CHAPTER 19. MEMORY DUMP

Source code discussed here is in the UTILITY.BLK file, screens 28 to 30 and KERNEL86.BLK screen 87.

Source code and text data can be displayed or printed using commands like LIST, SHOW, and TYPE at the primitive level. Non-ASCII data like object code and numeric data cannot be display conveniently. The dumping utility provided in F83 allows the user to review the binary data in a conveniently formatted form. Large area of memory and large numeric data set can be either displayed on terminal or listed on printer.

19.1. THE DUMB DUMP

A simple and primitive dump command is included in the kernel of F83 system. It helps the user to debug the system before it is fully checked out.

```
: DUMP      ( addr len --- )      Dump a range of memory from addr in bytes.
      0 DO                        Set up the loop.
            CR                    Start a new line.
            DUP 6 .R SPACE        Print the address first.
            16 0 DO               Dump 16 bytes.
            DUP C@                Get one byte.
            3 .R                  Print one byte.
            1+                    Increment address.
            LOOP  16 +LOOP        Loop for more lines.
      DROP ;
```

19.2. THE SMART DUMP

More sophisticated dumping routines present data in both numeric and ASCII forms because in many cases the ASCII data are intermixed with binary data. It is convenient to have both types of display showing side by side. It is also nice if one can scan the memory forward and backward. The more elaborate dumping utility in F83 has many features not available in other systems. A sample of memory dump was shown in Fig. 3.1.

```
: .2              ( n --- )       Display a 2 digit number followed by a space.
      0                           Make a double number of n.
      <# # # #>                   Convert two digits.
      TYPE SPACE                  Type two digits with a trailing space.
      ;

: D.2             ( addr len --- ) Display a line of 2 digit numbers.
      BOUNDS
      OVER + SWAP.                Convert addr len to the limit-index format.
```

```
        ?DO                             Skip if len=0.
                I C@ .2                 Print one number.
        LOOP ;

: EMIT.                 ( char --- )    Emit one character if it is printable. Otherwise display a period.
        127 AND                         Mask off MSB.
        DUP                             Save a copy of char.
        BL 126 BETWEEN                  Is it between 32 and 126, the printable range?
        NOT IF DROP ASCII .             If not printable, replace char with '.'.
        THEN
        EMIT                            Send either char or '.' .
        ;

: DLN                   ( addr --- )    Dump 16 bytes of data starting at addr.  Display address
                                        first, then 2 sets of 8 bytes, followed by the ASCII equivalent.
        CR                              New line.
        DUP 4 U.R 2 SPACES              Display the address.
        8 2DUP D.2 SPACE                Display 8 bytes.
        OVER + 8 D.2 SPACE              Second set of 8 bytes.
        16 BOUNDS DO                    Scan 16 bytes.
                I C@ EMIT.              Print ASCII characters.
        LOOP ;

: ?.N           ( n1 n2 --- n1 )        If n1=n2, display a downwards pointer, otherwise display
                                        the number.
        2DUP =                          n1=n2?
        IF ." \/" DROP          Equal.  Display pointer and drop n2.
        ELSE 2 .R THEN                  Otherwise, display n2.
        SPACE           ;


: ?.A           ( n1 n2 --- n1 )        If n1=n2, display a 'v' symbol. Otherwise display one character.
        2DUP =
        IF ." V" DROP
        ELSE 1 .R THEN                  Display only one character.
        ;

: .HEAD         ( addr len --- addr1 len1 )     Display the header field of a dump, making it easy to
                                                index into the data portion of the dump.
        SWAP                            Get addr to top of stack.
        DUP -16 AND                     Mask off the least significant 4 bits in the address.
        SWAP                            Second copy of address.
        15 AND                          Preserve only the lower 4 bits.
        CR 6 SPACES                     Skip the address field.
        8 0 DO I ?.N LOOP               Print numeric field markers.
        SPACE
        16 8 DO I ?.N LOOP              Second set of field markers.
        SPACE
```

```
        16 0 DO I ?.A LOOP          ASCII field markers.
        ROT +                       Leave addr1 and len1 on stack, enabling full line display.
        ;

: DUMP              ( addr len --- ) Dump a range of memory specified on stack.  The dump is
                                     always in HEX, but the current base is preserved.
        BASE @ -ROT                 Save base under addr.
        HEX                         Use hexadecimal conversion.
        .HEAD                       Print the display header.
        BOUNDS DO                   Scan the memory range.
                I DLN               Display a line.
        KEY? ?LEAVE                 Quit if any key is pressed.
        16 +LOOP                    16 bytes per line.
        BASE !                      Restore the original base.
        ;


: DU            ( addr --- addr+64 ) Dump 64 bytes at the specified address and increment the
                                     addr so that next block of memory can be display next.
        DUP 64 DUMP                 Dump 64 bytes in a block.
        64 +                        Increment addr.
        ;

: DL                ( line# --- )    Dump the specified line in the current screen to verify
                                     non- printable characters.
        C/L *                       Starting character count.
        SCR †BLOCK                  Get the current block buffer address.
        +                           Address of the specified line.
        C/L DUMP                    Dump one line.
        ;
```