.
                        ;       RESEARCH PARK
                        ;       PRINCETON, NEW JERSEY 08540
                        ;
                        ;       COPYRIGHT JAN. 1977 TDL INC.
                        ;
                        ;       ASSEMBLED by Roger Amidon
                        ;
                        .PREL   ;THIS MONITOR SUPPLIED IN RELOCATING FORMAT
                        ;
0400'                   LENGTH  = Z     ;SIZE OF THIS MONITOR
                        ;
                        .TITLE "     <Zap Monitor, Version 2.0, Jan. 16 1977>"
                        .SBTTL  /  Copyright 1977 by TECHNICAL DESIGN LABS, INC./
                        ;
                        ;       <I/O DEVICES>
                        ;
                        ;-TELEPRINTER
                        ;
0001                    TTI     = 1     ;DATA IN PORT
0001                    TTO     = 1     ;DATA OUT PORT
0000                    TTS     = 0     ;STATUS PORT (IN)
0001                    TTYDA   = 1     ;DATA AVAILABLE MASK BIT
0080                    TTYBE   = 80H   ;XMTR BUFFER EMPTY MASK
                        ;
0003                    RCP     = 3     ;READER CONTROL PORT.
                                        ;THIS PORT IS PULSED ONCE
                                        ;FOR EACH READER REQUEST
                                        ;TO SUPPORT A CONTROLLED
                                        ;READER.
                        ;
                        ;       <CONSTANTS>
                        ;
0000                    I       = 0             ;'I' REG. VALUE
0000                    FALSE   = 0             ;ISN'T SO
FFFF                    TRUE    = # FALSE       ;IT IS SO
000D                    CR      = 0DH           ;ASCII CARRIAGE RETURN
000A                    LF      = 0AH           ;ASCII LINE FEED
0007                    BELL    = 7             ;DING
00FF                    RUB     = 0FFH          ;RUB OUT
0000                    FIL     = 00            ;FILL CHARACTERS AFTER CRLF
0007                    MAX     = 7             ;NUMBER OF QUES IN EOF
                        ;
                        ;
                        ;       PROGRAM CODE BEGINS HERE
                        ;
0000' C3 0308'          ZAP:    JMP     BEGIN   ;GO AROUND VECTORS
                                                ; GET MEMORY SIZE,
                                                ; AND CONTINUE AHEAD
                        ;
```

```
                        ;
                        ;       <VECTORS FOR CALLING PROGRAMS>
                        ;
                        ; THESE VECTORS MAY BE USED BY USER WRITTEN
                        ; PROGRAMS TO SIMPLIFY THE HANDLING OF I/O
                        ; FROM SYSTEM TO SYSTEM.  WHATEVER THE CURRENT
                        ; ASSIGNED DEVICE, THESE VECTORS WILL PERFORM
                        ; THE REQUIRED I/O OPERATION, AND RETURN TO
                        ; THE CALLING PROGRAM. (RET)
                        ;
                        ; THE REGISTER CONVENTION USED FOLLOWS-
                        ;
                        ; ANY  INPUT OR OUTPUT DEVICE-
                        ;       CHARACTER TO BE OUTPUT IN 'C' REGISTER.
                        ;       CHARACTER WILL BE IN 'A' REGISTER UPON
                        ;       RETURNING FROM AN INPUT OR OUTPUT.
                        ; 'CSTS'-
                        ;       RETURNS TRUE (OFFH IN 'A' REG.) IF THERE IS
                        ;       SOMETHING WAITING, AND ZERO (00) IF NOT.
                        ; 'IOCHK'-
                        ;       RETURNS WITH THE CURRENT I/O CONFIGURATION
                        ;       BYTE IN 'A' REGISTER.
                        ; 'IOSET'-
                        ;       I/O CANNOT BE MODIFIED IN THIS 1K VERSION
                        ; 'MEMCK'-
                        ;       RETURNS WITH THE HIGHEST ALLOWED USER
                        ;       MEMORY LOCATION. 'B'=HIGH BYTE, 'A'=LOW.
                        ; 'TRAP'-
                        ;       THIS IS THE 'BREAKPOINT' ENTRY POINT.
                        ;       NOT USED IN THE 1K VERSION, GOES TO THE
                        ;       'ERROR' ROUTINE TO RESET THE MONITOR'S
                        ;       STACK.
                        ;
0003' C3 0374'                  JMP     CI      ;CONSOLE INPUT
0006' C3 037D'                  JMP     RI      ;READER INPUT
0009' C3 0222'                  JMP     CO      ;CONSOLE OUTPUT
000C' C3 0233'                  JMP     PO      ;PUNCH OUTPUT
000F' C3 0222'                  JMP     CO      ;LIST OUTPUT
0012' C3 0282'                  JMP     CSTS    ;CONSOLE STATUS
0015' 3E00                      MVI     A,0     ; I/O CHECK
0017' C9                IOSET:  RET             ;SET TO TTY CONFIGURATION
0018' C3 0017'                  JMP     IOSET   ;CAN'T SET I/O ON 1K VERSION
001B' C3 02FF'                  JMP     MEMCK   ;MEMORY LIMIT CHECK
001E' CD 0313'          ERROR:  CALL    MEMSIZ  ;RESET BACK TO MONITOR (TRAP)
0021' F9                        SPHL            ;RE-ESTABLISH A STACK
0022' 0E2A                      MVI     C,'*'   ;ANNOUNCE ERROR
0024' CD 0222'                  CALL    CO
0027' 1815                      JMPR    START
```

```
                        ;
                        ;       MONITOR NAME & VERSION
                        ;
0029' 0D0A000000 MSG:   .BYTE   CR,LF,FIL,FIL,FIL
002E' 5A61702056        .ASCII  'Zap V'
0033' 322E30            .ASCII  '2.0'
                        ;
000D                    MSGL    = .-MSG
                        ;
0034'                   STACK   = .-2           ;A FAKE STACK TO GET STARTED
                        ;
0036' 0038'             .WORD   AHEAD           ;AFTER MEMORY SIZE
                        ;
0038' F9        AHEAD:  SPHL                    ;SET TRUE STACK
0039' 060D              MVI     B,MSGL          ;SAY HELLO TO THE FOLKS
003B' CD 01F2'          CALL    TOM             ;OUTPUT SIGN-ON MSG
003E' 0E3E      START:  MVI     C,'>'           ;PROMPT CHARACTER
0040' 21 003E'          LXI     H,START         ;MAIN 'WORK' LOOP
0043' E5                PUSH    H               ;SET UP A RETURN TO HERE
0044' CD 0278'          CALL    CRLF
0047' CD 0222'          CALL    CO
004A' CD 03DC'  STARO:  CALL    TI              ;GET A CONSOLE CHARACTER
004D' E67F              ANI     7FH             ;IGNORE NULLS
004F' 28F9              JRZ     STARO           ;GET ANOTHER
0051' 0E02              MVI     C,2             ;SET-UP C REG.
0053' FE44              CPI     'D'             ;SEE IF 'DISPLAY' COMMAND
0055' 2017              JRNZ    EOF
                        ;
                        ; THIS DISPLAYS THE CONTENTS OF MEMORY IN BASE HEX
                        ; WITH THE STARTING LOCATION ON EACH LINE.(BETWEEN
                        ; THE TWO PARAMETERS GIVEN). 16 BYTES PER LINE MAX.
                        ;
0057' CD 0273'  DISP:   CALL    EXLF            ;GET DISPLAY RANGE
005A' CD 021A'  ..DO:   CALL    LFADR           ;CRLF & PRINT ADDR.
005D' CD 0220'  ..DI:   CALL    BLK             ;SPACE OVER
0060' 7E                MOV     A,M
0061' CD 02E3'          CALL    LBYTE
0064' CD 02BD'          CALL    HILOX           ;RANGE CHECK
0067' 7D                MOV     A,L
0068' E60F              ANI     0FH             ;SEE IF TIME TO CRLF
006A' 20F1              JRNZ    ..DI
006C' 18EC              JMPR    ..DO
                        ;
                        ; THIS OUTPUTS THE END OF FILE (EOF) PATTERN
                        ; FOR THE CHECKSUM LOADER. IT IS USED AFTER
                        ; PUNCHING A BLOCK OF MEMORY WITH THE 'W'
                        ; COMMAND.  AN ADDRESS PARAMETER MAY BE GIVEN,
                        ; WHICH WILL BE INCLUDED IN THE END FILE.
                        ;
006E' FE45      EOF:    CPI     'E'             ;SEE IF 'EOF'
0070' 201A              JRNZ    FILL
0072' CD 0296'          CALL    EXPRI           ;GET OPTIONAL ADDR.
0075' CD 022C'          CALL    PEOL            ;CRLF TO PUNCH
0078' 0E3A              MVI     C,':'           ;FILE MARKER CUE
007A' CD 0233'          CALL    PO
```

```
007D' AF                XRA     A       ;ZERO LENGTH
007E' CD 034D'          CALL    PBYTE
0081' E1                POP     H
0082' CD 0348'          CALL    PADR    ;PUNCH OPTIONAL ADDR.
0085' AF                XRA     A       ;FILE TYPE=0
0086' CD 034D'          CALL    PBYTE   ;PUNCH IT
0089' C3 025F'          JMP     NULL    ;TRAILER & RETURN
                        ;
                        ; THIS COMMAND WILL FILL A BLOCK OF MEMORY
                        ; WITH A VALUE. IE; F0,1FFF,0  FILLS FROM
                        ; <1> TO <2> WITH THE BYTE <3>. HANDY FOR
                        ; INITIALIZING A BLOCK TO A SPECIFIC VALUE, OR
                        ; MEMORY TO A CONSTANT VALUE BEFORE LOADING
                        ; A PROGRAM. (ZERO IS ESPECIALLY USEFUL.)
                        ;
008C' FE46      FILL:   CPI     'F'     ;SEE IF 'FILL'
008E' 200C              JRNZ    GOTO
0090' CD 028B'          CALL    EXPR3   ;GET 3 PARAMETERS
0093' 71        ..F:    MOV     M,C     ;STORE THE BYTE
0094' CD 02C3'          CALL    HILO
0097' 30FA              JRNC    ..F
0099' D1                POP     D       ;RESTORE STACK
009A' 18A2              JMPR    START   ; IN CASE OF ACCIDENTS
                        ;
                        ; THIS COMMAND ALLOWS EXECUTION OF ANOTHER
                        ; PROGRAM.
                        ;
009C' FE47      GOTO:   CPI     'G'     ;SEE IF 'GOTO'
009E' 2006              JRNZ    TEST
00A0' CD 0296'          CALL    EXPR1   ;GET AN ADDRESS TO GO TO
00A3' C3 0278'          JMP     CRLF    ;CRLF & EXECUTE
                        ;
                        ; 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> 04" WHERE, IN THIS PARTICULAR
                        ; EXAMPLE, BIT 2 IS THE BAD BIT.
                        ; BIT LOCATION OF THE FAILURE IS EASILY
                        ; DETERMINED. NON-R/W MEMORY WILL DISPLAY
                        ; <ADDR>  FF  (ALL BITS BAD)
                        ;
00A6' FE4A      TEST:   CPI     'J'     ;SEE IF 'TEST'
00A8' 201B              JRNZ    MOVE
00AA' CD 0273'          CALL    EXLF    ;GET TWO PARAMS
00AD' 7E        ..T1:   MOV     A,M     ;READ A BYTE
00AE' 47                MOV     B,A     ;SAVE IN B REG.
00AF' 2F                CMA
00B0' 77                MOV     M,A     ;READ/COMPLIMENT/WRITE
00B1' AE                XRA     M       ; & COMPARE
00B2' 280B              JRZ     ..T2    ;SKIP IF ZERO (OK)
00B4' 08                EXAF            ;SAVE BAD BYTE
00B5' CD 021D'          CALL    HLSP    ;PRINT BAD ADDR
```

```
00B8' 08                EXAF            ;GET BAD BYTE BACK
00B9' CD 02E3'          CALL    LBYTE   ;PRINT IT
00BC' CD 0278'          CALL    CRLF
00BF' 70        ..T2:   MOV     M,B     ;REPLACE BYTE
00C0' CD 02BD'          CALL    HILOX   ;RANGE TEST
00C3' 18E8              JMPR    ..T1
                ;
                ; THIS COMMAND MOVES MASS AMOUNTS OF MEMORY
                ; FROM <1> THRU <2> TO THE ADDRESS STARTING
                ; AT <3>.  THIS ROUTINE SHOULD BE USED WITH
                ; SOME CAUTION, AS IT COULD SMASH MEMORY IF
                ; CARELESSLY IMPLEMENTED.
                ;
                ;       M<1>,<2>,<3>
                ;
00C5' FE4D      MOVE:   CPI     'M'     ;SEE IF 'MOVE'
00C7' 200B              JRNZ    READ
00C9' CD 028B'          CALL    EXPR3   ;GET 3 PARAMETERS
00CC' 7E        ..M:    MOV     A,M     ;PICK UP
00CD' 02                STAX    B       ;PUT DOWN
00CE' 03                INX     B       ;MOVE UP
00CF' CD 02BD'          CALL    HILOX   ;CHECK IF DONE
00D2' 18F8              JMPR    ..M
                ;
                ; THIS COMMAND READS THE CHECK-SUMMED HEX FILES
                ; FOR BOTH THE NORMAL INTEL FORMAT AND THE TDL
                ; RELOCATING FORMAT. ON BOTH FILES, A 'BIAS' MAY
                ; BE ADDED, WHICH WILL CAUSE THE OBJECT CODE TO
                ; BE PLACED IN A LOCATION OTHER THAN ITS
                ; INTENDED EXECUTION LOCATION. THE BIAS IS ADDED TO
                ; WHAT WOULD HAVE BEEN THE NORMAL LOADING
                ; LOCATION, AND WILL WRAP AROUND TO ENABLE
                ; LOADING ANY PROGRAM ANYWHERE IN MEMORY.
                ;
                ; WHEN LOADING A RELOCATABLE FILE, AN ADDITIONAL
                ; PARAMETER MAY BE ADDED, WHICH REPRESENTS THE
                ; ACTUAL EXECUTION ADDRESS DESIRED. THIS ALSO MAY
                ; BE ANY LOCATION IN MEMORY.
                ;
                ; EXAMPLES:
                ;
                ; R[CR] =0 BIAS, 0 EXECUTION ADDR.
                ; R<ADDR1>[CR] =<1>BIAS, 0 EXECUTION ADDR.
                ; R,<ADDR1>[CR] =0 BIAS, <1> EXECUTION ADDR.
                ; R<ADDR1>,<ADDR2>[CR] =<1>BIAS, <2> EXECUTION ADDR.
                ;
00D4' FE52      READ:   CPI     'R'     ;SEE IF 'READ' COMMAND
00D6' C2 017C'          JNZ     SUBS
00D9' CD 0296'          CALL    EXPR1   ;GET BIAS, IF ANY
00DC' 78                MOV     A,B     ;LOOK AT DELIMITER
00DD' D60D              SUI     CR      ;ALL DONE?
00DF' 47                MOV     B,A     ;SET UP RELOCATION OF 0
00E0' 4F                MOV     C,A     ; IF CR ENTERED
00E1' D1                POP     D       ;BIAS AMOUNT
00E2' 2804              JRZ     ..R0    ;CR ENTERED
```

```
00E4' CD 0296'          CALL    EXPR1   ;GET RELOCATION
00E7' C1                POP     B       ;ACTUAL RELOCATION VALUE
00E8' EB        ..R0:   XCHG
00E9' D9                EXX             ;HL'=BIAS, BC'=RELOCATION
00EA' CD 0278'          CALL    CRLF
00ED' CD 020C'  LOD0:   CALL    RIFF    ;GET A CHARACTER
00F0' E67F              ANI     7FH     ;KILL PARITY BIT
00F2' D63A              SUI     ':'     ;ABSOLUTE FILE CUE?
00F4' 47                MOV     B,A     ;SAVE CUE CLUE
00F5' E6FE              ANI     0FEH    ;KILL BIT 0
00F7' 20F4              JRNZ    LOD0    ; NO, KEEP LOOKING
00F9' 57                MOV     D,A     ;ZERO CHECKSUM
00FA' CD 0162'          CALL    SBYTE   ;GET FILE LENGTH
00FD' 5F                MOV     E,A     ;SAVE IN E REG.
00FE' CD 0162'          CALL    SBYTE   ;GET LOAD MSB
0101' F5                PUSH    PSW     ;SAVE IT
0102' CD 0162'          CALL    SBYTE   ;GET LOAD LSB
0105' D9                EXX             ;CHANGE GEARS
0106' D1                POP     D       ;RECOVER MSB
0107' 5F                MOV     E,A     ;FULL LOAD ADDR
0108' C5                PUSH    B       ;BC'=RELOCATION
0109' D5                PUSH    D       ;DE'=LOAD ADDR
010A' E5                PUSH    H       ; HL'=BIAS
010B' 19                DAD     D       ; BIAS+LOAD
010C' E3                XTHL            ;RESTORE HL'
010D' DDE1              POP     X       ; X=BIAS+LOAD
010F' D9                EXX             ;DOWNSHIFT
0110' E1                POP     H       ;HL=LOAD ADDR
0111' CD 0162'          CALL    SBYTE   ;GET FILE TYPE
0114' 3D                DCR     A       ;1=REL. FILE, 0=ABS.
0115' 78                MOV     A,B     ;SAVE CUE BIT
0116' C1                POP     B       ;BC=RELOCATION
0117' 2003              JRNZ    ..A     ;ABSOLUTE FILE
0119' 09                DAD     B       ;ELSE RELOCATE
011A' DD09              DADX    B       ;BOTH X & HL
011C' 1C        ..A:    INR     E       ;TEST LENGTH
011D' 1D                DCR     E       ;0=DONE
011E' C8                RZ
011F' 3D                DCR     A       ;TEST CUE
0120' 2810              JRZ     LODR    ;RELATIVE
0122' CD 0162'  ..L1:   CALL    SBYTE   ;NEXT
0125' CD 0175'          CALL    STORE   ;STORE IT
0128' 20F8              JRNZ    ..L1    ;MORE COMING
012A' CD 0162'  LOD4:   CALL    SBYTE   ;GET CHECKSUM
012D' 28BE              JRZ     LOD0    ;GOOD CHECKSUM
012F' C3 001E'          JMP     ERROR   ;BAD, ABORT
0132' 2E01      LODR:   MVI     L,1     ;SET-UP BIT COUNTER
0134' CD 0152'  ..L1:   CALL    LODCB   ;GET THE BIT
0137' 3807              JRC     ..L3    ;DOUBLE BIT
0139' CD 0175'  ..L5:   CALL    STORE   ;WRITE IT
013C' 20F6              JRNZ    ..L1
013E' 18EA              JMPR    LOD4    ;TEST CHECKSUM
0140' 4F        ..L3:   MOV     C,A     ;SAVE LOW BYTE
0141' CD 0152'          CALL    LODCB   ;NEXT CONTROL BIT
0144' 47                MOV     B,A     ;SAVE HIGH BYTE
```

```
0145' D9                    EXX
0146' C5                    PUSH    B           ;GET RELOCATION
0147' D9                    EXX
0148' E3                    XTHL                ;INTO HL
0149' 09                    DAD     B           ;RELOCATE
014A' 7D                    MOV     A,L         ;LOW BYTE
014B' CD 0175'              CALL    STORE       ;STORE IT
014E' 7C                    MOV     A,H         ;HIGH BYTE
014F' E1                    POP     H           ;RESTORE HL
0150' 18E7                  JMPR    ..L5        ;DO THIS AGAIN
0152' 2D            LODCB:  DCR     L           ;COUNT BITS
0153' 2007                  JRNZ    ..LC1       ;MORE LEFT
0155' CD 0162'              CALL    SBYTE       ;GET NEXT
0158' 1D                    DCR     E           ;COUNT BYTES
0159' 67                    MOV     H,A         ;SAVE THE BITS
015A' 2E08                  MVI     L,8         ;8 BITS/BYTE
015C' CD 0162'      ..LC1:  CALL    SBYTE       ;GET A DATA BYTE
015F' CB24                  SLAR    H           ;TEST NEXT BIT
0161' C9                    RET
0162' C5            SBYTE:  PUSH    B           ;PRESERVE BC
0163' CD 0333'              CALL    RIBBLE      ;GET A CONVERTED ASCII CHAR.
0166' 07                    RLC
0167' 07                    RLC
0168' 07                    RLC
0169' 07                    RLC                 ;MOVE IT TO HIGH NIBBLE
016A' 4F                    MOV     C,A         ;SAVE IT
016B' CD 0333'              CALL    RIBBLE      ;GET OTHER HALF
016E' B1                    ORA     C           ;MAKE WHOLE
016F' 4F                    MOV     C,A         ;SAVE AGAIN IN C
0170' 82                    ADD     D           ;UPDATE CHECKSUM
0171' 57                    MOV     D,A         ;NEW CHECKSUM
0172' 79                    MOV     A,C         ;CONVERTED BYTE
0173' C1                    POP     B
0174' C9                    RET
0175' DD7700        STORE:  MOV     0(X),A      ;WRITE TO MEMORY
0178' DD23                  INX     X           ;ADVANCE POINTER
017A' 1D                    DCR     E           ;COUNT DOWN
017B' C9                    RET
                    ;
                    ; THIS ROUTINE ALLOWS BOTH INSPECTION OF &
                    ; MODIFICATION OF MEMORY ON A BYTE BY BYTE
                    ; BASIS. IT TAKES ONE ADDRESS PARAMETER,
                    ; FOLLOWED BY A SPACE.  THE DATA AT THAT
                    ; LOCATION WILL BE DISPLAYED. IF IT IS
                    ; DESIRED TO CHANGE IT, THE VALUE IS THEN
                    ; ENTERED.  A FOLLOWING SPACE WILL DISPLAY
                    ; THE NEXT BYTE.  A CARRIAGE RETURN [CR]
                    ; WILL TERMINATE THE COMMAND.  THE SYSTEM
                    ; ADDS A CRLF AT LOCATIONS ENDING WITH EITHER
                    ; XXX0 OR XXX8. TO AID IN DETERMINING THE
                    ; PRESENT ADDRESS, IT IS PRINTED AFTER
                    ; EACH CRLF.  A BACKARROW [_] WILL BACK
                    ; UP THE POINTER AND DISPLAY THE
                    ; PREVIOUS LOCATION.
                    ;
```

```
017C' FE53      SUBS:   CPI     'S'     ;SEE IF 'SUBSTITUTE'
017E' 202E              JRNZ    WRITE
0180' CD 0296'          CALL    EXPRI   ;GET STARTING ADDR.
0183' E1                POP     H
0184' 7E        ..S0:   MOV     A,M
0185' CD 02E3'          CALL    LBYTE   ;DISPLAY THE BYTE
0188' CD 0360'          CALL    COPCK   ;MODIFY?
018B' D8                RC              ; NO, ALL DONE
018C' 2814              JRZ     ..S1    ;DON'T MODIFY
018E' FE5F              CPI     '_'     ;BACKUP?
0190' 2819              JRZ     ..S2
0192' E5                PUSH    H       ;SAVE POINTER
0193' 0E01              MVI     C,1
0195' 21 0000           LXI     H,0
0198' CD 029E'          CALL    E>1     ;GET NEW VALUE
019B' D1                POP     D       ;VALUE IN E
019C' E1                POP     H
019D' 73                MOV     M,E     ;MODIFY
019E' 78                MOV     A,B     ;TEST DELIMITER
019F' FE0D              CPI     CR
01A1' C8                RZ              ;DONE
01A2' 23        ..S1:   INX     H
01A3' 7D        ..S3:   MOV     A,L     ;SEE IF TIME TO CRLF
01A4' E607              ANI     7
01A6' CC 021A'          CZ      LFADR   ;TIME TO CRLF
01A9' 18D9              JMPR    ..S0
01AB' 2B        ..S2:   DCX     H       ;DECREMENT POINTER
01AC' 18F5              JMPR    ..S3    ;AND PRINT DATA THERE.
                ;
                ;
                ; THIS ROUTINE DUMPS MEMORY IN THE STANDARD
                ; INTEL HEX-FILE FORMAT.  A START & END
                ; PARAMETER IS REQUIRED. AT THE CONCLUSION
                ; OF THE DUMP, AN "END OF FILE" SHOULD BE
                ; GENERATED WITH THE "E" COMMAND.
                ;
01AE' FE57      WRITE:  CPI     'W'     ;SEE IF 'WRITE' COMMAND
01B0' 2061              JRNZ    SIZE
01B2' CD 0273'          CALL    EXLF    ;GET TWO PARAMETERS
01B5' CD 0374'          CALL    CI      ;PAUSE FOR PUNCH-ON
01B8' CD 022C'  ..W0:   CALL    PEOL    ;CRLF TO PUNCH
01BB' 01 003A           LXI     B,':'   ;START-OF-FILE CUE
01BE' CD 0233'          CALL    PO      ;PUNCH IT
01C1' D5                PUSH    D       ;SAVE
01C2' E5                PUSH    H       ; POINTERS
01C3' 04        ..W1:   INR     B       ;CALCULATE FILE LENGTH
01C4' CD 02C3'          CALL    HILO
01C7' 3824              JRC     ..W4    ;SHORT FILE
01C9' 3E18              MVI     A,24    ;24 BYTES PER FILE
01CB' 90                SUB     B       ;ENOUGH YET?
01CC' 20F5              JRNZ    ..W1    ; NO.
01CE' E1                POP     H       ;GET START ADDR BACK.
01CF' CD 01D5'          CALL    ..W2    ;SEND THE BLOCK
01D2' D1                POP     D       ;RESTORE END OF FILE POINTER
01D3' 18E3              JMPR    ..W0    ;KEEP GOING
```

```
01D5' 57          ..W2:   MOV     D,A     ;INITIALIZE CHECKSUM
01D6' 78                  MOV     A,B     ;FILE LENGTH
01D7' CD 034D'            CALL    PBYTE   ;PUNCH IT
01DA' CD 0348'            CALL    PADR    ;PUNCH ADDRESS
01DD' AF                  XRA     A       ;FILE TYPE=0
01DE' CD 034D'            CALL    PBYTE   ;PUNCH IT
01E1' 7E          ..W3:   MOV     A,M     ;GET A DATA BYTE
01E2' CD 034D'            CALL    PBYTE   ;PUNCH IT
01E5' 23                  INX     H       ;POINT TO NEXT BYTE
01E6' 10F9                DJNZ    ..W3    ;DECREMENT FILE COUNT
01E8' AF                  XRA     A
01E9' 92                  SUB     D       ;CALCULATE CHECKSUM
01EA' C3 034D'            JMP     PBYTE   ;PUNCH IT, RETURN
01ED' E1          ..W4:   POP     H       ;CLEAR STACK
01EE' D1                  POP     D       ; OF POINTERS
01EF' AF                  XRA     A       ;SET-UP A
01F0' 18E3                JMPR    ..W2    ;FINISH UP & RETURN
                  ;
                  ;
                  ; THIS IS A MESSAGE OUTPUT ROUTINE.
                  ; IT IS USED BY THE SIGN-ON AND CRLF.
                  ; POINTER IS IN HL (WHEN ENTERED AT
                  ; TOM1) AND LENGTH IN B REG.
                  ;
01F2' 21 0029'    TOM:    LXI     H,MSG
01F5' 4E          TOM1:   MOV     C,M     ;GET A CHARACTER
01F6' 23                  INX     H       ;MOVE POINTER
01F7' CD 0222'            CALL    CO      ;OUTPUT IT
01FA' 10F9                DJNZ    TOM1    ;KEEP GOING TILL B=0
01FC' CD 0282'            CALL    CSTS    ;SEE IF AN ABORT REQUEST
01FF' B7                  ORA     A       ; WAITING.
0200' C8                  RZ              ;NO.
                  ;
                  ; SEE IF CONTROL-C IS WAITING
                  ; ABORT IF SO.
                  ;
0201' CD 0374'            CALL    CI
0204' E67F                ANI     7FH     ;KILL PARITY BIT
0206' FE03                CPI     3       ;CONTROL-C?
0208' C0                  RNZ
                  ;
0209' C3 001E'    ERRX:   JMP     ERROR
                  ;
                  ;
                  ; THIS GETS A READER CHARACTER,
                  ; AND COMAPRES IT WITH 'D' REG.
                  ; IT ABORTS ON AN 'OUT-OF-DATA'
                  ; CONDITION.
                  ;
020C' CD 037D'    RIFF:   CALL    RI      ;GET READER CHARACTER
020F' 38F8                JRC     ERRX    ;ABORT ON CARRY
0211' BA                  CMP     D       ;TEST D
0212' C9                  RET
                  ;
                  ; THIS ROUTINE WILL RETURN THE
```

```
                        ; CURRENT VALUE OF THE HIGHEST
                        ; READ/WRITE MEMORY LOCATION THAT
                        ; IS AVAILABLE ON THE SYSTEM.
                        ; IT WILL "SEARCH" FOR MEMORY
                        ; STARTING AT THE BOTTOM OF MEMORY
                        ; AND GO UPWARDS UNTIL NON-R/W MEMORY
                        ; IS FOUND.
                        ;
0213' FE5A              SIZE:   CPI     'Z'     ;SEE IF 'SIZE' COMMAND
0215' 2026                      JRNZ    UNLD
0217' CD 0313'                  CALL    MEMSIZ  ;GET THE VALUE
                        ;
                        ;
                        ; CRLF BEFORE HLSP ROUTINE
                        ;
021A' CD 0278'          LFADR:  CALL    CRLF
                        ;
                        ; PRINT THE CURRENT VALUE OF H&L,
                        ; AND A SPACE.
                        ;
021D' CD 02DE'          HLSP:   CALL    LADR
                        ;
                        ; PRINT A SPACE ON THE CONSOLE
                        ;
0220' 0E20              BLK:    MVI     C,' '
                        ;
                        ; THIS IS THE MAIN CONSOLE
                        ; OUTPUT ROUTINE.
                        ; TELEPRINTER CONFIGURATION
                        ; I/O DRIVER.
                        ;
0222' DB00              CO:     IN      TTS
0224' E680                      ANI     TTYBE
0226' 20FA                      JRNZ    CO
0228' 79                        MOV     A,C
0229' D301                      OUT     TTO
022B' C9                        RET
                        ;
                        ; SEND CRLF TO PUNCH DEVICE
                        ;
022C' 0E0D              PEOL:   MVI     C,CR
022E' CD 0233'                  CALL    PO
0231' 0E0A                      MVI     C,LF
                        ;
                        ; THIS IS THE 'PUNCH' OUTPUT
                        ; DRIVER. IT IS SET UP FOR THE
                        ; TTY PORTS, BUT MAY BE MODIFIED
                        ; FOR ANOTHER PORT, FOR TRUE
                        ; SEPARATION OF THE CONSOLE
                        ; AND READER/PUNCH DEVICES.
                        ;
                        ; (I.E. - PORT 6 & 7 FOR CASSETTE, ETC.)
                        ;
0233' DB00              PO:     IN      TTS     ;STATUS PORT
0235' E680                      ANI     TTYBE   ;TRANSMITTER BUFFER EMPTY?
```

```
0237' 20FA              JRNZ    PO      ;IF NOT, LOOP.
0239' 79                MOV     A,C     ;GET CHARACTER TO OUTPUT
023A' D301              OUT     TTO     ;TO DATA PORT
023C' C9                RET             ;DONE
                        ;
                        ; THIS IS A BINARY DUMP ROUTINE THAT MAY BE
                        ; USED WITH BOTH PAPER-TAPE AND/OR CASSETTE
                        ; SYSTEMS.  IT PUNCHES A START-OF-FILE MARK
                        ; AND THEN PUNCHES IN FULL 8-BITS DIRECTLY
                        ; FROM MEMORY.  IT IS FOLLOWED BY AN END-OF-
                        ; FILE MARKER. THESE DUMPS MAY BE LOADED
                        ; USING THE "L" COMMAND. THEY ARE USEFUL
                        ; FOR FAST LOADING.
                        ;
                        ;   U<A1>,<A2>[CR]
                        ; PUNCHES FROM <A1> THRU <A2>
                        ;
023D' FE55      UNLD:   CPI     'U'     ;SEE IF 'UNLOAD' COMMAND
023F' 201A              JRNZ    NULLX
0241' CD 0273'          CALL    EXLF    ;GET TWO PARAMETERS
0244' CD 0374'          CALL    CI      ;PAUSE FOR PUNCH-ON (TTY)
0247' CD 02F6'          CALL    LEAD    ;PUNCH LEADER
024A' CD 02F1'          CALL    MARK    ;PUNCH FILE MARKER
024D' 4E        ..U:    MOV     C,M     ;GET MEMORY BYTE
024E' CD 0233'          CALL    PO      ;PUNCH IT
0251' CD 02C3'          CALL    HILO    ;SEE IF DONE
0254' 30F7              JRNC    ..U
0256' CD 02F1'          CALL    MARK    ;PUNCH END FILE MARKER
0259' 1804              JMPR    NULL
                        ;
                        ; THIS PUNCHES NULLS (LEADER/TRAILER).
                        ; IT RETURNS "QUIET"
                        ;
025B' FE4E      NULLX:  CPI     'N'     ;SEE IF 'NULL'
025D' 206E              JRNZ    HEXN
025F' CD 02F6'  NULL:   CALL    LEAD    ;PUNCH NULLS
0262' C3 004A'          JMP     START0  ;RETURN QUIET
                        ;
                        ; CONVERT HEX TO ASCII
                        ;
0265' 0F        CBYTE:  RRC
0266' 0F                RRC
0267' 0F                RRC
0268' 0F                RRC
                        ;
0269' E60F      CONV:   ANI     0FH     ;LOW NIBBLE ONLY
026B' C690              ADI     90H
026D' 27                DAA
026E' CE40              ACI     40H
0270' 27                DAA
0271' 4F                MOV     C,A
0272' C9                RET
                        ;
                        ; GET TWO PARAMETERS, PLACE
                        ; THEM IN DE & HL, AND THEN
```

```
                        ; CRLF.
                        ;
0273' CD 0298'          EXLF:   CALL    EXPR
0276' D1                        POP     D
0277' E1                        POP     H
                        ;
                        ; CONSOLE CARRIAGE RETURN &
                        ; LINE FEED ROUTINE.
                        ;
                        ; THE NUMBER OF FILL CHARACTERS
                        ; MAY BE ADJUSTED TO 0-3 BY THE
                        ; VALUE PLACED IN THE B REG. MINIMUM
                        ; VALUE FOR "B" IS TWO (2). MAXIMUM
                        ; IS FIVE (5).
                        ;
0278' E5                CRLF:   PUSH    H       ;SAVE HL
0279' C5                        PUSH    B       ; & BC
027A' 0604                      MVI     B,4     ;CRLF LENGTH (SET FOR 2 FILLS)
027C' CD 01F2'                  CALL    TOM     ;SEND CRLF
027F' C1                        POP     B
0280' E1                        POP     H
0281' C9                        RET
                        ;
                        ; TEST THE CONSOLE'S
                        ; KEYBOARD FOR A KEY-PRESS.
                        ; RETURN TRUE (OFFH IN A REG)
                        ; IF THERE IS A CHARACTER
                        ; WAITING.
                        ;
0282' DB00              CSTS:   IN      TTS
0284' E601                      ANI     TTYDA
0286' 3E00                      MVI     A,FALSE
0288' C0                        RNZ             ;MAY NEED PATCHING***
0289' 2F                        CMA             ;IF DIFFERENT I/O USED
028A' C9                        RET
                        ;
                        ; GET THREE PARAMETERS AND
                        ; CRLF.
                        ;
028B' 0C                EXPR3:  INR     C
028C' CD 0298'                  CALL    EXPR
028F' CD 0278'                  CALL    CRLF
0292' C1                        POP     B
0293' D1                        POP     D
0294' E1                        POP     H
0295' C9                        RET
                        ;
                        ; GET ONE PARAMETER.
                        ; NO CRLF.
                        ;
0296' 0E01              EXPR1:  MVI     C,1
                        ;
                        ; THIS IS THE MAIN "PARAMETER-GETTING" ROUTINE.
                        ; THIS ROUTINE WILL ABORT ON A NON-HEX CHARACTER.
                        ; IT TAKES THE MOST RECENTLY 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.
                        ;
0298' 21 0000   EXPR:   LXI     H,0         ;INITIALIZE HL TO ZERO
029B' CD 03DC'  EX0:    CALL    TI          ;GET SOMETHING FROM CONSOLE
029E' 47        EX1:    MOV     B,A         ;SAVE IT
029F' CD 0338'          CALL    NIBBLE      ;CONVERT ASCII TO HEX.
02A2' 3808              JRC     ..EX2       ;ILLEGAL CHARACTER DETECTED
02A4' 29                DAD     H           ;MULTIPLY BY 16
02A5' 29                DAD     H
02A6' 29                DAD     H
02A7' 29                DAD     H
02A8' B5                ORA     L           ;OR IN THE SINGLE NIBBLE
02A9' 6F                MOV     L,A
02AA' 18EF              JMPR    EX0         ;GET SOME MORE
02AC' E3        ..EX2:  XTHL                ;SAVE UP IN STACK
02AD' E5                PUSH    H           ;REPLACE THE RETURN
02AE' 78                MOV     A,B         ;TEST THE DELIMITER
02AF' CD 0368'          CALL    QCHK
02B2' 3002              JRNC    ..EX3       ;DELIMITER ENTERED?
02B4' 0D                DCR     C           ;CR, SHOULD GO TO ZERO
02B5' C8                RZ                  ; RETURN IF IT DOES
02B6' C2 001E'  ..EX3:  JNZ     ERROR       ;SOMETHING WRONG
02B9' 0D                DCR     C           ;DO THIS AGAIN?
02BA' 20DC              JRNZ    EXPR        ; YES.
02BC' C9                RET                 ;ELSE RETURN
                        ;
                        ; RANGE TESTING ROUTINES.
                        ; CARRY SET INDICATES RANGE EXCEEDED.
                        ;
02BD' CD 02C3'  HILOX:  CALL    HILO
02C0' D0                RNC                 ;OK
02C1' D1                POP     D           ;RETURN ONE LEVEL BACK
02C2' C9                RET
                        ;
02C3' 23        HILO:   INX     H           ;INCREMENT HL
02C4' 7C                MOV     A,H         ;TEST FOR CROSSING 64K BORDER
02C5' B5                ORA     L
02C6' 37                STC                 ;CARRY SET=STOP
02C7' C8                RZ                  ;YES, BORDER CROSSED
02C8' 7B                MOV     A,E         ;NOW, TEST HL VS. DE
02C9' 95                SUB     L
02CA' 7A                MOV     A,D
02CB' 9C                SBB     H
02CC' C9                RET                 ;IF CARRY WAS SET, THEN STOP
                        ;
                        ;       HEXADECIMAL MATH ROUTINE
                        ;
                        ; THIS ROUTINE IS USEFUL FOR
                        ; DETERMINING RELATIVE JUMP
                        ; OFFSETS.  IT RETURNS THE SUM
                        ; & DIFFERENCE OF TWO PARAMETERS.
                        ;
```

```
                        ;    H<)>,<Y>
                        ;
                        ;   X+Y    X-Y
                        ;
02CD' FE48              HEXN:   CPI     'H'     ;SEE IF HEX MATH
02CF' C2 039C'                  JNZ     LOAD
02D2' CD 0273'                  CALL    E)LF
02D5' E5                        PUSH    H       ;SAVE HL FOR LATER
02D6' 19                        DAD     D       ;GET SUM
02D7' CD 021D'                  CALL    HLSP    ;PRINT IT
02DA' E1                        POP     H       ;THIS IS LATER
02DB' B7                        ORA     A       ;CLEAR CARRY
02DC' ED52                      DSBC    D       ;GET DIFFERENCE & PRINT IT
                        ;
                        ; PRINT H&L ON CONSOLE
                        ;
02DE' 7C                LADR:   MOV     A,H
02DF' CD 02E3'                  CALL    LBYTE
02E2' 7D                        MOV     A,L
02E3' F5                LBYTE:  PUSH    PSW
02E4' CD 0265'                  CALL    CBYTE
02E7' CD 0222'                  CALL    CO
02EA' F1                        POP     PSW
02EB' CD 0269'                  CALL    CONV
02EE' C3 0222'                  JMP     CO
                        ;
                        ; THIS ROUTINE SENDS EIGHT RUBOUTS
                        ; TO THE PUNCH DEVICE.
                        ;
02F1' 01 08FF           MARK:   LXI     B,08FFH ;SET-UP B&C
02F4' 1803                      JMPR    LEO
                        ;
                        ; THIS ROUTINE SENDS BLANKS TO THE
                        ; PUNCH DEVICE.
                        ;
02F6' 01 4800           LEAD:   LXI     B,4800H ;PRESET FOR SOME NULLS
02F9' CD 0233'          LEO:    CALL    PO
02FC' 10FB                      DJNZ    LEO
02FE' C9                        RET
                        ;
                        ; THIS ROUTINE RETURNS TO A USER
                        ; PROGRAM THE CURRENT TOP OF
                        ; MEMORY VALUE MINUS WORKSPACE
                        ; AREA USED BY THE MONITOR.
                        ;
02FF' E5                MEMCK:  PUSH    H
0300' CD 0313'                  CALL    MEMSIZ
0303' 44                        MOV     B,H
0304' 3EC0                      MVI     A,0C0H  ;LEAVE SOME ROOM FOR STACK
0306' E1                        POP     H
0307' C9                        RET
```

```
                        ;
                        ;       WE BEGIN IN THE MIDDLE.....
                        ;
0308' 3E00      BEGIN:  MVI     A,I     ;INITIAL 'I' REG. VALUE
030A' ED47              STAI            ;NEEDED IF USING INTERUPT.
030C' AF                XRA     A       ;CLEAR READER CONTROL
030D' D303              OUT     RCP     ; PORT.
030F' 31 0034'          LXI     SP,STACK        ;SET UP A FAKE STACK
                        ;
0312' 06                .BYTE   (MVI)           ;SKIP OVER PUSH
                        ;
                        ; THIS IS A CALLED ROUTINE USED
                        ; TO CALCULATE THE TOP OF MEMORY
                        ; STARTING FROM THE BOTTOM OF
                        ; MEMORY, AND SEARCHING UPWARD UNTIL
                        ; FIRST R/W MEMORY IS FOUND, AND THEN
                        ; CONTINUING UNTIL THE END OF THE R/W
                        ; MEMORY. THIS ALLOWS R.O.M. AT ZERO,
                        ; AND INSURES A CONTINUOUS MEMORY BLOCK
                        ; HAS BEEN FOUND.
                        ; IT IS USED BY THE ERROR ROUTINE TO
                        ; RESET THE STACK POINTER.
                        ;
0313' C5        MEMSIZ: PUSH    B
0314' 01 0000'          LXI     B,ZAP   ;POINT TO START OF MONITOR
0317' 21 FFFF           LXI     H,-1    ;RAM SEARCH STARTING PT.-1
031A' 24        ..MO:   INR     H       ;FIRST FIND R/W MEMORY
031B' 7E                MOV     A,M
031C' 2F                CMA
031D' 77                MOV     M,A
031E' BE                CMP     M
031F' 2F                CMA
0320' 77                MOV     M,A
0321' 20F7              JRNZ    ..MO
0323' 24        ..M1:   INR     H       ;R/W FOUND, NOW FIND END
0324' 7E                MOV     A,M
0325' 2F                CMA
0326' 77                MOV     M,A
0327' BE                CMP     M
0328' 2F                CMA
0329' 77                MOV     M,A
032A' 2004              JRNZ    ..M2
032C' 7C                MOV     A,H     ;TEST FOR MONITOR BORDER
032D' B8                CMP     B
032E' 20F3              JRNZ    ..M1    ;NOT THERE YET
0330' 25        ..M2:   DCR     H       ;BACK UP
0331' C1                POP     B
0332' C9                RET             ;VALUE IN HL
                        ;
                        ; THIS GETS A READER CHARACTER, AND
                        ; CONVERTS IT FROM ASCII TO HEX.
                        ;
0333' CD 020C'  RIBBLE: CALL    RIFF
0336' E67F              ANI     7FH
```

```
0338' D630          NIBBLE: SUI     '0'         ;QUALIFY & CONVERT
033A' D8                    RC                  ;<0
033B' FE17                  CPI     'G'-'0'     ;>F?
033D' 3F                    CMC                 ;PERVERT CARRY
033E' D8                    RC
033F' FE0A                  CPI     10          ;NMBR?
0341' 3F                    CMC                 ;PERVERT AGAIN
0342' D0                    RNC                 ;RETURN CLEAN
0343' D607                  SUI 'A'-'9'-1       ;ADJUST
0345' FE0A                  CPI     10          ;FILTER ":" THRU "@"
0347' C9                    RET
                    ;
                    ; SEND H&L VALUE TO PUNCH DEVICE
                    ;
0348' 7C            PADR:   MOV     A,H
0349' CD 034D'              CALL    PBYTE
034C' 7D                    MOV     A,L
                    ;
                    ; PUNCH A SINGLE BYTE
                    ;
034D' F5            PBYTE:  PUSH    PSW         ;NIBBLE AT A TIME
034E' CD 0265'              CALL    CBYTE
0351' CD 0233'              CALL    PO
0354' F1                    POP     PSW         ;NEXT NIBBLE
0355' F5                    PUSH    PSW         ;SAVE FOR CHECKSUM
0356' CD 0269'              CALL    CONV
0359' CD 0233'              CALL    PO
035C' F1                    POP     PSW         ;ORIGINAL BYTE HERE
035D' 82                    ADD     D           ;ADDED TO CHECKSUM
035E' 57                    MOV     D,A         ;UPDATE CHECKSUM
035F' C9                    RET
                    ;
                    ;
0360' 0E2D          COPCK:  MVI     C,'-'
0362' CD 0222'              CALL    CO
0365' CD 03DC'              CALL    TI
                    ;
                    ; TEST FOR DELIMITERS
                    ;
0368' FE20          QCHK:   CPI     ' '         ;RETURN ZERO IF DELIMITER
036A' C8                    RZ
036B' FE2C                  CPI     ','
036D' C8                    RZ
036E' FE0D                  CPI     CR          ;RETURN W/CARRY SET IF CR
0370' 37                    STC
0371' C8                    RZ
0372' 3F                    CMC                 ;ELSE NON-ZERO, NO CARRY
0373' C9                    RET
                    ;
                    ; MAIN CONSOLE INPUT ROUTINE
                    ;
0374' DB00          CI:     IN      TTS
0376' E601                  ANI     TTYDA
0378' 20FA                  JRNZ    CI
037A' DB01                  IN      TTI
```

```
037C' C9                        RET
                        ;
                        ; READER INPUT ROUTINE, WITH
                        ; TIME-OUT DELAY. INCLUDES
                        ; PULSING OF HARDWARE PORT
                        ; TO INDICATE REQUEST FOR
                        ; READER DATA.
                        ;
                        ; THIS MAY BE ALTERED TO ANY
                        ; I/O PORT CONFIGURATION TO ENABLE
                        ; SEPARATE READER/PUNCH DEVICE.
                        ;
037D' E5        RI:     PUSH    H
037E' 3EFF              MVI     A,0FFH  ;MAY BE ALTERED TO SUIT
0380' D303              OUT     RCP     ;PULSE READER CONTROL PORT
0382' AF                XRA     A       ;CLEAR IT
0383' D303              OUT     RCP
0385' 67                MOV     H,A     ;CLEAR FOR TIME-OUT TEST
0386' DB00      RIO:    IN      TTS     ;MAY BE MODIFIED ***
0388' E601              ANI     TTYDA   ;BUT ALWAYS USE 'ANI'
038A' 280C              JRZ     RI2     ;TO CLEAR CARRY
038C' C5                PUSH    B
038D' 06FF              MVI     B,0FFH  ;SHORTEN FOR HIGH-SPEED DEVICE
038F' E3        DLO:    XTHL            ;WASTE TIME
0390' E3                XTHL            ;FOR DELAY
0391' 10FC              DJNZ    DLO
0393' C1                POP     B
0394' 25                DCR     H
0395' 20EF              JRNZ    RIO
0397' 37        RI1:    STC             ;*NOTE: CARRY SET TO INDICATE
                                        ; NO DATA.
0398' DB01      RI2:    IN      TTI
039A' E1        RID:    POP     H
039B' C9                RET
                        ;
                        ; THIS ROUTINE READS A BINARY FILE
                        ; IMAGE, IN THE FORM AS PUNCHED IN
                        ; THE "U" (UNLOAD) COMMAND.  IT TAKES
                        ; ONE PARAMETER, WHICH IS THE STARTING
                        ; ADDRESS OF THE LOAD, AND WILL PRINT
                        ; THE LAST ADDRESS(+1) LOADED ON THE
                        ; CONSOLE DEVICE.
                        ;
039C' FE4C      LOAD:   CPI     'L'     ;SEE IF 'LOAD' COMMAND
039E' 205F              JRNZ    NEXT
03A0' CD 0296'          CALL    EXPR1   ;INITIAL LOAD ADDRESS
03A3' E1                POP     H
03A4' CD 0278'          CALL    CRLF
03A7' 16FF              MVI     D,0FFH  ;START-OF-FILE TAG
03A9' 0604      ..L0:   MVI     B,4     ;FIND AT LEAST FOUR 0FFH'S
03AB' CD 020C'  ..L1:   CALL    RIFF
03AE' 20F9              JRNZ    ..L0
03B0' 10F9              DJNZ    ..L1
03B2' CD 020C'  ..L2:   CALL    RIFF    ;4 FOUND, NOW WAIT FOR NON-0FFH
03B5' 28FB              JRZ     ..L2
```

```
03B7' 77                    MOV     M,A     ;FIRST REAL DATA BYTE
03B8' 3E07                  MVI     A,BELL  ;TELL TTY
03BA' D301                  OUT     TTO
03BC' 23            ..L3:   INX     H
03BD' CD 020C'              CALL    RIFF
03C0' 2803                  JRZ     ..EL    ;POSSIBLE END OF FILE
03C2' 77                    MOV     M,A
03C3' 18F7                  JMPR    ..L3
03C5' 0601          ..EL:   MVI     B,1     ;INITIALIZE
03C7' CD 020C'      ..EL0:  CALL    RIFF
03CA' 2009                  JRNZ    ..EL1
03CC' 04                    INR     B       ;COUNT QUES
03CD' 3E07                  MVI     A,MAX   ;LOOK FOR EOF
03CF' B8                    CMP     B       ;FOUND MAX?
03D0' 20F5                  JRNZ    ..EL0   ;NOPE
03D2' C3 02DE'              JMP     LADR    ;YEP, PRINT END ADDR
03D5' 72            ..EL1:  MOV     M,D
03D6' 23                    INX     H
03D7' 10FC                  DJNZ    ..EL1
03D9' 77                    MOV     M,A     ;REAL BYTE
03DA' 18E0                  JMPR    ..L3
                    ;
                    ;
                    ; THIS IS THE INTERNAL KEYBOARD
                    ; HANDLING ROUTINE. IT WILL IGNORE
                    ; RUBOUTS (OFFH) AND BLANKS (00),
                    ; AND IT WILL NOT ECHO CR'S & N'S.
                    ; (NO N'S FOR THE "NULL" COMMAND).
                    ; IT CONVERTS LOWER CASE TO UPPER
                    ; CASE FOR THE LOOK-UP OF COMMANDS.
                    ;
                    ; OTHER CHARACTERS ARE ECHOED AS THEY
                    ; ARE RECIEVED.
                    ;
03DC' CD 0374'      TI:     CALL    CI
03DF' E67F                  ANI     7FH     ;KILL PARITY BIT
03E1' 3C                    INR     A       ;IGNORE RUBOUTS
03E2' F8                    RM
03E3' 3D                    DCR     A       ;IGNORE NULLS
03E4' C8                    RZ
03E5' FE4E                  CPI     'N'     ;IGNORE N'S FOR NULL CMND
03E7' C8                    RZ
03E8' FE6E                  CPI     'n'
03EA' 2810                  JRZ     ..I
03EC' FE0D                  CPI     CR      ;IGNORE CR'S
03EE' C8                    RZ
03EF' C5                    PUSH    B
03F0' 4F                    MOV     C,A
03F1' CD 0222'              CALL    CO
03F4' 79                    MOV     A,C
03F5' C1                    POP     B
03F6' FE40                  CPI     'A'-1   ;CONVERT TO UPPER CASE
03F8' D8                    RC
03F9' FE7B                  CPI     'z'+1
03FB' D0                    RNC
```

```
03FC' E65F          ..T:    ANI     05FH
03FE' C9                    RET
                    ;
                    ;
                    ;
                    ;
03FF' C9            NEXT:   RET             ;ADDITIONAL COMMANDS
                                            ;MAY BE TESTED FROM HERE,
                                            ;AND THE MONITOR EXTENDED
                                            ;FROM BEYOND THIS POINT.
                    ;
                    ;
0400'                       Z:
                                            ;END OF PROGRAM
                    ;
                    ;
                    ;
0000'               .END    ZAP
```

++++ SYMBOL TABLE ++++

AHEAD	0038'	BEGIN	0308'	BELL	0007	BLK	0220'
CBYTE	0265'	CI	0374'	CO	0222'	CONV	0269'
COPCK	0360'	CR	000D	CRLF	0278'	CSTS	0282'
DISP	0057'	DLO	038F'	EOF	006E'	ERROR	001E'
ERRX	0209'	EXO	029B'	EXI	029E'	EXLF	0273'
EXPR	0298'	EXPR1	0296'	EXPR3	028B'	FALSE	0000
FIL	0000	FILL	008C'	GOTO	009C'	HEXN	02CD'
HILO	02C3'	HILOX	02BD'	HLSP	021D'	I	0000
IOSET	0017'	LADR	02DE'	LBYTE	02E3'	LEO	02F9'
LEAD	02F6'	LENGTH	0400'	LF	000A	LFADR	021A'
LOAD	039C'	LODO	00ED'	LOD4	012A'	LODCB	0152'
LODR	0132'	MARK	02F1'	MAX	0007	MEMCK	02FF'
MEMSIZ	0313'	MOVE	00C5'	MSG	0029'	MSGL	000D
NEXT	03FF'	NIBBLE	0338'	NULL	025F'	NULLX	025B'
PADR	0348'	PBYTE	034D'	PEOL	022C'	PO	0233'
QCHK	0368'	RCP	0003	READ	00D4'	RI	037D'
RIO	0386'	RI1	0397'	RI2	0398'	RIBBLE	0333'
RID	039A'	RIFF	020C'	RUB	00FF	SBYTE	0162'
SIZE	0213'	STACK	0034'	STARO	004A'	START	003E'
STORE	0175'	SUBS	017C'	TEST	00A6'	TI	03DC'
TOM	01F2'	TOM1	01F5'	TRUE	FFFF	TTI	0001
TTO	0001	TTS	0000	TTYBE	0080	TTYDA	0001
UNLD	023D'	WRITE	01AE'	Z	0400'	ZAP	0000'

NO PROGRAM ERRORS

GENERAL INFORMATION

A. Customer Service

Customer service falls into two broad categories:

1. Equipment troubleshooting
2. User applications counseling.

In the case of Equipment troubleshooting when you wish to return the unit for factory service, the following procedure should be adhered to whether the unit is under warranty or not.

1. Write up the exact symptoms of the problem. Give exact details of what you observed, what you noticed, what you were doing when the problem was first noticed, etc.

2. Describe the system you had in operation when the problem developed. Note kind of mainframe, accessory boards in use, program being run, etc. Also note if the other boards are still working correctly.

3. Describe what you have done to try and handle the problem. Please be as specific as possible.

4. Pack the unit well (You would be wise to keep the shipping carton and materials this unit came in for this possibility.) and return it postpaid to TDL.

5. If the unit is NOT under warranty, enclose an authorization to repair and bill to whatever dollar limit beyond which you would want to be informed before we continue.

6. If the unit is under warranty, it will be treated as per the conditions as laid out in the warranty.

In the case of user applications counseling, the service is generally free of charge. This service is designed to aid you in applications where your own ability or experience is not sufficient to provide the answer. This is not intended to provide a broad educational service of a general nature. Rather it is designed to answer specific applications problems where a "how to" may not be clear to a less than very experienced computerist. If your

questions are specific, you will receive an answer as fast as is possible.

For questions of a more general nature, such as those that might repeat from many users, or for items which we feel would be of interest to a broader public, such will be printed up and distributed as part of the Z-80 User's group Newsletter which is currently being established. The newsletter will publish any information, program development, novel computer applications etc. which are either submitted to us by you, the user's, or by our engineers and programmers. Please feel free to contribute to this effort in any way.

As our development progresses, and as your programming ideas come in, a software library will be established for your use.

TDL WARRANTY

TECHNICAL DESIGN LABS INC., in recognition of its responsibility to provide quality components and adequate instructions for their proper assembly, warrants its products as follows:

All components sold by Technical Design Labs Inc., (hereinafter referred to as TDL) are first quality prime and are procured from reputable distributors and/or factories and their representatives, and any part which fails because of defects in manufacture or material will be replaced at no charge for a period of 3 months for kits, and one year for assembled products following the date of purchase as shown on the customer's invoice. For replacement, the defective part must be returned to TDL postpaid within the warranty period.

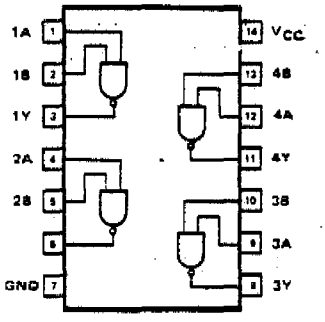
Any malfunctioning unit or subunit, purchased as a kit and returned to TDL within the 3 month warranty period, which in the judgement of TDL has been constructed with care, and has not been subject to electrical or mechanical abuse, will be restored to proper operating condition or replaced at TDL's discretion and returned, with a minimal charge to cover postage.

Any units or subunits purchased as a kit and returned to TDL within the 3 month warranty period, which in the opinion of TDL is not covered by the above conditions will be repaired and returned at a cost sommensurate with the work required. In no case will this charge exceed $30.00 without prior notification and approval of the owner.

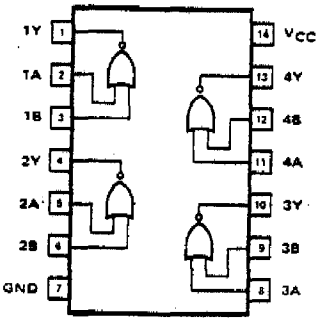
Any unit or subunit, purchased as assembled units are guaranteed to meet the specifications in effect at the time of manufacture for a period of at least one year following purchase. These units are additionally guaranteed against defects in materials or workmanship for the same one year period. All warranted factory assembled units returned to TDL postpaid will be repaired and returned without charge providing only that no evidence of electrical or mechanical abuse exists.

This warranty is made in lieu of all other warranties expressed or implied and is limited in any case to the repair or replacement of the unit or subunit involved.

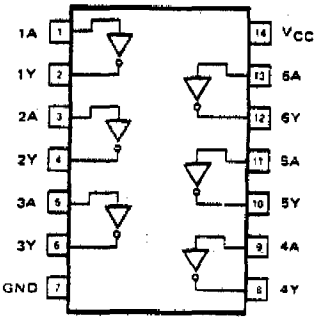
APPENDIX A: Pinout Diagrams of ICs on the ZPU



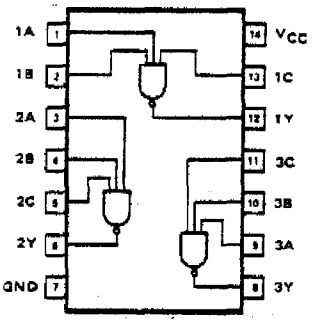
74LS00
ICs 18,21,22



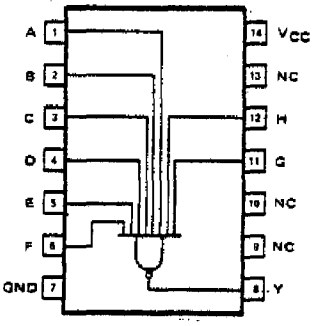
74LS02
ICs 11,12,20



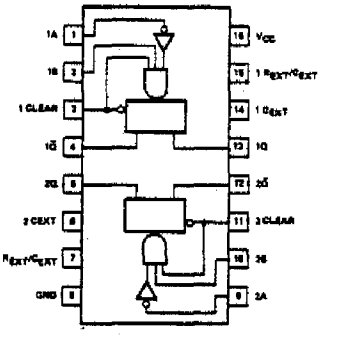
74LS04
ICs 13,15,19



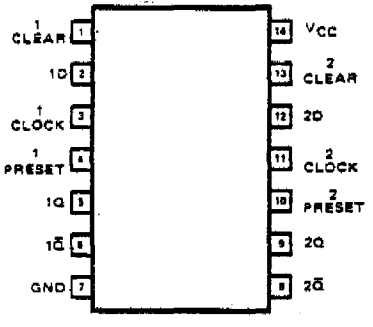
74LS10
IC17



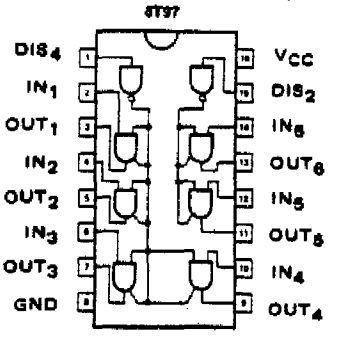
74LS30
IC14



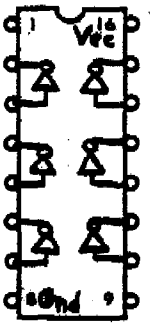
74LS74
IC16



74123
IC23



8T97,8097 or
74367
ICs 1 to 10



SCL 4049
IC24

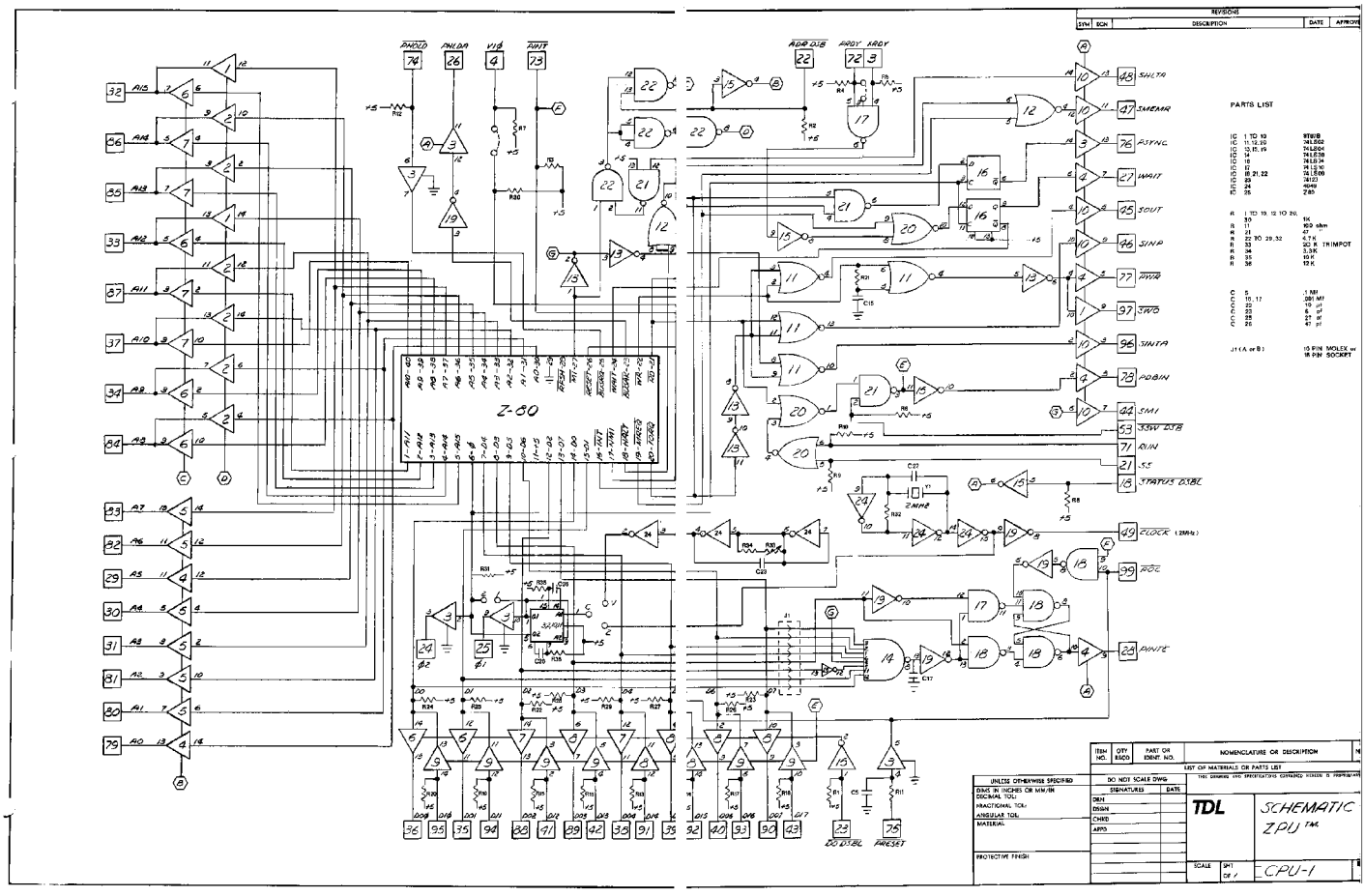
TDL ZPU CARD BUS SIGNAL LIST

(For explanation of asterisks (*) see next page.)

1	+8v	51	+8v
2	+16v	52	-16v
3	XRDY	53	SSW DSB
4	VI Ø	54	EXT CLR
5	VI 1	55	*
6	VI 2	56	**
7	VI 3	57	**
8	VI 4	58	**
9	VI 5	59	CMW (1)
10	VI 6	60	
11	VI 7	61	
12		62	
13		63	
14		64	
15		65	
16		66	
17		67	***
18	STATUS DSBL	68	MWRITE
19	CCDSBL	69	****
20	*****	70	******
21	SS	71	RUN
22	ADDR DSBL	72	PRDY
23	DO DSBL	73	PINT
24	Ø2	74	PHOLD
25	Ø1	75	PRESET
26	PHLDA	76	PSYNC
27	PWAIT	77	PWR
28	PINTE	78	PDBIN
29	A5	79	AØ
30	A4	80	A1
31	A3	81	A2
32	A15	82	A6
33	A12	83	A7
34	A9	84	A8
35	DO 1	85	A13
36	DO Ø	86	A14
37	A10	87	A11
38	DO 4	88	DO 2
39	DO 5	89	DO 3
40	DO 6	90	DO 7
41	DI 2	91	DI 4
42	DI 3	92	DI 5
43	DI 7	93	DI 6
44	SMI	94	DI 1
45	SOUT	95	DI Ø
46	SINP	96	SINTA
47	SMEMR	97	SWO
48	SHLTA	98	RFSH(optional)
49	CLOCK (2MHz)	99	POC
50	GND	100	GND

*	reserved for chassis ground
**	reserved for Altair 8800B
***	reserved for PTCO PHANTOM
****	reserved for protect status
*****	reserved for memory unprotect
******	reserved for memory protect

1. CMW reserved for Conditional Memory Write, which is a system protect signal on TDL's Memory Management Board.


REVISIONS
SYM | ECN | DESCRIPTION | DATE | APPROVE
PARTS LIST
Z-80
SHLTA
SAREAR
RSYNC
WAIT
SOUT
SINP
PINTA
SMO
SINTA
PDBIN
SMI
SSW DSB
RUN
SS
STATUS DSBL
CLOCK (2MHz)
SOC
PINTE
PRESET
DO DSBL
SCHEMATIC
ZPU TAK
CPU-1
TDL