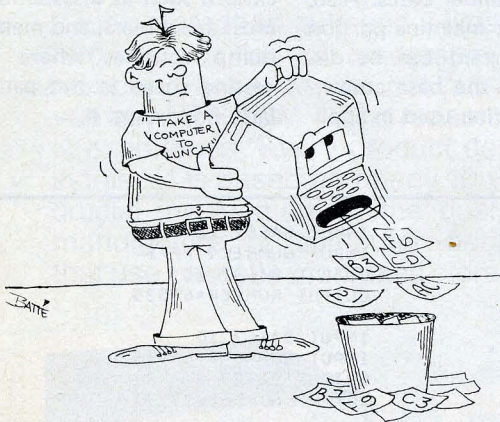


Hexadecimal Memory Dump



One good article generates another. This is a nice companion to one of last year's.

Kudos to Mark Borger-son. His 6800 assembly-language program for fast loading machine-language

programs as it appeared in the February 1977 issue of *Kilobaud* ("Cut 6800 Programming Time with this Extraor-

dinary Program," p. 104) works extremely well. I re-assembled that program to relocate it to a convenient

location in my memory.

After implementing the loader program, it appeared that a hexadecimal dump program would be a very useful companion program. The MIKBUG P command can be used for that purpose; however, it has two disadvantages. First, it is necessary to use the M command to enter the starting and ending addresses at A002-A004. The second, and more serious, disadvantage is that the output is formatted for the MIKBUG tape punch and is very difficult to read since there is no spacing between bytes.

Here is a program that overcomes those disadvantages. The display format is not new, by any means, but the only program I have seen for this format is written to run on an Altair 680b by Mits. Since Mits does not use MIKBUG, the program will not run on my SWTP 6800 system. My program will run

Program listing.

```

00001          NAM      HEXDUMP
00002          **
00003          * HEXADECIMAL MEMORY DUMP PROGRAM
00004          **
00005          * LOAD VIA MIKBUG "L" COMMAND
00006          * USE MIKBUG "G" COMMAND TO START
00007          **
00008          * ENTER ADDRESS OF FIRST BYTE TO DUMP
00009          **
00010          * ENTER ADDRESS OF LAST BYTE TO DUMP
00011          **
00012          * PUSH "RESET" ON COMPUTER TO ABORT
00013          * CONTROL RETURNS TO MIKBUG
00014          **
00015          OPT      NOG
00016          OPT      S
00017          OPT      0

00019          * MIKBUG ROUTINES USED
00020          E1D1     OUTEEE EQU    $E1D1
00021          E0BF     OUT2H  EQU    $E0BF
00022          E047     BADDR  EQU    $E047
00023          E0CC     OUTS   EQU    $E0CC
00024          E0C8     OUT4HS EQU    $E0C8
00025          E0E3     MONIT  EQU    $E0E3

00027          * START PROGRAM
00028          3F20          ORG      $3F20
00029          3F20 0002     TEMP    RMB    2          TEMP STORAGE FOR X REG.
00030          3F22 0002     LSTBYT RMB    2          ADDRESS OF LAST BYTE TO DUMP
00031          3F24 0001     COUNT  RMB    1          COLUMN COUNTER
  
```

on any 6800 system using MIKBUG, or one of the newer replacements for MIKBUG.

I have used the same basic dump technique as in the Altair program, but with input/output routines modified for MIKBUG. I have added some prompt messages at the beginning and have used Mr. Borgerson's technique of relocating the stack pointer to restart the program by simply typing G on the terminal.

To use the program after loading, set the program counter at A048-A049 to 3F25 (or the appropriate starting address if you have relocated the program) and use the MIKBUG G command to start execution. The program title will be printed, followed by a prompt message, FIRST BYTE TO PRINT. The address of the first byte to dump is entered and the computer responds with LAST BYTE TO PRINT. The address of the last byte is entered and the dump begins. The display format consists of 16 bytes per line with the address of the first byte being printed at the left (see Fig. 1).

There is no limit to the amount of memory that can be dumped at one time; any number of bytes from one to 65K can be dumped. (Hope you have a lot of paper for the larger numbers!) A word of caution: The address of the first byte to be dumped must be less than that of the last. If this is not the case, all memory locations *except* the region between the two addresses will be dumped! If both addresses are the same, only one byte will be displayed.

The dump shown in Fig. 1 is a dump of the dump program itself. This should prove to be a valuable debugging program; especially if your program has "gone to that never-never land known only to CPUs and covered its tracks in the process," to quote another *Kilobaud* author. ■

```

00032 3F25 8E A060 GO      LDS   #A060  RELOCATE STACK POINTER
00033 3F28 CE 3FB7      LDX   #TITLE POINT TO 'TITLE' MESS.
00034 3F2B A6 00      LDA A 00,X  GET CHARACTER TO PRINT
00035 3F2D C6 2E      LDA B #'    PUT ASCII PERIOD IN B
00036 3F2F 11          CBA          IS CHAR. IN A-REG A PERIOD?
00037 3F30 27 06      BEQ   ADRS1
00038 3F32 BD E1D1     JSR   OUTEEE PRINT CHAR IN A REG
00039 3F35 08          INX
00040 3F36 20 F3      BRA   AA    LOOP FOR MORE
00041 3F38 86 0D      LDA A #S0D  CARRIAGE RETURN
00042 3F3A BD E1D1     JSR   OUTEEE
00043 3F3D 86 0A      LDA A #S0A  LINE FEED
00044 3F3F BD E1D1     JSR   OUTEEE
00045 3F42 86 00      LDA A #S00  ASCII NULL
00046 3F44 BD E1D1     JSR   OUTEEE
00047 3F47 BD E1D1     JSR   OUTEEE
00048 3F4A CE 3FCF     LDX   #FIRST POINT TO 'FIRST' MESS.
00049 3F4D A6 00      LDA A 00,X  GET CHAR TO PRINT
00050 3F4F 11          CBA          IS CHAR IN A-REG A PERIOD?
00051 3F50 27 06      BEQ   GET
00052 3F52 BD E1D1     JSR   OUTEEE PRINT CHAR IN A REG
00053 3F55 08          INX
00054 3F56 20 F5      BRA   BB    LOOP FOR MORE
00055 3F58 8D 51      GET   BSR   GETADR  GET FIRST ADDRESS
00056 3F5A FF 3F20     STX   TEMP  STORE IT
00057 3F5D BD E0CC     JSR   OUTS
00058 3F60 CE 3FE3     LDX   #LAST  POINT TO 'LAST' MESS.
00059 3F63 A6 00      LDA A 00,X  GET CHARACTER TO PRINT
00060 3F65 C6 2E      LDA B #'    ASCII PERIOD IN B-REG
00061 3F67 11          CBA          IS CHAR IN A-REG A PERIOD?
00062 3F68 27 06      BEQ   ADRS2
00063 3F6A BD E1D1     JSR   OUTEEE PRINT CHAR IN A REG
00064 3F6D 08          INX
00065 3F6E 20 F3      BRA   CC    LOOP FOR MORE
00066 3F70 8D 39      ADRS2 BSR   GETADR  GET LAST ADDR
00067 3F72 08          INX          ADJUST IT
00068 3F73 FF 3F22     STX   LSTBYT STORE IT
00069 3F76 FE 3F20     LDX   TEMP  POINT TO FIRST BYTE
00070 3F79 86 0D      CRLF   LDA A #S0D  SEND CR,LF
00071 3F7B BD E1D1     JSR   OUTEEE
00072 3F7E 86 0A      LDA A #S0A
00073 3F80 BD E1D1     JSR   OUTEEE
00074 3F83 86 11      LDA A #17
00075 3F85 B7 3F24     STA A COUNT INIT COUNTER
00076 3F88 FF 3F20     STX   TEMP  STORE X REG
00077 3F8B CE 3F20     LDX   #TEMP
00078 3F8E BD E0C8     JSR   OUT4HS PRINT ADDRESS
00079 3F91 FE 3F20     LDX   TEMP  RESTORE XREG
00080 3F94 7A 3F24     NXTBYT DEC COUNT
00081 3F97 27 E0      BEQ   CRLF
00082 3F99 BD E0CC     JSR   OUTS  SEND A SPACE
00083 3F9C A6 00      LDA A X     BYTE TO A
00084 3F9E BD E0BF     JSR   OUT2H PRINT IT, & INCREMENT X-REG
00085 3FA1 EC 3F22     CPX   LSTBYT ARE WE DONE?
00086 3FA4 27 02      BEQ   JMONIT YES, RETURN TO MIKBUG
00087 3FA6 20 EC      BRA   NXTBYT
00088 3FA8 7E E0E3     JMONIT MONIT
00089 3FAB BD E0CC     GETADR JSR   OUTS  SEND SPACE
00090 3FAE 86 3F      LDA A #'?  SEND QUESTION MARK
00091 3FB0 BD E1D1     JSR   OUTEEE
00092 3FB3 BD E047     JSR   BADDR GET ADDRESS
00093 3FB6 39          RTS        RETURN

```

```

00095 3FB7 48          TITLE FCC   /HEXADECIMAL MEMORY DUMP./
00096 3FCF 46          FIRST FCC  /FIRST BYTE TO PRINT./
00097 3FE3 4C          LAST  FCC  /LAST BYTE TO PRINT./

```

```

00099 A048          ORG   $A048
00100 A048 3F20     FDB   $3F20  STARTING ADDRESS IN PROG CTR
00101          END
OUTEEE E1D1
OUT2H  E0BF
BADDR  E047

```

```

OUTS   E0CC
OUT4HS E0C8
MONIT  E0E3
TEMP   3F20
LSTBYT 3F22
COUNT 3F24
GO      3F25
AA      3F2B
ADRS1  3F38
BB      3F4D
GET     3F58
CC      3F63
ADRS2  3F70
CRLF   3F79
NXTBYT 3F94
JMONIT 3FA8
GETADR 3FAB
TITLE  3FB7
FIRST  3FCF
LAST   3FE3

```

TOTAL ERRORS 00000

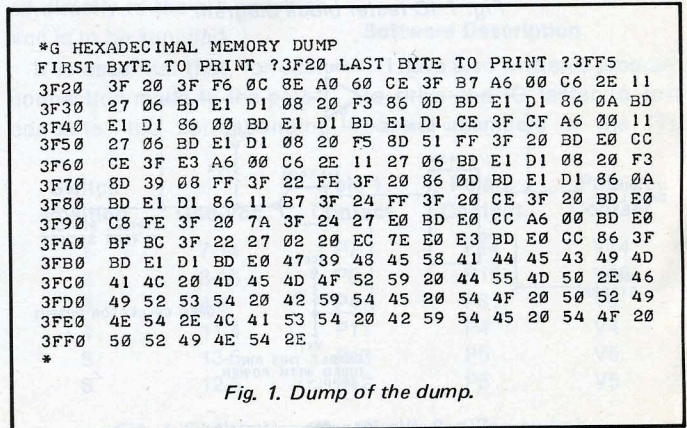


Fig. 1. Dump of the dump.