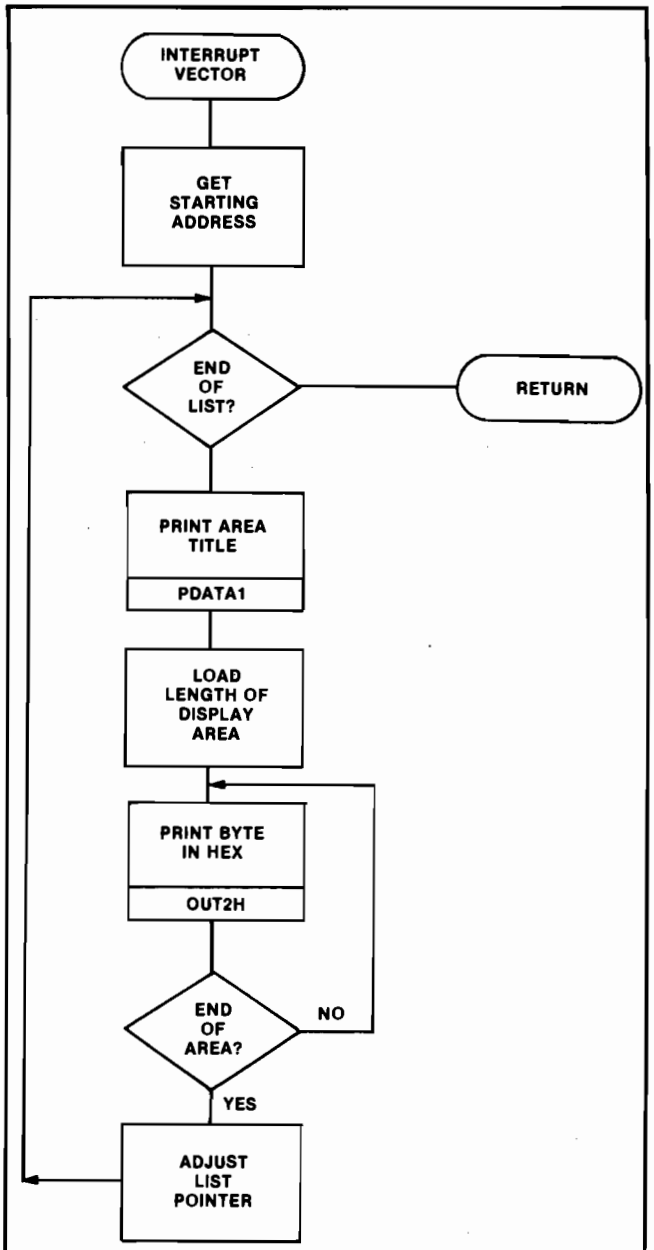
6800 MIKBUG DUMP

By Tom Munnecke

How often have you wished you could look at certain areas of memory at strategic times when debugging an assembler program? This 33-byte program allows you to do just that on 6800 systems using the MIKBUG™ monitor.



Flowchart of DUMP program.

The program uses a display list to define strategic locations in memory by name, length and address. Whenever the routine is invoked, it displays the pertinent data, then returns to let your program continue.

A display list could cause the following display:

COUNT = 44,MASK = 01,LAST = E1AC,NEXT = 0012

You can call the program directly, or you can drive it with an external interrupt. In order to use interrupts:

1. Use the RTI instruction at location A059
2. Set 'A050' into locations A000-A001.
3. Make sure the interrupt flag is clear in your program. If in doubt, execute a 'CLI' instruction.
4. If you do not have an interrupt button on your system, you can force an interrupt by momentarily connecting a wire between the IRQ pin on the MPU (pin 4) or the system bus, and ground.

After you set up your display list according to the formats shown in Figure 3, set memory locations A078-A079 to point to the beginning of the list.

Display List Format:

Descriptor 1	Descriptor 2		Descriptor N	ETX

Any number of descriptors may be in list. List is terminated by ASCII ETX character (HEX '04') after last descriptor. This list is pointed to by location A078.

Figure 1.

Each Descriptor is of the format:

NAME	ETX	LEN	ADDR

Where

1. Name is a string of ASCII characters of any length to be printed before the memory is displayed.
2. ETX (HEX '04') delimits the end of the Name field.
3. LEN is a 1-byte count of the number of bytes of memory to be displayed.
4. ADDR is a 2-byte address of the area to be displayed.

Figure 2.

Example: Print the following when DUMP is executed.

MASK = XX COUNT = YYYY
NAME = ZZZZZZZZZZZZZZZZZZZZ

Where XX is the value of location 003F and YY is the value of location 0100. ZZZZZZZZZZZZZZZZZZZZ is the data stored in location 0020 through 0033.

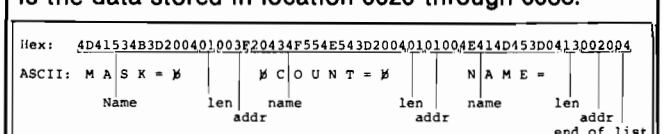


Figure 3.

Since the program uses only relative addressing, it may be located anywhere in memory.□

PROGRAM LISTING

```
00001                            NAM         DUMP
00002                  * 11/17/77 BY TOM MUNNECKE
00003                  * DUMP SELECTED PORTIONS OF
00004                  * MEMORY WITH TITLES.
00005                  *
00006                  * DISPLAY LIST POINTED TO
00007                  * BY A078 CONTROLS FORMATS
00008                  *
00009                  * THIS PROGRAM USES THE
00010                  * MIKBUG (TM MOTOROLA)
00011                  * ROUTINES PDATA1 AND OUT2H
00012                  * TO PRINT ASCII STRINGS
0013                  * (DELIMITED BY ETX) AND A
00014                  * BYTE IN HEX, RESPECTIVELY.
00015                  * THE INDEX REGISTER POINTS TO
00016                  * THE DATA AND IS ADJUSTED BY
00017                  * THE ROUTINES TO POINT TO THE
00018                  * NEXT BYTE BEYOND WHAT WAS
00019                  * PRINTED WHEN THEY RETURN.
00020                  *
00021                  *
00022                          OPT         S,P
00023        A078      START   EQU         $A078
00024        E07E      PDATA1  EQU         $E07E
00025        E0BF      OUT2H   EQU         $E0BF
00026        A07A      SAVE    EQU         $A07A
00027 A050                     ORG         $A050
00028                  *
00029 A050 FE A078 DUMP   LDX         START     GET START ADDR
00030 A053 A6 00   LOOP   LDA A       0,X       GET FIRST BYTE
00031 A055 81 04          CMP A       #$04      END OF LIST ?
00032 A057 26 01          BNE         MORE      NO, GET MORE
00033 A059 39             RTS                   END RETURN TO CALLER
00034                  *
00035                  * RTS CAN BE RTI IF FROM A
00036                  * INTERRUPTED ROUTINE
00037                  *
00038 A05A BD E07E MORE   JSR         PDATA1    PRINT NAME
00039 A05D E6 01          LDA B       1,X       GET LENGTH
00040 A05F FF A07A        STX         SAVE      SAVE CURR POSITION
00041 A062 EE 02          LDX         2,X       GET ADDR TO DISPLAY
00042 A064 BD E0BF HEX    JSR         OUT2H     BUMP BYTE
00043 A067 5A             DEC B                 DECREMENT LENGTH CTR
00044 A068 26 FA          BNE         HEX       DUMP NEXT
00045 A06A FE A07A        LDX         SAVE      RESTORE LIST POS.
00046 A06D 08             INX                   ADJUST LIST POSITION
00047 A06E 08             INX                   TO NEXT POSITION IN
00048 A06F 08             INX                   LIST
00049 A070 08             INX
00050 A071 20 E0          BRA         LOOP      PROCESS NEXT
00051                     END

START   A078
PDATA1  E07E
OUT2H   E0BF

SAVE    A07A
DUMP    A050
LOOP    A053
MORE    A05A
HEX     A064

ENTER PASS

S00B000044554D50202020203E
S11EA050FEA078A6008104260139BDE07EE601FFA07AEE02BDE0BF5A26FAFE71
S10BA06BA07A0808080820E0AF
S9030000FC
```