USER'S MANUAL

The Speakeasy™
cassette/general purpose interface

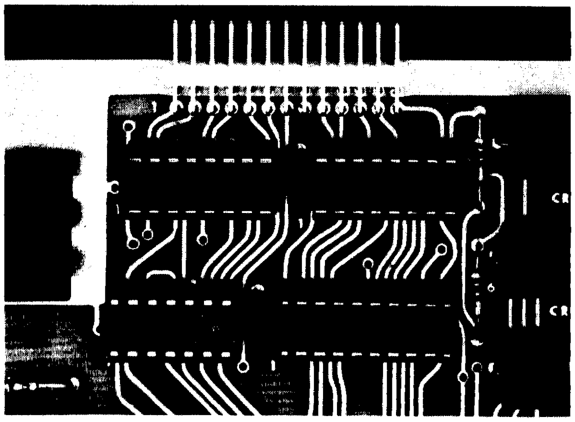


TABLE OF CONTENTS

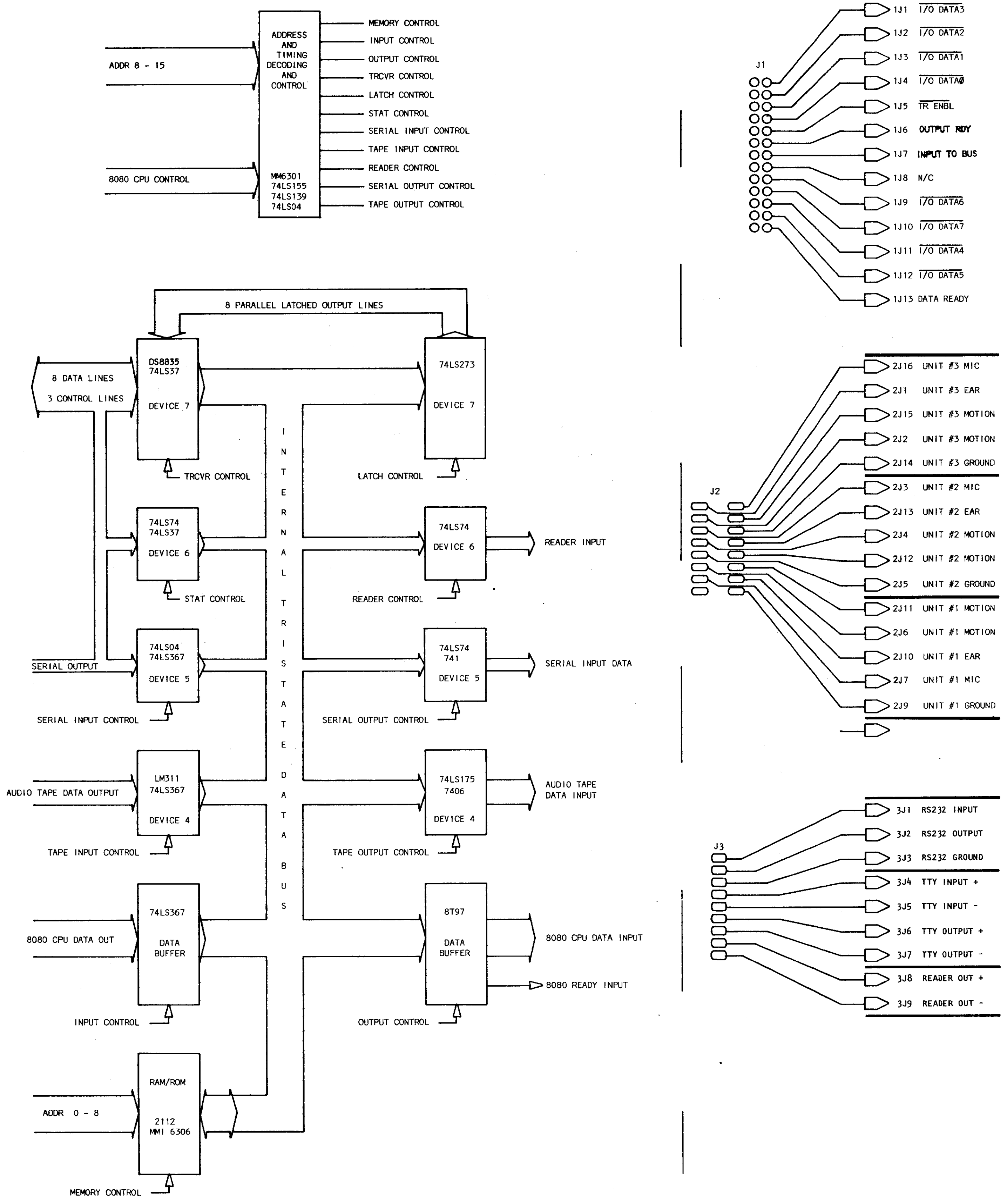
Thinker
Toys™ 1201 10th.
Berkeley, Calif.
94710

THE SPEAKEASY^{TM}
CASSETTE/GENERAL PURPOSE INTERFACE

INTRODUCTION

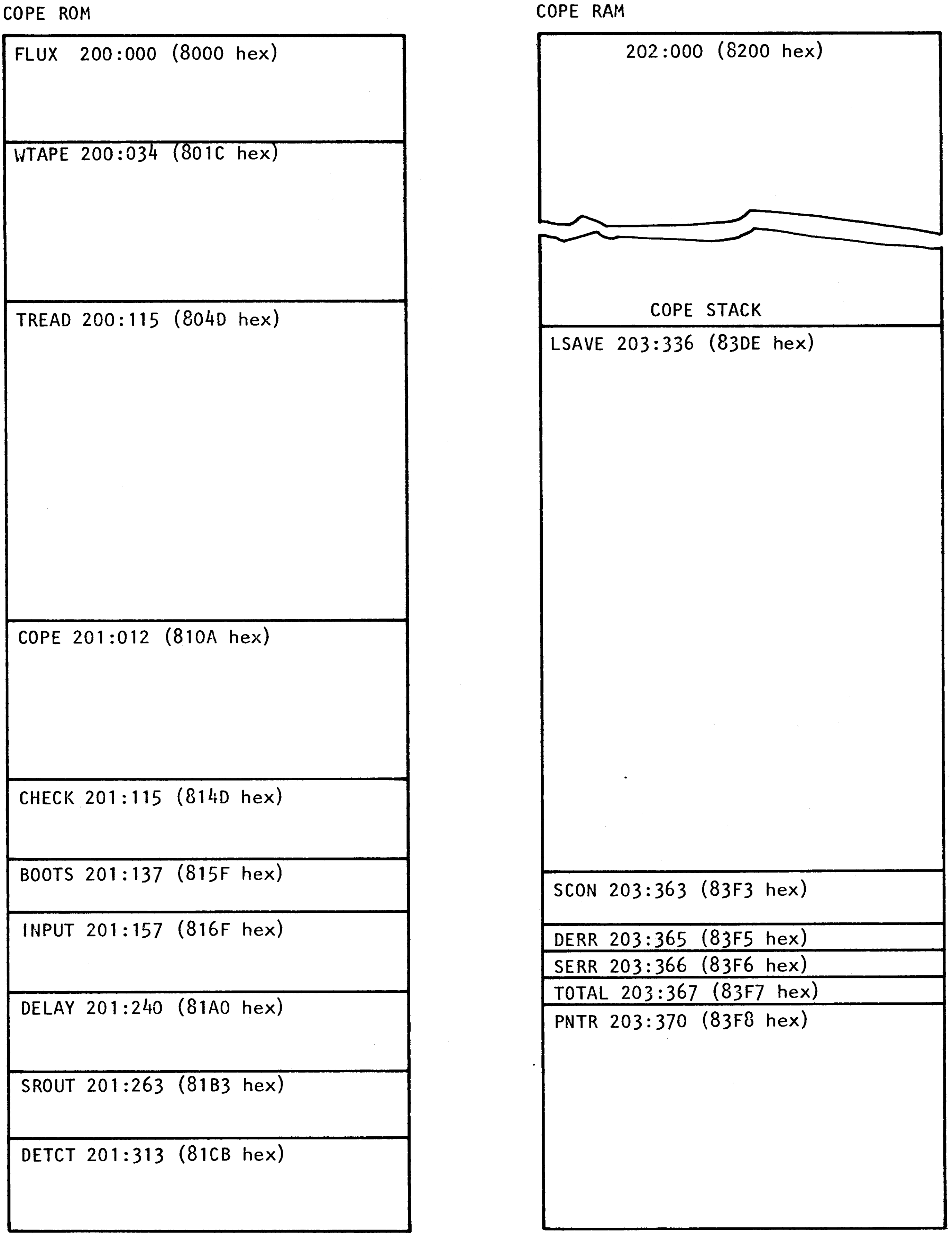
The Speakeasy was the first and is still the most versatile of the "intelligent" S-100 interface boards. It pioneered the use of RAM and ROM and PROM on the I/O board itself. There are three completely separate interfaces on the Speakeasy board. Yet, the board is not dense. There is very little chance of creating solder bridges because there is sufficient clearance for all the parts. Other I/O boards which are as versatile as the Speakeasy have upwards to seventy integrated circuits and traces ten mills wide. The reason for the vast difference in circuit complexity is that the Speakeasy has its complexity concentrated in its PROMs. The intelligence of the CPU is harnessed to keep the circuit layouts simple and clean.

However, there are certain trade-offs incurred by this novel approach to the I/O interfacing: the CPU communicates with the Speakeasy interfaces much differently than the standard "dumb" I/O interfaces. The purpose of this manual is to ease the problems associated with this novel and more efficient method of controlling I/O devices. The staff at Morrow's has worked diligently to make this manual as clear and complete as possible and, most importantly, useable. Comments on improving the material will be welcomed. Errors are always difficult to completely eliminate from a technical document; your help in finding them is solicited. Please feel free to write us with your comments.



THE SPEAKEASY I/O INTERFACE A THINKER TOY from MORROW'S
CONNECTOR LAYOUT AND BLOCK DIAGRAM

COPE MEMORY MAP



PRINCIPLES OF OPERATION

THE TAPE CASSETTE INTERFACE

GENERAL

The I/O board contains interface circuitry that allows an S-100 compatible computer to communicate with three audio cassette player/recorders. The interface can read from any one of the three tape channels and can write on any combination of the three tape channels. The data standard that the interface uses is the "Kansas City" 300 baud data format agreed upon several years ago by a diverse group of manufacturers of personal computing equipment.

THE "KANSAS CITY" DATA STANDARD

The Kansas City Standard (KCS) is a data format which specifies how digital data shall be encoded into tones that can be recorded on low cost audio cassette/player recorders. KCS defines a logic 1 to be a 2400 hertz signal which is eight complete cycles long. A logic 0 is a 1200 hertz signal which is four complete cycles long. KCS also specifies how logic 0s and 1s shall be grouped together in bytes on the audio cassette: there is to be at least two logic 1 bits (rest bits) followed by a logic 0 bit (start bit) followed by eight data bits.

The KCS is a conservative data format. There is a large factor of redundancy (a factor of eight for logic 1 and a factor of four for logic 0) which means that this is an extremely reliable method of data storage and retrieval. As with most things, the reliability of the KCS is not without penalties. In this case, the penalty is the relatively slow rate that data is stored or retrieved -- 300 bits per second. However, the first concern of any decent standard should be data integrity and in this sense the KCS performs admirably.

SUITABLE AUDIO CASSETTE PLAYER/RECORDERS

There are a wide variety of audio cassette units available which are suitable for use with the I/O board. However, there are several features which a cassette unit should have.

(1) The player should have a digital counter so that file positions on tape can be easily found.
(2) The unit should have AGC (automatic gain control) for making recordings so that volume settings can be restricted to one value rather than requiring one adjustment for writing and another for reading.
(3) There should be an input jack that allows a remote switch to activate the unit.
(4) There should be an auxiliary input other than the microphone input which will accomodate a radio or record player pre-amp output.

The signal level from the cassette interface of the I/O board is approximately 4 volts peak-to-peak and is too high for microphone input. This high level signal has a much better signal-to-noise ratio than the 50 millevolt signal necessary for the microphone input. During recording, the microphone input must be disabled. Sometimes, the microphone is disabled when a jack is inserted in the auxilliary input but more often the cassette unit comes supplied with a small plastic plug that is inserted into the microphone jack to disable the input.

THE SOFTWARE INTERFACE

A unique design feature of the I/O board is the use of PROM and RAM to link the 8080 CPU intimately with the cassette player/recorders connected to the board. Through the use of software in PROM, the I/O board directs the CPU to measure precise time intervals with time delay subroutines and to use these timing routines to generate the 1200/2400 Hz wave forms that are sent out to the recorder. Using other software routines stored in the on-board PROM, the CPU measures the time between zero-crossings of the incoming signals from the cassette player to convert the 1200/2400 Hz wave forms into digital information.

The software in the PROMs in highly modularized. For input, specific routines measure the time between wave form zero-crossings. Other routines use the zero-crossing measurements to build individual bits while other software uses the bit-building software to build bytes. High level routines take assembled bytes and transfer them to memory.

For output, the high level routines transfer bytes to routines which disassemble the bytes to bits. The bits are in turn transferred to subroutines which convert them to 1200/2400 Hz wave forms through the use of time measurement software.

The software has been designed to read or write entire blocks of information to or from memory with no intervention from external software. A program such as an editor or BASIC communicates with software of the tape cassette interface by issuing a CALL COPE instruction. Certain CPU registers must be initialized to furnish detailed information to the I/O board software about the number of bytes to be transferred, the address of the first byte and the type of transfer (read, write, verify). The rest of this section explains the details of how to use the powerful COPE software which is in the PROMs of the I/O board.

COPE

COPE is an acronym for Cassette OPerating Executive. It is the name given to that body of software in the PROMs which effects data transfers to and from the cassette player/recorders connected to the I/O board. Below are the kinds of tasks COPE will perform:

(1) Read a block of data (from 1 to 65k bytes) from any one of three cassette players.
(2) Verify a block of data on a cassette player against a block of data in memory.

(3) Move across a block of data on a cassette player.
(4) Write a block of data from memory to any one or combination of three cassette recorders.

COPE COMMAND STRUCTURE

The COPE software has exactly one entry point: 201:012 octal or 810A hex and is almost without exception entered via a CALL instruction (315 octal, CD hex). How does the COPE software distinguish between reading, writing, etc., if there is only one entry point? The A register or "accumulator" of the 8080A serves as the primary command register and the C register acts as a secondary command register. That is, the patterns or values present in the A and C registers when a "CALL COPE" instruction is executed determines what kind of operation the COPE software will perform. The pattern in the C register is ignored if the value in the A register specifies a write command. If the A register does not specify a write command, the value of the C register is used to specify whether the command is a read, verify or move.

A REGISTER COMMAND FORMATS

The convention used below to specify bit positions in 8080 registers is the same as the one used in the 8080 User's Manual published by INTEL CORP: the higher the bit number, the greater the significance the bit has in forming a value of the register.

A register command structure:

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit Ø

Bit Ø - The write/not write bit. If this bit is a one, data will be read from the computer's memory and written on any combination of the three cassette recorders connected to the interface. If this bit is a zero, data from exactly one of the cassette players is to be: (i) read from the tape into the computer's memory; (ii) read from the tape and compared with memory; or (iii) simply passed over so as to access the next block of data on the tape. The C register is used as a secondary command register when bit 0 is a zero to decide which of the three alternatives should be performed by the COPE software.

Bit 1 - Cassette channel #1 bit. This bit controls the motion of the cassette player/recorder connected to channel #1 of the I/O board. A one in this bit position turns the motor and electronics "on" while a zero turns them "off."

Bit 2 - Cassette channel #2 bit. This bit controls the motion of the cassette player/recorder connected to channel #2 of the I/O board. A one in this bit position turns the motor and electronics "on" while a zero turns them "off."

Bit 3 - Cassette channel #3 bit. This bit controls the motion of the cassette player/recorder connected to channel #3 of the I/O board. A one in this bit position turns the motor and electronics "on" while a zero turns them "off."

Bit 4 - Always a zero.

Bit 5 - Always a zero.

Bit 6 - Always a zero.

Bit 7 - If this bit is a one, all the motion control relays will be turned off when COPE completes a command. There are exceptions, however, and one will be described later.

C REGISTER COMMAND FORMATS

The value or pattern of the C register is of no importance to the COPE software unless bit 0 of the A register is zero. When bit 0 of the A register is a zero, the C register has the following command structure:

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

Bit 0 - The MOVE bit. If this bit is a one and bit 0 of the A register is a zero, no data will be transferred from tape to memory. Instead, the active tape transport will move the tape across a block of data. The length of this block is discussed in the next section.

Bit 1 - Always a zero.

Bit 2 - Always a zero.

Bit 3 - Always a zero.

Bit 4 - Always a zero.

Bit 5 - Always a zero.

Bit 6 - The VERIFY bit. If this bit is a one and bit 0 of the A register is a zero, a "compare" or "verify" operation will be performed. Data will be read from the transport connected to the active channel and compared byte-for-byte with memory data. If any data on the tape should differ from its counterpart in memory, an error flag will be set. This is discussed later.

Bit 7 - Always a zero.

If all the bits of the C register are zero and bit 0 of the A register is zero, a read operation will be performed. The COPE software will transfer a block of data from the cassette player connected to the active channel and write this data into the computer's memory.

COPE DATA BLOCK SPECIFICATIONS

As mentioned above, COPE software performs operations on blocks of data. When a block resides in a computer's memory, it has three attributes:

(1) A starting address.
(2) An ending address.
(3) A length (in bytes).

These attributes are not independent. Given any two, the third can be calculated. The two that COPE uses are the starting address of the data block and its length. COPE requires the starting address to be in the H-L register pair and the block length to be in the D-E register pair.

When COPE writes data from memory onto one or more cassette channels, five CPU registers must be initialized before transferring control to the COPE software:

(1) The A register must be initialized with the proper pattern.
(2) The H-L register pair must be initialized with the starting address of the data block that will be written on the tape(s).
(3) The D-E register pair must be initialized with the length of the data block.

When COPE reads or verifies data on tape, six registers must be initialized before transferring control to the COPE software:

(1) The A register must be initialized with the proper pattern.
(2) The C register must specify a read or verify operation.
(3) The H-L register pair must be initialized with the starting address of the data block.
(4) The D-E register pair must be initialized with the length of the data block.

When COPE spaces the tape head across a block of data, four registers must be initialized before transferring control to the COPE software.

(1) The A register must be initialized with the proper values.
(2) The C register must specify a move operation.
(3) The D-E register pair must be initialized with the length of the data block.

The H-L register pair does not have to be initialized since no data is transferred during a move operation. However, the value in this register pair is not preserved during a COPE move operation. In fact, any COPE operation changes the values of all the CPU registers except the stack pointer.

There are two memory locations in the RAM of the I/O board that COPE uses for error reporting. DERR (203:365 octal, 83F5 hex) is the Data ERRor memory cell. This location is over-written with some non-zero value whenever an error is encountered during verify operations. COPE never initializes this cell and writes in this cell only during verify operations when an error is encountered. If, for example, five errors were encoutered during a particular verify operation, COPE would write into the DERR location five times -- each time with some non-zero value. The proper way to use the DERR location is to initialize it to zero some time prior to a verify operation. After the verify is complete, this cell should be tested. If it is still zero, no error was encountered during the verify. If it has changed from zero, an error occurred.

The other error reporting memory location is SERR (203:366 octal, 83F6 hex). This is the Status ERRor memory cell. Whenever COPE is called on to perform a READ, VERIFY, or MOVE operation, one of the first things COPE does is turn on the motor and electronics of the tape cassette player that is connected to the active channel. If through an error or oversight, the cassette player is not plugged in, turned on, or connected to the channel, the COPE wave form detection software will see only spurious signals and not regular waveforms. If this situation persists for more than 45 seconds at a time, COPE will abort the operation and write some non-zero value in SERR and then return control to the calling program. As with the DERR cell, SERR should be initialized to zero before initiating a READ, VERIFY or MOVE operation when it is necessary to detect the absence of valid waveforms over an extended period of time. If a read operation were inadvertantly started before the play button of the player were pressed, no harm would be done as long as it is depressed within 45 seconds after COPE has taken control.

EXAMPLES USING COPE

This section presents several practical examples that illustrate how the technical specifications of the previous sections are used.

Example 1: Write $(728)_{10}$ bytes of data starting at location $(120)_{10}$ of page 7 in memory onto tape channels 2 and 3.

8080 machine code is usually expressed in octal or hex so 728 and 120 need to be converted.

$$(728)_{10} = (2D8)_{16} = (2{:}330)_8$$

and

$$(120)_{10} = (78)_{16} = (178)_8$$

2:330 is the so-called "split octal" representation. Numbers larger than 255 must use at least two bytes of storage. 330 is the value that occupies the lower of the two bytes while 2 is the value in the high order byte. The method of conversion is as follows: calculate how many times 256 will divide the number. This is the octal number to the left of the colon. The remainder is the octal number to the right of the colon. In the present case, 256 divides 728 twice with a remainder of 216. 216 has an octal representation of 330 $(3\cdot64 + 3\cdot8 + 0\cdot1)$.

The D-E register pair must be loaded with the length of data block which is 2:330 in octal or 208 in hex. The H-L register pair must be initialized with the starting address of the data block which has the split octal representation of 7:170 and the hex representation of 778.

Write operations do not use the secondary command register so the only other register that must be initialized is the A register -- the COPE command register. The motion should be terminated at the end of the operation so bit 7 should be a one. For all commands, bits 6, 5, and 4 must be zero. Tape channels two and three are to be active so bits 3 and 2 should be a one. Bit 1 should be a zero because tape channel 1 is not active. Finally, bit 0 must be a one so that COPE will perform a write operation. The bit pattern of A is summarized below:

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
1	0	0	0	1	1	0	1

This bit pattern corresponds to the octal expression 215 and the hex expression 8D. The initialization procedure to write the above data onto the tape can be expressed in a sequence of 8080 instructions.

```
       Hex                          Octal

3E 8D       MVI    A,8D      076 203       MVI    A,203        INITIALIZE CMD REG
11 08 02    LXI    D,208     021 330 002   LXI    D,2:330      INITIALIZE BLK LENGTH REG
21 78 07    LXI    H,778     041 178 007   LXI    H,7:178      INITIALIZE START ADDR REG
CD 0A 81    CALL   COPE      315 012 201   CALL   COPE         EXECUTE COPE OPERATION
```

The particular order of the first three instructions is not important. The CALL COPE instruction sets the operation in motion and must occur last. A word of warning concerning CALL instructions: it is vital that the 8080 CPU stack register hold an address which points to valid RAM memory whenever a CALL instruction is executed. Subroutines usually end with a return instruction. This is a special type of branch -- the branch address is furnished by the memory locations pointed to by the stack register. Under normal circumstances a CALL instruction has stored a return address in these two locations. If the stack pointer points to ROM or non-existent memory, a CALL instruction cannot store the correct return address.

Example 2: Test the write circuitry of the I/O board. In this example, test data will be written from the RAM of the I/O board onto tape channel 1. The test data along with the addresses is presented below:

Octal

```
202:100   000   001   002   003   004   005   006   007
202:110   010   011   012   013   014   015   016   017
202:120   020   021   022   023   024   025   026   027
202:130   030   031   032   033   034   035   036   037
202:140   040   041   042   043   044   045   046   047
202:150   050   051   052   053   054   055   056   057
```

Hex

```
8240  00  01  02  03  04  05  06  07  08  09  0A  0B  0C  0D  0E  0F
8250  10  11  12  13  14  15  16  17  18  19  1A  1B  1C  1D  1E  1F
8260  20  21  22  23  24  25  26  27  28  29  2A  2B  2C  2D  2E  2F
```

As in example 1, bit 7 of the A register should be 1 and bits 6, 5 and 4 must be 0. Because only channel 1 will be active, bits 3 and 2 are zero while bit 1 is one. Finally, since the operation is to write data onto tape, bit 0 is also 1. Thus the A register which is the COPE command register must have the following pattern:

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
1	0	0	0	0	0	1	1

This corresponds to the octal number 203 or the hex number 83. Again, as in example 1, because the operation is writing data onto tape, it is not necessary to initialize register C, the secondary command register. The starting address of the test data as outlined above is 202:100 octal or 8240 hex. The data block has a length of 60 octal or 30 hex.

The set of instructions to write the test data out on channel one's tape recorder will conveniently fit at the beginning of the I/O boards RAM: 202:000 octal (8200 hex).

Octal

```
202:000    061 370 202    LXI  SP,202:370    INITIALIZE STACK POINTER
202:003    076 203        MVI  A,203         INITIALIZE COMMAND REG
202:005    021 060 000    LXI  D,60          LOAD THE DATA BLOCK LENGTH
202:010    041 100 202    LXI  H,202:100     LOAD STARTING ADDR OF BLOCK
202:013    315 012 201    CALL COPE          EXECUTE COMMAND
202:016    303 016 202    JMP 202:016        COME TO DYNAMIC HALT
```

Hex

```
8200       31 F8 82       LXI  SP,82F8
8203       3E 83          MVI  A,83
8205       11 30 00       LXI  D,30
8208       21 40 82       LXI  H,8240
820B       CD 0A 81       CALL COPE
820E       C3 0E 82       JMP  820E
```

Load this program into the RAM of the I/O board starting at the indicated address. Connect cables from the I/O board to a suitable audio cassette player/recorder. Before connecting the motion control cable, load a cassette tape into the unit, position the tape past the blank leader, reset the digital counter on the unit to zero and connect the motion control cable. Next, depress the play/record buttons. Start the program. The tape will begin to move and will stop after a short while. When the tape stops, the above program will have completed the CALL COPE instruction and will be executing the JMP 820E dynamic halt instruction. Halt the the computer but do not turn it off. Remove the motion control and external speaker plugs from the cassette player/recorder.

Rewind the tape to the point where the recorded data started. Press the play button. There should be a steady 2400 Hz tone that lasts approximately ten seconds. The data should follow the tone and will not last more than several seconds. After verifying the presence of the recorded data, rewind the tape so that the head is positioned over the beginning of the leader tone. Reconnect the motion control and the external speaker cables.

Example 3: Test the read circuitry of the I/O board by reading the test data written in Example 2.

The bit pattern of the A register is exactly the same as in Example 2 except that bit 0 is a zero because the desired operation is a read and not a write. Thus, the A register has the following pattern:

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
1	0	0	0	0	0	1	0

This pattern corresponds to the octal number 202 or the hex number 82. Because the command is not a write operation, it is necessary to initialize the secondary command register which is the C register. This register must be initialized to zero since the desired operation is not a verify or move.

For comparison purposes, the data on the tape should be read into an area of RAM memory different than where the data of the previous example was stored. A convenient place in the I/O board RAM is between locations 202:200 and 202:257 octal. The set of instructions to read the previously written test data back into the I/O board RAM starting at location 202:200 octal is presented below:

Octal

```
202:030    061 370 202    LXI  SP,202:370    INITIALIZE STACK POINTER
202:033    076 202        MVI  A,202         INITIALIZE COMMAND REGISTER
202:035    016 000        MVI  C,000         SET UP SECONDARY COMMAND REG
202:037    021 060 000    LXI  D,60          DATA BLOCK LENGTH
202:042    041 200 202    LXI  H,202:200     DATA BLOCK STARTING ADDRESS
202:045    315 012 201    CALL COPE          EXECUTE READ COMMAND
202:050    303 050 202    JMP  202:050       DYNAMIC HALT
```

Hex

```
8218       31 F8 82       LXI  SP,82F8
821B       3E 82          MVI  A,82
821D       0E 00          MVI  C,0
821F       11 30 00       LXI  D,30
8222       21 80 82       LXI  H,8280
8225       CD 0A 81       CALL COPE
8228       C3 28 82       JMP  8228
```

Enter this program into the RAM of the I/O board starting at the indicated address. Adjust the volume setting of the cassette player at approximately 4/5 of the full volume. Occasionally, it will be necessary to experiment with the volume setting. If there is a tone control on the cassette unit, it should be set to maximum treble. Depress the PLAY button of the cassette and start the program. The tape will start to move and in a short while, it will stop and the COPE command will be done. Stop the computer and press the stop button of the cassette unit. Examine locations 202:200 through 202:257 octal. The data should have the following pattern:

Octal

```
202:200     000   001   002   003   004   005   006   007
202:210     010   011   012   013   014   015   016   017
202:220     020   021   022   023   024   025   026   027
202:230     030   031   032   033   034   035   036   037
202:240     040   041   042   043   044   045   046   047
202:250     050   051   052   053   054   055   056   057
```

Hex

```
8280    00  01  02  03  04  05  06  07  08  09  0A  0B  0C  0D  0E  0F
8290    10  11  12  13  14  15  16  17  18  19  1A  1B  1C  1D  1E  1F
82A0    20  21  22  23  24  25  26  27  28  29  2A  2B  2C  2D  2E  2F
```

If the data is not correct, experiment with the volume setting on the cassette player and restart the program. If errors persist, return the Speakeasy for service.

Example 4: Verify the data on the tape against data in memory.

Be sure that the data in the RAM memory between locations 202:100 and 202:157 is as described in Example 2 above, i.e., 000 through 057 octal. With the motion control cable removed, position the tape in the cassette player to the beginning of the leader tone that was recorded in Example 2. Replace the motion control cable, adjust the volume setting and depress the play button.

The command register should be initialized just as it was in Example 3. The starting address register and the data block length should be set just as they were in Example 2. However, the secondary command register (register C) needs to be configured for a verify operation. Bit 6 is the verify bit, therefore, the C register should have the following pattern:

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	1	0	0	0	0	0	0

This pattern corresponds to the octal number 100 or the hex 40.

The results of verify operations are stored at location 203:365 octal or 83F5 hex and has the label DERR which stands for Data ERRor. The COPE software writes in this location only if it encounters data on the tape which is different from corresponding data in memory. If an error is found, COPE loads DERR with some non-zero value. Before a verify operation, DERR should be initialized to zero. At the end of a verify operation, DERR should be checked. If DERR still has the value zero, then the verify was successful. If it is a non-zero, a compare error was detected and the data on the tape may be incorrect.

The following program contains the necessary instructions to verify the tape data against the data stored between locations 202:100 and 202:157 octal.

Octal

```
202:054      061 370 202     LXI  SP,202:370      INITIALIZE STACK POINTER
202:057      076 202         MVI  A,202           INITIALIZE COMMAND REGISTER
202:061      016 100         MVI  C,100           LOAD C WITH VERIFY COMMAND
202:063      021 060 000     LXI  D,60            INITIALIZE BYTE COUNT REG
202:066      041 100 202     LXI  H,202:100       INITIALIZE BLOCK STARTING ADDR
202:071      315 012 201     CALL COPE            EXECUTE COMMAND
202:074      303 074 202     JMP  202:074         DYNAMIC HALT
```

Hex

```
822C         031 F8 82       LXI  SP,82F8
822F         3E 82           MVI  A,82
8231         0E 40           MVI  C,40
8233         11 30 00        LXI  D,30
8236         21 40 82        LXI  H,8240
8239         CD 0A 81        CALL COPE
823C         C3 3C 82        JMP  823C
```

Examine location 203:365 octal (83F5 hex). This is the Data Error location DERR. Store a zero in this location. Enter the above program starting at location 202:054 octal (822C hex). Start the program. The tape will start moving and continue until the compare operation is finished. When the tape stops, halt the computer and examine DERR. The value of this memory location should still be zero.

Reposition the tape to the beginning tone leader and then examine location 202:120. Change this value from 20 to 377. Restart the program at location 202:054 octal (822C hex). Again the tape will start moving and continue until the verify operation has been completed. When the tape stops, halt the computer and re-examine location 203:365 octal (83F5 hex). DERR should now have a value other than zero (357 in this instance).

BYTE ORIENTED DATA TRANSFERS

In the applications discussed previously, the data managed by COPE was always in blocks many bytes long and transferred as a whole rather than a single byte at a time. There are applications, however, where the length of a block of data on a tape is not known before hand. The end of a block of data is usually marked with a special code which corresponds to an "end of file" mark. In this situation, a block transfer is not practical. Data must be transferred a byte at a time and each byte must be tested to see if the transfer is finished.

In effecting this type of transfer, COPE will still be transferring blocks of data. However, each block of data will be only one byte long and the command register will be modified so that the tape motion is not turned off in between transfers.

For a read operation from, say, channel 1 where the tape is <u>not</u> to be turned off after the command has finished, the command register has the pattern:

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	0	0	0	0	0	1	0

As with a normal read, the C register is initialized to zero.

Assuming that the file in question is made up of ASCII characters which have values of octal 40 or greater, the ASCII code 004, call EOT (end of tape), will be used for an "end of file" mark for the data. The following program will transfer data from the tape cassette unit to memory starting at location 0.

Octal

```
203:000  061 370 203   START  LXI  SP,203:370   INITIALIZE STACK POINTER
203:003  041 000 000          LXI  H,0          INITIALIZE DATA POINTER
203:006  345           LOOP   PUSH H            SAVE THE DATA POINTER
203:007  021 001 000          LXI  D,1          SET UP FOR 1 BYTE TRANSFER
203:012  076 002              MVI  A,2          LOAD COMMAND REGISTER
203:014  016 000              MVI  C,0          INITIAL SECONDARY CMD REG
203:016  041 047 203          LXI  H,TEMP       LOAD BYTE TRANSFER ADDR
203:021  315 012 201          CALL COPE         GET BYTE OF DATA FROM TAPE
203:024  072 047 203          LDA  TEMP         GET TAPE DATA BYTE IN ACC
203:027  376 004              CPI  EOT          COMPARE END OF FILE MARK
203:031  341                  POP  H            RESTORE STACK/GET DATA PNTR
203:032  312 042 203          JZ   DONE         TEST FOR END OF FILE
203:035  167                  MOV  M,A          MOVE DATA TO MEMORY
203:036  043                  INX  H            INCREMENT DATA POINTER
203:037  303 006 203          JMP  LOOP         GO GET MORE DATA
203:042  076 000       DONE   MVI  A,0
203:044  323 004              OUT  TAPE         TURN OFF TAPE MOTION
203:046  311                  RET               RETURN
203:047  000           TEMP   000               DATA CELL WHERE COPE
                                                  TRANSFERS DATA BYTE
```

```
Hex

8300      31 F8 83      START   LXI   SP,83F8
8303      21 00 00              LXI   H,0
8306      E5            LOOP    PUSH  H
8307      11 01 00              LXI   D,1
830A      3E 02                 MVI   A,2
830C      0E 00                 MVI   C,0
830E      21 27 83              LXI   H,TEMP
8311      CD 0A 81              CALL  COPE
8314      3A 27 83              LDA   TEMP
8317      FE 04                 CPI   EOT
8319      E1                    POP   H
831A      CA 22 83              JZ    DONE
831D      77                    MOV   M,A
831E      23                    INX   H
831F      C3 06 83              JMP   LOOP
8322      3E 00         DONE    MVI   A,0
8324      D3 04                 OUT   TAPE
8326      C9                    RET
8327      00            TEMP    000
```

THE CHECK SUM SOFTWARE

Oncedata is stored on some bulk magnetic media such as a disk or tape, it is always possible that through some fault in the media or from an accidental exposure to an intense magnetic field, the original data has been altered. It is also possible that the read electronics of the device(s) which transfer data from the media to the computer's memory are faulty or even worse periodically faulty. If a program stored on tape is read into memory with errors, it probably will not run, and so it is loaded again and usually the second or third time it goes in correctly. This is annoying but not disastrous. However, if a payroll file were to be read in with errors such that several employees were in effect given large raises, this might be considered disastrous. The field of error correcting and detecting codes and algorythms has received a good deal of study and experimentation for these and other reasons. Error correcting codes are quite complex and are beyond the scope of the ROM software on the I/O board. However, there is a short routine included in the ROM called CHECK whose starting address 201:115 octal (814D hex). This routine can be used to detect a wide variety of read errors which may occur in transferring data from the audio cassette player to the computer's memory. This section is devoted to explaining how to use the subroutine CHECK to enhance the integrity of data stored on audio cassette tapes.

CHECK is used to generate a "check sum" over a data block which is to be transferred to or from memory. This check sum is two bytes long. The most straightforward way to use CHECK is as follows:

(1) Just before writing a data block out on tape, generate a check-sum word over the data block with CHECK and then append these two bytes to the end of the data.

(2) Write augmented data block on the tape.

(3) Whenever the block of data is read into the computer, generate another check sum over all the data <u>except</u> the last two bytes.

(4) Compare the new check sum with the original check sum which is stored in the last two bytes of the data block.

The CHECK subroutine generates check sums over blocks of data. Just as with the COPE software, CHECK requires that the starting address of the data block be in the H-L register pair and that the block length be in the D-E register pair. When CHECK is through, the check sum is in the H-L register pair -- high order byte in H and low order byte in L. The D-E register pair has the address of the <u>next</u> location above the data block. This makes it easy to append the check sum to a data block or to compare a new check sum with an old one. Below are two examples of subroutines which use CHECK. The first appends a check sum to a data block and the second compares a new check sum with an old one at the end of the block. The zero flag is set if the two check sums agree. In each case, the data block is 4K (4096) bytes long and starts at location 0.

```
Octal                               I

041 000 000     APPEND:  LXI  H,0          INITIALIZE STARTING ADDR
021 000 020              LXI  D,20:000     INITIALIZE BLOCK LENGTH
315 115 201              CALL CHECK        COMPUTER CHECK SUM
353                      XCHG              XCHG CHECK SUM W/ ADDR
163                      MOV  M,E          STORE LOW ORDER BYTE OF CS
043                      INX  H            INCREMENT ADDR POINTER
162                      MOV  M,D          STORE HIGH ORDER BYTE OF CS
311                      RET               RETURN
```

```
Hex

21 00 00        APPEND:  LXI  H,0
11 00 10                 LXI  D,1000
CD 4D 81                 CALL CHECK
EB                       XCHG
73                       MOV  M,E
23                       INX  H
72                       MOV  M,D
C9                       RET
```

```
Octal                               II

041 000 000     COMPAR   LXI  H,0          INITIALIZE STARTING ADDR
021 000 020              LXI  D,20:000     INITIALIZE BLOCK LENGTH
315 115 201              CALL CHECK        COMPUTE NEW CHECK SUM
353                      XCHG              XCHG CHECK SUM W/ ADDR
173                      MOV  A,E          MOVE NEW LOW BYTE TO A
276                      CMP  M            COMPARE W/ OLD LOW BYTE
300                      RNZ               RETURN IF ERROR
043                      INX  H            INCREMENT ADDR
172                      MOV  A,D          MOVE NEW HIGH BYTE TO A
276                      CMP  M            COMPARE W/ OLD HIGH BYTE
311                      RET               RETURN
```

```
Hex

21 00 00        COMPAR  LXI  H,0
11 00 10                LXI  D,1000
CD 4D 81                CALL CHECK
EB                      XCHG
7B                      MOV  A,E
BE                      CMP  M
C0                      RNZ
23                      INX  H
7A                      MOV  A,D
BE                      CMP  M
C9                      RET
```

THE SOFTWARE UART

INTRODUCTION

The serial port of the I/O board has been designed to interface to a teletype with a paper tape reader and any serial RS232 device. The ADM3-A terminal from Lear Seigler or the Hazeltine 1500 are examples of relatively inexpensive RS232 video terminals which interface directly to the serial port.

Many I/O boards have a serial port -- it is a very useful interface to have on an I/O board. However, most I/O boards with serial ports have an integrated circuit called a UART connected to the port. It takes care of the serial-to-parallel conversion for incoming data from a terminal and the parallel-to-serial conversion for outgoing data from the computer to the terminal. The Speakeasy does not have a UART; instead, it has several pieces of software in its PROM which together constitute a software UART. This intelligent parallel-to-serial and serial-to-parallel converter has a great deal of flexibility and in many ways is superior to most hardware UARTS. Unfortunately, it is not more flexible in all respects.

An integrated circuit UART is actually two separate pieces of hardware in a single package. The parallel-to-serial conversion and the serial-to-parallel conversion functions of a UART are independent of each other. On many UARTS these two functions even have provision for separate clocks so that the two processes can take place at different rates. The parallel-to-serial and serial-to-parallel functions of the software UART are also separate pieces of code. Unlike the hardware UART, however, only one of these processes can take place at a given time since the CPU itself is the prime mover in the process. The consequence of this is that for slow devices such as a teletype it is possible that during output, an input character from from the keyboard may be missed. This would be a serious problem if it were not for an important option that is built into the input (serial-to-parallel) software. The input software has the ability to echo input as it is collected so that during input it is not necessary to do any output (parallel-to-serial). At data rates of 300 baud and better, characters are collected (serial-to-parallel) and echoed (parallel-to-serial) faster than most humans can type so that it is virtually impossible to miss input during output. At these speeds the two separate processes of the software UART appear to someone at a terminal to be occuring simultaneously.

INPUT

The input software of the PROM UART allows for several options while it is collecting input data. A paper tape reader of a TTY can be turned on and characters can be echoed <u>while</u> they are being collected. The B register of the CPU is used to enable these options. The starting address of the serial input routine is 201:157 (816F hex) and is executed using a CALL INPUT instruction. The input character is returned in the D register. The input routine uses all the CPU registers and six levels of the stack.

OUTPUT

The output routine of the software UART expects that the parallel data to be sent out to the terminal is in the accumulator. The starting address of the output routine is 201:263 (81B3 hex). As with the input routine all the CPU registers are used along with four levels of the stack.

THE SPEED CONSTANT SCON

The software UART can run at any speed between 0 and 2400 baud (bits per second). The baud rate is set by initializing locations 203:363 and 203:364 (83F3 and 83F4). 203:363 has the label SCON which stands for Serial CONstant. Both SCON and SCON+1 must be initialized before a valid baud rate for the software UART is determined. The formula for calculating SCON and SCON+1 is given below:

$$\text{SCON} = \frac{1{,}000{,}000}{K \times (\text{baud rate})}$$

For baud rates less than 600, K has the value 53 (decimal). For rates more than 600, K should be increased to 57. Below are some values for SCON for the more common baud rates:

Baud Rate	SCON+1, SCON
110	000:253 (00 AA hex)
300	000:075 (00 3D hex)
1200	000:016 (00 0E hex)
1800	000:011 (00 09 hex)
2400	000:006 (00 06 hex)

Some terminals may require a slight change from the values listed above which are average quantities.

Examples

Listed below is a short program ideal for testing the input and output circuitry of the serial port of the I/O board. This is a very low level routine which does not use SCON and has the effect of making a half duplex terminal out of a full duplex one.

```
    Octal                Hex

202:000  333 005       8200  DB 05        START  IN 5    GET THE INPUT
202:002  323 005       8202  D3 05               OUT 5   ECHO INPUT
202:004  303 000 202   8204  C3 00 82            JMP START
```

When the terminal is connected to the serial port and this program is running, whenever a key is pressed on the keyboard, it will be echoed on the screen or by the printer.

When bit 0 of register B is a zero, the input routine will automatically echo input. When bit 7 of register B is a 1, the TTY's paper tape reader will be turned on until a character is collected. It is always turned off when a character is collected even if some other piece of code has turned the reader on. Following is a program which exercises both the input and output routines. The speed constant is set for 300 baud.

Octal

```
202:000   061 000 203   START  LXI   SP,203:000   INITIALIZE STACK POINTER
202:003   041 075 000          LXI   H,75         INITIALIZE
202:006   042 363 203          SHLD  SCON           SCON
202:011   006 002       LOOP   MVI   B,1          SUPPRESS AUTOMATIC ECHO
202:013   315 157 201          CALL  INPUT        GO GET INPUT
202:016   172                  MOV   A,D          TRANSFER INPUT CHARAC TO A
202:017   315 263 201          CALL  SROUT        GO ECHO INPUT
202:022   303 011 202          JMP   LOOP         GO GET MORE INPUT
```

Hex

```
8200      31 00 83      START  LXI   SP,8300
8203      21 3D 00             LXI   H,3D
8206      22 F3 83             SHLD  SCON
8209      06 01         LOOP   MVI   B,1
820B      CD 6F 81             CALL  INPUT
820E      7A                   MOV   A,D
821F      CD B3 81             CALL  SROUT
8212      C3 09 82             JMP   LOOP
```

When calling SROUT, the output routine, the character to be printed must be in the A register. When calling INPUT, the character entered from the keyboard is returned to the D register. The program below will perform just like the previous example. The difference between the two is that the CALL SROUT is missing but the INPUT routine is now enabled to echo characters as they are collected.

Octal

```
202:000   061 000 203   START  LXI   SP,203:000
202:003   041 075 000          LXI   H,75
202:006   042 363 203          SHLD  SCON
202:011   006 000       LOOP   MVI   B,0
202:013   315 157 201          CALL  INPUT
202:016   303 011 202          JMP   LOOP
```

Hex

```
8200      31 00 83      START  LXI   SP,8300
8203      21 3D 00             LXI   H,3D
8206      22 F3 83             SHLD  SCON
8209      06 00         LOOP   MVI   B,0
820B      CD 6F 81             CALL  INPUT
820E      C3 09 81             JMP   LOOP
```

THE SERIAL STATUS FLIP-FLOP

Associated with the serial port is a status flip-flop which is accessed through the status port. This flip-flop is connected to bit 0 of input device 6. The serial status is set whenever a logical zero is present on either the TTY input or the RS232 input. Serial devices transmit a continuous stream of ones or rest bits when they are idle. When a serial device is ready to transmit data, it sends out one zero bit (start bit), seven or more data bits and one or more rest bits. The serial status flip-flop is a convenient signal to test whenever a program needs to know if a key (any key) has been struck on the keyboard. If a key has been struck,

```
333 006   IN  6
346 001   ANI 1
```

will reset the zero flag. If no key has been struck, the two instructions will set the zero flag. The status flip-flop is reset whenever an IN 5 instruction is executed.

THE PARALLEL PORT

The parallel port is perhaps the simplest of the four ports on the I/O board. It is a bidirectional port: data may be transmitted or received. There is a 1 kΩ pull-up resistor connected to pin #5 of J1 which disables the transmitters of the DM8835 transceivers. The default direction of the port is "input". This pin should be grounded if the port is to function as an output port. Associated with the output port is an 8-bit latch. This latch stores the most recent data sent to the parallel output port. There is a status bit of device 6 also associated with the parallel port. Pin #13 of J1 is connected to the clock input of a flip-flop. When a positive pulse is received at this pin, the status flip-flop for the port is set. When an IN 7 instruction is executed by the processor, the flip-flop is reset.

This port is called an inverting parallel port. The DM8835 transceivers buffering information from the outside world invert during both transmission and reception. Because of the higher noise margins, peripheral devices frequently send inverted data rather than positive data. When this is the case, the receivers of the DM8835s will automatically invert the data back to its natural positive state. If the peripheral transmits positive data, it will have the wrong polarity when an IN 7 instruction is executed. If this is the case, the IN 7 instruction should be followed by a CMA instruction to invert the data to its positive state. Below are two small routines -- the first for input and the second for output -- which illustrate the way a typical peripheral connected to the port should be accessed.

Input Routine
(Input data in A)

```
     Octal                 Hex

202:100  333 006       8240  DB 06      PIN   IN 6      GET STATUS
    102  346 002       8242  E6 02            ANI 2     STRIP OFF PARA PORT BIT
    104  312 100 202   8244  CA 40 82         JZ PIN    TEST IF DATA READY
    107  333 007       8247  DB 07            IN 7      GET DATA/RESET STATUS
    111  311           8249  C9               RET       RETURN
```

Output Routine
(Output data in B)

```
202:100  333 006       8240  DB 06      POUT  IN 6      GET STATUS
    102  346 002       8242  E6 02            ANI 2     STRIP OFF PARA PORT BIT
    104  312 100 202   8244  CA 40 82         JZ POUT   TEST FOR OUTPUT READY
    107  170           8247  78               MOV A,B   MOVE OUTPUT DATA TO ACC
    110  323 007       8248  D3 07            OUT 7     TRANSMIT DATA
    112  333 007       824A  DB 07            IN 7      RESET STATUS
    114  311           8246  C9               RET       RETURN
```

A final example is the routine to access data from an Oliver OAE paper tape reader connected to the parallel port:

```
202:200   333 006       8280   DB 06      OPIN IN 6    GET STATUS
202:202   346 002       8282   E6 02           ANI 2   STRIP PARALLEL PORT BIT
202:204   312 200 202   8284   CA 80 82        JZ OPIN TEST FOR INPUT READY
202:207   333 007       8287   DB 07           IN 7    GET INPUT DATA
202:211   057           8289   2F              CMA     COMPLIMENT
202:212   311           828A   C9              RET     RETURN
```

BOOTSTRAP SOFTWARE AND FILE STRUCTURES

INTRODUCTION

Almost all of the modern micro-processor based computers use semi-conductor memory exclusively. This type of memory has many advantages but has one major short coming when compared with "magnetic core" memory -- semi-conductor memory cannot remember the information stored in it when power was turned off. This means programs must be reloaded whenever power is turned off. The Speakeasy I/O board was designed to eliminate this problem.

There is a short routine contained in the COPE ROM called "BOOTS" which has the ability to automate the process of program loading and initialization. BOOTS stands for Bootstrap and the routine has been designed to perform system initialization by utilizing the COPE tape handling software. Specifically, this routine automatically loads a 256 byte record from the tape into the first page of the RAM on the I/O board. After this record is loaded, BOOTS forces an unconditional branch to the first location of this RAM. Just loading a short record and forcing a branch will not by itself perform system initialization. On the other hand, if the record that BOOTS loads is really a program which contains loading parameters pertaining to yet another program which follows immediately on the tape, then the process of initialization can be successfully completed.

The BOOTS routine is in ROM and always does the same task. The desired program that is to be loaded and started such as BASIC or ATE is on the second record of the tape and must be loaded. The bridge between BOOTS and the main program is the first 256 byte record/program on the tape that is loaded and then started by BOOTS. The structure of this record is the main topic of this section. This record has been designed both to facilitate program loading and to implement a file structure for records on audio cassette tapes.

The Speakeasy I/O board can be used in a variety of applications without any file structure. However, experience has shown that the board is even more effective when some type of file structure is implemented. Below is an example (suggestion) which can be ignored, followed, or used as a starting point for a more elaborate and effective file structure.

HEADER STRUCTURE

The 256 byte record that BOOTS loads and that is in the front of any program on tape will be called a "header" or "record header." This header contains instructions for reading and writing records using COPE. It also contains information pertaining to the record that it proceeds. The information implements the file structure and will be discussed first. The instructions along with a "map" of the header will be presented afterwards.

The header contains the following file dependent information:

(1) A checksum (2 bytes) for error detection.

(2) The load address in memory (2 bytes) for the record.

(3) The length of the record (2 bytes).

(4) The record type (1 byte).

(5) The tape cassette write channel number (1 byte).

(6) The title type (1 byte).

(7) The title (130 bytes).

Checksum has the label CHKSM+1 and contains the two byte checksum generated by CHECK when the record is originally created.

Load address has the label LODAD. If the record is a program, the starting address should agree with the load address.

Record length has the label LNGTH.

Record type has the label TYPE. Allowance has been made for up to 250 different record types. If TYPE contains an ASCII B, the record contains binary data or a program. If TYPE contains an ASCII S, the record consists entirely of ASCII characters and generally would be source code for a program.

Tape cassette write channel number has the label WUNIT. WUNIT contains 203, 205, or 211 (83, 85, 89 hex) and is in actuality a command to COPE to write on channel 1, 2, or 3. This byte is present to facilitate the reproduction (with perhaps minor changes) of records without the aid of any higher level software.

Title type has the label TITYP. Some records have no title or have an encoded title. If TITYP contains an ASCII B, the record has a binary title which consists of the first 130 bytes of the record in binary. If TITYP contains an ASCII S, the record has a title consisting of ASCII characters.

Title has the label TITLE. If the title type is an ASCII S, the title consists entirely of ASCII characters. It may be as long as 130 bytes. The last character of the title is an ASCII carriage return which is the octal number 15 (hex ØD).

Below are the instructions of the header program/record which uses the file structure information presented earlier to load and reproduce records.

Octal

```
202:000    333 377        HEADR  IN SSW          GET SENSE SWITCHES
202:002    017                   RRC             TEST FOR SSW Ø
202:003    332 000 203           JC HEADR        WAIT IF SSW Ø ON
202:006    061 066 203           LXI SP, TSTAK   INITIALIZE THE STACK
202:011    315 031 203           CALL LOADR      CALL LOAD RECORD ROUTINE
202:014    302 014 203    STOP1  JNZ STOP1       TEST FOR ERRORS
202:017    333 377        NOGO   IN SSW          GET SENSE SWITCHES
202:021    007                   RLC             TEST FOR SSW 7
202:022    332 017 203           JC NOGO         WAIT IF SSW 7 ON
202:025    052 000 203           LHLD VERAD      GET FIRST ADDRESS OF RECORD
202:030    351                   PCHL            BRANCH THERE
202:031    052 167 203    LOADR  LHLD LODAD      GET THE LOAD ADDRESS
202:034    042 000 203           SHLD VERAD      SAVE FOR LATER VERIFY
202:037    315 062 203           CALL LREGS      LOAD UP THE REGISTERS
202:042    315 136 203           CALL COPER      GET READY TO READ RECORD
202:045    300                   RNZ             RETURN IF STATUS ERROR
202:046    315 115 201           CALL CHECK      COMPUTE CHECK SUM
202:051    021 000 000    CHKSM  LXI D,Ø         LOAD UP OLD CHECK SUM
202:054    174                   MOV A,H         GET LOW BYTE OF CHECK SUM
202:055    272                   CMP D           COMPARE WITH OLD LOW BYTE
202:056    300                   RNZ             RETURN IF NO COMPARE
202:057    175                   MOV A,L         GET HIGH BYTE OF CHECK SUM
202:060    273                   CMP E           COMPARE WITH OLD HIGH BYTE
202:061    311                   RET             RETURN
202:062    052 171 203    LREGS  LHLD LNGTH      LOAD THE D-E REGISTER
202:065    352                   XCHG              PAIR WITH RECORD LENGTH
202:066    052 000 203           LHLD VERAD      LOAD H-L WITH STARTING ADDR
202:071    076 202               MVI A,202Q      INITIALIZE A & C WITH
202:073    016 000               MVI C,Ø           A READ COMMAND
202:075    311                   RET             RETURN
202:076    315 104 202    TAPEW  CALL TAPE       GO WRITE HEADER & RECORD
202:101    303 101 202    STOP2  JMP STOP2       DYNAMIC HALT
202:104    315 062 202    TAPE   CALL LREGS      LOAD UP CPU REG FOR COPE
202:107    315 115 201           CALL CHECK      COMPUTE CHECK SUM
202:112    042 052 202           SHLD CHKSM+1    SAVE CHECK SUM
202:115    041 000 202           LXI H,HEADR     STARTING ADDR FOR HEADER
202:120    021 000 001           LXI D,1:000     LENGTH OF HEADER
202:123    072 174 202           LDA WUNIT       WRITE COMMAND
202:126    365                   PUSH PSW        SAVE FLAGS & ACCUMULATOR
202:127    315 136 202           CALL COPER      GO WRITE THE HEADER
202:132    315 062 202           CALL LREGS      LOAD UP CPU REG FOR COPE
202:135    361                   POP PSW         RESTORE FLAGS & ACC
202:136    365            COPER  PUSH PSW        SAVE FLAGS & ACC
202:137    257                   XRA A           CLEAR ACC
202:140    062 365 203           STA DERR        INITIALIZE DATA ERROR CELL
202:143    062 366 203           STA SERR        INITIALIZE DATA ERROR CELL
202:146    361                   POP PSW         RESTORE FLAGS & ACC
202:147    345                   PUSH H          SAVE STARTING ADDRESS
202:150    335                   PUSH D          SAVE BLOCK LENGTH
202:151    315 012 201           CALL COPE       GO DO THE TAPE COMMAND
202:154    331                   POP D           RESTORE BLOCK LENGTH
202:155    341                   POP H           RESTORE STARTING ADDRESS
```

```
202:156      072 366 203             LDA SERR        GET THE STATUS ERROR CELL
202:161      107                     MOV B,A         SAVE IN B
202:162      072 365 203             LDA DERR        GET THE DATA ERROR CELL
202:165      250                     ORA B           MERGE ERROR DATA
202:166      311                     RET             RETURN
202:167      000 000         LODAD   DW 0            LOAD ADDRESS WORD
202:171      000 000         LENGTH  DW 0            BLOCK LENGTH WORD
202:173      000             TYPE    DW 0            FILE TYPE
202:174      203             WUNIT   DB 203Q         UNIT NUMBER & WRITE CMD
202:175      102             TITYP   DB 102Q         TITLE TYPE
202:176      000             TITLE   DS 202Q         TITLE AREA
```

Hex

```
8200         DB FF           HEADR   IN SSW
8202         0F                      RRC
8203         DA 00 82                JC HEADR
8206         31 36 83                LXI SP,TSTAK
8209         CD 19 82                CALL LOADR
820C         C2 0C 82        STOP1   JNZ STOP1
820F         DB FF           NOGO    IN SSW
8211         07                      RLC
8212         DA 0F 83                JC NOGO
8215         2A 00 83                LHLD VERAD
8218         E9                      PCHL
8219         2A 77 82        LOADR   LHLD LODAD
821C         22 00 82                SHLD VERAD
821F         CD 32 82                CALL LREGS
8222         CD 5E 82                CALL COPER
8225         C0                      RNZ
8226         CD 4D 81                CALL CHECK
8229         11 00 00        CHKSM   LXI D,0
822C         7C                      MOV A,H
822D         BA                      CMP D
822E         C0                      RNZ
822F         7D                      MOV A,L
8230         BB                      CMP E
8231         C9                      RET
8232         2A 79 82                LHLD LNGTH
8235         EB                      XCHG
8236         2A 00 83                LHLD VERAD
8239         3E 82                   MVI A,82
823B         0E 00                   MVI C,0
823D         C9                      RET
823E         CD 44 82        TAPEW   CALL TAPE
8241         C3 41 82        STOP2   JMP STOP2
8244         CD 32 82        TAPE    CALL LREGS
8247         CD 4D 81                CALL CHECK
824A         22 2A 82                SHLD CHKSM
```

```
824D        21 00 82                      LXI H,HEADR
8250        11 00 01                      LXI D,100
8253        3A 7C 82                      LDA WUNIT
8256        F5                            PUSH PSW
8257        CD 5E 82                      CALL COPER
825A        CD 32 82                      CALL LREGS
825D        F1                            POP PSW
825E        F5                     COPER  PUSH PSW
825F        AF                            XRA
8260        32 F5 83                      STA DERR
8263        32 F6 83                      STA SERR
8266        F1                            POP PSW
8267        E5                            PUSH H
8268        D5                            PUSH D
8269        CD 0A 81                      CALL COPE
826C        D1                            POP D
826D        E1                            POP H
826E        3A F6 83                      LDA SERR
8271        47                            MOV B,A
8272        3A F5 83                      LDA DERR
8275        B0                            ORA B
8276        C9                            RET
```

THE HEADER PROGRAM

Once the header program is in memory, it should be preserved for later use with other programs. Below are the steps to save a copy of the header program so that it will auto-load and not try to load any record that follows it. This copy should be used only when it is necessary to create a file structure for a program or a data record.

(1) Enter the header program starting at location 202:000 (8200 hex).

(2) Enter 101 (41 hex) at location 202:167.

(3) Enter 202 (82 hex) at location 202:170.

(4) Enter 001 (01 hex) at location 202:171.

(5) Put a tape in the recorder connected to channel one and press record.

(6) Start the header program at location 202:076 (823E hex).

The recorder will start and will write out the header record itself on tape. Next, the program will write exactly one byte from location 202:101 of the header program. When BOOTS is used to read in the header record, regardless of the position of the sense switches, the header program will load and then come to a dynamic halt. If none of the sense switches are on, this halt will occur at location 202:101.

AUTO-LOADING PROGRAMS

With the header program in memory, any other program can be preserved on tape with auto loading capability by following the steps below:

(1) Enter the lowest address of the program at locations 202:167 and 202:170; the low order byte of the address is loaded at 202:167 (8277 hex) and the high order byte is entered at 202:170 (8278 hex).

(2) Enter the length of the program at locations 202:171 and 202:172. The length is a two byte value to allow programs of any length to be preserved. The low order byte is stored at location 202:171 (8279 hex) and the high order byte at location 202:172 (827A hex).

(3) Start the header program at location 202:076 (823E hex).

The header program will first record the header on the tape after a leader tone of about five seconds has been recorded. Another five second leader tone is recorded, followed by the main program.

An auto-loading program consists of two records: the first is the header record and the second is the program itself. The header record contains file parameter information and code to load the program which follows it. To load an auto-loading program, the bootstrap program BOOTS is executed. There are two sense switches used by the header program which allows a considerable degree of flexibility: sense switch 0 and sense switch 7.

Sense switch 0: If sense switch 0 is turned on while the header record is loading, the header program will not load the main record until sense switch 0 is turned off. This feature is very useful when a record needs to be placed in a different area than actually called for. After the header program loads and is waiting for sense switch 0 to be turned off, stop the program. Change locations 202:167 and 202:170 (8277 and 8278 hex) to the new load address. Restart the header program at location 202:000 (8200 hex). Turn on sense switch 7 just after restarting the header program.

Sense switch 7: When this switch is turned on, the header program will load the main record but will <u>not</u> allow the main program to execute. This is useful if it is necessary to set certain parameters of the main program before execution. It is also a necessary procedure if the main program's lowest address is not its starting address or if the main program is not really a program but a data file.

AN "INTEL HEX" LOADER

There is a substantial body of software available to the small computer user which is recorded on KC Standard cassette tapes and has a data format which conforms to the so-called Intellec Hex Paper Tape Format or "Intel Hex" format. The details of this format along with a program which will load "Intel Hex" formatted KC Standard tapes is presented below.

THE INTELLEC HEX PAPER TAPE FORMAT

In this format, two ASCII hexadecimal characters are used to represent a byte of data. Preceding the first data field and following the last data field, there must be a leader/trailer of at least 200 rest bits (approximately eight seconds of 2400 hz tone). Each record consists of six fields. The fields must occur in the order presented below.

1. RECORD MARK FIELD: Frame 0

The ASCII code for a colon (:) is used to signal the start of a record.

2. RECORD LENGTH FIELD: Frames 1 and 2

The number of data bytes in the record is represented by two ASCII hexadecimal digits in this field. The high order digit is in Frame 1. The maximum number of data bytes in a record is 255 (FF in hex). An end-of-file record contains two ASCII zeros in this field.

3. LOAD ADDRESS FIELD: Frames 3-6

The four ASCII hexadecimal digits in Frames 3-6 give the address at which the data is loaded. The high-order digit is in Frame 3, the low-order digit in Frame 6. The first data byte is stored in the location indicated by the load address; successive bytes are stored in successive memory locations. This field in an end-of-file record contains zeros or the starting address of the program.

4. RECORD TYPE FIELD: Frames 7 and 8

The two ASCII hexadecimal digits in this field specify the record type. The high order digit is in Frame 7. All data records are type 0; end-of-file records are type 1.

5. DATA FIELD: Frames 9 to $9 + 2^{(\text{record length}) -1}$

A data byte is represented by two frames containing the ASCII characters 0-9 or A-F, which represent a hexadecimal value between 0 and FF (0 and 255 decimal). The high order digit is the first frame of each pair. There are no data bytes in an end-of-file record.

6. CHECKSUM FIELD: Frames $9 + 2^{\text{(record length)}}$ to $9 + 2^{\text{(record length)}} + 1$

The checksum field contains the ASCII hexadecimal representation of the two's complement of the 8-bit sum of the 8-bit bytes that result from converting each pair of ASCII hexadecimal digits to one byte of binary, from the record length field to and including the last byte of the data field. Therefore, the sum of all the ASCII pairs in a record after converting to binary from the record length field to and including the checksum field is zero.

Format Example:

```
:130160000E0C119231CD40000E09119031CD40006E
:0A3170007E3196310100000092311B
:10317C0092310100963180008C31923100009631F1
:04318E0092319231B7
```

The Loader Program

This program will load Intel Hex format KC Standard tapes. If the loader encounters an error, it will stop at ERROR. Otherwise, the loader stops at DONE. This program must reside in RAM.

Octal

```
203:000   061 200 203   START  LXI   SP,TSTAK   INITIALIZE STACK POINTER
203:003   315 020 203          CALL  READ       READ THE RECORD
203:006   076 000              MVI   A,0        STOP THE
203:010   323 004              OUT   4            CASSETTE PLAYER
203:012   302 012 203   ERROR  JNZ   ERROR      STOP HERE ON ERROR
203:015   303 015 203   DONE   JMP   DONE       STOP HERE IF NO ERRORS

203:020   315 135 203   READ   CALL  RBYTE      READ A BYTE FROM TAPE
203:023   376 072              CPI   ":"        COMPARE WITH START OF REC
203:025   302 020 203          JNZ   READ       WAIT FOR BEGINNING OF REC
203:030   036 000              MVI   E,0        INITIALIZE CHECKSUM
203:032   315 075 203          CALL  CHTR       GET A CHARACTER PAIR
203:035   120                  MOV   D,B        RECORD LENGTH TO D
203:036   315 075 203          CALL  CHTR       GET A CHARACTER PAIR
203:041   140                  MOV   H,B        HIGH ORDER ADDR BYTE TO H
203:042   315 075 203          CALL  CHTR       GET ANOTHER CHARACTER PAIR
203:045   150                  MOV   L,B        LOW ORDER ADDR BYTE TO L
203:046   315 075 203          CALL  CHTR       GET RECORD TYPE FIELD
203:051   076 001              MVI   A,1        COMPARE RECORD TYPE
203:053   270                  CMP   B            TO END OF RECORD
203:054   310                  RZ                 IDENTIFIER
203:055   315 075 203   LOOP   CALL  CHTR       GET A DATA BYTE
203:060   167                  MOV   M,A        STORE IN MEMORY
203:061   043                  INX   H          ADVANCE ADDRESS POINTER
203:062   025                  DCR   D          DECREASE LENGTH COUNTER
```

```
203:063    302 055 203          JNZ  LOOP       GET MORE DATA IF NON-ZERO
203:066    315 075 203          CALL CHTR       GET CHECKSUM
203:071    300                  RNZ             STOP IF CHECKSUM ERROR
203:072    303 020 203          JMP  READ       READ NEXT RECORD IN

203:075    315 135 203   CHTR   CALL RBYTE      READ A BYTE FROM TAPE
203:100    315 123 203          CALL HEX        CONVERT TO HEX
203:103    027                  RAL             MULTIPLY
203:104    027                  RAL               BY
203:105    027                  RAL               HEXADECIMAL
203:106    027                  RAL               TEN
203:107    107                  MOV  B,A        SAVE IN B
203:110    315 135 203          CALL RBYTE      GET LOW ORDER ASCII DIGIT
203:113    315 123 203          CALL HEX        CONVERT TO HEX
203:116    260                  ORA  B          MERGE WITH HIGH ORDER BYTE
203:117    107                  MOV  B,A        SAVE IN B
203:120    303                  ADD  E          UPDATE CHECKSUM
203:121    137                  MOV  E,A        SAVE NEW CHECKSUM
203:122    311                  RET             RETURN

203:123    076 000       HEX    MVI  A,0        GET THE DATA BYTE
203:125    326 060              SUI  60Q        SUBTRACT BASE
203:127    376 012              CPI  10         COMPARE WITH TEN
203:131    330                  RC              RETURN IF LESS THAN TEN
203:132    326 007              SUI  7          SUBTRACT OFFSET
203:134    311                  RET             RETURN
203:135    305           RBYTE  PUSH B          SAVE THE
203:136    325                  PUSH D            STATE OF
203:137    345                  PUSH H            THE LOADER
203:140    041 124 203          LXI  H,HEX+1    INITIALIZE STORAGE POINTER
203:143    021 001 000          LXI  D,1        INITIALIZE BYTE COUNT
203:146    076 002              MVI  A,2        INITIALIZE CMD REG
203:150    113                  MOV  C,E        INITIALIZE C TO ZERO
203:151    315 012 201          CALL COPE       READ A BYTE FROM TAPE
203:154    341                  POP  H          RESTORE THE
203:155    321                  POP  D            STATE OF
203:156    301                  POP  B            THE LOADER
203:157    311                  RET             RETURN
```

Hex

```
8300       31 80 83      START  LXI  SP,TSTAK
8303       CD 10 83             CALL READ
8306       3E 00                MVI  A,0
8308       D3 04                OUT  4
830A       C2 0A 83      ERROR  JNZ  ERROR
830D       C3 0D 83      DONE   JMP  DONE

8310       CD 5D 83      READ   CALL RBYTE
8313       FE 3A                CPI  ":"
8315       C2 10 83             JNZ  READ
8318       1E 00                MVI  E,0
831A       CD 3D 83             CALL CHTR
831D       50                   MOV  D,B
831E       CD 3D 83             CALL CHTR
8321       60                   MOV  H,B
```

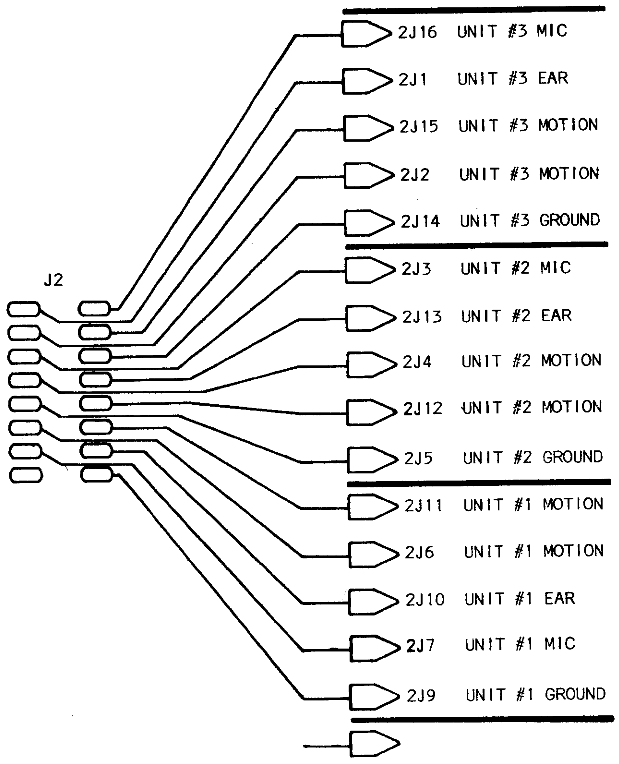
```
8322     CD 3D 83              CALL CHTR
8325     68                    MOV  L,B
8326     CD 3D 83              CALL CHTR
8329     3E 01                 MVI  A,1
832B     B8                    CMP  B
832C     C8                    RZ
832D     CD 3D 83      LOOP    CALL CHTR
8330     77                    MOV  M,A
8331     23                    INX  H
8332     1D                    DCR  D
8333     C2 2C 83              JNZ  LOOP
8336     CD 3D 83              CALL CHTR
8339     C0                    RNZ
833A     C3 10 83              JMP  READ
833D     CD 5D 83      CHTR    CALL RBYTE
8340     CD 53 83              CALL HEX
8343     17                    RAL
8344     17                    RAL
8345     17                    RAL
8346     17                    RAL
8347     47                    MOV  B,A
8348     CD 5D 83              CALL RBYTE
834B     CD 53 83              CALL HEX
834E     B0                    ORA  B
834F     47                    MOV  B,A
8350     83                    ADD  E
8351     5F                    MOV  E,A
8352     C9                    RET

8353     3E 00         HEX     MVI  A,0
8355     D6 30                 SUI  30H
8357     FE 0A                 CPI  10
8359     D8                    RC
835A     D6 07                 SUI  7
835C     C9                    RET
835D     C5            RBYTE   PUSH B
835E     D5                    PUSH D
835F     E5                    PUSH H
8360     21 54 83              LXI  H,HEX+1
8363     11 01 00              LXI  D,1
8366     3E 02                 MVI  A,2
8368     4B                    MOV  C,E
8369     CD 0A 81              CALL COPE
836C     E1                    POP  H
836D     D1                    POP  D
836E     C1                    POP  B
836F     C9                    RET
```

INTERFACE CABLES AND CONNECTION PROTOCOL

THE TAPE CASSETTE PORT

J2 is the cassette interface connector. It is a 16-pin DIP (dual-in-line) socket. Fifteen of the sixteen pins are used. Five connections are provided for each tape channel. The pin-out for this connector is detailed below.

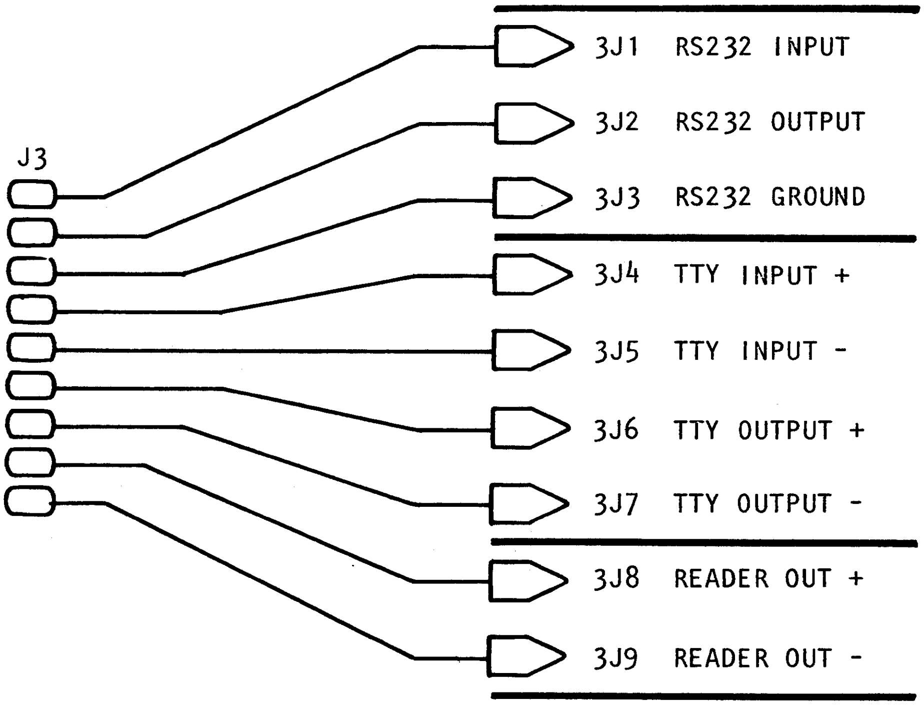


The input signal should come from the external speaker output of the cassette player/recorder. The output signal carries a wave form of approximately four volts peak and is much too high for the microphone input of the recorder. Instead, it should go to the auxiliary input. The two motion control outputs should be connected to the remote input of the cassette player/recorder. There are two choices for cables: flat ribbon or co-axial. If flat cable is used, it should terminate at one end with a DIP header (3M part #3416-0000 or equivalent). If co-axial cables are used, a 16-pin component carrier is a convenient way to terminate. There is one common ground connection which should be connected to both the auxiliary input ground connection and the external speaker output ground connection. (A 16-conductor flat cable with a DIP header is available from Morrow's to facilitate connecting J2 to a cassette player/recorder.)

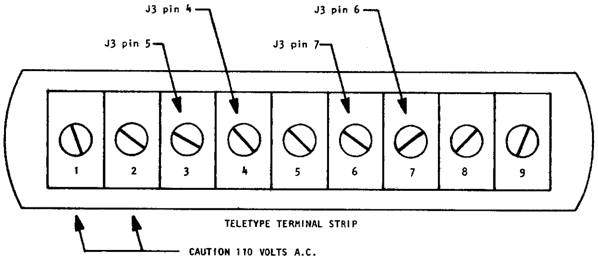
There are cassette players which have a ground loop between the ground connection of the external speaker jack and the auxiliary input jack. A ground loop between two common ground connections exists when there is a significant impedence between the two points. In a cassette player, ground loops generally are caused by component layout anomalies. Ground loop problems manifest themselves by hum or interference when both the auxiliary plug and the speaker plug are plugged in at the same time. If there is a noticeable background hum on tape data files, it may be caused by a ground loop in a cassette player. The problem can be averted by plugging in only one of the two signal plugs at a time: speaker plug during tape reads or auxiliary input during tape writes.

THE SERIAL PORT

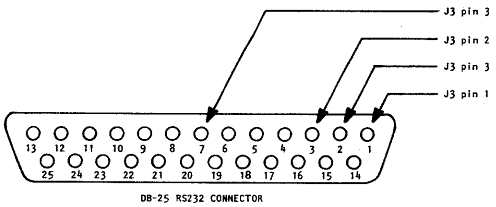
J3 is the serial interface connector. It is a 16-pin DIP socket. Nine of the sixteen pins are used. Three connections are dedicated to the RS232 section of the serial port and six are devoted to the TTY section of the port. The pin-out for this connector is detailed below.



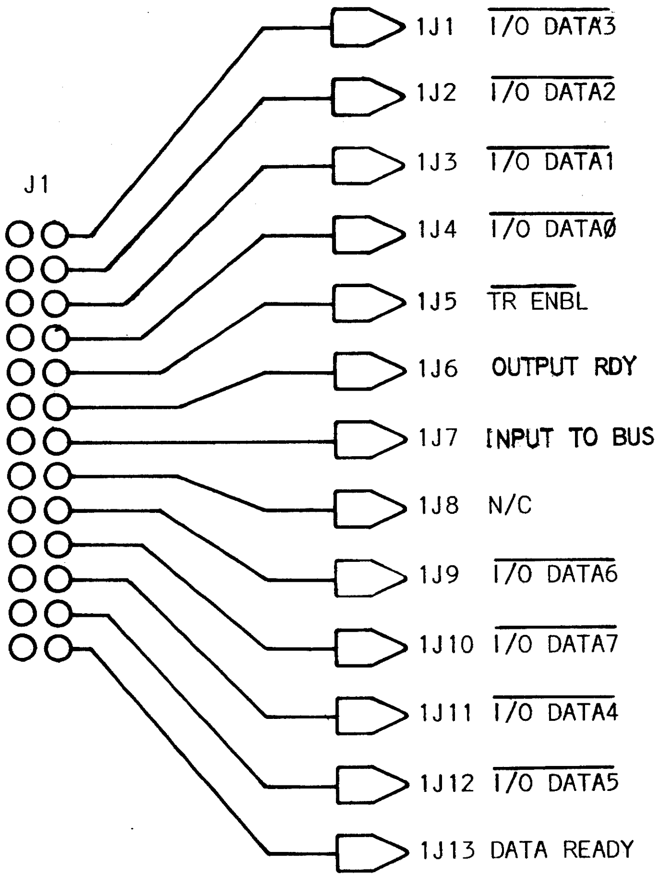
If a TTY is connected to the port, the RS232 inputs should be left open. On the other hand, if an RS232 device is connected to the port, the TTY inputs must be connected. That is, pins 3 and 4 of J3 must be connected together when an RS232 device is connected to pins 1, 2 and 15 of J3. The two layouts below detail how the connectors of a TTY or an RS232 device (DB25) should be wired to J3. A 16-conductor flat cable is available from Morrow's to facilitate connecting J3 to a TTY or RS232 device.



TELETYPE TERMINAL STRIP

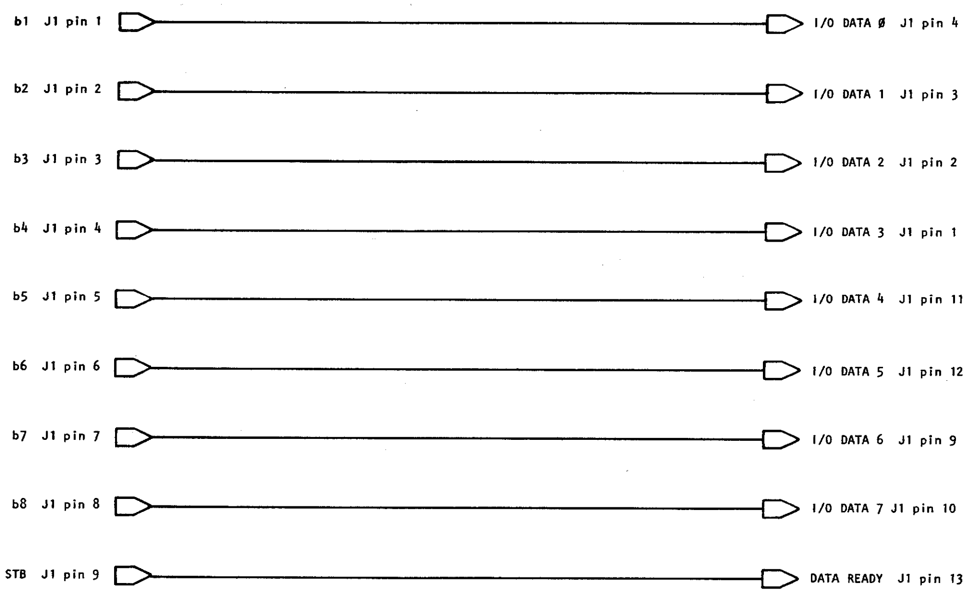


DB-25 RS232 CONNECTOR
REAR VIEW

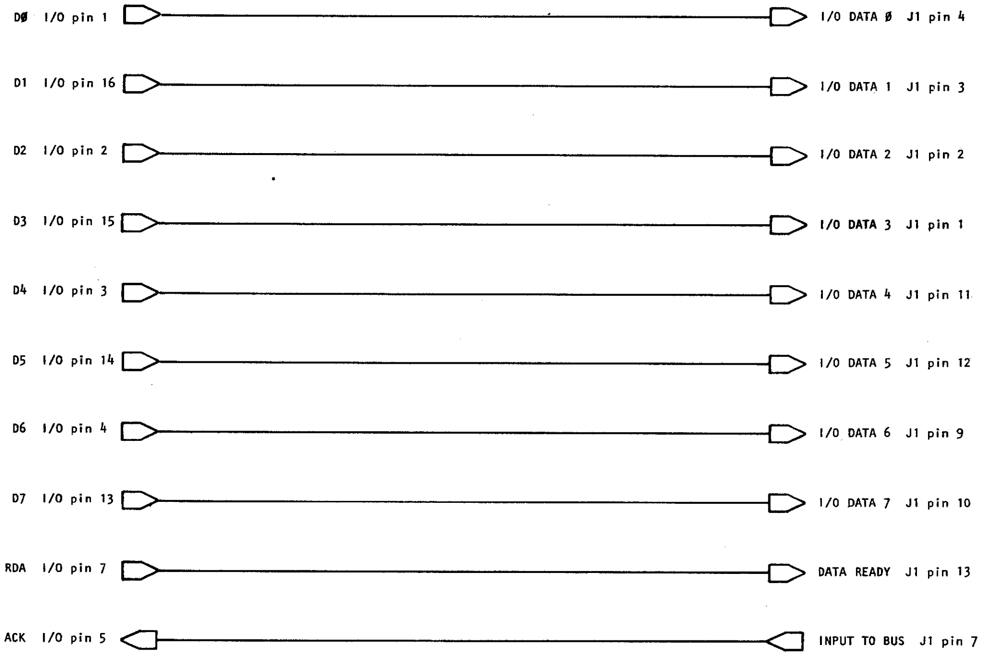


THE PARALLEL PORT

J1 is the parallel port connector. It is a 26-pin array designed to mate with a flat cable terminated with a 26-connection socket (3M part # 3399-0000 socket or equivalent). The pin out for this connector is detailed below.
A 26-conductor cable with a 3M socket is available from Morrow's to facilitate connecting the parallel port to an I/O device.



CHERRY "PRO" KEYBOARD INTERFACE EXAMPLE



OLIVER PAPER TAPE READER INTERFACE EXAMPLE

PARALLEL PORT INTERFACE EXAMPLES

A THINKER TOY PRODUCT FROM MORROW'S

SPEAKEASY I/O BOARD

PARTS LIST

1	8" x 10" glossy photograph	
1	5" x 10" printed circuit board	
3	24Ω ¼ watt resistors	(red-yellow-black)
3	240Ω ½ watt resistors	(red-yellow-brown)
2	470Ω 1 watt resistors	(yellow-purple-brown)
9	1kΩ ¼ watt resistors	(brown-black-red)
1	1.5kΩ "	(brown-green-red)
3	3.3kΩ "	(orange-orange-red)
5	4.7kΩ "	(yellow-purple-red)
2	10kΩ "	(brown-black-orange)
3	15kΩ "	(brown-green-orange)
1	22kΩ "	(red-red-orange)
2	27kΩ "	(red-purple-orange)
4	47kΩ "	(yellow-purple-orange)
1	330kΩ "	(orange-orange-yellow)
1	.001 μfd disk capacitor	
4	.01 "	
1	.1 μfd mylar capacitor	
2	.82 μfd tantulum capacitors	
3	39 μfd "	
17	by-pass capacitors*	
7	1N914/4820-0201 signal diodes	
2	1N4742 12 v. 1 watt zener diodes	
1	1N4585 rectifying diode	
3	Gordos 761A52 relays	

*by-pass capacitor values may vary from .01 μfd to .1 μfd

Parts List

3	2N3906 PNP transistors
1	2N3904 NPN transistor
17	low profile 16-pin sockets
6	low profile 14-pin sockets
1	low profile 20-pin socket
1	26 conductor right angle male header (position J1)
1	16 conductor right angle male header with shell & female connectors (J2)
1	7 or 9 conductor right angle male header with matching shell and female connectors (J3)
1	heat sink
1	set machine nut & screw
1	74LS00 quad 2-input NAND gate
1	74LS04 hex inverter
1	7406 hex open collector inverter/buffer
1	74LS37 quad 2-input NAND buffer
2	74LS74 dual D-type flip-flop
1	74LS139 dual 1 of 4 decoder
1	74LS155 dual 1 of 4 decoder
1	74LS175 quad dual rail register
1	74LS273 octal latch
2	74LS367 2-4 tri-state* buffer with enables
4	2112/5039927A 4x256 N-MOS RAM
2	MMI 6306 4x512 PROM (Monolithic Memories)
1	MMI 6301/82S129 4x256 PROM
2	DS8835 inverting tri-state* transceivers
2	8T97 2-4 tri-state* buffer/driver with enables
1	741 operational amplifier
1	LM311 comparator

*tri-state is a trade mark of National Semiconductor

ASSEMBLY INSTRUCTIONS

DO NOT INSTALL OR SOLDER ANY PARTS UNTIL YOU HAVE READ THESE INSTRUCTIONS SEVERAL TIMES AND HAVE FULLY DIGESTED THE INFORMATION!

CAUTION -- DO NOT SOLDER OR CLIP COMPONENT LEADS WITHOUT USING SAFETY GLASSES!

INSPECTION

Use the Parts List to make sure that there are no missing items in your kit. Please notify us of any shortages. Be sure to check for missing parts before you start to assemble.

COMPONENT LEAD WIDTHS

Bend the leads with the plastic bending block in your kit. Be sure that the leads are bent to the proper width before the part is inserted. Properly bent components solder easier and give the finished kit a professional appearance.

1. All the ¼ watt resistors have lead widths of .5 inches.

2. The ½ watt resistors R38 and R39 have lead widths of .5 inches.

3. The ½ watt resistor at R28 has a lead width of .6 inches.

4. The two 1 watt resistors have lead widths of .95 inches.

5. The two .82 µfd tantulum capacitors C5 and C8 have lead widths of ½ inch.

6. The three 39 µfd tantulum capacitors C9, C10, and C11 have lead widths of .6 inches.

7. The CR11 diode has a lead width of .7 inches; all other diodes have lead widths of ½ inch.

8. The transistors and ceramic disk capacitors have partially preformed leads and may be installed without preparation.

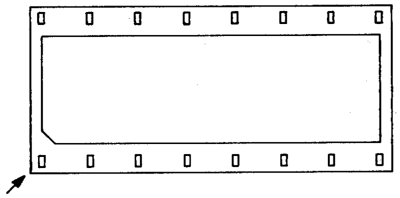
SOCKETS

A socket is furnished for every integrated circuit. It is important that you use the sockets; otherwise, a defective part will be extremely difficult to replace.

NO REPAIR OR SERVICE WILL BE PERFORMED ON A KIT WHICH HAS HAD INTEGRATED CIRCUITS SOLDERED TO THE CIRCUIT BOARD.

PARTS ORIENTATION

In all references throughout the instructions, the convention used is that the gold edge connector is the bottom of the board. Orientation identification is molded into the plastic of the sockets and is illustrated below:



This orientation mark identifies where pin #1 of the integrated circuit is to be positioned when inserted into the socket. The sockets should be inserted in the board so that the orientation mark is in the lower left hand corner.

Orientation of the transistors, tantulum capacitors, diodes and voltage regulator is specified in the component layout drawing. It is advisable to study this drawing and the 8 x 10 glossy photograph carefully before building the kit. Refer to both during parts installation.

SOLDERING AND SOLDER IRONS

The most desirable soldering iron for complex electronic kits is a constant temperature soldering iron with an element regulated at 650° F. The tip should be fine so that it can be brought in intimate contact with the pads of the circuit board. Both Unger and Weller have excellent products which fit the above requirements.

There are three important soldering requirements for building this kit:

1. Do not use an iron that is too cold (less than 600° F) or too hot (more than 750°).

2. Do not apply the iron to a pad for extended periods.

3. Do not apply excessive amounts of solder.

The proper procedure for soldering components to the circuit board is as follows:

1. Bring the iron in contact with both the component lead and the pad.

2. Apply a small amount of solder at the point where the iron, component lead, and pad all make contact.

3. After the initial application of solder has been accomplished with the solder flowing to the pad and component lead, the heat of the iron will have transferred to both the pad and the lead. Apply a small amount of additional solder to cover the joint between the pad and the lead. DO NOT PILE SOLDER ON THE JOINT! EXCESSIVE HEAT AND SOLDER CAUSE PADS AND LEADS TO LIFT FROM THE CIRCUIT BOARD. EXCESSIVE SOLDER IS THE PRIMARY CAUSE FOR BOARD SHORTS AND BRIDGED CONNECTIONS.

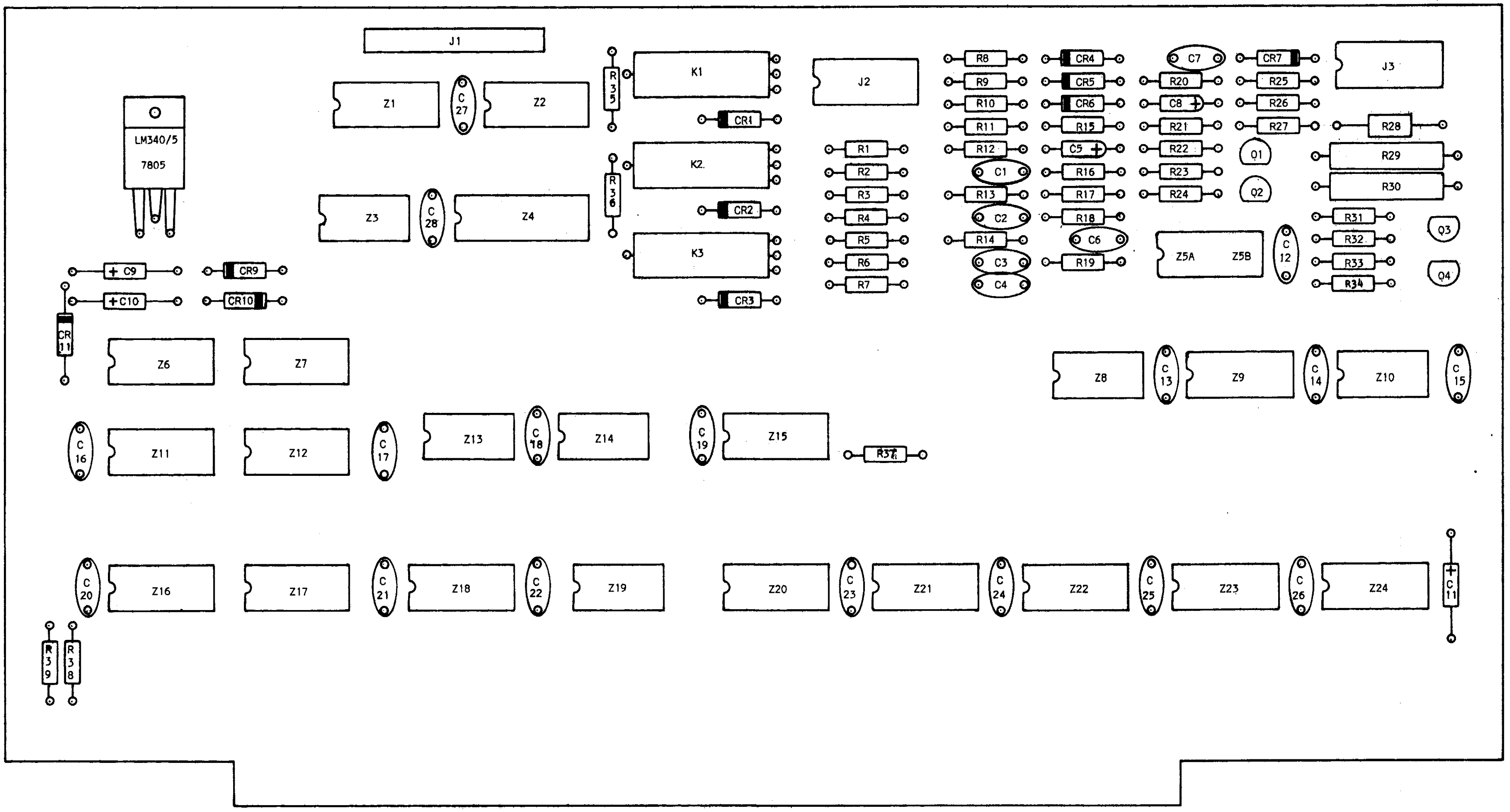
PARTS INSTALLATION

Before installing parts, bend the leads of the resistors, diodes and tantulum capacitors to the proper lengths. After a series of parts have been installed in the board, bend the leads slightly to hold them in place, solder the leads and trim the excess lead lengths before proceeding to the next step.

Install:

____	R38-R39	240Ω $\frac{1}{2}$ watt (bottom left)	(red-yellow-brown)
____	CR9-CR10	1N4742 12 v. 1 watt zener	Check for orientation!
____	R35-R36	1kΩ $\frac{1}{4}$ watt (top)	(brown-black-red)
____	CR1-CR3	1N914/4820-0201	Check for orientation!
____	R1, R3, R5	47kΩ $\frac{1}{4}$ watt	(yellow-purple-orange)
____	R2, R4, R6	24Ω $\frac{1}{4}$ watt	(red-yellow-black)
____	R7, R37	1kΩ $\frac{1}{4}$ watt	(brown-black-red)
____	R8-R11	4.7kΩ $\frac{1}{4}$ watt	(yellow-purple-red)
____	R12-R14	15kΩ $\frac{1}{4}$ watt	(brown-green-orange)
____	CR4-CR6	1N914/4820-0201	Check for orientation!
____	R15-R16	10kΩ $\frac{1}{4}$ watt	(brown-black-orange)
____	C5	.82 µfd tantulum	Check for orientation!
____	R17	22kΩ $\frac{1}{4}$ watt	(red-red-orange)
____	R18	330kΩ $\frac{1}{4}$ watt	(orange-orange-yellow)
____	R19	1kΩ $\frac{1}{4}$ watt	(brown-black-red)
____	R20, R24, R25	3.3kΩ $\frac{1}{4}$ watt	(orange-orange-red)
____	C8	.82 µfd tantulum	Check for orientation!
____	R21	4.7kΩ $\frac{1}{4}$ watt	(yellow-purple-red)
____	R22	47kΩ $\frac{1}{4}$ watt	(yellow-purple-orange)
____	R23	1.5kΩ $\frac{1}{4}$ watt	(brown-green-red)
____	CR7	1N914/4820-0201	Check for orientation!
____	R26, R27	27kΩ $\frac{1}{4}$ watt	(red-purple-orange)

____	R28	240Ω ½ watt	(red-yellow-brown)
____	R31-R34	1kΩ ¼ watt	(brown-black-red)
____	CR11	1N4585	Check for orientation!
____	Sockets Z1, Z2, Z5		Pin #1 to lower left.
____	Socket Z4		Pin #1 to lower left.
____	Sockets Z6, Z7, Z11, Z12, Z16, Z17, Z18		Pin #1 to lower left.
____	Sockets Z15, Z9, Z20, Z21, Z22, Z23, Z24		Pin #1 to lower left.
____	Sockets Z3, Z8, Z19		Pin #1 to lower left.
____	Sockets Z13, Z14, Z19		Pin #1 to lower left.
____	Connector J1		
____	Socket/connector J2		
____	Socket/connector J3		
____	C9, C10, C11	39 μfd tantulum	Check for orientation!
____	R29, R30	470Ω 1 watt	(yellow-purple-brown)
____	C27, C28, C7	by-pass capacitors	
____	C1-C3	.01 μfd blocking capacitors	
____	C4	.1 μfd filter capacitor	
____	C6	.001 μfd blocking capacitor	
____	C12-C15	by-pass capacitors	
____	C16-C19	by-pass capacitors	
____	C20-C26	by-pass capacitors	
____	Install the LM340.5/7805 5 v. regulator by bending leads, inserting and hand tightening the nut and bolt through the regulator, heat sink, and board. Solder the leads. If heat sink grease is available, apply a thin film between the board, heat sink and regulator. Finally, tighten nut firmly.		
____	Q1	2N3904 transistor	Check for orientation!
____	Q2-Q4	2N3906 transistor	Check for orientation!
____	K1-K3	761A05 Form A Gordos relay	Check for orientation!



R1	47	kΩ	
R2	24	Ω	
R3	47	kΩ	
R4	24	Ω	
R5	47	kΩ	
R6	24	Ω	
R7	1	kΩ	
R8	4.7	"	
R9	4.7	"	
R10	4.7	"	
R11	4.7	"	
R12	15	"	
R13	15	"	
R14	15	"	
R15	10	"	
R16	10	"	
R17	22	"	
R18	330	"	
R19	1	"	
R20	3.3	"	
R21	4.7	"	

R22	47	kΩ	
R23	1.5	"	
R24	3.3	"	
R25	3.3	"	
R26	27	"	
R27	27	"	
R28	240	Ω	½W
R29	470	"	1W
R30	470	"	1W
R31	1	kΩ	
R32	1	"	
R33	1	"	
R34	1	"	
R35	1	"	
R36	1	"	
R37	1	"	
R38	240	Ω	½W
R39	"	"	"

C1	.01	µfd
C2	.01	"
C3	.01	"
C4	.1	"
C5	.82	"
C6	.001	"
C7	.01	"
C8	.82	"
C9	39	"
C10	39	"
C11	39	"
C12	by-pass	
C13	by-pass	
C14	by-pass	
C15	by-pass	
C16	by-pass	
C17	by-pass	
C18	by-pass	
C19	by-pass	
C20	by-pass	
C21	by-pass	

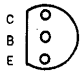
C22	by-pass
C23	by-pass
C24	by-pass
C25	by-pass
C26	by-pass
C27	~~by-pass~~
C28	by-pass
CR1	1N914/4820-0201
CR2	"
CR3	"
CR4	"
CR5	"
CR6	"
CR7	"
CR8	non-existant
CR9	1N4742 12V 1W
CR10	"
CR11	1N4585

K1	Gordos 761A52	
K2	"	
K3	"	
Q1	2N3904	
Q2	2N3906	
Q3	2N3906	
Q4	2N3906	
Z1	DS8835	NATIONAL
Z2	DS8835	
Z3	74LS37	
Z4	74LS273	
Z5A	LM311	
Z5B	741	
Z6	2112/5039927	INTEL
Z7	2112/5039927	
Z8	7406	
Z9	74LS175	
Z10	74LS74	
Z11	2112/5039927	INTEL

Z12	2112/5039927 INTEL
Z13	74LS74
Z14	74LS00
Z15	74LS155
Z16	6306-1 MONOLITHIC MEMORYS
Z17	6306-1
Z18	74LS139
Z19	74LS04
Z20	6301-1 MONOLITHIC MEMORYS
Z21	74LS367
Z22	74LS367
Z23	8T97 SIGNETICS
Z24	8T97

by-pass capicators vary from kit to kit from .01 µfd to .1 µfd

TRANSISTORS



BOTTOM VIEW

THE SPEAKEASY I/O INTERFACE   A THINKER TOY from MORROW'S
COMPONENT LAYOUT

POWER-UP AND SYSTEM CHECK OUT

POWER SUPPLY/VOLTAGE REGULATOR CHECK OUT

Voltage requirements: (reference to ground - pins 50 and 100)

Pins 1 and 51	not less than 7 volts not more than 10 volts	approx. .7 amps
Pin 2	not less than 14 volts not more than 22 volts	approx. .1 amps
Pin 52	not less than -22 volts not more than -14 volts	approx. .1 amps

Before installing any of the integrated circuits, apply power to pins 1 and 51, pin 2, and pin 52 (ground at pins 50 and 100) as specified above. Power can come from external supplies or from the host computer. If the board is to be plugged into the computer, it is advisable to use an extender board during testing of the interface. After applying power, perform the following measurements with a volt meter:

(1)	J3 pin 4	-12 volts
(2)	Z5 pin 11	+12 volts
(3)	Z1 pin 16	+5 volts
(4)	Z16 pin 16	+5 volts
(5)	Z15 pin 16	+5 volts
(6)	Z8 pin 14	+5 volts
(7)	Z24 pin 16	+5 volts

If the voltage at any of the check points differs by more than 5% of the required value, return the board for trouble shooting and repair.

POWER-UP CHECK OUT

Install the integrated circuits as per the layout sheet. Be very careful not to bend pins of the ICs under the package. Often a pin which is bent under an IC appears to be inserted in the socket. REMEMBER: BENT PINS ARE THE MOST COMMON REASON FOR A MALFUNCTIONING BOARD!

After the parts have been installed, repeat the seven voltage measurements specified in the previous section. If the test points differ from the required values, the board should be returned. Under no circumstance should sockets be removed from the board. Such abuse may void the warranty under which repair and service is performed.

SYSTEM CHECK-OUT INTRODUCTION

The I/O interface occupies 1024 bytes of memory: 200:000 through 203:377 octal or 8000 through 83FF hex. It also uses I/O device addresses 4, 5, 6 and 7. These addressing requirements can be easily customized for special applications but in the check-out procedures which follow, the standard addressing will be used. Users of customized boards should apply appropriate address offsets in the test programs below.

The 1024 bytes of memory used by the I/O board are evenly divided between RAM and PROM. The PROM occupies the first 512 bytes and the RAM uses the last 512 bytes. The board uses I/O device 4 for the cassette data input, data output, and tape motion control. Device 5 is dedicated to the serial data port which includes both a TTY and an RS232 interface. Device 7 controls the 8 bit bi-directional parallel port. Device 6 performs two distinct tasks: the output half controls the paper tape reader, if one is present, on the TTY connected to device 5. The input half furnishes data ready status bits for both the serial port and the parallel port.

The check out procedures which follow are designed to exercise each I/O device on the board along with the RAM and PROM. The RAM is also utilized to facilitate the task of checking the I/O devices.

At a number of places in the check out procedure, it is necessary to set sense switches and start short test programs. For IMSAI or ALTAIR computers a program is started by doing an "examine" at the starting address and then pressing the RUN switch. For the EQUINOX computer or a Keyed-Up 8080 Front Panel/CPU, a program is started by initializing the Program Counter of the CPU and pressing the M key. For computers which have no formal front panel such as the SOL or VECTOR GRAPHICS, a test program is started by having the system monitor perform an "EXECUTE" command to the test program's starting address.

There are several short 28 gauge wires included in I/O Bag #1. These wires are to be used to jumper pins J3 in the check out tests that follow. These wires are fine enough and stiff enough so that they may be inserted into the various connectors of the connector/socket J3.

ROM

Insert the I/O board in the main frame and turn on the computer. After power up, perform the following examine memory contents:

Octal		Hex	
200:000	303	8000	C3
200:001	003	8001	03
200:002	200	8002	80
200:003	075	8003	3D

201:012	107	810A	47
201:013	042	810B	22
201:014	336	810C	DE
201:015	203	810D	83

If there are any deviations from the above values, the board should be returned for service. This completes the ROM check out.

RAM

Using the computer's front panel or the system monitor, enter the following data starting at address 202:000 octal (8200 hex):

Octal		Hex		
202:000	333	8200	DB	START IN SSW
202:001	377	8201	FF	
202:002	323	8202	D3	OUT TAPE
202:003	004	8203	04	
202:004	303	8204	C3	JMP START
202:005	000	8205	00	
202:006	202	8206	82	

After the data has been entered, examine the locations to be sure the RAM has accepted it correctly. This program will be used along with the sense switches (front panel numeric switches on the EQUINOX or Keyed-Up 8080 panel) to check output port 4, the tape cassette command and data output port. If the RAM will not accept data correctly, return the board for service.

Again, using the system monitor or the computer's front panel, enter the following data starting at address 203:000 octal (8300 hex):

203:000	333	8300	DB	BEGIN IN SSW
203:001	377	8301	FF	
203:002	323	8302	D3	OUT PARALLEL
203:003	007	8303	07	
203:004	333	8304	DB	IN PARALLEL
203:005	007	8305	07	

```
203:006     062          8306    32              STA TEST

203:007     014          8307    0C

203:010     203          8308    83

203:011     303          8309    C3              JMP BEGIN

203:012     000          830A    00

203:013     203          830B    83

203:014     000          830C    00     TEST
```

As before, examine the data to be sure that the RAM has accepted it correctly. If there are memory cells which do not accept data, return the board.

This second sequence of data is another program which will help in checking out the input and output functions of the parallel port -- port 7. To check out the various I/O ports, several items are necessary:

1. Some jumpers at least three inches long (included in Bag #1).

2. Ohm meter.

3. Voltmeter or logic probe.

TAPE COMMAND AND DATA OUT PORT: I/O DEVICE 4

The first short program entered at 202:000 octal (8200 hex) will be used; it is recapped here for convenience.

```
         Octal                     Hex

202:000     333 377      8200    DB FF       START IN SSW

202:002     323 004      8202    D3 04             OUT 4

202:004     303 000 202  8204    C3 00 82          JMP START
```

Start this program with all the sense switches off. (The sense switches, i.e., the numeric keys of the EQUINOX's front panel are automatically reset whenever the front panel program transfers control to the user's program by pressing the M switch.)

Using the volt meter or logic probe, measure the voltage at the left side of R7 (the 7th resistor below J2). The voltage should be almost zero (TTL logic zero).

Turn on sense switch 0 (press numeric pad switch 0 on the EQUINOX or Keyed-Up 8080). Verify that the voltage is now approximately 4 volts (TTL logic 1). Turn sense switch 0 off again (press S, then M on the EQUINOX).

Verify that the level at the left side of R7 is again a logic zero. Stop the program. This completes the check out of the tape cassette data output.

Restart the program. Connect an ohm meter across pins 6 and 11 of J2. Verify that this is an open circuit. Also, verify that the logic level at pin 12 of Z8 (7406) is a logic 0. Turn on sense switch 1. Verify that pins 6 and 11 of J2 now make a short circuit and that the level at pin 12 of Z8 is now 5 volts. Turn off sense switch 1. Verify that pins 6 and 11 of J2 are again open circuited while the level at pin 12 of Z8 is again zero. This completes the check out of the unit #1 tape command port -- Z8 enables data input from the tape channel 1 and pins 6 and 11 connect to the motor control relay at channel 1.

Connect the ohm meter next across pins 4 and 12 of J2 and verify that this is an open circuit. Also, verify that pin 6 of Z8 is a logic 0. Turn on sense switch 2. Check that pins 4 and 12 of J2 are now short circuited and that pin 6 of Z8 is a logic 1 (5 volts). Turn off sense switch 2 and check that pin 6 of Z8 has returned to a logic zero (approximately 0 volts) and that pins 4 and 12 of J2 are again an open circuit. This completes the check out of the unit #2 command port.

Finally, connect the ohm meter across pins 2 and 15 of J2 and check that this is an open circuit. Also check that pin 10 of Z8 is at a logic 0. Turn on sense switch 3 and check that pins 2 and 15 of J2 are now shorted while pin 10 of Z8 is at 5 volts. Turn sense switch 3 off now and check that at open circuit condition is again maintained across pins 2 and 15 of J2 while the logic level at pin 10 of Z8 has returned to zero. This completes the checkout of the tape output data and command port. Should any of the above tests fail, please return board for service.

TAPE INPUT DATA PORT:

Enter the following program at 202:000 octal (8200 hex):

```
202:000    333 004        8200    DB 04        GET  IN 4
202:002    346 001        8202    E6 01             ANI 1
202:004    062 012 202    8204    32 0A 82          STA DATA
202:007    303 000 202    8207    C3 00 82          JMP GET
202:012    377            820A    FF           DATA 377
```

Remove the LM311 from socket Z5A. Connect a jumper from pin #15 of Z5 to pin 3 of J3. Start the above program. Stop the program and examine location 202:012. Verify that this location contains zero. Next, jumper pin 15 of Z5 to pin 4 of J3. Restart the above program. Stop the program and verify that location 202:012 now contains the value 001. Remove the jumper and replace the LM311 in Z5A.

This completes the check out of the tape data input channel. The test does not exercise the LM311 comparator or its input circuitry but it does verify that the output from the LM311 is being properly accessed by the CPU. Later, we will exercise the input circuitry of the tape data channel.

SERIAL INPUT DATA PORT:

Enter the following program at 202:000 octal (8200 hex):

```
202:000     333 005        8200     DB 05        GETS IN 5

202:002     346 001        8202     E6 01             ANI 1

202:004     062 021 202    8204     32 11 82          STA SDATA

202:007     333 006        8207     DB 06             IN 6

202:011     346 001        8209     E6 01             ANI 1

202:013     062 022 202    820B     32 12 82          STA STATUS

202:016     303 000 202    820E     C3 00 82          JMP GETS

202:021     377            8211     FF           SDATA 377

202:022     376            8212     FE           STATUS 376
```

After entering any data into a computer, especially a test program such as the one above, be sure to examine the contents for accuracy. After the program has been checked, start the test. Stop the program and examine locations 202:021 (8211 hex) and 202:022 (8212 hex). Location 202:021 should now contain 000 and location 202:022 should contain 001.

Using a jumper, connect pin #4 to pin #5 of J3. Restart the test program. Stop the program and examine locations 202:021 and 202:022 for values 001 and 000, respectively. Disconnect the jumper. This completes the serial input data port check out.

SERIAL OUTPUT DATA PORT:

Enter the following program at 202:000 octal (8200 hex):

```
            OCTAL                       HEX
202:000     333 377        8200     DB FF        TEST IN SW

202:002     323 005        8202     D3 05             OUT 5

202:004     303 000 202    8204     C3 00 82          JMP TEST
```

Using a jumper, connect pins 6 and 7 of J3 together. Make sure all the sense switches are off (the sense switches are always turned off when the front panel of the EQUINOX is active). Start the program. With a volt meter, verify that the voltage between pin 6 of J3 and ground is 0 volts and that the voltage between pin #2 of J3 and ground is +12 volts.

Turn on sense switch 0 (press the 0 digit on the EQUINOX). Measure the voltage between pin #6 of J3 and ground. It should be 5 volts. Measure again the voltage between pin #2 of J3 and ground. It should now read -12 volts instead of +12 volts. Stop the program and disconnect the jumpers. This completes the test of the serial output data port.

TTY/PAPER TAPE READER CONTROL PORT:

Enter the following program at 202:000 octal (8200 hex):

```
202:000    333 377        8200    DB FF       TEST IN SSW

202:002    323 006        8202    D3 06            OUT 6

202:004    333 000 202    8204    C3 00 82         JMP TEST
```

Using a jumper, connect pins 8 and 9 of J3. Make sure all the sense switches are off. Start the program. With a volt meter, measure the voltage between pin #8 of J3 and ground. The voltage should be -12 volts. Turn on sense switch 0 and again measure the voltage between pin #8 of J3 and ground. It should now be +5 volts. Stop the program and remove the jumper. This completes the check out of the TTY Paper Tape Reader Control port.

PARALLEL STATUS PORT:

Enter the following program at 202:000 octal (8200 hex):

```
202:000    333 007        8200    DB 07            IN 7

202:002    333 006        8202    DB 06       LOOP IN 6

202:004    346 002        8204    E6 02            ANI 2

202:006    062 014 202    8206    32 0C 82         STA STATUS

202:011    303 002 202    8209    C3 02 82         JMP LOOP

202:014    377            820C    FF          STATUS 377
```

Start and stop the program. Examine location 202:014 octal and verify that it contains 000. Start the program again. Locate R36 on the circuit board (just to the right of Z4, a 20 pin IC #74LS273). Using a jumper, ground the TOP lead of R36. Stop the program and examine location 202:014 octal (820C hex). This location should now contain 002. This completes the check out of the Parallel Status port.

PARALLEL DATA PORT:

Earlier, the following program was entered at location 203:000 octal (8300 hex):

```
    203:000    333 377          8300    DB FF       BEGIN IN SSW
    203:002    323 007          8302    D3 07             OUT 7
    203:004    333 007          8304    DB 07             IN 7
    203:006    062 014 203      8306    32 0C 83          STA TEST
    203:011    303 000 203      8309    C3 00 83          JMP BEGIN
    203:014    000              830C    00          TEST 000
```

Verify that the program is still in memory. Jumper pin #5 of J1 to ground Check as follows:

Step 1	Step 2	Step 3	Step 4	Step 5	Step 6
Turn off all sense switches.	Start the program.	Turn on sense switch:	Stop the program.	Examine location 203:014 (830C hex).	Contents should be:
"	"	0	"	"	001 octal (01 hex)
"	"	1	"	"	002 octal (02 hex)
"	"	2	"	"	004 octal (04 hex)
"	"	3	"	"	010 octal (08 hex)
"	"	4	"	"	020 octal (10 hex)
"	"	5	"	"	040 octal (20 hex)
"	"	6	"	"	100 octal (40 hex)
"	"	7	"	"	200 octal (80 hex)

This completes the check out of the parallel data port and the function modules of the I/O board. If any of the test data differs from what has been described above, return the board for service.

There is one further test which should be done, i.e., exercise the input circuitry of the tape cassette interface. The best way to do this is by performing the examples which illustrate the use of the tape cassette interface (see the tape cassette section of the Principles of Operation).

ROUTINE TO GENERATE FLUX CHANGES ON THE CASSETTE TAPE INTERFACE

Calling conventions:

1. The H-L pair contains the total number of flux changes desired.

2. Register C contains a value which determines the delay time between flux changes given by 32·(C) + 65 machine cycles.

3. Register A contains a value which determines the delay time from entry of the routine to the first flux changes given by 32·(A) + 23 machine cycles.

There is a delay of 55 machine cycles from the last flux change to the next instruction of the calling routine which follows the "CALL FLUX" instruction.

For a flux change frequency of 1200 hz, or a zero bit, register C should contain 24 decimal. For a flux change frequncy of 2400 hz, or a one bit, register C should contain 11 decimal.

All timing assumes one wait state per fetch.

```
200:000  303 003 200   FLUX    JMP   FLUX+3     TIMING PAD
    003  075                   DCR   A          DECREMENT DELAY COUNT
    004  302 000 200           JNZ   FLUX       TEST FOR EXPIRATION
    007  074                   INR   A          COMPLIMENT BIT Ø OF
    010  250                   XRA   B            REG B & LEAVE IN ACC
    011  323 004               OUT   TAPE       CHANGE THE FLUX
    013  107                   MOV   B,A        UPDATE REG B
    014  053                   DCX   H          DECREMENT FLUX CHANGE
    015  175                   MOV   A,L        COUNT & TEST FOR
    016  264                   ORA   H            COUNT EQUAL ZERO
    017  171                   MOV   A,C        LOAD A WITH DELAY COUNT
    020  302 000 200           JNZ   FLUX       STATUS REG IS SET BY ORA
    023  311                   RET              FLUX CHANGES DONE
```

WRITE TAPE CASSETTE SUBROUTINE

Routine to write a block of data on a cassette recorder using the cassette interface.

Calling conventions:

1. The entry point to the routine is WTAPE as opposed to the first instruction which has the label TLOOP.

2. The H-L register pair is the address pointer to the data block.

3. The D-E register pair holds the word count of the data block.

4. Register C contains a value which determines the delay time from entry of the routine to the first flux change given by 32·(C) + 125 machine cycles.

There is a delay of 158 machine cycles from the last flux change on the audio cassette interface and the instruction of the calling routine following the "CALL WTAPE" instruction.

Register Map:

A: General purpose and delay counts.

B: Tape command - except for bit Ø, register B is preserved. All outputs to the cassette interface are issued through this register.

C: Delay constants.

D-E: Data: word count.

H-L: Flux change count; address pointer.

Including the return address of the calling program, eight levels of the stack are used by the routine. All delays are calculated on the assumption that there is one wait state encountered per memory fetch (both data and instruction memory).

```
200:024  303 027 200  TLOOP  JMP   TLOOP+3   TIMING PAD
    027  267                 ORA   A         TIMING PAD
    030  341                 POP   H         CURRENT DATA POINTER
    031  321                 POP   D         CURRENT WORD COUNT
    032  016 022             MVI   C,18      DELAY CONSTANT
    034  172          WTAPE  MOV   A,D       TEST FOR WORD
    035  263                 ORA   E           COUNT EQUAL TO
    036  310                 RZ                ZERO
    037  033                 DCX   D         DECREMENT WORD COUNT
    040  325                 PUSH  D         SAVE NEW WORD COUNT
    041  136                 MOV   E,M       GET CURRENT DATA
    042  043                 INX   H         INCREMENT ADDRESS POINTER
    043  345                 PUSH  H         SAVE NEW ADDRESS POINTER
    044  171                 MOV   A,C       LOAD DELAY CONSTANT
    045  026 003             MVI   D,3       STOP BITS
    047  041 010 000  ZBIT   LXI   H,8       # OF FLUX CHANGES
    052  016 030             MVI   C,24        FOR A ZERO BIT
    054  315 000 200  WRITE  CALL  FLUX      GENERATE A BIT
    057  173                 MOV   A,E       CHECK FOR ALL
    060  262                 ORA   D           BITS WRITTEN
    061  312 024 200         JZ    TLOOP     TEST FOR NEXT WORD
    064  172                 MOV   A,D       SHIFT
    065  037                 RAR               THE
    066  127                 MOV   D,A         CURRENT
    067  173                 MOV   A,E         DATA BIT
    070  037                 RAR               TO THE
    071  137                 MOV   E,A         CARRY BIT
    072  076 000             MVI   A,Ø       TIMING PAD
    074  177                 MOV   A,A       TIMING PAD
    075  076 024             MVI   A,2Ø      SET UP FOR
    077  322 047 200         JNC   ZBIT        A ZERO BIT
    102  041 020 000         LXI   H,16      SET UP FOR
    105  016 013             MVI   C,11        A ONE BIT
    107  177                 MOV   A,A       TIMING PAD
    110  076 006             MVI   A,6       DELAY CONSTANT
    112  303 054 200         JMP   WRITE
```

READ TAPE CASSETTE WITH RUNNING SUM START BIT DETECTION AND STORE/VERIFY/MOVE OPTIONS

Calling conventions:

1. The H-L register pair is loaded with the starting address of the area where the data is to be conditionally stored or verified.

2. The D-E register pair is loaded with the length of the data block to be read.

3. Register C determines the mode the read routine will operate in.

a. If bit 6 of register C is set, the routine will verify data on the tape cassette against the contents of the block of memory determined by the two register pairs H-L and D-E.

b. If bit 7 is set, the routine will simply advance across the tape the number of bytes determined by D-E. This option allows the user to advance exactly one full record on the tape to facilitate computer controlled searches.

c. The routine assumes that motion control and channel selection will be performed by the calling program.

The routine has logic which rejects wave forms which are not in the region of 1200 to 2400 hz. As such, the tape mechanism may be engaged after the routine has been called but it is not advisable to do so. The routine will wait for signals for approximately one minute and if none are detected during this interval, the routine terminates and writes FF in SERR (STATUS ERROR = 203:366). This location is not initialized by the on-board software. If the verify option is chosen and there is a byte in memory which does not agree with its counterpart on the cassette, DERR (DATA ERROR = 203:365) is filled with some non-zero value. Again this cell is not initialized by the on-board software. Including the return address of the calling program, ten levels of the stack are used by the routine. All timing assumes one wait state per fetch.

```
200:115 305          TREAD   PUSH  B         SAVE OPTIONS
    116 325                  PUSH  D         SAVE WORD COUNT
    117 345                  PUSH  H         SAVE ADDRESS POINTER
    120 041 000 360          LXI   H,ØFØØØH  BAD WAVE FORM COUNT
    123 345          SYNC    PUSH  H         SAVE
    124 257                  XRA   A         INITIALIZE
    125 041 370 203          LXI   H,PNTR      8 MOST
    130 021 367 203          LXI   D,TOTAL     RECENT
    133 022                  STAX  D           TRANSITION TIME
    134 167          STORE   MOV   M,A         MEMORY LOCATION
    135 054                  INR   L           AND THEIR
    136 302 134 200          JNZ   STORE       TOTAL
```

COPE

```
200:141  107                 MOV   B,A
    142  333 004             IN    TAPE
    144  117                 MOV   C,A
    145  014         SUM     INR   C
    146  333 004             IN    TAPE      REG B
    150  251                 XRA   C           IS A MEASURE OF THE
    151  037                 RAR               TIME BETWEEN THE
    152  004                 INR   B           LAST TRANSITION AND THE NEXT
    153  312 341 200         JZ    LONG      ERROR IF B OVERFLOWS
    156  322 146 200         JNC   SUM+1
    161  054                 INR   L         ADVANCE TRANSITION
    162  076 370             MVI   A,MASK      TIME BUFFER
    164  265                 ORA   L           POINTER
    165  157                 MOV   L,A         & UPDATE
    166  076 034             MVI   A,HIGH    PROTECT AGAINST
    170  270                 CMP   B           LOW FREQUENCY
    171  332 341 200         JC    LONG        WAVE FORMS
    174  032                 LDAX  D         GET OLD RUNNING SUM
    175  226                 SUB   M         SUBTRACT OLDEST TRANSITION TIME
    176  200                 ADD   B         ADD NEWEST TRANSITION TIME
    177  022                 STAX  D         SAVE
    200  336 164             SBI   CONST     COMPARE FOR
    202  322 216 200         JNC   BITS        A START BIT TOTAL
    205  160                 MOV   M,B       SAVE NEWEST TRANSITION TIME
    206  006 003             MVI   B,3       OFF SET COUNT
    210  303 213 200 PAD1    JMP   PAD1+3    TIMING PAD
    213  303 145 200         JMP   SUM       MEASURE NEXT TRANSITION TIME

200:216  036 010     BITS    MVI   E,8       NUMBER OF BITS TO COLLECT
    220  056 013             MVI   L,11      OFFSET TO DETERMINE Ø/1 WAVE
    222  026 017     RENTR   MVI   D,15      BIT CELL TIME ON TAPE
    224  046 014             MVI   H,12      TRANSITION COUNTER
    226  014         COUNT   INR   C         MEASURE
    227  333 004             IN    TAPE        THE
    231  251                 XRA   C           NEXT
    232  037                 RAR               TRANSITION
    233  055                 DCR   L           TIME
    234  322 227 200         JNC   COUNT+1     ON THE TAPE
    237  056 015             MVI   L,13      OFFSET FOR TRANSITION COUNTER
    241  172                 MOV   A,D       GET CURRENT CELL TIME VALUE
    242  372 252 200         JM    TWO       SUBTRACT 1 IF WAVE FORM
    245  326 001             SUI   1           WAS PART OF A ZERO BIT
    247  303 257 200         JMP   NEXT        OTHERWISE SUBTRACT 2
    252  326 002     TWO     SUI   2
    254  303 257 200         JMP   NEXT
    257  127         NEXT    MOV   D,A       UPDATE CELL TIME VALUE
    260  045                 DCR   H         DECREMENT TRANSITION COUNTER
    261  322 226 200         JNC   COUNT     GO BACK IF CELL TIME NON-ZERO
    264  174                 MOV   A,H       IF H POSITIVE A ZERO BIT
    265  027                 RAL               OTHERWISE A ONE BIT
```

COPE

```
200:266  170                 MOV   A,B       SHIFT LATEST
    267  037                 RAR               BIT INTO
    270  107                 MOV   B,A         DATA REGISTER
    271  055                 DCR   L         ADJUST OFFSET
    272  055                 DCR   L           OF TRANSITION TIME COUNT
    273  177                 MOV   A,A       PADDING
    274  035                 DCR   E         DECREMENT BIT COUNT
    275  302 222 200         JNZ   RENTR
    300  361                 POP   PSW       DISCARD BAD WAVE FORM COUNT
    301  341                 POP   H         GET DATA POINTER
    302  321                 POP   D         GET WORD COUNT
    303  361                 POP   PSW       GET OPTIONS
    304  365                 PUSH  PSW       SAVE OPTIONS
    305  312 326 200         JZ    VERFY     TEST FOR VERIFY OPTION
    310  332 314 200         JC    INCR      TEST FOR NO LOAD OPTION
    313  160                 MOV   M,B       LOAD MEMORY WITH DATA
    314  043         INCR    INX   H         ADVANCE DATA POINTER
    315  033                 DCX   D         DECREMENT WORD COUNT
    316  172                 MOV   A,D       TEST FOR
    317  263                 ORA   E           WORD COUNT
    320  302 116 200         JNZ   TREAD+1     EQUAL ZERO
    323  361                 POP   PSW       DISCARD OPTIONS REGISTER
    324  257                 XRA   A         A=Ø MEANS SUCCESSFUL READ
    325  311                 RET

    326  176         VERFY   MOV   A,M       COMPARE PRESENT
    327  250                 XRA   B           DATA WITH MEMORY
    330  312 314 200         JZ    INCR      CONDITIONALLY SET DERR
    333  062 365 203         STA   DERR        WITH NON-ZERO VALUE
    336  303 314 200         JMP   INCR      RETURN

    341  341         LONG    POP   H         GET BAD WAVE FORM
    342  043                 INX   H           COUNT & INCREMENT
    343  174                 MOV   A,H       TEST
    344  265                 ORA   L           COUNT FOR
    345  302 123 200         JNZ   SYNC        OVERFLOW
    350  075                 DCR   A         OVER FLOW OCCURRED
    351  062 366 203         STA   SERR      SET STATUS ERROR
    354  341                 POP   H         RESTORE
    355  321                 POP   D           STACK
    356  301                 POP   B           & RETURN
    357  311                 RET               TO CALLING PROGRAM
```

CASSETTE OPERATING EXECUTIVE (COPE)

Calling conventions:

1. Register A is the command register.

   Bit Ø is the write/read command with a one signifying a write.

   Bit 1 activates channel 1.

   Bit 2 activates channel 2.

   Bit 3 activates channel 3.

   Bit 7 = 1 signifies that the activated channels shall be stopped upon completion of the present command.

   Bit 7 = 0 signifies that the selected channels shall continue to facilitate chained sequences of tape commands.

2. Register C is the options register for read commands, specifically:

   Bit 6 = 1 will prevent loading of the read data and will perform a compare operation with the tape data and the data pointed to by the H-L and D-E register pairs.

   Bit 0 = 1 will prevent loading of the read data and simply move the tape forward the number of bytes specified by the D-E register pair. This option is useful for passing over exactly one record so as to be properly positioned to read the next record.

3. Register pair H-L is the data pointer which should be set to the starting address of the block of data that is to be read, written, or verified.

4. Register pair D-E is the word or byte count register and should be initialized to the length of the data block.

The routine makes extensive use of the stack and as a result, rather than rely on the user to provide the necessary stack space, the calling program's stack pointer is saved. The local RAM of the interface is used as a temporary stack during the duration of the routine. After completing its operations, COPE restores the calling program's stack pointer.

```
201:012  107           COPE    MOV   B,A        INITIALIZE COMMAND REG FOR TAPE
    013  042 336 203           SHLD  LSAVE      SAVE ADDRESS POINTER
    016  041 000 000           LXI   H,Ø        GET
    021  071                   DAD   SP           STACK POINTER
    022  061 336 203           LXI   SP,LSAVE   SAVE STACK POINTER ON
    025  345                   PUSH  H            COPE STACK
    026  323 004               OUT   TAPE       START CASSETTE
    030  365                   PUSH  PSW        SAVE COMMAND
    031  041 000 000           LXI   H,Ø        DO A
    034  053           WCOPE   DCX   H            DELAY
    035  174                   MOV   A,H          OF APPROXIMATELY
    036  265                   ORA   L            ONE
    037  302 034 201           JNZ   WCOPE        SECOND
    042  361                   POP   PSW        GET COMMAND
    043  365                   PUSH  PSW          AND SAVE AGAIN
    044  037                   RAR              TEST FOR A READ
    045  322 104 201           JNC   READ         OPERATION
    050  041 300 135           LXI   H,24ØØØ    GENERATE
    053  016 013               MVI   C,11         5 SECONDS
    055  171                   MOV   A,C          OF 2400 HZ WAVE
    056  315 000 200           CALL  FLUX         REST BITS
    061  016 006               MVI   C,6        DELAY CONSTANT
    063  052 336 203           LHLD  LSAVE      GET THE ADDR POINTER
    066  315 034 200           CALL  WTAPE      WRITE RECORD
    071  361           EXIT    POP   PSW        GET COMMAND
    072  267                   ORA   A          TEST TO TURN
    073  362 101 201           JP    RETRN        OFF CASSETTE
    076  257                   XRA   A
    077  323 004               OUT   TAPE       STOP CASSETTE
    101  341           RETRN   POP   H          GET OLD STACK POINTER
    102  371                   SPHL             RESTORE
    103  311                   RET              RETURN

    104  052 336 203   READ    LHLD  LSAVE      GET ADDRESS POINTER
    107  315 115 200           CALL  TREAD
    112  303 071 201           JMP   EXIT
```

COMPUTE CHECK-SUM ROUTINE

Calling conventions:

1. The register pair H-L is loaded with the starting address of the data block on which the check-sum is to be computed.

2. The register pair D-E is loaded with the word count of the data block.

3. The computed check is returned in the register pair H-L.

Including the return address of the calling program, the routine uses four levels of the stack.

```
201:115  345           CHECK  PUSH   H               SAVE ADDRESS POINTER
    116  041 000 000          LXI    H,Ø             INITIALIZE CHECK SUM
    121  104                  MOV    B,H
    122  343           GDATA  XTHL                   SAVE & EXCHG/ADDR POINTER
    123  116                  MOV    C,M             GET DATA
    124  043                  INX    H               INCREMENT ADDRESS POINTER
    125  343                  XTHL                   SAVE & GET PARTIAL CHECK SUM
    126  011                  DAD    B               ADD NEW DATA
    127  033                  DCX    D               DECREMENT WORD COUNT
    130  172                  MOV    A,D             TEST FOR
    131  263                  ORA    E                 WORD COUNT
    132  302 122 201          JNZ    GDATA             EQUAL ZERO
    135  321                  POP    D               RESTORE STACK
    136  311                  RET

    137  061 336 203   BOOTS  LXI    SP,LSAVE        READ
    142  041 000 202          LXI    H,MSA+200:000     THE
    145  021 000 001          LXI    D,256             FIRST
    150  076 202              MVI    A,202Q            RECORD
    152  016 000              MVI    C,Ø               AND
    154  303 012 201          JMP    COPE              BRANCH THERE
```

THE SERIAL DATA INPUT ROUTINE WITH READER CONTROL AND ECHO OPTIONS

Calling conventions:

1. The on-board RAM location SCON (SERIAL SPEED CONSTANT) must be initialized to the proper value, preferably using the DETCT routine below. This is so that the cell time assumed by the routine and the cell time of the device connected to the interface are compatible.

2. Register B is the options register.

a. If bit Ø is a one, echoes to the printer are suppressed; otherwise, the data will be echoed as collected.

b. If bit 7 is a one, the paper tape reader, if one is connected to the interface, will be turned on until a start bit is encountered and then turned off.

c. The routine returns with the collected data byte in register D.

Including the return address of the calling program, six levels of the stack are used by the routine.

Register Map:

A: General purpose accumulator.

B: The data collection options register described above.

C: Not used.

D-E: 16 bit storage shift register for data collection.

H-L: Delay time counts.

All timing assumes one wait state per fetch.

```
201:157  170           INPUT  MOV   A,B      CONDITIONALLY
    160  007                  RLC              TURN ON
    161  323 006              OUT   READR      THE PAPER TAPE READER
    163  052 363 203          LHLD  SCON     GET THE SPEED CONSTANT
    166  345           SWAIT  PUSH  H          & SAVE ON THE STACK
    167  036 377              MVI   E,-1     INITIALIZE ½ THE SHIFT REG
    171  333 005       SLOOK  IN    SERAL    GET INPUT DATA
    173  037                  RAR              & TEST
    174  332 171 201          JC    SLOOK      FOR A START BIT
```

```
201:177  315 240 201          CALL DELAY    WAIT HALF A
    202  341                  POP   H         BIT TIME
    203  333 005              IN    SERAL     & VERIFY
    205  037                  RAR             THAT A START
    206  332 166 201          JC    SWAIT     BIT IS PRESENT
    211  026 377              MVI   D,-1    INITIALIZE OTHER ½ OF SHIFT REG
    213  257                  XRA   A       STOP THE
    214  323 006              OUT   READR     READER
    216  345          GTBIT   PUSH  H       UPDATE STACK
    217  051                  DAD   H       CALCULATE SPEED
    220  053                  DCX   H         CONSTANT FOR A FULL BIT TIME
    221  333 005              IN    SERAL   GET INPUT
    223  137                  MOV   E,A     UPDATE SHIFT REG
    224  315 240 201          CALL  DELAY   DELAY 1 BIT TIME & SHIFT D-E
    227  341                  POP   H       GET SPEED CONSTANT
    230  332 216 201          JC    GTBIT   WAS START BIT SHIFTED TO CARRY?
    233  076 001              MVI   A,1     YES.  LEAVE WITH
    235  323 005              OUT   SERAL     PRINTER IN SPACE MADE
    237  311                  RET
```

SERIAL DELAY ROUTINE USED IN CONJUNCTION WITH THE SERIAL INPUT AND OUTPUT ROUTINES

Calling conventions:

1. H-L is initialized with a value which determines the time before the routine returns to the calling program given by 53(H-L) + 58 machine cycles.

2. Bits Ø of registers B and E are initialized to values consistent with these bits being ored together and sent to the serial output device.

All timing assumes one wait state per fetch.

```
201:240  173          DELAY   MOV   A,E     CONDITIONALLY
    241  260                  ORA   B         OUTPUT BIT Ø OF E
    242  323 005              OUT   SERAL     TO THE SERIAL DEVICE
    244  053                  DCX   H       DECREMENT
    245  174                  MOV   A,H       H-L PAIR
    246  265                  ORA   L         AND TEST
    247  302 240 201          JNZ   DELAY     FOR ZERO
    252  172                  MOV   A,D     ROTATE D-E
    253  037                  RAR             ONE
    254  173                  MOV   A,E       BIT
    255  037                  RAR             POSITION
    256  137                  MOV   E,A       TO THE
    257  172                  MOV   A,D       RIGHT
    260  037                  RAR             WITH END AROUND
    261  127                  MOV   D,A       BIT PRESERVED
    262  311                  RET
```

SERIAL OUTPUT ROUTINE

Calling conventions:

1. Register A is initialized to an 8-bit value to be serially sent to the serial output device.

2. The on-board RAM location SCON (SERIAL SPEED CONSTANT) must be initialized to the proper value preferably using the DETCT routine below. This is so that the cell time assumed by the routine and the cell time for the device connected to the interface are compatible.

The routine starts by sending a zero for one cell time to the serial device connected to the interface. It next sends bit Ø of register A, followed by bits 1 through 7. It then sends a one for two cell times as rest bits and returns to the calling program. All timing assumes one wait state per fetch. Including the return address of the calling program, four levels of the stack are used by the routine.

Register Map:

A: Serial output data.

B: The routine loads B with twice A to force output when the delay routine is called.

C: Used as a bit counter and initialized to 11 decimal.

D-E: 16-bit storage shift register for output to the serial output device.

H-L: Delay time counts.

```
201:263  207          SROUT   ADD   A       ADD A START BIT
    264  107                  MOV   B,A     MAKE BIT Ø OF B = Ø
    265  137                  MOV   E,A     SHIFTED DATA TO E
    266  076 013              MVI   A,11    THIS IS BIT COUNT & REST BITS
    270  117                  MOV   C,A     COUNT TO REG C
    271  027                  RAL           LOAD D WITH REST BITS
    272  127                  MOV   D,A       & HIGH ORDER DATA BIT
    273  052 363 203  OLOOP   LHLD  SCON    GET SPEED CONSTANT
    276  052 363 203          LHLD  SCON    PADDING
    301  051                  DAD   H       ADJUST FOR OUTPUT
    302  053                  DCX   H         LOOP
    303  315 240 201          CALL  DELAY   OUTPUT DATA BIT & SHIFT
    306  015                  DCR   C       DECREMENT BIT COUNT
    307  302 273 201          JNZ   OLOOP
    312  311                  RET
```

SERIAL DEVICE SPEED DETECTION ROUTINE

This routine should be used when the host system is first turned on. The serial input device that will be communicating with the interface should be connected to the interface but should not be turned on until just before the routine is started. The starting address of the routine 201:313 (81CB hex) should be set on the switches followed by an examine. The serial device should then be turned on and the routine started directly afterward. Depending upon certain initial conditions in the host system, the interface itself, and the serial device, the printing mechanism of the serial output device may momentarily not be initialized correctly. However, the first task of the DETCT routine after it initializes the stack pointer is to make sure that the output to the serial printer is correct in accordance with the input from the serial keyboard device.

After these steps are accomplished, the user should then strike the return or carriage return key on the keyboard. The DETCT routine is expecting to see this ASCII character and when it does, it is possible for the routine to measure the baud rate of the device and initialize the on-board RAM location SCON (SERIAL SPEED CONSTANT). After accomplishing this task, the routine comes to a dynamic halt (JMP*) and the user can then halt the host system and start the COPE BOOTSTRAP routine.

All timing assumes one wait state per fetch.

```
201:313  061 340 203         LXI   6,MSA+203:340  INITIALIZE STACK POINTER
    316  026 006      DETCT  MVI   D,6            TRANSITION COUNTER
    320  112                 MOV   C,D            BIT Ø OF C IS
    321  014          TTOOP  INR   C                IMPORTANT BIT HERE
    322  041 001 000         LXI   H,1            OFFSET FOR H-L
    325  333 005      GETIN  IN    SERAL          GET INPUT
    327  323 005             OUT   SERAL          ECHO
    331  251                 XRA   C              COMPARE WITH BIT Ø OF C
    332  037                 RAR
    333  043                 INX   H
    334  322 325 201         JNC   GETIN          WAIT IF NO TRANSITION
    337  345                 PUSH  H              SAVE DELAY TIME
    340  025                 DCR   D
    341  302 321 201         JNZ   TTOOP
    344  341                 POP   H              GET 3 ZERO BITS TIME
    345  341                 POP   H              GET 2 ONE BITS TIME
    346  301                 POP   B              GET 1 ZERO BIT TIME
    347  011                 DAD   B              ADD TOGETHER
    350  301                 POP   B              GET 1 ONE BIT TIME
    351  011                 DAD   B              ADD TO PREVIOUS SUBTOTAL
    352  301                 POP   B              DISCARD START BIT
    353  301                 POP   B                & INFINITE STRING
    354  026 003             MVI   D,3              OF REST BITS
```

COPE

SERIAL DEVICE SPEED DETECTION ROUTINE

This routine should be used when the host system is first turned on. The serial input device that will be communicating with the interface should be connected to the interface but should not be turned on until just before the routine is started. The starting address of the routine 201:313 (81CB hex) should be set on the switches followed by an examine. The serial device should then be turned on and the routine started directly afterward. Depending upon certain initial conditions in the host system, the interface itself, and the serial device, the printing mechanism of the serial output device may momentarily not be initialized correctly. However, the first task of the DETCT routine after it initializes the stack pointer is to make sure that the output to the serial printer is correct in accordance with the input from the serial keyboard device.

After these steps are accomplished, the user should then strike the return or carriage return key on the keyboard. The DETCT routine is expecting to see this ASCII character and when it does, it is possible for the routine to measure the baud rate of the device and initialize the on-board RAM location SCON (SERIAL SPEED CONSTANT). After accomplishing this task, the routine comes to a dynamic (JMP*) halt and the user can then halt the host system and start the COPE BOOTSTRAP routine.

All timing assumes one wait state per fetch.

```
201:313  061 340 203         LXI   6,MSA+203:340INITIALIZE STACK POINTER
    316  026 006      DETCT  MVI   D,6          TRANSITION COUNTER
    320  112                 MOV   C,D          BIT Ø OF C IS
    321  014          TTOOP  INR   C              IMPORTANT BIT HERE
    322  041 001 000         LXI   H,1          OFFSET FOR H-L
    325  333 005      GETIN  IN    SERAL        GET INPUT
    327  323 005             OUT   SERAL        ECHO
    331  251                 XRA   C            COMPARE WITH BIT Ø OF C
    332  037                 RAR
    333  043                 INX   H
    334  322 325 201         JNC   GETIN        WAIT IF NO TRANSITION
    337  345                 PUSH  H            SAVE DELAY TIME
    340  025                 DCR   D
    341  302 321 201         JNZ   TTOOP
    344  341                 POP   H            GET 3 ZERO BITS TIME
    345  341                 POP   H            GET 2 ONE BITS TIME
    346  301                 POP   B            GET 1 ZERO BIT TIME
    347  011                 DAD   B            ADD TOGETHER
    350  301                 POP   B            GET 1 ONE BIT TIME
    351  011                 DAD   B            ADD TO PREVIOUS SUBTOTAL
    352  301                 POP   B            DISCARD START BIT
    353  301                 POP   B              & INFINITE STRING
    354  026 003             MVI   D,3            OF REST BITS
```

```
201:356  257          SLOOP   XRA   A        DIVIDE
    357  174                  MOV   A,H        THE
    360  037                  RAR              TOTAL
    361  147                  MOV   H,A        BY
    362  175                  MOV   A,L        EIGHT
    363  037                  RAR              TO
    364  157                  MOV   L,A        CALCULATE
    365  025                  DCR   D          HALF THE
    366  302 356 201          JNZ   SLOOP      DATA BIT TIME
    371  053                  DCX   H          ADJUST FOR OTHER
    372  042 363 203          SHLD  SCON       ROUTINES & SAVE
    375  303 375 201  STOP    JMP   STOP     DYNAMIC HALT
    000                       END
```

Hex Listing

ROUTINE TO GENERATE FLUX CHANGES ON THE CASSETTE TAPE INTERFACE

(see octal listing for comments)

```
8000  C3 03 80    FLUX   JMP    FLUX+3
8003  3D                 DCR    A
8004  C2 00 80           JNZ    FLUX
8007  3C                 INR    A
8008  A8                 XRA    B
8009  D3 04              OUT    TAPE
800B  47                 MOV    B,A
800C  2B                 DCX    H
800D  7D                 MOV    A,L
800E  B4                 ORA    H
800F  79                 MOV    A,C
8010  C2 00 80           JNZ    FLUX
8013  C9                 RET
```

WRITE TAPE CASSETTE SUBROUTINE

```
8014  C3 17 80    TLOOP  JMP    TLOOP+3
8017  B7                 ORA    A
8018  E1                 POP    H
8019  D1                 POP    D
801A  0E 12              MVI    C,18
801C  7A          WTAPE  MOV    A,D
801D  B3                 ORA    E
801E  C8                 RZ
801F  1B                 DCX    D
8020  D5                 PUSH   D
8021  5E                 MOV    E,M
8022  23                 INX    H
8023  E5                 PUSH   H
8024  79                 MOV    A,C
8025  16 03              MVI    D,3
8027  21 08 00    ZBIT   LXI    H,8
802A  0E 18              MVI    C,24
802C  CD 00 80    WRITE  CALL   FLUX
802F  7B                 MOV    A,E
8030  B2                 ORA    D
8031  CA 14 80           JZ     TLOOP
8034  7A                 MOV    A,D
8035  1F                 RAR
8036  57                 MOV    D,A
8037  7B                 MOV    A,E
8038  1F                 RAR
8039  5F                 MOV    E,A
```

```
803A  3E 00              MVI    A,Ø
803C  7F                 MOV    A,A
803D  3E 14              MVI    A,20
803F  D2 27 80           JNC    ZBIT
8042  21 10 00           LXI    H,16
8045  0E 0B              MVI    C,11
8047  7F                 MOV    A,A
8048  3E 06              MVI    A,6
804A  C3 2C 80           JMP    WRITE
```

READ TAPE CASSETTE WITH RUNNING SUM START BIT DETECTION AND STORE/VERIFY/MOVE OPTIONS

```
804D  C5          TREAD  PUSH   B
804E  D5                 PUSH   D
804F  E5                 PUSH   H
8050  21 00 F0           LXI    H,0F000H
8053  E5          SYNC   PUSH   H
8054  AF                 XRA    A
8055  21 F8 83           LXI    H,PNTR
8058  11 F7 83           LXI    D,TOTAL
805B  12                 STAX   D
805C  77          STORE  MOV    M,A
805D  2C                 INR    L
805E  C2 5C 80           JNZ    STORE
8061  47                 MOV    B,A
8062  DB 04              IN     TAPE
8064  4F                 MOV    C,A
8065  0C          SUM    INR    C
8066  DB 04              IN     TAPE
8068  A9                 XRA    C
8069  1F                 RAR
806A  04                 INR    B
806B  CA E1 80           JZ     LONG
806E  D2 66 80           JNC    SUM+1
8071  2C                 INR    L
8072  3E F8              MVI    A,MASK
8074  B5                 ORA    L
8075  6F                 MOV    L,A
8076  3E 1C              MVI    A,HIGH
8078  B8                 CMP    B
8079  DA E1 80           JC     LONG
807C  1A                 LDAX   D
807D  96                 SUB    M
807E  80                 ADD    B
807F  12                 STAX   D
8080  DE 74              SBI    CONST
8082  D2 8E 80           JNC    BITS
```

```
8085  70                 MOV    M,B
8086  06 03              MVI    B,3
8088  C3 8B 80    PAD1   JMP    PAD1+3
808B  C3 65 80           JMP    SUM
808E  1E 08       BITS   MVI    E,8
8090  2E 0B              MVI    L,11
8092  16 0F       RENTR  MVI    D,15
8094  26 0C              MVI    H,12
8096  0C          COUNT  INR    C
8097  DB 04              IN     TAPE
8099  A9                 XRA    C
809A  1F                 RAR
809B  2D                 DCR    L
809C  D2 97 80           JNC    COUNT+1
809F  2E 0D              MVI    L,13
80A1  7A                 MOV    A,D
80A2  FA AA 80           JM     TWO
80A5  D6 01              SUI    1
80A7  C3 AF 80           JMP    NEXT
80AA  D6 02       TWO    SUI    2
80AC  C3 AF 80           JMP    NEXT
80AF  57          NEXT   MOV    D,A
80B0  25                 DCR    H
80B1  D2 96 80           JNC    COUNT
80B4  7C                 MOV    A,H
80B5  17                 RAL
80B6  78                 MOV    A,B
80B7  1F                 RAR
80B8  47                 MOV    B,A
80B9  2D                 DCR    L
80BA  2D                 DCR    L
80BB  7F                 MOV    A,A
80BC  1D                 DCR    E
80BD  C2 92 80           JNZ    RENTR
80C0  F1                 POP    PSW
80C1  E1                 POP    H
80C2  D1                 POP    D
80C3  F1                 POP    PSW
80C4  F5                 PUSH   PSW
80C5  CA D6 80           JZ     VERFY
80C8  DA CC 80           JC     INCR
80CB  70                 MOV    M,B
80CC  23          INCR   INX    H
80CD  1B                 DCX    D
80CE  7A                 MOV    A,D
80CF  B3                 ORA    E
80D0  C2 4E 80           JNZ    TREAD+1
80D3  F1                 POP    PSW
80D4  AF                 XRA    A
80D5  C9                 RET
```

```
80D6  7E          VERFY  MOV    A,M
80D7  A8                 XRA    B
80D8  CA CC 80           JZ     INCR
80DB  32 F5 83           STA    DERR
80DE  C3 CC 80           JMP    INCR

80E1  E1          LONG   POP    H
80E2  23                 INX    H
80E3  7C                 MOV    A,H
80E4  B5                 ORA    L
80E5  C2 53 80           JNZ    SYNC
80E8  3D                 DCR    A
80E9  32 F6 83           STA    SERR
80EC  E1                 POP    H
80ED  D1                 POP    D
80EE  C1                 POP    B
80EF  C9                 RET
```

CASSETTE OPERATING EXECUTIVE (COPE)

```
810A  47          COPE   MOV    B,A
810B  22 DE 83           SHLD   LSAVE
820E  21 00 00           LXI    H,0
8111  39                 DAD    SP
8112  31 DE 83           LXI    SP,LSAVE
8115  E5                 PUSH   H
8116  D3 04              OUT    TAPE
8118  F5                 PUSH   PSW
8119  21 00 00           LXI    H,0
811C  2B          WCOPE  DCX    H
811D  7C                 MOV    A,H
811E  B5                 ORA    L
811F  C2 1C 81           JNZ    WCOPE
8122  F1                 POP    PSW
8123  F5                 PUSH   PSW
8124  1F                 RAR
8125  D2 44 81           JNC    READ
8128  21 C0 5D           LXI    H,24000
812B  0E 0B              MVI    C,11
812D  79                 MOV    A,C
812E  CD 00 80           CALL   FLUX
8131  0E 06              MVI    C,6
8133  2A DE 83           LHLD   LSAVE
8136  CD 1C 80           CALL   WTAPE
8139  F1          EXIT   POP    PSW
813A  B7                 ORA    A
813B  F2 41 81           JP     RETRN
813E  AF                 XRA    A
813F  D3 04              OUT    TAPE
8141  E1          RETRN  POP    H
8142  F9                 SPHL
8143  C9                 RET
```

```
8144  2A DE 83    READ   LHLD   LSAVE
8147  CD 4D 80           CALL   TREAD
814A  C3 39 81           JMP    EXIT
```

COMPUTE CHECK-SUM ROUTINE

```
814D  E5          CHECK  PUSH   H
814E  21 00 00           LXI    H,0
8151  44                 MOV    B,H
8152  E3          GDATA  XTHL
8153  4E                 MOV    C,M
8154  23                 INX    H
8155  E3                 XTHL
8156  09                 DAD    B
8157  1B                 DCX    D
8158  7A                 MOV    A,D
8159  B3                 ORA    E
815A  C2 52 81           JNZ    GDATA
815D  D1                 POP    D
815E  C9                 RET

815F  31 DE 83    BOOTS  LXI    SP,LSAVE
8162  21 00 82           LXI    H,8200H
8165  11 00 01           LXI    D,256
8168  3E 82              MVI    A,82
816A  0E 00              MVI    C,0
816C  C3 0A 81           JMP    COPE
```

THE SERIAL DATA INPUT ROUTINE WITH READER CONTROL AND ECHO OPTIONS

```
816F  78          INPUT  MOV    A,B
8170  07                 RLC
8171  D3 06              OUT    READR
8173  2A F3 83           LHLD   SCON
8176  E5          SWAIT  PUSH   H
8177  1E FF              MVI    E,-1
8179  DB 05       SLOOK  IN     SERAL
817B  1F                 RAR
817C  DA 79 81           JC     SLOOK
817F  CD A0 81           CALL   DELAY
8182  E1                 POP    H
8183  DB 05              IN     SERAL
8185  1F                 RAR
8186  DA 76 81           JC     SWAIT
8189  16 FF              MVI    D,-1
818B  AF                 XRA    A
818C  D3 06              OUT READR
```

```
818E  E5          GTBIT  PUSH   H
818F  29                 DAD    H
8190  2B                 DCX    H
8191  DB 05              IN     SERAL
8193  5F                 MOV    E,A
8194  CD A0 81           CALL   DELAY
8197  E1                 POP    H
8198  DA 8E 81           JC     GTBIT
819B  3E 01              MVI    A,1
819D  D3 05              OUT    SERAL
819F  C9                 RET
```

SERIAL DELAY ROUTINE USED IN CONJUNCTION WITH THE SERIAL INPUT AND OUTPUT ROUTINES

```
81A0  7B          DELAY  MOV    A,E
81A1  B0                 ORA    B
81A2  D3 05              OUT    SERAL
81A4  2B                 DCX    H
81A5  7C                 MOV    A,H
81A6  B5                 ORA    L
81A7  C2 A0 81           JNZ    DELAY
81AA  7A                 MOV    A,D
81AB  1F                 RAR
81AC  7B                 MOV    A,E
81AD  1F                 RAR
81AE  5F                 MOV    E,A
81AF  7A                 MOV    A,D
81B0  1F                 RAR
81B1  57                 MOV    D,A
81B2  C9                 RET
```

SERIAL OUTPUT ROUTINE

```
81B3  87          SROUT  ADD    A
81B4  47                 MOV    B,A
81B5  5F                 MOV    E,A
81B6  3E 0B              MVI    A,11
81B8  4F                 MOV    C,A
81B9  17                 RAL
81BA  57                 MOV    D,A
81BB  2A F3 83    OLOOP  LHLD   SCON
81BE  2A F3 83           LHLD   SCON
81C1  29                 DAD    H
81C2  2B                 DCX    H
81C3  CD A0 81           CALL   DELAY
81C6  0D                 DCR    C
81C7  C2 BB 81           JNZ    OLOOP
81CA  C9                 RET
```

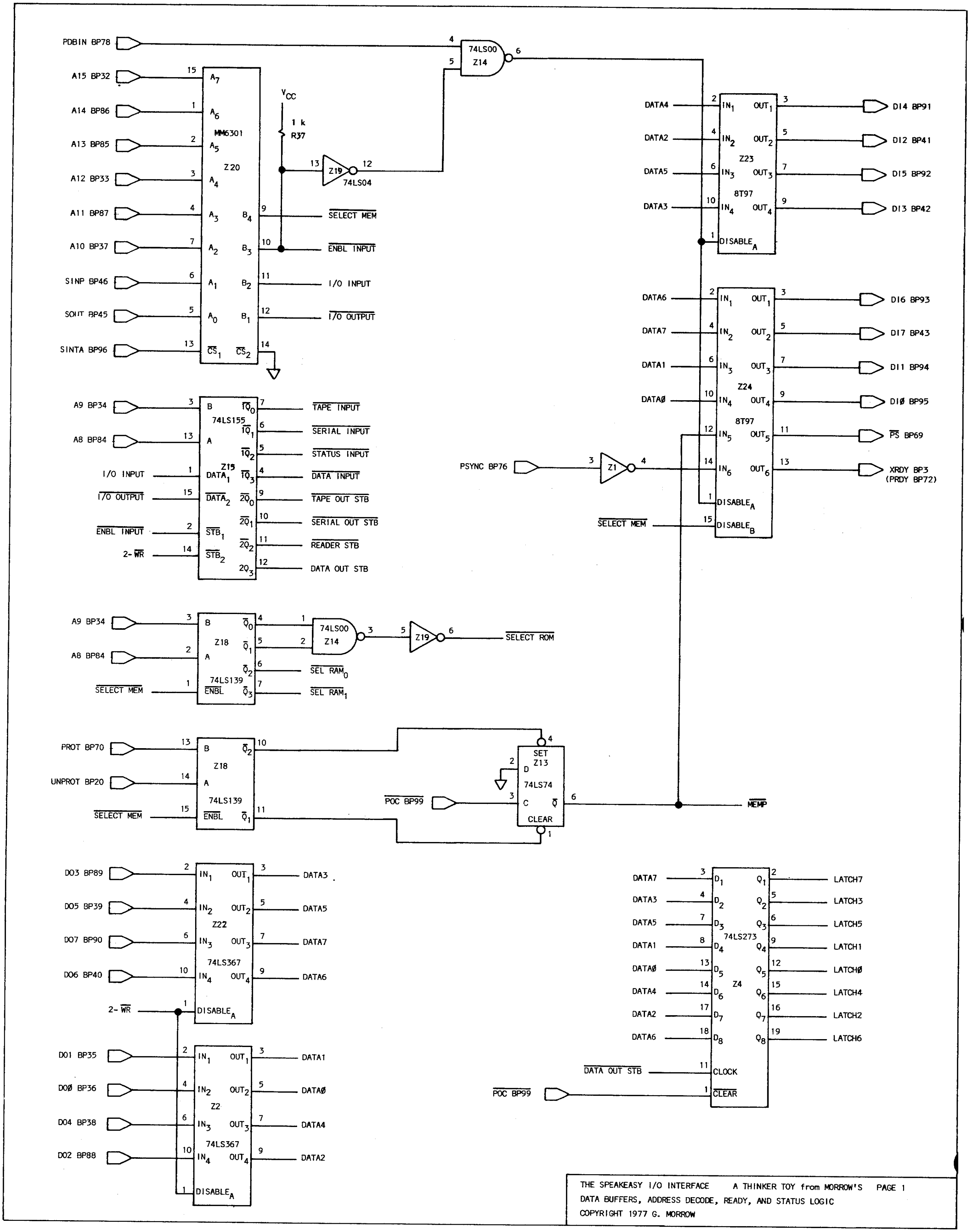
SERIAL DEVICE SPEED DETECTION ROUTINE

```
81CB  31 E0 83           LXI    6,83E0H
81CE  16 06       DETCT  MVI    D,6
81D0  4A                 MOV    C,D
81D1  0C          TLOOP  INR    C
81D2  21 01 00           LXI    H,1
81D5  DB 05       GETIN  IN     SERAL
81D7  D3 05              OUT    SERAL
81D9  A9                 XRA    C
81DA  1F                 RAR
81DB  23                 INX    H
81DC  D2 D5 81           JNC    GETIN
81DF  E5                 PUSH   H
81E0  15                 DCR    D
81E1  C2 D1 81           JNZ    TLOOP
81E4  E1                 POP    H
81E5  E1                 POP    H
81E6  C1                 POP    B
81E7  09                 DAD    B
81E8  C1                 POP    B
81E9  09                 DAD    B
81EA  C1                 POP    B
81EB  C1                 POP    B
81EC  16 03              MVI    D,3
81EE  AF          SLOOP  XRA    A
81EF  7C                 MOV    A,H
81F0  1F                 RAR
81F1  67                 MOV    H,A
81F2  7D                 MOV    A,L
81F3  1F                 RAR
81F4  6F                 MOV    L,A
81F5  15                 DCR    D
81F6  C2 EE 81           JNZ    SLOOP
81F9  2B                 DCX    H
81FA  22 F3 83           SHLD   SCON
81FD  C3 FD 81    STOP   JMP    STOP
```

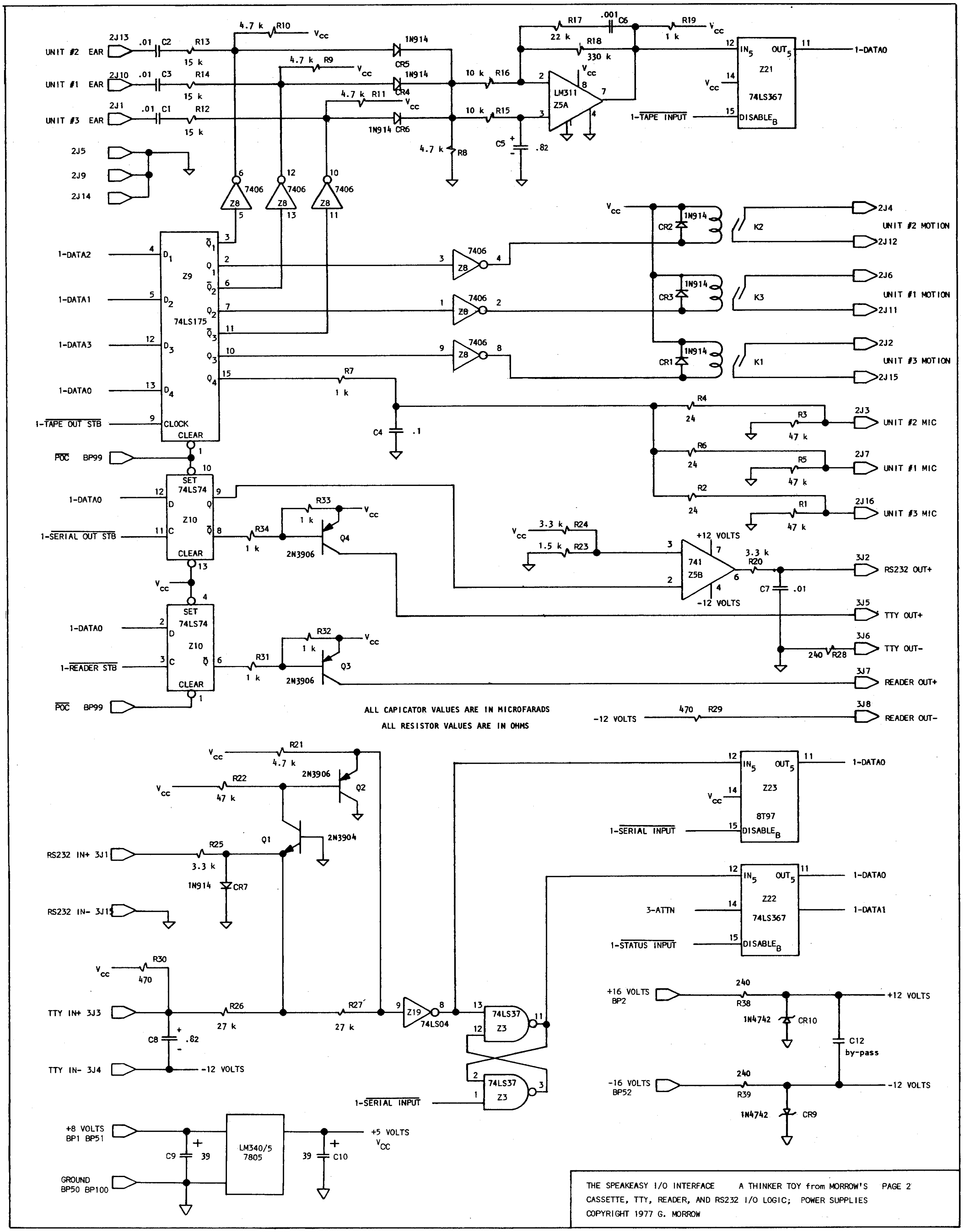
WARRANTY

Parts are warranted to be free from defects in material and workmanship. Defective parts returned postpaid will be exchanged free of charge. Thinker Toy products purchased in kit form are warranted for six months from date of invoice. Thinker Toy products purchased as assembled units are warranted for one year from invoice date. Malfunctioning units whether purchased in kit form or pre-assembled will be repaired, tested, and returned with a minimal charge for postage/handling if in the opinion of Morrow's Micro-Stuff or Thinker Toys care has been exercised in their assembly and/or use. Warranty is void if on inspection by Morrow's or Thinker Toys it is found that the product has been subject to improper assembly or abuse. Charges will be assessed accordingly for repair parts and labor. Repair fees will not exceed $25.00 unless prior approval has been obtained from purchaser.

The foregoing warranty is in lieu of all other warranties expressed or implied and in any event is limited to product repair or replacement.



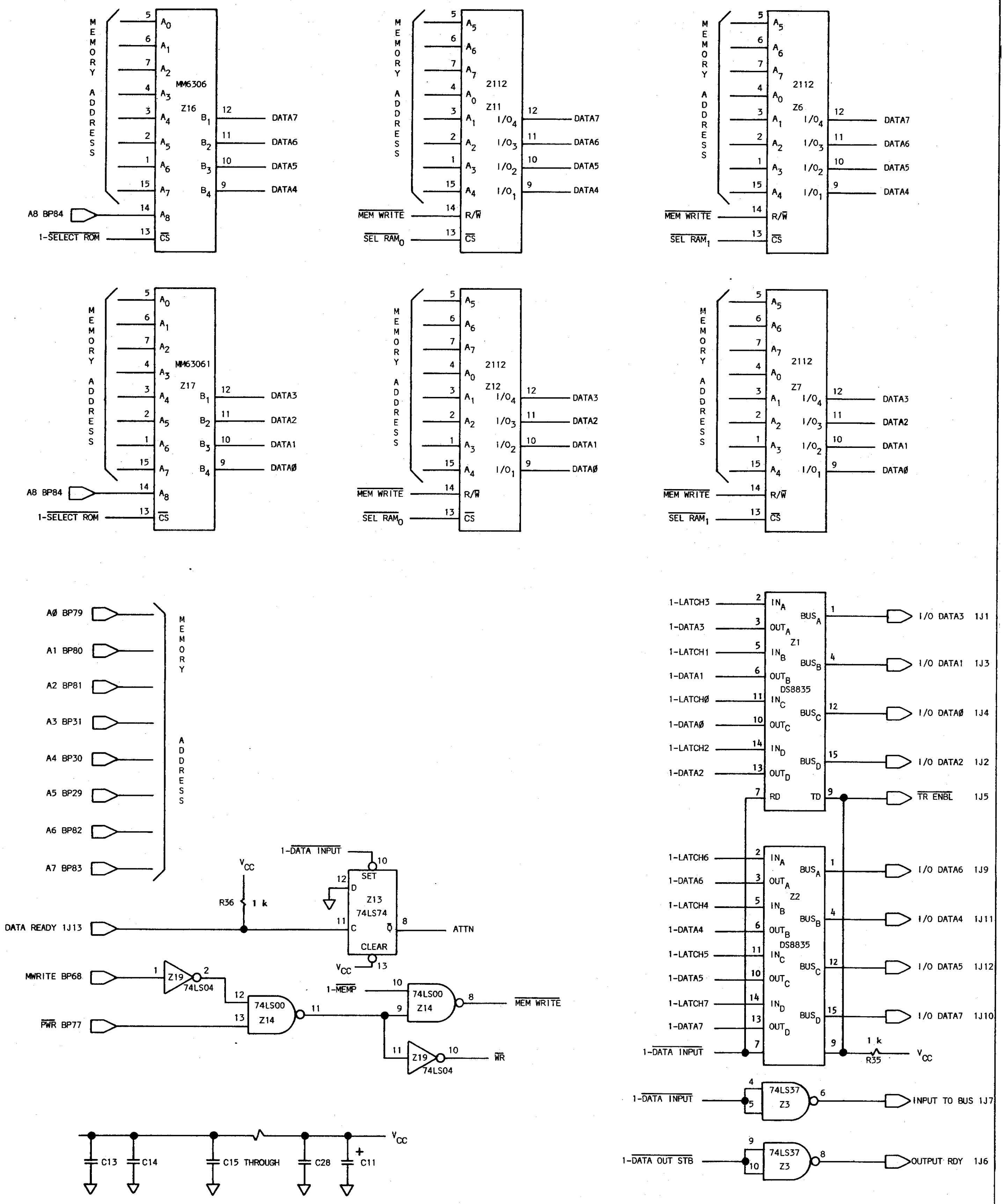
THE SPEAKEASY I/O INTERFACE A THINKER TOY from MORROW'S PAGE 1
DATA BUFFERS, ADDRESS DECODE, READY, AND STATUS LOGIC
COPYRIGHT 1977 G. MORROW



ALL CAPICATOR VALUES ARE IN MICROFARADS

ALL RESISTOR VALUES ARE IN OHMS

THE SPEAKEASY I/O INTERFACE   A THINKER TOY from MORROW'S   PAGE 2
CASSETTE, TTY, READER, AND RS232 I/O LOGIC; POWER SUPPLIES
COPYRIGHT 1977 G. MORROW



THE SPEAKEASY I/O INTERFACE A THINKER TOY from MORROW'S PAGE 3
MEMORY, MEMORY WRITE LOGIC, AND PARALLEL I/O LOGIC
COPYRIGHT 1977 G. MORROW

ROBERT F. HASSARD
3466 Tice Creek Drive, #4
Walnut Creek, Calif. 94595

#5347

October 8, 1979

The Speakeasy
Inv. 5898

Thinker Toys
5221 Central Ave., #9
Richmond, CA

In accordance with your telephoned instructions, I enclose (1) Rev. 3 of my Z-80 Version of Cope, and (2) a Tape Cassette with recordings made with this version.

Side 1 of the Cassette is a recording from 2K of ROM. When I read this back into RAM, the result was accurate, except that F8 was inserted at the beginning, everything displaced by one byte, and the last byte was lost.

Side 2 of the Cassette is a recording from the above described RAM. When I read this back into another section of RAM, another F8 was inserted at the beginning (making two F8s), everything displaced once more, and another byte lost at the end.

As explained in my Sept. 25 letter, the enclosed program takes care of two things. (1) It addresses Speakeasy on the upper eight address lines in the only way that my computer can (as now configured), and (2) it uses constants that adjust for the lack of WAIT states on each fetch.

I have checked BP3 (XRDY) on the oscilloscope in comparison with the CPU Clock (Ø). There are clear and distinct signals on Buss 3, but it is a chopped signal and clearly timed wrong for the purpose ofccreating a WAIT. I am certain that the reason for this is that the circuit on my CPU Board which makes the artificial PSYNC does not (and Can Not) make the signal fast enough to satisfy Speakeasy.

Because of the lack of a WAIT state on each fetch, I am not satisfied that there are not timing problems with the enclosed Z-80 version of COPE. For example, I doubt that the Serial Device Speed Detection will work. I have only tested this program as respects writing and reading tape.

I hope that you can (1) review the Tape handling portions of the program as respect to needs for more PADS, and (2) the balance in all respects.

The only changes that I have made in this revision are recapped on the next page.

Re Cap of changes made to
Cope to produce this Z-80 Version

1. INPUT and OUTPUT Macros are used that cause the CPU to address Speakeasy on the upper eight address lines.

2. The location of two sub-routines, (1) Compute Checksum, and (2) Boots, are moved ahead of the Executive Routine in order to make room for the IN and OUT macros in the Serial sub-routines.

3. Certain Constants are changed so as to compensate for the lack of WAIT states on each fetch.

ORIGINAL COPE		REVISED COPE		
Address	Hex Value	Hex Value	Address	Comment
801B	12	13	801C	Delay Constant
802B	18	1B	802C	# Flux changes for a Zero bit
803E	14	16	803E	Set up for a Zero bit
8046	0B	0C	8045	Set up flux change for a one bit
8049	06	07	8048	Delay Constant
812C	0B	ØC	812F	2400 Hz wave rest bits
8132	06	07	8135	Delay Constant

As stated before, my primary concern is that there may be need for other timing corrections by insering pads. Although there is little room left to do this. There are five NoOps at 80E7 to 80EB, one NoOp at 8154, and two more at the end. The number of machine cycles per line has been entered at the right margin of each page. These assume NO WAIT states.

Sincerely,

Robert F. Hassard

PAGE 1 Z-80 VERSION COPE REV 3 9/22/79

	INSTRUCTION				SOURCE CODE			
ADDRESS	BYTE 1	BYTE 2	BYTE 3	BYTE 4	LABEL	OP CODE	OPERAND	COMMENTS #5347
					Remarks	are	Same	as in
					SPEAKEASY	USER'S		MANUAL
8000	18	00			FLUX	JR	FLUX+2	12
2	3d					DEC	A	28 4
3	20	FB				JR NZ	FLUX	12/7
5	3C					INC	A	4
6	A8					XOR	B	4
7	d9					EXX		4
8	01	04	04			Ld BC	0404	10
B	Ed	79				OUT (c)	A	12
d	d9					EXX		4
E	47					Ld B	A	72 4
F	2B					DEC	HL	-5 6
10	7d					Ld A	L	4
1	B4					OR A	H	28C+67 4
2	79					Ld A	C	1=12 0CH 4
3	20	EB				JRNZ	FLUX	0=27 1BH 12/7
5	C9					RET		10
					WRITE	TAPE	CASSETTE	
8016	18	00			TLOOP	JR	TLOOP+2	12
8	B7					OR	A	4
9	E1					POP	HL	10
A	d1					POP	DE	10
B	0E	13				Ld C	19 Dec.	7
d	7A				WTAPE	Ld A	D	4
E	B3					OR	E	4
F	C8					RET Z		11/5
20	1B					DEC	DE	6
1	d5					PUSH	DE	11
2	5E					Ld E	(HL)	7
3	23					INC	HL	6
4	E5					PUSH	HL	11
5	79					Ld A,	C	4
6	16	03				Ld D	03	7
8	21	08	00		ZBIT	Ld HL	0008	10
B	0E	1B				Ld C	27 Dec.	7
d	Cd	00	80		WRITE	CALL	FLUX	17
30	7B					Ld A,	E	4
1	B2					OR	D	4
2	28	E2				JR Z	TLOOP	12/7
4	7A					Ld A,	D	4
5	1F					RRA		4
6	57					Ld D,	A	4
7	7B					Ld A,	E	4
8	1F					RRA		4
9	5F					Ld E,	A	4

PAGE 2 Z-80 VERSION CODE REV 3 9/22/79

ADDRESS	INSTRUCTION				SOURCE CODE			COMMENTS
	BYTE 1	BYTE 2	BYTE 3	BYTE 4	LABEL	OP CODE	OPERAND	
					WRITE	TAPE	(CONT.)	
803A	3E	00				Ld A	00	7
C	7F					Ld A	A	4
d	3E	16				Ld A	22 Dec.	7
F	30	E7				JR NC	2 BIT	12/7
41	21	10	00			Ld HL	16 Dec.	10
4	0E	0C				Ld C	12 Dec.	7
6	7F					Ld A,	A	4
7	3E	07				Ld A	7	7
9	18	E2				JR	WRITE	12
			READ		TAPE	CASSETTE	with running Sum, etc.	
804B	C5				TREAD	PUSH	BC	11
C	d5					PUSH	DE	11
d	E5					PUSH	HL	11
E	21	00	F0			Ld HL	F000	10
51	E5				SYNC	PUSH	HL	11
2	AF					XOR	A	4
3	21	F8	83			Ld HL	PNTR	10
6	11	F7	83			Ld DE	TOTAL	10
9	12					Ld (DE)	A	7
A	77				STORE	Ld (HL),	A	7
B	2C					INC	L	4
C	20	FC				JR NZ	STORE	12/7
E	47					Ld B	A	4
F	3E	04				Ld A	04	To Upper 8 Addr Lines 7
61	dB	04				IN A	(04)	11
3	4F					Ld C,	A	4
4	0C				SUM	INC	C	4
5	3E	04				Ld A	04	To Upper 8 Addr Lines 7
7	dB	04				IN A	(04)	11
9	A9					XOR	C	4
A	1F					RRA		4
B	04					INC	B	4
C	CA	d8	80			JMP Z	LONG	10
F	30	F4				JR NC	SUM+1	12/7
71	2C					INC	L	4
2	3E	F8				Ld A	MASK	7
4	B5					OR	L	4
5	6F					Ld L,	A	4
6	3E	1C				Ld A	HIGH	7
8	B8					CP	B	4
9	dA	d8	80			JMP C	LONG	10
C	1A					Ld A,	(DE)	7
d	96					SUB	(HL)	7
E	80					ADD A	B	4
F	12					Ld (DE),	A	7

Page 3 Z-80 VERSION CODE Rev 3 9/22/79

ADDRESS	INSTRUCTION				SOURCE CODE			COMMENTS
	BYTE 1	BYTE 2	BYTE 3	BYTE 4	LABEL	OP CODE	OPERAND	
					READ TAPE (cont.)			
8080	dE	74				SBC A	CONST	4
2	30	07				JR NC	BITS	12/7
4	70					Ld (HL)	B	7
5	06	03				Ld B	03	7
7	18	00			PAD 1	JR	PAD 1 +2	12
9	18	d9				JR	SUM	12
B	1E	08			BITS	Ld E	08	7
d	2E	0B				Ld L	11 dec	7
F	16	0F			RENTR	Ld D	15 dec	7
91	26	0C				Ld H	12 dec	7
3	0C				COUNT	INC	C	4
4	3E	04				Ld A	04	7
6	dB	04				IN A	(04)	11
8	A9					XOR	C	4
9	1F					RRA		4
A	2d					DEC	L	4
B	30	F7				JR NC	COUNT+1	12/7
d	2E	0d				Ld L	13 dec.	7
F	7A					Ld A,	D	4
A0	FA	A7	80			JMP M	TWO	10
3	d6	01				SUB	01	4
5	18	04				JR	NEXT	12
7	d6	02			TWO	SUB	02	4
9	18	00				JR	NEXT	12
B	57				NEXT	Ld D,	A	4
C	25					DEC	H	4
d	30	E4				JR NC	COUNT	12/7
F	7C					Ld A	H	4
B0	17					RLA		4
1	78					Ld A,	B	4
2	1F					RRA		4
3	47					Ld B	A	4
4	2d					DEC	L	4
5	2d					DEC	L	4
6	7F					Ld A,	A	4
7	1d					DEC	E	4
8	C2	8F	80			JMP NZ	RENTR	10
B	F1					POP	AF	10
C	E1					POP	HL	10
d	d1					POP	DE	10
E	F1					POP	AF	10
F	F5					PUSH	AF	11
C0	28	0d				JR Z	VERFY	12/7
2	38	01				JR C	INCR	12/7
4	70					Ld (HL),	B	7
5	23				INCR	INC	HL	4
6	1B					DEC	DE	6

ADDRESS	INSTRUCTION				SOURCE CODE			COMMENTS	
	BYTE 1	BYTE 2	BYTE 3	BYTE 4	LABEL	OP CODE	OPERAND		
					READ	TAPE	(cont.)		
80C7	7A					Ld A,	D		4
8	B3					OR	E		4
9	C2	4C	80			JMPNZ	TREAD+1		10
C	F1					POP	AF		10
d	AF					XOR	A		4
E	C9					RET			10
F	7E				VERFY	Ld A	(HL)		7
d0	A8					XOR	B		4
1	28	F2				JR Z	INCR		12/7
3	32	F5	83			Ld (83F5), A		DERR	13
6	18	Ed				JR	INCR		12
8	E1				LONG	POP	HL		10
9	23					INC	HL		4
A	7C					Ld A,	H		4
B	B5					OR	L		4
C	C2	51	80			JMP NZ	SYNC		10
F	3d					DEC	A		4
E0	32	F6	83			Ld (83F6), A		SERR	13
3	E1					POP	HL		10
4	d1					POP	DE		10
5	C1					POP	BC		10
6	C9					RET			10
7	00	00	00						
A	00	00							
				COMPUTE	CHECK-SUM	ROUTINE			
80EC	E5				CHECK	PUSH	H		11
d	21	00	00			Ld HL	0000		10
F0	44					Ld B,	H		4
1	E3				GDATA	EX (SP),	HL		19
2	4E					Ld C,	(HL)		7
3	23					INC	HL		6
4	E3					EX (SP),	HL		19
5	09					ADD HL,	BC		15
6	1B					DEC	DE		6
7	7A					Ld A,	D		4
8	B3					OR A	E		4
9	20	F6				JR NZ	GDATA		12/7
B	d1					POP	DE		10
C	C9					RET			10
80Fd	31	dE	83		BOOTS	Ld SP	83dE	LSAVE	10
8100	21	00	82			Ld HL	8200		10
3	11	00	01			Ld DE	256 dec		10
6	3E	82				Ld A	82H		7
8	0E	00				Ld C	00		7
						JMP	COPE	continue 810A	

PAGE 5   Z-80 VERSION COPE   REV 3   9/22/79

CASSETTE OPERATING EXECUTIVE

ADDRESS	INSTRUCTION BYTE 1	BYTE 2	BYTE 3	BYTE 4	SOURCE CODE LABEL	OP CODE	OPERAND	COMMENTS
810A	47				COPE	Ld B	A	4
B	22	dE	83			Ld(83DE)	HL	16
E	21	00	00			Ld HL	0000	10
11	39					ADD HL	SP	10
2	31	dE	83			Ld SP	83DE	10
5	E5					PUSH	HL	11
6	d9					EXX		4
7	01	04	04			Ld BC	0404	10
A	Ed	79				OUT(c)	A	12
C	d9					EXX		4
d	F5					PUSH	AF	11
E	21	00	00			LdHL	0000	10
21	2B				W COPE	DEC	HL	6
2	7C					Ld A	H	4
3	B5					OR	L	4
4	20	FB				JRNZ	WCOPE	12/7
6	F1					POP	AF	10
7	F5					PUSH	AF	11
8	1F					RRA		4
9	30	21				JR NC	READ	12/7
B	21	C0	5d			Ld HL	24000	10
E	0E	0C				Ld C	10	7
30	79					Ld A	C	4
1	Cd	00	80			CALL	FLUX	17
4	0E	07				Ld C	06	7
6	2A	dE	83			Ld HL	(83DE)	16
9	Cd	1d	80			CALL	WTAPE	17
C	F1				EXIT	POP	AF	10
d	B7					OR	A	4
E	F2	49	81			JP P	RETRN	10
41	AF					XOR	A	4
2	d9					EXX		4
3	01	04	04			Ld BC	0404	10
6	Ed	79				OUT(c)	A	12
8	d9					EXX		4
9	E1				RETRN	POP	HL	10
A	F9					Ld SP	HL	6
B	C9					RET		10
C	2A	dE	83		READ	LdHL	(83DE)	16
F	Cd	4B	80			CALL	TREAD	17
52	18	E8				JR	EXIT	12
4	00							4

. Page 6   Z-80 VERSION COPE   Rev 3   9/22/79

ADDRESS	INSTRUCTION BYTE 1	 BYTE 2	 BYTE 3	 BYTE 4	SOURCE CODE LABEL	 OP CODE	 OPERAND	COMMENTS
				SERIAL DATA INPUT ,				etc.
8155	78				INPUT	Ld A,	B	4
6	07					RLCA		4
7	d9					EXX		4
8	01	06	06			Ld BC	0606	10
B	Ed	79				OUT(c),	A READR	12
d	d9					EXX		4
E	2A	F3	83			Ld HL,	(83F3)	SCON 16
61	E5				SWAIT	PUSH	HL	11
2	1E	FF				Ld E,	-1	7
4	3E	05			SLOOK	Ld A,	05	7
6	dB	05				IN A,	(05) SERAL	11
8	1F					RRA		4
9	38	F9				JR C	SLOOK	12/7
B	Cd	98	81			CALL	DELAY	17
E	E1					POP	HL	10
F	3E	05				Ld A,	05	7
71	dB	05				INA,	(05) SERAL	11
3	1F					RRA		4
4	38	EB				JR C	SWAIT	12/7
6	16	FF				Ld d,	-1	7
8	AF					XOR	A	4
9	d9					EXX		4
A	01	06	06			Ld BC,	0606	10
d	Ed	79				OUT(c),	A READR	12
F	d9					EXX		4
80	E5				GTBIT	PUSH	HL	11
1	29					ADD HL,	HL	11
2	2B					DEC	HL	6
3	3E	05				Ld A,	05	7
5	dB	05				IN A,	(05) SERAL	11
7	5F					Ld E,	A	4
8	Cd	98	81			CALL	DELAY	17
B	E1					POP	HL	10
C	38	F2				JR C	GTBIT	12/7
E	3E	01				Ld A	01	7
90	d9					EXX		4
1	01	05	05			Ld BC,	0505	10
4	Ed	79				OUT(c),	A SERAL	12
6	d9					EXX		4
7	C9					RET		10

Page 7 Z-80 VERSION CODE Rev 3 9/22/79

	INSTRUCTION				SOURCE CODE			
ADDRESS	BYTE 1	BYTE 2	BYTE 3	BYTE 4	LABEL	OP CODE	OPERAND	COMMENTS
					SERIAL DELAY ROUTINE			
8198	7B				DELAY	Ld A,	E	4
9	B0					OR	B	4
A	d9					EXX		4
B	01	05	05			Ld BC,	0505	10
E	Ed	79				OUT (c),	A SERAL	12
A0	d9					EXX		4
1	2B					DEC	HL	6
2	7C					Ld A,	H	4
3	B5					OR	L	4
4	20	F2				JR NZ	DELAY	12/7
6	7A					Ld A,	d	4
7	1F					RRA		4
8	7B					Ld A,	E	4
9	1F					RRA		4
A	5F					Ld E,	A	4
B	7A					Ld A,	d	4
C	1F					RRA		4
d	57					Ld d,	A	4
E	C9					RET		10
					SERIAL OUTPUT ROUTINE			
81AF	87				SROUT	ADD A,	A	4
B0	47					Ld B,	A	4
1	5F					Ld E,	A	4
2	3E	0B				Ld A,	11 dec.	7
4	4F					Ld C,	A	4
5	17					RLA		4
6	57					Ld d,	A	4
7	2A	F3	83		OLOOP	Ld HL	(83F3) SCON	16
A	2A	F3	83			Ld HL,	(83F3) SCON	16
d	29					Add HL,	HL	11
E	2B					DEC	HL	6
F	Cd	98	81			CALL	DELAY	17
C2	0d					DEC	C	4
3	20	F2				JR NZ	OLOOP	12/7
5	C9					RET		10

SERIAL DEVICE SPEED DETECTION

ADDRESS	INSTRUCTION BYTE 1	BYTE 2	BYTE 3	BYTE 4	SOURCE CODE LABEL	OP CODE	OPERAND	COMMENTS	
81C6	31	E0	83			Ld SP,	83E0		10
9	16	06			DETCT	Ld d,	06		7
B	4A					Ld C,	d		4
C	0C				TLOOP	INC	C		4
d	21	01	00			Ld HL	0001		10
d0	3E	05			GETIN	Ld A,	05		7
2	dB	05				IN A,	(05) SERAL	Timing here	11
4	d9					EXX		is very	4
5	01	05	05			Ld BC,	0505	different	10
8	Ed	79				OUT(C),	A SERAL		12
A	d9					EXX			4
B	A9					XOR	C		4
C	1F					RRA			4
d	23					INC	HL		6
E	30	F0				JR NC	GETIN		12/7
E0	E5					PUSH	HL		11
1	15					DEC	D		4
2	20	E8				JR NZ	TLOOP		12/7
4	E1					POP	HL		10
5	E1					POP	HL		10
6	C1					POP	BC		10
7	09					ADD HL,	BC		11
8	C1					POP	BC		10
9	09					ADD HL,	BC		11
A	C1					POP	BC		10
B	C1					POP	BC		10
C	16	03				Ld d,	03		7
E	AF				SLOOP	XOR	A		4
F	7C					Ld A,	H		4
F0	1F					RRA			4
1	67					Ld H,	A		4
2	7d					Ld A,	L		4
3	1F					RRA			4
4	6F					Ld L,	A		4
5	15					DEC,	D		4
6	20	F6				JR NZ	S LOOP		12/7
8	2B					DEC	HL		6
9	22	F3	83			Ld	(83F3),HL SCON		16
C	18	FE			STOP	JR	STOP		12
E	00	00							