Pegasus Software
P.O. Box 10014
Honolulu, Hawaii 96816

September 29, 1981

Enclosed is your updated copy of the FBASIC compiler. I would appreciate hearing your comments and suggestions concerning the compiler package.

The compiler was not originally designed to produce ROMable code. We hope to fully support this feature in the future. We have several FBASIC users who are using the compiler to produce ROMable code by avoiding some features of the language.

The troublesome statements are as follows:

The FOR statement is self modifying.

The END statement is self modifying. The RETURN statement may be used instead if the stack is properly maintained.

Arrays must be set to absolute addresses in order to separate them from the code module.

The cursor position counter for print formatting is currently within the code module. Therefore the POS, SPC, and TAB functions will not produce the proper results.

I hope this information is of help to you. If you have additional questions please don't hesitate to write or call me (808) 735-5013.

Sincerely,

Richard Foulk

New features of the FBASIC Compiler Version 1.12 update:

When specifying a file name as outlined in the manual, you can now, at your option, select the disk drive to find it on. This is done by preceding the file name with a letter corresponding to the desired drive, and a colon after the letter.

```
B:FILE2
```

In the above example the compiler is instructed to find the file "FILE2" on drive B. The default drive is the one currently in use. That is, the drive the compiler itself was called from.

The INPUT statement is supported with this version of the compiler. It is of the same general form as used with OSI BASIC, except that multiple arguments are not supported.

```
INPUT A
INPUT "Enter a number"; NUM
```

No question mark is printed as a prompt. Therefore the user has full control of the prompt used, if any. As in the above examples the prompt message is optional.

The INPUT statement does not print REDO FROM START on non-numeric input, but instead flags this condition and leaves it up to the user to handle the error. The length of the user input is returned in the processors Y register. And non-numeric input is flagged by the Y registers high bit set.

```
INPUT A
T=.Y       :  REM see manual for register access
IF T>128 THEN PRINT "IMPROPER INPUT"
LN=T AND 127  :  REM get the length of input
```

The input is buffered at $2E79, so that non-numeric input may be accessed by the programmer. This buffer is used by the operating system to buffer the directory when searching for a file, it is not used otherwise.

At runtime the INPUT statement allows simple line editing. Both the RUBOUT and Underline (shift-O on video systems) can be used to delete characters. Control-U may be used to kill the whole line, instead of OSI's use of the commercial-at symbol (@).

The RND() function is supported with this release. In the example:

```
A=RND(10)
```

the RND() function will return a number between 0 and 9.

Also supported are the SPC(), POS(), and TAB() functions. These work just like their OSI BASIC counterparts.

```
PRINT SPC(20);   :  REM print 20 spaces
PRINT TAB(20);I  :  REM move to the 20th column & print I
A=POS(0)         :  REM get the current location of the cursor
```

The MOD function has also been added. In the example:

```
A=B MOD C
```

A is set to the remainder of the division of B by C. This is the same as the MOD function found in the larger Microsoft BASICs and in Pascal and other languages.

In response to a request from an advanced user, the facility to list the line numbers of a program along with their absolute addresses has been added to the compiler. This feature is enabled by typing a space following the "Address:" prompt. The compiler lists each line number followed by its address to the console. To get a listing to the printer use the IO command before invoking the compiler, or hit control-P during one of the compilers prompts. If a line is to be called from outside an FBASIC program the line should be referenced within the program (have a GOTO etc. going to it) to insure that the code can be safely entered there.

Last, but definitely not least, a Cross-reference utility has been added to the system. This is to help you keep track of variable usage and line referencing on larger BASIC and FBASIC programs. It is called from the operating system with the XQ command. As a printer device the program selects the line printer parallel port which corresponds to device number 4 in OSI BASIC. If your printer doesn't currently respond to that device number we suggest you adjust the output device table within the operating system so it does.

Your comments and suggestions on the compiler and it's various utilities are welcome. Help us make the next version even better!

Thank you.