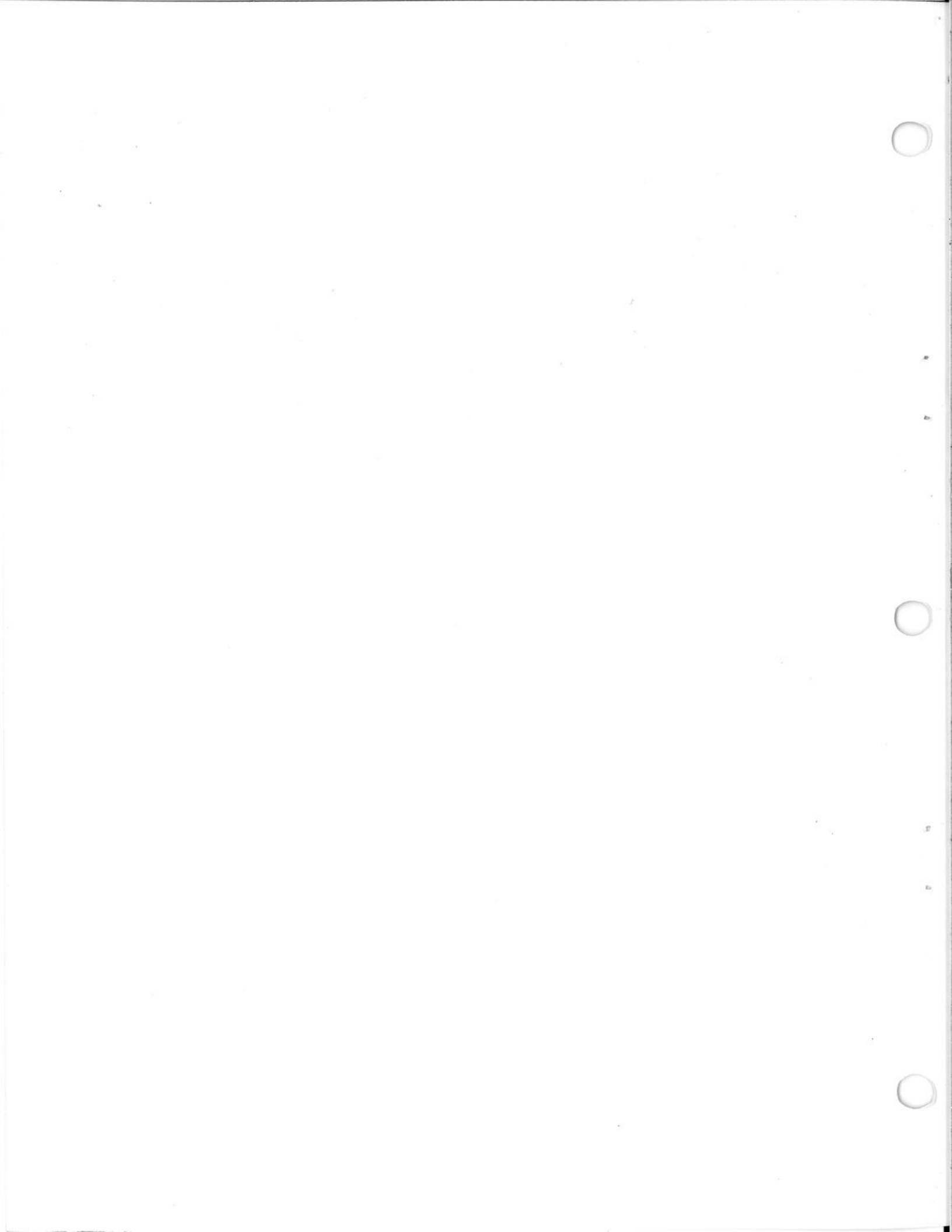


Boston University Computing Center

# The VPS HANDBOOK





## Preface

This publication is one of a series of four manuals which describe the features of VPS™. Whereas the other three manuals (VPS Utilities, VPS Guide to Programming Languages, and VPS System Facilities for Assembler Programmers) are aimed at specific uses of the VPS system; this manual describes the basic facilities of VPS - terminal commands, control statements, file structures, etc. It is intended for every user of VPS - from the most casual to the most experienced. To make it more advantageous to buy this manual (and the other three manuals, as well), it has been three-ring punched and left unbound for easy insertion into a loose leaf notebook.

This manual was produced entirely on VPS using the VPS Editor and a new, enhanced version of SCRIPT (an English language text preparation program). For information on these facilities the user is referred to the VPS Utilities Manual. During the course of writing this manual we found SCRIPT deficient in some areas and added new facilities which we feel will be beneficial to other users of SCRIPT (e.g. an underlining feature - much easier to use than backspace and a program which will take a special SCRIPT output file and produce a two column index - an example of which is the index to this manual). The manual itself is divided into two parts - Section 1 being a description of the VPS facilities and Section 2 being a description of each of the VPS terminal commands. Hopefully, this will make it possible to use Section 2 as a quick reference.

Marian G. Moore  
John H. Porter

Second edition, March 1979  
Revised, July 1980

© 1980 Boston University

VPS is a registered trademark of Boston University



Contents

Introduction . . . . .	1
Section 1: VPS System Facilities . . . . .	5
Chapter 1: Terminal Control Procedures . . . . .	5
Direct Wired Terminals . . . . .	5
Dial-up 2741-Like Terminals . . . . .	5
Dial-up ASCII Terminals . . . . .	6
Signing on to VPS Using an ASCII or 2741-like Terminal . . . . .	7
The Account Number . . . . .	7
Notes on Dial-ups or Be Prepared . . . . .	7
Notes on ASCII Terminals . . . . .	7
Signing on to VPS using a 3270 . . . . .	8
Operating the 3270 or Confusion Can Lead to Discovery . . . . .	8
Logical Line Editing Functions . . . . .	10
Tabs . . . . .	12
The Logical Tab Character . . . . .	14
The Attention Key . . . . .	14
Upper and Lower Case . . . . .	14
The Conversational Read List . . . . .	15
Chapter 2: Job Structure . . . . .	17
Format of System Control Statements . . . . .	17
The /FILE Statement . . . . .	19
Matrix of Required and Optional /FILE Parameters . . . . .	22
Summary of Default Input/Output Units . . . . .	23
The /LOAD Statement . . . . .	25
The /JOB Statement . . . . .	28
Symbolic Parameters . . . . .	29
Defining Symbolic Parameters . . . . .	29
Assigning Default Values to Symbolic Parameters . . . . .	30
Overriding Default Values . . . . .	31
System Defined Symbolic Parameters . . . . .	31
Using Symbolic Parameters . . . . .	32
Chapter 3: Library Files . . . . .	35
Access Control . . . . .	36
Library Limit . . . . .	36
Charges . . . . .	37
The Temporary Files * and *2 . . . . .	37
Reading Library Files in Programs . . . . .	37
The /INCLUDE Statement . . . . .	39
Creating Library Files . . . . .	41
Listing the Names of Saved Files . . . . .	43
Changing the Names and Access Codes of Library Files . . . . .	43
Purging Files From the Library . . . . .	44
Listing Library Files at the Terminal . . . . .	45
Listing Library Files on the Batch . . . . .	45
Making Little Holes in Cardboard . . . . .	45
Chapter 4: Work Files . . . . .	47
Structure of a PS Work File . . . . .	48

Structure of a DA Work File . . . . .	48
The VTOC and the Catalog . . . . .	49
Work File Track Limit . . . . .	49
Charges for Work File Space . . . . .	49
Defining a Work File Input/Output Unit . . . . .	50
Work File Naming Conventions . . . . .	54
Renaming Work Files . . . . .	55
Creation and Deletion of Work Files . . . . .	55
Temporary Work Files . . . . .	56
Editing Work Files . . . . .	56
 Chapter 5: Magnetic Tape . . . . .	 57
Defining Tape Input/Output Units . . . . .	57
Accessing More Than One File on a Tape . . . . .	59
Reading Standard Labeled Tapes . . . . .	60
Writing on Blank Tapes . . . . .	61
File Protection . . . . .	61
Using Tapes from Terminals . . . . .	61
Examples of /FILE Statements for TAPE Units . . . . .	61
 Chapter 6: Other Input/Output Units . . . . .	 63
TERMIN: Reading the Terminal Keyboard . . . . .	63
TERMOUT: Receiving Printout . . . . .	64
Virtual Unit Record Units . . . . .	66
READER: Virtual Card Reader . . . . .	66
PRINTER: Virtual Printer . . . . .	67
PUNCH: Virtual Card Punch . . . . .	69
DUMMY: When Less is More . . . . .	71
 Chapter 7: Accounting and the User Profile . . . . .	 73
Session or Job Related Accounting . . . . .	73
Resource Related Accounting . . . . .	73
The User Profile . . . . .	74
Changing the Password . . . . .	74
The /PROFILE Command . . . . .	74
/PROFILE Example . . . . .	77
 Chapter 8: Interactive Debugging . . . . .	 79
Debugging Facility Terminal Commands . . . . .	79
Interactive Debug Mode . . . . .	80
Debug Mode Error Messages . . . . .	81
Debug Mode Commands . . . . .	81
Examination and Modification of Storage . . . . .	82
Examination and Modification of General Purpose Registers . . . . .	83
Examination and Modification of Floating Point Registers . . . . .	84
Examination and Modification of the PSW . . . . .	84
Symbol Definition . . . . .	84
Calculations and Conversions . . . . .	85
Resumption of Execution . . . . .	85
Job Termination . . . . .	86
 Chapter 9: The Batch . . . . .	 87
The /ID Statement . . . . .	87
The /END Statement . . . . .	88
The /PW Statement . . . . .	88

Use of Privileged Accounts on the Batch . . . . .	89
/LIBRARY Command on the Batch . . . . .	89
/SAVE Command on the Batch . . . . .	89
/NEWS Command on the Batch . . . . .	90
 Section 2: VPS Terminal Commands . . . . .	 91
 VPS Terminal Command Table . . . . .	 92
/ACCT - Print Accounting Data . . . . .	95
/ATTN - Pass an Attention Interrupt to an Executing Program . . . . .	96
/BLIP - Print Program Status Information Periodically . . . . .	97
/BQA, /BQB, /BQC - Submit a Job to the Batch . . . . .	98
/CANCEL - Cancel the Execution and Printout of a Program . . . . .	100
/CATL - Print a Listing of Available Programs and Subroutines . . . . .	101
/CE - Cancel Execution of a Program . . . . .	102
/CLIST - Modify, List, or Clear the Conversational Read List . . . . .	103
/CO - Cancel Printout of a Program . . . . .	105
/CTL - Define or Change Session Control Functions . . . . .	106
/DAILY - List the Message of the Day . . . . .	111
/DEBUG - Control Interactive Debug Mode . . . . .	112
/DISPLAY - List a File with Line Numbers . . . . .	114
/DS - List Information Pertaining to Work Files . . . . .	115
/DSPURGE - Purge a Work File . . . . .	117
/DSRENAME - Rename a Work File . . . . .	118
/EDIT - Initiate the VPS File Editor . . . . .	119
/EXEC - Execute One or More Files or Control Statements . . . . .	120
/ID - Initiate a Terminal Session . . . . .	122
/LIBRARY - List Information Pertaining to VPS Library Files . . . . .	124
/LIST - List a File Without Line Numbers . . . . .	126
/LOG - Print Program Status Information At Program Termination . . . . .	127
/MESSAGE - Send a Message . . . . .	128
/NEWS - List Current Computing Center Information . . . . .	129
/OFF - Terminate a Terminal Session . . . . .	130
/PER - Initiate Program Event Recording . . . . .	131
/POST - Communicate With a Running Program . . . . .	134
/PROFILE - Inspect the User's Profile . . . . .	135
/PURGE - Delete Library Files . . . . .	136
/QUERY - Print Terminal Session Environment Information . . . . .	137
/RATES - List Current Computing Center Charging Rates . . . . .	139
/RENAME - Rename a Library File . . . . .	140
/REPLACE - Replace the Contents of a File with the * File . . . . .	143
/REPLY - Reply to a Message from an Executing Program . . . . .	145
/RO - Resume Printing of Output During Program Execution . . . . .	146
/RSAVE - Replace the Contents of a File with the *2 File . . . . .	147
/SAVE - Save the Contents of the * or *2 File in a New File . . . . .	149
/SCRIPT - Invoke the VPS Text Preparation Program . . . . .	151
/SCROLL - Pause After Every "n" Lines of Output . . . . .	153
/SKIP - Suppress the Printing of Output Conditionally . . . . .	154
/SLEEP - Lock the Terminal Keyboard . . . . .	155
/SORT - Sort the Contents of Library File(s) . . . . .	156
/STATUS - Print Accumulated Terminal Session Statistics . . . . .	157
/SUMMARIZE - List "/" Lines of a File . . . . .	158
/TABIN - Modify Input Tabs . . . . .	159
/TABOUT - Modify Output Tabs . . . . .	161

/TABS - Modify Input and Output Tabs . . . . .	163
/TABSET - Set the Terminal's Physical Tabs . . . . .	165
/TAPE - Read a Paper Tape from an ASCII Terminal . . . . .	167
/TEXT - Modify Terminal Input and Output Translation . . . . .	168
/USERS - Print the Number of Users on VPS . . . . .	169
/VERIFY - Request the System to Prompt for the Password . . . . .	170
/XEDIT - Initiate the VPS File Editor for Large Files . . . . .	171
/? - Print Program Status Information . . . . .	172
Index . . . . .	173



## Introduction

VPS (Virtual Processor System) is a high-performance general purpose timesharing system written by the systems staff of the Boston University Computing Center. This system has been in use by the Boston University community since February 1977, when it replaced a timesharing system called RAX, successive versions of which had been in use at the University since 1968.

VPS is designed specifically to run under a modified version of the IBM virtual machine operating system, VM/370. Briefly, each signed-on user and each VPS batch processor occupies a separate virtual machine. Each VPS virtual machine has a certain amount of virtual storage that it owns privately in which the user's program is run. The remaining storage seen by each machine is common to all the VPS virtual machines and contains the VPS supervisor routines and data areas. Various special interfaces to VM/370 allow the separate virtual machines to communicate and synchronize with each other and to make particularly efficient use of VM's internal facilities (e.g. paging, spooling, device handling). The basic design philosophy behind VPS has been to minimize memory requirements through program sharing and to minimize supervisor overhead by avoiding the duplication of functions between VPS and VM/370. (For a more elaborate discussion of the structure of VPS, see the Computing Center publication "VPS - A Virtual Multiprocessor Timesharing System", TP 77-01.)

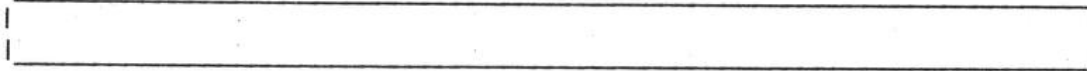
While VPS is totally different internally from the old RAX system, the appearance to the user differs mainly in the augmentation of certain facilities: Most of the commands, control statements, and programming techniques which were used under RAX remain valid with little or no modification. This approach was taken primarily because it permitted the system to be designed, implemented, and brought on line more quickly and easily than would have been the case had a more radical approach been chosen. Since the underlying design of VPS was still highly experimental, it was essential that it be tested in a real working environment before a more extensive effort was undertaken.

However, the net result of the many years of modifications that had been applied to the RAX system compounded with the changes dictated by the VPS structure have resulted in a user interface which is often inelegant, sometimes awkward, and in various ways lacking in unity. Indeed, the reasons for some of the more obscure syntax can be understood only from an historical perspective.

Consequently, the systems staff is now planning to design the "real" VPS system, in which the entire user interface will be defined and documented before implementation is begun. No particular aspect of the RAX-simulating version of VPS is considered sacred a priori - the goal being to produce an accessible but powerful new system by drawing on as large a pool of ideas as can be gathered. Users are invited to write up and add to this pool any suggestions they may have.

VPS Command and Statement Formats

Throughout this manual a box -



is used to enclose the definition of a VPS command or statement. The format of the information inside this box is defined as follows:

- The symbols [], {}, \_\_, and ... are used to indicate how a command or statement may be written. DO NOT CODE THESE SYMBOLS. Their general definitions are given below:

[] indicates optional operands. The operand enclosed in brackets may or may not be coded, depending on whether or not the associated option is desired or whether a default is to be taken. If more than one item is enclosed in brackets, one or none of the items may be coded.

{ } indicates that a choice must be made. One of the operands from the vertical stack must be coded, depending on which of the associated services is desired.

\_\_ indicates a value that is used in default of a specified value. This value is assumed if the operand is not coded.

... indicates that an operand, or set of operands, may be repeated. The number of repetitions is dependent upon the function desired.

- Upper and lower case letters in command and statement names and keyword options are used to denote the shortest allowable abbreviation. For example, in the /LIBRARY definition (page 124) the command name is written as -

/LIBrary

indicating that VPS will recognize any of the following as the /LIBRARY command:

/lib  
/libr  
/libra  
/librar  
/library

- Operands written entirely in lower case are used to denote that the user is to make the indicated substitution of a value, name, etc. For example, the definition of the /SUMMARIZE command (page 158) is given as follows:

```
/SUMmarize {filename}
```

The user is instructed to replace 'filename' with the actual VPS library file name that is to be summarized as shown below.

```
/sum myfile
```

- Where the selected operand is a combination of capital and lower case letters separated by an equal sign or a blank the user is to code the capital letters and equal sign (or blank) exactly as shown and then make the indicated substitution. For example, the /CTL command (page 106) has as an optional parameter the following:

```
[EScape c]
```

This indicates that the keyword, ESCAPE, may be abbreviated ES and that the user is to substitute the desired special character for 'c', as in the following:

```
/ct es "
```

- Commas and parentheses are to be coded exactly as shown, omitting only the comma following the last operand coded.

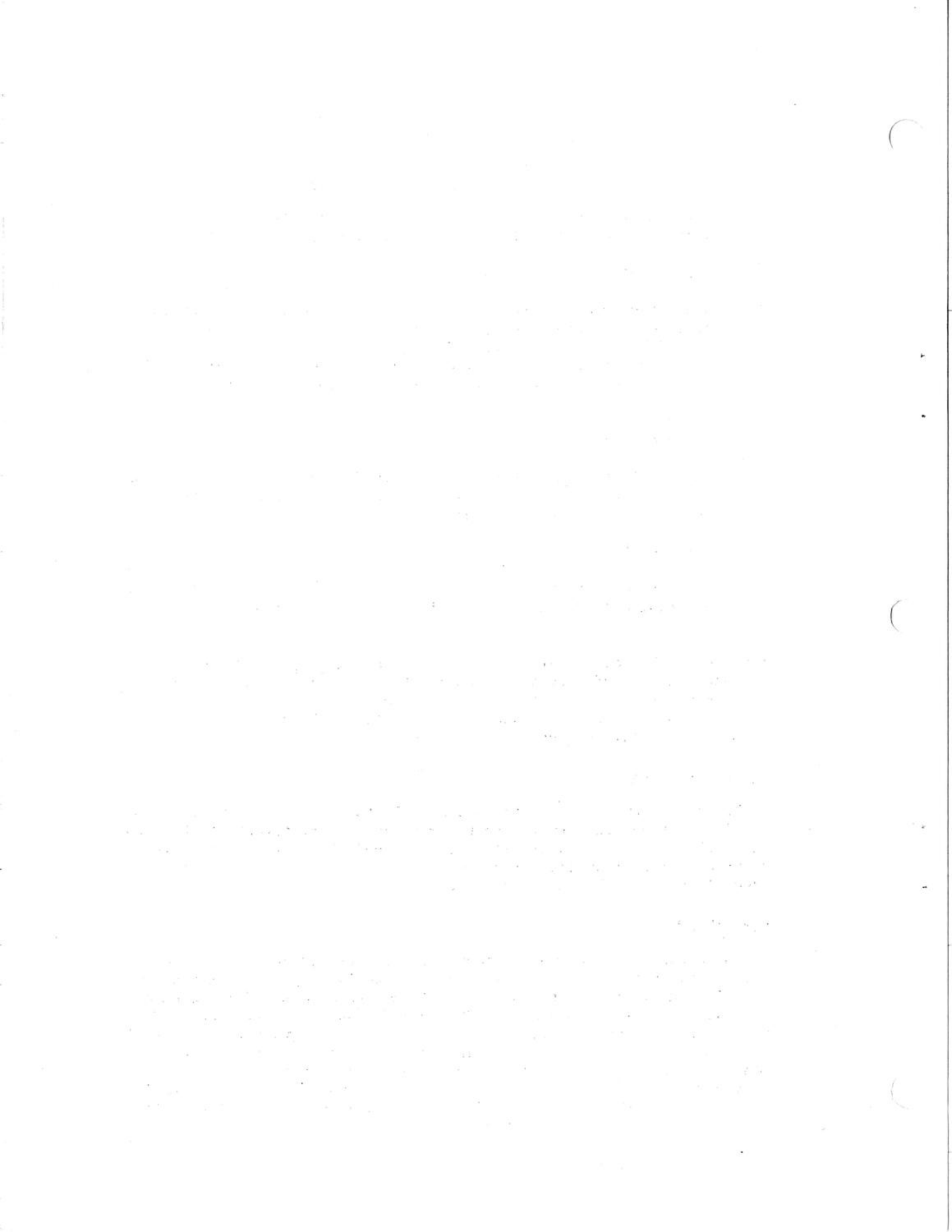
Note: Capital letters are used in this manual only to differentiate between symbols and keywords required by VPS commands or statements, and names or values selected by the user; and to indicate allowed abbreviations. If a command or statement is being typed in on a terminal it should not be typed in upper case.

#### Example Formats

In the examples shown in this manual the system responses (\*GO, etc.) are typed in upper case to more easily differentiate between what the imaginary user is entering and the associated system response. Note that depending upon the terminal type actual responses from VPS may also be in lower case.

#### What Is "OS"?

Throughout this manual we occasionally (and grudgingly) refer to "OS". Historically, "OS" was an operating system written and distributed by IBM to run on their large scale System/360 computers and came in three flavors - OS/PCP, OS/MFT, and OS/MVT. Today, "OS" encompasses a much broader range of operating system; all still written by IBM and all having, at least in their outward appearance, many similarities. Therefore, when we refer in this manual to "OS" we are referring to a function or process found in the IBM operating systems - OS/MFT, OS/MVT, OS/VSl, OS/VS2/SVS, and OS/VS2/MVS. They are still trying to get it right.



## Chapter 1: Terminal Operating Procedures

VPS supports a number of different terminals which fall into three categories:

- IBM 2741-like terminals - IBM 2741, AJ 841, etc.
- ASCII terminals - AJ 832, Tektronix 4006, Teletype models 33 and 35, Decwriter, GE Terminet, etc.
- IBM 3270 family of display terminals - 3277, 3278, 3279.

To initiate a terminal session with any of these types, the terminal must be connected to the Boston University 370/168 computer which runs VPS. The 3270 terminals used at Boston University are connected directly to the computer. ASCII and 2741-like terminals are connected by telephone lines. These terminals may be direct wired (i.e. the terminal is connected directly to the telephone line, no dialing is necessary) or dial-up (i.e. the terminal is not connected directly and the user must dial into VPS).

### Direct Wired Terminals

If the terminal, whether 2741-like or ASCII, is direct wired, the user need only turn the terminal on and press the carriage return key to initiate a terminal session. VPS will respond with a sign-on message.

### Dial-up IBM 2741-Like Terminals

To initiate a terminal session on a dial-up 2741-like terminal the following procedure should be used:

1. Turn the terminal on.
2. If a Dataphone is being used, press the "talk" button. If an acoustic coupler is being used, place the telephone receiver in the coupler.
3. Dial the VPS telephone number (353-4300, on campus Ext. 4300).
4. If a Dataphone is being used, listen for the high-pitched tone, press the "data" button, and hang-up the receiver. If an acoustic coupler is being used, wait for the carrier light to come on.
5. It should only be necessary to press the carriage return key. If difficulty dialing in with a 2741-like terminal is encountered, the user should try entering one of the following recognition characters before pressing carriage return:

EBCD - type the character #.  
CORRESPONDENCE - type the character 9.

If the characters on the type ball have the same arrangement as

those on office Selectric type balls, the terminal is a CORRESPONDENCE model, otherwise it is an EBCD model.

At this point the user should receive the VPS sign-on message. Until the user has signed on, VPS will alternately send EBCD code and CORRESPONDENCE code, so be sure to attempt to sign-on only after the non-garbled version of the message has printed. If a garbled message prints, press carriage return again.

#### Dial-up ASCII Terminals

The procedure for initiating a terminal session on a dial-up ASCII terminal is slightly different from the 2741-type terminal and the user should follow these steps:

1. Turn the terminal on.
2. Set the terminal function switches as follows:

originate-answer - set to originate  
half or full duplex - set to half duplex  
parity - set to off  
speed - set to one of the following:  
    110 Baud - for 10 characters/second  
    150 Baud - for 15 characters/second  
    300 Baud - for 30 characters/second

3. If a Dataphone is being used, press the "talk" button. If an acoustic coupler is being used, place the telephone receiver in the coupler.
4. Dial the VPS telephone number (353-4300, on campus Ext. 4300).
5. If a Dataphone is being used, listen for the high-pitched tone, press the "data" button, and hang up the receiver. If an acoustic coupler is being used, wait for the carrier light to come on.
6. Type one of the following terminal recognition characters, corresponding to the Baud setting chosen in Step 2, and hit carriage return:

10 characters/second - type an upper case S.  
15 characters/second - type an upper case Y.  
30 characters/second - type an upper case O.

At this point the user should receive the VPS sign-on message.

Note that when using an acoustic coupler, it is necessary to place the receiver in the coupler before the VPS telephone number is dialed, since it may be possible for noise to be sent down the telephone line, setting the terminal recognition character by accident before the user has a chance to enter it.

Signing on to VPS Using an ASCII or 2741-like Terminal

If the user has followed one of the above procedures successfully, the VPS sign-on message -

```
*VPS/370 SIGN ON.
```

should be printed on the terminal and the user may now enter the /ID command (see page 122) to sign-on to VPS. If the user successfully enters the /ID command, VPS will request the account password (VPS will automatically over-strike the password field as a security measure). Note that if the password is not entered correctly after three tries or the password is not entered within 20 seconds, VPS will terminate the sign-on. After the password is entered, VPS will respond with the signed-on message, the unspent account balance, and the message-of-the-day. A successful sign-on would look like the following:

```
*VPS/370 SIGN ON.
/id hb722arj
*PASSWORD?
*****
*SIGN-ON AT 14:40:49 TUE JULY 29, 1980 105 USERS
*UNSPENT BALANCE = $35.29
*SYSTEM SCHEDULE: /EX HOURS
*GO
```

The user is now at "\*go" level and may enter VPS terminal commands.

The Account Number

As you see above, the VPS account number consists of a string of letters and digits from one to eight characters in length. Students enrolled in courses using computer facilities will generally receive account numbers from the course instructor. Other persons may acquire account numbers by applying at the Computing Center. A course account number is eight characters in length and consists of a 5 character course code followed by a unique 3 character personal I.D. (PID) for the student. (Students taking more than one course will have the same PID in each course.) For example, a student enrolled in a computer course might be assigned the account number mall8rak.

Notes on Dial-ups or Be Prepared

VPS will automatically terminate any terminal session on a dial-up line where the session has been in an idle state for 15 minutes. A session is idle when the terminal is sitting at \*go or during program execution when the program (this includes commands such as /EDIT) is waiting for you to type something in.

Notes on ASCII Terminals

On some ASCII terminals it may be necessary to have the system pad output lines sent to the terminal for proper operation of the terminal. The user is referred to the descriptions of the /ID command (page 122) and the /CTL command (page 106) for more information.

Signing on to VPS Using a 3270

To initiate a terminal session on a 3270 terminal use the following procedure:

1. Turn the terminal on by pressing in the top portion of the large red switch to the left of the screen. When the terminal has warmed up, the VM-VPS logo will appear in the center of the screen. The upper left-hand corner of the screen will contain \*VPS/370 SIGN ON.
2. When the logo appears, press the ENTER key. The screen will clear, the cursor will appear in the upper left-hand corner, and CP READ will be displayed in the lower right.
3. Type the /ID command (/id followed by a space, followed by your account number) and press ENTER.
4. If the /ID command has been successfully entered, VPS will request the account password with \*PASSWORD?. (Note that as with other terminals, if the password is not entered within 20 seconds, VPS will terminate the sign-on.) Type the password and press the ENTER key. The password will not appear on the screen. After the password is entered, VPS will respond with the sign-on message, the unspent account balance, and any messages-of-the-day. You are now at "\*go" level and VPS terminal commands can be entered.

Operating the 3270 or Confusion Can Lead to Discovery

3270 display terminals are distinctly different from hardcopy terminals (IBM 2741s, Decwriters, AJ 832s, etc.) and ASCII display terminals (DEC VT100s, Televideos, etc.). The operation of an ASCII display terminal is pretty much identical to its hardcopy counterpart; only the display medium has changed - a screen for the former, paper for the latter. 3270s, however, can have their screen formats and contents controlled by the operating system. (Technically, only the characters you type before pressing carriage return are read by the operating system on hardcopies and ASCII displays. All or part of the screen can be read or written on a 3270.) This section is intended to give brief explanation of the 3270. It is assumed that users have some experience using a hardcopy terminal.

Screen Format

The screen format of the 3270 is controlled by VPS. The lower right-hand corner of the screen is used to display terminal status. The following is a list of the possible status conditions:

CP READ - normally, this appears only when signing on to VPS. It indicates that the system is ready to accept the /ID command.

VM READ - appears when either VPS or an executing program is waiting for input from the terminal.

RUNNING - appears when a command or job execution is in progress.



MORE... - appears when the screen is full and there is more output waiting. If no action is taken (see below), the system will automatically scroll the screen after one minute.

HOLDING - appears when the screen has been switched from MORE... to HOLDING (see below). If no action is taken, the screen will remain in this state.

NOT ACCEPTED - appears when a key has been pressed, and the terminal is in a state not defined for that key (e.g. pressing a PF key, see below, while in HOLDING). After 3 seconds the terminal will return to the previous state.

All screen lines above the status line are used as the input/output area. The cursor (underscore "\_") will appear at the next available line on the screen when the system is waiting for terminal input. When the terminal is in RUNNING, MORE..., HOLDING, and NOT ACCEPTED, the cursor sits in the lower left-hand corner and input cannot be entered.

### The Keyboard

The keyboard of the 3270 looks overwhelming at first because of the extraordinary number of keys. If it's any consolation - a few are not functional. The standard alphabetic, numeric, and special character sets are there and operate the same as a hardcopy terminal. Shifting to upper case is performed as on any keyboard and a shift lock key is provided on the left side. (Note that when the keyboard is shifted an up arrow appears at the bottom of the screen.)

The dark gray keys perform special terminal and system functions. Pressing a key will cause the function that appears on the top face of the key to be performed. The ALT key (to the right of the space bar) is used to enter functions appearing on the front of some keys. To enter these functions hold down ALT and press the particular key desired. The following is a list of the functional special keys and the action performed when pressed:

ENTER - enters the line of input on which the cursor resides in either CP READ or VM READ status. (Identical to the carriage return key on hardcopy terminals.) In RUNNING, it causes a single attention interrupt to VPS (see "The Attention Key" later in this chapter). In MORE... and HOLDING it causes the screen to scroll.

PA1 - in CP READ, VM READ, RUNNING, MORE..., and HOLDING causes a single attention interrupt to VPS (see "The Attention Key" later in this chapter).

PA2 - in CP READ, VM READ, RUNNING, MORE..., and HOLDING causes a double attention interrupt to VPS (see "The Attention Key" later in this chapter).


SYSREQ - in MORE... causes the screen to go to HOLDING. In HOLDING it causes the screen to go to MORE...


PF1-PF24 - each causes the action defined for that key to be performed. These are called program function keys and can be defined by the user to perform special terminal functions controlled by the system or as lines of input to be entered when that PF key is pressed. See the /CTL command, page 106, for information on defining PF keys.

CLEAR - clears the screen and resets the cursor to the top line. (Note that pressing CLEAR while output is being sent to the screen, may cause a terminal disconnect - see below.)

RESET - resets the terminal after an illegal terminal operation (an X will appear in the lower lefthand corner when this occurs).

 - resets or sets the annoying key clacker.


 - puts the terminal into character insert mode. In character insert mode, characters typed will be inserted in the line at the cursor location. Press the RESET key to return to normal character input mode.


 - deletes the character from the screen that is under the cursor.

ERASE EOF - clears the screen from the cursor to the bottom.

CURSOR BLINK - sets or resets the cursor to blinking.

ALT CURSOR - sets or resets the cursor from an underscore to a box.

 - cursor controls.

 - speedy cursor controls.

One of the real advantages of using a 3270 is that when the terminal is in VM READ status any line on the screen can be used as the input line. This can be done by moving the cursor to the line you wish to enter, making any necessary changes, and pressing the ENTER key. The line will be redisplayed at the next available line location.

It should be obvious at this point that some experimentation is necessary to really become comfortable using this terminal. However, USER BEWARE, because of the unique design of this terminal (known inside IBM as a feature), certain operations will cause the terminal to be disconnected from the terminal session. This has occurred if the VM-VPS logo appears on the screen and X PROG 407 is displayed in the lower lefthand corner. Signing-on within 15 minutes will return you to your terminal session at the point where you so abruptly left it. A disconnect can be caused on a 3270 by pressing the CLEAR key while lines are being sent to the screen. See the /CTL command for a system provided clear function which will avoid this behavior.

### Logical Line Editing Functions

Lines of information (commands, editor input, etc.) are sent to VPS by entering the line and then pressing the carriage return or ENTER

key. The system will take no action on the line until it has been sent in this manner.

VPS supports five logical line editing functions which allow error correction, multiple line input, special character input, etc. To use the logical line editing functions, the operations must be performed on the line before it has been sent by pressing the carriage return key. The VPS system defaults for the logical line editing functions are:

<u>Function</u>	<u>Default</u>
Logical character delete (BS)	Physical backspace key
Logical delimiter (DELIM)	;
Logical line delete (LDEL)	(off)
Logical escape (ESCAPE)	(off)
Hex substitution (HEX)	(off)

Note that on 3270s local line editing (e.g. moving the cursor to insert or delete characters, etc.) may be performed before the line is sent since the screen is not read until the ENTER key is pressed. Up to this point, system editing characters are not needed.

#### Logical Character Delete (BS)

The logical character delete character allows the user to delete one or more of the previous characters entered (one character per BS character, including other logical line editing characters) in the event of minor typing errors. Normally, the physical backspace key will be used (also the system default). In the following example a back arrow (" $\leftarrow$ ") is used to represent the physical backspace key:

<u>Keyed Data</u>	<u>System Accepted Data</u>
abxy $\leftarrow\leftarrow$ cde	abcde
124 $\leftarrow$ 3467 $\leftarrow\leftarrow$ 567	1234567

#### Logical Delimiter (DELIM)

The logical delimiter character allows the user to enter multiple logical lines on the same physical line to minimize wait time. The logical delimiter character is inserted at the end of each logical line. In the following example, the system default (";") is used:

<u>Keyed Data</u>	<u>System Accepted Data</u>
12345;abcdef	12345 abcdef
/acct;/log on;/list myfile	/acct /log on /list myfile

Logical Line Delete (LDEL)

The logical line delete character deletes the entire previous logical line back to and including the previous logical delimiter character (if present). In the following example, the user has previously set the logical line delete character to a dollar sign ("\$"):

<u>Keyed Data</u>	<u>System Accepted Data</u>
abcde\$12345	12345
abc;123\$456	abc456
abc;123\$;456	abc
	456

Logical Escape Character (ESCAPE)

The logical escape character causes VPS to interpret the next character as a data character, even if it is one of the logical line editing characters. In the following example, the user has previously set the logical escape character to a double quote (""):

<u>Keyed Data</u>	<u>System Accepted Data</u>
abc";123	abc;123
""abc""	"abc"

Hexadecimal Substitution (HEX)

Occasionally it is necessary to enter hexadecimal data from the terminal (e.g. to edit data files, etc.). The hexadecimal substitution character allows this function on a limited basis. After the hexadecimal substitution has been set, whenever the substitution character is found in an input file, it will be replaced with its hexadecimal substitution value. In the following example, the user has previously set the hexadecimal substitution character "&" to the hexadecimal value 02:

<u>Keyed Data</u>	<u>System Accepted Data in Hex</u>
123&456	F1F2F302F4F5F6

Tabs

Another line editing function of VPS is tab support. (Note that the 3270 does not provide a tab key. See the /CTL command, page 106, for 3270 tabbing functions using a PF key.) Tab support works in much the same manner as the tab on a typewriter - once you have notified VPS of the tab locations and you have set the tabs on your terminal - pressing the tab key while entering input will cause VPS to accept the next character typed at the next tab location set.

To use tab support properly, the user must be aware of the difference between the physical tab settings of the terminal and the logical tab settings known to VPS. The physical tab settings are the actual tab stop locations set on the terminal. VPS has no way of knowing where these positions are, since, when the tab key is pressed, only one character (the TAB character) appears in the line sent to VPS (though the carriage may move a number of spaces). Therefore, it is

necessary for the user to inform VPS of the tab locations - these are the logical tab locations. VPS supports two types of logical tab settings, input tabs and output tabs.

#### Input Tabs

Input tabs are the logical tab settings used for data entered from the terminal. When input tabs are set, each time the user presses the tab key while entering input, VPS will accept the next character typed by the user as if it were in the column position corresponding to the next input tab location. This holds true regardless of the physical tab settings on the terminal. The system default input tab settings are columns 7 and 73.

#### Output Tabs

Output tabs are the logical tab settings used for data being sent to the terminal. When output tabs are set, VPS will automatically use the physical tabs on the terminal where possible instead of spacing with blanks. This is done to save print time. The user must set the physical tabs to the output tab settings for the output line formats to be correct.

#### Setting the Tabs

Logical tabs may be set using either the /TABIN command - to set input tabs (page 159), the /TABOUT command - to set output tabs (page 161), or the /TABS command - to set both input and output tabs (page 163).

After the logical tabs have been set, the user may use the /TABSET command (page 165) to aid in setting the physical tabs to the same locations as the logical tabs. In the following example, the logical tabs (both input and output) are set to columns 10, 20, and 30. The /TABSET command is then used to set the physical tabs to the same locations.

```
*GO
/tabs 10 20 30
*OK
/tabset

INPUT TABS - 10 20 30

OUTPUT TABS - 10 20 30

SET LEFT MARGIN TO COLUMN 1; THEN HIT RETURN.

CLEAR ALL PHYSICAL TABS; THEN HIT RETURN.

.....T.....T.....T
         T         T         T
*GO
```

To avoid confusion with tab support it is recommended that users:

- set logical input and output tabs to the same column locations.
- set logical and physical tabs to the same column locations.
- set the left margin on the terminal to column 1.

### The Logical Tab Character (TAB)

For terminals without a tab key, tabbing may be performed on input by using the logical tab character. The logical tab character may be specified using either the /PROFILE command (page 135) which will set a default logical tab character or the /CTL command (page 106) which will set the logical tab character for a specific terminal session. After a logical tab character has been specified and VPS has been notified of the logical input tabs, the occurrence of the logical tab character in a line will cause VPS to place the next character entered at the next logical tab position.

### The Attention Key

The attention, PA1, or break key on a terminal may be used to control different aspects of the terminal session. If the attention key is pressed while VPS is printing a system message, the message will be deleted.

If, however, the user presses the attention key when a program produced message is being printed, when responding to a program or attention read, or when the terminal is idle and a program is executing, the terminal session will enter the attention level (note that the program will continue to execute during the attention read if it was executing when attention level was entered). At attention level special program control commands may be entered which will allow the user to cancel output, skip output, cancel the program, set debugging options, etc. Commands allowed at attention level are noted as such in Section II of this manual. An attention level read can be easily distinguished from a read issued by the program, since a colon (":") will be printed out as a prompt character for an attention level read.

Programs can be written on VPS to recognize attention interrupts (known as program attentions). To cause an attention interrupt to a program a double attention must be used. A double attention can be caused by pressing the PA2 key on a 3270, rapidly pressing the break key twice on ASCII or 2741-like terminals, or causing a single attention interrupt to VPS and entering the /ATTN command when the ":" appears.

### Upper and Lower Case

Under normal operation when a user enters a line of input, that input is translated (all upper case characters are changed to lower case EBCDIC and all lower case characters are changed to upper case EBCDIC). If the input line is being placed in a file, say with the VPS editor, the translated version of the line is used. This is done for two reasons:

- processors (i.e. compilers, assemblers, executing programs, etc.) expect input in upper case (as if it were punched on cards).
- it is much easier to enter terminal input in lower case without having to use the shift key.

Conversely, when a line of output is printed at the terminal it is translated (upper case is changed to lower case and lower case is changed to upper case). This is done for two reasons:

- on some terminals lower case printing is faster.
- if a file is being listed which the user previously entered, the listing is more easily used if it is printed in the same case as it was entered.

Because of this double translation, English language text may be entered and printed normally on terminals.

An alternative mode of operation is to have all input translated to upper case (note that this is the default on ASCII terminals). The user may specify the translation mode with the /TEXT command (page 168).

#### The Conversational Read List

As mentioned above, it is possible to enter several logical input lines as a single physical input line by using the logical delimiter character. Any such group of logical input lines entered in response to a "\*go" level read or conversational (program-generated) read is stored by the system as a list. Each time a new line of input is requested by the system (at "\*go" level) or by a program, the next line is taken off this list and passed back to the requestor. Input will be requested from the keyboard only if the list is found to be empty.

Since this list exists independently of any program which may be running, it is possible to invoke sequentially one or more programs and supply responses to them on a single physical input line. For example, the following input line will edit a file, replace it with the edited version, and execute it (";" is the user's delimiter):

```
/edit newspell;l fubar;c /fubar/foobar;/rsav;/exec newspell
```

This is equivalent to entering the following separate lines in sequence:

```
/edit newspell  
l fubar  
c /fubar/foobar/  
rsav  
/exec newspell
```

The conversational read list will be cleared if a program terminates abnormally or if a program issues a rewind request to a TERMIN unit (see page 63). The list may also be cleared, listed, or

## Chapter 1: Terminal Operating Procedures

added to at either end via the /CLIST command (page 103).



## Chapter 2: Job Structure

A job to be run under VPS consists of one or more control statements followed by whatever data records are appropriate to the particular load module (compiler, loader, etc.) invoked by the control statements. The control statements themselves define the input/output units which are needed for the program, the size of the storage area in which the program is to be run, and so on. In general, the control statements will consist of as many /FILE and /JOB statements as are required followed by a /LOAD statement. While /FILE and /JOB statements are optional, a job must always have a /LOAD statement, and this statement must always be the last statement in the set of system control statements. All records which follow the /LOAD statement are ignored by the system and are available to be read by the module that is invoked by the /LOAD statement.

### Format of System Control Statements

The system control statements have the general format

```
/xxxx parameter1,parameter2,...
```

where /xxxx is /FILE, /JOB, or /LOAD, defining the statement type. This identification must start in column 1 of the statement and must be separated by at least one blank from the first parameter. The parameters may extend to the last character of the record or column 255, whichever is reached first.

A statement may be continued by breaking it at any comma and placing the next parameter(s) on a continuation statement of the form

```
/ parameter,...
```

where the slash is in column 1 and at least one blank appears before the first parameter. Any number of continuation statements may be used. For example, the group of statements:

```
/LOAD (PROG6,  
/      FILE=14),  
/      PARM='X Y Z',  
/      NSEGS=(4,2)
```

is equivalent to the statement

```
/LOAD (PROG6,FILE=14),PARM='X Y Z',NSEGS=(4,2)
```

The format of the parameters on system control statements is very similar to that used in OS Job Control Language, for users familiar with that system. Words shown in upper case are to be coded precisely as they appear (although they should, of course, be entered in lower case at the terminal - see page 14). Words shown in lower case are to be replaced by specific information when the statement is coded. The

designation "m" or "n" means that a decimal number is required, most other cases require character strings. Parameters are separated from each other by commas and, unlike terminal commands, system control statements may not contain blanks between parameters. Braces {} and brackets [] are used, as in all parts of this manual, to delimit required and optional parameters, respectively, and are not part of the parameters. All other special characters (commas, parentheses, equal signs, etc.) are to be coded as shown, except that parentheses shown enclosing two or more parameters may be eliminated if only one of the parameters is being coded. For example, DISP=(OLD) is equivalent to DISP=OLD, but DISP=(OLD,KEEP) may not be coded without the parentheses.

Note that since VPS recognizes the ampersand as the start of a symbolic parameter (see page 29), a double ampersand must be coded in control statements when not identifying a symbolic parameter.

The /FILE Statement

The /FILE statement defines an input/output unit for the program being run. The various types of units, which are described in detail in other parts of this manual, are summarized below:

unit type	see page	description
LIBIN	37	Library file input
LIBOUT	41	Library file output
DISK	50	Work file input/output
TAPE	57	Magnetic tape input/output
TERMIN	63	Terminal keyboard input
TERMOUT	64	Terminal output (printer on batch)
READER	66	Virtual card reader
PRINTER	67	Virtual printer
PUNCH	69	Virtual card punch
DUMMY	71	Dummy input/output

The general form of the /FILE statement is given here. The only parameter which must be present on any /FILE statement irrespective of the unit type is the UNIT parameter. Any other parameter may be required for some unit types and optional for others (see the matrix on page 22). A parameter which is undefined for the type of unit specified will be ignored. A maximum of 200 units may be defined for any given job.

```

/FILE {UNIT=( [n,] {type} [,NAME=ddname] [,COND] )}
      [,DSNAME=dsname]    [,VOL=SER=volser]
      [,RECFM=r]    [,LRECL=n]    [,BLKSIZE=n]
      [,KEYLEN=n]    [,LEVELS=n]
      [ ,SPACE=( REC [(m,n)] ) ] [ DISP=( OLD ,KEEP ,KEEP ) ]
      [          BLK          ] [     SHR ,DELETE ,DELETE ]
      [          TRK          ] [     NEW          ]
      [          CYL          ] [     MOD          ]
      [,CLASS=c]    [,DEN=n]    [,BUFNO=n]
      [ ,LABEL=( n ,NL ) ]    [,LIMCT=n]    [,DSORG=dd]
      [          ,BLP ]
      [,OUTLIM=n]    [,COPIES=n]    [,OPT=option]

```

We make here such remarks as may be put forth without undue reference to specific unit types. For detailed discussions, refer to the appropriate sections of this manual (see table on page 19).

#### UNIT

The field designated "type" in this parameter defines the unit type and must be one of the unit types shown in the table on page 19 (LIBIN, LIBOUT, etc.).

If n is specified, the unit will be assigned the unit number n when it is created and may be accessed by the program this way. The value of n must be between 1 and 200. If NAME=ddname is specified, where "ddname" is a one to eight character name, the "ddname" will be assigned to the unit. (The term ddname comes from OS, where it refers to the name identifying a DD statement, the OS equivalent of the VPS /FILE statement. Since most of the languages on VPS which reference units by ddname were originally implemented on OS, this terminology seems the most appropriate.) Which of these is appropriate depends on what programming language is being used; both n and NAME may be given, in which case the unit may be referred to either way. If neither n nor NAME is given, the system will consider the file definition to be in error, since there would be no way for any program to access the unit.

The parameter COND may be given, indicating that this unit definition is conditional - that is, this /FILE statement is to be ignored if a /FILE statement defining a unit with the same unit number or ddname has appeared previously in the job. This is useful when the user wishes to put together a job containing unit definitions which may then be optionally overridden at the time of execution by appending the prepared job to the overriding /FILE statements.

#### DISP

The DISP parameter is used to determine the initial and final disposition of the file (library file or work file) being accessed by the unit. The first field specifies the initial condition, the second specifies the final disposition for normal termination of the job, and the third designates the final disposition for cases of abnormal termination. Abnormal termination includes program cancellation and termination caused by program errors.

#### LRECL

The LRECL parameter, when defined for a unit type, specifies the logical record length of the file. This does not mean that the program accessing the file must use this length for reading or writing: The system will truncate or pad any records as necessary when they are passed between the program and the file. For a more detailed discussion, see the "VPS System Facilities for Assembler Programmers" manual.

#### RECFM

This parameter also refers to the file, not to the form of the records being passed to or from the program. Any necessary conversions will be made by the system. That is to say, a program may write fixed length records to a file containing variable length records, read variable length records from a file containing fixed length records, and so forth. The format of the records seen by the program is determined by the program itself at the time the input or output is requested. How this is done and whether it is under the user's control depends on which programming language is being used. In most cases, the user need not be concerned with this problem.

#### DSNAME

This parameter, where applicable, defines the dataset name (library file name, work file name, etc.) of the file to be accessed by the unit being created. If the name contains only alphameric characters and periods, it may be substituted directly for "dsname". If the name contains other characters as well, it should be enclosed in apostrophes, with any apostrophes in the name doubled. The DSNAME keyword may be abbreviated DSN, if desired.

Matrix of Required and Optional /FILE Parameters

We give below the matrix of fields which are required (R) or optional (O) for the various unit types. If a matrix element is blank, the field is undefined for that unit type.

	L	L	T	T	D	T	R	P	P	D
	I	I	E	E	I	A	E	R	U	U
	B	B	R	R	S	P	A	I	N	M
	I	O	M	M	K	E	D	N	C	M
	N	U	I	O			E	T	H	Y
		T	N	U			R	E		
				T				R		
UNIT	R	R	R	R	R	R	R	R	R	R
DSNAME	O	O			O					
VOL						R				
RECFM				O	O	O		O		
LRECL	O	O			O	R				
BLKSIZE					O	R				
KEYLEN					O					
LIMCT					O					
SPACE					O					
LEVELS	O									
DISP	O	O			O			O	O	
CLASS							O	O	O	
LABEL						R				
DEN						O				
BUFNO					O	O				
DSORG					O					
OUTLIM								O	O	
COPIES								O	O	
OPT					O			O		

Summary of Default Input/Output Units

Unless the /FILE statements appearing in a job indicate otherwise, the system will provide certain units automatically when the job is run. A default unit will not be defined if a /FILE statement supplied in the job defines a unit with the same unit number or ddname. We give below the /FILE statements equivalent to the default units for jobs run at the terminal and for jobs run on the batch. For details about each unit, see the appropriate section of this manual (refer to the table on page 19).

For a job run at the terminal, the default units are equivalent to:

```
/FILE UNIT=(5,LIBIN,NAME=SYSIN),DSNAME=<input stream>,
/   LEVELS=5
/FILE UNIT=(6,TERMOUT,NAME=SYSPRINT),RECFM=FA
/FILE UNIT=(9,TERMIN)
/FILE UNIT=(10,LIBOUT,NAME=SYSPUNCH),DSNAME=*2,
/   DISP=(OLD,KEEP,KEEP)
```

while a job run on the batch receives:

```
/FILE UNIT=(5,LIBIN,NAME=SYSIN),DSNAME=<input stream>,
/   LEVELS=5
/FILE UNIT=(6,TERMOUT,NAME=SYSPRINT),RECFM=FA
/FILE UNIT=(7,PUNCH,DSNAME=SYSPUNCH)
/FILE UNIT=(10,LIBOUT,NAME=SYSSAVE)
```

where DSNAME=<input stream> is used to indicate that the LIBIN unit is initially defined (at start of job) to be the job input stream. That is, if the job is being run at the terminal via /EXEC (page 120), then unit 5, ddname SYSIN, will be set up initially as though it were reading a library file at the top level consisting of a single /INCLUDE statement (see page 39), where the parameters on the /INCLUDE statement are those given on the /EXEC command. If the job is being run on the batch and was submitted via the /BQx command (page 98), this unit will be similarly defined, except that the /INCLUDE parameters are taken from the /BQx command. If the job is being run on the batch and was submitted on cards, then this unit will be initially defined at the top level to consist of the series of cards which constitute the job. In any case, 4 additional levels are available for resolution of nested /INCLUDE statements. This unit may be modified by the program to access a different library file at the top level if the programming language being used permits (see "VPS System Facilities for Assembler Programmers" and the "VPS Guide to Programming Languages").

Besides the units described above, other default units will be created when certain language processors or system utilities are invoked. These are specific to the particular processor or utility and are described in the appropriate manuals ("VPS Utilities" and "VPS Guide to Programming Languages").

In addition to providing default input/output units, the system

will also provide a default unit translation (see TRANS, page 26) to make the terminal and batch environments a bit more compatible.

If the default unit 10 is in effect for a terminal job, and unit 7 has not been defined or rerouted to another unit, a TRANS=7->10 will be done by the system, so that program output to unit 7, which would be the card punch if the job were being run on the batch, will be sent to the LIBOUT unit defined as unit 10.

Similarly, for a job run on the batch, if unit 5 has not been redefined by the job, and unit 9 has not been defined or rerouted to another unit, a TRANS=9->5 will be done. Therefore, reads issued to what would be the terminal keyboard if the job were running at the terminal will be directed to the input stream.



The /LOAD Statement

The /LOAD statement defines the operating environment for the load module to be executed and specifies the name of that load module. The load module will be a language processor (PLIOPT, VSASM, FORTG, etc.), loader (LOADER), or other system facility (e.g., OSLOAD, COPY, LKED); or a user load module which has been created with the system linkage editor (see the "VPS Utilities" manual).

The general form of the /LOAD statement is

```

/LOAD  [ (lname ,LIB=syslibnm ) ]  [,PARM=pppp]
       [ ,FILE=fileid ]
       [,NSEGS=( [m] [,n] ) ]  [,TRANS=(m->n,...)]
       [ ,OPT= NOCC ]          [,TIME=t]
       [ NOSKIP ]

```

where the parameters have the following meanings:

The positional parameter

The first field of this parameter, shown as lname, gives the name of the load module to be executed. If the load module name is omitted, the name LOADER is assumed.

If the load module is in the standard system load module library, the LIB and FILE parameters are omitted. If the load module is in a system load library other than the standard one, the LIB parameter must be given, with syslibnm replaced by the name of the library (at this writing, no such load libraries exist). If the load module is in a user load library, the FILE parameter must be specified, with fileid replaced by either the unit number or the ddname (NAME parameter) specified on the /FILE statement defining the user load library.

PARM

This parameter specifies a character string which is to be passed to the load module when it begins execution. If the string does not contain blanks, commas, or other special characters, it may simply be substituted directly for pppp. If the string is a collection of alphameric strings, commas, and balanced sets of parentheses, it may be specified by enclosing the entire string in an additional set of parentheses. The outermost set of parentheses will not be passed on to the load module. Finally, any sort of string may be passed by doubling all apostrophes in the string and enclosing the result in single apostrophes. The string passed to the load module in this last case will be the coded string with the outermost apostrophes removed and all double apostrophes changed to single apostrophes.

## Examples:

parameter given	string passed to load module
PARM=LIST	LIST
PARM=(LIST,XREF(SHORT))	LIST,XREF(SHORT)
PARM='DONT''T BOMB'	DON'T BOMB

When the load module is entered, general register 1 contains the address of a fullword containing the address of the parameter string area. The parameter string area consists of a halfword containing the length of the string followed by the string. If no PARM field is specified, the halfword will be zero.

## NSEGS

This parameter specifies the amount of main storage to be allocated for the job. The unit of allocation is the 64K (64 X 1024 bytes) segment. The first value in the parameter, shown as m, indicates the amount of storage to be allocated for the program and its data areas (arrays, common, etc.), and the second, n, indicates the amount to be allocated for system storage (input/output unit control areas, buffers, and so forth).

If the first value is omitted, a value of 2 is assumed, making 128K available to the program. For performance reasons, the system imposes an upper limit on this value for each account. At this writing, the default limit is set to 8, allowing the user to allocate up to 512K. Users requiring more virtual storage should contact the Computing Center.

The second value rarely needs to be specified. The default for this value is 1 at the terminal and 2 on the batch. A larger value may have to be specified if an unusually large number of input/output units or units with unusually large physical block sizes are being defined.

## TRANS

This parameter specifies reroutings for input/output unit references. The reroutings are specified as entries of the form m->n, where m and n are unit numbers. This entry means "a reference by the program to unit m is to be redirected to unit n". If more than one such entry is given, the entries must be separated by commas and the list of entries enclosed in parentheses. The translation is only one level deep, so that if, for example, TRANS=(1->2,2->3) is specified, a program reference to unit 1 will access unit 2, not unit 3.

## OPT

This parameter sets options for the default TERMOUT unit (unit 6, ddname SYSPRINT). NOCC indicates that the records sent to this unit are not to be interpreted as having carriage control characters in column 1. The entire record will be printed verbatim. NOSKIP, which is meaningful only on the batch, prevents

the system from automatically skipping to the top of the next printer page each time the bottom of a page is reached. Printing will be allowed to continue undisturbed right over the perforations. Note that these options are not mutually exclusive, and both will be set if OPT=(NOCC,NOSKIP) is specified.

#### TIME

This parameter sets the CPU time limit and is meaningful only in jobs being run at the terminal (CPU time limits for the batch are controlled by the /ID statement, described on page 122). The system will make a reasonable effort to terminate the job after it has used the indicated amount of CPU time, but some overrun may be expected due to the nature of the internal timing mechanisms, particularly in jobs which do a relatively large amount of disk or tape i/o. The time t may be specified as a decimal number, indicating minutes (e.g., TIME=5 will set the limit to 5 minutes) or as a decimal number followed by the letter S, indicating seconds (e.g., TIME=30S sets the limit to 30 seconds).

The /JOB Statement

The /JOB statement is used to indicate whether the system should print out each system control statement (/FILE, /JOB, /LOAD) as it is read. In addition, the TRANS parameter (see page 26) may be specified on a /JOB statement. This is useful, for example, when a job has been set up in a library file, and the user wishes to run the job with one or more units rerouted without modifying the file, as might be done for testing purposes or to save in the library the output that would normally be printed. The /JOB statement has the format:

```
/JOB [PRINT ] [,TRANS=(m->n,...)]  
     [NOPRINT]
```

where the parameters are:

PRINT  
NOPRINT

The parameters PRINT and NOPRINT turn control statement printing on and off, respectively. If PRINT is specified, printing will commence with the statement following the /JOB statement on which PRINT appears. If NOPRINT is given, printing will be suppressed starting with the next statement. On the terminal, the initial setting is NOPRINT, while the initial setting on the batch is PRINT.

TRANS

This parameter specifies unit reroutings and is identical with the TRANS parameter on the /LOAD statement, described on page 26.

Symbolic Parameters

Normally, when a program is written or installed on VPS to be executed frequently, a library file is created containing VPS control statements needed to invoke the program (i.e. /FILE statements, /JOB statement, /LOAD statement, etc.). The general structure of a control statement stream to run a program appears below:

```

/JOB ...
/FILE ...
.
.
.
/FILE ...
/LOAD ...
      (input to the program)

```

For programs that are executed frequently which require modifications to these control statements (such as creating unique output file names for each program execution, changing the amount of virtual storage allotted for program execution, etc.), their associated execution files must be edited and resaved before each execution. The VPS symbolic parameter facility may be used to eliminate this chore.

A symbolic parameter is a symbol preceded by an ampersand (&) that represents a parameter, a subparameter, or value. In the following control statements the symbolic parameters are underlined:

```

/FILE UNIT=(&UTYPE,NAME=SYSPRINT),DSN=&FNAME
/LOAD PLIOPT,NSEGS=&VSTOR

```

When this file is executed (see the /EXEC command, page 120), each symbolic parameter must be given a value or nullified using two special VPS control statements (/DFLT and /SET) which will be discussed below. The changes are in effect only for that execution and the control statements that are modified are not changed in the file.

Defining Symbolic Parameters

Any parameter or subparameter that may vary with each execution is a good candidate for definition as a symbolic parameter. A symbolic parameter is one to seven alphanumeric characters (the first must be alphabetic) preceded by an ampersand. Note that since VPS recognizes the ampersand as the start of a symbolic parameter, a double ampersand must be coded in control statements when not identifying a symbolic parameter. For example:

```

/FILE UNIT=(DISK,NAME=FT12FO01),DISP=(NEW,DELETE,DELETE),
/ DSN=&&FT12,SPACE=(TRK,&NUMTS)

```

In the above, &&FT12 represents a temporary file name, whereas &NUMTS is a symbolic parameter.

A period is required after a symbolic parameter when the symbolic parameter precedes either an alphameric character or a period in a control statement. The system recognizes the period as a delimiter and the period will not appear in the generated control statement. (A single period will appear when two periods are coded immediately following a symbolic parameter.)

### Assigning Default Values to Symbolic Parameters

The /DFLT statement is used to assign default values to symbolic parameters in a control statement stream. The format of the /DFLT statement is written below.

```
| /DFLT {symbol} = [string],... |
```

symbol

specifies the symbolic parameter (without the ampersand) to which the default is to be given.

string

specifies the default value to be assigned to the symbolic parameter. The maximum length of the string is 255 characters. If the string contains either blanks, commas, ampersands, or other special characters it must be enclosed in apostrophes. If the string is omitted, a null value will be assigned to the symbolic parameter.

The following example shows the use of the /DFLT statement to assign default values.

```
/DFLT STOR=,BLK=4560,UTYPE=LIBIN,FILE=MYINPUT,  
/ PARM='MAP,LIST'  
/FILE UNIT=(&UTYPE,NAME=INPUT),DSNAME=&FILE,DISP=SHR  
/FILE UNIT=(DISK,NAME=OUTPUT),DSNAME=UR.HBK200.OUTPUT,DISP=OLD,  
/ RECFM=FB,LRECL=80,BLKSIZE=&BLK  
/LOAD LOADER,PARM='&PARM',NSEGS=&STOR
```

If a program execution occurred using the above control statement stream and none of the defaults were overridden, the following control statements would be in effect for execution.

```
/FILE UNIT=(LIBIN,NAME=INPUT),DSNAME=MYINPUT,DISP=SHR  
/FILE UNIT=(DISK,NAME=OUTPUT),DSNAME=UR.HBK200.OUTPUT,DISP=OLD,  
/ RECFM=FB,LRECL=80,BLKSIZE=4560  
/LOAD LOADER,PARM='MAP,LIST',NSEGS=
```

Note that since the NSEGS= parameter has been nullified it has no effect on the job.

Overriding Default Values

The /SET statement is used to set or override default values of symbolic parameters in a control statement stream. The format of the /SET statement is written below.

```
| /SET {symbol} = [string],... |
```

**symbol**

specifies the symbolic parameter (without the ampersand) to which the value is to be given.

**string**

specifies the value to be assigned to the symbolic parameter. The maximum string length is 255 characters. If the string contains either blanks, commas, ampersands, or other special characters it must be enclosed in apostrophes. If the string is omitted, a null value will be assigned to the symbolic parameter.

In the following example the /SET statement is used to override certain defaults.

```
/SET STOR=6,FILE=,UTYPE=TERMIN
/DFLT STOR=,BLK=4560,UTYPE=LIBIN,FILE=MYINPUT,
/ PARM='MAP,LIST'
/FILE UNIT=(&UTYPE,NAME=INPUT),DSNAME=&FILE,DISP=SHR
/FILE UNIT=(DISK,NAME=OUTPUT),DSNAME=UR.HBK200.OUTPUT,DISP=OLD,
/ RECFM=FB,LRECL=80,BLKSIZE=&BLK
/LOAD LOADER,PARM='&PARM',NSEGS=&STOR
```

The above example would generate the following control statement stream:

```
/FILE UNIT=(TERMIN,NAME=INPUT),DSNAME=,DISP=SHR
/FILE UNIT=(DISK,NAME=OUTPUT),DSNAME=UR.HBK200.OUTPUT,DISP=OLD,
/ RECFM=FB,LRECL=80,BLKSIZE=4560
/LOAD LOADER,PARM='MAP,LIST',NSEGS=6
```

System Defined Symbolic Parameters

VPS provides five system defined symbolic parameters which can be used in control statements. The system automatically sets these parameters at each program execution. If any of these symbols appears on /SET or /DFLT statements in a control statement stream the system defined value will be overridden. The symbolic parameters are:

&SYSACCT - will be replaced with the account number of the job (on the batch) or terminal session.

&SYSPID - will be replaced with the last three characters of the account number of the job (on the batch) or terminal session. (Note: the last three characters of course account numbers are the Personal ID of the student or instructor owning that account.)

&SYSDATE - will be replaced with the current date in the form - YMMDD.

&SYSTIME - will be replaced with the current time in the form - HHMMSS.

&SYSTEM - in a terminal execution will be replaced with a null string. In a batch job it will be replaced with the string '/\*'.

&SYSBAT - in a terminal execution will be replaced with the string '/\*\'. In a batch job it will be replaced with a null string.

Assume that the following control statements exist in a VPS library file.

```
/FILE UNIT=(DISK,NAME=INPUT),DSN=UR.&SYSACCT..DATA,DISP=SHR
/&SYSBAT.FILE UNIT=(DUMMY,NAME=SYSPUNCH)
/LOAD LOADER,PARM='/&SYSDATE,&SYSTIME'
```

If this file were executed on 25 January 1979 at 8:15:05 AM during a terminal session under the account HBK412, the following control statements would be generated:

```
/FILE UNIT=(DISK,NAME=INPUT),DSN=UR.HBK412.DATA,DISP=SHR
/** UNIT=(DUMMY,NAME=SYSPUNCH)
/LOAD LOADER,PARM='/790125,081505'
```

Note that the /FILE statement for SYSPUNCH has been turned into a comment and the system defined file for SYSPUNCH will be used. Now assume that the same file were executed the next day on the batch under the account UTL293 at 2:32:37 in the afternoon. The following control statements would be generated:

```
/FILE UNIT=(DISK,NAME=INPUT),DSN=UR.UTL293.DATA,DISP=SHR
/FILE UNIT=(DUMMY,NAME=SYSPUNCH)
/LOAD LOADER,PARM='/790126,143237'
```

### Using Symbolic Parameters

/SET and /DFLT statements may appear anywhere in a control statement stream. The value of a symbol at any given point in the stream will be equivalent to the last value it was set to on a /SET statement or its default from a /DFLT statement. Symbols may be reset at any time in the control statement stream using a /SET statement (note that the /DFLT statement will never override a previously set symbol). For example, the following control statements -

```
/DFLT NAME=KEYS,TYPE=LIBIN
```



```

/FILE UNIT=(&TYPE,NAME=FT02F001),DSN=&NAME
/SET TYPE=DISK
/FILE UNIT=(&TYPE,NAME=FT14F001)
/SET TYPE=TERMOUT
/FILE UNIT=(&TYPE,NAME=FT06F001)
/LOAD FORTX

```

would generate the following:

```

/FILE UNIT=(LIBIN,NAME=FT02F001),DSN=KEYS
/FILE UNIT=(DISK,NAME=FT14F001)
/FILE UNIT=(TERMOUT,NAME=FT06F001)
/LOAD FORTX

```

Symbolic parameters need not appear on /DFLT statements. However, if a symbol is neither defaulted on a /DFLT statement nor set on a /SET statement before its use, VPS will issue an error message and terminate the execution. If a symbol is not used in a control statement stream yet it has appeared on a /SET statement, VPS will issue a warning message.

Usually, a VPS library file is created containing /DFLT statements followed by the skeleton control statements (though, of course, there is no requirement that these statements appear in the same file, see the discussion of the /INCLUDE statement, page 39). Assume, for example, that the file BASEX contains the following:

```

/DFLT VIRT=6,INUTYPE=LIBIN,LSTTYPE=TERMOUT,LSFILE=
/FILE UNIT=(&INUTYPE,NAME=INPUT),DSN=&INFILE,DISP=SHR
/FILE UNIT=(DISK,NAME=OUTPUT),DSN=UR.&SYSACCT.OUTPUT,DISP=OLD
/FILE UNIT=(&LSTTYPE,NAME=SYSPRINT),DSN=&LSFILE,DISP=(NEW,KEEP)
/LOAD PLIOPT,NSEGS=&VIRT
      (object)
/DATA

```

Assume the file SQUARES contains:

```

/SET LSTTYPE=LIBOUT,LSFILE=*2,INFILE=MATRIX

```

A user logged on to HBK412 wishing to execute BASEX using the data file DATA78 and the file SQUARES might type (at "\*go" level);

```

/ex squares,basex,data78

```

The effective job stream for this execution appears below.

```

/FILE UNIT=(LIBIN,NAME=INPUT),DSN=MATRIX,DISP=SHR
/FILE UNIT=(DISK,NAME=OUTPUT),DSN=UR.HBK412.OUTPUT,DISP=OLD
/FILE UNIT=(LIBOUT,NAME=SYSPRINT),DSN=*2,DISP=(NEW,KEEP)
/LOAD PLIOPT,NSEGS=6
      (object)
/DATA
      (data)

```

Note that the same program execution could have been achieved by using the following at "\*go" level:

```
/ex '/set lsttype=libout,lsfile=*2,infile=matrix',basex,data78
```

In which case the file SQUARES is not used. This might be more convenient if repeated executions of BASEX with these parameters is not anticipated.

### Chapter 3: Library Files

The VPS library is used for the great majority of program and data storage applications. The user creates files in the library using the editor (/EDIT, page 119) or some other system utility, or by writing to a unit of the type LIBOUT in a program. A number of facilities are provided which allow the user to list, modify, delete, rename, and control access to these files.

The library is a space-efficient file system designed to provide the user with a sequentially accessible storage facility. Unlike the more flexible work file (see page 47), a library file cannot be accessed randomly or updated in place and cannot have a logical record length greater than 255. A little reflection, however, will show that for many program and data storage applications these restrictions are irrelevant, since most such files are read or written only sequentially and need not contain records longer than the 255 character limit.

These restrictions on library files are related to the internal structure of the file system. Briefly, records written into the library are compressed by the deletion of trailing blanks and the reduction of all embedded strings of three or more contiguous blanks to two bytes of control data. These compressed records are placed in 512 byte blocks along with an additional 12 bytes of control information per block and written onto disk. The required disk space is allocated from a common pool of free blocks in "space sets" of four contiguous 512 byte blocks each. (These space sets are known colloquially as "library tracks" but do not correspond to physical tracks on the disk. For IBM 3350 disks, six and three-quarter space sets fit on one physical track.) The first block of any library file is called the file directory and does not contain any data records. Instead, this directory block contains information about the file such as its name and owner, public and special access codes, logical record length, number of records, and a table describing the disk locations of all the space sets in the file, up to a maximum of 135 space sets (if a file exceeds this number of space sets, additional directory blocks will be automatically placed in the file as required). This structure permits (typically) two to four times as much information to be stored in a given amount of disk space as more conventional schemes.

Each file placed in the library is pointed to by an index. An index is a separate disk area containing an entry giving the name, owner, and disk location of each library file saved under that index. The full name of any library file has the format

INDEXNAME.FILENAME

where INDEXNAME is the one to eight character name of the index and FILENAME is the one to eight character file name to be found in that index. A file name must start with an alphabetic character and must consist entirely of alphabetic or numeric characters. The symbols \$,

@, and # are considered alphabetic characters for this purpose. Files saved by users are placed in the index USERLIB, which is assumed as the default index name for any file reference in which no index is specified. The user, therefore, does not have to type "USERLIB." in front of each file name. Since all user accounts save files in the same index, a file saved by one user can be referenced using the same name by any other user (assuming the file's access codes permit the reference) and once a particular file name has been taken by one user, it cannot then be used by anyone else. Indices other than USERLIB are used for certain groups of system files, such as assembler macro libraries, and are discussed in appropriate sections of this and other VPS manuals.

### Access Control

A file's owner may control three types of access to the file: READ, EXEC, and PURGE. The user of an account which has READ access to the file may list it or access it as input data for a program. If the account has EXEC access to the file, the account user may execute the file as a program. Finally, PURGE access to a file allows the file to be replaced or deleted by the account user. The account which owns a file always has READ, EXEC, and PURGE access to that file. Each file also contains indicators which specify which types of access accounts other than the owner are to be given. In addition, one other account or one range of accounts may be specified as having its own set of access specifications. Alternatively, another library file, rather than an account, may be given READ or PURGE access to a file. The specified access will be gained only when the program contained in the named file is executed (via the /EXEC command, page 120) and is independent of the account under which the program is run. The default values for these access indicators for a newly created file are specified in the user profile (see the /PROFILE command, page 135), and the values for a replaced file are inherited from the old file unless explicitly overridden.

Normally, unless the executing program has specific access to a referenced file, the account under which the program is being run will be used to determine whether or not the reference is to be allowed. However, in the case where a file is referenced by a /INCLUDE statement (see below) which is itself contained in the executed file (the file containing the /LOAD statement), the account which owns the executed file will be used, and READ access will be required. Thus, a user may create a file which uses, via /INCLUDE, other files to which that user's account has READ access and give the public or some group of accounts EXEC access to the top-level file, without having to make the included file directly accessible by that same group.

### Library Limit

Since library space is a limited resource, each account has associated with it a library space limit which specifies the maximum number of library space sets that can be occupied by that account's library files. No creation of a new file or replacement of an old file which would cause this limit to be exceeded will be allowed by the system.

Charges

Library space usage is charged for continuously by the system with a time resolution of about one minute. The accumulated charge is updated and the account's balance adjusted accordingly whenever the amount of space being used changes, and whenever the new charge increment is found to exceed ten cents at the time the system is brought up each morning. The currently accumulated library space charge may be determined via the /ACCT command (page 95) or the /PROFILE command (page 135). The /RATES command (page 139) may be used to determine the current charging rates.

The Temporary Files \* and \*2

The terminal user can work with two temporary files using the names \* and \*2. These files can be created, replaced, and read in the same manner as ordinary library files, but they are defined separately for each user account. Therefore, a user cannot refer to the \* or \*2 file of another user, and there is no name conflict between two or more users referencing \* or \*2. Also, the \* and \*2 files cannot be accessed from the batch, and the space they occupy is not included in the user's library space usage count. Each of these files may occupy a number of library space sets equal to the user's library space limit. These files are referred to as temporary files because they are purged by the system when it is freshly loaded each morning. Any data which the user has placed in the \* or \*2 file must be saved as a regular library file before the end of the day if it is to be preserved.

Reading Library Files in Programs

Three techniques are available which allow programs to read library files (program in this context includes compilers, assemblers, loaders, etc.); which is appropriate in a given situation depends on the programming language and the results desired (see the "VPS System Facilities for Assembler Programmers" manual and the "VPS Guide to Programming Languages").

The simplest approach, and the one generally used for compilations, execution of programs saved as object decks, and accessing of single data streams, is to include the file or files to be accessed in the data stream which comprises the batch or terminal job using one or more /INCLUDE statements, discussed below (see page 17 for an explanation of the structure of VPS jobs). Unless specifically redefined by a /FILE statement supplied by the user, this data stream is available to the program as unit 5 with ddname SYSIN.

The other two approaches are methods of defining which file is being accessed by a library input unit. A library input unit is a file of the type LIBIN, for which the general form of the /FILE statement is

```

/FILE {UNIT=( [n,] LIBIN [,NAME=ddname] )} [,DSNAME=dsname]

      [ ,DISP=  $\frac{\text{OLD}}{\text{NEW}}$  ]    [ ,LEVELS=  $\frac{1}{n}$  ]

```

where the meanings of the parameters are:

#### UNIT

The keyword LIBIN in this parameter indicates that this unit is to be a library input unit. If n is specified, the unit will be assigned the unit number n when it is created and may be accessed by the program this way. If NAME=ddname is specified, where "ddname" is a one to eight character name, the "ddname" will be assigned to the unit. Which of these is appropriate depends on what programming language is being used; both n and NAME may be given, in which case the input unit may be referred to either way. If neither n nor NAME is given, the system will consider the file definition to be in error, since there would be no way for any program to access the unit.

#### DSNAME

The optional parameter DSNAME specifies that the file "dsname" is to be read when the program accesses this unit. If DSNAME is omitted, and the program does not itself define which file is to be accessed, a read issued to this unit will raise an immediate end-of-file condition. The DISP parameter, described below, determines whether the file specified by the DSNAME parameter must exist at the time the job starts.

#### DISP

The DISP parameter specifies whether the system is to check for the existence of the file named in the DSNAME parameter at the time the job starts. If DISP=OLD is specified, the file must exist or the job will not be allowed to continue, whereas specification of DISP=NEW postpones the test for existence of the file until the program actually attempts to access this unit. The DISP=NEW parameter is useful when the file to be read is being created by the job that is reading it and therefore does not exist when the job starts.

#### LEVELS

This parameter indicates how many levels of /INCLUDE nesting are to be allowed for this unit. For example, if the file to be read by this unit contains a /INCLUDE statement and LEVELS=1 (which is the default) has been specified, then the /INCLUDE statement itself will be passed back to the program. On the other hand, if the LEVELS specification is 2 or greater, the file or files named on the /INCLUDE statement will be read and passed back to the program line for line when this point is reached in reading the top-level file.

Obviously, then, one way of defining the file to be accessed by a

LIBIN unit is to specify its name in the DSNNAME parameter of the /FILE statement for that unit. It is also possible in several of the available programming languages (see the VPS System Facilities for Assembler Programmers manual and the VPS Guide to Programming Languages) to redefine the file being read by any LIBIN unit or the normal job data stream (unit 5, ddname SYSIN, discussed above).

#### The /INCLUDE Statement

Library files may be nested by the inclusion of /INCLUDE (abbreviation /INC) statements. The general form of the /INCLUDE statement is

```

/INCLUDE field1[,field2[,...[fieldN]...]]
/INC

```

where the keyword /INCLUDE or /INC must begin in column 1 and at least one blank must separate it from "field1", which is taken as starting with the first non-blank character after /INCLUDE. Subsequent fields, if present, are each taken as starting with the first character (whether blank or non-blank) following the comma which defines the end of the preceding field.

Each of the fields must have one of the following forms:

#### 1) A Library File Name

When a library file name is encountered on a /INCLUDE statement, the nesting level will be increased by one and the entire file named will be read in and returned to the program line by line. Any field which starts with an alphabetic character or one of the symbols \$, @, and # or which consists of "\*" or "\*2" is assumed to be a library file name. The nesting level available to an included file can be explicitly reduced by the addition of a parameter of the form "(L=n)" immediately following the file name, where n is the number of nesting levels to be allowed to this file. In this context, (L=0) and (L=1) are synonymous.

As an example, suppose that a unit defined as having 5 nesting levels is reading a file at the top level, so that a maximum of 4 additional nesting levels are available. If the statement

```
/INCLUDE FILE2
```

is encountered, the nesting level will increase by one and FILE2 will be read, with 3 nesting levels still available for /INCLUDE resolution. If, on the other hand, the statement is

```
/INCLUDE FILE2(L=1)
```

then FILE2 will still be read, but it will be allowed no additional nesting levels, and any /INCLUDE statements encountered

in FILE2 will be returned to the program without inspection. Note that inclusion of the (L=n) parameter cannot increase the number of nesting levels that would otherwise be available (i.e. as implied by the number of levels defined for the input unit).

## 2) A Reference to Another Input Unit

If a field of the form "\*Un", where n is the unit number of any existing input unit, is found, the nesting level will be increased by one, and unit n will be read and returned to the program line by line. When end-of-file is encountered on unit n, the nesting level will decrease again, and normal processing will continue at the level where the /INCLUDE statement was encountered. If additional nesting levels are available when the \*Un field is encountered, /INCLUDE statements read from the specified input unit will be resolved. Apart from the finite number of available levels, the only restriction imposed on the nesting of \*Un references is that the LIBIN unit itself may not be referenced (other LIBIN units may, of course, be specified). Finally, if the input unit specified has a record length greater than 255, the records will be truncated to 255 bytes when they are read by the LIBIN unit.

## 3) A String of Literal Data

### a) Delimited Form

If the first character of a field is an apostrophe, the field is assumed to be a character string enclosed in apostrophes, and the string, with the enclosing apostrophes removed, will be returned as a record to the program. Apostrophes which are to appear in the string must be coded as double apostrophes, in the usual manner. This feature is useful for passing a line or two of data to a program on a /EXEC (page 120) or /BQx (page 98) command.

### b) Open Form

If the first character of a field is not alphabetic, is not \$, @, #, or an apostrophe, and the field is not \*, \*2, or of the \*Un form, then the field is assumed to be a string of literal data extending from that character to the next unpaired comma or the end of the /INCLUDE statement. Paired commas will become single commas in the string passed back to the program.

For example, consider a job for which the data stream is defined by

```
/INCLUDE PROG1  
/DATA  
1,2,3,4  
/INCLUDE DATA6
```

This may alternatively be specified as

```
/INC PROG1,/DATA,1,,2,,3,,4,DATA6
```



or as

```
/INC PROG1,/DATA,'1,2,3,4',DATA6
```

Thus, while the first form would have to be entered into a library file in order to be run as a job (or punched on separate cards for the batch), the second and third forms may be entered directly as arguments of /EXEC or /BQx.

### Creating Library Files

The creation of a library file involves writing the records which are to constitute the file to an output unit of the type LIBOUT and then saving the resulting file under a new name or replacing an already existing file with the new file. Various system programs exist which perform this function, most notably the editor (see page 119), and user programs may be written in most of the available programming languages to perform the same function. The form of the /FILE statement which is used to define a LIBOUT unit is:

```
/FILE {UNIT=( [n,] LIBOUT [,NAME=ddname] )} [,DSNAME=dsname]
      [ ,DISP=( OLD ,KEEP ,KEEP ) ] [,LRECL=m]
      [ ,NEW ,DELETE ,DELETE ]
```

where the parameters have the following meanings:

#### UNIT

The keyword LIBOUT in this parameter indicates that this unit is to be a library output unit. If n is specified, the unit will be assigned the unit number n when it is created and may be accessed by the program this way. If NAME=ddname is specified, where "ddname" is a one to eight character name, the "ddname" will be assigned to the unit. Which of these is appropriate depends on what programming language is being used; both n and NAME may be given, in which case the output unit may be referred to either way. If neither n nor NAME is given, the system will consider the file definition to be in error, since there would be no way for any program to access the unit.

#### DSNAME

The optional parameter DSNAME specifies that the file created by writing to this unit is to be saved in the library under the name "dsname" when the job terminates. If DSNAME=\* or DSNAME=\*2 is specified, the \* or \*2 file, respectively, will be replaced by the new file. The action taken by the system in the case where the file to be saved already exists depends on the DISP parameter, described below.

#### DISP

The first subfield of this parameter indicates whether the system is to replace an existing file if one is found. If NEW is specified and the file named in the DSNAME parameter is found to already exist, the new file will not replace it, and an error message will be produced. If OLD is specified, the new file will be saved under the specified name or will replace the file already saved under that name if it exists (access codes permitting, of course). The second and third subparameters indicate whether the file saving action is to be attempted if the job terminates normally or abnormally, respectively. (Abnormal termination includes termination via the /CANCEL command and termination caused by programming errors detected by the system.) If KEEP is specified, the save will be attempted, whereas DELETE will cause the new file to be deleted and no save to be done. For example, a specification of DISP=(OLD,KEEP,DELETE), which is the default if no DISP parameter is explicitly given, means that if the job terminates normally the newly written file is to replace the file specified by the DSNAME parameter, or a new file of this name saved if the file did not previously exist; and that if the job terminates abnormally the new file is to be deleted. If the program terminates without having written any records to the LIBOUT unit, no action will take place, and any previously existing file will remain intact.

#### LRECL

If LRECL is specified, the file produced by writing to this LIBOUT unit will have the stated logical record length. The value *m* must be between 20 and 255. If LRECL is not specified, the system will assign a record length equal to the length of the longest record written to the unit, except that records longer than 255 bytes will be truncated to 255 and the minimum value that will be set is 20. Note that due to the compression scheme described above the record length does not affect the amount of disk space actually occupied by the file as long as the number and arrangement of non-blank characters in the file's records remain the same.

Since it is often necessary for programs to write into the library, the system provides a default LIBOUT unit with the unit number 10, unless the user specifically overrides this unit number with a /FILE statement. The /FILE statement which is equivalent to this default unit is:

```
/FILE UNIT=(10,LIBOUT),DSNAME=*2,DISP=(OLD,KEEP,KEEP)
```

Thus, if a program writes to unit 10 and then terminates, the records written to this unit will be found in the temporary file \*2 and may be listed, read by another program, or saved as a permanent library file. If the program is running on the the batch, the file will be deleted, since \* and \*2 cannot be created or accessed by batch jobs.

Once a group of records have been written to a LIBOUT unit, they must be saved in the library if they are to become a library file. This may be done automatically by the system through the DSNAME parameter, as described above, or it may be done directly by a program

(see the VPS System Facilities for Assembler Programmers manual or the VPS Guide to Programming Languages) or by the use of one of several system commands which save the \* or \*2 file as a permanent file or replace an existing library file with the \* or \*2 file. These commands, which are valid only at "\*go" level, are /SAVE (page 149), which is used to save either the \* or the \*2 file as a new permanent file; /RSAVE (page 147), which is used to replace an existing library file with the \*2 file; and /REPLACE (page 143), which is used to replace an existing file with the \* file.

#### Listing the Names of Saved Files

The user may list the names and attributes of saved files by issuing the /LIBRARY command (page 124). This command will produce a list of the names, record counts, record sizes, disk space counts, creation times and dates, dates of last reference, and access codes of all the files saved under the user's account or, if specified, of selected files or other accounts (the latter requires a privilege). The /LIBRARY command will optionally produce the file name listing as a library file which can then be read by a program, allowing the user to produce and process lists of library file names with a minimum of effort.

#### Changing the Names and Access Codes of Library Files

The /RENAME command (page 140) may be used to change the name of a library file or to modify the file's access codes. The account attempting to perform the /RENAME function on a file must have PURGE access to that file, or the request will be rejected. For example, to change the name of the file TESTPROG to PROG6, one would issue the command

```
/RENAME TESTPROG,PROG6
```

This command is also available as a program, called RENAME, which may be run using the /EXEC command (page 120) which will read in an arbitrarily large number of arguments in the same format as the argument of the /RENAME command and perform the indicated operations. This is useful when a large number of file changes are to be made, as when a collection of files is being given a different set of access codes. The data to be read by the RENAME program must consist of the appropriate arguments arranged in a file so that the first non-blank character in each line of the file is the first character of a /RENAME-type argument. Each line may contain only one such argument, and each argument must be followed by at least one blank.

Since this is perhaps best described through an example, let us consider how we might change all the files belonging to the account we are currently using to have the access code READ. First, we produce a list of all these files using the /LIBRARY command by issuing

```
/LIBRARY BRIEF,SAVE
```

When this command has completed, and the terminal has returned to "\*go" level, we will find the list of file names in the \*2 file.

Suppose that this list consists of

```
FILE1
FILE6
MORJOKES
YAWN
```

We now edit the \*2 file (/EDIT, page 119) and replace it with the edited version so that it becomes

```
FILE1,,READ
FILE6,,READ
MORJOKES,,READ
YAWN,,READ
```

and \*2 is now a file containing legal arguments for the /RENAME command. Finally, we complete the task by running the program RENAME with our \*2 file as data, using /EXEC:

```
/EXEC RENAME,*2
```

and the files in our library now have their access codes modified as we wished.

#### Purging Files from the Library

The process of deleting a file from the library is referred to as purging the file and is performed by the /PURGE command (page 136). For an account user to purge a file the account must have PURGE access to the file. If, for example, we wish to purge the file OLDPROG, we issue (at "\*go" level):

```
/PURGE OLDPROG
```

and the system will purge the file.

Like the /RENAME command, /PURGE is available as a program, called PURGE, which will read in a list of file names and perform the /PURGE function on each file. The data file for PURGE must consist of a list of valid /PURGE arguments. Note that the unmodified SAVE output from the /LIBRARY command constitutes such a list: For instance, if we had wished to purge the files discussed above in the example under "Changing the Names and Access Codes of Library Files", we could have fed the \*2 file produced by the /LIBRARY command directly to the PURGE program by issuing

```
/EXEC PURGE,*2
```

immediately after the /LIBRARY command had terminated, and all the files in the list would have been purged.

Listing Library Files at the Terminal

If a user's account has READ access to a library file, the file may be partially or completely listed at the terminal with either the /LIST command (page 126) or the /DISPLAY command (page 114). The only difference between the two commands is that /DISPLAY prefixes a line number to each file record typed out and /LIST does not. Neither of these commands is available on the batch.

Listing Library Files on the Batch

Users wishing to list files on the batch may use the program FILELIST, which will produce a printer-formatted listing complete with leading banner pages, page titles, and a record count. The proper form of a batch job for listing files BIGSTUFF and LILSTUFF is

```

/ID account          (see page 122)
/INC FILELIST
BIGSTUFF
LILSTUFF
/END

```

Any number of files may be listed by a single VPS batch job using this program; a new banner page will be produced for each file.

Making Little Holes in Cardboard

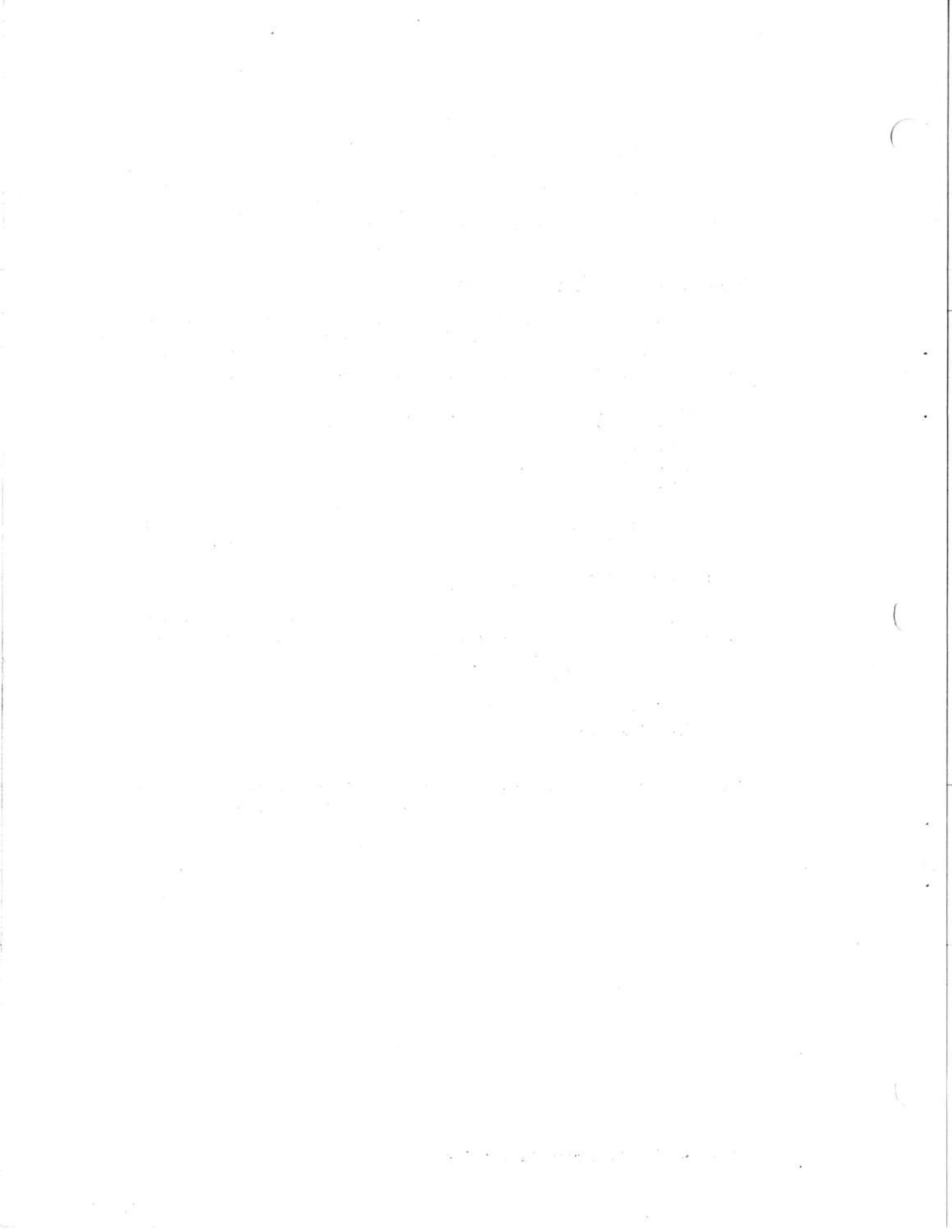
Library files may be punched out on the batch through the machinations of the program PUNCH, which simply copies all the data it reads to the card punch. The batch job which would be used to punch out the file DATACRDS is

```

/ID account
/INC PUNCH,DATACRDS
/END

```

Users should bear in mind that punched cards are not an appropriate medium for files containing records longer than 80 bytes.



## Chapter 4: Work Files

In addition to the library file (page 35), which is used for most program and data storage applications, VPS supports a type of disk file called the work file. A work file is an OS compatible dataset of either the PS type (physical sequential, as supported by BSAM) or the DA type (direct access, as supported by BDAM). This file consists of a specifically reserved disk area containing blocks of user-specified length. The blocks may contain fixed length or variable length records and may or may not have keys, depending on the specifications given when the file was created.

Compared with library files, a work file has both advantages and disadvantages. On the positive side, a work file can be accessed either sequentially or randomly, updated in place, searched by key (if it has keys), and a single unit (defined by a single /FILE statement) can be used by the program to both read from and write to the file. The physical block size is limited only by the capacity of a disk track (19069 bytes for IBM 3350 disks).

On the other hand, unlike a library file, a work file's records are not compressed by the system, the space occupied by the file does not expand or contract when the number of records in use increases or decreases (although for PS files the user may request that extra chunks of space be added to the file automatically when the existing allocation overflows), and the creation of work files is somewhat less convenient than the creation of library files. Also, a work file is modified by overwriting the old records with the new records, whereas a library file is modified by writing out a complete new set of records in a different disk location and then changing the index pointer for that file. Consequently, a library file is far less likely to be damaged than a work file if something goes awry in the midst of an update.

These considerations, then, must be weighed in the balance in deciding whether a particular situation indicates the use of a work file rather than a library file. If it appears that the ability to access the data randomly, the higher limit on logical record length, or the possibility of performing a simple write-rewind-read sequence on a single unit outweighs the loss in economy and safety or the greater inconvenience, then a work file is the indicated choice. The user should consult the appropriate manual ("VPS System Facilities for Assembler Programmers", "VPS Guide to Programming Languages") to determine whether restrictions imposed by the programming language being used will affect this decision.

It must be noted that some compilers and other programs require the use of work files for their temporary work areas or even for permanent data storage (e.g. SPSS, Datatext). The user should check out the related documentation if there is a question about a specific case.

### Structure of a PS Work File

A PS (physical sequential) work file consists of a permanently allocated disk area onto which data blocks are written in sequential order. The records of which the blocks are composed may be either fixed length or variable length format, or the file may simply be treated by the program at the block level, and the contents of the blocks will be ignored by the system. A given programming language may restrict the user to only some of the available formats and access methods, so the user should read the documentation for the language being used.

While a PS file may be treated as a random access file if the programming language permits - that is, blocks may be read and overwritten ("updated in place") in any order, the usual mode of use is sequential reading and writing of the file. In any case, the initial creation of the file must take place sequentially, as each block put into the file in normal output mode is written out using the "formatting" write operation ("write count, key, and data"), which destroys all data on the accessed track following the block being created. This technique allows each new block of data to have a length different from any block or blocks it overwrites, but at the same time obviously restricts normal output to sequential writing.

### Structure of a DA Work File

VPS supports two types of DA (direct access) files. The first type is the standard IBM VS BDAM file used by the major programming languages (REGIONAL in PL/I, file organization D, W, or R in COBOL, DEFINE FILE in FORTRAN G1 or H Extended, or BDIO and BIO in Assembler Language). DA files identical to those created on VS of the record formats F, U, and V (unspanned), with or without keys, can be created and accessed on VPS. The second type is a restricted form of DA file permitting only fixed length records without keys created and accessed using QIO. (Note that QIO allows blocked DA files - not available in BDAM.) This type is used by the Linkage Editor, FORTG (an old version of FORTRAN G1 no longer supported by the Computing Center), and can be used through DIO or QIO from Assembler Language. More technical discussions of these two file types can be found in the "VPS Guide to Programming Languages" and the "VPS System Facilities for Assembler Programmers" manual.

A DA file, like a PS file, consists of a permanently allocated disk area. However, the DA file is specifically intended for random access - either by block number addressing or by key search. A newly created fixed DA file must be formatted before it is used for random access, as all random access writes to a fixed DA file are update-type writes and require that the target block already exist. Most programming languages (e.g. PLIOPT, FORTX, COBOL) provide methods for formatting fixed DA files (see the documentation for the language being used). However, if the user is writing in Assembler Language or using a language or program which does not provide this service (e.g. FORTG or the Linkage Editor), then the file must be formatted in some other way. The simplest approach is to request on the /FILE statement which creates the file that the system automatically format the file



when the disk space is allocated (see the OPT parameter, below). For files of record formats V and U, formatting is not done. Instead when the file is first created, only those areas of the file where records are written are "formatted" and the program must leave space available in the file if records are to be added later.

### The VTOC and the Catalog

Each work file is described by an entry called a dataset control block (DSCB) in an area of the disk called the volume table of contents (VTOC). The DSCB contains, among other things, the block size, key length, logical record length, and dataset organization (PS or DA) of the file. Therefore, it is not necessary to specify these values on a /FILE statement used to access an already existing work file.

There is one VTOC for each physical volume (disk pack) on the system; each VTOC contains DSCBs only for files allocated on its own volume. However, it is not necessary to know the name of the volume containing a work file in order to access it, as VPS also maintains a central catalog containing the volume serial and other related information for every work file on the system. When a new file is created, VPS automatically selects a volume on which to allocate the requested space and puts an entry in the catalog. When a file is deleted, its entry is automatically removed from the catalog.

The /DS command (page 115) may be used to list the names of work files belonging to a given account. Other parameters associated with each file, such as the LRECL, BLKSIZE, and SPACE values, will also be printed.

### Work File Track Limit

Each account has associated with it a work file track limit and a count of the work file tracks currently in use. The user will not be allowed to create any new work file if the updated work file track count for the user's account would exceed the account's limit. Also, VPS will not allow a work file to be extended (see the SPACE parameter, below) if the extension would cause the account owning the file to exceed its limit.

### Charges for Work File Space

Charges for work file space are computed continuously with a time resolution of about one minute. The current charge and the user's balance are adjusted appropriately any time the number of work file tracks in use changes (i.e., whenever a work file is created or deleted) and whenever the new charge increment exceeds ten cents at the time the system is brought up in the morning. The charge for the current billing period (normally starting at the beginning of the current month) can be determined via the /ACCT (page 95) or /PROFILE (page 135) command. The /RATES command (page 139) may be used to list the current charging rates.

Defining a Work File Input/Output Unit

A work file is accessed via input and output requests to a unit of the type DISK. The general form of the /FILE statement for a DISK unit is:

```

/FILE {UNIT=( [n,] DISK [,NAME=ddname] )} [,DSNAME=dsname]

      [,LRECL=n]          [,BLKSIZE=n]

      [,KEYLEN=n] [SPACE=( REC , (m,n) ) ] [RECFM= F ]
                  [BLK      ]             [FB      ]
                  [TRK      ]             [FBS     ]
                  [CYL      ]             [V       ]
                                      [VB      ]
                                      [VS      ]
                                      [VBS     ]
                                      [U       ]

      [DISP=( OLD      ,KEEP      ,KEEP      ) ] [,LIMCT=n]
          [SHR      ,DELETE    ,DELETE    ]
          [NEW      ]
          [MOD      ]

      [,BUFNO=n] [DSORG= PS ] [,OPT=FORMAT]
                  [DA      ]

```

where the parameters have the following meanings:

**UNIT**

The keyword DISK in this parameter indicates that the unit is to be used to access a work file. If n is specified, the unit will be assigned the unit number n when it is created and may be accessed by the program this way. If NAME=ddname is specified, where "ddname" is a one to eight character name, the "ddname" will be assigned to the unit. Which of these is appropriate depends on what programming language is being used; both n and NAME may be given, in which case the unit may be referred to either way. If neither n nor NAME is given, the system will consider the file definition to be in error, since there would be no way for any program to access the unit.

**DSNAME**

The name of the work file (see "Work File Naming Conventions", below) is specified by this parameter. If the DSNAME parameter is omitted, the system assumes that a temporary work file is to be created. If the file name consists only of alphameric characters, ampersands, and periods, it need not be enclosed in apostrophes. However, if other characters are to appear in the file name, the name should be enclosed in apostrophes, with any apostrophes which form part of the name entered as double apostrophes.

**LRECL**

The LRECL parameter specifies the logical record length in bytes of the file. If the file was originally created with a record length different from the one specified by LRECL, the new specification will be used to access the file (i.e., the blocks will be divided up using the new record length). If the file is only read by the program, the permanent file description on the disk (DSCB) will still indicate the original record length, but if the program writes to the file, this permanent description will be altered to indicate the new length. If the record length is not specified when the file is created, a default of 128 will be used.

**BLKSIZE**

This parameter specifies the data length of the physical blocks in the file in bytes and must not exceed the track capacity of the disk (19069 bytes for unkeyed blocks on IBM 3350 disks). If the file has no keys (see the KEYLEN parameter, below), the physical block size is equal to the data length.

For an existing PS file, this parameter will override the value in the file's DSCB. If the file is overwritten by the program, the new data will be put into the file using the new data length.

For a DA file, this parameter is meaningful only if the /FILE statement specifies a file which is to be newly created. A /FILE statement for an already existing DA file which specifies a BLKSIZE value other than that with which the file was originally created will cause the file to be accessed with this new data length (possibly resulting in i/o error indications) but will not cause the file to be reformatted or its permanent description on disk changed. If BLKSIZE is not specified when the file is created, a default of 512 will be used.

In choosing a BLKSIZE value for a particular application, the user should bear in mind that as smaller and smaller blocks are put on a disk track, more and more space is taken up by the gaps between the blocks, so the effective track capacity decreases. Exactly how much the capacity decreases is a function of the particular type of disk drive that is being used and whether or not the blocks have keys. To facilitate the decision process, a program called BLKSIZE is supplied on VPS to make the necessary calculations for the current hardware for the case of blocks without keys. The program should be run at the terminal and is self-explanatory.

**KEYLEN**

The KEYLEN parameter specifies the length in bytes of the key portion for each block in the dataset and must not exceed 255. If KEYLEN is not specified, no keys will be created. The physical block size of the file is equal to the key length (KEYLEN) plus the data length (BLKSIZE).

For a PS file, KEYLEN will override the DSCB value in the same manner as BLKSIZE.

For a DA file, the KEYLEN parameter is meaningful only if the file is being created, as the file cannot subsequently be reformatted.

#### SPACE

This parameter indicates how much disk space is to be allocated to the file. The first field of this parameter specifies what units of space are to be used for allocation, and the second field indicates how many of these units are to be allocated (rounded up to an integral number of tracks). The numeric parameters must not exceed 32767. If the second parameter is given as a single number, an area containing that many allocation units will be used for the file, but no further expansion of the file will be allowed. For example, if a file is to contain 600 records, the parameter may be given as SPACE=(REC,600), indicating that the unit of allocation is the record (REC) and that 600 records are required. Similarly, SPACE=(BLK,100) defines a file containing 100 physical blocks, SPACE=(TRK,20) specifies that 20 disk tracks are needed, and SPACE=(CYL,4) means that 4 cylinders are to be allotted. If the second parameter is given as two numbers (enclosed in parentheses), the first of these numbers will determine the initial size (the primary allocation) of the file. The second number specifies the secondary allocation quantity.

If the file is of the PS type, and the secondary allocation quantity is non-zero, then each time a program attempts to place more data in the file than it will hold the file will be increased in size by this many allocation units. This may occur up to a maximum of 15 times, provided the necessary space exists on disk, and the account which owns the file does not exceed its work file track limit.

For a DA file, the secondary allocation quantity will be ignored, and the file cannot be expanded.

#### RECFM

The RECFM parameter specifies the record format to be used when the file is accessed on a record-by-record basis. The following record formats may be defined:

F	fixed length records, unblocked
FB	fixed length records, blocked
FBS	fixed length records, blocked, no short blocks except last
V	variable length records, unblocked
VB	variable length records, blocked
VS	spanned variable length records, unblocked
VBS	spanned variable length records, blocked
U	undefined record format

All the above record formats may be specified for a PS file, although some programming languages may not support them. In particular, spanned variable length records (RECFM=VS or VBS) are

currently supported only by FORTX, FORTG1, and some application programs written in FORTRAN, such as SPSS. Also, the QIO access method used in assembler language does not support spanned records. The remaining formats are supported by most languages and application programs on VPS.

As noted previously, a DA file may have only fixed length records, so the only record formats which should be specified for such a file are F, FB, and FBS.

#### DISP

The first field of the DISP parameter specifies whether the file already exists or is to be created, and whether the program is to be allowed to write to it. If OLD is given, the file must already exist, and the program will be allowed to both read from and write to it. If SHR is specified, an already existing file is still implied, but the program will only be allowed to read from the file. If a user whose account has only READ access to a file wishes to specify that file in a /FILE statement, SHR must be specified for the DISP parameter. If the first field is NEW, the system will attempt to create a new work file with the name supplied in the DSNNAME parameter (see "Creation and Deletion of Work Files", below). If MOD is specified, the system will use an existing file of the specified name unless none exists, in which case a new file will be created. In addition, specifying MOD for an existing PS file on which the first operation performed by the program is a write will cause the written data to be placed after the last block in the file (i.e. appended to the end).

The second and third fields of the DISP parameter indicate whether the file is to be deleted at normal and abnormal termination, respectively. If KEEP is specified in the appropriate position, the file will not be deleted; if DELETE is specified, it will be. KEEP is the default setting for both these fields unless a new file is being created, in which case the default for the normal termination field is KEEP, and the default for the abnormal termination field is DELETE. If the DISP field is omitted, the default is DISP=(OLD,KEEP,KEEP).

#### BUFNO

This field, meaningful only for QIO on DA files, indicates how many buffers the system is to provide for accessing the work file. If the field is omitted, the system will allocate a number of buffers based on the physical block size and organization type of the file. For a more detailed discussion of the meaning of the BUFNO parameter in the case of DA files, see the "VPS System Facilities for Assembler Programmers" manual.

#### DSORG

This parameter specifies the dataset organization. PS, the default if DSORG is not specified, indicates that the file is a physical sequential dataset, as supported by OS BSAM. DA means that the organization is to be equivalent to that used for OS BDAM.

#### LIMCT

The LIMCT parameter, valid only for keyed DA files, specifies the number of tracks or blocks to be searched when the extended search option is used (see the Guide to Programming Languages or the System Facilities for Assembler Programmers manual for information on extended search). Some higher level languages and assembler programs using BDIO can override or set this value during execution. If the work file referred to by the /FILE statement is not a keyed DA file, this parameter will be ignored.

#### OPT

The OPT parameter, when used with a DISK unit, can specify only the FORMAT option, which indicates that a new fixed DA file is to be formatted by the system when it is created. If the work file referred to by the /FILE statement is not a DA file or is not being newly created, this parameter will be ignored. This parameter is necessary only when creating DA files to be used with FORTG (not FORTRAN G1 or H Extended) or the linkage editor.

#### Work File Naming Conventions

The name of a work file must follow a specific naming convention, since this name indicates which account owns the file and what type of access other accounts are to have. The general form of a work file name is

URmWn.account.name

where "account" is the user's account number and "name" may be any collection of characters provided the entire name, periods included, does not exceed 44 characters. The initial field URmWn must begin with the letter U, indicating that this dataset is a user work file. The remainder of this field specifies whether READ and WRITE access is to be given to accounts other than the owner's.

If Rm is included, then only accounts for which the first m characters of the account number match those of the file owner's account are allowed to read the file. The value of m must be between 0 and 8. If R is given but m is not, a value of 0 is assumed, so that all accounts are allowed READ access to the file. If Rm is omitted entirely, only the account which owns the file is allowed to read it.

The Wn field is entirely analogous to the Rm field, except that it controls WRITE access.

To clarify the situation, we give here some file names belonging to the account XYZ100 each followed by a description of which accounts are allowed to read from or write to the file:

U.XYZ100.FILE1

Since the Rm and Wm fields are not given, only the account XYZ100 is allowed to read from or write to this file.

UR.XYZ100.MORE.STUFF

All accounts may read from this file, but only XYZ100 may write to it.

UR3W6.XYZ100.EVEN.MORE.STUFF

Only accounts starting with the three characters XYZ may read from this file, and only accounts starting with the six characters XYZ100 may write to it.

### Renaming Work Files

Work files may be renamed with the /DSRENAME command (page 118). Besides allowing the user-assigned name (the part of the name following the account number) to be changed, this command also allows the access code appearing at the beginning of the file name to be modified.

### Creation and Deletion of Work Files

A user creates a work file by running a job with an appropriate /FILE statement in it. If the first field of the DISP parameter is NEW, the system attempts to create a new dataset; if a dataset of the same name already exists, an error condition is raised, and the job will not be allowed to run. If MOD is specified for this field, and the named dataset does not exist, the dataset will be created; if the dataset does exist, the existing dataset will be used for the job.

For convenience, the system provides a dummy program called BR14, which can be specified on the /LOAD statement (see page 25). This program terminates as soon as it is entered.

For example, to create the 6 track sequential file UR.XYZ100.NEW.JOKE with record length 300 and block size 900, we may use the job

```
/FILE UNIT=(1,DISK),DSNAME=UR.XYZ100.NEW.JOKE,
/   LRECL=300,BLKSIZE=900,SPACE=(TRK,(6,2)),
/   DISP=(NEW,CATLG)
/LOAD BR14
```

A secondary allocation quantity of 2 tracks has also been specified.

On the other hand, if this file were to be used for random access, we would create it as a DA file, using the job:

```
/FILE UNIT=(1,DISK),DSNAME=UR.XYZ100.NEW.JOKE,
/   RECFM=V,BLKSIZE=808,SPACE=(TRK,6),
/   DISP=(NEW,CATLG),DSORG=DA,KEYLEN=4
```

where we have specified no secondary allocation quantity, since it is meaningless for a DA file.

A user may delete a file either by running a job containing a /FILE statement indicating deletion is to occur at end-of-job or by using the /DSPURGE command (page 117).

On a /FILE statement, the second and third fields of the DISP parameter determine whether the file is deleted. If the job terminates normally, and the second DISP field is DELETE, the file will be deleted, as it would be if the job were to terminate abnormally with DELETE specified for the third DISP field.

For instance, the file created in the above examples could be deleted by running the job

```
/FILE UNIT=(1,DISK),DSNAME=UR.XYZ100.NEW.JOKE,  
/   DISP=(OLD,DELETE)  
/LOAD BR14
```

### Temporary Work Files

Work files are often used by programs as temporary work areas rather than for the permanent storage of data. Since a work file to be used this way need not exist except while the program is actually running, it is usually specified on the /FILE statement as a temporary file. A DISK file is considered temporary if the DSNAME parameter is omitted or if the dataset name is given as a name not exceeding 8 characters beginning with an ampersand (e.g. DSNAME=&&TEMP). Note that an ampersand must be represented by two consecutive ampersands so that the name does not appear to VPS to be a symbolic parameter (see page 29).

The system maintains a pool of temporary work file tracks; if the space required for the file does not exceed that available in the pool, the file will be allocated from the pool, and the cost in CPU time and I/O operations associated with manipulating the VTOC will be avoided. If the file does not fit, a regular work file will be created automatically. In any case, the temporary file will be deleted when the job terminates. The disk tracks allocated to temporary files are not included in an account's usage total or track charge.

### Editing Work Files

The editor which is invoked for library file modification with the /EDIT command (page 119) may also be used to edit a work file, provided the logical record length of the file does not exceed 256. For details, consult the "VPS Utilities" manual.



## Chapter 5: Magnetic Tape

VPS currently provides the user with a subset of OS tape support. Non-labeled tapes are fully supported, but standard labeled tapes cannot yet be created by the system. It is, however, possible to read standard labeled tapes through the "bypass label processing" option. Tapes to be read or written may be specified as having fixed length or variable length records, with or without blocking, or as having an undefined record format. The terms have the same meanings for VPS as they have for OS; for a detailed discussion of these record formats, see the "VPS System Facilities for Assembler Programmers" manual.

### Defining Tape Input/Output Units

A tape file is accessed through a unit of the type TAPE. The general form of the /FILE statement for a TAPE unit is:

```
/FILE {UNIT=( [n,] TAPE [,NAME=ddname] )}
      {,VOL=SER=volser}    {,LRECL=m}    {,BLKSIZE=n}
      [ ,RECFM=  F
                FB
                V
                VB
                VS
                VBS
                U ] { ,LABEL=( [k] ,NL
                               ,BLP ) } [ ,DEN=  2
                                           3
                                           4
                                           800
                                           1600
                                           6250 ]
      [,BUFNO=n]
```

where the parameters have the following meanings:

#### UNIT

The keyword TAPE in this parameter indicates that this unit is to be used to access a tape file. If n is specified, the unit will be assigned the unit number n when it is created and may be accessed by the program this way. If NAME=ddname is specified, where "ddname" is a one to eight character name, the "ddname" will be assigned to the unit. Which of these is appropriate depends on what programming language is being used; both n and NAME may be given, in which case the unit may be referred to either way. If neither n nor NAME is given, the system will consider the file definition to be in error, since there would be no way for any program to access the unit.

VOL=SER

The VOL=SER parameter is required and indicates the name of the tape volume which is to be mounted by the operator. This name must also appear in written form on the tape reel itself, preferably on a paper label. The name must consist of one to six alphanumeric characters and is to be substituted directly for "volser" (e.g., VOL=SER=TAPE01).

LRECL

This parameter specifies the logical record length of the file in bytes and must not exceed 32767. If the file is defined to have fixed length records (RECFM=F or RECFM=FB), the record length must be an integer submultiple of the block size. If the file has variable format records (RECFM=V or RECFM=VB), the record length must be at least as great as the length of the longest record in the file and must be at least 4 less than the block size. If the record format is specified as undefined (RECFM=U), this parameter has no meaning.

BLKSIZE

The BLKSIZE parameter specifies the physical block size of the tape file in bytes and must not be less than the length of the largest block actually in the file. The largest value which may be specified is 32767. If the file has fixed length records, the lengths of all blocks in the file must be integer multiples of the record length. If variable length records are used, the block size must be at least 4 greater than the record length, and each block in the file must begin with a valid block descriptor word (see the "VPS System Facilities for Assembler Programmers" manual).

RECFM

This parameter describes the record format of the tape file. If F is specified, the file has unblocked fixed length records, and the block size and record length must be equal. FB implies fixed length blocked records, and all blocks in the file must be integer multiples of the record length. V and VB indicate unblocked and blocked variable length records, respectively; all blocks and records in the file must conform to the standard OS specifications for variable length format. If the record format is given as U, the file is assumed to contain blocks of arbitrary length not exceeding the stated physical block size, and each record is taken as an entire block.

VS and VBS, indicating unblocked and block spanned variable length records, may also be specified, but many languages and programs do not currently support these formats. They are, however, supported by FORTX, FORTG1, and some application programs written in FORTRAN, such as SPSS.

**LABEL**

The first field of this parameter indicates which file on the tape is to be accessed by this unit. The first file is file 1, the second file 2, and so forth. The second parameter specifies what form of label processing is to be done. If NL is given, the tape must be non-labeled. If a standard labeled tape is mounted when a non-labeled tape is called for, it will be rejected by the system. Label checking will be bypassed if BLP ("Bypass Label Processing") is given instead, and standard labeled tapes may be accessed to a limited degree through the use of this parameter, as described below under "Reading Standard Labeled Tapes".

**DEN**

The DEN parameter is used to specify the recording density for a tape file which is being written. This parameter will be ignored unless BLP is specified in the LABEL parameter; if NL is specified, the tape will be recorded at the same density as the records already present on the tape. The density may be specified in either OS form (2, 3, 4) or human form (800, 1600, 6250). A value of 2 or 800 means 800 bits per inch (BPI), 3 or 1600 means 1600 BPI, and 4 or 6250 means 6250 BPI.

**BUFNO**

This optional parameter is used to specify how many buffers the system should allocate for reading or writing the tape file. If this parameter is omitted, the system will allocate two buffers for tapes with physical block sizes up to 8k (8 X 1024 bytes) and one buffer for tapes with block sizes greater than this value. This parameter is rarely needed, but might be used, for example, to force the system to allocate only one buffer even though the block size of the file would indicate that two should be used, in order to decrease the amount of storage needed for the job, or to force the system to use two buffers when one would normally be used, in order to increase the overlap of execution and i/o channel activity for jobs in which these activities are well matched. A number greater than two would be of value only for a job which exhibits a ratio of CPU activity to tape i/o activity that varies appreciably in time. In any case, the system will not allocate more than five buffers, and any request for more than this number will be treated as a request for five buffers.

Accessing More Than One File on a Tape

It is possible, in a single program, to read or write more than one file on a tape. However, since only one file on a tape may be accessed at any given moment, due to the physical limitations of the tape drive, the program must be written in such a way that each file accessed on the tape is "closed" before the next file is accessed. Normally, a separate /FILE statement, defining a separate TAPE i/o unit, must be supplied for each file to be accessed on the tape. All these /FILE statements must specify the same volume name (VOL=SER parameter). (There is a system facility available which allows the user program to modify the file number (first field of the LABEL parameter), block size (BLKSIZE), record length (LRECL), and record format (RECFM) of a TAPE unit if that unit and all other units

accessing the same tape volume are closed at the time. See the "VPS System Facilities for Assembler Programmers" manual for details.) A new file may be accessed at the following points in the execution of a program:

No file on the tape has been accessed.

The file being read has reached end-of-file.

The file previously being read or written out has been closed, rewound, or end-file has been requested, depending on the programming language. Some examples of operations in various languages which have this effect are:

Operations which rewind the tape file (reposition the file to the beginning):

Assembler (VSASM): The QREW macro.

PL/I (PLIOPT): CLOSE (file) ENVIRONMENT(REREAD);

COBOL (COBOLVS): CLOSE file

FORTRAN (FORTG1 or FORTX): REWIND n

Operations which cause end-file to be requested (reposition the file to the end):

Assembler (VSASM): The QEOF macro.

PL/I (PLIOPT): CLOSE (file) ENVIRONMENT(LEAVE);

COBOL (COBOLVS): CLOSE file WITH NO REWIND

FORTRAN (FORTG1 or FORTX): ENDFILE n

### Reading Standard Labeled Tapes

Standard labeled tapes are not yet specifically supported by VPS, but such tapes may be read if BLP is specified in the LABEL parameter of the /FILE statement. However, it is necessary to recompute the file number (first field of the LABEL parameter), since one must account for the labels on the tape. Each file on a standard labeled tape is preceded by a header label and followed by a trailer label; each of these labels is itself a tape file. Therefore, the first data file is actually file 2, the second file 5, and so on. In general, if M is the file number in labeled tape terminology, the file number which must be used is

$$N = 3M - 1$$

Writing on Blank Tapes

When a new tape is purchased, it is completely blank and cannot be processed if NL is specified in the LABEL parameter, because the tape will be run straight off the reel when the system attempts to verify that a labeled tape has not been mounted. Therefore, when a new tape is first written upon, the user must specify BLP in the LABEL parameter. This approach must also be used if the user wishes to overwrite a tape at a new density.

File Protection

In order to protect a tape from being written on, the user must remove the write ring from the tape reel. No /FILE statement parameter is available which affords this protection.

Using Tapes from Terminals

VPS allows the use of tapes by programs running at terminals as well as by programs running on the batch: There is no difference between the two environments related to /FILE statements or programming techniques; if a terminal job containing TAPE unit definitions is run, the operator will receive requests for the tape volumes indicated. However, the user should communicate with the operator, either by telephone or through the /MSG command (page 128) to insure that the operator has the proper tape in hand, and that the necessary tape drives are available.

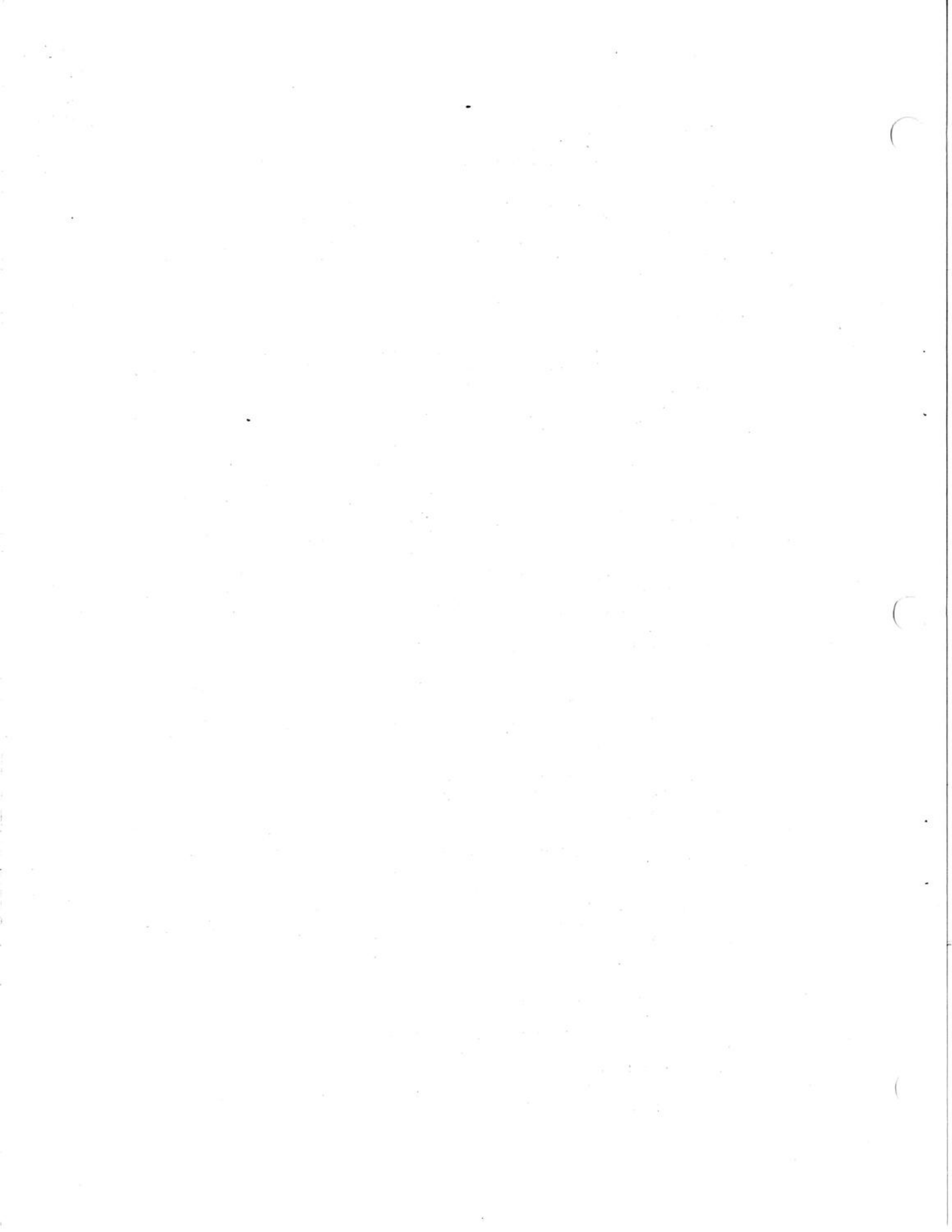
Examples of /FILE Statements for TAPE Units

The following /FILE statement defines a TAPE unit which will access the fourth file on a tape named JNK841 which has blocked variable format records 500 bytes or less in length and a maximum physical block length of 5000 bytes.

```
/FILE UNIT=(25,TAPE),VOL=SER=JNK841,
/   LRECL=500,BLKSIZE=5000,RECFM=VB,
/   LABEL=(4,NL)
```

The following three /FILE statements define three units which will access separate files all on the same tape volume, named TAPE12, with various block sizes, record lengths, and record formats. Note that no unit numbers are assigned by the /FILE statements, so the units will have to be accessed by the ddnames TFILE1, TFILE2, and TFILE3 (by, for example, a PL/I program).

```
/FILE UNIT=(TAPE,NAME=TFILE1),VOL=SER=TAPE12,
/   LRECL=80,BLKSIZE=800,RECFM=FB,
/   LABEL=(3,NL)
/FILE UNIT=(TAPE,NAME=TFILE2),VOL=SER=TAPE12,
/   LRECL=200,BLKSIZE=200,RECFM=F,
/   LABEL=(6,NL)
/FILE UNIT=(TAPE,NAME=TFILE3),VOL=SER=TAPE12,
/   LRECL=150,BLKSIZE=154,RECFM=V,
/   LABEL=(1,NL)
```



## Chapter 6: Other Input/Output Units

This chapter describes the unit types which have not been described in the preceding chapters. These types are grouped into a single chapter simply because they do not require extensive explanations.

### TERMIN: Reading the Terminal Keyboard

If a program is running at a terminal, it may receive input directly from the user by issuing a read request to a unit of the type TERMIN. Records up to 200 characters in length may be entered this way, and the input will be edited by the system in accordance with the user's input editing characters and settings (see page 10) before it is passed to the program. (The input editing and even the translation to EBCDIC from terminal code can be suppressed by the program - see the "VPS System Facilities for Assembler Programmers" manual for details. The input spooling facility is also described in that manual.)

As described in Chapter 1 (page 5), all records entered through the terminal at "\*go" level and in response to program-generated reads are put into a single list of records which is accessed in FIFO (first in first out) order by all requests for terminal input generated by the system at "\*go" level or by programs. This list may also be cleared, typed out, or added to at either end with the /CLIST command (page 103), which is valid only in response to an attention level read.

A default TERMIN unit, with unit number 9, is provided for jobs run at the terminal (see page 23).

The form of the /FILE statement for a TERMIN unit is

```
/FILE {UNIT=( [n,] TERMIN [,NAME=ddname] )}
```

where the parameters are:

#### UNIT

The keyword TERMIN in this parameter indicates that the unit is to be a terminal input unit. If n is specified, the unit will be assigned the unit number n when it is created and may be accessed by the program this way. If NAME=ddname is specified, where "ddname" is a one to eight character name, the "ddname" will be assigned to the unit. Which of these is appropriate depends on what programming language is being used; both n and NAME may be given, in which case the input unit may be referred to either way. If neither n nor NAME is given, the system will consider the file definition to be in error, since there would be no way for any program to access the unit.

TERMOUT: Receiving Printout

Program output which is to appear on the terminal (the line printer for batch jobs) is sent to one or more units of the type TERMOUT. All TERMOUT units defined for a given job collectively feed a single output stream, so that records written to several different TERMOUT units will appear interspersed in the same order as they are produced by the program. The system provides a default TERMOUT unit as unit 6, ddname SYSPRINT (see page 23), unless the user overrides it. User-supplied /FILE statements may be employed to define additional TERMOUT units and to specify whether these units are to interpret the first character of each record as a carriage control character.

The following carriage controls are supported on both terminal and batch (carriage control characters are not printed - the data to be printed starts with the character following the carriage control character):

<u>character</u>	<u>effect</u>
space	advance one line, print data
0 (zero)	advance two lines, print data
- (minus)	advance three lines, print data
1 (one)	eject page on batch, print data; same as "-" on terminal
#	suppress carriage return on terminal, print data; equivalent to space on batch

In addition, the following carriage controls are supported on the batch only:

<u>character</u>	<u>effect</u>
+	overprint previous line with this data
C	skip to bottom of page, print data
X'01'	print data, no carriage advance
X'09'	print data, advance one line
X'11'	print data, advance two lines
X'19'	print data, advance three lines
X'89'	print data, eject page
X'E1'	print data, skip to bottom of page
X'03'	ignore data
X'0B'	ignore data, advance one line
X'13'	ignore data, advance two lines
X'1B'	ignore data, advance three lines
X'8B'	ignore data, eject page
X'E3'	ignore data, skip to bottom of page

(X'hh' is the Assembler language representation for a byte with hexadecimal value hh.) The characters shown as hexadecimal values are machine operation codes, whereas the other carriage control characters supported are standard ANSI control characters. The set of carriage control characters supported by VPS is thus the union of the most commonly used subsets of ANSI and machine code control characters. If the first character of a line sent to a TERMOUT unit which is expecting carriage control characters is not a valid carriage control, the entire line, including the first character, will be printed.



If carriage control is in effect for a TERMOUT unit being used at a terminal, the following two skip control characters will also be recognized:

<u>character</u>	<u>effect</u>
!	print line despite /SKIP setting
%	print line and clear /SKIP setting

Note that these control characters will force printout if print suppression is being controlled by /SKIP (page 154), but not if it is being controlled by /CO (page 105) or if the job has been cancelled (/CANCEL, page 100). The skip control character must precede the carriage control character in the line, so that the carriage control character becomes the second character. If a line with one of these skip control characters appearing in the first position is sent to a TERMOUT unit by a batch job, the character will be ignored and the line considered to start with the second character for the purpose of carriage control interpretation.

The general form of a /FILE statement for a TERMOUT unit is:

```
/FILE {UNIT=( [n,] TERMOUT [,NAME=ddname] )}
```

```
    [ ,RECFM=  $\frac{FA}{F}$  ]
```

#### UNIT

The keyword TERMOUT in this parameter indicates that the unit is to be a printout unit. If n is specified, the unit will be assigned the unit number n when it is created and may be accessed by the program this way. If NAME=ddname is specified, where "ddname" is a one to eight character name, the "ddname" will be assigned to the unit. Which of these is appropriate depends on what programming language is being used; both n and NAME may be given, in which case the input unit may be referred to either way. If neither n nor NAME is given, the system will consider the file definition to be in error, since there would be no way for any program to access the unit.

#### RECFM

The RECFM parameter is used to indicate whether the system is to interpret the first character of each record as a carriage control character. If FA is specified, the records will be so interpreted. If F is specified, the entire line will be printed, the first character remaining uninspected. The default is FA.

### Virtual Unit Record Units

The user may define virtual card readers, printers, and punches for the duration of a job via appropriate /FILE statements (this requires a privilege at the terminal).

#### READER: Virtual Card Reader

A READER unit is a virtual card reader. When a given READER unit is first accessed, it attaches itself to the highest priority otherwise unattached READER spool file of the same class as itself belonging to the account (on terminal) or batch processor running the job and returns the first virtual card image to the program. Each subsequent read request fetches another card image until no more spool files of that description exist. At that point, an end-of-file condition is raised. A READER spool file is created by reading in a deck of cards or by "spooling" a virtual card punch or printer to the appropriate account. This is a rarely needed facility, and the user should consult with User Services before messing around with it.

The number of READER units the user may define at one time for a terminal job is defined in the user's account (zero for most accounts). For a batch job, up to eight READER units may be created.

The general form of the /FILE statement for a READER unit is:

```
/FILE {UNIT=([n,] READER [,NAME=ddname])}  [,CLASS=c]
```

#### UNIT

The keyword READER in this parameter indicates that the unit is to be a virtual card reader. If n is specified, the unit will be assigned the unit number n when it is created and may be accessed by the program this way. If NAME=ddname is specified, where "ddname" is a one to eight character name, the "ddname" will be assigned to the unit. Which of these is appropriate depends on what programming language is being used; both n and NAME may be given, in which case the input unit may be referred to either way. If neither n nor NAME is given, the system will consider the file definition to be in error, since there would be no way for any program to access the unit.

#### CLASS

The CLASS parameter determines the spool file class that will be accessed by this unit and is specified as a single letter. The default is CLASS=B.

PRINTER: Virtual Printer

Each PRINTER unit is a separate virtual printer. The records sent to a given PRINTER unit form a separate output stream and will not appear interspersed with records sent to TERMOUT units or other PRINTER units. The stuff sent to a PRINTER unit always appears on the line printer, whether the job is run at the terminal or on the batch.

The number of PRINTER units the user may define at one time for a terminal job is four. For a batch job, up to seven PRINTER units may be defined.

The /FILE statement for a PRINTER unit has the form:

```

/FILE {UNIT=( [n,] PRINTER [,NAME=ddname] )}    [,CLASS=c]
      [ ,RECFM=FA ] [ ,COPIES=n ] [ ,OUTLIM=n ] [ ,OPT=NOSKIP ]
              F      1
      [ ,DISP=( ,KEEP ,KEEP ) ]
                ,DELETE ,DELETE

```

## UNIT

The keyword PRINTER in this parameter indicates that the unit is to be a virtual printer. If n is specified, the unit will be assigned the unit number n when it is created and may be accessed by the program this way. If NAME=ddname is specified, where "ddname" is a one to eight character name, the "ddname" will be assigned to the unit. Which of these is appropriate depends on what programming language is being used; both n and NAME may be given, in which case the input unit may be referred to either way. If neither n nor NAME is given, the system will consider the file definition to be in error, since there would be no way for any program to access the unit.

## CLASS

The CLASS parameter is used to specify the class of the spool file to be produced by this unit. The default is CLASS=A, which is the normal class for files to be printed on standard forms. This parameter is useful when the user wishes the file to be held for special forms or for printing at another time. Consultation with the operator is necessary to insure that a mutually agreeable output class is chosen.

## RECFM

The RECFM parameter for a PRINTER unit has the same effect as the RECFM parameter for a TERMOUT unit (see page 64, above). The same carriage control characters are supported for PRINTER units as are supported for batch TERMOUT units.

OUTLIM

This parameter sets the output limit in pages for the PRINTER unit being defined. Bear in mind that the total output of all PRINTER and TERMOUT units for a batch job may not exceed the limit set by the /ID statement (see page 87). The default is 20 pages and the maximum is 32767.

COPIES

The COPIES parameter controls the number of copies of this file to be printed on the high speed line printer. The default is COPIES=1 and the maximum is 255.

OPT

NOSKIP prevents the system from automatically skipping to the top of the next page each time the bottom of a page is reached. Printing will be allowed to continue undisturbed over the perforations.

DISP

The second and third fields of the DISP parameter indicate whether the output is to be printed for normal and abnormal termination, respectively. If DELETE is coded for the applicable field, the output will be flushed from the system at end-of-job and the user's account will not be charged for the printing.

PUNCH: Virtual Card Punch

Each PUNCH unit is a separate virtual card punch. The records sent to each particular PUNCH unit will produce a separate deck of punched cards. (Note that to punch cards at the Computing Center a real dollar balance must exist for the account punching the cards.)

The number of PUNCH units the user may define at one time for a terminal job is four. For a batch job, up to eight PUNCH units may be defined, including the default PUNCH unit normally supplied as unit 7, ddname SYSPUNCH (see page 23).

The form of the /FILE statement for a PUNCH unit follows:

```

/FILE {UNIT=([n,] PUNCH [,NAME=ddname])}  [,CLASS=c]
      [,OUTLIM=n]      [,COPIES=n]
                        [ 1 ]
      [,DISP=(  ,KEEP  ,KEEP  )]
                  [,DELETE ,DELETE ]

```

## UNIT

The keyword PUNCH in this parameter indicates that the unit is to be a virtual card punch. If n is specified, the unit will be assigned the unit number n when it is created and may be accessed by the program this way. If NAME=ddname is specified, where "ddname" is a one to eight character name, the "ddname" will be assigned to the unit. Which of these is appropriate depends on what programming language is being used; both n and NAME may be given, in which case the input unit may be referred to either way. If neither n nor NAME is given, the system will consider the file definition to be in error, since there would be no way for any program to access the unit.

## CLASS

The CLASS parameter determines the output class for this unit. The default is CLASS=A, which will direct the spool file to the card punch.

## OUTLIM

This parameter sets the output limit in cards for the PUNCH unit being created. Remember that the total output of all PUNCH units for a batch job may not exceed the limit set by the /ID statement (see page 87). The default is 50 cards and the maximum value is 32767.

## COPIES

The COPIES parameter controls the number of copies of this file to be punched on the card punch. The default is COPIES=1 and the maximum is 255.

DISP

The second and third fields of the DISP parameter indicate whether the output is to be punched for normal and abnormal termination, respectively. If DELETE is coded for the applicable field, the output will be flushed from the system at end-of-job and the user's account will not be charged for the punched cards.

DUMMY: When Less is More

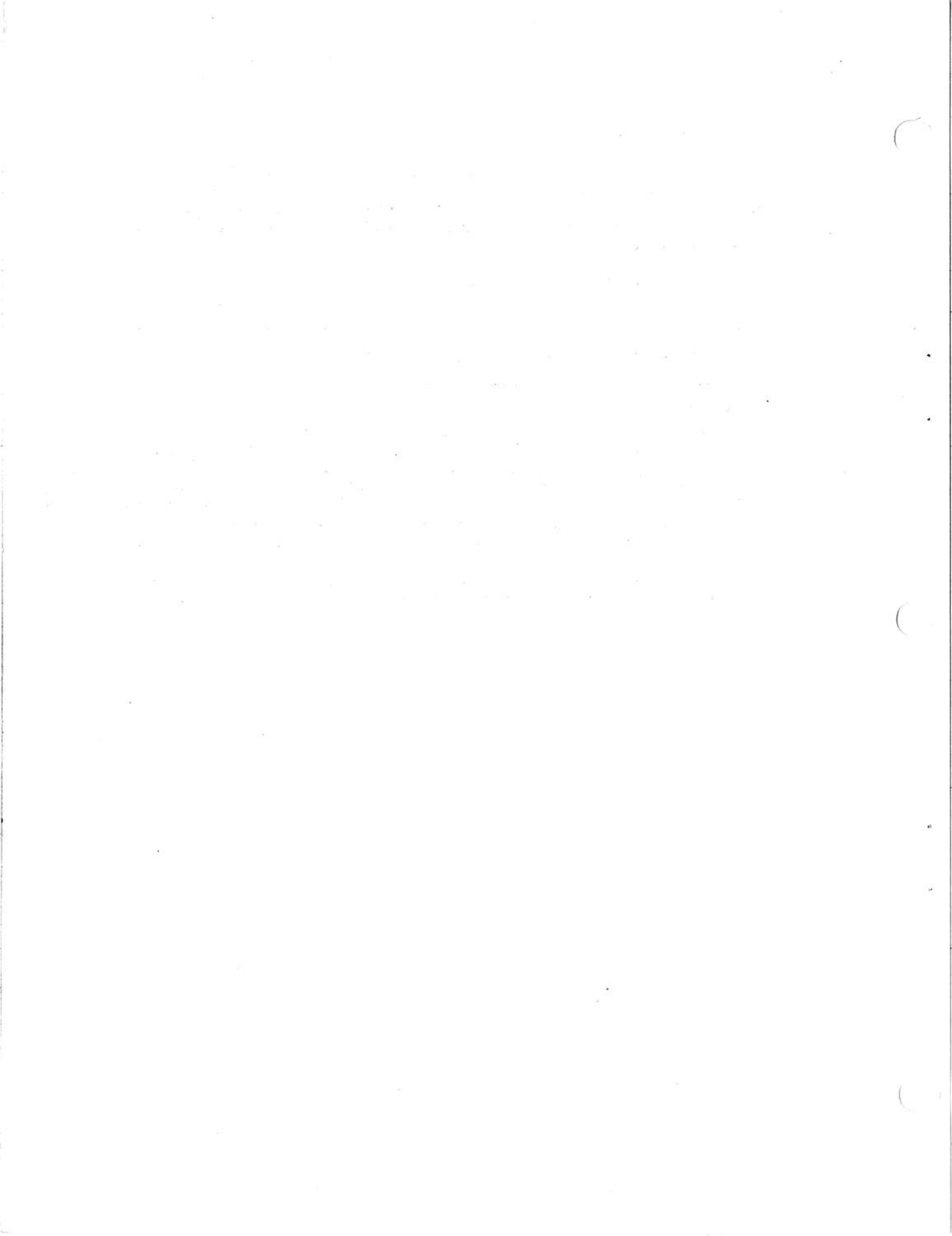
A DUMMY unit may be used when a program requires that a particular unit exist, but the user does not wish to supply an actual unit. A DUMMY unit will ignore all write, rewind, backspace, and end-file requests and will raise the end-of-file condition for any read requests.

The /FILE statement for a DUMMY unit has the form:

```
| /FILE {UNIT=([n,] DUMMY [,NAME=ddname])}
```

## UNIT

The keyword DUMMY in this parameter indicates that the unit is to be a dummy unit. If n is specified, the unit will be assigned the unit number n when it is created and may be accessed by the program this way. If NAME=ddname is specified, where "ddname" is a one to eight character name, the "ddname" will be assigned to the unit. Which of these is appropriate depends on what programming language is being used; both n and NAME may be given, in which case the input unit may be referred to either way. If neither n nor NAME is given, the system will consider the file definition to be in error, since there would be no way for any program to access the unit.





## Chapter 7: Accounting and the User Profile

One of the tasks performed by the VPS system is the collection of information for accounting and billing purposes. Session or job related accounting begins when a user signs on to the system through a terminal or runs a batch job and is accumulated during the terminal session or job. A charge is produced at sign-off or job termination. Resource related accounting, such as for direct access space, is independent of terminal sessions and batch jobs and is accumulated at small intervals during system operation. For either of these types of charges, the system automatically reduces the associated account balance by the charge.

All of these charges are collected on a storage device and at the end of a billing cycle (usually the end of the month) statements are produced giving a summary of the charges and the new balance for each account. Users are notified of their balance at sign-on, and at sign-off the system prints session charge and new balance. During the session, charging information may be listed with the /ACCT command (see page 95) or the /PROFILE command (discussed later in this chapter).

### Session or Job Related Accounting

The information accounted and billed for during terminal sessions or batch jobs includes:

Central Processor Time (CPU Time) - the time spent by the CPU executing instructions for the user.

Terminal Connect Time (for terminal sessions only) - the time the user is connected, or signed-on, to VPS.

I/O Accesses - the accumulated reads and writes to disk and tape files.

Paper and Cards - the total number of pages printed on the high-speed line printer and cards punched. (Note that to print pages or punch cards above the limit specified for an account, a real dollar balance must exist for that account. The punch limit for most accounts is zero.)

Session or job related charging takes place only at sign-off or job termination.

### Resource Related Accounting

The information accounted for independent of terminal sessions and batch jobs includes:

Library file space sets  
Work file tracks

Resource related accounting is done dynamically by the system with a resolution of approximately one minute - whether the user is active or not. Therefore, an account balance can change during the course of a

terminal session or batch job due to resource related charging and may decrease from day to day even though the account is not being used.

### The User Profile

When an account is created on VPS a "user profile" for that account is also created. Profile information consists of the account password and default values for such things as tab settings, access attributes, the editor flip character, etc. These default values free the user from the necessity of entering this information at each sign-on since VPS will automatically set these parameters from the user profile.

From time to time a user may find it advantageous to change or set parameters in the profile. This may be done with the /PROFILE command. If an account is to be used by several individuals, the profile may be made unchangeable when the account is created or at any time thereafter.

### Changing the Password

The most common change to the user profile is changing the password. The password may be changed by using either the /PROFILE command (discussed later in this chapter) or by executing the password changing program - PASSWD. PASSWD is a conversational program which prompts the user for all necessary information; an example of its use is shown below.

```
*GO
passwd
```

```
THIS PROGRAM ALLOWS YOU TO CHANGE THE PASSWORD
OF THE ACCOUNT YOU ARE CURRENTLY SIGNED ON WITH.
```

```
THE ACCOUNT YOU ARE SIGNED ON WITH IS MA140HBK.
```

```
ENTER CURRENT PASSWORD
```

```
*****
```

```
ENTER DESIRED NEW PASSWORD (OR /CAN)
```

```
*****
```

```
PASSWORD SUCCESSFULLY CHANGED.
```

```
*GO
```

```
.
.
.
```

### The /PROFILE Command

The user profile values may be inspected and/or changed using the /PROFILE command (see page 135). The /PROFILE command invokes an interactive program which will prompt the user for profile commands. Whenever the program is ready to process a command, the profile program's prompt character ("-") will be typed. At that point, the

user may enter one of the profile commands described below.

The GET Command

The GET command will list accounting information and any parameters previously set by the user. Options having system defaults (see page 10) will not be listed. The standard form of the GET command is written below.

```

| GET  [ (null)
|       FULL
|       HEX ]
|

```

parameter

specifies the type of listing to be produced. If no parameter is specified, the listing will include only those profile parameters which have been previously set or changed by the user. If FULL is specified, the listing will also include account balance information. If HEX is specified, the profile record will be dumped in hexadecimal.

The CHANGE Command

The CHANGE command will change one or more of the profile parameters. Note that CHANGE may be abbreviated C. The standard form of the CHANGE command is written below.

```

| CHANGE  parm1=value1,parm2=value2,...
| C
|

```

parameter

specifies the profile parameter and new default value to be assigned to it ("value1", "value2" above). If a list of profile parameters are to be changed, they must be separated by commas. The supported profile parameters are described below:

- PASSWORD= (PW=) - specifies a new password.
- AUTOPROG= (AP=) - specifies the name of the program to be automatically executed each time the user signs-on.
- OPTION= (OPT=) - specifies a series of parameters which must be enclosed in parentheses (if more than one parameter is specified) and separated by commas. Options available are:
  - ETERNAL (NC) - autoprogram cannot be canceled.
  - NOETERNAL (NONC) - resets ETERNAL.
  - FIXPW (FPW) - password cannot be changed.
  - FIXPROF (FPR) - user profile cannot be changed.
  - BEGINNER (BEG) - beginner mode will be entered at sign-on.
  - NOBEGINNER (NOBEG) - beginner mode will not be

- entered at sign-on.
- DEBUG (DEB) - equivalent to entering the /DEBUG ON command at sign-on (see page 112).
- NODEBUG (NODEB) - resets DEBUG.
- LOGMSG (LOG) - equivalent to entering the /LOG ON command at sign-on (see page 127).
- NOLOGMSG (NOLOG) - resets LOGMSG.
- CTLCMDON (CMD) - equivalent to entering the /CTL CMD ON command at sign-on (see page 106).
- NOCTLCMDON (NOCMD) - resets CTLCMDON.
- TAB= - specifies the logical tab character.
- BS= - specifies the logical backspace character.
- LDEL= - specifies the logical line delete character.
- ESCAPE= - specifies the logical escape character.
- DELIM= - specifies the logical line end character.
- HEX= - specifies the hex substitution character followed by a comma and the 2-digit hexadecimal code it represents, all enclosed in parentheses.
- TABIN= - specifies input tab settings in the non-column form used for the /TABIN command (see page 159).
- TABOUT= - specifies output tab settings in the non-column form used for the /TABOUT command (see page 161).
- TABS= - specifies input and output tab settings in the non-column form used for the /TABS command (see page 163).
- FLIP= - specifies the editor flip character.
- EDFLAG= - either BRIEF or VERIFY can be specified to set the editor default mode of operation.
- TERMCPU= (TCPU=) - specifies a default value in seconds of CPU time after which a program executing at the terminal will be canceled. (This may be overridden using the TIME= parameter on the /LOAD statement.)
- BATCHPW= (BPW=) - specifies a batch password.
- ACCESS= - specifies access attributes for any future library save operations (see page 149).
- PERMIT= - specifies special access attributes for any future library save operations (see page 149).

Notes:

Parameters may be reset by specifying the parameter with no value. Changes of the password or the access or permit parameters will take effect immediately. All other changes will not take effect until the next sign-on.

The END Command

The END command terminates the profile program. QUIT may be specified in place of END - both have the same effect. The standard form of the END command is written below.

```

| END
| QUIT
|

```

/PROFILE Example

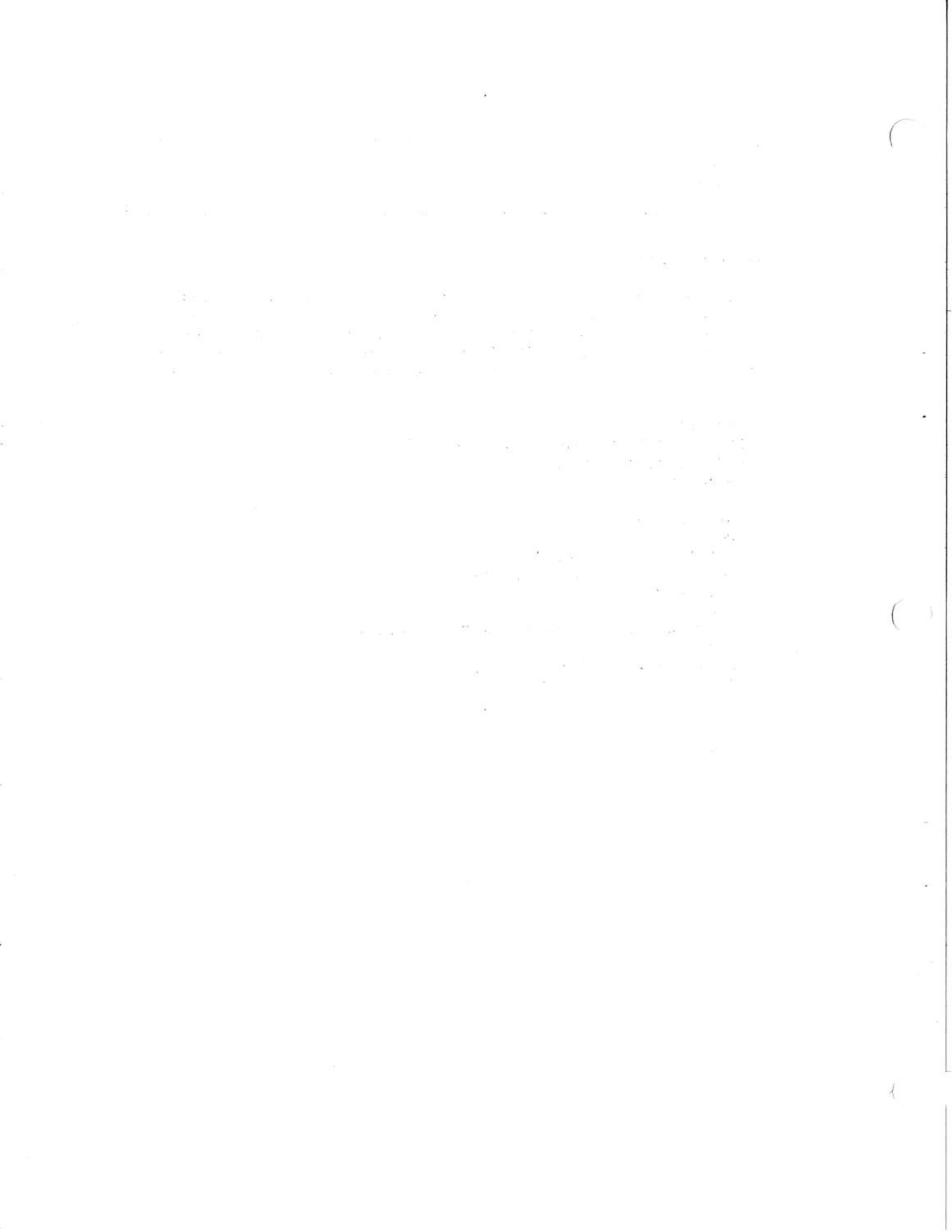
In the following example the user invokes the profile program using the /PROFILE command. The GET command is used to inspect the previously changed parameters. The CHANGE command is then used to set the line end character (DELIM=), set the hexadecimal substitution character (HEX=), and reset the logical back space character (BS=).

```

*GO
/profile
PROFILE CHANGING PROGRAM FOR AM470V07
ENTER CURRENT PASSWORD.
*****

ENTER COMMANDS.
-get
ACCTNO=AM470V07,TABIN=ASM,BS=<
-change delim=%,hex=($,02),bs=
CHANGED OK.
-get
ACCTNO=AM470V07,TABIN=ASM,DELIM=%,HEX=($,02)
-end
ACCOUNT UPDATE PROGRAM FINISHED.
*GO
.
.
.

```



## Chapter 8: Interactive Debugging

VPS provides the user with several aids that facilitate the interactive debugging of programs - particularly programs written in assembler language. Mechanisms are provided which allow the user to trace program execution, to detect certain events, and to interactively examine and modify registers and storage. Two terminal commands are implemented which give access to these facilities: /DEBUG (page 112) and /PER (page 131). Associated with these commands is a "debug mode" which may be entered during the execution of a program. In this mode, program execution is suspended, and a number of commands are made available which allow the user to inspect and modify the program and to resume execution. No special coding conventions or subroutine calls are required in programs to be debugged using these facilities.

### Debugging Facility Terminal Commands

The terminal command /DEBUG is used to indicate when debug mode is to be entered and to control tracing of supervisor call instructions. /DEBUG ON specifies that debug mode is to be entered when abnormal conditions arise during program execution, such as program interrupts or fatal errors. /DEBUG EOJ specifies that debug mode is to be entered even if the program terminates normally, allowing a post mortem examination to be made. If the /DEBUG command is issued without operands (either at "\*go" or attention level), or if the keyword ENTER is specified, debug mode will be entered immediately.

The /DEBUG command is also used to specify that supervisor call (SVC) instructions are to be traced during program execution. This tracing may be done either one SVC at a time (/DEBUG SVC), with debug mode being entered immediately after each SVC is issued by the program, or in a continuously running mode (/DEBUG TSVC), in which the trace message is typed each time an SVC is executed by the program, but the program is then allowed to continue running. In the latter case, the user may enter debug mode at any time by interrupting one of the trace messages using the attention or break key and issuing /DEBUG. When debug mode is entered in this way, the instruction following the SVC instruction will not have been executed and the supervisor service associated with the SVC will not have been performed. The user then has the option of resuming execution and having the service performed normally or of resuming at the next instruction without having the SVC serviced by the system at all.

With the exception of the /DEBUG ON setting, all /DEBUG options are turned off when the terminal is returned to "\*go" level after the termination of the program. The /DEBUG ON setting remains in effect for the duration of the terminal session unless it is explicitly turned off with /DEBUG OFF. The user profile can be modified (see page 73) so that /DEBUG ON is the default setting. (See page 112 for a complete description of the /DEBUG command.)

The other terminal command that gives access to the interactive debugging mode is /PER, which makes the "Program Event Recording" facility of System/370 available for program execution tracing. This command allows access to all aspects of this hardware facility and may be issued at "\*go" or attention level. When issued at "\*go" level, any setting applies to the next program executed, while settings issued during an attention read apply to the currently executing program. All settings are turned off when the terminal is returned to "\*go" level after the program has terminated. The /PER command allows the user to monitor the executing program for instruction execution within a specified address range (the INST keyword), modification of storage within a specified address range (STORE), successful branching (BRANCH), and modification of specified general purpose registers (GR). When a /PER event occurs, a trace message is produced, and debug mode will be entered or program execution continued depending on whether RUN or NORUN was specified, respectively. (See page 131 for a complete description of the /PER command.)

A few remarks are in order regarding the two classes of tracing: supervisor call tracing with /DEBUG and /PER event tracing. As currently implemented, the /PER tracing will not trace supervisor call instructions, so a complete trace of execution requires the use of both /DEBUG and /PER. Also, when debug mode is entered following a /PER trace event, the instruction traced has completed execution, whereas upon entrance to debug mode following a /DEBUG SVC (or TSVC) trace event, the supervisor call request will not yet have been serviced by the system. In this latter case, if program execution is resumed via the RES command (see below) the system will complete the request before execution of the next program instruction. If execution is resumed using RESAT (see below), on the other hand, the program will be reentered immediately, and the request will not be serviced at all. Individual supervisor calls may therefore be "handled" by the user through the debug mode commands, and program execution continued with "RESAT &".

The current status of /DEBUG and /PER settings may be determined by issuing /QUERY DEBUG or /QUERY PER (see page 137), at "\*go" or attention level.

### Interactive Debug Mode

Debug mode allows the terminal user to examine the executing program, modify the instruction and data areas, and continue execution. As long as the terminal is in debug mode, program execution is suspended.

Debug mode will be entered if /DEBUG ON is set and the executing program produces an abend or program interruption. Debug mode will also be entered if /DEBUG is issued without operands or with the operand ENTER, if a supervisor call instruction is issued with /DEBUG SVC set, or if one of the /PER trace settings is on with NORUN set, and a /PER trace event occurs.

In the discussion of debug mode commands, the general purpose registers, floating point registers, and PSW are those possessed by



the problem program at the point of suspension of execution.

### Debug Mode Error Messages

Apologies are made for the primitive state of debug mode error messages. Except for a few cases, the only error message produced by debug mode is "???". This usually means that the command is invalid or that an expression in the command argument is invalid (e.g. invalid digit, undefined symbol). If the message "PGM INTERRUPT EXECUTING COMMAND" is produced, the user has tried to reference protected or non-existent storage. The message "\*NO TERMIN UNIT FOUND - PRDUMP ASSUMED" means that no TERMIN unit (see page 63) was defined at the time debug was entered. This will occur if the user has overridden the default unit 9 by including a /FILE statement for this unit with a unit type other than TERMIN and has not defined any other unit as a TERMIN unit. A statement of the form "/FILE UNIT=(n,TERMIN)", where n is any otherwise unused unit number, should be inserted into the program. If no TERMIN unit is found, a full memory dump will be produced, and the job terminated.

### Debug Mode Commands

In the following command descriptions, values shown as aN, iN, or rN may be specified as simple hexadecimal numbers; as expressions made up of hexadecimal numbers, decimal numbers, symbols, the operators +, -, \*, and / (with the usual hierarchy observed), and up to five parenthetical levels; or as expressions followed by "base register" specifications. A base register specification consists of one or more register numbers (hexadecimal 0 through F or decimal 0 through 15) separated by commas and all enclosed in parentheses. The contents of each register indicated is added to the value of the expression. Decimal numbers are specified by typing the symbol "#" followed by the decimal digits. Some valid expressions are shown here (X, PLACE, and VALUE are symbols which are assumed to have been previously defined).

```
18DC0
2F+10C0-#280
X*(6B/VALUE)+PLACE
VALUE*200(3)
```

Note that no blanks may appear within an expression.

An additional unary operator, @, is used to specify that the immediately following quantity is the address of a four byte area containing the value to be used rather than the value itself (indirect addressing). For example, the value of the expression "1A00" is simply hexadecimal 1A00, but the value of "@1A00" is the contents of the four byte area starting at address 1A00. If the address of the area is to be specified as an expression, the expression must be enclosed in parentheses. For example, the contents of the fullword at address 1A10 could be specified as "@1A10" or as "@(1A00+10)". The value of the expression "@1A00+10", on the other hand, would be hexadecimal 10 plus the contents of the four bytes starting at address 1A00.

Values shown as v are specified in generalized byte-string form.

This form consists of pairs of hexadecimal digits arbitrarily interspersed with EBCDIC character strings enclosed in apostrophes (using double apostrophes to indicate single apostrophes, in the usual way). Blanks may be placed between pairs of hexadecimal digits, if desired for clarity. Examples of valid string expressions are shown here.

```
C1C2C3
'ABC'
0654 6B 'HERE''S STUFF ' 768B
```

Values shown as s are "regular symbols". A regular symbol is created and modified by using it on the left side of an assignment statement (see below) and may be used in any aN, iN, or rN expression. The symbol must consist of 1 to 8 alphameric characters with the first character alphabetic.

Values shown as f are floating point values and must be specified in base 10. Some valid floating point values are shown here.

```
1.0
-6.54e3
120
254.6e-12
```

Note that the decimal point does not have to be specified, since in all cases it is clear from the context whether a floating point value or fixed point value is intended.

#### Examination and Modification of Storage

DUMP a1,a2,a3 (abbreviation D)

The contents of storage from a1+a3 through a2+a3 is printed in both hexadecimal and literal form, sixteen bytes to the line. If a3 is not specified, a value of zero is assumed. If a2 is not specified, only a single printed line is produced.

DF a1,i1,i2

The address a1 is assumed to be the address of a fixed point fullword array (INTEGER\*4 in FORTRAN, FIXED BIN (31) in PL/I, etc.) of which the first element has subscript (1). The elements (i1) through (i2) are printed in decimal, with the subscript of the first element on each line appearing at the left. If i2 is omitted, it is set equal to i1. If i1 is omitted, a value of 1 is assumed.

DH a1,i1,i2

This command is similar to DF, except that the array elements are assumed to be fixed point halfwords (INTEGER\*2 in FORTRAN, FIXED BIN (15) in PL/I).

DE al,il,i2

This command is similar to DF, except that the array elements are assumed to be fullword floating point values (REAL\*4 in FORTRAN, FLOAT DEC (7) in PL/I).

DD al,il,i2

Again, similar to DF, except that the elements are now assumed to be floating point doublewords (REAL\*8 in FORTRAN, FLOAT DEC (15) in PL/I).

FIND al,a2,v

The area of storage from al through a2 is searched for the first location containing the string v. If found, a one-line dump is printed of that location.

FINDN al,a2,v

The area of storage from al through a2 is searched for the first location not containing the string v. The first comparison is made at address al, and each succeeding comparison is made at the address computed by adding the number of bytes in the string v to the address at which the previous comparison was made.

REP al,v (abbrev. R)

The string v is moved into storage starting at address al, replacing that portion of storage byte for byte.

STF al,il,a2

The address al is assumed to be the address of element (1) of an array of fixed point fullwords (cf. DF, above). The (il)th element of this array is replaced by the value of a2. If il is omitted, a value of 1 is assumed.

STH al,il,a2

Similar to STF, except that the array is assumed to consist of fixed point halfwords (cf. DH, above).

STE al,il,f

Similar to STF, except that the array is taken to be made up of fullword floating point values (cf. DE, above).

STD al,il,f

Not unlike STF, except that doubleword floating point values are assumed to make up the array (cf. DD, above).

#### Examination and Modification of General Purpose Registers

DR rl

The current contents of general purpose register rl is printed in hexadecimal form, character form, and as a signed decimal number. If rl is typed as a value between hexadecimal 10 and 15, the value is taken as equivalent to the corresponding value between hexadecimal A and F. rl may therefore be regarded as a decimal or hexadecimal number, as the user prefers.

REGS

All 16 general purpose registers are printed in hexadecimal form.

STR rl,al

The current contents of general purpose register rl is replaced by the value al.

Examination and Modification of Floating Point Registers

DFR rl

The contents of floating point register rl is printed in hexadecimal, character, and decimal floating point form.

STFR rl,f

The current contents of floating point register rl is replaced by the value f.

Examination and Modification of the PSW

PSW

The current value of the PSW is printed in hexadecimal.

SPM al

The rightmost 8 bits of the value al replace the "instruction length - condition code - program mask" byte of the PSW.

Note that the instruction address portion of the PSW may be modified by the RESAT command, described below under "Resumption of Execution".

Symbol Definition

assignment statement: s=al

The regular symbol s is defined, if necessary, and is given the value al.

SYMBOLS <param> (abbrev. SYM)

If <param> is not specified, all currently defined regular symbols are printed along with their values. If <param> is given as SYSTEM, the current symbol table will be dropped and the system symbol table (the table of all CSECT and entry point names in the current resident nucleus of VPS) substituted. Note that the system symbol table is read-only, and new symbols may not be defined while SYMBOLS SYSTEM is in effect. If <param> is specified as \*, debug will return to using its own table, and if SYMBOLS \*SYM is specified while SYMBOLS \* is in effect, debug will return to using the table that was defined prior to issuing SYMBOLS \*. The address of this latter table always appears in the SYMBOLS listing opposite the name \*SYM, but \*SYM cannot be used in expressions. Finally, <param> may be specified as an aN-type expression, in which case the value of the expression will be used as the symbol table origin.

**ERASE s**

The regular symbol *s* is deleted from the symbol table.

**Special Symbols**

Two special symbols are always defined and may only appear in *aN*, *iN*, and *rN* expressions. These are:

\$ The *a1* value from a storage examination command (DUMP, DF, DH, DE, DD) or storage modification command (REP, STF, STH, STE, STD) or the printed address from a successful FIND command, whichever is most recent.

& The instruction address from the PSW.

Calculations and Conversions**HEX *a1*** (abbrev. see text)

The expression *a1* is evaluated, and the result printed in hexadecimal. Note that the command word HEX need not be typed: If a valid expression is entered which does not look like a debug command, the resulting value will be typed back. This interpretation can always be forced by typing a blank in front of the expression.

**DEC *a1***

The expression *a1* is evaluated, and the result printed as a signed decimal number.

**FLOAT *v***

The string *v* is left-justified in a field of 8 binary zeroes, and the result interpreted as a long precision floating point number. The value is printed out in decimal floating point form.

**FHEX *f***

The decimal floating point number *f* is converted to internal floating point representation and printed out in hexadecimal form.

**TOD *a1,a2***

The 64 bit value with left half *a1* and right half *a2* is taken as a time of day clock value reckoned from 0:00:00 Monday, 1 January 1900 and converted to time and date. If *a2* is omitted, a value of zero is assumed.

Resumption of Execution**RES**

Program execution is resumed with the current register and PSW values.

**RESAT *a1***

The instruction address in the PSW is replaced by the value of *a1* and program execution is resumed.

RETURN

The instruction address in the PSW is replaced by the contents of bits 8 through 31 of general purpose register 14 and program execution is resumed. This is particularly handy when resuming execution after a program interruption caused by a branch and link to location zero, as happens when a program attempts to call a non-existent (unresolved) subroutine via a "BALR 14,15".

Job Termination

/CANCEL

The system /CANCEL (page 100) command may be issued to abnormally terminate execution and return to \*go.

QUIT

This command is used when return to "\*go" level is desired, but the user does not wish the termination to be considered abnormal by the system. The consideration which usually indicates use of QUIT rather than /CANCEL is the choice of DISP field for determination of the final disposition of one or more files. For example, if a LIBOUT or DISK file has DISP=(NEW,KEEP,DELETE), /CANCEL will cause the file to be deleted, whereas QUIT will cause the file to be kept.

## Chapter 9: The Batch

In addition to the normal terminal mode, VPS may also be accessed in a batch mode. The printed output from a job run on the batch appears on the high speed line printer and must be picked up by the user at the input/output window in the Computing Center (basement, 111 Cummington Street). A job may be submitted to the batch from a terminal via the /BQx command (page 98) or by punching it on cards and leaving it at the input/output window.

While the batch mode is obviously less desirable than the terminal mode for program development, it is useful for applications which produce a relatively large amount of printout or which the user wishes to run without having to attend a terminal.

A job submitted with the /BQx command is in the same format as a job run at the terminal (see page 17), but a job submitted on cards must also contain one /ID statement, one /END statement, and, if a batch password is defined for the account being used, one /PW statement.

When either type of batch job is submitted it is placed in a queue based on its time limit and class. The job is placed in the queue so as to interleave jobs submitted by a single user with those submitted by other users. The queues can be displayed by executing the program Q (/EXEC Q) at the terminal.

Users who do not require fast turnaround and would like to take advantage of reduced cost may submit their job to the background batch. The background batch selects jobs during non-prime-time hours and operates at low priority. The total cost of a job executed in the background batch is reduced by 50%. To enter a job into the background batch, use the /BQZ command (see page 98).

### The /ID Statement

The /ID statement, which must be the first card of the job, specifies the account number to be charged and is also used to override the system default limits on CPU time, printed output, and punched output. The format of the /ID statement is

```
| /ID aaaaaaaa          ttt ppp ccc |
|                          ↑  ↑  ↑  |
|                          30 34 38  |
| column                  |
```

where /ID starts in column 1 and at least one blank separates it from the account number aaaaaaaa. The fields ttt, ppp, and ccc are the CPU time, printed output, and punched output overrides, respectively, and must appear in the columns shown (30, 34, and 38). Each field is three columns wide.

The CPU time override may be given as a 3 digit decimal number of minutes (e.g., 010 means 10 minutes) or as a 2 digit decimal number of seconds followed by the letter S (e.g., 30S means 30 seconds). In addition, the character string MAX may be coded in this field, indicating that the system is to allow the job the maximum time limit (currently 24 hours). If this override is omitted, a value of 30 seconds is used.

The printed output override is a 3 digit number of pages and applies to the total output of all TERMOUT and PRINTER units defined for the job (including the system default TERMOUT unit, normally addressed as unit 6, ddname SYSPRINT). If this override is omitted, a value of 10 pages will be used. If 999 is coded here, no limit will be enforced.

Finally, the punched output override is a 3 digit number of punched cards which applies to the total output of all PUNCH units (including the default PUNCH unit addressable as unit 7, ddname SYSPUNCH). The default value for this limit is 50 cards, and a coded value of 999 means that no limit is to be enforced.

#### The /END Statement

The /END statement is used to indicate the end of the job and must, therefore, be the last card of the job - following all system control statements and all data cards. The /END statement consists of simply:

```
| /END |
```

where /END starts in column 1.

#### The /PW Statement

If a batch password is defined for the user's account, a /PW statement must be included in any batch job submitted on cards under that account. The /PW statement may appear anywhere between the /ID statement and the /LOAD statement. In the interest of security and accountability, it is recommended that all accounts have batch passwords, and that these passwords be changed periodically. Batch passwords are created and modified with the /PROFILE terminal command (page 135).

Batch jobs submitted with the /BQx command do not need /PW statements. A maximum of one /PW statement will be ignored for a batch job submitted via /BQx and for a terminal job. A second /PW statement will cause rejection of the job in any of the several environments. The format of the /PW statement is:



```
| /PW pppppppp |
```

where /PW starts in column 1, and at least one blank separates it from the password pppppppp. It is advisable to flip the "print" switch off on the keypunch when punching the card so that the password is not printed at the top. Obviously, complete security requires that the /PW card not be allowed to fall into unauthorized hands.

#### Use of Privileged Accounts on the Batch

A privileged account may be used to run a batch job in the same manner as an ordinary account. However, as a protective measure, the system will disable all privileges for any job submitted on cards, irrespective of the account being used. In order to run a batch job with one's privileges intact, it is necessary to submit the job via /BQx.

#### /LIBRARY Command on the Batch

The /LIBRARY command (page 124), which is used at the terminal to obtain a listing of the names of library files saved under the user's account, may also be used on the batch to get this listing printed out on the high speed line printer. This is done by running a job consisting of the statement:

```
| /LIBRARY <parameters> |  
| /LIBR |
```

where <parameters> represents a list of parameters of the same form as the parameters which would be used with the /LIBRARY terminal command. If no parameters are given, FULL is assumed. For example, to print out a library listing for the account BS980B04, one would use the job:

```
/ID BS980B04  
/LIBRARY  
/END
```

#### /SAVE Command on the Batch

The /SAVE command (page 149) may be used on the batch to save a deck of cards in the library. The card bearing the /SAVE command must immediately follow the /ID statement, and the cards to be saved must be placed immediately after the /SAVE command. The /SAVE command has the form:

```
/SAVE <parameters>
```

where <parameters> represents a list of parameters in the same form as would be entered on a /SAVE command issued at the terminal. For example, to save a deck of cards under the name BICYCLE using the account FH82822, we would submit the job:

```
/ID FH82822
/SAVE BICYCLE
....
.
.   the deck of cards
.
....
/END
```

The /RSAVE and /REPLACE commands are not available on the batch.

#### /NEWS Command on the Batch

The /NEWS command (page 129) may be used on the batch to obtain a printout of the current Computing Center news file. The command has the same form on batch as at the terminal, namely:

```
/NEWS
```

For example:

```
/ID HBK760
/NEWS
/END
```

## Section 2: VPS Terminal Commands

This section describes the VPS terminal commands, which are arranged here in alphabetical order. The table on the following two pages indicates in which modes the individual commands are valid and where their descriptions are to be found. The table also indicates which commands use the "comma" syntax. This is a syntax in which parameters are separated by commas, and embedded blanks are not allowed. The commands not indicated as following the comma syntax have their parameters separated by blanks, and any number of blanks may appear between two parameters.

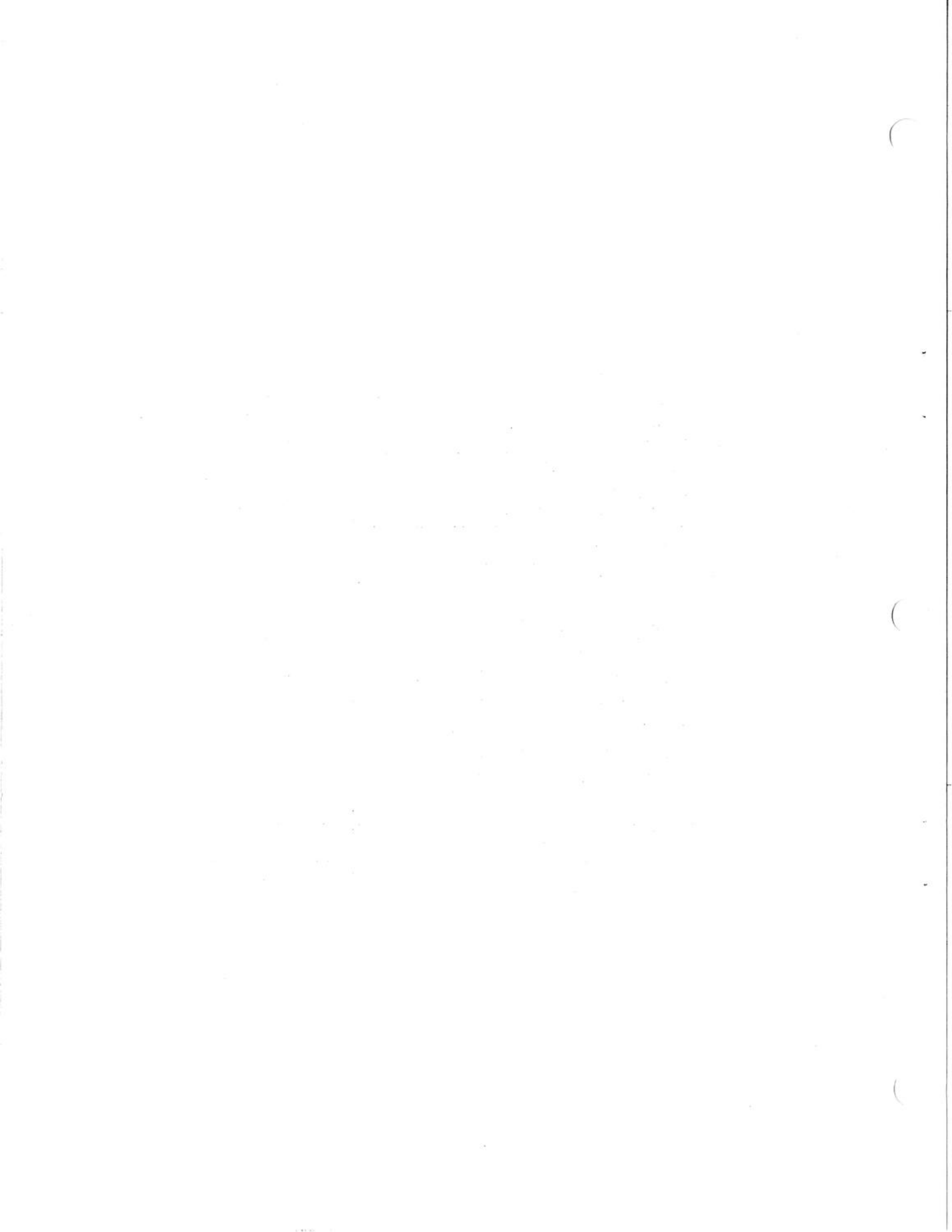
VPS Terminal Command Table

command	page	valid for				uses
		*go read	attn read	conv read	sign on	comma syntax
/ACCT	95	X				
/ATTN	96		X			
/BLIP	97	X	X			
/BQA	98	X	X			note 1
/BQB	98	X	X			note 1
/BQC	98	X	X			note 1
/BQZ	98	X	X			note 1
/CANCEL	100		X	X		
/CATL	101	X				
/CE	102		X			
/CLIST	103		X			
/CO	105		X			
/CTL	106	X	X			
/DAILY	111	X	X			
/DEBUG	112	X	X			
/DISPLAY	114	X				X
/DS	115	X				X
/DSPURGE	117	X				X
/DSRENAME	118	X				X
/EDIT	119	X				X
/EXEC	120	X				note 1
/ID	122				X	X
/LIBRARY	124	X				X
/LIST	126	X				X
/LOG	127	X	X			
/MESSAGE	128	X	X			
/NEWS	129	X				
/OFF	130	X	X		X	
/PER	131	X	X			

Note 1: This command borrows part or all of its syntax from the /INCLUDE statement and must be considered as a special case.

VPS Terminal Command Table

command	page	valid for				uses
		*go read	attn read	conv read	sign on	comma syntax
/POST	134		X			
/PROFILE	135	X				
/PURGE	136	X				X
/QUERY	137	X	X			
/RATES	139	X				
/RENAME	140	X				X
/REPLACE	143	X				X
/REPLY	145	X	X			
/RO	146		X			
/RSAVE	147	X				X
/SAVE	149	X				X
/SCRIPT	151	X				X
/SCROLL	153	X	X			
/SKIP	154		X			
/SLEEP	155	X				
/SORT	156	X				X
/STATUS	157	X	X			
/SUMMARIZE	158	X				
/TABIN	159	X	X			
/TABOUT	161	X	X			
/TABS	163	X	X			
/TABSET	165	X				
/TAPE	167	X	X			
/TEXT	168	X	X			
/USERS	169	X	X			
/VERIFY	170	X	X			
/XEDIT	171	X				X
/?	172	X	X			



Levels:

"\*go"

The /ACCT command may be used to print accounting data at the user's terminal. The format of the /ACCT command is written below.

```
| /ACCT |
```

Notes:

The accounting data consists of the accumulated terminal and batch activity since the account was last reset (normally, this will be the accumulated activity since the account was created). Information printed will consist of CPU and connect time for terminal activity and CPU time plus the number of cards read and punched and pages printed for batch activity.

Accounting information for library files and work files is also listed. In this case, the information is accumulated for the current billing period (normally, the billing period begins on the first of each month) and consists of the tracks in use, the track limit, and the accumulated track charges.

Note that the information listed does not include data for the current terminal session, as the information is updated only at sign-off.

/ATTN

Levels:

Attention

The /ATTN command may be used during program execution to pass an attention interrupt to an executing program. This command is useful when executing programs that have attention exit routines, in that when it is entered, control will be passed to the exit routine. If the program has no attention exit routine, entering this command will have no effect. The format of the /ATTN command is written below.

```
| /ATtn |
```

Notes:

The /ATTN command can be used only at attention level. Attention level is reached by striking the attention, or break, key during program execution.

Reaching attention level and entering the /ATTN command is equivalent to striking the attention, or break, key twice rapidly.



Levels:  
 Attention  
 "\*go"

The /BLIP command may be used to cause program status information to be printed at the user terminal every few seconds of CPU time during the execution of a program. The format of the /BLIP command is written below.

```
/BLip [ON  

       [OFF]
```

The keywords available on the /BLIP command are:

ON

specifies that the system should print program status information at the user terminal every few seconds of CPU time. ON is the default.

OFF

specifies that the system should suspend printing of program status information.

Notes:

Program status information will be printed in the form:

\*hh:mm:ss cpu xx.xx io yyy

where -

hh:mm:ss - is the time of day.

xx.xx - is the accumulated seconds of CPU time used during the execution of the current program.

yyy - is the accumulated number of input/output operations performed during the execution of the program.

/BQA, /BQB, /BQC, /BQZ

Levels:

Attention

"\*go"

The /BQ commands (/BQA, /BQB, /BQC, /BQZ) are used to submit jobs to the VPS batch. The last character of the command (a, b, c, or z) identifies the class of the job. A job submitted in this manner will not be run as part of the user's terminal session, rather it will be queued by the system and run as if the job had been punched on cards and submitted through the card reader. Output, in the form of printer output and punched cards, may be picked up at the Computing Center Job Submission Window after the job has run. The job classes are ordered by priority, A being the highest. All terminal users may use either the /BQC or the /BQZ command to submit jobs. The /BQZ command is used to submit jobs to the background batch. Only privileged accounts may use /BQA and /BQB. (For more information on the background batch and batch queues see page 87.) The format of the /BQ commands is written below.

```
| /BQx [Time nnn] [Pages nnn] [PUunch nnn] {Job ...}  
|      [Time MAX] [Pages 10] [PUunch 50]  
|      [Time 30S]
```

The /BQ commands all have the same format. The time, pages, and punch keywords are used by each command processor to form a valid VPS batch "/id" statement. Therefore, the user should not place a "/id" statement in the stream being submitted as a batch job. The keyword options are defined below.

TIME

specifies the maximum amount of CPU time the submitted job should take for execution. It may be specified as the number of minutes (if the number is followed by an 'S' it indicates seconds) or as 'MAX' which will set the limit to 24 hours. If the job exceeds this limit, the system will automatically cancel it. The default is 30 seconds.

PAGES

specifies the maximum number of pages to be printed. It must be specified as a number. If 999 is specified, no limit will be placed on the number of pages printed. If the limit is exceeded, the job will be cancelled automatically by the system. The default is 10 pages.

PUNCH

specifies the maximum number of cards to be punched. It must be specified as a number. If 999 is specified, no limit will be placed on the number of cards punched. If the limit is exceeded, the job will be cancelled automatically by the system. The default is 50 cards.

JOB

specifies the beginning of the string (represented above by "...") defining the job to be run. Everything from the first non-blank character after the Job keyword to the end of the command line will be taken as the job definition. Most likely, the job definition will consist of one or more file names, separated by commas, which when merged together by the system form the job being submitted. Note, the Job keyword must follow the Time, Pages, and PUnch keywords.

Example:

The command -

```
/BQC J MYFILE
```

is equivalent to the following deck of punched cards submitted to the VPS batch -

```
/ID MYACCT  
/INC MYFILE  
/END
```

Notes:

The VPS files, \* and \*2 may not be specified as part of the job definition.

/CANCEL

Levels:

Attention

Conversational Read

The /CANCEL command may be used during program execution to cancel the execution and printout of the program. The format of the /CANCEL command is written below.

```
| /CANcel |
```

Notes:

The /CANCEL command can be used only at attention level or in response to a conversational read. Attention level is reached by striking the attention, or break, key during program execution.

Levels:  
  "\*go"

The /CATL command may be used to print a listing of available programs and library subroutines at the user's terminal. The format of the /CATL command is written below.

```
| /CATL |
```

/CE

Levels:

Attention

The /CE command may be used during program execution to terminate program execution without terminating program output. After this command is entered, the user will continue to receive program output generated prior to entering the /CE command. The format of the command is written below.

```
| /CE |
```

Notes:

The /CE command can be used only at attention level. Attention level is reached by striking the attention, or break, key during program execution. After the remaining output is printed, the terminal session is returned to the "\*go" level.

Levels:  
Attention

The /CLIST command may be used to modify, list, or clear the conversational read list. The conversational read list is a list of statements/commands to be processed by the system at the "\*go" level or by an executing program which reads the conversational read unit (defined by the keyword TERMIN on a /FILE statement, see page 63). The format of the /CLIST command is written below.

```

/CList [ List
        Clear
        Head statement
        Tail statement ]

```

**LIST**

specifies that the current contents of the conversational read list are to be listed at the terminal. LIST is the default.

**CLEAR**

specifies that the current contents of the conversational read list are to be cleared. Note that after the list is cleared the next conversational read by the system or a read of the conversational read unit by an executing program will be issued to the user terminal.

**HEAD**

specifies that the statement following the HEAD keyword is to be placed at the top of the conversational read list. If, after this command is entered the executing program issues a read to the conversational read unit, the "line" read will be the statement following the HEAD keyword. If the program issues no read to the conversational read unit, the statement following the HEAD keyword will be the next command executed after the terminal session returns to the "\*go" level.

**TAIL**

specifies that the statement following the TAIL keyword is to be placed at the bottom of the conversational read list.

Notes:

The conversational read list may be thought of as a queue which can be added to at either end, but accessed by the system only at the head.

/CLIST

The /CLIST command can be used only at attention level. Attention level is reached by striking the attention, or break, key during program execution.

The conversational read list will be cleared if the program abnormally terminates or is cancelled.

Example

In the following example a program, MATRICES, is executed. After execution begins, the terminal user enters attention level and builds a conversational read list which will be passed to the program as it successively reads the conversational read unit. In this example conversational reads are terminated by the string "endinput". Note that the program cannot have issued a conversational read before the list had been built or the terminal would have unlocked.

```
*GO
matrices
!                               (attention level entered)
:/cl h  4.6  7.2  9.6
!                               (attention level entered)
:/cl t 10.4 18.6 19.2
!                               (attention level entered)
:/cl t endinput
!                               (attention level entered)
:/cl list
  4.6  7.2  9.6
10.4 18.6 19.2
ENDINPUT
.
.                               (program continues execution)
.
```



Levels:

Attention

The /CO command may be used during program execution to cancel program output without terminating program execution. After the /CO command has been entered, no further output will be printed at the user's terminal until one of the following events occurs: the program terminates execution; the /RO command is entered which resumes terminal output; or the program issues a read to the conversational read unit - in which case terminal output will resume with output generated after the conversational read. The format of the /CO command is written below.

```
| /CO |
```

Notes:

The /CO command can be used only at attention level. Attention level is reached by striking the attention, or break, key during program execution.

/CTL

Levels:

Attention  
"\*go"

The /CTL command may be used to define or change terminal and session control functions. The format of the /CTL command is written below.

```
/CTL [DElim c] [BS c] [LDe1 c] [EScape c] [HEX c hex]
      [DElim ;] [BS <-]
      [TAb c] [FLip c] [LINEsize nnn] [CMd ON]
      [TAb tab key] [LINEsize 150] [CMd OFF]
      [BEginner ON] [Msgs ON] [CPI nn] [LMIN nn]
      [BEginner OFF] [Msgs OFF]
      [PFnn function]
```

ACL

DELIM

defines the logical delimiter character. The character specified (represented by "c" above) must be a special or control character (any character other than a "/", all upper and lower case alphabetic characters, and all numeric characters). The default is ";".

BS

defines the logical backspace character. The character specified must be a special or control character. The default is the physical backspace key (represented by "<-" above) on all 2741 type terminals and most ASCII terminals (note that on all ASCII terminals it is also Control h).

LDEL

defines the logical line delete character. The character specified must be a special or control character. If the LDEL keyword is entered with no character, the logical line delete character function is disabled.

ESCAPE

defines the logical escape character. The character specified must be a special or control character. If the ESCAPE keyword is entered with no character, no logical escape character is in effect.

## HEX

defines the logical hexadecimal substitution character followed by the 2-digit hexadecimal code it represents. The character must be a special or control character. After the hexadecimal substitution character is defined, any occurrence of the substitution character in an input line from the terminal will automatically be replaced with the 2-digit hexadecimal code. If the HEX keyword is entered with no parameter, no hexadecimal substitution will occur.

## TAB

defines the logical tab character. The character specified must be a special or control character. The default is the physical tab key.

## FLIP

defines the /EDIT flip character which when appended to editor commands flips the verify/brief setting (see the VPS Utilities Manual for further information). The character specified must be a special or control character. If the FLIP keyword is entered with no character, the /EDIT flip function is disabled.

## LINESIZE

defines the output line size for the terminal. Output lines will automatically be split into chunks of the size specified for terminal output. The default is 150.

## CMD

defines whether commands written to the conversational read unit (by executing programs), their responses, and any error messages generated by these commands will be printed at the terminal. If ON is specified the commands, responses, and any error messages will be printed. If OFF is specified no information on commands written to the conversational read by executing programs will be printed. If ERROR is specified responses and error messages will be printed. OFF is the normal mode unless set to ON in the user profile. ON is the default.

## BEGINNER

specifies whether the terminal session is to enter "beginner mode". ON specifies that beginner mode is to be entered. OFF specifies that normal VPS mode is to be entered. ON is the default.

## MSG

specifies whether the user wishes to receive messages from the VPS operator and other users. If ON is specified messages sent to the user's terminal will be printed. If OFF is specified messages sent to the user's terminal will not be printed. ON is the

/CTL

default.

#### CPI

specifies the number of characters per idle. On some ASCII terminals it may be necessary to pad the output line sent by VPS with idle characters after the carriage return character. This is needed since the carriage on these terminals may not return to the left margin before the next line is received by the terminal for printing - causing loss of characters on output. The CPI value may be set at sign-on using special terminal numbers specified on the /ID command (see the /ID command for more information). In some cases it may be necessary to change the CPI during the terminal session. The CPI may be thought of as the return velocity measured in character spaces per character time - as the CPI value is decreased the number of idle characters placed after the carriage return character is increased. For example, a CPI value of 10 will cause the system to place 1 idle character after the carriage return character for every 10 characters in the line to be printed. The maximum value allowed is 25.

#### LMIN

specifies the shortest line length the user wishes to have sent to the terminal on output. If the LMIN parameter is set, lines shorter than the value specified will be padded with idle characters (the idles will be placed before the carriage return character). This option is generally used for terminals of the print-belt type, which require a specific minimum amount of time to print each line. The maximum value allowed is 36.

#### PFnn function (for 3270 display terminals only)

defines the screen modification, command, or set of commands which is to be performed when the designated program function key is pressed. "nn" is the PF key number (01 through 24) and "function" may be any of the following:

```
BTAB
CLEAR
CLEOF
LEND
LLOCK
TAB x x x x...
```

Action upon pressing the defined PF key -

BTAB - cursor returns to the first position of the line.

CLEAR - clears the screen, resets the cursor. Identical to ALT-CLEAR keys only better (you have only one key to hit and you can't become disconnected which can occasionally happen using the ALT-CLEAR keys).

copy l.n-adr  
form l.n-adr

/CTL

CLEOF - clears the screen from and including the line in which the cursor is found and then resets the cursor to that line.

LEND - cursor is set to the end of the line.

LLOCK - locks the line on which the cursor is found. Locked lines will not be scrolled off the screen. Locked lines are erased when the screen is cleared.

TAB - causes the cursor to be placed at the next tab position on the line.

Other Functions -

Besides the specific internal functions mentioned above, program function keys may be set to any string of characters - such as a VPS terminal command, a string of commands separated by the line end character, Editor commands separated in the same way, etc. The form of the /CTL command in this case is as follows.

```
/ctl pfnn delay STRING  
          immed
```

Note: Due to the nature of the VM - VPS interface the STRING must be entered in upper case (this is not true of the internal functions).

delay - when the PF key is hit the string is displayed on the current line and the cursor is set to the end of that line.

immed - when the PF key is hit the string is displayed on the current line and sent to the system as if the ENTER key had been hit.

For example, a program function key has been defined as follows:

```
/ctl pf13 immed L RASPBERRIES;P 5
```

Each time the PF13 key is hit the following would be sent to the system as if you had typed it in and hit ENTER (in this example it's advantageous to be in the Editor when you do this):

```
l raspberries;p 5
```

Another program function key has been defined as follows:

```
/ctl pf18 /BQC T 1 P 20 J
```

In this case, the PF key is set as a delayed function (the default) so that when PF18 is hit the following would appear on the screen -

```
/bqc t 1 p 20 j_
```

/CTL

At this point the rest of the command would be completed and the ENTER key hit.

Levels:  
"\*go"

The /DAILY command may be used to list the message of the day which is also printed on the user's terminal at sign-on. The format of the /DAILY command is written below.

```
| /DAILY |
```

Levels:

Attention  
"\*go"

The /DEBUG command is used to enter interactive debug mode or to set or change program trace options (see Chapter 8 for a full explanation of interactive debugging). The format of the /DEBUG command is written below.

/DEBug	[ON OFF ENTER]	[TSvc Svc NOSvc]	[Eoj NOEoj]
--------	----------------------	------------------------	----------------

ON

specifies that an abend or program interrupt resulting from program execution will place the terminal in debug mode.

OFF

specifies that an abend or program interrupt resulting from program execution will not place the terminal in debug mode and all debug settings are to be turned off.

ENTER

specifies that interactive debug mode is to be entered immediately.

TSVC

specifies that a message is to be produced each time a supervisor call instruction is issued by the program, but that the program is to continue execution.

SVC

specifies that a message is to be produced each time a supervisor call instruction is issued by the program, program execution is to be suspended at this point, and the terminal is to be placed in debug mode.

NOSVC

specifies that TSVC and SVC are to be turned off.

EOJ

specifies that debug mode is to be entered when the program terminates execution whether it terminates normally or abnormally.



NOEOJ

specifies that EOJ is to be turned off.

Notes:

The /DEBUG command may be entered at either "\*go" or attention level. Attention level is reached by striking the attention, or break, key during program execution. The /QUERY command may be used to determine the current /DEBUG settings.

When the /DEBUG command is entered at "\*go" level any settings will apply to the next program executed, while settings issued at attention level apply to the currently executing program. Except for /DEBUG ON, all settings are turned off when the terminal is returned to "\*go" level after program execution terminates. The ON setting remains active for the remainder of the terminal session unless it is explicitly turned off with /DEBUG OFF.

The format of the supervisor call trace message is as follows:

aaaaaa SVC nnn (ssssssss)

aaaaaa - is the address of the SVC (or EX) instruction.

nnn - is the decimal representation of the SVC number.

ssssssss - is the standard system symbol for the SVC.

/DISPLAY

Levels:  
"\*go"

The /DISPLAY command may be used to list all or part of a file at the user's terminal. After the command is entered, the lines specified will be printed at the terminal with a line number assigned by the /DISPLAY command appended to the left of each line. The format of the /DISPLAY command is written below.

```
/DISPlay [ num1,          ] [ num2,          ] [ filename ]  
          [ 1,            ] [ LAST,           ] [ *         ]  
          [ LAST,         ] [ LAST-num4,    ] [ *2        ]  
          [ LAST-num3,    ]
```

**first parameter**

specifies the line number in the file where printing is to begin, the first line in the file being line number 1. The parameter may be specified either as a number (represented by "num1" above) relative to the beginning of the file; as the keyword LAST, indicating that the last line in the file is to be printed; or as the keyword LAST followed by a minus sign ("-") and a number (represented by "num3" above), indicating that the first line to be printed is line number "LAST-num" relative to the end of the file. If this is the only parameter specified, only one line will be displayed. Line number 1 is the default.

**second parameter**

specifies the line number in the file where printing is to end. The parameter may be specified as either a number (represented by "num2" above) relative to the beginning of the file; as the keyword LAST, indicating the printing should continue until the end of the file is reached; or as the keyword LAST followed by a minus sign ("-") and a number (represented by "num4" above), indicating that the last line to be printed is line number "LAST-num" relative to the end of the file. LAST is the default only if the first parameter is not specified.

**filename**

specifies the name of the file to be printed. It may be specified as a valid file name or a "\*" or "\*2" indicating the appropriate temporary file. The default is the temporary file "\*".

Levels:  
 "\*"go"

The /DS command may be used to list information pertaining to work files. The format of the /DS command is written below.

```

/DS      [Brief] [Dlname=workfilename]      [,SAVE
         [Full]  [ALL]                      [,SAVE=filename
                                         [,RSV=filename]

         [,Volume=volser
         [,Volume=(volser,volser,...)]

         [,Acct=account
         [,Acct=acct*
         [,Acct=account1-account2
         [,Acct=acct1*-acct2*
         [,Acct=(acctspec,acctspec,...)]
  
```

#### first parameter

specifies the type of listing desired. If BRIEF is specified, the following information is listed for each work file: logical record length, block size, record format, data set organization, creation date, last reference date, tracks allocated to the file, and tracks used by the file. If FULL is specified the following additional information will also be listed: the key length (if present), cylinder and head location of the work file, number of extents, secondary allocation quantity and type (if specified when the file was created). BRIEF is the default.

#### DSNAME=

specifies a work file name for which information is to be listed. Due to the naming conventions for work files (see page 54), the access code field and/or the name field may be replaced by an asterisk("\*"). In this case, the field(s) appearing as an asterisk will not be used in the search for the work files (i.e. if an asterisk appears in the access code field of the work file name, information about all of the work files matching the account and name portion will be listed).

#### ALL

specifies that all data sets found for the account number(s) or account number range(s) are to be listed. ALL is the default.

/DS

SAVE

specifies that the listing is not to be printed at the terminal. Instead the listing is placed in the temporary file \*2.

SAVE=

specifies the name of a new file into which the /DS command will place the listing. If this keyword is specified, the listing will not be printed at the terminal.

RSAV=

specifies the name of an existing file into which the /DS command is to place the listing. If this keyword is specified the listing will not be printed at the terminal.

VOLUME= (for privileged accounts only)

specifies the volume(s) which are to be searched for work files. It may be specified as a single volume serial number or as a list of volume serial numbers enclosed in parentheses and separated by commas. When this parameter is omitted, information is printed for all work files that satisfy the account specifications and are cataloged in the VPS catalog.

ACCT= (for privileged accounts only)

specifies the account number(s) or range(s) of account numbers of the files to be listed. It may be specified as a single account number; a range of account numbers separated by a dash ("-"), in which case all account numbers within that range will be used; a "partial account number" (represented by "acct\*" above) followed by an asterisk ("\*") indicating that all account numbers beginning with that "partial account number" should be used; a range of "partial account numbers" where each "partial account number" is followed by an asterisk and the two numbers are separated by a dash, indicating that the account numbers to use are all account numbers starting with the first account where the number begins with the "partial account number" (represented by "acct1\*" above) to the last account where the number begins with the "partial account number" (represented by "acct2\*" above); or finally, a list of any of these above specifications enclosed in parentheses and separated by commas. The default is the terminal user's account number.

Levels:  
 "\*"go"

The /DSPURGE command may be used to delete work files from the VPS storage volumes. Note that this command will also remove the data set entries from the VPS catalog. The format of the /DSPURGE command is written below.

<pre> /DSPurge {accesscode.account.name } ,...           *.account.name           accesscode.account.*           *.account.* </pre>
---

#### workfilename

specifies the work file(s) to be purged. Note that due to the naming conventions for work files (see page 54), the access code field and/or the name field may be replaced by an asterisk ("\*"). In these cases, the field(s) appearing as an asterisk will not be used in the search for work files to purge (i.e. if an asterisk appears in the access code field of the work file name, all files matching the account and name portion will be purged). Multiple work file names may be specified on the same command line but must be separated by commas.

/DSRENAME

Levels:  
"\*go"

The /DSRENAME command may be used to rename (and recatalog) work files. The format of the /DSRENAME command is written below.

```
| /DSRename {oldworkfilename},{newworkfilename} |
```

oldworkfilename

specifies the name of an existing work file.

newworkfilename

specifies the name to which the file is to be renamed. An alternative form of this parameter allows an equal sign ("=") to be specified in place of the access code portion of the work file name, the account portion, and/or the name portion. An equal sign indicates that that portion of the name is to remain unchanged.

Levels:  
 "\*"go"

The /EDIT command may be used to initiate the general purpose VPS file manipulation program which allows the user to change or create files (see the VPS Utilities Manual for more extensive information). The format of the /EDIT command is written below.

```
/Edit [filename] [,LRecl=number]
      *
      *2
```

**filename**

specifies the name of an existing file which is to be changed. The file name parameter may be either a valid file name or a "\*" or "\*2" indicating the appropriate temporary file. If the /EDIT command is entered with no file name the command assumes a new file is being created.

**LRECL=**

specifies that the maximum line length of the file being changed or of the file being created is to be set to the value indicated. If LRECL is not specified and a new file is being created, the maximum line length will default to 80. If LRECL is not specified and an old file is being changed the maximum line length will be taken from the old file. If an old file is specified and an LRECL is coded which is smaller than the maximum line length, the file will be truncated. The smallest LRECL value which can be specified is 20, the largest is 255.

**Notes:**

The largest file that can be edited is about 5000 records if the maximum line length is 80 bytes (for files with a maximum line length different from 80 bytes the maximum record number will vary accordingly). The reader is referred to the description of the /XEDIT command (page 171) for editing large files.

Levels:  
"\*go"

The /EXEC command may be used to execute one or a series of files and/or VPS control statements. The format of the /EXEC command is written below.

```
/EXec {file name | cntl stmt, file name | cntl stmt, ...}
```

Notes:

If more than one file name/control statement is specified they must be separated by commas.

The /EXEC command can be thought of as a pseudo /INCLUDE statement, in that the system will process the command as though it were reading a library file consisting of one /INCLUDE statement having the parameters as specified on the /EXEC command (see Chapters 2 and 3).

VPS supports an implicit form of the /EXEC command where the command verb itself is optional. If the first or only parameter of the /EXEC command is a file name, then the characters "/EXEC " may be omitted and the file name should be typed starting at the left margin of the keyboard. If the first parameter of the /EXEC command is a VPS control statement then the characters "/EXEC" may be omitted and the VPS control statement should be typed after at least one blank is typed.

Example

In the following example an assembler program residing in the file EDGES is assembled. Note that the first example follows the open form of the /INCLUDE statement, the second follows the delimited form (see Chapter 2 for a full explanation of the /INCLUDE statement). A user could enter either of the following - both being equivalent.

```
*GO
/ex /load vsasm,,parm='deck,,list',edges
.
.
.
```

(or)



```
*GO  
  '/load vsasm,param='deck,list'',edges      (note leading blank)
```

```
  .  
  .  
  .
```

If the VPS control statement "/LOAD VSASM" was the first line in the file EDGES, the program could be assembled in either of the following ways - both being equivalent.

```
*GO  
/ex edges
```

```
  .  
  .  
  .
```

(or)

```
*GO  
edges
```

```
  .  
  .  
  .
```

/ID

Levels:

Valid only at \*VPS/370 SIGN ON.

The /ID command is used to identify a user to VPS and to allow the user access to the system. The format of the /ID command is written below.

```
/ID [termnum,] {account}
```

termnum

specifies a terminal number to VPS. The terminal number need not be specified unless the terminal being signed on to VPS is an ASCII terminal having unusual characteristics. With some ASCII terminals it is necessary for VPS to modify output lines being sent to the terminal due to the physical characteristics of the specific terminal. Output lines may be modified by VPS in two ways - idle characters may be inserted into the lines after the carriage return character (for terminals whose carriage may not return to the left margin before a new output line has been received - causing possible loss of characters) and idle characters may be added before the carriage return character for terminals which require a specific minimum amount of time to print each line (usually print-belt type terminals). To specify the number of idle characters to be placed after the carriage return character, VPS uses the "characters per idle" value (CPI). A CPI of 10 indicates that VPS will add 1 idle character after the carriage return character for every 10 characters in the output line to be printed. To specify the number of idle characters to be placed before the carriage return character, VPS uses the "minimum line length" value (LMIN). For example, if LMIN is set to 20 each output line of less than 20 characters will be padded with idle characters to make a 20 character line. Both these values may be set at sign-on using a special terminal number or modified after signing on using the /CTL command. The reader is referred to the /CTL command description for further information on CPI and LMIN. Below is a list of terminal numbers, the CPI and LMIN values used, and an example of the type of terminal which uses the terminal number.

<u>Terminal Number</u>	<u>CPI</u>	<u>LMIN</u>	<u>Terminal Type</u>
0-9F	0	0	(default settings)
A0-AF	20	0	TI 700 Portable
CO-CF	9	0	AJ 832
FA-FF	0	36	GE Terminet (at 1200 BAUD)

account  
the account number to be charged with this terminal session.

/LIBRARY

Levels:  
"\*go"

The /LIBRARY command may be used to list information pertaining to VPS library files. The format of the command is written below.

```
/LIBrary [BRIEF  
FULL  
NAME  
RCDS  
RSIZ  
TRKS  
CREATED  
REF  
PUBL  
PERMIT ] [ ,FILE=filename  
,FILE=name*  
,FILE=filename1-filename2  
,FILE=name1*-name2*  
,FILE=(filespec,filespec,...) ]  
  
[ ,INDEX=indexname  
,INDEX=(indexname,indexname,...)  
,INDEX=USERLIB ]  
  
[ ,ACCT=account  
,ACCT=acct*  
,ACCT=account1-account2  
,ACCT=acct1*-acct2*  
,ACCT=(acctspec,acctspec,...) ] [ ,SAVE  
,SAVE=filename  
,RSAV=filename ]
```

first parameter

specifies the type of listing desired. If BRIEF is specified the listing will include: the account(s), the index name, and the number of files found. For each file the listing will include: file name (NAME), number of records in the file (RCDS), the record size (RSIZ), the number of library tracks used (TRKS), and the date and time the file was created (CREATED). If FULL is specified the listing will include all of the above information plus, for each file, the following additional information: date last referenced (REF), public access code (PUBL), and the account range permitted special access (PERMIT). Additionally, a listing keyword (NAME, RCDS, RSIZ, TRKS, CREATED, REF, PUBL, PERMIT) may be specified producing a listing of information up to and including that keyword for each file found. The default is CREATED at the terminal, and FULL in the batch.

FILE=

specifies the file name(s) or range(s) of file names for which information is to be listed. It may be specified as a single file name; a range of file names separated by a dash ("-"), in which case all file names within that range will be listed; a "partial

name" (represented by "name\*" above) followed by an asterisk ("\*") indicating that all file names beginning with the "partial name" should be listed; a range of "partial names" where each "partial name" is followed by an asterisk and the two names are separated by a dash, indicating that a listing should be produced of all the files starting with the first file where name begins with the "partial name" (represented by "name1\*" above) to the last file where name begins with the "partial name" (represented by "name2\*" above); or, finally, a list of any of these above specifications enclosed in parenthesis and separated by commas. The default is all file names falling within the account and index specifications. Note that FILE may be abbreviated as F.

## INDEX=

specifies the index name or list of index names of the files (defined by the FILE keyword) for which information is to be listed. It may be specified as a single index name or a list of index names enclosed in parentheses and separated by commas. The default is USERLIB. Note that INDEX may be abbreviated as I.

## ACCT= (for privileged accounts only)

specifies the account number or range of account numbers of the files to be listed. It may be specified in exactly the same manner as the FILE keyword. The default is the terminal user's account number. Note that ACCT may be abbreviated as A.

## SAVE

specifies that the listing is not to be printed at the terminal. Instead the listing is placed in the temporary file \*2.

## SAVE=

specifies the name of a new file into which the /LIBRARY command will place the listing. If this keyword is specified, the listing will not be printed at the terminal.

## RSAV=

specifies the name of an existing file into which the /LIBRARY command is to place the listing. If this keyword is specified the listing will not be printed at the terminal.

/LIST

Levels:  
"\*go"

The /LIST command may be used to list all or part of a file at the user's terminal. Unlike the /DISPLAY command, /LIST does not print an assigned line number for each line. The format of the /LIST command is written below.

/List	[ num1, 1, LAST, LAST-num3, ]	[ num2, LAST, LAST-num4, ]	[ filename * *2 ]
-------	--	----------------------------------	-------------------------

first parameter

specifies the line number in the file where printing is to begin. The first line in the file being line number 1. The parameter may be specified either as a number (represented by "num1" above) relative to the beginning of the file; as the keyword LAST, indicating that the last line in the file is to be printed; or as the keyword LAST followed by a minus sign ("-") and a number (represented by "num3" above), indicating that the first line to be printed is line number "LAST-num" relative to the end of the file. If this is the only parameter specified, only one line will be listed. Line number 1 is the default.

second parameter

specifies the line number in the file where printing is to end. The parameter may be specified as either a number (represented by "num2" above) relative to the beginning of the file; as the keyword LAST, indicating the printing should continue until the end of the file is reached; or as the keyword LAST followed by minus sign ("-") and a number (represented by "num4" above), indicating that the last line to be printed is the line number "last-num" relative to the end of the file. LAST is the default only if the first parameter is not specified.

filename

specifies the name of the file to be printed. It may be specified as a valid file name or as "\*" or "\*2" indicating the appropriate temporary file. The default is the temporary file "\*".

Levels:

Attention  
 "\*go"

The /LOG command may be used to cause program status information to be printed at the user's terminal at the completion of each program executed. The format of the /LOG command is written below.

```
/LOG  [ ON  
      OFF ]
```

ON

specifies that the system should print program status information after each program or command execution. ON is the default.

OFF

specifies that the system should suspend the /LOG function.

Notes:

Program status information will be printed in the form:

```
*hh:mm:ss  cpu xx.xx  io yyy
```

where -

hh:mm:ss - is the time of day.

xx.xx - is the seconds of CPU time used during the execution of the last program or command.

yyy - is the number of input/output operations performed during the execution of the last program or command.

/MESSAGE (/MSG)

Levels:

Attention  
"\*go"

The /MESSAGE command may be used to send a message to another user, the operator, or yourself. Note that /MSG may be used in place of /MESSAGE. The format of the /MESSAGE command is written below.

/MESsage	{ account OPERator * ALL }	{message}
/MSG		

**first parameter**

specifies the destination of the message to be sent. It may be specified as an account number of another user; OPERATOR, indicating the message is to be sent to the VPS operator; \*, indicating the message is to be printed only at the sending terminal; or ALL, indicating the message is to be sent to all signed-on VPS users (note that ALL can be specified only by a privileged account).

**message**

specifies the message to be sent.

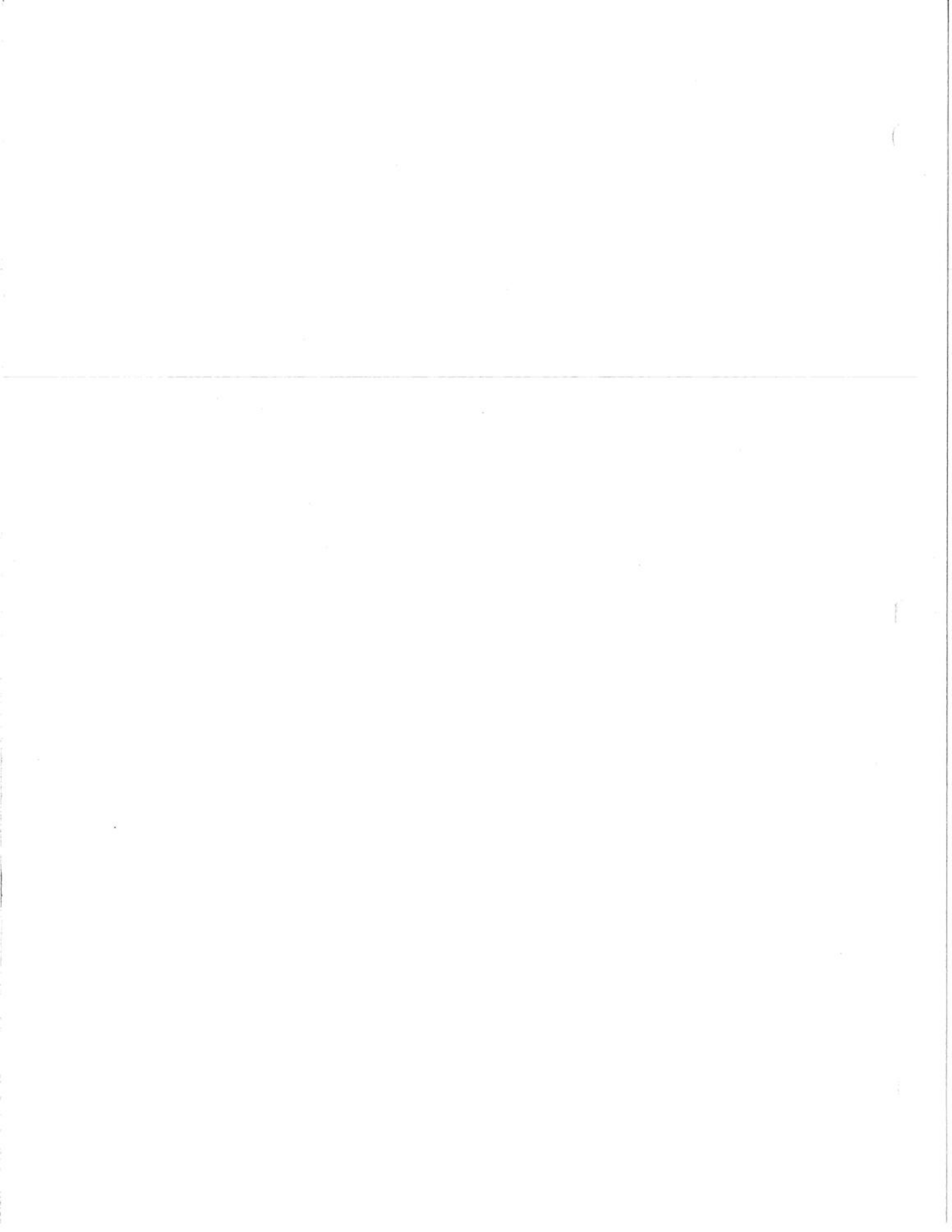


## L R E A D

The LREAD utility is used to print a formatted summary of the contents of an IBM standard labelled tape. It prints the file position, dataset name (dsname), record length (lrecl), block size, record format, creation date and system, and the number of blocks in each file on the tape. The program is invoked by creating a VPS library file containing the following records:

```
/SET VOL=tapenm  
/INC LREAD
```

where "tapenm" is the name (or volume serial) of the tape to be scanned. The library file can either be executed on the terminal or sent to batch (see the /BQx command) for processing. In the latter case, the report will be printed on the high-speed printer.



Levels:  
"\*go"

The /NEWS command may be used to print a listing of the Computing Center news file which is in reverse chronological sequence and includes documentation on recent additions and changes to VPS as well as information of general interest. The format of the /NEWS command is written below.

```
| /NEWS |
```

/OFF

Levels:

Attention  
"\*go"

The /OFF command is used to terminate a terminal session and sign the user off the system. The format of the /OFF command is written below.

```
/OFF [ Drop  
      Hold ]
```

DROP (meaningful for dial-up terminals only)  
specifies that the telephone connection is to be disconnected after the terminal session is terminated. DROP is the default.

HOLD (meaningful for dial-up terminals only)  
specifies that the telephone connection is to be preserved after the terminal session has terminated so that a new sign-on can be entered without redialing.

Notes:

The /OFF command may be entered at either "\*go" or attention level. Attention level is reached by striking the attention, or break key during program execution. If /OFF is entered at the attention level, it will have the same effect as entering the /CANCEL command followed at "\*go" by a /OFF command.

Levels:

Attention

"\*go"

The /PER command makes the Program Event Recording facility of System/370 available for program execution tracing. For a complete description of program tracing and interactive debugging, the reader is referred to Chapter 8. The format of the /PER command is written below.

```

| /PER [ON ] [INSt [a1 [a2]]] [STore [a1 [a2]]]
|      [OFF] [NOINSt] [NOSTore]
|      [Clear]
|
|      [BRanch] [GR [r1 ... [r16] ] ] [RUn] [OR]
|      [NOBRanch] [NOGR] [NORUn] [ANd]
|

```

## ON

specifies that the PER facility is to be enabled and all current tracing specifications will be in effect. Note that ON is implied by INST, STORE, BRANCH, and GR.

## OFF

specifies that the PER facility is to be disabled and the current tracing specifications preserved - tracing is halted.

## CLEAR

specifies that the current tracing specifications are to be cleared.

## INST

specifies that instruction tracing is to be performed within a specified address range, where the parameters "a1" and "a2" are two hexadecimal storage addresses indicating the start and end of the address range. That is, execution of a problem program instruction residing in the specified area of storage will cause a trace message to be printed at the terminal. If a1 and a2 are not specified, the address range defaults to the entire user region. If only a1 is specified, a2 is set equal to a1 and only the instruction beginning at address a1 will produce the trace message. If a1 and a2 are both specified, all instructions within that range will be traced (if a2 is smaller than a1, only instructions outside the range a2 to a1 will be traced).

/PER

NOINST

specifies that instruction tracing is to be turned off.

STORE

specifies that the PER facility should monitor the specified address range for modification of data. Any instruction causing such a modification will produce a trace message. The address range specification follows the same rules as that for INST. Note that between INST and STORE only one address range may be active at any time, although both INST and STORE may be simultaneously in use.

NOSTORE

specifies that monitoring for data modification should be turned off.

BRANCH

specifies that a trace message is to be produced each time a successful branch is taken in the executing program.

NOBRANCH

specifies that trace messages produced by a successful branch in the executing program should be turned off.

GR

specifies that a trace message is to be produced whenever execution of a problem program instruction modifies the contents of the indicated general purpose register(s). It may be specified as the keyword followed by a decimal number or a list of decimal numbers separated by blanks indicating the register(s) to be monitored.

NOGR

specifies that trace messages produced by the modification of the general purpose register(s) specified in a previous GR parameter are to be turned off.

RUN

specifies that the program should continue to execute after each trace message is printed.

NORUN

specifies that DEBUG mode should be entered after each trace message is printed (see Chapter 8).

OR

specifies that if one or more of the stop conditions are met debug mode will be entered. OR is the default.

AND

specifies that all of the stop conditions must be met simultaneously for debug mode to be entered. Thus, for example, one might trace only branch instructions (BRANCH) within a specific range of instruction addresses (INST) which change a particular general purpose register (GR).

Notes:

The /PER command may be entered either at "\*go" or attention level. Attention level is reached by striking the attention, or break, key during program execution. When issued at "\*go" level, any settings apply to the next program executed, while settings issued at attention level apply to the currently executing program. All settings are turned off when the terminal returns to "\*go" level after the program terminates.

The format of the PER trace message is as follows:

```
aaaaaa mmmm hhhh ->bbbbbb GR STORE
```

aaaaaa - is the address of the instruction (always appears).  
 mmmm - is the assembler mnemonic for the instruction (always appears).  
 hhhh - is the image of the instruction in hexadecimal (always appears).  
 bbbbb - is the target address for a successful branch - appears only if BRANCH is on and the trace message was caused by a successful branch.  
 GR - specifies that GR tracing is on and this message was produced when the indicated instruction modified one of the monitored general purpose registers.  
 STORE - specifies that STORE tracing is on and this message was produced when the indicated instruction modified the monitored area.

/POST

Levels:

Attention

The /POST command may be used during program execution to communicate with the running program. Note that the user must make provisions in the program prior to using the command (Assembler users are referred to the description of the PSTCOD macro in the VPS System Facilities for Assembler Programmers Manual for more information). The format of the /POST command is written below.

```
| /POST {pstcod} |
```

pstcod

specifies a 1 to 8 character post code which is to be passed to the executing program.

Notes:

The /POST command can be used only at attention level. Attention level is reached by striking the attention, or break, key during program execution.



Levels:  
"\*go"

The /PROFILE command may be used to invoke the VPS program to interactively inspect the user's profile. The program may be used to set or change profile information only if that privilege was activated when the account was created. See Chapter 7 (page 73) for a description of the commands which may be used after the profile program has been invoked. The format of the /PROFILE command is written below.

```
| /PROFile |
```

/PURGE

Levels:  
"\*go"

The /PURGE command may be used to delete library files from the VPS library. The format of the /PURGE command is written below.

```
| /PURge filename[(acct)],filename[(acct)],... |
```

filename

specifies the name of an existing VPS library file which the terminal user wishes to erase. Multiple files may be erased by specifying a list of file names separated by commas. If the file exists under an account different from the user's account, the file name must be immediately followed (no intervening blanks) by the owning account number enclosed in parentheses (this requires a privilege).

Levels:

Attention  
 "\*go"

The /QUERY command may be used to obtain various types of information about the current environment of the user's terminal session. The format of the /QUERY command is written below.

```

| /Query { Names
|         Per
|         Debug
|         Reply   Sent
|         Reply   Rcvd
|         Ctl
|         Batch   [nnnn]
|         PF
| }
  
```

## NAMES

specifies that all account numbers currently signed-on and their physical line addresses should be listed at the terminal. Note that the occurrence of DSC in the listing indicates a disconnected user.

## PER

specifies that the current PER settings (see Chapter 8) should be listed at the terminal.

## DEBUG

specifies that the current Debug settings (see Chapter 8) should be listed at the terminal.

## REPLY

specifies that information on messages requiring replies which have been sent or received by the user's terminal during this terminal session should be listed. If SENT is specified, all messages sent from this terminal will be listed along with their message numbers. If RCVD is specified, all messages received by this terminal will be listed along with their message numbers. The default is RCVD.

## CTL

specifies that the current terminal control functions (see the /CTL command description) should be listed at the terminal.

/QUERY

BATCH

specifies that information on jobs executing in the VPS batch is to be listed. Alternatively, a batch job number ("nnnn") may be specified in which case only information about that executing job will be displayed.

PF

(for 3270 display terminals only)

specifies that the current setting of all the program function keys is to be displayed on the user's terminal.

Notes:

The /QUERY command may be entered at either "\*go" or attention level. Attention level is reached by striking the attention, or break, key during program execution.

Levels:  
"\*go"

The /RATES command may be used to list the current Computing Center charging rates for VPS terminal and batch usage. The format of the /RATES command is written below.

```
/RATES
```

## /RENAME

Levels:  
"\*go"

The /RENAME command may be used to rename and/or to change the access attributes of an existing library file. The format of the /RENAME command is written below.

```
/REName {oldfilename[(acct)],} [newfilename[(acct)]]  
        [,accattr,accattr,...]
```

### oldfilename

specifies the name of an existing VPS library file whose name and/or access attributes the user wishes to change. If the file exists under an account different from the user's account, the file name must be immediately followed (no intervening blanks) by the owning account number enclosed in parentheses (this requires a privilege).

### newfilename

specifies the new name the user wishes the file to be given. If the newly named file is to be charged to an account other than the account it currently exists under, the new name must be immediately followed (no intervening blanks) by the new account number enclosed in parentheses (this requires a privilege unless it is the user's account). Note that this parameter should be omitted when only the access attributes are to be modified, but the separating comma must always be present (see the example below). An alternative form of this parameter allows an equal sign ("=") to be specified in place of the index portion (if present) and/or the name portion of the new file name. An equal sign indicates that that portion of the old file name is to be used for that portion of the new file name.

### accattr

specifies an access attribute the user wishes to change. It may be specified as a single attribute or a list of attributes separated by commas. The access attributes permitted are:

- READ (R) - specifies that the file may be read or copied by other accounts.
- NOREAD (NOR) - specifies that the file may not be read or copied by other accounts.
- EXEC (E) - specifies that the file may be the argument of the /EXEC command (the file may be executed) by other accounts.
- NOEXEC (NOE) - specifies that the file may not be the argument of the /EXEC command (the file may not be executed) by other

accounts.

PURGE (P) - specifies that the file may be updated, edited, replaced, or purged by other accounts.

NOPURGE (NOP) - specifies that the file may not be updated, edited, replaced, or purged by other accounts.

P=(A=acctrange,acattr) - specifies the account or account range permitted special access to the file and the access attributes allowed for these account(s). 'acctrange' may be specified as a single account; a range of accounts separated by a dash ("-"), in which case all accounts within that range will be given the special access attributes; a "partial account" (represented by a partial account number followed by an asterisk ("\*")), indicating that all accounts beginning with the "partial account" should be given the special access attributes; or, finally, a range of "partial accounts" separated by a dash, indicating that all accounts starting with the first account beginning with the first "partial account" to the last account beginning with the second "partial account" should be given the special access attributes. 'acattr' is any of the access attributes listed above.

P=(F=index.filename,acattr) - specifies the full name (index name and file name) of the program which is permitted special access to the file during its execution and the type of access allowed. 'acattr' is any of the access attributes listed above.

Alternatively, the following access attributes may also be specified:

PUBL - which corresponds to "READ" and "EXEC" access.

PRIV - which corresponds to "NOREAD" and "NOEXEC" access.

XO - which corresponds to "NOREAD" and "EXEC" access.

NOXO - which corresponds to "NOEXEC" access.

#### Examples:

In the following example the file TEST is renamed PROD:

```
*GO
/rename test,prod
.
.
.
```

In the following example the file EXAM1, whose access attributes are currently EXEC, READ, and NOPURGE, will be changed. All accounts beginning with the string "MA1" through all the accounts beginning with the string "MA4" will be given read access, all other account will be given NOEXEC, NOREAD, and NOPURGE access to the file.

/RENAME

\*GO

/rename exam1,,priv,p=(a=mal\*-ma4\*,r)

.  
.  
.



Levels:

"\*go"

The /REPLACE command may be used to replace an existing library file with the current contents of the temporary "\*" file and, if desired, modify the access attributes of the replaced file. The format of the /REPLACE command is written below.

```
| /REplace {filename[(acct)]} [,acattr,acattr,...] |
```

**filename**

specifies the name of the library file to be replaced with the temporary "\*" file. If the file exists under an account different from the user's account, the file name must be immediately followed (no intervening blanks) by the owning account number enclosed in parentheses (this requires a privilege).

**acattr**

specifies an access attribute the user wishes to change. It may be specified as a single attribute or a list of attributes separated by commas. The access attributes permitted are:

- READ (R) - specifies that the file may be read or copied by other accounts.
- NOREAD (NOR) - specifies that the file may not be read or copied by other accounts.
- EXEC (E) - specifies that the file may be the argument of the /EXEC command (the file may be executed) by other accounts.
- NOEXEC (NOE) - specifies that the file may not be the argument of the /EXEC command (the file may not be executed) by other accounts.
- PURGE (P) - specifies that the file may be updated, edited, replaced, or purged by other accounts.
- NOPURGE (NOP) - specifies that the file may not be updated, edited, replaced, or purged by other accounts.
- P=(A=acctrange,acattr) - specifies the account or account range permitted special access to the file and the access attributes allowed for these account(s). 'acctrange' may be specified as a single account; a range of accounts separated by a dash ("-"), in which case all accounts within that range will be given the special access attributes; a "partial account" (represented by a partial account number followed by an asterisk ("\*")), indicating that all accounts beginning with the "partial account" should be given the special access attributes; or, finally, a range of "partial accounts" separated by a dash, indicating that all accounts starting with the first account beginning with the first "partial account" to the last account beginning with the

/REPLACE

second "partial account" should be given the special access attributes. 'accattr' is any of the access attributes listed above.

P=(F=index.filename,accattr) - specifies the full name (index name and file name) of the program which is permitted special access to the file during its execution and the type of access allowed. 'accattr' is any of the access attributes listed above.

Alternatively, the following access attributes may also be specified:

- PUBL - which corresponds to "READ" and "EXEC" access.
- PRIV - which corresponds to "NOREAD" and "NOEXEC" access.
- XO - which corresponds to "NOREAD" and "EXEC" access.
- NOXO - which corresponds to "NOEXEC" access.

Levels:

Attention  
 "\*go"

The /REPLY command is used to reply to a message from an executing program which requires a response. This message may be generated by a program executing at the user's terminal, at another user's terminal, or in the VPS batch. The user must make provisions in the program prior to its execution (see the VPS System Facilities for Assembler Programmers Manual for a description of how these messages are generated). The /QUERY command may be used to list the text of messages to which the user has not yet responded. Note that /Y may be used instead of /REPLY. The format of the /REPLY command is written below.

```

/REPLY {repid} {reply}
/Y

```

repid

specifies the reply id number of the message to which the user is responding.

reply

specifies the reply to the message.

Notes:

The /REPLY command can be used either at "\*go" level or at attention level. Attention level is reached by striking the attention, or break, key during program execution.

Example:

In the following example a program in the file TRANSACT is executed. At the end of the first phase of processing the program asks if the user wishes the program to continue. The user replies with 'NO' and execution is terminated.

```

*GO
transact
<01> END PHASE 1, TO CONTINUE REPLY YES, OTHERWISE REPLY NO
!                                     (attention level entered)
:/reply 1 no
.
.
.

```

/RO

Levels:

Attention

The /RO command may be used during program execution to resume printing of program output halted by the /CO command (see the description of the /CO command for further information). The format of the /RO command is written below.

```
| /RO |
```

notes:

The /RO command can be used only at attention level. Attention level is reached by striking the attention, or break, key during program execution.

Levels:  
 "\*go"

The /RSAVE command may be used to replace an existing library file with the current contents of the temporary "\*2" file and, if desired, modify the access attributes of the replaced file. The format of the /RSAVE command is written below.

```
/RSAVE {filename[(acct)]} [,acattr,acattr,...]
```

#### filename

specifies the name of the library file to be replaced with the temporary "\*2" file. If the file exists under an account different from the user's account, the file name must be immediately followed (no intervening blanks) by the owning account number enclosed in parentheses (this requires a privilege).

#### acattr

specifies an access attribute the user wishes to change. It may be specified as a single attribute or a list of attributes separated by commas. The access attributes permitted are:

- READ (R) - specifies that the file may be read or copied by other accounts.
- NOREAD (NOR) - specifies that the file may not be read or copied by other accounts.
- EXEC (E) - specifies that the file may be the argument of the /EXEC command (the file may be executed) by other accounts.
- NOEXEC (NOE) - specifies that the file may not be the argument of the /EXEC command (the file may not be executed) by other accounts.
- PURGE (P) - specifies that the file may be updated, edited, replaced, or purged by other accounts.
- NOPURGE (NOP) - specifies that the file may not be updated, edited, replaced, or purged by other accounts.
- P=(A=acctrange,acattr) - specifies the account or account range permitted special access to the file and the access attributes allowed for these account(s). 'acctrange' may be specified as a single account; a range of accounts separated by a dash ("-"), in which case all accounts within that range will be given the special access attributes; a "partial account" (represented by a partial account number followed by an asterisk ("\*")), indicating that all accounts beginning with the "partial account" should be given the special access attributes; or, finally, a range of "partial accounts" separated by a dash, indicating that all accounts starting with the first account beginning with the first "partial account" to the last account beginning with the

second "partial account" should be given the special access attributes. 'accattr' is any of the access attributes listed above.

P=(F=index.filename,accattr) - specifies the full name (index name and file name) of the program which is permitted special access to the file during its execution and the type of access allowed. 'accattr' is any of the access attributes listed above.

Alternatively, the following access attributes may also be specified:

- PUBL - which corresponds to "READ" and "EXEC" access.
- PRIV - which corresponds to "NOREAD" and "NOEXEC" access.
- XO - which corresponds to "NOREAD" and "EXEC" access.
- NOXO - which corresponds to "NOEXEC" access.

Levels:

"\*go"

The /SAVE command is used to save the contents of the temporary "\*" or "\*2" file by either creating a new library file and placing the contents of the designated temporary file in it or by replacing an existing library file with the contents of the designated temporary file. At the same time the user may specify access attributes for the new or replaced file. The format of the /SAVE command is written below.

```

/SAVE {filename[(acct)]} [,SV] [,REPL]
      [,accattr,accattr,...]

```

**filename**

specifies the name of the file to be either created or replaced depending upon whether the REPL keyword is present. If this is a file to be replaced and the file exists under an account different from the user's account, the file name must be immediately followed (no intervening blanks) by the owning account number enclosed in parentheses (this requires a privilege).

**SV**

is written as shown and indicates that the contents of the temporary "\*2" file are to be saved. If this keyword is not entered, the contents of the temporary "\*" file are saved.

**REPL**

is written as shown and indicates that the name specified in the 'filename' parameter is an existing file and this is a replace operation. If this keyword is not entered, the system will attempt to create a new file with a name corresponding to the 'filename' parameter.

**accattr**

specifies an access attribute the user wishes to change. It may be specified as a single attribute or a list of attributes separated by commas. The access attributes permitted are:

- READ (R) - specifies that the file may be read or copied by other accounts.
- NOREAD (NOR) - specifies that the file may not be read or copied by others.
- EXEC (E) - specifies that the file may be the argument of the

- /EXEC command (the file may be executed) by other accounts.
- NOEXEC (NOE) - specifies that the file may not be the argument of the /EXEC command (the file may not be executed) by other accounts.
- PURGE (P) - specifies that the file may be updated, edited, replaced, or purged by other accounts.
- NOPURGE (NOP) - specifies that the file may not be updated, edited, replaced, or purged by other accounts.
- P=(A=acctrange,acctr) - specifies the account or account range permitted special access to the file and the access attributes allowed for these account(s). 'acctrange' may be specified as a single account; a range of accounts separated by a dash ("-"), in which case all accounts within that range will be given the special access attributes; a "partial account" (represented by a partial account number followed by an asterisk ("\*")), indicating that all accounts beginning with the "partial account" should be given the special access attributes; or, finally, a range of "partial accounts" separated by a dash, indicating that all accounts starting with the first account beginning with the first "partial account" to the last account beginning with the second "partial account" should be given the special access attributes. 'acctr' is any of the access attributes listed above.
- P=(F=index.filename,acctr) - specifies the full name (index name and file name) of the program which is permitted special access to the file during its execution and the type of access allowed. 'acctr' is any of the access attributes listed above.

Alternatively, the following access attributes may also be specified:

- PUBL - which corresponds to "READ" and "EXEC" access.
- PRIV - which corresponds to "NOREAD" and "NOEXEC" access.
- XO - which corresponds to "NOREAD" and "EXEC" access.
- NOXO - which corresponds to "NOEXEC" access.



Levels:  
 "\*go"

The /SCRIPT command invokes the VPS program that allows the user to print English language text in a form suitable for term papers, theses, publications, resumes, etc. /SCRIPT will perform such operations as automatic page numbering, left and right margin control, top and bottom margin control, underlining, and margin justification. The text must first be typed on a terminal and saved in the VPS library. The reader is referred to the VPS Utilities Manual for information on the preparation of the text file. The format of the /SCRIPT command is written below.

```
/SCRIPT {filename} [,option,option,....]
```

**filename**

specifies the name of the library file the user wishes to have formatted and printed.

**option**

specifies a /SCRIPT control option. It may be specified as a single option or as a list of options separated by commas. The following is a list of the control options available:

**C**enter - specifies that the listing should be centered on l32 position printout paper.

**N**owait - specifies that the printing of the text should begin immediately without pausing for the user to adjust the paper.

**N**umber - specifies that the name of the text file and the line number of each line should be listed next to the left margin.

**P**AGExxx - specifies that scripting should begin at page "xxx". Note "xxx" must be entered as three numeric digits.

**S**ingle - specifies that the listing should terminate after one page is printed.

**S**top - specifies that the listing of the text should stop after each page is printed. This allows the insertion of standard typewriter paper if desired.

**U**nformatted - specifies that the file should be listed without formatting.

**T**ranslate - specifies that character translation should occur using a translation table that may be present as part of the text file (see the VPS Utilities Manual for further information).

**2**PASS - specifies that the /SCRIPT program should make one pass through the text file performing all indicated formatting without producing any output. A second pass is then made,

/SCRIPT

during which output is produced. This option is primarily useful for handling a text file which uses reference names defined later in the file (see the VPS Utilities Manual for further information).

Offline - specifies that though the program is being run at the terminal the script output is being printed offline (i.e. there is no need to position the paper, etc.). See the VPS Utilities Manual for further information.

Levels:

Attention  
"\*go"

The /SCROLL command may be used to cause VPS to pause and issue an attention read after every "n" lines of output are printed. The format of the /SCROLL command is written below.

```
/SCroll {number}
        {OFF}
```

number

specifies the number of lines the user wishes to have printed between each pause.

OFF

specifies that normal output processing should resume and no pauses are to occur in output printing.

Notes:

The /SCROLL command can be used either at "\*go" level or at attention level. Attention level is reached by striking the attention, or break, key during program execution.

/SKIP

Levels:  
Attention

The /SKIP command may be used during program execution to suppress the printing of output. Output printing will resume when one of the following events occurs: the skip parameter has been satisfied, the program issues a conversational read, or the program terminates. The format of the /SKIP command is written below.

```
/SKip {number  
      /string/}
```

number

specifies the number of output lines to be skipped.

/string/

specifies a character string which is contained in the next output line the user wishes to have printed. In other words, the system will suppress the printing of output until a line is found containing that string. Note that the character string must be immediately preceded and followed by a special character, not appearing within the string, indicating the beginning and end of the character string (any non-alphameric character not appearing in the string can be used).

Notes:

The /SKIP command may be used only at attention level. Attention level is reached by striking the attention, or break, key during program execution.

Levels:  
"\*go"

The /SLEEP command may be used to lock the terminal keyboard until the attention, or break, key is struck. The format of the /SLEEP command is written below.

```
| /SLleep |
```

Notes:

The /SLEEP command may be used to cause messages (if sent by another user or the VPS operator) to be printed on the terminal eventhough the signed-on terminal is inactive.

/SORT

Levels:  
"\*go"

The /SORT command may be used to sort library files into ascending or descending sequence using one to ten sort fields and producing a single sorted output file. See the VPS Utilities manual for a complete description of the commands which may be entered after the sort program has been invoked. The format of the /SORT command is written below.

```
| /SORT filename,filename,... |
```

**filename**

specifies the library file(s) to be sorted. It may be specified as a single file name or as a list of file names separated by commas.

Levels:

Attention  
"\*go"

The /STATUS command may be used to obtain accumulated terminal session statistics. Information printed will include: the current time and date, the CPU time in seconds since sign-on, the number of I/O operations since sign-on, the hookup time since sign-on, and the current session cost. The format of the /STATUS command is written below.

```
| /STATUS |
```

Notes:

The /STATUS command can be used either at "\*go" level or at attention level. Attention level is reached by striking the attention, or break, key.

/SUMMARIZE

Levels:  
"\*go"

The /SUMMARIZE command may be used to list all lines of a file which have a slash "/" in column 1. The file line number is prefixed to each line as it is listed. The total number of lines in the file is also printed. The format of the /SUMMARIZE command is written below.

```
| /SUMmarize {filename} |
```

filename

specifies the library file the user wishes to summarize.



Levels:

Attention  
 "\*go"

The /TABIN command is used to set, change, or turn off the terminal input tabs. After setting or changing the input tab locations, each time the user presses the tab key while entering input, VPS will accept the next character typed by the user as if it were in the column position corresponding to the next input tab location. This holds true regardless of the physical tab settings on the terminal. The format of the /TABIN command is written below.

```
|
| /TABIn [ column column ... ]
|        Asm
|        COBol
|        Fort
|        Pli
|        Spss
|        Every n
|        Off
|
```

**column**

specifies the column number(s) at which input tabs are to be set. It may be specified as a single column number or as a list of column numbers separated by blanks. Up to 11 column numbers may be specified and they must be entered in ascending order.

**ASM**

specifies that the input tabs should be set at columns 10, 16, 35, and 72.

**COBOL**

specifies that the input tabs should be set at columns 8, 12, 16, 20, 24, 28, 32, ... ,72.

**FORT**

specifies that the input tabs should be set at columns 7 and 73.

**PLI**

specifies that the input tabs should be set at columns 5, 10, 15, 20, 25, 30, 35, 40, 45, etc. - equivalent to specifying EVERY 5.

**SPSS**

specifies that the input tabs should be set at column 16.

/TABIN

EVERY

specifies that an input tab should be set at every "n" columns. For example, if EVERY 3 is specified, tabs would be set at columns 3, 6, 9, 12, etc.

OFF

specifies that the current input tabs should be turned off.

Notes:

The /TABIN command may be used at either "\*go" or attention level. Attention level is reached by striking the attention, or break, key.

Levels:

Attention  
 "\*go"

The /TABOUT command is used to set, change, or turn off the terminal output tabs. After setting or changing the output tab locations, VPS will automatically use the physical tabs on the terminal to save time while printing. Issuing the /TABOUT command does not set the physical tabs on the terminal. Therefore, the user must set the physical tabs to the output tab settings for output line formats to be correct. The format of the /TABOUT command is written below.

```

/TABOut [column column ... ]
        Asm
        COBo1
        Fort
        Pli
        Spss
        EVERY n
        Off
  
```

column

specifies the column number(s) at which output tabs are to be set. It may be specified as a single column number or as a list of column numbers separated by blanks. Up to 11 column numbers may be specified and they must be entered in ascending order.

## ASM

specifies that the output tabs should be set at columns 10, 16, 35, and 72.

## COBOL

specifies that the output tabs should be set at columns 8, 12, 16, 20, 24, 28, 32, ... ,72.

## FORT

specifies that the output tabs should be set at columns 7 and 73.

## PLI

specifies that the output tabs should be set at columns 5, 10, 15, 20, 25, 30, 35, 40, 45, etc. - equivalent to specifying EVERY 5.

/TABOUT

SPSS

specifies that the output tabs should be set at column 16.

EVERY

specifies that an output tab should be set at every "n" columns. For example, if EVERY 3 is specified, output tabs would be set at columns 3, 6, 9, 12, etc.

OFF

specifies that the current output tabs should be turned off.

Notes:

The /TABOUT command may be used at either "\*go" or attention level. Attention level is reached by striking the attention, or break, key.

Levels:

Attention  
 "\*go"

The /TABS command may be used to set, change, or turn off the input and output terminal tab settings simultaneously. The user should refer to the description of the /TABIN and /TABOUT commands for further information on input and output settings. The format of the /TABS command is written below.

```

/TABS [column column ... ]
      Asm
      COBo1
      Fort
      Pli
      Spss
      Every n
      Off
  
```

**column**

specifies the column number(s) at which input and output tabs are to be set. It may be specified as a single column number or as a list of column numbers separated by blanks. Up to 11 column numbers may be specified and they must be entered in ascending order.

**ASM**

specifies that the input and output tabs should be set at columns 10, 16, 35, and 72.

**COBOL**

specifies that the input and output tabs should be set at columns 8, 12, 16, 20, 24, 28, 32, ... ,72.

**FORT**

specifies that the input and output tabs should be set at columns 7 and 73.

**PLI**

specifies that the input and output tabs should be set at columns 5, 10, 15, 20, 25, 30, 35, 40, 45, etc. - equivalent to specifying EVERY 5.

/TABS

SPSS

specifies that the input and output tabs should be set at column 16.

EVERY

specifies that an input and output tab should be set at every "n" columns. For example, if EACH 3 is specified, tabs would be set at columns 3, 6, 9, 12, etc.

OFF

specifies that the current input and output tabs should be turned off.

Notes:

The /TABS command may be used at either "\*go" or attention level. Attention level is reached by striking the attention, or break, key.

Levels:  
 "\*go"

The /TABSET command may be used to assist the user in setting the physical tabs to correspond to a previous /TABIN, /TABOUT, or /TABS command. The /TABSET command invokes an interactive program which will direct the user through the necessary steps in setting the physical tabs. The format of the /TABSET command is written below.

```

/TABSET  [ INPUT
          OUTPUT ]
```

**INPUT**

specifies that the physical tabs are to be set to the current input tab locations.

**OUTPUT**

specifies that the physical tabs are to be set to the current output tab settings. OUTPUT is the default.

Notes:

After the user has entered the /TABSET command the system will respond with the current input and output tab settings, and then instruct the user to set the left margin to column 1 and clear the current physical tabs. /TABSET will then type a single line consisting of periods and the letter "t" wherever the user has defined a tab position. Each time the letter "t" is printed, the terminal pauses for several seconds, allowing the user to press the tabset key. When all the tabs have been set the program will print a second line using the tab character to verify the settings.

Example:

In the following example, the user sets the input and output tabs using the /TABS command to columns 10, 20, and 30. The /TABSET command is then used to set the physical tabs.

/TABSET

```
*GO
/tabs 10 20 30
*OK
/tabset
```

INPUT TABS - 10 20 30

OUTPUT TABS - 10 20 30

SET LEFT MARGIN TO COLUMN 1; THEN HIT RETURN.

CLEAR ALL PHYSICAL TABS; THEN HIT RETURN.

```
.....T.....T.....T
      T          T          T
*GO
```



Levels:  
 "g0"

The /TAPE command is used to allow paper tapes to be read by ASCII terminals. The format of the /TAPE command is written below.

```
/TAPE { CR
      { OFF }
```

#### CR

specifies that a paper tape is about to be read. VPS will then access the terminal in the following manner:

- when VPS is ready to accept another record it will send an XON (DC1) to the terminal, starting the paper tape reader.
- when VPS receives an End-of-Line character from the terminal it will send an XOFF (DC3), stopping the paper tape reader.

#### OFF

specifies that paper tape handling should be turned off and VPS should access the terminal normally.

#### Notes:

To successfully read paper tapes the user should be aware of a potential problem. If the End-of-Line character is anything other than an XOFF (e.g. carriage return), the tape reader will not stop when an End-of-Line character is sent. This will cause loss of data. The problem can be bypassed in one of the following ways (these remarks apply particularly to the vagaries of the venerable Model 33 Teletype. Precisely what is required for another type of terminal must be determined from a consideration of that terminal's properties):

- When the paper tape is created at least 4 pad characters (NULs) should be written after every End-of-Line character and the terminal should be switched to full duplex when the tape is read. The padding allows VPS to send the XOFF to stop the reader before there is any loss of data.
- If a carriage return character is used on the tape as the End-of-Line character and no padding follows it, the terminal should be wired so that carriage return stops the reader.
- The XOFF character should be used as the End-of-Line character when creating the tape.

/TEXT

Levels:

Attention  
"\*go"

The /TEXT command is used to modify terminal input and output translation. To understand the function of the /TEXT command, it is necessary to be familiar with the way in which VPS handles terminal input and output. Under normal operation when a user enters a line of input, that input is translated (all upper case characters are changed to lower case EBCDIC and all lower case characters are changed to upper case EBCDIC). If the input line is being placed in a file, say with the VPS Editor, the translated version of the line is used. Conversely, when a line of output is printed at the terminal it is translated before printing (upper case is changed to lower case and lower case is changed to upper case). The format of the /TEXT command is written below.

/Text { Full Upper }
-------------------------

**FULL**

specifies normal VPS terminal input and output handling. Both input and output will be translated in the manner described above.

**UPPER**

specifies that all input is to be translated to upper case EBCDIC (if the input is to be placed in a file - it will be stored as upper case EBCDIC), and all output to the terminal is to be translated to upper case.

Notes:

This command is normally used for ASCII terminals which have the lower case feature, since when signing-on to an ASCII terminal VPS will default to UPPER. The user may inform the system that lower case is supported by specifying FULL on the /TEXT command.

The /TEXT command may be entered at "\*go" level or at attention level. Attention level is reached by striking the attention, or break, key.

Levels:

Attention  
"\*go"

The /USERS command may be used to obtain the number of users currently signed-on to VPS. The format of the /USERS command is written below.

```
| /USers |
```

Notes:

The /USERS command may be used at either "\*go" level or attention level. Attention level is reached by striking the attention, or break, key during program execution.

/VERIFY

Levels:  
    "\*go"

The /VERIFY command will cause the system to request that the user's password be entered. If the password is not entered correctly in three attempts, the user will be signed-off by the system. This command provides a mechanism which allows the user to place the terminal in a "safe" condition so that it may be left unattended. The format of the /VERIFY command is written below.

```
| /VERify |
```

Levels:  
 "\*go"

The /XEDIT command may be used to initiate the general purpose VPS file manipulation program which allows the user to change or create files (see the VPS Utilities Manual for more extensive information). This command should be used for very large files (see below). The format of the /XEDIT command is written below.

```
/XEdit [filename] [,LRecl=number]
      *
      *2
```

filename

specifies the name of an existing file which is to be changed. The file name parameter may be either a valid file name or a "\*" or "\*2" indicating the appropriate temporary file. If the /XEDIT command is entered with no file name the command assumes a new file is being created.

LRECL=

specifies that the maximum line length of the file being changed or of the file being created is to be set to the value indicated. If LRECL is not specified and a new file is being created, the maximum line length will default to 80. If LRECL is not specified and an old file is being changed the maximum line length will be taken from the old file. If an old file is specified and an LRECL is coded which is smaller than the maximum line length, the file will be truncated. The smallest LRECL value which can be specified is 20, the largest is 255.

Notes:

The /XEDIT command invokes the same program as does the /EDIT command but uses a larger workspace. The largest file that can be edited is about 27,000 records if the maximum line length is 80 bytes (for files with a maximum line length different from 80 bytes the maximum record number will vary accordingly).

/?

Levels:

Attention  
"\*go"

The /? command may be used to cause program status information to be printed at the user terminal. The format of the /? command is written below.

```
| /? |
```

Notes:

Program status information will be printed in the form:

\*hh:mm:ss cpu xx.xx io yyy users nn

where -

hh:mm:ss - is the time of day.

xx.xx - is the accumulated seconds of CPU time used during the execution of the current program.

yyy - is the accumulated number of input/output operations performed during the execution of the program.

nn - is the total number of users currently signed-on.

The /? command may be used at either "\*go" or attention level. Attention level is reached by striking the attention, or break, key during program execution. At attention level ? may be used instead of /?.

&SYSACCT  
     definition 31  
     example 32,33  
 &SYSBAT  
     definition 32  
     example 32  
 &SYSDATE  
     definition 32  
     example 32  
 &SYSPID definition 32  
 &SYSTEM definition 32  
 &SYSTIME  
     definition 32  
     example 32  
  
 \* file  
     description 37  
     replacing with /REPLACE  
         143  
     saving with /SAVE 149  
  
 \*go  
     getting there 7  
     what next 91  
  
 \*2 file  
     default output unit for 23  
     description 37  
     replacing with /RSAVE 147  
     saving with /SAVE 149  
  
 /? definition 172  
 /ACCT definition 95  
 /ATTN definition 96  
 /BLIP definition 97  
 /BQA definition 98  
 /BQB definition 98  
 /BQC  
     definition 98  
     example 99  
 /BQZ  
     definition 98  
     use 87  
 /CANCEL definition 100  
 /CATL definition 101  
 /CE definition 102  
 /CLIST  
     definition 103  
     example 104  
 /CO definition 105  
 /CTL definition 106  
 /DAILY definition 111  
 /DEBUG  
     definition 112  
     use of 79  
 /DFLT statement  
     definition 30  
     example 30,31,33  
 /DISPLAY definition 114  
 /DS definition 115  
 /DSPURGE definition 117  
 /DSRENAME definition 118  
 /EDIT definition 119  
 /END statement description  
     88  
 /EXEC  
     definition 120  
     example 120  
 /FILE statement  
     DISK 50  
     DUMMY 71  
     LIBIN 37  
     LIBOUT 41  
     PRINTER 67  
     PUNCH 69  
     READER 66  
     TAPE 57  
     TERMIN 63  
     TERMOUT 64  
     description 20  
     required and optional  
         fields 22  
     unit types 19  
 /ID  
     definition 122  
     example 7  
 /ID statement description 87  
 /INCLUDE statement  
     description 39  
 /JOB statement description  
     28  
 /LIBRARY  
     definition 124  
     use on batch 89  
 /LIST definition 126  
 /LOAD statement description  
     25  
 /LOG definition 127  
 /MESSAGE definition 128  
 /MSG definition 128  
 /NEWS definition 129  
 /OFF definition 130  
 /PER  
     definition 131  
     use of 79  
 /POST definition 134  
 /PROFILE  
     definition 135  
     example 77  
     use 74-77  
 /PURGE  
     definition 136

- use of 44
- /PW statement description 88
- /QUERY definition 137
- /RATES definition 139
- /RENAME
  - definition 140
  - example 141
  - use of 43
- /REPLACE definition 143
- /REPLY
  - definition 145
  - example 145
- /RO definition 146
- /RSAVE definition 147
- /SAVE
  - definition 149
  - use on batch 89
- /SCRIPT definition 151
- /SCROLL definition 153
- /SET statement
  - definition 31
  - example 31,33,34
- /SKIP definition 154
- /SLEEP definition 155
- /SORT definition 156
- /STATUS definition 157
- /SUMMARIZE definition 158
- /TABIN definition 159
- /TABOUT definition 161
- /TABS definition 163
- /TABSET
  - definition 165
  - example 13
- /TAPE definition 167
- /TEXT definition 168
- /USERS definition 169
- /VERIFY definition 170
- /XEDIT definition 171
- /Y
  - definition 145
  - example 145
- Access control
  - library files 36,43
  - work files 54
- Account number description 7
- Accounting
  - library file charges 37
  - resource related charges 73
  - session related charges 73
  - work file charges 49
- Attention handling
  - causing an interrupt 96
  - definition 14

- Auto program modifying 75
- BLKSIZE parameter
  - of /FILE statement 51,58
- BUFNO parameter
  - of /FILE statement 53,59
- Background batch 87
- Batch
  - /LIBRARY command 89
  - /SAVE command 89
  - CPU time limit 88
  - background 87
  - password 88
  - password changing 76
  - printout limit 88
  - privileged accounts on 89
  - punched output limit 88
  - use of 87
- Beginner mode 75
- CHANGE Command (/PROFILE)
  - definition 75
  - example 77
- CLASS parameter
  - of /FILE statement 66,67,69
- COPIES parameter
  - of /FILE statement 68,69
- Cardboard making little holes in 45
- Carriage control defaults for batch and term 64
- Catalog description 49
- Characters per idle
  - changing 108
  - definition 108
  - setting at sign-on 122
- Control statements format of 17
- Conversational read list
  - TERMIN unit 63
  - definition 15
  - example 104,15
  - modifying with /CLIST 103
- DEN parameter of /FILE statement 59
- DISP parameter
  - of /FILE statement 21,38,42,53,68,70
- DSNAME parameter
  - of /FILE statement 21,38,41,50
- Debug mode
  - changing 76



commands 81  
 entering 80  
 use of /DEBUG 79  
 use of /PER 79

END Command (/PROFILE)  
 definition 76  
 example 77

Editor flip character  
 changing 107  
 changing default 76

Example formats 3

File access attributes  
 changing defaults 76  
 displaying 124  
 with /RENAME 140  
 with /REPLACE 143  
 with /RSAVE 147  
 with /SAVE 149

GET Command (/PROFILE)  
 definition 75  
 example 77

Hex substitution  
 changing 107  
 changing default 76  
 default 11  
 definition 12  
 example of use 12

Input/output units  
 /FILE statement 20  
 default units 23  
 dummy unit 71  
 magnetic tape 57  
 printed output 64  
 terminal input 63  
 terminal output 64  
 virtual card punch 69  
 virtual card reader 66  
 virtual printer 67

KEYLEN parameter of /FILE  
 statement 51

LABEL parameter of /FILE  
 statement 59

LEVELS parameter of /FILE  
 statement 38

LRECL parameter  
 of /FILE statement 21,42,  
 51,58

Library files

/INCLUDE statement 39  
 access control 36,43  
 charges 37  
 creating 41  
 deleting 44  
 description 35  
 library space limit 36  
 listing names of 43  
 purging 136  
 reading 37  
 renaming 43

Line editing functions  
 10-12

Line length  
 changing 108  
 definition 108  
 setting at sign-on 122

Logical backspace character  
 changing 106  
 changing default 76  
 default 11  
 definition 11  
 example of use 11

Logical delimiter  
 default 11  
 definition 11  
 example of use 11

Logical escape character  
 changing 106  
 changing default 76  
 default 11  
 definition 12  
 example of use 12

Logical line delete  
 changing 106  
 changing default 76  
 default 11  
 definition 12  
 example of use 12

Logical line end  
 changing 106  
 changing default 76

Logical tab character  
 changing default 76  
 definition 14  
 example 107

NAME subparameter of UNIT  
 parameter 20

NOPRINT parameter of /JOB  
 statement 28

NSEGS parameter of /LOAD  
 statement 26

OPT parameter

- of /FILE statement 54,68
- of /LOAD statement 26
- OS
  - BDAM type data sets 48
  - BSAM type data sets 48
  - QSAM type data sets 48
  - labelled tapes 57
  - what it is 3
- OUTLIM parameter
  - of /FILE statement 68,69
- PARM parameter of /LOAD
  - statement 25
- PF keys
  - querying 138
  - setting 108
  - use 10
- PRINT parameter of /JOB
  - statement 28
- PSW examination and
  - modification 84
- PURGE program for library
  - files 44
- Password
  - changing 74,75
  - use 7
- Program tracing
  - setting /DEBUG mode 112
  - setting /PER options 131
- RECFM parameter
  - of /FILE statement 21,52,58,65,67
- RENAME program for library
  - files 43
- Registers
  - floating point 84
  - general purpose 83
- SPACE parameter of /FILE
  - statement 52
- Sign-off procedure 130
- Sign-on
  - example 7
  - procedure 7
  - using a 3270 8
- Storage examination and
  - modification 82
- Symbolic parameters
  - assigning defaults 30
  - defining 29
  - overriding defaults 31
  - system defined 31
  - use 29,32
- TIME parameter of /LOAD
  - statement 27
- TRANS parameter
  - of /JOB statement 28
  - of /LOAD statement 26
- Tabs
  - changing in/out defaults 76
  - changing input default 76
  - changing output default 76
  - input 13
  - output 13
  - physical 13
  - setting in/out tabs 163
  - setting input tabs 159
  - setting output tabs 161
  - setting physical tabs 165
  - use 12-14
- Tapes
  - labelled 57
  - non-labelled 57
  - write protecting 61
- Temporary work files 56
- Terminal input 63
- Terminal linesize changing
  - 107
- Terminal output 64
- Terminals
  - ASCII sign-on procedure 6
  - VPS supported 5
  - direct wired 5
  - 2741 sign-on procedure 5
  - 3270 operation 8
  - 3270 sign-on procedure 8
- UNIT parameter of /FILE
  - statement 20
- Upper/lower case translation
  - setting 168
  - use 14
- User profile
  - definition 74
  - modifying 74-77
- VOL parameter of /FILE
  - statement 58
- VPS
  - command and stmt formats 2
  - description 1
- VTOC description 49
- Virtual card punch 69
- Virtual card reader 66
- Virtual printer 67

Work files

access control 54  
advantages 47  
catalog 49  
creating 55  
defining 50  
deleting 117,55  
description 47  
direct 48  
disadvantages 47  
editing 56  
naming conventions 54  
renaming 118,55  
sequential 48  
space charges 49  
structure 48  
temporary 56  
track limit 49

