MenuBASIC™

Program Development Software

For Interactive Programming
and Automatic Command Implementation

For Use with the 1722A Instrument Controller
and the 1752A Data Acquisition System

FLARE™
Fluke Application Software

Model Number 17XXA-903 (Compiled Version)
Model Number 17XXA-905 (Extended Version)
P/N 792960
March 1896

TABLE OF CONTENTS

Page

Introduction 1
How MenuBASIC Works 2
Default Device 3
System Requirements 3
Hardware Requirements 3
Software Requirements 4
System Software 4
Language Software 4
MenuBASIC Software 5
How To Use This Manual 5
Notation Conventions 6
Getting Started 8
Before You Begin 8
To Create A Working Disk 9
A Sample Program 10
Entering MenuBASIC 15
Configuring The E-Disk 16
Configuring The E-Disk For Compiled BASIC. 16
Configuring The E-Disk For Extended BASIC. 17
Writing And Editing A Program 20
Creating A New Program: The C Command 20
Entering The Editor: The E Command 21
Using The Editor 21
Command Mode 21
Moving Blocks Of Text 23
Exiting The Editor 24
Programming Errors 24
Beautifying Your Program: The B Command 25
Saving And Running A Program 26
Saving Your Program: The S Command 27
Recalling Your Program: The O Command 27

TABLE OF CONTENTS (CONT)

Page

Listing Your Program: The L Command 28
Running Your Program: The R Command 29
Compiling Your Program: The CO Command 29
Changing the Default Library: The LI Command .. 30
Chaining Programs Together 30
Using The Language 31
BASIC Language Syntax 31
BASIC Compiler and Extended BASIC Compiler Options 32
Commands That Should Be Avoided 33
SET SHELL 33
ASSIGN SYSTEM DEVICE 33
EXECUTE 33
Linking Predefined Modules 34
Errors 37
Program Errors 37
MenuBASIC Errors 38
Using A Printer 39
Setting Up A Printer: The SP Command 39
Selecting The Printer Port: The PP Command 39
Printing Your Program: The P Command 40
Printing Error Messages: The PE Command 40
Printing Other Data 40
File Utilities: The F Command 41
Automating Start-Up And Other Functions 41
Appendices
A. The Editor: A Quick Reference Guide A-1
B. Program Listings B-1

INTRODUCTION

MenuBASIC is a language tool designed to simplify program development for the Fluke 1722A Instrument Controller and the 1752A Data Acquisition System. MenuBASIC provides a simple, interactive environment with a three page menu for command selection. Using only a few key strokes, the programmer can command MenuBASIC to automatically compile, link and run programs. With menu selection, single key commands can perform functions that may require long command strings on other systems.

There are two versions of MenuBASIC: one for users of Fluke Compiled BASIC (CBASIC) and another for users of Fluke Extended BASIC (XBASIC). Because MenuBASIC functions similarly in both applications, this manual is written for users of both CBASIC and XBASIC. Both versions of MenuBASIC provide access to the powerful BASIC language, complete with advanced structures that make programs more logical, more readable and easier to debug.

MenuBASIC also provides features that simplify systems development such as full IEEE-488 support and a multi-level interrupt structure. A powerful editor is provided that can search, replace, cut and paste, and interactively merge programs.

MenuBASIC provides complete file and serial port control, allows you to link other BASIC, FORTRAN or Assembly Modules, and to create command files to automate common tasks.

How MenuBASIC Works

MenuBASIC software utilizes standard Fluke programs, including the Floppy Disk Operating System (FDOS), Editor (EDIT.FD2), File Utility Program (FUP.FD2) and Set Utility Program (SET.FD2). In addition, the CBASIC version uses the standard BASIC compiler (BC) and linking loader (LL). The XBASIC version uses the extended BASIC compiler (XBC) and the extended linking loader (XLL).

MenuBASIC automatically creates three versions of a file. The first version is the source file. The source file is created when you invoke the editor and is given the extension .BAS. This is the file that you can edit.

The second version of the file, the object file, is created when you exit the editor. The compiler takes the source file and changes it into a format that the linking loader can use. The object file has a .OBJ extension in CBASIC and a .OBX extension in XBASIC.

The final version of the file is an executable file, also created when you exit the editor. The executable file is required for the operating system to be able to run your program. The linking loader creates this file, gives it a .FD2 extension and automatically deletes the object file.

MenuBASIC is simple to use because it automatically does all of these steps for you. However, if you change a file outside of the MenuBASIC program, it will not be automatically compiled and linked and your existing executable file will not be changed. In order to run the new file, you will have to use the CO command or go through the steps required to compile and link the program yourself. See the 1722A or 1752A System Guide for more information on compiling and linking a program.

Two BASIC programs make up MenuBASIC: MenuBASIC itself, and ATOCMP, the automatic compilation program. During most operations, MenuBASIC is the system shell. MenuBASIC sets itself to the shell so that control will always return to MenuBASIC, rather than FDOS, when a program has finished execution.

After creating or editing a program, the ATOCMP program creates two command files: AUTOBC.CMD, which performs the compilation and AUTOLL.CMD, which performs the linking. These command files are then automatically executed and control is returned to MenuBASIC.

Default Device

MenuBASIC performs all operations on the E-Disk™. Because E-Disk is faster and provides more memory than a floppy disk, automatic compilation is faster and more efficient. MenuBASIC assigns the E-Disk (ED0:) as the system device.

Note, however, that most MenuBASIC commands (such as Save and Old) will use the system device that was in effect when MenuBASIC was invoked. For example, if you insert a MenuBASIC disk and type **mbasic** at the FDOS> prompt, once MenuBASIC is loaded, ED0: will be the system device, but MenuBASIC will consider the floppy disk (MF0:) to be the default device.

SYSTEM REQUIREMENTS

Hardware Requirements

MenuBASIC requires a 1722A Instrument Controller or 1752A Data Acquisition System for proper functioning. In this manual, the 1722A and 1752A will both be referred to as the 17XXA.

MenuBASIC requires at least 450 blocks of E-Disk for operation. To meet this requirement, optional memory modules must be installed on the 17XXA. For programs with less than 50 blocks (25K bytes) of source code, the 17XXA-006 256K-byte RAM Expansion is adequate. If larger programs are anticipated, the 17XXA-007 512K-byte RAM Expansion is recommended.

Software Requirements

In order to power-up or restart the 17XXA and to invoke MenuBASIC, the following software must be installed on the system device:

- System Software
- Compiled or Extended BASIC Software
- MenuBASIC Software

System Software

System Software is shipped standard with the 17XXA, and includes the operating system (FDOS), the File Utility Program (FUP), and the System Editor. The following system software is required to run MenuBASIC:

FDOS2.SYS	FUP.HLP
MACRO.SYS	SET.FD2
FUP.FD2	EDIT.FD2

Language Software

The Compiled BASIC software package (17XXA-203) includes the BASIC Compiler (BC) and Linking Loader (LL). The Extended BASIC software package (17XXA-205) includes the Extended BASIC Compiler (XBC) and Linking Loader (XLL). The following software is required to run Compiled MenuBASIC:

BC.FD2 B$LOAD.OBJ
LL.FD2 BASIC.LIB
BSCRUN.FD2

The following software is required to run Extended MenuBASIC:

XBC.FD2 BSXRUN.FD2
XLL.FD2 BASIC.LIB

MenuBASIC Software

The programs provided on the MenuBASIC disk are:

MBASIC.FD2 HEXMOD.BAS
ATOCMP.FD2 HEXTST.BAS

HEXMOD.BAS and HEXTST.BAS are not required for MenuBASIC operation, but are required for the sample program provided in this manual.

HOW TO USE THIS MANUAL

This manual presents information on the use of MenuBASIC program development software. To help you to get acquainted with the manual, this section presents manual use instructions and a chart of conventions that are used in the Flare software manuals. The next section explains how to get started in MenuBASIC. To provide an overview of MenuBASIC operation, a sample program is included. Other sections explain how to use the editor to write and edit a program, and how to save, recall, compile, link and run your program. Tips on BASIC language syntax and options follow. Printer use and automated start-up are also covered. A quick reference guide for the editor and program listings for the sample program are presented in the appendices.

Notation Conventions

Fluke manuals use certain conventions to illustrate keyboard entries and to differentiate these entries from surrounding text. The braces, { }; brackets, [ ]; and angle brackets, < > are not part of the key stroke sequence and should not be typed in.

<xxx>	Means "press the xxx key". Example: <RETURN> indicates the Return key.
<xxx>/y	Means "hold down key xxx and then press y". Example: <CTRL>/C means to hold down the key labeled CTRL and then press the key labeled C.
[xxx]	Indicates an optional input. Example: [input filename] means to type the name of an input file name. If no file name is typed, a default name will be used.
xxx	Means to type the name of the input as shown. Example: BASIC means to type the program name BASIC as shown.
{xxx}	Indicates a required user-defined input. Example: {device} means to type a device name of your choice, as in MF0: for floppy disk drive 0.

(xxx) This construction has two uses:

1. As a separate word, (xxx) means that xxx is printed by the program. Example: (date) means that the program prints today's date at this point.

2. Attached to a procedure or function name, (xxx) means that xxx is a required input of your choice; the parentheses must also be typed in. Example: TIME(parameter) means that the word TIME must be typed in, followed by a parameter that must be enclosed in parentheses.

GETTING STARTED

This manual is written with the assumption that you are familiar with the Fluke 1722A Instrument Controller or the Fluke 1752A Data Acquisition System and either the Fluke Compiled BASIC (CBASIC) or Fluke Extended BASIC (XBASIC) language. If you are not, please read the following materials:

- Getting Started: A New User's Guide to the 17XXA Instrument Controller.
- The 17XXA System Guide.
- The CBASIC manual, or the XBASIC manual.

Before You Begin

Before you begin, you will need to make a backup copy of the MenuBASIC disk and put the original copy of the disk in a safe place.

To make copies of your disks, use the Touch-copy program (Tcopy). Tcopy is an easy to learn, menu-driven program that utilizes the Touch-Sense Display to transfer files between the 17XXA's file-structured devices. Tcopy is explained in more detail in the 17XXA System Guide.

As shipped, the MenuBASIC disk will not independently load and run. You must either create a MenuBASIC working disk or copy the MenuBASIC executable files onto the system device (SY0:).

To Create A Working Disk

Creating a working disk will allow you to load and run the MenuBASIC program on your 17XXA using only the working disk.

1. Turn on the 17XXA.

2. Load the system disk.

3. Type

   FDOS> tcopy <RETURN>

4. Follow the simple instructions, using the individual option, to copy the following executable files from the system disk onto your working disk:

FUP.FD2	EDIT.FD2
FUP.HLP	FDOS2.SYS
SET.FD2	MACRO.SYS

5. If you will be using CBASIC, copy the following files from your CBASIC backup disk:

BC.FD2	BSCRUN.FD2
LL.FD2	BASIC.LIB
B$LOAD.OBJ	

   If you will be using XBASIC, copy the following files from your XBASIC backup disk:

XBC.FD2	BSXRUN.FD2
XLL.FD2	BASIC.LIB

6. Now copy the following files from your MenuBASIC disk onto your working disk:

   MBASIC.FD2 HEXMOD.BAS
   ATOCMP.FD2 HEXTST.BAS

A Sample Program

A short tutorial is provided in this section to familiarize you with MenuBASIC. A program is compiled, linked and run automatically by the MenuBASIC routines.

The program HEXTST and the subroutine HEXMOD are provided on the MenuBASIC disk. HEXTST converts hexadecimal numbers into decimal numbers, calling the subroutine HEXMOD, which is responsible for the mathematical calculations. The subroutine HEXMOD must be linked to HEXTST for HEXTST to run correctly. Program listings for HEXMOD and HEXTST may be found in Appendix B.

To compile and run HEXTST.BAS the following steps must be performed:

- Compile the subroutine HEXMOD
- Compile the program HEXTST
- Link HEXTST and HEXMOD together
- Run HEXTST

1. To begin, enter MenuBASIC by typing:

   FDOS> mbasic <RETURN>

MenuBASIC will begin its start-up routines and will notify you if insufficient E-Disk space was allocated before MenuBASIC was invoked.

If this is the case, the screen will display:

You have XXX blocks of E-Disk allocated, but MenuBASIC needs at least 450 blocks.

Type a 0 to exit MenuBASIC and prevent E-Disk configuration, or enter number of blocks you wish to allocate?

2. Type in the number 450, and MenuBASIC will copy the files that it needs to run from SY0: onto E-Disk. It will also set MenuBASIC as the shell. While MenuBASIC is copying the files, the screen will flash the message: **Loading MenuBASIC System.**

3. The screen will now display the first of three menu pages. (See Figure 1 for an illustration of the first page of the menu.) Press <RETURN> twice to see the commands available on the other two screens.

Compiling And Linking HEXMOD

1. Since HEXMOD is an existing file, select the Old (recall) command at the Enter Command? prompt by typing:

   o hexmod

   This will recall the file HEXMOD.BAS. The message **User program: HEXMOD.BAS** will appear in the upper right-hand corner of the screen to indicate that HEXMOD is the active program.

2. At the Enter Command? prompt, enter the editor by typing

   e

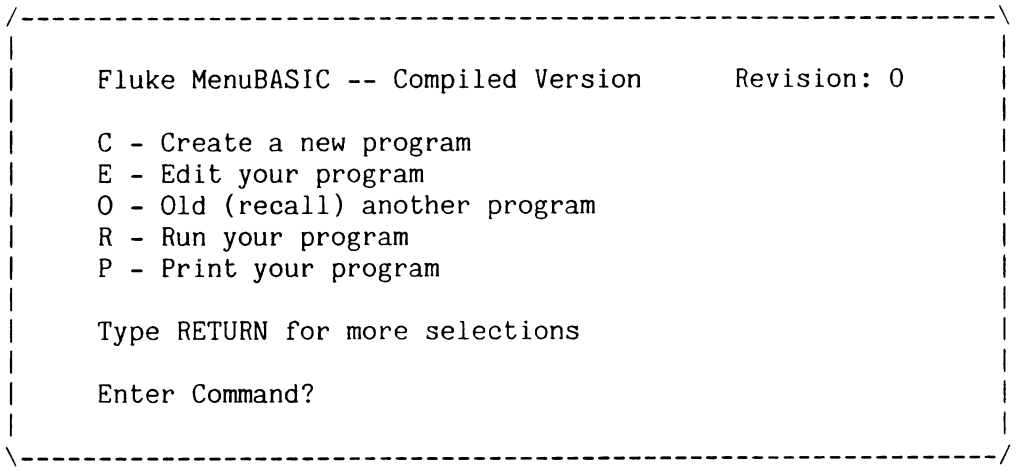

Figure 1. The MenuBASIC Menu Screen (first page)

This puts you into the system editor with the current user program, HEXMOD.BAS. Notice the first two lines of the program.

```
!# /nl/e
!# m
```

The first line tells MenuBASIC that the program has no line numbers and that it uses extended syntax. Extended syntax means that the source code contains extended language features such as continuation lines, statement labels, long variable names, true subroutines, and extended control statements. The second line tells MenuBASIC that the following program is a module, that it should not be linked, and that no .FD2 file should be created for the module. The rest of the program, HEXMOD.BAS, is an ordinary subroutine.

3. To exit the editor, type

   <CTRL>/C

   When you leave the editor it automatically compiles and links the program.

Compiling And Linking HEXTST

1. Recall the main program, HEXTST, at the Enter Command? prompt by typing:

   o hextst

   MenuBASIC will provide the following prompt:

   Do you wish to save MF0: HEXMOD.BAS?

2. Since you did not make any changes to the file, respond by typing no, or simply n.

If you had changed the file and wanted to save it, you would type y, or yes. The message on the right-hand corner of the screen now reads:

User program: MF0:HEXTST.BAS

3. Enter the editor at the Enter Command? prompt by typing:

   e

   Notice the first two lines of the program:

   ```
   !# /nl/e
   !#i hexmod
   ```

The first line tells MenuBASIC that there are no line numbers and that extended syntax is used, and the second line tells MenuBASIC to include the file HEXMOD when it links the program.

4. Exit the editor by typing:

   <CTRL>/C

When you leave the editor, it automatically compiles and links the program HEXTST and the subroutine that it calls, HEXMOD.BAS.

Running HEXTST

1. To run the interactive HEXTST program, type

   Enter Command? r

   The screen will look like this:

   Testing the sethex utility
   Enter a hex string (max four characters) ?

2. Enter any four of the following characters: the numbers 0 through 9, and the letters A through F. The program will respond with the corresponding integer.

3. To exit the program and return to the MenuBASIC menu, type:

   <CTRL>/C

   The program prints this information to the screen:

   !Abort at line 7 in module $MAIN$
   Type any key to continue

4. Type any key to return to the MenuBASIC menu.

If you try to leave MenuBASIC at this point, it will check to see if the current user program has been modified. If it has, MenuBASIC will ask if you want to save it. ALL MODIFIED FILES MUST BE SAVED, OR THE WORK WILL BE LOST.

Entering MenuBASIC

To enter MenuBASIC, insert your MenuBASIC working disk and type

```
FDOS> mbasic
```

MenuBASIC will begin its start-up routines. If E-Disk space was not allocated for your program before Menu-BASIC was invoked, MenuBASIC will ask you to configure the E-Disk. Program operation requires that at least 450 blocks of memory space be allocated as E-Disk.

Configuring The E-Disk

If the E-Disk was not configured before MenuBASIC was invoked, the following message will be displayed:

You have XXX blocks of E-Disk allocated, but MenuBASIC needs at least 450 blocks.

Type a 0 to exit MenuBASIC and prevent E-Disk configuration, or enter number of blocks you wish to allocate?

The number of blocks that you must allocate as E-Disk depends on the version of MenuBASIC that you are using.

Configuring The E-Disk For Compiled BASIC

With Compiled MenuBASIC, all available memory space should be allocated as E-Disk. For example, if you have a 17XXA-006 256K-byte RAM expansion module, at least 512 blocks are available to be configured as E-Disk. If you have a 17XXA-007 512K-byte RAM expansion module, at least 1024 blocks are available to be configured as E-Disk.

If there are important files on the E-Disk, enter 0 to prevent their loss. Use the File Utility Program (FUP) or the Tcopy program to save them to another device, and then configure the E-Disk. (See the 17XXA System Guide for more information on copying files.)

Configuring The E-Disk For Extended BASIC

With Extended MenuBASIC, you must decide how much space to dedicate for program space and how much for E-Disk space. If there are important files on the E-Disk, enter 0 and use the File Utility Program (FUP) or the Tcopy program to save them to another device, and then configure the E-Disk. (See the 17XXA System Guide for more information on copying files.)

To decide on the number of blocks to allocate:

1. Determine the amount of memory available:
   a. Take into account the 30K bytes provided as standard program memory.
   b. Add in the amount of additional memory available on the optional memory board.

2. Estimate the size of your program. Unless your program is unusually large, 100K bytes will be adequate.

3. Subtract the amount of program space from the memory available to find the E-Disk space available.

4. Multiply this number by 2 (2 blocks per K-byte) to arrive at the number of blocks to allocate.

Refer to Figure 2 for a sample configuration using a 17XXA with a 17XXA-007 512 K-byte RAM expansion.

It doesn't hurt to round down, so in this case, 850 blocks can be allocated as E-Disk.

After configuring the E-Disk, MenuBASIC will copy the files that it needs from SYS: to ED0:, set MenuBASIC as the shell and create a temporary data file. While MenuBASIC is copying the files, the screen will flash the message: **Loading MenuBASIC System.** MenuBASIC will then display its menu. See Figure 1 for an illustration of the first screen of the MenuBASIC menu. Hit <RETURN> to view the second screen of the menu. Hit <RETURN> again to view the third screen.

NOTE

MenuBASIC commands are not case-sensitive, and the program will accept the abbreviations y and n for yes and no.

Step 1a.	1b.		2.		3.		4.		
30K +	512K	-	100K	=	432K	x	2 blocks/K	=	864 blocks
Stand-ard Program Memory	Optional Memory		Program Space		E-Disk Space		Number Of Blocks Per K-byte		Number Of Blocks To Allocate

Figure 2. E-Disk Configuration Example

WRITING AND EDITING A PROGRAM

Creating A New Program: The C Command

The C command is used to create a new program. From the MenuBASIC menu, type:

```
Enter Command? c
```

MenuBASIC will ask you to enter the name of the program to be created. If an alternate file name extension is not specified, MenuBASIC will provide the extension .BAS.

Or you can type:

```
Enter Command? c {filename}
```

If you specify the file name of a program that already exists, MenuBASIC will ask if you wish to replace it. Answering yes will overwrite the original program with your new program.

Once the user program is created, MenuBASIC will invoke the editor. For more information on the editor, see the section, **Using the Editor**, and the appendix to this manual, as well as the 17XXA System Guide.

Exit the editor by typing:

```
<CTRL>/C
```

The program will be stored on the system device (ED0:) unless a device name is specified. When you return to the MenuBASIC menu, the program that you have just created is the current user program. Its name appears after the message, **User Program:**.

Entering The Editor: The E Command

To edit a program from the main menu, simply type e. This will allow you to edit the current user program.

Using The Editor

The editor is an easy to use screen-oriented editor with advanced capabilities such as searching, file merging and cut-and-paste. A few of the fundamentals are covered in this manual. A complete discussion of the editor's capabilities is provided in the 17XXA System Guide, and a quick reference guide is provided in Appendix A of this manual.

To move the cursor around in your program, use the arrow keys. To insert text at any position, just place the cursor where you want the text and type it in. To delete text, use the delete (DELETE) and delete character (DEL CHAR) keys. To delete the entire line to the right of the cursor, use the delete line (DEL LINE) key. To insert a new line, move to the end of the previous line and press <RETURN>.

Command Mode

Pressing the Escape key <ESC> puts you into Command mode, allowing you to access other, more powerful editing features, such as cursor and text manipulation, searching, and exiting the program.

To move the cursor to line n of your program, type:

<ESC> n G

To search for a pattern, type:

<ESC> /

The program will provide a slash at the upper left of the screen and the cursor will appear after the slash. Enter the string for which you are searching, followed by <RETURN>.

To move ahead to the next occurrence of the string, type

<ESC> n

To search backwards in the text, type:

<ESC> ?

The program will provide a question mark at the upper left of the screen and the cursor will appear after the question mark. Type in the string for which you are searching.

To repeat the search, type:

<ESC> n

Another feature is the ability to place invisible markers anywhere in the text. There can be 26 such markers in any file, one for each lower-case letter in the alphabet. This command places the marker n at the cursor position:

<ESC> mn

To return to position n after subsequent editing, use the command:

<ESC> `n

NOTE

The ` character is not an apostrophe, but it is the back quote character on the upper right of the top row of keys.

There is no provision for deleting markers. However, they are not recorded with the file and do not remain after the current editing session. Also, the same marker can be moved simply by placing it elsewhere. The editor will remember only the most recent placement.

Moving Blocks Of Text

The editor also allows you to remove (yank) and insert (paste) portions of text by storing the text in a buffer. This temporary storage location in memory is called a yank buffer. First, a copy is made of the text between the cursor location and a specified marker and stored in the yank buffer. The yanked text can then be placed at any location in the file. The text remains in the yank buffer until it is replaced by new text, or the editor is exited.

The yank buffer is especially useful if part of a program has inadvertently been left out, and its inclusion requires program restructuring. The yank buffer can be used to remove program sections and hold them in memory until the new section is written. Then they can be placed in the appropriate location in the text.

Another good use for the yank buffer is as a holding area for a frequently written line of code, such as a tightly formatted PRINT statement or a very long line that you do not want to re-type. See the 17XXA System Guide for more information.

A summary of the yank commands follows:

<ESC> y`n removes text from the cursor to marker n.

<ESC> p (lower-case p) places the contents of the buffer into the text following the cursor position.

<ESC> P (upper-case P) places the contents of the buffer into the text in front of the cursor position.

Exiting The Editor

To exit the editor, save the changes you have made, and return to the menu, type:

<CTRL>/C

There will be a short delay while MenuBASIC checks for program errors. If any errors are found, the program will display them before returning to the menu. See the **Errors** section for more information.

If you want to discard the changes you made during the editing session, press the escape key, followed by a colon (:). The cursor will move to the top of the screen. Type the letter q, and an exclamation point:

<ESC>:q!

Programming Errors

Any errors that you make in writing or editing your program will appear on the screen when you exit the editor. Refer to the CBASIC or XBASIC manual for an explanation of these errors.

MenuBASIC may also provide a message that certain words are not recognized by the compiler. The cause of these errors is usually mistyping. If they refer to modules or application programs that are called, it may be that these programs were not defined. Refer to the section, **Linking Predefined Modules,** for more information.

The following errors usually mean that your E-Disk is full:

I/O error: No room on device — Stop
I/O error: Read/write past physical end of file — Stop

The File Utilities command (F) will allow you to remove files from your E-Disk so that there will be sufficient room. See the heading, **File Utilities: The F Command,** for more information. Selecting F puts you into FUP, the File Utility program. The MenuBASIC program does not have a delete command.

Beautifying Your Program: The B Command

The Beautify (B) command formats a program so that it is easier to read and debug. The B command performs the function of "Pretty Print" in other systems. Selecting B from the MenuBASIC menu allows you to insert the number of spaces (3 or 4 per tab stop) and the tab stop (indent or start on the left) desired. MenuBASIC notifies you when the file has been successfully translated.

SAVING AND RUNNING A PROGRAM

Like BASIC, MenuBASIC allows the user to work with one program at a time. The current program is called the user program and its name is displayed in the upper right-hand corner of the menu screen. When a new program is created or recalled, the message is updated to reflect the current user program. When the Save command is executed, the program is stored under the current user name. If no device name is present in the user program name, the default device name will be used. The default device will be the device that was the system device when MenuBASIC was invoked.

NOTE

Unlike BASIC, MenuBASIC allows you to work with multiple programs in memory (E-Disk). While this is convenient, if you get too many programs onto the E-Disk, you may run out of space.

Saving Your Program: The S Command

The S command is used to save the current user program. The program will be saved under the current user program name. If no device is specified in the user program name, the program will be stored on the device that was the system device when MenuBASIC was invoked.

A program is automatically saved when you switch from one user program to another using the C or O command and answer yes to the prompt:

Do you wish to save (filename)?

Recalling Your Program: The O Command

To recall a program for use, use the O (Old) command. You may specify the program name by typing:

Enter Command? o {program name}

If you specify a device name as part of the program name, MenuBASIC will bring in the program from that device. If no device is specified, MenuBASIC will first check the E-Disk, and, if the program is not found, will check the default device. If the program is found, it will become the current user program.

Examples:

```
Enter Command ? o
Enter name of program to be olded ? cat
```

will look for the program cat.bas, first on E-Disk, and then on the default device. Cat.bas will become the current user program.

```
Enter Command ? o mb0:test.001
```

will look for the program test.001 on bubble memory (MB0:). Mb0:test.001 will become the current user program.

Listing Your Program: The L Command

To obtain a directory listing of your files, select the L command from the menu. All files on ED0: (the system device) will be listed to the screen. Pressing any key will then cause MenuBASIC to list the files on the default device to the screen. Type any key to return to the MenuBASIC menu.

Running Your Program: The R Command

You may execute a program using the R (Run) command. If MenuBASIC has not compiled the program, it will automatically compile it before attempting to execute it. Typing:

```
Enter Command ? r
```

will run the current user program, or you may specify the program to be run by typing:

```
Enter Command ? r {program name}
```

If a device name is specified in the R command, MenuBASIC will search for the program on that device.

Compiling Your Program: The CO Command

When you exit the editor, your program is automatically compiled and linked for you. With long programs, this can be a slow process. The CO (Compile) command allows you to break your program into smaller subroutines and compile and link them without having to enter the editor.

For example, HEXTST.BAS is the main program that calls the subroutine HEXMOD, which is contained in the module HEXMOD.BAS. If you compile HEXMOD with the CO command, an object file (.OBJ or .OBX) is produced that is required for HEXTST to perform its link. Compiling and linking HEXTST with the CO command produces an executable file (.FD2) that you can run using the R (Run) command.

NOTE

HEXTST could have been compiled and linked with the R command, as long as there was not an executable version of HEXTST already present. If an executable file is present, it will be run and HEXTST will not be re-compiled.

Changing the Default Library: The LI Command

A library is always required for operation of the MenuBASIC compiler. BASIC.LIB is the default library when MenuBASIC is powered-up. The Library (LI) command lists the current default library to the screen and prompts you to enter a new default library. Enter the default library name, omiting the .LIB extension. MenuBASIC provides the extension for you. If you input a library name that is not found by the program, the default library will not be changed.

Chaining Programs Together

Chaining is a method that allows one program (Program A) to execute another program (Program B). There are two ways to chain two programs together. The first requires that an executable (.FD2) version of Program B be available. Program A can then execute a command to get Program B from the floppy disk and execute the program. In this example, the code in Program A might read:

```
12400 IF I% < 0 THEN RUN "MF0:B"
```

Note that this instruction has no effect on MenuBASIC's user program. If A was the user program before this instruction was executed, it will continue to be afterward.

The second chaining method doesn't require an executable (.FD2) file. MenuBASIC's Execute (EXEC) command is used in combination with the R command. For example, if a program were to run another program, phase2.002, stored in bubble memory, the BASIC command in the program would be:

```
32000 EXEC "MBASIC" WITH "R MB0:PHASE2.002"
```

This instruction does affect the current user program. The program chained to (in this case, phase2.002) will become the MenuBASIC user program when the program is executed. (See the heading, **Commands That Should Be Avoided**, for more information on the EXEC command.)

USING THE LANGUAGE

BASIC Language Syntax

MenuBASIC allows the use of both Standard Syntax and Extended Syntax. Standard Syntax is the BASIC syntax used in the interpreted version of BASIC. Extended Syntax is used in CBASIC and XBASIC to free the programmer from the conventional BASIC restrictions of single lines and two-character variable names.

Compiler and linker options are comment lines embedded in a program that provide program information to MenuBASIC. These options may fill as many lines as needed, but must begin within the first three lines of the program. MenuBASIC compiler options are covered in the following section. Linker options are covered under the heading, **Linking Predefined Modules.** For detailed information on compiler and linker options, see the CBASIC or XBASIC Manual.

BASIC Compiler and Extended BASIC Compiler Options

The compiler automatically assumes Standard Syntax and line numbers. However, CBASIC and XBASIC support a number of compiler options that can be accessed with MenuBASIC:

- Extended Language Syntax - The /E Option
- Integer Conversion - The /I Option
- No Line Numbers - The /NL Option
- No Markers - The /NM Option

Syntax

```
!# {menubasic option}
```

Parameters

- Compiler options are preceded by a slash (/).
- All Compiler options must be on one line.
- Compiler options are not case-sensitive.

Example

A program using extended syntax, no line numbers and integer conversion, might look like this:

```
! Automatic Calibration Program by T.R.D.
!# /nl/e/i
! Begin Instrument Set-Up
.
.
.
```

Commands That Should Be Avoided

MenuBASIC is a program development tool that is, by design, not completely bullet-proof. Programs developed by MenuBASIC will run with no errors under the fully Fluke-supported CBASIC or XBASIC system. However, the lesser-used BASIC commands: SET SHELL, ASSIGN System Device, and EXEC (Execute) can short-circuit MenuBASIC's operation and should be avoided.

SET SHELL

MenuBASIC sets the shell to itself for proper operation. If you must use the Set Shell command, have your program return the shell to EXEC MBASIC when it has finished execution. Otherwise, you will either be exited to FDOS or have to reboot the system.

ASSIGN System Device

If MenuBASIC is not resident on the device that you assign as the system device, this command can cause you to exit to FDOS and generate the error message: **? Can't load shell.**

EXECUTE

This command allows your program to chain to other programs. (See the heading, **Chaining Programs Together,** for more information). To Execute (EXEC) a command file, you must set the shell to FDOS at the beginning of the file you are chaining to, and invoke a return to MenuBASIC at the end of the command file.

If you have a startup command file, STRTUP.CMD, that looks like this:

```
& Startup Command File in Progress
fup
ed0:/c-1
ed0:=mf0:/w
/x
mbasic r test1.bas
```

a program, TEST1.BAS, that looks like this:

```
.
.
.
if x = "c" then
  set shell
  exec "cop"
endif
```

and a command file, COP.CMD, that looks like this:

```
fup
wd0:mf0:/w
/x
mbasic r test1
```

This series of files allows the system to boot-up and begin running the main program, TEST1.BAS. Test1 then calls the command file COP.CMD, which calls MenuBASIC to run test1 again.

LINKING PREDEFINED MODULES

MenuBASIC allows you to link modules developed in BASIC, FORTRAN, or Assembly language to your program. Files to be linked must have an .OBJ extension in CBASIC or an .OBX extension in XBASIC - they must be compiled before they can be linked.

MenuBASIC linker options are comment lines embedded in your program that provide program information to the linking loader. MenuBASIC options may fill as many lines as needed, but must begin within the first three lines of the program. The linker options are

- Include - the I option, which links different modules together

- Find - the F option, which locates library files

Syntax

```
!# i {object module}[,object module]
!# f {library file}[,library file]
```

Parameters

- Include and find options may take up as many lines as necessary, but must not be broken up by other program lines.

- MenuBASIC provides file name extensions for the linking loader; do not type file names with extensions.

Example

A program linking two modules, intcon and comm, might look like this:

```
1000! System Test Program by D.S.
1010! Specify automatic integer conversion
1020!# /i
1030! Link modules
1040!# i intcon,comm
1050! Begin system initialization
    .
    .
    .
```

A program linking the program HEXMOD, and a library, TTBOX, and specifying no line numbers and extended syntax, might look like the program listed below. (TTBOX is the Touchscreen Toolbox, another Flare application package available from Fluke.)

```
!# /nl/e
!# i hexmod
!# f ttbox
!  Begin System Initialization
   .
   .
   .
```

ERRORS

Program Errors

Program errors fit into one of three classifications: syntax errors, run-time errors, or logical errors. For complete descriptions of these errors, see the CBASIC or XBASIC manual.

Syntax errors are language usage errors. They will be displayed on the screen when you leave an edit session, or you may use the Print Errors (PE) command to list these errors to a printer. The following program illustrates a syntax error:

```
!# /nl/e
!
prnt "hello, world"
end
```

Since "print" in line 3 is misspelled, MenuBASIC will display:

```
Total of 1 error in compilation.
Fatal errors found:
!Error 501 (illegal statement terminator) at line 3
```

Run-Time errors occur when you try to run your program. If you have not used the No Markers (/NM) option, MenuBASIC will display the errors and the line numbers on which they occurred. The following program illustrates a run-time error:

```
!# /nl/e
!
print "hello,world"
print A$
end
```

MenuBASIC will compile and link this program, but because you have not defined A$, when you try to run it, the following will be displayed:

```
hello,world

!Error 900 at line 4 in module $MAIN$
```

Logical errors are caused by errors in your program logic. MenuBASIC cannot troubleshoot this type of error. It's up to you to find these.

MenuBASIC Errors

Most errors in MenuBASIC are trapped, and appropriate messages are printed. Some errors escape through the filter, and may appear on the screen. For example, if MenuBASIC bombed out with the message:

```
Error 305 at line 562 in FIND_F
```

The file you specified probably does not exist, since an error 305 means File not found.

See the CBASIC or XBASIC language manual for a complete listing of the errors that may be reported by the run-time or compiler programs.

USING A PRINTER

Setting Up A Printer: The SP Command

Before a printer can be used, the Setup Serial Ports command (SP) must be invoked. SP puts you into the Set Utility Program and provides you with the prompt: SET>.

The purpose of the Set Utility program is to configure the 17XXA to enable it to communicate with virtually any other piece of equipment that uses the RS-232-C standard. The port parameters are set to default values when the operating system is loaded, and some applications will not require changing the defaults.

The default values are:

Device: KB1:
Baud Rate: 9600
Data Bits: 8
Parity: even
Stop Bits: 1
End Of Line: 10
End Of File: 26
Stall Input: disabled
Stall Output: enabled
Time Out: 0

SET is exited by typing: SET> exit. Refer to the 17XXA System Guide for more information on the SET program.

Selecting The Printer Port: The PP Command

Select the printer port that you wish to use with the PP command. KB1: is the default printer port.

Printing Your Program: The P command

To print your program, type P at the Enter Command? prompt. The program will print out on the device specified in the Select Printer Port (PP) command. For example, if KB1: is specified in the PP command, P will provide the message: Listing program on KB1:, and will print to the device on KB1:.

Printing Error Messages: The PE Command

To print error messages, type PE in the menu. The program will print out the program errors on the printer specified in the Select Printer Port (PP) command.

Printing Other Data

To print anything else, you must print it using the File Utility Program (FUP). To enter FUP, type F at the MenuBASIC prompt. For more information on FUP, see the 17XXA System Guide.

In FUP, if you wanted to print a directory listing for both MF0: and ED0: on the printer hooked up to KB1:, you would type the following:

```
FUP> kb1:=mf0:/q   Lists a directory of MF0: on the
                   printer
FUP> kb1:=ed0:/q   Lists a directory of ED0: on the
                   printer
```

To exit FUP, type:

```
FUP> /X
```

FILE UTILITIES: THE F COMMAND

Type F in the menu to enter the File Utility Program (FUP). FUP is a utility that gives the user full control over files on any of the devices. MenuBASIC allows you to enter FUP without having to save the current user program. FUP provides you with the prompt, FUP>. To exit FUP, type /X. See the 17XXA System Guide for more information on FUP.

AUTOMATING START-UP AND OTHER FUNCTIONS

System command files may be used to run and compile programs through MenuBASIC. Only two MenuBASIC commands can be used in command files. They are the CO (Compile) and R (Run) commands. For example, if a program, HEXTST, has already been compiled, a command file setting up the serial port to a baud rate of 4800, and then running HEXTST could look like this:

```
set
kb1:
br 4800
ex
mbasic r hextst
```

If you are not familiar with command files, refer to the 17XXA System Guide under the heading, **Automating System Functions.**

APPENDICES

Appendix A contains a quick reference guide for the system editor. Appendix B contains listings for the sample programs HEXMOD.BAS and HEXTST.BAS, which are used in the **Getting Started** section of this manual.

APPENDIX A

The Editor: A Quick Reference Guide

This appendix provides a quick reference guide for the system editor (EDIT.FD2) provided with MenuBASIC. Insertion mode, Command mode, and Global commands are covered. For more information on the editor, see the 17XXA System Guide.

The editor has two modes of operation, Insertion mode and Command mode. The Insertion mode is the default, and is primarily used for inserting text, although it does allow for some cursor and text manipulation. The Command mode is used for more powerful cursor and text manipulation, searching, and exiting. Many edit program commands result in a temporary return to the Insertion mode. When these commands are used, <ESC> will return you to Command mode.

INSERTION MODE

<backspace>, <linefeed>, ↑ ↓ ← →

COMMAND MODE

Use these commands in Insertion Mode by preceding with <ESC>

CURSOR POSITIONING

<backspace>, <linefeed>, ↑ ↓ ← →

[n]g	To line n of file
H	To top line of screen
[n]^F	Forward n screens
[n]^B	Backward n screens
[n]W	Forward n strings
[n]|	To column n

SEARCHING

COMMANDS

?{pattern}	Backward in buffer to pattern
!{pattern}	Forward to end of file for pattern
/{pattern}	Forward in buffer to pattern
n	Repeat previous search
N	Repeat search in opposite direction

SEARCHING (contd)

METACHARACTERS (Match:)

.	Single character wild card
^	Beginning of line
$	End of line
[]	Class (example: [aeiou])
[c-c]	Class range (example: [0-9])
!	Not in class (example: [!a-z]
#	Zero or more occurrences (example: .#)
\	Treats following character as literal (example:\$)

MARKER COMMANDS

m(x)	Insert marker x at cursor
`x	Move to marker x

TEXT MANIPULATION

y`(x)	Copy from cursor to marker x into yank buffer
[n]p	Copy yank buffer into text n times after cursor
Y	Clear yank buffer

DELETION

[n] <delete>, <del char>, <del line>

d`(x)	Delete to marker
[n]dl	Delete to column n
{n}dB	Delete m strings

GLOBAL COMMANDS

INSERTION MODE: <ESC>: {single character} <RETURN>

COMMAND MODE: :{global character} <RETURN>

:@	Toggle default mode
:p	Move forward 1 page
:m	Memory left
:v	Version number
:q	Exit and save
:q!	Exit and do not save
:s/ (old pattern)/(new pattern)/	Substitute old pattern for new pattern in a file

APPENDIX B

Program Listings

The program HEXTST and the subroutine HEXMOD are provided on the MenuBASIC disk. HEXTST converts hexadecimal numbers into decimal numbers. HEXTST calls the subroutine HEXMOD, which is responsible for the mathematical calculations. The subroutine HEXMOD must be linked to HEXTST for HEXTST to run correctly.

HEXTST.BAS

The main program HEXTST.BAS is included on the MenuBASIC disk to allow you to run the sample program in the **Getting Started** section of this manual. HEXTST.BAS calls the subroutine HEXMOD.BAS.

```
!#/nl/e
!# i hexmod
hexval%=0
loop
    print "Testing the sethex utility"
    print "Enter a hex string (max four characters) ";
    input string$
    sethex(string$,hexval%)
    if hexval%=0 then
            print "Illegal value entered:";string$
    else
            print "The integer was set to:";hexval%
    endif
    print
endloop
```

HEXMOD.BAS

The subroutine HEXMOD.BAS is included on the MenuBASIC disk. This subroutine performs some mathematical calculations and must be linked to the main program HEXTST.BAS in order for HEXTST to run correctly.

```
!# /nl/e
!# m
sub sethex(s$,v%)
    s$=ucase$(s$)
    v%=0
    l%=len(s$)
    if l%>4 then l%=0
    for i%=1 to l%
       d%=instr(1%,"0123456789ABCDEF",mid(s$,i%,1))-1
       if d%<0 then
          v%=0
          i%=l%
       else
          d%=lsh(d%,4*(l%-i%))
          v%=v% or d%
       endif
    next i%
    subret
subend
```

RESTRICTED RIGHTS