Chapter 15
Forth 'BLOCK' Environment

Before the advent of sophisticated operating system environments as that available on the Amiga, Forth systems have been uniquely self-sufficient in their approach to utilizing mass-storage.

The resulting 'BLOCK' environment has been implemented as a pseudo- standard for disk-interfacing, and as such, does provide a small degree of transportability for high-level forth code.

It is important to note, however, that SCREEN files, those created under BLOCK, are not compatible with AmigaDOS text files, being defferent in format (SCREENs do not have end-of-lines).

If you want to convert your BLOCK files to text you can do so using BLOCK2TEXT which is described at the end of this chapter.

AmigaDOS Incompatibilities

Newcomers should note that files produced under the 'BLOCK' environment are not compatible with AmigaDOS text-based programs (editors, word-processors, spoolers, sorters, etc.). For example, they cannot be listed (or printed) via the AmigaDOS TYPE command. For those that wish to use the BLOCK environment, JForth provides both a line editor and a full screen editor, each described below.

Source files are typically a*at least* 3 times larger with BLOCK files, since all white space on a SCREEN is actual ASCII blank characters; there are no end-of-line characters.

The management of source code is often cumbersome in SCREEN files as the programmer is forced to reference sections of the file via SCREEN NUMBERS, sequentially numbered 1024-byte 'blocks' of characters, with no standard means of referencing other files.

For these and other reasons, we strongly discourage the use of SCREEN files for Amiga development work; it is provided for compatibility with existing source code and for those that prefer SCREENS.

JForth will support the standard 'BLOCK' environment, after compiling certain source files; but has been specifically designed to fully utilize the AmigaDOS file-system and accompanying ASCII text files. Certain JForth tools/utilities are not available under the 'BLOCK' environment:

1. TYPEFILE - since SCREEN FILES do not contain end-of-line characters, the output from thisASCII-file oriented program is not readable.
2. READLINE - again rendered unusable, for the same reason.
3. FILE? - not supported for words compiled from SCREENs, it requires end-of-lines.
4. ASM - The Forward Assembler is also line-oriented.
5. DEBUGGER - JForth can only compile in DEBUG mode within ascii files.

JForth supplied SCREEN utilities

Support for the BLOCK environment is provided in 3 files:

1. BLOCK - this file provides the read/write, open/close, and display primitives, along with the necessary buffer management' words.
2. EDITOR - loads a 'line-editor' similar to that promoted by FIG (Forth Interest Group). Defines the EDITOR vocabulary.
3. SCRED - loads the JForth SCReen EDitor that fully interfaces with the above line-editor. Extends

the EDITOR vocabulary. Line editor commands may be issued within SCRED.

Issuing the command INCLUDE JDEV:SCRED will cause these files to be compiled; they will require about 25K of dictionary area.

Note that the command INCLUDE JDEV:EDITOR will load only BLOCK and EDITOR, saving about 10K, but providing only line-editing capabilities.

Line Editor Operation and Glossary

To begin an editing session using the line editor, open the screen file via:

```
OPEN-SCR <screen-filename>
```

To enter the EDITOR vocabulary, making available the words listed below, enter:

```
EDITOR
```

To both display SCREEN #1 and select it for editing, type:

```
1 LIST
```

The EDITOR commands may now be used to modify the source. As changes are made, the user may SAVE or SAVE-BUFFERS to write changes to disk, but these changes may not become permanently applied to the file until CLOSE-SCR is executed.

Any time while the SCREEN file is open, you may LOAD the file, specifying the SCREEN number to start LOADing from on the stack.

An example session might be...

```
OPEN-SCR ram:myScreens    \ will ask create? if doesn't exist
EDITOR                    \ expose the EDITOR vocabulary
1 LIST                    \ list out screen 1
1 CLEAR                   \ empty out the entire screen
1 P : Hello  ( -- )       \ enter a small definition
2 P   >newline ." Hello, world!" cr
3 P ;
L                         \ LIST out the entire screen
1 LOAD                    \ LOAD the screen (compile it)
HELLO                     \ execute the word
CLOSE-SCR                 \ all done with the file, tell AmigaDOS
```

The commands in the EDITOR vocabulary...

B (-- , Back-up the cursor by the number of characters at PAD.)

C (-- Copy the following text, inserting it at the cursor.)

```
C INSERT THIS WHERE THE CURSOR IS
```

D (line# -- , Delete line#, but place its text at PAD.)

E (line# -- , Erase line# with blanks.)

F (<line> -- , search for line that follows)

Search forward from the cursor position, until the text is found, leaving the cursor just past the text. If the text is not found, an error message is printed, and QUIT executed.

```
F THIS STRING
```

H (line# -- , Hold line# at PAD)

Leave it untouched in the SCREEN.

`I  ( line# -- , Insert the line at PAD before line# )`

Move line# and all lines below it down. Line 15 is lost.

`L  ( -- , Relist the current screen.)`

`M  ( n -- , Move the cursor N characters.)`

Negative is to the left. Display the final destination line.

`N  ( -- , Find next occurrence of string last found by F or N.)`

`P  ( line# -- , Put new text on a line.)`

```
    3 P THIS GOES TO LINE 3.
```

`R  ( line# -- , Replace line# with the text in PAD.)`

`S ( line# -- , Spread the screen at line#)`

Move it and all lines below it down. Line# becomes blank. Line 15 is lost.

`T  ( line# -- line# , Type line# on the standard EMIT device)`

Save its text in PAD.

`X  ( -- , Delete the next occurrence of the following text.)`

```
    X THIS TEXT
```

`CLEAR   ( scr# -- , CLEAR screen and select for editing.)`

`COPY  ( scr1 scr2 -- , COPY screen SCR1 to SCR2.)`

`DELETE  ( #chars -- , DELETE #chars before the cursor.)`

`EMPTY-BUFFERS  ( -- , Empty current contents of all buffers. )`

`SAVE-BUFFERS   ( -- , Write any updated buffers to disk.)`

Leave in buffer.

`SAVE  ( -- , Alias for SAVE-BUFFERS.)`

`TOP   ( -- , Move the cursor to the top of the screen.)`

`TILL  ( <line> -- , Delete all text from cursor to located text)`

```
    TILL THIS TEXT
```

`MORE-SCREENS ( #scrs -- , Add #SCRS to current SCREEN file size)`

SCRED ... the JForth SCReen EDitor

SCRED is a sophisticated editing tool, useful when dealing with SCREEN files. It fully interfaces with the EDITOR line-editor, providing instantaneous access to and from the editing environment.

SCRED incorporates a command line feature, from which any editor or forth command may be executed. This is particularly useful in using line-editor commands while in SCRED.

A SCRED editing session may be invoked with:

```
SCRED <screen-filename>
```

This command will perform the above described line-editor OPEN-SCR, enter the EDITOR vocabulary and display SCREEN 1 of the file.

Once SCRED is entered, the user is presented with a text menu of possible single-stroke key operations. At the top is listed the current screen number and filename while on the left side are the screen line numbers, for use with the line-editor.

To the right is displayed the status of INSERT and WRAP. When SCRED is in INSERT mode, characters typed will 'push' existing characters over, otherwise existing characters are simply overwritten. The INSERT mode may be toggled via a key-stroke listed in the menu.

The WRAP parameter indicates the number of lines that will be affected by the INSERT mode (or when characters are BACKSPACEd). It is set from the command line as follows: 3 WRAP ! or 0 WRAP ! .

```
1 WRAP !
```

The 'Forth Cmd' key will cause the cursor to move below the displayed screen to the 'command line' at the bottom of the window. There it will display the message '<SCRED>:' informing the user that it is waiting for input.

At this point, the user may type in any FORTH or EDITOR command; when the command finishes, the cursor will return to the screen, available for more editing. If the command causes the displayed screen contents to change, its appearance will be updated before the cursor returns.

If the command generates an error, causing the system to execute QUIT, the message 'any key QUITs' will appear in the command line until the user presses a key. This gives him the chance to see any error message in the command line before the SCRED window closes.

SCRED may be exited by typing

```
QUIT
```

or

```
END-ED
```

from the command-line, or simply typing the control key for the EXIT function.

NOTE that *SCRED does not close the SCR-FILE when it exits*. This not only allows the programmer to LOAD the screen(s) he just edited, but also quickly get back to the same SCRED environment by typing:

```
SE
```

This command will reopen the SCRED window to the same screen and cursor position it was last on when SCRED was exited.

Once the programmer has finished with the file and exited SCRED, he should finally close the file with

```
CLOSE-SCR
```

It is this command that tells AmigaDOS to make all the changes permanent on the disk.

BLOCK2TEXT

If you decide you don't want to use BLOCKs, we don't, you can convert your BLOCK file to a normal TEXT file. This will let you use normal text editors like Textra and to use Amiga DOS commands like TYPE. A facility has been provided to do that. It is in the file JDEV:BLOCK2TEXT. To convert a file called PROJECT.blk to a file called PROJEECT.txt, enter:

```
INCLUDE JDEV:BLOCK2TEXT
BLOCK2TEXT PROJECT.blk PROJECT.txt
```