

ProvideX™

Utilities and Subprograms

Version4.x
September 1997

This manual contains information on the ProvideX utilities and subprograms. Chapter 1 of this manual is targeted at all ProvideX users while the rest of this manual is primarily targeted at the application programmer and analyst.

The complete set of ProvideX manuals consists of:

- ProvideX™ Installation Guide
- ProvideX™ Language Reference Manual
- ProvideX™ User's Guide
- ProvideX™ Utilities and Subprograms
- ProvideX™ NOMADS and Data Dictionary
- ProvideX™ WindX
- ProvideX™ ODBC

Copyright © 1986, 1990, 1994, 1995, 1996, 1997 by Sybex Ltd. (Ontario, Canada). All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopied, recorded or other, without prior written consent of Sybex Ltd.

The software described in this manual is furnished to the user under a license for a specified number of users and may be copied (with the inclusion of all copyright notices) only in accordance with the terms of such a license.

Sybex Ltd. assumes no responsibility for any errors or omissions that may appear in this document.

Sybex Ltd.

#204-8920 Woodbine Ave
Markham, Ontario
Canada, L3R 9W9

Voice: (905) 470-1025
Fax: (905) 470-9349
Compuserve: 74777,3445

ProvideX™ is a trademark of Sybex Ltd., Ontario, Canada
Sybex® is a Canadian registered trademark of Sybex Ltd., Ontario Canada
UNIX is a registered trademark of Bell Laboratories
MS-DOS and XENIX are registered trademarks of Microsoft Inc.
SCO is a registered trademark of The Santa Cruz Operation.
BB^X and BASIS are registered trademarks of BASIS International Ltd.

Other brand or product names are the trademarks or registered trademarks of their respective holders.

Contents

Chapter 1

<i>Utility Programs</i>	1
Screen Interface	1
Utility Subsystem main menu (**)	2
Built-in Calculator (*C)	2
Keyed file Trace (*CHKKEY)	4
Full Screen Program editor (*E)	4
Display of Open Files (*F)	9
Display File information (*FI)	9
Generic File Maintenance (*FM)	10
Syntax Table edit Utility (*LEXEDIT)	11
Message file maintenance (*MSGUPD)	12
Phone Directory (*P)	13
Main Utility Menu (*U)	13
System configuration sub-menu (*UC)	14
Keyboard Configuration (*UCK)	15
Configure Link files (*UCL)	16
Configure system parameters (*UCP)	18
Directory Utilities Sub-menu (*UD)	19
Directory Delete (*UDD)	19
Change to a different Directory (*UDG)	20
Make a new sub-directory (*UDM)	21
Directory Print (*UDP)	21
Change Directory name (*UDR)	22
Directory View (*UDV)	23
File Maintenance sub-menu (*UF)	24
File Administration Sub-menu (*UFA)	24
Keyed file integrity check (*UFAC)	25
Change file record count limit (*UFAM)	27
Recover Keyed file (*UFAR)	28
File Copy (*UFC)	29
File Delete (*UFD)	30
File Data Erase (*UFE)	31
Display file Information (*UFI)	31
Make a data file (*UFM)	32
File print (*UFP)	34
Change File name (*UFR)	35
File Update (*UFU)	36
File View (*UFV)	38
General Utility Sub-menu (*UG)	39
Mortgage Calculation program (*UGM)	39
Simple Spreadsheet Export (*UGS)	40

Program File sub-menu (*UP)	42
Program Bulk Search & Replace (*UPB)	42
Program compare (*UPC)	44
Program file Delete (*UPD)	45
Program file list (*UPL)	45
Program file Create (*UPM)	46
Change Program file name (*UPR)	47
Program Security (*UPS)	47

Chapter 2

<i>Subprograms</i>	49
Date Validation subprogram (*DATE)	49
Generate File List (*FL.LST)	50
Compare File Name to Mask (*FL.MTH)	50
Return name of a work file (*FL.NME)	51
Generic Option Selector (*OPTSEL)	51
Convert program (*PG.CNV)	52
Popup Selection Box (*POPSEL)	53
Abort print file (*PR.ABT)	53
Close print file (*PR.CLS)	53
Get print device (*PR.GET)	54
Open print file (*PR.OPN)	54
Select print file (*PR.SEL)	54
Restore current window (*SCR.RST)	55
Save window image (*SCR.SVE)	55
Item selection utility (*SELECT)	55
Input Validation routine (*VLDATE)	56

Chapter 3

<i>System Programs/Files</i>	57
System activation information (ACTIVATE.PVX)	57
Hot-Key intercept program (*CONTROL)	58
Device Drivers (*DEV)	58
Generic Error Handler (*ERROR)	59
On-line help display (*HELP)	59
System utility help information (*HELP.xx)	60
On-line Program help display (*HELP.PRG)	60
On-line Program help password (*HELP.PWD)	60
Keyboard definition (*KYBRD.CFG/*KYBRD.STD)	61
Syntax tables (*LEXTBL.xx/*LEXDEF.xx)	61
Message Library (*MLFILE.xx)	62
Parameter definitions (*PRMDEF.xx)	62
On-line Query processor (*QUERY)	63
Query maintenance program (*QUERY.DEF)	63
SYSTEM START_UP (*START.UP)	64
Load Keyboard Definitions (*TTY)	64

Chapter 4

<i>Windows Development Kit</i>	65
Overview of Subprograms and Objects	65
Push buttons (**BUTTON)	68
Check Boxes (**CHKBOX)	69
Check List (**CHKLST)	70
Combo boxes (**COMBOX)	71
Control buttons (**CTLBTN)	72
Drop boxes (**DRPBOX)	74
Error Box (**ERROR.BOX)	75
Horizontal movement processor (**HMOVE.CHK)	75
Horizontal Scroll Bar (**HSCRBR)	76
Input Box (**INPBOX)	77
List boxes (**LSTBOX)	78
Pull-down menus (**MENU)	79
Open File (**OPEN.FLE)	81
Option Box (**OPTION.BOX)	83
Prompt Box (**PROMPT.BOX)	83
Variable List boxes (**VARBOX)	84
Vertical movement processor (**VMOVE.CHK)	85
Vertical scroll Bar (**VSCRBR)	85
Warning Box (**WARN.BOX)	87
Window Horizontal scroll Bar (**WHSCRL)	87
Window Vertical scroll Bar (**WVSCRL)	88

Chapter 1

Utility Programs

This section of the manual describes the utilities that are included with the ProvideX system. These utilities provide the ability to create and maintain files, programs, and directories. They also provide built-in functions such as a calculator and a telephone directory.

All system utilities start with an asterisk (*) and are maintained in the ProvideX 'Library'. By default, this library is defined as the sub-directory **'lib'** in the directory containing the PVX executive. If desired, the path to the ProvideX utilities may be altered by setting the environment variable PVXLIB to the name of directory where the utilities are kept.

All references to system utilities, and all other references to any file residing in the ProvideX library (utilities, subprograms, files, etc.) whose first character begins with an asterisk, are case insensitive. A call to either **"*U"** or **"*u"** will access the same utility.

The majority of the utilities are available by a function or control key sequence. To access the utilities, press function key 5 (default value). This default function key can be changed by the system configuration utility ***UCK**.

Less common utilities and those requiring detailed knowledge of ProvideX internals, including changing the message library and language tables, must be run manually.

Screen Interface

Most of the utilities use the top two lines of the screen for input. The first line describes the action and the second line is used for either input entry or option selection. In cases where the input allows for the selection of one of a variety of options, each option will be displayed on the second line with the most common (or least destructive) option appearing first. Type the first letter of the option or use the arrow keys to highlight the desired option and press ENTER.

Throughout this document, examples showing the top two lines have been shaded with the top, left and right sides boxed in. An example of an option line with 6 values follows:

```
Selection:  
Calculator Files Phone Utilities Edit Quit
```

NOTE: All input to the utilities should be terminated by an ENTER key, a single character code, or a function key. The ENTER key is marked RETURN on some keyboards, either one being acceptable.

NOTE: All utilities support the Mouse for selecting options. 

**

Utility Subsystem main menu

Description:

This is the main utility subsystem menu. Typically, it is called by the input intercept routine "CONTROL" when a CTL value of -1 is received. It can also be CALLED or RUN directly.

Selection: Calculator Files Phone Utilities Edit Quit

Select:

<i>Calculator</i>	to run the Built-in calculator. (*C)
<i>Files</i>	to run the display of open files. (*F)
<i>Phone</i>	to run the Phone directory utility. (*P)
<i>Utilities</i>	to run the top level of the System utilities. (*U)
<i>Edit</i>	to run the program full screen editor. (*E)
<i>Quit</i>	to exit the utility.

Unable to process. Try another selection
--

This message indicates that an unrecoverable error occurred in a subordinate utility program.

*C

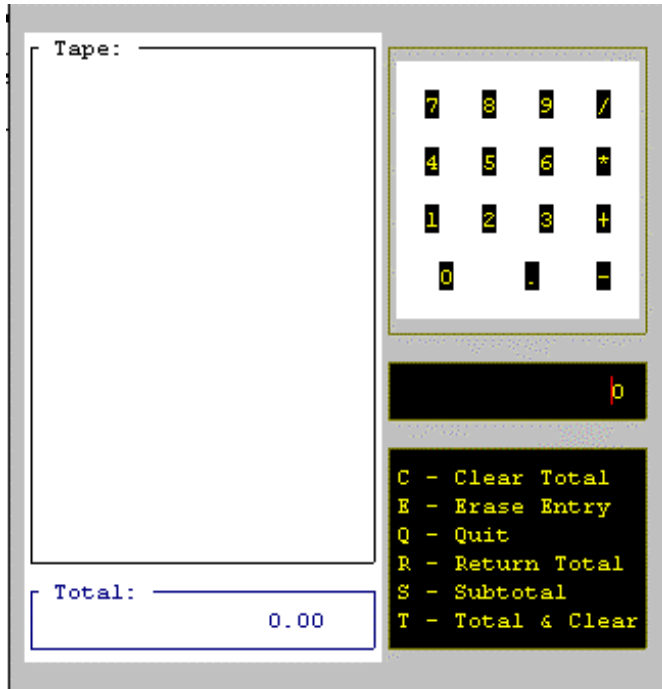
Built-in Calculator

Description:

This utility provides a built-in calculator function within ProvideX. It allows for the addition, subtraction, multiplication, and division of numbers. The display includes a 'TAPE' column that lists the last few entries.

Key	Action
0-9 and .	These numbers are used to enter values
+	Adds the next entry to the total
-	Subtracts the next entry from the total
/	Divides the current total by the next entry
*	Multiplies the current total by the next entry
ENTER	Ends entry and prepares for next entry. If hit twice in a row, a subtotal is entered in the TAPE.

Key	Action
C	Clears the current total.
E	Erases the current entry field
Q	Quits the calculator
R	Places the current total value into the input buffer, thereby returning it to the program currently running.
S	Inserts a Sub-Total in the TAPE listing
T	Totals the input and inserts it in the TAPE listing.


***C – Built-in Calculator (continued):**

Input to this utility is done in 'Polish notation', which means that the value is entered first, then the operation. This is based on the standard method employed by automatic adding machines.

Example:

To multiply 5 and 10 together:

Press	C	- to clear the calculator
	5	- to enter 5
	ENTER	- to end the entry of 5
	1	
	0	- to now have 10 in the entry field
	*	- to multiply the numbers

NOTE: This utility supports the Mouse. 

***CHKKEY**

Keyed file Trace

Description:

This utility is used to trace a key file chain to determine the tree structure for a given record. Its purpose is to aid in the repair of a damaged key file.

Note: This program is not accessible directly from the standard utility menus. To invoke these programs issue a RUN or CALL directive.

Once invoked, enter the name of the file to be processed, the key number to trace, and the key value desired. The program will walk the file's key tables and display each entry as it is encountered.

NOTE: To use the information that this utility provides, requires a thorough understanding of the internals of ProvideX keyed file structures. This is beyond the scope of this manual. Please contact your local ProvideX distributor if you need additional technical information regarding keyed file structures.

***E**

Full Screen Program editor

Description:

This utility provides a full screen editor for program development and maintenance. It edits the current 'main-line' program (the program at level 1). To invoke the utility, select "E" from the main utilities menu or enter CALL "*E".

'xxxxxx' is running. Terminate it (Y/N)

This message is displayed if the editor was invoked while a program was running. Enter a 'Y' to terminate it or enter an 'N' to abandon the editor session. The editor cannot edit a program that is currently running.

The editor displays the current program on the screen. The program may be viewed and edited directly on the screen through the use of the cursor movement keys (Up arrow, Down arrow, Page Up, Page Down, Home, End). If active, the mouse may be used. Any Program compile errors are displayed on the screen below the lines on which they occur.

The top of the screen contains the editor menu.

F1-Text edit F2-Line edit F3-Program F4-Quit

Press:

F1 to select the text editor "String Functions" which provide the ability to search, replace, copy, and paste strings.

***E – Full screen program editor (continued):**

- | | |
|----|--|
| F2 | to select the line editor "Line Functions" which provide the ability to insert lines, go to a line, delete line(s), copy, and paste lines. F2 also provides access to an external Text Line library. |
| F3 | to select the Program functions which include saving, loading, renumbering, and starting a new program. |
| F4 | to exit the editor. The edited program will remain the current mainline program after exiting. |

String functions
Find Replace Snip Copy Paste Quit

Select:

- | | |
|----------------|---|
| <i>Find</i> | to locate a string within the program. |
| <i>Replace</i> | to replace a string with another. |
| <i>Snip</i> | to highlight and place a string into the copy buffer. |
| <i>Copy</i> | to highlight and copy a string into the copy buffer. |
| <i>Paste</i> | to insert the copy buffer into a line of code. |
| <i>Quit</i> | to exit the utility. |

Enter string to find

If FIND is selected, enter the string to locate and press ENTER.

Find more occurrences
No Yes Quit

Select:

- | | |
|-------------|----------------------|
| <i>No</i> | to stop searching. |
| <i>Yes</i> | to continue. |
| <i>Quit</i> | to exit the utility. |

String not found! Press return to continue.

The above message is displayed if the string was not found.

Enter string to be replaced

If REPLACE is selected, enter the string to be found and changed.

Replace 'xxxxxx' with what?

Enter the new value for the string. Once entered, the editor will start to scan for the original string within the program and highlight it.

Replace this occurrence?
Yes No All Quit

E - Full screen program editor (continued):*Select:**

<i>Yes</i>	to replace the highlighted occurrence.
<i>No</i>	to skip this replacement.
<i>All</i>	to have the system automatically replace this and all subsequent occurrences of the specified string.
<i>Quit</i>	to exit the utility.

Highlight string to snip and press <Return>

If SNIP is selected, highlight the text to be moved to the copy buffer. The cursor keys (or mouse) are used to highlight text starting from the point the SNIP option was selected, up to the point the ENTER key is pressed. Once ENTER is pressed, the string highlighted is moved to the copy buffer and removed from the program.

Highlight strings to copy and press <Return>

If COPY is selected, highlight the text to be copied into the copy buffer. The cursor keys (or mouse) are used to highlight text starting from the point the COPY option was selected, up to the point the ENTER key is pressed.

:

Line functions**Goto Insert Delete Snip Copy Paste Move Undo Library Quit****Select:**

<i>Goto</i>	to move the cursor to a specific line.
<i>Insert</i>	to start inserting lines.
<i>Delete</i>	to highlight and delete a series of lines.
<i>Snip</i>	to highlight and move a series of lines into the copy buffer.
<i>Copy</i>	to highlight and copy a series of lines into the copy buffer.
<i>Paste</i>	to insert the contents of the copy buffer into the program at this point. Inserted lines will have all internal line references adjusted.
<i>Move</i>	to highlight and move lines from one point to another.
<i>Undo</i>	to reset the last changed line back to what it was. This is effective only on the last single line changed.
<i>Library</i>	to access the code library.
<i>Quit</i>	to exit the utility.

Line number

If GOTO is selected, enter the line number of the line to go to.

Enter line number to start insertion at

If INSERT is selected, enter the starting line number. When inserting lines, the system automatically displays new line numbers to allow the entry of the contents of the line. This continues until a null line is entered or another system command is executed. If required, the editor automatically renumbers the program allowing new lines to be entered.

E - Full screen program editor (continued):*Highlight lines to delete and press <Return>**

If DELETE is selected, highlight the lines to be removed. The cursor keys (or mouse) are used to highlight starting from the line where the DELETE option was selected on, up to and including the line where the ENTER is pressed.

Highlight line to snip and press <Return>

If SNIP is selected, highlight the lines to be moved to the copy buffer. The cursor keys (or mouse) are used to highlight the lines, starting from the line the SNIP option was selected on, up to and including the line where the ENTER is pressed.

Highlight lines to copy and press <Return>

If COPY is selected, highlight the lines to be copied to the copy buffer. The cursor keys (or mouse) are used to highlight the lines, starting from the line the COPY option was selected on, up to and including the line where the ENTER is pressed.

Highlight lines to move and press <Return>

If MOVE is selected, highlight the lines to be moved. The cursor keys (or mouse) are used to highlight the lines, starting from the line the MOVE option was selected on, up to and including the line where the ENTER is pressed.

Position to line desired and press <Return>

Once the lines to MOVE have been highlighted, position the cursor to the line before which the lines are to be inserted and press ENTER.

Cannot move -- Program too large -- Press <Return>

If the movement of lines causes an overflow in the line numbers (exceeding 65000), the above message appears.

If LIBRARY is selected, the editor allows you to copy elements of the program to a 'Line Library' or the insertion of lines from a library into the program. A Line Library consists of a ProvideX keyed file where lines of code are stored. Each group of lines in the library has a name, which is used to identify the contents.

The name of the library must be identified the first time that the Line Librarian component of the editor is used.

Name of program text library

Enter the name of the file containing the text library. If the library does not exist, the editor creates it.

Text library functions
Copy-to-library Paste-from-library Set-library Quit

E - Full screen program editor (continued):*Select**

<i>Copy-to-library</i>	to highlight a series of lines and copy them to the library.
<i>Paste-from-library</i>	to insert a series of lines from the library.
<i>Set-library</i>	to change the library file name.
<i>Quit</i>	to exit the utility.

Highlight text to copy to library
Name of library element to write to: _

If COPY_TO_LIBRARY is selected, enter the name to be assigned to the library entry that will receive the highlighted lines, then highlight the lines as in standard line COPY above.

Name of library element to paste _

If PASTE_FROM_LIBRARY is selected, enter the name of the entry in the text library which is to be inserted. The lines are inserted before the current line in the program.

General program functions
Save Load Renumber Delete Quit

Select:

<i>Save</i>	to save the program.
<i>Load</i>	to load a different program for editing.
<i>Renumber</i>	to renumber the current program.
<i>Delete</i>	to delete the complete program and start from scratch on a new one.
<i>Quit</i>	to exit the utility.

Name of program to save to

If SAVE is selected, enter the name of the program file to be saved to. By default, the program will be saved back onto the current file.

What program do you want to load

If LOAD is selected, enter the name of the program file.

'xxxxxx' is not a program -- Try again

The above message appears if LOAD is selected for a non-program file.

No such program

The above message appears if LOAD is selected for a non-existent file.

F**Display of Open Files*****Description:**

This utility is used to display all the files currently open. It can be selected from the main utility menu (**). Once invoked, this program will display the open files with their file number, type, record size information and full path name.

If more files are open than will fit on one screen, the following question is asked:

<p>More open files to follow. Continue? Yes No Quit</p>

If all open files have been displayed, the following is asked:

<p>Open files display. Re-display? No Yes Quit</p>
--

In either case...

Select:

<i>No</i>	to stop the display.
<i>Yes</i>	to continue display.
<i>Quit</i>	to exit the utility.

FI**Display File information*****Description:**

This combination utility program and subprogram is used to display the current information about a file. If CALLED as a subprogram, the file name passed in the first argument is used, otherwise, the user is prompted (via **OPEN.FLE) to enter the name of the file.

Calling sequence:

CALL "*FI", IN_PTH\$

Where:

IN_PTH\$ must contain the name of the file for which the information is to be displayed.

<p>Note: This program is not accessible directly from the standard utility menus. To invoke this program issue a RUN or CALL directive.</p>

*FM

Generic File Maintenance

Description:

This program is used to display and update a keyed data file. It provides a spreadsheet-like format for the display and update of the file. Each field of a record is a column with each record being a single row. The menu allows column formats, column widths, and search criteria to be specified.

```

File:C:\$BBLIB\ ERROR.INF
File Edit Format Goto Select Quit
Rec#  Key  Fld1
 1  00000 Returned for one of the following reasons:
 2  00001 - On the OPEN indicates that the file is LOCKED by
 3  00002 another user.
 4  00003 - On a READ, FIND, EXTRACT, or INPUT the record is
 5  00004 EXTRACTED by another user.
 6  00005 - Also returned if a device timeout occurred on a device.
 7  00100 Returned for one of the following reasons:
 8  00101 - The data elements that comprise the record caused its
 9  00102 overall length to exceed the maximum record length
10  00103 defined for the file.
11  00104 - Attempting to DROP the only window
12  00200 The end of the input file has been reached or the output
13  00201 file is full or has reached its preset maximum record count.
14  00300 A physical error was returned from the device being
15  00301 accessed. In the case of a recurring error on a disk drive,
16  00302 the problem should be recorded and reported to your hardware
17  00303 maintenance supplier.
18  00400 A "not ready" status was returned from the device being
19  00401 accessed. If the device is a printer, the likely cause is
20  00402 that either it has run out of paper or that the off-line
F2-Menu F4-Quit * TEST VERSION *

```

This utility can be called as a subprogram by passing it the name of the file to process. The calling sequence is:


```
CALL "*FM", IN_PTH$
```

Where:

IN_PTH\$ is the name of the file to present.

NOTE: This program is not accessible directly from the standard utility menus. To invoke this program, issue a RUN or CALL directive.

This is a **preliminary** version of the utility and is representative of the next generation of ProvideX tools. It has been included with the current version of the utility set due to its unique capabilities and to serve as an example in its use of Mouse and scroll bars.

NOTE: This utility supports the Mouse. 

***LEXEDIT**

Syntax Table edit Utility

Description:

This utility allows the systems programmer to modify the internal ProvideX syntax tables thereby changing the names of the various directives, system variables, functions, mnemonics and options.

The modification of the syntax table is a three-step process:

- 1) Define new keywords, functions, etc.
- 2) Generate the syntax table from the new information.
- 3) Load the new table.

These steps can be accomplished through the options available in the *File* menu. To access this menu, press <F2> and select *File*:

Select:

<i>Open</i>	to add, modify, or delete keyword expressions. (See ' <i>Modifying a syntax table</i> ').
<i>Generate</i>	to generate a new version of the syntax table.
<i>Load</i>	to load a syntax table for use.
<i>Print</i>	to print a report which lists all of the keywords sorted in either keyword or object code sequence. The report includes information about output attributes (leading, trailing, and embedded spaces), and whether the keyword can be used as a label or as a variable. Alternate keywords include corresponding primary keywords.
<i>About</i>	to view copyright information.
<i>Quit</i>	to exit the utility.

*LEXEDIT is primarily designed to support multilingual systems. When accessing various menu options, you must first identify which language file to process. A language file is identified by a language code, which may be up to three characters long (For example, the code for the standard English lexicon is 'EN').

***LEXEDIT - syntax table edit utility (continued):**

```

Keyword Table Maintenance
File Quit
Processing C:\SBB\LIB\LEXDEF.EN
Keyword.....: [GO TO      ]

Primary keyword: < GO TO >
Object code:          b4
Can be variable:      Y
Can be label:         Y
Output with leading space (Y/N): [Y]
Output with trailing space (Y/N): [Y]
Output with embedded space (Y/N): [N]

Alternate keywords:
<NONE>

Keyword Syntax
Directive: X          Mnemonic: 'X'      Function: X (
or: X X              or: 'X' (         Option: , X =

Primary keyword
Modify Cancel
<F2> Menu  <F3> Prior Input  <F4> Quit

```

At the *keyword* prompt, enter the word or expression which is to be added, modified or deleted. If a new keyword is to be added, the *related keyword* (i.e. a keyword to which the new one corresponds) must be entered. The program will then display the following information:

If the keyword is a primary keyword (i.e. the one which is displayed in program listings), the output attributes (leading, trailing, embedded spaces) can be modified. Alternate (non-primary) keywords can be changed to primary keywords, but this results in the original primary keyword being changed to an alternate keyword. Only alternate keywords may be deleted.

Note: This program is not directly accessible from the standard utility menus. To invoke this program issue a RUN or CALL directive.

***MSGUPD**

Message File Maintenance

Description:

This utility program is used to update the message library. The ProvideX message library contains all of the "canned" text for error messages used by ProvideX and the default text strings used in the generation of alphanumeric day and month names used by the DTE function.

***MSGUPD – Message file maintenance (continued):**

The utility initially asks for the language code of the message library file to be updated. The language code is determined from the system Environment variable "LANG". It is used to define the file name by appending it to the name "*MLFILE.". If no environment variable is defined, ProvideX uses a default of "EN". Once the language has been determined, enter the error message number, then edit its associated text.

Note: This program is not accessible directly from the standard utility menus. To invoke this program issue a RUN or CALL directive.

P**Phone Directory*****Description:**

This utility allows the user to maintain a list of public and private phone numbers. Each entry in the phone directory consists of a person's name, company, and telephone number.

Persons Name	Company	Phone
Charles Dickens	Scrooge Incorporated	01144-71-5551212
* Mike King	Sybex Ltd	(905) 470-1025
Mary Poppins	Home Care Ltd	555-1234

<*> End of file <*>

To enter new names into the phone directory, press the ENTER key. The system presents a window for the entry of the person's name, company, phone number and an option to indicate whether this phone number is Public (available to all users) or Private (available to you only).

To search for a name, select the search mode by pressing F1 or F2, then enter the name desired. Press ENTER to begin the search.

U**Main Utility Menu*****Description:**

This utility program displays the major categories of utilities. The user selects a category from the list presented and a subsequent menu (or program) is run.

***U – Main utility directory (continued):**

System utilities: Files Directories Programs Configuration General Quit

Select:

<i>Files</i>	to create, delete, rename, view, update or otherwise manipulate data files. (*UF)
<i>Directories</i>	to create, delete, rename, or view directories. (*UD)
<i>Programs</i>	to create, delete, rename, edit, list or otherwise manipulate program files. (*UP)
<i>Configuration</i>	to alter the configuration of the ProvideX environment. (*UC)
<i>General</i>	to access the general utilities, such as the Mortgage computation program and the Spreadsheet export program. (*UG)
<i>Quit</i>	to exit the utility.

Unable to process. Try another selection
--

This message indicates that an unrecoverable error occurred in a subordinate utility program.

UC**System Configuration Sub-menu*****Description:**

This utility provides the configuration Sub-menu.

Configuration utilities: Linkfiles Keyboard Parameters Quit

Select:

<i>Linkfiles</i>	to define/update link files. These Linkfiles are used to establish a relationship between a file name in a program and the physical device. For example, a Linkfile 'LP' may be made to link to the real file '/dev/lp'. (*UCL)
<i>Keyboard</i>	to define the keyboard input control sequences that your terminal will send. (*UCK)
<i>Parameters</i>	to view and adjust any of the ProvideX system parameters. (*UCP)
<i>Quit</i>	to exit the utility.

Unable to process. Try another selection
--

This message indicates that an unrecoverable error occurred in a subordinate utility program.

***UCK**

Keyboard Configuration

Description:

This utility is used to define the input sequences that will be received from the various terminal keyboards. It allows editing or the re-assignment of the key sequences by manually pressing the keys on the keyboard rather than having to look them up in the terminal technical manual.

All keyboard input sequences are maintained by terminal type and optionally, by user-id. This allows custom implementations for specific users.

Keyboard definition utility for xxxxxx. Is this specific to xxxxxx
No Yes Quit

Select:

No if the keyboard definition will **not** be specific to the current user-id.
Yes if the keyboard definition will be specific to the current user-id.
Quit to exit the utility.

The system presents a screen to define or alter the key input sequences for your terminal. The screen display consists of four columns each with up to twenty key labels. Initially, the key label F1 is highlighted. To change the input sequence that F1 represents, press the ENTER key and then press the key (or keys) on the terminal that are to correspond to F1.

To select a different key to change, press U to move up, D to move down, R to move right, or L to move left. The Up, Down, Left, and Right arrows may also be used, once they have been defined. Pressing Q will quit the utility.

```

Using cursor keys (or U,D,L,R) highlight key to define
Press <Return> when ready or Q to quit:
F1 :003B      ESCAPE:03      UTILS :0041      CTL-10:
F2 :003C      ERASE :7F       PRTSCR:0019     CTL-11:
F3 :003D      BKSPC :08       SWITCH:         CTL-12:
F4 :003E      LEFT  :004B     STATS :005A     CTL-13:
F5 :          RIGHT :004D     HELP  :0040     CTL-14:
F6 :          DELETE:0053    QUERY :003F     CTL-15:
F7 :          INSERT:0052   PGINF:0059     CTL-16:
F8 :          INSSPC:0092   SCNRST:0097    CTL-17:
F9 :          ADVSPC:      CTL-18:
F10 :         HOME  :0047    CTL-19:
F11 :         UP    :0048
F12 :         DOWN :0050
F13 :         PGUP :0049
F14 :         PGDN :0051
F15 :         TABFWD:09
F16 :         TABBCK:000F      ALT-F1:
F17 :         RSTART:         RSTORE:0077    ALT-F2:
F18 :         END  :004F      CLREOL:0093   ALT-F3:
F19 :         SHLEFT:0084     NXTHRD:0074   ALT-F4:1B
F20 :         SHRGT:0076     PRVHRD:0073

```

***UCK - Keyboard configuration (continued):**

To restore all the key sequences to their default (system standard) definition, press *. Pressing * resets the terminal key definitions to the ProvideX defined standard, if one exists.

Are you sure you wish to restore from system default value?
No Yes Quit

Select:

<i>No</i>	to abandon the restore command.
<i>Yes</i>	to continue with the restore.
<i>Quit</i>	to exit the utility.

If a duplicate key input sequence is detected, the utility issues the following message.

Already defined as function 'xxxxxx' Cancel old definition?
Yes No Quit

Select:

<i>Yes</i>	to remove the old definition for the key pressed.
<i>No</i>	to re-enter the keystroke.
<i>Quit</i>	to exit the utility.

UCL**Configure Link files*****Description:**

This utility is used to create and maintain device, file, and printer link files and for the specification of the link's path name and device driver.

'Link' files utility
Name of 'Link' file? _

Enter the name of the Link file to create/update.

File already exists and is not a link!!

The above error message is displayed if a non-link file of the same name exists.

The following message is displayed if a link file of the same name exists.

Do you want to delete 'xxxxxx'
No Yes Quit

Select:

<i>No</i>	to maintain and modify the link file.
<i>Yes</i>	to delete the link file.
<i>Quit</i>	to exit the utility.

***UCL - Configure link files (continued):**

'xxxxxx' has been erased

The above message is displayed to confirm that the link file has been deleted.

Unable to delete:xxxxxx

The above message indicates that an error occurred during the delete procedure. The most common cause is lack of permissions.

The following message is displayed if the link file did not already exist.

File 'xxxxxx' does not exist. Create?
Yes No Quit

Select:

<i>Yes</i>	to create the link file.
<i>No</i>	to re-enter a different link file name.
<i>Quit</i>	to exit the utility.

'xxxxxx' links to...
Name of file/device: _

Enter the name of the file or device that this link file will point to.

What type of link is 'xxxxxx'
File Printer Device Quit

Select:

<i>File</i>	if the link points to a file.
<i>Printer</i>	if the link points to a printer.
<i>Device</i>	if the link points to a device of another type.
<i>Quit</i>	to exit the utility.

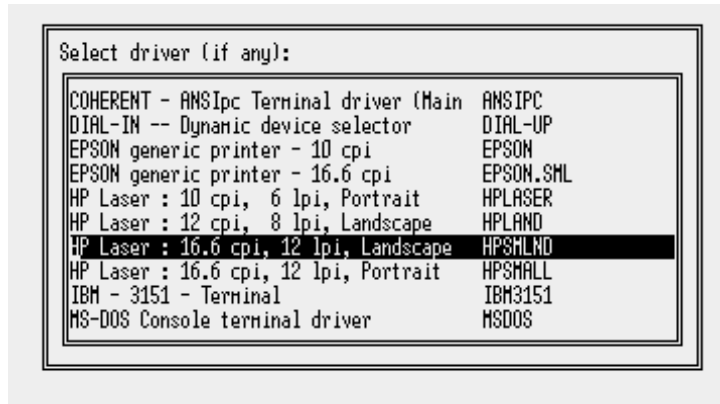
If PRINTER is selected, the utility asks how the printer is attached to the system.

Is this printer connected via a Terminal Aux port?
No Yes Quit

Select:

<i>No</i>	to indicate that this printer is connected directly to the computer system, or the operating system device drivers make it appear so.
<i>Yes</i>	to indicate the device is attached to a terminal and all output must have a Printer select command before the transmission and a de-select command after.
<i>Quit</i>	to exit the utility.

Once the type and path name have been determined for the link file, select the device type from the list presented. Use the Up/Down arrow keys to change the highlight on the screen and press ENTER.

***UCL - Configure link files (continued):**

```

Link file 'xxxxxx' created to xxxxxx (xxxxxx)
Name of 'Link' file? _

```

The above message indicates that the link file was created. If you wish to create more link/device files enter the name of another link file to create or change.

UCP**Configure system parameters*****Description:**

This utility is used to alter the current system parameters. It displays all current parameters in the system and allows them to be changed.

```

Parameter maintenance
Use cursor keys to highlight parameter then press ENTER. Press F4 to quit
AH=Off  Alt Window hdr  LC=Off  Lower case Var  UM=Off  Use UMBs
BF=10   # Keyed IO bfrs  LP=0n   Line prtr mode  VM=0n   Direct Video
BX=Off  BBX Compatible  MS=0    Real swap KB    WD=10000 Defer writes
BY=1970 Base julian year  NE=Off  Error in Sub    WK=Off  Keep windows
CD=Off  Current DIR 1st  NI=Off  Ignore spaces   WL=Off  Use write locks
CU=36   Currency char.  OP=Off  ORIGINAL paths  WT=2    #Secs for busy
DC=Off  Destructive cur  OH=0    SAVE Owner ID#  XF=Off  Extend file #s
DD=Off  Use / for DIR   PC=0    Program cache   XI=Off  Extract blocks
DP=46   Decimal point  PF=0n   Use Page frame  XS=0    XMS Swap KB
DT=0    Device timeout   PS=0    Max Swap size   XT=Off  Quit at END
FF=0    Fid format      QS=Off  Quick Start
FI=Off  Ignore mask err  QT=Off  Quiet No-prompt
FL=Off  Filenames lower  RR=Off  RESET on RUN
FP=0n   Floatingpoint      SD=Off  Append slashes
FS=138  Field Separator    SF=Off  Short VAR names
FU=Off  Filenames upper  SL=32   # saved crnds
HC=0n   Hide Cursor      SV=1    Save version
IC=Off  Ignore case      SZ=243  Max memory size
IZ=Off  Ignore SZ param  TH=44   Thousands sep

When the 'AH' parameter is On, the system will display the alternate form of
window and box heading lines. The alternate heading line consists if the title
centered in an Inverse video line. Default Off.

```


***UCL - Configure link files (continued):**

Use the Up/Down/Right/Left arrow keys on the keyboard to select the parameter to change. Then press ENTER. Parameters, which have only an On/Off state, will toggle their status when selected. Parameters which have values associated with them ask for a new value when selected.

NOTE: For information on the various system parameters, see the ProvideX Language Reference manual.

UD**Directory Utilities Sub-menu*****Description:**

This is the Directory utilities sub-menu program. It presents the directory functions available within the system and allows the user to chose one.

Directory utilities:
Make Delete Rename Print View Goto Quit

Select:

<i>Make</i>	to create new directories. (*UDM)
<i>Delete</i>	to delete existing directories. (*UDD)
<i>Rename</i>	to change the name of an existing directory. (*UDR)
<i>View</i>	to view the names and characteristics of the files contained in a directory. (*UDV)
<i>Print</i>	to print the names and characteristics of the files contained in a directory. (*UDP)
<i>Goto</i>	to change to a specific directory. (*UDG)
<i>Quit</i>	to exit the utility.

Unable to process. Try another selection

The above message indicates that an unrecoverable error occurred in a subordinate utility program.

UDD**Directory Delete*****Description:**

This utility deletes a sub-directory.

Delete directory utility
Directory to delete: _

Enter the name of the directory to delete

***UD - Directory utilities sub-menu (continued):**

'xxxxxx' is not a directory

The above message indicates that the name entered is not the name of a valid directory.

Are you sure you wish to delete 'xxxxxx'
No Yes Quit

Select:

No to abandon the deletion request.
Yes to delete the directory.
Quit to exit the utility.

Delete sub-ordinate files/directories if present?
No Yes Quit

Select:

No to delete the specified directory. The directory **MUST** be empty before it can be deleted.
Yes to delete all files and subordinate directories in the specified directory.
Quit to exit the utility.

Directory 'xxxxxx' deleted. More deletes?
No Yes Quit

Select:

No if there are no more deletes to be performed.
Yes to delete other directories.
Quit to exit the utility.

UDG**Change to a different Directory*****Description:**

This utility changes the current directory.

Goto directory utility (Change working directory)
Directory to goto (*=Starting directory): _

Enter the name of the directory to become the 'Current' directory. Enter an asterisk (*) to return to the directory that you were in when you started ProvideX.

'xxxxxx' is not a directory!

The above message appears if the name entered is not the name of a directory.

No such directory!

The above message appears if the name entered does not exist.

UDM**Make a new sub-directory***

Description:

This utility creates new directories.

Make a directory utility

Directory to make: _

Enter the name of the directory to be created.

Directory 'xxxxxx' already exists

The above message is displayed if a file or directory of the name specified already exists.

Directory 'xxxxxx' has been made. More to make?

No Yes Quit

Select:

<i>No</i>	if no more directories are to be made.
<i>Yes</i>	to make more directories.
<i>Quit</i>	to exit the utility.

UDP**Directory Print***

Description:

This utility program prints the contents of a directory on a printer. This listing contains file name, type, record size, key information and other related data.

Print directory utility

Directory to print: _

Enter the name of the directory to print.

What type of printout?

Detailed Summary Quit

Select:

<i>Detailed</i>	to print file information such as file type, key size, etc.
<i>Summary</i>	to print file names only.
<i>Quit</i>	to exit the utility.

Sort report by filename?

Yes No Quit

UPD – Directory Print (continued):*Select:**

Yes	if the printout is to be sorted by file name.
No	if the printout is not to be sorted.
Quit	to exit the utility.

Print other directories?
No Yes Quit

Select:

No	if no more directories are to be printed.
Yes	to print more directories.
Quit	to exit the utility.

UDR**Change Directory name*****Description:**

This utility renames a directory. *Not all operating systems support this functionality.*

Directory rename utility
Directory to rename: _

Enter the name of an existing directory to rename.

'xxxxxx' is not a directory

The above message is displayed if there is no directory with the name specified.

Renaming directory 'xxxxxx'
Enter new name: _

Enter the new name for the directory.

Directory 'xxxxxx' now 'xxxxxx'. More renames?
No Yes Quit

Select:

No	if there are no more directories to rename.
Yes	to rename other directories.
Quit	to exit the utility.

*UDV *Directory View*

Description:

This utility allows the viewing of the file names within a directory. The display consists of the file names, type, and other related information such as key size, record size, etc.

View directory utility
Directory to view: _

Enter the name of the directory to view.

'xxxxxx' is not a directory

The above message is displayed if the name entered is not a valid directory.

What type of display?
Detailed Summary Quit

Select:

<i>Detailed</i>	to display file information such as file type, key size, etc..
<i>Summary</i>	to display file names only.
<i>Quit</i>	to exit the utility.

Partial listing. Continue display
Yes No Quit

Select:

<i>Yes</i>	to continue the display.
<i>No</i>	to terminate the display.
<i>Quit</i>	to exit the utility.

Do you wish to view more/other directories
No Yes Quit

Select:

<i>No</i>	if there are no further directories to display.
<i>Yes</i>	to display other directories.
<i>Quit</i>	to exit the utility.

Partial listing. Continue display
Yes No Quit

-OF-

Partial list of alt. keys. Continue display
Yes No Quit

Select:

<i>Yes</i>	to continue display.
<i>No</i>	to terminate display.
<i>Quit</i>	to exit the utility.

UF**File Maintenance sub-menu*****Description:**

This utility program allows the user to select a sub-function of the file maintenance portion of the system utilities.

File system utilities:

Make Delete Rename Print View Update Info Copy Erase Admin Quit

Select:

<i>Make</i>	to create a new file. (*UFM)
<i>Delete</i>	to delete a file. (*UFD)
<i>Rename</i>	to change the name of a file. (*UFR)
<i>Print</i>	to print the contents of a file. (*UFP)
<i>View</i>	to view the contents of a file. (*UFV)
<i>Update</i>	to change the contents of a file. (*UFU)
<i>Info</i>	to obtain file information such as type, record size, keys, etc.. (*UFI)
<i>Copy</i>	to copy the contents of a file. (*UFC)
<i>Erase</i>	to clear the contents of a file. (*UFE)
<i>Admin</i>	to access the file administration sub-menu. This sub-menu contains utilities to change the file maximum record counts, validate the integrity of a file, and recover a corrupted file. (*UFA)
<i>Quit</i>	to exit the utility.

Unable to process. Try another selection

The above message indicates that an unrecoverable error occurred in a subordinate utility program.

UFA**File Administration Sub-menu*****Description:**

This utility program provides the file administration sub-menu .

File administration utilities:

Maximum_size Check Re-construct Quit

Select:

<i>Maximum_size</i>	to change the maximum size of a file. (*UFAM)
<i>Check</i>	to verify the integrity of a keyed file. (*UFAC)
<i>Re-construct</i>	to repair a damaged keyed file. This utility copies each physical record into a holding file then re-creates and re-loads the keyed file. (*UFAR)
<i>Quit</i>	to exit the utility.

Unable to process. Try another selection

The above message indicates that an unrecoverable error occurred in a subordinate utility program.

*UFAC

Keyed file integrity check

Description:

This utility is used to verify the integrity of the key structure of a keyed data file. Typically, it would be used after some form of hardware or operating system malfunction where the writing of data to the disk drives was unable to be guaranteed successful. This utility can also be CALLED as a subprogram to verify a file.

Calling sequence:

CALL "*UFAC", CALL_FILE\$, CALL_DSP

Where:

CALL_FILE\$ The file to be verified.

CALL_DSP If this argument is set to 1, the utility will provide a status display as it is verifying the file. If zero (0), no display is provided.

Check keyed file
Name of file: _

Enter the name of the file to be verified.

Pause at each error?
Yes No Quit

Select:

Yes to stop after each error is detected.

No to continue processing and simply report the total number of errors at the end of the verification run.

Quit to exit the utility.

Display index trace?
Yes No Quit

Select:

Yes to have the utility display the various keyed index entries as they are being verified. While this display is helpful in detecting a problem, but it can be very time consuming on a slow terminal.

No to prevent a trace of the index entries.

Quit to exit the utility.

Log errors to printer?
No Yes Quit

Select:

No to display the errors.

Yes to display and print an error log.

Quit to exit the utility.

UFAC – Keyed file integrity check (continued):*Errors:**

The following are the errors detected by the file integrity check utility:

Key/record mismatch - Idx (xxxxxx) - Rec (xxxxxx)
Continue Quit

The key table entry and the contents of the record do not match. Select 'Continue' to proceed.

Key block @xxxxxx has invalid header address of xxxxxx
Continue Quit

When reading the key tables, a disk block was read whose key block header address does not match that of the block that was requested.

Record count error - xxxxxx - xxxxxx of xxxxxx records
Continue Quit

After verifying the key tables, the total count of records in the file found by keys, does not match the number of records expressed in the file header.

Deleted record - address xxxxxx
Continue Quit

A deleted record was detected while reading using the key table. This should not occur since deleted records have their associated key table entries removed.

Key [xxxxxx] at unknown address xxxxxx does not exist
Continue Quit

A key table entry pointed to a non-existent record address.

Free record link list loop
Continue Quit

There is a loop within the free (deleted) record list.

Key block @xxxxxx has invalid type. Should be \$xxxxxx\$ was \$xxxxxx\$
Continue Quit

When reading the key tables, a disk block was read whose key block header type indicator was invalid.

Key block @xxxxxx has invalid length.
Continue Quit

When reading the key tables, a disk block was read whose key block header size field was invalid.

Record @xxxxxx has invalid length
Continue Quit

The data record read has an invalid record size field.

***UFAC - Keyed file integrity check (continued):**

Record @xxxxxx has invalid index
Continue Quit

While verifying a variable length record file, the record address (which includes data block number and record number within the block) indicated a non-existent record in the data block.

Record @xxxxxx has invalid offset
Continue Quit

While verifying a variable length record file, the offset to the record within the data block was invalid.

NOTE: In general, if an error is detected, use the utility "*UFAR" to attempt to recover the data. If this fails restore the file from a backup.

UFAM**Change file record count limit*****Description:**

This utility alters the maximum number of records that may be placed on a file. Normally, ProvideX files are created with no upper limit to the number of records other than disk capacity. During the development cycle, it may be advantageous to set a limit to avoid allocating disk space to a file if a program goes into a loop.

Adjust maximum number of records
Name of file: _

Enter the name of the file to be changed.

Adjust maximum number of records
Enter maximum number of records (**=no limit) _

Enter the new maximum number of records for the file. Enter an asterisk (*) to set no limit.

Adjust maximum number of records
Enter maximum record size:(current size)

Enter the new maximum record size (increase only, to decrease you must still reload the data file.)

Record adjustment complete for 'xxxxxx'. Adjust another file?
No Yes Quit

Select:

No	if there are no other files to adjust.
Yes	to change the maximum record count on other files.
Quit	to exit the utility.

*UFAR

Recover Keyed file

Description:

This utility is used to recover a damaged keyed file. It reads all of the records in a data file by record index and copies them to a holding file. Once this is complete, the original file is cleared and the holding file records are rewritten back to the original file.

KEYED/DIRECT file key reconstruction utility
Name of KEYED/DIRECT file: _

Enter the name of the file to be reconstructed.

'xxxxxx' is not a recoverable KEYED/DIRECT file

The above message indicates that the selected file is not a recoverable keyed file.

Cannot open 'xxxxxx':xxxxxx

The above message indicates the file could not be opened. The actual reason for the open failure appears in the message after the colon.

A temporary file will be needed to hold the data during recovery
Enter name of temporary file: _

The system will provide the name of a temporary holding file based on the original file name. Press ENTER to use the name provided or enter a different name. If the file being recovered is large, the holding file may need to be redirected to a different system device.

Temporary file 'xxxxxx' already exists. Overwrite it?
Yes No Quit

Select:

<i>Yes</i>	to overwrite the existing holding file
<i>No</i>	to enter a different holding file name.
<i>Quit</i>	to exit the utility.

Recover to highest active index or physical end of file?
Active_index Physical_EOF Quit

Select:

<i>Active_index</i>	to recover the data up to the location indicated in the file header as the end of the file. If the file was re-initialized accidentally or there is some question as to the validity of the file header, DO NOT select this option.
<i>Physical_EOF</i>	to recover to the physical end of the disk file as allocated by the operating system. This may result in some unwanted records appearing in the output file.
<i>Quit</i>	to exit the utility.

***UFAR – Recovered keyed file (continued):**

At this point the utility will try to read the first few records in the file in an attempt to develop some data validation rules. These rules will include the number of fields in each record, key sizes, and key content. The rules will be displayed and the user is asked to confirm their validity for all records on the file.

Is this correct for ALL records in this file?

Yes No Quit

Select:

Yes to indicate that the rules are valid for ALL records in the file.
No to indicate that the rules DO NOT apply to all the data records in the file.
Quit to exit the utility.

Now the utility reads all the records from the corrupted file by record index. If the record appears to be correct, it is copied to the holding file. This proceeds until the specified end point of the file is reached.

Recovered xxxxxx records. Clear and reload original file?

Yes No Quit

Select:

Yes to clear the contents of the original file and to copy the holding file into it.
No to leave the original file unchanged and leave the data on the holding file.
Quit to exit the utility.

File 'xxxxxx' repaired successfully. Erase temporary file 'xxxxxx.BAK'

Yes No Quit

Select:

Yes to delete the holding file.
No to keep the holding file.
Quit to exit the utility.

***UFC**

File Copy

Description:

This utility program is used to copy the contents of a ProvideX file. The copy reads and writes physical records between the files. If the input file has an external key and the output file does not, the key will be inserted in front of each record's data separated by a ProvideX separator character (Hex 8A). If the output file has an external key and the input file does not, all data up to the first ProvideX separator character is used as the external key and is dropped from each record's data. Optionally this utility may be CALLED as a subprogram giving the input and output file names.

Calling sequence:

CALL "*UFC", CALL_FR\$, CALL_FR\$, CALL_TO\$, CALL_CL\$, CALL_DSP

UFC – File copy (continued):*Where:**

CALL_FR\$ must contain the name of the input file.
CALL_TO\$ must contain the name of the output file.
CALL_CL\$ is set to "Y" to pre-clear the output file.
CALL_DSP if set to 1, causes the utility to display a progress report on the screen.

File copy utility
 What is the name of the file to copy from? _

Enter the name of the file from which to copy.

Input file is 'xxxxxxx'
 What is the name of the output file? _

Enter the name of the file, which is to receive the data.

The utility asks if the output file is to have all existing data removed before adding the data from the input file.

Pre-clear output file 'xxxxxxx'
 No Yes Quit

Select:

No to only add the data to the output file.
Yes to have all existing data in the output file removed prior to adding the input data.
Quit to exit the utility.

***UFD**

File Delete

Description:

This utility deletes a file from the system and returns its disk space to the operating system.

Delete file utility
 File to delete: _

Enter the name of the file to delete.

'xxxxxx' is not a data file

The above message is displayed if the name specified is not the name of a data file.

Are you sure you wish to delete 'xxxxxx'
 No Yes Quit

UFD – File delete (continued):*Select:**

<i>No</i>	to keep the file.
<i>Yes</i>	to delete the file.
<i>Quit</i>	to exit the utility.

File 'xxxxxx' deleted. More deletes?
No Yes Quit

Select:

<i>No</i>	if there are no more deletes.
<i>Yes</i>	to delete other files.
<i>Quit</i>	to exit the utility.

NOTE: Once a file has been deleted, there is no way to recover its information other than reloading it from a backup.

UFE**File Data Erase*****Description:**

This utility is used to erase the contents of a data file while preserving the file definition.

Erase file contents utility
File to clear: _

Enter the name of the file to clear.

'xxxxxx' is not a data file

The above message is displayed if the name specified is not the name of a data file.

File 'xxxxxx' erased. More erasures?
No Yes Quit

Select:

<i>No</i>	if there are no more files to clear.
<i>Yes</i>	to clear other files.
<i>Quit</i>	to exit the utility.

UFI**Display file Information*****Description:**

This utility displays file information consisting of the file type, number of records, key information, record size, and other related information (block size, link references, etc.) depending on file type.

***UFI – Display file information (continued):**

Display file information
Name of file: _

Enter the name of the file, which is to be displayed. Once the file name is entered, the utility displays the true path name to the file and all other applicable file information.

***UFM**

Make a data file

Description:

This utility is used to create new data files.

Make file utility
File to make? _

Enter the name of the file to create.

'xxxxxx' already exists

The above message is displayed if the name specified is the name of a file (or directory) that already exists.

What type of file will 'xxxxxx' be?
Keyed Indexed Serial Quit

Select:

<i>Keyed</i>	if the file is to have a key (or keys).
<i>Indexed</i>	if the file is to be indexed.
<i>Serial</i>	if the file is to be Serial. There are no other questions asked regarding Serial files. Once this option is selected, the file will be created.
<i>Quit</i>	to exit the utility.

Creating Keyed file 'xxxxxx'
Maximum record size (0=Sort file -- Key only)? _

Enter the maximum record size for the file.

Do you want FIXED or VARIABLE length records
Variable Fixed Quit

Select:

<i>Variable</i>	to create a variable length record file.
<i>Fixed</i>	to create a fixed length record file.
<i>Quit</i>	to exit the utility.

What percentage of space do you want reserved for record expansion
Enter percentage (5-80)? _

***UFM – Make a data file (continued):**

If a VARIABLE length record is selected, enter the percentage of growth to reserve for each record. Records can grow beyond the value entered, but performance is better if the value is accurate.

Define keys for file 'xxxxxx'
Size of EXTERNAL key (0=No external key): _

Enter the size of any external key.

Are there any more keys to be defined?
No Yes Cancel Quit

Select:

No if there are no additional keys to define.
Yes to define additional keys.
Cancel to change the file definition.
Quit to exit the utility.

More components for 'xxxxxx'
No Yes Cancel Quit

Select:

No if there are no additional components in the key.
Yes if there are additional components in the key.
Cancel to change the file definition.
Quit to exit the utility.

Define xxxxxx - xxxxxx
Enter field number containing key data xxxxxx: _

Enter the field number within the record that contains the key information. Enter 0 for the complete record without regard to field separators, or KEY for the external key, if one exists.

Define xxxxxx - xxxxxx
Enter offset within field xxxxxx: _

Enter the offset within the field. Enter 1 for the start of the field.

Define xxxxxx - xxxxxx
Enter length of key component: _

Enter the length of the key segment.

Key sequence for xxxxxx - xxxxxx
Ascending Descending Cancel Quit

Select:

Ascending to specify an ascending key.
Descending to specify a descending key.
Cancel to change the file definition.
Quit to exit the utility.

***UFM – Make a data file (continued):**

Making SORT file 'xxxxxx'
What is the maximum key size (0<n<99)? _

Enter the maximum key size for the SORT file.

Making INDEXED file 'xxxxxx'
Maximum record size? _

Enter the maximum record size for the INDEXED file.

What is the maximum number of records (*=no limit)? _

Enter the maximum number of records that the file will be allowed to contain. Entering an asterisk (*) indicates that there is no preset limit to the number of records.

File 'xxxxxx' created. More files to make?
No Yes Quit

Select:

<i>No</i>	if there are no more files to create.
<i>Yes</i>	to create additional files.
<i>Quit</i>	to exit the utility.

***UFP** *File print*

Description:

This utility is used to print the contents of a data file.

Print file utility
File to print: _

Enter the name of the file to print.

'xxxxxx' is not a printable file

The above message will be displayed if the file name entered does not refer to a file that can be printed.

Print multiple fields per line
Yes No Quit

Select:

<i>Yes</i>	to print multiple fields on one line separated by a vertical bar.
<i>No</i>	to print one field per line.
<i>Quit</i>	to exit the utility.

***UFP – File print (continued):**

Printing file 'xxxxxx'
Enter starting key: _

-Or-

Printing file 'xxxxxx'
Enter record number to start printing at: _

Identify the starting record at which to begin printing.

Starting key 'xxxxxx' for file 'xxxxxx'
Enter ending key: _

-Or-

Starting record number 'xxxxxx' for file 'xxxxxx'
Enter record number to end printing at: _

Identify the record at which to end printing. If no ending key is entered, the file is printed until the end of file is reached.

UFR**Change File name*****Description:**

This utility renames data file(s).

File rename utility
File to rename: _

Enter the name of the existing file to be renamed.

File 'xxxxxx' does not exist

The above message is displayed if no file with the name specified can be found.

'xxxxxx' is not a data file

The above message is displayed if the name specified is not the name of a data file.

Renaming file 'xxxxxx'
Enter new name: _

Enter the new name for the file.

File 'xxxxxx' already exists

The above message is displayed if a file with the new name already exists.

File 'xxxxxx' now 'xxxxxx'. More renames?
No Yes Quit

UFR – Change file name (continued):*Select:**

<i>No</i>	if there are no more renames required.
<i>Yes</i>	to rename other files.
<i>Quit</i>	to exit the utility.

***UFU**

File Update

Description:

This utility updates the contents of a data file. It allows for the addition, deletion, and modification of data records.

Update data file utility
File to update: _

Enter the name of the data file to be updated.

'xxxxxx' is a program file - not modifiable

The above message is displayed if the file name entered refers to a program file.

'xxxxxx' is a sequential file - not modifiable

The above message is displayed if the file name entered refers to a SERIAL file.

Modify file 'xxxxxx'
Record index number: _

Enter the record number of the record to update. The record number must not be less than zero, otherwise, the following message is displayed.

Invalid record number. Must be > or = zero

If a record number is entered for a record that does not exist, the utility will assume that a new record is being created. See the following description on 'Creating new records'. If the record did exist, see 'Modifying existing records'.

Modify file 'xxxxxx'
Record access key (nnn chr.): _

Enter the primary access key for the record to update. The key must not be longer than the defined primary key, otherwise, the following message is displayed.

Key size too long

***UFU – File update (continued):**

If a key is entered for a record that does not exist, the utility will assume that a new record is being created. See the following description on 'Creating new records'. If the record did exist, see 'Modifying existing records'.

Creating new records:

If the record identified does not exist the following question is asked.

```
'xxxxxx' record 'nnnnnn' does not exist. Add new record?
Yes No Quit
```

Select:

<i>Yes</i>	to option create a new record.
<i>No</i>	to change the record number.
<i>Quit</i>	to exit the utility.

If YES is selected, the contents of the new record can be added to the file by entering its data one field at a time.

```
'xxxxxx' record 'nnnnnn' does not exist. Add new record?
F1 - Prior field  F2 - End record
```

Each field of the record is entered. The system prompts you for the contents of field 1, 2, and so on until the F2 key is pressed. If you make a mistake and need to reenter a prior field, press the F3 key.

If the record identified does exist the following question is asked.

```
Modify 'xxxxxx', record '
Ignore Change Delete Quit
```

Select:

<i>Ignore</i>	if no changes are to be made to the specified record.
<i>Change</i>	to change a field within the record.
<i>Delete</i>	to remove the specified record from the file. This option is available only for files with a key and is not displayed when updating an Indexed file.
<i>Quit</i>	to exit the utility.

```
Modify 'xxxxxx', record '(record identifier)'
Field #: _
```

Enter the field number to be changed/added. As the field numbers are entered, their contents will be displayed for editing. Press ENTER to physically update the file. Press the F4 key to abandon the changes.

```
'xxxxxx' record length error - record '
```

The above message is displayed if the resultant output record does not fit within the file.

The utility confirms the deletion request if the Delete option is selected.

***UFU – File update (continued):**

Delete record 'xxxxxx' of 'xxxxxx'?
No Yes Quit

Deleting existing records:**Select:**

<i>No</i>	to cancel the delete.
<i>Yes</i>	to confirm the delete.
<i>Quit</i>	to exit the utility.

Record 'xxxxxx' of 'xxxxxx' deleted

The above message is displayed to confirm that the requested record has been deleted from the file.

***UFV**

File View

Description:

This utility allows the contents of any datafile to be viewed.

View file utility
File to view: _

Enter the name of the data file that is to be viewed.

'xxxxxx' is not a viewable file

The above message appears if the file name specified does not refer to a valid viewable data file. Program files and directories are not considered viewable data files.

View multiple fields per line
Yes No Quit

Select:

<i>Yes</i>	to display records with multiple fields on the same line. The vertical bar is used as a field separator.
<i>No</i>	to display each field on a new line.
<i>Quit</i>	to exit the utility.

Viewing file 'xxxxxx'
Enter record number to start display at: _

For Indexed or Serial files, enter the record number of the first record to begin viewing. The record number must be greater than or equal to zero.

Viewing file 'xxxxxx'
Enter starting key: _

***UFV – File view (continued):**

For files with keys, enter the primary key of the record to begin viewing. The size of key specified must not exceed the size of the defined primary key.

The utility displays the data records starting at the point specified. Any non-printable characters will be converted to their Hexadecimal value and displayed within curly brackets {}.

If there is more information than will fit on one screen, the following message is displayed.

```
More data on file.
Enter new record number or <Return> to continue: _
```

Press ENTER to continue the display or enter a new starting record number or primary key.

The following message is displayed when the utility reaches the end of the data file. To continue the display from the start of the file, press ENTER or enter a new starting record number (or key). Press F4 to terminate the utility.

```
End of file reached!
Enter record number to start display at: _
```

UG**General Utility Sub-menu*****Description:**

This program allows the user to select one of the General utilities included with ProvideX.

```
General system utilities
Mortgage Spreadsheet Quit
```

Select:

<i>Mortgage</i>	to compute mortgage payments. (*UGM)
<i>Spreadsheet</i>	to export data to a spreadsheet. (*UGS)
<i>Quit</i>	to exit the utility.

```
Unable to process. Try another selection
```

The above message indicates that an unrecoverable error occurred in a subordinate utility program.

UGM**Mortgage Calculation program*****Description:**

This sample program does mortgage calculations. The starting balance, mortgage rate, and period must be entered. Enter either the desired payment amount or the number of payments.

***UG – General utility sub-menu (continued):**

```

03/27/94 16:59                                     (c) Copyright 1994, Sybex Ltd.
-----
*UGM - Sample Mortgage Calculation Program
-----

Basic annual interest rate .....: -
* Interest compounded how often ..:
* Payments made how often .....:
Mortgage amount .....:
Payment amount .....:
Payments until paid off .....:
First payment date (MM/DD/YY) ..:
Number of payments to print .....:
Report title .....:

-----

* Enter one of the following...

      A -- Annual      S -- Semi-annual    M -- Monthly
      B -- Bi-weekly  H -- Weekly         D -- Daily

```

A payment table may be printed once the mortgage particulars have been entered.

Note: This sample program is designed to assist the programmer to become familiar with some of the simpler aspects of ProvideX.

UGS**Simple Spreadsheet Export*****Description:**

This utility generates an export file for subsequent importing into a spreadsheet. The name of the file and the data fields to be extracted are required.

Each export definition is saved on a file called "PCXPRT.DEF" and must consist of the name of the export definition, the input data file, the output file, and the columns that are to be generated.

***UGS – Simple spreadsheet setup (continued):**

```

PC Export utility -- Data file to Spreadsheet Interface
-----
Name of export definition .....: EXPORT1
Get data from file .....: MK
Put data into PC file .....: FOR123
Column:B
Field name/number .....: ADDR
Offset to start of data .....: 0
Length of data .....: 0
Name of field (opt.) .....:

A: 1(1,5) CUSTID
B: 2(1,20) ADDR

-----
Enter 'W' to proceed, 'C' to change, 'D' to delete: █
F3 - Prior input, F4 - Quit

```

The columns that this program outputs are comprised of the fields specified. Each column must be defined with a field number, an offset within a field, and a field length. A name may be associated with a field.

As each column is defined, it is displayed on the screen as:

X:F(O,L) NAME

Where:

X	is the column letter.
F	is the field number.
O	is the offset.
L	is the length.
NAME	is the optional field name.

The name associated with each field is recorded in the PCXPRT.DEF file. Future references to the specific input file can use the field name rather than field number, offset, and length. In the preceding example, all future references to file MK can refer to fields CUSTID and ADDR by name rather than by field numbers, offsets, and lengths.

Once the desired fields have been defined, press F3. The export of the data can then be run.

NOTE: All data records from the input file are exported.

UP**Program File sub-menu*****Description:**

This program is the Program Utilities sub-menu. It is used to select the function to be performed on program files.

Program utilities:
Make Delete Rename Compare Edit List Bulk-edit Secure Quit

Select:

<i>Make</i>	to create a program file.
<i>Delete</i>	to delete a program file.
<i>Rename</i>	to change the name of an existing program file.
<i>Compare</i>	to compare two programs.
<i>Edit</i>	to edit a program using the Full screen Editor.
<i>List</i>	to get a listing of a program.
<i>Bulk-edit</i>	to perform bulk editing of programs.
<i>Secure</i>	to set general security on programs.
<i>Quit</i>	to exit the utility.

Unable to process. Try another selection

The above message indicates that an unrecoverable error occurred in a subordinate utility program.

UPB**Program Bulk Search & Replace*****Description:**

This utility performs a common string search and/or replacement against a series of programs. It accepts up to five different search strings which can be changed. The search can be done by exact words, or by regular expression, or without regard to case (see MSK function in Language Reference for definition of expressions).

Bulk program search and replace
String to search for: _

Enter the string to be searched for.

String #n:'xxxxxx'
Enter replacement string: _

If the string is to be changed, enter the replacement text, otherwise, press the ENTER key. The system will repeat the preceding two questions for up to five strings or until the ENTER key alone is pressed in response to "String to search for".

***UPB - Program bulk search and replace (continued):**

What type of search is desired?
 Exact_match Ignore_case Word_only Mask Quit

Select:

<i>Exact_match</i>	if the string must match exactly.
<i>Ignore_case</i>	if the string can occur in either upper or lower case.
<i>Word_only</i>	if the string desired must occur as an independent word (variable name, command, etc.).
<i>Mask</i>	if the strings being searched for are MSK() expressions.
<i>Quit</i>	to exit the utility.

Output results to
 Screen Printer Both Quit

Select:

<i>Screen</i>	to show the lines found on the terminal.
<i>Printer</i>	to print the lines found in a report.
<i>Both</i>	to both show and print the lines found.
<i>Quit</i>	to exit the utility.

Where are the programs located?
 Enter starting directory: _

Enter the name of the directory where the programs are located. By default, the current directory is searched.

Sorry, 'xxxxxx' is not a directory

The above error message is displayed if the name entered is not the name of a directory.

Include sub-directories within 'xxxxxx'
 No Yes Quit

Select:

<i>No</i>	if only the directory entered is to be searched.
<i>Yes</i>	if the directory entered and all sub-directories are to be searched.
<i>Quit</i>	to exit the utility.

Which programs in xxxxxx?
 Enter mask/name: _

Enter a pattern match for the desired programs. Press ENTER for all programs.

The utility scans the directory and outputs any lines containing the desired strings.

When a program contains a string being searched (and not replaced), all matching lines are displayed/printed and the following question is asked.

***UPB - Program bulk search and replace (continued):**

Matches in program 'xxxxxx'
Continue Non_stop Quit

Select:

<i>Continue</i>	to continue searching more programs.
<i>Non_stop</i>	to continue searching but not to pause at the end of each program.
<i>Quit</i>	to exit the utility.

When a program containing a string to be replaced is found, all changed lines are displayed/printed and the following question is asked.

Changes in program 'xxxxxx'
Update Skip All Quit

Select:

<i>Update</i>	to change the program.
<i>Skip</i>	to leave the program unchanged.
<i>All</i>	to change this and all subsequent programs.
<i>Quit</i>	to exit the utility

UPC**Program compare*****Description:**

This utility is used to compare two existing programs. The comparison is done on a line by line basis. There is no attempt made to match lines whose line numbers have changed.

Program compare utility
Original program file: _

***UPC - Program compare (continued):**

Enter the original program file name.

'xxxxxx' is not a program file

The above message occurs if no program file with the name specified exists.

Original program file:xxxxxx
Revised program file: _

Enter the name of the program to compare with the original program.

'xxxxxx' is not a program file

The above message occurs if no program file with the second name exists.

***UPC - Program compare (continued):**

Print only the lines, which have been changed
Yes No Quit

Select:

Yes	to print only the lines that are different.
No	to print all lines and flag only those lines that are different.
Quit	to exit the utility.

The output consists of a side by side listing of both programs with a flag noting the differences in the middle of the report. These flags are:

Add	New line added
Chg	Line has been changed
Del	Line has been deleted

UPD**Program file Delete*****Description:**

This utility deletes program files from the system.

Delete program utility
Program to delete: _

Enter the name of the program to be deleted.

'xxxxxx' is not a program

The above message occurs if no program file with the name specified exists.

Program 'xxxxxx' deleted. More deletes?
No Yes Quit

Select:

No	if there are no more programs to delete.
Yes	to enter more programs to delete.
Quit	to exit the utility.

UPL**Program file list*****Description:**

This utility provides a hard copy listing of a program or series of programs.

***UPL - Program file list (continued):**

Program print utility
Enter program name: _

Enter the name (or mask) of the programs to print.

No programs found to match '

The above error message occurs if no programs were found to match the specified input.

Select the type of listing to be produced.

Do you want a formatted print (Takes longer...)
No Yes Quit

Select:

<i>No</i>	to produce an unformatted program listing.
<i>Yes</i>	to produce a formatted program listing.
<i>Quit</i>	to exit the utility.

If desired, select a program cross-reference listing.

Full listing and variable cross reference table?
Both Listing_only Xref_only Quit

Select:

<i>Both</i>	to produce both a cross-reference and program listing.
<i>Listing_only</i>	to produce a program listing only.
<i>Xref_only</i>	to produce a cross-reference listing only.
<i>Quit</i>	to exit the utility.

UPM**Program file Create*****Description:**

This utility is used to create program files.

Make program file utility
Program to make: _

Enter the name of the program file to create.

Program 'xxxxxx' already exists

The above message is displayed if a file with the name entered already exists.

***UPG – Program file create (continued):**

Program 'xxxxxx' has been made. More to make?
No Yes Quit

Select:

<i>No</i>	if there are no more program files to create.
<i>Yes</i>	to create more program files.
<i>Quit</i>	to exit the utility.

UPR**Change Program file name*****Description:**

This utility is used to rename existing program files.

Program rename utility
Program to rename: _

Enter the name of the existing program file that is to be renamed.

'xxxxxx' is not a program

The above message is displayed if no program file of the name given exists.

Renaming program 'xxxxxx'
Enter new name: _

Enter the new name for the program file.

Program 'xxxxxx' now 'xxxxxx'. More renames?
No Yes Quit

Select:

<i>No</i>	if there are no more program files to rename.
<i>Yes</i>	to rename more program files.
<i>Quit</i>	to exit the utility.

UPS**Program Security*****Description:**

This utility is used to set a default password for all program loads and saves. This utility can be used during the development cycle to eliminate the need to constantly enter the PASSWORD for a program after loading and just before saving.

This utility sets the default password (PASSWORD * ".....").

***UPS – Program security (continued):**

ProvideX removes and re-assigns passwords from all programs with the default password. New programs are created with the default password assigned.

Set general program security password Security password (<ENTER> for no password): _
--

Enter the default program password.

Chapter 2

Subprograms

This section of the manual contains a description of the subprograms included with the ProvideX system. These subprograms provide the developer with a comprehensive set of tools to facilitate the development of new applications and simplify the task of upgrading existing ones.

While every effort has been made to ensure that these subprograms function under most conditions, Sybex cannot guarantee that they will function under all conditions. Should any problems with these subprograms be encountered, please contact your local ProvideX distributor.

Note: If the developer wishes to change the functionality provided by these subprograms, it is recommended that the revised versions be maintained and referenced in a separate directory. This will prevent the overwriting of custom alterations with the new versions of the ProvideX standard routines, as they become available.

***DATE**

Date Validation subprogram

Description:

This subprogram is used to validate an input date and to convert it to a standard format. The input procedure is flexible in accepting the date. Both numeric and alphanumeric input is accepted.

Numbers containing 4 digits or exceeding 31 are assumed to be a year. Numbers between 13 and 31 are considered to be a month, and numbers between 1 and 12 are considered to be the day of the month.

When two or more numbers are found which satisfy either month, day or year, the first number is considered to be the month, the second to be the day, and the third to be the year. If only one number between 1 and 31 is entered, it is assumed to be the day.

If no month or year is found, the current month and year is used. If the date is invalid, an error is generated in the subprogram which can be trapped using a ERR= clause in the CALL statement.

Calling sequence:

```
CALL "*DATE", IN_DT$, IN_FM$, OT_JUL$, OT_DAY
```

Where:

<i>IN_DT\$</i>	contains the input date and returns the output.
<i>IN_FM\$</i>	contains the format of the date required where DD is the numeric day, MM is the numeric month, YY or YYYY is the numeric year, and MMM is the alphanumeric month. All other characters are copied unaltered into the date returned.

***DATE - Date validation subprogram (continued):**

<i>OT_JUL\$</i>	returns the date as a value of YYDDD where YY is the last two digits of the year and DDD is the day number within the year (Jan 1 is day 1, etc.).
<i>OT_DAY</i>	returns the day of the week as a value of 1 through 7 with Monday being equal to 1 through Sunday being equal to 7.

***FL.LST**

Generate File List

Description:

This subprogram returns a sorted list of files in a directory. The caller specifies the directory name, mask, and file types to return.

Calling sequence:

CALL "*FL.LST", OT_LST\$, IN_DIR\$, IN_MSK\$, IN_TYPS\$

Where:

<i>OT_LST\$</i>	returns the file names found in the directory. Each entry consists of a 12 character file name followed by a 1 character file type (13 characters per file).
<i>IN_DIR\$</i>	if specified and non-null, contains the name of the directory to be searched. By default, the current directory is searched.
<i>IN_MSK\$</i>	if specified and non-null, must contain the file name mask to compare against. Only files that match this mask will be returned. By default, this is all files.
<i>IN_TYPS\$</i>	if specified and non-null, must contain the types of files to be returned.

The single character file type codes are as follows:

I	- Indexed file
K	- Keyed/Direct/Sort file
L	- Link file
D	- Directory
S	- Serial (or unknown file type)

***FL.MTH**

Compare File Name to Mask

Description:

This subprogram is used to compare a file name to a specified input file name or mask. It handles pattern matches such as *.BAT, ???DAT.IDX.

Calling sequence:

CALL "*FL.MTH", IN_MSK\$, IN_NAME\$

FL.MTH – Compare file name to mask (continued):*Where:**

IN_MSK\$ contains the desired file name or mask to compare.
IN_NAME\$ contains the file name to be compared.

If the name specified (*IN_NAME\$*) does not match the mask (*IN_MSK\$*), this subprogram exits with an error #11. If the name does matches, this subprogram exits normally.

FL.NME**Return name of a work file*****Description:**

This subprogram returns the name of a work file. The file name returned consists of:

Xfffhhmm.WRK

Where:

fff is the value of FID(0).
hhmm is the current time.

If the Environment variable PVXTMP is defined, the name returned is prefixed with its value.

Calling sequence:

CALL "*FL.NME", OT_NAME\$

Where:

OT_NAME\$ returns the path name of a work file to be used.

OPTSEL**Generic Option Selector*****Description:**

This generic option selection utility displays a prompt line and an option selection line. The user can chose an option either by entering the first letter or by shifting through the options via the cursor movement keys (left/right arrow or tab) and then pressing ENTER. The first letter of the selected option is returned to the calling program. The prompt and option lines are cleared and only the selected option is left on the screen.

By default, the two lines are placed on the bottom of the screen. The calling program can override this.

Calling sequence:

CALL "*OPTSEL", OT_ANS\$, IN_MSG\$, IN_OPT\$, IN_LNO

OPTSEL – Generic option selector (continued):*Where:**

OT_ANS\$ returns the selected option. Only the first character of the selected option is returned. A null string is returned if no option was selected and a function key was pressed.

IN_MSG\$ must be initialized with the prompt line to appear above the option selection line.

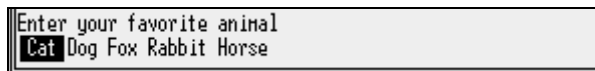
IN_OPT\$ must be initialized with the options to be presented to the user. The options must be separated with a space and the first character of each option must be unique.

IN_LNO must be initialized with the line number in the current window on which to start the prompt and option selection. If omitted, the prompt and option lines appear as the last two lines of the current window.


***OPTSEL Example:** Given the following calling parameters:

```
CALL "*OPTSEL",
      X$,
      "Enter your favorite animal",
      "Cat Dog Fox Rabbit Horse"
```

The screen/window appears as follows:



```
Enter your favorite animal
Cat Dog Fox Rabbit Horse
```

NOTE: This subprogram supports the Mouse. 

PG.CNVConvert program***Description:**

This subprogram is called to convert a ProvideX program file to a serial file or vice-versa. Two parameters are required, the first is the input file name, and the second is the output file name.

Calling sequence:

```
CALL "*PG.CNV", IN_PGM_FRM$, IN_PGM_TO$
```

Where:

IN_PGM_FRM\$ is the name of the input file (either program or serial) to be converted. This file **must** exist.

IN_PGM_TO\$ is the name of the output file (either program or serial) to receive the converted program. If this parameter is null, the utility generates a work file and returns its name in this field. If the file does not exist, a Serial file is produced.

***POPSEL**

Popup Selection Box

Description:


This subprogram displays a pop-up window with a selection. The user can choose one of the selections, which is then returned to the calling program.

Calling sequence:

CALL *"*POPSEL"*, OT_ANS\$, IN_OPT\$, IN_MSG\$

Where:

<i>OT_ANS\$</i>	returns the first character of the selection chosen. A null string is returned if the user did not select any of the options presented, but instead pressed a function key.
<i>IN_OPT\$</i>	must contain all the options to be presented, each separated by a back slash (\). These options are placed on the bottom line of the box with the first letter highlighted.
<i>IN_MSG\$</i>	This parameter string must contain the message line to be displayed in the box.

NOTE: This subprogram supports the Mouse. 

***PR.ABT**

Abort print file

Description:

This subprogram is called to abort the current output for the printer file specified.

Calling sequence:

CALL *"*PR.ABT"*, PR_LFN

Where:

PR_LFN is the logical file number for the printer.

***PR.CLS**

Close print file

Description:

This subprogram is called to close a print file. The print file to be closed must be specified.

Calling sequence:

CALL *"*PR.ABT"*, PR_LFN

Where:

PR_LFN is the logical file number for the printer.

***PR.GET**

Get print device

Description:

This subprogram is called to allocate a printer from the system. Typically, it is called for physical devices, which require the program to wait for access. If the device is unavailable, a message appears on the terminal and the program tries to get the device later.

Calling sequence:

CALL **"*PR.GET"**, PRTR\$, LN

Where:

PRTR\$	contains the name of the printer to be opened.
LN	contains the line number on the screen on which the wait message is displayed.

***PR.OPN**

Open print file

Description:

This subprogram is used to obtain and open a print file. It asks the user for the name of the print file to open. By default, it opens the printer (or file) defined in the global system variable %Z_PRTR\$.

Calling sequence:

CALL **"*PR.OPN"**, PR.LFN, LN

Where:

PR.LFN	contains the file number to be used when opening the printer. If zero, this subprogram dynamically assigns a new file number (via HFN) and returns it in this argument.
LN	contains the line number to use on the screen when asking for the device/file name. If this parameter is not specified, the question appears at the bottom of the screen.

***PR.SEL**

Select print file

Description:

This subprogram selects the printer to receive a print file.

Calling sequence:

CALL **"*PR.SEL"**, PRTR\$, LN

Where:

PRTR\$	returns the name of the printer selected.
LN	is the line number to display/ask for the printer selection.

SCR.RSTRestore current window***Description:**

This subprogram is used to restore the screen saved by a prior call to *SCR.SVE.

The current window only is restored. If the window size has changed, the original saved data is placed in the upper left corner of the current window and truncated to fit. Any unused area is cleared.

CALL **"*SCR.RST", SV_SCRN\$**

Where:

SV_SCRN\$ is the value returned from *SCR.SVE which contains the screen image information.

SCR.SVESave window image***Description:**

This subprogram saves the current screen image. It is called with a variable, which receives the window contents. The contents of the current window only are preserved. To restore the window, the subprogram *SCR.RST is called with the same variable.

Calling sequence:

CALL **"*SCR.SVE", SV_SCRN\$**

Where:

SV_SCRN\$ is the variable to receive the screen image information.

SELECTItem selection utility***Description:**

This subprogram displays a scrolling list of items within a window on the screen. An item can be selected by first highlighting it with the Up/Down arrow keys and then pressing ENTER. The selected entry's index is returned to the calling program.

Calling sequence:

CALL **"*SELECT", SEL_IDX, SEL_TBL\${ALL}, SEL_TTL\$, SEL_WIDE, WDW_CLR\$**

Where:


SEL_IDX returns the selected index. On input, it can be used to initialize the display.

SEL_TBL\${ALL} is a string table that contains the options to be presented.

SEL_TTL\$ contains the title/prompt that will appear in the selection window.

***SELECT – Item selection utility (continued):**

<i>SEL_WIDE</i>	optional parameter containing the width of the window to be created. If not provided (or zero), the window will be 6 columns wider than the largest entry in table up to a maximum of 70 columns.
<i>WDW_CLR\$</i>	optional parameter containing a string of mnemonics to be used to define the colour of the window. If not provided, the current colours will be used.

NOTE: This subprogram supports the Mouse. 

VLDATE**Input Validation routine***

Description:

This subprogram is used to validate an input string against a series of validation rules. If the validation fails, an error message is returned. The rules provided to this routine must contain a series of values or ranges separated by commas.

Example:

1, 4-5, A-X.

Calling sequence:

CALL *VLDATE*, *INP_VAL\$*, *VLD_TBL\$*, *ERR_MSG\$***

Where:

<i>INP_VAL\$</i>	is the input value to validate.
<i>VLD_TBL\$</i>	is a string containing the validation rules.
<i>ERR_MSG\$</i>	returns the error message if the validation fails or a null string, if the validation is successful.

Chapter 3

System Programs/Files

This section of the manual contains a description of the programs, which are used by ProvideX to handle Hot-keys and built-in Help and Query facilities. Normally, application programs should not call these routines. They have been included in this manual to provide the application designer with a better understanding of how the system internals operate.

The following files have a two or three character language code as part of the file name. This language code is taken from the system environment variable LANG. If undefined, the value of EN (for English) will be used.

*LEXTBL.EN
*LEXDEF.EN
*MLFILE.EN
*PRMDEF.EN

ACTIVATE.PVX

System activation information

Description:

This file contains the internal activation information for your version of ProvideX. The information includes:

- System name
- Software serial number
- System identification number
- Maximum simultaneous user count
- Expiry date
- Package activation information

   **WARNING**   

This file should not be modified or moved under any conditions. Any attempt to modify or move this file may result in the partial or total deactivation of your software!! Reactivation of your software caused by tampering with this file is a billable service.

***CONTROL**

Hot-Key intercept program

Description:

This is the Hot-Key intercept utility program. It is invoked internally by the ProvideX executive whenever a terminal input encounters a CTL value between -1 and -999. It processes the following:

CTL value	Processing
-1	Calls the system utilities main menu *U
-2	Prints the contents of the current screen
-3	Line-switch (on some systems)
-4	Display current statistics
-5	Input field help (Calls *HELP)
-6	Input field query (Calls *QUERY)
-7	Program help (Calls *HELP.PRG)
-8	Resets screen
-9	<i>(Reserved for future usage)</i>
-10 thru -999	Calls the program \$CTL- <i>nnn</i> where ' <i>nnn</i> ' is the CTL value. Before calling this program, a new window consisting of the entire screen is created. It is dropped when the program returns.

Once *CONTROL has completed processing the CTL function, ProvideX will re-execute the INPUT (or OBTAIN) directive that initiated the process.

If the program reads raw data directly from the keyboard using the mnemonics 'BI' or 'ME', or the directives READ RECORD or ACCEPT, the program must call *CONTROL to handle all CTL values between -1 and -999.

Example:

Read single character but detect Hot-Key call to system functions

```
0100 OBTAIN (0,SIZ=100) 'ME',X$, 'MN'
0110 IF CTL < 0 AND CTL > -1000 THEN
0110: CALL "*CONTROL"; GOTO 0100
```

***DEV**

Device Drivers

Description:

ProvideX supports a wide variety of device drivers. Each device driver has its own ProvideX basic program which resides in the *DEV directory. Each device driver is responsible for defining the terminal characteristics, mnemonics, and CTL definitions.

The device driver for your terminal is loaded and executed during system initialization. Device drivers are also executed whenever a ProvideX device link file is opened.

***DEV – Device drivers (continued):**

Device drivers are standard ProvideX programs and have access to the full complement of ProvideX functions and directives. See the ProvideX Users Guide for more details on writing device drivers.

***ERROR**

Generic Error Handler

Description:

This program is intended to serve primarily as a sample error handling program.

This utility displays the message associated with the error condition and the current program stack. The user may abort the program by entering 'A', retry the error by entering 'R', or restart the session by entering 'S'.

If the system variable %Z_PASSWORD\$ is defined, the 'A' and 'S' options both require the entry of the value stored in this variable as a password.

Note: Normally, each application designer will develop his/her own error handler.

***HELP**

On-line help display

Description:

This is the main program responsible for on-line help. It is invoked by "*CONTROL" to retrieve and to display the on-line help text and to allow it to be maintained.

It retrieves any help information for the current program based on either the current screen location or the value in the system variable 'HLP'. This information is obtained from either the user help file "HELP.xx" or "*HELP.xx" for the system utilities ('xx' is the current language code). Any help information found is displayed, otherwise, a message indicating that there is no help is displayed.

The systems designer can enter or change the help text by entering the help password (password on the program *HELP.PWD). The help text and any related Query definition can be changed if the password is entered, followed by the ENTER key.

Common help messages may be defined by entering the first line of help text in the form =XXXXXX, where XXXXXX is the name given to common help text to use.

HELP.xxSystem utility help information*

Description:

This keyed file contains the help text for the system utilities. It is referenced by the system utility *HELP and *HELP.PRG in response to a help request within a system utility.

HELP.PRGOn-line Program help display*

Description:

This program is responsible for the display and maintenance of the on-line Program help screens. It is invoked by "*CONTROL" to retrieve and display the on-line help text and to allow it to be maintained.

It opens and retrieves the help information for the current program from either the users help file "HELP.xx" or "*HELP.xx" for system utilities ('xx' is the current language code). Any help information found is displayed, otherwise, a message indicating that there is no help is displayed.

The systems designer can enter or change the help text by entering the help password (password on the program *HELP.PWD). The system editor (as defined by environment value PVXEDIT) will be invoked to allow the editing of the program help if the password is entered.

HELP.PWDOn-line Program help password*

Description:

This utility program's sole purpose is to verify the Help subsystem password. The program accepts the password for the help subsystem as its only parameter. It validates the password and returns normally if successful. Failure of the password causes an error exit.

This subprogram functions by attempting to de-password itself with the password given. By default, the password for this program is:

SYBEX

KYBRD.CFG/*KYBRD.STD**Keyboard definition files***

Description:

These two keyed files contain the keyboard command sequences and their respective CTL values. The system utility *TTY uses the information contained in these files to load the CTL key definitions for terminals.

The *KYBRD.STD file contains the standard keyboard definitions as provided by Sybex. The *KYBRD.CFG file contains any custom definitions that have been defined by the users. The system utility *UCK creates and updates the *KYBRD.CFG file. These files contain records with 80 elements where each element has the input sequences for various CTL functions. The CTL values for each of the 80 elements is maintained in the *KYBRD.STD file along with the keyword description of the function it represents.

When *TTY is looking for a keyboard definition, it first looks in the *KYBRD.CFG file. If no entry is found, the *KYBRD.STD file is used.

LEXTBL.xx/*LEXDEF.xx**ProvideX LEX definition tables******(xx indicates language)***

Description:

The *LEXTBL file contains an internal format image of the syntax tables used by the ProvideX compiler and lister components. The *LEXDEF file contains the same information but in a conventional keyed file format to be used by the utility *LEXEDIT. The contents of these files have the textual display formats for all internal ProvideX object code elements.

Whenever ProvideX compiles a statement, it converts it to an internal code. This internal code consists of a one or two character code, which represents a directive, function, or system variable. It is these codes that are saved to program files. The codes are translated back to viewable text when a statement is listed.

Altering the syntax tables within ProvideX can change the translation process. This functionality allows for the definition of a completely different set of directives, functions and variables as far as the entry and listing of the program is concerned.

One of the primary uses for this functionality is to allow for multi-lingual systems. By defining directives in other languages, users in the field who may be responsible for interfacing with support personnel can be provided an environment that allows them to read the program in their native language.

A secondary use of this functionality is to assist in the conversion of systems to ProvideX.

***MLFILE.xx**

Message Library

Description:

This file contains all the textual elements of the ProvideX executive. All error messages are maintained within this file as 80 byte records indexed by a message number.

The system date function (DTE) maintains the literals containing the names of the months and days in the records starting at index 129 through index 135. These messages in English are:

129 : January,February,March,April,May,June

130 : July,August,September,October,November,December

131 : Monday,Tuesday,Wednesday,Thursday,Friday,Saturday,Sunday

132 : am,pm,AM,PM

133 : Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec

134 : Mon,Tue,Wed,Thu,Fri,Sat,Sun

135 : am,pm,AM,PM

In addition, the upper case character definition table is maintained in the first 64 bytes of the four records starting at index 140. The lower case table is maintained starting at index 144.

***PRMDEF.xx**

Parameter definitions

Description:

This file is used by the system utility program *UCP to maintain the descriptions and the valid ranges for the system parameters. This keyed file has one record for each system parameter keyed on the parameter code.

The record layout is:

Field #	Description
1	Short description of parameter (16 char max)
2	Minimum value
3	Maximum value. If the maximum value equals the minimum value, the parameter simply toggles OFF and ON.
4	Default value.
5	Reset indicator. A value of "Y" in this field indicates that changing this parameter may effect other parameters. Therefore, *UCP must refresh the screen whenever a change occurs.
6	Help text

***QUERY**

On-line Query processor

Description:


This utility is the 'Standard' query processor. It uses the information created by '*QUERY.DEF' to create and display a query.

When a query is invoked, a window is displayed in the center of the screen with the first 10 records of the specified query file. The Up/Down arrow, Pgup/Pgdn, Home or End keys allow for movement within the file. Optionally, full or partial keys may be entered to position directly within the file.

If the user selects one of the entries displayed, its associated values (defined by the query definition) are placed into the input buffer to be processed by the mainline program.

This routine is Mouse sensitive and provides scroll bars for direct positional manipulation.

See *QUERY.DEF following for details on query definition.

NOTE: This subprogram supports the Mouse. 

***QUERY.DEF**

Query definition/maintenance program

Description:

This utility program is called by the help subsystem to create or modify 'Standard' query definitions. Standard queries present a series of items from the records of a specified file. The definition utility allows the programmer to define the file name, fields to display, and their format.

The query definition must include the following:

Query title:	The title line for the query window
Query file:	The name of the keyed file to be presented.
Query key no:	The key number of the keyed file to be used. This will determine the order of the records to be displayed.
 Query flds:	 There can be up to 9 fields displayed within a query. Each field must define the field within a record of the file to display, the offset within the field, the length of the field, a format specifier, a return indicator, and title. Valid format specifiers are:
	0-9 Number of decimal points
	D Date format 99/99/99
	\$ Currency display \$#,##0.00

All fields with the return indicator set to "Y" are returned to the program which invoked the Query. Each field is placed into the input queue, one field at a time.

***START.UP**

SYSTEM START_UP

Description:

This is the general system startup program and is always executed during ProvideX initialization or after a START directive. It displays the main system start up screen that contains the ProvideX copyright and serial number messages. The program then calls the users "START_UP" program if it exists. The user "START_UP" program is where default system parameters would be altered or global variables would be initialized.

<p>NOTE: Since this program and any user START_UP program are CALLED during system initialization, they should not attempt to initiate any application programs. If you attempt to have your START_UP program run your application, any error in your application may result in a exit back to ProvideX command mode with no current program.</p>

***TTY**

Load Keyboard Definitions

Description:

This program is used by most terminal device drivers to load the keyboard control sequences for the terminal. It uses the tables defined by "UCK" to configure the device which is identified by the system variable LFO (Last File Opened).

Whenever a terminal device driver is executed, it should transfer control to this program to load the CTL definitions from the *KYBRD.CFG or *KYBRD.STD files.

Chapter 4

Windows Development Kit

In order to provide a transition to a Windows and Object Oriented development environment, a variety of subprograms and special Mnemonics are provided. These subprograms can be used to display and control various Windows compatible display/input 'objects'. The current routines run in standard Text mode and will be converted to full windows routines as they become available in the future with no changes to the programs written today.

Overview of Subprograms and Objects:

In general, each **object** is defined by a call to `***xxxx.def`, where `xxxx` indicates the object type. A series of definition parameters such as the line, column, and size of the object are provided with the call. Typically, the first parameter is a string where the routine will store its own internal control information. The second parameter is the CTL value (or EVENT identifier) to be returned on an INPUT statement whenever the specified object is selected by the mouse.

To process an '**object event**', the subprogram `***xxxx.xeq` is CALLED with the control structure and the return value fields.

To remove an object, the subprogram `***xxxx.del` is CALLED. Please note that the objects are always destroyed whenever the current window is DROPEd.

For objects which require updating by the program, the subprogram `***xxxx.dsp` is provided.

The types of objects that are supported are:

BUTTON	A button is used to signal a event. Typical uses include OK or CANCEL buttons on a screen to signal that the user has completed input or wishes to cancel the process. The programmer can activate a button by calling the event processing subprogram.
CTLBTN	A control button is like a standard button but it only returns a CTL value when selected and cannot be activated directly by the program.
CHKBOX	A check box is an ON/OFF indicator. The state of a check box toggles from On to Off (or vice-versa) when selected by the user.
CHKLST	A checklist is a related series of ON/OFF indicators, only one of which may be active at any time. When the user selects one of the indicators, it is turned ON, all other indicators in the check list are turned Off. This is similar to the push buttons on a radio.

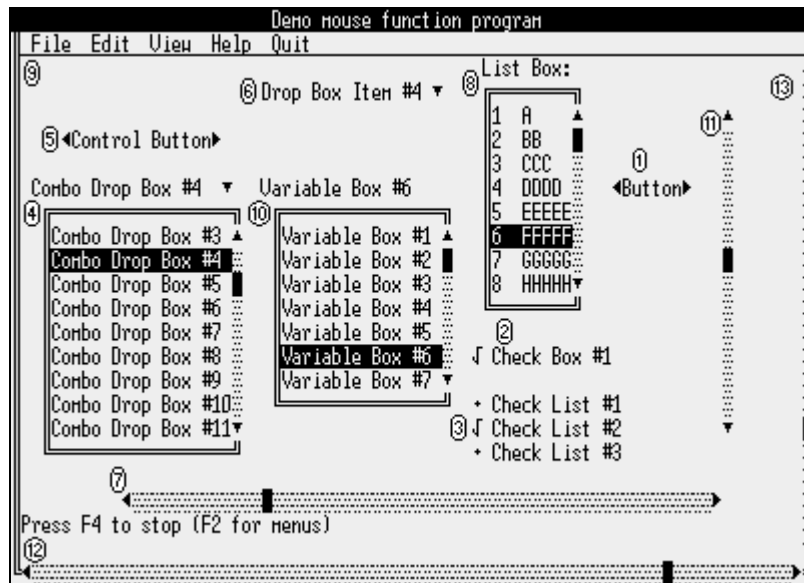
Overview of Sub-programs and Objects (Continued):

MENU	A menu is displayed on the top line of the current window. A string that contains the main menu line contents, all submenu items, and a series of control flags defines a menu. Each entry in a menu must have a single character identified as the 'Access' character. Normally this is the first character of the menu item, such as 'F' for File, 'E' for Edit, etc.. When a menu is selected, the program is passed the series of letters that comprise the selected option such as "FS" for 'File' and then 'Save'.
HSCRBR	A horizontal scroll bar is displayed as a line proceeding from left to right upon which there is a visual representation of a slider. It returns a numeric value between 1 and a program defined maximum. The user may control the value of the scroll bar by selecting the slider and moving it left (lower value) or right (higher value). On the left and right sides of the bar are arrows which move the slider one position left (value down by one) or right (value up by one) when selected. If the user selects the line to the right of the slider, the slider will increase by a 'Big jump'. Usually, the 'Big jump' is defined by the program as one page of information. If the user selects the line to the left of the slider, the slider will decrease by a 'Big jump'.
VSCRBR	A vertical scroll bar is displayed as a line proceeding from top to bottom upon which there is a visual representation of a slider. The user may control the value of the vertical scroll bar by selecting the slider and moving it up (lower value) or down (higher value). At the top and bottom of the bar are arrows that move the slider one position up or down when selected. If the user selects the line above the slider, the slider will move up in a 'Big jump'. If the user selects the line below the slider, the slider will move down a 'Big jump'.
WHSCRL	A window horizontal scroll bar consists of horizontal scroll bar which occupies the bottom line of a frame around a window.
WVSCRL	A window vertical scroll bar consists of vertical scroll bar which occupies the right edge of a frame around a window.
LSTBOX	A list box presents a series of options from which the user may select one. The list is presented within a box on the screen. A scroll bar is placed on the right side of the box if more entries exist than there are lines in the box.
VARBOX	A variable list box is similar to a list box except that the user can enter a selection not contained within the list. The user either types in the desired value or uses the Up/Down arrow keys to select an entry from the list box which appears below the entry line.
DRPBOX	A drop box consists of a single line on the screen below which a list box will be displayed. When the user selects the drop box, a list box containing the valid values for the entry will appear. After the user selects the desired entry from the list box, it will disappear and the single line will reflect the chosen entry.

Overview of Sub-programs and Objects (Continued):

COMBOX A combo drop box consists of a single line on the screen below which a list box will be displayed. The user may directly enter a value into a combo box or press the Down arrow key to have a list box appear. A value from the list box then may be selected or the user can continue to enter a manual selection.

Included in the ProvideX EXAMPLES directory is the demonstration program "DEMO". It illustrates the use and functionality of some of the various Windows-like subprograms as follows:



Item #	Routine Name	Description
(1)	**button	Push Button
(2)	**chkbox	Check Box
(3)	**chklist	Check List
(4)	**combox	Combo Drop Box
(5)	**ctlbtn	Control Button
(6)	**drpbox	Drop Box
(7)	**hscrbr	Horizontal Scroll Bar
(8)	**lstbox	List Box
(9)	**menu	Pull Down Menus
(10)	**varbox	Variable List Box
(11)	**vscrbr	Vertical Scroll Bar
(12)	**whscr1	Horizontal Window Scroll Bar
(13)	**wvscr1	Vertical Window Scroll Bar

Overview of Sub-programs and Objects (Continued):

The following special mnemonics have been defined for use in the formatting of scroll bars, menu markings, etc. These mnemonics are used via the MNM function to insert a single character of output and are not used independently as standard mnemonics.

The MS-DOS definitions have been provided and dummy definitions for other terminal types are defined in *TTY.

Mnemonic	Function
'!'	Character to output for scroll bars
'!X'	Character to output for current position in bar
'!>'	Right arrow character
'!<'	Left arrow character
'!^'	Up arrow character
'!V'	Down arrow character
'!*'	Menu tick mark indicator
'!M'	Sub-menu indicator

NOTE: If '!X' is not defined, then scroll bars are not supported. The routines simply return without effecting the display.

In and Out Flags:

Most of the Windows subprograms have input and output flags defined. Presently they are not used in many of the subprograms, but have been reserved for future use. These flags are designed to hold a series of options. All options within a flag are separated by commas. Append a comma followed by the option when adding new options to a flag. Use the POS() function to check for the presence of an option within a flag.

Passing Tables to Window functions:

All tables passed to functions, which require a list of elements must have a base of one, **not zero**.

****BUTTON**

Push buttons

Description:

This function provides the ability to define Push buttons, which return CTL events when, activated by the mouse.

Define Calling sequence:

CALL "*button.def", b_def\$, btn_text\$, ctl_val, col, line, in_flg\$, out_flg\$**

Where:

b_def\$ is the Button control structure variable.
btn_text\$ is the text to display in the button.

****BUTTON – Push button (continued):**

<i>ctl_val</i>	contains the CTL event to occur when the mouse selects the button region.
<i>col</i>	is the column for the button.
<i>line</i>	is the line for the button.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

Invocation Calling sequence:

CALL "*button.xeq", b_def\$, in_flg\$, out_flg\$**

Where:

<i>b_def\$</i>	is the Button control structure variable.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output options. This returns one of the following flags which indicates how the button was selected:
MSE	the Mouse was pressed
RET	the Enter key was used
SPC	the Space bar was pressed

To delete a button:

CALL "*button.del", b_def\$, in_flg\$, out_flg\$**

Where:

<i>b_def\$</i>	is the Button control structure variable.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

****CHKBOX***Check Boxes***Description:**

A check box is a ON/OFF indicator. The state of a check box will toggle from On to Off (or vice-versa) when selected by the user. The state indicator for a check box contains a "Y" if the check box is ON, otherwise, the indicator contains a NULL string ("").

Define Calling sequence:

CALL "*checkbox.def", chk_def\$, chk_text\$, chk_val\$, ctl_val, col, line, in_flg\$, out_flg\$**

Where:

<i>chk_def\$</i>	is the Check Box control structure variable.
<i>chk_text\$</i>	contains the text to display in the check box.
<i>chk_val\$</i>	is the current check box state.
<i>ctl_val</i>	contains the CTL event to occur when mouse selects the check box region.
<i>col</i>	contains the column for the check box.

****CHKBOX – Check boxes (continued):**

<i>line</i>	contains the line for the check box.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

Invocation Calling sequence:

CALL "*chkbox.xeq", chk_def\$, chk_val\$, in_flg\$, out_flg\$**

Where:

<i>chk_def\$</i>	is the Check box control structure variable.
<i>chk_val\$</i>	contains the check box state.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

To delete a check box:

CALL "*chkbox.del", chk_def\$, in_flg\$, out_flg\$**

Where:

<i>chk_def\$</i>	is the Check box control structure variable.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

****CHKLST**

Check List

Description:

Check Lists have multiple choices with only one being selected at any given time. A check mark is displayed beside the selected item.

Define Calling sequence:

CALL "*chklist.def", chkl_def\$, ctl_val, chkl_idx, chk_lst\${ALL}, col, line, width, height, in_flg\$, out_flg\$**

Where:

<i>chkl_def\$</i>	is the Check List control structure variable.
<i>ctl_val</i>	contains the CTL event to occur when mouse selects the check list region.
<i>chkl_idx</i>	contains the index of the current item selected (default is 1).
<i>chk_lst\${ALL}</i>	contains the list of items to be displayed.
<i>col</i>	contains the column for the check list.
<i>line</i>	contains the line for the check list.
<i>width</i>	contains the column width of the check list display.
<i>height</i>	contains the number of check list items.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

****CHKLST – Check list (continued):****Invocation Calling sequence:**

CALL "*chklst.xeq", chk_l_def\$, chk_l_idx, chk_lst\${ALL}, in_flg\$, out_flg\$**

Where:

chk_l_def\$ is the Check list control structure.
chk_l_idx contains and returns the index of the selected item.
chk_lst\${ALL} contains the list of items to choose from.
in_flg\$ contains input flags (Future).
out_flg\$ receives output flags (Future).

CALL "*chklst.del", chk_l_def\$, in_flg\$, out_flg\$**

Where:

chk_l_def\$ is the Check list control structure.
in_flg\$ contains input flags (Future).
out_flg\$ receives output flags (Future).

****COMBOX***Combo boxes*

A combo box is similar to a drop box except that it allows for the entry of values not contained with the pre-specified list of items. Only a single line is occupied on the screen until the combo box is activated. At this point, a 'List box' drops down from the input containing the possible entries. The user may select one of the entries in the 'List box' or enter data directly.

To define a Combo box:

CALL "*combox.def", cbox_def\$, cbox_val\$, ctl_val, cbox_tbl\${ALL}, col, line, width, height, in_flg\$, out_flg\$**

Where:

cbox_def\$ is the variable to receive the control structure for the Combo box.
cbox_val\$ contains the initial value to display.
ctl_val contains the CTL event to occur when the mouse selects the box.
cbox_tbl\${ALL} contains the table of values for the box.
col contains the column to start the combo box.
line contains the line to start the combo box.
width contains the width of the box. If omitted (or <3), this is set to the length of the largest item in the list plus two for the border up to a maximum of 40 (or the edge of the window).
height contains the height of the box. If omitted (or <3), this is set to the number of entries in the list plus two up to a maximum of 12 (or to the bottom of window).
in_flg\$ contains input flags (Future).
out_flg\$ receives output flags (Future).

****COMBOX – Combo boxes (continued):**

To process a Combo Box event:

CALL "*combox.xeq", cbox_def\$, cbox_val\$, cbox_tbi\${ALL}, in_flg\$, out_flg\$**

Where:

<i>cbox_def\$</i>	is the Combo Box control structure.
<i>cbox_val\$</i>	returns the value selected from the drop box or manually entered.
<i>cbox_tbi\${ALL}</i>	contains the table of values for the box.
<i>in_flg\$</i>	contains input flags. The flag '+FOCUS' forces execution of this routine. This is useful for systems which do not employ a Mouse.
<i>out_flg\$</i>	receives output flags (Future).

To delete a Combo Box:

CALL "*combox.del", cbox_def\$, in_flg\$, out_flg\$**

Where:

<i>cbox_def\$</i>	is the Combo Box control structure.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

****CTLBTN**

Control buttons

Description:

This function provides the ability to define Control buttons which returns CTL events when activated by the mouse.

Control Buttons can be accessed only with a Mouse. If you are writing software for systems which may not support a mouse, you might consider using the **BUTTON routines. Otherwise, you will have to define a keyboard control sequence and use the **ctlbtn.psh & **ctlbtn.rel routines.

Define Calling sequence:

CALL "*ctlbtn.def", ctl_val, col, line, btn_text\$, b_attr\$, in_flg\$, out_flg\$**

Where:

<i>ctl_val</i>	contains the CTL event to occur when the mouse selects the button region.
<i>col</i>	contains the column for the button.
<i>line</i>	contains the line for the button.
<i>btn_text\$</i>	contains the text to display in the button.
<i>b_attr\$</i>	contains the Screen attributes for the button.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

****CTLBUT – Control button (continued):****Invocation Calling sequence:**

CALL "*ctlbtn.xeq", col, line, btn_text\$, ctl_val, b_attr\$, in_flg\$, out_flg\$**

Where:

<i>col</i>	contains the column for the button.
<i>line</i>	contains the line for the button.
<i>btn_text\$</i>	contains the text to display in the button.
<i>ctl_val</i>	contains CTL value of the button region.
<i>b_attr\$</i>	contains the Screen attributes for the button.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

To delete a button:

CALL "*ctlbtn.del", col, line, btn_text\$, in_flg\$, out_flg\$**

Where:

<i>col</i>	contains the column for the button.
<i>line</i>	contains the line for the button.
<i>btn_text\$</i>	contains the text displayed in the button.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

To highlight a button:

CALL "*ctlbtn.psh", col, line, btn_text\$, b_attr\$, in_flg\$, out_flg\$**

Where:

<i>col</i>	contains the column for the button.
<i>line</i>	contains the line for the button.
<i>btn_text\$</i>	contains the text displayed in the button.
<i>b_attr\$</i>	contains the screen attributes for the button.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

To turn a button highlight off:

CALL "*ctlbtn.rel", col, line, btn_text\$, b_attr\$, in_flg\$, out_flg\$**

Where:

<i>col</i>	contains the column for the button.
<i>line</i>	contains the line for the button.
<i>btn_text\$</i>	contains the text displayed in the button.
<i>b_attr\$</i>	contains the screen attributes for the button.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

****DRPBOX**

Drop boxes

Description:

A drop box is similar to a list box except that a single line only is occupied on the screen until the drop box is activated. At this point, a 'List box' drops down from the input containing the possible entries. When one of the entries is selected, the 'List box' disappears and the value selected appears on the single line.

Drop box definition:

CALL **drpbox.def*, *dbox_def*\$, *dbox_val*\$, *ctl_val*, *dbox_tbl*{*ALL*}, *col*, *line*, *width*, *height*, *in_flg*\$, *out_flg*\$**

Where:

<i>dbox_def</i> \$	is the variable to receive the control structure for the drop box.
<i>dbox_val</i> \$	contains the value to display in the input line for the drop box. If the value does not exist in the table, the first item from the table is displayed.
<i>ctl_val</i>	contains the CTL event to occur when mouse enters input field.
<i>dbox_tbl</i> { <i>ALL</i> }	contains the table of values to use.
<i>col</i>	contains the column to start the drop box.
<i>line</i>	contains the line to start the drop box.
<i>width</i>	contains the width of the box. If omitted (or <3), this is set to the length of the largest item in the list plus two for the border up to a maximum of 40 (or the edge of the window).
<i>height</i>	contains the height of the box. If omitted (or <3), this is set to the number of entries plus two in the list up to a maximum of 12 (or to the bottom of window).
<i>in_flg</i> \$	contains input flags (Future).
<i>out_flg</i> \$	receives output flags (Future).

Drop box event processing:

CALL **drpbox.xeq*", *dbox_def*\$, *dbox_val*\$, *dbox_tbl*{*ALL*}, *in_flg*\$, *out_flg*\$**

Where:

<i>dbox_def</i> \$	is the drop box control structure.
<i>dbox_val</i> \$	returns the value selected.
<i>dbox_tbl</i> { <i>ALL</i> }	contains the Table of values to use.
<i>in_flg</i> \$	contains input flags (Future).
<i>out_flg</i> \$	receives output flags (Future).

To delete a Drop box:

CALL **drpbox.del*", *dbox_def*\$, *in_flg*\$, *out_flg*\$**

Where:

<i>dbox_def</i> \$	is the drop box control structure.
<i>in_flg</i> \$	contains input flags (Future).
<i>out_flg</i> \$	receives output flags (Future).

****ERROR.BOX***Error Box*

Description:

This routine is used to display an Error or Warning message. It simply places the message in a window on the screen and prompts the user to press enter.

Define Calling sequence:

CALL "*error.box", err_msg\$, in_flg\$, out_flg\$**

Where:

<i>err_msg\$</i>	contains the message to display in the window. Multiple lines of description may be used by inserting the SEP variable at the end of each line.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

****HMOVE.CHK***Horizontal movement processor*

Description:

This is an internal routine used by the list box functions to intercept keyboard movement. It is used to process cursor movement keys →, ←, TAB, HOME, END, Scroll left, and Scroll Right.

Define Calling sequence:

CALL "*hmove.chk", hb_new, hb_max, hb_bjmp, in_flg\$, out_flg\$**

Where:

<i>hb_new</i>	on input has the current value and returns the new value calculated.
<i>hb_max</i>	contains the largest value in the list.
<i>hb_bjmp</i>	contains the amount to adjust the new value by when detecting scroll left/right.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

****HSCRBR**

Horizontal Scroll Bar

A horizontal scroll bar is used when there is more information to display than fits on one screen. It shows the relative position of the information currently displayed. The scroll bar typically appears within a window region. When the mouse is activated within the scroll bar region, a CTL value EVENT is initiated. The program must call ****HSCRBR.XEQ** to handle this event and subsequently ****HSCRBR.DSP** to update the scroll bar display. ****HSCRBR.DSP** must be CALLED to update the display whenever the location on the scroll bar changes via other program functions (timer, file position, initial setting...).

These scroll bar functions should not be confused with the Window region horizontal scroll bar, which appears on the bottom of the current window.

Defining scroll bar:

CALL **hscrbr.def*, hscr_def\$, ctl_val,\$ col, line, width, height, in_flg\$, out_flg\$**

Where:

<i>hscr_def\$</i>	is the variable to receive the control structure for the scroll bar.
<i>ctl_val</i>	contains the CTL event to occur when mouse enters the scroll bar.
<i>col</i>	contains the column to start the scroll bar.
<i>line</i>	contains the line to start the scroll bar.
<i>width</i>	contains the scroll bar width (default 1).
<i>height</i>	contains the scroll bar height (default is to bottom of current window).
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

Updating scroll bar:

CALL **hscrbr.dsp*, hscr_def\$, new_val, max_val, big_jump, sml_jump, in_flg\$, out_flg\$**

Where:

<i>hscr_def\$</i>	is the scroll bar control structure
<i>new_val</i>	contains the new value within scroll bar (minimum value = 1)
<i>max_val</i>	contains the end value within scroll bar
<i>big_jump</i>	contains the amount to increase/decrease for BIG Jump (mouse pressed left/right of current position). If < 1 or not specified, this value defaults to max_val divided by # of entries in bar.
<i>sml_jump</i>	contains the amount to increase/decrease for SMALL Jump (arrows at either end of bar). If not specified or < 1, a value of 1 is used.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

Processing Scroll bar event:

CALL **hscrbr.xeq*, hscr_def\$, new_val, max_val, big_jump, sml_jump, in_flg\$, out_flg\$**

****HSCRBR – Horizontal scroll bar (continued):****Where:**

<i>hscr_def\$</i>	is the scroll bar control structure
<i>new_val</i>	returns the value corresponding to the selected position within the scroll bar. The scroll bar must be updated by calling **HSCRBR.DSP after the value is returned and processed. (minimum value = 1).
<i>max_val</i>	contains the maximum value within scroll bar.
<i>big_jmp</i>	contains the amount to increase/decrease for BIG Jump (mouse pressed left/right of current position). If < 1 or not specified, this value defaults to <i>max_val</i> divided by # of entries in bar.
<i>sml_jmp</i>	contains the amount to increase/decrease for SMALL Jump (arrows at either end of bar). If not specified or < 1, a value of 1 is used.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

Deleting scroll bar:

CALL **hscrbr.del**, *hscr_def\$*, *in_flg\$*, *out_flg\$***

Where:

<i>hscr_def\$</i>	is the scroll bar control structure
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

****INPBOX***Input Box*

An input box allows the user to click the mouse in the middle of a text field and have an INPUT EDIT start at that position.

To define an Input box:

CALL **inibox.def**, *ibox_def\$*, *ibox_val\$*, *ctl_val*, *col*, *line*, *width*, *height*, *in_flg\$*, *out_flg\$***

Where:

<i>ibox_def\$</i>	is the variable to receive the Input box control structure.
<i>ibox_val\$</i>	contains the initial value.
<i>ctl_val</i>	contains the CTL event to occur when mouse enters box.
<i>col</i>	contains the column to start the combo box.
<i>line</i>	contains the line to start the combo box.
<i>width</i>	contains the width of the value.
<i>height</i>	contains the height of the region.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

To process an Input Box event:

CALL **inibox.xeq**, *ibox_def\$*, *ibox_val\$*, *in_flg\$*, *out_flg\$***

****INPBOX – Input box (continued):****Where:**

<i>ibox_def\$</i>	is the Input box control structure.
<i>ibox_val\$</i>	return the value input.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	contains output flags. If the user enters text from the keyboard and presses ENTER, a flag of 'RET' is returned.

To update the display of an Input Box:

CALL "*inpbox.dsp", *ibox_def\$*, *ibox_val\$*, *in_flg\$*, *out_flg\$***

Where:

<i>ibox_def\$</i>	is the Input box control structure.
<i>ibox_val\$</i>	contains the value to display.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

To delete an Input Box:

CALL "*inpbox.del", *ibox_def\$*, *in_flg\$*, *out_flg\$***

Where:

<i>ibox_def\$</i>	is the Input box control structure.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

****LSTBOX***List boxes*

A list box is used to provide the user with a predetermined list of possible choices in a scrolling window. The input returned from a list box can only be one of the items in the list.

Definition call sequence:

CALL "*lstbox.def", *ibox_def\$*, *ctl_val*, *lst_tbl\${ALL}*, *col*, *line*, *width*, *height*, *in_flg\$*, *out_flg\$***

Where:

<i>ibox_def\$</i>	is the variable to receive List box control structure.
<i>ctl_val</i>	contains the CTL event to occur when mouse enters box.
<i>lst_tbl\${ALL}</i>	contains the values to place in the list box.
<i>col</i>	contains the column to start list box
<i>line</i>	contains the line to start the list box
<i>width</i>	contains the width of the box. If omitted (or <3), this is set the length of the largest item in the list plus two for the border, up to a maximum of 40 (or the edge of the window).
<i>height</i>	contains the height of the box. If omitted (or <3), this is set to the number of entries in the list plus two up to a maximum of 12 (or to the bottom of window).
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

****LSTBOX – List Box (continued):****List box display updating:**

```
CALL "***lstbox.dsp", lbox_def$, new_val$, lst_tbl${ALL}, reset_ind, in_flg$,
out_flg$
```

Where:

<i>lbox_def\$</i>	is the List box control structure.
<i>new_val\$</i>	contains the value to be highlighted within the list box.
<i>lst_tbl\${ALL}</i>	contains the values to place in the box.
<i>reset_ind</i>	is an optional parameter that when set to non-zero causes the complete contents of the list box need to be re-displayed.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

List box event processing call:

```
CALL "***lstbox.xeq", lbox_def$, new_val$, lst_tbl${ALL}, reset_ind, in_flg$,
out_flg$
```

Where:

<i>lbox_def\$</i>	is the List box control.
<i>new_val\$</i>	returns the value selected within list box.
<i>lst_tbl\${ALL}</i>	contains the values to place in box.
<i>reset_ind</i>	is an optional parameter that should be set to non-zero if the complete contents of the list box need to be re-displayed.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	contains output flags. If the user chooses a selection using the keyboard and presses ENTER, a flag of 'RET' is returned. A flag of 'MSE' is returned if the mouse is used.

List box delete:

```
CALL "***lstbox.del", lbox_def$, in_flg$, out_flg$
```

Where:

<i>lbox_def\$</i>	is the list box control structure.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

****MENU*****Pull-down menus***

These functions provide access to pull down menus. The menu is defined by the **MENU.DEF subprogram. When a menu event is detected, **MENU.XEQ provides the menu interface and returns the characters selected.

Example:

A typical menu would be defined as:

****MENU – Pull-down menus (continued):**

```
" [ &File , &Edit , &Help ] , F : [ &Open , &Save , E&Xit ] , E : [ &Delete , &Insert ] "
```

This results in a menu of "File Edit Help" across the top of the window with the first letter (the one prefixed by &) as the access key. If "F" is entered, a pull down menu of "Open Save EXit" will appear.



If Open is selected, the **MENU.XEQ routine returns "FO" for File/Open. If no menu was selected, a Null string is returned.

A 'menu flag string' can be provided to contain the sequence:

```
,{code}+      indicates that the specified menu item is
               toggled ON.
,{code}-      indicates that the specified menu item is
               not available.
```

Defining a menu:

```
CALL "***menu.def", menu_dta$, ctl_val, menu_flg$, in_flg$, out_flg$
```

Where:

```
menu_dta$      contains the menu/sub-menu items. The selection letters are
               prefixed by a '&'.
ctl_val        contains the CTL event to occur when mouse selects the menu
               region.
menu_flg$      contains the menu flag string (optional).

in_flg$        contains input flags (Future).
out_flg$       receives output flags (Future).
```

Processing a menu event:

```
CALL "***menu.xeq", menu_dta$, menu_resp$, menu_flg$, in_flg$, out_flg$
```

Where:

```
menu_dta$      contains the menu/sub-menu items. The selection letters are
               prefixed by a '&'.
menu_resp$     returns the code for a selected menu item.
menu_flg$      contains the menu flag string (optional).
in_flg$        contains input flags (Future).
out_flg$       receives output flags (Future).
```

Inserting a checkmark beside a menu selection:

```
CALL "***menu.on", menu_flg$, menu_req$, in_flg$, out_flg$
```

****MENU – Pull-down menus (continued):****Where:**

menu_flg\$ is the menu flag string to update.
menu_req\$ contains the menu code for checkmark.
in_flg\$ contains input flags (Future).
out_flg\$ receives output flags (Future).

Removing a checkmark from a menu selection:

CALL "*menu.off", menu_flg\$, menu_req\$, in_flg\$, out_flg\$**

Where:

menu_flg\$ is the menu flag string to update.
menu_req\$ contains the menu code for checkmark.
in_flg\$ contains input flags (Future).
out_flg\$ receives output flags (Future).

Disabling a menu selection:

CALL "*menu.dis", menu_flg\$, menu_req\$, in_flg\$, out_flg\$**

Where:

menu_flg\$ is the menu flag string to update.
menu_req\$ contains the menu code to disable.
in_flg\$ contains input flags (Future).
out_flg\$ receives output flags (Future).

Enabling a menu selection:

CALL "*menu.ena", menu_flg\$, menu_req\$, in_flg\$, out_flg\$**

Where:

menu_flg\$ is the menu flag string to update.
menu_req\$ contains the menu code to enable.
in_flg\$ contains input flags (Future).
out_flg\$ receives output flags (Future).

****OPEN.FILE***Open File*

This utility provides a Windows-like open file routine.

To invoke the Open File utility:

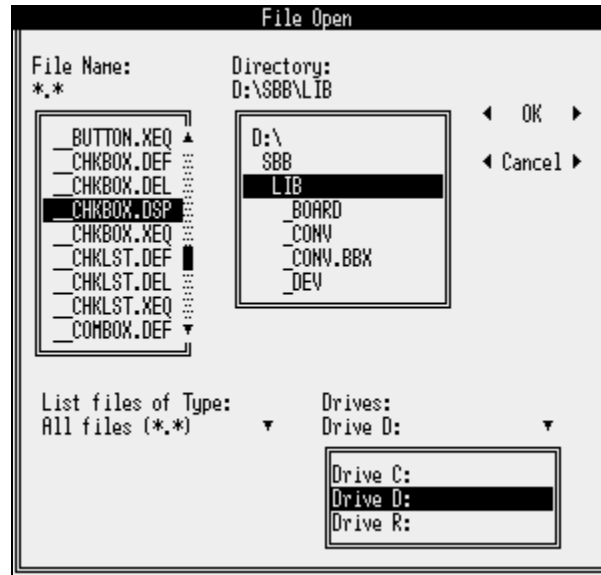
CALL "*open.file", fle_val\$, fle_pth\$, fle_ttl\$, typ_tbl\${ALL}, in_flg\$, out_flg\$**

Where:

fle_val\$ contains the default file name on entry, returns the selected file name. If no file is selected, "" is returned.
fle_pth\$ contains the initial directory path.
fle_ttl\$ contains the window title (default is 'File Open').
typ_tbl\${ALL} is the table of descriptions and valid file name masks. (see below)

****OPEN.FILE – Open file (continued):**

in_flg\$ contains input flags (Future).
out_flg\$ receives output flags (Future).



Each entry in the Table of Descriptions (typ_tbl\$(ALL)) must be in the following format:

TYP_TBL\$(X)="Description | Mask"

Where:

Description is the text that is displayed in the window.
 | is the separator between the description and the mask.
Mask are the filters used to identify the files associated with the description. The standard '*' and '?' wild card characters can be used.

i.e. TYP_TBL\$(1)="All files (*.*) | *.*"
 TYP_TBL\$(2)="Batch files (*.bat) | *.bat"

****OPTION.BOX***Option Box*

Description:

This routine is used to display a message in a window and to allow the user to select from a series of short (one word) options.

CALL "*option.box", yorn_resp\$, opt_msg\$, yorn_data\$, in_flg\$, out_flg\$**

Where:

<i>yorn_resp\$</i>	is the variable to receive the selection.
<i>opt_msg\$</i>	contains the message to display in the window. Multiple lines of description may be used by inserting the SEP variable at the end of each line.
<i>yorn_data\$</i>	is a string containing the options separated by commas. The selection letters are to be prefixed by the & (ie. "&Yes,&No" or "&OK").
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

****PROMPT.BOX***Prompt Box*

Description:

This routine displays a message in a window and prompts for an alphanumeric response. Prompt boxes accept input wider than the actual screen by implementing a scrolling input field.

Define Calling sequence:

CALL "*prompt.box", desc\$, fld_siz, resp\$, in_flg\$, out_flg\$**

Where:

<i>desc\$</i>	contains the text to display before the input field.
<i>fld_siz</i>	contains the maximum size of the input field.
<i>resp\$</i>	contains the initial value passed in and returns the result of the input.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

****VARBOX**

Variable List boxes

Description:

A variable list box consists of an input field with a list box displayed below which contains a list of the possible 'canned' input choices (such as a file name, etc..). The input returned from a variable list box can be one of the entries from the list, or any input entered by the user.

To define a variable list box:

CALL **varbox.def***, *vlst_def*\$, *vlst_val*\$, *ctl_val*, *vlst_tbl*{*ALL*}, *col*, *line*, *width*,
height, *in_flg*\$, *out_flg*\$

Where:

<i>vlst_def</i> \$	is the variable to receive the list box control structure.
<i>vlst_val</i> \$	contains the initial value to display.
<i>ctl_val</i>	contains the CTL event to occur when mouse enters box or the input field.
<i>vlst_tbl</i> { <i>ALL</i> }	contains the values to place in box.
<i>col</i>	contains the column to start the variable input.
<i>line</i>	contains the line to start the variable input.
<i>width</i>	contains the width of the input. The box size is set 2 characters longer than this value to accommodate a space and drop activation field following the input field.
<i>height</i>	contains the height of the box. If omitted (or <3) this is set to the number of entries in the list plus two up to a maximum of 12 (or to the bottom of window).
<i>in_flg</i> \$	contains input flags (Future).
<i>out_flg</i> \$	receives output flags (Future).

To process a Variable list box event:

CALL **varbox.xeq***, *vlst_def*\$, *vlst_val*\$, *vlst_tbl*{*ALL*}, *vb_reset*, *in_flg*\$,
out_flg\$

Where:

<i>vlst_def</i> \$	is the Variable list box control structure.
<i>vlst_val</i> \$	returns the value selected/entered.
<i>vlst_tbl</i> { <i>ALL</i> }	contains the values to place in box.
<i>lb_reset</i>	is an optional parameter that when set to non-zero causes the complete contents of the list box to be re-displayed.
<i>in_flg</i> \$	contains input flags. To force execution of this routine, the flag '+FOCUS' can be passed in through <i>in_flg</i> \$. This is useful for systems which do not employ a Mouse.
<i>out_flg</i> \$	receives output flags (Future).

To delete a Varbox:

CALL **varbox.del***, *vlst_def*\$, *in_flg*\$, *out_flg*\$

****VARBOX – Variable list boxes (continued):****Where:**

<i>vlst_def\$</i>	is the Variable list box control structure.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

****VMOVE.CHK*****Vertical movement processor*****Description:**

This is an internal routine used by the list box functions to intercept keyboard movement. It is used to process cursor movement keys ↑, ↓, HOME, END, PGUP, and PGDN.

Calling sequence:

CALL "*vmove.chk", vb_new, vb_max, vb_bjmp, in_flg\$, out_flg\$**

Where:

<i>vb_new</i>	on input, has the current value and returns the new value calculated.
<i>vb_max</i>	contains the largest value in the list
<i>vb_bjmp</i>	contains the amount to adjust the new value by when detecting a PGUP or PGDN.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

****VSCRBR*****Vertical scroll Bar***

A vertical scroll bar is used to allow the user to set a value or position. The list boxes use this routine when there are more selections than will fit on one screen to indicate that more choices exist. Typically, the scroll bar appears within a window region. When the mouse is activated within the scroll bar region, a CTL value EVENT is initiated. The program must call **VSCRBR.XEQ to handle this event and subsequently, **VSCRBR.DSP to update the scroll bar display. **VSCRBR.DSP must be called to update the display whenever the value of the scroll bar changes via other program functions (timer, file position, initial setting...).

These scroll bar functions should not be confused with the Window region vertical scroll bar which appears to the right side of the current window.

Defining scroll bar:

CALL "*vscrbr.def", vscr_def\$, ctl_val, col, line, width, height, in_flg\$, out_flg\$**

Where:

<i>vscr_def\$</i>	is the variable to receive the Vertical scroll bar control structure.
<i>ctl_val</i>	contains the CTL event to occur when mouse enters scroll bar.
<i>col</i>	contains the column to start the scroll bar.

****VSCRBR – Vertical scroll bar (continued):**

<i>line</i>	contains the line to start the scroll bar.
<i>width</i>	contains the scroll bar width (default 1).
<i>height</i>	contains the scroll bar height (default is to bottom of current window).
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

Updating scroll bar display:

CALL **"**vscrbr.dsp", vscr_def\$, new_val, max_val, big_jump, sml_jump, in_flg\$, out_flg\$**

Where:

<i>vscr_def\$</i>	is the scroll bar control structure.
<i>new_val</i>	contains the value within scroll bar (minimum value = 1).
<i>max_val</i>	contains the maximum value within scroll bar.
<i>big_jump</i>	contains the amount to increase/decrease for BIG Jump (mouse pressed above/below current position). If < 1 or not specified, this value defaults to max_val divided by # of entries in bar.
<i>sml_jump</i>	contains the amount to increase/decrease for SMALL Jump (arrows at either end of bar). If not specified or < 1, a value of 1 is used.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

Processing scroll bar event:

CALL **"**vscrbr.xeq", vscr_def\$, new_val, max_val, big_jump, sml_jump, in_flg\$, out_flg\$**

Where:

<i>vscr_def\$</i>	is the scroll bar control structure.
<i>new_val</i>	returns the value corresponding to the selected position within the scroll bar. The scroll bar must be updated by calling "**VSCRBR.DSP" after the value is returned and processed. (minimum value = 1).
<i>max_val</i>	contains the maximum value within scroll bar.
<i>big_jump</i>	contains the amount to increase/decrease for BIG Jump (mouse pressed above/below current position). If < 1 or not specified, this value defaults to max_val divided by # of entries in bar.
<i>sml_jump</i>	contains the amount to increase/decrease for SMALL Jump (arrows at either end of bar). If not specified or < 1, a value of 1 is used.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

Deleting scroll bar:

CALL **"**vscrbr.del", vscr_def\$, in_flg\$, out_flg\$**

Where:

<i>vscr_def\$</i>	is the scroll bar control structure.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

****WARN.BOX**

Warning Box

Description:

This routine is used to display an Error or Warning message. It places the message in a window on the screen and prompts for a response. The prompt selections passed in are typically Yes or No.

Define Calling sequence:

CALL "*warn.box", yorn_resp\$, err_msg\$, yorn_data\$, in_flg\$, out_flg\$**

Where:

<i>yorn_resp\$</i>	returns the letter of the selected option
<i>err_msg\$</i>	contains the message to display in the window. Multiple lines of description may be used by inserting the SEP variable at the end of each line.
<i>yorn_data\$</i>	contains a string containing the options separated by commas. The selection letters are to be prefixed by the & (ie. "&Yes,&No").
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

****WHSCRL**

Window Horizontal scroll Bar

A horizontal window scroll bar is used to represent a logical placement within a horizontally scrollable data item such as a long line or record. It always appears on the bottom line (border) of a window. When the mouse is activated within the scroll bar region, a CTL value EVENT is initiated. The program must call **WHSCRL.XEQ to handle this event and subsequently **WVSCRL.DSP to update the scroll bar display. **WVSCRL.DSP must be CALLED to update the display whenever the value of the scroll bar changes via other program functions (timer, file position, initial setting...).

CALL "*whscrl.def", hscr_def\$, ctl_val, in_flg\$, out_flg\$**

Where:

<i>hscr_def\$</i>	is the Variable to receive the control structure for the scroll bar.
<i>ctl_val</i>	contains the CTL event to occur when mouse enters scroll bar.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

To update the scroll bar display:

CALL "*whscrl.dsp", hscr_def\$, new_val, max_val, big_jmp, sml_jmp, in_flg\$, out_flg\$**

****WHSCRL – Window horizontal scroll bar (continued):****Where:**

<i>hscr_def\$</i>	is the Variable with the control structure for the scroll bar.
<i>new_val</i>	contains the value within scroll bar to be displayed (minimum value = 1).
<i>max_val</i>	contains the maximum value within scroll bar.
<i>big_jmp</i>	contains the amount to increase/decrease for BIG Jump (mouse pressed left/right of current position). If < 1 or not specified, this value defaults to <i>max_val</i> divided by # of entries in bar.
<i>sml_jmp</i>	contains the amount to increase/decrease for SMALL Jump (arrows at either end of bar). If not specified or < 1, a value of 1 is used.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

To process scroll bar event:

CALL "*whscrl.xeq", , in_flg\$, out_flg\$**

Where:

<i>hscr_def\$</i>	is the scroll bar control structure
<i>new_val</i>	returns the value corresponding to the selected position within the scroll bar. The scroll bar must be updated by calling "***WHSCRL.DSP" after the value is returned and processed. (minimum value = 1).
<i>max_val</i>	contains the maximum value within the scroll bar.
<i>big_jmp</i>	contains the amount to increase/decrease for BIG Jump (mouse pressed left/right of current position). If < 1 or not specified, this value defaults to <i>max_val</i> divided by # of entries in bar.
<i>sml_jmp</i>	contains the amount to increase/decrease for SMALL Jump (arrows at either end of bar). If not specified or < 1, a value of 1 is used.
<i>in_flg\$</i>	contains input flags (Future).
<i>out_flg\$</i>	receives output flags (Future).

****WVSCRL*****Window Vertical scroll Bar***

A vertical window scroll bar is used to represent a position within the current window. It always appears on the right border of the current window. When the mouse is activated within the scroll bar region, a CTL value event is initiated. The program must call **WVSCRL.XEQ to handle this event and then subsequently, **WVSCRL.DSP to update the scroll bar display. **WVSCRL.DSP must be CALLED to update the display whenever the value of the scroll bar changes via other program functions.

Defining scroll bar:

CALL "*wvscrl.def", vscr_def\$, ctl_val, in_flg\$, out_flg\$**

Where:

<i>vscr_def\$</i>	is the Variable to receive the control structure for the scroll bar.
-------------------	--

****WVSCRL – Window vertical scroll bar (continued):**

ctl_val contains the CTL event to occur when mouse enters scroll bar.
in_flg\$ contains input flags (Future).
out_flg\$ receives output flags (Future).

Updating display:

CALL "*wvscrl.dsp", vscr_def\$, new_val, max_val, big_jmp, sml_jmp, in_flg\$,
 out_flg\$**

Where:

vscr_def\$ is the Variable with the control structure for the scroll bar.
new_val contains the value within scroll bar to be displayed (1 = first value)
max_val contains the maximum value within scroll bar
big_jmp contains the amount to increase/decrease for BIG Jump (mouse pressed above/below current position). If < 1 or not specified, this value defaults to *max_val* divided by # of entries in bar.
sml_jmp contains the amount to increase/decrease for SMALL Jump (arrows at either end of bar). If not specified or < 1, a value of 1 is used.
in_flg\$ Additional input parameters. Since the scaled representation on the bar can change, a value of 'SCALE_RESET' can be passed through *in_flg\$* to force a recalculation and redisplay of the bar.
out_flg\$ receives output flags (Future).

Processing Scroll bar event:

CALL "*wvscrl.xeq", vscr_def\$, new_val, max_val, big_jmp, sml_jmp, in_flg\$,
 out_flg\$**

Where:

vscr_def\$ is the scroll bar control structure.
new_val returns the value corresponding to the selected position within the scroll bar. The scroll bar must be updated by calling "***WVSCRL.DSP" after the value is returned and processed. (minimum value = 1).
max_val contains the maximum value within scroll bar
big_jmp contains the amount to increase/decrease for BIG Jump (mouse pressed above/below current position). If < 1 or not specified, this value defaults to *max_val* divided by # of entries in bar.
sml_jmp contains the amount to increase/decrease for SMALL Jump (arrows at either end of bar). If not specified or < 1, a value of 1 is used.
in_flg\$ contains input flags (Future).
out_flg\$ receives output flags (Future).

Index

- *
- **BUTTON.xxx, **68–69**
- **CHKBOX.xxx, **69–70**
- **COMBOX.xxx, **71–72**
- **CTLBTN.xxx, **72–74**
- **DRPBOX.xxx, **74–75**
- **ERROR.BOX, **75**
- **HMOVE.CHK, **75–76**
- **INPBOX.xxx, **77–78**
- **LSTBOX.xxx, **78–79**
- **MENU.xxx, **79–81**
- **OPEN.FLE, **81–83**
- **OPTION.BOX, **83**
- **PROMPT.BOX, **83–84**
- **VSCRBR, **85–87**
- **WARN.BOX, **87**
- **WVSCRL.xxx, **88–91**
- *C, **2–3**
- *CHKKEY, **4**
- *CONTROL, **58, 59, 60**
- *DATE, **49–50**
- *DEV, **58–59**
- *E, **4–8**
- *ERROR, **59**
- *F, **8–9**
- *FI, **9**
- *FL.LST, **50**
- *FL.NME, **51**
- *FM, **10**
- *HELP, **59**
- *HELP.INF, **59**
- *HELP.PRG, **60**
- *HELP.PWD, **59, 60**
- *HELP.xx, **59–60**
- *KYBRD.CFG, **60–61**
- *KYBRD.STD, **60–61**
- *LEXDEF.xx, **61**
- *LEXEDIT, **11–12, 61**
- *LEXTBL.xx, **61**
- *MLFILE.xx, **62**
- *MSGUPD, **12–13**
- *OPTSEL, **51–52, 51–52, 51–52, 51–52, 51–52, 51–52, 51–52**
- *PG.CNV, **52**
- *POPSEL, **52–53**
- *PR.ABT, **53**
- *PR.OPN, **54**
- *PR.SEL, **54**
- *PRMDEF.xx, **62**
- *QUERY, **62–63**
- *QUERY.DEF, **63**
- *SCR.SVE, **55**
- *SELECT, **55–56**
- *START.UP, **64**
- *TTY, **64–65**
- *U, **13–14**
- *UC, **14**
- *UCK, **15–16**
- *UCL, **16–18**
- *UCP, **18–19, 62**
- *UD, **19**
- *UDD, **19–20**
- *UDG, **20**
- *UDM, **21**
- *UDP, **21–22**
- *UDR, **22**
- *UDV, **23**
- *UF, **24**
- *UFA, **24**
- *UFAC, **25–27**
- *UFAM, **27–28**
- *UFAR, **28–29**
- *UFC, **29–30**
- *UFD, **30–31**
- *UFE, **31**
- *UFI, **31–32**
- *UFM, **32–34**
- *UFR, **35–36**
- *UFU, **36–38**
- *UFV, **38–39**

*UG, 39
 *UGM, 39–40
 *UGS, 40–41
 *UP, 42
 *UPB, 42–44
 *UPC, 44–45
 *UPD, 45
 *UPL, 45–46
 *UPM, 46–47
 *UPR, 47
 *UPS, 47–48
 *VLDATE, 56–57

A

Abort print file, 53
 ACTIVATE.PVX, 57
 Activation file, 57

B

Boxes

combo, 71–72
 drop, 74–75
 error, 75
 input, 77–78
 list, 78–79
 option, 83
 prompt, 83–84
 warning, 87

Bulk program scan/edit, 42–44

Buttons, 68–69

Buttons-Control, 72–74

C

Calculator, 2–3

Change

current directory, 20
 max records, 27–28

Check Boxes, 69–70

Check file keys, 25–27

Clear file contents, 31

Combo Boxes, 71–72

Compare programs, 44–45

Configuration menu, 14

Configure keyboard, 15–16, 60–61

Convert program file, 52

Copy

file, 29–30

program, 52
 Create
 directory, 21
 file, 32–34
 program file, 46–47
 Cross reference, 45–46

D

Date validation, 49–50

Define queries, 63

Delete

directory, 19–20
 file, 30–31
 program, 45

Device drivers, 58–59

Directory

change, 20
 create, 21
 delete, 19–20
 list of files, 50
 print, 21–22
 rename, 22
 utilities, 19
 view, 23

Drop boxes, 74–75

E

Editor

multiple programs, 42–44
 program, 4–8

Erase

file, 31

Error boxes, 75

Error handler, 59

Export - Spreadsheet, 40–41

F

File

admin menu, 24
 change max records, 27–28
 check keys, 15–16
 copy, 29–30
 create, 32–34
 data erasure, 31
 delete, 30–31
 information, 23, 31–32
 information display, 9

keyboard config, 60–61
 modify data, 10, 36–38
 recovery, 4, 28–29
 rename, 35–36
 system messages, 62
 temp file name, 51
 update, 36–38
 utility menu, 24
 verify keys, 25–27
 view, 38–39

File Open selector, 81–83

Files

directory list, 50
 open list, 8–9

Flags

In and out, 68

G

General Utilities, 39

H

Help

field - display/edit, 59
password, 60
 program display/edit, 60
 text file, 59–60

HELP.xx, 59, 60

Home directory, 20

Horizontal

movement check, 75–76

Hot-Key interceptor, 58

I

In and Out Flags, **68**

Information

file, 31–32

Input Boxes, 77–78

Input validate, 56–57

Intercept Hot-key, 58

K

Keyboard

config file, 60–61
 configuration, 15–16, 64–65

Keys

verification, 25–27

L

Linkfile maintenance, 16–18

List

boxes, 78–79
 select entry, 55–56

List program, 45–46

M

Maximum record count, 27–28

Menu, 79–81

configuration, 14
 directory, 19
 file admin, 24
 file functions, 24
 general utilities, 39
 program utilities, 42
 utilities, 13–14

Message

file update, 12–13
 library, 62

Modify

data records, 36–38
 file data, 10
 parameters, 18–19
 syntax tables, 11–12

Mortgage Calculations, 39–40

Movement check

horizontal, 75–76

N

Name for temporary file, 51

O

Object types, **65–68**

On-line help, 59

On-line Query, 62–63

Open file - Selector, 81–83

Open file list, 8–9

Open printer, 54

Option boxes, 83

Option selector, 51–52, 51–52, 51–52, 51–53,
 51–52, 51–52, 51–52

P

Parameter settings, 18–19, 62

Password

- help subsystem**, 60

- Popup option selector, 52–53

Print

- directory, 21–22
- screen, 58

Printer

- abort, 53
- open, 54
- select, 54

Program

- compare, 44–45
- copy/convert, 52
- create, 46–47
- delete, 45
- editor, 4–8
- help text, 60
- list, 45–46
- rename, 47
- scan/edit, 42–44
- security, 47–48
- utilities menu, 42

- Prompt boxes, 83–84

ProvideX

- activation, 57
- initialization, 64

Q**Query**

- definition, 63
- handler, 62–63

R

- Range checking, 56–57

- Recover file, 28–29

Rename

- directory, 22
- file, 35–36
- program, 47

S

- Sample error handler, 59

- Save screen contents, 55

- Scan programs, 42–44

Screen

- print contents, 58
- save contents, 55

Scroll bars

- vertical, 85–87
- window vertical, 88–91

- Security password, 47–48

Select

- from list, 55–56
- option, 51–52, 51–52, 51–52, 51–53, 51–52, 51–52, 51–52
- printer, 54

- Spreadsheet Export, 40–41

- Syntax tables, 11–12, 61

System

- parameters, 62
- startup, 64

T

- Tables for Windows functions, **68**

- Temporary file name, 51

- Terminal keyboard map, 64–65

- Tracing keys, 4

U**Update**

- file, 36–38

V**Validate**

- dates, 49–50
- input, 56–57

- Verify file keys, 25–27

Vertical

- scroll bars, 85–87

View

- directory, 23
- file, 36–39

W

- Warning boxes, 87

Window frame

- vertical scroll bar, 88–91

Windows

- object types, 65–68