

MicroByte

Software for the 2650

SBCOS

SBCOS

Monitor and Cassette Operating System
for the SBC-2560 Single Board Computer

By Ian Binnie and Martin Hood
Copyright MicroByte, 1982.

CONTENTS

1	Description	5
1.1	BINBUG	5
1.2	ACOS	6
2	System Requirements	7
3	Monitor Commands	9
3.1	A Access and Alter memory	9
3.2	B Breakpoint program	10
3.3	C Clear breakpoint	11
3.4	D Dump memory to tape	11
3.5	G Goto program start	12
3.6	K Cold start DOS	13
3.7	L Load tape into memory	13
3.8	S Set register	15
3.9	W Warm start DOS	15
4	Extending the Monitor command set	17
5	External character output routine	19
5.1	External output vector	19
5.2	Requirements of external output subroutine	19
6	Memory Mapped VDU Driver routine	21
6.1	Control Characters	21
6.2	Cursor	21
6.3	Scrolling and Paging modes	22
6.4	Fast Scrolling	22
6.5	Partial Scrolling	22
7	Serial Output Routine	25
7.1	Setting the speed	25
7.2	Fill characters	26
7.3	Stop bits	26
8	Keyboard Input Routines	29
8.1	Serial Keyboard	29
8.2	Parallel Keyboard	30
8.3	Break Key	30
9	Cassette Operating System	31
9.1	Data encoding and control	31
9.2	ACOS interface	33
9.3	Tape Hygiene	34

10	Cassette utility programs	35
10.1	110 Baud and 300 Baud Loader/dumpers	36
10.2	Catalog	37
10.3	Copy	38
10.4	Offload	39
10.5	Srcload	40
11	Monitor compatibility with PIPBUG	41
11.1	User accessible routines	41
11.2	Monitor independence of programs	42
12	Memory Map	43
13	Appendices	45
a.	Modifying the SBC-2650 board for SBCOS	45
b.	Installing BINBUG	46
c.	Installing ACOS	48
d.	Setting ACOS phase	50
14	Source listings	51

Chapter 1

DESCRIPTION

1.1 MONITOR

SBCOS contains both BINBUG v6 and v7. These are monitor programs for the 2650 intended for use with the SBC-2650 single board computer. The BINBUG v6 section supports a memory mapped Video Display Unit as the terminal display device, while BINBUG v7 is designed for serial output via the FLAG pin of the 2650. Either monitor can accept keyboard input serially through the SENSE pin, or in parallel through the AM2520 keyboard port. Both monitors utilise the MicroByte ACOS cassette operating system for mass storage and contain commands to branch to the MicroByte or VHS Disc Operating Systems.

To ensure compatibility with programs which access the monitor directly, all major user accessible routines have the same entry addresses as those in PIPBUG.

All PIPBUG commands are supported:

A	Access and Alter memory
B	Set Breakpoint
C	Clear Breakpoint
D	Dump memory to tape
G	Goto program and execute
L	Load from tape into memory
S	Set and display registers

Commands to enter MicroDOS or the VHS DOS at 6800 are also provided:

K	Cold start DOS
W	Warm start DOS

The set of monitor commands may be extended by the user through a vector contained in read/write memory.

The Breakpoint function has been enhanced. It now displays the contents of all registers, as well as the breakpoint address each time a breakpoint is executed. Only one breakpoint is supported by the monitors in SBCOS.

The Load and Dump commands communicate with the ACOS cassette operating system routines which are located at 6000 in EPROM. ACOS allows the recording on tape of named files at high speed with

exceptional reliability.

Both BINBUG v6 and BINBUG v7 can accept keyboard input from either a serial keyboard through the sense pin or from a parallel keyboard through the parallel keyboard port.

BINBUG v6 contains a set of VDU driver routines which output characters to a memory mapped display such as the DG640. The display may be operated in Scrolling, Paged or Mixed mode. Scrolling of the graphics bits may be inhibited to increase the speed of writing to the display.

BINBUG v7 contains a flexible serial output routine which can operate at a variety of speeds, provide one or two stop bits, and variable fill after carriage return and linefeed characters.

Monitor output may be directed to another output routine by changing a vector in read/write memory.

1.2 CASSETTE OPERATING SYSTEM

ACOS is an acronym for 'Audio Cassette Operating System'. ACOS provides a convenient and efficient means for storing onto, and retrieving from cassette tape, named files at high speed. The data transfer achieved by ACOS requires about 1.5 seconds per block of 256 bytes, or 6 seconds per K. Motor control for two cassette recorders is also provided.

The main routines for ACOS are contained in EPROM addressed at 6000. BINBUG v6 and BINBUG v7 include the ACOS commands as part of the monitor command set and contain routines to transfer data between the ACOS buffer and memory. ACOS commands may be written into user programs, because the ACOS program is written as a series of user accessible subroutines.

Files are broken up into 256 byte data blocks and excellent reliability is assured by the 16 bit cyclic redundancy check performed on each block when writing and reading files. The CRC, filename and other pertinent information is included in a postamble block which follows the data block.

Each block of data is transferred into a buffer area, both when loading and dumping, to allow the CRC to be calculated and to ensure that only error free blocks are transferred to main memory. ACOS will display a variety of prompts while loading a file to inform you of the progress of the load.

Utility programs are provided to dump and load 110 Baud PIPBUG ASCII format tapes, and 300 Baud BINBUG Binary format tapes using the ACOS hardware. Programs are also supplied to CATALOG and COPY ACOS tapes, to load ACOS object tapes at an offset address and to load BASIC source programs using ACOS hardware.

Chapter 2

SYSTEM REQUIREMENTS

- o SBC-2650 Single Board computer or equivalent, with a 2560 CPU operating at a clock speed of 1MHz or 2MHz and modified to address two 2716 EPROMs, one containing BINBUG in EPROM socket A (0000 - 03FF) and the other containing ACOS in EPROM socket B (6000 - 63FF). Details of the modifications required are contained in the appendices. These modifications will not be required if an EPROM board is available which can contain the ACOS EPROM.
- o An 8255 or 2655 PPI addressed at extended port 30 - 33. Only port C is actually used by SBCOS.
- o Initialisation switches connected to Bits 4, 5, and 6 of the 8255 port C to establish parallel/serial keyboard and communication baud rate.
- o Exclusive use of read/write memory between 0400 - 055F for monitor scratch and ACOS buffer.
- o Serial ASCII keyboard with an output data rate the same as the serial output routine and which is input through the SENSE pin,

OR

Parallel output keyboard which is input through the AM2520 keyboard port addressed at 35 (status) and 36 (data).
- o Memory mapped VDU addressed at 7800 - 7FFF in memory space with a format of 16 line x 64 characters per line. e.g. DG640,

OR

Serial VDU operating at a speed of 300 Baud. Various additional speeds are available depending on clock speed.
- o ACOS cassette operating system hardware connected to port C of the on board PPI, bits 0, 1, 2 and 7.

Optional:
 - o MicroByte or VHS Disc Operating System.

Chapter 3

MONITOR COMMANDS

The general format for all MONITOR commands is

Command [hex] [hex] [hex]<cr or lf>

where

hex is a hexadecimal number between 0 and 7FFF and is either an address
 or the number of a register.
cr is a Carriage return
lf is a Linefeed,
< > implies that the component is essential, and
[] implies the component is optional.

Hexadecimal numbers are separated from each other by a space.

3.1 ACCESS AND ALTER MEMORY

A <address><cr or lf>

This command allows a specified memory location to be examined and, if in read/write memory, altered.

A <address><cr or lf>

will display the address in memory and its contents in the following format:

aaaa xx

where aaaa is the address in memory and xx is the contents. To alter the contents of this memory location, enter the hexadecimal value which is to replace the present contents, followed by <cr> or <lf>.

<cr> will exit to the monitor command level.

<lf> will cause the following address and its contents to be displayed.

If no change is to be made, type <lf> to display the next address, or <cr> to exit to the monitor command level.

3.2 SET BREAKPOINT

B <address><cr or lf>

Setting a breakpoint in a program allows the execution of that program to be stopped at a specific instruction and the state of the 2650's registers saved for later inspection.

The Breakpoint function replaces two bytes of code at the breakpoint address specified with a ZBRR indirect instruction which branches to the breakpoint handling code in the monitor. Because of this implementation, a Breakpoint may only be set in read write memory. When the Breakpoint is executed, the two bytes which have been replaced by the ZBRR instruction will be restored, and the address of the breakpoint and the contents of all the registers printed.

B 450<cr or lf>

will set a Breakpoint at 0450 by replacing the two program bytes at 0450 and 0451 by 9B 9B (ZBSR *1B). When the breakpoint instruction at 0450 is executed, the original contents of memory will be restored and the register states displayed in the following format:

ADDRESS	RO	R1	R2	R3	R1	R2	R3	PSU	PSL
		----lower---			----upper---				

Caution:

1. Since a breakpoint vector replaces TWO bytes of code, generally it is not possible to install a breakpoint at the address of a subroutine return instruction. The exception to this rule occurs when the code immediately following the return instruction is NOT executed before the breakpoint. This restriction occurs because the second byte of the breakpoint vector modifies the first byte of the following instruction and would cause an unpredictable program branch if executed.
2. The breakpoint address specified must be the address of the first byte of an instruction, otherwise the breakpoint will not be executed correctly.
3. If a breakpoint is NOT executed, the breakpoint should be cleared using the 'C' command before another breakpoint is set, otherwise the modified bytes of the program will not be restored.

DE xxxx [filename] Dump with EXECUTE address.

Enter the execute address, xxxx, and the filename of the file to be dumped. ACOS will then prompt with a

D

and expects the upper and lower address limits of the program to be entered. These addresses should be in the same form as for the DT command with the start address first followed by the end address. The filename does not need to be re-entered.

If the DUMP TAPE command is terminated with a LINEFEED instead of a carriage return, ACOS will dump the specified area of memory to tape and then prompt again with another 'D'.

The addresses of another area of memory may be entered and will be dumped as a continuation of the original file. When loaded, the sections will be loaded into memory as one file. As many sections as required may be appended to the one file by terminating the command line with a linefeed. The last command line should be terminated with a carriage return. When dumping files with an execute address or when dumping a file composed of a number of blocks, the filename and the execute address only need to be specified in the first command. Subsequent command lines need only contain the addresses of the block to be dumped.

The motor of the DUMP recorder will be turned off after each block dumped. The COPY utility may be used to compact the file into one consisting of continuous blocks.

Note: ACOS uses the area of memory between 0440 and 055F for scratch and its block buffer. For this reason it is NOT possible to LOAD or DUMP programs which reside in this area of memory. You are encouraged to adopt our programming standard, which defines the area of memory below 0800 as a scratch area, and to write all your programs commencing at address 0800 or higher. MicroByte products will adhere to this standard in the future.

3.5 GOTO ADDRESS AND EXECUTE PROGRAM

G <address><cr or lf>

This command will cause control to be passed to a program in memory at the address specified. All registers are preset to the state displayed by the 'S' command. If a breakpoint has been executed, the registers will be restored to the contents they had immediately prior to the breakpoint. The registers are all set to zero when the computer is reset.

3.6 COLD START MicroDOS or VHS DOS (if installed)

K<cr or lf>

A test is performed to determine if the Disk Operating System program is present at 6800 and if it is, control is passed to the cold start entry point.

3.7 LOAD ACOS FORMAT TAPE

L <qualifier> [parameters]<cr or lf>

Load an ACOS format object tape.

LN LOAD recorder ON.

Switch on the motor of the load recorder to enable the tape to be rewound etc.

LF LOAD recorder OFF

Switch off the motor of the load recorder.

LT [filename] LOAD TAPE

LOAD from tape the file with the specified filename. If no filename is specified, the first object file encountered on the tape will be loaded. During loading the following characters will be displayed on the terminal.

An error free block has been read, but the filename does not correspond to the filename specified.

^ An error free block has been read, but the file is not of the appropriate TYPE. For example, this symbol will be displayed if an attempt is made to load a source code or data file using the monitor LOAD commands. Only OBJECT type files may be loaded using the monitor LOAD commands.

? A CRC error has been detected in the last block read.

When the correct filename has been found, one of the following characters will be displayed:

nn This is the hexadecimal number of the last error free block loaded.

S The block read had the correct filename but the first block of the file has been missed.

M An error was detected in a block in the middle of the file.

L The last block of the file has been missed.

A question mark will be displayed on the terminal at the conclusion of the load if an error has been detected during the loading process. If no errors have been detected, the monitor prompt will be displayed.

LE [filename] LOAD and EXECUTE

This command will LOAD the specified file into memory in the same way as the LT command. If no filename is specified, the first object file encountered on the tape will be loaded. If no errors occurred during loading and the file was dumped with an execute address, the program will commence execution. If no execution address was recorded, or if an error was detected during loading, control will be passed to the monitor.

LL [filename] LOAD LEADER

This command reads blocks from a file and displays the same messages as the LT command, however the data blocks will not be transferred into memory. The LL command will also ignore the file type and allow any type of file to be checked. It may be used to:

- o Find the last file on a tape to allow another file to be dumped immediately following it, or to
- o Verify that a newly recorded tape file may be read without error.

3.8 DISPLAY AND SET REGISTER CONTENTS

S <register number><cr or lf>

Allows the state of each register at the time of the last breakpoint to be examined, either singly or collectively. The state of each register may be set to a particular value before continuing execution of the program.

S n<cr or lf>

will display the value which register n contained at the last breakpoint. Typing in a valid hexadecimal value will alter the value to which register n will be restored when execution of the program is resumed using the 'G' command.

<cr> will return to the monitor command level,
<lf> will display the next sequential register.

	Lower		Upper
S0	0		
S1	1	S4	1
S2	2	S5	2
S3	3	S6	3
S7	PSU	S8	PSL

SA displays ALL registers on a single line.

3.9 WARM START MicroDOS or VHS DOS (if installed)

W<cr or lf>

If Disk Operating System is present at 6800, this command will perform a 'warm' start by branching indirectly through the ACON at 6803 to the DOS re-entry address.

Chapter 4

EXTENDING THE COMMAND SET

Additional commands may be added to BINBUG by extending the command chain.

The ADDRESS CONSTANT (ACON) named COMD (0431-2), normally points to EBUG (001D), the monitor error re-entry address. This ACON can be altered by the user to point to further command decoding. The extended command decoding should terminate by branching to EBUG (001D) if the command is not found. R0 contains the first letter of the command and each of the extended command routines should exit to MBUG (0022).

If the computer is reset, or the monitor entered at 0000, COMD will be reset to include only the monitor commands.

Chapter 5

EXTERNAL CHARACTER OUTPUT

All output from the monitor which is to be displayed on the terminal, and output from most user programs, will branch to the subroutine COUT (02B4). This subroutine normally will call either the memory mapped VDU driver routine (v6) or the serial output routine (v7), to output the character to the terminal display.

However, it is often desirable to be able to direct the output from the monitor to another device, for example, a printer.

5.1 EXTERNAL OUTPUT VECTOR

If the ADDRESS CONSTANT named EOUT (043B-C) is altered to point to the location of a subroutine which will output a character to the printer, output may then be switched between the terminal and the printer by switching OP (0409). If OP contains 00, characters will be output through the standard terminal driver, but if OP is non-zero, output will be directed to the external character output routine, indirectly via EOUT.

5.2 REQUIREMENTS OF EXTERNAL OUTPUT SUBROUTINE

The alternate output subroutine must emulate the normal COUT subroutine in all ways if compatibility is to be maintained.

Specifically, any alternate output routine must:

- o Nest subroutines three or fewer levels.
- o Return with the lower register bank selected.
- o Preserve the contents of R1, R2, and R3.
- o Not alter the status of the LCOM bit in PSL.
- o Not return with the WC bit of the PSL set.

Chapter 6

MEMORY MAPPED VDU DRIVER

BINBUG v6 contains a video display driver for the DG640 Video Display Unit. The DG640 VDU is a 16 line by 64 character display which supports lower case characters, inverted video, graphics and hardware flashing. To operate with BINBUG v6, the DG640 must occupy the 2K of memory space located at 7800 - 7FFF, at the top of the address space of the 2650. BINBUG v6 is selected by connecting A10 of the BINBUG EPROM (IC16-19) to OV.

6.1 CONTROL CHARACTERS

The video display driver will display on the VDU, all 96 printable ASCII characters, and will decode the following control characters.

Carriage return	CR	OD CTRL/M	Position to left of current line.
Linefeed	LF	OA CTRL/J	Advance to next line (depends on mode - see section 6.3)
Backspace	BS	OS CTRL/H	Back one position in same line (non - destructive)
Formfeed	FF	OC CTRL/L	Clear screen and enter paging mode.
Horizontal Tab	HT	O9 CTRL/I	Advance to next TAB position. Tabs are set at columns 9, 17, 25...
		OE CTRL/N	Clear screen but do not change mode.
ESCAPE	ESC	1B CTRL/[If in Paging mode then return to scrolling mode (see section 6.3)

All other control characters will be ignored.

6.2 CURSOR

The current position of the cursor is indicated by the character under the cursor being displayed in inverted video. Each call to the video driver with a printing character will cause that character to be written to the display at the current cursor position and the cursor to advance to the next position on the display.

6.3 SCROLLING AND PAGING MODES

The video driver permits output in either scrolling or paging mode but will always commence in scrolling mode after the computer has been reset. Paging mode may be entered at any time by sending an ASCII Formfeed (OC) to the video driver. The formfeed character will also clear the screen. Paging mode may also be selected by setting bit 7 of 0433.

In scrolling mode, writing will proceed down the screen until the screen is full. Then each linefeed will cause the screen to scroll up one line and the bottom line of the display will be cleared.

In paging mode, writing will proceed down the screen until the bottom is reached. The cursor will then disappear from the screen and the display will be 'frozen' until a character is entered at the keyboard. The action taken will depend upon the character entered.

ESCAPE	Return the display to the scrolling mode.
RETURN	Scroll the display up one line, but remain in the paging mode.
Any other	Clear the screen and display the next page of output on the screen.

6.4 FAST SCROLLING

When in the scrolling mode, the VDU driver will normally scroll the graphics and flashing bits (7C00 - 7FFF) as well as the character bytes (7800 - 7BFF) of the DG640. A 'fast scroll' mode is available by setting bit 4 of SCRF (0433). If this bit is set, only the display memory between 7800 - 7BFF will be scrolled, and not the graphics and flashing bits. Selecting the 'fast scroll' mode will double the scrolling rate and minimise flashing of the display during scrolling.

6.5 PARTIAL SCROLLING

Sometimes it is desirable to scroll only part of the screen. The video driver routine has provision to limit the scrolling to the bottom section of the screen. Setting bits 0 - 3 of SCRF (0433) to a value between 00 - 0F will prevent the top 0 - 15 lines of the display from being scrolled. This feature may be used in conjunction with the fast scroll mode, if desired. For example, if SCRF is set to H'15', the top five lines of the display will remain static - only the bottom eleven lines will be scrolled. None of the graphics bits will be scrolled.

Bit 7 of SCRF is set when the display is being written in the paging mode. Selection of the paging mode will override any of the scrolling options which are selected. Any pre-existing scrolling options will be restored when the scrolling mode is reselected.

SCROLL FLAG SCRF 0433

Bit	Description
-----	-------------

7	1 = page mode, 0 = scroll mode
6	x
5	x
4	1 = fast scroll, 0 = normal scroll
3	n Number of unscrolled lines at the top of screen.
2	n
1	n
0	n

Chapter 7

SERIAL OUTPUT ROUTINE

BINBUG v7 is designed for use with a video display terminal using a serial communications link and serial output is provided through the FLAG terminal. Software timing loops are used to generate the serial output, so the accuracy of the communication rate depends mainly on the accuracy of the microprocessor's clock. BINBUG v7 is selected by connecting A10 of the BINBUG EPROM (IC16-19) to +5V.

7.1 SETTING THE SPEED

Unlike PIPBUG, the serial output Baud rate is not fixed at 110 baud but may be one of a number of standard data communications rates.

The speed of the serial output routine is determined by the state of bits 4 and 5 of extended port 33 when the computer is reset or the monitor is entered at 0000.

Bit			Speed		Delay constants		Fill
6	5	4	1MHz	2MHz	DLY1 (042D)	DLY2 (042E)	
x	0	0	150	300	DD	DA	2
x	0	1	300	600	6E	6B	2
-	-	-	600	1200	36	34	-
x	1	0	1200	2400	1A	18	4
x	1	1	2400	4800	0C	0A	4

The monitor scratch locations 042D and 042E contain the constants which are used to determine the baud rate and these locations are loaded with an appropriate value on initialisation. Non-standard baud rates may be selected by storing different constants in these locations. This MUST be performed from within a program (unless the parallel keyboard port is used for input) and cannot be done using the 'A' command, since the monitor will cease to recognise the characters typed at the keyboard after the first constant has been altered and it will then be impossible to alter the second constant.

The following BASIC program will calculate the values of the constants and percentage error for a given Baud rate.

```
100 REM Program to calculate delay constants for BINBUG v5/6/7
110 F = 1E6 : REM 1 MHz
120 INPUT "Baud rate",R
130 D2 = ( (F/6) / R - 20 ) / 5
140 N2 = INT( D2 + .5 ) : REM round to nearest integer
150 D1 = ( (F/3) / R - 27 ) / 5 - N2
160 N1 = INT( D1 + .5 ) : REM round to nearest integer
165 REM Calculate actual rates and error percentages
170 S1 = (F/3) / ( 27 + 5*( N1 + N2 ) )
180 E1 = ( S1/R - 1 ) * 100
190 S2 = (F/3) / ( 28 + 5*( N1 + N2 ) )
200 E2 = ( S2/R - 1 ) * 100
210 PRINT R; " baud", "DLY1="; N1, "DLY2="; N2,
220 PRINT "Error"; E1; "% /"; E2; "%"
9999 END
```

7.2 FILL CHARACTERS

Most printer and many VDU type terminals require that a delay be provided after the Carriage Return or Linefeed characters to allow the printhead to move to left margin, the paper to feed up, or the display to scroll. Often this delay is accomplished by sending 'NULL' or other FILL characters which are ignored by the terminal, but waste sufficient time to ensure that no data is lost.

BINBUG v7 allows a delay to be inserted after either a Carriage Return or a Linefeed or after both of these characters. The delay may be equivalent to a delay of between one and thirty one fill characters. The contents of memory location 0433 (OUTF) controls this delay and also the number of stop bits appended to each character.

PADDING FLAG OUTF 0433

Bit	Effect
7	0 = 2 stop bits, 1 = 1 stop bits
6	1 = delay after Carriage Return
5	1 = delay after Linefeed
0 - 4	Number of character times of delay

The default number of character times of delay added after the Carriage Return and after the Linefeed is variable and is designed to maintain the delay added approximately constant at all communication rates.

The delay may be disabled by making bits 5 and 6 of OUTF 0.

7.3 STOP BITS

Bit 7 controls the number of stop bits inserted on output characters. Bit 7 = 1 causes one stop bit to follow each character while Bit 7 = 0 yields two stop bits. Bit 7 only has effect for output. The input routine will correctly interpret characters with either one or two stop bits irrespective of the status of bit 7.

Chapter 8

KEYBOARD INPUT ROUTINES

Both BINBUG v6 and BINBUG v7 can accept characters either from a serial keyboard connected to the SENSE pin or from a parallel keyboard connected to the SBC-2650 parallel keyboard input port, an AM2520, addressed at extended port 35,36.

The user accessible subroutine CHIN (0286) in SBCOS will input characters serially via the SENSE pin if bit 6 of port C of the on board PPI is HIGH and via the parallel keyboard port if bit 6 is LOW.

8.1 SERIAL KEYBOARD

Like PIPBUG, BINBUG v6 and BINBUG v7 can input serial data from a keyboard through the sense pin of the 2650. However unlike PIPBUG, the baud rate is not fixed at 110 baud but may be 300 Baud or another standard data communication rate. The other Baud rates which may be selected depends whether the CPU clock speed is 1MHz or 2MHz and whether BINBUG V6 or V7 is being used.

Bit			V6 (DG640)		V7 (Serial)	
6	5	4	1MHz	2MHz	1MHz	2MHz
0	0	0	150	300	150	300
0	0	1	300	600	300	600
0	1	0	1200	2400	1200	2400
0	1	1	N/A	N/A	2400	4800

BINBUG determines which speed to use for input (and output as well for BINBUG v7) by reading the status of bits 4 and 5 of port C of the on board PPI when the computer is reset or the monitor is entered at 0000.

The BINBUG monitors in SBCOS may only be used with a CPU clock rate of either 1MHz or 2MHz. They will not function correctly at 'odd' clock speeds.

The monitor scratch locations 042D and 042E contain the constants which are used to determine the baud rate and these locations are loaded with an appropriate value on initialisation. Non-standard baud rates may be selected by storing different constants in these locations. This MUST be performed from within a program and cannot be done using the 'A' command, since the monitor would cease to recognise the characters typed at the keyboard after the first constant was altered and it would then be impossible to alter the

second constant. Refer to section 7.1 for a table of delay constants.

8.2 PARALLEL KEYBOARD

A parallel keyboard may be connected to the AM2520 8 bit data latch on the SBC-2650. BINBUG expects to find this port addressed at extended port address 35 for the keyboard read latch and port 36 for the reset keyboard latch.

The AM2520 latches input data when it receives a positive going strobe pulse. While waiting for a key to be pressed, BINBUG continually reads the data port of the AM2520 until the contents of the data port are non-zero indicating that a character has been strobed in. After BINBUG has read the character input, it resets the data latch in the AM2520 to contain 00 by writing to port 36. No meaningful data is written to port 36. The action of writing to this port address is sufficient to reset the data latch. Because the data latch must contain a non-zero character for BINBUG to read its contents, the NULL character (00) cannot be entered using the monitor character input routine.

8.2 BREAK KEY

Many MicroByte programs poll the SENSE pin of the 2650 to detect a keyboard 'break' to indicate the program should be interrupted. To allow this feature to continue to function when using a parallel input keyboard, the BREAK key should be connected to the SENSE pin of the 2650. Provision is made on the SBC-2650 board to make this connection by means of straps, however the circuit diagram as published in ETI (December 1981) does not correctly describe the actual connections on the printed circuit board.

To enable BREAK from a parallel input keyboard, cut the etch link between 1 and 2 on pads W2a. This is the unlabeled strapping field immediately above W2. Connect pad 2 to 3 in the same field. This change disconnects the serial input from the SENSE pin and connects the BREAK key from the parallel keyboard to it instead.

Chapter 9

CASSETTE OPERATING SYSTEM

SBCOS requires the ACOS interface hardware to be connected to bits 0, 1, 2 and 7 of the PPI port C (extended port 33) A circuit design of the ACOS interface is included in section 9.2.

This section of the manual contains details about how ACOS works and information about how to setup ACOS. Details of the modifications required to the SBC-2650 board to install SBCOS are included in the appendices.

9.1 ACOS DATA ENCODING AND CONTROL

9.1.1 DATA FORMAT

The data recorded on tape is Frequency Modulated. Only a single cycle of tone is used for each 'unit' of code. A 'unit' of code consists of two bits of data. The data is encoded into four major frequencies, and each cycle of tone represents two bits or one 'nybble' of data. A BYTE of data consists of a start unit '00', four two bit data nybbles and a stop unit '11'. All the tones are generated and decoded by software.

9.1.2 ERROR CHECKING

Error checking is performed by calculating a 16 bit CRC for the contents of the buffer. If a file is being dumped, the CRC is recorded on the tape as part of the postamble block. When the tape is subsequently reloaded, the recorded CRC is compared to the calculated CRC. If the two CRC's fail to match, then a read error has occurred. A sequential block number is also included in the postamble. When loading, ACOS will detect an error if the numbers of the blocks read from tape are not sequential.

9.1.3 FILE FORMAT

A file to be recorded on tape is split up into 256 byte blocks of data. Appended to each block is an eighteen byte postamble.

The structure of one block of a file is:

- o Interfile gap '11' tone, approximately 2 seconds
- o Synchronising bytes, 8 x H'FF'
- o Start of Block Sync. Bytes, 2 x H'33'
- o 256 bytes of raw data.
- o Postamble, 18 bytes
- o Inter-record gap, approximately 0.5 seconds.

9.1.4 POSTAMBLE (044E - 045F)

byte	address	function
1 - 2	045E - 045F	Execute address
3	045D	File identification
4	045C	Number of data bytes in block
5 - 6	045A - 045B	Store address of first byte
7 - 14	0452 - 0459	Filename (backwards)
15	0451	Unassigned
16	0450	Block number in file
17 - 18	044E - 044F	16 bit CRC word

9.1.5 FILE IDENTIFICATION BYTE (0447)

A calling program may specify the TYPE of a file by loading a file type code into TYPE (0447). Only the four most significant bits are used for the file type.

The following file types have been defined:

0X	Object file, ie. Executable program machine code.
1X to 7X	Data file
8X	BASIC source file
9X	Assembler source file
AX	Text file
BX to FX	Unassigned Text type files

9.1.6 TAPE CONTROL BYTE (0446)

LTIP (0446) is used for a variety of control functions.

Bit 0	Dump tone output.
Bit 1	Dump machine ON if set.
Bit 2	Load machine ON if set.
Bit 3	Not used
Bit 4	Not used
Bit 5	If set, ACOS will disregard any errors encountered during loading, and will transfer the data from the buffer to memory despite the errors. This function is disabled during the loading of object type files. A block of an object file which results in a CRC error will NOT be transferred to memory even if Bit 5 of LTIP is set.

- Bit 6 If set, the output of block numbers during the load process will be inhibited. Instead an asterisk will be output for each error free block with the correct filename, which is loaded by ACOS.
- Bit 7 If set, this bit will prevent ACOS from transferring the blocks of data from the buffer into memory.

9.2 TAPE HYGIENE

The meticulous care of cassettes is extremely important when using a cassette system which records data at a speed in excess of 3000 Baud, since very little disturbance to the tape is required to 'drop a bit'. The word 'hygiene' has been used to emphasise this point.

110 and 300 Baud Cassette systems are not particularly critical with regard to tape handling. Tapes recorded by ACOS are sensitive to particles of dust and spots of finger oil on the tape, because these will cause drop-outs which will in turn cause read errors. However, if the following suggestions are adhered to ACOS will be a thoroughly reliable cassette system.

- o NEVER remove a cassette from the recorder, or even press the eject button, unless the cassette is fully rewound. It is probably not worth risking the loss of valuable data by attempting to record ACOS files on a tape which has been previously mistreated in this way.
- o Regularly clean the heads of the cassette recorders with a cotton bud and alcohol.
- o Use good quality cassettes. In particular, avoid using cassettes on which the oxide is loose. This will cause a gradual buildup of oxide which will foul the heads and cause read errors. Many cheap cassettes suffer from this problem of loose oxide, and many also contain excessive dropouts. Generally speaking, the 'low noise' category of cassette is sufficient and the use of 'ultra-dynamic' cassettes represents overkill.
- o Periodically de-magnetise the heads of the cassette recorder.

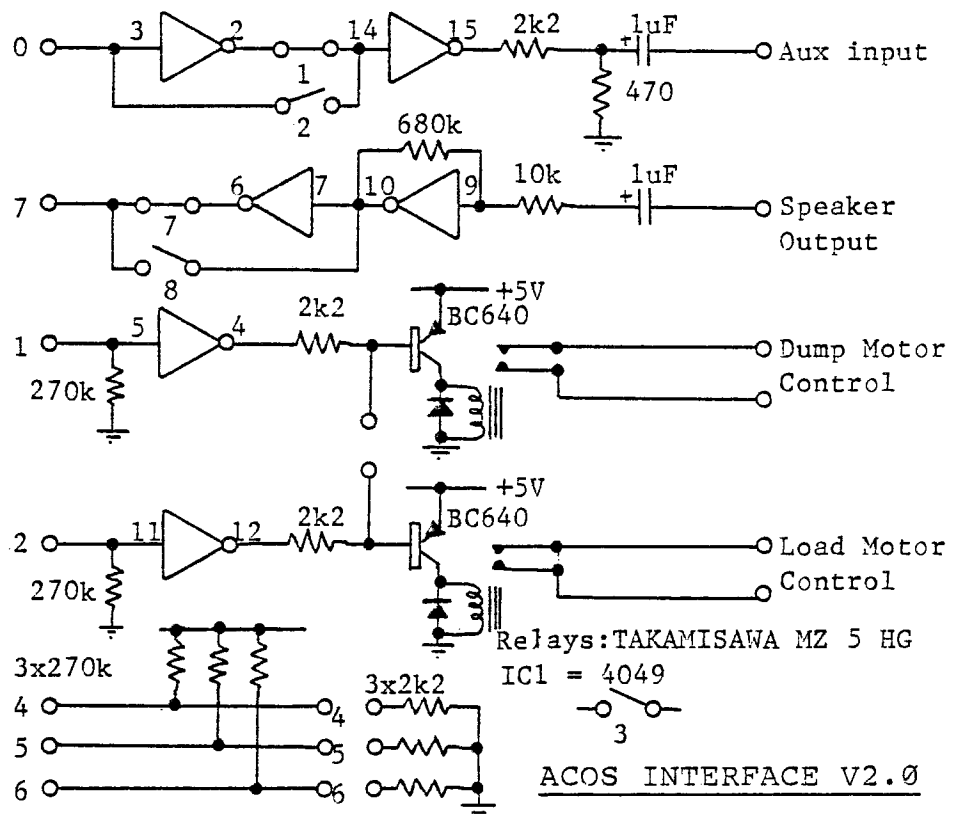
9.3 ACOS INTERFACE

9.3.1 PORT ALLOCATION TABLE

	Dump Machine	Load Machine
Audio	Port 32 bit 0.	Port 32 bit 7.
Motor	Port 32 bit 1.	Port 32 bit 2.

9.3.2 ACOS INTERFACE CIRCUIT DIAGRAM

The following diagram is the recommended hardware configuration when using two cassette recorders. ACOS may also be used with only one cassette recorder by connecting the motor control lines in parallel.



Chapter 10

ACOS UTILITY PROGRAMS

Six utility programs are provided with SBCOS. These programs can:

- o LOAD and DUMP files in other cassette formats using the ACOS hardware,
- o Catalog the files on an ACOS tape,
- o Copy files from one ACOS tape to another,
- o Load ACOS files at an displaced address, and
- o Load the 300 Baud program tapes produced by the BINBUG 3.6 version of MicroByte BASIC.

10.1 KANSAS CITY STANDARD LOADER/DUMPER PROGRAMS.

LD110	110 Baud PIPBUG ASCII format load-dump utility. (0800) and (2000)
LD300	300 Baud BINBUG Binary format load-dump utility. (0800) and (2000)

Both programs use the ACOS hardware to load and dump cassettes which are recorded in the Kansas City standard. The 1200/2400 Hz tones are generated by software and no additional cassette interface is required. Each load / dump utility is supplied as two versions which are located at different addresses in memory. The first version of each utility begins at memory location 0800 and is suffixed 'L', while the alternate versions all begin at 2000 and have their filename suffixed 'H'.

10.1.1 USING THE PROGRAMS

On entry to the utility the header line and an option prompt are printed. The command set is tabulated below and is identical for both the load / dump utility programs.

L	Switch on the LOAD recorder and LOAD into memory the first valid file found. The file will NOT auto-execute at the conclusion of loading.
Loooo	Load into memory the first valid object file found at the address specified. If no address is specified or 0000 is entered, the file will be loaded at the address recorded on the tape.
Dnnnn mmmm xxxx	Switch on the DUMP machine and dump to tape the contents of memory from nnnn to mmmm with an execute address of xxxx.
H	Output a 10 second HEADER of 3300 Hz tone on the DUMP machine.
E	EXIT to the monitor.

10.2 CAT (0800)

This utility will read a previously recorded ACOS format tape and list a CATALOG of the files contained on it. If the file is an object file, the load address, end address and execute address will also be listed. An object file which consists of non-contiguous blocks, will have this indicated with an 'N' after the start address.

10.2.1 USING THE PROGRAM

- o Load CATALOG into memory.
- o Place the tape to be cataloged into the load machine.
- o Type G800 to execute the CATALOG program. CATALOG will print its header line and then prompt:

PRINTER <N> ?

A response of 'Y' will cause the output to be directed to a printer as well as to the terminal. Any other response will cause the catalog to be listed on the terminal only.

- o CATALOG should be terminated by pressing the BREAK key, since the program has no way of determining when it has read the last file on the tape.

10.2.2 PRINTER OUTPUT

In order to use the printer option, the branches located at 0806, 0809 and 080C must be altered to point to printer handler routines. These routines should:

0806 Output the character in R0 to the printer.
0809 Initialise the printer.
080C Turn the printer off.

As supplied, the initialise (0809) printer output (0806) and terminate (080C) branches all point to a subroutine return instruction. If the printer used does not require initialisation or termination, the branches at 0809 and 080C should remain pointing to the return instruction.

VDU initialise (080F) and VDU terminate (0812), function with a memory mapped VDU to clear the screen, and prevent scrolling of the top line of the screen. Currently the first byte of each branch is a RETC,UN (17). To enable memory mapped VDU initialisation, alter 080F and 0812 from 17 to 1F.

You may try this utility by inserting the utilities cassette in the LOAD machine and typing 'LE CAT'.

10.3 COPY (0800)

This program will automatically copy all the files on one tape to another. This is achieved by loading each file into memory and then dumping the file in precisely the same format as it was loaded. Details of the files being copied are displayed as the copying proceeds. If an error is detected while loading, the copy procedure will be terminated. Copy requires that two cassette recorders be connected, a LOAD machine and a DUMP machine.

10.3.1 MEMORY REQUIREMENTS

Sufficient RAM must be available to contain the longest file in memory in its entirety. In practice, since the postamble blocks are also stored in memory, about 7% more memory is required than the actual length of the loaded file.

10.3.2 USING THE PROGRAM

- o Load COPY into memory.
- o Place the tape to be copied in the load machine.
- o Place a blank tape in the dump machine.
- o Type G800 to execute the COPY program.

Similar headings to those used by the CATALOG utility will be displayed and a ten second header will be output to the DUMP machine. The load machine will be turned on and the first file will be loaded into a buffer area of memory.

The file is stored in memory as a mirror image of the data on the tape, including the postamble. When the file has been completely loaded, the filename and its details will be displayed as a catalog entry and a duplicate of the file will be output to the dump machine.

This process will repeat until copying is terminated by a BREAK from the keyboard during the loading of a file, the computer is reset, or an error is detected during loading. COPY has no way of determining when it has reached the last file on the tape, so it relies on the user to terminate the COPY process.

10.3.4 MEMORY MAPPED VDU INITIALISE.

To enable memory mapped VDU initialisation, which prevents scrolling of the top line of the screen, alter 080F and 0812 from 17 to 1F.

10.3.5 ERRORS

If the tape copying process is stopped by a read error, the copy may be continued without generating a header by entering the utility at 0803, however the file which generated the error will not be copied. A memory error, or insufficient memory will also terminate the copying process.

10.4 OFFLOAD (0800) and (2000)

OFFLOAD is an ACOS utility program which will load an ACOS object or source file and store it in memory at an address different from the location in memory from which it was originally dumped. This facility can be useful for loading an object file into an EPROM programming buffer. No 'relocation' is performed on the contents of an object file when it is loaded. The file will be transferred to memory precisely as it is recorded on the tape.

10.4.1 USING THE PROGRAM

- o Load OFFLOAD into memory with the command LT OFFLOADL and type G800 to execute the program (the low memory version).
- o The program's header line will be printed.

```
MicroByte ACOS offset loader v 2.1
Load address [File name] ?
```

- o Enter the address in memory at which the file is to be loaded. The name of the file to be loaded may be included as well but if is omitted, OFFLOAD will load the first file it encounters.

For example:

```
*LT OFFLOADL
0102
*G800
MicroByte ACOS offset loader v 2.1
Load address [File name] ? 2800 TESTFIL
010203
*
```

Alternatively, the load address and/or filename can be included in the command line. e.g.

```
*G800 3000 COPY
```

- o OFFLOAD will always return to the monitor at the conclusion of loading a file since, in general, the loaded file will not be executable at its offset location.

Note: Attempting to auto-execute OFFLOAD with the LE <filename> command will cause the program to exit to the monitor in error.

10.5 SRCLOAD (0600)

This program will load the specially formatted source file produced by the BINBUG 3.6 version of BASIC. BASIC saves a program as an image of memory. The tape contains the ASCII characters which comprise the program preceded by an STX character (02) and terminated by an ETX character (03). The data is not broken into blocks and does not contain any checksum or address information. The end of the file is detected by encountering the end of text marker (03). Most inconvenient of all, the tapes produced by BINBUG 3.6 BASIC can only be loaded by that same version of BASIC. The BINBUG 3.6 monitor cannot load them into memory.

These circumstances make it very tedious to convert programs saved under BINBUG 3.6 BASIC into ACOS format tapes. It involves configuring the BINBUG 3.6 monitor and tape hardware, loading BINBUG 3.6 BASIC, using this program to load the BASIC program, changing to the ACOS monitor and tape hardware, loading ACOS BASIC and then using BASIC to save the new file. An almost impossible task!

SRCLOAD eases the pain of this procedure considerably. It resides in memory at 0600, and will read BINBUG 3.6 BASIC program format tapes into memory at 2000 using the ACOS hardware.

To convert a BASIC program from BINBUG 3.6 to ACOS

- o Load ACOS BASIC into memory.
- o Load SRCLOAD into memory and execute the program by typing G600.
- o Use SRCLOAD to load a BINBUG format BASIC program.
- o Execute ACOS BASIC. Open the OLD file in memory and SAVE it to ACOS format tape. Running BASIC will destroy SRCLOAD so it will be necessary to go back to the second step and reload SRCLOAD to convert the next file.

Chapter 11

MONITOR COMPATABILITY WITH PIPBUG

To ensure that previously written programs which make subroutine calls to the PIPBUG monitor will still function correctly with BINBUG, the following entry points and subroutine start addresses are the same.

0000		Cold start - initialise most of scratch RAM. and establish serial I/O rate and keyboard type.
001D	EBUG	Error return - Outputs '?'.
0022	MBUG	Normal re-entry to monitor.

11.1 USER ACCESSIBLE SUBROUTINES

005B	LINE	Input a line of characters into the 20 character buffer at 0413. DEL (7F) will delete the last character entered and backspace the cursor. Input is terminated by a <cr> or <lf>.
008A	CRLF	Output Carriage Return and Linefeed.
00A4	STRT	Store R1,R2 into TEMP (040D,040E).
0269	BOUT	Output R1 as two hexadecimal digits.
0286	CHIN	Input a character from the keyboard into R0.
02B4	COUT	Output R0 to the terminal or to an external device if OP (409) is not zero.
02DB	GNUM	Get the next hexadecimal number from the input buffer into R1, R2.

All these subroutines MAY be used by user programs, however considering that CRLF, STRT and BOUT are only a few bytes in length it is preferable to code these routines into a program rather than rely on their existence in the monitor.

On the contrary, the monitor calls CHIN and COUT should ALWAYS be used by programs for input and output to ensure that the program can be transported between computers which have different means of getting characters from the terminal and outputting characters to

the display.

LINE and GNUM are very useful routines and their use by a small user program can often be justified to save memory. However, LINE suffers the limitation that its input buffer is only 20 characters in length, and GNUM exits to the monitor, rather than back to the calling program, if it encounters a non-hexadecimal character. Any program which needed to input longer lines or recover from possible user errors could not use these monitor subroutines.

11.2 MONITOR INDEPENDENCE OF PROGRAMS

It cannot be emphasised too strongly how important it is to write programs which are independent of the monitor being used. If you write your programs in such a way that the program is heavily dependent on routines contained in a particular monitor, then you will almost certainly guarantee that the program will not work on a computer which uses a different monitor. This is especially likely to be the case if calls are made to the monitor at non-standard entry points.

- o If monitor calls must be made from a program - I/O calls to CHIN and COUT are an example of two monitor calls made by almost every program - these calls should be made from the program through a single branch instruction or indirectly through a single Address Constant (ACON). If external calls are made this way, the program may be easily adapted for use with a different monitor, by altering a single branch to each monitor routine, rather than by changing many monitor references throughout the program.
- o Branching indirectly through monitor ACONs to access monitor subroutines, using a ZBRR * or ZBSR * instruction, is a poor practice, because no guarantee exists that these ACONs will exist at all, or be in the same position in another monitor. The one byte saved is not worth the difficulties encountered when attempting to alter the program to function with another monitor.

Chapter 12

MEMORY MAP

Monitor entry points.

0000	Monitor initialisation entry.
001D	Monitor error re-entry point.
0022	Monitor warm entry point.
005B	Input line subroutine.
008A	Output CR and LF.
00A4	Store R1, R2 in TEMP.
0269	Print R1 as two hexadecimal digits.
0286	Input character from keyboard into R0.
02B4	Output character in R0 to terminal.
02DB	Resolve hexadecimal number from line buffer into R1, R2.
03E7	Transfer data from memory to ACOS buffer.
03F3	Transfer data from ACOS buffer to memory.
03FF	Version number.

Monitor scratch RAM.

0400 - 0408	Storage for registers and PSW.
0409	Output switch
040A - 040C	Used by GOTO.
040D - 040E	TEMP, 16 bit address store.
040F - 0410	TEMQ, Second 16 bit address store.
0411	TEMR
0412	spare
0413 - 0426	BUFF Input buffer.
0427	BPTR Input buffer pointer.
0428	spare
0429	CNT
042A	CODE
042B	KFLAG 0 => Parallel Keyboard, non zero => Serial Keyboard.
042C	MARK Flag that breakpoint has been set.
042D	DLY1 Serial I/O delay constant.
042E	DLY2 Serial I/O delay constant.
042F - 0430	FROM (v6) spare (v7)
0431 - 0432	COMD Pointer to extension of monitor commands.
0433	SCRF Scrolling control byte. (v6)
	OUTF Padding control byte. (v7)
0434 - 0437	Breakpoint scratch.
0438	spare
0439 - 043A	PTR Current cursor location (v6), spare (v7).
043B - 043C	EOUT Pointer to external COUT subroutine.

ACOS scratch RAM.

0446	Tape control byte.
0447	File type.
0448 - 0449	CRC generate scratch.
044E - 044F	Cyclic Redundancy Checksum.
0450	Block Number.
0452 - 0459	Filename..
045A - 045B	Load address.
045C	Number of bytes in block.
045D	File Identification byte.
045E - 045F	Execution address.
0460 - 055F	ACOS block buffer.

ACOS entry points.

6000	LOAD functions entry point.
6003	DUMP functions entry point.
601A	Turn on DUMP machine.
6023	Turn off DUMP machine.
604D	Output 10 second header tone.
60C5	Turn on LOAD machine.
60CE	Turn off LOAD machine.
6203	Get filename from monitor input buffer.
62B8	Generate CRC in scratchpad.
62D5	Check CRC with CRC in postamble.
6302	Generate CRC for data in block buffer.
630E	Output one cycle.
6323	Encode and output one byte.
6346	Output 3 second header tone.
6353	Output block buffer and postamble to tape.
6389	Time one input cycle.
63AB	Input one byte.
63D6	Input one block from tape into block buffer.

Note: These documented entry points to ACOS are the only ones which should be used by user utility programs.

Appendix A

MODIFYING THE SBC-2650 BOARD FOR SBCOS

SBCOS has been specially designed to operate on the SBC-2650. Unfortunately the SBC-2650 was not designed to operate with SBCOS and some modifications to the CPU board are required to install and correctly address the ACOS EPROM. If an EPROM board is available which can be addressed to 6000-63FF, the ACOS EPROM can be installed in this device and the modifications described here will not be necessary.

SBCOS is supplied in two 2716 EPROMs.

- o One EPROM contains two special versions of BINBUG. The lower half supports the DG640 memory mapped VDU (v6), and the upper half supports serial terminals (v7).
- o The second EPROM contains two copies of ACOS. The lower half contains a 1MHz version of ACOS, while the upper half contains a 2MHz version.

SBCOS utilises port C of the on board Programmable Peripheral Interface (PPI) (8255 or 2655). The PPI must be addressed to occupy the port address range 30 - 33. The Port is configured so that bits 0 - 3 are output and bits 4 - 7 are input.

Modifications are required to the CPU board to alter the addressing range of the on board EPROM B socket (IC15) to address 6000 - 63FF (ACOS). These modifications are detailed in Appendix C.

An interface board is available which connects directly to the 16 pin DIL socket for the PPI port C on the SBC-2650.

Contents of the SBCOS 2716 EPROMs.

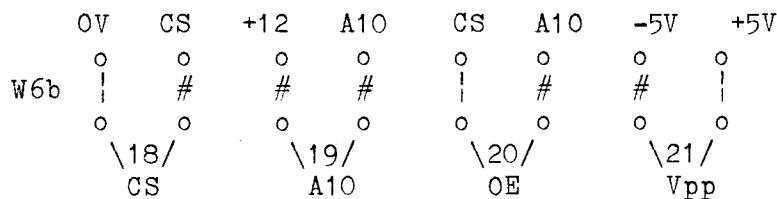
BINBUG v7.x Serial I/O	A10 = 1	ACOS 3.x 2MHz	A10 = 1
BINBUG v6.x DG640	A10 = 0	ACOS 3.x 1MHz	A10 = 0

Appendix B

INSTALLING BINBUG

The BINBUG EPROM will function normally if the SBC-2650 CPU board is set up as though the SBCBUG monitor were to be installed. To do this:

1. Solder wire-wrap pins into the EPROM strapping field holes, AFTER cutting the existing etched traces. This will make it very much easier to modify the EPROM strapping in the future.
2. Connect the link fields for EPROM A (IC16) as detailed in the diagram below. '|' is a link and '#' is a cut. This change will enable one half of the 2716 EPROM which contains one of the special BINBUG monitors.



Connect EPROM pin 19 (A10) to 0V to enable BINBUG V6.x for a memory mapped VDU, or to +5V to enable BINBUG V7.x for a serial output VDU.

3. If the MicroByte Interface Board is being used, ensure that the required modifications have been made to the PORT C socket to supply power to the interface. This modification involves connecting +5V to pin 16 of the port C socket and is detailed in the ACOS Interface Board instructions.

Plug the interface board into the port C socket and set the DIP switches appropriately for your terminal and keyboard devices. Bits 4, 5 and 6 of port C determine the characteristics of the I/O devices.

If a custom interface is being used, then bits 4, 5 and 6 of port C must be strapped either high or low to establish the keyboard input speed and whether a serial or parallel keyboard is being used.

Bit 6	Bit 4	Bit 5	
1	x	x	Parallel Keyboard input.
0	x	x	Serial input via SENSE pin.

Serial input and output data rate.

			1MHz	2MHz
x	0	0	150 Baud	300 Baud
x	1	0	300 Baud	600 Baud
x	0	1	1200 Baud	2400 Baud
x	1	1	2400 Baud	4800 Baud (V7.x only)

Set the switches in the MicroByte ACOS Interface Board to reflect the desired configuration.

Note: The setting of the switches is only read during monitor initialisation which occurs when the CPU is RESET, so altering the speed setting switches, for example, will not alter the I/O baud rate until the monitor is reinitialised or the CPU is RESET.

4. Insert the BINBUG 2716 into EPROM socket A (IC16).
5. Setup the PPI to be addressed to a base address of H'30'. Ensure that at least 1K of RAM is installed, addressed at 0400-07FF. Set the RAM and I/O address switch (SW1) as shown below.

1	2	3	4	5	6	7	8
ON	ON			ON	ON	ON	ON
		OFF	OFF				

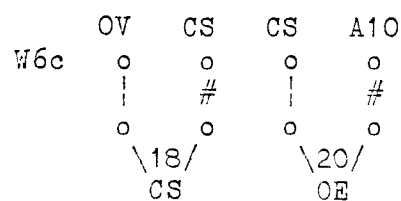
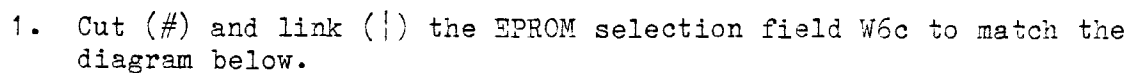
PORT 3x RAM 0000-OFFF

6. Connect the Keyboard and display device to the CPU board and apply power. The Monitor's prompt, an asterisk, should be displayed and all monitor commands other than the ACOS commands L and D should function normally as detailed in the BINBUG User's Guide. Executing either of the ACOS commands 'L' or 'D' at this stage will 'hang' the computer and require the CPU to be RESET to regain control.

Ensure that the SBC-2650 board and the selected monitor are functioning correctly before proceeding further.

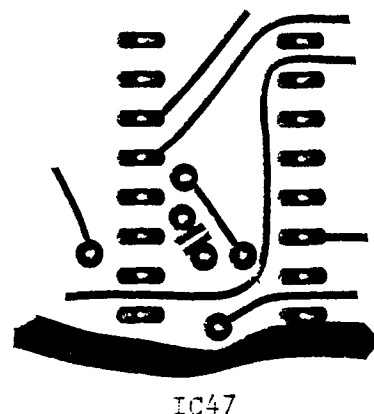
INSTALLING ACOS

The following modifications will cause one of the halves of the ACOS 2716 EPROM installed in EPROM B socket (IC15) to be addressed at 6000-63FF.



2. The remaining connections are most easily made using wire-wrap wire.
3. Connect W7-4 to W7-3 (EPROM B CS).
4. Install a link between IC30-18 (A14) and pad 2 on W6a.
5. Cut the trace which runs between IC18-3 and IC18-13 on the copper side of the board.
6. Connect IC18-3 to IC30-3 (A11).

7. Identify the trace which connects to IC50-10 and continues through two plated through holes under IC47 and around the right edge of the board when viewed from the copper side. Cut the track on the copper side of the board between the two plated through holes. Refer to the diagram as this trace can be difficult to identify.



8. Connect IC50-10 to IC30-3 (A11).
9. On the component side of the board, cut the trace which goes to IC18-13 just before it connects to pin 13.
10. Connect to IC18-13 to IC30-19 (A13).
11. Cut the trace which connects A14 to IC50-11 on the copper side of the board just before it connects to pin 11.
12. Install a link between IC50-11 and IC30-4 (A10).
13. On the copper side of the board, cut the trace from IC15-19 (EPROM B A10).
14. Connect IC15-19 (EPROM B A10) to OV to select the 1MHz ACOS or to +5V to select the 2MHz ACOS. The CPU clock speed must be configured appropriately as well.

CHECK YOUR WORK CAREFULLY! I got a connection wrong while checking these instructions, and I wrote them. It is possible for you to make a mistake too!

15. Install the ACOS 2716 in EPROM socket B (IC15).
16. Power up the SBC-2560 board. The monitor prompt should be displayed as before. If the prompt is not displayed, power down immediately and determine the cause of the problem before proceeding.

If all connections have been made correctly it will be possible to examine the contents of the ACOS EPROM between 6000-63FF with the BINBUG 'A' command and also to execute the ACOS LOAD and DUMP commands.

SETTING ACOS PHASE

Once both the hardware and the ACOS EPROM have been installed, it is necessary to adjust the phase of both the output and the input of the interface. This must be done to ensure that tapes recorded by all ACOS systems will use the same recording phase, so that tapes recorded on one system can be read by another.

To set up the phase to the standard, proceed as follows:

1. Connect the LOAD phase strap to either sense.
2. Attempt to load the supplied ACOS format tape by inserting it in the load machine, pressing the play button, and typing 'LT' at the terminal. After about ten seconds of leader tone during which time nothing should be displayed, either numbers will be displayed on the terminal indicating that blocks of data are being read from the tape, or absolutely nothing will continue to happen. If no response is observed, the phase is most probably incorrect.
3. Abort from the load mode by pressing the BREAK key or by the resetting the computer, rewind the tape and alter the LOAD phase strap to the opposite position.
4. Try loading the tape again. This time numbers should be displayed on the screen indicating that ACOS is loading a file.
5. If you still obtain no response, check the circuit very carefully and try loading at different recorder level settings. Although ACOS is relatively immune to variation in input levels, an optimum setting does exist for each recorder. This level must be determined by trial and error.
6. Once the load phase has been set to the standard phase, the phase of the data recorded by the dump machine can be adjusted. To set the dump phase, connect the dump phase strap to one of the two possible positions and dump a file to tape. Next attempt to load this file using the load machine. If no response is obtained, change the dump phase strap to the other position and repeat the procedure. This time the dumped file should load satisfactorily.
7. The correct phase strap connections are those which allow the supplied tape to be loaded, and a tape dumped by the system to be re-loaded.

SOURCE LISTINGS

```

;*****
;*
;*      Program      :   BINBUG v6      *
;*      Version      :   v6.1          *
;*      Edit Date    :   02-08-82      *
;*
;*      Copyright MicroByte 1982      *
;*
;*****

```

```

;* This version of BINBUG supports:

```

```

;*      VHS Data DOS
;*      ACOS Cassette System
;*      DG640 VDU
;*      ETI-685 processor board

```

```

:DELE EQU H'7F'
:CR EQU H'0D'
:LF EQU H'0A'
:BS EQU H'08'
:HT EQU H'09'
:FF EQU H'0C'
:CL EQU H'0E'
:ESC EQU H'1B'
:SPAC EQU H'20'
:BLEN EQU 20      Input buffer length
:PAGE EQU H'78'   VDU memory at 7800

```

```

:      ORG      0

```

```

0000 0734 :INIT LDI,R3 HDAT-COM Cold start
0002 20 : EORZ,R0
0003 CF4400 :AINI STRA,R0 COM,R3- Clear most of scratch
0006 5B7B : BRNR,R3 AINI
0008 0477 : LDI,R0 H'77' Setup 'GOTO' branch
000A CC040A : STRA,R0 XGOT in RAM to restore
000D 041F : LDI,R0 H'1F' PSW lower.
000F CC040C : STRA,R0 XGOT+2
0012 0498 : LDI,R0 B'10011000'
0014 D433 : WRTE,R0 H'33' initialise 8255
0016 1F0355 : BCTA,UN INCC Continue initialisation

```

```

0019 02B4 :ZOUT ACON COUT

```

```

;*Breakpoint vectors indirectly through
;*this ACON

```

```

001B 012B :VEC ACON BK01

```

```

;*COMMAND HANDLER

```

```

;*Input line and jump to routine depending
;*on first character of command. DOS commands
;*W and K are also supported. Final branch is
;*indirectly through ACON in RAM which may be
;*altered to extend command table.

```

```

001D 043F :EBUG LDI,R0 A'? ' Error re-entry
001F 3F02B4 : BSTA,UN COUT
0022 75FF :MBUG CPSL H'FF' Monitor warm start
0024 3F008A : BSTA,UN CRLF
0027 042A : LDI,R0 A'* '
0029 3F02B4 : BSTA,UN COUT Output prompt
002C 3B2D : BSTR,UN LINE Input command line to buffer
002E 20 : EORZ,R0
002F CC0427 : STRA,R0 BPTR Point to start of buffer
0032 0C0413 : LODA,R0 BUFF Get command character
0035 E441 : COMI,R0 A'A '
0037 1C00B9 : BCTA,EQ ALTE ALTER memory
003A E442 : COMI,R0 A'B '
003C 1C0190 : BCTA,EQ BKPT Set BREAKPOINT
003F E443 : COMI,R0 A'C '
0041 1C0186 : BCTA,EQ CLR CLEAR breakpoint
0044 E444 : COMI,R0 A'D '
0046 1C03DB : BCTA,EQ DUMP DUMP ACOS object tape
0049 E447 : COMI,R0 A'G '
004B 1C0318 : BCTA,EQ GOTO GOTO program and execute
004E E44C : COMI,R0 A'L '
0050 1C03C2 : BCTA,EQ LOAD LOAD ACOS object tape
0053 E453 : COMI,R0 A'S '
0055 1C00EF : BCTA,EQ SREG SET register
0058 1F0341 : BCTA,UN XCMD EXTENDED COMMAND PROCESSOR

```

:*INPUT COMMAND LINE INTO INPUT BUFFER

:*DEL will delete the last character entered
 :*and output a BS to erase it on the terminal.
 :*CODE indicates type of line entered.
 :*CODE = 1 if CR
 :* 2 if LF
 :* 3 if CR and data
 :* 4 if LF and data

```
005B 07FF :LINE  LODI,R3 -1
005D CF0427 :      STRA,R3 BPTR      Initialise buffer pointer
0060 E714 :LLIN  COMI,R3 BLEN      At end of buffer?
0062 1818 :      BCTR,EQ ELIN
0064 3F0286 :      BSTA,UN CHIN      Input a character
0067 E47F :      COMI,R0 DELE      If DELETE and
0069 980D :      BCFR,EQ ALIN
006B E7FF :      COMI,R3 -1      if not at start of line
006D 1871 :      BCTR,EQ LLIN
006F 0408 :      LODI,R0 BS      then output BS and
0071 3F02B4 :      BSTA,UN COUT
0074 A701 :      SUBI,R3 1      decrement buffer pointer
0076 1B68 :      BCTR,UN LLIN
0078 E40D :ALIN  COMI,R0 CR      If return then
007A 9816 :      BCFR,EQ BLIN
007C 0501 :ELIN  LODI,R1 1      set CODE odd
007E 03 :CLIN  LODZ,R3      If buffer length zero then
007F 1A02 :      BCTR,N DLIN
0081 8502 :      ADDI,R1 2      add 2 to code
0083 CD042A :DLIN  STRA,R1 CODE
0086 CF0429 :      STRA,R3 CNT      save length of line
0089 C0 :      NOP
```

:* Output CR & LF

```
008A 040D :CRLF  LODI,R0 CR      Output Carriage Return and
008C BB99 :      ZBSR *ZOUT
008E 040A :      LODI,R0 LF      Line Feed
0090 9B99 :      ZBRR *ZOUT
```

```
0092 0502 :BLIN  LODI,R1 2
0094 E40A :      COMI,R0 LF      if LF then set CODE =2
0096 1866 :      BCTR,EQ CLIN      and return
0098 CF2413 :      STRA,R0 BUFF,R3+ else insert character in
009B 3F02B4 :      BSTA,UN COUT      buffer and echo
009E 1B40 :      BCTR,UN LLIN
```

```
00A0 C0 :      NOP
```

:*Get HEX number from input buffer and
 :*store in TEMP.

```
00A1 3F02E3 :GAST  BSTA,UN GNUM
:
00A4 CD040D :STRT  STRA,R1 TEMP
00A7 CE040E :      STRA,R2 TEMP+1
00AA 17 :      RETC,UN
```

:*Increment TEMP by R2

```
00AB 0500 :INCT  LODI,R1 0
00AD 8E040E :      ADDA,R2 TEMP+1
00B0 7708 :      PPSL WC
00B2 8D040D :      ADDA,R1 TEMP
00B5 7508 :      CPSL WC
00B7 1B6B :      BCTR,UN STRT
```

:*Access memory at specified address and
 :*if hex number is entered then replace
 :*memory contents.
 :*If command line is terminated by LF then
 :*display next memory location.

```
00B9 3B66 :ALTE  BSTR,UN GAST      Get address and store
00BB 3F0269 :LALT  BSTA,UN BOUT      Print MSB of address
00BE 0D040E :      LODA,R1 TEMP+1
00C1 3F0279 :      BSTA,UN FOUT      Print LSB of address + 3 spaces
00C4 0D840D :      LODA,R1 *TEMP
00C7 3F0279 :      BSTA,UN FOUT      Print contents of memory
00CA 3F005B :      BSTA,UN LINE      Input line
00CD 0C042A :      LODA,R0 CODE
00D0 E402 :      COMI,R0 2      if CR only
00D2 1E0022 :      BCTA,LT MBUG      then exit
00D5 1811 :      BCTR,EQ DALT      else if only LF do next
00D7 CC0411 :      STRA,R0 TEMR
00DA 3F02E3 :      BSTA,UN GNUM      Get new contents
00DD CE840D :      STRA,R2 *TEMP      and alter memory
00E0 0C0411 :      LODA,R0 TEMR
00E3 E404 :      COMI,R0 4      if terminated by CR
00E5 9C0022 :      BCFA,EQ MBUG      then exit
00E8 0601 :DALT  LODI,R2 1      else advance to
00EA 3F00AB :      BSTA,UN INCT      next location
00ED 1B4C :      BCTR,UN LALT      and repeat
```

:*DISPLAY AND SET REGISTER STACK

:*SA will display all registers

```

00EF 3F02E3 :SREG  BSTA,UN GNUM      Get register number
00F2 E608   :LSRE  COMI,R2 8      If > 8
00F4 1D015D :      BCTA,GT SEE      then print all registers
00F7 CE0411 :      STRA,R2 TEMR
00FA 0E6400 :      LODA,R0 COM,R2      Get register contents
00FD C1      :      STRZ,R1      from store area
00FE 3F0279 :      BSTA,UN FOUT      display register contents
0101 3F005B :      BSTA,UN LINE      Input command line
0104 0C042A :      LODA,R0 CODE
0107 E402   :      COMI,R0 2      If CR only
0109 1E0022 :      BCTA,LT MBUG      then exit else if LF only
010C 1815   :      BCTR,EQ CSRE      then get next register
010E CC040F :      STRA,R0 TEMQ      else
0111 3F02E3 :      BSTA,UN GNUM      get next value
0114 02      :      LODZ,R2
0115 0E0411 :      LODA,R2 TEMR
0118 CE6400 :      STRA,R0 COM,R2      and update register store
011B 0C040F :      LODA,R0 TEMQ
011E E403   :      COMI,R0 3      If line terminated by CR
0120 1C0022 :      BCTA,EQ MBUG      then exit
0123 0E0411 :CSRE  LODA,R2 TEMR      else point to
0126 8601   :      ADDI,R2 1      next register
0128 1F00F2 :      BCTA,UN LSRE      and repeat

```

:*SINGLE BREAKPOINT ROUTINE

:*Breakpoint vector branches indirectly through
:*the ACON at 1B to this routine.

```

012B CC0400 :BK01  STRA,R0 COM      Store the contents
012E 13      :      SPSL      of PSW and all registers
012F CC0408 :      STRA,R0 COM+8      on stack at 400
0132 12      :      SPSU
0133 CC0407 :      STRA,R0 COM+7
0136 7710   :      PPSL  RS
0138 CD0404 :      STRA,R1 COM+4
013B CE0405 :      STRA,R2 COM+5
013E CF0406 :      STRA,R3 COM+6
0141 7510   :      CPSL  RS
0143 CD0401 :      STRA,R1 COM+1
0146 CE0402 :      STRA,R2 COM+2
0149 CF0403 :      STRA,R3 COM+3
014C 3B1C   :      BSTR,UN CLBK      Clear breakpoint vector
014E 3F008A :      BSTA,UN CRLF
0151 0D040D :      LODA,R1 TEMP      and print address of
0154 3F0269 :      BSTA,UN BOUT      breakpoint
0157 0D040E :      LODA,R1 TEMP+1
015A 3F0279 :      BSTA,UN FOUT      followed by spaces

```

:*DISPLAY CONTENTS OF ALL REGISTERS

```

015D 06F7   :MINUS9 EQU 256-9      Counts up 9 registers
015F 0E6309 :SEE  LODI,R2 MINUS9
0162 C1      :DISP  LODA,R0 COM-MINUS9,R2
0163 3F0279 :      STRZ,R1
0166 DA77   :      BSTA,UN FOUT      Print byte + spaces
0168 9B22   :      BIRR,R2 DISP
      ZBRR  MBUG

```

:*CLEAR SINGLE BKPT

```

016A 20      :CLBK  EORZ,R0
016B CC042C :      STRA,R0 MARK      Clear BP flag
016E 0D0436 :      LODA,R1 HADR      Get address of BP,
0171 0E0437 :      LODA,R2 LADR
0174 3F00A4 :      BSTA,UN STRT      stuff in TEMP to print
0177 0C0434 :      LODA,R0 HDAT      and restore memory
017A CC840D :      STRA,R0 *TEMP      to original contents.
017D 0C0435 :      LODA,R0 LDAT
0180 0701   :      LODI,R3 1
0182 CFE40D :      STRA,R0 *TEMP,R3
0185 17      :      RETC,UN

```

:*BP clear is inhibited after reset but

:*may be cleared by re-executing BP vector

:*or G18C.

```

0186 0C042C :CLR  LODA,R0 MARK      is a BP set?
0189 1C001D :      BCTA,Z  EBUG
018C 3B5C   :      BSTR,UN CLBK
018E 9B22   :      ZBRR  MBUG

```

:*SET BREAKPOINT VECTOR AT ADDRESS SPECIFIED.

:*The contents two memory locations are saved

:*and replaced by ZBRR *1B to vector to to

:*BP service routine.

```

0190 3F00A1 :BKPT  BSTA,UN GAST      Get BP address into TEMP,
0193 CD0436 :      STRA,R1 HADR      save address
0196 CE0437 :      STRA,R2 LADR
0199 3B02   :      BSTR,UN BKST      and save memory contents.
019B 9B22   :      ZBRR  MBUG
019D 0C840D :BKST  LODA,R0 *TEMP      Save contents of memory
01A0 CC0434 :      STRA,R0 HDAT
01A3 0701   :      LODI,R3 1
01A5 0FE40D :      LODA,R0 *TEMP,R3
01A8 CC0435 :      STRA,R0 LDAT
01AB 049B   :      LODI,R0 H'9B'      and replace with ZBRR *1B.
01AD CC840D :      STRA,R0 *TEMP
01B0 049B   :      LODI,R0 VEC+128
01B2 CFE40D :      STRA,R0 *TEMP,R3
01B5 04FF   :      LODI,R0 -1
01B7 CC042C :      STRA,R0 MARK      Set BP flag
01BA 17      :      RETC,UN

```

:*ADVANCED VIDEO DRIVER ROUTINE

:*Decodes following Control characters

```
:*CR      OD      Position to left of line
:*LF      0A      Scroll one line
:*BS      08      Back one position
:*FF      0C      Clear screen and enter page mode
:*HT      09      Advance to next TAB stop
:*        0E      Clear screen
:*ESC     1B      Select scroll mode
```

:*loads cursor pointer

```
01BB 7710 :LPTR PPSL RS
01BD 0D0439 : LODA,R1 PTR
01C0 0E043A : LODA,R2 PTR+1
:*mask and store pointer
01C3 4503 :MAST ANDI,R1 3
01C5 6578 :SPT IORI,R1 PAGE
01C7 CE043A : STRA,R2 PTR+1
01CA CD0439 : STRA,R1 PTR
01CD 17 : RETC,UN
```

:*output ASCII character in R0

```
01CE 3B6B :KOUT BSTR,UN LPTR Normal entry point
01D0 E420 : COMI,R0 SPAC
01D2 1A19 : BCTR,LT WCC
01D4 CC8439 : STRA,R0 *PTR Output char
01D7 DA02 : BIRR,R2 SCUR Increment cursor
01D9 8501 : ADDI,R1 1
```

:*if bit 3 of R1 is set at this point

:*page overflow has occurred

```
:SCUR TMI,R1 4
01DB F504 : BCTA,EQ TSCR
01DD 1C0224 :
```

:*set cursor

```
:SC BSTR,UN MAST Store new
01E0 3B61 : LODI,R0 H'80'
01E2 0480 : EORA,R0 *PTR
01E4 2C8439 : STRA,R0 *PTR
01E7 CC8439 : RRB CPSL RS
01EA 7510 : RETC,UN
01EC 17 :
```

:*decodes control characters

```
01ED E40D :WCC COMI,R0 CR If CR
01EF 9802 : BCFR,EQ NCR
01F1 46C0 : ANDI,R2 H'C0' then set cursor left.
01F3 E40A :NCR COMI,R0 LF If LF
01F5 9808 : BCFR,EQ NLF
01F7 8640 : ADDI,R2 64 then advance to next line.
01F9 7708 : PPSL WC
01FB 8500 : ADDI,R1 0
01FD 7508 : CPSL WC
01FF E40C :NLF COMI,R0 FF If FF
0201 1838 : BCTR,EQ PAGC then clear screen.
0203 E408 : COMI,R0 BS If BS
0205 9806 : BCFR,EQ NBS
```

```
0207 A601 : SUBI,R2 1 then backspace but not
0209 F63F : TMI,R2 63 to previous line.
020B 185D : BCTR,EQ RRB
020D E409 :NBS COMI,R0 HT If HT
020F 9804 : BCFR,EQ NHT
0211 B60B : ADDI,R2 8 then expand to spaces.
0213 46F8 : ANDI,R2 H'F8'
0215 E40E :NHT COMI,R0 CL
0217 182A : BCTR,EQ PGC1
:*entry point to erase old cursor
0219 0480 : LODI,R0 H'80'
021B 2C8439 : EORA,R0 *PTR
021E CC8439 : STRA,R0 *PTR
0221 1F01DB : BCTA,UN SCUR
:*test if scroll flag set
0224 0D0433 :TSCR LODA,R1 SCRF
0227 9E0374 : BCFA,N SCRL
022A 3F0286 : BSTA,UN CHIN Pause for prompt
022D 7710 : PPSL RS
022F E41B : COMI,R0 ESC If ESC then
0231 1C0374 : BCTA,EQ SCRL enter scrolling mode.
0234 0500 : LODI,R1 0 If CR then scroll
0236 E40D : COMI,R0 CR one line, else clear screen
0238 1C037C : BCTA,EQ SCRL and display next page.
:*subroutine to clear page
023B 0D0433 :PAGC LODA,R1 SCRF
023E 6580 : IORI,R1 H'80' Set page mode flag
0240 CD0433 : STRA,R1 SCRF
0243 20 :PGC1 EORZ,R0 Home cursor
0244 C1 : STRZ,R1
0245 C2 : STRZ,R2
0246 3F01C5 :KLR BSTA,UN SPT and clear screen.
0249 0420 : LODI,R0 SPAC
024B CEE439 :KL STRA,R0 *PTR,R2
024E DA7B : BIRR,R2 KL
0250 8501 : ADDI,R1 1
0252 4507 : ANDI,R1 7
0254 9870 : BCFR,EQ KLR
0256 1F01E0 : BCTA,UN SC
```

```

0259 30313233 : ANSI      ORG      H'259'
          : DATA      A'0123456789ABCDEF' ASCII list of HEX digit
          : *Print R1 as two HEX. digits.

0269 01      : BOUT      LODZ,R1
026A 50      :           RRR,R0
026B 50      :           RRR,R0
026C 50      :           RRR,R0
026D 50      :           RRR,R0
026E 3B01    :           BSTR,UN CON
0270 01      :           LODZ,R1
0271 440F    : CON      ANDI,R0 H'F'
0273 0C6259  :           LODA,R0 ANSI,R0
0276 1F02B4  :           BCTA,UN COUT      and return

          : *Print byte + 3 spaces
0279 3B6E    : FOUT      BSTR,UN BOUT
027B 0703    : FORM      LODI,R3 3
027D 0420    : AGAP      LODI,R0 SPAC
027F 3F02B4  :           BSTA,UN COUT
0282 FB79    :           BDRR,R3 AGAP
0284 17      :           RETC,UN

0285 C0      :           NOP

          : *Keyboard input routines
0286 0C042B  : CHIN      LODA,R0 KBFLAG      keyboard flag
0289 9809    :           BCFR,Z SIN      Serial input
028B 5435    : PORTIN     REDE,R0 H'35'      keyboard port
028D 187C    :           BCTR,Z PORTIN
028F D436    :           WRTE,R0 H'36'      reset keyboard
0291 447F    :           ANDI,R0 H'7F'      Mask off parity bit
0293 17      :           RETC,UN

          : *Inputs character from keyboard through SENSE
0294 7710    : SIN        PPSL RS
0296 0500    :           LODI,R1 0
0298 0608    :           LODI,R2 8      No. data bits
029A 12      : CHI        SPSU      Look for start bit
029B 1A7D    :           BCTR,LT CHI
029D 3B26    :           BSTR,UN DLY      1
029F 12      :           SPSU
02A0 1A78    :           BCTR,LT CHI      False start bit
02A2 3B1B    : BTIN      BSTR,UN DLAY      Wait till middle of bit
02A4 12      :           SPSU      Read SENSE
02A5 4480    :           ANDI,R0 H'80'      Mask off data bit
02A7 51      :           RRR,R1
02A8 61      :           IORZ,R1      Merge into one byte
02A9 C1      :           STRZ,R1
02AA FA76    :           BDRR,R2 BTIN      Do it 8 times
02AC 3B11    :           BSTR,UN DLAY      Wait for stop bit
02AE 01      :           LODZ,R1
02AF 7510    :           CPSL RS
02B1 447F    :           ANDI,R0 H'7F'      Mask off parity bit
02B3 17      :           RETC,UN

```

```

: *OUTPUT CHARACTER in R0 to VDU
: *or to external routine

```

```

02B4 7710    : COUT      PPSL RS      PIPBUG COMPATIBLE ENTRY
02B6 0D0409  :           LODA,R1 OP      If OP not zero then
02B9 1C01CE  :           BCTA,Z KOUT
02BC 1F843B  :           BCTA,UN *EOUT      use external routine.

```

```

: *SERIAL I/O TIMING DELAY LOOPS

```

```

: *      1MHz clock      -      150  300  600 1200 2400 Baud
: *      2MHz clock      -      300  600 1200 2400 4800 Baud

```

```

02BF 0C042D  : DLAY      LODA,R0 DLY1      DD  6E  36  1A  0C
02C2 C0      :           NOP
02C3 F87D    :           BDRR,R0 $-1
02C5 0C042E  : DLY      LODA,R0 DLY2      DA  6B  34  18  0A
02C8 C0      :           NOP
02C9 F87D    :           BDRR,R0 $-1
02CB 17      :           RETC,UN

```

```

: * Initial delay constants

```

```

02CC DA      : DLI      DATA H'DA'      150/300 baud
02CD 6B      :           DATA H'6B'      300/600 baud
02CE 18      :           DATA H'18'      1200/2400 baud

```

:*Convert HEX digit in R0 to BINARY in R3

```
02CF 0710 :LKUP LODI,R3 16
02D1 EF4259 :ALKU COMA,R0 ANSI,R3-
02D4 14 : RETC,EQ
02D5 E701 : COMI,R3 1
02D7 9A78 : BCFR,LT ALKU
02D9 9B1D : ZBRR EBUG
```

:*Get HEX. number from input buffer into R1,R2.

```
02DB 1B06 : BCTR,UN GNUM PIPBUG COMPATIBLE ENTRY

02DD 0C042A :DNUM LODA,R0 CODE If code = 1
02E0 1808 : BCTR,Z LNUM then doing second byte.
02E2 17 : RETC,UN
02E3 20 :GNUM EORZ,R0
02E4 C1 : STRZ,R1
02E5 C2 : STRZ,R2
02E6 C3 : STRZ,R3
02E7 CC042A : STRA,R0 CODE
02EA 0F0427 :LNUM LODA,R3 BPTR
02ED EF0429 : COMA,R3 CNT End of line?
02F0 14 : RETC,EQ
02F1 0F2413 : LODA,R0 BUFF,R3+ Get character from buffer
02F4 CF0427 : STRA,R3 BPTR and update pointer.
02F7 E420 : COMI,R0 SPAC
02F9 1862 : BCTR,EQ DNUM
02FB 3B52 : BSTR,UN LKUP Convert to binary
02FD 040F : LODI,R0 H'0F'
02FF D2 : RRL,R2
0300 D2 : RRL,R2
0301 D2 : RRL,R2
0302 D2 : RRL,R2
0303 42 : ANDZ,R2
0304 D1 : RRL,R1
0305 D1 : RRL,R1
0306 D1 : RRL,R1
0307 D1 : RRL,R1
0308 45F0 : ANDI,R1 H'F0'
030A 46F0 : ANDI,R2 H'F0'
030C 61 : IORZ,R1
030D C1 : STRZ,R1
030E 03 : LODZ,R3
030F 62 : IORZ,R2
0310 C2 : STRZ,R2
0311 0401 : LODI,R0 1
0313 CC042A : STRA,R0 CODE
0316 1B52 : BCTR,UN LNUM
```

:*GOTO program and execute.

:*Restore all registers first.

```
0318 3F00A1 :GOTO BSTA,UN GAST Get execute address
031B 0C0407 : LODA,R0 COM+7 Restore all registers
031E 92 : LPSU
031F 0D0401 : LODA,R1 COM+1
0322 0E0402 : LODA,R2 COM+2
0325 0F0403 : LODA,R3 COM+3
0328 7710 : PPSL RS
032A 0D0404 : LODA,R1 COM+4
032D 0E0405 : LODA,R2 COM+5
0330 0F0406 : LODA,R3 COM+6
0333 0C0408 : LODA,R0 COM+8 get PSW lower and store
0336 CC040B : STRA,R0 XGOT+1 in RAM. Self modifying code
0339 0C0400 : LODA,R0 COM is the only way to
033C 75FF : CPSL H'FF' restore PSL correctly.
033E 1F040A : BCTA,UN XGOT
```

:* EXTENDED COMMAND PROCESSOR

:*To check for existence of of VHS data DOS

:*and implement W and K commands.

```
0341 0990 :XCMD LODR,R1 *DOS does DOS exist at 6800?
0343 E51F : COMI,R1 H'1F'
0345 9809 : BCFR,EQ EXTCMD
0347 E44B : COMI,R0 A'K'
0349 1888 : BCTR,EQ *DOS DOS cold start
034B E457 : COMI,R0 A'W'
034D 1CE803 : BCTA,EQ *H'6803' DOS warm start
0350 1F8431 :EXTCMD BCTA,UN *COMD external commands

0353 6800 :DOS ACON H'6800' pointer to DOS
```

:* Continue initialisation

```
0355 041D :INCQ LODI,R0 >EBUG Non-extend command table
0357 CC0432 : STRA,R0 COMD+1
035A 5532 : REDE,R1 H'32' keyboard option
035C 0602 : LODI,R2 2
035E D1 : RRL,R1
035F 1A03 : BCTR,N INCQ parallel keyboard
0361 CE042B : STRA,R2 KBFLAG serial keyboard
0364 D1 :INCQ RRL,R1
0365 D1 : RRL,R1
0366 D1 : RRL,R1
0367 4503 : ANDI,R1 3
0369 0D62CC : LODA,R0 DL1,R1
036C CE642C :INCRAT STRA,R0 DLY1-1,R2
036F 82 : ADDZ,R2 calculate 2nd delay constant
0370 FA7A : BDRR,R2 INCRAT
0372 9B22 : ZBRR MBUG
```

:*SCROLLING ROUTINE

```

0374 0D0433 :SCRL  LODA,R1 SCRF
0377 451F : ANDI,R1 H'1F'
0379 CD0433 : STRA,R1 SCRF

:*scroll 1 line
037C 2510 :SCRL  EORI,R1 H'10'
037E 20 : EORZ,R0
037F CC043A : STRA,R0 PTR+1
0382 0440 : LODI,R0 64
0384 CC0430 : STRA,R0 FROM+1

0387 0600 :SC1  LODI,R2 0
0389 7708 : PPSL  WC
038B 51 : RRR,R1
038C 52 : RRR,R2
038D 51 : RRR,R1
038E 52 : RRR,R2
038F 7508 : CPSL  WC
0391 6578 : IORI,R1 PAGE

0393 CD0439 :SC2  STRA,R1 PTR
0396 CD042F : STRA,R1 FROM
0399 0EE42F :SC3  LODA,R0 *FROM,R2
039C CEE439 : STRA,R0 *PTR,R2
039F DA78 : BIRR,R2 SC3
03A1 F503 : TMI,R1 3
03A3 1802 : BCTR,EQ CLRB
03A5 D96C : BIRR,R1 SC2

:*clear bottom line
03A7 06C0 :CLRB  LODI,R2 H'C0'
03A9 0420 : LODI,R0 SPAC
03AB CEE439 :SSPC  STRA,R0 *PTR,R2
03AE DA78 : BIRR,R2 SSPC
03B0 F504 : TMI,R1 4
03B2 9807 : BCFR,EQ SCND
03B4 0D0433 : LODA,R1 SCRF
03B7 450F : ANDI,R1 H'F'
03B9 1B4C : BCTR,UN SC1
03BB 0503 :SCND  LODI,R1 3
03BD 06C0 : LODI,R2 H'C0'
03BF 1F01E0 : BCTA,UN SC

```

:*ACOS LOAD/DUMP ROUTINES

```

:TCTL  EQU  H'446'
:EXF  EQU  H'44B'
:TYPE  EQU  H'447'
:NBT  EQU  H'45C'
:EXAD  EQU  H'45E'
:ADD  EQU  H'45A'
:OBC  EQU  H'460'

:  ORG  H'3C2'
03C2 0500 :LOAD  LODI,R1 0
03C4 CD0446 : STRA,R1 TCTL
03C7 CD0447 : STRA,R1 TYPE
03CA 3F6000 : BSTA,UN H'6000'  ACOS LOAD subroutine
03CD 0C044B : LODA,R0 EXF
03D0 E4FF : COMI,R0 H'FF'
03D2 9C0022 : BCFA,EQ MBUG
03D5 3F008A : BSTA,UN CRLF
03D8 1F845E : BCTA,UN *EXAD  Jump start program

03DB 20 :DUMP  EORZ,R0
03DC CC0446 : STRA,R0 TCTL
03DF CC0447 : STRA,R0 TYPE
03E2 3F6003 : BSTA,UN H'6003'  ACOS DUMP subroutine
03E5 9B22 : ZBRR  MBUG

:*Subroutine to transfer memory to buffer
03E7 0D045C :TSB  LODA,R1 NBT
03EA 0DC45A :LPX  LODA,R0 *ADD,R1-
03ED CD6460 : STRA,R0 OBC,R1
03F0 5978 : BRNR,R1 LPX
03F2 17 : RETC,UN

:*Subroutine to transfer buffer to memory.
03F3 0D045C :TFR0  LODA,R1 NBT
03F6 0D4460 :LPJ  LODA,R0 OBC,R1-
03F9 CDE45A : STRA,R0 *ADD,R1
03FC 5978 : BRNR,R1 LPJ
03FE 17 : RETC,UN

03FF 61 : DATA  H'61'  Version

```

:*BINBUG SCRATCH AREA

: ORG H'400'

```

0400      :COM      RES      9      Register & PSW stack
0409      :OP       RES      1      Output switch
040A      :XGOT     RES      3      PSL restore instr.
040D      :TEMP     RES      2      16 bit address store
040F      :TEMQ     RES      2
0411      :TEMR     RES      1
0412      :         RES      1
0413      :BUFF     RES      BLEN   Input buffer
0427      :BPTR     RES      1      Buffer pointer
0428      :         RES      1
0429      :CNT      RES      1
042A      :CODE     RES      1

042B      :KBFLAG   RES      1      Keyboard flag; 0 => Parallel
042C      :MARK     RES      1      BP flag. FF => BP set.

:* Serial I/O delay constants
:*      1MHz clock      -   150   300   600 1200 2400 Baud
:*      2MHz clock      -   300   600 1200 2400 4800 Baud
042D      :DLY1     RES      1      DD   6E   36   1A   0C
042E      :DLY2     RES      1      DA   6B   34   18   0A

042F      :FROM     RES      2

:* Indirect pointer to extended command processor
0431      :CMD      RES      2      Normally 001D (EBUG)

:* Scroll flag - pxxfnnnn
:*      p          - 1 page mode    / 0 - scroll mode
:*      f          - 1 fast scroll   / 0 - normal scroll
:*      nnnn       - number of unscrolled lines at top
0433      :SCRF     RES      1      Scroll flag

:*RAM not cleared from here on
0434      :HDAT     RES      1      Bytes which were replaced
0435      :LDAT     RES      1      by BP vector
0436      :HADR     RES      1      BP address
0437      :LADR     RES      1

0438      :         RES      1
0439      :PTR      RES      2      current cursor location

043B      :EOUT     RES      2      indirect to external COUT

```

```

;*****
;*
;*      Program      :   BINBUG v7
;*      Version       :   v7.1
;*      Edit Date    :   17-08-82
;*
;*      Copyright MicroByte 1982
;*
;*****

```

```

;* This version of BINBUG supports:

```

```

;*      VHS Data DOS
;*      ACOS Cassette System
;*      Serial VDU
;*      ETI-685 processor board

```

```

:DELE EQU H'7F'
:CR EQU H'0D'
:LF EQU H'0A'
:BS EQU H'08'
:SPAC EQU H'20'
:BLEN EQU 20          Input buffer length

```

```

:      ORG 0

```

```

0000 7640 :INIT PPSU FLAG
0002 0734 : LODI,R3 HDAT-COM Cold start
0004 20 : EORZ,R0
0005 CF4400 :AINI STRA,R0 COM,R3- Clear most of scratch
0008 5B7B : BRNR,R3 AINI
000A 0477 : LODI,R0 H'77' Setup 'GOTO' branch
000C CC040A : STRA,R0 XGOT in RAM to restore
000F 041F : LODI,R0 H'1F' PSW lower.
0011 CC040C : STRA,R0 XGOT+2
0014 0440 : LODI,R0 FLAG ensure FLAG set on GOTO
0016 1F0355 : BCTA,UN INCC Continue initialisation

```

```

0019 02B4 :ZOUT ACON COUT

```

```

;*Breakpoint vectors indirectly through
;*this ACON

```

```

001B 012B :VEC ACON BK01

```

```

;*COMMAND HANDLER

```

```

;*Input line and jump to routine depending
;*on first character of command. DOS commands
;*W and K are also supported. Final branch is
;*indirectly through ACON in RAM which may be
;*altered to extend command table.

```

```

001D 043F :EBUG LODI,R0 A'? ' Error re-entry
001F 3F02B4 : BSTA,UN COUT
0022 75FF :MBUG CPSL H'FF' Monitor warm start
0024 3F008A : BSTA,UN CRLF
0027 042A : LODI,R0 A'* '
0029 3F02B4 : BSTA,UN COUT Output prompt
002C 3B2D : BSTR,UN LINE Input command line to buffer
002E 20 : EORZ,R0
002F CC0427 : STRA,R0 BPTR Point to start of buffer
0032 0C0413 : LODA,R0 BUFF Get command character
0035 E441 : COMI,R0 A'A '
0037 1C00B9 : BCTA,EQ ALTE ALTER memory
003A E442 : COMI,R0 A'B '
003C 1C0190 : BCTA,EQ BKPT Set BREAKPOINT
003F E443 : COMI,R0 A'C '
0041 1C0186 : BCTA,EQ CLR CLEAR breakpoint
0044 E444 : COMI,R0 A'D '
0046 1C03DB : BCTA,EQ DUMP DUMP ACOS object tape
0049 E447 : COMI,R0 A'G '
004B 1C0318 : BCTA,EQ GOTO GOTO program and execute
004E E44C : COMI,R0 A'L '
0050 1C03C2 : BCTA,EQ LOAD LOAD ACOS object tape
0053 E453 : COMI,R0 A'S '
0055 1C00EF : BCTA,EQ SREG SET register
0058 1F0341 : BCTA,UN XCMD EXTENDED COMMAND PROCESSOR

```

```

      :*Get HEX number from input buffer and
      :*store in TEMP.

00A1 3F02E3 :GAST   BSTA,UN GNUM

      :*Store R1, R2 in TEMP

00A4 CD040D :STRT   STRA,R1 TEMP
00A7 CE040E :      STRA,R2 TEMP+1
00AA 17      :      RETC,UN

      :*Increment TEMP by R2

00AB 0500   :INCT   LODI,R1 0
00AD 8E040E :      ADDA,R2 TEMP+1
00B0 7708   :      PPSL   WC
00B2 8D040D :      ADDA,R1 TEMP
00B5 7508   :      CPSL   WC
00B7 1B6B   :      BCTR,UN STRT

      :*Access memory at specified address and
      :*if hex number is entered then replace
      :*memory contents.
      :*If command line is terminated by LF then
      :*display next memory location.

00B9 3B66   :ALTE   BSTR,UN GAST      Get address and store
00BB 3F0269 :LALT   BSTA,UN BOUT      Print MSB of address
00BE 0D040E :      LODA,R1 TEMP+1
00C1 3F0279 :      BSTA,UN FOUT      Print LSB of address + 3 space
00C4 0D840D :      LODA,R1 *TEMP
00C7 3F0279 :      BSTA,UN FOUT      Print contents of memory
00CA 3F005B :      BSTA,UN LINE      Input line
00CD 0C042A :      LODA,R0 CODE
00D0 E402   :      COMI,R0 2      if CR only
00D2 1E0022 :      BCTA,LT MBUG      then exit
00D5 1811   :      BCTR,EQ DALT    else if only LF do next
00D7 CC0411 :      STRA,R0 TEMR
00DA 3F02E3 :      BSTA,UN GNUM      Get new contents
00DD CE840D :      STRA,R2 *TEMP      and alter memory
00E0 0C0411 :      LODA,R0 TEMR
00E3 E404   :      COMI,R0 4      if terminated by CR
00E5 9C0022 :      BCFA,EQ MBUG      then exit
00E8 0601   :DALT   LODI,R2 1      else advance to
00EA 3F00AB :      BSTA,UN INCT      next location
00ED 1B4C   :      BCTR,UN LALT    and repeat

```

```

      :*INPUT COMMAND LINE INTO INPUT BUFFER

```

```

      :*DEL will delete the last character entered
      :*and output a BS to erase it on the terminal.
      :*CODE indicates type of line entered.
      :*CODE = 1 if CR
      :*        2 if LF
      :*        3 if CR and data
      :*        4 if LF and data

```

```

005B 07FF   :LINE   LODI,R3 -1
005D CF0427 :      STRA,R3 BPTR      Initialise buffer pointer
0060 E714   :LLIN   COMI,R3 BLEN      At end of buffer?
0062 1818   :      BCTR,EQ ELIN
0064 3F0286 :      BSTA,UN CHIN      Input a character
0067 E47F   :      COMI,R0 DELE      If DELETE and
0069 980D   :      BCFR,EQ ALIN
006B E7FF   :      COMI,R3 -1      if not at start of line
006D 1871   :      BCTR,EQ LLIN
006F 0408   :      LODI,R0 BS      then output BS and
0071 3F02B4 :      BSTA,UN COUT      decrement buffer pointer
0074 A701   :      SUBI,R3 1
0076 1B68   :      BCTR,UN LLIN
0078 E40D   :ALIN   COMI,R0 CR      If return then
007A 9816   :      BCFR,EQ BLIN
007C 0501   :ELIN   LODI,R1 1      set CODE odd
007E 03      :CLIN   LODZ,R3      If buffer length zero then
007F 1A02   :      BCTR,N DLIN
0081 8502   :      ADDI,R1 2      add 2 to code
0083 CD042A :DLIN   STRA,R1 CODE
0086 CF0429 :      STRA,R3 CNT      save length of line
0089 C0      :      NOP

      :* Output CR & LF

008A 040D   :CRLF   LODI,R0 CR      Output Carriage Return and
008C BB99   :      ZBSR   *ZOUT
008E 040A   :      LODI,R0 LF      Line Feed
0090 9B99   :      ZBRR   *ZOUT

0092 0502   :BLIN   LODI,R1 2
0094 E40A   :      COMI,R0 LF      if LF then set CODE =2
0096 1866   :      BCTR,EQ CLIN      and return
0098 CF2413 :      STRA,R0 BUFF,R3+  else insert character in
009B 3F02B4 :      BSTA,UN COUT      buffer and echo
009E 1B40   :      BCTR,UN LLIN

00A0 C0      :      NOP

```

:*DISPLAY AND SET REGISTER STACK
 :*SA will display all registers

```

00EF 3F02E3 :SREG  BSTA,UN GNUM      Get register number
00F2 E608   :LSRE  COMI,R2 8        If > 8
00F4 1D015D :       BCTA,GT SEE      then print all registers
00F7 CE0411 :       STRA,R0 TEMR
00FA 0E6400 :       LODA,R0 COM,R2      Get register contents
00FD C1      :       STRZ,R1      from store area
00FE 3F0279 :       BSTA,UN FOUT      display register contents
0101 3F005B :       BSTA,UN LINE      Input command line
0104 0C042A :       LODA,R0 CODE
0107 E402   :       COMI,R0 2        If CR only
0109 1E0022 :       BCTA,LT MBUG      then exit else if LF only
010C 1815   :       BCTR,EQ CSRE      then get next register
010E CC040F :       STRA,R0 TEMQ      else
0111 3F02E3 :       BSTA,UN GNUM      get next value
0114 02      :       LODZ,R2
0115 0E0411 :       LODA,R2 TEMR
0118 CE6400 :       STRA,R0 COM,R2    and update register store
011B 0C040F :       LODA,R0 TEMQ
011E E403   :       COMI,R0 3        If line terminated by CR
0120 1C0022 :       BCTA,EQ MBUG      then exit
0123 0E0411 :CSRE  LODA,R2 TEMR      else point to
0126 8601   :       ADDI,R2 1        next register
0128 1F00F2 :       BCTA,UN LSRE      and repeat
  
```

:*SINGLE BREAKPOINT ROUTINE

:*Breakpoint vector branches indirectly through
 :*the ACON at 1B to this routine.

```

012B CC0400 :BK01  STRA,R0 COM      Store the contents
012E 13     :       SPSL          of PSW and all registers
012F CC0408 :       STRA,R0 COM+8    on stack at 400
0132 12     :       SPSU
0133 CC0407 :       STRA,R0 COM+7
0136 7710   :       PPSL  RS
0138 CD0404 :       STRA,R1 COM+4
013D CE0405 :       STRA,R2 COM+5
013E CF0406 :       STRA,R3 COM+6
0141 7510   :       CPSL  RS
0143 CD0401 :       STRA,R1 COM+1
0146 CE0402 :       STRA,R2 COM+2
0149 CF0403 :       STRA,R3 COM+3
014C 3B1C   :       BSTR,UN CLBK    Clear breakpoint vector
014E 3F008A :       BSTA,UN CRLF
0151 0D040D :       LODA,R1 TEMP      and print address of
0154 3F0269 :       BSTA,UN BOUT      breakpoint
0157 0D040E :       LODA,R1 TEMP+1
015A 3F0279 :       BSTA,UN FOUT      followed by spaces
  
```

:*DISPLAY CONTENTS OF ALL REGISTERS

```

015D 06F7   :MINUS9 EQU 256-9      Counts up 9 registers
015F 0E6309 :SEE  LODI,R2 MINUS9
0162 C1      :DISP  LODA,R0 COM-MINUS9,R2
0163 3F0279 :       STRZ,R1
0166 DA77   :       BSTA,UN FOUT      Print byte + spaces
0168 9B22   :       BIRR,R2 DISP
          :       ZBRR  MBUG
  
```

:*CLEAR SINGLE BKPT

```

016A 20     :CLBK  EORZ,R0
016B CC042C :       STRA,R0 MARK      Clear BP flag
016E 0D0436 :       LODA,R1 HADR      Get address of BP,
0171 0E0437 :       LODA,R2 LADR
0174 3F00A4 :       BSTA,UN STRT      stuff in TEMP to print
0177 0C0434 :       LODA,R0 HDAT      and restore memory
017A CC840D :       STRA,R0 *TEMP      to original contents.
017D 0C0435 :       LODA,R0 LDAT
0180 0701   :       LODI,R3 1
0182 CFE40D :       STRA,R0 *TEMP,R3
0185 17     :       RETC,UN
  
```

:*BP clear is inhibited after reset but
 :*may be cleared by re-executing BP vector
 :*or G18C.

```

0186 0C042C :CLR  LODA,R0 MARK      is a BP set?
0189 1C001D :       BCTA,Z  EBUG
018C 3B5C   :       BSTR,UN CLBK
018E 9B22   :       ZBRR  MBUG
  
```

:*SET BREAKPOINT VECTOR AT ADDRESS SPECIFIED.
 :*The contents two memory locations are saved
 :*and replaced by ZBRR *1B to vector to to
 :*BP service routine.

```

0190 3F00A1 :BKPT  BSTA,UN GAST      Get BP address into TEMP,
0193 CD0436 :       STRA,R1 HADR      save address
0196 CE0437 :       STRA,R2 LADR
0199 3B02   :       BSTR,UN BKST      and save memory contents.
019B 9B22   :       ZBRR  MBUG

019D 0C840D :BKST  LODA,R0 *TEMP      Save contents of memory
01A0 CC0434 :       STRA,R0 HDAT
01A3 0701   :       LODI,R3 1
01A5 0FE40D :       LODA,R0 *TEMP,R3
01A8 CC0435 :       STRA,R0 LDAT
01AB 049B   :       LODI,R0 H'9B'      and replace with ZBRR *1B.
01AD CC840D :       STRA,R0 *TEMP
01B0 049B   :       LODI,R0 VEC+128
01B2 CFE40D :       STRA,R0 *TEMP,R3
01B5 04FF   :       LODI,R0 -1
01B7 CC042C :       STRA,R0 MARK      Set BP flag
01BA 17     :       RETC,UN
  
```

:*SERIAL OUTPUT ROUTINE

```

:*Output character in R0 as serial asynchronous
:*character through the FLAG terminal.
:*Contents of OUTF control pad characters after CR/LF
:STF EQU H'80' bit 7 = 1 if only 1 stop bit
:CRF EQU H'40' set if pad after CR
:LFF EQU H'20' set if pad after LF
:* bit 0-4 no of pad characters

```

```

01BB 7710 :KOUT PPSL RS
01BD 7508 : CPSL WC
01BF 7640 : PPSU FLAG
01C1 C2 : STRZ,R2
01C2 0508 : LODI,R1 8 No. Data bits to output
01C4 3F02BF : BSTA,UN DLAY output 1 stop bit
01C7 0C0433 : LODA,R0 OUTF
01CA 8E02BF : BSFA,N DLAY second stop bit?
01CD 7440 : CPSU FLAG
01CF 3F02BF :BTOU BSTA,UN DLAY
01D2 52 : RRR,R2
01D3 1A04 : BCTR,LT B1
01D5 7440 : CPSU FLAG Zero bit
01D7 1B04 : BCTR,UN B0
01D9 7640 :B1 PPSU FLAG One bit
01DB 1B00 : BCTR,UN B0
01DD F970 :B0 BDRR,R1 BTOU
01DF 3F02BF : BSTA,UN DLAY
01E2 7640 : PPSU FLAG

```

```

:*Insert optional stop bits after CR and/or LF
:*for printer or slow VDU

```

```

01E4 0D0433 : LODA,R1 OUTF
01E7 E60D : COMI,R2 CR
01E9 9804 : BCFR,EQ TLF
01EB F540 : TMI,R1 CRF test if pad after CR
01ED 1808 : BCTR,EQ PAD
01EF E60A :TLF COMI,R2 LF
01F1 980F : BCFR,EQ ZEND
01F3 F520 : TMI,R1 LFF test if pad after LF
01F5 980B : BCFR,EQ ZEND
01F7 451F :PAD ANDI,R1 H'1F' mask off no of pad characters
01F9 060B :PAD1 LODI,R2 11
01FB 3F02BF :NULL BSTA,UN DLAY
01FE FA7B : BDRR,R2 NULL
0200 F977 : BDRR,R1 PAD1
0202 7510 :ZEND CPSL RS
0204 17 : RETC,UN

```

:* Serial I/O delay & pad constants

```

0205 DD6E1A0C :DL1 DATA H'DD,6E,1A,0C' 150,300,1200,2400 baud
0209 DA6B1808 :DL2 DATA H'DA,6B,18,08'
020D E2E26464 :DPC DATA H'E2,E2,64,64' Pad characters

```

```

0259 30313233 : ANSI ORG H'259'
DATA A'0123456789ABCDEF' ASCII list of HEX digit
:*Print R1 as two HEX. digits.

```

```

0269 01 :BOUT LODZ,R1
026A 50 : RRR,R0
026B 50 : RRR,R0
026C 50 : RRR,R0
026D 50 : RRR,R0
026E 3B01 : BSTR,UN CON
0270 01 : LODZ,R1
0271 440F :CON ANDI,R0 H'F'
0273 0C6259 : LODA,R0 ANSI,R0
0276 1F02B4 : BCTA,UN COUT and return

```

:*Print byte + 3 spaces

```

0279 3B6E :FOUT BSTR,UN BOUT
027B 0703 :FORM LODI,R3 3
027D 0420 :AGAP LODI,R0 SPAC
027F 3F02B4 : BSTA,UN COUT
0282 FB79 : BDRR,R3 AGAP
0284 17 : RETC,UN

```

```

0285 C0 : NOP

```

:*Keyboard input routines

```

0286 0C042B :CHIN LODA,R0 KBFLAG keyboard flag
0289 9809 : BCFR,Z SIN Serial input
028B 5435 :PORTIN REDE,R0 H'35' keyboard port
028D 187C : BCTR,Z PORTIN
028F D436 : WRTE,R0 H'36' reset keyboard
0291 447F : ANDI,R0 H'7F' Mask off parity bit
0293 17 : RETC,UN

```

:*Inputs character from keyboard through SENSE

```

0294 7710 :SIN PPSL RS
0296 0500 : LODI,R1 0
0298 0608 : LODI,R2 8 No. data bits
029A 12 :CHI SPSU Look for start bit
029B 1A7D : BCTR,LT CHI
029D 3B26 : BSTR,UN DLY 1
029F 12 : SPSU
02A0 1A78 : BCTR,LT CHI False start bit
02A2 3B1B :BTIN BSTR,UN DLAY Wait till middle of bit
02A4 12 : SPSU Read SENSE
02A5 4480 : ANDI,R0 H'80' Mask off data bit
02A7 51 : RRR,R1
02A8 61 : IORZ,R1 Merge into one byte
02A9 C1 : STRZ,R1
02AA FA76 : BDRR,R2 BTIN Do it 8 times
02AC 3B11 : BSTR,UN DLAY Wait for stop bit
02AE 01 : LODZ,R1
02AF 7510 : CPSL RS
02B1 447F : ANDI,R0 H'7F' Mask off parity bit
02B3 17 : RETC,UN

```

:*OUTPUT CHARACTER in R0 to VDU
:*or to external routine

```
02B4 7710 :COUT PPSL RS PIPBUG COMPATIBLE ENTRY
02B6 0D0409 : LODA,R1 OP If OP not zero then
02B9 1C01BB : BCTA,Z KOUT
02BC 1F843B : BCTA,UN *EOUT use external routine.
```

:*SERIAL I/O TIMING DELAY LOOPS

```
:* 1MHz clock - 150 300 600 1200 2400 Baud
:* 2MHz clock - 300 600 1200 2400 4800 Baud
```

```
02BF 0C042D :DLAY LODA,R0 DLY1 DD 6E 36 1A 0C
02C2 C0 : NOP
02C3 F87D : BDRR,R0 $-1
02C5 0C042E :DLY LODA,R0 DLY2 DA 6B 34 18 0A
02C8 C0 : NOP
02C9 F87D : BDRR,R0 $-1
02CB 17 : RETC,UN
```

```
02CC 000000 : DATA 0,0,0
```

:*Convert HEX digit in R0 to BINARY in R3

```
02CF 0710 :LKUP LODI,R3 16
02D1 EF4259 :ALKU COMA,R0 ANSI,R3-
02D4 14 : RETC,EQ
02D5 E701 : COMI,R3 1
02D7 9A78 : BCFR,LT ALKU
02D9 9B1D : ZBRR EBUG
```

:*Get HEX. number from input buffer into R1,R2.

```
02DB 1B06 : BCTR,UN GNUM PIPBUG COMPATIBLE ENTRY

02DD 0C042A :DNUM LODA,R0 CODE If code = 1
02E0 1B08 : BCTR,Z LNUM then doing second byte.
02E2 17 : RETC,UN
02E3 20 :GNUM EORZ,R0
02E4 C1 : STRZ,R1
02E5 C2 : STRZ,R2
02E6 C3 : STRZ,R3
02E7 CC042A : STRA,R0 CODE
02EA 0F0427 :LNUM LODA,R3 BPTR
02ED EF0429 : COMA,R3 CNT End of line?
02F0 14 : RETC,EQ
02F1 0F2413 : LODA,R0 BUFF,R3+ Get character from buffer
02F4 CF0427 : STRA,R3 BPTR and update pointer.
02F7 E420 : COMI,R0 SPAC
02F9 1B62 : BCTR,EQ DNUM
02FB 3B52 : BSTR,UN LKUP
02FD 040F : LODI,R0 H'0F' Convert to binary
02FF D2 : RRL,R2
0300 D2 : RRL,R2
0301 D2 : RRL,R2
0302 D2 : RRL,R2
0303 42 : ANDZ,R2
0304 D1 : RRL,R1
0305 D1 : RRL,R1
0306 D1 : RRL,R1
0307 D1 : RRL,R1
0308 45F0 : ANDI,R1 H'F0'
030A 46F0 : ANDI,R2 H'F0'
030C 61 : IORZ,R1
030D C1 : STRZ,R1
030E 03 : LODZ,R3
030F 62 : IORZ,R2
0310 C2 : STRZ,R2
0311 0401 : LODI,R0 1
0313 CC042A : STRA,R0 CODE
0316 1B52 : BCTR,UN LNUM
```

;*GOTO program and execute.
;*Restore all registers first.

```

0318 3F00A1 :GOTO   BSTA,UN GAST      Get execute address
031B 0C0407 :       LODA,R0 COM+7      Restore all registers
031E 92      :       LPSU
031F 0D0401 :       LODA,R1 COM+1
0322 0E0402 :       LODA,R2 COM+2
0325 0F0403 :       LODA,R3 COM+3
0328 7710    :       PPSL   RS
032A 0D0404 :       LODA,R1 COM+4
032D 0E0405 :       LODA,R2 COM+5
0330 0F0406 :       LODA,R3 COM+6
0333 0C0408 :       LODA,R0 COM+8      get PSW lower and store
0336 CC040B :       STRA,R0 XGOT+1    in RAM. Self modifying code
0339 0C0400 :       LODA,R0 COM       is the only way to
033C 75FF    :       CPSL   H'FF'   restore PSL correctly.
033E 1F040A :       BCTA,UN XGOT

```

;* EXTENDED COMMAND PROCESSOR
;*To check for existence of of VHS data DOS
;*and implement W and K commands.

```

0341 0990    :XCMD   LODR,R1 *DOS      does DOS exist at 6800?
0343 E51F    :       COMI,R1 H'1F'
0345 9809    :       BCFR,EQ EXTCMD
0347 E44B    :       COMI,R0 A'K'
0349 188B    :       BCTR,EQ *DOS      DOS cold start
034B E457    :       COMI,R0 A'W'
034D 1CE803 :       BCTA,EQ *H'6803'   DOS warm start
0350 1F0431 :EXTCMD  BCTA,UN *CMD      external commands
0353 6800    :DOS     ACON   H'6800'   pointer to DOS

```

;* Continue initialisation

```

0355 CC0407 :INCC   STRA,R0 COM+7      set FLAG on GOTO
0358 041D    :       LODI,R0 >EBUG   Non-extend command table
035A CC0432 :       STRA,R0 CMD+1
035D 0498    :       LODI,R0 B'10011000'
035F D433    :       WRTE,R0 H'33'   initialise 8255
0361 5532    :       REDE,R1 H'32'   keyboard option
0363 D1      :       RRL,R1
0364 1A03    :       BCTR,N INCQ      parallel keyboard
0366 CC042B :       STRA,R0 KBFLAG   serial keyboard
0369 D1      :INCC   RRL,R1
036A D1      :       RRL,R1
036B D1      :       RRL,R1
036C 4503    :       ANDI,R1 3
036E 0D6205 :       LODA,R0 DL1,R1   1st delay constant
0371 CC042D :       STRA,R0 DLY1
0374 0D6209 :       LODA,R0 DL2,R1   2nd delay constant
0377 CC042E :       STRA,R0 DLY2
037A 0D620D :       LODA,R0 DPC,R1   default padding
037D CC0433 :       STRA,R0 OUTF
0380 9B22    :       ZBRR   MBUG

```

;*ACOS LOAD/DUMP ROUTINES

```

:TCTL   EQU   H'446'
:EXF    EQU   H'44B'
:TYPE   EQU   H'447'
:NBT    EQU   H'45C'
:EXAD   EQU   H'45E'
:ADD    EQU   H'45A'
:OBC    EQU   H'460'

:       ORG    H'3C2'
03C2 0500    :LOAD   LODI,R1 0
03C4 CD0446 :       STRA,R1 TCTL
03C7 CD0447 :       STRA,R1 TYPE
03CA 3F6000 :       BSTA,UN H'6000'   ACOS LOAD subroutine
03CD 0C044B :       LODA,R0 EXF
03D0 E4FF    :       COMI,R0 H'FF'
03D2 9C0022 :       BCFA,EQ MBUG
03D5 3F008A :       BSTA,UN CRLF
03D8 1F845E :       BCTA,UN *EXAD     Jump start program

03DB 20      :DUMP   EORZ,R0
03DC CC0446 :       STRA,R0 TCTL
03DF CC0447 :       STRA,R0 TYPE
03E2 3F6003 :       BSTA,UN H'6003'   ACOS DUMP subroutine
03E5 9B22    :       ZBRR   MBUG

```

;*Subroutine to transfer memory to buffer

```

03E7 0D045C :TSB    LODA,R1 NBT
03EA 0DC45A :LPX    LODA,R0 *ADD,R1-
03ED CD6460 :       STRA,R0 OBC,R1
03F0 5978    :       BRNR,R1 LPX
03F2 17      :       RETC,UN

```

;*Subroutine to transfer buffer to memory.

```

03F3 0D045C :TFR0   LODA,R1 NBT
03F6 0D4460 :LPJ    LODA,R0 OBC,R1-
03F9 CDE45A :       STRA,R0 *ADD,R1
03FC 5978    :       BRNR,R1 LPJ
03FE 17      :       RETC,UN

```

```

03FF 71      :       DATA   H'71'   Version

```

```

;*BINBUG SCRATCH AREA

:      ORG      H'400'

0400      :COM      RES      9      Register & PSW stack
0409      :OP       RES      1      Output switch
040A      :XGOT     RES      3      PSL restore instr.
040D      :TEMP     RES      2      16 bit address store
040F      :TEMQ     RES      2
0411      :TEMR     RES      1
0412      :         RES      1
0413      :BUFF     RES      BLEN   Input buffer
0427      :BPTR     RES      1      Buffer pointer
0428      :         RES      1
0429      :CNT      RES      1
042A      :CODE     RES      1

042B      :KBFLAG   RES      1      Keyboard flag; 0 => Parallel
042C      :MARK     RES      1      BP flag. FF => BP set.

;* Serial I/O delay constants
;*      1MHz clock      -   150   300   600  1200  2400 Baud
;*      2MHz clock      -   300   600  1200  2400  4800 Baud
042D      :DLY1     RES      1      DD   6E   36   1A   0C
042E      :DLY2     RES      1      DA   6B   34   18   0A

042F      :         RES      2

;* Indirect pointer to extended command processor
0431      :COMD     RES      2      Normally 001D (EBUG)

;* Padding flag- sclnnnnn
;*      s          - 1 two stop bits / 0 - one stop bit
;*      c          - 1 pad after CR
;*      l          - 1 pad after LF
;*      nnnnn      - number of padding characters
0433      :OUTF     RES      1      pad after CR/LF

;*RAM not cleared from here on
0434      :HDAT     RES      1      Bytes which were replaced
0435      :LDAT     RES      1      by BP vector
0436      :HADR     RES      1      BP address
0437      :LADR     RES      1

0438      :         RES      1
0439      :         RES      2

043B      :EOUT     RES      2      indirect to external COUT

```

```

:VERS2 EQU 1 zero => version 2.x
:VERS3 EQU 0 zero => version 3.x
:ONEMHZ EQU 0 zero => One MHZ version
:TWOMHZ EQU 1 zero => Two MHZ version
:*****
:* ACOS V3.E *
:* By R.A.ARMSTRONG *
:* COPYRIGHT MicroByte 1979,1982 *
:*****
:*
:* 1MHz version
:* DUMP ENTRY HEX 6003
:* LOAD ENTRY HEX 6000
:*
:* ACOS - Audio Cassette Operating System.
:*
:* This program is written as a series of
:* subroutines which are designed to be called
:* by user written tape routines as well as the
:* monitor load and save functions.
:*
:*APORT EQU H'32' I/O via extended port 32
:*
:*----- MEMORY USAGE
:* 446 TAPE CONTROL
:* 447 FILE TYPE
:* 448,9 CRC GENERATE BUFFER
:* 44A-D SCRATCH PAD
:* 44E-45F TAPE POST-AMBLE
:* 460-55F BUFFER
:*
:*----- POST-AMBLE
:* 44E-F CRC
:* 450 BLOCK NUMBER
:* 452-9 FILE NAME
:* 45A,B ADDRESS
:* 45C NO. OF BYTES
:* 45D FILE ID.
:* 45E,F EXECUTE ADDRESS
:*
:*----- TAPE CONTROL
:* BIT 7 - NO LOAD
:* BIT 6 - NO BLOCK COUNT
:* BIT 5 - IGNORE ERROR S+D
:* BIT 4
:* BIT 3 - INVERT LOAD PHASE
:* BIT 2 - DRIVE LOAD
:* BIT 1 - DRIVE DUMP
:* BIT 0

```

```

:*----- FILE TYPE
:* UPPER NIBBLE
:* 0 - OBJECT
:* 1-7 - DATA
:* 8 - SOURCE (ASS)
:* 9 - SOURCE (BAS)
:* A-F - SOURCE
:* LOWER NIBBLE
:* BIT 3
:* BIT 2
:* BIT 1 - FIRST BLOCK
:* BIT 0 - LAST BLOCK
:*
:*----- MONITOR EQUATES
:*COUT EQU H'02B4'
:*BOUT EQU H'0269'
:*GNUM EQU H'02DB'
:*STRT EQU H'00A4'
:*TFB EQU H'03E7'
:*TFRO EQU H'03F3'
:*LINE EQU H'005B'
:*
:* ORG H'6000'
:******LOAD ENTRY
:*
:*LXR BCTA,UN LMR

```

Move memory to buffer
Move buffer to memory

6000 1F60BB

Branch to loader routines

```

:*****DUMP ENTRY
:*
6003 20 :DMR EORZ,R0 Zero existing
6004 CC8047 : STRA,R0 *EXAD 45E execute address
6007 CC8049 : STRA,R0 *EXAD+2 45F
600A CC8100 : STRA,R0 *FLFL 450 and block number.
600D 0C8201 :DMJJ LODA,R0 *CODE 42A
6010 CC822B : STRA,R0 *CIDE 44C
6013 3F625A : BSTA,UN TFD Calculate BXa index
6016 C3 : STRZ,R3 and
6017 9F601A : BXa TNT Jump to command.

:* TURN ON TAPE (N)
:*
601A 08AF :TNT LODR,R0 *LTAP 446
601C 6402 : IORI,R0 2
601E C8AB :SAV STRR,R0 *LTAP 446
6020 D432 : WRTE,R0 APORT
6022 17 : RETC,UN

:* TURN OFF TAPE (F)
:*
6023 08A6 :TOT LODR,R0 *LTAP 446
6025 44FD : ANDI,R0 H'FD' Mask off dump motor bit
6027 1B75 : BCTR,UN SAV
6029 FFFFFFFF : DATA H'FF,FF,FF,FF'

:* OUTPUT CYCLE (continued)
602D FA7E :LP3 BDRR,R2 LP3
602F 7510 : CPSL RS
6031 17 : RETC,UN

:* DUMP WITH EXECUTE ADDRESS (E)
:*
6032 3F02DB :EAD BSTA,UN GNUM Get EXEC. address
6035 C990 : STRR,R1 *EXAD 45E and store in
6037 CA90 : STRR,R2 *EXAD+2 45F postamble.
6039 3F6203 : BSTA,UN GNAM Resolve filename,
603C 0444 :NXLN LODI,R0 A'D' prompt for dump
603E 3F02B4 : BSTA,UN COUT limit addresses,
6041 3F005B : BSTA,UN LINE input the addresses
6044 1F600D : BCTA,UN DMJJ & continue as for dump.

:*-----
6047 045E :EXAD ACON H'045E'
6049 045F : ACON H'045F'
604B 0446 :LTAP ACON H'0446'
:*-----

:* LEADER DELAY (L)
:*
604D 3B4B : BSTR,UN TNT Turn ON dump rec.
604F 0703 : LODI,R3 3 and output 3 x 3sec.
6051 3F6346 :A BSTA,UN HED headers.
6054 FB7B : BDRR,R3 A
6056 1B4B : BCTR,UN TOT Turn OFF dump rec. +RTN

```

```

:* DUMP TAPE (T)
:*
6058 7702 : PPSL LCOM
:* CHECK TYPE
605A 0C8120 : LODA,R0 *TYPE 447 If object file, then
605D 3C61C0 : BSTA,Z GADD get limit addresses,
:* SET UP IDD & FIRST FLAG
6060 0C8120 : LODA,R0 *TYPE 447
6063 6402 : IORI,R0 2 set first block flag,
6065 CC82B6 : STRA,R0 *IDD 45D and store in FILE ID.
:* GET NAME
6068 08DD : LODR,R0 *EXAD 45E If no execute address
606A 68DD : IORR,R0 *EXAD+2 45F
606C 6C8100 : IORA,R0 *FLFL 450 AND first file section,
606F 3C6203 : BSTA,Z GNAM then get filename,
6072 3F601A : BSTA,UN TNT turn on tape
6075 3F6346 : BSTA,UN HED & do 3 sec. header

6078 3F627E :DUMP BSTA,UN PPA Prepare postamble block,
607B 3F03E7 : BSTA,UN TFB transfer data to buffer
607E 0C82B6 : LODA,R0 *IDD 45D duplicate FILE ID.
6081 C8AA : STRR,R0 *ODD 44B
6083 0C822B : LODA,R0 *CIDE 44C If command line ends in
6086 F401 : TMI,R0 H'01' LF, then don't dump the
6088 1808 : BCTR,EQ NXT last block.
:* CLEAR LAST PAGE FLAG
608A 0C82B6 : LODA,R0 *IDD 45D
608D 44FE : ANDI,R0 H'FE'
608F CC82B6 : STRA,R0 *IDD 45D
6092 0C8100 :NXT LODA,R0 *FLFL 450
6095 8401 : ADDI,R0 1 Increment block number
6097 CC8100 : STRA,R0 *FLFL 450
609A 3F6302 : BSTA,UN CRCG Generate CRC for block
609D 3F6353 : BSTA,UN ZOTT Dump 256 bytes & Postamble.
60A0 088B : LODR,R0 *ODD 44B If last block
60A2 F401 : TMI,R0 H'01'
60A4 1809 : BCTR,EQ OUTO then terminate,
60A6 44FD : ANDI,R0 H'FD' else clear first
60A8 CC82B6 : STRA,R0 *IDD 45D page flag
60AB 1B4B : BCTR,UN DUMP and continue.

:*-----
60AD 044B :ODD ACON H'044B'
:*-----

:* TURN OFF TAPE
60AF 3F6023 :OUTO BSTA,UN TOT Turn off tape
60B2 0C822B : LODA,R0 *CIDE 44C If command line ended
60B5 F401 : TMI,R0 H'01' with LF, then get
60B7 9C603C : BCFA,EQ NXLN next set of addresses.
60BA 17 : RETC,UN

```

```

: *----- ENTRY FOR ACOS LOAD ROUTINES
: *
60BB 20 :LMR EORZ,R0
60BC C8A5 : STRR,R0 *EXF 44B Reset execute flag.
60BE 3F625A : BSTA,UN TFD Calculate BXA index
60C1 C3 : STRZ,R3
60C2 9F60C5 : BXA TLT & Jump to LOAD.

: * TURN ON TAPE (N)
: *
60C5 089E :TLT LODR,R0 *LTIP 446
60C7 6404 : IORI,R0 4 Load ON
60C9 C89A :SIV STRR,R0 *LTIP 446
60CB D432 : WRTE,R0 APORT
60CD 17 : RETC,UN

: * TURN OFF TAPE (F)
: *
60CE 0895 :TFT LODR,R0 *LTIP 446
60D0 44FB : ANDI,R0 H'FB' Mask load ctrl. bit
60D2 1B75 : BCTR,UN SIV

60D4 FFFFFFFF : DATA H'FF,FF,FF,FF,FF,FF,FF,FF,FF' reserved

: * LOAD AND EXECUTE (E)
: *
60DD 04FF :EXD LODI,R0 H'FF' Enable execute jump
60DF C882 : STRR,R0 *EXF 44B
60E1 1B20 : BCTR,UN LODX and proceed with load.

: *-----
60E3 044B :EXF ACON H'044B'
60E5 0446 :LTIP ACON H'0446'
: *-----

60E7 0AFC :CHEC LODR,R2 *LTIP 446
60E9 F640 : TMI,R2 H'40'
60EB 9C0269 : BCFA,EQ BOUT Print block number
60EE 1F6171 : BCTA,UN MOUT or '*'.

60F1 043F :EBAG LODI,R0 A'? ' Error block
60F3 1F02B4 : BCTA,UN COUT + RTN

: *-----
60F6 044C :FPF ACON H'044C'
: *-----

```

```

: * LOCATE AND VERIFY FILE (L)
: *
60F8 08EB : LODR,R0 *LTIP 446
60FA 6480 : IORI,R0 H'80' Set noload flag.
60FC C8E7 : STRR,R0 *LTIP 446
60FE 1B03 : BCTR,UN LODX and proceed as for load.

: *-----
6100 0450 :FLFL ACON H'0450'
6102 C0 : NOP

: * LOAD OBJECT FILE FROM TAPE (T)
: *
6103 20 :LODX EORZ,R0 Clear 1st page flag,
6104 C8F0 : STRR,R0 *FPF 44C
6106 3F60C5 : BSTA,UN TLT turn on load rec.

: * LOAD TAPE TO BUFFER
: *
6109 3F63D6 :LOD BSTA,UN ZINN Input block,
610C 3F62B8 : BSTA,UN CRX calculate CRC,
610F 3F62D5 : BSTA,UN CRK check CRC for match
6112 C1 : STRZ,R1 if CC=0 then
6113 1823 : BCTR,Z NZT CRC is OK.

: * ERROR - OUTPUT MESSAGE
: *
6115 08DF : LODR,R0 *FPF 44C If NOT loading file
6117 9809 : BCFR,Z E1
6119 043F : LODI,R0 A'? ' then print '?'
611B 3F6171 : BSTA,UN MOUT
611E 1B07 : BCTR,UN NYT

: *-----
6120 0447 :TYPE ACON H'0447'
: *-----

6122 044D :E1 LODI,R0 A'M' else print 'M' (Missed)
6124 3F616B : BSTA,UN WOUT

: * DISREGARD ERROR IF SOURCE OR DATA
6127 09F7 :NYT LODR,R1 *TYPE 447
6129 45F0 : ANDI,R1 H'F0'
612B 185C : BCTR,Z LOD

: * CHECK CONTROL BYTE FOR DISREG. FLAG
612D 08B8 : LODR,R0 *LTYP 446
612F F420 : TMI,R0 H'20'
6131 9856 : BCFR,EQ LOD
6133 0480 : LODI,R0 H'80'
6135 CC80F6 : STRA,R0 *FPF 44C

```

```

6138 3F622F :NZT BSTA,UN CHNM Check filename for match
613B 580A : BRNR,R0 NNT If name is wrong and
613D 08AA : LODR,R0 *FPF1 44C
613F 9837 : BCFR,Z OXU loading a file, then abort;
6141 0423 : LODI,R0 A'#' else if not loading, then
6143 3B2C :ZOUT BSTR,UN MOUT print '#', and look for
6145 1B42 : BCTR,UN LOD next block.

```

```

:* CHECK FOR FILE TYPE MATCH

```

```

6147 089E :NNT LODR,R0 *LTYT 446
6149 1A1A : BCTR,N NQT1 SKIP TYPE
614B 08D3 : LODR,R0 *TYPE 447
614D C1 : STRZ,R1
614E 2C82B6 : EORA,R0 *IDD 45D
6151 44F0 : ANDI,R0 H'F0'
6153 1804 : BCTR,Z WRX Print '^' if incorrect
6155 045E : LODI,R0 A'^' file type.
6157 1B6A : BCTR,UN ZOUT

```

```

:* BRANCH FOR SOURCE OR DATA

```

```

6159 45F0 :WRX ANDI,R1 H'F0' If NOT object file,
615B 182F : BCTR,Z NQT then if NOT ignoring errors,
615D F580 : TMI,R1 H'80'
615F 9C60CE : BCFA,EQ TFT then turn off tape +RTN,
6162 3F61DE : BSTA,UN SRCE else get source load origin
6165 1B25 :NQT1 BCTR,UN NQT and transfer to memory.

```

```

:*-----
6167 0446 :LTYT ACON H'446'
6169 044C :FPF1 ACON H'44C'
:*-----

```

```

:* MESSAGE OUTPUT ROUTINES

```

```

616B 09FA :WOUT LODR,R1 *LTYT 446
616D F540 : TMI,R1 H'40'
616F 9804 : BCFR,EQ QAQ
6171 3B02 :MOUT BSTR,UN QAQ
6173 0408 : LODI,R0 H'8' BS
6175 1F02B4 :QAQ BCTA,UN COUT + RTN
:* ERROR WHILE LOADING. OUTPUT "L" & ABORT
6178 044C :OXU LODI,R0 A'L'
617A 3B6F : BSTR,UN WOUT
617C 3F60CE :LEND BSTA,UN TFT
617F 08E8 : LODR,R0 *FPF1 44C
6181 EC8100 : COMA,R0 *FLFL 450
6184 14 : RETC,EQ
6185 20 :LOINT EORZ,R0 Prevent auto execute
6186 CC80E3 : STRA,R0 *EXF 44B
6189 1F60F1 : BCTA,UN EBAG Error '?' +RTN

```

```

618C 09DB :* NAME OK - FILE LOADING?
618E 181D :NQT LODR,R1 *FPF1 44C
: BCTR,Z FBK
:* FILE IS LOADING - PRINT NUMBER OR '*' & CR.
6190 042A :OPT LODI,R0 A'*'
6192 0D8100 : LODA,R1 *FLFL 450
6195 3F60E7 : BSTA,UN CHEC
6198 08CF :INC LODR,R0 *FPF1 44C
619A 8401 : ADDI,R0 1 Increment block count
619C C8CB : STRR,R0 *FPF1 44C
619E 08C7 : LODR,R0 *LTYT 446
61A0 BE03F3 : BSFA,N TFRO If transfer is permitted
then move buffer to memo
:* CHECK IF LAST BLOCK
61A3 0C82B6 : LODA,R0 *IDD 45D
61A6 F401 : TMI,R0 H'01'
61A8 1852 : BCTR,EQ LEND
61AA 1F6109 : BCTA,UN LOD YES
:* NOT YET LOADING - CHECK FOR FIRST BLOCK
61AD 0C82B6 :FBK LODA,R0 *IDD 45D
61B0 F402 : TMI,R0 H'02'
61B2 1807 : BCTR,EQ QUQ
:* NOT FIRST BLOCK
61B4 0453 : LODI,R0 A'S' Start block missed.
61B6 3F616B : BSTA,UN WOUT
61B9 1B5D : BCTR,UN INC
:* FIRST BLOCK
61BB 20 :QUQ EORZ,R0
61BC C1 : STRZ,R1
61BD 1B51 : BCTR,UN OPT
61BF C0 : NOP

```

```

:*****
:*
:*****
SUBROUTINES
:*****
:* GET ADDRESSES
:*
61C0 3F02DB :GADD BSTA,UN GNUM Get lower limit
61C3 3F00A4 : BSTA,UN STRT and put in TEMP
61C6 3F02DB : BSTA,UN GNUM Get upper limit.
61C9 ED82A8 : COMA,R1 *TEMP 40D
61CC 1E60F1 : BCTA,LT EBAG Error if upper < lower
61CF 1906 : BCTR,GT OKK
61D1 EE82AA : COMA,R2 *TEMP+2 40E
61D4 1E60F1 : BCTA,LT EBAG
61D7 CD82AC :OKK STRA,R1 *TEMQ 40F Save upper limit
61DA CE82AE : STRA,R2 *TEMQ+2 410
61DD 17 : RETC,UN

:* GET SOURCE LOAD ORIGIN
:*
61DE 0D82A8 :SRCE LODA,R1 *TEMP 40D
61E1 CD82B0 : STRA,R1 *ADD 45A
61E4 0C82AA : LODA,R0 *TEMP+2 40E
61E7 CC82B2 : STRA,R0 *ADD+2 45B
61EA 0E82B4 : LODA,R2 *NBT 45C
61ED 180E : BCTR,Z XAX

:* NOT FULL PAGE
61EF 82 : ADDZ,R2
61F0 7708 : PPSL WC
61F2 8500 : ADDI,R1 0
61F4 7508 : CPSL WC
61F6 CC82AA :SAPR STRA,R0 *TEMP+2 40E
61F9 CD82A8 : STRA,R1 *TEMP 40D
61FC 17 : RETC,UN
61FD 8501 :XAX ADDI,R1 1
61FF 1B75 : BCTR,UN SAPR

:*-----
6201 042A :CODE ACON H'042A'
:*-----

:* MOVE FILE NAME FROM INPUT BUFFER INTO POSTAMBLE.
:*
6203 0BA2 :GNAM LODR,R3 *BPTR 427 Get buffer pointer.
6205 0608 : LODI,R2 8 Maximum name length
6207 EBA0 :CXK COMR,R3 *CNT 429
6209 1814 : BCTR,EQ END
620B 0FA225 : LODA,R0 *BUF,R3+ Step over spaces
620E E420 : COMI,R0 A' '
6210 1875 : BCTR,EQ CXK
6212 CEC22D :XY STRA,R0 *PAB,R2- Store filename in Postamble
6215 EB92 : COMR,R3 *CNT 429
6217 1806 : BCTR,EQ END
6219 0FA225 : LODA,R0 *BUF,R3+
621C 5A74 : BRNR,R2 XY
621E 17 : RETC,UN
621F 0403 :END LODI,R0 3 Terminate filename with <3>
6221 CEC22D : STRA,R0 *PAB,R2-

```

```

6224 17 : RETC,UN

:*-----
6225 0413 :BUF ACON H'0413'
6227 0427 :BPTR ACON H'0427'
6229 0429 :CNT ACON H'0429'
622B 044C :CIDE ACON H'044C'
622D 0452 :PAB ACON H'0452'
:*-----

:* COMPARE FILE NAME IN BUFFER FOR A MATCH
:* WITH THE FILENAME IN THE POSTAMBLE.
:*
622F 0BF6 :CHNM LODR,R3 *BPTR 427 Buffer pointer
6231 0608 : LODI,R2 8
6233 EBF4 :CCK COMR,R3 *CNT 429
6235 1813 : BCTR,EQ CORR
6237 0FA225 : LODA,R0 *BUF,R3+ Skip leading spaces
623A E420 : COMI,R0 H'20'
623C 1875 : BCTR,EQ CCK
623E C1 :SKY STRZ,R1
623F 0EC22D : LODA,R0 *PAB,R2-
6242 E1 : COMZ,R1
6243 9808 : BCFR,EQ CEOT
6245 0FA225 : LODA,R0 *BUF,R3+
6248 5A74 : BRNR,R2 SKY
624A 04FF :CORR LODI,R0 H'FF' CC=LT if OK.
624C 17 : RETC,UN

:*
624D A701 :CEOT SUBI,R3 1
624F EF8229 : COMA,R3 *CNT 429
6252 9804 : BCFR,EQ EROR
6254 E403 : COMI,R0 H'03'
6256 1872 : BCTR,EQ CORR
6258 20 :EROR EORZ,R0 CC=EQ if not OK
6259 17 : RETC,UN

:* CHECKS SECOND CHARACTER OF COMMAND
:* AND CALCULATES BXA INDEX.
:*
625A 0D8227 :TFD LODA,R1 *BPTR 427 Point to command string
625D 0DA225 : LODA,R0 *BUF,R1+ get 2nd character
6260 0606 : LODI,R2 6
6262 EF4272 :XYZ COMA,R0 TUX,R2- Compare to table
6265 1804 : BCTR,EQ OHT
6267 5A79 : BRNR,R2 XYZ If not found then
6269 A501 : SUBI,R1 1 point back one.
626B CD8227 :OHT STRA,R1 *BPTR 427
626E 0E6277 : LODA,R0 TXB,R2 BXA index
6271 17 : RETC,UN

6272 544E4645 :TUX DATA A'TNFEL'
6277 3E000918 :TXB DATA H'3E,00,09,18,33'
627C 0000 : DATA 0,0

```

```

:* PREPARE POST-AMBLE
:*

```

```

627E 7702 :PPA PPSL LCOM
6280 7508 : CPSL WC
6282 09A4 : LODR,R1 *TEMP 40D
6284 C9AA : STRR,R1 *ADD 45A
6286 0AA2 : LODR,R2 *TEMP+2 40E
6288 CAA8 : STRR,R2 *ADD+2 45B
: * OVER BLOCK BOUNDARY?
628A E9A0 : COMR,R1 *TEMQ 40F
628C 9A0C : BCFR,LT XYX
: * BOUNDARY CROSSED LONG BLOCK - CALCULATE NBT
628E 20 : EORZ,R0
628F A2 : SUBZ,R2
6290 C8A2 : STRR,R0 *NBT 45C
: * CALCULATE NEW TEMP
6292 20 : EORZ,R0
6293 C895 : STRR,R0 *TEMP+2 40E
6295 8501 : ADDI,R1 1
6297 C98F : STRR,R1 *TEMP 40D
6299 17 : RETC,UN
: * SHORT BLOCK
629A 0892 :XYX LODR,R0 *TEMQ+2 410
629C A2 : SUBZ,R2
629D 8401 : ADDI,R0 1
629F C893 : STRR,R0 *NBT 45C
: * SET LAST PAGE FLAG
62A1 0893 : LODR,R0 *IDD 45D
62A3 6401 : IORI,R0 1
62A5 C88F : STRR,R0 *IDD 45D
62A7 17 : RETC,UN

: *-----
62A8 040D :TEMP ACON H'040D'
62AA 040E : ACON H'040E'
62AC 040F :TEMQ ACON H'040F'
62AE 0410 : ACON H'0410'
62B0 045A :ADD ACON H'045A'
62B2 045B : ACON H'045B'
62B4 045C :NBT ACON H'045C'
62B6 045D :IDD ACON H'045D'
: *-----

```

```

: * GENERATE CRC FOR BUFFER
: *
62B8 20 :CRX EORZ,R0
62B9 C1 : STRZ,R1
62BA C2 : STRZ,R2
62BB C3 : STRZ,R3
62BC 7708 : PPSL WC
62BE 0FE2E7 :XXX LODA,R0 *OBC,R3 460
62C1 3B26 : BSTR,UN CRU
62C3 FB79 : BDRR,R3 XXX
62C5 0710 : LODI,R3 H'10'
62C7 0FE2E1 :VVV LODA,R0 *PUB+2,R3 44F
62CA 3B1D : BSTR,UN CRU
62CC FB79 : BDRR,R3 VVV
62CE C995 : STRR,R1 *CRC+2 449
62D0 CA91 : STRR,R2 *CRC 448
62D2 7508 : CPSL WC
62D4 17 : RETC,UN

: * CHECK WITH POST-AMBLE
: *
62D5 088C :CRK LODR,R0 *CRC 448
62D7 2888 : EORR,R0 *PUB+2 44F
62D9 098A : LODR,R1 *CRC+2 449
62DB 2982 : EORR,R1 *PUB 44E
62DD 61 : IORZ,R1
62DE 17 : RETC,UN

: *-----
62DF 044E :PUB ACON H'044E'
62E1 044F : ACON H'044F'
62E3 0448 :CRC ACON H'0448'
62E5 0449 : ACON H'0449'
62E7 0460 :OBC ACON H'0460'
: *-----

: * GENERATE CRC FOR ONE BYTE
62E9 22 :CRU EORZ,R2
62EA 0608 : LODI,R2 H'08'
62EC 60 : IORZ,R0
62ED 9A0C :TES BCFR,N CRZE
62EF 7701 : PPSL CAR
62F1 2440 : EORI,R0 H'40'
62F3 2502 : EORI,R1 2
62F5 D1 :SHF RRL,R1
62F6 D0 : RRL,R0
62F7 FA74 : BDRR,R2 TES
62F9 C2 : STRZ,R2
62FA 17 : RETC,UN
62FB 7501 :CRZE CPSL CAR
62FD 1B76 : BCTR,UN SHF

62FF FFFFFFFF : DATA H'FF,FF,FF'

```

```

6302 3F62B8 : * CRC GENERATE
6305 08DC : CRCG BSTA,UN CRX
6307 C8D8 : LODR,R0 *CRC 448
6309 08DA : STRR,R0 *PUB+2 44F
630B C8D2 : LODR,R0 *CRC+2 449
630D 17 : STRR,R0 *PUB 44E
        : RETC,UN

: * 1 MHz timing constants {2MHz constants}
: * OUTPUT ONE CYCLE OF TONE.
: * R0 CONTAINS TIMING CONSTANT.
: *
630E 7710 :CYC PPSL RS
6310 0D804B : LODA,R1 *LTAP 446
6313 6503 : IORI,R1 3 set OP bit
6315 D532 : WRTE,R1 APORT
: * DELAY 1/2 CYCLE
6317 C2 : STRZ,R2
6318 F87E :LP0 BDRR,R0 LP0
631A 45FE : ANDI,R1 H'FE' clear OP bit
631C D532 : WRTE,R1 APORT
: * DELAY 1/2 CYCLE LESS EXTERNAL DELAY
631E A602 : SUBI,R2 2 {ADDI,R2 8}
6320 1F602D : BCTA,UN LP3

: * ENCODE AND OUTPUT ONE BYTE
: *
6323 CB9F :ZOT STRR,R3 *TEX 44A Start bit
6325 C1 : STRZ,R1
6326 0432 : LODI,R0 H'32' {6E}
6328 3B64 : BSTR,UN CYC
632A 0604 : LODI,R2 4 Set loop count
632C 01 :XZX LODZ,R1
632D 51 : RRR,R1
632E 51 : RRR,R1
632F 4403 : ANDI,R0 3
6331 C3 : STRZ,R3
6332 0F6340 : LODA,R0 TZB,R3
6335 3B57 : BSTR,UN CYC
6337 FA73 : BDRR,R2 XZX
6339 0412 : LODI,R0 H'12' {2E} Stop bit
633B 3B51 : BSTR,UN CYC
633D 0B85 : LODR,R3 *TEX 44A
633F 17 : RETC,UN

: * timing constants 1MHz { 2MHz }
: *
6340 3224170B :TZB DATA H'32,24,17,0B' {66,4A,30,18}

```

```

6344 044A : *-----
:TEX ACON H'044A'
: *-----

: * OUTPUT 3 SECOND HEADER
: *
6346 053F :HED LODI,R1 63
6348 0600 :XX LODI,R2 0
634A 040B :YY LODI,R0 H'0B' {18}
634C 3B40 : BSTR,UN CYC
634E FA7A : BDRR,R2 YY
6350 F976 : BDRR,R1 XX
6352 17 : RETC,UN

: * DUMP BUFFER & POST-AMBLE
: *
6353 0502 :ZOTT LODI,R1 2
6355 0689 :UU LODI,R2 H'89' {EF}
6357 040B :ZZ LODI,R0 H'0B' {18}
6359 3F630E : BSTA,UN CYC
635C FA79 : BDRR,R2 ZZ
635E F975 : BDRR,R1 UU
6360 0708 : LODI,R3 8 8 x H'FF'
6362 04FF :JJ LODI,R0 H'FF'
6364 3F6323 : BSTA,UN ZOT
6367 FB79 : BDRR,R3 JJ
6369 0702 : LODI,R3 2 2 Sync bytes
636B 0433 :H LODI,R0 H'33'
636D 3F6323 : BSTA,UN ZOT
6370 FB79 : BDRR,R3 H
6372 0FE2E7 :LPD LODA,R0 *OBC,R3 Dump buffer
6375 3F6323 : BSTA,UN ZOT
6378 DB78 : BIRR,R3 LPD
637A 0712 : LODI,R3 H'12'
637C 0FE385 :LPE LODA,R0 *PIB,R3 Dump postamble
637F 3F6323 : BSTA,UN ZOT
6382 FB78 : BDRR,R3 LPE
6384 17 : RETC,UN

: *-----
6385 044D :PIB ACON H'44D'
6387 6185 :ERBK ACON LOINT
: *-----

```

```

      ;* TIME INPUT CYCLE LENGTH
      ;*
6389 7710 :CON PPSL RS
638B 5532 :CIN REDE,R1 APORT Count time high
638D C0 : NOP
638E 1A04 : BCTR,N ZER
6390 8401 : ADDI,R0 1
6392 1B77 : BCTR,UN CIN
6394 5532 :ZER REDE,R1 APORT Count time low
6396 C0 : NOP
6397 9A04 : BCFR,N IXI
6399 8401 : ADDI,R0 1
639B 1B77 : BCTR,UN ZER
639D 7510 :IXI CPSL RS
639F B480 : TPSU SENS Test for Keyboard break
63A1 14 : RETC,EQ
63A2 12 : SPSU
63A3 A403 : SUBI,R0 3 adjust stack pointer
63A5 92 : LPSU
63A6 3F60CE : BSTA,UN TFT turn off tape
63A9 1BDC : BCTR,UN *ERBK and exit via error routine.

      ;* INPUT ONE BYTE
      ;*
63AB 20 :ZIN EORZ,R0 Look for start bit
63AC 3B5B : BSTR,UN CON
63AE E414 : COMI,R0 H'14' {2A}
63B0 9979 : BCFR,GT ZIN
      ;* NYBBLE COUNT TO 4
63B2 0704 : LODI,R3 4
63B4 0500 : LODI,R1 0
63B6 20 :XXY EORZ,R0
63B7 51 : RRR,R1
63B8 20 : EORZ,R0
63B9 3B4E : BSTR,UN CON
63BB E40D : COMI,R0 H'0D' {1D}
63BD 190D : BCFR,GT ZXZ
63BF 6501 : IORI,R1 1
63C1 51 : RRR,R1
63C2 E407 : COMI,R0 H'07' {11}
63C4 190E : BCTR,GT NINE
63C6 6540 :III IORI,R1 H'40'
63C8 FB6C :ZYZ BDRR,R3 XXY
63CA 01 : LODZ,R1
63CB 17 : RETC,UN
63CC 51 :ZXZ RRR,R1
63CD E413 : COMI,R0 H'13' {29}
63CF 1902 : BCTR,GT FIFT
63D1 1B73 : BCTR,UN III
      ;*** EQUALISE DELAYS
63D3 C0 :FIFT NOP
63D4 1B72 :NINE BCTR,UN ZYZ

```

```

      ;* INPUT BUFFER & POST-AMBLE
      ;*
63D6 3B53 :ZINN BSTR,UN ZIN Look for sync bytes
63D8 E433 : COMI,R0 H'33'
63DA 987A : BCFR,EQ ZINN
63DC 3B4D : BSTR,UN ZIN
63DE E433 : COMI,R0 H'33'
63E0 9874 : BCFR,EQ ZINN
63E2 0600 : LODI,R2 00
63E4 CA98 :DDD STRR,R2 *CRQ 44A
63E6 3B43 : BSTR,UN ZIN Input data block
63E8 0A94 : LODR,R2 *CRQ 44A
63EA CEE2E7 : STRA,R0 *OBC,R2
63ED DA75 : BIRR,R2 DDD
63EF 0612 : LODI,R2 H'12'
63F1 CABB :EEE STRR,R2 *CRQ 44A
63F3 3F63AB : BSTA,UN ZIN Input Postamble
63F6 0A86 : LODR,R2 *CRQ 44A
63F8 CEE385 : STRA,R0 *PIB,R2
63FB FA74 : BDRR,R2 EEE
63FD 17 : RETC,UN

      ;*-----
63FE 044A :CRQ ACON H'044A'
      ;*-----

```