

SSI SanSoft, Inc.

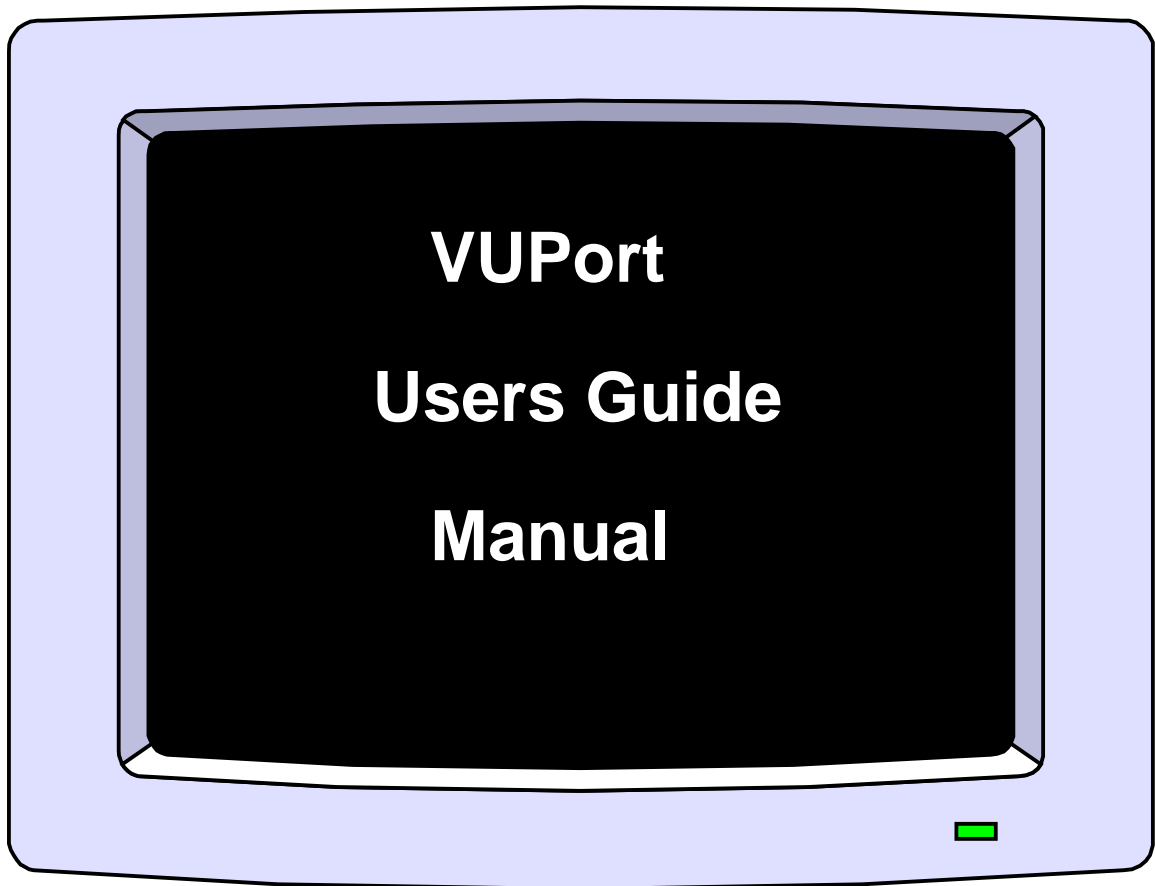


Table of Contents

1 CONCEPTS AND CONVENTIONS.....	1
1.1 INTRODUCTION	1
1.2 CURSOR CONTROL.....	2
1.3 FILE NAMING.....	5
2 SYSTEM CONSIDERATIONS.....	7
2.1 ENVIRONMENT VARIABLES	7
2.2 UNIX ENVIRONMENT VARIABLES.....	8
2.3 MICRO FOCUS COBOL ENVIRONMENT VARIABLES.....	10
2.4 VUPOINT ENVIRONMENT VARIABLES	12
2.5 VUPOINT USAGE CONSTANTS	15
2.6 SYSTEM INITIALIZATION	16
2.7 SYSTEM TERMINATION	17
2.8 STARTING THE COMMAND PROCESSOR.....	18
2.9 NATIVE LANGUAGE SUPPORT	19
3 COMMAND PROCESSOR.....	21
3.1 RUNNING PROGRAMS	21
3.2 WINDOW CONTROL	23
3.3 MANAGE FILES	25
3.4 MANAGE PRINT QUEUE	27
3.5 PRINTER CONTROL	29
3.6 DEVICE CONTROL.....	31
4 PROCEDURE LANGUAGE	33
4.1 INTRODUCTION	33
4.2 ASSIGN STATEMENT	36
4.3 ATTACH STATEMENT.....	37
4.4 CALL STATEMENT.....	38
4.5 DECLARE STATEMENT.....	39
4.6 DISMOUNT STATEMENT.....	40
4.7 DESTROY STATEMENT.....	41
4.8 EXTRACT STATEMENT	42
4.9 GO TO STATEMENT	44
4.10 HELP STATEMENT.....	45
4.11 IF STATEMENT	46
4.12 LOGOFF STATEMENT	48
4.13 MESSAGE STATEMENT.....	49
4.14 MOUNT STATEMENT.....	51
4.15 OPTIONS STATEMENT	52
4.16 PRINT STATEMENT.....	53
4.17 PROMPT STATEMENT.....	55
4.18 RENAME STATEMENT	57
4.19 RETURN STATEMENT	58
4.20 RUN STATEMENT.....	59
4.21 SCRATCH STATEMENT	61
4.22 SET STATEMENT	62
4.23 SHELL STATEMENT	63
4.24 SUBMIT STATEMENT.....	64
4.25 USING STATEMENT	66

5 FORMS CONTROL..... 67

- 5.1 INTRODUCTION 67
- 5.2 FORM NUMBER 68
- 5.3 LINES PER PAGE 69
- 5.4 PRINT OVER PERFORATION..... 70
- 5.5 HEADER/TRAILER PAGE FORMAT 71
- 5.6 SIMILAR FORMAT..... 72
- 5.7 EXPAND TABS..... 73
- 5.8 DOWNLOAD INFORMATION 74

6 APPLICATION INTERFACING.....ERROR! BOOKMARK NOT DEFINED.

- 6.1 STARTING THE VUPORT SYSTEM **ERROR! BOOKMARK NOT DEFINED.**
- 6.2 KSARC, INITIAL PROCEDURE FILE..... **ERROR! BOOKMARK NOT DEFINED.**
- 6.3 EXECUTING DIFFERENT APPLICATIONS **ERROR! BOOKMARK NOT DEFINED.**
- 6.4 EXECUTING MICRO FOCUS COBOL APPLICATIONS **ERROR! BOOKMARK NOT DEFINED.**
- 6.5 LPRB, THE LINE PRINTER SPOOLER **ERROR! BOOKMARK NOT DEFINED.**
- 6.6 HINTS ON SETTING UP MENUS AND PROCEDURES **ERROR! BOOKMARK NOT DEFINED.**

List of Tables

TABLE 1 CURSOR KEY FUNCTIONS.....	2
TABLE 2 GENERAL FUNCTION KEYS	3
TABLE 3 USAGE CONSTANTS.....	15
TABLE 4 WINDOW STATUSES	23
TABLE 5 FILE MANAGEMENT FUNCTION KEYS.....	25
TABLE 6 PRINT QUEUE STATUSES	28
TABLE 7 PRINTER STATUSES	29
TABLE 8 LOGICAL EXPRESSIONS	34
TABLE 9 EXTRACT KEYWORDS	43
TABLE 10 LOGICAL EXPRESSIONS AND ABBREVIATIONS.....	46
TABLE 11 LIST OF SET KEYWORDS.....	62

1 Concepts and Conventions

1.1 Introduction

The VUPort system is a utility package designed to host applications which have been migrated from a Wang VS environment. The package contains many features including a command processor, a print spooler, a batch job scheduler and many file maintenance utilities.

This document is intended to describe to a system administrator and end users how to administer and operate the VUPort system. This document in parts makes the assumption that the reader is familiar with the Unix operating system and understands some general concepts of computer operation.

In general, the VUPort system can be broken into four main parts. These parts are:

1. The Command Processor
2. The Procedure Interpreter
3. The System utilities
4. The Development Environment

This documents focus is on the Command Processor and Procedure Interpreter. If additional information is required about the system utilities, please consult the VUPort Utility Guide.

1.2 Cursor Control

When entering data on a screen, there is a set of control keys used for moving through the screen. Users are free to fill in any fields they wish on the screen until they hit the Return Key (Also known as Execute Key) or any allowable function key. At this point, screen entry is completed and the program continues.

VUPort understands thirty-two Function Keys. However, they may not all be allowable on every screen. Below is a list of the other control and field editing keys understood within VUPort.

Table 1 Cursor Key Functions

KEY	KEY ACTION
INSERT KEY	Insert one space into a modifiable field at the current cursor position. Move all characters to current cursor position.
DELETE KEY	Delete the character at the current cursor position. Moves all characters to the right of the cursor one character to the left and places a space into the last character in the field.
CLEAR FIELD	Blanks all characters from the cursor to the end of the field.
RIGHT ARROW	Moves the cursor one position to the right
LEFT ARROW	Moves the cursor one position to the left.
UP ARROW	Moves the cursor up one line, leaving it at the same column.
DOWN ARROW	Moves the cursor down one line, leaving it at the same column.
HOME	Moves the cursor to the first modifiable field on the screen.
TAB	Moves the cursor to the next modifiable field on the current line or on the line below, if there are no more modifiable fields on the current line.
BACK TAB	Moves the cursor to the previous modifiable field on this line or on the line above the current line, if there are no modifiable fields on the current line.
NEW LINE	Moves the cursor to the next modifiable field at least one line below the current line.

General Function Keys

Throughout VUPort certain function keys provide similar actions. In most cases these function keys all produce the same action. Depending on the location, these pre-defined function keys are used to scroll through lists of items. Below is a list of the function keys and their general meaning.

Table 2 General Function Keys

KEY	KEY ACTION
F2	Display first screen of a list.
F4	Display last screen of a list.
F4	Display previous screen of a list.
F5	Display next screen of a list.
F15	Print command Screen.
F16	Exit current screen and return to previous screen. Also terminate or cancel current action.

Screen Layout

Throughout VUPort there is a consistent screen format. Most screens are terminated with a function key to specify the next action to take or by the execute key to specify that the screen is complete. The allowable function keys and their associated actions are located at the bottom of each screen. In every case, pfkey 16 is used to terminate or cancel the current screen and return to the previous screen. Within the VUPort Help Processor, pfkey 15 is always the Print Command Screen. When a screen has modifiable fields, such as the Run Program Screen, the screen can be terminated with the Execute Key. This usually indicates that the action is to be taken.

Some screens contain many objects for which an action can apply. To specify which object you wish to affect by an action, you need to place the cursor next to the object and press the associated pfkey for that action. For instance, the File Display screens contain up to 30 files or libraries per screen. If you wish to delete a particular file, (pfkey 8) you would position the cursor, by using the arrow keys or other cursor movement keys, next to the file and then press pfkey 8.

An additional method of moving the cursor is available on screens that display lists of items. When selecting an item from a menu or list you can move the cursor directly to the location of that item by pressing the character that immediately follows the protected tab stop. For example, if a menu contains four items each starting with the letters A, B, C and D you can move the cursor directly to the item starting with D by pressing the character D.

1.3 File Naming

VUPort provides a file structure much simpler than the file structure native to the UNIX environment. VUPort file structure contains 3 levels: Volumes, Libraries and Files. This file structure is how VUPort identifies all files.

A volume is the highest grouping structure. Volume names contain up to six alphanumeric upper case characters. The system will default the name to upper case if it is specified in lower case. Volumes contain libraries. Library names are up to eight alphanumeric characters. Case is significant since these can be either upper or lower case, therefore, case. Libraries contain files. File names are up to fourteen alphanumeric characters. Filename and library names can also be either upper or lower case. Unless you have the VUPort environment variable VUPSW set, the case is significant. See the section in this manual on Environment Variables for more information on the VUPSW variable.

2 System Considerations

2.1 Environment Variables

VUPort uses a number of environment variables. Environment variables to customize the behavior of the system, as well as to define the location of certain files and directories on your computer. These values must be set in order for VUPort to operate properly. In addition to the environment variables VUPort requires, the Micro Focus COBOL system has a set of variables it requires. Since VUPort using the Micro Focus COBOL compiler and runtime, the variables which are required by Micro Focus must also be set in order for these components to operate properly.

There are a number of options available to you to set the environment variables before starting VUPort. The option you choice will depend on the method you use during a user login. In many cases a user will log on using a UNIX system shell. Typically the users shell will either be the bourne shell, the kourne shell or the C shell. If the user is logging in using either the bourne shell or the kourne shell the environment variables can be set in the users .profile file. If the user is logging in using the C shell, the environment variables can be set in the users .cshrc file. The .profile and .cshrc files is located in the users HOME directory. Below are examples command which can be added to the users .profile file to set the PATH environment variable.

Bourne Shell Environment Variable Example

```
PATH=/bin:/usr/bin:/usr/lib/cobol/bin  
export PATH
```

Kourne Shell Environment Variable Example

```
export PATH=/bin:/usr/bin:/usr/lib/cobol/bin
```

C Shell Environment Variable Example

```
setenv PATH /bin:/usr/bin:/usr/lib/cobol/bin
```

2.2 UNIX Environment Variables

2.2.1 HOME

The HOME environment variable is used by the VUPort system to locate the VUPort startup procedure `.ksarc` when the VUPort command processor (`ksamain`) is invoked with the `-I` startup option. See the section on System startup for more information regarding starting the VUPort command processor.

2.2.2 PATH

The PATH environment variable is used by the system shell to locate command files. This variable can contain multiple directories which are searched in the order specified when a command is entered. A typical PATH variable contains a list of system directories which contain the UNIX commands the user executes. In addition to these directories, you need to list the location of the directory which contains the VUPort commands you wish to execute. During the installation you were prompted for the location of the VUPort program files and possibly the location of the VUPort developer programs. These two locations must be added to the PATH environment variable if they are not already in it.

Example

```
export PATH=/bin:/usr/bin:/opt/vuport/bin
```

2.2.3 SHELL

The SHELL environment variable is used to determine which shell VUPort will use when executing UNIX commands. Since it is possible to execute shell scripts and commands from within the VUPort environment, you must specify which shell VUPort will use in order to carry out the UNIX commands you request. It is recommended that you use the bourne shell when executing UNIX commands from within VUPort. The location of the bourne shell on most systems is **/bin/sh**.

Example

```
export SHELL=/bin/ksh
```

2.2.4 TERM

The TERM environment variable is used to specify the type of terminal you are using. This variable corresponds to the terminfo entry which will be used. Note that when using the SanSoft terminal emulation product VUWin, the value of the TERM value must be `vu320`.

Example

```
export TERM=vu320
```

2.2.5 TERMINFO

The TERMINFO environment variable is used to determine the location of the terminfo database. If you do not want to use the system default terminfo database, you can use this environment variable to override the default location.

Example

```
export TERMINFO=/opt/vuport/terminfo
```

2.3 Micro Focus COBOL Environment Variables

2.3.1 COBDIR

The COBDIR environment variable defines the directory where the Micro Focus COBOL compiler is installed. The default location for the Micro Focus COBOL Compiler is /usr/lib/cobol. You only have to set the COBDIR environment variable if you have installed the Micro Focus COBOL compile in a directory other than the default directory.

Example

```
export COBDIR=/opt/vuport/cobol41
```

2.3.2 COBSW

The COBSW environment variable is used to set Micro Focus COBOL Runtime Switches. This switch is fully documented in the Micro Focus COBOL Users Guide, however it is recommended that the following values be set in order for VUPort to properly execute.

- A Do not animate programs. Optionally you can set the +A switch to turn animation on.
- F Do not check numeric data items for non-numeric data.
- N Do not pad null characters in line sequential files with null characters. (Optional)

Example

```
export COBSW=-A-F-N
```

2.3.3 COBCPY

The COBCPY environment variable is used by the Micro Focus COBOL Compiler and the VUPort preprocessor to locate your COBOL COPY. This variable is used during the compilation process and the animation (debugging) process. If you use copy statement qualification in your source code by including both the file name and library name in the COPY statement, you must specify both the location of the file and the location of the library when you set the COBCPY variable.

Example

```
export COBCPY=/u/apps:/u/apps/APCOPY
```

2.3.3.1 COBPATH

The COBPATH environment variable defines the location the Micro Focus COBOL runtime will search for subroutines which are CALL. Note that this variable is used only when a CALL is being executed from a COBOL program which has not be pre-processed by the VUPort pre-processor, or when a CALL is being executed from a COBOL program which has been pre-processed by the VUPort pre-processor, but the VUPort configuration file option Call_Nowarn is set on.

2.3.4 LIBPATH, SHLIB_PATH, LD_LIBRARY_PATH

These variables are used to specify the location where the shared libraries can be found. The version of the operating system will determine which of one of these variables you would use.

LIBPATH Use on IBM AIX

SHLIB_PATH Use on HP-UX

LD_LIBRARY_PATH Use on SCO Unix and AT&T System V.4

Example

```
export LIBPATH=/lib:/usr/lib:$COBDIR/coblib
```

2.4 VUPort Environment Variables

2.4.1 DISPOS

The DISPOS environment variable is used to override the default print file disposition of delete. By default when a print out is sent to the VUPort print queue, the disposition is D for delete. To override this behavior, set the DISPOS environment variable to the value of the disposition you wish. The valid options are:

- D Delete print file after printing
- K Keep the print file on the system after printing
- R Requeue the print file in the print queue with a status of hold after printing

Example

```
export DISPOS=R
```

2.4.2 MAXASIZE

The MAXASIZE environment variable is used to override the default maximum size of a parameter being passed between a procedure and a COBOL program. The MAXASIZE variable should be supplied in bytes and has a default value of 256 bytes. Setting this value has an effect on the amount of shared memory which is used to pass the parameters, so it should be set with care. Note that the size of a parameter being passed between two COBOL programs has a fixed limit of 65,000 bytes.

Example

```
export MAXASIZE=2048
```

2.4.3 SDCHAR

The SDCHAR environment variable is used to specify an alternate character to be displayed when a protected tab stop field is active. By default the '>' character is used to signify that a protected tab stop field is active.

Example

```
export SDCHAR=+
```

2.4.4 SPCHAR

The SPCHAR environment variable is used to specify an alternate character to be displayed for spaces in modifiable fields. By default the '_' character is used to signify that a space character is modifiable.

Example

```
export SPCHAR=.
```

2.4.5 VUCFILE

The VUCFILE environment variable is used to specify an alternate VUPort configuration file. The file path specified must be an absolute path including the actual file name. By default VUPort uses the configuration file called vuport.cfg in the directory pointed to by the VUPATH variable or /usr/lib/ksa if VUPATH is not set.

Example

```
export VUCFILE=/home/john/vuport.cfg
```

2.4.6 VUMAP

The VUMAP environment variable is used to specify an alternate base file name for the native language mapping configuration file. By default the value of the TERM environment variable is used to locate the native language mapping configuration file. See the section on native language support for more information on the native language mapping file.

Example

```
export VUMAP=altname
```

2.4.7 VUPATH

The VUPATH environment variable defines the location of the VUPort system files. By default the VUPort system files are installed in the directory /usr/lib/ksa. If you installed these file in an alternate location, you must set this environment variable to indicate where you installed the files.

Example

```
export VUPATH=/opt/vuport/lib
```

2.4.8 VUPSW

The VUPSW environment variable is used to set the VUPort runtime switches. By default, all the VUPort runtime switches are off. If you wish to turn on one or more switches, you must include the switch with a + sign before it in the VUPSW environment variable. The following are the possible values options which can be set using the VUPSW environment variable.

- +L Force all filenames and library names to lowercase.
- +M Causes VUPort not to automatically set a new files security access rights to 666. When set new files will be created with the security access rights specified with the UNIX umask setting.
- +R Causes the procedure interpreter to save the screen it is displaying prior to each program being run, then redisplay the procedure screen after it completes. The net effect of this option is that the procedure interpreter will redisplay the screen it believes to be on when the procedure it is executing uses the ERASE=NO option in MESSAGE or PROMPT statements.
- +S Causes the VUPort command processor to suppress any status message displays. This option works the same as the Security program option, but will not override the security option to suppress status messages.
- +U Force all filenames and library names to uppercase.

Example

```
export VUPSW=+U+M
```

2.5 VUPort Usage Constants

When starting the VUPort command processor, or any of the utilities in stand alone mode, your usage constants are initialized from your environment variables. The following is a table listing each of the usage constants which can be initialized through your environment variables.

Table 3 Usage Constants

INLIB	INVOL	JOBCLASS	JOBPRIO
OUTLIB	OUTVOL	PRINTER	PRNTMODE
PROGLIB	PROGVOL	PRTCLAS	PRTFORM
RUNLIB	RUNVOL	SPOOLIB	SPOOLVOL
TIMEOUT	WORKVOL		

Example:

```
Export PRINTER=1
```

2.6 System Initialization

In order for VUPort to operate, you must have the VUPort control daemon running. This control daemon is called **ksactl**. When the control daemon is first started it reads the config file which is created using the GENEDIT utility. All active devices are brought on line.

The ksactl program has the following usage.

Usage : ksactl [-l <log file>] [-s] [-h]

-l <log file> Place all console messages to the log file

-s No start up processing.

-h Show usage

When using the -l <log file> option, all messages ksactl normally puts on the system console, will be put in the specified logfile.

When using the -s option, ksactl will start without configuring predefined devices defined in the config file.

Using the -h option, ksactl will only show the usage message telling you how you can invoke ksactl.

Once you have started ksactl, your system will be ready to use. The VUPort program ksactl will remain active in the background.

2.7 System Termination

Prior to shutting down your UNIX machine, it is important that you terminate all running applications and stop the VUPort control daemon. A command line utility called **ksastop** is provided to stop the VUPort control daemon, but it is your responsibility to make sure all users logged into the system have been logged off.

The ksastop program has the following usage:

```
ksastop [-f]
```

-f Force all printers and initiators to be released

2.8 STARTING THE COMMAND PROCESSOR

The VUPort Command Processor is started by running the program ksamain. In order for the VUPort Command Processor to operate correctly, the users environment must be set up before it is invoked. In order to set up the users environment, you can set all the necessary variables in the users UNIX login script (.profile), or use the ksalogin program to automatically set the users environment variables, and then invoke ksamain.

In order to control the initial actions of ksamain, the program has the following options.

Usage: ksamain {-i | -u | -f <filepath>}

-i = Run \$HOME/.ksarc

-u = Run Procedure specified by SECURITY

-f = Run Procedure specified by <filepath>

If you start km using the -i option, ksamain will look in your home directory for a file called .ksarc. If it finds this file, ksamain will directly execute the procedure found in the file .ksarc.

If you start km using the -u option, ksamain will lookup in the userlist file, what your startup procedure is, registered using the SECURITY program. If a startup procedure is defined, ksamain will directly execute the specified startup procedure.

If you start km using the -f <file path> option, ksamain will start directly the specified program. Make sure you specify a absolute file path, otherwise ksamain will not be able to find the file.

2.9 NATIVE LANGUAGE SUPPORT

Using VUPort the data in your data files is still stored in Wang VS format. The consequence of this is that special characters that are input from the keyboard first have to be converted to Wang VS format and that WANG VS special characters needs to be converted before displayed on the screen or printed on the printer. To take care of this conversion VUPort supports a form of Native Language Support. To use Native Language Support you have to create a special directory called `nls` in the `$VUPATH` directory. First we discuss what must be done for terminals and then what must be done for printers.

Terminals

For every terminal type there must be two files in the `nls`-library, These are the `$TERM.in` and the `$TERM.out` files. `$TERM` must be replaced by the value of the `TERM` environment variable for that terminal.

The `$TERM.in` file takes care of the conversion of special characters as they come from the keyboard to the Wang VS format in which they are stored. A line in this file must look like this:

```
\low  \Wanglow\Wangup
```

or

```
\up   \Wangup\Wangup
```

In this example `low` stands for the value generated by input from the keyboard with a lowercase special character as input. `Up` stands for an uppercase special character generated by input from the keyboard. `Wanglow` is the lowercase Wang VS value for the special character and `Wangup` for the uppercase value of it. All values must be inserted in octal format.

The `$TERM.out` file takes care of the conversion of Wang VS special characters as they are stored in the data files, to the format in which they are displayed on screen. A line in this file must look like this:

```
\Wanglow  \low
```

or

```
\Wangup   \up
```

For this file the same applies as described above for the `$TERM.in` files. Remember that the values must be in octal.

Printers

For printers only a **.out** file has to be made where the name of the file doesn't matter. This file is then used in conjunction with the xlp program. You can use xlp with the following command:

```
xlp <map file> <command> [options]
```

xlp is a program and uses the standard input as input and searches for characters specified in the map file, and globally replace these characters with there new values. In this command the map file is the **.out** file and instead of <command> a print command can be inserted (with some arguments, if needed). In VUPort this command can be inserted where printers are attached. You should enter this command where normally the device name must be entered. For example:

Examples

```
!/bin/xlp printer.map lpr -P1 -h
```

```
cat infile | xlp test.map cat - > outfile
```

The exclamation mark indicates that a UNIX program must be executed when something must be printed.

3 Command Processor

3.1 Running Programs

Executing programs from within the VUPort System is accomplished by pressing pfkey 1 from the VUPort Help Menu. The screen displayed will require the entry of a RUNPROG, and optionally a RUNLIB and RUNVOL. The RUNPROG is the name of the file you wish to run. This file can be an executable file, an 'a.out' format, a VUPort Procedure file or a Cobol Object File.

If you wish to execute a program in a specific Library and Volume, you need to specify all three parameters on this screen. If the program you wish to execute is a VUPort Utility (e.g. ksaforms) or the program is located in the library called 'library' on the volume **SYSTEM**; you need only enter the file name. The system will automatically look in the library named 'library' on the volume named **SYSTEM** for the file.

In addition to searching for a program or procedure file in the system library, the VUPort System will also attempt to locate the file by appending the appropriate Cobol object file extensions to the name. After attempting to locate the file by the name given, the system will append the following suffixes to the name and then repeat the search sequence for each newly formed name.

Example:

```
Cobol Object File Suffixes: .GNT, .gnt, .INT,  
.int, .COB, .cob
```

Once a program is started, pressing the 'delete' or 'break' key on the terminal can interrupt it. This will temporarily stop the program and return to the VUPort Help Processor. At that point, the option to run a program is no longer available. The option has been changed to **Continue Program**.

It is possible also to turn the ability to interrupt a program off by pressing pfkey 1 to complete the Run Screen and start the execution of the program. This will disable the 'Break' or 'Delete' keys from interrupting the program. See the chapter on VUPort Procedures 'HELP OFF', for more information on disabling the interrupt key.

The VUPort System also provides a method of entering the Unix shell. By pressing pfkey 6 from the Help Menu, the system will execute a shell. The user will then be able to run shell commands from typically the '\$' prompt. By default, the shell created is the Unix Bourne Shell. Setting an environment variable called 'SHELL' to point to an alternative shell can change this. One alternative shell is the Berkeley 'C' Shell,. To execute this shell instead set the SHELL environment variable to **/usr/ucb/csh**.

Example:

```
SHELL=/usr/ucb/csh
export SHELL
```

When entering commands at a shell prompt, you can return to the Help Menu without terminating the shell. To do this you need to execute the program 'kh'. This will display the Help Menu and still keep the shell active. To return to the shell, press the pfkey 1. NOTE: The shell program cannot be canceled, as can other programs.

3.2 Window Control

The VUPort System controls up to three application windows. A window is simply a set of programs attached to a terminal. With three windows you can have up to three sets of programs running on one terminal. Pressing pfkey 10 from the VUPort Help Processor enters window control. At this point you see a menu of all the windows active or not, on your terminal. The window number is the number of the window from 0 to 2. The program is the current program in the window. This is the program that has control over the terminal. Below is a list of the statuses of the window and their meaning.

Table 4 Window Statuses

STATUS	MEANING
ACTIVE	This is the current window, but no program is currently running.
CANCEL	An attempt has been made to cancel the active program in this window.
DEAD	The active program has died and the system is in the process of returning control to the program that has placed the link, or to the VUPort Help Processor.
HOLD	The active program is on hold, waiting for the user to reattach to it.
INTRPT	The active program in the current window has been interrupted because the user has pressed the delete or Break key.
LINK	The window is in transition between a program that has placed a link request and the new program.
NEW	This is a new window that has just been opened and no program has been run from it.
SHELL	The window has entered a Shell, and then requested help by running the program 'kh'.
UNAVAIL	This window is not available because the physical terminal does not support enough pages of memory.
UNUSED	This window is available but is currently not being used. It is closed.

The process number and comment fields are used to supply the user with information regarding the Unix Process Id, and state of the processes.

When the VUPort System is started, window zero is the active window. To open a new window, press pfkey 1 from the Window Processor Menu. If an additional window is available, the current window will be placed into the 'hold' status, and the next available window number will become the current window. The program then returns to the VUPort Help Menu.

To reattach to a window, position the cursor in front of the line of the window you wish to reattach to and press pfkey 2. The current window will be placed into the 'hold' status, and this window will become the current window. The window will return to the status it was prior to being placed on hold.

When more than one window is open, pfkey 16 on the Help Processor menu has a special meaning. By pressing pfkey 16 from the Help Processor when no program is active in the window, the window will be closed, and the next open window will become the active window. When all but one window is closed, pressing pfkey 16 will terminate the VUPort System's program.

When a program is active in the current window, pfkey 16 will attempt to cancel the running program. The system will then return to the program that called the program that was canceled. Note: It can be dangerous to cancel programs that have files open. Data Loss can occur!

During program execution, the user can print the program screen by interrupting the program and then pressing pfkey 14. The difference between the pfkey 14, and pfkey 15 is that, pfkey 15 prints the screen being displayed by the VUPort System's Help Processor while pfkey 14 prints the screen being displayed by the application program.

3.3 Manage Files

By pressing pfkey 5 from the VUPort Help Processor, the program enters the Files Display sub-menu. This portion of the program allows you to display the names and other information on the files located on your system. The first screen contains a list of all the Volume names currently mounted on your system. The top line of the screen also has a location to enter a volume name, library name, and/or a file name. The volume and library names will default to the values in INLIB and INVOL. By entering a volume name only, or by leaving the VOLUME name blank and placing the cursor in front of the volume you wish to choose, and pressing the execute key, the system will display all the libraries on that volume. If entered, the volume name must be exact.

By entering a volume and library name, the system will display all the files in the library you have entered. If the library does not exist on the volume, the system will respond by displaying the list of libraries on the volume starting one name (alphabetically) after where library would have been had it existed. The same concept holds true for files. Entering a volume and library name, and a name of a nonexistent file (or a partial name) will cause the system to display all the files in the library starting one name after to the name entered.

If you are displaying the list of libraries or a list of files you can scroll through the names by using the following pfkeys.

Table 5 File Management Function Keys

PFKEY	FUNCTION
2	Display the list of names at the beginning.
3	Display the list of names starting on the last screen.
4	Display the previous screen of names.
5	Display the next screen of names.

Pfkey 6 allows you to rename a library or file. By positioning the cursor in front of the name and pressing pfkey 6, you will be allowed to enter the new file/library name. To accept the new file/library name press the execute key. To cancel the rename request press pfkey 16.

Pfkey 7 allows you to delete a library of file. By positioning the cursor in front of the name and pressing pfkey 7, you will be asked to confirm the delete request. By pressing the execute key, the system will remove the file or library. **Deleting a library causes all the files in the library to be deleted.** To cancel the delete request press pfkey 16.

To display the detailed information on a file, position the cursor and press the execute key. The system will display the information about the file. This information includes; the size of the file, its access permissions, and the dates when it was created, last accessed, and modified.

3.4 Manage Print Queue

To enter Queue management press pfkey 3 from the Help Menu. Pressing pfkey 2 will then display the Print Queue. This menu contains all the documents in the system print queue. The printouts are ordered in two groups. At the top of the display are all the printouts that are currently printing or are queued to print. The remaining entries are the printouts not released for printing. Printouts owned by you are highlighted for easier reading. These are the only queue entries you can change unless you are the superuser. The superuser's queue entries will be highlighted for the superuser, but he can change any of the print queue entries. The superuser is a user with the numerical user id of 0. (Typically login id 'root')

You can use the pfkeys to scroll through the display. They are the same keys used throughout VUPort. Use pfkey 2 to display first, 3 for last, 4 for previous, and 5 for next.

The display contains all the information on each printout. 'Ptr' is the printer number the printout will be printed on. File, Lib, and Vol are the File name, library and volume where the print file resides. Pages is the number of pages in the document. See the chapter on Misc Utilities 'lpr' for more information on the page count. The copies column can be set by the user to indicate the number of copies of the document to be printed. Form is the form the printout is to be printed on. See the chapter on **'ksaforms'** for additional information on the print form. Class is Class the class the printout is assigned to. This is used to determine the order the printout is to be printed. The 'Disp' is the disposition of the print file once it has been printed. The options are 'D' to delete the file after printing, 'R' to requeue the file and print queue. The owner is the name of the user that submitted the printout. Finally the status is the current state the printout is in. Below is a table of all the statuses and their meanings.

Table 6 Print Queue Statuses

STATUS	MEANING
CANLD	The printout was canceled by the operator and placed back into the print queue.
HOLD	The printout has been placed into the print queue with a print mode of hold. It has not been released or printed yet.
PRINT	The printout is currently printing.
PRNTD	The printout has been placed back in the queue after it has been printed. This happens after it has been printed with a disposition of 'R', Requeue.
QUED	The printout is released to print but the printer it is assigned to is currently printing something else.

Positioning the cursor in front of the line you wish to change and pressing pfkey 10 can change any parameter of a print queue entry. You cannot change printouts that have the status of qued or print. This is because they are active printouts. Only inactive printouts can be changed. If a printout has a status of qued, you must first place it on hold, by using pfkey 8. You may then change its parameters. The parameters you can change are the printer number, the disposition, the class, the form, and the number of copies. Pressing the execute key will confirm the changes. Pressing pfkey 16 will cancel the change request.

Inactive printouts can also be removed. By removing the print queue entry you will delete the print file. Position the cursor in front of the printout you wish to remove and press pfkey 9. The computer will request you to confirm the delete request. Press the execute key to confirm the remove request. Press pfkey 16 to cancel the delete request.

Finally you can display the printout by positioning the cursor in front of the print entry and pressing return or pfkey 11. This will execute the display utility and display the print file. NOTE: The help key is always disabled while you are displaying the print file in this manner.

3.5 Printer Control

By pressing pfkey 11 from the Help Menu, you enter the operator mode. From this menu you can control the printers by pressing pfkey 2. The system will display all the printers attached to the system and a summary of each printer's status. A maximum of ten printers can be attached to a system at any one time.

The printer number is displayed in the first column of the screen. The currently loaded form is displayed next. The status of the printer is then displayed. Below is a table summarizing all the statuses of a printer.

Table 7 Printer Statuses

STATUS	MEANING
PRNT	The printer is currently busy printing a file. The file name and current page number will be displayed.
IDLE	The printer is currently not doing anything.
HELP	The printer is in trouble and requires help. When the printer is placed under control, a message detailing what type of help it needs will be displayed.
CONT	Another operator is currently controlling the printer.
DEAD	The printer has encountered a fatal error and is no longer attached. The system is requesting acknowledgment of the death of the printer. This usually occurs during printer initialization and is possibly caused by a hardware or system error.

If the printer is idle, the last column on the display shows the device to which the printer is attached. If the printer is dead, the reason for its death will be displayed in this column. If the printer is currently attempting to print a file, the file name and the current page it is printing will appear here.

To attach a new printer to the system, press pfkey 6 from the Printer Control Menu. The computer will request the printer number, the initial form the printer has loaded, and the device where the printer is attached. After entering this information, press the execute key to complete the printer attach. Press pfkey 16 to cancel the attach request. See the chapter on VUPort Procedures **ATTACH** for an alternative method of attaching printers to the system.

Releasing a printer, in essence, logically removes the printer from the system. The VUPort System will no longer be able to spool printouts to this printer. To do this, position the cursor in front of the printer line you wish to release and press pfkey 7. The computer will request a confirmation for the release request. Press the execute key to confirm the request or pfkey 16 to cancel the request. Only printers with a status of idle or dead can be released from the system.

As long as a printer is alive you can control it, by positioning the cursor in front of the printer line you wish to control and pressing pfkey 9. The computer will stop the printer. If the printer is currently printing a file, the buffer in the printer might need to be drained before the printer actually stops printing.

If the printer was printing a file, you can restart the printout at any page by supplying the page number and pressing pfkey 2. You can also cancel the printout by pressing pfkey 7. This will take the printout and place it back in the print queue with a status of **cancl'd**. In either case, if the operation was successful, the computer will return the display back to the printer control menu.

When a printer is under control you always have the ability to load a form and print form alignments. By supplying the new form number and pressing pfkey 9 the computer will load a new form on the printer. Form alignments are printed by pressing pfkey 10. See **ksaforms** for more information on form alignments. To continue printing a file where it was stopped or to exit from the control printer screen press pfkey 1.

3.6 Device Control

By pressing pfkey 3 from the operator menu, you can control devices. Device control is where you can mount and dismount volumes. The computer will display the volumes currently mounted on the system. At this point volumes can be mounted and dismounted. To mount a volume, press pfkey 6. The system will prompt you to enter the volume name, and a Unix path name. Press the execute key to confirm the mount request. Pressing pfkey 16 will cancel the mount request.

Positioning the cursor in front of the volume you wish to dismount and pressing pfkey 7 does dismounting volumes. The computer will ask for a confirmation of the dismount request. Press the execute key to confirm the dismount request. Press pfkey 16 to cancel the dismount request.

Dismounting a volume makes the volume inaccessible through the VUPort System. A special volume called **SYSTEM** is usually mounted on the system. This volume is where many of the system utilities are located. Do not dismount this volume unless you know why you need to do so.

4 Procedure Language

4.1 Introduction

The VUPort Procedure language is an interpretive language used to control the KSA Systems environment. Procedures can be run directly by the KSA Systems Help Processor or by invoking the procedure interpreter with the name of the procedure file from the shell. When invoking the procedure interpreter from the shell the procedure file is specified using the `-i` argument.

Example:

```
ksaproc -i /etc/ksarc
```

Each procedure file is free format, with the exception of the following general rules. The procedure file is a line sequential file made up of lines with no more than 72 characters. It can be created and edited with the system editor. Each line contains zero or more words. Blank lines are allowed. White space characters separate words. White Space characters are one or more spaces or tabs. A new line character terminates lines. Lines starting with a '*' are considered commentary. In-line comments are started with a '[' and ended with a ']'. Nested in-line comments are not allowed.

The first four non-white space characters in the procedure file must be **proc** or **PROC**. The next word is considered the procedure name. The remaining part of the line must be blank. Comment lines are allowed before the procedure header. A sample procedure header is:

Example:

```
PROCEDURE ARAPP
```

All words are converted to upper case unless quoted by ' or ". Keywords are always upper case. Variable and label names are not case significant. Variable names must start with a **&**. Label declarations must be in column one and have a ':' at the end.

Constants are numeric, alphanumeric or Boolean. The procedure interpreter will convert the value to the appropriate type depending on the context of its usage. Numeric values are integer values in the range 0 to 2,147,483,647. Alphanumeric values are upper and lower case letters, digits, or special characters including '!', '@', '#', '\$', '%', '^', '*', '-', '_', '{', '}', ':', ';', ',', '.', '|'. Substrings can be created by using variables with a substring expression at the end.

Format:

& variable (start[, length])

Start is one relative and is the starting character in the variable to be used. Length is optional. It is the number of characters to be included in the substring expression. If length is not given, the remainder of the variable is used. Sub strings can only be used with string variables.

Boolean values can be alphanumeric or numeric. A transient value of 0 or spaces is considered false. Everything else is considered true. Besides transient values, booleans can be created through logical expressions. Below is a table of the expressions available. They are listed in order of priority. Note: Parenthesis always takes precedence over built in priority.

Table 8 Logical Expressions

EXPRESSION	MEANING
+ - * /	Binary Arithmetic
=	Equal to
<>	Not equal to
>,LT	Less Than
>,GT	Greater than
<=,LE,NGT	Less than equal to, or Not greater than
>=,GE,NLT	Greater than equal to, or Not less than
NOT	Negation
AND	Logical and
OR	Logical or
!!	Concatenation, also known as BANG BANG .

File names are given in either directory path format or File/Lib/Vol Path. Directory paths are Unix path names. They must start with a '/' and can contain, upper and lower case characters, digits, '.' and optionally the library and volume.

Format:

filename [IN library [ON volume]]

If the library and/or the volume is not supplied, one of the usage constants will be used depending on the statement the file name reference is located in.

A backward reference enables a procedure statement to obtain some or all of its required parameters from a preceding, previously executed DISPLAY or enter clause. A backward reference is specified by enclosing in parenthesis the label of a DISPLAY or ENTER clause optionally followed by a field name. Values that are generated by backward referencing have the format:

Format:

(label[,field])

4.2 Assign Statement

Format:

ASSIGN & variable = expression

The assign statement places a value into a variable. The variable can be a full variable or a substring expression. The expression type is dependent on the type of the receiving variable. Conversion occurs automatically. The assigned value will be truncated to the length of the receiving field.

Example:

```
ASSIGN &ivar1 = &ivar2 + 20
ASSIGN &svar = "Hello"
```

Has the same effect as:

```
ASSIGN &svar(1,2) = "He"
ASSIGN &svar(3,3) = "llo"
```

4.3 Attach Statement

Format:

```
ATTACH printer ON path [FORM number]
```

The attach statement logically attaches a printer to the system. The printer must be a number in the range 0 to 99. The path must be a directory path of an existing character special file. If FORM is given, number is the initial form that is loaded on the printer. If FORM is not given, the default form of 0 is used.

Example:

```
ATTACH 1 ON /dev/lp01  
ATTACH 1 ON /dev/tty/01 FORM 100
```

4.4 CALL Statement

Format:

[label :]CALL target - label

The CALL statement transfers control to a specified label. The target of a call must be a string expression corresponding to a label defined within the procedure. Rules for calling a label when the procedure contains more than one occurrence of the label are the same as for the GOTO statement. Calls can be nested. The maximum nesting level is 16. If no label exists, an error occurs. If an END statement is executed when there is no corresponding call, an error occurs.

Example:

```
PROCEDURE
CALL X
CALL Y
RETURN
X:   PROMPT `SUBROUTINE X' END
Y:   PROMPT `SUBROUTINE Y' END
```

4.5 Declare Statement

Format:

```
DECLARE & var [, & var] [GLOBAL] STRING(number) [INITIAL expression]
                              INTEGER
```

The declare statement declares one or more variables. Every variable must be declared prior to using it in the procedure file. The scope of the variable is within the procedure file in which it has been declared. If GLOBAL is specified, the scope extends to include any procedure file invoked from that point, until the file which declared it exits.

String variables are alphanumeric variables with a length of number. If the initial statement is supplied, the value of expression is assigned to the variables. If the initial statement is not given, string variables are initialized to spaces, and integer variables are initialized to zero.

There are two built-in variables called &DATE &TIME. &DATE is the system date, and is in the format YYMMDD. &Time is the systems time and is in the format HHMMSS.

Example:

```
DECLARE &svar STRING(15) INITIAL "abcdefghijklmno"
DECLARE &ivar1,&ivar2 INTEGER
```

4.6 Dismount Statement

Format:

DISMOUNT		DISK		expression
		TAPE		
		DIR		
		DIRECTORY		

Dismount logically dismounts a mounted volume. When the dismount statement is processed, the volume is logically removed from the system. DISK, TAPE, DIR and DIRECTORY are commentary and optional. Expression is the name of the volume to dismount. The name will be converted to upper case. If the volume is not mounted, or the system is unable to dismount the volume, an error message will occur.

Example:

```
DISMOUNT VOL111
DISMOUNT TAPE 'TAPE1'
```

4.7 Destroy Statement

Format:

```
[label :].DESTROY PROCEDURE
```

The DESTROY statement closes the currently executing procedure file, attempts to scratch it, and returns control to the invoking routine. The currently executing procedure file is closed, an attempt is made to scratch the procedure file, and control is returned to the invoking routine (which can be a program, a procedure₁ or the command processor). The return code passed back to the invoking routine is that resulting from the scratch. label: is a string of one to eight alphanumeric characters that must begin with an alphabetic character and be immediately followed, with no intervening spaces, by a colon. The label may be the target of a GOTO or CALL statement. The DESTROY statement does not assign a return code to the label.

Example:

```
PROCEDURE TEST TO BE DESTROYED AFTER EXECUTION  
RUN TEST IN WORKLIB ON FLOPPY DESTROY PROCEDURE
```

4.8 Extract Statement

Format 1:

EXTRACT & var = keyword[,&var = keyword]

Format 2:

EXTRACT & var =

BLOCKS ALLOCATED FOR	filename	[IN libname	[ON volname]
RECORDS USED BY	<i>path</i>			

Format 3:

EXTRACT & var =

[NEW]PATH OF filename IN libname ON volname						
[NEW]PATH OF libname [ON volname]						
PATH OF VOLUME volname						
PATH OF <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="padding: 5px;">FILE</td> <td style="padding: 5px;"> </td> <td style="padding: 5px;">pathname</td> </tr> <tr> <td style="padding: 5px;">DIRECTORY</td> <td style="padding: 5px;"> </td> <td></td> </tr> </table>	FILE		pathname	DIRECTORY		
FILE		pathname				
DIRECTORY						
PATH OF TAPE volname						
NAME OF TAPE tapedev						

The extract statement assigns system information to a variable. 'Var' can be a variable or a substring. The expression type is dependent on the type of receiving variable. Conversion occurs automatically. The length of the expression is truncated to the length of the receiving field.

The format 1 extract statement is used to extract the value of a usage constant or another value within the users environment. Format 1 extract statements are only valid when the procedure is running from the VUPort Help Processor. Unpredictable results will occur if a format 1 extract is executed when the procedure interpreter is in a stand-alone mode. Below is a list of all the valid keywords.

Table 9 Extract Keywords

FILECLAS	FORM#	INLIB	INVOL
JOBCLASS	JOBLIMIT	JOBQUEUE	LINES
OUTLIB	OUTVOL	PRINTER	PRNTMODE
PROGLIB	PROGVOL	PRTCLAS	PRTFCLASS
PRTFORM	RUNLIB	RUNVOL	SPOOLIB
SPOOLSYS	SPOOLSYS	SPOOVOL	SYSLIB
SYSVOL	TASKTYPE	USERID	USERID
USERNAME	USERNAME	WORKLIB	WORKVOL
WS			

The format 2 extract statement is used to extract information associated with a file. In this statement 'var' is considered to be an integer variable. The BLOCKS ALLOCATED FOR statement assigns to 'var' the number of full 1024 byte blocks used by the file. The RECORDS USED BY statement assigns to 'var' the number of bytes in the file.

If file name is given, but libname or volname is not, the system will default them to OUTLIB and OUTVOL. If path name is given, it must be a complete Unix path. In either case, if the file is not found, an error will occur.

Example:

```
EXTRACT &svar1 = INLIB, &svar2 = INVOL
EXTRACT &ivar = BLOCKS ALLOCATED FOR
datafile IN aplib ON VOL111
EXTRACT &ivar = RECORDS USED BY /tmp/motd
```

The format 3 will extract statement is used to extract absolute path of the file, library and volume. The NEW option will create the file specified. The DIRECTORY option will verify the existence of the absolute path given. The TAPE option extracts the device path name associated with the TAPEDEV. Lastly the "NAME OF TAPE" extracts the device number associated with the tape volume.

Example:

```
EXTRACT &FPATH = PATH OF &FILE1 IN &LIB1 ON &VOL1
```

4.9 Go to Statement

Format:

GOTO label

The go to statement will transfer control of the procedure to a different part of the procedure.

Example:

```
GOTO endproc
```

4.10 Help Statement

Format:

```
HELP |ON |  
     |OFF|
```

The help statement is used to turn help processing on and off. When help is on, any programs or procedures executed from this procedure or any subsequent procedures from this point upward can be interrupted. When help is off, the programs and procedures executed cannot be interrupted. Help will be turned on when the procedure file terminates. This statement will have no effect if the procedure file has not been executed directly or indirectly through the VUPort Help Processor.

Example:

```
HELP OFF
```

4.11 If Statement

Format 1:

IF expression [THEN] *anystatement*

Format 2:

IF [NOT] EXISTS

FILE	[IN libname	[ON volname]]
LIBRARY	libname	[ON volname]
VOLUME	volname	

 [THEN] *anystatement*

The if statement is used to control the execution of the procedure based on a conditional statement. The **THEN** keyword is commentary.

The format 1 if statement will execute any statement when expression evaluates to true. The expression can be made up of Boolean or logical expressions. Boolean expressions are variables or constants that evaluate to either true or false. A value zero or spaces is considered false. Everything else is considered true. Logical expressions are made up of one or more relational operators. Below is a table of the available relational operators and their meaning. The symbol and the abbreviation are equal in precedence and meaning.

Table 10 Logical Expressions and Abbreviations

SYMBOL	ABBREVIATED MEANING
> G	Greater than
>= GE, NLT	Greater than or equal to (Not less than)
< LT	Less than
<= LE,NGT	Less than or equal to (Not greater than)
= EQ	Equal to
<> NEQ,NE	Not equal to

Any relation can contain the **NOT** keyword. This is used to negate the value. Any number of relational expressions can be combined by using the **and/or** operators. The relational expressions can also be grouped by using '(' and ')'. These always take precedence over the operators.

The format 2 **'if statement'** is used to determine if a file, library or volume exists. The any statement is executed if the statement evaluates to true. If libname or volname is not given, the system will default to INLIB and INVOL.

Example

```
IF &pfk = 5 THEN RUN "display"
```

```
IF ((&DATE(1,2) = "12") AND (&DATE(3,2) = "25"))  
PROMPT CENTER BLINK "Merry Christmas"
```

```
IF EXISTS FILE "lockdata" IN "apwork" ON VOL222  
THEN SCRATCH "apwork" ON VOL222
```

4.12 Logoff Statement

Format:

LOGOFF

The logoff statement is used to terminate the procedure and exit the VUPort Help Menu. When executed the procedure will exit and return to the VUPort Help Processor. The VUPort Help Processor will then exit. This statement can only be executed in procedures executed from the VUPort Help Processor. An error will occur if more than one window is open when this statement is executed.

Example:

```
LOGOFF [ Log off the system when done ]
```

4.13 Message Statement

Format:

MESSAGE [option[, option]][ROW expression][line][; line]..

The message statement is used to display messages on the screen. Up to twenty-four lines can be displayed with one statement.

One or more message options can be given. A comma must separate them.

Format:

$$\text{ALARM} = \begin{array}{|c} \text{YES} \\ \text{NO} \end{array}$$

$$\text{ERASE} = \begin{array}{|c} \text{YES} \\ \text{NO} \\ \text{FORCE} \end{array}$$

If the alarm keyword is given, the terminal bell is sounded if the '= YES' statement is used. Using 'ALARM = NO' and leaving the ALARM statement out have the same effect.

If the erase keyword is given, the terminal display is erased if the '= YES' statement is used. If the '= NO' statement is used the terminal display is not erased before the message is displayed. Giving 'ERASE = YES' and leaving the ERASE statement out have the same effect.

If the ROW statement is given, the expression is considered numeric. The message lines will start at the row number expression where expression is in the range 1 to 24. If the ROW statement is not supplied, the procedure will center the display on the terminal based on the number of lines given to display.

One or more lines can be supplied. Each line is separated by a '; . A ;' with no fields interpreted as a blank line.

Format:

[CENTER]field[, field[, field]..]

When the CENTER keyword is given, the line is centered on the screen. If it is not given the procedure interpreter will start the line display at column two on the terminal.

Each field can have zero or more attributes associated with it. If no attributes are given, the field will be displayed in BRIGHT mode. If more than one attribute is supplied, they must be separated by a comma ','.

Format:

[attribute[,attribute...]]expression

The available field attributes are: BLANK, BLINK, BRIGHT, DIM and LINE. The expression can be a variable, a substring, or a constant. Numeric variables and constants are displayed with a length of 11. Alphanumeric (string) variables, constants, and substring are displayed with a length equal to that of the variable, constant, or substring. Each field in the line has one character reserved for the attribute whether it is supplied or not.

Example:

```
MESSAGE ROW 12 "HELLO LINE 12";
      CENTER "HELLO CENTER LINE 13"

MESSAGE CENTER BLINK, LINE
      "This is blinking and underlined"

MESSAGE CENTER BLINK "This line is blinking";;
      DIM "Then this line two lines down"
```

4.14 Mount Statement

Format:

MOUNT		DISK		volname ON path
		TAPE		
		DIRECTORY		
		DIR		

The mount statement is used to logically mount a volume. The keywords DISK, TAPE, DIR, and DIRECTORY are commentary. The volname is the name of the volume to mount. Path must be a Unix path of an existing directory. If successful, the volume will be mounted for global use within the VUPort System.

4.15 Options Statement

Format:

$$\text{OPTIONS PROCMSG} = \begin{array}{|l} \text{YES} \\ \text{NO} \end{array}$$

The options statement is used to set options within the Procedure interpreter and VUPort environment. Currently only the 'procmsg' option available. The 'procmsg' statement controls the displaying of the program name currently being executed. By default it is 'on'. When 'on' the VUPort System will display the name of the currently executing program or procedure. When "off" the system will run all the programs without displaying the "Program in progress" message. When the option is changed, the new value stays in effect until changed by the same procedure or a different procedure.

Example:

```
Turn the Program in progress message off
OPTIONS PROCMSG = NO
```

4.16 Print Statement

Format:

$$\begin{aligned} & \text{PRINT filename [IN libname] [ON volname] [, CLASS = prtclass]} \\ & \left[, \text{COPIES} = \text{integer - expression} \right] \left[, \text{STATUS} = \left\{ \begin{array}{l} \text{SPOOL} \\ \text{HOLD} \end{array} \right\} \right] \\ & \left[, \left\{ \begin{array}{l} \text{DISPOSITION} \\ \text{DISP} \end{array} \right\} = \left\{ \begin{array}{l} \text{SCRATCH} \\ \text{REQUEUE} \\ \text{SAVE} \end{array} \right\} \right] \end{aligned}$$

The PRINT statement places a print file in the print queue. CLASS, STATUS, FORM#, COPIES and DISPOSITION can be specified in any order, and each may be specified only once. The PRINT statement places the file identified by filename IN libname ON volname in the print queue with the specified options.

If no library name for the print file is specified SPOOLIB is used. If no volume is specified, SPMOLVOL is used. CLASS specifies the class to which the print request is to be assigned. This value must be A through Z. It can be specified as a constant, a variable, a partial backward reference, or any other string expression.

If CLASS is not specified, the value of PRTCLASS from the SET Usage Constants function from the command processor is used.

STATUS specifies when the file is printed. SPOOL indicates that the file is printed as soon as the designated printer is available. HOLD places the file on the print queue, but it does not print until you release it.

FORM# specifies the printer form number. The value of the integer expression must be in the range 0 to 254. If FORM# is not specified, the value of FORM# from the SET Usage Constants function from the command processor is used.

COPIES specifies the number of copies to be printed. The value of the integer expression must be in the range 1 to 32,767. If COPIES is not specified, one copy is printed.

DISPOSITION specifies what happens to the file after it is printed. SCRATCH deletes the files from the library; REQUEUE places the file at the bottom of the print queue; SAVE retains the file within the library, but does not requeue. The default is SCRATCH.

Example:

```
PROC  
PRINT REPORT IN #GSPRT, CLASS A, COPIES = 3
```

4.17 Prompt Statement

Format:

```
PROMPT [option[,option]..] [ROW expression] [line[:line]..]
```

The prompt statement is used to display information to the terminal and allow the user to enter information. The procedure will display the message to the terminal then wait for the user to press the execute key or any pfkey.

The prompt option has the same options as the message statement and a few more.

Format:

```
PFKEY = variable
CURROW = variable
CURCOL = variable
ALARM = |YES|
        |NO |
ERASE  = |YES|
        |NO |
```

Zero or more options can be supplied in the prompt statement. They must be separated by a comma ','. The PFKEY option tells the procedure to assign the number of the pfkey pressed by the users to the variable given. The values are 0 to 32 where 0 is assigned if the execute key is pressed, and 1 to 32 is used to correspond to the function key number.

The CURROW option tells the procedure to assign the row number location to the variable at the time the screen is terminated with either the execute or a pfkey. The value of the variable is also used as the initial cursor location. The values are 1-24 corresponding to the 24 lines on the terminal.

The CURCOL option tells the procedure to assign the column number location to the variable at the time the screen is terminated with either the execute or a pfkey. It is also used as the initial cursor location. The values are 1 to 80 corresponding to the 80 columns on the terminal. Both the CURROW and CURCOL options are handy to design procedures that use the point and press method of determining which actions to take.

The ALARM and ERASE options have the same meaning in the Prompt statement as they do in the Message Statement.

If the ROW statement is given, the expression is considered numeric. The Row prompt lines will start at the row number expression where expression is in the range 1 to 24. If the ROW statement is not supplied, the procedure will center the prompt lines on the terminal based on the number of lines given to display.

One or more lines can be supplied. Each line is separated by a ';', and a ';' with no fields considered a blank line.

Format:

```
[CENTER] field[,field[,field]..]
```

If the CENTER keyword is given, the line is centered on the screen. If it is not given the procedure interpreter will start the line display at column 2 on the terminal.

Each field can have zero or more attributes associated with it. If no attributes are given, the field will be displayed in BRIGHT mode. If more than one attribute is supplied, they must be separated by a comma ','.

Format:

```
[attribute[,attribute..]]expression
```

The field attributes are broken into two types. The first type is the attributes to specify how the field is to be displayed. The valid attributes are BLANK, BLINK, BRIGHT, DIM, and LINE.

The second type of attributes is used to specify what type of input is allowed in the fields. The valid attributes are NUMERIC, UPLOW, and UPPER. If the field is to be modifiable, the expression must be a variable name (or sub-string) and the attributes must contain at least one of the three modifiable attributes.

The expression can be a variable, sub-string or constant. Numeric variables and constants are displayed with a length of 11. Alphanumeric (string) variables and constants, and sub strings are displayed with a length equal to that of the variable, constant or sub-string. Each field in the line has one character reserved for the attribute whether it is supplied or not.

Example:

```
PROMPT ERASE = NO, PFKEY = &I ROW 20
      'Procedure Complete Press Return'
PROMPT ALARM = YES ROW 15
      CENTER BRIGHT 'Please enter your name:',
      DIM,LINE,UPLOW &S
```

4.18 Rename Statement

Format 1:

```
RENAME filename [IN libname [ON volname]]  
          TO filename [IN libname]
```

Format 2:

```
RENAME LIBRARY libname [ON volname]  
          TO libname
```

The rename statement is used to rename files or libraries. The format 1 rename statement renames a file. If libname or volname are not specified in either the source or target file reference, OUTLIB and OUTVOL are used. A different library can be supplied in the source and target library to effectively move the file from one library to another on the same volume.

The format 2 rename statement is used to rename a library. If volname is not specified in the source library reference, OUTVOL is used. The library will be renamed and must remain on the same volume.

Example:

```
RENAME 'editfile' IN '#worklib' TO 'editsave'  
RENAME LIBRARY '#ksawork' ON VOL111 TO 'ksasave'
```

4.19 Return Statement

Format:

```
RETURN[CODE = expression]
```

The return statement is used to terminate the procedure. When executed the procedure will end and control will be returned to the calling program or procedure. By default the end of the procedure file includes an implicit return statement.

For the RETURN statement, the value specified within the optional CODE clause is returned as the procedure return code. If the CODE clause is not specified, the return code is zero.

Example:

```
IF &pfk = 16 THEN RETURN CODE = &TEST1
```

4.20 Run Statement

Format:

```

RUN filename [IN libname] [ON volname]
             [ERROR EXIT [IS] label ]
             [CANCEL EXIT [IS] label ]
             [DISPLAY prname [keyword = value[, ...] ] ]
             [ENTER prname [pfkey] [keyword = value[, ...] ] ]
             [USING arglist]

```

The run statement is used to execute programs and procedures. This statement can only be used when the VUPort Help Processor has executed the procedure. The filename is the program or procedure to run. When executed, the procedure will look first for the program in the library and volume given. If it is not found there, or the library and/or volume is not given, the computer will search for the program in PROCLIB on PROCVOL, then in RUNLIB on RUNVOL, and finally, in the library named library on the volume name SYSTEM.

The filename must be an exact match. The procedure interpreter can determine the type of file and will add the extensions if needed, (e.g. .gnt). When the program is executed, the system will set the current working directory to the current value of INLIB on INVOL.

Program and procedure errors can be trapped by using the ERROR EXIT statement. If an error occurs while the procedure is searching for the program to execute, the procedure will transfer control to the label named in the ERROR EXIT statement. If an error occurs while searching for the program and the ERROR EXIT statement is not given, an error will occur and the procedure will stop.

It is usually desirable to trap a cancel request from the user. If the CANCEL EXIT statement is supplied and the user cancels the program named in the run statement during execute, the procedure will transfer control to the label named in the CANCEL EXIT statement. If the CANCEL EXIT statement is omitted, and the user cancels the program during execution, the procedure will continue with the next statement after the run statement. In both the ERROR and CANCEL EXIT statements the IS clause is commentary and can be omitted.

Parameters can be supplied to a program by using the DISPLAY and ENTER statements. Any number of DISPLAY and ENTER statements can be supplied in a run statement. A comma must separate them. The Display statement allows the user to supply parameters to a getparm screen without terminating the screen. The enter statement will supply parameters to a getparm screen and will terminate the screen.

The pname field in the DISPLAY and ENTER statements is the name of the screen found on the third line of the getparm request. The keyword fields are the field names where values are placed. Any number of keyword '=' value statements can be supplied for each ENTER or DISPLAY statement, but they must be separated by a comma. If pfkey is supplied in the ENTER statement, the getparm screen is terminated with that pfkey. Valid values are 0 to 32, where 0 is the execute key. If pfkey is not given, 0 is used.

Example:

```
* Execute Screen format utility
* Do not allow file saves
RUN 'kzform' ENTER OPTION 2,
                ENTER GENERATE 16

* Execute some application supplying parameters
RUN 'applic' ENTER FUNCTION 4,
                ENTER COMPANY          COMPNUM = 9998, COMPPER
= 3,
                COMPDATE = 123175,
                ENTER FUNCTION 16
* Execute system display utility trapping exits
RUN 'display'
    ERROR EXIT IS errxit
    CANCEL EXIT IS canxit
```

4.21 Scratch Statement

Format 1:

```
SCRATCH filename [IN libname] [ON volname]
```

Format 2:

```
SCRATCH LIBRARY libname [ON volname]
```

The scratch statement is used to remove files and libraries from the system. The format 1 scratch statement is used to remove a file. If libname and/or volname is not given, OUTLIB and OUTVOL is used. It is NOT an error to scratch a non-existent file or library.

The format 2 scratch statement is used to remove an entire library. If volname is not given, OUTVOL is used. WARNING: scratching a library removes all the files in the library.

Note: Scratching the last file in a library does not scratch the library.

Example:

```
SCRATCH 'junkfile' IN 'junklib' ON VOL111  
SCRATCH LIBRARY '#ksawork'
```

4.22 Set Statement

Format:

```
SET keyword = value [,keyword = value[. .]]
```

The set statement is used to set usage constants. One or more keyword '=' value statements can be in a set statement. Each set of keyword '=' value clauses must be separated by a comma. Below is a list of all the valid set keywords. Keywords with a '†' must be set with numeric value although conversion and truncation will take place where needed.

Table 11 List of Set Keywords

FORM#	INLIB	INVOL
JOBCLASS	JOBLIMIT	JOBQUEUE
OUTLIB	OUTVOL	PRINTER
PRNTMODE	PROGLIB	PROGVOL
PRTCLAS	PRTFCLASS	PRTFORM
RUNLIB	RUNVOL	SPOOLIB
SPOOLSCR	SPOOLSYS	SPOOVOL
SYSLIB	SYSVOL	TASKTYPE
WORKLIB	WORKVOL	

Example:

```
SET inlib = 'data', invol = vol111, printer = 5,  
    prtform = 132, prntmode = s
```

```
SET prntmode = h
```

4.23 Shell Statement

Format 1:

```
SHELL
```

Format 2:

```
SHELL COMMAND = value
```

Format 3:

```
SHELL | filename [INlibname] [ON volname]
      | path
```

The shell statement is used to invoke a Unix shell or execute a Unix shell statement. During execution of the shell, the interrupt key is always turned off. See HELP OFF statement for more information on interrupting programs.

The format 1 shell command creates an interactive shell. Before the shell is created, the system will set the current working directory to INLIB on INVOL. The shell created uses the value of the environment variable SHELL to determine which shell to execute. During the execution of the shell, the user can request help from the VUPort Help Processor by executing the command 'kh'.

The format 2 shell statement is used to execute one Unix command. The command executing is held in value. Value can be a variable, constant, substring or backward reference. The length of the value is limited to 80 characters.

The format 3 shell statement is used to execute a Unix shell file. The file name can be given in either a direct Unix path name, which must start with a /, or in a file name reference. If libname and/or volname is not given, RUNLIB and RUNVOL are used.

Example:

```
SHELL COMMAND = "pg /usr/dict/words"
SHELL /usr/bin/wp
SHELL 'ksashell' in 'utility' on system
```

4.24 Submit Statement

Format:

$$\begin{aligned}
 & \text{SUBMIT } \{ \text{procname} [\text{IN libname}] [\text{ON volname}] \} \\
 & [\text{AS procedure - id } \{ \text{USING } [\text{parameter}] \dots \}] \\
 & \left[, \text{GLOBALS} = \left\{ \begin{array}{l} \text{YES} \\ \text{NO} \end{array} \right\} \right] \left[, \text{ENVIRONMENT} = \left\{ \begin{array}{l} \text{YES} \\ \text{NO} \end{array} \right\} \right] \\
 & \left[, \text{CLASS} = \text{class} \right] \left[, \text{STATUS} = \left\{ \begin{array}{l} \text{RUN} \\ \text{HOLD} \end{array} \right\} \right] \\
 & \left[, \text{DUMP} = \left\{ \begin{array}{l} \text{PROGRAM} \\ \text{YES} \\ \text{NO} \end{array} \right\} \right] \left[, \text{ACTION} = \left\{ \begin{array}{l} \text{CANCEL} \\ \text{WARN} \\ \text{PAUSE} \end{array} \right\} \right] \\
 & \left[, \left\{ \begin{array}{l} \text{DISPOSITION} \\ \text{DISP} \end{array} \right\} = \text{Requeue} \right]
 \end{aligned}$$

The SUBMIT statement submits a program or procedure file to the program or procedure queue for non-interactive execution. Selected local variables can be passed to the submitted procedure or program. All global varieties can also be passed to the submitted procedure.

GLOBALS, ENVIRONMENT, CLASS, STATUS, DUMP, CPULIMIT, ACTION, and DISPOSITION can be specified in any order, but each may be specified only once.

The SUBMIT statement submits a program or procedure file into the program or procedure queue for non-interactive execution. With the USING clause, selected local variables can be passed to the submitted procedure or program. With the GLOBALS option, all global variables can also be passed to the submitted procedure.

If a filename, library, and volume are all specified, and the file does not exist, the SUBMIT is unsuccessful. If no library and no volume name are specified, the file is searched for in PROGLIB on PROGVOL. If the file does not exist there, the file is searched for in SYSLIB on SYSVOL. If the file does not exist there, an error occurs. If a library, but no volume name, is specified, the file is searched for in the specified library on PROGVOL. If the file does not exist there, the file is searched for in the specified library on SYSVOL. If the file does not exist there, an error occurs.

If AS procedure-id is specified, the procedure-id is used to identify the procedure on the procedure queue. If AS procedure-id is not specified, the system automatically assigns one.

If USING is specified, the listed parameters are passed by value to the submitted procedure or program.

GLOBALS specifies whether all global variables accessible to the submitting procedure are to be made available to the submitted procedure. If GLOBALS YES, global variables are made available to the submitted procedure; otherwise, they are not.

ENVIRONMENT specifies whether the user's current usage constants apply to the submitted procedure. If ENVIRONMENT = YES, current usage constants apply; otherwise, they do not.

CLASS specifies the class to which the submit request is to be assigned. This value must be A through Z. It can be specified as a constant, a variable, a partial backward reference, or any other string expression. If CLASS is not specified, the value of JOBCCLASS from the SET Usage Constants function from the command processor is used.

STATUS specifies the queuing status for the submitted procedure. RUN specifies that the file is executed. HOLD places the file on the procedure queue, but it does not execute until it is released.

DUMP specifies whether a program dump is produced on abnormal termination. If DUMP = YES, a dump is produced for the job submitted. If DUMP NO, no dump is created. If DUMP = PROGRAM, a dump is produced only if the program terminates abnormally.

DISPOSITION specifies whether the procedure is queued again after that execution. REQUEUE is the only option and specifies that the procedure is requeued.

The value of GLOBALS, ENVIRONMENT, CLASS, STATUS, DUMP, ACTION, DISPOSITION, as well as hh, mm, and ss of CPULIMIT, can also be specified as a constant, a variable, a partial backward reference, or any other expression that evaluates to a valid value for the given option.

The Procedure Interpreter generates a procedure and submits it if the USING clause is coded, GLOBALS is equal to YES, ENVIRONMENT is equal to YES, or if a program file is being submitted.

Example:

```
PROC
  DECLARE &PARAM2 AS INTEGER
  SUBMIT JOBCOO IN GSLIB OF VOL600 AS DEMO USING
    "TEST" , &PARAM2
```

4.25 Using Statement

Format:

```

USING variable [, variable]...
AS { STRING(expression) }
   { INTEGER          }
[ , variable [, variable]... AS { STRING (expression) } ]
                                { INTEGER          } ]

```

The USING statement declares the formal parameter to a procedure. When the USING statement is used, it must be the first statement following the PROCEDURE statement. The USING statement declares the formal parameters to a procedure, specified by the listed variables and the attributes of the parameters. The parameter type can be STRING or INTEGER. If the parameter type is STRING, it appears in the form STRING (expression), where expression is an integer expression. This value defines the size of a string parameter. The number of parameters must equal the number of parameters declared in the USING statement.

Example:

```

PROCEDURE SUBPROC, WHICH ILLUSTRATES A USING
STATEMENT USING &FILE, &LIBRARY STRING(8), &VOLUME
STRING(6)
RUN COBOL
ENTER INPUT FILE = &FILE, LIBRARY = &LIBRARY,
VOLUME = &VOLUME ENTER OPTIONS
ENTER OUTPUT FILE = &FILE, LIBRARY = GSOBJ,
VOLUME = LANG

```

5 Forms Control

5.1 Introduction

VUPort Forms control is designed to allow for maximum control over your printers and printouts. Each printout is assigned a form number. VUPort Systems will determine the actions required based on the parameters of the form. To define forms you need to run the program **ksaforms**. This program is part of the standard distribution of the VUPort System and is normally located in the library named **library** on the volume named **SYSTEM**.

The utility **ksaforms** is a standard file maintenance program. It includes the usual functions to Add/Change/Delete and print the forms definitions. When executing **ksaforms**, the main menu is displayed first. Pressing Pfkey 1 will enter add mode, pfkey 2 enters change mode, and pfkey 3 enters delete mode. In addition, pfkey 4 will print all the records in the forms file.

Once you have entered a mode, it will stay active until you press pfkey 16. Pressing pfkey 16 will return to the program to the main menu. Pressing pfkey 16 from the Main Menu will exit the program and return to the calling program or the VUPort Help Processor or to the program or procedure which executed the program.

5.2 Form Number

The form number is the identifying field to the form definition. The form number is a numeric field. It can be in the range of 0 to 999. The user has control over which forms are in the file except for form zero. Form Zero is a special form. It can not be deleted from the forms file. The user may wish to change the parameters for form zero, although any requests to delete it will produce an error message.

The form number is the number that corresponds to the form number in the print queue. When a printout comes up to be printed the document's form is loaded on the printer. More information is available on loading print forms in the section on Printer Control.

5.3 Lines Per Page

In each form definition the lines per page are defined. This parameter is used to determine how page control is handled. This field is used in a few ways.

First, let's discuss form alignments. When printing form alignments, the system will look at this field to determine where to print alignment lines. When an alignment is printed, the system will generate a form feed, then six lines of alignment 'X's. Two lines are printed at the top of the page, two lines in the middle, and two lines on the bottom of the page. Based on the lines per page field, the last line of a form alignment should be one line away from the top of the next page. One line feed should then place the printer at the next page of the form on the printer.

The second use of the lines per page field is to determine the number of pages in the printout. When calculating the number of pages the program looks for form feeds and counts lines. When the number of lines per page is reached the computer will consider this one page. The alternative is to re-set the line count when a form feed is reached. A combination of the number of form feeds and the number of over run pages found determines the number of pages that are in a printout.

Another use of this field is to avoid printing on the form perforation. Most forms have a perforation between pages. When a printout does not contain form feeds, lines can end up being printed on the perforation line resulting in a printout that is very difficult to read. If the print over perforation field is set to 'N' (No), the system will generate a form feed when the number of lines per page has been reached.

Another reason for generating form feeds involves the use of laser printers. Certain laser printers will lose a line at the end of a page if they do not receive a form feed. This can be very annoying, and will lead to printouts that are virtually useless. In either case, the lines per page field will determine when a form feed character should be generated to avoid these unpleasant aspects of your computer's printers.

5.4 Print Over Perforation

The Print Over Perforation field can have two values. A 'N' value meaning No, or a 'Y' value meaning Yes. Note that all fields will be converted to upper case as they are typed in. You can type a small 'y' with the same results as typing a capital 'Y'.

When this field contains a 'N', the system will be very careful not to print lines on the perforation between pages. The system will use the lines per page field to determine when a form feed character should be generated to avoid printing on the perforation. Another useful part of this feature is to force laser printers to eject a page when it is full. Some laser printers require either a form feed or an extra line at the end of a page to eject pages. If this field is set to 'N', the system will force the laser printers to eject a page when the printout has printed lines per page number of lines.

If this field is 'Y', the computer will not require that the application generate form feed characters where needed. During the printing operation the system will not generate any extra form feeds to avoid printing on the perforation of the pages.

5.5 Header/Trailer Page Format

The header page and trailer page are extra pages printed before a printout is printed and after it has completed. The pages contain useful information such as, when the printout was printed, how many copies of the printout were submitted, which copy this printout is and who the printout was printed for. Both the header and trailer page format field must be one of four values.

A 'L' (Large) header and trailer page format indicates that the computer should generate a header and trailer page designed for a 132 character print line. A large header page will be generated before each copy of a printout is printed, and a large trailer page will be generated at the end of each copy of the printout.

A 'S' (Small) header and trailer page format indicates that the computer should generate a header and trailer page designed for an 80 character print line. A small header page will be generated before each copy of a printout is printed, and a small trailer page will be generated at the end of each copy of the printout.

A 'F' (Form Feed) header and trailer page format indicates that the computer should generate only a form feed character at the beginning and the end of a printout. The extra form feed character at the beginning of the printout will guarantee that the new printout will start at the beginning of the page. The extra form feed character at the end of the printout guarantees that the printer will not be left stopped in the middle of the page after it has completed printing a document. The trailer form feed character is also useful for clearing a laser printer's buffer. Some laser printers will not eject a page, or will require a certain time-out period to elapse before ejecting the last page of a document. The 'F' format trailer page, will guarantee that the laser printer's buffer will be cleared after the last page is printed.

5.6 *Similar Format*

The similar format field is very significant in the operation of the print spooler. During print operations, the computer is constantly loading and reloading forms. Most of the form changes are due to new parameters being applied to a printout. They do not actually require a physical stock change on the printer.

The similar forms field will determine if the change of form requires that a new form be physically loaded on a printer. Let's take for example when you are using a laser printer to print most of your documents and reports. During the normal operation of the printer, different forms are loading based on the font you wish to use for each particular printout. Only a small portion of the time do you actually require letterhead paper to be loaded in the laser.

This being the case, the laser printer can continue to print on its normal white bond paper most of the time. But, the laser must be told to switch from portrait to landscape mode and back, for example, if you are printing a spreadsheet or a simple report. Now, each form for the portrait format, and each form for the landscape format will all contain the same similar form number, usually form zero. However, the form number 100 indicates letter head stock is to be used, so the similar form number would for that form be anything but form zero.

During the printing operation, the system will determine if the new form to be loaded on the printer requires a stock change. This is done by seeing if: 1) the current form number matches the new form number, 2) the current form number matches the new similar form, 3) the current forms similar form number matches the new forms similar form number. If any of these conditions are met, the system will change the currently loaded print form and continue printing uninterrupted.

However, if all of the above conditions are not met, the system will stop the printer and place the printer into the 'help' state. The system will display a message stating which form is required on which printer. After the operator has loaded the form and informed the computer that it has been loaded, the system will start to print the printout. The next printout's form number will determine when the computer will request that the old form be reloaded on the printer.

5.7 *Expand Tabs*

The expand tabs field can contain three values. The field is used to determine if the system should expand tabs within the printout to hard spaces or leave them as tabs.

A value of 'Y' (Yes) tells the system to expand all tabs in the printout to enough spaces to fill characters up to the next eight-character boundary. If the printer does not support tab stops, this field should be set to yes.

A value of 'N' (No) tells the system to leave tabs alone. During the printing processes, the computer will pass the tabs through to the printer.

Finally a value of 'D' (Data) tells the computer not to process the printout at all. This value will turn off all line counting and tab expansion. This value should be used with printouts that do their own pen positioning. Some applications like **troff** and other graphics-type applications generate cursor or pen positioning commands directly in the document. A value of 'D' informs the system that this is happening so that it can react accordingly.

5.8 Download Information

The download information is the location of a file that should be sent to the printer at the time the form is loaded. This file can contain control printer control sequences. The file might contain information to tell the computer to print in portrait mode. Large, more complex files can also be used to load soft fonts or other alternate character type information.

One word of advice can be added. When creating the download files, the user should not assume the state of the printer. Each download file should contain all the information to place the printer into the state you want it in.

A few sample download files have been supplied with the software. Each can be found in the library 'fonts' on the volume SYSTEM. They are designed to control a Hewlett Packard Laser Jet or compatible laser printer.

- | | |
|----------------|--|
| font80p | This file will place the HP laser printer into 80 column portrait mode. |
| font32p | This file will place the HP laser printer into 132 column portrait mode. |
| font32l | This file will place the HP laser printer into 132 column portrait mode. |

6 Application Interfacing

6.1 Starting The VUPort System

There are several ways to start the VUPort System. Technical users and applications designed to run from the shell can continue to run from the shell. From this level, to enter the Help menu, execute the program **ksamain**. The program will start up and the usual program initialization will be done.

Shielding the user from the shell is sometimes desirable. You can achieve this in two ways. The first method is by using the **ksamain** program as the users shell. The VUPort System will be started as soon as the user logs into the Unix System. Below is an example of the password file entry needed to accomplish this. See the [Unix Programmer Reference Manual, Volume 4, Passwd File Format](#), for a complete description of the password file line.

Example:

```
ksa::100:100:KSA
User:/usr/acct/users/ksa:/bin/ksamain
```

The second method is by invoking a shell like '/bin/sh' from the password file and then using the **.profile** file to execute the VUPort System. This is a good way of allowing the application to set up specific environment variables needed by your application before entering the VUPort System. By using the **exec** shell command, the user will be logged out when the VUPort System terminates. Below is a sample profile file to accomplish this.

Example:

```
APPDIR=/usr/acct/application
PATH=/usr/nbin:$PATH
COBDIR=/usr/lib/ncobol
SHELL=/bin/sh
TERM=wyse50
export APPDIR
COBDIR PATH SHELL TERM
exec /bin/ksamain
```

6.2 *KSARC, Initial Procedure File*

When **ksamain** is started with the '-i' argument it will look for the file **.ksarc** in the users HOME directory. If the file is found it will be executed as the initial procedure file. This file can be used to set-up the user's VUPort environment and execute the application, this allows the user to be completely shielded from the Unix Shell and VUPort System as a whole. Below is a sample procedure to set-up the users environment, execute a main menu program, and then log off the computer.

Example:

```
PROCEDURE LOGIN

* Set global application usage constants
SET PRINTER = 5, PRTFORM = 80, PRNTMODE = H,
  INLIB = 'compdata', INVOL = 'VOL111'
  RUNLIB = 'acctobj', RUNVOL = 'VOL111'
  PROGLIB = 'acctproc', PROGVOL = 'VOL111'
* Run users application menu
RUN 'menu'
* Exit VUPort System to log off the system
LOGOFF
```

6.3 Executing different applications

Certain applications cannot be directly executed from the VUPort System. An example of this is RM/COBOL (Ryan-McFarland Cobol) applications. In general application programs that require command line arguments cannot be executed from a RUN Statement in a procedure or from the RUN Program screen within the VUPort Help Processor.

One method of executing these types of applications is to use the **SHELL** Statement from a procedure. This is sometimes acceptable, although the system will turn the Help key off while the application is executing. An alternative is to write a very small 'C' program to invoke your application. Remember only programs and procedures directly executed by the VUPort System can be interrupted. These programs can reside in a common program library pointed to by RUNLIB. Below is an example of a 'C' program that is used to execute the RM/COBOL verification programs. Note that this program uses an **execl** statement and therefore, will be replaced by the application program. This will allow the application to execute with the help key enabled. In this example the procedure that runs this program has already set the INLIB to the library where the data files reside.

Example:

```
/* Program to execute the RM/Cobol Verify Suite */
main()
{
    execl("/usr/bin/runcobol", "runcobol",
          "VERIFY", "-k", 0);
    perror( "Unable to execute RM/Cobol runtime");
    exit( 100);
}
```

6.4 Executing Micro Focus Cobol Applications

The VUPort System will automatically determine if a file to be executed is a Micro Focus Cobol Application. They will be invoked with the proper runtime arguments at execution time. Usually the intermediate object files and the Native code object filenames have an extension. The .INT and .GNT extensions are common. These extensions need not be supplied in the RUN statement or on the Run Screen. The VUPort System will also determine the type of object file the File is.

The VUPort System understands two different versions of MF/Cobol. The older version is the non-Extended Technologies (ET Cobol) version. The VUPort System will invoke the Cobol object using the runtime interpreter **cbrun** found in the directory **/usr/bin**.

In an attempt to support multiple compiler versions, the VUPort System will invoke Micro Focus ET Cobol objects with the runtime interpreter **cbrun** found in the directory **/usr/nbin**. These constant path names are inconsistent with the path names of the other programs the VUPort System looks for. For systems with the Cobol/2 product the VUPort System will invoke the Micro Focus runtime with the name 'rts' found in the directory **/usr/lib/ksa**. In all other instances, the program will attempt to locate the programs by searching the directory's names in the PATH variable.

The environment variable COBDIR is used by Micro Focus ET Cobol to locate the programs it requires. It is the responsibility of the login process to set this variable if it is anything other than the standard **/usr/lib/cobol**.

6.5 LPRB, The line printer spooler

The replacement line printer spooler program is named **lprb**. This program is renamed to replace your system spooler. It usually replaces the **lp** or **lpr** program depending on the one your system uses. The renaming process makes the swap transparent to your application. This program is invoked by many different applications in many different ways. Below are a few considerations to keep in mind when interfacing it to your application.

When using the **lprb** spooler from applications started by the VUPort System the Usage constants play an important role. The `PRINTER`, `PRNTMODE` and `PRTFORM` constants are read from the users set-up and used accordingly, this is always the case. 'T' is also true when using the spooler from a shell started by the VUPort System.

When invoking the `lprb` program from applications running without an active VUPort Help Processor you need to consider a few things. First, if the application is executing the spooler without any arguments, for instance MF/Cobol applications, the user can set-up environment variables to replace to Usage constants. The names of the environment variables are the same as the usage constants and have the same meaning and effect. The most important variables are `PRINTER`, `PRNTMODE` and `PRTFORM`. Remember that if your application is running in a Bourne Shell environment, these variables must be exported.

Example:

```
PRINTER=1
PRNTMODE=S
PRTFORM=80
export PRINTER PRNTMODE PRTFORM
```

When you have the liberty of supplying arguments to the spooler they can override the environment variables. The arguments can be supplied in any order, and the argument values can be given with or without a space between the flag and the value. The available flags are:

-p printer-number

-f form-number

-m print-mode (S=Spool, H=Hold, K=Keep)

-x copies

When the **lprb** programming is running from an application such as Micro Focus Cobol, it is not always possible to supply command line arguments. When the application needs to supply a copy count, it can be done by sending the lprb program a copy count request in the first record to be printed. This is done by sending a null (Hex 00) character followed by a ASCII copy count. The definitions required to do this in a Micro Focus Cobol program are below.

Example:

```

.
.
.
01 COPY-REQUEST.
   05 FILLER      PIC X(01) VALUE LOW-VALUES.
   05 COPY-CNT   PIC 9(05) VALUE 1.
.
.
.
WRITE REPORT-RECORD FROM COPY-REQUEST BEFORE 1.

```

When an active VUPort Help Processor can be located, the values of the usage constants are used. When an active VUPort Help Processor cannot be located, the spooler will default the proper usage constants to the value found in the environment variables and then override them with the command line arguments. When all else fails, the spooler will default to printer 0, the print mode to S (Spool), and the print form to 0.

6.6 Hints on setting up Menus and Procedures

The VUPort Procedure language is truly a language. That means that as with any other languages, there are different methods and conventions that can always be debated on how to solve a particular problem. It has been said that if you give one hundred programmers one problem to solve, chances are you will receive one hundred different solutions to the problem. If you give the same programmers the same problem to solve one year later, chances are you will receive one hundred different answers none of which are the same as the first one hundred you received.

The point of this section is not to debate the different methods available to solve problems, but simply to suggest some possible guidelines to use. These guides have proven helpful in many different applications over a considerable period of time.

First, you should avoid hard coding library and volume names into your menus. You should take advantage of the auto search features built into the VUPort System. The menu programs should contain references to the program or procedure name only. The main application procedure or even the login procedure should set the **RUNLIB** and **PROGLIB** usage constants to the location of the application programs and procedures. By convention the **RUNLIB** should be used for executable files, and **PROGLIB** be used for procedure files.

Next, it is also good practice to avoid hard coding volume names into your application procedures. It is a good practice to have the upper most procedures set the **INVOL** and **OUTVOL** usage constants and let the specific application procedures set the library names. This is very handy when you are reorganizing your system. If you isolate the references to volume names to a single procedure, only that procedure needs to be changed when the location of the data files or program files change.

In regards to printers, assume nothing. For the most part, the user should have control over the printer settings. The login procedure can set up the default printer number, form number, and desired print mode. From that point on leave the rest up to the user. The application procedures can and in most cases should set the form number as it applies to the particular application; however, the printer number and form number should be set as little as possible. It can be very annoying to the user to have each procedure file reset the print mode to spool after they have manually set it to hold.

Volumes should be mounted at the time the system is brought up and should remain mounted. Although there are always exceptions to the rule, it is good practice to leave volumes mounted all the time. The same concept applies to printers. Each printer should be attached to the system at the time the system is initialized and remains attached for the entire time the system is active. For the most part this is forever. By doing this you can avoid any unneeded confusion of now you see it, now you don't.

