

PROM Monitor

TABLE OF CONTENTS

I	ABSTRACT	page 1
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



©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 MONITORA) Power up sequence

- 1) Strap the appropriate bits at location F002 to indicate the presence of a terminal, the type of terminal, and the number of stop bits to be used. (See the 680B Operator's Manual.)
- 2) Turn the Altair<sup>T.M.</sup> 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 (00F6 and 00F7), 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 COMMANDSM - 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 00A2, the following command is used:

.M 00A2 FF <CR>

- 2) To deposit a 09 in location 0072, the following command is used:

.M 0072 E1 09

(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 0050, use the following commands:

.M 0050 00 4D

.N 0051 00 49

.N 0052 00 54

.N 0053 00 53

- 2) To check and correct a sequence of instructions located at 0015 through 0018, the following commands are used:

.M 0015 4C <CR>

.N 0016 5C <CR>

.N 0017 36 32

.N 0018 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 02F3, the following command is used:

.J 02F3

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 S0) 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 00F3). The command

M 00F3 03 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 00F3 FF 00

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 00F3 which has bit 7 set will suppress echoing, and any number loaded into 00F3 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 00F3 and to have the last data block load a 00 into location 00F3. 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 S00B000004D454D5445535420B5
SI0400F3FF08
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 MONITORSetting 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 00F6 and 00F7 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 00F6 and 00F7. 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 (=0) then the Monitor assumes processor control. (This is the normal course of events since the Monitor initializes BRKADR to 03 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 00F2 03 FF

or including the instruction

COM      \$F2      SET BRKADR FLAG

in a program, then control is transferred to location 0000 when a program breakpoint occurs. This feature can be used to perform special functions when program breakpoints occur. 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      0          BREAKPOINT ROUTINE ADDRESS
                                LDA B    #015        SEND CR AND LF
                                JSR      OUTCH       TO TERMINAL
                                LDA B    #012
                                JSR      OUTCH
                                TSX
                                LDA B    #7          X POINTS TO PROCESSOR STATUS
                                LOOP     LDA A    X    INITIALIZE COUNTER
                                PSH B
                                JSR      OUT2H       BYTE OF STATUS TO A REG
                                JSR      OUTS        OUT2H & OUTS CLOBBER B REG
                                PUL B    OUTS        PRINT OUT BYTE OF STATUS
                                INX          SPACE OVER
                                DEC B    RESTORE B REG
                                BNE      LOOP        BUMP POINTER
                                RTI          DECREMENT COUNTER
                                           IF NOT DONE, KEEP PRINTING
                                           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      0
                                JMP      $0300      THIS BREAKPOINT ROUTINE
                                ORG      $0300      STARTS AT 0300
                                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.

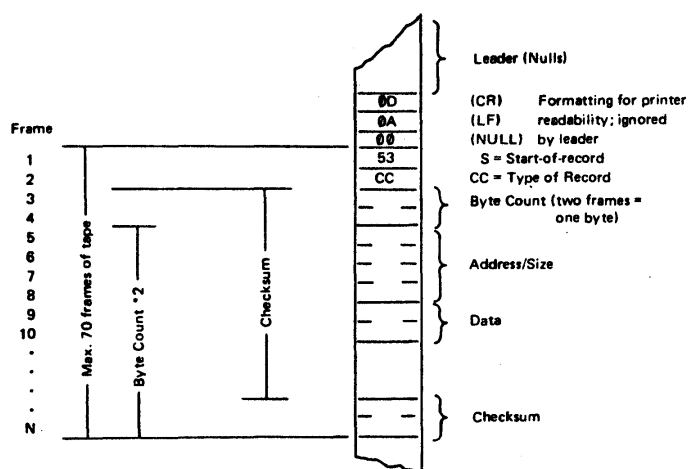
S0 = Header Record  
S1 = Data Record  
S9 = End of File Record

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:

S1NNAAAADDDDDDDDDDD.....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 checksum 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.

	CC = 30 Header Record		CC = 39 Data Record		CC = 39 End-of-File Record	
Frame						
1. Start-of-Record	53	S	53	S	53	S
2. Type of Record	30	0	31	1	39	9
3. Byte Count	31		31		30	
	32	12	36	16	33	03
4.	30		31		30	
6. Address/Size	30		31	1100	30	0000
7.	30	0000	30		30	
8.	30		30		30	
9. Data	34		39		46	FC
10.	38	48-11	38	98	43	
.	34		30			
.	34	44-D	32	32		(Checksum)
.	35					
.	32	52-R	41			
.			48	A8 (Checksum)		
N. Checksum	39					
	45	9E				

Reprinted with permission from Motorola.

**FIGURE 6-1. Paper Tape Format**



VII PROM MONITOR MEMORY USE INFORMATIONMonitor Memory Location

The ACIA version of the PROM Monitor is 256 bytes long and resides in locations FF00 through FFFF. The Baudot version of the Monitor is 512 bytes long and resides in locations FE00 through FFFF.

Monitor Stack

The stack pointer is initialized to 00F1 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 00F2 through 00FF 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 (00F2) - BREAKPOINT ADDRESS FLAG

If bit 7 of BRKADR is clear (=0) the Monitor gains processor control when a program breakpoint occurs. If bit 7 is set, control is transferred to location 0000 when a breakpoint occurs. See Section V for further information.

## ECHO (00F3) - 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 (00F4) - 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 (00F5) - 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 (00F6-00F7)

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

## TEMP (00F8)

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

## BYTECT (00F9) - BYTE COUNT

BYTECT contains the byte count during paper tape loading.

XHI (00FA)

XHI stores the high order byte of the index register.

XLO (00FB)

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 (00FC)

(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 (00FD-00FE)

(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 (00FF)

(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 0104.

The maskable interrupt vector points to location 0100 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 6800 MPU. Therefore, the interrupt mask (bit 4 in the processor condition codes register) must be clear (=0) to enable input from the Baudot Teletype.

The maskable interrupt vector points to location FE00. When a maskable interrupt request is acknowledged, the Monitor checks to see if 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 0104.

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.

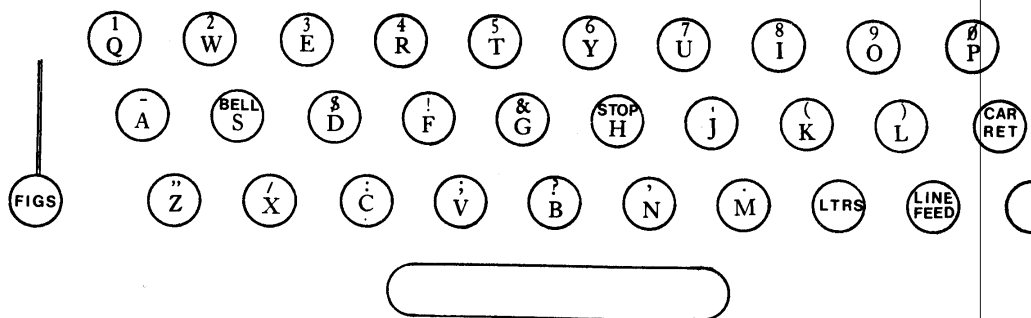


Figure 8-1. Baudot Keyboard

BAUDOT (OCTAL)	LOWER CASE	UPPER CASE	EXTENDED CASE
0	NULL	NULL	SEE *2 BELOW
1	E	3	
2	LINE FEED	LINE FEED	
3	A	-	
4	BLANK	BLANK	
5	S	CONTROL-G	
6	I	8	
7	U	7	
10	CAR RETURN	CAR RETURN	
11	D	\$	
12	R	4	ESCAPE
13	J	'	
14	N	,	@
15	F	!	~
16	C	:	
17	K	(	<
20	T	5	
21	Z	"	#
22	L	)	>
23	W	2	
24	H	SEE *1 BELOW	
25	Y	6	
26	P	0	
27	Q	1	
30	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 &lt;-&gt;ASCII Conversion



PAGE 001 PROM MON IX PROM MONITOR SOURCE LISTING (ACIA VERSION)

PAGE 002 PROM MON

```

00059          **
00060          * INPUT ONE HEX DIGIT INTO B REG
00061          * RETURN TO CALLING PROGRAM IF
00062          * CHARACTER RECEIVED IS A HEX
00063          * DIGIT. IF NOT HEX, GO TO CRLF
00064          **
00065 FF0F 8D EF  INHEX  BSR      INCH      GET A CHARACTER
00066 FF11 C0 30          SUB B      #'0
00067 FF13 2B 3C          BMI        C1      NOT HEX
00068 FF15 C1 09          CMP B      #$9
00069 FF17 2F 0A          BLE        IN1HG   NOT HEX
00070 FF19 C1 11          CMP B      #$11
00071 FF1B 2B 34          BMI        C1      NOT HEX
00072 FF1D C1 16          CMP B      #$16
00073 FF1F 2E 30          BGT        C1      NOT HEX
00074 FF21 C0 07          SUB B      #7      IT'S A LETTER-GET BCD
00075 FF23 39          IN1HG  RTS      RETURN
00076          **
00077          * POLE FOR CHARACTER
00078          * SETS CARRY IF CHARACTER IS IN BUFFER
00079          * CLOBBERS B REG
00080          **
00081          **
00082 FF24 F6 F000 POLCAT LDA B      ACIACS   ACIA STATUS TO B
00083 FF27 57          ASR B      ROTATE RDRF BIT INTO CARRY
00084 FF28 39          RTS      RETURN
00085          **
00086          * LOAD PAPER TAPE
00087          * LOAD ONLY S1 TYPE RECORDS
00088          * TERMINATE ON S9 OR CHECKSUM ERROR
00089          **
00090          **
00091          **
00092          **
00093 FF29 8D D5  LOAD   BSR      INCH      READ FRAME
00094 FF2B C0 53          SUB B      #'S
00095 FF2D 26 FA          BNE        LOAD    FIRST CHAR NOT (S)
00096 FF2F 8D CF          BSR      INCH      READ FRAME
00097 FF31 C1 39          CMP B      #'9
00098 FF33 27 1C          BEQ        C1      S9 END OF FILE
00099 FF35 C1 31          CMP B      #'1
00100 FF37 26 F0          BNE        LOAD    SECOND CHAR NOT (1)
00101 FF39 4F          CLR A      ZERO THE CHECKSUM
00102 FF3A 8D 17          BSR      BYTE     READ BYTE
00103 FF3C C0 02          SUB B      #2
00104 FF3E D7 F9          STA B      BYTECT   BYTE COUNT
00105 FF40 8D 20          BSR      BADDR    GET ADDRESS OF BLOCK
00106 FF42 8D 0F          BSR      BYTE     GET DATA BYTE
00107 FF44 7A 00F9  LOAD11 DEC      BYTECT  DECREMENT BYTE COUNT
00108 FF47 27 05          BEQ        LOAD15 DONE WITH THIS BLOCK
00109 FF49 E7 00          STA B      X      STORE DATA
00110 FF4B 08          INX      BUMP POINTER
00111 FF4C 20 F4          BRA        LOAD11 GO BACK FOR MORE
00112 FF4E 4C          LOAD15 INC A    INCREMENT CHECKSUM
00113 FF4F 27 D8          LLOAD  BEQ      LOAD  ALL OK - IT'S ZERO
00114 FF51 20 58          C1      BRA      CRLF  CHECKSUM ERROR - QUIT

```

PAGE 003 PROM MON

```

00117          **
00118          * READ BYTE (2 HEX DIGITS)
00119          * INTO B REG
00120          * A IS USED FOR PAPER TAPE CHECKSUM
00121          **
00122 FF53 8D BA  BYTE  BSR      INHEX  GET FIRST HEX DIG
00123 FF55 58      ASL B      SHIFT TO HIGH ORDER 4 BITS
00124 FF56 58      ASL B
00125 FF57 58      ASL B
00126 FF58 58      ASL B
00127 FF59 1B      ABA        ADD TO CHEKSUM
00128 FF5A D7 F8  STA B      STORE DIGIT
00129 FF5C 8D B1  BSR      INHEX  GET 2ND HEX DIG
00130 FF5E 1B      ABA        ADD TO CHECKSUM
00131 FF5F DB F8  ADD B      TEMP   COMBINE DIGITS TO GET BYTE
00132 FF61 39      RTS        RETURN
00133          **
00134          * READ 16 BIT ADDRESS INTO X
00135          * STORE SAME ADDRESS IN XHI & XLO
00136          * CLOBBERS B REG
00137          **
00138 FF62 8D EF  BADDR  BSR      BYTE   GET HIGH ORDER ADDRESS
00139 FF64 D7 FA  STA B      XHI      STORE IT
00140 FF66 8D EB  BSR      BYTE   GET LOW ORDER ADDRESS
00141 FF68 D7 FB  STA B      XLOW    STORE IT
00142 FF6A DE FA  LDX      XHI      LOAD X WITH ADDRESS BUILT
00143 FF6C 39      RTS        RETURN
00147          **
00148          * PRINT BYTE IN A REG
00149          * CLOBBERS B REG
00150          **
00151 FF6D 16  OUT2H  TAB      COPY BYTE TO B
00152 FF6E 54      LSR B      SHIFT TO RIGHT
00153 FF6F 54      LSR B
00154 FF70 54      LSR B
00155 FF71 54      LSR B
00156 FF72 8D 01  BSR      OUTHR   OUTPUT FIRST DIGIT
00157 FF74 16  TAB      BYTE INTO B AGAIN
00160 FF75 C4 0F  OUTHR  AND B      #$F  GET RID OF LEFT DIG
00161 FF77 CB 30  ADD B      #$30  GET ASCII
00162 FF79 C1 39  CMP B      #$39
00163 FF7B 23 04  BLS      OUTCH   IF IT'S A LETTER ADD 7
00164 FF7D CB 07  ADD B      #7     LINE UP OUTCH ENTRY POINTS
00165 FF7F 01  NOP
00166 FF80 01  NOP

```

PAGE 004 PROM MON

```

00167 FF81 8C      OUTCH FCB      $8C      USE CPX SKIP TRICK
00168 FF82 C6 20   OUTS  LDA B      #$20     OUTS PRINTS A SPACE
00171
00172 **
00173 ** OUTCH OUTPUTS CHARACTER IN B
00174 FF84 37
00175 FF85 8D 9D   OUTC1 BSR      POLCAT  ACIA STATUS TO B REG
00176 FF87 57      ASR B
00177 FF88 24 FB   BCC      OUTC1  XMIT NOT READY
00178 FF8A 33      PUL B      CHAR BACK TO B REG
00179 FF8B F7 F001 STA B      ACIADA  OUTPUT CHARACTER
00180 FF8E 39      RTS
00183 **
00184 * EXAMINE AND DEPOSIT NEXT
00185 * USES CONTENTS OF XHI & XLO AS POINTER
00186 **
00187 FF8F DE FA   NCHANG LDX      XHI      INCREMENT POINTER
00188 FF91 08      INX
00189 FF92 DF FA   STX      XHI
00190 FF94 96 FA   LDA A      XHI
00191 FF96 8D D5   BSR      OUT2H  PRINT OUT ADDRESS
00192 FF98 96 FB   LDA A      XLOW
00193 FF9A 8D D1   BSR      OUT2H
00194 FF9C 8C      FCB      $8C      USE CPX SKIP TRICK
00195
00196 **
00197 ** EXAMINE & DEPOSIT
00198 **
00198 FF9D 8D C3   CHANGE BSR      BADDR  BUILD ADDRESS
00199 FF9F 8D E1   BSR      OUTS     PRINT SPACE
00200 FFA1 A6 00   LDA A      X      BYTE INTO A
00201 FFA3 8D C8   BSR      OUT2H  PRINT BYTE
00202 FFA5 8D DB   BSR      OUTS     PRINT SPACE
00203 FFA7 8D AA   BSR      BYTE    GET NEW BYTE
00204 FFA9 E7 00   STA B      X      STORE NEW BYTE
00206
00207 **
00208 ** COMMAND DECODING SECTION
00209 **
00209 FFAB 9E F6   CRLF  LDS      SAVSTK
00210 FFAD C6 0D   LDA B      #$D      CARRIAGE RETURN
00211 FFAF 8D D0   BSR      OUTCH
00212 FFB1 C6 0A   LDA B      #$A      LINE FEED
00213 FFB3 8D CC   BSR      OUTCH
00214 FFB5 C6 2E   LDA B      #.      PROMPT CHARACTER
00215 FFB7 8D C8   BSR      OUTCH
00216 FFB9 BD FF00 JSR      INCH     READ CHARACTER
00217 FFBF 17      TBA      MAKE A COPY
00218 FFBF 8D C3   BSR      OUTS     PRINT SPACE
00219 FFBF 81 4C   CMP A      #'L
00220 FFC1 27 8C   BEQ      LLOAD  LOAD PAPER TAPE

```

PAGE 005 PROM MON

```

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 FFCE 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        PROCEED FROM BREAKPOINT
00233
00234      **
00235      * RESET ENTRY POINT
00236 FFD8 8E 00F3  RESET  LDS        #ECHO
00237 FFDB C6 03      LDA B        #3      INITIALIZE STACK POINTER
00238 FFDD 37      PSH B      INIT ECHO AND BRKADR FLAGS
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        #SD1
00245 FFEB F7 F000  STA B      ACIACS  INIT ACIA PORT
00246
00247      **
00248      * SOFTWARE INTERRUPT ENTRY POINT
00249 FFE8 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
00254
00255      **
00256      * NOW COME THE INTERRUPT VECTORS
00257
00258
00259
00260 FFF8      ORG        $FFF8
00261 FFF8      FDB        MIVEC  MI VECTOR
00262 FFF9      FDB        INTRPT  SWI VECTOR
00263 FFFA      FDB        NMIVEC  NMI VECTOR
00264 FFFB      FDB        RESET  RESET VECTOR
00265 FFFC      FDB
00266 FFFD      FDB
00267 FFFE      FDB
00268

```

PAGE 006 PROM MON

MIVEC 0100  
NMIVEC 0104  
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  
INCH FF00  
INHEX FF0F  
INLHG FF23  
POLCAT FF24  
LOAD FF29  
LOAD11 FF42  
LOAD15 FF4E  
LOAD FF4F  
C1 FF51  
BYTE FF53  
BADDR FF62  
OUT2H FF6D  
OUTHR FF75  
OUTCH FF81  
OUTS FF82  
OUTC1 FF85  
NCHANG FF8F  
CHANGE FF9D  
CRLF FFAB  
NOTJ FFCE  
RESET FFD8  
INTRPT FFEE

TOTAL ERRORS 00000

--- PAGE 001 PROM MON X PROM MONITOR SOURCE LISTING (BAUDOT VERSION)

```

00001          NAM          PROM    MONITOR
00002          **
00003          **  ALTAIR 680B PROM MONITOR
00004          **  BAUDOT VERSION 1.0
00005          **
00006          OPT          S          PRINT SYMBOL TABLE
00007          OPT          PAGE      PAGINATED LISTING
00008          FE00        MIVEC EQU   $FE00
00009          0104        NMIVEC EQU  $104
00010          0100        CRAZY EQU   $100
00011          F002        STRAPS EQU  $F002
00012          0000        NOTERM EQU  0
00013          F000        ACIACS EQU   $F000
00014          F001        ACIADA EQU   $F001
00015          00F1        ORG        $F1
00016          00F1 0001    STACK RMB 1          BOTTOM OF MONITOR'S STACK
00017          00F2 0001    BRKADR RMB 1          BREAKPOINT ADDRESS FLAG
00018          00F3 0001    ECHO RMB 1          TTY ECHO FLAG
00019          00F4 0001    EXTFLG RMB 1          EXTENDED CHARACTER FLAG
00020          00F5 0001    BUFULL RMB 1          BUFFER FULL FLAG
00021          00F6 0002    SAVSTK RMB 2          TEMP FOR STACK POINTER
00022          00F8 0001    TEMP RMB 1          TEMPORARY STORAGE
00023          00F9 0001    BYTECT RMB 1          BYTE COUNT
00024          00FA 0001    XHI RMB 1          XREG HIGH
00025          00FB 0001    XLOW RMB 1          XREG LOW
00026          00FC 0001    SHIFT RMB 1          BAUDOT SHIFT FLAG
00027          00FD 0002    SAVEX RMB 2          TEMP FOR INDEX REG
00028          00FF 0001    BUFFER RMB 1          BAUDOT CHARACTER BUFFER
00029          **
00030          * START OF PROM
00031          **
00032          FE00        ORG        $FE00
00033          **
00034          * MASKABLE INTERRUPT VECTOR POINTS TO GET
00035          **
00036          FE00 86 40    GET      LDA A      #$40    THIS BIT ROTATES INTO CARRY
00037          *                      TO SIGNAL STOP BIT ARRIVAL
00038          FE02 F6 F002    LDA B      STRAPS    IF BIT 0 OF F002 IS LOW
00039          FE05 56        ROR B                      THEN INTERRUPT CAME FROM BAUDOT
00040          FE06 24 21    BCC          GETBIT    SO CLOCK IN CHAR CODE
00041          FE08 7E        FCB          $7E      IF BIT 0 IS HIGH
00042          FE09 01        FCB          001      JUMP TO 0100 (HEX)
00043          **
00044          * THIS IS THE UPPPER CASE CONVERSION TABLE
00045          **
00046          FE0A 00        UPCAS FCB 0          NULL
00047          FE0B 33        FCC          /3/
00048          FE0C 0A        FCB          $A      LINE FEED
00049          FE0D 2D        FCC          /-/
00050          FE0E 20        FCB          $20     BLANK
00051          FE0F 07        FCB          7      CONTROL G (BELL)
00052          FE10 38        FCC          /87/
00053          FE11 37        FCB          $D      CARRIAGE RETURN
00054          FE12 0D

```

---  
PAGE 002 PROM MON

```

00054 FE13 24      FCC      /$4'/
      FE14 34
      FE15 27
00055 FE16 2C      FCC      /:/
00056 FE17 21      FCC      /!:(5/
      FE18 3A
      FE19 28
      FE1A 35
00057 FE1B 22      FCC      /"/
00058 FE1C 29      FCC      /)/
00059 FE1D 32      FCC      /2/
00060 FE1E 00      FCB      0
00061 FE1F 36      FCC      /6019?/  SLOT FOR STOP
      FE20 30
      FE21 31
      FE22 39
      FE23 3F
00062 FE24 00      FCB      0
00063 FE25 00      FCB      0
00064 FE26 2E      FCC      /:/
00065 FE27 2F      FCC      /!:/
00066 FE28 3B      FCC      /:/
00067
00068      **
00069      * END OF UPPER CASE TABLE
00070 FE29 8D 3D  GETBIT BSR      WAIT11  WAIT HALF A BIT TIME
00071 FE2B F6 F002 LDA B      STRAPS
00072 FE2E 56      ROR B      PUT DATA BIT INTO CARRY
00073 FE2F 8D 37  BSR      WAIT11  FINISH UP BIT TIME
00074 FE31 46      ROR A      COLLECT CODE IN A
00075 FE32 24 F5  BCC      GETBIT  IF MORE TO COME GO GET EM
00076 FE34 48      ASL A      GET RID OF STOP BIT
00077 FE35 44      LSR A      RIGHT JUSTIFY CODE
00078 FE36 44      LSR A
00079 FE37 44      LSR A
00080
00081      **
00082      * WE HAVE THE CODE IN A NOW
00083 FE38 81 1B      CMP A      #S1B  IF IT'S AN UPSHIFT
00084 FE3A 26 03      BNE      NTUP  SET THE SHIFT FLAG
00085 FE3C D7 FC  CLRSF STA B      SHIFT  AND RETURN FROM INTERRUPT
00086 FE3E 3B      RTI
00088 FE3F 5F      NTUP CLR B
00089 FE40 81 1F      CMP A      #S1F  IF IT'S A DOWNSHIFT
00090 FE42 27 F8      BEQ      CLRSF  CLEAR THE SHIFT FLAG
00092 FE44 D1 F4      CMP B      EXTFLG  IF EXTENDED CHARACTER
00093 FE46 2B 31      BMI      EXTCAR  IS SET GO TO EXT
00094      CHARACTER SEARCH
00095 FE48 CE FEE2      LDX      #LOWCAS-2
00096      * SET POINTER TO LOWER CASE
00097 FE4B D1 FC      CMP B      SHIFT  IF SHIFT FLAG IS SET
00098 FE4D 2B 20      BMI      UPCAR  THEN INDEX INTO UPPER CASE TABLE

```



PAGE 003 PROM MON

```

00099 FE4F 08      ADDAX INX      ADD A REG TO X REG
00100 FE50 4A      DEC A
00101 FE51 2A FC    BPL      ADDAX
00102 FE53 53      DONE      COM B   FORM MASK
00103 FE54 D7 F5    STA B      BUFULL SET BUFFER FULL FLAG
00104 FE56 E4 01    AND B      1,X    MASK OFF LOW 6 OR ALL 8
00105 FE58 D7 FF    STA B      BUFFER  STORE CHAR INTO BUFFER
00106 FE5A 3B      RTI        RETURN FROM THE INTERRUPT
00107
00108      **
00109      * PUT CLOCKS OUT THE CHARACTER CODE
00110      **
00110 FE5B 48      PUT      ASL A      ROTATE IN START BIT
00111 FE5C 8A 40    ORA A      #$40   OR IN STOP BIT
00112 FE5E B7 F002  NXTBIT STA A    $F002 SEND A BIT
00113 FE61 8D 05    BSR      WAIT11
00114 FE63 8D 03    BSR      WAIT11 WAIT AROUND FOR 22 MIL SECS
00115 FE65 44      LSR A      SHIFT TO NEXT BIT
00116 FE66 26 F6    BNE      NXTBIT IF MORE TO SEND THEN DO SO
00118 FE68 CE 02AF WAIT11 LDX      #687 11 MIL SEC DELAY
00119 FE6B 09      WAIT      DEX
00120 FE6C 26 FD    BNE      WAIT
00121 FE6E 39      RTS
00123 FE6F CE FE08  UPCAR LDX      #UPCAS-2 POINT TO UPPER CASE TABLE
00124 FE72 81 1A    CMP A      #$1A   IF IT'S THE EXTEND CHAR THEN
00125 FE74 26 D9    BNE      ADDAX   SET THE EXTENDED CHAR FLAG
00126 FE76 97 F4    STA A      EXTFLG AND RETURN FROM INTERRUPT
00127 FE78 3B      RTI
00129 FE79 CE FFE0  EXTCAR LDX      #EXTEND-2 POINT TO EXTENDED CHAR TABLE
00130 FE7C D7 F4    STA B      EXTFLG CLEAR THE EXTEND FLAG
00131 FE7E 08      CHKNXT INX
00132 FE7F 08      INX
00133 FE80 A1 00    CMP A      X
00134 FE82 27 CF    BEQ      DONE     SEARCH THE EXTENDED CHAR TABLE
00135 FE84 6D 00    TST      X         IF MATCH FOUND THEN WE ARE DONE
00136 FE86 2A F6    BPL      CHKNXT IF MINUS ENCOUNTERED THEN CODE NOT
00137 FE88 CE FEE2  LDX      #LOWCAS-2 IN TABLE SO MAKE INTO CONTROL CHAR
00138 FE8B C6 C0    LDA B      #SC0   BY TAKING LOWER CASE ASCII AND
00139 FE8D 20 C0    BRA      ADDAX   SETTING MASK TO GET RIG OF HI
00140 FE8F 96 FC    CHKUP  LDA A      ORDER 2 BITS
00141 FE91 26 06    BNE      SHIFT  BEFORE CHECKING UPPER CASE TABLE
00142 FE93 86 1B    LDA A      OKUP   CHECK THE SHIFT FLAG
00143 FE95 97 FC    STA A      #S1B  SEND OUT FIGURES SHIFT AND SET
00144 FE97 8D C2    BSR      SHIFT  SHIFT FLAG AS NECESSARY
00145 FE99 CE FE0A  OKUP  LDX      PUT
00146 FE9C 8D 39    BSR      #UPCAS  SET POINTER TO UPPER CASE TABLE
00147 FE9E 2A 2F    BPL      SEARCH  CALL SEARCH ROUTINE
00148 FEA0 86 1A    LDA A      RESTR   IF POSITIVE, SEARCH WAS SUCCESSFUL
00149 FEA2 8D B7    BSR      #S1A  SEARCH FAILED SO OUTPUT EXTEND
00150 FEA4 CE FFE0  LDX      PUT      CHARACTER
00151 FEA7 E1 01    NXT     DEX      #EXTEND-2
00152 FEA9 27 24    BEQ      1,X    SEARCH THROUGH EXTENDED CHAR
                                RESTR  TABLE

```

---  
PAGE 004 PROM MON

```

00153 FEAB 08      INX
00154 FEAC 08      INX
00155 FEAD A6 00   LDA A      X      BUMP POINTER TWICE
00156 FEAF 2A F6   BPL      NXT  LOAD THE BAUDOT CODE INTO B
00157 FEB1 C6 20   LDA B      #$20  IF MINUS - END OF TABLE
00158 FEB3 8D 04   BSR      BOUT2  NO MATCH FOUND - OUTPUT BLANK
00159 FEB5 20 1A   BRA      REST2
00160
00161      **
00162      * BOUTCH IS THE OUTPUT CHARACTER ROUTINE
00163 FEB7 DF FD   BOUTCH STX      SAVEX  SAVE X,A,&B
00164 FEB9 0F     BOUT2  SEI      DISENABLE INTERRUPTS
00165 FEBB 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
00173 FEC8 86 1F   LDA A      #$1F   IF FLAG IS SET THEN SEND OUT
00174 FECA 8D 8F   BSR      PUT      LETTERS SHIFT AND CLEAR FLAG
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     RESTR  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      **
00185      * SUBROUTINE TO SEARCH CONVERSION TABLES
00186      * RETURNS WITH CODE IN A IF FOUND
00187      * RETURNS WITH N BIT SET IF NOT FOUND
00188
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      **
00198      * LOWER CASE CONVERSION TABLE
00199
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/
00205 FEEA 49
00206 FEEB 55

```

PAGE 005 PROM MON

```

00205 FEED 0D          FCB          $D          CARRIAGE RETURN
00206 FEED 44          FCC          /DRJNFCKTZLWHYPQOBG/
      FEEF 52
      FEEF 4A
      FEF0 4E
      FEF1 46
      FEF2 43
      FEF3 4B
      FEF4 54
      FEF5 5A
      FEF6 4C
      FEF7 57
      FEF8 48
      FEF9 59
      FEFA 50
      FEFB 51
      FEFC 4F
      FEFD 42
      FEFE 47
      FEFF 00
00207          FCB          0          SLOT FOR FIGURES SHIFT
00208          **
00209          * INCH ENTRY POINT MUST BE AT START OF SECOND PROM
00210          **
00211 FF00 4D          INCH          FCC          /MXV/
      FF01 58
      FF02 56
00213 FF03 8D 1F      HANG          BSR          POLCAT          IF BUFFER IS EMPTY
00214 FF05 24 FC      BCC          HANG          HANG AROUND FOR INTERRUPT
00215 FF07 7F 00F5    CLR          BUFULL        CLEAR THE BUFFER FULL FLAG
00216 FF0A D6 FF      LDA B          BUFFER        PUT CHAR INTO B
00217 FF0C 39          RTS          RETURN
00218          **
00219          * INPUT ONE HEX DIGIT INTO B REG
00220          * RETURN TO CALLING PROGRAM IF
00221          * CHARACTER RECEIVED IS A HEX
00222          * DIGIT. IF NOT HEX, GO TO CRLF
00223          **
00224 FF0D 8D F1      INHX          BSR          INCH          GET A CHARACTER
00225 FF0F C0 30      SUB B          #'0
00226 FF11 2B 3D      BMI          C1          NOT HEX
00227 FF13 C1 09      CMP B          $9
00228 FF15 2F 0A      BLE          INHX          NOT HEX
00229 FF17 C1 11      CMP B          $11
00230 FF19 2B 35      BMI          C1          NOT HEX
00231 FF1B C1 16      CMP B          $16
00232 FF1D 2E 31      BGT          C1          NOT HEX
00233 FF1F C0 07      SUB B          #7          IT'S A LETTER-GET BCD
00234 FF21 39          INHX          RTS          RETURN
00235          **
00236          * THIS HELPS LINE UP ENTRY POINTS
00237          **
00238 FF22 20 93      BBOUTC        BRA          BOUTCH
00239          **

```

PAGE 006 PROM MON

```

00240          * POLE FOR CHARACTER
00241          * SET CARRY IF CHAR IN BUFFER IS CURRENT
00242          * CLEAR CARRY IF NOT CURRENT
00243          **
00244 FF24 D6 F5 POLCAT LDA B      BUFULL
00245 FF26 57      ASR B
00246 FF27 39      RTS
00247          **
00248          * LOAD PAPER TAPE
00249          * LOAD ONLY S1 TYPE RECORDS
00250          * TERMINATE ON S9 OR CHECKSUM ERROR
00251          **
00252 FF28 8D D6 LOAD BSR INCH READ FRAME
00253 FF2A C0 53 SUB B #'S
00254 FF2C 26 FA BNE LOAD FIRST CHAR NOT (S)
00255 FF2E 8D D0 BSR INCH READ FRAME
00256 FF30 C1 39 CMP B #'9
00257 FF32 27 1C BEQ C1 S9 END OF FILE
00258 FF34 C1 31 CME B #'1
00259 FF36 26 F0 BNE LOAD SECOND CHAR NOT (1)
00260 FF38 4F CLR A ZERO THE CHECKSUM
00261 FF39 8D 17 BSR BYTE READ BYTE
00262 FF3B C0 02 SUB B #2
00263 FF3D D7 F9 STA B BYTECT BYTE COUNT
00264 FF3F 8D 20 BSR BADDR GET ADDRESS OF BLOCK
00265 FF41 8D 0F LOAD11 BSR BYTE GET DATA BYTE
00266 FF43 7A 00F9 DEC BYTECT DECREMENT BYTE COUNT
00267 FF46 27 05 BEQ LOAD15 DONE WITH THIS BLOCK
00268 FF48 E7 00 STA B X STORE DATA
00269 FF4A 08 INX BUMP POINTER
00270 FF4B 20 F4 BRA LOAD11 GO BACK FOR MORE
00271 FF4D 4C LOAD15 INC A INCREMENT CHECKSUM
00272 FF4E 27 D8 LLOAD BEQ LOAD ALL OK - IT'S ZERO
00273 FF50 20 4D C1 BRA CRLF CHECKSUM ERROR - QUIT
00274          **
00275          * READ BYTE (2 HEX DIGITS)
00276          * INTO B REG
00277          * A IS USED FOR PAPER TAPE CHECKSUM
00278          **
00279 FF52 8D B9 BYTE BSR INHEX GET FIRST HEX DIG
00280 FF54 58 ASL B SHIFT TO HIGH ORDER 4 BITS
00281 FF55 58 ASL B
00282 FF56 58 ASL B
00283 FF57 58 ASL B
00284 FF58 1B ABA ADD TO CHEKSUM
00285 FF59 D7 F8 STA B TEMP STORE DIGIT
00286 FF5B 8D B0 BSR INHEX GET 2ND HEX DIG
00287 FF5D 1B ABA ADD TO CHECKSUM
00288 FF5E DB F8 ADD B TEMP COMBINE DIGITS TO GET BYTE
00289 FF60 39 RTS RETURN
00290          **
00291          * READ 16 BIT ADDRESS INTO X
00292          * STORE SAME ADDRESS IN XHI & XLO
00293          * CLOBBERS B REG

```

PAGE 007 PROM MON

```

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  **
00303  * PRINT BYTE IN A REG
00304  * CLOBBERS B REG
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      $SF      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     IF IT'S A LETTER ADD 7
00316 FF7C CB 07      ADD B      #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      BOUTC
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 008 PROM MON

```

00348 FF9F 9E F6 CRLF LDS SAVSTK
00349 FFA1 C6 0D LDA B #SD CARRIAGE RETURN
00350 FFA3 8D DC BSR OUTCH
00351 FFA5 C6 0A LDA B #SA LINE FEED
00352 FFA7 8D D8 BSR OUTCH
00353 FFA9 C6 2E LDA B #. PROMPT CHARACTER
00354 FFAB 8D D4 BSR OUTCH
00355 FFAD BD FF00 JSR INCH READ CHARACTER
00356 FFB0 17 TBA MAKE A COPY
00357 FFB1 8D CC BSR OUTS PRINT SPACE
00358 FFB3 81 4C CMP A #L
00359 FFB5 27 97 BEQ LLOAD LOAD PAPER TAPE
00360 FFB7 81 4A CMP A #J
00361 FFB9 26 04 BNE NOTJ
00362 FFB8 8D A4 BSR BADDR GET ADDRESS TO JUMP TO
00363 FFB6 6E 00 JMP X JUMP TO IT
00364 FFBF 81 4D NOTJ CMP A #M
00365 FFC1 27 CE BEQ CHANGE EXAMINE & DEPOSIT
00366 FFC3 81 4E CMP A #N
00367 FFC5 27 BC BEQ NCHANG E & D NEXT
00368 FFC7 81 50 CMP A #P
00369 FFC9 26 D4 BNE CRLF
00370 FFCB 3B RTI
00371 FFCC 8E 00F5 RESET LDS #BUFULL PROCEED FROM BREAKPOINT
00372 FFCE 4F CLR A INIT STACK POINTER
00373 FFD0 36 PSH A
00374 FFD1 36 PSH A INIT BUFFER FULL FLAG
00375 FFD2 36 PSH A INIT EXT CHAR FLAG
00376 FFD3 36 PSH A INIT BRKADR FLAG
00377 **
00378 * SOFTWARE INTERRUPT ENTRY POINT
00379 **
00380 FFD4 9F F6 INTRPT STS SAVSTK SAVE STACK POINTER
00381 FFD6 9F FA STS XHI SAVE SP FOR N COMMAND
00382 FFD8 0E CLI ENABLE INTERRUPTS
00383 FFD9 B6 F002 LDA A STRAPS IF NO TERMINAL BIT IS SET
00384 FFD0 9A F2 ORA A BRKADR OR BIT 7 OF BRKADR IS SET
00385 FFDE 2B 20 BMI NOTERM JUMP TO 0
00386 FFE0 20 BD BRA CRLF TO COMMAND DECODER
00387 **
00388 * EXTENDED CHARACTER TABLE
00389 **
00390 FFE2 03 EXTEND FCB 3
00391 FFE3 5F FCC //
00392 FFE4 1E FCB $IE
00393 FFE5 3D FCC /=
00394 FFE6 09 FCB $9
00395 FFE7 1B FCB $B ESCAPE CHARACTER
00396 FFE8 0D FCB $D
00397 FFE9 5E FCC //
00398 FFEA 1A FCB $A
00399 FFEB 2B FCC /+
00400 FFEC 0F 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	/e/	
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  
 NIUP FE3F  
 ADDAX FE4F  
 DONE FE53  
 PUT FE5B  
 NXTBIT FE5E  
 WAIT11 FE68  
 WAIT FE6B  
 UPCAR FE6F  
 EXTCAR FE79  
 CHKNXT FE7E

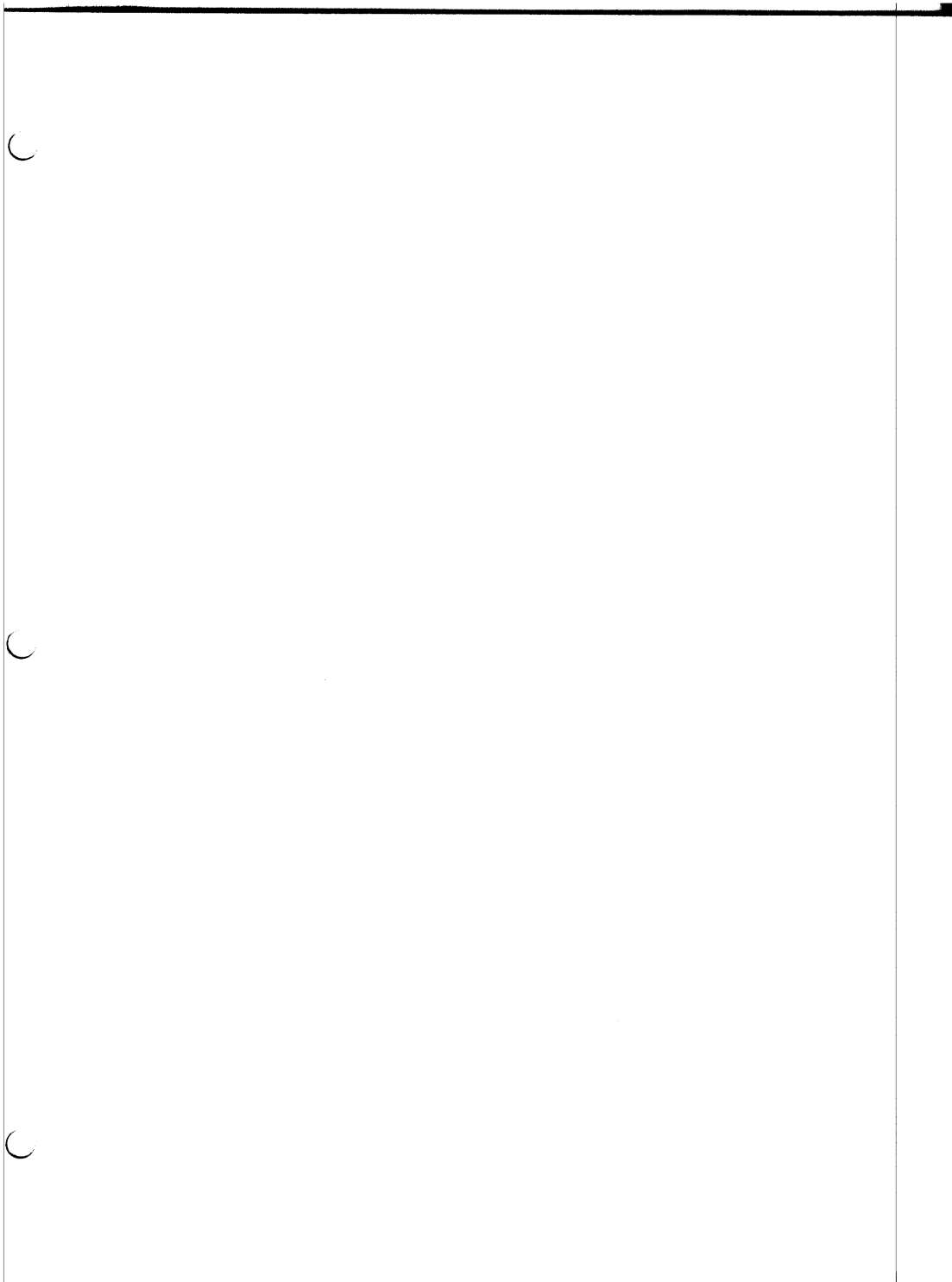
---  
PAGE 010 PROM MON

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
INLHG	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**

