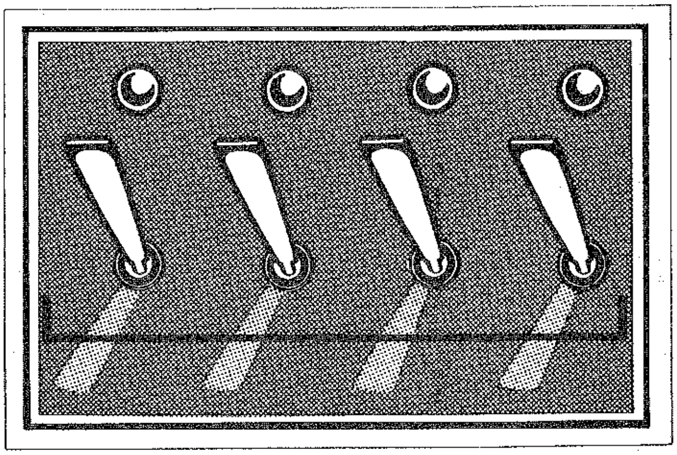
altair T.M. 680b



System Monitor Manual

TABLE OF CONTENTS

I ABSTRACT page 2

II NOTES ON THE FORMAT OF THIS MANUAL page 3

III STARTING UP THE PROM MONITOR page 4

IV DESCRIPTION OF THE MONITOR COMMANDS page 6

V USER PROGRAM DEBUGGING WITH THE PROM MONITOR page 12

VI PAPER TAPE FORMAT page 15

VII PROM MONITOR MEMORY USE INFORMATION page 17

VIII BAUDOT TELETYPE OPTION INFORMATION page 21

IX PROM MONITOR SOURCE LISTING (ACIA VERSION) page 25

X PROM MONITOR SOURCE LISTING (BAUDOT VERSION) page 31

MITS, Inc. ©1976
2450 Alamo S.E.
Albuquerque, N.M. 87106

I ABSTRACT

This document describes the functions and operating procedures of the Altair 680b PROM Monitor, a system program which allows the user to examine and change the contents of memory locations, load formatted object tapes into memory, start program execution at a specified address, and debug user programs. A source listing of the PROM Monitor is included so that its I/O and hexadecimal conversion routines may be utilized by user programs.

II NOTES ON THE FORMAT OF THIS MANUAL

1) All numbers used in this document are hexadecimal (base 16) unless otherwise indicated.

2) In the examples provided in this document, underscoring is used to indicate user typed information.

3) The symbol <CR> is used to represent a carriage return.

4) There are two versions of the PROM Monitor, one which supports the use of the ACIA chip, and one for use with a Baudot Teletype. All information in this manual applies to both versions of the Monitor, except where otherwise noted.

5) Symbolic addresses which are referenced but not defined in the examples, such as OUTCH and OUT2H, are entry points in the PROM Monitor. Refer to appropriate source listing (Section IX for the ACIA version and Section X for the Baudot version) for detailed information on these routines.

6) Assembly code examples follow the conventions of the 680B Resident Assembler.

III STARTING UP THE PROM MONITOR

A) Power up sequence

1) Strap the appropriate bits at location FØØ2 to indicate the presence of a terminal, the type of terminal, and the number of stop bits to be used. (See the 68ØB Operator's Manual.)

2) Turn the Altair T.M. computer on.

3) Turn the terminal on.

4) Switch the Halt-Run switch to the Halt position.

5) Actuate the Reset switch.

6) Switch the Halt-Run switch to the Run position.

7) The PROM Monitor will respond by sending a carriage return and line feed to the terminal and printing a ".". The "." is the Monitor's prompt character which indicates that the Monitor is ready to accept a command.

> NOTE
>
> Use steps 4 through 7 to start the Monitor if the system is already powered up.

B) Entering the PROM Monitor from a User Program

There are three methods of entering the Monitor from a user program. The first method is to include the following instructions at the appropriate place in the program.

```
LDX $FFFE  RESTART VECTOR TO X REGISTER
```

JMP X JUMP TO RESTART ADDRESS

This has the same effect as doing a Reset from the front panel. The Monitor is entered at its reset entry point, causing the stack pointer and all system parameters to be initialized.

> NOTE
>
> If the user program is outputting to the terminal just prior to the execution of these instructions, the last character sent to the terminal may be lost when the Monitor initializes the terminal control register.

The second method of entering the Monitor from a user program is to include the following instruction at the appropriate place in the program.

```
JMP CRLF
```

The symbol CRLF must be correctly defined in the user program for the version of the Monitor being used (ACIA or Baudot). The Monitor is entered, the stack pointer is loaded from SAVSTK (ØØF6 and ØØF7), and a carriage return, line feed, and the Monitor's prompt character are sent to the terminal.

The third method of entering the Monitor from a user program is to place a SWI (software interrupt) instruction at the appropriate place in the program. This method is generally used for program debugging and therefore discussion of this feature is delayed until section V.

IV DESCRIPTION OF MONITOR COMMANDS

M - Memory Examine and Deposit Command

Purpose - To examine and optionally modify the contents of a single memory byte.

Usage -

1) Type M in response to the Monitor's ".".

2) A space will be printed.

3) Type the four digit hexadecimal address of the byte to be examined.

4) The two digit hexadecimal contents of the specified byte will be printed, preceded by and followed by a space.

5) To change the contents of the specified byte, enter the new contents by typing two hexadecimal digits.

6) To leave the contents of the specified byte unaltered, type a carriage return (or any other non-hexadecimal character).

Examples -

1) To examine and leave unaltered the contents of ØØA2, the following command is used:

```
.M ØØA2 FF <CR>
```

2) To deposit a Ø9 in location ØØ72, the following command is used:

```
.M ØØ72 E1 Ø9
```

(Note that a carriage return is not used.)

> NOTE
>
> The contents of the specified byte are not changed until two valid hexadecimal digits are entered. Therefore, if an invalid digit is typed, the contents of the location will remain unchanged.

N - Memory Deposit and Examine Next Command

Purpose - Used after an M command to examine and optionally modify the contents of the next sequential memory byte.

Usage -

1) Type N in response to the Monitor's ".".

2) The Monitor will type the next sequential memory address, preceded by and followed by a space. The contents of the byte will be printed, followed by a space.

3) To change the contents of the specified byte, enter the new contents by typing two hexadecimal digits.

4) To leave the contents of the specified byte unaltered, type a carriage return (or any other non-hexadecimal character).

Examples -

1) To load a string of ASCII characters into successive memory bytes starting at location ØØ5Ø, use the following commands:

```
.M ØØ5Ø ØØ 4D

.N ØØ51 ØØ 49

.N ØØ52 ØØ 54

.N ØØ53 ØØ 53
```

2) To check and correct a sequence of instructions located at ØØ15 through ØØ18, the following commands are used:

```
.M ØØ15 4C <CR>

.N ØØ16 5C <CR>

.N ØØ17 36 32

.N ØØ18 37 <CR>
```

J - Jump to Specified Address Command

Purpose - To start program execution at a specified address.

Usage -

1) Type J in response to the Monitor's ".".

2) A space will be printed.

3) Type the four digit hexadecimal address at which execution is to begin.

4) The processor will jump to the specified location and start execution of the program stored there.

Example -

To start execution of a program which starts at Ø2F3, the following command is used:

```
.J Ø2F3
```

L - Load Paper Tape Command

Purpose - To load formatted object tapes into memory. (See Section VI for paper tape format.)

Usage -

1) Type L in response to the Monitor's ".".

2) Place the paper tape in the reader and start the reader.

Loading begins with the first data record (type S1). Any information preceding the first data record, including the header record (type SØ) is ignored.

Normal termination of the load occurs when an end of file record (type S9) is encountered. Control returns to the Monitor's command decoding section and any information following the S9 on the tape is interpreted as Monitor commands. Therefore, the paper tape reader should be turned off as soon as the S9 is printed on the terminal.

If a checksum error occurs while the tape is being read, control is returned to the Monitor's command decoding section and the rest of the information on the tape is interpreted as Monitor commands. If this occurs, the paper tape reader should be turned off and the paper tape should be reloaded from its beginning.

Suppressing Teletype Echo

NOTE

This information applies only to the ACIA version of the PROM Monitor.

While loading a paper tape, Teletype echo can be suppressed by one of two methods. The first method is to use the Monitor's M command to store an FF into the Monitor's echo flag (location ØØF3). The command

```
M ØØF3 Ø3 FF
```

turns off Teletype echoing. The L command can then be used to load the paper tape. (The L will not be echoed!) When the load is completed, the command

```
M ØØF3 FF ØØ
```

is used to restore Teletype echoing. (Only the FF, which is printed by the Monitor, will appear on the terminal!)

NOTE

Only the most significant bit of the echo flag affects Teletype echoing. Therefore, any number loaded into ØØF3 which has bit 7 set will suppress echoing, and any number loaded into ØØF3 which has bit 7 clear will restore echoing.

The second method of suppressing Teletype echo is to have the first data block of the paper tape load an FF into location ØØF3 and to have the last data block load a ØØ into location ØØF3. This can be accomplished by including the following mnemonics in an assembly code program.

```
NAM EXAMPL
ORG $00F3
FCB $FF          TURN OFF ECHO FOR LOAD

(PROGRAM STATEMENTS)

ORG $00F3
FCB 0            RESTORE TTY ECHO
END
```

This is the method used on all MITS supplied paper tapes. When using this method, a typical load looks like:

```
.L S00B00004D454D5445535420B5
S10400F3FF08
S9
.
```

If a checksum error occurs, Teletype echoing will remain off. The command

```
.M 00F3  FF 00
```

can be used to restore echoing. (Only the FF will appear on the terminal!)

P - Proceed From Program Breakpoint Command

Purpose - To proceed from a program breakpoint.

Usage -

1) Type P in response to the Monitor's ".".

2) Program execution will be resumed.

> **NOTE**
>
> A discussion of program breakpoints is included in Section V.

V USER PROGRAM DEBUGGING WITH THE PROM MONITOR

Setting Program Breakpoints

When a program is not performing properly, it is often helpful to stop program execution at strategic points for the purpose of displaying and/or modifying the contents of the processor registers and memory locations. This is known as setting program breakpoints.

The PROM Monitor allows a program breakpoint to be set by insertion of a SWI (software interrupt) instruction at the point in the program where the break is to occur. When the SWI instruction is executed, the status of the processor is pushed onto the stack according to the format shown in Table 5-1. The PROM Monitor gains control of the processor and may be used to examine and/or modify the contents of the registers and memory locations.

Stack Pointer	>	
SP+1	>	Condition Codes
SP+2	>	Accumulator B
SP+3	>	Accumulator A
SP+4	>	Index Reg (High Order Byte)
SP+5	>	Index Reg (Low Order Byte)
SP+6	>	Program Counter (High Order Byte)
SP+7	>	Program Counter (Low Order Byte)

TABLE 5-1

When the Monitor is entered at a program breakpoint, the stack pointer is saved in locations 00FA and 00FB. When an N command is executed, the contents of 00FA and 00FB are incremented by one and then used as the address of the next memory byte to be examined. Therefore, if an N command is issued directly after entering the Monitor at a breakpoint, the address displayed will be SP+1 (see Table 5-1) and the contents displayed will be the contents of the condition codes register. Further N commands will display the contents of the remaining processor registers in the order shown in Table 5-1.

Alternatively, the contents of the stack pointer can be determined by using the M and N commands to examine locations 00F6 and 00F7, where the Monitor stores the high and low bytes of the stack pointer, respectively. Once the contents of the stack pointer have been determined, the M and N commands can be used in conjunction with Table 5-1 to examine and/or modify the contents of the processor registers.

The P command is used to continue program execution after a breakpoint. The P command causes the stack pointer to be loaded from locations ØØF6 and ØØF7 and the other processor registers to be pulled from the stack. Program execution is resumed at the address of the SWI instruction that caused the break, plus one.

> NOTE
>
> The contents of the stack pointer may be changed by modifying the contents of locations ØØF6 and ØØF7. However, great caution should be exercised when so doing since the P command causes the processor registers to be pulled from the stack.

Any number of breakpoints may be present in a program at one time. It should be clear that insertion of a SWI instruction may make re-assembly of the program necessary. A breakpoint can be removed by replacing the SWI instruction with a NOP or by deleting the SWI instruction and re-assembling the program.

Breakpoint Routines

Whenever the PROM Monitor is entered at a program breakpoint, the flag BRKADR (location F2) is checked. If the most significant bit (bit 7) of BRKADR is clear (=Ø) then the Monitor assumes processor control. (This is the normal course of events since the Monitor initializes BRKADR to Ø3 whenever the Reset function is performed.) However, if the most significant bit of BRKADR is set (=1), which can be accomplished by using the command

```
M ØØF2 Ø3 FF
```

or including the instruction

```
COM      $F2      SET BRKADR FLAG
```

in a program, then control is transferred to location ØØØØ when a program breakpoint occurs. This feature can be used to perform special functions when program breakpoints occcur. Two examples of the use of this feature are given below.

1) This example illustrates the use of a breakpoint routine to print the contents of the processor's registers and continue program execution each time a program breakpoint occurs.

```
                ORG      Ø        BREAKPOINT ROUTINE ADDRESS
                LDA B    #@15     SEND CR AND LF
                JSR      OUTCH    TO TERMINAL
                LDA B    #@12
                JSR      OUTCH
                TSX               X POINTS TO PROCESSOR STATUS
                LDA B    #7       INITIALIZE COUNTER
        LOOP    LDA A    X        BYTE OF STATUS TO A REG
                PSH B             OUT2H & OUTS CLOBBER B REG
                JSR      OUT2H    PRINT OUT BYTE OF STATUS
                JSR      OUTS     SPACE OVER
                PUL B             RESTORE B REG
                INX               BUMP POINTER
                DEC B             DECREMENT COUNTER
                BNE      LOOP     IF NOT DONE, KEEP PRINTING
                RTI               CONTINUE PROGRAM EXECUTION
```

2) This example illustrates the use of a breakpoint routine to examine the contents of the A register and transfer control to the Monitor if A is clear (contains all zeroes). If A is not clear, program execution continues. This type of routine is used to implement "conditional breakpoints".

```
                ORG      Ø
                JMP      $Ø3ØØ    THIS BREAKPOINT ROUTINE
                ORG      $Ø3ØØ    STARTS AT Ø3ØØ
                TST A             TEST CONTENTS OF A REG
                BNE      CONTIN   A ALL ZEROES?
                JMP      CRLF     YES, JUMP TO MONITOR
        CONTIN  RTI               NO, CONTINUE PROG EXEC
```

VI PAPER TAPE FORMAT

The PROM Monitor supports the paper tape format established by Motorola.

The first character of a record is an S. The digit following the S defines the type of record.

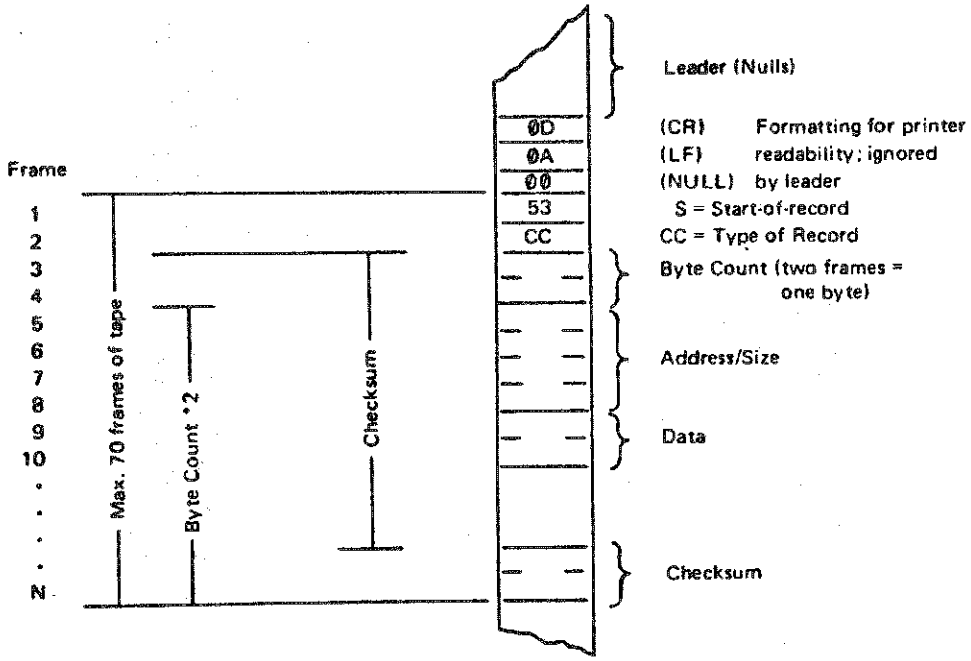
S0 = Header Record
S1 = Data Record
S9 = End of File Record

Header records (type S0) contain the program name, and are ignored by the PROM Monitor. The end of file record (type S9) causes the Monitor to terminate the loading process. Data records (type S1) contain the actual data to be loaded and are of the form:

```
S1NNAAAADDDDDDDDDD.........DDCC
```

where S1 specifies that the record is a data record, NN is a two digit hexadecimal byte count specifying the number of remaining bytes in the record (1 byte = 2 frames of tape), AAAA is the 4 digit hexadecimal starting address of the data block, each DD pair consists of two hexadecimal digits which are combined to form a byte, and CC is the cnecksum of all preceding frames (excluding the S and 1). The checksum is the one's complement of the binary sum of the byte count, the address, and the data bytes.

Further information concerning the paper tape format is given in Figure 6-1.



Frames 3 through N are hexadecimal digits (in 7-bit ASCII) which are converted to BCD. Two BCD digits are combined to make one 8-bit byte.

The checksum is the one's complement of the summation of 8-bit bytes.

Frame	CC = 30 Header Record		CC = 31 Data Record		CC = 39 End-of-File Record	
1. Start-of-Record	53	S	53	S	53	S
2. Type of Record	30	0	31	1	39	9
3. Byte Count	31	12	31	16	30	03
4.	32		36		33	
5.	30		31		30	
6. Address/Size	30	0000	31	1100	30	0000
7.	30		30		30	
8.	30		30		30	
9. Data	34	48-H	39	98	46	FC
10.	38		38		43	(Checksum)
.	34	44-D	30	32		
.	34		32			
.	35	52-R	41	A8 (Checksum)		
.	32		48			
.						
.						
N. Checksum	39	9E				
	45					

Reprinted with permission from Motorola.

FIGURE 6-1. Paper Tape Format

VII PROM MONITOR MEMORY USE INFORMATION

Monitor Memory Location

The ACIA version of the PROM Monitor is 256 bytes long and resides in locations FFØØ through FFFF. The Baudot version of the Monitor is 512 bytes long and resides in locations FEØØ through FFFF.

Monitor Stack

The stack pointer is initialized to ØØF1 whenever the Monitor is entered at its reset entry point. The stack pointer can be changed by using the Monitor's M and N commands to alter the contents of SAVSTK (see Monitor flags below)

NOTE

The contents of SAVSTK should generally not be changed when the Monitor is entered at a program breakpoint as this will cause the P command to operate improperly.

Monitor Flags

Locations ØØF2 through ØØFF are reserved for use by the Monitor. These locations are assigned as described below. With the exceptions of BRKADR, ECHO, and SAVSTK, these locations should generally not be tampered with.

BRKADR (ØØF2) - BREAKPOINT ADDRESS FLAG

If bit 7 of BRKADR is clear (=Ø) the Monitor gains processor control when a program breakpoint occurs. If bit 7 is set, control is transferred to location ØØØØ when a breakpoint occurs. See Section V for further information.

ECHO (ØØF3) - TELETYPE ECHO FLAG

(Applies to ACIA version only)

If bit 7 of ECHO is clear, Teletype input is echoed. If bit 7 is set, Teletype echo is suppressed. See Page 9 for further information.

EXTFLG (ØØF4) - EXTENDED CHARACTER FLAG

(Applies to Baudot version only)

EXTFLG is set when the Baudot character input routine receives the extend character and cleared after the extended character is received. See Section VIII for information on the Baudot version of the Monitor.

BUFULL (ØØF5) - BUFFER FULL FLAG

(Applies to Baudot version only)

If BUFULL is clear then the contents of the character buffer are not current. If BUFULL is set (any bits high) then the contents of the character buffer are current.

SAVSTK (ØØF6-ØØF7)

SAVSTK is used to save and restore the contents of the stack pointer.

TEMP (ØØF8)

TEMP is used for temporary storage during computation of paper tape checksums.

BYTECT (ØØF9) - BYTE COUNT

BYTECT contains the byte count during paper tape loading.

XHI (ØØFA)

XHI stores the high order byte of the index register.

XLO (ØØFB)

XLO stores the low order byte of the index register.

> NOTE
>
> XHI and XLO are also used to store the stack pointer when the Monitor is entered at a program breakpoint. This allows the N command to be used to examine the processor status. (See Section V for further information.)

SHIFT (ØØFC)

(Applies to Baudot version only)

SHIFT is set whenever the Baudot Teletype is in the upper case mode. SHIFT is clear whenever the Baudot Teletype is in the lower case mode.

SAVEX (ØØFD-OOFE)

(Applies to Baudot version only)

SAVEX is used by the Baudot output character routine to save and restore the contents of the index register.

BUFFER (ØØFF)

(Applies to Baudot version only)

BUFFER is the character buffer used by the Baudot input character routine.

Interrupt Vectors

The non-maskable interrupt vector points to location Ø1Ø4.

The maskable interrupt vector points to location Ø1ØØ in the ACIA version of the Monitor. See Section VIII for information concerning the maskable interrupt vector in the Baudot version.)

VII BAUDOT TELETYPE OPTION INFORMATION

The Baudot version of the PROM Monitor is a 512 byte, 2 PROM chip version of the Monitor, which contains the necessary software to support a Baudot Teletype (using bit banger I/O) and convert between Baudot (5 level code) and 7 bit ASCII.

> NOTE
>
> The Monitor supports Baudot Teletypes wired for half duplex only.

Baudot Input

Input from the Baudot Teletype is handled by using the maskable interrupt feature of the 68ØØ MPU. Therefore, the interrupt mask (bit 4 in the processor condition codes register) must be clear (=Ø) to enable input from the Baudot Teletype.

The maskable interrupt vector points to location FEØØ. When a maskable interrupt request is acknowledged, the Monitor checks to see if the the interrupt request was originated by the Baudot Teletype. If so, the character code is clocked in. If the request was originated by a device other than the Baudot Teletype, control is transferred to location Ø1Ø4.

The Baudot input routine converts from Baudot to ASCII and then stores the ASCII character into a 1 byte buffer. Therefore, one character type ahead is possible.

> NOTE
>
> The Baudot output character routine masks out interrupts and therefore a character typed while output is occurring is likely to be either misread or lost entirely.

Baudot < > ASCII Conversion

Figure 8-1 shows the Baudot keyboard which the Monitor's Baudot < > ASCII conversion is based on. The Baudot character set contains 55 (decimal) useable codes. For most computer applications this is an insufficient number of character codes, and therefore the PROM Monitor supports an extended Baudot character set. Table 8-2 shows the characters supported by the Baudot version of the Monitor.

The following is a list of conventions used for Baudot < > ASCII conversion.

1) Extended characters are formed by combining an & (the extend character) with another upper case character. For example, an "=" sign is represented by "&;" .

2) On output, if an ASCII code cannot be matched with a Baudot code, the extend character is printed, followed by a blank.

3) On input, control characters are formed by combining an & (the extend character) with the appropriate lower case character. For example, to send a control-A, the extend character must be typed, followed by a letters shift, followed by an A.

4) On input, any upper case extended character which is not explicitly defined in Table 8-2 is matched to the ASCII control character of its associated lower case. For example, an extended ":" (&:) is matched to a control-C.

5) On input, the codes for null, line feed, and carriage return are unaffected by case. For example, a lower case line feed, an upper case line feed, and an extended line feed are all matched to an ASCII 12 (octal).

6) The letters and figures shift codes are not matched to ASCII codes. They serve only to change the character case.

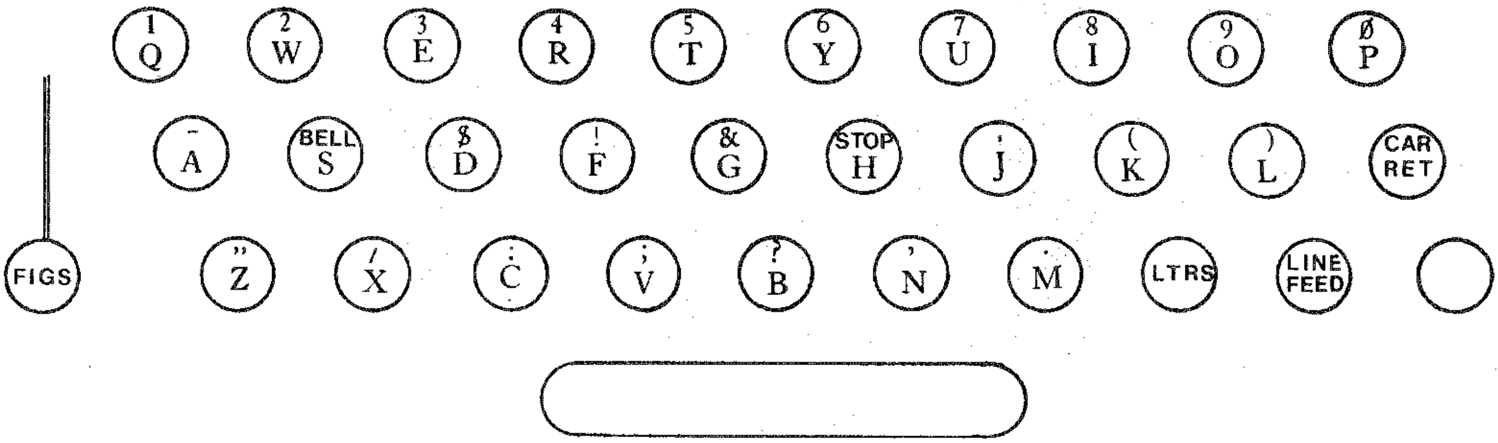


Figure 8-1. Baudot Keyboard

BAUDOT (OCTAL)	LOWER CASE	UPPER CASE	EXTENDED CASE
Ø	NULL	NULL	
1	E	3	
2	LINE FEED	LINE FEED	
3	A	-	_ SEE *2 BELOW
4	BLANK	BLANK	
5	S	CONTROL-G	
6	I	8	
7	U	7	
1Ø	CAR RETURN	CAR RETURN	
11	D	$	ESCAPE
12	R	4	
13	J	'	
14	N	,	@
15	F	!	^
16	C	:	
17	K	(	<
2Ø	T	5	
21	Z	"	#
22	L	)	>
23	W	2	
24	H	SEE *1 BELOW	
25	Y	6	
26	P	Ø	
27	Q	1	
3Ø	O	9	
31	B	?	%
32	G	&(EXT CHAR)	+
33	FIG SHIFT	FIG SHIFT	
34	M	.	*
35	X	/	
36	V	;	=
37	LTR SHIFT	LTR SHIFT	

*1 ON INPUT A STOP IS MATCHED TO A NULL. THERE IS NO ASCII CODE WHICH WILL OUTPUT A STOP.

*2 THIS CHARACTER IS PRINTED AS A BACK ARROW ON TELETYPE MODEL 33.

TABLE 8-2 Baudot <>ASCII Conversion

IX PROM MONITOR SOURCE LISTING (ACIA VERSION)

```
PAGE  001    PROM MON

00001                          NAM          PROM       MONITOR
00002                 **
00003                 ** ALTAIR 680B PROM MONITOR
00004                 ** ACIA VERSION 1.0
00005                 **
00006                          OPT          S          PRINT SYMBOL TABLE
00007                          OPT          PAGE       PAGINATED LISTING
00008           0100  MIVEC    EQU          $100
00009           0104  NMIVEC   EQU          $104
00010           F002  STRAPS   EQU          $F002
00011           0000  NOTERM   EQU          0
00012           F000  ACIACS   EQU          $F000
00013           F001  ACIADA   EQU          $F001
00014                 **
00015                 * MONITOR STACK AND FLAGS
00016                 **
00017 00F1                     ORG          $F1
00018 00F1 0001       STACK    RMB          1          BOTTOM OF MONITOR'S STACK
00019 00F2 0001       BRKADR   RMB          1          BREAKPOINT ADDRESS FLAG
00020 00F3 0001       ECHO     RMB          1          TTY ECHO FLAG
00021 00F4 0001       EXTFLG   RMB          1          EXTENDED CHARACTER FLAG
00022 00F5 0001       BUFULL   RMB          1          BUFFER FULL FLAG
00023 00F6 0002       SAVSTK   RMB          2          TEMP FOR STACK POINTER
00024 00F8 0001       TEMP     RMB          1          TEMPORARY STORAGE
00025 00F9 0001       BYTECT   RMB          1          BYTE CGUNT
00026 00FA 0001       XHI      RMB          1          XREG HIGH
00027 00FB 0001       XLOW     RMB          1          XREG LOW
00028 00FC 0001       SHIFT    RMB          1          BAUDOT SHIFT FLAG
00029 00FD 0002       SAVEX    RMB          2          TEMP FOR INDEX RG
00030 00FF 0001       BUFFER   RMB          1          BAUDOT CHARACTER BUFFER
00031                 **
00032                 * START OF PROM
00033                 *
00034 FF00                     ORG          $FF00
00035                 **
00036                 * INPUT ONE CHAR INTO A-REGISTER
00037                 * ECHO CHAR IF BIT 7 OF ECHO FLAG IS CLEAR
00038                 **
00040 FF00 8D 22      INCH     BSR          POLCAT     ACIA STATUS TO A REG
00041 FF02 24 FC               BCC          INCH       RECEIVE NOT READY
00042 FF04 C6 7F               LDA B        #$7F       MASK FOR PARITY REMOVAL
00043 FF06 D1 F3               CMP B        ECHO       CHECK ECHO FLAG
00044 FF08 F4 F001             AND B        ACIADA     GET CHARACTER
00045 FF0B 24 74               BCC          OUTCH      ECHO
00046 FF0D 39                  RTS                     NO ECHO
00048                 **
00049                 * THE FOLLOWING NOP LINES UP THE ENTRY
00050                 * POINTS TO POLCAT IN THE TWO VERSIONS
00051                 * OF THE MONITOR
00052                 **
00054 FF0E 01                  NOP
```

```
PAGE  ØØ2   PROM MON



                  **
                  * INPUT ONE HEX DIGIT INTO B REG
                  * RETURN TO CALLING PROGRAM IF
                  * CHARACTER RECEIVED IS A HEX
                  * DIGIT.  IF NOT HEX, GO TO CRLF
                  **
FFØF 8D EF        INHEX  BSR       INCH      GET A CHARACTER
FF11 CØ 3Ø               SUB B     #'Ø
FF13 2B 3C               BMI       C1        NOT HEX
FF15 C1 Ø9               CMP B     #$9
FF17 2F ØA               BLE       IN1HG     NOT HEX
FF19 C1 11               CMP B     #$11
FF1B 2B 34               BMI       C1        NOT HEX
FF1D C1 16               CMP B     #$16
FF1F 2E 3Ø               BGT       C1        NOT HEX
FF21 CØ Ø7               SUB B     #7        IT'S A LETTER-GET BCD
FF23 39           IN1HG  RTS                 RETURN
                  * *
                  * POLE FOR CHARACTER
                  * SETS CARRY IF CHARACTER IS IN BUFFER
                  * CLOBBERS B REG
                  **
FF24 F6 FØØØ      POLCAT LDA B     ACIACS    ACIA STATUS TO B
FF27 57                  ASR B               ROTATE RDRF BIT INTO CARRY
FF28 39                  RTS                 RETURN
                  **
                  * LOAD PAPER TAPE
                  * LOAD ONLY S1 TYPE RECORDS
                  * TERMINATE ON S9 OR CHECKSUM ERROR
                  **
FF29 8D D5        LOAD   BSR       INCH      READ FRAME
FF2B CØ 53               SUB B     #'S
FF2D 26 FA               BNE       LOAD      FIRST CHAR NOT (S)
FF2F 8D CF               BSR       INCH      READ FRAME
FF31 C1 39               CMP B     #'9
FF33 27 1C               BEQ       C1        S9 END OF FILE
FF35 C1 31               CMP B     #'1
FF37 26 FØ               BNE       LOAD      SECOND CHAR NOT (1)
FF39 4F                  CLR A               ZERO THE CHECKSUM
FF3A 8D 17               BSR       BYTE      READ BYTE
FF3C CØ Ø2               SUB B     #2
FF3E D7 F9               STA B     BYTECT    BYTE COUNT
FF4Ø 8D 2Ø               BSR       BADDR     GET ADDRESS OF BLOCK
FF42 8D ØF        LOAD11 BSR       BYTE      GET DATA BYTE
FF44 7A ØØF9             DEC       BYTECT    DECREMENT BYTE COUNT
FF47 27 Ø5               BEQ       LOAD15    DONE WITH THIS BLOCK
FF49 E7 ØØ               STA B     X         STORE DATA
FF4B Ø8                  INX                 BUMP POINTER
FF4C 2Ø F4               BRA       LOAD11    GO BACK FOR MORE
FF4E 4C           LOAD15 INC A               INCREMENT CHECKSUM
FF4F 27 D8        LLOAD  BEQ       LOAD      ALL OK - IT'S ZERO
FF51 2Ø 58        C1     BRA       CRLF      CHECKSUM ERROR - QUIT
```

```
ØØ117                  **
ØØ118                  * READ BYTE (2 HEX DIGITS)
ØØ119                  * INTO B REG
ØØ12Ø                  * A IS USED FOR PAPER TAPE CHECKSUM
ØØ121                  **
ØØ122 FF53 8D BA       BYTE   BSR        INHEX     GET FIRST HEX DIG
ØØ123 FF55 58                 ASL B                SHIFT TO HIGH ORDER 4 BITS
ØØ124 FF56 58                 ASL B
ØØ125 FF57 58                 ASL B
ØØ126 FF58 58                 ASL B
ØØ127 FF59 1B                 ABA                  ADD TO CHEKSUM
ØØ128 FF5A D7 F8              STA B      TEMP      STORE DIGIT
ØØ129 FF5C 8D B1              BSR        INHEX     GET 2ND HEX DIG
ØØ13Ø FF5E 1B                 ABA                  ADD TO CHECKSUM
ØØ131 FF5F DB F8              ADD B      TEMP      COMBINE DIGITS TO GET BYTE
ØØ132 FF61 39                 RTS                  RETURN
ØØ133                  **
ØØ134                  * READ 16 BIT ADDRESS INTO X
ØØ135                  * STORE SAME ADDRESS IN XHI & XLO
ØØ136                  * CLOBBERS B REG
ØØ137                  **
ØØ138 FF62 8D EF       BADDR  BSR        BYTE      GET HIGH ORDER ADDRESS
ØØ139 FF64 D7 FA              STA B      XHI       STORE IT
ØØ14Ø FF66 8D EB              BSR        BYTE      GET LOW ORDER ADDRESS
ØØ141 FF68 D7 FB              STA B      XLOW      STORE IT
ØØ142 FF6A DE FA              LDX        XHI       LOAD X WITH ADDRESS BUILT
ØØ143 FF6C 39                 RTS                  RETURN
ØØ147                  **
ØØ148                  * PRINT BYTE IN A REG
ØØ149                  * CLOBBERS B REG
ØØ15Ø                  **
ØØ151 FF6D 16          OUT2H  TAB                  COPY BYTE TO B
ØØ152 FF6E 54                 LSR B                SHIFT TO RIGHT
ØØ153 FF6F 54                 LSR B
ØØ154 FF7Ø 54                 LSR B
ØØ155 FF71 54                 LSR B
ØØ156 FF72 8D Ø1              BSR        OUTHR     OUTPUT FIRST DIGIT
ØØ157 FF74 16                 TAB                  BYTE INTO B AGAIN
ØØ16Ø FF75 C4 ØF       OUTHR  AND B      #$F       GET RID OF LEFT DIG
ØØ161 FF77 CB 3Ø              ADD B      #$3Ø      GET ASCII
ØØ162 FF79 C1 39              CMP B      #$39
ØØ163 FF7B 23 Ø4              BLS        OUTCH
ØØ164 FF7D CB Ø7              ADD B      #7        IF IT'S A LETTER ADD 7
ØØ165 FF7F Ø1                 NOP                  LINE UP OUTCH ENTRY POINTS
ØØ166 FF8Ø Ø1                 NOP
```

```
ØØ167 FF81 8C        OUTCH  FCB       $8C      USE CPX SKIP TRICK
ØØ168 FF82 C6 2Ø     OUTS   LDA B     #$2Ø     OUTS PRINTS A SPACE
ØØ171                **
ØØ172                * OUTCH OUTPUTS CHARACTER IN B
ØØ173                **
ØØ174 FF84 37               PSH B              SAVE CHAR
ØØ175 FF85 8D 9D     OUTC1  BSR       POLCAT   ACIA STATUS TO B REG
ØØ176 FF87 57               ASR B
ØØ177 FF88 24 FB            BCC       OUTC1    XMIT NOT READY
ØØ178 FF8A 33               PUL B              CHAR BACK TO B REG
ØØ179 FF8B F7 FØØ1          STA B     ACIADA   OUTPUT CHARACTER
ØØ18Ø FF8E 39               RTS
ØØ183                **
ØØ184                * EXAMINE AND DEPOSIT NEXT
ØØ185                * USES CONTENTS OF XHI & XLO AS POINTER
ØØ186                **
ØØ187 FF8F DE FA     NCHANG LDX       XHI      INCREMENT POINTER
ØØ188 FF91 Ø8               INX
ØØ189 FF92 DF FA            STX       XHI
ØØ19Ø FF94 96 FA            LDA A     XHI
ØØ191 FF96 8D D5            BSR       OUT2H    PRINT OUT ADDRESS
ØØ192 FF98 96 FB            LDA A     XLOW
ØØ193 FF9A 8D D1            BSR       OUT2H
ØØ194 FF9C 8C               FCB       $8C      USE CPX SKIP TRICK
ØØ195                **
ØØ196                * EXAMINE & DEPOSIT
ØØ197                **
ØØ198 FF9D 8D C3     CHANGE BSR       BADDR    BUILD ADDRESS
ØØ199 FF9F 8D E1            BSR       OUTS     PRINT SPACE
ØØ2ØØ FFA1 A6 ØØ            LDA A     X        BYTE INTO A
ØØ2Ø1 FFA3 8D C8            BSR       OUT2H    PRINT BYTE
ØØ2Ø2 FFA5 8D DB            BSR       OUTS     PRINT SPACE
ØØ2Ø3 FFA7 8D AA            BSR       BYTE     GET NEW BYTE
ØØ2Ø4 FFA9 E7 ØØ            STA B     X        STORE NEW BYTE
ØØ2Ø6                **
ØØ2Ø7                * COMMAND DECODING SECTION
ØØ2Ø8                **
ØØ2Ø9 FFAB 9E F6     CRLF   LDS       SAVSTK
ØØ21Ø FFAD C6 ØD            LDA B     #$D      CARRIAGE RETURN
ØØ211 FFAF 8D DØ            BSR       OUTCH
ØØ212 FFB1 C6 ØA            LDA B     #$A      LINE FEED
ØØ213 FFB3 8D CC            BSR       OUTCH
ØØ214 FFB5 C6 2E            LDA B     #'.      PROMPT CHARACTER
ØØ215 FFB7 8D C8            BSR       OUTCH
ØØ216 FFB9 BD FFØØ          JSR       INCH     READ CHARACTER
ØØ217 FFBC 17               TBA                MAKE A COPY
ØØ218 FFBD 8D C3            BSR       OUTS     PRINT SPACE
ØØ219 FFBF 81 4C            CMP A     #'L
ØØ22Ø FFC1 27 8C            BEQ       LLOAD    LOAD PAPER TAPE
```

```
00221 FFC3 81 4A          CMP A     #'J
00222 FFC5 26 04          BNE       NOTJ
00223 FFC7 8D 99          BSR       BADDR     GET ADDRESS TO JUMP TO
00224 FFC9 6E 00          JMP       X         JUMP TO IT
00225 FFCB 81 4D   NOTJ   CMP A     #'M
00226 FFCD 27 CE          BEQ       CHANGE    EXAMINE & DEPOSIT
00227 FFCF 81 4E          CMP A     #'N
00228 FFD1 27 BC          BEQ       NCHANG    E & D NEXT
00229 FFD3 81 50          CMP A     #'P
00230 FFD5 26 D4          BNE       CRLF
00231 FFD7 3B             RTI                 PROCEDE FROM BREAKPOINT
00233              **
00234              * RESET ENTRY POINT
00235              **
00236 FFD8 8E 00F3 RESET  LDS       #ECHO     INITIALIZE STACK POINTER
00237 FFDB C6 03          LDA B     #3        INIT ECHO AND BRKADR FLAGS
00238 FFDD 37             PSH B
00239 FFDE 37             PSH B
00240 FFDF F7 F000        STA B     ACIACS    MASTER RESET ACIA
00241 FFE2 F6 F002        LDA B     STRAPS    LOOK AT STRAPS
00242 FFE5 2B 19          BMI       NOTERM    NO TERM - JUMP TO 0
00243 FFE7 C4 04          AND B     #4        GET # OF STOP BITS
00244 FFE9 CA D1          ORA B     #$D1
00245 FFEB F7 F000        STA B     ACIACS    INIT ACIA PORT
00246              **
00247              * SOFTWARE INTERRUPT ENTRY POINT
00248              **
00249 FFEE 9F F6   INTRPT STS       SAVSTK    SAVE STACK POINTER
00250 FFF0 9F FA          STS       XHI       SAVE SP FOR N COMMAND
00251 FFF2 D6 F2          LDA B     BRKADR    IF BIT 7 OF BRKADR IS SET
00252 FFF4 2B 0A          BMI       NOTERM    JUMP TO 0
00253 FFF6 20 B3          BRA       CRLF      GOTO COMMAND DECODER
00256              **
00257              * NOW COME THE INTERRUPT VECTORS
00258              **
00260 FFF8                ORG       $FFF8
00263 FFF8 0100           FDB       MIVEC     MI VECTOR
00264 FFFA FFEE           FDB       INTRPT    SWI VECTOR
00265 FFFC 0104           FDB       NMIVEC    NMI VECTOR
00266 FFFE FFD8           FDB       RESET     RESET VECTOR
00268                     END
```

```
PAGE  ØØ6    PROM MON



MIVEC   Ø1ØØ
NMIVEC  Ø1Ø4
STRAPS  FØØ2
NOTERM  ØØØØ
ACIACS  FØØØ
ACIADA  FØØ1
STACK   ØØF1
BRKADR  ØØF2
ECHO    ØØF3
EXTFLG  ØØF4
BUFULL  ØØF5
SAVSTK  ØØF6
TEMP    ØØF8
BYTECT  ØØF9
XHI     ØØFA
XLOW    ØØFB
SHIFT   ØØFC
SAVEX   ØØFD
BUFFER  ØØFF
INCH    FFØØ
INHEX   FFØF
IN1HG   FF23
POLCAT  FF24
LOAD    FF29
LOAD11  FF42
LOAD15  FF4E
LLOAD   FF4F
C1      FF51
BYTE    FF53
BADDR   FF62
OUT2H   FF6D
OUTHR   FF75
OUTCH   FF81
OUTS    FF82
OUTC1   FF85
NCHANG  FF8F
CHANGE  FF9D
CRLF    FFAB
NOTJ    FFCB
RESET   FFD8
INTRPT  FFEE


TOTAL ERRORS ØØØØØ
```

PAGE ØØ1 PROM MON X PROM MONITOR SOURCE LISTING (BAUDOT VERSION)

```
                       NAM       PROM      MONITOR
               **
               ** ALTAIR 68ØB PROM MONITOR
               ** BAUDOT VERSION 1.Ø
               **
                       OPT       S         PRINT SYMBOL TABLE
                       OPT       PAGE      PAGINATED LISTING
     FEØØ      MIVEC   EQU       $FEØØ
     Ø1Ø4      NMIVEC  EQU       $1Ø4
     Ø1ØØ      CRAZY   EQU       $1ØØ
     FØØ2      STRAPS  EQU       $FØØ2
     ØØØØ      NOTERM  EQU       Ø
     FØØØ      ACIACS  EQU       $FØØØ
     FØØ1      ACIADA  EQU       $FØØ1
ØØF1                   ORG       $F1
ØØF1 ØØØ1      STACK   RMB       1         BOTTOM OF MONITOR'S STACK
ØØF2 ØØØ1      BRKADR  RMB       1         BREAKPOINT ADDRESS FLAG
ØØF3 ØØØ1      ECHO    RMB       1         TTY ECHO FLAG
ØØF4 ØØØ1      EXTFLG  RMB       1         EXTENDED CHARACTER FLAG
ØØF5 ØØØ1      BUFULL  RMB       1         BUFFER FULL FLAG
ØØF6 ØØØ2      SAVSTK  RMB       2         TEMP FOR STACK POINTER
ØØF8 ØØØ1      TEMP    RMB       1         TEMPORARY STORAGE
ØØF9 ØØØ1      BYTECT  RMB       1         BYTE COUNT
ØØFA ØØØ1      XHI     RMB       1         XREG HIGH
ØØFB ØØØ1      XLOW    RMB       1         XREG LOW
ØØFC ØØØ1      SHIFT   RMB       1         BAUDOT SHIFT FLAG
ØØFD ØØØ2      SAVEX   RMB       2         TEMP FOR INDEX REG
ØØFF ØØØ1      BUFFER  RMB       1         BAUDOT CHARACTER BUFFER
               **
               * START OF PROM
               **
FEØØ                   ORG       $FEØØ
               **
               * MASKABLE INTERRRUPT VECTOR POINTS TO GET
               **
FEØØ 86 4Ø     GET     LDA A     #$4Ø      THIS BIT ROTATES INTO CARRY
               *                             TO SIGNAL STOP BIT ARRIVAL
FEØ2 F6 FØØ2           LDA B     STRAPS    IF BIT Ø OF FØØ2 IS LOW
FEØ5 56                ROR B               THEN INTERRUPT CAME FROM BAUDOT
FEØ6 24 21             BCC       GETBIT    SO CLOCK IN CHAR CODE
FEØ8 7E                FCB       $7E       IF BIT Ø IS HIGH
FEØ9 Ø1                FCB       ØØ1       JUMP TO Ø1ØØ (HEX)
               **
               * THIS IS THE UPPPER CASE CONVERSION TABLE
               **
FEØA ØØ        UPCAS   FCB       Ø         NULL
FEØB 33                FCC       /3/
FEØC ØA                FCB       $A        LINE FEED
FEØD 2D                FCC       /-/
FEØE 2Ø                FCB       $2Ø       BLANK
FEØF Ø7                FCB       7         CONTROL G (BELL)
FE1Ø 38                FCC       /87/
FE11 37
FE12 ØD                FCB       $D        CARRIAGE RETURN
```

```
PAGE  ØØ2    PROM MON


ØØØ54 FE13 24              FCC        /$4'/
      FE14 34
      FE15 27
ØØØ55 FE16 2C              FCC        /,/
ØØØ56 FE17 21              FCC        /!:(5/
      FE18 3A
      FE19 28
      FE1A 35
ØØØ57 FE1B 22              FCC        /"/
ØØØ58 FE1C 29              FCC        /)/
ØØØ59 FE1D 32              FCC        /2/
ØØØ6Ø FE1E ØØ              FCB        Ø         SLOT FOR STOP
ØØØ61 FE1F 36              FCC        /6Ø19?/
      FE2Ø 3Ø
      FE21 31
      FE22 39
      FE23 3F
ØØØ62 FE24 ØØ              FCB        Ø         SLOT FOR &
ØØØ63 FE25 ØØ              FCB        Ø         SLOT FOR FIGURES SHIFT
ØØØ64 FE26 2E              FCC        /./
ØØØ65 FE27 2F              FCC        !/!
ØØØ66 FE28 3B              FCC        /;/
ØØØ67                **
ØØØ68                * END OF UPPER CASE TABLE
ØØØ69                **
ØØØ7Ø FE29 8D 3D     GETBIT BSR       WAIT11    WAIT HALF A BIT TIME
ØØØ71 FE2B F6 FØØ2          LDA B     STRAPS
ØØØ72 FE2E 56               ROR B               PUT DATA BIT INTO CARRY
ØØØ73 FE2F 8D 37            BSR       WAIT11    FINISH UP BIT TIME
ØØØ74 FE31 46               ROR A               COLLECT CODE IN A
ØØØ75 FE32 24 F5            BCC       GETBIT    IF MORE TO COME GO GET EM
ØØØ76 FE34 48               ASL A               GET RID OF STOP BIT
ØØØ77 FE35 44               LSR A               RIGHT JUSTIFY CODE
ØØØ78 FE36 44               LSR A
ØØØ79 FE37 44               LSR A
ØØØ8Ø                **
ØØØ81                * WE HAVE THE CODE IN A NOW
ØØØ82                **
ØØØ83 FE38 81 1B            CMP A     #$1B      IF IT'S AN UPSHIFT
ØØØ84 FE3A 26 Ø3            BNE       NTUP      SET THE SHIFT FLAG
ØØØ85 FE3C D7 FC     CLRSF  STA B     SHIFT     AND RETURN FROM INTERRUPT
ØØØ86 FE3E 3B               RTI
ØØØ88 FE3F 5F        NTUP   CLR B
ØØØ89 FE4Ø 81 1F            CMP A     #$1F      IF IT'S A DOWNSHIFT
ØØØ9Ø FE42 27 F8            BEQ       CLRSF     CLEAR THE SHIFT FLAG
ØØØ92 FE44 D1 F4            CMP B     EXTFLG    IF EXTENDED CHARACTER
ØØØ93 FE46 2B 31            BMI       EXTCAR    IS SET GO TO EXT
ØØØ94                *                          CHARACTER SEARCH
ØØØ95 FE48 CE FEE2          LDX       #LOWCAS-2
ØØØ96                * SET POINTER TO LOWER CASE
ØØØ97 FE4B D1 FC            CMP B     SHIFT     IF SHIFT FLAG IS SET
ØØØ98 FE4D 2B 2Ø            BMI       UPCAR     THEN INDEX INTO UPPER CASE TABLE
```

```
PAGE  ØØ3   PROM MON


ØØØ99 FE4F Ø8        ADDAX   INX                    ADD A REG TO X REG
ØØ1ØØ FE5Ø 4A                DEC A
ØØ1Ø1 FE51 2A FC             BPL        ADDAX
ØØ1Ø2 FE53 53        DONE    COM B                  FORM MASK
ØØ1Ø3 FE54 D7 F5             STA B      BUFULL      SET BUFFER FULL FLAG
ØØ1Ø4 FE56 E4 Ø1             AND B      1,X         MASK OFF LOW 6 OR ALL 8
ØØ1Ø5 FE58 D7 FF             STA B      BUFFER      STORE CHAR INTO BUFFER
ØØ1Ø6 FE5A 3B                RTI                    RETURN FROM THE INTERRUPT
ØØ1Ø7                **
ØØ1Ø8                * PUT CLOCKS OUT THE CHARACTER CODE
ØØ1Ø9                **
ØØ11Ø FE5B 48        PUT     ASL A                  ROTATE IN START BIT
ØØ111 FE5C 8A 4Ø             ORA A      #$4Ø        OR IN STOP BIT
ØØ112 FE5E B7 FØØ2 NXTBIT    STA A      $FØØ2       SEND A BIT
ØØ113 FE61 8D Ø5             BSR        WAIT11
ØØ114 FE63 8D Ø3             BSR        WAIT11      WAIT AROUND FOR 22 MIL SECS
ØØ115 FE65 44                LSR A                  SHIFT TO NEXT BIT
ØØ116 FE66 26 F6             BNE        NXTBIT      IF MORE TO SEND THEN DO SO
ØØ118 FE68 CE Ø2AF WAIT11    LDX        #687        11 MIL SEC DELAY
ØØ119 FE6B Ø9        WAIT    DEX
ØØ12Ø FE6C 26 FD             BNE        WAIT
ØØ121 FE6E 39                RTS
ØØ123 FE6F CE FEØ8 UPCAR     LDX        #UPCAS-2 POINT TO UPPER CASE TABLE
ØØ124 FE72 81 1A             CMP A      #$1A        IF IT'S THE EXTEND CHAR THEN
ØØ125 FE74 26 D9             BNE        ADDAX       SET THE EXTENDED CHAR FLAG
ØØ126 FE76 97 F4             STA A      EXTFLG      AND RETURN FROM INTERRUPT
ØØ127 FE78 3B                RTI
ØØ129 FE79 CE FFEØ EXTCAR    LDX        #EXTEND-2 POINT TO EXTENDED CHAR TABLE
ØØ13Ø FE7C D7 F4             STA B      EXTFLG      CLEAR THE EXTEND FLAG
ØØ131 FE7E Ø8        CHKNXT  INX
ØØ132 FE7F Ø8                INX
ØØ133 FE8Ø A1 ØØ             CMP A      X           SEARCH THE EXTENDED CHAR TABLE
ØØ134 FE82 27 CF             BEQ        DONE        IF MATCH FOUND THEN WE ARE DONE
ØØ135 FE84 6D ØØ             TST        X           IF MINUS ENCOUNTERED THEN CODE NOT
ØØ136 FE86 2A F6             BPL        CHKNXT      IN TABLE SO MAKE INTO CONTROL CHAR
ØØ137 FE88 CE FEE2           LDX        #LOWCAS-2 BY TAKING LOWER CASE ASCII AND
ØØ138 FE8B C6 CØ             LDA B      #$CØ        SETTING MASK TO GET RIG OF HI
ØØ139 FE8D 2Ø CØ             BRA        ADDAX       ORDER 2 BITS
ØØ14Ø FE8F 96 FC     CHKUP   LDA A      SHIFT       BEFORE CHECKING UPPPER CASE TABLE
ØØ141 FE91 26 Ø6             BNE        OKUP        CHECK THE SHIFT FLAG
ØØ142 FE93 86 1B             LDA A      #$1B        SEND OUT FIGURES SHIFT AND SET
ØØ143 FE95 97 FC             STA A      SHIFT       SHIFT FLAG AS NECESSARY
ØØ144 FE97 8D C2             BSR        PUT
ØØ145 FE99 CE FEØA OKUP      LDX        #UPCAS      SET POINTER TO UPPER CASE TABLE
ØØ146 FE9C 8D 39             BSR        SEARCH      CALL SEARCH ROUTINE
ØØ147 FE9E 2A 2F             BPL        RESTR       IF POSITIVE, SEARCH WAS SUCCESSFUL
ØØ148 FEAØ 86 1A             LDA A      #$1A        SEARCH FAILED SO OUTPUT EXTEND
ØØ149 FEA2 8D B7             BSR        PUT         CHARACTER
ØØ15Ø FEA4 CE FFEØ           LDX        #EXTEND-2
ØØ151 FEA7 E1 Ø1     NXT     CMP B      1,X         SEARCH THROUGH EXTENDED CHAR
ØØ152 FEA9 27 24             BEQ        RESTR       TABLE
```

```
PAGE  004    PROM MON


00153 FEAB 08                INX
00154 FEAC 08                INX                 BUMP POINTER TWICE
00155 FEAD A6 00             LDA A     X         LOAD THE BAUDOT CODE INTO B
00156 FEAF 2A F6             BPL       NXT       IF MINUS - END OF TABLE
00157 FEB1 C6 20             LDA B     #$20      NO MATCH FOUND - OUTPUT BLANK
00158 FEB3 8D 04             BSR       BOUT2
00159 FEB5 20 1A             BRA       REST2
00160                **
00161                * BOUTCH IS THE OUTPUT CHARACTER ROUTINE
00162                **
00163 FEB7 DF FD     BOUTCH  STX       SAVEX     SAVE X,A,&B
00164 FEB9 0F        BOUT2   SEI                 DISENABLE INTERRUPTS
00165 FEBA 36                PSH A
00166 FEBB 37                PSH B
00167 FEBC CE FEE4           LDX       #LOWCAS   SET POINTER TO LOWER CASE
00168 FEBF 8D 16             BSR       SEARCH    TABLE AND CALL SEARCH ROUTINE
00169 FEC1 2B CC             BMI       CHKUP     IF MINUS, THEN SEARCH FAILED
00170 FEC3 D6 FC             LDA B     SHIFT     CHECK THE SHIFT FLAG
00171 FEC5 27 08             BEQ       RESTR
00172 FEC7 36                PSH A               IF FLAG IS SET THEN SEND OUT
00173 FEC8 86 1F             LDA A     #$1F      LETTERS SHIFT AND CLEAR FLAG
00174 FECA 8D 8F             BSR       PUT
00175 FECC 97 FC             STA A     SHIFT     A IS CLEAR ON RETURN FROM PUT
00176 FECE 32                PUL A
00177 FECF 8D 8A     RESTR   BSR       PUT
00178 FED1 33        REST2   PUL B               RESTORE B
00179 FED2 32                PUL A               RESTORE A REG
00180 FED3 DE FD             LDX       SAVEX     RESTORE X REG
00181 FED5 0E                CLI                 ENABLE INTERRUPTS
00182 FED6 39        RET     RTS                 RETURN
00183                **
00184                * SUBROUTINE TO SEARCH CONVERSION TABLES
00185                * RETURNS WITH CODE IN A IF FOUND
00186                * RETURNS WITH N BIT SET IF NOT FOUND
00187                **
00188 FED7 4F        SEARCH  CLR A
00189 FED8 6D 00     NXTCHK  TST       X
00190 FEDA 2B FA             BMI       RET       IF MINUS - END OF TABLE
00191 FEDC E1 00             CMP B     X
00192 FEDE 27 F6             BEQ       RET       MATCH - RETURN
00193 FEE0 08                INX                 INCREMENT POINTER
00194 FEE1 4C                INC A               INCREMENT OUTPUT CODE
00195 FEE2 20 F4             BRA       NXTCHK    CONTINUE SEARCH
00196                **
00197                * LOWER CASE CONVERSION TABLE
00198                **
00199 FEE4 00        LOWCAS  FCB       0         NULL
00200 FEE5 45                FCC       /E/
00201 FEE6 0A                FCB       $A        LINE FEED
00202 FEE7 41                FCC       /A/
00203 FEE8 20                FCB       $20       BLANK
00204 FEE9 53                FCC       /SIU/
      FEEA 49
      FEEB 55
```

```
PAGE  ØØ5   PROM MON


FEEC ØD                FCB       $D        CARRIAGE RETURN
FEED 44                FCC       /DRJNFCKTZLWHYPQOBG/
FEEE 52
FEEF 4A
FEFØ 4E
FEF1 46
FEF2 43
FEF3 4B
FEF4 54
FEF5 5A
FEF6 4C
FEF7 57
FEF8 48
FEF9 59
FEFA 5Ø
FEFB 51
FEFC 4F
FEFD 42
FEFE 47
FEFF ØØ                FCB       Ø         SLOT FOR FIGURES SHIFT
                **
                * INCH ENTRY POINT MUST BE AT START OF SECOND PROM
                **
FFØØ 4D         INCH   FCC       /MXV/
FFØ1 58
FFØ2 56
FFØ3 8D 1F      HANG   BSR       POLCAT    IF BUFFER IS EMPTY
FFØ5 24 FC             BCC       HANG      HANG AROUND FOR INTERRUPT
FFØ7 7F ØØF5           CLR       BUFULL    CLEAR THE BUFFER FULL FLAG
FFØA D6 FF             LDA B     BUFFER    PUT CHAR INTO B
FFØC 39                RTS                 RETURN
                **
                * INPUT ONE HEX DIGIT INTO B REG
                * RETURN TO CALLING PROGRAM IF
                * CHARACTER RECEIVED IS A HEX
                * DIGIT.  IF NOT HEX, GO TO CRLF
                **
FFØD 8D F1      INHEX  BSR       INCH      GET A CHARACTER
FFØF CØ 3Ø             SUB B     #'Ø
FF11 2B 3D             BMI       C1        NOT HEX
FF13 C1 Ø9             CMP B     #$9
FF15 2F ØA             BLE       IN1HG     NOT HEX
FF17 C1 11             CMP B     #$11
FF19 2B 35             BMI       C1        NOT HEX
FF1B C1 16             CMP B     #$16
FF1D 2E 31             BGT       C1        NOT HEX
FF1F CØ Ø7             SUB B     #7        IT'S A LETTER-GET BCD
FF21 39         IN1HG  RTS                 RETURN
                **
                * THIS HELPS LINE UP ENTRY POINTS
                **
FF22 2Ø 93      BBOUTC BRA       BOUTCH
                **
```

```
ØØ240                    * POLE FOR CHARACTER
ØØ241                    * SET CARRY IF CHAR IN BUFFER IS CURRENT
ØØ242                    * CLEAR CARRY IF NOT CURRENT
ØØ243                    **
ØØ244 FF24 D6 F5         POLCAT LDA B     BUFULL
ØØ245 FF26 57                   ASR B
ØØ246 FF27 39                   RTS
ØØ247                    **
ØØ248                    * LOAD PAPER TAPE
ØØ249                    * LOAD ONLY S1 TYPE RECORDS
ØØ25Ø                    * TERMINATE ON S9 OR CHECKSUM ERROR
ØØ251                    **
ØØ252 FF28 8D D6         LOAD   BSR       INCH      READ FRAME
ØØ253 FF2A CØ 53                SUB B     #'S
ØØ254 FF2C 26 FA                BNE       LOAD      FIRST CHAR NOT (S)
ØØ255 FF2E 8D DØ                BSR       INCH      READ FRAME
ØØ256 FF3Ø C1 39                CMP B     #'9
ØØ257 FF32 27 1C                BEQ       C1        S9 END OF FILE
ØØ258 FF34 C1 31                CMP B     #'1
ØØ259 FF36 26 FØ                BNE       LOAD      SECOND CHAR NOT (1)
ØØ26Ø FF38 4F                   CLR A               ZERO THE CHECKSUM
ØØ261 FF39 8D 17                BSR       BYTE      READ BYTE
ØØ262 FF3B CØ Ø2                SUB B     #2
ØØ263 FF3D D7 F9                STA B     BYTECT    BYTE COUNT
ØØ264 FF3F 8D 2Ø                BSR       BADDR     GET ADDRESS OF BLOCK
ØØ265 FF41 8D ØF         LOAD11 BSR       BYTE      GET DATA BYTE
ØØ266 FF43 7A ØØF9              DEC       BYTECT    DECREMENT BYTE COUNT
ØØ267 FF46 27 Ø5                BEQ       LOAD15    DONE WITH THIS BLOCK
ØØ268 FF48 E7 ØØ                STA B     X         STORE DATA
ØØ269 FF4A Ø8                   INX                 BUMP POINTER
ØØ27Ø FF4B 2Ø F4                BRA       LOAD11    GO BACK FOR MORE
ØØ271 FF4D 4C            LOAD15 INC A               INCREMENT CHECKSUM
ØØ272 FF4E 27 D8         LLOAD  BEQ       LOAD      ALL OK - IT'S ZERO
ØØ273 FF5Ø 2Ø 4D         C1     BRA       CRLF      CHECKSUM ERROR - QUIT
ØØ274                    **
ØØ275                    * READ BYTE (2 HEX DIGITS)
ØØ276                    * INTO B REG
ØØ277                    * A IS USED FOR PAPER TAPE CHECKSUM
ØØ278                    **
ØØ279 FF52 8D B9         BYTE   BSR       INHEX     GET FIRST HEX DIG
ØØ28Ø FF54 58                   ASL B               SHIFT TO HIGH ORDER 4 BITS
ØØ281 FF55 58                   ASL B
ØØ282 FF56 58                   ASL B
ØØ283 FF57 58                   ASL B
ØØ284 FF58 1B                   ABA                 ADD TO CHEKSUM
ØØ285 FF59 D7 F8                STA B     TEMP      STORE DIGIT
ØØ286 FF5B 8D BØ                BSR       INHEX     GET 2ND HEX DIG
ØØ287 FF5D 1B                   ABA                 ADD TO CHECKSUM
ØØ288 FF5E DB F8                ADD B     TEMP      COMBINE DIGITS TO GET BYTE
ØØ289 FF6Ø 39                   RTS                 RETURN
ØØ29Ø                    **
ØØ291                    * READ 16 BIT ADDRESS INTO X
ØØ292                    * STORE SAME ADDRESS IN XHI & XLO
ØØ293                    * CLOBBERS B REG
```

```
00294                  **
00295 FF61 8D EF       BADDR  BSR        BYTE       GET HIGH ORDER ADDRESS
00296 FF63 D7 FA              STA B      XHI        STORE IT
00297 FF65 8D EB              BSR        BYTE       GET LOW ORDER ADDRESS
00298 FF67 D7 FB              STA B      XLOW       STORE IT
00299 FF69 DE FA              LDX        XHI        LOAD X WITH ADDRESS BUILT
00300 FF6B 39                 RTS                   RETURN
00301                  **
00302                  * PRINT BYTE IN A REG
00303                  * CLOBBERS B REG
00304                  **
00305 FF6C 16          OUT2H  TAB                   COPY BYTE TO B
00306 FF6D 54                 LSR B                 SHIFT TO RIGHT
00307 FF6E 54                 LSR B
00308 FF6F 54                 LSR B
00309 FF70 54                 LSR B
00310 FF71 8D 01              BSR        OUTHR      OUTPUT FIRST DIGIT
00311 FF73 16                 TAB                   BYTE INTO B AGAIN
00312 FF74 C4 0F       OUTHR  AND B      #$F        GET RID OF LEFT DIG
00313 FF76 CB 30              ADD B      #$30       GET ASCII
00314 FF78 C1 39              CMP B      #$39
00315 FF7A 23 05              BLS        OUTCH
00316 FF7C CB 07              ADD B      #7         IF IT'S A LETTER ADD 7
00317 FF7E 8C                 FCB        $8C
00318 FF7F C6 20       OUTS   LDA B      #$20       OUTS PRINTS A SPACE
00319                  **
00320                  * OUTCH OUTPUTS CHAR IN B
00321                  **
00322 FF81 20 9F       OUTCH  BRA        BBOUTC
00323                  **
00324                  * EXAMINE AND DEPOSIT NEXT
00325                  * USES CONTENTS OF XHI & XLO AS POINTER
00326                  **
00327 FF83 DE FA       NCHANG LDX        XHI        INCREMENT POINTER
00328 FF85 08                 INX
00329 FF86 DF FA              STX        XHI
00330 FF88 96 FA              LDA A      XHI
00331 FF8A 8D E0              BSR        OUT2H      PRINT OUT ADDRESS
00332 FF8C 96 FB              LDA A      XLOW
00333 FF8E 8D DC              BSR        OUT2H
00334 FF90 8C                 FCB        $8C
00335                  **
00336                  * EXAMINE & DEPOSIT
00337                  **
00338 FF91 8D CE       CHANGE BSR        BADDR      BUILD ADDRESS
00339 FF93 8D EA              BSR        OUTS       PRINT SPACE
00340 FF95 A6 00              LDA A      X          BYTE INTO A
00341 FF97 8D D3              BSR        OUT2H      PRINT BYTE
00342 FF99 8D E4              BSR        OUTS       PRINT SPACE
00343 FF9B 8D B5              BSR        BYTE       GET NEW BYTE
00344 FF9D E7 00              STA B      X          STORE NEW BYTE
00345                  **
00346                  * COMMAND DECODING SECTION
00347                  **
```

```
PAGE  ØØ8    PROM MON


ØØ348 FF9F 9E F6    CRLF   LDS         SAVSTK
ØØ349 FFA1 C6 ØD           LDA B       #$D       CARRIAGE RETURN
ØØ35Ø FFA3 8D DC           BSR         OUTCH
ØØ351 FFA5 C6 ØA           LDA B       #$A       LINE FEED
ØØ352 FFA7 8D D8           BSR         OUTCH
ØØ353 FFA9 C6 2E           LDA B       #'.       PROMPT CHARACTER
ØØ354 FFAB 8D D4           BSR         OUTCH
ØØ355 FFAD BD FFØØ         JSR         INCH      READ CHARACTER
ØØ356 FFBØ 17              TBA                   MAKE A COPY
ØØ357 FFB1 8D CC           BSR         OUTS      PRINT SPACE
ØØ358 FFB3 81 4C           CMP A       #'L
ØØ359 FFB5 27 97           BEQ         LLOAD     LOAD PAPER TAPE
ØØ36Ø FFB7 81 4A           CMP A       #'J
ØØ361 FFB9 26 Ø4           BNE         NOTJ
ØØ362 FFBB 8D A4           BSR         BADDR     GET ADDRESS TO JUMP TO
ØØ363 FFBD 6E ØØ           JMP         X         JUMP TO IT
ØØ364 FFBF 81 4D    NOTJ   CMP A       #'M
ØØ365 FFC1 27 CE           BEQ         CHANGE    EXAMINE & DEPOSIT
ØØ366 FFC3 81 4E           CMP A       #'N
ØØ367 FFC5 27 BC          BEQ         NCHANG   E & D NEXT
ØØ368 FFC7 81 5Ø           CMP A       #'P
ØØ369 FFC9 26 D4           BNE         CRLF
ØØ37Ø FFCB 3B              RTI                   PROCEDE FROM BREAKPOINT
ØØ371 FFCC 8E ØØF5 RESET   LDS         #BUFULL   INIT STACK POINTER
ØØ372 FFCF 4F              CLR A
ØØ373 FFDØ 36              PSH A                 INIT BUFFER FULL FLAG
ØØ374 FFD1 36              PSH A                 INIT EXT CHAR FLAG
ØØ375 FFD2 36              PSH A                 INIT ECHO FLAG
ØØ376 FFD3 36              PSH A                 INIT BRKADR FLAG
ØØ377               **
ØØ378               * SOFTWARE INTERRUPT ENTRY POINT
ØØ379               **
ØØ38Ø FFD4 9F F6    INTRPT STS         SAVSTK    SAVE STACK POINTER
ØØ381 FFD6 9F FA           STS         XHI       SAVE SP FOR N COMMAND
ØØ382 FFD8 ØE              CLI                   ENABLE INTERRUPTS
ØØ383 FFD9 B6 FØØ2         LDA A       STRAPS    IF NO TERMINAL BIT IS SET
ØØ384 FFDC 9A F2           ORA A       BRKADR    OR BIT 7 OF BRKADR IS SET
ØØ385 FFDE 2B 2Ø           BMI         NOTERM    JUMP TO Ø
ØØ386 FFEØ 2Ø BD           BRA         CRLF      TO COMMAND DECODER
ØØ387               **
ØØ388               * EXTENDED CHARACTER TABLE
ØØ389               **
ØØ39Ø FFE2 Ø3       EXTEND FCB         3
ØØ391 FFE3 5F              FCC         / /
ØØ392 FFE4 1E              FCB         $1E
ØØ393 FFE5 3D              FCC         /=/
ØØ394 FFE6 Ø9              FCB         $9
ØØ395 FFE7 1B              FCB         $1B       ESCAPE CHARACTER
ØØ396 FFE8 ØD              FCB         $D
ØØ397 FFE9 5E              FCC         / /
ØØ398 FFEA 1A              FCB         $1A
ØØ399 FFEB 2B              FCC         /+/
ØØ4ØØ FFEC ØF              FCB         $F
```

```
PAGE  009    PROM MON


00401 FFED 3C               FCC        /</
00402 FFEE 12               FCB        $12
00403 FFEF 3E               FCC        />/
00404 FFF0 1C               FCB        $1C
00405 FFF1 2A               FCC        /*/
00406 FFF2 11               FCB        $11
00407 FFF3 23               FCC        /#/
00408 FFF4 19               FCB        $19
00409 FFF5 25               FCC        /%/
00410 FFF6 0C               FCB        $C
00411 FFF7 40               FCC        /@/
00412                **
00413                * NOW COME THE INTERRUPT VECTORS
00414                **
00415 FFF8                  ORG        $FFF8
00416 FFF8 FE00             FDB        MIVEC      MI VECTOR
00417 FFFA FFD4             FDB        INTRPT     SWI VECTOR
00418 FFFC 0104             FDB        NMIVEC     NMI VECTOR
00419 FFFE FFCC             FDB        RESET      RESET VECTOR
00420                       END


MIVEC  FE00
NMIVEC 0104
CRAZY  0100
STRAPS F002
NOTERM 0000
ACIACS F000
ACIADA F001
STACK  00F1
BRKADR 00F2
ECHO   00F3
EXTFLG 00F4
BUFULL 00F5
SAVSTK 00F6
TEMP   00F8
BYTECT 00F9
XHI    00FA
XLOW   00FB
SHIFT  00FC
SAVEX  00FD
BUFFER 00FF
GET    FE00
UPCAS  FE0A
GETBIT FE29
CLRSF  FE3C
NTUP   FE3F
ADDAX  FE4F
DONE   FE53
PUT    FE5B
NXTBIT FE5E
WAIT11 FE68
WAIT   FE6B
UPCAR  FE6F
EXTCAR FE79
CHKNXT FE7E
```

```
PAGE 010    PROM MON



CHKUP  FE8F
OKUP   FE99
NXT    FEA7
BOUTCH FEB7
BOUT2  FEB9
RESTR  FECF
REST2  FED1
RET    FED6
SEARCH FED7
NXTCHK FED8
LOWCAS FEE4
INCH   FF00
HANG   FF03
INHEX  FF0D
IN1HG  FF21
BBOUTC FF22
POLCAT FF24
LOAD   FF28
LOAD11 FF41
LOAD15 FF4D
LLOAD  FF4E
C1     FF50
BYTE   FF52
BADDR  FF61
OUT2H  FF6C
OUTHR  FF74
OUTS   FF7F
OUTCH  FF81
NCHANG FF83
CHANGE FF91
CRLF   FF9F
NOTJ   FFBF
RESET  FFCC
INTRPT FFD4
EXTEND FFE2


TOTAL ERRORS 00000
```

mits

2450 Alamo SE
Albuquerque, NM 87106