TOOLBOX LIBRARY
REFERENCE MANUAL

REVISION A
MAY 1995

1. GETTING STARTED

1.1 Introduction

This manual documents the use of the FORTRAN Toolbox Library with the Lahey Computer System's F77L, F77L-EM/32 language systems. This library is used in conjunction with the Spindrift Utility and Spindrift Windows Libraries. This manual assumes that the FORTRAN programmer has already obtained the Spindrift Libraries, and is familiar of their syntax and usage. For more information, refer to the *LAHEY SPINDRIFT UTILITY LIBRARY REFERENCE MANUAL* and *LAHEY SPINDRIFT WINDOWS LIBRARY REFERENCE MANUAL*. The Toolbox Library system requirements for compilation, linking and execution are identical to those of the Spindrift Utility and Spindrift Windows Libraries.

This documentation is divided into three chapters as follows:

Chapter 1 - GETTING STARTED

This chapter is about using the library routines, linking, typing and other conventions.

Chapter 2 - REFERENCE

This chapter explains all the SUBROUTINES and FUNCTIONS found in the Toolbox Library with examples for each one.

Chapter 3 - CREATING APPLICATIONS

This chapter proposes a general methodology for constructing menu-driven FORTRAN applications. It also illustrates the use of the Toolbox Library routines thru an example.

1.2 Installation

Use the following procedure to install the Toolbox Library.

1. Change to the subdirectory where the Spindrift Utility and Spindrift Windows libraries are (e.g. \F77L or \F77L3\LIB)

2. Copy the Toolbox Libraries in that directory

Installation is now complete

1.3 Linking Toolbox Library Subroutines

The Toolbox Library is provided in a single file that is compatible with the Lahey OPTLINK Linker, DOS Link (for F77L programs) and PharLap 386Link (for F77L-EM/32 programs).

NOTE:	The Toolbox Library needs to be linked with the Spindrift Utility and Spindrift Windows Library. If this is not done, the linker will fail and issue an “unresolved externals” error.

1.3.1 Using F77L’s Optlink Linker

Syntax:

```
optlink obj,[exe],[nul],toolbox+wind+util[+other libs];
```

Where:

“optlink” is the linker executable;

“obj” is the program object (compiled) code file

“exe” is the executable filename to create;

“nul” surpresses any map file generation ; and

“toolbox”, “wind”, “util” are the Toolbox, Spindrift Windows and Spindrift Utility Libraries.

1.3.2 Using DOS LinkWith F77L

Syntax:

```
link obj,[exe],[nul],toolbox+wind+util[+other libs];
```

Where:

"link" is the DOS Link executable;

"obj" is the program object (compiled) code file

"exe" is the executable filename to create;

"nul" surpresses any map file generation ; and

"toolbox", "wind", "util" are the Toolbox, Spindrift Windows and Spindrift Utility Libraries.

1.3.3 Using PharLap 386LINK With F77L-EM/32

Syntax:

```
386link obj -EXE exe -LIB toolbox wind util[ other libs]
```

Where:

"386link" is the PharLap 386Link executable;

"obj" is the program object (compiled) code file

"exe" is the executable filename to create;

"toolbox", "wind", "util" are the Toolbox, Spindrift Windows and Spindrift Utility Libraries for the F77L-EM/32 language system.

1.4 General Overview of Library Routines

1.4.1 File Realted

APPEND: Appends one file to another

BROWSE: Browses the contents of a file

CHKEXST: Checks the existance of a file

DELFILE: Deletes a DOS file

DIRSHOW: Displays the contents of a directory

EDITFILE: ASCII file editor

EXEHDR: Checks the compression status of an executable file

FEXIST: Checks for the existance of a file

FILE: Prompts the user for a file name

MORE: Displays the contents of a file

PUSHEND: Pushes the pointer of a file to its end

SELECTFILE: Selects a file from a file list

1.4.2 String Related

APPENDEXT: Edits DOS filename strings

CHRBLN: Finds position of last nonblank character in a string

INPUTCHR: Reads input from window

READCHAR: Prompts for input

SHFTRGHT: Performs a right - justification function on a string

1.4.3 Windows-Realted

MENU:	Creates user menus
MESSAGE:	Displays messages in a box
PAUSEW:	Pauses in a window
RESETBKG:	Creates background designs
SETITLE:	Creates titles
STRETCHW:	Resizes windows
YORN:	Dialog routine

1.4.4 Other

EXEC:	Spawns other DOS tasks from FORTRAN
FREEUNIT:	Obtains a free (unused) file or window unit number
SYSINFO:	Obtains system information

2. REFERENCE

2.1 APPEND Subroutine

This subroutine is used to append a text file to another.

Syntax:

```
CALL APPEND (INfile, OUTfile)
```

Where:

“INfile” is a CHARACTER variable with the file, the contents of which are appended to “OUTfile”.

“OUTfile” is a CHARACTER variable with the file to be appended.

Example:

```
. . .
FILENAME1 = 'TEST1.DAT'
FILENAME2 = 'TEST2.DAT'
CALL APPEND (FILENAME1, FILENAME2)
. . .
```

after execution of this code segment, the new version of 'TEST2.TXT' contains both the old 'TEST2.DAT' and 'TEST1.DAT', in the specified order.

Note:	Both files must already exist for APPEND to work properly. No check for file existance is done.

2.2 APPENDEXT Subroutine

This subroutine is used to edit the extension of a DOS file name string.

Syntax:

```
CALL APPENDEXT (Filename, Defext, Overwrite)
```

Where:

“Filename” is the CHARACTER variable with the file name

“Defext” is the CHARACTER variable with extension (including '.')

“Overwrite” is a LOGICAL value, .TRUE. overwrites any existing extension in "Filename" with "Defext", .FALSE. puts "Defext" only if there is no extension in "Filename".

Example:

```
. . .
FILENAME1 = 'TEST.DAT'
FILENAME2 = 'TEST.DAT'
CALL APPENDEXT (FILENAME1, '.TXT', .FALSE.)
CALL APPENDEXT (FILENAME2, '.TXT', .TRUE. )
. . .
```

after execution of this code segment, "FILENAME1" has the value 'TEST.DAT', while "FILENAME2" has the value 'TEST.TXT'.

2.3 BROWSE Subroutine

This subroutine is used as a text file browser.

Syntax:

```
CALL BROWSE (Fname, Iw)
```

Where:

“Fname” is the CHARACTER variable with the file browsed.

“IW” is the INTEGER variable containing the browse window number.

Example:

```
. . .
IBROWSE   = 1
FILENAME1 = 'TEST1.DAT'
CALL BROWSE (FILENAME1, IBROWSE)
. . .
```

after execution of this code segment, the file 'TEST1.TXT' is browsed in window 1 (which is previously defined and opened).

Note: The maximum size file size is 64356 bytes. If the file size is larger, only the first 64356 bytes are viewed.

See also: MORE, EDITFILE

2.4 CHKEXST Subroutine

This subroutine checks for a file that exists in the same DOS path of executing program.

Syntax:

```
CALL CHKEXST (Filen, Filenf, Ex)
```

Where:

“Fname” is the CHARACTER variable with the file which existance is to be checked for.

“Filenf” is the CHARACTER variable with the full path of file “Filen”

“Ex” is a LOGICAL variable which is .TRUE. if “Filen” exists and .FALSE. if “Filen” doesn't.

Example:

```
. . .
FILENAME1 = 'TEST1.DAT'
CALL CHKEXST (FILENAME1, FULLPATH, EXIST)
IF (EXIST) THEN
   WRITE (*,*) 'Full path of file: ',FULLPATH
ELSE
   WRITE (*,*) 'File does not exist'
ENDIF
. . .
```

after execution of this code segment, 'TEST1.TXT' is checked for existance in path of the program. If it does, then the full path of it is printed.

2.5 CHRBLN Subroutine

This subroutine returns the position of the last non-blank character in a string.

Syntax:

```
CALL CHRBLN (CHAR, IBLK)
```

Where:

“CHAR” is the CHARACTER variable containing the string examined.

“IBLK” is the INTEGER variable with the position of the last non-blank character in string “STRING”

Example:

```
. . .
STRING = 'ABCDE    '
CALL CHRBLN (STRING, IBLNK)
```

after execution of this code segment, variable “IBLNK” has the value 5.

Note:	If “CHAR” is blank, IBLK has a value of zero

2.6 DELFILE Subroutine

This subroutine deletes a DOS file.

Syntax:

```
CALL DELFILE (Filen)
```

Where:

“Filen” is the CHARACTER variable containing the file to be deleted.

Example:

```
. . .
FILENAME1 = 'TEST1.DAT'
CALL DELFILE (FILENAME1)
. . .
```

after execution of this code segment, 'TEST1.DAT' is deleted.

Note:	If “FILEN” does not exist, no action is taken place.

2.7 DIRSHOW Subroutine

This subroutine displays current directory at screen.

Syntax:

```
CALL DIRSHOW (IDIR, Filespec)
```

Where:

“IDIR” is a INTEGER variable with the directory window.

“Filespec” is the CHARACTER variable with the dos file mask to be shown.

Example:

```
. . .
IDIR     = 1
FILESPEC = '*.DAT'
CALL DIRSHOW (IDIR, FILESPEC)
. . .
```

after execution of this code segment, all files with a '.DAT' extention are displayed in window "IDIR".

Warning:	In this implementation, “IDIR” must be a window with 16 rows high and 45 columns wide.

2.8 EDITFILE Subroutine

This subroutine provides a simple ASCII file editor. Changes are made by using the edit keys. The editor is in Overwrite mode always. If the user presses the ESC key, editing is aborted and control is returned to the calling subroutine. If the user presses the function F2 key, the edit buffer is saved to disk and control is returned to the calling subroutine.

Syntax:

```
CALL EDITFILE (Fil, Iedit)
```

Where:

“Fil” is the CHARACTER variable with the file to be edited.

“Iedit” is the INTEGER variable with the editor window.

Example:

```
. . .
FILENAME1 = 'TEST1.DAT'
CALL EDITFILE (FILENAME1,IEDIT)
. . .
```

after execution of this code segment, 'TEST1.DAT' is edited in window IEDIT.

Warning: The “IEDIT” window must be 25 rows high and 80 columns wide, and MUST NOT have a WRAP attribute.

The maximum file size is 1000 lines and 78 columns wide. Any text out of these ranges is truncated when saved.

See Also: MORE, BROWSE

2.9 EXEC Subroutine

This subroutine executes an external program from FORTRAN. The program checks for:

- COMMAND.COM in the path specified by DOS environment variable COMSPEC
- Existance of the program file first in the current dir, then in the DOS path.

In case if one of these files are not found, an error message is displayed and control is returned back to the FORTRAN program.

Syntax:

```
CALL EXEC (PROG, COMLINE, Ierr)
```

Where:

“Prog” is the CHARACTER variable with the executable file name (with extension).

“Comline” is the CHARACTER variable with command line arguments.

“Ierr” is a INTEGER variable containing the error window.

Example:

```
. . .
PROG = 'MEM.EXE'
OPTS = '/D /P'
CALL EXEC (PROG, OPTS, 10)
. . .
```

after execution of this code segment, 'MEM.EXE' is checked for existance in current directory and DOS path. If found, a DOS shell command "MEM /D /P" is executed. If not, an error message is displayed in window 10.

2.10 EXEHDR Subroutine

This subroutine checks and sees if an EXE file is compressed and finds the compressor used.

Syntax:

```
CALL EXEHDR (EXEfile, Itype)
```

Where:

“EXEfile” is the CHARACTER variable with the file appended

“Itype” is an INTEGER flag:

“Itype” value	EXE file type
0	File not compressed
1	File compressed with PKLITE
2	File compressed with LZEXE

Example:

```
. . .
FILENAME1 = 'TEST1.EXE'
CALL EXEHDR (FILENAME1, Itype)
. . .
```

after execution of this code segment, 'TEST1.EXE' is tested for its compression status.

Warning: “EXEfile” must exist in the specified path, for the SUBROUTINE to work properly.

2.11 FEXIST Function

This LOGICAL function determines the existance of a file and optionally prints error message.

Syntax:

```
Logical Variable = FEXIST (Filenm, Ierror, Imes)
```

Where:

“Filenm” is the CHARACTER variable with the file appended

“Imes” is the INTEGER variable with the message window.

“Ierror” is an INTEGER flag for displaying a file does not exist message:

“Ierror” value	Message type
0	display message
1	surpress message

Example:

```
. . .
FILENAME1 = 'TEST1.EXE'
A = FEXIST (FILENAME1, 0, 10)
. . .
```

after execution of this code segment, 'TEST1.EXE' is tested for existance in current directory. If it exists, variable “A” has a /TRUE. value. If not, “A” has the value .FALSE. and an error message is displayed at window 10.

2.12 FILE Subroutine

This subroutine prompts the user for a file name, with a default extension. Optionally, the user can select from a file list (with a user-specified file mask).

Syntax:

```
CALL File (Filen, Defext, Filesp, FEXIST, IRdfn, Isel)
```

Where:

“Filen” is the CHARACTER variable with the file name specified by the user.

“Defext” is the CHARACTER variable with the default file extension.

“Filesp” is the CHARACTER variable with the file mask for the file list (when used).

“FEXIST” a LOGICAL flag, which is .TRUE. if a file was specified (and its name is in “Filename”), or .FALSE. if no file was specified.

“Irdfn” is the INTEGER variable with the file name prompt window.

“Isel” is an INTEGER flag for displaying a file list.

Example:

```
. . .
DEFEXT   = 'DAT'
FILESPEC = '*.XLS'
IRDFN    = 1
ISEL     = 2
CALL FILE (FILENAME,DEFEXT,FILESPEC,FEXIST,IRDFN,ISEL)
IF (FEXIST) THEN
   WRITE (*,*) 'File specified:',FILENAME
ELSE
   WRITE (*,*) 'No file specified.'
ENDIF
. . .
```

after execution of this code segment, a prompt is shown in window “IRDFN” asking for a file name, with a default extension of '.DAT'. If function key F5 is pressed, a file list with all '.XLS' of the current directory are shown. If no file is

selected or specified, flag “FEXIST” has a value of .FALSE., other wise “FEXIST” is .TRUE. and the file name is saved in variable “FILENAME”.

Warning:	In this implementation, “ISEL” must be a window with 16 rows high and 45 columns wide.
Note:	If no file is specified, ‘FILENAME’ is blank.
See Also:	SELECTFILE

2.13 FREEUNIT Subroutine

With this subroutine, the user get a free file or window unit (i.e. which is not connected to a file or a window).

Syntax:

```
CALL Freeunit (Iunit)
```

Where:

“Iunit” is the INTEGER variable containing the free unit number.

Example:

```
. . .
CALL FREEUNIT (IFREE)
. . .
```

after execution of this code segment, IFREE has the number of a free (unused) unit.

2.14 INPUTCHR Subroutine

This subroutine obtains input from a certain window. No intialization of the window takes place, so everything on screen is preserved.

Syntax:

```
CALL INPUTCHR (IWIN,STRING,ESC)
```

Where:

“IWIN” is the INTEGER variable with the input window unit.

“STRING” is the CHARACTER variable with the input string.

“ESC” is a LOGICAL flag ; which has the value .TRUE. if ESC key was pressed, and .FALSE. if ESC was not pressed.

Example:

```
. . .
CALL CLRB (IWIN)
WRITE (IWIN,*) 'Give string :'
CALL INPUTCHR (IWIN,STRING,ESC)
IF (.NOT.ESC) THEN
   WRITE (IWIN,*) 'String given [',CHARNB(STRING),']'
ELSE
   WRITE (IWIN,*) 'ESC key pressed.'
ENDIF
. . .
```

after execution of this code segment, the window "IWIN" is initialized and prompts the user for input. After that, the results are displayed.

See Also: READCHAR

2.15 MENU Subroutine

This subroutine displays a command menu in a window, with optional context-sensitive help.

Syntax:

```
CALL MENU (HED,Opt,Hlp,Iopt,Nopt,Ichoice,Show,Imenu)
```

Where:

“HED” is the CHARACTER variable with the menu window caption.

“Opt” is an array of CHARACTER strings with the menu options.

“Hlp” is an array of CHARACTER strings, one string for each menu option (context-sensitive).

“Iopt” is an array of INTEGER variables with the code of each menu option.

“Nopt” is an INTEGER variable containing the number of options of the menu.

“Ichoice” is the INTEGER variable containing the code of the selected option.

“Show” is a LOGICAL flag ; whin it is .TRUE., the context help is shown, and when it is .FALSE., the help is not shown.

“Ichoice” is the INTEGER variable containing the menu window unit number.

Example:

```
       . . .
C
C *** Option 1 ************************************************
C
       Opt (1) = ' Write message ’
       Iopt(1) = 1
       Hlp (1) = 'Displays a sample message'

C
C *** Option 999 **********************************************
C
       Opt (2) = ' Quit '
```

```
       Iopt(2) = 999
       Hlp (2) = 'Terminates Program and exits to DOS'
C
C *** Call  menu subroutine ********************************
C
10     CALL MENU ('Main Menu', Opt, Hlp, Iopt, 2, ICH, .TRUE.,6)
C
C *** Execute Options **************************************
C
       IF (Ichoice.EQ.1) THEN
          WRITE (*,*) 'Demo message'
       ELSE IF (Ichoice.EQ.999) THEN
          STOP 'Program terminated'
       ENDIF
C
C *** Redo menu loop ***************************************
C
       GOTO 10
       . . .
```

after execution of this code segment, a menu of two items is displayed in window 6. After selecting an item, the appropriate code is executed.

2.16 MESSAGE Subroutine

Displays a window with a message.

Syntax:

```
CALL Message (Mess, Ilines, Paus, Imes)
```

Where:

“Mess” is a CHARACTER variable (when Ilines=1) or Array (when Ilines>1) with the message to diplay.

“Ilines” is an INTEGER variable with the number of lines to display

“Paus” is a LOGICAL variable ; .TRUE. is to wait for the user to press the [Enter] key ; .FALSE. is to display the message and continue running.

“Imes” is a INTEGER variable, with the message window unit number.

Example:

```
. . .
MESSG = 'This is an example message'
CALL APPEND (MESSG, 1, .TRUE., 10)
. . .
```

after execution of this code segment, the message 'This is an example message' is displayed at window 10 and waits for the user to press the [Enter] key.

Note:	When “PAUS” is .TRUE., a ‘[OK]’ button is shown at the bottom line of the window. This implies that the user must provide enough space in thw window (an additional 2 rows) for the message to be displayed correctly.

2.17 MORE Subroutine

Serves as a file displayer (simulates UNIX “more”).

Syntax:

```
CALL MORE (MOREn, IMORE, Imes)
```

Where:

“MOREn” is the CHARACTER variable with the file shown.

“IMORE” is the INTEGER variable with the MORE window (must be full screen, i.e. 23 rows x 78 cols work space).

“Imes” is the INTEGER variable with the MESSAGE window unit.

Example:

```
. . .
FILENAME1 = 'TEST1.DAT'
CALL MORE (FILENAME1, IMORE, IMES)
. . .
```

after execution of this code segment, ‘TEST1.TXT’ is displayed at window “IMORE” one screenful at a time. If the file does not exist, a “File not found” error message is diplayed at window “Imes”.

Note: Window “IMORE” MUST NOT have the WRAP attribute and must be 25 rows high.

See Also: BROWSE, EDITFILE

2.18 PAUSEW Subroutine

Pauses and displays a “Press any key to continue...” message at a certain window.

Syntax:

```
CALL PAUSEW (IPAUSE)
```

Where:

“IPAUSE” is a INTEGER variable, the window to pause in.

Example:

```
. . .
CALL PAUSEW (10)
. . .
```

after execution of this code segment, the program pauses and a “Press any key to continue...” message is displayed at window 10.

2.19 PUSHEND Subroutine

Positions the pointer of a sequential file at its end. Simulates the ACCESS='APPEND' clause of a F77L OPEN statment, but with the use of Standard Fortran 77 commands.

Syntax:

```
CALL PUSHEND (IUNIT)
```

Where:

"IUNIT" is a INTEGER variable, the file unit number.

Example:

```
. . .
CALL PUSHEND (10)
. . .
```

after execution of this code segment, the pointer of unit 10 is pushed to its end.

2.20 READCHAR Subroutine

Prompts the user for input.

Syntax:

```
CALL READCHAR (Mess, Buf, Iwin)
```

Where:

“Mess” is a CHARACTER variable, the message to display before the input prompt.

“Buf” is a CHARACTER variable, containing the user input.

“Iwin” is an INTEGER variable, with the window unit of the prompt.

Example:

```
. . .
CALL PUSHEND ('Give the value of ALPHA: ',Buf, Iwin)
READ (Buf,*) ALPHA
. . .
```

after execution of this code segment, the user is prompted for the value of the REAL variable “ALPHA” in window “Iwin”. The input is saved in CHARACTER variable “Buf”. Then “ALPHA” is assigned from a READ statment (“Buf” is used as a FORTRAN internal file).

See Also: INPUTCHR

2.21 RESETBKG Subroutine

Paints a design suitable for a background.

Syntax:

```
CALL RESETBKG (Ibckgrnd, Iwin)
```

Where:

“Ibckgrnd” is an INTEGER variable, with the window used as a background.

“Iwin” is an INTEGER variable, when >0 it initializes window "Iwin", when 0, no initialization of another window takes place.

Example:

```
. . .
CALL RESETBKG (Ibckgrnd,Iwin)
. . .
```

after execution of this code segment, the background window “IBckgrnd” is initialized. Then Window “Iwin” is shown.

2.22 SELECTFILE Subroutine

With this subroutine, the user can select a specific file from a file list (with a user-specified file mask).

Syntax:

```
CALL Selectfile (INFILE, FILESPEC, ISEL)
```

Where:

“INFILE” is the CHARACTER variable with the file name specified by the user.

“Filespec” is the CHARACTER variable with the file mask for the file list (when used).

“Isel” is the INTEGER variable with the message window.

Example:

```
. . .
FILESPEC = '*.XLS'
ISEL     = 1
CALL SELECTFILE (FILENAME,FILESPEC,ISEL)
. . .
```

after execution of this code segment, a file list with all '.XLS' of the current directory are shown. The file selected is saved in variable “FILENAME”.

Warning: In this implementation, “ISEL” must be a window with 16 rows high and 45 columns wide.

Note: If no file is selected, ‘FILENAME’ is blank.

See Also: FILE

2.23 SETITLE Subroutine

Assigns windows with good looking titles. The caption is centered, and the title bar is in reverse video attribute.

Syntax:

```
CALL SETITLE (Iwin, Wintitle)
```

Where:

“Iwin” is an INTEGER variable, the window which the title is placed.

“WinTitle” is a CHARACTER string containing the title bar caption.

Example:

```
. . .
CALL SETITLE (Iwin,'Main Window')
. . .
```

after execution of this code segment, window "Iwin" is assigned the title ‘Main Window’.

2.24 SHFTRGHT Subroutine

Performs a right-justification function on a string.

Syntax:

```
CALL SHFTRGHT (STRING)
```

Where:

“STRING” is a CHARACTER string containing the string for justification.

Example:

```
. . .
STRING = ‘AAAA    ‘
CALL SHFTRGHT (STRING)
. . .
```

after execution of this code segment, the window “STRING” has the value ‘ AAAA’.

2.25 STRETCHW Subroutine

Stretches (resizes) an existing window.

Syntax:

```
CALL STRETCHW (Iwin, Nrow, Ncol, Iflag, Tit)
```

Where:

“Iwin” is an INTEGER variable, the window which to be streched.

“NRow” is an INTEGER variable, the number of rows to stretch; positive Nrow pulls down, negative Nrow pushes up.

“NCol” is an INTEGER variable, the number of columns to strech; positive Ncol pulls right,negative Nrow pushes left.

“Iflag” is an INTEGER variable; when=0, top left corner is moved; when=1, bottom right corner is moved.

“Tit” is a CHARACTER string containing the window's new title.

Example:

```
. . .
CALL STRECHW (Iwin,10,10,1,'New Window')
. . .
```

after execution of this code segment, the window “Iwin” has its bottom right corner moved 10 rows down and 10 columns right. The window's title becomes ‘New Title’.

2.26 SYSINFO Subroutine

Returns system information.

Syntax:

```
CALL SYSINFO (MAXCOL)
```

Where:

"MAXCOL" is an INTEGER variable with the maximum number of colors available in graphics mode.

Example:

```
. . .
CALL SYSINFO (MAXCOL)
. . .
```

after execution of this code segment, the variable "MAXCOL" contains the number of colors (starting from 0) available in graphics mode.

NOTE:	This routine must be called from graphics mode to return the most appropriate values. If it is called from text mode, the value returned is an estimate.

2.27 YORN Subroutine

Dialog routine; the user is provided with a window of options.

Syntax:

```
CALL YORN (Msg, Nmsg, Opt, Nopt, Ians, Iyorn)
```

Where:

“Msg” is a CHARACTER variable (when Nmsg=1) or Array (when Nmsg >1) with the message to display on the dialog.

“Nmsg” is an INTEGER variable with the number of lines in Msg.

“Opt” is a CHARACTER variable array with the options on the dialog.

“Nopt” is a INTEGER variable containing the number of options.

“Ians” is a INTEGER variable returns the option selected.

“Iyorn” is a INTEGER variable with the dialog window number.

Example:

```
. . .
Msg    = ‘Do you want to terminate program?’
Opt(1) = ‘[ Yes ]’
Opt(2) = ‘[ No  ]’
CALL YORN (Msg, 1, Opt, 2, Ians, Iyorn)
IF (Ians.EQ.1) THEN
   STOP ‘Program terminated’
ELSE
   RETURN
ENDIF
. . .
```

after execution of this code segment, a dialog with two options is presented.

3. CREATING APPLICATIONS

3.1 INTRODUCTION

This chapter presents a general methodology for developing menu-driven FORTRAN applications.

3.2 METHODOLOGY

In writing FORTRAN menu-driven applications, the user is advised to place all COMMON block and I/O (file & Window unit) PARAMETERs in a separate file. This file is read in by the compiler by inserting INCLUDE statments in the SUBROUTINES which need this information.

When using the Toolbox Library routines, all windows must be previously opened by OPENWIND statements, one for each window used. A title may then be assigned to each one of them.The user is advised to include an initialization SUBROUTINE, that opens all windows and assigns them a title.

The program below illustrates the required procedure and calling order for a typical menu-driven program.

```
      PROGRAM MENU
      CHARACTER*30  OPT(3), HLP(3)
      INTEGER IOPT(3), SCRN(1024)
      INCLUDE 'UNITS.INC'
C
C *** GRAB ENTRY SCREEN *********************************
C
      CALL GETSCN (SCRN)
C
C *** INITIALIZE ALL WINDOWS ****************************
C
      CALL WININIT
C
C *** MENU SYSTEM AND OTHER PROGRAM ROUTINES *******
C
10    CALL MENU (......)
      . . .
C
```

```
C *** REDO MENU LOOP ************************************
C
       GOTO 10
C
C *** TERMINATION PROCEDURE ***************************
C
100    CALL DISPSCN (SCRN)
       STOP 'Normal Termination.'
       END

       SUBROUTINE WININIT
       INCLUDE 'UNITS.INC'
       CALL OPENWIND (IMENU, ....)
       CALL OPENWIND (IWIND2, ....)
       CALL SETITLE  ('SAMPLE TITLE', IWIND2)
       RETURN
       END
```

File UNITS.INC:

```
       PARAMETER (IMENU=1001, IWIND2=1002)
```

3.3 EXAMPLE APPLICATION

The following program demontrates how to construct a FORTRAN menu-driven application with a background wallpaper and multiple menu-levels. This program follows the general methodology described above. It also demonmstrates the capabilities of the Toolbox Library.

Application Source Code (APPLIC.FOR)

```
CC************************************************************
CC
CC  Program unit   : PROGRAM MAIN
CC  Purpose        : THIS IS THE MAIN ROUTINE OF THE PROGRAM.
CC
CC************************************************************
CC
      PROGRAM MAIN
CC
CC************************************************************
      INCLUDE 'APPLIC.INC'
      INTEGER   Iopt(6),ENTRYSCRN(1024)
      CHARACTER Opt(6)*23,Hlp(6)*49
      CHARACTER Yrn(2)*7
      CHARACTER Buf*64
      LOGICAL   FEX
C
```

```
C *** INITIALIZATIONS ****************************************
C
      CALL CAPTSCN (ENTRYSCRN)
      CALL WININIT
      CALL CUROFF
C
C *** SETUP MENU SYSTEM **************************************
C
      Opt (1) = ' Another Menu '
      Iopt(1) = 123
      Hlp (1) = 'Uses MENU '
C
      Opt (2) = ' Edit [SAMPLE.TXT]'
      Iopt(2) = 234
      Hlp (2) = 'Uses EDITFILE '
C
      Opt (3) = ' Browse Text file '
      Iopt(3) = 345
      Hlp (3) = 'Uses SELECTFILE, BROWSE '
C
      Opt (4) = ' View Sample Text file '
      Iopt(4) = 456
      Hlp (4) = 'Uses FILE, MORE '
C
      Opt (5) = ' About '
      Iopt(5) = 567
      Hlp (5) = 'Information about this demo '
C
      Opt (6) = ' Exit  '
      Iopt(6) = 678
      Hlp (6) = 'Displays the use of YORN Subroutine '
C
C *** SHOW ENTRY *********************************************
C
      CALL ABOUT (.TRUE.,ENTRYSCRN)
C
C *** CALL MENU SYSTEM ***************************************
C
10    CALL RESETBKG (IBCKGRND, 0)
      CALL MENU     ('Main Menu', Opt, Hlp, Iopt, 6, ICH,
     &               .TRUE., IMENU1)
C
C *** FILE OPTION ********************************************
C
      IF (ICH .EQ. 123) THEN
         CALL SUBMENU

C
C *** EDIT TEXT FILE OPTION **********************************
C
      ELSE IF (ICH .EQ. 234) THEN
         CALL EDITFILE ('APPLIC.INC',IEDITOR)
C
C *** BROWSE TEXT FILE OPTION ********************************
```

```
C
      ELSE IF (ICH .EQ. 345) THEN
         CALL SELECTFILE (BUF,'*.*',ISEL)
         IF (CHARNB(BUF).NE.' ') CALL BROWSE (BUF, IBRW)
C
C *** VIEW TEXT FILE OPTION ***********************************
C
      ELSE IF (ICH .EQ. 456) THEN
         BUF=' '
         CALL FILE (BUF, '.TXT', '*.*', FEX, IRDF1, ISEL)
         IF (CHARNB(BUF).NE.' ') CALL MORE (BUF,IMORE,IMES)
C
C *** ABOUT OPTION ********************************************
C
      ELSE IF (ICH .EQ. 567) THEN
         CALL ABOUT (.FALSE.,ENTRYSCRN)
C
C *** TERMINATE PROGRAM & EXIT TO DOS *************************
C
      ELSE IF (ICH .EQ. 678) THEN
         Yrn(1)='[ Yes ]'
         Yrn(2)='[ No  ]'
         CALL YORN ('Exit this demo?',1,Yrn,2,IANS,IYORN)
         IF (IANS.EQ.1) THEN
            CALL DISPSCN (ENTRYSCRN)
            CALL CURON
            STOP
         ENDIF
      ENDIF
C
C *** CALL MENU SYSTEM ***************************************
C
      GOTO 10
C
C *** END OF MAIN PROGRAM ************************************
C
      END




CC************************************************************
CC
CC  Program unit   : SUBROUTINE SUBMENU
CC  Purpose        : Displays and executes a submenu
CC
CC************************************************************
CC
      SUBROUTINE SUBMENU
CC
CC************************************************************
      INCLUDE 'APPLIC.INC'
```

```
      CHARACTER Opt(4)*17,Hlp(4)*47,BUF*64
      INTEGER   Iopt(4),SCRN(1024)
C
C *** DISPLAY AND RUN MENU ***********************************
C
      Opt (1)=' Input a number '
      Hlp (1)='Displays the use of INPUTCHR Subroutine'
      Iopt(1)=1
C
      Opt (2)=' Dummy Option 1'
      Hlp (2)='Does nothing '
      Iopt(2)=2
C
      Opt (3)=' Dummy Option 2 '
      Hlp (3)='Does nothing '
      Iopt(3)=2
C
      Opt (4)=' Main Menu '
      Hlp (4)='Return to main menu'
      Iopt(4)=3
C
      CALL CAPTSCN (SCRN)
C
C *** CALL MENU SYSTEM ***************************************
C
100   CALL DISPSCN (SCRN)
      CALL MENU ('SubMenu',Opt,Hlp,Iopt,4,ICH,.TRUE.,IMENU2)
C
C *** READ NUMBER OPTION *************************************
C
      IF (ICH.EQ.1) THEN
         CALL CURON
         BUF=' '
         CALL READCHAR ('Enter number: ',BUF,INDAT)
         READ (BUF,*,ERR=10) A
         CALL CUROFF



C
C *** DUMMY OPTION *******************************************
C
      ELSE IF (ICH.EQ.2) THEN
         CALL MESSAGE('This is a dummy option',1,.TRUE.,IMES)
C
C *** RETURN OPTION *******************************************
C
      ELSE IF (ICH.EQ.3) THEN
         RETURN
      ENDIF
      GOTO 100
C
C *** ERROR HANDLER *******************************************
C
```

```
10    CALL MESSAGE('This is a not a valid number',1,.TRUE.,IMES)
      GOTO 100
      END

CC************************************************************
CC
CC  Program unit   : SUBROUTINE ABOUT
CC  Purpose        : ABOUT DIALOG.
CC
CC************************************************************
CC
      SUBROUTINE ABOUT (SWITCH,ENTRYSCRN)
CC
CC************************************************************
      INCLUDE 'APPLIC.INC'
      INTEGER*2 KEY1
      INTEGER*4 ENTRYSCRN(1024)
      LOGICAL   Switch
      CHARACTER CH1*1
      CALL RESETBKG (IBCKGRND, IW)
      CALL BLDB(Iw, 1, 1,'          SAMPLE APPLICATION  ')
      CALL BLDB(Iw, 3, 1,'       for MENU-driven FORTRAN')
      CALL BLDB(Iw, 5, 1,'             APPLICATIONS     ')
      IF (SWITCH) THEN
        CALL LOCATE (22,3)
        CALL PRINTS (' Press ESC to exit, another key to
     &                continue')
      ENDIF
      CALL GETCHT(CH1, KEY1, 3, *50)
      IF(KEY1.EQ.16 .AND. Switch) THEN
         CALL DISPSCN (ENTRYSCRN)
         STOP
      ENDIF
50    RETURN
      END
CC************************************************************
CC
CC  Program unit   : SUBROUTINE WININIT
CC  Purpose        : INITIALIZES ALL PROGRAM WINDOWS
CC
CC************************************************************
CC
      SUBROUTINE WININIT
CC
CC************************************************************
      INCLUDE 'APPLIC.INC'
      INCLUDE 'WINDOWS.INC'
C
C ABOUT window
C
      CALL OPENWIND(IW,4,17,12,60,SINGLBD,BRIGHT+DBLUBKG)
      CALL SETITLE (IW, 'About sample application')
C
C More window
```

```
C
      CALL OPENWIND(IMORE,0,0,24,79,SINGLBD,BRIGHT+DBLUBKG)
      CALL SETITLE (IMORE, 'MORE subroutine')
C
C Message window
C
      CALL OPENWIND(IMES,6,9,13,70,SINGLBD+WRAP+SCROLL,
     &                             dbluBKG+bright)
      CALL SETITLE (IMES, 'Message')
C
C Input data window
C
      CALL OPENWIND(INDAT,8,12,13,67,SINGLBD+SCROLL,
     &                                    bright+dbluBKG)
      CALL SETITLE (INDAT, 'Request user for input')
C
C Yorn window
C
      CALL OPENWIND(IYORN,6,12,13,67,SINGLBD+SCROLL,
     &                               BRIGHT+DBLUBKG)
      CALL SETITLE (IYORN, 'Selection')
C
C Background window
C
      CALL OPENWIND(IBCKGRND,0,0,24,79,SINGLBD+SCROLL,
     &                                BRIGHT+DBLUBKG)
      CALL SETITLE (IBCKGRND,
     &             'FORTRAN Toolbox Library Sample Application')
C
C Menu windows
C
      CALL OPENWIND(IMENU1,4,22,14,48,SINGLBD,DBLUBKG+BRIGHT)
      CALL OPENWIND(IMENU2,9,51,17,71,SINGLBD+SCROLL,
     &                                BRIGHT+DBLUBKG)
C
C Read File name window
C
      CALL OPENWIND(IRDF1,6,12,10,67,SINGLBD+SCROLL,
     &                               BRIGHT+DBLUBKG)
      CALL SETITLE (IRDF1, 'ESC aborts, F5 lists directory')
C
C Directory window
C
      CALL OPENWIND(ISEL,6,18,21,62,SINGLBD+SCROLL,
     &                              BRIGHT+DBLUBKG)
      CALL SETITLE (ISEL,'Select file')
C
C Browse window
C
      CALL OPENWIND(IBRW,6,18,21,62,SINGLBD+WRAP+SCROLL,
     &                              BRIGHT+DBLUBKG)
      CALL SETITLE (IBRW,'EDIT keys move, ESC ends')
C
C Editor window
```

```
C
      CALL OPENWIND(IEDITOR,0,0,24,79,SINGLBD+SCROLL,
     &                               BRIGHT+DBLUBKG)
      CALL SETITLE (IEDITOR,
     &     '[SAMPLE.TXT]  **  ESC aborts, F2 exits with save')
C
      RETURN
      END
```

Application Include File (APPLIC.INC)

```
CC************************************************************
CC
CC  APPLIC.INC
CC  THIS IS THE INCLUDE FILE OF APPLICATION DEMO. IT CONTAINS
CC  THE WINDOW UNIT NUMBERS
CC
CC************************************************************
CC
      PARAMETER(IW=1001,IMORE=1002,IMES=1003,INDAT =1004,
     &         IYORN=1005,IBCKGRND=1006,IMENU1=1007,IMENU2=1008,
     &         IRDF1=1009,ISEL=1010,IBRW=1011,IEDITOR=1012)
```

Windows Attribute INCLUDE file (WINDOWS.INC)

```
      INTEGER BLKBKG,BLKCHR,DBLUBKG,DBLUCHR,GREENBKG,GREENCHR,
     &       LBLUBKG,LBLUCHR, REDBKG,  REDCHR,  LAVBKG, LAVCHR,
     &       YELLOBKG,YELLOCHR,WHITEBKG,WHITECHR,FLASH,UNDRLINE,
     &       NORMAL, BRITE,   BRIGHT,   INVERSE,  SCROLL,  WRAP,
     &       DBLBDR, SINGLBD, UNITZERO, DEFAULT
C
C *** CHARACTER COLOR DEFINITIONS ****************************
C
      PARAMETER (BLKCHR =0, DBLUCHR= 1, GREENCHR= 2,LBLUCHR = 3,
     &           REDCHR =4, LAVCHR = 5, YELLOCHR= 6,WHITECHR=7)
C
C *** BACKGROUND COLOR DEFINITIONS ***************************
C
      PARAMETER(BLKBKG=0, DBLUBKG =16, GREENBKG=32, LBLUBKG =48,
     &          REDBKG=64,LAVBKG =80, YELLOBKG=96, WHITEBKG=112)
C
C *** TEXT ATTRIBUTE DEFINITIONS *****************************
C
      PARAMETER (UNDRLINE=1,   NORMAL  =7, BRITE = 8, BRIGHT=15,
     &           INVERSE  =112, FLASH  =128)
C
C *** WINDOWN TYPE DEFINITIONS *******************************
C
      PARAMETER (NONE    =0, DEFAULT  = 0, SINGLBD= 1, DBLBDR=3,
     &           WRAP    =4, SCROLL   = 8, LIST   = 16)
      PARAMETER (UNITZERO=-32762)
```

```
C
C *** TEXT FORMATTING & WINDOW TITLE DEFINITIONS **************
C
      PARAMETER (TOPw   =0, BOTTOMw  = 1, LEFTw  = 2, RIGHTw=3,
     &           CENTERw=0, LJUSTw   = 1, RJUSTw = 2, TJUSTw=1,
     &           BJUSTw =2)
C
C *** END OF INCLUDE FILE 'WINDOWS.INC' ***********************
C
```

3.4 RUNNING THE SAMPLE APPLICATION

The following figures show various screens after running the sample application. All windows, menus, etc. are displayed against a background.

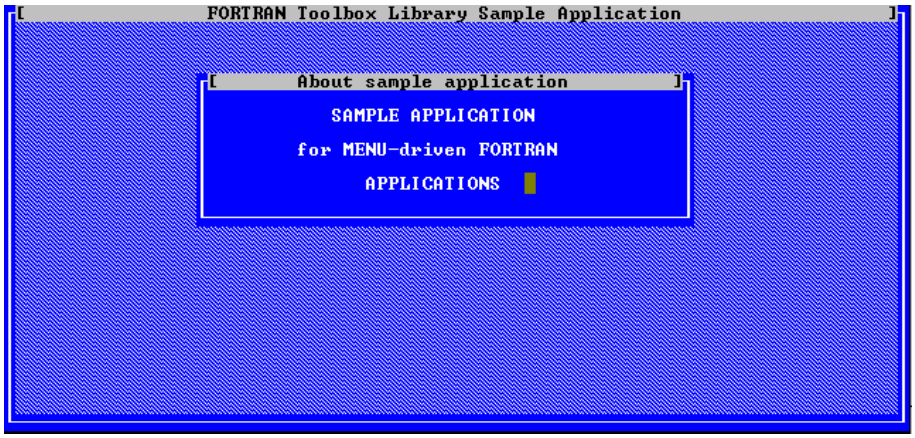

Figure 1: Entry screen (showing ABOUT dialog)

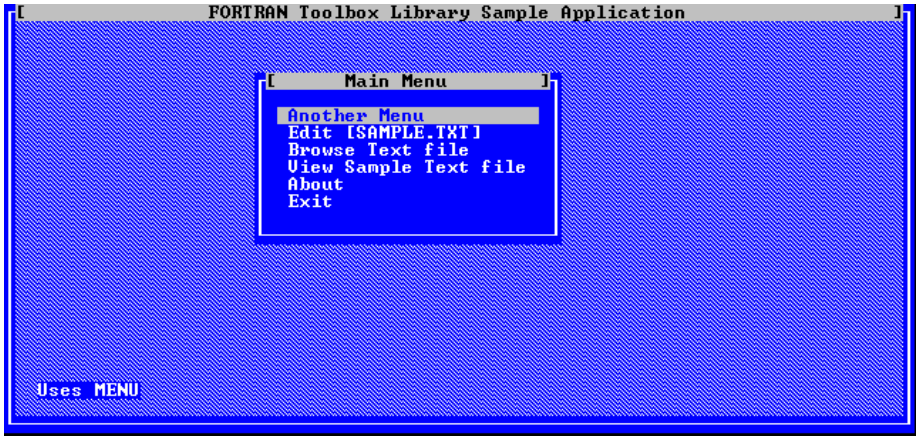

Figure 2: Main Menu of program (help is displayed on the left bottom corner)

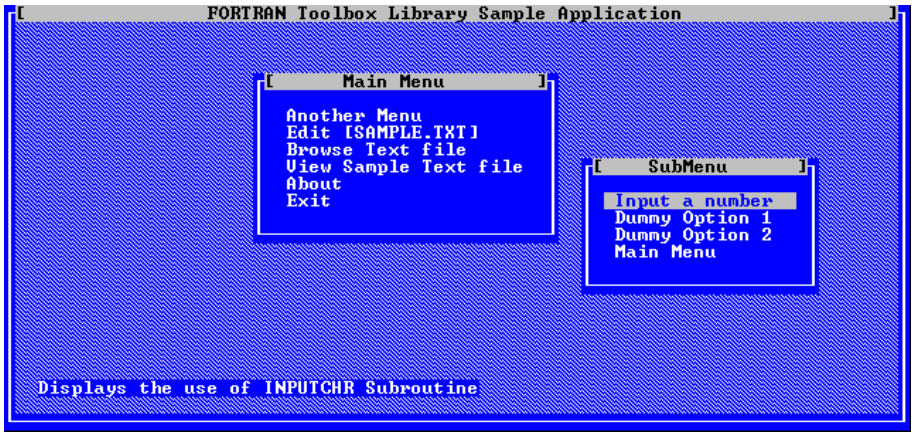

Figure 3: Submenu of Sample Application

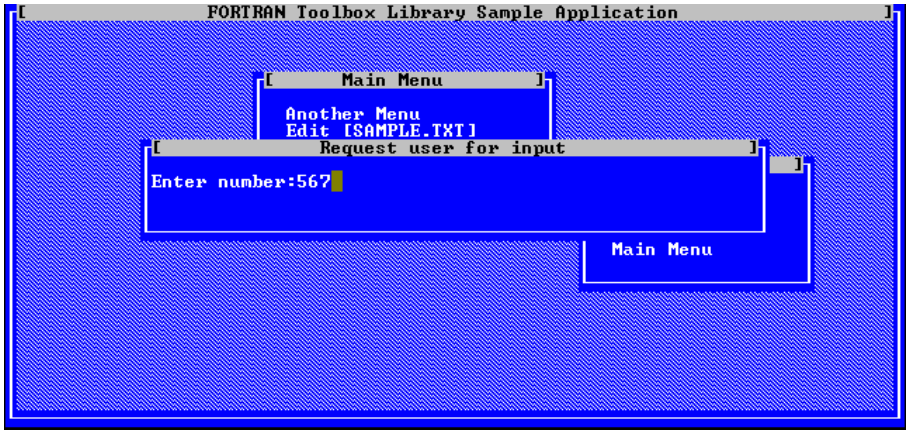

Figure 4: Reading input from a window.

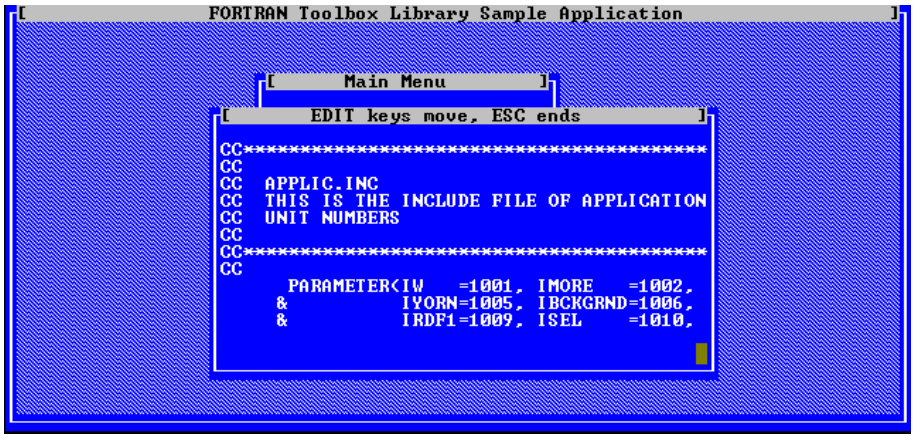

Figure 5: Browsing a text file

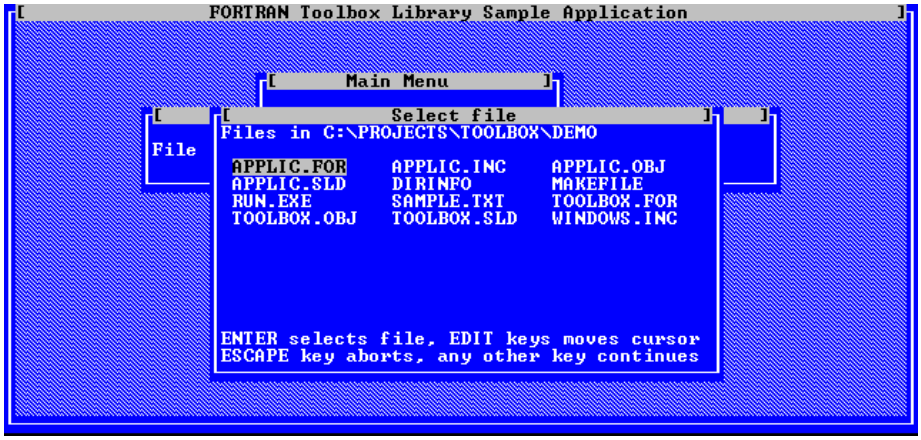

Figure 6: Selecting a file name