CHAPTER 6. TERMINAL INPUT AND OUTPUT

The source code discussed in this chapter is in file KERNEL86.BLK, screens 41 to 49.

Forth is an interpretive language which intimately interacts with the user through a CRT terminal. Terminal input and output control is a very important part of the Forth system, allowing the user to enter commands and data into the computer and display the results or messages on the CRT. Many Forth implementations have input/output commands coded in the host machine codes which access the terminal interface directly. These Forth systems are often stand-alone system which do not need support from a traditional operating system. F83 was design to run under the popular CP/M or MS-DOS system, so that it can be transported between different host computers. The terminal input/output commands in F83 thus utilize the CP/M or DOS BIOS routines to receive information from the keyboard and send information to the CRT display.

6.1. THE BDOS I/O CALLS TO THE OPERATING SYSTEM

The fundamental interface between Forth terminal I/O commands and the CP/M or DOS is the Forth word BDOS:

```
CODE  BDOS ( entry function --- return-value )      Load function code into C register
                                                    and entry parameter into D register. Call the BIOS.  Return
                                                    result are then pushed on the data stack.
        CX POP                                      Load function code into C register.
        DX POP                                      Load entry parameter into D register.
        33 INT                                      Call BIOS by a software interrupt. This is the MS-DOS
                                                    interrupt vector.  For CP/M, it is 224 INT.
        AH AH SUB                                   Clear the high byte in the AX register.
        1PUSH                                       Return with the result on stack.
        END-CODE
```

BDOS is not only used for terminal I/O, it can also be used for most of the disk I/O calls, making Forth I/O commands very neat and simple.

```
: (KEY?)   `( --- f )                   Return a true flag if the user presses a key.  Otherwise,
                                        return a false flag.
        0 11 BDOS                       Function 11 is the direct console I/O call.  Entry 0 specifies
                                        a console status command.  If no character is ready, 0 is
                                        returned. If a character is entered, FFH is returned.
        0<>                             Reversed the BDOS flag.
        ;

: (KEY)        ( --- char )             Wait until a key is pressed, and then return the ASCII code.
        BEGIN                           Enter the wait loop.
                PAUSE                   Release the CPU to other tasks so that the multitasking
                                        scheme can work smoothly.
```

```
        (KEY?)              Is a key pressed?
    UNTIL                   Yes, then exit the loop.  Otherwise, wait another round.
    0 8 BDOS                Entry parameter 0 specifies a console input function and
                            returns an ASCII code on the stack.
    ;

: (CONSOLE)     ( char --- )    Send the character on stack to the terminal for display.
    PAUSE                   Let other task have a run on the CPU.
    6 BDOS                  Call BDOS to send out the character.
    DROP                    BDOS always returns a number on the stack.  It has to be
                            dropped.
    1 #OUT +!               Increment user variable #OUT, keeping track of the output
                            character count.
    ;

: (PRINT)   ( char --- )    Send a character to the printer.
    PAUSE                   Always pause before an I/O operation because I/O operations
                            are generally slow.  CPU can then be freed to serve other
                            tasks or other users.
    5 BDOS                  Function 5 is the BDOS call for output to listing device.
    DROP                    Clear the stack.
    1 #OUT +!               Increment #OUT.
    ;

: (EMIT)    ( char --- )    Send the character to both the terminal and the printer.
    PRINTING @              Is the printing flag set?
    IF                      Yes.  Sent character to the printer.
        DUP (PRINT)         Print character.
        -1 #OUT +!          Back up #OUT so it will not be incremented twice.   THEN

    (CONSOLE)               Output to the terminal.
    ;
```

(KEY?), (KEY), and (EMIT) are the actual operators vectored to by KEY?, KEY, and EMIT. EMIT can vector to (PRINT) or (EMIT) like the regular CP/M system to activate the printer with the console.

6.2. TERMINAL OUTPUT COMMANDS

The following output words are all simple derivatives of EMIT and they do not need extensive comments:

```
: CRLF      ( --- )     Send a carriage return and a line feed to the console.
    13 EMIT             Carriage return.
    10 EMIT             Line feed.
    #OUT OFF            Clear the output character count.
    1 #LINE +!          Increment the line count.
    ;
```

```
: TYPE          ( addr len --- )        Display a string on the console.
        0 ?DO                           Repeat len times, but skip if len is zero.
                DUP C@ EMIT             Send one character.
                1+                      Increment character address.
        LOOP    DROP                    Clear stack.
        ;

: SPACE         ( --- )                 Send a space to console.
        BL EMIT  ;

: SPACES        ( n --- )               Send n spaces to console.
        0 MAX                           Eliminate negative counts.
        0 ?DO                           Repeat n times.
                SPACE
        LOOP   ;

: BACKSPACES    ( n --- )               Send n backspaces to console.
        0 ?DO
                BS EMIT
        LOOP   ;
```

6.3. INTERPRETING CONTROL CHARACTERS

Forth is capable of using most of the ASCII characters for word names. Only a few ASCII codes are reserved for system functions. Many fig-Forth system reserve the NUL (ASCII 0), CR (ASCII 13), and SP (ASCII 32) as delimiters for Forth words. The DEL (ASCII 127) is used to nullify the previously entered character, which is important in correcting typing errors. Other non-printable characters or control character can be used freely to name definitions. Because it is difficult to document the non-printable characters, embedding them in names is discouraged unless you want a very secured environment.

F83, on the other hand, provides a mechanism for you to implement special functions for control characters. EXPECT checks to see if an input character is a control character. When a function is defined for a particular control character, the function will be executed immediately when that character is entered on the keyboard. A jump table is maintained for all the 32 control characters. A few of them are used for special purposes which are defined as follows:

```
: BS-IN         ( n char --- n-1 )      Back up the input character buffer by dropping the character
                                        off the stack and decrementing n by 1. If n is zero, sound
                                        the bell instead. Used for ctrl-H.
        DROP                            Discard the character.
        DUP IF                          Is n=0?
                1- BS                   Yes. Decrement n and backspace.
        ELSE  BELL                      n=0. Sound the bell.
        THEN  EMIT                      Send either BS or BELL to console.
        ;
```

```
: (DEL-IN)   ( n char --- n-1 )      Backup the input and erase the previous character.  If n=0,
                                     sound the bell.  Used for DEL (127).
        DROP   DUP IF
                1- BS                Backspace.
                SPACE BS             Send a space and backspace again. Erase the previous
                                     character.
        ELSE BELL
        THEN EMIT   ;

: BACK-UP   ( n char --- 0 )         Erase the current line and set the character count to zero.
                                     Used for ctrl-U and ctrl-X.
        DROP                         Discard the character on the stack.
        DUP BACKSPACES               Backup to the beginning of the current line.
        DUP SPACES                   Erase all the characters on this line.
        BACKSPACES                   Backup
        0 ;                          Clear character count.

: RES-IN   ( char --- )              Reset the Forth system to a clean start again. (ctrl-C)
        FORTH                        Set default vocabulary.
        TRUE                         Force system abort.
        ABORT" Reset"                Abort with a message.
        ;

: P-IN          ( char --- )         Toggle the printer on or off. (ctrl-P)
        DROP    PRINTING @           Get the flag in PRINTING.
        NOT                          Complement it to turn the printer on or off.
        PRINTING !                   Store it back.
        ;

: CR-IN   ( m addr n char --- m addr m )   Finish input and remember the number of characters
                                           in SPAN. (ctrl-M or CR)
        DROP    SPAN !               Store n in SPAN.
        OVER                         Duplicate m.
        BL EMIT                      Send out a space.
        ;


: (CHAR)      ( addr n char --- addr n+1 )   Process a normal character by appending it to the
                                             input buffer.
        3DUP    EMIT                 Send character to console.
        +                            Addr+n, the memory address for the current character.
        !                            Store the character into the input buffer at addr+n.
        1+                           Increment n by 1 for the next character.
        ;

DEFER  CHAR                          CHAR will be vectored to (CHAR).
DEFER  DEL-IN                        DEL-IN will be vectored to (DEL-IN).
```

```
VARIABLE  CC                    CC will be used to point to the current control character
                                table.
CREATE  CC-FORTH                The control character table which can handle each control
                                character as a special case.  It is actually an execution array
                                which is indexed into by EXPECT to do the right thing
                                when it receives a control character.
        ]                       Enter compilation mode to compile 32 execution address for
                                the 32 control characters.
CHAR   CHAR   CHAR   CHAR   CHAR   CHAR CHAR   CHAR
BS-IN  CHAR   CHAR   CHAR   CHAR   CR-IN  CHAR   CHAR
P-IN   CHAR   CHAR   CHAR CHAR   BACK-UP CHAR   CHAR
BACK-UP CHAR  RES-IN CHAR  CHAR   CHAR   CHAR   CHAR
        [                       Reenter the execution mode.
```

6.4. MORE SOPHISTICATED INPUT COMMANDS

KEY is the most elementary word to accept keyboard input. It simply gets a character and puts its ASCII code up on the data stack: not a very intelligent word. Once we have the control character table, we can build a very intelligent input definition which can respond to many control characters to do a wide range of different things in response to our keyboard strokes. This definition is EXPECT:

```
: EXPECT     ( addr len --- ) Get a string from the terminal and place it in the buffer at
                              addr specified.  Perform a limited amount of line editing.
                              Save the number of characters input in the variable SPAN.
                              Process control characters as specified by the control
                              character table pointed to by CC.
        DUP SPAN !                  Save len in SPAN.
        SWAP 0                      Stack is now:  len addr 0  ---
        BEGIN                       Start the input loop.
              2 PICK                Copy len to top of stack.
              OVER -                ( len addr count #left --- )
        WHILE                       If all characters have been received, exit the loop.  If #left is
                                    not 0, continue on.
              KEY                   Get one more character.
              DUP BL <              Is it less than 32, i.e., a control character?
              IF                    Yes. A control character.
                    DUP 2*          Offset into the CC table.
                    CC @ +          The table entry address.
                    PERFORM         Execute the CC table entry.
              ELSE                  Not a control character.
                    DUP 127 =       Is it a DEL?
                    IF DEL-IN       Yes. Do delete the prior character.
                    ELSE CHAR       No.  A regular character.  Append it to the input buffer.
                    THEN
              THEN
        REPEAT                      End of string input loop.
        2DROP DROP                  Clear the stack.
```

```
        ;

: TIB           ( --- addr )        Get the address of the terminal input buffer.
        'TIB @                      TIB is vectored through 'TIB.
        ;

: QUERY         ( --- )             Get an input stream of text from the terminal and store it in
                                    the terminal input buffer. Prepare the system to interpret this
                                    input text.
        TIB 80 EXPECT               Receive upto 80 characters into the terminal input buffer.
        SPAN @                      Get the actual length of the input stream, which may be less
                                    than 80.
        #TIB !                      Store it in #TIB so that the text interpreter will know when
                                    the text is exhausted.
        BLK OFF                     Clear BLK so that the text interpreter will use the terminal
                                    input buffer for text input.
        >IN OFF                     Clear the character pointer to start from the beginning of the
                                    terminal input buffer.
        ;
```

QUERY is the Forth input word at the highest level. It waits on the user to type a line of text on the keyboard. The line is terminated either by receiving 80 characters from the keyboard or by receiving a carriage return key. The line of text is stored in the terminal input buffer. All the pertinent parameters are set so that the text interpreter can take over and interpret or execute the commands given in the input line.

6.5. STRING COMMANDS

Screens 41 to 43 are a set of commands to operate on strings in memory. A string in Forth is a sequence of ASCII characters preceded by a byte count. A string may have zero to 255 characters. It is generally identified by the address of the count byte. However, most string commands require the address of the first character in the string as argument, not the address of the count byte. String commands use the following generalized syntax:

<source addr> <dest addr> <length> <string command>

Destination address is optional in cases of single string operations.

Most of the string commands are standard Forth-83 words and their definitions are simple and straightforward. I will only list here their functions and stack parameters:

TABLE 6.1. STRING COMMANDS

COUNT (addr --- addr+1 len)	Convert the string address to address-length representation.
LENGTH (addr --- addr+2 len)	Return address-length for long strings whose character count is 16-bits.
FILL (addr len char ---)	Initialize a string to char.
ERASE (addr len ---)	Initialize a string to NUL.
BLANK (addr len ---)	Initialize a string to blanks.
MOVE (sour dest len ---)	Move a string without overlapping.
UPC (char --- char')	Convert a character to upper case.
UPPER (addr len ---)	Convert s string to upper case.
-TRAILING (addr len --- addr len')	Delete trailing blanks from a string by changing its length.
COMP (sour dest len --- n)	Compare source string with destination string. Return -1 if source<destination. Return 1 if source>destination. Return 0 if strings are the same.
CAPS-COMP (sour dest len --- n)	Compare two strings regardless of character cases.
COMPARE (sour dest len --- n)	Compare two strings. If CAPS is true, convert to upper case before comparing.

Figure 6.1 Representation of strings