BINBUG V3.6

Written by Ian Binnie

CONTENTS

1. DESCRIPTION

BINBUG 3.6 is a monitor program for the 2650 which is intended to replace the original PIPBUG monitor when a memory mapped Video Display Unit is being used as the terminal display device. 300 baud Binary format cassette load and dump routines provide program storage which is approximately six times faster than PIPBUG.

To ensure compatibility with programs which access PIPBUG 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 (one only)
C	Clear Breakpoint
D	Dump memory to tape
G	Goto program and execute
L	Load from tape into memory
S	Set and display registers

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

Only one breakpoint is supported by BINBUG 3.6.

BINBUG 3.6 contains a set of VDU driver routines which output characters to a memory mapped display. The display may be operated in Scrolling or Mixed mode.

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

2. REQUIREMENTS

Exclusive use of read/write memory between 0400 - 043F.

Provision to install a 2708 EPROM containing BINBUG addressed between 0000 - 03FF.

ASCII keyboard with serial output at 300 Baud, which is input through the sense terminal.

Memory mapped VDU addressed at 7800 - 7FFF in memory space with a format of 16 line x 64 characters per line. eg. DG640

2650 microprocessor using a 1MHz clock.

3. COMMANDS

The general format for all BINBUG 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 A<address><cr or lf>

ACCESS AND ALTER MEMORY

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, simply type 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 next sequential address 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 B<address><cr or lf>

SET BREAKPOINT

The ability to terminate execution of a program at a known point and then examine the state of each of the resisters is very powerful. The Breakpoint function replaces two bytes of code at the breakpoint address specified with a ZBRR indirect instruction and because of this 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 will be printed.

B450<cr or lf>

will set a Breakpoint at 0450 by replacing the two bytes at 0450 and 0451 by 9B 9B (ZBSR *1B). When the instruction at 0450 is executed, the address of the break point will be displayed and control passed back to the monitor.

Caution: 1. Since a breakpoint vector replaces TWO bytes of code, it is not possible to install a breakpoint at the address of a subroutine return instruction, unless the code immediately following the return instruction is NOT executed before the breakpoint. This is because the second byte of the breakpoint vector modifies the first byte of the following instruction.

2. The breakpoint address specified must be the address of the first byte of an instruction, otherwise the breakpoint will not be executed.

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.

3.3 C<cr or lf>

CLEAR BREAKPOINT

Once a breakpoint has been set, it may be cleared and the modified bytes of program restored, by typing:

C

If no breakpoint is currently set, BINBUG will output a question mark.

3.4 D<start> <end> [execute]<cr or lf>

DUMP MEMORY TO TAPE IN BINBUG BINARY FORMAT AT 300 Baud

The contents of memory between the start and end addresses inclusive will be output in BINBUG Binary format at 300 Baud via the FLAG terminal.

Each block of data will contain a maximum of 255 bytes of data recorded in the following format:

: (3A) Start of block mark.

-- (1000) Load address as two eight bit bytes.

- (FF) Number of bytes in block.

- (4A) Checksum of address packet.

---..... n bytes of data.

- (F5) Checksum of data block.

As many blocks of data as is required to dump the specified area of memory will be output. The 'end of file block' contains the EXECUTE ADDRESS instead of the normal load address and is indicated as such because the 'number of bytes in block' byte is zero.

3.5 G<address><cr or lf>

GOTO ADDRESS AND EXECUTE PROGRAM

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, this is the content of the registers immediately prior to to the breakpoint. Otherwise the registers are all set to zero when the computer is reset.

3.6 L<cr or lf>

LOAD BINBUG BINARY FORMAT TAPE

Loads an object tape in the same format as was dumped by the DUMP command.

If an execute address was included when dumping then the program will automatically branch to the execute address at the conclusion of an error free load.

If a checksum error is detected in the address packet checksum or the data block checksum, then a question mark will be printed and loading will be aborted.

3.7 S<register number><cr or lf>

DISPLAY AND SET REGISTER CONTENTS

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.

Sn<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 of to which register n will be restored when execution of the program is resumed using the 'G' command.

<cr> will abort 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

4. EXTENDING THE COMMAND SET

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

The ADDRESS CONSTANT (ACON) named COMD (0431,0432), 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.

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

5. EXTERNAL CHARACTER OUTPUT

All output from BINBUG - and output from most user programs - which is to be displayed on the terminal, will branch to the subroutine COUT (02B4). This subroutine normally will call the VDU driver routines to output the character to the display.

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

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 is 0, characters will be output through the VDU driver, but if OP is non-zero, output will be directed to the external character output routine, indirectly via EOUT.

6. VDU DRIVER ROUTINES

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

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

Carriage return	CR	0D	^M	Position to left of current line.
Linefeed	LF	0A	^J	Advance to next line.
Backspace	BS	08	^H	Back one position in same line (non - destructive)

All other control characters will be ignored.

6.1 CURSOR

The current positon of the cursor is indicated by the character under the cursor being displayed in inverted video.

6.2 PARTIAL SCROLLING

It is not always desirable to scroll the entire display. The video driver routine has provision to limit the scrolling to the bottom section of the screen. Setting bits 0-3 of SCRF (0437) to a number between 0 - F will prevent the top 0 - 15 lines of the display from being scrolled.

7. PIPBUG COMPATABILITY

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.
001D	EBUG	Error return - Outputs '?'.
0022	MBUG	Normal re-entry to monitor.

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 character from keyboard through the sense terminal, at 300 Baud, into R0.
02DB	GNUM	Get the next hexadecimal number from the input buffer into R1, R2.
02B4	COUT	Output R0 to the terminal or to an external device if OP (0409) is set to 1.

8. WRITING MONITOR INDEPENDENT PROGRAMS

The desirability of writing programs which are as independent as possible of the monitor being used, cannot be overemphasised. Users who write their programs in such a way that the program is heavily dependent on routines contained in a particular monitor, 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.

a. 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 this is done, 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.

b. 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.

SOURCE LISTING

```
                    :*****************************************
                    :*                BINBUG 3.6             *
                    :*        COPYRIGHT MicroByte 1979       *
                    :*****************************************
                    :*
                    :* This version of BINBUG supports:
                    :*
                    :*        BINBUG Binary format tape I/O
                    :*        300 Baud serial input
                    :*        DG640 VDU compatible
                    :*
                    :         ORG     0
                    :*
                    :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
                    :PAGE     EQU     H'78'         VDU MEMORY AT 7800
                    :*
0000  0738          :INIT     LODI,R3 CURS-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          :         LODI,R0 H'77'         Setup GOTO branch
000A  CC040A        :         STRA,R0 XGOT          in RAM to restore
000D  041F          :         LODI,R0 H'1F'         PSW lower
000F  CC040C        :         STRA,R0 XGOT+2
0012  041D          :         LODI,R0 H'1D'         Default end of command table
0014  CC0432        :         STRA,R0 COMD+1
0017  1B09          :         BCTR,UN MBUG
                    :*
0019  02B4          :ZOUT     ACON    COUT
                    :* Breakpoint vectors indirectly through this ACON
001B  014A          :VEC      ACON    BK01
```

```
                    :* COMMAND HANDLER
                    :* Input line and jump to routine depending on
                    :* the first character of the command.
                    :* Final branch is indirectly through and ACON
                    :* in RAM which may be altered to extend the
                    :* 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  1C01A1        :         BCTA,EQ BKPT        Set BREAKPOINT
003F  E443          :         COMI,R0 A'C'
0041  1C0197        :         BCTA,EQ CLR         CLEAR breakpoint
0044  E444          :         COMI,R0 A'D'
0046  1C02E5        :         BCTA,EQ DUMP        DUMP binary tape
0049  E447          :         COMI,R0 A'G'
004B  1C0121        :         BCTA,EQ GOTO        GOTO program and execute
004E  E44C          :         COMI,R0 A'L'
0050  1C03C4        :         BCTA,EQ LOAD        LOAD Binary tape
0053  E453          :         COMI,R0 A'S'
0055  1C00F1        :         BCTA,EQ SREG        SET register
0058  1F8431        :         BCTA,UN *COMD       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 VDU display.
                    :* CODE indicates the 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 if not
0069  980D          :        BCFR,EQ ALIN
006B  E7FF          :        COMI,R3 -1          at start of line
006D  1871          :        BCTR,EQ LLIN
006F  0408          :        LODI,R0 BS          the 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 not zero
007F  1A02          :        BCTR,N  DLIN
0081  8502          :        ADDI,R1 2           Then add 2 to code
0083  CD042A        :DLIN    STRA,R1 CODE
0086  CF0429        :        STRA,R3 CNT         Save length of line
0089  C0            :        NOP
008A  040D          :CRLF    LODI,R0 CR          Output Carriage return
008C  BB99          :        ZBSR    *ZOUT
008E  040A          :        LODI,R0 LF          and linefeed
0090  9B99          :        ZBRR    *ZOUT
0092  0502          :BLIN    LODI,R1 2
0094  E40A          :        COMI,R0 LF          If linefeed then set
0096  1866          :        BCTR,EQ CLIN        CODE = 2 and return
0098  CF2413        :        STRA,R0 BUFF,R3+    else insert character
009B  3F02B4        :        BSTA,UN COUT        in buffer and echo.
009E  1B40          :        BCTR,UN LLIN
                    :*
00A0  C0            :        NOP
```

```
                    :* Get HEX number from input buffer and store
                    :* in TEMP
                    :*
00A1  3F029E        :GAST   BSTA,UN GNUM        Get hex number
                    :*
00A4  CD040D        :STRT   STRA,R1 TEMP        Store R1, R2 in 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
00C4  0D840D        :       LODA,R1 *TEMP
00C7  3B19          :       BSTR,UN OTIN        Print contents and set new
00C9  1811          :       BCTR,EQ DALT        Just a LF
00CB  CC0411        :CALT   STRA,R0 TEMR        Something new
00CE  3F029E        :       BSTA,UN GNUM        Get the new byte
00D1  CE840D        :       STRA,R2 *TEMP       stuff in memory
00D4  0C0411        :       LODA,R0 TEMR
00D7  E404          :       COMI,R0 4           Was it followed by LF?
00D9  9C0022        :       BCFA,EQ MBUG
00DC  0601          :DALT   LODI,R2 1
00DE  3B4B          :       BSTR,UN INCT        Increment to next address
00E0  1B59          :       BCTR,UN LALT
                    :*
00E2  3F0279        :OTIN   BSTA,UN FOUT        Print current memory byte
00E5  3F005B        :       BSTA,UN LINE        set next (if any)
00E8  0C042A        :       LODA,R0 CODE
00EB  E402          :       COMI,R0 2           If < 2 then no byte
00ED  1E0022        :       BCTA,LT MBUG        was entered
00F0  17            :       RETC,UN



                    :* DISPLAY AND SET REGISTER STACK
                    :*
00F1  3F029E        :SREG   BSTA,UN GNUM        Get register number
00F4  E608          :LSRE   COMI,R2 8           Error if bigger than 8
00F6  1D001D        :       BCTA,GT EBUG
00F9  CE0411        :       STRA,R2 TEMR
00FC  0E6400        :       LODA,R0 COM,R2      Get register contents
```

```
00FF  C1        :          STRZ,R1              from store area and
0100  3B60      :          BSTR,UN OTIN         Display resister contents
0102  1815      :          BCTR,EQ CSRE         and set new contents
0104  CC040F    :ASRE      STRA,R0 TEMQ
0107  3F029E    :          BSTA,UN GNUM         Get new value
010A  02        :          LODZ,R2
010B  0E0411    :          LODA,R2 TEMR
010E  CE6400    :          STRA,R0 COM,R2       update resister store
0111  0C040F    :BSRE      LODA,R0 TEMQ
0114  E403      :          COMI,R0 3            If line terminated by CR
0116  1C0022    :          BCTA,EQ MBUG         then EXIT
0119  0E0411    :CSRE      LODA,R2 TEMR         else point to next
011C  8601      :          ADDI,R2 1            resister and
011E  1F00F4    :          BCTA,UN LSRE         repeat


                :* GOTO PROGRAM AND EXECUTE IT.
                :* Restore all resisters first.
                :*
0121  3F00A1    :GOTO      BSTA,UN GAST         Get execute address
0124  0C0407    :          LODA,R0 COM+7        Restore all resisters
0127  92        :          LPSU
0128  0D0401    :          LODA,R1 COM+1
012B  0E0402    :          LODA,R2 COM+2
012E  0F0403    :          LODA,R3 COM+3
0131  7710      :          PPSL    RS
0133  0D0404    :          LODA,R1 COM+4
0136  0E0405    :          LODA,R2 COM+5
0139  0F0406    :          LODA,R3 COM+6
013C  0C0408    :          LODA,R0 COM+8        Get PSW lower and store
013F  CC040B    :          STRA,R0 XGOT+1       in RAM. Self modifying code
0142  0C0400    :          LODA,R0 COM          Is the only way to
0145  75FF      :          CPSL    H'FF'        restore the PSL correctly.
0147  1F040A    :          BCTA,UN XGOT


                :* SINGLE BREAKPOINT ROUTINE
                :* Breakpoint vector branches indirectly though
                :* the ACON at 001B to this routine.
                :*
014A  CC0400    :BK01      STRA,R0 COM          Store the contents of
014D  13        :          SPSL                 PSW and all resisters on
014E  CC0408    :          STRA,R0 COM+8        stack at 0400.
0151  12        :          SPSU
0152  CC0407    :          STRA,R0 COM+7
0155  7710      :          PPSL    RS
0157  CD0404    :          STRA,R1 COM+4
015A  CE0405    :          STRA,R2 COM+5
015D  CF0406    :          STRA,R3 COM+6
0160  7510      :          CPSL    RS
0162  CD0401    :          STRA,R1 COM+1
0165  CE0402    :          STRA,R2 COM+2
0168  CF0403    :          STRA,R3 COM+3
016B  3B0E      :          BSTR,UN CLBK         Clear breakpoint vector
016D  0D040D    :          LODA,R1 TEMP         and print address of
0170  3F0269    :          BSTA,UN BOUT         breakpoint.
```

```
0173  0D040E    :         LODA,R1 TEMP+1
0176  3F0269    :         BSTA,UN BOUT
0179  9B22      :         ZBRR    MBUG


                ;*CLEAR SINGLE BREAKPOINT
                ;*
017B  20        ;CLBK     EORZ,R0
017C  CC042C    :         STRA,R0 MARK          Clear breakpoint flag
017F  0D042F    :         LODA,R1 HADR          Get address of BP.
0182  0E0430    :         LODA,R2 LADR
0185  3F00A4    :         BSTA,UN STRT          stuff in TEMP to print
0188  0C042D    :         LODA,R0 HDAT          and restore memory to
018B  CC840D    :         STRA,R0 *TEMP         orisinal contents.
018E  0C042E    :         LODA,R0 LDAT
0191  0701      :         LODI,R3 1
0193  CFE40D    :         STRA,R0 *TEMP,R3
0196  17        :         RETC,UN
                ;*
0197  0C042C    ;CLR      LODA,R0 MARK          is a BP set?
019A  1C001D    :         BCTA,Z  EBUG
019D  3B5C      :         BSTR,UN CLBK
019F  9B22      :         ZBRR    MBUG


                ;* SET A BREAKPOINT AT THE ADDRESS SPECIFIED.
                ;* The contents of two memory locations are saved
                ;* and replaced by ZBRR *001B to vector to BP
                ;* service routine.
                ;*
01A1  3F00A1    ;BKPT     BSTA,UN GAST          Get BP address into TEMP,
01A4  CD042F    :         STRA,R1 HADR          Save address,
01A7  CE0430    :         STRA,R2 LADR
01AA  0C840D    :         LODA,R0 *TEMP         and save memory contents.
01AD  CC042D    :         STRA,R0 HDAT
01B0  0701      :         LODI,R3 1
01B2  0FE40D    :         LODA,R0 *TEMP,R3
01B5  CC042E    :         STRA,R0 LDAT
01B8  049B      :         LODI,R0 H'9B'         replace with ZBRR *001B
01BA  CC840D    :         STRA,R0 *TEMP
01BD  049B      :         LODI,R0 VEC+128
01BF  CFE40D    :         STRA,R0 *TEMP,R3
01C2  04FF      :         LODI,R0 -1
01C4  CC042C    :         STRA,R0 MARK          Set BP Flag
01C7  9B22      :         ZBRR    MBUG
```

```
                    :* SCROLLING VIDEO DRIVER ROUTINE
                    :*LOADS CURSOR POINTER
01C9  7711          :LPTR     PPSL    RS+CAR
01CB  0E0438        :         LODA,R2 CURS
                    :* STORE POINTER TO BOTTOM LINE
01CE  05C0          :         LODI,R1 H'C0'
01D0  CD043A        :         STRA,R1 PTR+1
01D3  057B          :         LODI,R1 PAGE+3
01D5  CD0439        :         STRA,R1 PTR
01D8  17            :         RETC,UN
                    :*OUTPUT ASCII CHARACTER IN R0
01D9  E420          :KOUT     COMI,R0 SPAC
01DB  1A19          :         BCTR,LT WCC
                    :*ENTRY POINT TO WRITE ALL 128 CHARS AND INVERSE
01DD  3B6A          :WCHR     BSTR,UN LPTR
01DF  CEE439        :         STRA,R0 *PTR,R2     OUTPUT
01E2  DA00          :         BIRR,R2 $+2
                    :*ENTRY POINT TO SET CURSOR AT R2
01E4  E63F          :         COMI,R2 63
01E6  393A          :         BSTR,GT SCRL
01E8  0EE439        :SCUR     LODA,R0 *PTR,R2
01EB  6480          :         IORI,R0 H'80'
01ED  CEE439        :         STRA,R0 *PTR,R2
01F0  CE0438        :         STRA,R2 CURS
01F3  7510          :         CPSL    RS
01F5  17            :         RETC,UN
                    :*DECODES CONTROL CHARACTERS
01F6  3B51          :WCC      BSTR,UN LPTR
01F8  C1            :         STRZ,R1
01F9  0EE439        :         LODA,R0 *PTR,R2
01FC  447F          :         ANDI,R0 H'7F'
01FE  CEE439        :         STRA,R0 *PTR,R2
0201  E508          :         COMI,R1 BS
0203  9804          :         BCFR,EQ NBS
0205  A601          :         SUBI,R2 1           CAN'T BS TO PREV LINE
0207  1A04          :         BCTR,LT CRRR
0209  E50D          :NBS      COMI,R1 CR
020B  9802          :         BCFR,EQ NCR
020D  0600          :CRRR     LODI,R2 0
020F  E50A          :NCR      COMI,R1 LF
0211  380F          :         BSTR,EQ SCRL
0213  1B53          :         BCTR,UN SCUR
                    :*
                    :*CONVERTS NUMBER 0 TO F IN R1 TO
                    :* STARTING ADDRESS OF CORRESP LINE IN R1,R2
0215  0600          :CONV     LODI,R2 0
0217  7708          :         PPSL    WC
0219  51            :         RRR,R1
021A  52            :         RRR,R2
021B  51            :         RRR,R1
021C  52            :         RRR,R2
021D  7508          :         CPSL    WC
021F  6578          :         IORI,R1 PAGE
0221  17            :         RETC,UN
                    :*SUBROUTINE TO SCROLL PART OF SCREEN
0222  20            :SCRL     EORZ,R0
0223  CC0434        :         STRA,R0 TO+1
0226  0440          :         LODI,R0 64
```

```
0228  CC0436    :         STRA,R0 FROM+1
                :* LOAD LINE NO (0-F) TO BEGIN SCROLLING
022B  0D0437    :         LODA,R1 NUMB
022E  3B65      :S1       BSTR,UN CONV
0230  CD0433    :S2       STRA,R1 TO
0233  CD0435    :         STRA,R1 FROM
0236  0EE435    :S3       LODA,R0 *FROM,R2
0239  CEE433    :         STRA,R0 *TO,R2
023C  DA78      :         BIRR,R2 S3
023E  F503      :         TMI,R1  3             BOTTOM YET ?
0240  1802      :         BCTR,EQ CLRB
0242  D96C      :         BIRR,R1 S2
                :*CLEAR BOTTOM LINE
0244  06C0      :CLRB     LODI,R2 H'C0'
0246  0420      :         LODI,R0 SPAC
0248  CEE433    :SSPC     STRA,R0 *TO,R2
024B  DA7B      :         BIRR,R2 SSPC
024D  F504      :         TMI,R1  4             SCROLLED GRAPHIC YET ?
024F  14        :         RETC,EQ
0250  0D0437    :         LODA,R1 NUMB
0253  8510      :         ADDI,R1 16
0255  1B57      :         BCTR,UN S1
                :*
0257  C0        :         NOP
0258  C0        :         NOP


0259  30313233  :ANSI      DATA    A'0123456789ABCDEF'
                :*
                :* 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          + RETURN


                :* Print byte and three 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
```

```
                    :* INPUT CHARACTER THROUGH SENSE TERMINAL.
                    :* Uses a software timing loop and speed accuracy
                    :* depends on the CPU clock speed being 1MHz.
                    :*
0286  3F03A4      :CHIN     BSTA,UN SIN          PIPBUG compatible address
0289  447F        :         ANDI,R0 H'7F'
028B  17          :         RETC,UN


                    :* Convert HEX digit in R0 to binary in R3
                    :*
028C  0710        :LKUP     LODI,R3 16
028E  EF4259      :ALKU     COMA,R0 ANSI,R3-
0291  14          :         RETC,EQ
0292  E701        :         COMI,R3 1
0294  9A78        :         BCFR,LT ALKU
0296  9B1D        :         ZBRR    EBUG


                    :* Get HEX number from input buffer into R1, R2.
                    :* Enter at GNUM.
                    :*
0298  0C042A      :DNUM     LODA,R0 CODE         If CODE is 1 then
029B  1808        :         BCTR,Z  LNUM         doing second byte.
029D  17          :         RETC,UN
029E  20          :GNUM     EORZ,R0
029F  C1          :         STRZ,R1
02A0  C2          :         STRZ,R2
02A1  C3          :         STRZ,R3
02A2  CC042A      :         STRA,R0 CODE
02A5  0F0427      :LNUM     LODA,R3 BPTR
02A8  EF0429      :         COMA,R3 CNT          End of line?
02AB  14          :         RETC,EQ
02AC  0F2413      :         LODA,R0 BUFF,R3+     Get character from buffer
02AF  CF0427      :         STRA,R3 BPTR         and update pointer.
                    :*
02B2  1B0B        :         BCTR,UN $+13


                    :* OUTPUT CHARACTER IN R0 TO DG640 VDU DRIVER
                    :* or through external output routine.
                    :*
02B4  7710        :COUT     PPSL    RS           PIPBUG COMPATIBLE ENTRY
02B6  0D0409      :         LODA,R1 OP           If OP not zero then
02B9  1C01D9      :         BCTA,EQ KOUT
02BC  1F843B      :         BCTA,UN *EOUT        use external routine.


                    :* Continue GNUM.......
02BF  E420        :         COMI,R0 SPAC
02C1  1855        :         BCTR,EQ DNUM
02C3  3F028C      :         BSTA,UN LKUP         Convert to Binary
02C6  040F        :         LODI,R0 H'0F'
02C8  D2          :         RRL,R2
02C9  D2          :         RRL,R2
```

```
02CA  D2        ;         RRL,R2
02CB  D2        ;         RRL,R2
02CC  42        ;         ANDZ,R2
02CD  D1        ;         RRL,R1
02CE  D1        ;         RRL,R1
02CF  D1        ;         RRL,R1
02D0  D1        ;         RRL,R1
02D1  45F0      ;         ANDI,R1 H'F0'
02D3  46F0      ;         ANDI,R2 H'F0'
02D5  61        ;         IORZ,R1
02D6  C1        ;         STRZ,R1
02D7  03        ;         LODZ,R3
02D8  62        ;         IORZ,R2
02D9  1B02      ;         BCTR,UN $+4
                ;*
                ;* Get Hex number from buffer into R1, R2.
                ;*
02DB  1B41      ;         BCTR,UN GNUM        PIPBUG COMPATIBLE ENTRY
                ;*
02DD  C2        ;         STRZ,R2
02DE  0401      ;         LODI,R0 1
02E0  CC042A    ;         STRA,R0 CODE
02E3  1B40      ;         BCTR,UN LNUM



                ;* DUMP SPECIFIED AREA OF MEMORY AT 300 baud via
                ;* FLAG terminal in BINARY format.
                ;*
02E5  3F00A1    ;DUMP     BSTA,UN GAST        Get and store Dump start
02E8  3F029E    ;         BSTA,UN GNUM        and end addresses.
02EB  8601      ;         ADDI,R2 1           include last byte
02ED  7708      ;         PPSL    WC
02EF  8500      ;         ADDI,R1 0
02F1  7508      ;         CPSL    WC
02F3  CD040F    ;         STRA,R1 TEMQ        Save end address
02F6  CE0410    ;         STRA,R2 TEMQ+1
02F9  0720      ;         LODI,R3 32          Output 32 FF as leader
02FB  04FF      ;GAP      LODI,R0 H'FF'
02FD  3F036F    ;         BSTA,UN SOUT
0300  FB79      ;         BDRR,R3 GAP
0302  04FF      ;FDUM     LODI,R0 -1
0304  CC0412    ;         STRA,R0 TEMS
0307  043A      ;         LODI,R0 A':'        Start of block mark
0309  3F036F    ;         BSTA,UN SOUT
030C  20        ;         EORZ,R0
030D  CC042B    ;         STRA,R0 BCC         Zero checksum scratch
0310  0D040F    ;         LODA,R1 TEMQ        See if full or partial block
0313  0E^P410    ;         LODA,R2 TEMQ+1      is to be dumped.
0316  AE040E    ;         SUBA,R2 TEMP+1
0319  7708      ;         PPSL    WC
031B  AD040D    ;         SUBA,R1 TEMP
031E  7508      ;         CPSL    WC
0320  1E001D    ;         BCTA,LT EBUG        negative length????
0323  1911      ;         BCTR,GT ADUM
0325  5A11      ;         BRNR,R2 BDUM
0327  3F029E    ;         BSTA,UN GNUM        Get execute address
032A  01        ;         LODZ,R1             and output it
```

```
032B  3F036F    :         BSTA,UN SOUT
032E  02        :         LODZ,R2
032F  3B3E      :         BSTR,UN SOUT
0331  20        :         EORZ,R0             with zero length byte
0332  3B3B      :         BSTR,UN SOUT        as end of dump block.
0334  9B22      :         ZBRR    MBUG
                :*
0336  06FF      :ADUM     LODI,R2 255         Full length (255 bytes)
0338  CE0428    :BDUM     STRA,R2 MCNT        Short block
033B  0C040D    :         LODA,R0 TEMP        Dump load address
033E  3B2F      :         BSTR,UN SOUT
0340  0C040E    :         LODA,R0 TEMP+1
0343  3B2A      :         BSTR,UN SOUT
0345  0C0428    :         LODA,R0 MCNT        Number of bytes in block
0348  3B25      :         BSTR,UN SOUT
034A  0C042B    :         LODA,R0 BCC         Address packet checksum
034D  3B20      :         BSTR,UN SOUT
034F  0F0412    :DDUM     LODA,R3 TEMS
0352  0FA40D    :         LODA,R0 *TEMP,R3+   Now dump the data bytes
0355  EF0428    :         COMA,R3 MCNT
0358  1807      :         BCTR,EQ EDUM        until at end
035A  CF0412    :         STRA,R3 TEMS
035D  3B10      :         BSTR,UN SOUT
035F  1B6E      :         BCTR,UN DDUM
0361  0C042B    :EDUM     LODA,R0 BCC         Data block checksum
0364  3B09      :         BSTR,UN SOUT
0366  0E0428    :         LODA,R2 MCNT
0369  3F00AB    :         BSTA,UN INCT        increment load address
036C  1F0302    :         BCTA,UN FDUM        next block


                :* OUTPUT CHARACTER IN R0 AT 300 BAUD through FLAG
                :* terminal. All 8 bits are output by this S/R.
                :*
036F  7710      :SOUT     PPSL    RS          Use upper resister bank
0371  C1        :         STRZ,R1
0372  3B1F      :         BSTR,UN CBCC        Accumulate Checksum.
0374  7640      :         PPSU    FLAG
0376  C2        :         STRZ,R2
0377  0508      :         LODI,R1 8           No. of data bits.
0379  3B20      :         BSTR,UN DLAY        Two stop bits after
037B  3B1E      :         BSTR,UN DLAY        previous character.
037D  7440      :         CPSU    FLAG        Start bit
037F  3B1A      :ACOU     BSTR,UN DLAY
0381  52        :         RRR,R2
0382  1A04      :         BCTR,LT ONE
0384  7440      :         CPSU    FLAG
0386  1B02      :         BCTR,UN ZERO
0388  7640      :ONE      PPSU    FLAG
038A  F973      :ZERO     BDRR,R1 ACOU
038C  3B0D      :         BSTR,UN DLAY
038E  7640      :         PPSU    FLAG
0390  7510      :         CPSL    RS
0392  17        :         RETC,UN
                :*
                :* Accumulate checksum
                :*
```

```
0393  2D042B   :CBCC    EORA,R1 BCC
0396  D1        :        RRL,R1
0397  CD042B   :        STRA,R1 BCC
039A  17        :        RETC,UN
                :*
                :* Software timing loop for 300 Baud serial output.
                :*
039B  04B7      :DLAY    LODI,R0 H'B7'       Full bit delay
039D  F87E      :        BDRR,R0 $
039F  04B3      :DLY     LODI,R0 H'B3'       Half bit delay
03A1  F87E      :        BDRR,R0 $
03A3  17        :        RETC,UN


                :* INPUT 8 bit serial character through SENSE.
                :*
03A4  7710      :SIN     PPSL    RS          Use upper resisters
03A6  0500      :        LODI,R1 0
03A8  0608      :        LODI,R2 8           No. data bits
03AA  12        :ACHI    SPSU                Look for start bit
03AB  1A7D      :        BCTR,LT ACHI
03AD  3B70      :        BSTR,UN DLY
03AF  12        :        SPSU
03B0  1A78      :        BCTR,LT ACHI
03B2  3B67      :BCHI    BSTR,UN DLAY
03B4  12        :        SPSU
03B5  4480      :        ANDI,R0 H'80'       Mask out sense bit
03B7  51        :        RRR,R1
03B8  61        :        IORZ,R1             merge in this bit
03B9  C1        :        STRZ,R1
03BA  FA76      :        BDRR,R2 BCHI
03BC  3B5D      :        BSTR,UN DLAY
03BE  01        :        LODZ,R1
03BF  3B52      :        BSTR,UN CBCC        Accumulate checksum
03C1  7510      :        CPSL    RS
03C3  17        :        RETC,UN


                :* LOAD 300 Baud BINARY FORMAT TAPE.
                :*
03C4  3B5E      :LOAD    BSTR,UN SIN
03C6  E43A      :        COMI,R0 A':'        Look for start of block
03C8  987A      :        BCFR,EQ LOAD
03CA  20        :        EORZ,R0
03CB  CC042B    :        STRA,R0 BCC         Zero checksum
03CE  3B54      :        BSTR,UN SIN         Get load address
03D0  CC040D    :        STRA,R0 TEMP
03D3  3B4F      :        BSTR,UN SIN
03D5  CC040E    :        STRA,R0 TEMP+1
03D8  3B4A      :        BSTR,UN SIN         Block length - zero
03DA  5803      :        BRNR,R0 ALOA        means last block so
03DC  1F840D    :        BCTA,UN *TEMP       jump start program.
                :*
03DF  CC0428    :ALOA    STRA,R0 MCNT        Save block length
03E2  3F03A4    :        BSTA,UN SIN         set checksum byte
03E5  0C042B    :        LODA,R0 BCC
```

```
03E8  9813     :         BCFR,EQ LERR        Error if not zero
03EA  C3       :         STRZ,R3
03EB  3F03A4   :BLOA     BSTA,UN SIN         set data bytes
03EE  EF0428   :         COMA,R3 MCNT        until the right number
03F1  1805     :         BCTR,EQ CLOA        has been got.
03F3  CFE40D   :         STRA,R0 *TEMP,R3    store in memory
03F6  DB73     :         BIRR,R3 BLOA
03F8  0C042B   :CLOA     LODA,R0 BCC         test data checksum.
03FB  1847     :         BCTR,EQ LOAD
03FD  9B1D     :LERR     ZBRR    EBUG        Error if not zero
```

```
                :* BINBUG SCRATCH RAM
                :*
                :       ORG     H'400'
                :*
0400            :COM    RES     9           Resister & 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            :TEMS   RES     1
0413            :BUFF   RES     BLEN        Input buffer
0427            :BPTR   RES     1           Buffer pointer
0428            :MCNT   RES     1
0429            :CNT    RES     1
042A            :CODE   RES     1           Buffer status
042B            :BCC    RES     1           Checksum scratch
042C            :MARK   RES     1           BP Flag. FF => BP set.
042D            :HDAT   RES     1           Bytes which were replaced
042E            :LDAT   RES     1           by a BP vector.
042F            :HADR   RES     1           BP address.
0430            :LADR   RES     1
                :*INDIRECT POINTER TO EXTENDED COMMAND PROCESSOR
0431            :COMD   RES     2           Normally 001D
                :*
                :* VDU Driver scratch
                :*
0433            :TO     RES     2
0435            :FROM   RES     2
0437            :NUMB   RES     1
                :*RAM NOT CLEARED FROM HERE ON
0438            :CURS   RES     1           VDU cursor count
0439            :PTR    RES     2
                :*
043B            :EOUT   RES     2           INDIR TO EXTERNAL COUT
```