SH20-9062-2

Program Product

# VSPC FORTRAN Terminal User's Guide

IBM

SH20-9062-2

Program Product

# VSPC FORTRAN Terminal User's Guide

**Program Number 5748-FO2**

IBM

**Third Edition (September 1978)**

This edition, as amended by technical newsletter SN20-9303, applies to Release 1 of VSPC FORTRAN, program product 5748-FO2, and to any subsequent releases until otherwise indicated in new editions or technical newsletters. Information pertaining to Release 2 of VSPC is for OS/VS2 MVS and OS/VS1 only.

The changes for this edition are summarized under "Summary of Amendments" following the list of figures. Specific changes are indicated by a vertical bar to the left of the change. These bars will be deleted at any subsequent republication of the page affected. Editorial changes that have no technical significance are not noted.

Changes are periodically made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370 Bibliography,* GC20-0001, for the editions that are applicable and current.

Publications are not stocked at the address given below; requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, P. O. Box 50020, Programming Publishing, San Jose, California, U.S.A. 95150. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

# PREFACE

This publication provides the tutorial and reference information necessary to develop programs in the VSPC FORTRAN language under one of the VS Personal Computing program products:

- VSPC OS/VS1 (Program Number 5740-XR5)
- VSPC OS/VS2 MVS (Program Number 5740-XR6)
- VSPC DOS/VS (Program Number 5746-XR3)

The material in this guide covers information necessary for a typical user to run VSPC FORTRAN under VSPC. The information provided includes the following:

- The structure of VSPC, including workspaces, libraries, filenames, and passwords
- Operation of terminals with the assumption that the user is typing on the IBM 3270 Information Display System, on an IBM 2741 Communication Terminal, or on an IBM 3767 Communication Terminal. A chart outlining essential information for all VSPC terminals is in Appendix C.
- Techniques of logon, error correction, and logoff
- Command language for designing, modifying, and running programs
- VSPC and VSPC FORTRAN programming considerations
- Description of the VSPC FORTRAN Subprogram Library
- Library maintenance, controls, file organization, and a sample session

Appendixes are included that contain information on the following topics:

- Data File and Program Conversion Considerations
- Device Dependencies
- VSPC Terminal Quick Reference Chart
- Mathematical Function Accuracy Statistics
- Batch Considerations
- VSPC FORTRAN Error Messages

The reader is assumed to be familiar with the FORTRAN IV language as explained in the publication *IBM System/360 and System/370 FORTRAN IV Language,* which is a prerequisite to this guide.

See *VS Personal Computing (VSPC) Terminals* for detailed explanations of terminals supported by VSPC.

## Related Publications

The following related publications are referenced in this book:

- *VS Personal Computing (VSPC) General User's Guide and Command Language,* SH20-9071
- *VS Personal Computing (VSPC) Terminals,* SH20-9073
- *VS Personal Computing (VSPC): Writing Processors,* SH20-9074
- *IBM System/360 and System/370 FORTRAN IV Language,* GC28-6515
- *VS Personal Computing (VSPC) FORTRAN Installation Reference Material,* SH20-9063

# CONTENTS

# FIGURES

# SUMMARY OF AMENDMENTS

## July, 1979

### *Enhancements to VSPC FORTRAN*

**New Programming Features**

New functions added to the OPSYS subroutine are: COMMAND, DELAY, FSM, and RESET.

Technical additions and changes resulting from Release 2 of VSPC are also included.

## September, 1978

### *VSPC Release 2*

**New Programming Features**

Support for additional display system devices

Improved terminal support
- VIEW Mode (Full screen) editing
- Program function keys

Error message clarification

Current line pointer support

New commands: PFKEY, VIEW, NEWLINE, BACKSPACE, COPY, LOCATE, EXTRACT, JOIN, SPLIT, TEXT, RELEASE, ACQUIRE.

Improved Commands: LINESIZE, TRANSLATE, HARDCOPY, INPUT, LIST, CHANGE, FILE, PUNCH.

## September, 1977

### *VSPC FORTRAN Link Processor (OS/VS Only)*

**New Programming Feature**

A description of the VSPC FORTRAN subroutine link processor has been added to the "Program Development" chapter.

**Service Change**

The logon procedure with 3270 terminals has been clarified.

## August, 1976

### *VSPC AID*

**New Programming Feature**

A description of VSPC AID has been added to the section on VSPC commands.

### *Response to Input Request*

**Specification Changes**

A note has been added to the entry on "Conversing with VSPC" about replies to input requests that may disconnect you from VSPC.

# DESCRIPTION OF VSPC

VS Personal Computing (VSPC) makes a computing system available to many terminal users at the same time. Scientific computation, computer programming, and information retrieval are all possible to VSPC users through communication terminals.

The problem-solving nature of VSPC FORTRAN and the interactive nature of VSPC are particularly suited to each other. Together, they provide a quick and easy means to help you get your work done.

VSPC, which runs under OS/VS1, OS/VS2 MVS, and DOS/VS, offers a set of functions tailored for users without extensive data processing knowledge. VSPC has commands (instructions) that let the time-sharing user create and execute VSPC FORTRAN programs from a special typewriter or display station called a terminal. With the commands available, you can:

- Start and end terminal sessions
- Format your work
- Create programs
- Modify programs
- Execute programs
- Maintain libraries
- Send and receive messages

## User Profile

Before VSPC can recognize you and allow you to do work, your VSPC administrator must introduce you to VSPC through a *user profile* as shown in Figure 1. Your *user profile* defines you to VSPC.

| | |
|---|---|
| **Your User Profile** | |
| Your user number (usernum) | 12345 |
| Your workspace attribute | FORTRAN |
| Your library type | private |
| Your assigned project library | 22222 |
| Your logon password | TABOO |
| Your job entry code | C12345 |
| **Other Information** | |
| Public Libraries available | 100<br>200<br>300 |
| Logon message | WELCOME TO VSPC |

Figure 1. Sample User Profile, Libraries Available, and Logon Message

### *User Number*

Your user number identifies you to VSPC. Whenever you log on, you identify yourself by typing your *user number,* which is one through seven digits long, and is a unique value.

## *Workspace Attribute*

Your workspace *attribute* tells VSPC what kind of work you'll most often be using your workspace for. Your workspace *attribute* can be:

*CLIST:* which specifies you'll be entering VSPC commands in the form of data—you'll learn about it in the "Processing Command Lists" section of a following chapter.

*DATA:* which specifies you'll be entering information not associated with any processor.

*processor-name:* which names a foreground processor available at your installation and specifies that you'll be writing programs to be treated as data input to the processor. The processor-names you will be using most often are FORTRAN (for free-form FORTRAN), XFORTRAN (for fixed-form FORTRAN), and FLINK (for linking object programs).

## *Library*

Information about the libraries you'll use is included—what kind of library is provided for your use, how much computer storage you can use for it, and what additional libraries are available to you.

## *Logon Password*

A logon password can, optionally, be assigned to you. If one is assigned, you must supply the correct password each time you log on to do work. If you wish, you can change your logon password during your terminal session; you must then supply the new password the next time you log on.

## *Job Entry Code*

When you are assigned a job entry code, you are allowed to submit batch processing jobs through VSPC facilities. Your job entry code is unique, and identifies you to the batch processing system.

Once your VSPC administrator has introduced you to VSPC by defining your user profile, you can log on to do work. You do work by specifying VSPC commands.

## VSPC Commands

**A VSPC *command* is a word that tells VSPC to perform a specific action. The kinds of actions the commands perform can be thought of in groups of processing capabilities as follows:**

*Terminal Oriented Commands:* which help you use the terminal itself in communicating with VSPC. Using them, you can log on and log off (start and end a terminal session), change your logon password, change the line length and tab settings you send to VSPC, enter lines, and change lines you have already entered. On the IBM 3270 you can set the program function keys to often-used commands or character strings. You can specify a character to translate into a tab, new line, or a backspace character.

*Message Commands:* which let you send messages to users on other terminals, and prepare your own terminal to accept or refuse messages from others.

*Data Editing Commands:* which let you add or change a line in the workspace, delete characters, delete lines, replace lines, move lines or copy up or down, split or join lines, change line numbers, and locate occurrences of specific groups of characters. You can specify whether or not you want certain commands to display the text of the affected line at your terminal.

*Workspace Oriented Commands:* which let you change the *attribute* of your workspace, name your workspace contents and save them in a library, and erase the workspace contents when they are no longer needed.

*Library Management Commands:* which let you add and remove files in a library, transfer files among users, specify that other users can or cannot read a file, and specify that the file cannot be changed. You can also display – that is, print or obtain some other visual presentation – the library directory.

*Job Entry Commands:* which let you submit jobs for batch processing, to query their status, to cancel them, to specify that job output is to be sent to some specific destination, and to erase job output once it has been produced. (Job entry commands are not always available for your use; check with your VSPC administrator.)

*AID Commands:* which let you receive online explanations of VSPC messages, and online prompting in the formation of valid VSPC commands. (AID commands are not always available for your use; check with your VSPC administrator.)

### *List of VSPC Commands and Actions*

VSPC general commands are listed in Figure 2, in alphabetic order. Those commands you will use every day in order to do your VSPC FORTRAN work are described later in this publication. Those which you may occasionally want to use, and which are not described here, are detailed in *VS Personal Computing (VSPC) General User's Guide and Command Language.*

| **Command** | **Action** |
|---|---|
| ? | VSPC AID asks for explanation or prompting |
| ACQUIRE | Acquires ownership of a noncontrolled PROJECT library file that has been released by another user |
| ALLOCATE | Associates a file with a FORTRAN unit-number |
| BACKSPACE | Assigns logical backspace character |
| CHANGE | Alters a group of data characters in a workspace |
| CLEAR | Erases a workspace |
| COPY | Duplicates lines from one location to another in a workspace |
| DELETE | Erases lines from a workspace |
| ENTER | Sets the workspace attribute |
| EXTRACT | Deletes all but unspecified lines from a workspace |
| FILE | Creates or modifies a file |
| FIND | Locates a group of data charcters in your workspace |
| FREE | Removes previous unit-number allocations |
| HARDCOPY | Allocates a 3270 attached printer |
| INPUT | Begins automatic VSPC line numbering |
| JOIN | Unites two consecutive lines |
| KEY | Specifies the keyboard as the input source for CPT-TWX terminals |
| LINESIZE | Sets session line length |
| LIST | Displays current workspace contents |
| LOAD | Places a file in a workspace |
| LOCATE | Sets current line pointer |
| MERGE | Combines a file with the current workspace contents |
| MESSAGE | Sets conventions for message reception and message header printing |
| MOVE | Repositions lines of data in the workspace |
| NAME | Specifies an identifying name for a workspace |
| NEWLINE | Assigns logical new line character |
| OFF | Ends terminal session |
| PASSWORD | Specifies new logon password |
| PFKEY | (3270 only) Requests that a PF key be interpreted as specified |
| PROTECT | Specifies or removes file access restrictions |
| PUNCH | Specifies paper tape output for CPT-TWX terminals |
| PURGE | Removes a file from a library |
| QUERY | Displays information about VSPC usage |
| QUERY FILE | Displays status information about a file |
| QUERY LIBRARY | Displays a list of files in a library |
| RELEASE | Releases a file in a noncontrolled PROJECT library so another user can gain ownership of it with ACQUIRE |
| RENUMBER | Changes workspace line numbers |
| RUN | Compiles and/or executes a source or object program, a combined object program, or a command list |

Figure 2 (Part 1 of 2). VSPC General Commands

| Command | Action |
|---|---|
| SAVE | Places workspace contents into a library |
| SEND | Transmits messages from the terminal |
| SHARE | Makes a file available to other users |
| SPLIT | Splits one line into two lines |
| STORE | Compiles a source program and places the object program in a library; combines object programs and places the composite object program in a library. |
| TABSET | Sets session tabs, assigns logical tabset character, specifies expansion into blanks, or sets FORTRAN IV column tabs |
| TAPE | Specifies paper tape as input for CPT-TWX terminals |
| TEXT | Sets global default for the TEXT/NOTEXT operands of the CHANGE, LOCATE, and FIND commands |
| TRANSLATE | Sets upper/lowercase usage or usage of a translate table |
| VIEW | (3270 only) Changes VSPC screen format to VIEW mode (full screen edit) or, if you are already in VIEW mode, issuing the VIEW command changes the display on the screen |
| VSPC | Begins a terminal session |

Figure 2 (Part 2 of 2). VSPC General Commands

## *Command Format*

All VSPC commands have a similar format, as shown in Figure 3. The first word of a command is the *command-name* which specifies an action to be taken. The command name is followed, optionally, by one or more operands that provide additional information or that modify the action specified by the command.

Each *command* begins on a line by itself, and the first meaningful characters on the line must be the command name. If anything but spaces, commas, or tabs precedes the command name on the line, the command cannot be recognized by VSPC.

VSPC accepts commands entered at your terminal typed in any combination of lowercase and uppercase letters. Before they are acted upon, VSPC translates them into uppercase. Thus, if you enter any of the following:

```
save 200 quotient
SAVE 200 QUOTIENT
save 200 QUOTIENT
SAVE 200 quotient
Save 200 Quotient
```

or any other such combination, VSPC considers them to be equivalent to:

```
SAVE 200 QUOTIENT
```

When VSPC responds to a command, the message is always displayed in uppercase letters. For example:

```
READY
```

So that you can distinguish in the following text between what you enter and what VSPC displays, all examples in this book show what you enter in lowercase letters; the normal VSPC response in uppercase letters – as it is displayed at your terminal.

```
save 200 quotient
ENTER PASSWORD
```

The meaning of the combinations of words, spaces, and special characters is explained in this section.

## COMMAND-NAME

The command-name, in the lefthand column, is a VSPC keyword.

The command-name must be spelled exactly as shown, except when it is abbreviated, as described later in this section.

| | | |
|---|---|---|
| COMMAND-NAME | *operand-1* | Explanatory information about operand-1. |
| | *operand-2*<br>*operand-2*<br>*operand-2* | Explanatory information about operand-2. |
| | *operand-3*<br>*operand-3* | Explanatory information about operand-3. |

Figure 3. Command Format Presentation

## Command Operands

Operands can be either variable or keyword.

Variable operands supply VSPC with information that you, the VSPC FORTRAN terminal user, determine.

Keyword operands define a predetermined action to VSPC.

Some commands use only variable operands; other commands use both variable and keyword operands. Only a few commands use only keyword operands. Some commands use no operands at all.

Punctuation in operands has special meanings, as described in the following paragraphs:

- Parentheses enclose variable information that governs the action of the keyword operand. Parentheses may be omitted unless their omission makes the command ambiguous, such as when changing a file password with the PROTECT command.
- A colon (:) placed between two numbers specifies a range of numbers (as illustrated in the explanation of the LIST command later in this publication).
- Single quotation marks ( ' ) precede and follow groups of characters – called character strings – that are to be treated by VSPC exactly as specified.

The operands valid for a command-name are listed in the middle column, using the following conventions:

- Operands enclosed between two horizontal lines represent available options for one operand. If all options may be omitted, the action taken is explained in the third column.
- Operands separated by a horizontal line represent separate operands. When specified, each must be separated from the next, and from the command-name, by a space, comma, or tab.
- Operands printed in uppercase represent keyword operands. These operands, when specified, must be spelled exactly as shown, except when they are abbreviated (see "Command and Keyword Abbreviations"). Capitalized operands may have words in *italicized lowercase letters* following them; such words represent user-supplied information that modifies the keyword operand.
- Operands printed in *italicized lowercase letters* represent information supplied by the user. If the operand ends with a hyphen and a numeral, the basic meaning of the operand is unchanged. The suffix is added only to clarify text references.
- If the order in which operands are written is significant, an entry in the righthand column defines the required order. If no order is given, the operands may be written in any order.
- Options *within* operands must be written in the order shown.
- Special characters—some operands contain one or more of the following characters as part of the optional or required information:

| Special Character | Meaning |
|---|---|
| ( ) | enclose a data subfield |
| = | equates the logon request with a specific user number |
| ' | encloses a character string |
| * | specifies your current line, the workspace, or the terminal |
| : | denotes a range of numbers |
| / | requests a password prompt |

When such portions of the operands are entered from the terminal, these characters must be entered exactly as shown. For example:

| Format | Valid Entry | Invalid Entry |
|---|---|---|
| *begin-line*:*end-line* | `100:200` | `100 200` |
| '*string*' | `'host'` | `host` |
| (*name*) | `(FILEB)` | `FILEB` |

- Text references agree with the format notation. That is, a reference to a specific keyword operand is shown in uppercase letters; a reference to a specific *italicized* operand is *italicized.* For example: a reference to a *line* is a reference to a specific *line* operand; a reference to "line", however, is a reference to lines in general.

If command operands allow a choice of two mutually exclusive keywords, more than one of the keywords may be specified. In this case, the last keyword specified is the one VSPC uses. For example:

```
protect * noread nowrite read
```

specifies that the READ option is to override the NOREAD option. (Note that if more than two options for a keyword are possible, then only one option may be specified, or the user gets an error message.)

## Command and Keyword Abbreviations

In a command, both the command name and the keyword operands can be abbreviated to their leftmost significant characters. Among command names, the characters must be unique among all command names (that is, the SAVE, SHARE, and STORE commands can be abbreviated respectively as SA, SH, and ST). A keyword operand must be uniquely spelled within a command (that is, the TABSET operand of the QUERY command can be abbreviated as TAB or TA but not T).

Successively longer abbreviations—adding characters at the right—are permitted. For example, the PROFILE operand of the QUERY command can be abbreviated as P, PR, PRO, PROF, PROFI, and PROFIL.

An easy rule to remember is that in VSPC you can always abbreviate a command name or keyword operand to its first three characters.

## Separators

In the examples in this book, you'll notice that the operand is always separated from the command name by a space. This is because VSPC requires a *separator* between each operand and command name: valid separators are blanks, commas, and (if workspace tabs have been set) tabs. VSPC recognizes one or more separators as logically one separator only.

## Command Continuation

When the command *entry* ends with a minus sign (-), the next succeeding line is treated as a command continuation; logically, it immediately follows the last character preceding the minus sign.

When the command *entry* ends with a plus sign (+), the next succeeding line is treated as a field continuation; logically, the first non-separator character of the succeeding line immediately follows the last character preceding the plus sign.

### Comments

Comments in VSPC are valid. A comment must begin with the characters /* and end with the characters */. For example:

```
/* this is a comment */
```

Comments do not affect the execution of the command or command list in which they are included.

### *Explanatory Information*

Explanatory information about each operand is given in the righthand column: if an operand is required or optional, if its position in relation to other operands is significant, if it is optional, what action is taken when it is omitted, and so forth.

## VSPC AID

If VSPC AID has been included at installation time, you can ask the program to help you specify the operands for VSPC commands correctly. For any VSPC command, enter the command name preceded by a question mark (?). VSPC AID will reply with a series of prompts that question you about the operation you want to perform. From your replies, AID forms the command in valid format and then allows you to choose whether or not to execute it.

VSPC AID will also give you an explanation of error messages sent by VSPC. To display an explanation of the *n* th preceding message, enter "?n".

If you have forgotten the proper name of the command you want to specify, you can get a categorized list of VSPC commands by entering "?COMMANDS" or "?CMDS", and to get an online explanation of all the capabilities of VSPC AID, enter "?AID".

## VSPC Data

When you're typing at the terminal, you can enter commands, as previously described, or you can enter *data.*. Data can be anything you want it to be: a specific problem solution, testing information, a VSPC FORTRAN program, anything at all.

Data always begins with a *line number.* The line number indicates that whatever follows on this line is data, and is to be placed in your workspace. The line number identifies this line of information, and makes it possible to refer back to it at any future time—to display it, change it, replace it, or to delete it. For example, if you type:

```
query library (100)
```

that is a command; VSPC recognizes it as such, and performs the indicated action. However, if you type:

```
080 query library (100)
```

VSPC recognizes 080 as a line number, and treats the LIBRARY command that follows as data, placing it in the workspace. For example, all of the following lines are treated as data by VSPC:

```
100   write (6,100):
080   query library,100
120   80 100 120 140 160 180 200
250   The quick brown fox jumps over the lazy dog
```

In VSPC, the most frequently-used forms of data are:

- a program you write for later execution (such as your FORTRAN programs).
- a command list you'll execute later (command list processing is described elsewhere in this publication).
- information you'll use during the processing of your FORTRAN program.

## *Line Numbers*

In VSPC, any one of these forms of data must be preceded by a *line number.* The line number must be from one through five digits long. You create line numbers in one of two ways:

- typing each line number and then typing the line of information (you can type the line numbers in any numerical order, and VSPC automatically rearranges the data lines in ascending numerical order for you). If your line numbers contain fewer than five digits and your first data character is numeric, you must type a space between the line number and the data.
- asking VSPC to generate line numbers for you.

## *Using Data*

Once you've created your data, you can use it as you please: you can use it immediately and then erase it, or you can save it in a library under a unique *filename,* or you can use it immediately *and* save it for future reference. If you want to, you can erase it without using it at all.

There are three important things you should always keep in mind when you are creating data:

- During one terminal session, unless you have explicitly erased the data from your workspace, it still exists in the workspace. Suppose, for example, you have created and saved one group of test data and now want to begin creating a second set. If you don't erase the first set of data before you begin work on the second, you will simply add the second set of test data to the first (and, depending on the line numbers you're assigning; at the beginning, middle, or end of the first set). This may not be what you want to do at all.

- At the end of your terminal session, unless you specifically tell VSPC to save your current data, VSPC will erase it when you log off. That is, if you do not save your data in a library during the session and do not explicitly specify that it should be saved when you log off, the data will not be available the next time you log on.
- If you are working at the terminal and for some reason you're disconnected, VSPC saves your current data for you—unless the shutdown is due to actual machine failure (in which case VSPC still makes the attempt). If VSPC can save your data, when the system begins working again, you'll be able to retrieve the data you were working on.

## VSPC Libraries

In VSPC, your user profile always defines a type of library for your own use; your library has a *library number* (which is the same as your user number), and you are the *library manager* of this library. Your own library is always available to you, and other libraries may also be available.

Three kinds of libraries can be defined in VSPC, and your VSPC administrator may make you the library manager of any kind. The three types of libraries are private libraries, project libraries, and public libraries.

### *Private Libraries*

If your library is a private library, it is meant primarily for your use only. You are both the *library manager* and the *owner* of every file. You can, if you wish, specify that some files can be read by other users; you can also specify that some of your files cannot be changed even by yourself.

In addition to your own library, your VSPC administrator may have made a project library available to you as well. The public libraries are available to you, as they are to every other terminal user.

### *Project Libraries*

If your library is a project library, then a specific group of VSPC users is allowed to read files from your library. Your VSPC administrator can define your project library in either of the following ways:

1. Noncontrolled – Other project users are allowed to place files in the library; each such user is the *owner* of the files he puts in it. You are the *library manager*; however, you are the *file owner* only of those files you place in the library. Only the *owner* of a file can modify it; however, ownership of a file can be transferred from one project user to another. Only the *owner* and you, the *library manager*, can delete it.
2. Controlled – Other project users can read files in the project library; they cannot put files in it. Thus, you are the *owner* of every file, and so only you can modify or delete anything in the library. Such libraries are known as controlled project libraries.

If yours is a project library, your VSPC administrator may also make another project library available for your use.

## Public Libraries

If your library is a public library, then every VSPC user can read files from your library. Public libraries can be of the same two kinds as project libraries, and the same rules apply to file modification and deletion, except that ownership cannot be transferred.

If yours is a public library, your VSPC administrator may also make a project library available for your use.

## Filenames

When you save new work in a library, you become the *owner* of the file you create, and you must always specify a unique *filename* for it. When you specify a filename, you specify the following components:

*libnum name/password*

- *libnum*—The optional *library number* (which is always the one-digit through seven-digit user number of the library manager). When you're referring to a file in your own library, you can omit the library number.
- *name*—which is required, and which must be a unique identifying name within this library; a *name* is one through eight characters long—the first character being alphabetic, and the following characters being A through Z, 0 through 9, or $, #, or @.
- */password*—The optional *file password.* (It is described in the "VSPC Access Control Features" section.)

Every file in your library has an *attribute,* such as DATA or FORTRAN. The file attribute is the attribute your workspace had when you placed the file in the library.

## File Types

In any VSPC library you can put any VSPC files: sequential, direct, object program, undefined sequential, or undefined direct. Each type is described in the following paragraphs.

Files which can be processed by the LOAD, SAVE, and VSPC editing commands are considered to be editable files. Thus, for example, DATA files for input to VSPC FORTRAN programs can be created and/or maintained by the VSPC editing commands, and VSPC FORTRAN output files can be inspected for correctness while the program is being tested.

***Sequential Files:*** are files in which FORTRAN always processes the records in the order of their line numbers. The records are placed in the file in line number order, and FORTRAN always retrieves them from the file in that order. However, using VSPC commands you can edit the lines of a sequential file in any order. Records can be of varying lengths, and the increment between line numbers is of no importance.

***Direct Files:*** are files in which FORTRAN can process a specific record through a reference to its line number. Your program can also process records sequentially in line number order. You can edit the lines of a direct file in any order. The records are placed in the file in line number order, and the line numbers must increment by one (that is, line numbers must be 1, 2, 3, 4, 5, etc.). All records must be the same length.

***Object Program Files:*** are produced by the FORTRAN and XFORTRAN compilers and the FLINK processor (in OS/VS) when you specify the STORE command. Such files consist of executable machine instructions; that is, your compiled FORTRAN programs. They may be combined using the link processor in OS/VS. You can use object program files with the VSPC RUN command, but you cannot edit or use them for any other purpose.

***Undefined Files:*** are special purpose data files created by a program. They contain data in a format tailored to VSPC FORTRAN's needs, and cannot be edited by the user. Only an executing program can process or update such files. Undefined files can be either sequential or direct.

Note: Access to external VSAM files is not supported for VSPC FORTRAN.

## *Special File CONTINUE*

Any VSPC library can contain a special file with the *filename* CONTINUE. This file is used by VSPC to ensure that current work in the user's workspace is automatically saved, under certain circumstances, at the end of a terminal session.

When the workspace contains editable data, the CONTINUE file is created when either:

- The user issues an OFF CONTINUE command at the end of the session.
- The session is ended by conditions beyond the user's control (called a "forced ending").

In either case, the saved work can be made available during the next session with the following command:

```
LOAD CONTINUE
```

CONTINUE usage conforms to the following rules:

- The user can retrieve a CONTINUE file only from his own library.
- Unless the user specifies OFF CONTINUE, a CONTINUE file is automatically removed from his library when the next OFF command is executed.
- A CONTINUE file can be overlaid by a CONTINUE file of another type.
- The SAVE command can specify CONTINUE. Note, however, that this destroys any data already present in the CONTINUE file—data automatically saved at the end of a previous terminal session. Note, also that any file saved with the name CONTINUE will normally be purged at the end of the present session.
- The FILE, NUMBER, PROTECT, SHARE, STORE, and SUBMIT commands may not specify CONTINUE.
- The CONTINUE file cannot be used for processing as a data file in any mode.

## VSPC Access Control Features

VSPC access control features help you keep unauthorized people from using your VSPC user number and/or reading your library. They also make it possible to share files from your library with those who have need to use them. VSPC controls access through passwords, file protection, and file sharing.

### *Passwords*

A *password* is a group of characters you can use to limit access to VSPC itself or to your files. There are two kinds of passwords—*logon passwords* and *file passwords*. Both kinds of passwords have the same rules of formation: they can be from one through eight characters in length, and they can be made up of any combination of alphabetic characters and numerals 0 through 9. You can specify passwords in lowercase letters; they'll always be translated to uppercase. For example, if you specify:

```
TABOO02
```

or

```
taboo02
```

as a password, VSPC treats it as TABOO02. Note that the letter O and the number 0 are recognized as separate characters.

For security purposes, passwords are not displayed at your terminal. Some terminals suppress the printing of passwords; others print a series of overstruck characters called a "blot" over which you type your password.

#### Logon Passwords

The VSPC administrator can assign you a *logon password* in your user profile. You must provide this password before you are allowed to log on to work with VSPC. Once you have been given a *logon password,* you can change it during a terminal session. The next time you log on, you'll be required to provide the new password, not the original one. You can also give yourself a password if one hasn't been assigned to you.

#### File Passwords

You can specify *file passwords* when you are saving files for future reference. You add the password at the end of the *filename.* Any time you retrieve the file, you must first supply the correct file password before VSPC lets you access it.

### *File Protection*

You can specify *file protection* for any file you own. That is, you can limit access to the file, either by other users or even by yourself.

You can specify that no other terminal user can read, display, or copy your file. That is, even if other users can retrieve the file for use as a source program, they still cannot find out what is in it.

You can also specify that even you cannot modify a specific file. Other users can still read it and use it for processing as you can.

When both forms of file protection are in effect, others may be able to use the file as a source program, but only you can read it; you cannot change it or delete it.

## *File Sharing*

Files in a VSPC private library can ordinarily only be read by the library manager (who also is the owner of every file in the library). Similarly, files in a VSPC project library can only be read by members of the project.

If you own files in either type of library, VSPC lets you specify that files you choose to share can be accessed by any user. You can also specify that files you previously shared can be returned to their original unshared state.

Note that files in a public library are always able to be shared by all users.

## *Transfer of File Ownership*

If you own files in a noncontrolled project library (and you aren't the library manager), you can make any file you own available for transfer of ownership to another project library number. You do this by specifying, for example,

```
release 22222 quotient
```

When this RELEASE command is executed, file 22222 QUOTIENT is made available for another user of noncontrolled project library 22222 to acquire ownership; however, you still own the file.

After you have executed the RELEASE command, you or another user (but not the library manager) can issue an ACQUIRE command:

```
acquire 22222 quotient
```

If another project library member executes this ACQUIRE command, the ownership of the file is transferred to that library member, the space it occupies is subtracted from your library space and added to the other library member's space, and the file no longer has the *released* status. On the other hand, if another project library member runs a program that accesses the file for output or update, then at the time he accesses the file, the ownership of the file is automatically transferred to that library member (and the space accounting is transferred), but in this case, the file retains the *released* status (so that the file may continue to be accessed for output or update by other members of the project library).

If you issue an ACQUIRE command on a file that you have released, you're retracting the RELEASE command and, after the ACQUIRE command is executed, the file is no longer eligible for ownership transfer.

*Note:* The noncontrolled project library manager can also issue the RELEASE command, but only for files that he does not own; the library manager cannot issue the ACQUIRE command to take over ownership of other users' project library files.

## VSPC TERMINALS

The terminals you can use with VSPC are described in the following paragraphs. For each terminal there are several different keyboards available, and for each, some of the features may vary. The variations are described in the *VSPC Personal Computing (VSPC) Terminals* publication. Your VSPC administrator can tell you which terminals your organization uses.

**Note:** The valid VSPC switch settings for the terminals described below are shown in Appendix C.

### IBM 3270 Information Display System

The IBM 3270 Information Display System can be one of several display terminals —each has a display screen, a typewriter-like keyboard, and other auxiliary equipment. Optionally, a printer and/or a light pen may be attached to the terminal. A cursor (a one-character underline) indicates the screen position you are using–it is easily moved about for character correction, entry, or other functions. Several different keyboards are available for the variety of 3270 system terminals. One of the keyboard layouts available for the 3275/3277 is shown in Figure 4; one for the 3276/3278 is shown in Figure 5.

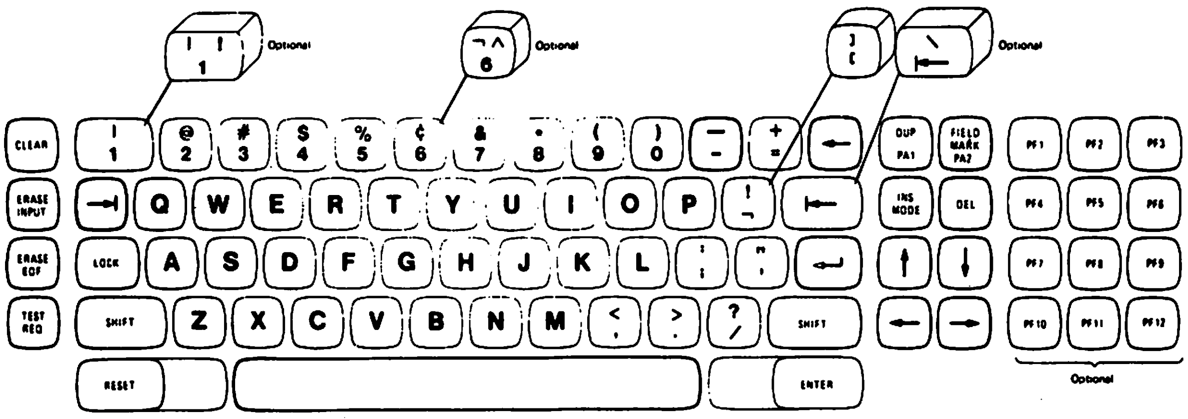


Figure 4. 3275 and 3277 Typewriter-Like Keyboard

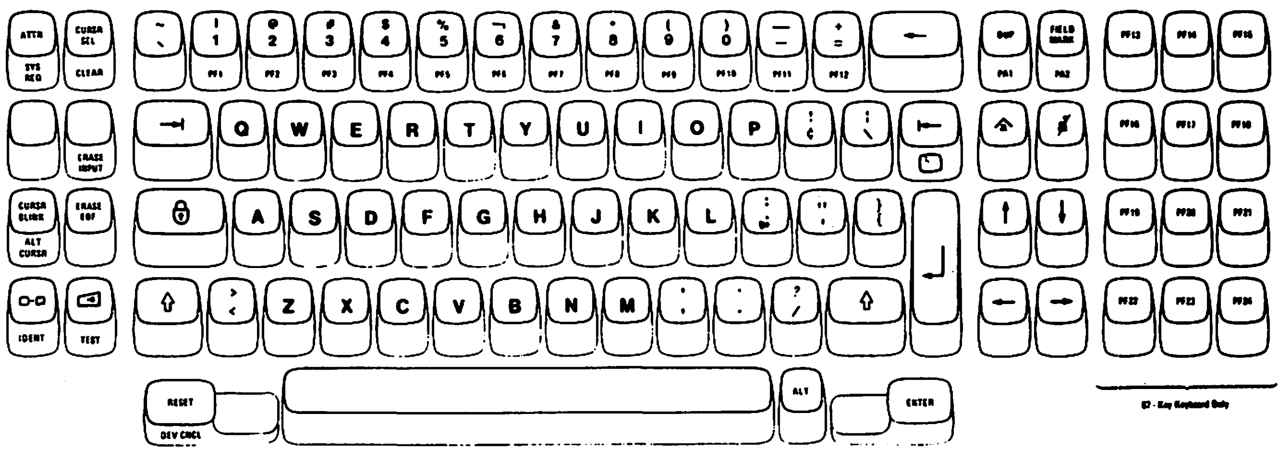


Figure 5. 3276 and 3278 Typewriter-Like Keyboard

## IBM 2741 Data Communication System

The IBM 2741 Communication Terminal looks like and acts like a conventional IBM Selectric® typewriter with two added controls that allow computer connection. One of the 2741 terminal keyboards is shown in Figure 6.

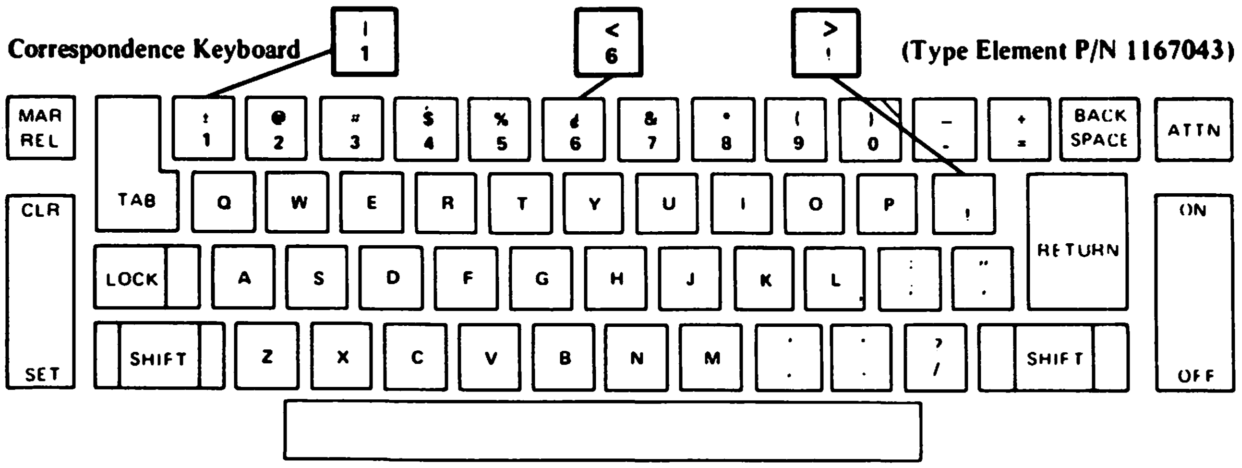


Figure 6. IBM 2741 Communication Terminal Keyboards

## IBM 3767 Communication Terminal

The IBM 3767 Communication Terminal is a convenient desk-top terminal that has a typewriter-like keyboard. When you are entering input, the print mechanism prints only in a forward direction; when it is printing computer output, however, the mechanism can print alternatively forward and backward (which speeds up the process). There is an audible alarm that warns you when you are approaching the end of an input line or when system problems occur. A three-character indicator shows the next print position in the input line, or (when an error occurs) the cause of an error condition. Some controls on the 3767 vary, depending on whether you are in start/stop (S/S) mode or in synchronous data link control (SDLC) mode; your VSPC administrator will tell you which modes you can use. One typical layout of the 3767 terminal controls is shown in Figure 7.

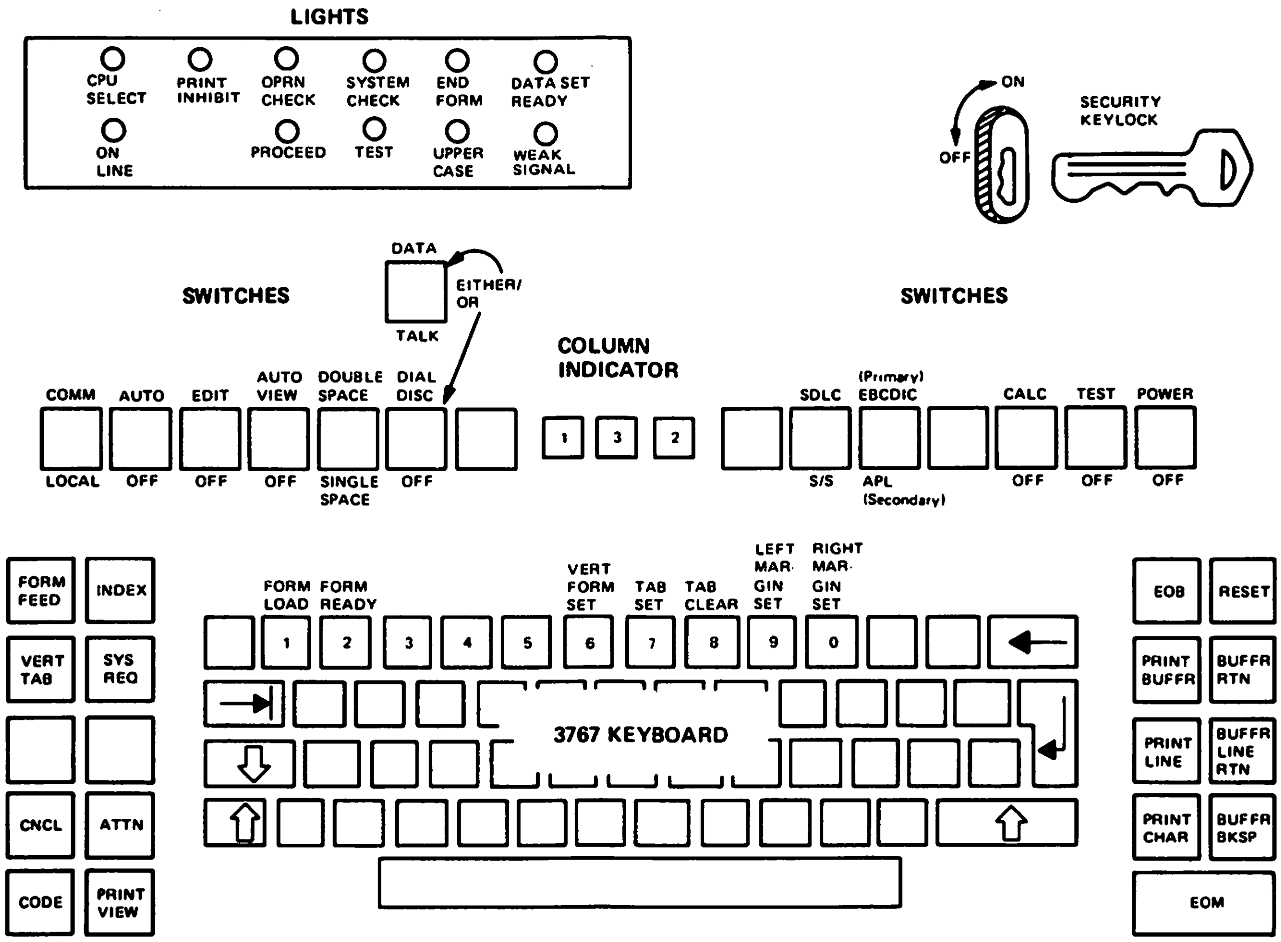


Figure 7. IBM 3767 Keyboard, Switches, and Light Layout

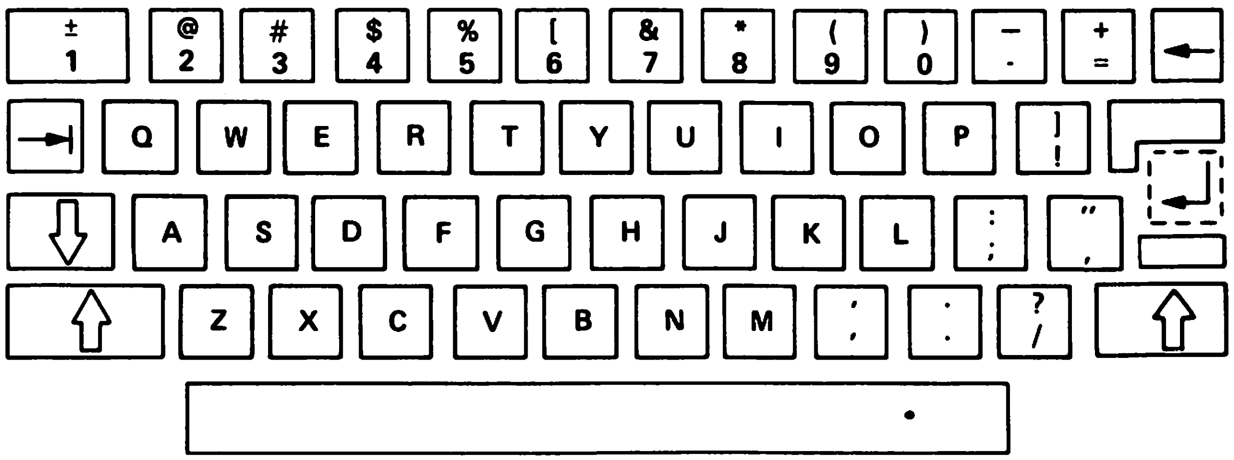


Figure 8. IBM 3767 Communication Terminal Correspondence Keyboard

## IBM 3770 Data Communication System

The IBM 3770 Data Communication System is a family of multipurpose keyboard/printer terminals. The IBM 3771 and 3773 terminals are designed for data entry, inquiry, and remote printing. The IBM 3774 and the IBM 3775 terminals each combine the features of the 3771 and 3773. For details on these terminals, see *VS Personal Computing (VSPC) Terminals.* For switch settings and logon instructions for the 3770, see the terminal reference chart.

## IBM 1050 Data Communication System

The IBM 1050 Data Communication System has special keys for computer connection, plus a printer-keyboard that resembles a conventional IBM Selectric ® typewriter.

## CPT-TWX (Models 33 and 35)

The CPT-TWX (Models 33 and 35) include a control panel, a print mechanism, and a keyboard. All alphabetic characters are uppercase letters; a special key generates control characters. The control panel is used to edit and transmit lines.

# VSPC TERMINAL-ORIENTED COMMANDS

The following activities are accomplished in the same way on any VSPC terminal:

- Specifying an attribute or password
- Formatting
- Displaying the status of your workspace, profile, or library
- Sending and receiving messages
- Signing off

## Specifying a Workspace Attribute

If the workspace attribute specified in the logon message is the one you want to use, you can begin making entries. Otherwise, with the ENTER command, you can specify a different attribute.

### *ENTER Command*

At the time your user profile was established, your VSPC administrator specified the default attribute your workspace assumes. This is the attribute it has every time you log on. You can explicitly change the attribute through the ENTER command.

| | | |
|---|---|---|
| ENTER | *processor-name*<br>CLIST<br>DATA | Optional—choose one or omit. If omitted, attribute of current workspace is displayed. |

*processor-name*
specifies a workspace attribute associated with one of the IBM program products that run under VSPC, or a user-written compiler or interpreter so designed. The named processor must be installed with VSPC. Valid processor-names you'll be using most frequently are:

FORTRAN (free-format FORTRAN)
XFORTRAN (fixed-format FORTRAN)
FLINK (FORTRAN link processor)

CLIST
specifies a workspace that can contain VSPC command lists (see the "Processing Command Lists" section).

DATA
specifies a workspace attribute that is not associated with any specific processing capability. The workspace can be used for any form of editable data.

When the ENTER command is executed, the new attribute replaces any previous attribute for this workspace.

## Adding or Changing Your Password

You can change an existing logon or file password or you can add a password if none exists.

### *PASSWORD Command*

The PASSWORD command supplies a new logon password.

| | | |
|---|---|---|
| **PASSWORD** | *password* | Optional—specify or omit. If omitted, password prompt is issued. |

*password*
specifies your new logon password.

Your password must comply with VSPC rules for passwords. See "Passwords" under "Description of VSPC."

The new logon password replaces the one in your profile and will be required at your next logon.

For additional security, the PASSWORD command can be entered without specifying the replacement *password*. VSPC then responds with the prompt, ENTER PASSWORD, and the print suppression method or eight blots to disguise your password.

Once you are successfully logged on, you can delete your password. Issue the PASSWORD command and press carrier return instead of a password after the prompt and you will no longer be protected by a logon password.

If your profile does not specify a password, you can use the PASSWORD command to create one.

*Examples:*

```
password pass
```

creates a logon password of PASS.

```
pas
ENTER PASSWORD
```

In the last example, when you enter the PASSWORD command (pas) immediately followed by ENTER (or carrier return), VSPC prompts you (ENTER PASSWORD) and uses either the print suppression method or the blot for you to add a new password or delete an existing one by pressing ENTER (or carrier return) again.

#### Password Prompting

VSPC prompts for a password by displaying one of the following messages:

```
ENTER PASSWORD
ENTER PASSWORD FOR filename
ENTER NEW PASSWORD FOR filename
```

You enter the password.

During logon, ENTER PASSWORD is always displayed if a logon password is included in your profile.

In other commands, when the required file password is not supplied in the appropriate operand, a prompting message is displayed. It is issued when a file password is being created or changed or when a specific matching file password must be supplied.

Before the command can be executed, the valid password must be entered from the terminal.

If an incorrect matching password is entered, VSPC issues the following message:

```
INCORRECT PASSWORD - RE-ENTER
```

If the valid password is not supplied within three reprompting attempts, VSPC prints the following message:

```
PASSWORD ERROR
```

The command is then rejected.

If a syntactically invalid new or changed password is entered, only one attempt is allowed, and VSPC issues the following message:

```
PASSWORD ERROR
```

The command is then rejected.

## Current Line Pointer

One of the lines in your workspace is considered your current line. (It is highlighted on the 3270 display screen in VIEW mode.) VSPC maintains a pointer to this line. Your current line pointer changes with the commands you issue during your editing session. Commands that use line numbers act on your current line when you enter an asterisk (*) in place of a line number.

As you enter data into your workspace (using the INPUT command or line-numbered entry), the pointer points to the last line entered. The pointer changes with several VSPC commands (for example, COPY, MOVE, and DELETE). You can change the pointer with the LOCATE command.

When you begin an editing session, the current line pointer points to the first line in your workspace. As you issue commands referring to different lines, the current line pointer is changed to the line you're working on.

When you're editing, there will be times you want to find the current line (it may not be displayed on your screen). You can use the LOCATE command to display or relocate your current line.

### *LOCATE Command*

The LOCATE command finds or relocates the current line pointer. It can be relocated by absolute line number, by its display line position relative to the current line, or by its context.

| | | |
|---|---|---|
| LOCATE | *line*<br>+*number*<br>–*number*<br>*begin-line* '*string*'<br>*begin-line:end-line* '*string*'<br>'*string*'<br>* | Optional first operand – specify one or omit. If omitted, current line is assumed. |
| | NOTEXT<br>TEXT | Optional second operand – ignored in VIEW mode; otherwise, if omitted, NOTEXT is assumed (unless TEXT ON has been specified). |

*line*
specifies an absolute line number to which the current line pointer is to be set.

*+number or -number*
specifies a relative number of lines. The current line pointer is set higher or lower by the specified number of lines.

*begin-line* '*string*'
specifies that the search for '*string*' begin with *begin-line* and search to the end of the document.

*begin-line:end-line* '*string*'
specifies a range of absolute line numbers to be searched for '*string*'.

If *end-line* is omitted, the search begins at *begin-line* and goes to the end of the workspace. (The search does *not* then go to the beginning of the workspace and continue to *begin-line*.)

If both *begin-line* and *end-line* are omitted, the search begins with the line after the current line and continues to the end of the workspace.

'*string*'
specifies a set of characters for which VSPC is to search. The current line pointer is then set to the first line found that contains those characters.

*
specifies your current line. It can be used to specify *line, begin-line,* or *end-line* as defined above.

**NOTEXT**
specifies that only the line number of the new current line be displayed.

**TEXT**
specifies that the text of the new current line be displayed at the terminal after the line number.

When the LOCATE command is entered in VIEW mode, both TEXT and NOTEXT are ignored. In this case no explicit response is made but the workspace display area is changed to reflect the new current line. In VIEW mode the current line is displayed and highlighted as the first line on the screen.

If an absolute line number is specified that does not exist, the current line pointer is set to the next lower line number. (If no lower line number exists, the current line pointer is set to the first line of the workspace.)

If LOCATE * is specified or if LOCATE is specified without a first operand, the current line pointer is found and the line number (if NOTEXT) or the line number and text of line (if TEXT) is displayed.

**Examples**

To find your current line:

```
locate
00050
```

or

```
locate * text
00010 BEGIN PROGRAM
```

The response tells you that 00010 is your current line. In VIEW mode this changes the display so that the current line is the first line (highlighted) at the top of the screen.

If you want to search for a certain set of characters:

```
locate 'x+y=z' text
00030 X+Y=Z
```

If *'string'* is specified and cannot be found, the current line pointer is unchanged and a message is issued:

```
LINE NOT FOUND
```

If the request is for a relative line number that exceeds the limits of the workspace, the current line pointer is set to the first line in the workspace if a negative relative line number is specified; or the last line if a positive line number is specified.

## Formatting

VSPC has a logical character-setting feature to help format material.

The computer recognizes a set of characters as tabs, backspaces, and line separators, but these characters are not printable at your terminal. You can set a printable character equal to one of these nonprintable characters to insert any of these three actions into your workspace.

- TABSET sets session tabs, specifies a tab character, or specifies whether or not entered tabs are to be expanded to blanks.
- NEWLINE specifies a character that translates into an nonprintable newline character.
- BACKSPACE specifies a character that translates into an unprintable backspace character.

The following two commands are available for additional formatting.

- TRANSLATE tells VSPC how to interpret each keystroke of your input and what to do with output.
- LINESIZE specifies the maximum length of the line to be displayed at your terminal.

## *TABSET Command*

The TABSET command can:

- communicate session tab positions to VSPC (or remove all tabs).
- translate a specified character into the tab character.
- specify that tab characters be expanded into blanks (or not).
- set the appropriate FORTRAN IV column tabs for fixed-format source entry.

| | | |
|---|---|---|
| **TABSET** | *tablist*<br>**OFF** | Optional operand—choose one or omit. |
| | 'x'<br>' ' | Optional operand—choose one or omit. |
| | **EXPAND**<br>**NOEXPAND** | Optional operand—choose one or omit. |
| | If an operand is omitted, any previous setting is unchanged. If all operands are omitted, the current values are displayed. | |

*tablist*
specifies where session tabs are to be set. The tab list must consist of 1 through 26 integers representing tab positions relative to the left margin. The integers must be written in ascending order with separators between them. The left margin of the terminal is position 1, except for a display terminal in VIEW mode, in which case the input area is indented 6 columns and the assumed columns are as shown in the separator line. When using INPUT NOPROMPT in VIEW mode, position 1 is assumed to be the first character following each logical newline character.

**OFF**
removes all current tab settings.

'*x*'
specifies a character to be translated into the nonprintable tab character. (On a 3270 terminal, the FIELD MARK key is interpreted as the tab character.)

' '
specifies that any existing tab character be removed.

**EXPAND**
specifies that tab characters (or their substitute) be expanded into blanks.

**NOEXPAND**
specifies that tab characters (or their substitute) are not to be expanded into blanks, but are to be passed to VSPC as literal tab characters (X'05')

The column tab settings are used to:

- Speed output to typewriter terminals with hardware tabs.
- Resolve output containing tab characters directed to a screen into an appropriate number of blanks.
- Control expansion of real or substitute tab characters, on input, into strings of blanks if the EXPAND option is in effect.
- Set the hardware tab settings on SDLC 3767 and 3770 terminals.

If you use tabs in your input, you must set them at each session because VSPC does not carry them over from one session to the next (assumes TABSET OFF at logon). Tab settings are not carried with your files. If you attempt to enter tabbed input without setting the session tabs, VSPC will not accept the line.

Tab positions can be changed during a session, but session settings should match the physical terminal settings. If they don't match, the appearance of your output may not match your input. You can display your current tab settings with the QUERY command.

When you have set session tabs (even if you don't use them in your input), VSPC uses them for output in combination with spaces and backspaces to speed up the process. The use of tabs does not change the appearance of the material.

If a tab character is contained in output and either no tab columns are defined or the line position is beyond the last set tab column, then the tab character displays as the logical tab character if one is set, or a blot character if not. Tabs contained in data displayed in the top part of a screen in VIEW mode, or displayed in the command area of a screen as a result of a CHANGE command with only a line number operand, are also displayed as the logical tab character or a blot character.

*Examples:*

```
tabset 5 10 15 20 25 30 35
```

sets session tabs at 5 10 15 20 25 30 and 35.

If your terminal does not have a TAB key (3270, for example), use TABSET to define a substitute character:

```
tabset '%' expand
```

Each time you enter a line with a % character in it, VSPC translates it into the nonprintable tab character and expands it to the number of blanks necessary to reach the next tab position you have specified for your VSPC session. Thus, if you enter

```
00010 amount paid % amount received
```

the data in your workspace will look approximately like this, depending on previous tab settings:

```
00010 AMOUNT PAID     AMOUNT RECEIVED
```

You must be sure your position on the current line is within the range of the tab positions you've set when you use the EXPAND operand. Otherwise, you will get an error message.

```
tab off
```

clears session tabs.

## NEWLINE Command

The NEWLINE command specifies a character that is translated into the nonprintable newline character when entered from your terminal.

| | | |
|---|---|---|
| NEWLINE | 'x'<br>'' | Optional operand–choose one or omit. If omitted, current value is displayed. |

'x'
: specifies a character to be translated into the nonprintable newline character.

''
: this operand (two adjacent single quotation marks) specifies that any currently specified newline character be removed.

After a NEWLINE command specifying a character is executed, a single command line can contain several VSPC commands separated by the newline character, and the commands are executed as if each were entered on a separate line.

If you specify a character as a line separator and then find you need to use that character in your data, you can issue the NEWLINE command with the two single quotation marks. This removes the translation and allows the normal use of the character.

If you are entering data using INPUT mode, the substitute newline character tells VSPC where to begin each new line. If a newline character is entered in a command in a quoted string, it is treated as a literal newline character--not as a line separator. That is, when the character is encountered, the current line is ended and the next line number and a space are generated followed by the data you entered after the newline character.

*Examples:*

```
  newline '$'
```

translates the $ into a nonprintable character that signals the start of a new line.

```
input prompt
  00010 do 10 i=1,5 $ a(i)=b(i)**2+4*i
  00030 +3*b(i) $ 10b(i)=0
```

VSPC treats these lines as if you'd entered:

```
  00010 do 10 i=1,5
  00020 a(i)=b(i)**2+4*i
  00030 +3*b(i)
  00040 10b(i)=0
```

## BACKSPACE Command

The BACKSPACE command lets you define a substitute backspace character, in case your terminal does not have a backspace character key (a 3277 display terminal, for example).

| | | |
|---|---|---|
| BACKSPACE | '*x*'<br>'' | Optional–choose one or omit. If omitted, current value is displayed. |

'x'
specifies a character to be translated into a nonprintable backspace character.

''
this operand (two adjacent single quotation marks) specifies that any currently specified backspace character be removed.

If you specify a character as a backspace and then find you need to use that character in your data, you can issue the backspace command with the two single quotation marks. This removes the translation and allows the normal use of the character.

**Note:** VSPC does not treat the substitution character as a backspace for character correction.

*Example:*

```
backspace'#'
```

translates the # into the nonprintable backspace character.

## *TRANSLATE Command*

You can type your entries in either lowercase letters or uppercase letters. Immediately after you log on, VSPC translates every lowercase letter you type into uppercase before placing it in your workspace.

The TRANSLATE command specifies:

- That a graphics substitution table, defined by the system administrator, is to be used.
- Whether or not terminal input will have lowercase alphabetic letters translated into uppercase letters.

| | | |
|---|---|---|
| **TRANSLATE** | *tablename*<br>**CAPS**<br>**APL**<br>**OFF** | Optional operand—choose one or omit. If omitted, current value is displayed. |

*tablename*
specifies that a table of graphic substitution characters has been defined by the system administrator, and that this *tablename* identifies it.

**CAPS**
specifies that VSPC is to translate lowercase alphabetic characters entered from the terminal into uppercase (capital) alphabetic characters.

**APL**
specifies that VSPC is to assume that the terminal has an APL type element.

**OFF**
specifies that VSPC is to accept both lowercase and uppercase alphabetic characters exactly as they are entered.

At logon, TRANSLATE CAPS is assumed, unless an APL type element was in use at logon time, in which case TRANSLATE APL is assumed.

The TRANSLATE specification remains in effect until another TRANSLATE command is successfully issued, or until the end of the terminal session.

Except for the character-string operands of the CHANGE, FIND, LOCATE, and SPLIT commands, the TRANSLATE command has no effect on VSPC commands—whether entered from the keyboard or within a command list. When commands are entered in lowercase letters, VSPC automatically translates all translatable portions into uppercase letters.

## *LINESIZE Command*

The LINESIZE command specifies the session line length for the terminal.

| | | |
|---|---|---|
| LINESIZE | *n* | Optional first operand–specify or omit. (Required if other operand is specified.) |
| | *x* | Optional second CPT-TWX operand–specify or omit. If omitted and first operand is specified, idles specification unchanged. |
| | If both operands are omitted, current linesize is displayed. | |

*n*

specifies the number of character positions to be used for a display line at the terminal; it must be an integer with a value from 18 through 255 inclusive, except for terminals that allow paper tape input and output, for which *n* may have a value from 18 through 32767, inclusive.

*x*

specifies the number of idle characters to be used following each carrier return on a CPT-TWX terminal. It must be between 0 and 100. Idle characters may be necessary to ensure enough time for an actual carrier return on the CPT-TWX terminal. If none is specified, zero is assumed.

The value of *n* remains in effect during a terminal session until changed by another LINESIZE command (or by an equivalent interpreter command).

At logon time, the assumed line length (in character positions) for the terminals is as follows:

| Terminal | Line Length |
|---|---|
| 3270 | 80 |
| 3767 SDLC | 132 |
| 3767 S/S, 2741, 1050 | 120 |
| 3770 | 132 |
| CPT-TWX | 72 |

Physical margins are not affected by the LINESIZE command, nor is the allowable length of terminal entries.

**Note:** Care must be used in specifying the LINESIZE command. If the session line length specified is longer than the terminal's physical line length, and if lines of data are also longer than the physical line length, some of the terminal output will be lost during line display.

## Questioning Your Status

Information is available to you about your tab settings, the current name and size of your workspace, your workspace attribute, your user number, type of library access, the length of time you've been connected to the system, and which translate table you are using.

## *QUERY Command*

There may be times when you need information about your current terminal settings or about information that's in your user profile—if, for example, you get called away from the terminal and now that you're resuming work, you're unsure of your tab settings and translation convention.

The VSPC QUERY command lets you ask for this information.

Using the QUERY command, you can also ask for accounting information and workspace status information. The "Program Development" section of this publication documents the queries of file status and VSPC library directories.

| | | |
|---|---|---|
| **QUERY** | **TABSET**<br>**TRANSLATE**<br>**ACCOUNT**<br>**PROFILE**<br>**WORKSPACE** | Optional—choose one or omit. If omitted, WORKSPACE is assumed. |

**TABSET**
requests that the current session tab settings be displayed. If none are in effect, an informative message is displayed.

**TRANSLATE**
requests that the current translation convention—*tablename,* CAPS, or OFF—be displayed.

**ACCOUNT**
requests that accounting information be displayed either as separate items or as composite processing units depending on how your installation has your accounting procedures defined.

**PROFILE**
requests that your profile be displayed, as follows:

USERNUM: *usernum*—is your user number.

LIBTYPE: *type*—specifies whether your library is PRIVATE, PROJECT, or PUBLIC (see "Libraries").

CONTENT: *attribute*—specifies your default workspace attribute (which may be different from that of the current workspace). The *attribute* can be DATA, CLIST, FORTRAN, XFORTRAN, FLINK, one of the other IBM program products installed with VSPC, or a user-written compiler or interpreter.

CPUMAX: *nnn* SECONDS—specifies, in seconds, the maximum CPU (central processing unit) time that can elapse between terminal inputs to a running program. The default is zero which indicates that there is no limit. Omitted if CPUMAX is zero.

SPACE: *nnn*—specifies the maximum number of characters your files can occupy in any library.

SIZE *mmm* EDITABLE, *nnn* OBJECT—specifies the maximum workspace sizes as follows: *mmm* characters for editable data (DATA, CLIST, FORTRAN, XFORTRAN, FLINK, or other compiler source programs), *nnn* characters for object programs.

INTPRMAX: *mmm* MAXIMUM, *nnn* DEFAULT—specifies the maximum size (*mmm* characters) and default size (*nnn* characters) for an interpreter workspace.

SSMAX: *mmm* CHARS, *nnn* ITEMS—specifies the maximum size (*mmm* characters) for a single data item in shared storage, and the maximum number (*nnn*) of items the user may make known to the shared storage manager at one time.

JECODE: *code*—When this entry is present, you are allowed to submit jobs for batch processing through the job entry commands. It specifies the first six characters of the jobname that identifies you to the VSPC job entry facility. It is omitted if you do not have job entry privileges.

PRIVILEGE: *privilege*—specifies whether or not you can use the privileged VSPC commands. ADM and ACTADM specify that you are allowed to use those commands. It is omitted if you do not have a privileged status.

PROJLIB *(libnum)*—specifies the project library to which you have access. It is omitted if you do not have access to a project library.

ACCOUNT: '(A16729,707-A23)' JOB—the accounting number your installation has assigned to you. If the word JOB appears, this number is inserted as the accounting information in the JOB statement of any job submitted by you with a SUBMIT command.

The following is an example of a listing you could receive after specifying QUERY PRO:

```
USERNUM: 12345
LIBTYPE: PRIVATE
CONTENT: FORTRAN
CPUMAX: 5 SECONDS
SPACE: 500,000
SIZE: 40,000 EDITABLE,  80,000 OBJECT
INTPRMAX: 60,000 MAXIMUM,  30,000 DEFAULT
SSMAX: 0 CHARS, 0 ITEMS
JECODE: DEPT87
PROJLIB: 22222
ACCOUNT: '(A16729,707-A23)'JOB
```

**Note:** If the VSPC administrator alters your profile after you've logged on, the QUERY PROFILE command displays the altered values; these values, however, do not take effect until the next time you log on.

**WORKSPACE**
requests that information about the workspace be displayed as follows:

NAME *filename*—the filename currently identifying the workspace. The libnum field in *filename* is displayed only if it is different from your library number (see "Filenames").

CONTENT *attribute*—the current attribute of the workspace contents.

LENGTH *nnn* CHARS—the number of data characters contained in the workspace.

*nnn* LINES—the number of lines contained in the workspace.

HIGHEST LINE NUMBER *n*—the line number of the last line in the workspace.

For example:

```
NAME: 3400 PROGLIST
CONTENT: XFORTRAN
LENGTH: 922 CHARS, 50 LINES
HIGHEST LINE NUMBER: 500
```

When the QUERY command is executed, the requested information is displayed at the terminal in the format indicated. When no operand is specified, WORKSPACE is assumed.

## Message Facilities

Usually during a terminal session you can send messages to other VSPC terminal users and (when you're not entering input or receiving output) receive messages from them as well. In addition, you can specify that you want to keep your terminal continuously ready to receive messages, or that you want to prevent messages from being routed to your terminal.

### *SEND Command*

The SEND command transmits messages from the terminal to a specific destination.

| | | |
|---|---|---|
| **SEND** | '*message-text*' | Required first operand. |
| | **CONSOLE**<br>**OPERATOR**<br>*usernum* | Optional—choose one or omit. If omitted, message is sent to OPERATOR. |

'*message-text*'
represents the message to be transmitted; it can be made up of any combination of the following characters:

the 26 alphabetic characters (upper- and lowercase)
the 10 numeric characters 0 through 9
blanks or spaces / + – * . , = ( ) ' : ; ?

enclosed in single quotation marks. If the alphabetic characters are entered in lowercase letters, they are translated into uppercase letters, even if TRANSLATE OFF is in effect.

If the apostrophe or single quotation mark (') is contained within the message text, it must be specified as two adjacent characters (''). In the message text, the adjacent characters are considered to be one.

*'message-text'* can be 1 through 64 characters long.

**CONSOLE, OPERATOR,** or *usernum*
when specified, identify the message destination.

**CONSOLE**
specifies that the message is to be sent to the operating system console.

**OPERATOR**
specifies that the message is to be sent to the VSPC operator. If the destination operand is omitted, the message is sent to the VSPC operator.

*usernum*
specifies that the message is to be sent to the user having that user number.

When *usernum* is specified, SEND command execution is unsuccessful if the receiving user is:

- In MESSAGE BLOCK mode.
- Not logged on.

In either case, VSPC returns a message to you.

**Note:** If the message is not displayed at the receiving user's terminal before he logs off, the message is purged.

## *MESSAGE Command*

The MESSAGE command specifies whether or not system (VSPC and compiler) messages are to be displayed at the terminal and/or whether or not message headers are to be displayed.

| | | |
|---|---|---|
| MESSAGE | ID<br>NOID | Optional—choose one or omit. If omitted, present display convention is unchanged. |
| | BLOCK<br>OPEN<br>WAIT | Optional—choose one or omit. If omitted, present message mode is unchanged. |
| | If all operands are omitted, current value is displayed. | |

**ID and NOID**
specify the display convention for message headers. (A unique identifying header precedes each VSPC and VSPC FORTRAN message; the header form for VSPC messages is ASU*nnn* and, for VSPC FORTRAN, the header form is AFP*nnn*, where *nnn* is a 3-digit number.)

**ID**
specifies that VSPC and VSPC FORTRAN message headers are to be displayed with each message.

**NOID**
specifies that VSPC and VSPC FORTRAN message headers are to be omitted—only the message text is to be displayed.

When both are omitted, the current display convention for message headers is unchanged.

**BLOCK, OPEN, and WAIT**
specify the message mode in which the terminal is to operate.

**BLOCK**
prevents display of unsolicited messages at the terminal. VSPC responses to commands are not suppressed.

**OPEN**
specifies normal terminal operation—that is, messages can be received whenever the terminal is not open for input. Terminal input can be entered without a previous attention signal.

**WAIT**
specifies that the terminal is continuously ready to receive messages. In this mode, an attention signal must precede any input from the terminal.

When BLOCK, WAIT, and OPEN are omitted, the present message mode is unchanged.

At logon time, the NOID and OPEN operands are in effect.

**Note:** If the MESSAGE mode is changed from BLOCK to OPEN, the most recent general message from the VSPC operator (if any was sent after MESSAGE BLOCK went into effect) is displayed at the terminal.

## Ending a Session

To end a terminal session, enter the OFF command.

### *OFF Command*

The OFF command ends a terminal session.

| | | |
|---|---|---|
| **OFF** | **CONTINUE** | Optional—specify or omit. If omitted, workspace contents not saved in CONTINUE file. |

**CONTINUE**
specifies that the current work is to be saved. If the workspace is password protected, the work is saved with this password.

When CONTINUE is omitted, the current workspace, and any existing CONTINUE file, is deleted when the OFF command is executed.

When the OFF command is successfully executed, the current terminal session is ended, and the logoff messages are displayed.

Another logon request can usually be entered. If it cannot, VSPC is shutting down.

If a new CONTINUE file is not being saved, the OFF command causes any previous CONTINUE file in the user's library to be automatically purged. If CONTINUE is specified, the previous CONTINUE file (if any) is replaced when the new CONTINUE file is saved successfully.

However, if the new CONTINUE file cannot be saved (for example, if your workspace is empty), the previous CONTINUE file—if any—remains in your library. When this happens, the following message is displayed at your terminal:

```
ASU228 UNABLE TO SAVE 'CONTINUE'
```

## *Saving Your Work*

There can be times when your terminal session ends unexpectedly; there may be a communication line failure, a power failure, or the system administrator may be halting VSPC. When this happens, VSPC tries to save your current work for you in the CONTINUE file.

When a system administrator halts VSPC, VSPC tells you whether or not it was able to save your work, with ending messages like the following:

```
14:22:00 06/30/76 12345
CONNECTED 01:30:30
CPU TIME: 12
```

This series of messages means your work was saved. If VSPC couldn't save your work, you'll receive one additional message:

```
UNABLE TO SAVE 'CONTINUE'
```

VSPC may be able to save your work, even if you haven't received these messages (if, for example, a communication line failed and your terminal was disconnected). If VSPC saved your work, then the next time you log on to VSPC, the VSPC response could be:

```
DATA 16:22:22 06/03/76 12345
CONTINUE CLIST PASSWORD
WELCOME TO VSPC
READY
```

The message CONTINUE CLIST PASSWORD tells you that your work has been saved, that it has the attribute CLIST, and that you specified a password for it at some previous time. You can now retrieve the work from CONTINUE—supplying the correct password—and then go on working with it. (When you bring the work into your workspace, your attribute is implicitly changed to CLIST.)

## CONDUCTING A 3270 TERMINAL SESSION

Conducting a session on a 3270 Display System differs from the method used for nondisplay terminals. But to use VSPC on any terminal, you must first contact the computer, then log on to VSPC before you can type data and commands and use the VSPC editing facilities.

Entering and editing data or commands in VSPC can be handled in two distinctly different ways on 3270 terminals.

When you log on to VSPC your entries (commands and data) appear at the bottom of the screen in the input area (see Figure 9). When you press ENTER, these lines appear in the output area. You can edit lines in the input area with the general editing procedures described in this section. In this book this screen format and method of input and editing is referred to as "command mode."

When you issue the VIEW command, your screen format changes to allow direct editing of lines in your workspace. This book refers to this screen format and method of input and editing as "VIEW mode."

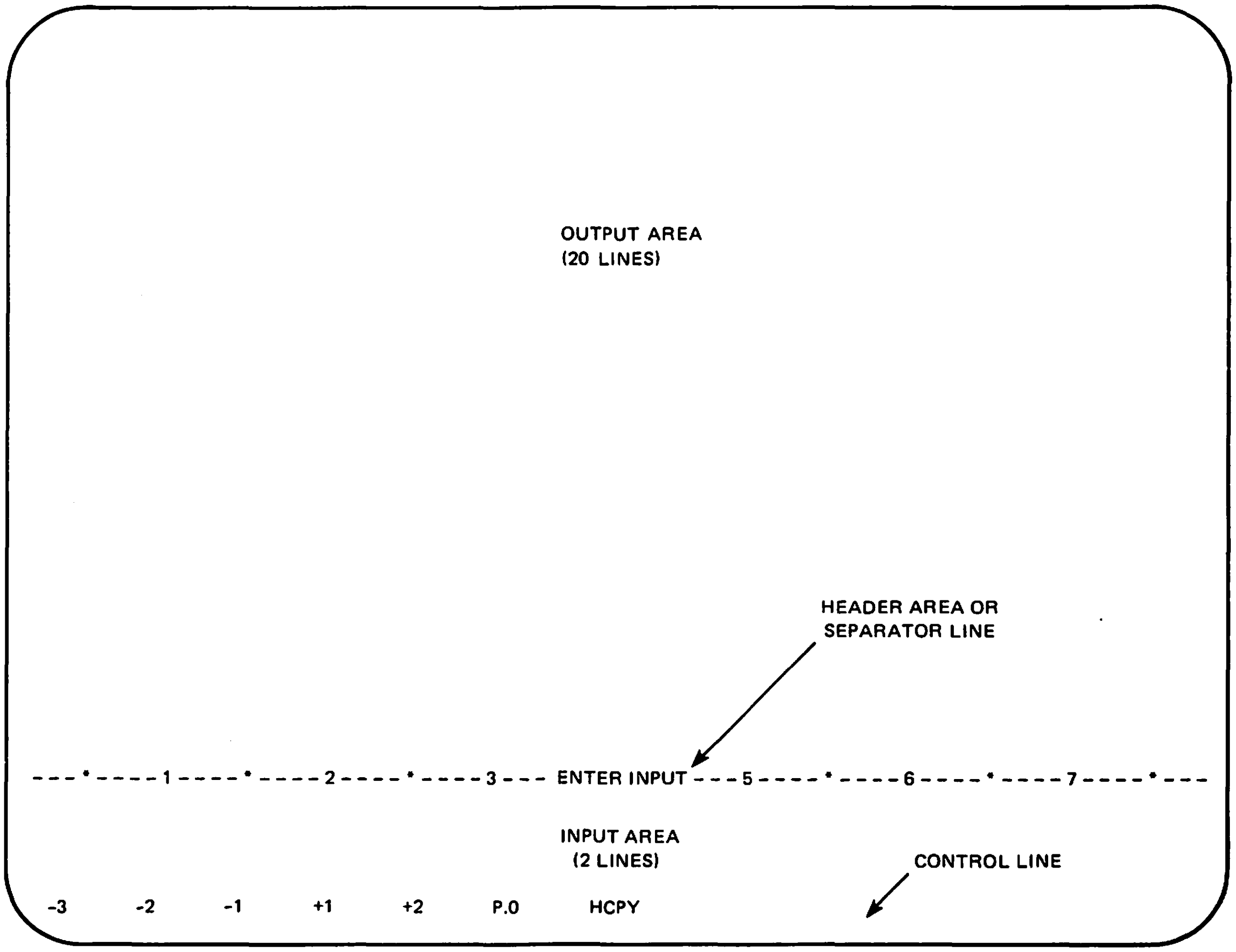


Figure 9. IBM 3270 VSPC Screen Format in Command Mode

## Contacting the Computer

An IBM 3270 can be locally connected to a computer system through a channel, or remotely connected through a nondial (leased) line. These instructions assume a display station equipped with the IBM Line Adapter feature. If your station is not so equipped, check with your VSPC administrator about how to establish a connection using the equipment at your location.

Pull out the switch labeled OFF-PUSH to apply power to your display station. The SYSTEM AVAILABLE light turns on and the cursor appears in the upper-left portion of the screen.

### *VSPC Command (Logon)*

After establishing contact with the computer, press the CLEAR key and enter your logon command as follows (no abbreviation):

| | |
|---|---|
| VSPC | ID= *usernum* |

**ID=** *usernum*
identifies you to VSPC. This number is assigned by your VSPC administrator. A usernum contains from 1 to 7 digits; a value of zero (0) is not a valid user number.

Press the ENTER key.

**Note:** For certain 3270 Display Systems that use the SDLC line discipline, the TEST REQ key may be required instead of ENTER. If your IBM 3277 is connected via a 3791, the procedure is also different. Consult *VSPC Terminals* or check with your VSPC administrator.

If your logon is not successful, you will receive a message stating the reason. You then must repeat the logon command, correcting the indicated problem. If you receive no response after typing your logon request, VSPC is not available.

If your logon is successful, the VSPC screen format appears as in Figure 9. If your installation requires a password, VSPC prompts:

```
ENTER PASSWORD
```

When you type your password it does not display. If the password you provide is not accepted, VSPC repeats the prompt and you can enter another password. If you have not provided the proper password within the number of prompts allowed (four) or within the allotted time (two minutes), you are disconnected from VSPC.

When you have successfully logged on (and typed a valid password), the following messages are displayed:

```
attribute time date usernum
[CONTINUE attribute[PASSWORD]]
[logon-message-text]
READY
```

- attribute designates the content classification of your workspace as specified in your profile. If none is specified,your workspace attribute defaults to DATA.

- time date displays the time and date you logged on.
- usernum states your user identification number.
- CONTINUE (when present) indicates that the contents of your workspace at the end of your last terminal session have been saved and are available again with a LOAD CONTINUE.
- attribute (in the second line) is the attribute of the CONTINUE workspace.
- PASSWORD is displayed if CONTINUE is protected with a password, meaning you will need to supply one when you load the CONTINUE workspace.
- logon-message-text relays information from your VSPC operator.
- READY indicates VSPC is ready to accept input.

If the workspace attribute specified is the one you want to use, you can begin making entries. Otherwise, with the ENTER command, you can change your workspace attribute. See *VS Personal Computing (VSPC) General User's Guide and Command Language* for additional information.

### Example of 3270 Logon

You contact VSPC:

```
vspc id=1234567
```

VSPC prompts for your password:

```
ENTER PASSWORD
```

(You enter your password, but it does not display.) After you type your password and press ENTER, VSPC sends you some messages:

```
FORTRAN  02/04/78  09:16:20  1234567
CONTINUE FORTRAN
WELCOME TO VSPC
READY
```

The READY response is standard after most successfully executed commands and indicates that VSPC is ready for your next entry. An explanatory error message is issued to your terminal if VSPC is unable to execute your command.

## General Editing

The VSPC screen format shown in Figure 9 is in effect when you log on to VSPC. When you type at the keyboard, your data or commands appear on the screen in the input area on the lower part of the screen. As you enter more lines and as VSPC responds to you, the entire dialog appears in the output area. On the control line you specify terminal actions–either by use of an optional light pen or by cursor positioning. The header area tells you what kind of processing you are currently using.

With this method you tell VSPC what to do from your terminal, and VSPC responds. Responses from VSPC vary depending on the type of terminal and whether you are typing lines of data or running a program. This interaction may vary on some 3270 terminals; this is the general procedure.

1. VSPC notifies you in one or more of the following ways that it is ready to accept your input:
   - Displays a question mark (?)
   - Displays a message (for example, READY)
   - Displays a line number (for example, 00010)
   - Displays ENTER INPUT in the 3270 Header Area
2. You type a line of input as follows:
   a. Type the line of data or a command.
   b. Press ENTER.
   c. Wait for VSPC to respond in one or more of the above ways before you type another line.

You can use all of the terminal editing facilities (INS MODE, DEL, etc.) to modify the lines displayed in the input area as described below.

### *Character Correction*

Move the cursor back to the incorrect character with the cursor control key; then type the correct character.

### *Character Insertion*

Press the INS MODE key and move the cursor to the position in which the additional character is to be inserted. Type the missing character. As characters are inserted, all characters to the right of the cursor are shifted to the right. If the insertion causes the line to exceed the line limit, the excess characters shift from the end of the first line of your input line to the beginning of the next. The INPUT INHIBITED indicator turns on and your keyboard is disabled if you try to insert more characters than the area will hold.

Press the RESET key to return the keyboard to its normal mode of operation and to turn off the INSERT MODE indicator.

### *Character Deletion*

Move the cursor to the position of the extra character and press the DEL key. The characters to the right of it move left one space.

### *Line Correction*

With the backspace key, return the cursor to the first character of the data you want to replace. Then press the ERASE EOF key. This erases all characters from the cursor position to the end of the line. Retype the line.

### *Special Requests*

Three terminal conditions allow you to make special requests.

1. When the MORE...PRESS CLEAR message is displayed on the screen, in addition to pressing CLEAR to continue output, you can:
   - Press PA1 to signal attention
   - Request backpaging
   - Request hardcopy
   - Press PA2 to cancel output
   - Press ENTER to hold the display longer than the automatic change time of 30 seconds. The message changes to HELD...PRESS CLEAR and any of the above actions may be taken before pressing CLEAR to continue output.
2. When ENTER INPUT appears on the screen, in addition to entering input, you can:
   - Press PA1 to escape from input
   - Request backpaging
   - Request hardcopy
3. When the terminal is idle, you can:
   - Press PA1 to signal attention
   - Request backpaging
   - Request hardcopy

### *Display Screen Hold*

When your output needs more than the remaining space available on the screen to display, VSPC pauses at the end of the screen with MORE...PRESS CLEAR shown in the screen header area. Pressing CLEAR displays the next page.

If you press neither CLEAR nor ENTER within 30 seconds, the next page is displayed automatically.

If, while MORE...PRESS CLEAR is showing, you press the ENTER key, the message changes to HELD...PRESS CLEAR and the current display remains until you press the CLEAR key.

### *Invalid Output Characters*

If you find a double quotation mark (") in your output that you didn't expect, VSPC has found an invalid character in your data.

## Program Function (PF) Keys

The program function keys provide shortcuts to entering commands, sets of characters, or control functions. You can assign a command, a set of characters, or a control function to any PF key on your terminal with the PFKEY command. They are interpreted the same way in both screen formats with one exception: If the FORWARD or BACKWARD option of the PFKEY command is used in VIEW mode, the workspace display area is moved instead of normal paging.

## *PFKEY Command*

The PFKEY command specifies the interpretation of a program function (PF) key on the 3270 terminal.

| | | |
|---|---|---|
| PFKEY | *n* | Optional first operand–specify or omit. (Required if other operand is specified.) |
| | ' '<br>'*string*'<br>'*string*' ENTER<br>'*string*' NOENTER<br>'*string*' COMMAND<br>'*string*' COMMAND ENTER<br>'*string*' COMMAND NOENTER<br>COMMAND<br>COMMAND ENTER<br>COMMAND NOENTER<br>LAST<br>LAST ENTER<br>LAST NOENTER<br>HCPY<br>FORWARD (x)<br>BACKWARD (x) | Optional second operand–choose one or omit. (Required if other operand is specified.) |
| | If all operands are omitted, current values are displayed. | |

*n*
specifies the number of the PF key to which you want to assign the interpretation. It must be between 1 and 24 inclusive.

' '
this operand (two adjacent single quotation marks) specifies that any currently specified PF key definition be removed.

'*string*'
specifies a string of characters to be inserted into the screen image at the location of the cursor when the selected PF key is pressed. (The action is as if the INS MODE key is pressed, the characters typed on the keyboard, and then the RESET key is pressed). The cursor is relocated at the character position following the inserted string.

The maximum length of a string is 255 characters. The maximum total length of character strings assigned to PF keys is 600 characters.

COMMAND
specifies that the cursor is to be logically moved to the beginning of the command area before performing the function specified for the key. If COMMAND is not specified, the function (such as string insertion) is performed at the current location of the cursor on the screen.

**LAST**
specifies that when the PF key is pressed, the contents of the last previous terminal input line are inserted into the command area at the current location of the cursor. The cursor is repositioned following the inserted characters.

**ENTER**
specifies that the string be copied to the screen as described above, and then the appropriate action taken as if the ENTER key had been pressed.

**NOENTER**
specifies that the string be copied to the screen but no other action taken. You can then modify it before pressing ENTER. Your line is truncated if the insert extends beyond the current input field.

If neither ENTER nor NOENTER is specified, ENTER is assumed.

**HCPY**
specifies that subsequent use of the named PF key should simulate selecting the HCPY field on the screen through cursor positioning and the ENTER key.

**FORWARD (x)**
specifies that subsequent use of the named PF key should page the display forward *x* pages. The number of pages may be any number between 1 and 99.

**BACKWARD (x)**
specifies that subsequent use of the named PF key should page the display backward *x* pages. The number of pages may be any number between 1 and 99.

*Examples:*

```
pfkey 4 'go to 40'
```

sets PF key number 4 so that every time you press it, the character string 'go to 40' is inserted at the location of the cursor.

```
pfkey 3 forward (1)
```

pages your screen forward 1 screen every time you press PF3.

## VIEW Mode

When you issue the VIEW command, your screen is considered to be in VIEW mode, and the format shown in Figure 10 appears on your screen. Your active workspace is displayed with line numbers in the 20-line workspace display area. You can edit directly with the general editing procedures described above.

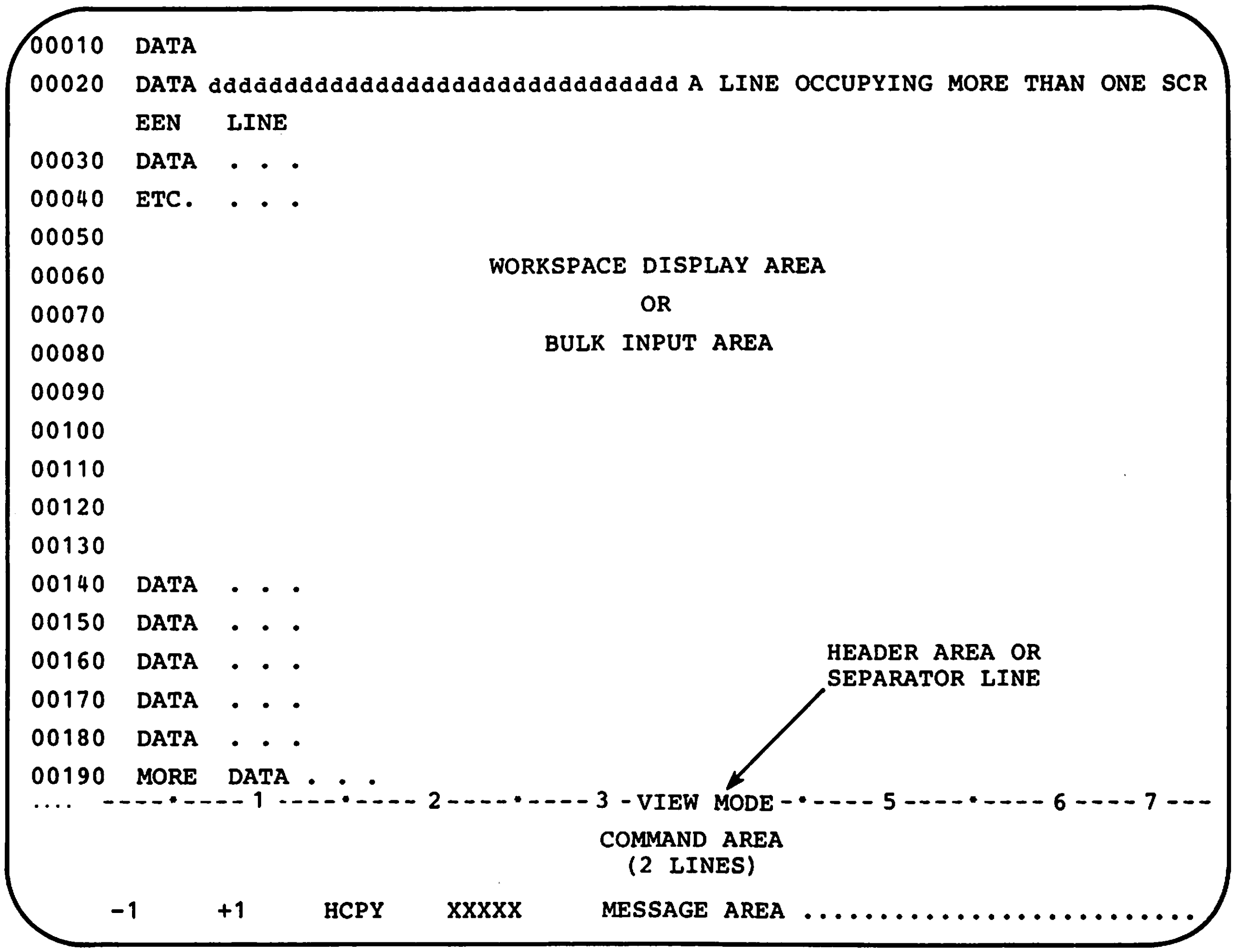
The separator line (or header area) indicates whether you are in VIEW mode, INPUT mode, or another mode.

You enter VSPC commands in the command area.

The control line contains a message area to display messages and responses to your commands. If a message is too long to fit, it is displayed in the workspace display area. When this happens, press CLEAR and your current screen is restored. You can point to the other items on the control line with the cursor, CURSR SEL key, or light pen.

- +1 or -1 moves your display one page forward or backward across your workspace without changing the current line pointer.
- HCPY requests the current display be printed. (You must have selected a printer with the HARDCOPY command.)
- *xxxxx* displays your current line number. If you point to it with cursor, CURSR SEL, or light pen, your current line is moved to the top of the screen.

**Note:** If you use the light pen or CURSR SEL key to select fields and cause a change in the display screen, any changes you've made while in VIEW mode since you last pressed ENTER are lost. Use PF keys or cursor positioning and ENTER, to be sure your changes are saved in your workspace.



```
00010  DATA
00020  DATA ddddddddddddddddddddddddddddddd A LINE OCCUPYING MORE THAN ONE SCR
       EEN   LINE
00030  DATA  . . .
00040  ETC.  . . .
00050
00060                        WORKSPACE DISPLAY AREA
00070                                  OR
00080                           BULK INPUT AREA
00090
00100
00110
00120
00130
00140  DATA  . . .
00150  DATA  . . .
00160  DATA  . . .                                      HEADER AREA OR
00170  DATA  . . .                                      SEPARATOR LINE
00180  DATA  . . .
00190  MORE  DATA . . .
....  ----•----1----•----2----•----3 -VIEW MODE-•----5----•----6----7---
                                COMMAND AREA
                                 (2 LINES)

    -1      +1      HCPY     XXXXX      MESSAGE AREA ........................
```

Figure 10. IBM 3270 VSPC Screen Format in VIEW Mode

## *VIEW Command*

The VIEW command allows you to use the full screen editing facilities of VSPC. After issuing this command, your screen is considered to be in VIEW mode and will change from that in Figure ~~6~~ 9 to that in Figure 10. To get out of VIEW mode, press the ENTER key when you have made no changes to the screen. Certain VSPC commands (RUN, STORE, or ENTER) also take you out of VIEW mode (change your screen format back to that shown in Figure 9).

If you are already in VIEW mode and issue the VIEW command, a different part of the workspace is displayed without altering the current line pointer.

| | | |
|---|---|---|
| VIEW | *line*<br>*<br>*+line*<br>*–line* | Optional – choose one or omit. If omitted, current line is displayed at top of screen. |

*line*
: specifies an absolute line number. The number and the text of the line will be displayed at the top of the output area. The current line pointer is not changed.

*
: specifies that your current line be displayed at the top of the screen.

*+line* or *–line*
: specifies the relative number of lines the display is to be moved. +line moves the display toward the end of the workspace; –line toward the beginning.

If the VIEW command is issued with no operand while already in VIEW mode, the screen is displayed with the current line at the top of the workspace display area.

In the VIEW mode, any command that changes the contents of the workspace (for example, RENUMBER, MOVE, DELETE) updates the display. Any command that moves the current line pointer to a line not displayed on the screen redisplays the workspace with the new current line at the top of the screen.

Responses to commands you issue in VIEW mode are displayed on the control line if they fit. If not, your screen is taken out of VIEW mode and the VSPC screen format shown in Figure 9 is displayed with the response as the last line of the output area and MORE . . . PRESS CLEAR in the separator line. In this case, press the CLEAR key to return to VIEW mode.

*Examples:*

```
view
```

changes the screen format from that in Figure 6 to that in Figure 10. You are now in VIEW mode and can use the full screen editing facilities.

```
view 30
```

puts your screen in VIEW mode with line 30 at the top of the screen.

If you press ENTER with no changes on the screen, your screen is taken out of VIEW mode and you can page back to see the VSPC commands you've issued and their responses.

## *LOCAL Commands*

Local commands are used only in VIEW mode. They are entered over a line number displayed in the output area starting in the left margin of the screen. Any characters after the command are ignored. These commands usually specify an action to be taken on the line over whose number they are entered. The actual number of the line is normally not changed by the entry of the command, and is restored when the screen is redisplayed following execution of the command.

The form of a local command is:

*nnnnX*

*nnnn*
: Is any number of lines (up to four digits) you want to add, delete, or repeat.

*X*
: is the command character: A, D, R, or /.

A–Add lines

The specified number of blank lines is added to the workspace after the selected line. The cursor is positioned at the start of the first of these lines. Line numbers for the added lines are generated in increments of one beginning with the line number of the selected line plus one. If there are insufficient line numbers available before the next existing line, lines are renumbered in increments of one as far as is necessary. If this occurs, a warning message is displayed in the message area of the control line. The added lines have a length of one character unless the workspace contains a 'DIRECT' file, in which case the length is the same as that of the selected line.

D–Delete lines

The specified number of lines is deleted from the workspace beginning with the selected line. The lines being deleted are deleted without inspection; any local commands entered on them are ignored.

R–Repeat lines

The specified number of lines is added after the selected line with identical text. Line numbers for the added lines are generated in the same manner as for the add lines command (A).

/–Change current line pointer

The selected line becomes the current line of the workspace. If this command is entered more than once on the same screen, the indicated line nearest the bottom of the screen becomes the current line.

You may enter multiple local commands on the screen before you press the ENTER or a PF key. The commands take effect in a sequence determined by their physical position on the screen, starting at the top.

*Examples:*

To make line 00050 instead of 00060 the current line, position the cursor under any digit of line 00050 and type the slash:

```
00040 data A
/0050 data B
00060 data C
```

Then, when you press ENTER, line 00050 becomes your current line.

```
00040 data A
00050 data B
00060 data C
```

You can add lines in the workspace:

```
00040 data A
4A050 data B
00060 data C
```

VSPC adds four blank lines after line 00050:

```
00040 data A
00050 data B
00051
00052
00053
00054
00060 data C
```

## *CLEAR, ENTER, and Program Access Keys*

While in VIEW mode you can move your cursor around the screen, type over data, insert characters (INS MODE), and delete characters (DEL) to modify the display of your workspace.

The CLEAR, ENTER, and Program Access (PA) keys are interpreted in a special way in VIEW mode.

### CLEAR Key

CLEAR nullifies changes made to the displayed text and displays the screen as it was before you made the changes.

Another use of CLEAR is to exit from 'message mode.' When a message or a response resulting from a command occupies multiple lines or is too long to be displayed on the message portion of the control line, enter 'message mode.' VSPC displays the command mode editing screen (Figure 9) with the message as the last line in the output area, and MORE ... PRESS CLEAR in the separator line. Press the CLEAR key to return to VIEW mode.

### ENTER Key

Press the ENTER key to signal that all your changes are to be entered into the workspace. If ENTER is used with no modifications made to the display, you are returned to Command mode editing.

### Program Access (PA) Keys

These program access keys are ignored in VIEW mode. While in message mode, PA2 may be used to cancel output.

## *Order of Processing*

The order in which VSPC processes requests depends on the way in which you enter them:

1. If you use the light pen to select control-line fields, nothing else on the screen is processed.

   For example, if you use the light pen to select HCPY, the copy printed is of your screen *before* you last pressed ENTER. The changes shown on the screen are entered into the workspace the next time you press the ENTER key.

2. If you use a PF key defined with a PFKEY NOENTER command to set a character-string, nothing is changed in the workspace until you press ENTER.

   If you use a PF key defined with a PFKEY ENTER command to set a character-string, it's as if you placed the character-string into the screen position of the cursor and then pressed the ENTER key.

3. If you request HCPY using a preset PFKEY or using cursor positioning followed by ENTER, changes on the screen are made in the workspace; then the screen is printed incorporating the changes.

4. When you press ENTER, changes you've made are placed in the workspace in the following order:

   | (1) Line changes and D (delete) local commands, beginning at the top of the screen and continuing to the bottom.

(2) Then A (add) and R (repeat) local commands, beginning at the bottom of the screen and continuing to the top, and the bottom-most of any / (set current line pointer) commands.

(3) Forward or backward paging requested by placing the cursor in the control line and pressing ENTER.

(4) A PFKEY preset to FORWARD or BACKWARD.

(5) General commands entered in the command area.

You can do the following at any time during the processing:

- If you press the PA1 key to signal attention, nothing else on the screen is processed.
- If you press the PA2 key while MORE...PRESS CLEAR is displayed, your output is canceled.
- If you press CLEAR, any screen changes you've made are canceled, and the screen is redisplayed in its original form.

**Cursor Positioning**

After processing the modifications to the screen, the new screen is formatted for the next display. When this new screen is displayed, the cursor is positioned at the start of the command area unless any of the following apply:

- You used a program function key with the NOENTER option to insert a set of characters. In this case the cursor is positioned after the string associated with the PF key.
- You issued the 'A' or 'R' local command. In this case the cursor is positioned at the start of the added lines. However, if another command entered on the same screen causes the workspace display to change such that the lines are not displayed, the cursor is at the start of the command area.

  If you enter multiple 'A' and/or 'R' commands on the same screen, the cursor is set to the position corresponding to the command nearest the top of the screen.
- You issue the INPUT command. In this case the cursor is positioned at the start of the area for adding new text.

## *HARDCOPY Command*

For IBM 3270 Information Display Systems, the HARDCOPY command specifies which printer you want to use.

| | | |
|---|---|---|
| HARDCOPY | *node-name*<br>*<br>OFF | Optional operand—choose one or omit. If omitted, current value is displayed. |

*node-name*
requests the use of a separate 3270 Display System printer. The *node-name* is predefined as part of the VTAM network definition.

*

**indicates that a 3284 printer is directly attached to this 3275 Display Station, and that this printer is to be used.**

**OFF**

specifies that the printer designated by the *node-name* currently in use is to be disconnected.

The HARDCOPY command must be executed before printing operations can be requested.

The * and *node-name* operands are used to request the use of a printer. If the indicated printer is available, VSPC responds with the following message:

```
READY
```

Requests for printing operations can then be entered.

Printing of the currently displayed page is requested by positioning the cursor at the HCPY field on the control line and pressing the ENTER or CURSR SEL key, by use of a program function (PF) key predefined as HCPY, or, if available, by selecting the HCPY field with the light pen. The current page is then printed.

If the indicated printer cannot be obtained, VSPC responds with the following message:

```
NOT AVAILABLE
```

In this case, the HARDCOPY command cannot successfully be executed at this time.

The OFF operand is specified when the user wishes to release the previously obtained printer. When execution is successful, VSPC responds with the following message:

```
READY
```

The printer previously selected is no longer available for use.

*Examples:*

If your 3270 display terminal has an associated printer, you can request that printer services be made available to you through the HARDCOPY command.

hardcopy *

On the 3275 terminal, this command specifies that output is to be sent to the attached 3284 printer.

A 3270 Information Display System printer that is not physically attached to your 3270 display terminal, but is shared by several users, may be associated with your display terminal by entering:

hardcopy *node-name*

where *node-name* is a name specified by your installation to identify that particular printer.

Later, when you want to print the current page, you can do so with the HCPY field in the control line.

When you no longer need the printer, specify

har off

This command disconnects the printer.

# CONDUCTING A 2741 OR 3767 START-STOP TERMINAL SESSION

On any terminal you must first contact the computer, then log on to VSPC before you can type data and commands and use the VSPC editing facilities.

## Contacting the Computer

Before you can log on to VSPC you must make contact with the computer. The procedure you use varies depending on whether the connection is through a leased line or a switched line. Your VSPC administrator will tell you which type of connection you're using.

If your connection is by a leased line, you can enter the VSPC logon command as soon as you turn the power switch on.

If your connection is by a switched (telephone) line, then additional steps such as are shown in Figure 11 are necessary.

| Telephone Data Set | Acoustic Coupler |
|---|---|
| 1. Press TALK button. | 1. Make sure coupler is: connected to power, turned off, connected to terminal. |
| 2. Remove handset from cradle; dial telephone number supplied by your system administrator. | 2. Remove handset from cradle; dial telephone number supplied by your system administrator. |
| 3. When you hear high-pitched tone, go on to Step 4. If line is busy or there is no answer, hang up and re-dial. | 3. When you hear high-pitched tone, go on to Step 4. If line is busy or there is no answer, hang up and re-dial. |
| 4. Press DATA button. DATA light should go on and keyboard unlock. | 4. Place handset face down in coupler box; make sure cord is in slot. Close and latch coupler lid. |
| 5. Place handset in cradle. | 5. Turn on coupler switch within 20 seconds. Keyboard should unlock. |
| 6. If DATA light goes out during session, retry from Step 1. | |

**Note:** These procedures depend on the particular Data Set or Acoustic Coupler you are using. See your system administrator for particulars on the set you're using.

Figure 11. Switched Line Terminal Connection Procedures

For the 2741 terminal, before you enter the logon command, it may be necessary to enter the following characters:

```
/"[nn]
```

where *nn* is a 2-digit number optionally predefined by your system administrator. Ask your system administrator if this entry is required for your 2741 terminal.

## VSPC Command (Logon)

The VSPC command connects the terminal with VSPC.

| | | |
|---|---|---|
| **VSPC** | ID=*usernum* | Required first operand. |
| | *x* | Required operand for S/S terminals. Ignored for others. |

ID=*usernum x*
identifies the user to VSPC.

*usernum*
is assigned by the system administrator for each user of the subsystem. It is a numeric identifier from one through seven digits long; a value of zero (0) is not a valid user number.

*x*
when specified, must be the upshift character of the "2" key. This operand is required for the following terminals: 3767 in start/stop mode, 1050, and 2741. For other terminals, this operand is ignored.

The VSPC command must be entered either entirely in uppercase letters or entirely in lowercase letters, as follows:

```
VSPC ID=12345
vspc id=12345
```

or as determined by your VSPC administrator.

**Note:** Depending on the default your organization has chosen, you may or may not be able to abbreviate the VSPC command. Check with your system administrator.

When the connection between the terminal and VSPC is made, either or both of the following two prompting messages may be received:

```
ENTER UP-SHIFT 2
```

When the ENTER UP-SHIFT 2 message is received, the *x* operand has been erroneously omitted, or incorrectly entered. The upshift character of the 2 key must be typed in response. If the response is incorrect, the prompt is repeated until logon is successful or the two-minute logon time limit has been reached.

```
ENTER PASSWORD
```

When the ENTER PASSWORD message is received, a password is required to complete a successful logon. The correct password must be typed in (see "Passwords" for syntax rules). If a correct password is not supplied after three prompting messages, the logon is unsuccessful.

When logon is successful, VSPC sends some or all the following responses:

```
attribute hh:mm:ss mm/dd/yy usernum
CONTINUE attribute
(or CONTINUE attribute PASSWORD)
logon-message
```

**Note:** The CONTINUE line is not always printed—see text.

*attribute,* in the first line, is the default workspace attribute for this user profile. Always displayed.

*hh:mm:ss* (hours:minutes:seconds) specifies the logon time. Always displayed.

*mm/dd/yy* (month/day/year) specifies the logon date. Always displayed. (This field may alternatively be displayed as *dd/mm/yy*—check with your VSPC administrator.)

*usernum* is your user number. Always displayed.

CONTINUE *attribute* or CONTINUE *attribute* PASSWORD appears only if there is a CONTINUE file in your library. The attribute is that of the CONTINUE file (it may be different from the default attribute). If a password is needed to obtain the CONTINUE file, the word PASSWORD is printed; otherwise it is omitted. This access password is the one that was set for the contents of the CONTINUE file.

*logon-message* contains a general information message. This line is displayed only if a logon message has been defined.

**Note:** If the workspace attribute or the CONTINUE file attribute is not presently known to VSPC, that part of the message contains asterisks (*). In this case, the workspace attribute is set to DATA.

Once a logon request has been entered, it must be completed within two minutes or the logon is unsuccessful. The two minutes begin as soon as VSPC accepts the logon request; they end when VSPC has accepted as valid the usernum and, if necessary, the upshift 2 and the password.

### Example of 2741 or 3767 Start-Stop Logon

For the 3767 in S/S mode and the 2741, you enter the logon command as follows:

```
vspc id=12345 @
ENTER PASSWORD
```
(you type the word TABOO over the blot characters)
```
FORTRAN 12:45:16 05/31/76 12345
WELCOME TO VSPC
READY
```

The @ (which represents the upshift of the 2 key) must be entered for the 2741 and for the 3767 operating in start-stop mode.

When you enter your logon password—TABOO—VSPC prints ■■■■■■■■ over which you type the word TABOO. In this way, no one else can read the password you've typed in.

If you make a mistake when you are typing the password—suppose you typed TABU instead of TABOO—VSPC prompts you again to enter the correct password. After you've made four attempts, VSPC no longer allows you to continue. In this case, you must reenter the logon command.

After you've entered the correct password, VSPC prints the following information:

- your profile *attribute,* (FORTRAN), the logon time, (12:45:16), the date (05/31/76), and your user number (12345).
- an optional *logon message* (WELCOME TO VSPC)—this message will vary from one computer system to another; it can vary from day to day, and it sometimes may not be printed at all.

## Conversing with VSPC

When engaged in a conversation with VSPC, you tell VSPC what to do from your terminal, and VSPC responds. Responses from VSPC vary depending on the type of terminal and whether you are typing lines of data or are running a program.

This discussion applies to the 3767 in start-stop mode and 2741, with some reference to other terminals. For more detail, see *VS Personal Computing (VSPC) Terminals.*

1. VSPC notifies you in one or more of the following ways that it is ready to accept input:
   - Unlocks the keyboard
   - Displays a question mark (?)
   - Lights the ON LINE and PROCEED indicators
   - Prints a message (for example, READY)
   - Prints a line number (for example, 00010)

2. You type a line of input as follows:
   a. Type the line of information.
   b. Press carrier return.
   c. Wait for VSPC to respond in one or more of the above ways before you type another line.

**Note:** Some users of 2741, 1050, and CPT-TWX terminals should not enter '99999' as the first five characters of a terminal input request; some users of 3270 terminals should not respond by pressing the test request key. Either of these actions may cause the terminal to be disconnected from VSPC. See your VSPC administrator to determine whether these restrictions apply to your installation.

## *Correcting Errors*

To correct typing errors in your current line:

1. Backspace to the error and signal attention.
2. VSPC responds with a series of actions:
   a. Prints an underscore (__) beneath the erroneous character.
   b. Backspaces one space.
   c. Advances one line and opens the keyboard.
3. Type the correction and the remainder of the line.
4. Press carrier return and wait for the VSPC response.

For example:

```
100 A=12E
        3
```

## *Deleting Lines*

To delete an entire line you are typing:

1. Press attention without an immediately preceding backspace.
2. VSPC prints an asterisk (*), effects a carrier return, and opens the keyboard.

For example:

```
100 A=12A*
```

## *Use of Attention Key*

In addition to character correction and line deletion, you can use the attention (ATTN) key for four other purposes:

- To signal the system–When your terminal is idle, pressing ATTN sends a signal to the system.
- To cancel output–When your terminal is printing, pressing ATTN ends the printing.
- To escape from your current input request–When it is the only response to an input request, pressing ATTN lets you cancel that request.

- To unlock your terminal for input when a MESSAGE WAIT command is in effect.

VS BASIC provides two modes for entering an attention during program execution: with an ON ATTN in effect or with ON ATTN not in effect.

If ON ATTN is in effect, you can hit the carriage return to begin execution at the point specified in the ON ATTN statement, or you can enter RES to resume execution at the point of the interruption. Any other action you take terminates execution of the program.

## *Invalid Output Characters*

If you find an N backspace Z (⌧) character in your output, it is an indication that VSPC has found an invalid character in your data. On other terminals an invalid character may be shown by other symbols. See Appendix ~~D~~. C.

# PROGRAM DEVELOPMENT

## VSPC FORTRAN Source Programs

VSPC FORTRAN source programs are VSPC sequential files that have been created with the VSPC Editor, and stored (VSPC SAVE command) with a content of either FORTRAN (if the source statements are free-format) or XFORTRAN (if the source statements are fixed-format). For example:

```
.
.
.
200 END
ENTER FORTRAN
SAVE PROGA
```

stores the source program in your workspace with a name of PROGA and content attribute of FORTRAN. Fixed-format source lines are truncated to 72 characters if they are longer, or padded with blanks to 72 characters if they are shorter. Each free-format source statement is restricted to 1320 characters, not including the line number, statement label, non-significant blanks, and continuation characters. The first character of a record follows the first blank after the line number. Since blanks are not significant in FORTRAN source programs, the padding of fixed-format source records will only change the program if there is a Hollerith or quoted literal continued over the end of that line. Thus, you must ensure that any continued literal stops exactly in position 72 of the line. Because the VSPC Editor can alter the length of a line when a field in the line is changed, it is good programming practice to avoid the use of continued literals in fixed-format source programs.

See the "VSPC FORTRAN Programming Considerations" section of this publication for additional information on language considerations.

Several steps are necessary to create and execute a VSPC FORTRAN program.

1. First, you name your program with the NAME command; then, you create your program line-by-line through the line entry procedure (optionally using the INPUT command for automatic line numbering); then use the SAVE command to place it in a library as source code.
2. You can modify your program with the various VSPC edit commands.
3. You execute your program using the RUN command; compile it and place it in a library as an object program with the STORE command.
4. In OS/VS, when an object program has been compiled, it can be combined with other object programs in the VSPC library using the RUN or STORE command and a linking control file. See the section "Combining Object Programs" in this manual.

These commands, plus others useful in VSPC FORTRAN program development are described in the following sections. Useful VSPC FORTRAN programming limitations, restrictions, considerations, and assumptions of which you will also want to be aware while developing your programs are in the "VSPC FORTRAN Programming Considerations" section of this publication.

## *NAME Command*

The NAME command assigns a name to your workspace. This name becomes the "filename" when a SAVE or STORE command is issued.

| | | |
|---|---|---|
| NAME | *filename* | Optional operand–choose one or omit. If omitted, name of current workspace is displayed. |

*filename*
specifies the identifying name to be used.

When the name field of the *filename* is CONTINUE, only your own library number can be specified or implied.

When the NAME command is executed, the specified *filename* replaces any previous filename identifying this workspace. When:

- the new *filename*—not including the optional password—is different from the workspace filename, and
- the library contains a non-empty file with the same name as the new *filename,*

the following warning message is issued:

```
WARNING - FILE ALREADY EXISTS
```

If the workspace with the new *filename* is later placed in the library, it replaces the previously existing file with the same name. If this is not what you want to do, issue a new NAME command specifying a different name before you save or store the workspace.

## Line-Numbered Data Entry

In VSPC, you can enter data lines by typing the line number, an optional space, the data, and a carrier return (you can enter the lines in any numerical order). VSPC then unlocks the keyboard so you can enter another line. For example, you type:

```
110 do 4 i=100,1000,2
120 k=sqrt(float(i))
140 2 continue
130 do 2 j=3,k,2
160 4 continue
```

However, VSPC treats the lines as if you'd specified them in the following order:

```
110 do 4 i=100,1000,2
120 k=sqrt(float(i))
130 do 2 j=3,k,2
140 2 continue
160 4 continue
```

By entering line numbers in increments greater than one, you'll later be able to insert new lines between already existing lines. For example, you decide to add one more statement to your program:

```
135 if(mod(i,j) .eq. c) go to 4
```

VSPC places line 135 in numerical order between lines 130 and 140.

You can replace a line you've already entered simply by entering a new line with the same line number. For example, if you type:

```
110 do 4 i=11,1000,2
```

this new line 110 completely replaces the original line 110, even though the new line is shorter than the original line 110.

To delete a line you've already entered, you enter the line number followed immediately with a *cr*. For example, you want to delete line 00280 of a program you just wrote. You enter:

`280` (you press *cr* key)

Line 00280 now no longer exists in your workspace. VSPC does not tell you, however, that the line has been deleted. (You can also use the DELETE command described later in this section.)

## INPUT Command

The INPUT command begins automatic line numbering for line entry.

| | | |
|---|---|---|
| INPUT | *begin-line*<br>*begin-line increment*<br>* | Optional first operand–choose one or omit. |
| | OVERLAY | Optional–specify or omit. If omitted, existing lines are not replaced. |
| | PROMPT<br>NOPROMPT | Optional–choose one or omit. If omitted, PROMPT is assumed. |

*begin-line* and *increment*
specify the line numbering conventions to be used.

*begin-line*
specifies the first generated INPUT line number. When specified, it must be an integer from 0 through 99999, inclusive.

If that line number already exists and the OVERLAY option was not specified, the line number entered will be the first number after that specified.

*increment*
specifies the increment by which succeeding line numbers are generated. When specified, it must be an integer from 1 through 99999, inclusive. When *increment* is omitted, an increment of 10 is assumed.

*
denotes your current line. It may be specified instead of *begin-line.*

**PROMPT and NOPROMPT**
specify the display convention to be used.

**PROMPT**
specifies that automatically generated line numbers will be displayed at the terminal.

**NOPROMPT**
specifies that automatically generated line numbers will not be displayed at the terminal.

When both PROMPT and NOPROMPT are omitted, PROMPT is assumed.

OVERLAY
specifies that entered lines are to overlay existing lines if they have the same line numbers.

If OVERLAY is not specified and you enter new lines that have numbers identical with existing lines in your workspace, the workspace is renumbered as necessary to avoid erasing lines. A message tells you that your lines are renumbered.

*Examples:*

```
input * prompt
```

in VIEW mode formats the screen with line numbers, starting with your current line, in the first five character positions (as shown in Figure 10).

Note: You cannot backspace and alter a generated line number.

When *begin-line* is omitted and the workspace contains line entries, the first generated INPUT line number is the multiple of 10 next higher than the highest line number in the workspace; the assumed increment is 10.

When PROMPT is in effect, each generated line number, followed by a space, is displayed at the terminal and the keyboard unlocks for input. Pressing ENTER ends automatic line numbering. The keyboard then unlocks for input, but no line number is generated.

When NOPROMPT is specified, the following message indicates the starting line number:

LINE *line-number*

where *line-number* is the first generated INPUT line number. For succeeding INPUT line entries, line numbers are automatically generated, but are not displayed at the terminal. Automatic line numbering ends when you press ENTER before any line entry data. The following message is displayed:

INPUT ENDED AT LINE *line-number*

where *line-number* is the last number assigned. However, if no lines have been entered, the response is READY. The keyboard then unlocks for entry of the next command.

## Editing Commands

You can modify your source program using the DELETE, MOVE, COPY, EXTRACT, JOIN, SPLIT, and CHANGE commands.

Use FIND to display a line containing specified sets of characters.

Use the TEXT command to tell VSPC whether or not to display the text of the line affected by your CHANGE, FIND, and LOCATE commands.

You can also renumber, save, and list your program, or erase it all with CLEAR.

### *DELETE Command*

Previously, you've seen how to delete a single line entry at your terminal or from your workspace. The DELETE command allows you to erase one or more lines from your workspace.

| | | |
|---|---|---|
| DELETE | *line*<br>*line*:*end-line*<br>* | Required–choose one. |

*line* and *end-line*
  must both be integers that specify line numbers within the workspace.

*line*
  specifies that this one line is to be erased. The line specified must actually exist in the workspace.

*line*:*end-line*
  specifies that the range of lines between *line* and *end-line* is to be erased. In this case, neither of the specified lines need actually exist in the workspace. When *end-line* is specified, it must identify a line number equal to or greater than *line.*

*
  specifies your current line. It can denote either *line* or *end-line.*

When execution is successful, the following message is printed:

```
nnn  LINES DELETED
```

where *nnn* is the number of lines which have been deleted.

### *MOVE Command*

The MOVE command repositions one or more lines of editable data within the workspace.

| | | |
|---|---|---|
| MOVE | *line*<br>*line:end-line*<br>* | Required first operand–choose one. |
| | *destination-line*<br>* | Required second operand. |
| | *increment* | Optional–specify or omit. If omitted, 10 is assumed. |

*line* and *end-line*
must be integers within the range 0 through 99999.

*line*
specifies the present line number of a single line to be moved. The line number specified must actually exist in the workspace.

*line:end-line*
specifies the present position of a range of lines to be moved; *line* specifies the first line in the range, and *end-line* specifies the last. The line numbers specified need not actually exist in the workspace; if they do not, then existing lines within the range specified are moved. The *end-line* must be equal to or greater in value than *line*.

*
specifies your current line. It can denote either *line, end-line,* or *destination-line.*

*destination-line*
is an integer that specifies the line number after which the range of lines is to be positioned; it need not actually exist within the workspace. It may be either lower in value than *line* or greater in value than *end-line.* That is, it must not be within the range from *line* through *end-line,* inclusive.

*increment*
specifies the increment by which moved lines are to be renumbered. It must be an integer in the range 1 through 99999.

- If *line* is specified and *end-line* is omitted, then the one corresponding line is moved to a position immediately following *destination-line,* if *destination-line* actually exists. If *destination-line* does not actually exist, the line is inserted using the specified destination line number.
- If *line* and *end-line* are both specified, then the inclusive range of corresponding lines is moved to a position immediately following *destination-line,* if *destination-line* actually exists. If it does not, they are inserted with the first moved line using the *destination-line* line number.
- Moved line(s) are renumbered sequentially higher than the line that corresponds to *destination-line.* If *increment* is specified, that increment is used in renumbering; if *increment* is omitted, the increment is 10.

- If the last generated line number overlaps the next existing line number after the inserted lines, then the succeeding original lines are also renumbered—as far as is necessary to maintain the ascending sequence of line numbers—according to the previous rule. The following message is sent to the user:

  ```
  RENUMBERED FROM LINE n
  ```

  where *n* is the number of the first original line to be renumbered.

When execution is successful, the following message is sent:

```
n LINES MOVED
```

where *n* is the number of lines that have been moved.

**Note:** If a MOVE command is executed for a workspace that corresponds to a direct file, the workspace will no longer meet direct file requirements, and will cause an error message when you attempt to save it in a library.

## *COPY Command*

The COPY command duplicates an existing line or range of lines in another location in the workspace.

| | | |
|---|---|---|
| COPY | *line*<br>*line*:*end-line*<br>* | Required first operand; choose one. |
| | *destination-line*<br>* | Required second operand. |
| | *increment* | Optional third operand; specify or omit. If omitted, 10 is assumed. |

*line*
specifies one line to be copied. The line number must exist in your workspace.

*line*:*end-line*
specifies a range of lines to be copied. If any line number specified in a range does not exist, then existing lines within the range specified are copied. The *end-line* must be greater than or equal to *line.* Line numbers can be between 0 and 99999.

*destination-line*
specifies the line number after which the lines are to be copied. It need not exist in your workspace; it may be higher or lower, but not within the range. If it does not exist, it becomes, the number of the first copied line.

*
specifies your current line. It can denote *line, end-line,* or *destination-line.*

*increment*
specifies the increment by which the copied lines are to be numbered. It must be an integer between 1 and 99999. When *increment* is omitted, an increment of 10 is assumed.

If the newly inserted line numbers overlap existing numbers, the lines following the insert are renumbered with the same increment as far as is necessary to maintain the line number sequence.

If the destination line is within the range of lines to be copied, the command is rejected.

The rules that apply to MOVE also apply to COPY.

*Examples:*

```
copy 290 400
```

copies line 290 with the number 00400 at line 00400. If necessary, lines are renumbered and a message returned:

```
LINES RENUMBERED FROM 410
```

```
copy 290:310 400 2
```

copies lines 290 through 310, places them at line 400, and increments by 2: 00402, 00404, 00406 .... No renumbering is done.

## *EXTRACT Command*

The EXTRACT command clears your workspace leaving the specified lines.

| | | |
|---|---|---|
| **EXTRACT** | *line*<br>*line*:*end-line*<br>* | Required—choose one. |

*line*
specifies a single line to be retained in the workspace. The line number must be in the workspace.

*line*:*end-line*
specifies the range of lines to be retained in the workspace. The *end-line* must be greater than or equal to *line.*

The line numbers specified need not be in the workspace; if they are not, existing lines within the range specified are retained.

*
specifies your current line. It can denote either *line* or *end-line.*

When EXTRACT is executed, the name of the resulting workspace is erased but the attribute is unchanged. The line numbers of the remaining lines are unchanged.

The current line pointer is unchanged if the current line is within the part of the workspace being kept. If the current line preceded the part retained, it is set to the first line of the resulting workspace. If the current line was after the retained part, it is set to the last line of the resulting workspace.

*Examples:*

```
extract 190
```

erases your entire workspace except line 00190.

```
extract 20:80
```

erases every line except 00020 through 00080.

## *JOIN Command*

The JOIN command unites two consecutive lines of the workspace.

| | | |
|---|---|---|
| **JOIN** | *line*<br>* | Required; choose one. |

*line*
specifies the first line number of the two consecutive lines to be joined.

*
specifies that your current line be joined to the following line.

The joined line has the line number of the first of the joined lines. The line number of the second line is erased; succeeding lines are not renumbered.

Your current line number is unchanged when using the JOIN command unless the second line joined is your current line. In this case the joined line becomes your current line.

## *SPLIT Command*

The SPLIT command divides a single line into two consecutive lines.

| | | |
|---|---|---|
| **SPLIT** | *line*<br>* | Required first operand; choose one. |
| | *number*<br>'*string*' | Required second operand; choose one. |

*line*
specifies the number of the line to be split.

*
specifies that your current line be split.

*number*
specifies the number of characters to be included in the first line. (The line number and following blank are not counted.) If *number* is greater than the number of characters in the line, the command is rejected.

'*string*'
specifies a set of characters that is to start the second line.

If 'string' occurs more than once in *line*, the first one is assumed.

The line number to be given to the second line is calculated from the line number of the line being split and the number of the line which originally followed it in the workspace.

- If the line being split is the last line of the workspace or the line following it has a line number 20 or more higher, then the second line is given a number 10 higher than the original line.
- If the original interval was less than 20, the second line will have a number midway between the two original lines.
- If the line that originally followed the line being split has a number only one higher than the line being split, then the second line is given a number one higher than the original line and further lines are renumbered with an increment of one as far as is necessary to preserve the sequence of lines. You will be informed if this renumbering is performed:

```
LINES RENUMBERED FROM n
```

where *n* is the first changed line number.

## *CHANGE Command*

We previously showed you how to make corrections during line entry. You can also replace characters in workspace lines—in one line, in a range of lines, or throughout the entire workspace. The CHANGE command allows you to do this.

| | | |
|---|---|---|
| CHANGE | *line*<br>*line:end-line*<br>*<br>ALL | Required first operand—choose one. |
| | '*string-1*'<br>'*string-1*' '*string-2*' | Required—choose one (optional on display terminals with line or *). |
| | TEXT<br>NOTEXT | Optional—choose one or omit. If omitted, specification in TEXT command is assumed; if specified, must follow 'string' operand. |

*line* and *end-line*
specify integers that correspond to line numbers in the workspace.

*line*
when specified alone represents a single line to be changed; the line number specified must actually exist in the workspace.

*line*:*end-line*
when specified, represents a range of lines to be searched. If the integers specified are not actual line numbers, all existing lines in the range

specified are searched. The value of *end-line* must be equal to or greater than that of *line*.

When both *line* and *end-line* are omitted, all lines of editable data in the workspace are searched.

*
specifies your current line. It can denote either *line* or *end-line*.

ALL
specifies that all lines in the workspace be searched for *'string'*.

*string-1* and *string-2*
specify the characters to be used; each may consist of 0 through 255 characters enclosed in single quotation marks; *string-1* and *string-2* need not contain the same number of characters.

All characters are valid. However, if a single quotation mark ( ' ) is contained within either string, it must be specified as two adjacent single quotation marks ( ' ' ). In the workspace, the adjacent quotation marks are treated as one single quotation mark.

*string-1*
specifies the characters to be changed.

*string-2*
specifies the replacement characters.

Any number of separators may be placed between *string-1* and *string-2*.

**TEXT** or **NOTEXT**
specifies the display convention to be used.

**TEXT**
specifies that changed lines are to be displayed at the terminal.

**NOTEXT**
specifies that only the line numbers of changed lines are to be displayed.

When both TEXT and NOTEXT are omitted, the last specification of the TEXT command is in effect.

On a 3270 Display System terminal you can issue the CHANGE command with only the line number. VSPC responds with a display of the line in the command area and you can change it (type over it, use DEL or INS MODE) and press ENTER.

When the CHANGE command is executed, all occurrences of *string-1* within the specified range of lines are replaced by *string-2*. The following replacement rules apply:

- If *string-1* only is specified, then *string-1* is deleted. (*string-2,* when omitted, is considered to contain 0 characters.)
- If both *string-1* and *string-2* are specified, *string-1* is replaced by *string-2*.
- If *string-1* is entered as having 0 characters (that is, if *string-1* is entered as two adjacent single quotation marks ( ' ' ), and *string-2* is specified, then *string-2* is added at the end of the specified lines.

- **If both strings are entered as having 0 characters, all lines or line numbers—depending on whether TEXT or NOTEXT is in effect—within the specified range are displayed.**

**When execution is successful and TEXT is in effect, all changed lines are displayed at the terminal.**

**When execution is successful and NOTEXT is in effect, only the line numbers of changed lines are displayed.**

**In either case, VSPC responds with the following message:**

```
nnn LINES CHANGED
```

**where *nnn* is the number of lines which have been changed.**

The current line pointer is unchanged unless you are using a display terminal. Enter

change *line*

and then reenter the displayed line. In this case the reentered line becomes your current line.

*Examples:*

You can specify a change to be made on every line of your program.

```
change all 'number' 'figure'
```

## *FIND Command*

The FIND command locates occurrences of a specific character sequence within the workspace.

| | | |
|---|---|---|
| FIND | *line*<br>*line*:*end-line*<br>* | Optional first operand–choose one or omit. If omitted, all lines are searched. |
| | '*string*' | Required. |
| | TEXT<br>NOTEXT | Optional–choose one or omit. If omitted, specification in TEXT command is assumed; if specified must follow 'string' operand. |

*line* and *end-line*
  specify integers that correspond to line numbers in the workspace.

*
  specifies your current line. It can denote either *line* or *end-line*.

'*string*'
  represents the characters searched for; it must consist of 0 through 255 characters enclosed in single quotation marks.

All characters are valid. However, if a single quotation mark (') is contained within the string, it must be specified as two adjacent single quotation marks (''). In the workspace, the adjacent quotation marks are treated as one single quotation mark.

**TEXT and NOTEXT**
specify the printing convention to be used.

The same rules as in the CHANGE command apply to the line and text operands.

## *TEXT Command*

The TEXT command tells VSPC whether or not to display the text of the lines affected by the CHANGE, FIND, and LOCATE commands.

| | | |
|---|---|---|
| TEXT | ON<br>OFF | Optional operand–choose one or omit. If omitted, current value is displayed. |

**ON**
specifies that the text of lines affected by CHANGE, FIND, and LOCATE are to be displayed unless the command itself specifies NOTEXT.

**OFF**
specifies that the text of lines affected by CHANGE, FIND, and LOCATE are not to be displayed unless the command itself specifies TEXT.

Specification of the TEXT and NOTEXT operands in CHANGE, FIND, and LOCATE commands overrides this TEXT command. When you log on, NOTEXT is assumed.

## *RENUMBER Command*

You can change the line numbers for all or part of your workspace with the RENUMBER command, and you can specify an increment to be used in the process.

| | | |
|---|---|---|
| **RENUMBER** | *<br>*old-line*<br>*old-line new-line*<br>*old-line new-line increment* | Optional—choose one or omit. If omitted, all lines renumbered with increment of 10. |

*
specifies your current line. It can denote *old-line* or *new-line*

*old-line*
identifies a line number in the workspace. It must be an integer with a value from 0 through 99999.

*new-line*
when specified must be an integer with a value from 0 through 99999. When specified, *new-line* represents the new value to be assigned to *old-line.*

*increment*
when specified, must be an integer that represents the increment used in assigning new line numbers; *increment* may have a value from 1 through 99999, inclusive.

When the RENUMBER command is executed, line numbers in the workspace are reassigned. The workspace must not be empty. The following rules apply:

- If all operands are omitted, all lines of data in the workspace are renumbered. The first line is assigned line number 10; succeeding line numbers are incremented by 10.
- If only *old-line* is specified, the editable data beginning at *old-line* and continuing to the end of the data are renumbered. The new line numbers are multiples of 10, and are incremented by 10.
- If *old-line* and *new-line* are specified, then *old-line* (the old line number) is replaced by *new-line* (the new line number), and succeeding line numbers are incremented by 10. The new line number must be higher in value than the last line number preceding the old line number; that is, there must be no overlapping of old and new line numbers.
- If the value of *old-line* falls between the values of existing line numbers in the workspace, the next higher existing line number is the first line renumbered.
- If *increment* is specified, then the new line numbers are incremented by the value specified in *increment.*

When execution is successful, VSPC responds with the following message:

```
n LINES RENUMBERED
```

where *n* specifies the number of lines renumbered.

**Note:** If a RENUMBER command is issued for a workspace that corresponds to a direct file, and the default *increment* of 10 is used, the workspace will no longer meet VSPC requirements for a direct file, and an error will occur when an attempt is made to place it in a library.

## *SAVE Command*

You've entered your source program from the terminal, and now it's ready to be used—you've done all the editing you want to do. Now you can save it in a library.

The library you can place it in can be your own private library; it can also be your project library; or it can be any public library. The naming convention you use depends on the library you want to save it in.

The SAVE command causes the workspace contents to be placed in the specified file.

| | | |
|---|---|---|
| SAVE | *filename* | Optional—specify or omit. If omitted, current workspace name is used. |

*filename*
when specified, creates or changes the *filename* for the saved file. The *filename* must conform to the rules for VSPC filenames.

When *filename* is specified, the workspace contents are placed in the file specified by *filename*. This *filename* now identifies the workspace (replacing any previously identifying *filename*).

If the SAVE command creates a new file, it can be placed in the user's private or project library, or any public library (except when the file is CONTINUE).

Only a file's owner can use the SAVE command to replace an existing file. The file type of the workspace and of the existing file must be the same (except when the file is CONTINUE). The file must not have NOWRITE protection.

If the *filename*—not including the optional password—is different from the workspace name, and the library specified already contains a non-empty file with the same name, SAVE command execution is unsuccessful, and an error message is printed. This prevents accidental overwriting of an existing file by a new file.

When *filename* is omitted, workspace contents are placed in the file, using the *filename* of the workspace. No check for duplicate names is made, because VSPC assumes one of the following:

- The workspace contents were loaded from this *filename*, and the SAVE command simply updates the old file.
- This *filename* was specified with the NAME command, and at that time a warning message was issued which you chose to ignore (see "NAME Command").

When *filename* is omitted and the workspace is unnamed, an error message is issued, and execution is unsuccessful.

When CONTINUE is the name field for this file, your own libnum must be specified or implied. The SAVE command overlays an existing CONTINUE file, even if CONTINUE is explicitly specified, and even if the file types are different.

When the library file is a direct file and the workspace does not qualify as a direct file, an error message is issued, and execution is unsuccessful.

## LIST Command

The LIST command displays all or part of the workspace contents.

| | | |
|---|---|---|
| LIST | *line*<br>*line:end-line*<br>* | Optional first operand–choose one or omit. If omitted, all lines are displayed. |
| | LINE<br>NOLINE | Optional–choose one or omit. If omitted, LINE is assumed. |

*line*
specifies the line you want displayed.

*line*:*end-line*
specifies that every line within the specified range of line numbers is to be displayed.

*
specifies your current line. It can denote either *line* or *end-line.*

When no line number is specified, all lines in your workspace are displayed.

**LINE** or **NOLINE**
specify the display convention to be used for line numbers.

**LINE**
specifies that line numbers are to be displayed.

**NOLINE**
specifies that line numbers are to be omitted.

When LINE and NOLINE are omitted, LINE is assumed.

When the LIST command is executed, the workspace is displayed according to the following rules:

- The workspace must not be empty.
- If the workspace contents are not owned by you and have PROTECT NOREAD in effect, the workspace is not displayed and an error message is issued.

### *CLEAR Command*

The CLEAR command provides an empty workspace.

| | | |
|---|---|---|
| **CLEAR** | | No operands allowed. |

When the CLEAR command is executed, VSPC erases all lines from the workspace, and also removes the following associated items, if present:

- The *filename* (including the password)
- Any PROTECT specification
- Any SHARE specification
- Any NUMBER specification differing from the default

The workspace attribute (whether the default attribute, or one specified through an ENTER command) is retained.

## Using VSPC Libraries

The VSPC library commands control access to the VSPC libraries available to you. There are commands for placing a file into your workspace, for creating and removing files, for specifying access control, and for obtaining a directory of current filenames. Each is explained in the following paragraphs.

## *LOAD Command–to Retrieve VSPC Files*

The LOAD command in this form places a specified file in the workspace.

| | | |
|---|---|---|
| **LOAD** | *filename* | Required. |

*filename*
identifies the file to be placed in the workspace.

The *filename* must identify a file that contains editable data. That is, the file may contain a command list, a data file, or a compiler source program. The file must not be an object program, an interpreter workspace, or an undefined file.

When the LOAD command is executed, VSPC locates the specified file and reads it into the workspace. Any previous workspace contents are completely replaced. All workspace characteristics—such as name, attribute, access control options, and NUMBER conventions—are changed to match those of the new contents.

## *FILE Command*

You can use the FILE command to create directory entries for new library members and to modify directory entries for existing files.

You can also use the FILE command to modify the size and/or the characteristic of the file.

| | | |
|---|---|---|
| FILE | *filename* | Required first operand. |
| | *n* | Optional second operand. If omitted for new files, editable file size limit is assumed, for existing files present file size limit is assumed. |
| | SEQUENTIAL<br>DIRECT | Optional–choose one or omit. If omitted, SEQUENTIAL is assumed for new files; for existing files, the present organization is assumed. |
| | UNDEFINED | Optional–specify or omit. If omitted, file is editable. |
| | CONTENT *(attribute)* | Optional; choose one or omit. If omitted, attribute is DATA for new files; unchanged for existing files. |

*filename*
identifies the file to be created or modified; it must conform to the rules for VSPC filenames (see "Filenames" and "Passwords").

The *filename* must not specify the special file CONTINUE.

*n*
specifies in thousands of bytes the maximum size permitted for this file; when specified it must be an integer in the range 0 through 65,535.

**SEQUENTIAL, DIRECT, and/or UNDEFINED**
when specified identifies the file organization for this file.

**SEQUENTIAL**
identifies this file as sequentially organized.

**DIRECT**
identifies this file as a direct file.

**UNDEFINED**
identifies this file as a sequentially organized foreground processor data file that cannot be edited by VSPC. The file organization can be sequential or direct; if neither the SEQUENTIAL nor DIRECT operands is specified, SEQUENTIAL is assumed.

VSPC Library commands and Job Entry commands can process UNDEFINED files.

CONTENT *(attribute)*
specifies attribute for the resulting file: a foreground processor, CLIST, or DATA. If CONTENT is omitted for new files, the DATA attribute is assumed. If CONTENT is omitted for existing files, the content attribute is unchanged.

When the FILE command is executed, the file identified by *filename* is either created or modified.

## File Creation

If the *filename* file does not already exist, a directory entry with the DATA attribute is created for it, but no space is allocated to it.

In this case, *n* specifies the maximum size for this file. If *n* is omitted, your user profile size limit for editable files is assumed.

## File Directory Entry Modification

If the file already exists, the following warning message is sent:

```
WARNING - FILE ALREADY EXISTS
```

and the FILE command is executed. (At least one operand in addition to *filename* must be specified.)

In this case, *n* specifies the new maximum size allowed for this file. If the contents of the file are larger than *n*, the following warning message, instead of the previous one, is sent:

```
WARNING - FILE LIMIT LESS THAN EXISTING FILE SIZE
```

The limit is set to the value of *n*. The size of present file can no longer be increased, and, if it is ever rewritten, *n* becomes the new maximum file size. (Note, therefore, that zero (0) is a legal maximum file size specification.)

If *n* is omitted, the present maximum size is unchanged.

The SEQUENTIAL, DIRECT, and/or UNDEFINED operand respecifies the file organization. If the operand is omitted, the present file organization is unchanged.

If DIRECT is specified, and the existing file does not meet the requirements for a direct file (see "File Organization"), the command is rejected.

A nonempty sequential or direct file cannot be changed to an undefined sequential or direct file, or vice-versa.

An object file cannot be changed to any other type.

### File Structure

The internal structure of all VSPC files is the same; therefore, they can all be read and written by FORTRAN programs, and their attributes can be changed freely by the FILE, ENTER, and NAME commands. Each record in the file has a number, called either its line number or its record number (the terms are interchangeable). Line numbers are provided by you when you create a file using the VSPC Editor, and record numbers are provided when the record is written by a FORTRAN program. In the latter case, the numbers start at one and increment by one. The remainder of the record is called the record text, and can vary in length from 0 to 4058 bytes. Other than that, the internal structure of VSPC is of no concern to the VSPC FORTRAN terminal user.

## *SHARE Command*

The SHARE command controls access to a file by users other than the owner.

You can make any file you own available to all VSPC users by specifying the SHARE command. You can also withdraw the permission to access a file.

| | | |
|---|---|---|
| SHARE | *filename*<br>* | Optional first operand–choose one or omit. (Required if other operand is specified.) |
| | YES<br>NO | Optional–choose one or omit. If omitted and the first operand is specified, YES is assumed. |

*filename* or *
  specifies the file to which the SHARE command will apply.

  *filename*
    when specified, applies the SHARE characteristic to the named file.

  *
    when specified, associates the SHARE characteristic with the workspace. The SHARE characteristic does not take effect until the workspace is placed in a library.

**YES** or **NO**
specifies the SHARE characteristic to be applied.

**YES**
specifies that the file can be accessed by all VSPC users.

**NO**
specifies that the file can be accessed only by users with normal access to this library.

The SHARE NO command negates the effect of a previous SHARE YES command. Before any SHARE command is issued, SHARE NO is in effect.

After SHARE NO is specified for a private library file, only the owner may access the file. After SHARE NO is specified for a project library file, only users of that project library may access the file.

When a SHARE command is issued and both YES and NO are omitted, YES is assumed.

The SHARE command does not affect access to a public library file. It cannot be issued for the special file CONTINUE.

After a SHARE command is issued, it remains in effect until another SHARE command changes the SHARE characteristic.

## *PROTECT Command*

The PROTECT command changes access to a currently existing file or to one that may be created later.

| | | |
|---|---|---|
| PROTECT | *filename*<br>* | Optional first operand–choose one or omit. (Required if other operands are specified.) |
| | PASSWORD<br>PASSWORD(*password*)<br>NOPASS | Optional–choose one or omit. If omitted and first operand is specified, password protection unchanged. |
| | READ<br>NOREAD | Optional–choose one or omit. If omitted and first operand is specified, access characteristic unchanged. |
| | WRITE<br>NOWRITE | Optional–choose one or omit. If omitted and first operand is specified, access characteristic unchanged. |
| | If all operands are omitted, the value applying to the current workspace is displayed. | |

*filename* or *
specifies the VSPC area to which the command will apply.

*filename*
specifies the file to which the PROTECT characteristics will apply.

*
specifies that the workspace contents are to have the PROTECT characteristics. The PROTECT characteristics go into effect when the workspace is placed in a library as a file.

**PASSWORD(*password*)** or **NOPASS**
specifies a change in password protection.

**PASSWORD(*password*)**
supplies a new *password* for the file. If the new *password* is omitted from the command specification, VSPC prompts the user to supply one (see "Passwords").

**NOPASS**
removes an existing password from the file.

**READ** and **NOREAD**
specify read characteristics for users other than the owner.

**READ**
allows users other than the owner to use the file as input for an executing program or to include it in a SUBMIT job entry command. READ also changes a previous NOREAD characteristic.

**NOREAD**
prevents users other than the owner from using the file as input for an executing program, or including it in a SUBMIT job entry command. It also changes a previous READ characteristic. Another user may, however, specify this file in a LOAD, MERGE, or RUN command, if he has access to it. The NOREAD operand cannot be specified for an object program file.

When a LOAD, RUN, or MERGE command places a file with NOREAD characteristics in the workspace, the entire workspace assumes the PROTECT characteristic. This means that the following commands are rejected: SAVE, STORE, CHANGE, FIND, and LIST.

**WRITE** and **NOWRITE**
specify write characteristics for the owner.

**WRITE**
specifies that the file's owner can modify this file.

**NOWRITE**
protects the file from modification, even by its owner. The file cannot be opened for output or update by an executing program, nor can it be named in a SAVE, STORE, or PURGE command.

Execution of the PROTECT command changes access to the file. Only the owner of the file can specify the PROTECT command with one exception: A library manager can specify PROTECT WRITE for a file in his library that he does not own.

When the PROTECT command is issued, at least one of the optional operands must be specified.

The PROTECT command cannot be issued for the special file CONTINUE.

A cleared workspace has the READ and WRITE characteristics; these may be explicitly changed through the PROTECT command; they can also be implicitly changed by loading a file which has the NOREAD or NOWRITE characteristics. A newly created file has the READ and WRITE characteristics, unless it was created by saving a workspace with the NOREAD or NOWRITE characteristics.

If WRITE is the only keyword operand, and a password for the file exists, the password can be omitted when specifying the PROTECT command. However, if the password is specified, it must be specified correctly, or the command is rejected. Figure 12 shows the type of access control for each library type.

| | **Library Type** | | |
|---|---|---|---|
| **Access Command** | **Private** | **Project** | **Public** |
| no access control commands | no other user can access | all project members can read | all users can read |
| SHARE YES | all other users can read | all users can read (as well as project members) | all users can read (as they always can) |
| PROTECT NOREAD | no other user can access | non-owners in project can load and run but not read | non-owners can load or run but not read |
| PROTECT NOWRITE | no other user can access owner can't modify | all project members can read; owner can't modify | all users can read; owner can't modify |
| PROTECT NOREAD PROTECT NOWRITE | no other user can access; owner can't modify | non-owners in project can load or run but not read; owner can't modify | non-owners can load or run but not read; owner can't modify |
| SHARE YES PROTECT NOREAD | non-owners can load or run but not read | non-owners can load or run but not read | non-owners can load or run but not read |
| SHARE YES PROTECT NOWRITE | all users can read; owner can't modify | all users can read; owner can't modify | all users can read; owner can't modify |
| SHARE YES PROTECT NOREAD PROTECT NOWRITE | non-owners can load or run but not read; owner can't modify | non-owners can load or run but not read; owner can't modify | non-owners can load or run but not read; owner can't modify |

Figure 12. Library Types and Access Control Commands

## RELEASE Command

The RELEASE command allows another user to gain ownership of your noncontrolled project file.

| | | |
|---|---|---|
| RELEASE | *filename* | Required. |

*filename*
specifies the name of the file to be made available for transfer of ownership.

The owner of a file makes it eligible for transfer with the RELEASE command. If must be a noncontrolled project library file. The file is still owned by the original user until another user issues an ACQUIRE command naming the file, or runs a program that accesses the file for output or update. If the ownership is transferred explicity by use of the ACQUIRE command, the *released* status is removed; if the ownership is transferred implicitly by access for output or update, the file remains *released.*

A file belonging to the library manager is not eligible to be transferred.

## ACQUIRE Command

The ACQUIRE command transfers ownership of a noncontrolled project file.

| | | |
|---|---|---|
| ACQUIRE | *filename* | Required. |

*filename*
specifies the name of the file to be transferred. The file must be in a noncontrolled project library.

When you issue the ACQUIRE command, VSPC checks that the file is in a noncontrolled project library, that you have enough space to store the file, and that the current owner has issued the RELEASE command.

If all these conditions are met, ownership transfers to you and the *released* status is removed. If not, the command is rejected and transfer does not take place.

The library manager of the project library cannot use the ACQUIRE command to become the owner of files owned by other project users.

*Note*: The current owner of a file can use the ACQUIRE command to retract the result of a previous RELEASE command.

## QUERY FILE Command—To Display File Status

This command displays status information about a file.

| | | |
|---|---|---|
| QUERY | FILE(*filename*) | Required. |

FILE(*filename*)
specifies that selected information about the named file is to be displayed.

*filename*
specifies the selected file. The file password, if any, is not required; if it is specified, however, it is checked.

When this command is executed, the following information is displayed:

OWNER: *usernum*—where *usernum* is the user number of this file's owner (displayed only if the file resides in a non-controlled PROJECT or PUBLIC library).

FILE TYPE: *type* CONTENT: *content*—where *type* can be SEQUENTIAL, DIRECT, OBJECT, UNDEFINED SEQUENTIAL, or UNDEFINED DIRECT, and where *content* can be CLIST, DATA, or *processor-name*.

FILE SIZE: *m* LIMIT: *n*—where *m* is the present file size in direct access storage, expressed in characters, and where *n* is the maximum allowable size, expressed in characters.

NUMBER OF RECORDS: *n*—where *n* is the number of logical records in the file (omitted if the file type is OBJECT).

RECORD SIZE: *n*—where *n* is the length of the logical records in the file (displayed only if the file has the specifications required for a DIRECT or UNDEFINED DIRECT file, and the file is not empty).

SEQUENCE NUMBERS: *option*—where *option* can be one of the following:

| **Option** | **Explanation** |
|---|---|
| LIST *n* | left margin, reserving *n* + 1 characters for a sequence number followed by a blank |
| LEFT *n* | left margin, reserving *n* number of characters |
| RIGHT *n* | right margin, reserving *n* number of characters |

(Omitted if the NUMBER attribute for the file is NONE.)

WORKSPACE LENGTH: *n*—where *n* is the length of object program when it is running or loaded in main storage (displayed only if the file type is OBJECT).

ACCESS: *access-option(s)*—where *access-option(s)* can be NOREAD, NOWRITE, and/or SHARE (displayed only if at least one of these options is in effect for the file).

DATE LAST WRITTEN: *mm/dd/yy*—where *mm/dd/yy* is the date the file was last opened for output or update, or a SAVE or STORE command was issued for it, or the file was created by a FILE command.

DATE LAST READ: *mm/dd/yy*—where *mm/dd/yy* is the date the file was last opened for input or update, or a LOAD or RUN command was issued for it. (Omitted if the file has not been opened for input or update, nor loaded nor run since it was last created or replaced.)

**Note:** Depending on the startup option chosen, the date field may be either *mm/dd/yy* (month/day/year) or *dd/mm/yy* (day/month/year).

## *QUERY LIBRARY Command–to Display Library Directory*

This command displays a list of the files in a library.

| | | |
|---|---|---|
| **QUERY** | **LIBRARY**<br>**LIBRARY(***libnum***)** | Required first operand—choose one. If (*libnum*) is omitted, your own library is assumed. |
| | **OWN** | Optional—specify or omit. If omitted, files of all owners assumed. |
| | **CONTENT(CLIST)**<br>**CONTENT(DATA)**<br>**CONTENT**<br>(*processor-name*) | Optional—choose one or omit. If omitted, files of all attributes assumed. |
| | **TYPE(SEQUENTIAL)**<br>**TYPE(DIRECT)**<br>**TYPE(OBJECT)**<br>**TYPE(UNDEFINED SEQUENTIAL)**<br>**TYPE(UNDEFINED DIRECT)** | Optional—choose one or omit. If omitted, all file types assumed. |
| | **FROM(***name***)** | Optional—specify or omit. If omitted, first file in library assumed. |

**LIBRARY** and **LIBRARY(***libnum***)**
identify the library to be listed. If *libnum* is omitted, your library is assumed. When *libnum* is specified, it must identify a library to which you normally have access. If a *libnum* of zero (0) is specified, your project library is assumed.

**OWN**
when specified, requests that only names of files which belong to you, within the specified library, be displayed.

When OWN is specified for a project or public library, only your own files are included in the list. OWN has no meaning, and is ignored, when specified for a private library.

**CONTENT**
when specified, requests that only names of files of the specified parenthesized attribute be displayed.

When CONTENT is specified, only names of files with the specified attribute (CLIST, DATA, or *processor-name*) are displayed. The *processor-name,* when specified, must be valid for this VSPC installation.

**TYPE**
when specified, requests that only files of the specified parenthesized type be displayed.

When TYPE is specified, only files of the specified type (SEQUENTIAL, DIRECT, OBJECT, UNDEFINED SEQUENTIAL, or UNDEFINED DIRECT) are included in the list.

**FROM**
when specified, identifies the file *name* at which the list is to begin.

Display begins with the first name which is equal to or higher than the specified *name,* and continues through the end of the library in ascending order according to the EBCDIC collating sequence—that is, in alphameric order as follows: $, #, @, A, ... Z, 0, 1, ... 9. (The EBCDIC collating sequence is given in *System/370 Reference Summary,* Order No. GX20-1850.)

Signaling attention during directory display causes the operation to end.

## *PURGE Command*

The PURGE command removes a file from a library.

| | | |
|---|---|---|
| **PURGE** | *filename* | Required. |

*filename*
identifies the file to be removed.

If the file has a password, the password can be omitted when the PURGE command is issued. If, however, the password is specified, it must be specified correctly or the command is rejected.

When the PURGE command is executed, the specified file is physically removed from the library.

For any file, the file owner can validly specify the PURGE command. For files in public or project libraries, the library manager can also specify the command. For other users, the PURGE command is rejected.

If PROTECT NOWRITE is in effect for this file, the PURGE command is rejected.

## Compiling and Executing a Program

VSPC provides facilities to compile and execute source programs, to execute object programs, and to store both source and object programs in VSPC libraries. FORTRAN source programs can be combined using the VSPC MERGE command. In OS/VS, FORTRAN object programs can be combined using the VSPC FORTRAN link processor (FLINK). The following sections explain how to use VSPC commands in conjunction with the FORTRAN or XFORTRAN compiler and the FLINK processor.

You can run a VSPC FORTRAN program from your workspace with the VSPC command RUN. This compiles the source program and, provided that there are no severe errors, executes it. When it has finished, the source program is still available in your workspace.

You can also run a source program which is saved in your library by specifying:

```
run proga
```

This command is equivalent to:

```
load proga
run
```

When the run has finished, the source program is still in your workspace, and you can use the VSPC Editor to correct any errors that you may have found. If you do, you can then run it again—this time from your workspace. When you are satisfied that it is correct, use the SAVE command to replace the corrected version in your library.

Specifying "save proga" puts PROGA in your own library, overlaying the version of PROGA that is already there.

When you are satisfied that your program is correct, and if you intend to run it frequently, you can save the output from the compiler in your library. You can then run the saved program without the overhead of compilation. An example of how to do this is:

```
load proga
store obja
```

The STORE command compiles the program in your workspace and saves the compiler output in the file OBJA, which has a content of OBJECT. You can then run it with:

```
run obja
```

The contents of your workspace are destroyed when you run an OBJECT program.

The use of program chaining, as in:

```
90 call opsys('chain','second')
```

is functionally equivalent to the RUN command, and normally causes the current workspace to be overwritten by the next program in the chain. (See the "VSPC FORTRAN Subprogram Library" section of this publication for a complete description of the call to the OPSYS subroutine.)

## *RUN Command*

| | | |
|---|---|---|
| RUN | *filename* | Optional—specify or omit. |
| | *'string'* | Optional–specify or omit. |

*filename*

when specified, identifies the file to be processed.

When *filename* is specified, the RUN command loads the specified file and operates on that workspace. The attribute of the file must be FORTRAN, XFORTRAN, FLINK, or CLIST.

- If *filename* specifies a FORTRAN source program, the program is loaded, compiled, and executed.
- If *filename* specifies a FORTRAN, XFORTRAN, or FLINK object program, the current data in the workspace is erased and the program is executed.
- If *filename* specifies a command list, the list is loaded and executed.

When *filename* is omitted, the RUN command operates on the contents of the workspace. In this case, the attribute of the workspace must be FORTRAN, XFORTRAN, FLINK, or CLIST.

*'string'*

specifies a character string to be passed to the program. It may consist of 0 through 32 characters enclosed in single quotation marks. All characters are valid. However, if a single quotation mark ( ' ) is contained within the string, it must be specified as two adjacent single quotation marks ( ' ' ).

The string may be retrieved by the program using the PARM function of OPSYS as described in the section "VSPC FORTRAN Subprogram Library" in this manual.

When the RUN command successfully operates on a FORTRAN program, the program is compiled and/or executed. A program header is displayed followed by any VSPC FORTRAN messages, followed by terminal interactions with the executing program (input and output).

When the RUN command successfully executes a command list, VSPC executes the commands in the list (see "Processing Command Lists").

When a CLIST contains a RUN command naming a second CLIST, return is made to the next command in the first CLIST following completion of the second CLIST. (See "Command List Nesting.")

## *STORE Command*

You can use the STORE command to compile a source program into a machine language object program; to name the resulting object program and place it in a library; and, in OS/VS, to combine object programs and place the result in a library. The object program is then available for execution at any future time.

| | | |
|---|---|---|
| **STORE** | *filename* | Required. |
| | **'NOLINK'**<br>**'LINK'** | Optional—choose one or omit.<br>Default discussed below. |

*filename*
specifies the name of the file in which the object program is to be placed. CONTINUE must not be specified.

**'NOLINK' and 'LINK'**
specify whether this object program is to be stored in a form that allows later linking by the FLINK processor.

**'NOLINK'**
specifies that linkages are to be resolved. The resultant object program cannot be used by the FLINK processor. 'NOLINK' is the default when the object program has a main procedure. When 'NOLINK' is specified with no main procedure, 'LINK' is imposed.

**'LINK'**
specifies that all linkages are not to be resolved. The resultant object program may be processed by the FLINK processor. 'LINK' is imposed when the object program does not have a main procedure.

The STORE command can be used to create a new object program file or to replace one that already exists.

The workspace must have FORTRAN, XFORTRAN, or FLINK as its attribute.

When the STORE command is executed, if the workspace has the FORTRAN or XFORTRAN attribute, the source program in the workspace is compiled. If the compilation is successful, the resulting object program is placed in the specified file identified by *filename*. After execution is completed, the source program in the workspace is unchanged. If the workspace has the FLINK attribute, the control file in the workspace is processed and the specified object programs linked. The resulting combined object program is placed in the specified file identified by *filename*.

If the STORE command creates a new object program file, that object program file can be placed into your private or project library, or into any public library.

If the STORE command replaces a file that already exists, the existing file must be an object program file and only the file's owner can replace it.

During STORE command execution, an attention signal causes execution to be terminated.

When compilation is successful, VSPC displays a header, the size of the object program, and the CPU time needed for compilation.

## MERGE Command

You may want to combine one file with another. You can do this using the MERGE command.

You can use the MERGE command to insert one file at a specific position following or within another file.

Using the MERGE command, you can also intersperse records from one file between records of another file.

| | | |
|---|---|---|
| MERGE | *filename* | Required first operand. |
| | * <br> *destination-line* <br> *destination-line increment* <br> OVERLAY | Optional—choose one or omit. If omitted, file placed at end of workspace contents. |

*filename*
identifies the file to be retrieved; it must contain editable data.

*
specifies your current line. It can denote your *destination-line.*

*destination-line*
identifies the line in the workspace after which the file is to be inserted. It must be an integer from 0 through 99999. If the *destination-line* specified is not an actual line number, the next lower existing line in the workspace is assumed.

*increment*
specifies the increment to be used for numbering the inserted file. It must be an integer from 1 through 99999. When *increment* is omitted, an increment of 10 is assumed.

**OVERLAY**
specifies that the file and the workspace are to be combined, based on their existing line numbers.

When the MERGE command is executed, VSPC retrieves the file and combines it with the workspace contents, according to the following rules:

- If both *destination-line* and OVERLAY are omitted, the file is placed at the end of the workspace contents. The line-number given to the first added line is the next multiple of 10 higher than any existing line-number. For the added lines, the increment is 10.
- When *destination-line* is specified, the file is inserted into the workspace as if it were the subject of a MOVE command (see "MOVE Command"). If *increment* is specified, that increment is used for numbering the inserted lines; otherwise the increment is 10. If the last generated line-number overlaps the next existing line-number after the inserted lines, then the succeeding original lines are also renumbered—as far as is necessary to

maintain the ascending sequence of line numbers—according to the previous rules. The following message is sent:

```
RENUMBERED FROM LINE nnn
```

where *nnn* is the number of the first original line to be renumbered.

- When OVERLAY is specified, the file and the workspace contents are combined according to the ascending sequence of their existing line numbers. If two line numbers are equal, the file line replaces the workspace line.

The attribute of the workspace is not changed. If the inserted file has a different attribute, MERGE command execution is completed, and the following warning message is issued:

```
WARNING - DIFFERENT FILE TYPE MERGED
```

If the file has the PROTECT NOREAD characteristic, and is not owned by you, then after the MERGE command is executed, your workspace assumes the protected characteristic, and any commands (as, for example, LIST) that would reveal the contents of the workspace cannot be executed.

## Combining Source Programs

FORTRAN source programs can be combined using the VSPC MERGE command. The combined source workspace need not necessarily include all the programs for a particular application. If desired, the combined workspace can be saved in the library using the SAVE command.

For example, if a complete application requires main program TAXES and subroutines SUB1 and SUB2, the following commands would combine the source programs and save the result in the library with *filename* TAXJOB.

```
load taxes
merge sub1
merge sub2
save taxjob
```

Since it consists of a complete application, the merged file can be immediately compiled. The following commands will store the compiled object program in the library with *filename* TAXOBJ and then run it.

```
store taxobj
run taxobj
```

If a portion of an application is subject to frequent modification, it is possible to merge the more stable routines. These can then be stored as a combined source file or compiled and stored as a partially linked object program. For example, the following commands would merge TAXES and SUB2, storing the partially linked object program as file PORTION1.

```
load taxes
merge sub2
store portion1 'link'
```

If SUB1 were later modified and recompiled, the object programs could be linked using the FORTRAN link processor (FLINK).

## Combining Object Programs

In OS/VS, FORTRAN object programs can be combined using the VSPC FORTRAN link processor (FLINK).

The FORTRAN and FLINK processors create two forms of the object program.

1. All references are resolved. This form cannot be used by the FLINK processor. You create this form by entering a STORE command either with no operand or with an operand of 'NOLINK'.
2. Some references are not resolved. That is, some references to subprograms or COMMON areas are not compiled with the program; these are called unresolved references. This form can be processed by the FLINK processor. You create this form with a STORE command and the 'LINK' operand. If an object program does not have a main procedure, then an option of 'LINK' will be assumed on the STORE command.

Both forms of object program can be executed with a RUN command provided a main procedure exists.

### *Copying Files with the FORTRAN Link Processor*

FLINK copies files from the VSPC library, either explicitly because the file is named in an INCLUDE statement, or implicitly because the name of an unresolved reference matches the name of a file in the library specified in a SEARCH statement. In either case, the file is only copied if it is an object file that was created by one of the following:

- STORE with the 'LINK' operand of a VSPC FORTRAN source program
- STORE with the 'LINK' operand of a FLINK control file
- IMPORT of an externally compiled program

**INCLUDE Control Statement**

| | | |
|---|---|---|
| INCLUDE | *filename* | |

*filename*
names a VSPC file which must have the characteristics described above.

The named file is copied from the VSPC library and incorporated into the output object program.

The file may be in your own library, in your project library, in a public library, or in any other library if the file has the SHARE characteristic. Files having the NOREAD characteristic may be included.

If the file is password protected, then the password must be specified following the file name.

INCLUDE may result in unresolved references as explained above; it may also cause outstanding unresolved references to be satisfied.

If the file does not exist, or does not have the appropriate characteristics, or is password protected and a password is not supplied, then the INCLUDE statement is ignored, and any outstanding unresolved references that would have been satisfied remain unresolved.

**SEARCH Control Statement**

|  |  |  |
|---|---|---|
| **SEARCH** | *libnum* |  |

*libnum*
specifies a VSPC library to be scanned for files having the name of unresolved references. A *libnum* of zero is a reference to the user's project library; specification of an asterisk (*) is a reference to the user's private library.

The SEARCH statement causes FLINK to search the given library for files whose names are the same as outstanding unresolved references. If the file has the appropriate characteristics as described above, then it is combined with the output object program. If the file is not found, has a password, or does not have the appropriate characteristics, then the reference remains unresolved. If the inclusion of a file results in further unresolved references, then an attempt is also made to satisfy these.

Any unresolved references that remain after completion of the SEARCH are left to be satisfied by succeeding INCLUDE or SEARCH statements.

## *Example Using the FORTRAN Link Processor*

Code and save the source for a new main program with calls to subroutines SUBA and SUBB that already exist in a public library (456) and a call to subroutine OWNSUB in your private library.

```
enter fortran
input
00010 . . .
00020 . . .
  .
  .
  .
00080 call suba
  .
  .
  .
00110 call subb
  .
  .
  .
00150 call ownsub
  .
  .
  .
00200 end
save mainsrc
```

Store the object level of the main program with the 'LINK' operand so that it may be processed by the FLINK processor.

```
store mainobj 'link'
```

Build the FLINK control file to control the link process.

```
clear
enter flink
input
00010 include mainobj
00020 include ownsub
00030 search 456
```

Save the source for the control file. The attribute of this workspace is FLINK.

```
save control
```

Link the subroutines and execute the program using the file still in the workspace.

```
run
```

Link the subroutines and store the resulting composite object program under the name CONTROL1 for future execution.

```
store control1
```

## *The AFPPREP Utility*

Subroutines not compiled under VSPC FORTRAN must be converted to VSPC workspace format before being brought into the VSPC library. The AFPPREP utility program converts an OS subroutine load module that was linked by a linkage editor into a VSPC workspace that can be imported by the VSPC Service Program using the IMPORT command. The subroutine can then be linked to a FORTRAN main program using the VSPC FORTRAN link processor. An OS load module should be brought into VSPC only if the following conditions are met:

1. It must have been link-edited by the VS linkage editor without error.
2. No host system SVCs can be issued.
3. It must not have any VSPC foreground interface service request.
4. It must not store any address within itself.
5. It must not be a main program.
6. It cannot make use of extended precision arithmetic.
7. Certain linkage editor attributes are not allowed.

For further information on the pre-IMPORT utility, see Appendix A of this manual or *VSPC FORTRAN Installation Reference Material.*

## Interrupting VSPC FORTRAN

There is a type of interrupt called an asynchronous interrupt that can occur at any time, and is not connected with your FORTRAN program. Four such interrupts are:

- The attention interrupt, which is a request to communicate with VSPC FORTRAN.
- Cancel output, which is a request to cancel all output to the terminal.
- Processing time limit, which may or may not be specified in your user profile. If it is, it is a limit on the amount of processing time you can use between terminal interactions. It is included as the CPUMAX field when you issue a QUERY PROFILE command.
- Session termination, which occurs when you are forced to sign off from VSPC; for example, when your telephone connection to the system is broken.

The first two interrupts described above are the results of your actions. Some terminals have two interrupt buttons—one for attention and one for cancel.

Others have only one, marked ATTN or BREAK. In this case, pressing the button causes both an attention and a cancel output interrupt to occur.

When session termination occurs, the VSPC supervisor attempts to save your editable workspace under the name CONTINUE.

### *Interrupts During Compilation of Your FORTRAN Program*

#### Attention

An attention interrupt causes VSPC FORTRAN to terminate processing with the message:

```
AFP034 ATTENTION INTERRUPT:COMPILATION TERMINATED
```

#### Cancel Output

A cancel output interrupt causes VSPC FORTRAN to suppress all output to the terminal. This status is reset:

- At the start of execution
- When you press attention

If execution is inhibited, the final message (AFP000) is sent to the terminal.

#### Processing Time Limit

This is ignored during the compilation of your program.

#### Session Termination

Compilation is terminated immediately.

### *Interrupts During Execution of Your FORTRAN Program*

**Attention**

An attention causes VSPC FORTRAN to interrupt execution of your program, and sends you the message:

```
AFP140 LINE xxxxx ATTENTION INTERRUPT
```

You are then prompted for input from your terminal. If you respond with a null line (carrier return only), execution continues. If you enter any data, or press attention, execution is terminated.

If you press attention in response to a terminal input request, and then elect to continue (by responding with a null line), you are prompted again for the input which was interrupted.

**Cancel Output**

A cancel output interrupt causes all further output from your program to be suppressed. This status is reset:

- When terminal input is required, either for a READ or a PAUSE statement (you will receive the '?' prompt)
- When you press attention

**Processing Time Limit**

When your processing time limit is reached, VSPC FORTRAN sends you the message:

```
AFP139 PROCESSING TIME LIMIT REACHED
```

You are then prompted for input from your terminal. If you respond with a null line (carrier return only), execution continues. If you enter any data or press attention, execution is terminated.

**Session Termination**

Execution of your program is terminated immediately.

### *Interrupts During Linking of Your FORTRAN Program*

An attention during FLINK processing causes VSPC FORTRAN to terminate processing and send you the message:

```
AFP034 ATTENTION INTERRUPT: LINKING TERMINATED
```

## Assigning Reference Numbers

To make referenced files available to your program during execution, you must issue the ALLOCATE command. The FREE command removes the assigned unit-number.

## *ALLOCATE Command*

In order to be able to perform an input/output request such as:

```
200 read( 1 )num
```

VSPC FORTRAN must know the name of the file you want to have read. To do this, you can issue a VSPC ALLOCATE command before you issue the RUN command:

```
allocate workfile 1
```

Then, whenever your program references unit 1, VSPC FORTRAN will reference WORKFILE.

| | | |
|---|---|---|
| **ALLOCATE** | *filename*<br>* | Optional first operand—choose one or omit. (Required if other operands are specified.) |
| | *unit-number* | Optional second operand. (Required if other operands are specified.) |
| | **MOD** | Optional—specify or omit. If omitted and first and second operands are specified, existing records overwritten. |
| | **RUN**<br>**SESSION** | Optional—choose one or omit. If omitted and first and second operands are specified, RUN is assumed. |
| | If all operands are omitted, current values are displayed. | |

*filename* or *
specifies the file or terminal assigned to the FORTRAN *unit-number.*

*filename*
specifies that the designated VSPC file is assigned to the FORTRAN *unit-number.*

*
specifies that your terminal is assigned to the FORTRAN *unit-number.*

*unit-number*
specifies the FORTRAN *unit-number*. It must be an integer with a value from 1 through 99, inclusive. If a given *unit-number* has already been allocated, the new allocation replaces the former one.

**Note:** The *unit-number* is equivalent to the data set reference number used in FORTRAN input/output statements.

**MOD**
has special meaning to VSPC FORTRAN. For details, see the discussion of MOD in the "ALLOC" section under "OPSYS" in the "VSPC FORTRAN Subroutine Library" chapter.

**RUN** or **SESSION**
specifies the length of time the allocation will stay in effect.

**RUN**
specifies that the *unit-number* allocation is to remain in effect until after the next RUN command is executed.

**SESSION**
specifies that the *unit-number* allocation is to remain in effect until the end of the current terminal session.

When both RUN and SESSION are omitted, **RUN** is assumed.

When the ALLOCATE command is executed, the specified *unit-number* allocation goes into effect immediately.

A check on the validity of the file specification is made when the file is opened at execution time.

There is a service subroutine provided by VSPC FORTRAN, called OPSYS (see "VSPC FORTRAN Subprogram Library") which enables you to perform the allocation within your program:

```
190 call opsys('alloc','workfile',1)
```

which has precisely the same effect. The terminal can be associated with a unit by specifying an '*' as the filename in the VSPC ALLOCATE command, and, unless they have been explicitly allocated otherwise, units 5 and 6 are taken to refer to the terminal.

If you want the allocation to remain in effect for more than one run, you must specify the SESSION option on the ALLOCATE command. This option is not available in the CALL OPSYS equivalent.

When you issue the ALLOCATE command for a file, the file need not exist in your library. However, that file must exist if a FORTRAN READ or WRITE tries to access it—even if it doesn't contain any data. The command:

```
file f
```

creates an empty file called F in your library.

## *FREE Command*

The FREE command allows you to remove a previously-specified allocation of a FORTRAN unit-number.

| | | |
|---|---|---|
| **FREE** | *unit-number* | Optional. |

*unit-number*
specifies a previously-allocated FORTRAN unit-number. When specified, it must be an integer with a value from 1 through 99, inclusive.

When the FREE command is executed, previous allocations of FORTRAN *unit-numbers* are removed according to the following rules:

- If *unit-number* is specified, only the allocation of the specified *unit-number* is removed.
- If *unit-number* is omitted, all of your *unit-number* allocations are removed.

## File Usage

The distinction is made in VSPC FORTRAN between two types of files—sequential and direct.

### *VSPC FORTRAN Use of Sequential Files*

In a sequential file, the order of the records is the order in which they were written. Records can only be retrieved in the order they were written, although the BACKSPACE statement can temporarily reverse this order. Normally, when you write such a file, the first WRITE statement creates the first record in the file. However, if you specify the MOD option on the ALLOCATE command, the first record written is added to the end of the file. The RESET function of OPSYS can also be used to specify whether records are written to the beginning or added to the end of the file, overriding the MOD option.

### *VSPC FORTRAN Use of Direct Files*

In a direct access file, records can be referenced in any order, and READ and WRITE statements can be freely intermixed. If you want to use a direct access file, you must inform VSPC and VSPC FORTRAN that it is to be direct. The command:

```
  file da direct
```

creates an empty direct access file in your library. When VSPC FORTRAN encounters an empty file referenced by a DEFINE FILE statement, it writes blank records to the file as specified by the statement. When a non-empty file is referenced by a DEFINE FILE statement, VSPC FORTRAN verifies that the record size is the same as specified in the DEFINE FILE statement and that the number of records is not less than the number specified in the DEFINE FILE statement; if these tests are met, the FORTRAN program can access all of the records in the file, even if there are more records than specified in the DEFINE FILE statement. If you edit a DIRECT file, the SAVE command ensures that you do not change the length of any records, and that your line numbers start at one and increment by one. If your program accesses the file sequentially, it is your responsibility to ensure that the file still has the attributes required to be DIRECT. VSPC FORTRAN does not pad or truncate records in this case.

The terminal is, by design, only able to handle sequential records; and so cannot be allocated to a unit which is to be accessed directly. If your program is going to modify the file, you can improve the efficiency of the VSPC FORTRAN by specifying MOD when the file is allocated, or by using the RESET function of OPSYS to specify the processing mode before the DEFINE FILE statement is executed (or to change the processing mode as required during actual processing).

### *Terminal Output Files*

You can use FORTRAN sequential list-directed or formatted WRITE statements and the PRINT statement to type at your terminal. When writing formatted records to the terminal, the first character of each record is taken to be an ASA control character, and should be one of the following:

| Character | Meaning |
|---|---|
| + | No space (not applicable on display terminals) |
| blank | single spaced lines |
| 0 | double spaced lines |
| - | triple spaced lines |

Any other character will result in a single space.

You have control over the printing width of your terminal using the LINESIZE command. If you try to write a record longer than that specified in LINESIZE to the terminal, it will be continued on as many lines as necessary. Since list-directed output can take up to 29 characters to write a number, LINESIZE specifications less than 30 are not recommended.

### *Terminal Input Files*

You can use FORTRAN sequential list-directed or formatted READ statements to read from your terminal, although list-directed input is often more convenient. Whenever your program requires you to enter a record, VSPC FORTRAN types a '?' to prompt you for input. It is good practice to always WRITE a message to the terminal indicating precisely what input is required.

If the record is required for a formatted READ, it is padded with blanks, if necessary, to the maximum terminal line width.

### *DATA and Non-DATA Files*

When you WRITE files with a content attribute of DATA, VSPC FORTRAN provides the record numbers that are required by VSPC, and when you READ these files, the record numbers are not passed on to the program.

However, if you WRITE a file that has a content other than DATA, VSPC FORTRAN expects you to provide the record numbers as the first characters of the record: leading blanks are replaced by zeros, and the line number is taken from the start of the record up to the first non-numeric character, with a maximum of five digits. If the character after the line number is a blank, it is ignored, and the remainder of the record is written as the record text.

When a non-DATA file is read, the record number is converted to a five-digit field, followed by a blank. It is used to prefix the record text.

The VSPC line numbers are useful for files which will be referenced both by FORTRAN programs and from the terminal using the VSPC Editor: the Editor treats the line number as a key, allowing you to refer to any record directly, and your program also can process the line number when it reads the file.

### *FORTRAN Undefined Files*

Undefined files are special purpose data files created by FORTRAN. They contain data in a format tailored to VSPC FORTRAN's needs, and cannot be edited by the user. Only programs can process or update such files. Undefined files can be either sequential or direct. They can be read and written using list-directed statements only, but all the I/O control statements and direct-access forms of statements are valid. They are created with the VSPC command:

```
  file u undefined
```

and have a content attribute of DATA. Undefined files are written in internal format, and so avoid the overhead of conversion to character form. However, each data item has its type identified, and so input data can be checked. Invalid assignments, such as LOGICAL to REAL, are not allowed, and if necessary, the input data is converted accordingly. You do not have to know the precise data types used to write the file, although, if you do, unformatted I/O is more efficient and takes less file space. Each item in an undefined file takes one character more than its associated length, and so the record length required for an UNDEFINED DIRECT file is easily calculated.

## Processing Command Lists

A command list is a series of VSPC commands entered into your workspace as data. VSPC commands are immediately executable, but a command list is not—it is first created and then executed later as a group of commands.

Once a command list is created it can be executed immediately, and/or it can be saved in a library for future reference. The sequence of commands can be executed using the RUN command.

When you create a command list, you enter the series of commands, preceding each one with a line number. You can either type the line numbers yourself, or you can use INPUT mode to generate them automatically.

Once the entire list is completed, you must make sure the workspace attribute is CLIST before you process the list. Then you can execute the list immediately using the RUN command, and/or you can place it in a library using the SAVE command.

You can create command lists to execute a series of commands you know you'll always use whenever you're doing a particular type of processing.

For example, you might want to run a program that requires a temporary file. With the following CLIST, you can create and purge the file automatically:

```
enter clist
10 file temp direct
20 run sort
30 purge temp
run
```

Or you may want to run two programs in succession, making the output of the first program available as input to the second program:

```
enter clist
5 /* this is the clist */
10 file work
20 allocate work 6
30 run prog 1
40 allocate work 5
50 run prog 2
60 purg work
70 /* end of clist */
save both12
  .
  .
  .
run both12
```

In either case, you'll be able to use one RUN command in place of the series of commands in the list.

During execution (with one exception) each command is displayed at the terminal in the following form just before its execution:

```
*line-number command
```

The exception is that if the command list has the PROTECT attribute of NOREAD, the commands are not displayed.

In either case, however, output is displayed at the terminal. Such output can be from an executing command (for example, a FIND command) or from a program (for example, a FORTRAN program invoked by a RUN command).

As soon as one command has been processed, the next command in the list is executed. After the last command in the list has been processed, the terminal keyboard unlocks for further terminal input.

### *Command List Nesting*

You can write command lists in such a way that one list will cause another list to be executed; when the second list completes execution, control is automatically returned to the first command list, beginning at the next command after the RUN command.

You use the RUN command within the first command list to begin the processing of the second. The second command list can itself contain a RUN command that causes the processing of a third command list. In this way, you can nest any multiple command lists.

## Exchanging Data with Programs Written in Other Languages

VSPC FORTRAN has the ability to exchange data with VS APL and VS BASIC, two IBM program products that run under VSPC. This data can be exchanged using VSPC sequential and direct access files. Access is through the use of either formatted, list-directed, or unformatted input/output statements.

Data written by one of the three supported VSPC languages that is to be used by a program written in one of the other languages, must be in a form that can be read or interpreted by the receiving program. Of course, because the structures of the languages vary, there are restrictions on the types of data that can be exchanged. It is your responsibility to ensure type compatibility when exchanging data with programs written in other languages.

### *Exchanging Data with VS APL*

VS APL's EBCDIC files can be read by the VSPC FORTRAN formatted and list-directed input/output statements. Each character vector written appears as a separate record. If the character vector was created using the monadic format operator, the resulting record will be in list-directed format. Otherwise, the record will be formatted.

VSPC FORTRAN's formatted and list-directed files can be read by VS APL. Each record is read into a character vector. If the data to be written is all numeric (excluding COMPLEX), list-directed WRITE statements can be used, and the record can be decoded using the Execute operator to perform an assignment. Otherwise, formatted WRITE statements must be used, and the receiving function must decode the resulting string. For a description of APL functions that perform conversion of data between APL and other language formats, see the listings under "DATACVGP" in the publication *VS APL for VSPC: Terminal User's Guide,* SH20-9066.

## *Exchanging Data with VS BASIC*

VS BASIC has input/output statements that correspond to FORTRAN's formatted and unformatted input/output statements. In an unformatted file, there is no conversion, so the data types involved in the two languages must correspond; therefore, the only unformatted file data types that can be exchanged between VS BASIC and VSPC FORTRAN are REAL*4, REAL*8, and variables having character string values.

In a formatted file, certain conversion types have counterparts in VS BASIC. These types can be exchanged between programs written in either language. Unreadable data can be skipped using X or T format items.

It should be noted that VS BASIC writes arrays in row major order and VSPC FORTRAN writes arrays in column major order. After being read, these arrays will have to be transposed by the receiving program before they can be used.

VS BASIC's formatted input/output statements can be used with S (short precision floating point) and L (long precision floating point) format items to create a file that can be read by VSPC FORTRAN's input/output statements. These S and L format items can be read into REAL*4 and REAL*8 variables using A4 and A8 format, respectively.

VS BASIC has a stream I/O facility that is similar to FORTRAN's list-directed I/O. Because of differences between the languages in the handling of record boundaries, individual items in these files cannot be exchanged. VSPC FORTRAN can, however, read a VS BASIC "list-directed" file by using a single READ statement to retrieve the entire file at one time.

VS BASIC can read VSPC FORTRAN's list-directed files that do not contain the LOGICAL data type. Only the real part of COMPLEX data is obtainable.

Corresponding conversion specifications in VSPC FORTRAN and VS BASIC are:

| FORTRAN | VS BASIC |
|---|---|
| D or E | PIC (numeric with exponential notation) |
| F | PIC (numeric with decimal point) |
| I | PIC (numeric with no decimal point) |
| A | C |
| A(4) using a REAL*4 variable | S |
| A(8) using a REAl*8 variable | L |
| X | X |
| T | POS |

**Note:** FORTRAN cannot read:

NC (zoned decimal)
PD (packed decimal)
PIC with editing characters

VS BASIC cannot read:

COMPLEX data
LOGICAL data

# VSPC FORTRAN SUBPROGRAM LIBRARY

VSPC FORTRAN contains all the FORTRAN library subroutines and functions, and so these do not take up any space in your object workspace.

These subprograms fall into two groups: mathematical and arithmetic functions, such as SQRT and MOD; and service subroutines, such as OVERFL.

If your program contains a subprogram that has the same name as one of the library subprograms, VSPC FORTRAN chooses yours for execution. However, if the library subprogram is a mathematical function, the compiler expects your function to take the same number and type of arguments as the library function, and to return the same type of value. This can be avoided if you include your function name in an EXTERNAL statement. For example:

```
external dsin,log
```

This, effectively, detaches those names from the library.

## Mathematical Functions

VSPC FORTRAN supplies all the standard mathematical functions as described in the publication *IBM System/360 and System/370 FORTRAN IV Language,* GC28-6515, Appendix C.

There is more information about the expected accuracy of these functions in "Appendix D. Mathematical Function Accuracy Statistics."

## Service Subroutines

The service subroutines available in VSPC FORTRAN include all the standard FORTRAN IV subroutines, and one other, OPSYS, which supplies utility functions specific to the VSPC environment. They are invoked through VSPC FORTRAN's CALL statement.

### *EXIT*

EXIT terminates execution of your program and returns control to the VSPC system. EXIT performs a function similar to the STOP statement, except that no message is written to the terminal. EXIT has no arguments:

```
call exit
```

### *DVCHK*

DVCHK tests whether a zero divide check has occurred. The divide check indicator is initially off, and is turned off after a call to DVCHK. The source statement is:

```
call dvchk(i)
```

*i* is set to 1 if the divide-check is on, or 2 if it is off.

## *OVERFL*

OVERFL tests whether or not an exponent overflow or exponent underflow exception has occurred. The overflow indicator is initially off, and is turned off after a call to OVERFL. The source statement is:

```
call overfl(i)
```

*i* is set to 2 if no overflow or underflow has occurred, 1 if the last condition was overflow, or 3 if the last condition was underflow.

## *SLITE*

SLITE sets the pseudo sense lights on or off. There are four such sense lights, and they are initially off. The source statement is:

```
call slite(i)
```

If *i* has a value of 1, 2, 3, or 4, that particular sense light is turned on. If the value of *i* is 0, all four sense lights are turned off. Any other value for *i* is invalid.

## *SLITET*

SLITET tests the value of a pseudo sense light. The source statement is:

```
call slitet(i,j)
```

*i* must have a value of 1, 2, 3, or 4, and indicates the sense lights to be tested. After being tested, the sense light is turned off.

*j* is set to 1 if the sense light is on, and 2 if it is off.

## *DUMP and PDUMP*

Because VSPC FORTRAN runs in the terminal-oriented, VSPC interactive environment, the storage dump facilities of other FORTRAN IV compilers are not essential to the user. They are therefore not supported by VSPC FORTRAN. If your FORTRAN program contains a call to the PDUMP subroutine, the following message is printed at your terminal:

```
AFP156  LINE xxx PDUMP FUNCTION NOT AVAILABLE
```

The DUMP subroutine causes the same message to be printed, except that the listed function is DUMP instead of PDUMP. DUMP also causes termination of execution of your program.

## *OPSYS*

OPSYS is provided to perform functions specific to the VSPC environment. The required function is indicated by the first argument, which must be a literal character string. For example:

```
call opsys('alloc','fortfile',16)
```

provides the VSPC file "FORTFILE" with the FORTRAN unit number 16.

The functions provided are ALLOC, RESET, FREE, CHAIN, PARM, COMMAND, DELAY, and FSM.

In the function descriptions that follow, all integer variables are required to be INTEGER*4.

## Filenames

Some OPSYS functions require a VSPC filename as an argument. Filenames are made up of three parts: the library number, the name of the file, and the password. The library number is not required when you are referring to your private library, and the password is not required when the file is not password protected, or if you want to be prompted for the password when the file is accessed. For example:

```
workfile
73000 input/pass
0 common/
```

are all valid filenames. There is a full description of filenames earlier in this publication, under "Filenames" in the chapter titled "Description of VSPC." Filenames, in a call to OPSYS, are limited to 32 characters, which is sufficient to contain any valid VSPC filename.

The filename can be contained in a literal character string, a scalar, or an array. However, if an array is used, it must be at least 32 characters long.

## ALLOC

You use the ALLOC function of OPSYS to assign a FORTRAN *unit-number* to a VSPC file. The source statement for the ALLOC function of OPSYS is:

call opsys('alloc',*filename,unit* [,'mod'])

which is similar to the VSPC ALLOCATE command, except that the VSPC SESSION-RUN option has no counterpart in OPSYS. Allocation, with OPSYS, is only for the duration of the run. A filename of '*' specifies that the terminal is to be the FORTRAN unit.

The VSPC unit number is equivalent to the FORTRAN *dataset-identifier* used in input/output statements. The unit number assigment identifies a data file in the executing object program.

For example, if your source program uses unit number 16 for an input file, you can code the OPSYS statement as follows:

```
call opsys('alloc','fortfile',16,'mod')
```

The VSPC file "FORTFILE" is now available to the object program as unit number 16.

"MOD" specifies that sequential data written by the object program will be added after any existing data in FORTFILE. (If MOD is omitted, the present contents of FORTFILE are replaced by new data and the object program creates a new file.) When you allocate a file that is to be processed by direct-access input/output statements, the MOD option determines the initial processing mode that will be set for a non-empty file when it is opened during execution of your DEFINE FILE statement (unless you open the file explicity with the RESET function of OPSYS before the DEFINE FILE statement is executed). If a non-empty file is opened by DEFINE FILE, the processing mode is set to non-exclusive update if MOD was specified when the file was allocated; otherwise, the processing mode is set to direct input. (When an empty file is referenced by DEFINE FILE, it is formatted with blank records and the processing mode is set to non-exclusive update.)

If the indicated unit is currently associated with another file, that file is freed. If the file is currently open, it will be closed before it is freed.

This function allows your program to use a file without requiring you to enter ALLOCATE commands before running it. For example:

```
100 real*8 fn
110 write(6,10)
120 10format('enter input file name')
130 read(5,20)fn
140 20format(a8)
150 call opsys('alloc',fn,1)
160 read(1,*)n
```

## RESET

The source statement for the RESET function of OPSYS is:

call opsys('reset',*unit*,'*mode*'[,*return*[,*reason*]])

This causes the file referred to by *unit* (which must previously have been allocated) to be opened for access in the file processing mode specified by *mode*; or, if the file is already open, it is reset for access in the specified processing mode. The specified processing mode overrides the MOD option (or lack thereof) which may have been specified when the file was allocated. However, use of the RESET function to set the processing mode explicitly does not prevent the FORTRAN program from changing processing mode implicitly in the usual FORTRAN manner later in the execution of your program (for example, if your program issues READ statements and later executes WRITE statements). The values of *mode* that you can specify are as follows:

| **Mode** | **Explanation** |
|---|---|
| input | allows reading records sequentially |
| output | allows writing records sequentially, which replace the existing file's contents, if any |
| append | allows writing records sequentially, which are added to the end of the existing file |
| dinput | allows direct-access FIND and READ statements |
| update | allows direct-access FIND, READ, and WRITE statements; no other user may be processing the same file in any mode but input or dinput while you are processing it in update mode |
| updhold | allows direct-access FIND, READ, and WRITE statements; no other user may access this file at all while you are processing it in this mode |

When the RESET function is used to set the mode for direct-access file processing, it may be executed before the DEFINE FILE statement and it may be executed any number of times to change the processing mode during the actual file processing after the DEFINE FILE statement (the DEFINE FILE statement must always be executed before any direct-access FIND, READ, and WRITE statements can be executed). The RESET function does not alter the value of the FORTRAN associated variable. If the RESET function is not executed before the DEFINE FILE statement, the DEFINE FILE statement will open the file in update mode if MOD was specified when the file was allocated, or in dinput mode if MOD was not specified. However, when DEFINE FILE formats a previously empty file, the file is set to update mode, regardless of any previous specification of processing mode.

The optional *return* and *reason* parameters in the RESET function are integer variables in which OPSYS will return the register 15 return code and register 0 reason code, respectively, which result from the attempt to open the file or change its processing mode. If *return* is zero, the attempt was successful. If *return* is non-zero, the attempt was unsuccessful and the file is no longer open (if further direct-access processing is desired, another DEFINE FILE statement must be executed). For the specific meanings of non-zero values of *return* and *reason*, see the description of the DUNIT, DOPEN, and DRESET service requests in *VS Personal Computing (VSPC): Writing Processors.* If the *return* and *reason* parameters are omitted and the RESET function fails, the program terminates.

The following example shows a use of the RESET and DELAY functions of OPSYS. When two users are using the example program, and one is writing to TESTFILE, the CALL OPSYS ('RESET') request by the other will be rejected, delayed, and retried, rather than terminating the program as would be the case if lines 60 through 140 were not coded.

```
 10        dimension iarray(10)
 20        call opsys('alloc','0 testfile',8)
 30        write (6,1)
 40    1   format (1x,'enter record number')
 50        read (5,*) i
 60    2   call opsys('reset',8,'update',irtn,irsn)
 70        if (irtn) 10,20,10
 80   10   if (irtn-12) 13,11,13
 90   11   if (irsn-258) 13,12,13
100   12   call opsys('delay',1.0d6)
110        goto 2
120   13   write (6,*) irtn,irsn
130        stop
140   20   continue
150        define file 8(100,10,u,ipoint)
160        read (8'i) iarray
170        do 30 j = 1,10
180   30   iarray(j) = iarray(j) + 1
190        write (8'i) iarray
200        call opsys('free',8)
210        end
```

## FREE

The source statement for the FREE function of OPSYS is:

```
call opsys('free',unit)
```

The FREE command removes previous FORTRAN unit number allocations:

```
call opsys('free',16)
```

The previous assignment of unit number 16 has now been removed.

If the unit is currently associated with an open file, the file is closed before it is freed.

## CHAIN

The source statement for the CHAIN function of OPSYS is:

call opsys( 'chain' ,*filename[,parameter]* )

This terminates execution of your program, and initiates a RUN of the file indicated by *filename.* The file can be a source or an object program written in FORTRAN or any other language available under VSPC, or it may be a CLIST file. If the program is a source or object program, you can pass a parameter to it by including a third argument in the OPSYS call. This parameter can be a literal character string, a scalar, or an array. 32 characters will be passed to the receiving program, and if necessary, the argument will be padded with blanks. If an array is used, it must be at least 32 characters long. For example:

```
call opsys('chain','prog2',filen)
```

causes the program currently executing to stop execution; PROG2 to start execution with FILEN being the variable containing the parameter passed to PROG2.

Using this function, it is possible to divide a large program into several small programs, which may be necessary if your program exceeds your maximum object workspace size.

## PARM

The source statement for the PARM function of OPSYS is:

call opsys( 'parm',*variable* )

This function retrieves the parameter passed to your program from another program through the CHAIN function (or its equivalent in other languages) or specified on the RUN command. The parameter is assigned to *variable,* and truncated if necessary. *Variable* can be the name of a scalar or an array. If an array is used, it must be at least 32 characters long because the complete parameter will be assigned.

If no parameter was passed in the CHAIN, or if the program was not invoked through the CHAIN facility, the variable is assigned blanks.

## COMMAND

The source statement for the COMMAND function of OPSYS is:

call opsys( 'command' ,'*option*',*return,message,string*[,*length*])

This causes the VSPC system command in the variable *string* to be executed, with the result returned in the integer variables *return* and *message*. The *string* parameter can be a literal character string, a scalar, or an array; if it is a literal character string that is not longer than 32 characters, the integer parameter *length* may be omitted—otherwise, the number of characters in the VSPC command must be specified in the *length* parameter, which may be an integer constant or integer variable. The values of *option* that you can specify are as follows:

| Option | Explanation |
|---|---|
| all | allows VSPC to display all terminal output that results from execution of your command |
| error | VSPC displays only error and warning messages that result from execution of your command; any other terminal output is suppressed |
| none | VSPC suppresses all terminal output that results from execution of your command |

The *return* variable will contain zero if the command was executed successfully, 4 if the command was executed but a warning message was generated (even if output was suppressed), 8 if execution of the command was terminated by an attention signal from the terminal, 12 if the command was unsuccessful (an error message was generated), and 16 if the command was not a syntactically-valid VSPC command. The *message* variable will contain the VSPC message number of the last message that was generated by execution of the command, even if display of the message was suppressed because of the *option* parameter.

Only the following user commands and privileged commands may be specified; other commands will be rejected with *return* set to 16 (the privileged commands are also rejected unless the user executing the FORTRAN program has the correct authority):

**General User Commands**

| | |
|---|---|
| ACQUIRE | PASSWORD |
| ALLOCATE | PFKEY |
| BACKSPACE | PROTECT |
| FILE | PUNCH |
| FREE | PURGE |
| HARDCOPY | QUERY |
| KEY | RELEASE |
| LINESIZE | SEND |
| MESSAGE | SHARE |
| NAME | TABSET |
| NEWLINE | TAPE |
| NUMBER | TEXT |
| OFF | TRANSLATE |

**VSPC Administrator Commands**

ALTER
DEFINE

**VSPC Monitor Commands**

MONITOR
PMONITOR

**VSPC Operator Commands**

DISPLAY
HALT
JENTRY
SET

The following example shows the COMMAND function of OPSYS being used to create and later to purge a temporary file:

```
  .
  .
  .
call opsys('command','error',irtn,imsg,'file tempfile')
  .
  .
  .
call opsys('command','error',irtn,imsg,'purge tempfile')
  .
  .
  .
```

## DELAY

The source statement for the DELAY function of OPSYS is:

```
call opsys('delay', time )
```

This causes execution of the FORTRAN program to be suspended for the number of microseconds specified by the double-precision floating-point parameter *time*, which may be a D-format constant or a REAL*8 variable. During the time your program execution is suspended, the processor is not being utilized. The suspension of execution will be terminated if you press the attention key on your terminal.

An example of the DELAY function is shown above, with the RESET function of OPSYS.

## FSM

The source statement for the FSM function of OPSYS is:

```
call opsys('fsm', request,return,reason
      [,control,ctlsize [ ,data,datalength ] ] )
```

This function invokes the VSPC Full Screen Manager for IBM 3270 Display System terminals. The values that you can specify for the *request* parameter, which may be an integer constant or integer variable, are as follows:

| | |
|---|---|
| 16 | FSM Page function |
| 32 | FSM Read function |
| 64 | FSM Write function |
| 128 | FSM Exit function |

The *return, reason, ctlsize,* and *datalength* parameters must all be integer scalar variables, while *control* must be an integer array variable and *data* can be any variable used to hold character data. Unless *request* specifies the FSM Exit function, *control* and *ctlsize* must be specified, and if the FSM Read or FSM Write functions are specified, then *data* and *datalength* must be specified as well.

At the completion of the requested FSM function, OPSYS returns the register 15 return code and register 0 reason code in the variables *return* and *reason*, respectively. If *return* is zero, the FSM function executed successfully. For the meanings of other values of *return* and *reason*, and a detailed description of all four FSM functions and the uses of the *control, ctlsize, data*, and *datalength* parameters, consult the description of the TFSCRN service request in *VS Personal Computing (VSPC): Writing Processors.*

Note that in the TFSCRN service request issued by the FSM function of OPSYS, the WSHPARM1 field is set to the address of the *control* variable, the WSHPARM2 field is set to the value of the *ctlsize* variable, the WSHPARM3 field is set to the address of the *data* variable, and the WSHPARM4 field is set to the value of the *datalength* variable.

**The following example uses the FSM function of OPSYS, together with the assignment statements needed to assign values to its subparameters, to display the message FULL SCREEN TEST on the 9th row of the screen, and to read a one-position input field from the 23rd row of the screen:**

```
 10   dimension ictl(18),ictlr(18),idat(8),idatr(8)
 20   data ictl/0,23,1,16,0,0,269,9,1,16,1,16,12,23,1,1,0,1/
 30   data idat(1)/'full'/,idat(2)/' scr'/,idat(3)/'een '/,-
 40   idat(4)/'test'/
 50   data ictlr/0,0,0,4,0,0,0,0,0,0,0,0/
 60   ictlsz=18
 70   idatln=16
 80   ireq=64
 90   call opsys('fsm',ireq,irtn,irsn,ictl,ictlsz,idat,-
100   idatln)
110   ireq=32
120   ictlsr=12
130   idatlr=32
140   call opsys('fsm',ireq,irtn,irsn,ictlr,ictlsr,idatr,-
150   idatlr)
160   end
```

# VSPC FORTRAN PROGRAMMING CONSIDERATIONS

## Language Considerations for VSPC FORTRAN

### *Free-Form Source Statements*

The following rules govern free-format source statements:

- A source statement cannot contain more than 1320 characters, excluding line numbers, the statement label, continuation characters, and non-significant blanks.
- *First line of statement:* This can start in any typing position.
- *Statement numbers:* The first line of a statement can contain, as the first non-blank characters of that line, a statement number consisting of from one through five decimal digits. Blanks and leading zeros in a statement number are ignored as are any blanks preceding the statement number. A blank need not separate a statement number from the first non-blank character that follows the statement number.
- *Continued line:* A line of a statement to be continued is indicated by a hyphen as the last non-blank character on the line.
- *Continuation line:* A line following a continued line. A continuation line can begin in any typing position except where a literal constant is being continued, in which case the line must begin after the blank which follows the line numbers.
- *Comment line:* Any non-continuation line with an asterisk (*) or double quotation marks (") as its first non-blank character. A comment line cannot be continued, but multiple comment lines may be used.

Thus the fixed-form statements:

```
00010 c   sample text
00020       do 10 i=1,5
00030       a( i )=b( i )**2+4*i
00040      c+3*b( i )
00050 10    b( i )=0.
```

could be written as:

```
00010 * sample text
00020 do 10 i=1,5
00030 a( i )=b( i )**2+4*i-
00040 +3*b( i )
00050 10 b( i )=0.
```

VSPC FORTRAN compiles free-format source if it is invoked by the name FORTRAN, and fixed-format source if it is invoked under the name XFORTRAN, using the ENTER command as shown:

```
enter fortran
```

or:

```
enter xfortran
```

## *STOP Statement*

The STOP statement returns control to the VSPC supervisor, but it does not pass a return code, since there is no corresponding facility in VSPC.

## *PAUSE Statement*

The PAUSE statement message or number is displayed at your terminal, and you are prompted for input by a '?'. When you enter any (or no) data, followed by a carrier return, your program continues execution.

## *EXTERNAL Statement*

The appearance of a library subprogram name such as SQRT, in an EXTERNAL statement effectively detaches that name from the library for the duration of compilation. Unless you have included a FUNCTION subprogram with that name, the library function will still be called, but all checking of arguments is suppressed. The returned value is taken by default from the name, unless it has been explicitly or implicitly typed.

## *FORTRAN II I/O Statements*

VSPC FORTRAN supports the FORTRAN II I/O statements READ, PRINT, and PUNCH as shown below:

```
print 100,a        is equivalent to   write(6,100)a
punch 100,a,b      is equivalent to   write(7,100)a,b
read *,a           is equivalent to   read(5,*)a
```

Units 5 and 6 are normally allocated to the terminal, although this can be overridden by the VSPC ALLOCATE command. Since punched card output has little or no use in a terminal environment such as VSPC, unit 7 is normally not allocated, and if you wish to use it, you must issue an ALLOCATE command for it.

## *END= Parameter of the READ Statement*

If the END= parameter is included in a READ statement which is reading your terminal, the branch is never taken because there is no way to indicate end-of-file from the terminal.

## *List-Directed I/O*

VSPC FORTRAN provides the list-directed (or free-format) I/O facility, which is easier to use than formatted I/O and is particularly useful for input from the terminal, or from files which are created or maintained at the terminal.

To use list-directed I/O, you must code an asterisk instead of the format statement number in the READ, WRITE, PRINT, or PUNCH statement.

To enter data using list-directed I/O, type in a list of data items separated by blanks, tabs, commas, or carrier returns. The data items can be in the form of integer, real, double precision, complex, or logical constants, or literal strings enclosed in quotation marks.

If your program contains the statement:

```
150 read(5,*)a,b,c,d
```

and the numbers you want to enter are:

```
2.0, 6.74, 2.E+07, 6.4E-03
```

the following may happen (remember that VSPC FORTRAN types out a '?' every time it wants you to enter a line):

```
?2,6.74   2E7
?6.4E-3
```

You could just as well have strung the four values out on one line, all separated by commas, or all separated by one or more spaces, or you could have entered each value on a separate line. If you want to omit a value, and leave the variable unchanged, you can use successive commas:

```
?2,,2E7,6.400E-03
```

If the value or values you are leaving out are the last ones in the list, end the list with a slash. For example, to read in just A, B, and C, you would enter:

```
?2.0,6.74,20000000/
```

You can mix conventional and list-directed I/O in your program. This may be necessary since list-directed I/O cannot handle the writing of literal data.

The following is an example of mixed list-directed and formatted I/O as it might occur in a simple program to average three numbers.

Your program reads as follows:

```
100 50  read(5,*)x,y,z
110     sum=x+y+z
120     avg=sum/3.
130     write(6,100)
140 100 format (' the average of the three numbers is:')
150     write(6,*)avg
160     end
```

When your program runs, the I/O at your terminal looks like this:

```
?70,105,92
THE AVERAGE OF THE THREE NUMBERS IS:
89.0
STOP
```

Because of differences between the internal and external representation of real numbers, it is not always possible to convert numbers precisely.

If a real number is written to a file and read in again using list-directed I/O, the internal representation will be the same, and for this reason list-directed output produces up to nine significant digits for single precision numbers, and up to eighteen for double precision. However, if you require a number which is read in and then written out to have the same external representation, it should only be written with six significant digits for single precision, or fifteen for double precision, and so formatted output should be used.

## *Literals in Data Initialization*

In initializing an array, you should consider the following:

- Any element of an array can be initialized by subscripting the array name. Only one element is initialized. If excess characters are specified, they are not placed into the next element; they are truncated. An array element that is partially filled is padded on the right with blanks. The example below illustrates how individual array elements are initialized:

```
10 dimension a(5)
20 data a(1),a(2),a(4)/'abcd','qrstuvw','12'
```

The array elements contain the following:

```
A(1) A(2) A(3) A(4) A(5)

ABCD QRST      12
```

- Several consecutive elements of an array can be initialized with a single literal constant by specifying the array name without a subscript. The data is placed in the array, filling as many elements as are necessary to insert the entire constant (as long as the constant does not exceed the limits of the array). The example below illustrates how several array elements can be initialized with a single literal constant. Using this method, initialization always begins with the first element of the array. For example:

```
10 dimension a(9)
20 data a/'abcdefghijklmnopqrstuvwxyz'/
```

The array elements contain the following:

```
A(1) A(2) A(3) A(4) A(5) A(6) A(7) A(8) A(9)

ABCD EFGH IJKL MNOP QRST UVWX YZ
```

To begin initialization with an element other than the first, you can use an EQUIVALENCE statement:

```
10 dimension a(10),b(5)
15 equivalence(a(6),b(1))
20 data b/'abcdefghijklmnopqrst'/
```

The arrays will now be initialized as follows:

```
A(1) A(2) A(3) A(4) A(5) A(6) A(7) A(8) A(9) A(10)

                         B(1) B(2) B(3) B(4) B(5)

                         ABCD EFGH IJKL MNOP QRST
```

- Individual elements of an array can be initialized after initializing several elements with an unsubscripted array name. Each constant must follow the array name that is to be initialized. The example below shows how the two methods can be combined into one operation. After initializing the first several elements of an array with an unsubscripted array name, you can later initialize additional elements using the subscripted array name. For example:

```
10 dimension a(5)
20 data a/'abcdefgh'/,a(4)/'4444'/,a(5)/'5555'/
```

The array will be initialized as follows:

```
A( 1 ) A( 2 ) A( 3 ) A( 4 ) A( 5 )

ABCD EFGH        4444 5555
```

If the constants that are to initialize a part of the array are not specified immediately after the elements that will contain them, data that has overflowed into subsequent array elements may be replaced by data that was incorrectly specified. The example below illustrates one possible unintentional result:

```
10 dimension a( 3 )
20 data a,x/'abcdefghijkl',10.0/
```

The variables will be initialized as follows:

```
A( 1 ) A( 2 ) A( 3 ) X

ABCD 10.0 IJKL
```

As you can see, the compiler assumes that the second constant is intended for the second array element. The correct way to code this array is shown in the following example:

```
10 dimension a( 3 )
20 data a/'abcdefghijkl'/,x/10.0/
```

This would result in the following initialization:

```
A( 1 ) A( 2 ) A( 3 ) X

ABCD EFGH IJKL 10.0
```

## User-Written Subprograms

If your applications require frequent use of common subprograms not provided by VSPC FORTRAN, you can place these subprograms in the VSPC library and combine them with your main program using the appropriate VSPC Editor commands. Once a subprogram has been compiled with its main program (via the RUN or STORE command), the resulting executable program may be retained and executed, as often as you desire, without recompilation.

With the FORTRAN link processor, FLINK, user-written object programs can be kept in a library and retrieved for use with a FORTRAN main program without recompiling each time. You must first convert the program to a VSPC FORTRAN workspace using the pre-IMPORT utility (AFPPREP), and then move it into VSPC with the IMPORT command of the VSPC Service Program. For further discussion of the FORTRAN link processor, see Appendix A of this manual or *VSPC FORTRAN Installation Reference Material.*

## Techniques for Coding Efficient and Accurate Programs

In VSPC FORTRAN, as with programming in general, there are usually several ways to approach a problem and code a program. However, not all the methods produce equally efficient or accurate results. The techniques below have been included because they take advantage of VSPC FORTRAN's ability to produce efficient and accurate executable code.

## Array Notation in I/O Statements

The use of array notation is more efficient than an implied DO. In the example below, line 4 is more efficient than line 3:

```
2 dimension a(10)
3 read(9)(a(i),i=1,10)
4 read(9)a
```

## DO Loops

Within nested DO loops, VSPC FORTRAN optimizes references to arrays whose subscripts contain induction variables of the containing loops. For example:

```
100     do 10 i=1,5
110     do 10 j=1,5
120 10  a(2*i+3*j+4)=0.0
```

is more efficient than the following code because the subscript in line 120 contains I and J, the DO-loop variables, whereas line 220 does not:

```
200     do 10 i=1,5
210     do 10 j=1,5
215     k=2*i+3*j+4
220 10  a(k)=0.0
```

## Statement Labels

The VSPC FORTRAN compiler keeps track of certain information (such as simple subscripts) in order to avoid duplication of calculations in consecutive statements. This cannot be done, however, if a statement is labeled, because labeled statements can be the target of a GOTO statement. Avoid labeling statements if the label is not referred to anywhere else in the program.

A side effect of this is that a logical IF statement may result in more efficient code than an arithmetic IF statement.

For instance, the following code:

```
100 if(i.eq.0)goto 10
110 i=1
```

may be more efficient than the code:

```
100     if (i)20,10,20
110 20  i=1
```

The existence of a label at line 110 can cause information to be lost, even though statement label 20 is not referenced elsewhere.

## Efficient Use of I/O Statements

Certain combinations of I/O statements can result in inefficient execution, and so should be avoided if possible.

Multiple consecutive BACKSPACE statements may cause inefficient execution, whereas a single BACKSPACE between READs or WRITEs will not.

If a direct access file is to be updated, specify MOD in the ALLOCATE command or call to OPSYS routine.

In the VSPC environment, the FIND statement causes no improvement in efficiency. Provided that the indicated record is required, however, it has no adverse effects.

## *Efficient Use of Files*

VSPC files are kept in blocked format with a blocksize of 4063 bytes, and each record takes five bytes more than the length of the record text. Records are not spanned across blocks, so consistent use of certain large record sizes (such as 2027 bytes) is wasteful of space, and increases the time taken to read or write such files.

The unformatted form of an input or output statement results in a faster data transfer rate into and out of storage, since no data conversions are performed. When operations are being performed on intermediate files (those which are used internally by the program and which you never see), the use of unformatted data increases program efficiency. In the example below, line 130 is more efficient than line 110:

```
100 dimension a(100)
110 write(20,9)a
120 9format(100e13.3)
130 write(20)a
```

However, when you use unformatted I/O, it is your responsibility to ensure that the output and input data types agree precisely because no checking can be done, and no conversions are performed. Use of UNDEFINED files, with the list-directed I/O statements, provides this checking and conversion, and, although less efficient than unformatted I/O, it is much more efficient than formatted I/O.

## *IF Statements*

The internal representation of REAL numbers is different from the external representation, and so some numbers cannot be represented exactly, although there is a high degree of precision. For instance, the number 1/10 cannot be represented exactly internally, in the same way that 1/3 cannot be represented exactly in decimal notation.

Small errors of this kind can build up when the number is used in calculations, and so you cannot always rely on the precise equality of two numbers, or on a real number being exactly zero.

For instance the statements:

```
100 if(a)20,30,20
110 if(b.eq.c)goto30
```

may not give the expected results if the numbers are the results of computation. In this case a statement of the form:

```
110if(abs(b-c).lt.1e-5)goto30
```

will give the desired result because it allows for a small difference between B and C.

Avoid using tests that rely on the equality of real numbers of differing precision. In such a comparison, the single precision number is padded, internally, with binary zeros, whereas the corresponding part of the double precision number may contain significant information, even if the external representation of the two numbers is the same. For instance, if you code the following, line 130 will branch to statement 10:

```
100 real a*4,x*8
110 a=1e-1
120 x=1d-1
130 if(a-x)10,20,10
```

## Limitations, Restrictions, and Assumptions Connected with VSPC Files

There are various limitations and restrictions imposed on file usage by VSPC and VSPC FORTRAN.

- You cannot have more than 15 files open concurrently.
- You cannot have more than 17 concurrently active FORTRAN unit numbers.
- The maximum size a file can be is included in the FILE command:

  ```
  file f 20
  ```

  which specifies a limit of 20,000 characters. If it is omitted, the maximum editable workspace size is used.
- Your library is limited in size by your user profile.
- Files cannot contain records longer than 4058 characters.
- Files cannot contain more than 8,338,607 records.
- Multiple file names can be associated with the same FORTRAN unit number in VSPC FORTRAN programs by use of the CALL to the OPSYS service subroutine.

# SAMPLE TERMINAL SESSION

This section contains a hypothetical terminal session using the VSPC FORTRAN compiler, and provides a detailed description of that session. The session introduces you to many VSPC features and illustrates the interaction that takes place between the user, the system, and VSPC FORTRAN in a typical application. If you follow the session and its description closely, you will gain a feel for developing a program at a terminal. The sample session illustrates many, but certainly not all, of the commands that are directly relevant to you as a VSPC FORTRAN user. An explanation of all of the relevant VSPC commands is contained in this publication.

A printout of the sample session appears in Figure 11 following the descriptive text. In the session, we have entered all our input in lowercase letters. This makes it easy to distinguish our entries on the printout from the system's messages and responses, which are always typed in uppercase letters.

The program that is written in the sample session generates a magic square. A magic square consists of an array of numbers (none of them repeating) in a square, which when added up by rows, columns, or diagonals, yield the same total. For example, the magic square for the number 45 will add up to 45 if you add along rows, columns, or diagonals. The magic square for 45 looks like this:

```
18    11    16

13    15    17

14    19    12
```

To keep the program simple, it has been coded to generate 3-by-3 magic squares only. In the program, the user is asked to supply, as input, a number for which a magic square is to be generated. Input numbers for these squares must be integers equal to or greater than 15, and divisible by 3. The output from the program is a 3-by-3 square whose sum across, down, and diagonally, is the input number.

An IBM 3767 communications terminal in S/S mode, or an IBM 2741 communication terminal, is the terminal used for the sample session.

## Preliminary Procedures and Sign-On

The first thing you do to start a terminal session is turn on the power according to instructions provided by your installation.

## Start of Session

Entry (1) in the sample session is the VSPC command. It is the first entry of every session; the command is used to identify you to the system.

The system is set up to recognize authorized users by user number, upshift 2 (depending on the terminal type used) and, optionally, a password. Type in your user number and upshift 2 following ID=. VSPC responds with a request for your password which you enter (2) by typing it over the blot characters. Before your first working session, therefore, you should contact the person responsible for authorizing your use of the system, called in this book the VSPC or system administrator. He or she will be able to give you the logon information you need for using VSPC.

It is important to remember that operands must be separated by one or more spaces, commas, or tabs (after you have specified session tabs with the TABSET command).

In response to the logon command, the system types out informational messages followed by READY, and is now prepared to accept any command you choose to enter. The word "FORTRAN" in the first message tells you that your workspace attribute is free-form FORTRAN. The remainder of the message gives time, date, and your user number. The second message may change from day to day.

In order to get message identification numbers printed with any messages we get from VSPC or VSPC FORTRAN, the next command (3) that we type is message id.

Name—the command we enter next (4)—gives a name (SQUARES) to your workspace.

The next command (5), input, requests VSPC to automatically generate line numbers. (Don't confuse VSPC line numbers with FORTRAN statement numbers. Line numbers provide easy reference points for changing or deleting lines at some later time. You don't refer to line numbers in, say, GOTO statements; you refer to FORTRAN statement numbers.)

The system types out the first line number, 00010, (6) on the following line, and is prepared for the first line of input to be typed.

We begin entering our program. After each statement is entered, press the carrier return, and the system types out a line number. Everything goes along smoothly until line 50, (7) at which point, after pressing carrier return, we notice we made an error. In order to correct the error, we press carrier return after VSPC types the 00060 line number (8). This allows us to enter a command. To correct our statement—we're missing a required parenthesis—we use the CHANGE command (9). We specify the line number of the line to be replaced (ignoring leading zeros), the data to be replaced, in quotation marks, and the replacement data, also in quotation marks. By specifying text we tell VSPC to type out the corrected line along with the information message.

To continue writing our program, we enter another input command, and VSPC generates the next higher line number.

(Our READ statements use 5 as the data set reference number, and our WRITE statements use 6. These are the default data set reference numbers for specifying the terminal as the desired I/O device. Because these reference numbers may differ from installation to installation, and because they can be temporarily overidden by the use of ALLOCATE, check with your system administrator before using 5 and 6.)

Line 100 and line 180 are list-directed I/O statements (10). As you will see later, the processor requests input for these statements when the statements are encountered during execution (that is, after you have entered the RUN command).

A good programming practice related to input is to write your own prompting messages for READ statements that are not in list-directed format. This practice is recommended because, for non-list directed READ statements, the system does not signal you that is it ready for your input data. If, however, you code a WRITE statement (11) before each of your READ statements, the typing out of the WRITE statement by the system when you execute your program lets you know that it's ready to receive your input for the READ that follows. Line 30 is an example of a self-prompting WRITE statement. (In this particular program, the WRITE statement serves a somewhat broader function than a simple self-prompt.)

The END statement signals the end of the FORTRAN program (12). If any subroutines or function subprograms were to be entered, they would be entered now, immediately following the END statement of the main program.

We are ready to execute. If you've been looking at the coding, though, you'll have seen some mistakes. The next few lines of the session illustrate a few of the commands available to let you make corrections or changes.

The first error we see in checking over the program is that we've accidentally entered the same statement twice (13), at line 190 and line 200. To get rid of one of these lines, we use the DELETE command. After entering delete 200 (14), we press the carrier return. The system deletes the specified line and is now ready to receive another command.

Next, we notice a spelling error in several of our WRITE statements containing literal data; the word "your" appears twice as "youre." This kind of mistake is easy to correct. We enter the change command (15) to change the misspelling "youre" wherever it appears from line 260 through line 420. Notice what we've specified as the operands of CHANGE. First, the line number at which the search for the word is to begin, then the line number at which the search is to end; next we've entered, enclosed in quotation marks, the word we want to change; then we've typed in the replacement for the word—also enclosed in quotation marks.

Still one more review of the program shows that we didn't write a FORMAT statement for the WRITE at line 480 (16). To add the missing statement, we simply type it, including a VSPC line number, a FORTRAN statement number, and the FORMAT statement (17). VSPC puts this line in sequence between 480 and 490.

Going over the program a last time turns up no further specification errors. Before executing, though, you probably want to see a copy of the program with our modifications and additions incorporated, a "clean" copy of the program. That's what we use the LIST command for. Next, before issuing LIST, we issue a RENUMBER command. RENUMBER causes the VSPC line numbers of the program to be revised (18). Notice that we have used a valid abbreviation for this command. The lines starting with double quotes are not in error. In free-form source, the double quote is used to identify a comment statement. Lines 410 and 420 are not wrong either. They constitute a statement that has been continued. In free-form, continuation is signalled just by a hyphen. In fixed-form, you need a non-blank character in typing position 6 of the next line.

To get the listing, we enter the command name LIST, with the LINE operand (19). VSPC prints out the program, as requested, with the revised line numbers.

The program is printed. In our final check we see no additional errors. Before requesting execution, you should save your source statements (20) for possible later retrieval. If you've saved your program, you can retrieve it (for example, by using the LOAD or RUN command) without wasting time on typing.

To compile and store the program we type in a STORE command (21), the next entry you see; VSPC responds by compiling the source program in the workspace. We thought our program was error-free. However, the compiler finds errors and prints out several error messages (22): (The message text is preceded by a message identification code and the line number of the line in error and followed by VSPC STORE command message.)

It turns out we've neglected to label a statement, referred to in the arithmetic IF statement at line 210 (23). The statement we meant to refer to is the one at line 230 (24).

We now retype line 230, supplying the missing number (25), and re-enter both the SAVE (26) and STORE (27) commands. This time the compiler finds no errors and continues by putting the compiled object program in your own library under the filename SQUAROBJ. Now we execute our object program by specifying run squarobj (28).

Input for our non-list-directed READ statement is entered when the system prints out our self-prompting WRITE statement (29).

Our results are now printed (30); they check out as correct. All horizontal, vertical, and diagonal rows add up to 24 as indicated in Figure 13.

```
1    vspc id=7654321 @
     ENTER PASSWORD
2    [redacted]
     FORTRAN 09:16:20 09/04/76 7654321
     VSPC IS UP 16 HOURS TODAY
     READY
3    message id
     ASU201 READY
4    name squares
     ASU201 READY
5    input
6    00010 " program for generating a magic square.
     00020 " get the name of the person requesting the magic square.
11   00030 write (6, 5)
     00040 5 format (' please enter your name preceded by a blank')
7    00050 read (5,10
8    00060 (press carrier return)
     ASU210 READY
9    change 50 '10' '10)' text
          50 READ (5,10)
     ASU518 1 LINES CHANGED
     input
     00060 10 format ('name                              ')
     00070 "request input number using list i/o to read it in.
11   00080 write (6, 15)
     00090 15 format (' enter a number greater than 14 and divisible by 3')
10   00100 20 read (5,*) num
     00110 "test input number to see if it is 15 or larger.
     00120 if (num-15) 25,40,40
     00130 "number too small. print message using formatted and list i/o and
     00140 "request a new number.
     00150 25 write (6,10)
     00160 write (6,30)
     00170 30 format (' your number must be greater than 14. your number was:')
10   00180 35 write (6,*) num
13   00190 go to 70
13   00200 go to 70
     00210 "test if input number is divisible by 3.
     00220 40 if (mod(num, 3)) 45,55,45
     00230 "not divisible by 3.
     00240 write (6, 10)
     00250 write (6, 50)
```

Figure 13 (Part 1 of 3). Sample Terminal Session

```
     00260 50 format (' youre number must be divisible by 3. your number was:')
     00270 go to 35
     00280 "number is ok. compute magic square.
     00290 55 ib=num/3-4
     00300 ia=ib+7
     00310 ic=ib+5
     00320 id=ib+2
     00330 ie=ib+4
     00340 if=ib+6
     00350 ig=ib+3
     00360 ih=ib+8
     00370 ii=b+1
     00380 "print out magic square.
     00390 write (6, 10)
     00400 write (6, 60) num
     00410 60 format (i8,' is the number whose magic square you requested.-
     00420 youre magic square is:')
     00430 write (6, 65) ia, ib, ic
     00440 write (6, 65) id, ie, if
     00450 write (6, 65) ig, ih, ii
     00460 65 format (3(i10))
     00470 go to 80
16   00480 70 write (6,75)
     00490 go to 20
     00500 80 continue
     00510 stop
12   00520 end
     00530 (presscarrierreturn)
     ASU201 READY
14   delete 200
     ASU517 1 LINES DELETED
15   change 260:420 'youre' 'your'
     260 420
     ASU518 2 LINES CHANGED
17   485 75 format (' please enter a new number according to the rules.')
18   renum
     ASU510 52 LINES RENUMBERED
19   list line
     10 "PROGRAM FOR GENERATING A MAGIC SQUARE.
     20 "GET THE NAME OF THE PERSON REQUESTING THE MAGIC SQUARE.
     30 WRITE (6, 5)
     40 5 FORMAT (' PLEASE ENTER YOUR NAME PRECEDED BY A BLANK')
     50 READ (5,10)
     60 10 FORMAT ('NAME                                  ')
     70 "REQUEST INPUT NUMBER USING LIST I/O TO READ IT IN.
     80 WRITE (6, 15)
     90 15 FORMAT (' ENTER A NUMBER GREATER THAN 14 AND DIVISIBLE BY 3')
     100 20 READ (5,*) NUM
     110 "TEST INPUT NUMBER TO SEE IF IT IS 15 OR LARGER.
     120 IF (NUM-15) 25,40,40
     130 "NUMBER TOO SMALL. PRINT MESSAGE USING FORMATTED AND LIST I/O AND
     140 "REQUEST A NEW NUMBER.
     150 25 WRITE (6,10)
     160 WRITE (6,30)
     170 30 FORMAT (' YOUR NUMBER MUST BE GREATER THAN 14. YOUR NUMBER WAS: ')
     180 35 WRITE (6,*) NUM
     190 GO TO 70
     200 "TEST IF INPUT NUMBER IS DIVISIBLE BY 3.
23   210 40 IF (MOD(NUM, 3)) 45, 55, 45
     220 "NOT DIVISIBLE BY 3.
24   230 WRITE (6, 10)
     240 WRITE (6, 50)
     250 50 FORMAT (' YOUR NUMBER MUST BE DIVISIBLE BY 3. YOUR NUMBER WAS: ')
     260 GO TO 35
```

Figure 13 (Part 2 of 3). Sample Terminal Session

```
     270 "NUMBER IS OK. COMPUTE MAGIC SQUARE.
     280 55 IB=NUM/3-4
     290 IA=IB+7
     300 IC=IB+5
     310 ID=IB+2
     320 IE=IB+4
     330 IF=IB+6
     340 IG=IB+3
     350 IH=IB+8
     360 II=IB+1
     370 "PRINT OUT MAGIC SQUARE.
     380 WRITE (6, 10)
     390 WRITE (6, 60) NUM
     400 60 FORMAT (I8, ' IS THE NUMBER WHOSE MAGIC SQUARE YOU REQUESTED.-
     410 YOUR MAGIC SQUARE IS:')
     420 WRITE (6, 65) IA, IB, IC
     430 WRITE (6,65) ID, IE, IF
     440 WRITE (6,65) IG, IH, II
     450 65 FORMAT (3(I10))
     460 GO TO 80
     470 70 WRITE (6,75)
     480 75 FORMAT (' PLEASE ENTER A NEW NUMBER ACCORDING TO THE RULES.')
     490 GO TO 20
     500 80 CONTINUE
     510 STOP
     520 END
20   save
     ASU201 READY
21   store squarobj
     ASU656 SQUAROBJ 09/04/76 09:16:42
22   00230 W?RITE (6,10)
     AFP002 LABEL
     AFP022 MAIN#-UNDEFINED LABELS
     00045
     AFP000 SEVERITY CODE 8: EXECUTION INHIBITED
     ASU664 COMPILED PROGRAM NOT STORED
     ASU658 TIME 0.05 SECS.
25   230 45 write (6,10)
26   save
     ASU201 READY
27   store squarobj
     ASU656 SQUAROBJ 09/04/76 09:17:59
     ASU657 COMPILED SIZE 6984 BYTES
     ASU658 TIME 0.05 SECS.
28   run squarobj
     ASU656 SQUAROBJ 09/04/76 09:18:19
29   PLEASE ENTER YOUR NAME PRECEDED BY A BLANK
     ? harry
     ENTER A NUMBER GREATER THAN 14 AND DIVISIBLE BY 3
     ? 24
     HARRY
           24 IS THE NUMBER WHOSE MAGIC SQUARE YOU REQUESTED.YOUR MAGIC SQUARE IS:
              11         4         9
30             6         8        10
               7        12         5
     STOP
     ASU658 TIME 0.05 SECS.
     off continue
     ASU211 09:47:18 09/04/76 7654321
     ASU212 CONNECTED 00:55:36
     ASU213 CPU TIME 0
```

Figure 13 (Part 3 of 3). Sample Terminal Session

# APPENDIX A. DATA FILE AND PROGRAM CONVERSION CONSIDERATIONS

Many FORTRAN source programs and data files, created under other than VSPC, can be converted by means of the VSPC Service Program to formats acceptable by VSPC. The VSPC Service Program converts VSPC files containing FORTRAN source programs to and from fixed-length or variable-length sequential data sets, and, for OS/VS, partitioned data set members. It is supplied by IBM as part of the VSPC program product. It runs as a batch program.

OS subroutine load modules also may be brought into VSPC FORTRAN. To do this, the user must convert the load module to a VSPC FORTRAN workspace using the AFPPREP utility, then bring it into VSPC using the IMPORT command of the VSPC Service Program.

## FORTRAN IV Data Files

FORTRAN IV data files must be copied into the appropriate VSPC library by the VSPC Service Program before they can be accessed by VSPC FORTRAN programs.

## FORTRAN IV Source Programs

Syntactically correct FORTRAN IV source programs, which don't require the use of the Debug Facility or extended precision data types, can be compiled and run under VSPC FORTRAN after they have been copied into the appropriate VSPC library by the VSPC Service Program. Calls made by these programs to the DUMP and PDUMP service subroutines will produce error messages. After the message following a CALL to PDUMP, control is passed to the next executable statement, and the program continues. In programs calling DUMP, execution is terminated after the error message.

## CALL-OS FORTRAN Data Files

CALL-OS FORTRAN data files must be placed into normal OS data sets (by means of the CALL-OS utility program, DIBCADBU, with the FORMDATA option) before being copied into the VSPC library by the VSPC Service Program. These files can then be used by VSPC FORTRAN programs.

## CALL-OS FORTRAN Source Programs

CALL-OS FORTRAN source programs containing non-standard FORTRAN statements or syntax will require modification to be run under VSPC. These modifications are in the areas of calls to the OPEN and CLOSE subroutines, special CALL-OS relational operators that must be changed to standard FORTRAN relational operators, percent sign continuation symbols that must be changed to minus signs, and filenames which must conform to the standard set for VSPC filenames. When these modifications are made, the source programs must be placed in OS data sets using DIBCADBU, and then copied into the VSPC library by the VSPC Service Program.

DIBCADBU converts the relational operators to standard FORTRAN operators and converts the entire program to fixed-form FORTRAN, thus eliminating both percent sign and minus sign continuation symbols. The remaining modifications may be made under VSPC, using the VSPC editing facilities.

## OS Subroutine Load Modules

Subroutines not compiled under VSPC FORTRAN must be converted to VSPC workspace format before being brought into the VSPC library. The AFPPREP utility program converts an OS subroutine load module that was linked by a linkage editor into a VSPC workspace that can be imported by the VSPC Service Program using the IMPORT command. The subroutine can then be linked to a FORTRAN main program using the VSPC FORTRAN link processor. The routine being converted must conform to the following rules:

1. It must be self-contained. That is, it must have no unresolved external references. Because of the special requirements of a VSPC foreground processor, the calling sequences, linkage, and register conventions are not compatible with routines compiled on VSPC FORTRAN.
2. No host system SVCs can be issued. This is a restriction placed on all VSPC foreground processors in the interest of system security and performance.
3. No VSPC foreground interface services can be requested. Issuance of such a request can cause the FORTRAN workspace to be physically relocated in storage, requiring special action by the VSPC FORTRAN executor that cannot be achieved in the unknown environment of a non-VSPC FORTRAN subprogram.
4. Code can store no addresses within itself that are required to be correct on subsequent entry. This is because the FORTRAN workspace can be relocated at any time the subprogram is not in control, and could have been moved between entries to the subprogram.
5. The imported load program cannot be the FORTRAN main program. The FORTRAN main program generally contains the bulk of the interface between the executing program and the host environment. There are major differences between the interfaces for VSPC foreground processors and those for programs in other environments. The main program can, of course, be imported in source form and be compiled by the VSPC FORTRAN compiler.

6. The imported module cannot make use of the extended-precision floating point instruction set. This is because the VSPC FORTRAN executor does not support extended precision simulations or arithmetic interrupts resulting from the use of extended-precision instructions in a non-VSPC FORTRAN subprogram.
7. The imported module cannot have any of the following linkage editor attributes:

   TEST
   OVERLAY
   ONLY LOADABLE
   NOT EDITABLE
   SCATTER FORMAT

For further details on the AFPPREP utility, see *VSPC FORTRAN Installation Reference Material.*

# APPENDIX B. DEVICE DEPENDENCIES

In addition to the commands already described, special VSPC commands can be used with the CPT-TWX terminals.

## CPT-TWX Terminals Special Commands

Three commands are unique to the CPT-TWX terminal:

- KEY Command to determine whether or not the keyboard is the source of input
- PUNCH Command to determine whether or not the output is to be punched on paper tape
- TAPE Command to determine if paper tape is the source of input

The LINESIZE command has an operand (x) used only with CPT-TWX terminals to specify the number of idle characters to be inserted following each carrier return.

### *KEY Command*

For CPT-TWX terminals, the KEY command specifies the keyboard as the input source.

| | | |
|---|---|---|
| **KEY** | | No operands allowed. |

At logon, the keyboard is the assumed input source. Whether or not terminal input is read from the keyboard depends on the manual setup of the CPT-TWX terminal. If the manual setup is wrong, input is incorrectly formatted on the console sheet.

When the KEY command is in effect, VSPC automatically causes a line feed after each input line is entered.

The KEY command is valid only from CPT-TWX terminals.

**Note:** The TAPE command specifies the paper tape reader as the input source, reversing the effect of the KEY command.

### *PUNCH Command*

For CPT-TWX terminals, the PUNCH command specifies whether or not terminal output will be punched on paper tape.

| | | |
|---|---|---|
| PUNCH | ON<br>OFF<br>CONTROL | Optional operand—choose one or omit. If omitted, current value is displayed. |

**ON and OFF**

**ON**
specifies that terminal output will be punched on paper tape. When PUNCH ON is in effect, VSPC adds the characters:

```
XOFF RUBOUT RUBOUT RUBOUT
```

at the end of each output line.

**OFF**
specifies that terminal output will not be punched on paper tape. When PUNCH OFF is in effect, no characters are added at the end of a line.

At logon, PUNCH OFF is in effect. Whether or not punched output is being produced, output is printed at the terminal as well.

**CONTROL**
specifies that data sent to the terminal or the attached paper tape punch contains all ASCII characters, including control characters, and that the data should not be edited by VSPC.

Whether or not terminal output is punched on paper tape depends on the manual setup of the terminal. If the setup is wrong, punched output is either incorrectly formatted or nonexistent.

## *TAPE Command*

For CPT-TWX terminals, the TAPE command specifies the paper tape reader as the input source.

| | | |
|---|---|---|
| TAPE | | No operands allowed. |

After the TAPE command is executed, VSPC assumes the paper tape reader as the input source. Immediately before each input line, VSPC sends an X-ON character; after each input line no line feed is sent by VSPC. Whether or not terminal input is read from tape depends on the manual setup of the CPT-TWX terminal. If the setup is wrong and TAPE is specified, lines may be overprinted.

**Note:** The KEY command specifies the keyboard as the input source, reversing the effect of the TAPE command. If no TAPE command has been specified, the keyboard is assumed to be the input source.

# APPENDIX C. VSPC TERMINALS QUICK REFERENCE CHART

| Action | TERMINALS | | | | | | CPT-TWX Mod 33/35 |
|---|---|---|---|---|---|---|---|
| | 3270 | 2741* | 3767 | | 3770 | 1050* | |
| Set switches for computer connection | 3275/3277: Pull out OFF/PUSH<br><br>3276/3278: Push top (I) of power switch | COM/LCL=COM<br>ON/OFF=ON | Keylock=ON<br>COMM/LOCAL=COMM<br>AUTO/OFF=AUTO (SDLC only)<br>EDIT/OFF=OFF (SDLC only)<br>AUTO VIEW/OFF=as desired<br>DOUBLE/SINGLE SPACE=as desired<br>DATA/TALK=ask administrator<br>DIAL/DISC=ask administrator<br>SDLC/SS=ask administrator<br>Primary/Secondary=as desired<br>CALC/OFF=OFF<br>TEST/OFF=OFF<br>POWER/OFF=POWER<br>Press SYS REQ | | Keylock=ON<br>POWER=ON<br>HOLDPRINT=OFF<br>EXTENT/ALARM=OFF<br>PUNCH=OFF<br>DISK=OFF<br>UPDATE/MONITOR=OFF<br>INTRP=OFF<br>HALF SPEED=OFF or NORMAL<br>SDLC/BSC=SDLC<br>AUTO=ON<br>(CODE/1 if 3771 or 3773)<br>Press SYS REQ | MAIN-LINE=ON<br>SYSTEM=ATTEND<br>MASTER=ON<br>PRINTER 1=SEND/REC<br>PRINTER 2=HOME<br>KEYBOARD=SEND<br>PUNCH=NORMAL<br>SYSTEM=PROGRAM<br>EOB=AUTO<br>SYSTEM=UP<br>All others=OFF | Press ORIG button<br>Model 35: Press K button<br>On high-pitched tone, press CTRL, WRU |
| Terminal ready to accept logon | SYSTEM AVAILABLE<br>Cursor appears | Keyboard unlocks | DATA SET READY, PROCEED lights on<br>ONLINE (leased line) | | PROCEED, KBD, LINE, and AUTO lights on | POWER, PROCEED lights on | Key mode: paper advance optional ? printed |
| | | | S/S | SDLC | | | |
| Logon: VSPC ID= | usernum | usernum x | usernum x | usernum | usernum | usernum x | usernum |
| End a line | ENTER | RETURN | ↵ (return) | ↵ (return) | ↵(return) | RETURN<br>ALTN CODING + EOT | RETURN |
| Reset terminal | RESET | --- | RESET | RESET | CODE and RESET | REQUEST | BRK-RLS<br>Model 35: K |
| Terminal ready to accept input | READY<br>Line number<br>Cursor appears<br>SYSTEM READY | READY<br>Line number<br>Unlocks keyboard | READY<br>Line number<br>ON LINE and PROCEED lights on | READY<br>Line number<br>ON LINE and PROCEED lights on | READY<br>Line number<br>ON LINE and PROCEED light on | READY<br>Line number<br>Unlocks keyboard<br>PROCEED light on | READY<br>Paper advance<br>Line number<br>Stops noise<br>Bell |
| Correct current line | Position cursor, ERASE EOF | BKSP, ATTN, retype, RETURN | ← (backspace), ATTN, retype, ↵(return) | ← (backspace), INDEX, retype, ↵(return) | ←(backspace) INDEX, retype, ↵(return) | BACKSPACE, ATTENTION, retype, RETURN | Backarrow or underline, retype, RETURN |
| Delete current line(s) | ERASE INPUT | ATTN | ATTN | BUFFER RTN, CNCL | BUFFER RTN, CNCL | ATTENTION | CTRL/X |
| Send signal to system | PA1 | ATTN | ATTN | ATTN | ATTN | ATTENTION | BREAK |
| Escape from input | PA1 | ATTN | ATTN | ATTN | ATTN | ATTENTION | BREAK |
| End output | PA2 | ATTN | ATTN | CNCL | CNCL | ATTENTION | BREAK |
| Logoff | OFF | OFF | OFF | OFF | OFF | OFF | OFF |
| Turn off terminal | Push OFF/ PUSH or (O) | ON/OFF=OFF | POWER=OFF<br>Keylock=OFF | POWER=OFF<br>Keylock=OFF | (unnecessary) | MAIN-LINE=OFF | CLR |
| Default line length | 80 | 120 | 120 | 132 | 132 | 120 | 72 |
| Invalid character prints as | "<br>(double quote) | ⊠<br>(N backspace Z) | ⊠<br>(N backspace Z) | –<br>(hyphen) | –<br>(hyphen) | ⊠<br>(N backspace Z) | "<br>(double quote) |

*Terminals controlled by the Multiple Terminal Access (MTA) facility of the Network Control Program (NCP) may need special logon procedures. See your VSPC administrator.

# APPENDIX D. MATHEMATICAL FUNCTION ACCURACY STATISTICS

This appendix contains accuracy statistics for the explicitly called mathematical functions. These statistics are presented in Figure 14, and are arranged in alphabetic order, according to the entry names.

| Entry Name | Argument Range | Sample E/U | Accuracy Figures | | | |
|---|---|---|---|---|---|---|
| | | | Relative | | Absolute | |
| | | | M ($\epsilon$) | $\sigma$ ($\epsilon$) | M (E) | $\sigma$ (E) |
| ALGAMA | $0 < X < 0.5$ | U | $1.16 \times 10^{-6}$ | $3.54 \times 10^{-7}$ | | |
| | $0.5 \leqq X < 3.0$ | U | | | $9.43 \times 10^{-7}$ | $3.42 \times 10^{-7}$ |
| | $3.0 \leqq X < 8.0$ | U | $1.25 \times 10^{-6}$ | $3.04 \times 10^{-7}$ | | |
| | $8.0 \leqq X < 16.0$ | U | $1.18 \times 10^{-6}$ | $3.80 \times 10^{-7}$ | | |
| | $16.0 \leqq X < 500.0$ | U | $9.85 \times 10^{-7}$ | $1.90 \times 10^{-7}$ | | |
| ALOG | $0.5 \leqq X \leqq 1.5$ | U | | | $6.85 \times 10^{-8}$ | $2.33 \times 10^{-8}$ |
| | $X < 0.5, X > 1.5$ | E | $8.32 \times 10^{-7}$ | $1.19 \times 10^{-7}$ | | |
| ALOG 10 | $0.5 \leqq X \leqq 1.5$ | U | | | $7.13 \times 10^{-8}$ | $2.26 \times 10^{-8}$ |
| | $X < 0.5, X > 1.5$ | E | $1.05 \times 10^{-6}$ | $2.17 \times 10^{-7}$ | | |
| ARCOS | $-1 \leqq X \leqq +1$ | U | $8.85 \times 10^{-7}$ | $3.19 \times 10^{-7}$ | | |
| ARSIN | $-1 \leqq X \leqq +1$ | U | $9.34 \times 10^{-7}$ | $2.06 \times 10^{-7}$ | | |
| ATAN | The full range | Note 7 | $1.01 \times 10^{-6}$ | $4.68 \times 10^{-7}$ | | |
| ATAN 2 | The full range | Note 7 | $1.01 \times 10^{-6}$ | $4.68 \times 10^{-7}$ | | |
| CABS | The full range | Note 1 | $9.15 \times 10^{-7}$ | $2.00 \times 10^{-7}$ | | |
| CCOS | $\|X_1\| \leqq 10, \|X_2\| \leqq 1$ | U | $2.50 \times 10^{-6}$ See Note 2 | $7.66 \times 10^{-7}$ | | |
| CDABS | The full range | Note 1 | $2.03 \times 10^{-16}$ | $4.83 \times 10^{-17}$ | | |
| CDCOS | $\|X_1\| \leqq 10, \|X_2\| \leqq 1$ | U | $3.98 \times 10^{-15}$ See Note 3 | $2.50 \times 10^{-16}$ | | |
| CDEXP | $\|X_1\| \leqq 1, \|X_2\| \leqq \pi/2$ | U | $3.76 \times 10^{-16}$ | $1.10 \times 10^{-16}$ | | |
| | $\|X_1\| \leqq 20, \|X_2\| \leqq 20$ | U | $2.74 \times 10^{-15}$ | $9.64 \times 10^{-16}$ | | |
| CDLOG | The full range except (1 + Oi) | Note 1 | $2.72 \times 10^{-16}$ | $5.38 \times 10^{-17}$ | | |
| CDSIN | $\|X_1\| \leqq 10, \|X_2\| \leqq 1$ | U | $2.35 \times 10^{-15}$ See Note 4 | $2.25 \times 10^{-16}$ | | |
| CDSQRT | The full range | Note 1 | $1.76 \times 10^{-16}$ | $4.06 \times 10^{-17}$ | | |
| CEXP | $\|X_1\| \leqq 170, \|X_2\| \leqq \pi/2$ | U | $9.93 \times 10^{-7}$ | $2.67 \times 10^{-7}$ | | |
| | $\|X_1\| \leqq 170,$ $\pi/2 < \|X_2\| \leqq 20$ | U | $1.07 \times 10^{-6}$ | $2.73 \times 10^{-7}$ | | |
| CLOG | The full range except (1 + Oi) | Note 1 | $7.15 \times 10^{-7}$ | $1.36 \times 10^{-7}$ | | |
| COS | $0 \leqq X \leqq \pi$ | U | | | $1.19 \times 10^{-7}$ | $4.60 \times 10^{-8}$ |
| | $-10 \leqq X < 0,$ $\pi < X \leqq 10$ | U | | | $1.28 \times 10^{-7}$ | $4.55 \times 10^{-8}$ |
| | $10 < \|X\| \leqq 100$ | U | | | $1.14 \times 10^{-7}$ | $4.60 \times 10^{-8}$ |
| COSH | $-5 \leqq X \leqq +5$ | U | $1.27 \times 10^{-6}$ | $2.63 \times 10^{-7}$ | | |
| COTAN | $\|X\| \leqq \pi/4$ | U | $1.07 \times 10^{-6}$ | $3.58 \times 10^{-7}$ | | |
| | $\pi/4 < \|X\| \leqq \pi/2$ | U | $1.40 \times 10^{-6}$ (Note 5) | $2.56 \times 10^{-7}$ | | |
| | $\pi/2 < \|X\| \leqq 10$ | U | $1.30 \times 10^{-6}$ (Note 5) | $3.11 \times 10^{-7}$ | | |
| | $10 < \|X\| \leqq 100$ | U | $1.49 \times 10^{-6}$ (Note 5) | $3.15 \times 10^{-7}$ | | |
| NOTES: (See end of table.) | | | | | | |

Figure 14 (Part 1 of 4). Accuracy Figures

| Entry Name | Argument Range | Sample E/U | Accuracy Figures | | | |
|---|---|---|---|---|---|---|
| | | | Relative | | Absolute | |
| | | | M (ε) | σ (ε) | M (E) | σ (E) |
| CSIN | $\lvert X_1\rvert \leqq 10, \lvert X_2\rvert \leqq 1$ | U | $1.92 \times 10^{-6}$ See Note 6 | $7.38 \times 10^{-7}$ | | |
| CSQRT | The full range | Note 1 | $7.00 \times 10^{-7}$ | $1.71 \times 10^{-7}$ | | |
| DARCOS | $\lvert X\rvert \leqq 1$ | U | $2.07 \times 10^{-16}$ | $7.05 \times 10^{-17}$ | | |
| DARSIN | $\lvert X\rvert \leqq 1$ | U | $2.04 \times 10^{-16}$ | $5.15 \times 10^{-17}$ | | |
| DATAN | The full range | Note 7 | $2.18 \times 10^{-16}$ | $7.04 \times 10^{-17}$ | | |
| DATAN2 | The full range | Note 7 | $2.18 \times 10^{-16}$ | $7.04 \times 10^{-17}$ | | |
| DCOS | $0 \leqq X \leqq \pi$ | U | | | $1.79 \times 10^{-16}$ | $6.53 \times 10^{-17}$ |
| | $-10 \leqq X < 0$, $\pi < X \leqq 10$ | U | | | $1.75 \times 10^{-16}$ | $5.93 \times 10^{-17}$ |
| | $10 < X \leqq 100$ | U | | | $2.64 \times 10^{-15}$ | $1.01 \times 10^{-15}$ |
| DCOSH | $\lvert X\rvert \leqq 5$ | U | $3.63 \times 10^{-16}$ | $9.05 \times 10^{-17}$ | | |
| DCOTAN | $\lvert X\rvert \leqq \pi/4$ | U | $2.46 \times 10^{-16}$ (Note 5) | $8.79 \times 10^{-17}$ | | |
| | $\pi/4 < \lvert X\rvert \leqq \pi/2$ | U | $2.78 \times 10^{-13}$ (Note 5) | $8.61 \times 10^{-15}$ | | |
| | $\pi/2 < \lvert X\rvert \leqq 10$ | U | $5.40 \times 10^{-13}$ (Note 5) | $1.13 \times 10^{-14}$ | | |
| | $10 < \lvert X\rvert \leqq 100$ | U | $8.61 \times 10^{-13}$ (Note 5) | $4.61 \times 10^{-14}$ | | |
| DERF | $\lvert X\rvert \leqq 1.0$ | U | $1.89 \times 10^{-16}$ | $2.60 \times 10^{-17}$ | | |
| | $1.0 < \lvert X\rvert \leqq 2.04$ | U | $2.87 \times 10^{-17}$ | $9.84 \times 10^{-18}$ | | |
| | $2.04 < \lvert X\rvert < 6.092$ | U | $1.39 \times 10^{-17}$ | $8.02 \times 10^{-18}$ | | |
| DERFC | $-6 < X < 0$ | U | $2.08 \times 10^{-16}$ | $6.52 \times 10^{-17}$ | | |
| | $0 \leqq X \leqq 1$ | U | $1.40 \times 10^{-16}$ | $2.59 \times 10^{-17}$ | | |
| | $1 < X \leqq 2.04$ | U | $4.11 \times 10^{-16}$ | $8.86 \times 10^{-17}$ | | |
| | $2.04 < X < 4$ | U | $3.26 \times 10^{-16}$ | $8.65 \times 10^{-17}$ | | |
| | $4 \leqq X < 13.3$ | U | $3.51 \times 10^{-15}$ | $1.96 \times 10^{-15}$ | | |
| DEXP | $\lvert X\rvert \leqq 1$ | U | $2.04 \times 10^{-16}$ | $5.43 \times 10^{-17}$ | | |
| | $1 < \lvert X\rvert \leqq 20$ | U | $2.03 \times 10^{-16}$ | $4.87 \times 10^{-17}$ | | |
| | $20 < \lvert X\rvert \leqq 170$ | U | $1.97 \times 10^{-16}$ | $4.98 \times 10^{-17}$ | | |
| DGAMMA | $0 < X < 1$ | U | $2.14 \times 10^{-16}$ | $7.84 \times 10^{-17}$ | | |
| | $1 \leqq X \leqq 2$ | U | $2.52 \times 10^{-17}$ | $6.07 \times 10^{-18}$ | | |
| | $2 < X < 4$ | U | $2.21 \times 10^{-16}$ | $8.49 \times 10^{-17}$ | | |
| | $4 \leqq X < 8$ | U | $5.05 \times 10^{-16}$ | $1.90 \times 10^{-16}$ | | |
| | $8 \leqq X < 16$ | U | $6.02 \times 10^{-15}$ | $1.78 \times 10^{-15}$ | | |
| | $16 \leqq X < 57$ | U | $1.16 \times 10^{-14}$ | $4.11 \times 10^{-15}$ | | |
| DLGAMA | $0 < X \leqq 0.5$ | U | $2.77 \times 10^{-16}$ | $9.75 \times 10^{-17}$ | | |
| | $0.5 < X < 3$ | U | | | $2.24 \times 10^{-16}$ | $7.77 \times 10^{-17}$ |
| | $3 \leqq X < 8$ | U | $2.89 \times 10^{-16}$ | $8.80 \times 10^{-17}$ | | |
| | $8 \leqq X < 16$ | U | $2.86 \times 10^{-16}$ | $8.92 \times 10^{-17}$ | | |
| | $16 \leqq X < 500$ | U | $1.99 \times 10^{-16}$ | $3.93 \times 10^{-17}$ | | |
| DLOG | $0.5 \leqq X \leqq 1.5$ | U | | | $4.60 \times 10^{-17}$ | $2.09 \times 10^{-17}$ |
| | $X < 0.5, X > 1.5$ | E | $3.32 \times 10^{-16}$ | $5.52 \times 10^{-17}$ | | |
| NOTES: (See end of table.) | | | | | | |

Figure 14 (Part 2 of 4). Accuracy Figures

| Entry Name | Argument Range | Sample E/U | Accuracy Figures | | | |
|---|---|---|---|---|---|---|
| | | | Relative | | Absolute | |
| | | | M (ε) | σ (ε) | M (E) | σ (E) |
| DLOG10 | $0.5 \leqq X \leqq 1.5$ | U | | | $2.73 \times 10^{-17}$ | $1.07 \times 10^{-17}$ |
| | $X < 0.5, X > 1.5$ | E | $3.02 \times 10^{-16}$ | $6.65 \times 10^{-17}$ | | |
| DSIN | $\|X\| \leqq \pi/2$ | U | $3.60 \times 10^{-16}$ | $4.82 \times 10^{-17}$ | $7.74 \times 10^{-17}$ | $1.98 \times 10^{-17}$ |
| | $\pi/2 < \|X\| \leqq 10$ | U | | | $1.64 \times 10^{-16}$ | $6.49 \times 10^{-17}$ |
| | $10 < \|X\| \leqq 100$ | U | | | $2.68 \times 10^{-15}$ | $1.03 \times 10^{-15}$ |
| DSINH | $\|X\| \leqq 0.88137$ | U | $2.06 \times 10^{-16}$ | $3.74 \times 10^{-17}$ | | |
| | $0.88137 < \|X\| \leqq 5$ | U | $3.80 \times 10^{-16}$ | $9.21 \times 10^{-17}$ | | |
| DSQRT | The full range | E | $1.06 \times 10^{-16}$ | $2.16 \times 10^{-17}$ | | |
| DTAN | $\|X\| \leqq \pi/4$ | U | $3.41 \times 10^{-16}$ | $6.27 \times 10^{-17}$ | | |
| | $\pi/4 < \|X\| \leqq \pi/2$ | U | $1.43 \times 10^{-12}$ (Note 5) | $2.95 \times 10^{-14}$ | | |
| | $\pi/2 < \|X\| \leqq 10$ | U | $2.78 \times 10^{-13}$ (Note 5) | $7.23 \times 10^{-15}$ | | |
| | $10 < \|X\| \leqq 100$ | U | $3.79 \times 10^{-12}$ (Note 5) | $9.50 \times 10^{-14}$ | | |
| DTANH | $\|X\| \leqq 0.54931$ | U | $1.91 \times 10^{-16}$ | $3.86 \times 10^{-17}$ | | |
| | $0.54931 < \|X\| \leqq 5$ | U | $1.54 \times 10^{-16}$ | $1.87 \times 10^{-17}$ | | |
| ERF | $\|X\| \leqq 1.0$ | U | $8.16 \times 10^{-7}$ | $1.10 \times 10^{-7}$ | | |
| | $1.0 < \|X\| \leqq 2.04$ | U | $1.13 \times 10^{-7}$ | $3.70 \times 10^{-8}$ | | |
| | $2.04 < \|X\| \leqq 3.9192$ | U | $5.95 \times 10^{-8}$ | $3.41 \times 10^{-8}$ | | |
| ERFC | $-3.8 < X < 0$ | U | $9.10 \times 10^{-7}$ | $2.96 \times 10^{-7}$ | | |
| | $0 \leqq X \leqq 1.0$ | U | $7.42 \times 10^{-7}$ | $1.27 \times 10^{-7}$ | | |
| | $1.0 < X \leqq 2.04$ | U | $1.54 \times 10^{-6}$ | $3.78 \times 10^{-7}$ | | |
| | $2.04 < X \leqq 4.0$ | U | $2.28 \times 10^{-6}$ | $3.70 \times 10^{-7}$ | | |
| | $4.0 < X \leqq 13.3$ | U | $1.55 \times 10^{-5}$ | $8.57 \times 10^{-6}$ | | |
| EXP | $\|X\| \leqq 1$ | U | $4.65 \times 10^{-7}$ | $1.28 \times 10^{-7}$ | | |
| | $1 < \|X\| \leqq 170$ | U | $4.42 \times 10^{-7}$ | $1.15 \times 10^{-7}$ | | |
| GAMMA | $0 < X < 1.0$ | U | $9.86 \times 10^{-7}$ | $3.66 \times 10^{-7}$ | | |
| | $1.0 \leqq X \leqq 2.0$ | U | $1.13 \times 10^{-7}$ | $3.22 \times 10^{-8}$ | | |
| | $2.0 < X \leqq 4.0$ | U | $9.47 \times 10^{-7}$ | $3.79 \times 10^{-7}$ | | |
| | $4.0 < X < 8.0$ | U | $2.26 \times 10^{-6}$ | $8.32 \times 10^{-7}$ | | |
| | $8.0 \leqq X \leqq 16.0$ | U | $2.20 \times 10^{-5}$ | $7.61 \times 10^{-6}$ | | |
| | $16.0 < X \leqq 57.0$ | U | $4.62 \times 10^{-5}$ | $1.51 \times 10^{-5}$ | | |
| SIN | $\|X\| \leqq \pi/2$ | U | $1.32 \times 10^{-6}$ | $1.82 \times 10^{-7}$ | $1.18 \times 10^{-7}$ | $4.55 \times 10^{-8}$ |
| | $\pi/2 < \|X\| \leqq 10$ | U | | | $1.15 \times 10^{-7}$ | $4.64 \times 10^{-8}$ |
| | $10 < \|X\| \leqq 100$ | U | | | $1.28 \times 10^{-7}$ | $4.52 \times 10^{-8}$ |
| SINH | $-5 \leqq X \leqq +5$ | U | $1.26 \times 10^{-6}$ | $2.17 \times 10^{-7}$ | | |
| SQRT | The full range | E | $4.45 \times 10^{-7}$ | $8.43 \times 10^{-8}$ | | |
| NOTES: (See end of table) | | | | | | |

Figure 14 (Part 3 of 4). Accuracy Figures

| Entry Name | Argument Range | Sample E/U | Accuracy Figures | | | |
|---|---|---|---|---|---|---|
| | | | Relative | | Absolute | |
| | | | M (ε) | σ (ε) | M (E) | σ (E) |
| TAN | $\|X\| \leqq \pi/4$ | U | $1.71 \times 10^{-6}$ | $2.64 \times 10^{-7}$ | | |
| | $\pi/4 < \|X\| \leqq \pi/2$ | U | $1.05 \times 10^{-6}$ (Note 5) | $3.59 \times 10^{-7}$ | | |
| | $\pi/2 < \|X\| \leqq 10$ | U | $6.49 \times 10^{-6}$ (Note 5) | $3.38 \times 10^{-7}$ | | |
| | $10 < \|X\| \leqq 100$ | U | $1.57 \times 10^{-6}$ (Note 5) | $3.07 \times 10^{-7}$ | | |
| TANH | $\|X\| \leqq 0.7$ | U | $8.48 \times 10^{-7}$ | $1.48 \times 10^{-7}$ | | |
| | $0.7 < \|X\| \leqq 5$ | U | $2.44 \times 10^{-7}$ | $4.23 \times 10^{-8}$ | | |

NOTES:

[1] The distribution of sample arguments upon which these statistics are based is exponential radially and is uniform around the origin.

[2] The maximum relative error cited for the CCOS function is based upon a set of 2000 random arguments within the range. In the immediate proximity of the points $\left(n + \frac{1}{2}\right) \pi + 0i$ (where $n = 0, \pm 1, \pm 2, \ldots,$) the relative error can be quite high, although the absolute error is small.

[3] The maximum relative error cited for the CDCOS function is based upon a set of 1500 random arguments within the range. In the immediately proximity of the points $\left(n + \frac{1}{2}\right) \pi + 0i$ (where $n = 0, \pm 1, \pm 2, \ldots,$) the relative error can be quite high, although the absolute error is small.

[4] The maximum relative error cited for the CDSIN function is based upon a set of 1500 random arguments within the range. In the immediate proximity of the points $n\pi + 0i$ (where $n = \pm 1, \pm 2, \ldots,$) the relative error can be quite high, although the absolute error is small.

[5] The figures cited as the maximum relative errors are those encountered in a sample of 2500 random arguments within the respective ranges.

[6] The maximum relative error cited for the CSIN function is based upon a set of 2000 random arguments within the range. In the immediate proximity of the points $n\pi + 0i$ (where $n = \pm 1, \pm 2, \ldots,$) the relative error can be quite high, although the absolute error is small.

[7] The sample arguments were tangents of numbers uniformly distributed between $-\frac{\pi}{2}$ and $+\frac{\pi}{2}$.

Figure 14 (Part 4 of 4). Accuracy Figures

The following information is given in the table:

*Entry Name:* This column gives the function name.

*Argument Range:* This column gives the argument range used to obtain the accuracy figures. For each function, accuracy figures are given for one or more representative segments within the valid argument range. In each case, the figures given are the most meaningful to the function and range under consideration.

The maximum relative error and standard deviation of the relative error are generally useful and revealing statistics. They are useless, however, for the range of a function where its value becomes 0, because the slightest error in the argument can cause an unpredictable fluctuation in the magnitude of the answer. When a small argument error would have this effect, the maximum absolute error and standard deviation of the absolute error are given for the range. For example, absolute error is given for sin ($\chi$) for values $\chi$ near $\pi$.

*Sample:* This column indicates the type of sample used for the accuracy figures. The type of sample depends on the function and range under consideration. The statistics may be based either on an exponentially distributed (E) argument sample or a uniformly distributed (U) argument sample.

*Accuracy Figures:* This column gives accuracy figures for one or more representative segments within the valid argument range. The accuracy figures supplied are based upon the assumption that the arguments are perfect (that is, without error and, therefore, having no error propagation effect on the answers). The only errors in the answers are those introduced by the functions. See Figure 15 for the symbols used in the presentation of accuracy figures.

| | |
|---|---|
| $M(\epsilon) = \text{Max}\left\lvert \frac{f(x) - g(x)}{f(x)} \right\rvert$ | The maximum relative error produced during testing. |
| $\sigma(\epsilon) = \sqrt{\frac{1}{N}\sum_i \left\lvert \frac{f(x_i) - g(x_i)}{f(x_i)} \right\rvert^2}$ | The standard deviation (root-mean-square)of the relative error. |
| $M(E) = \text{Max}\,\lvert f(x) - g(x) \rvert$ | The maximum absolute error produced during testing. |
| $\sigma(E) = \sqrt{\frac{1}{N}\sum_i \lvert f(x_i) - g(x_i) \rvert^2}$ | The standard deviation (root-mean-square) of the absolute error. |

Figure 15. Accuracy Figures Symbols

In case of complex functions, the absolute value signs used in the above definitions are to mean the complex absolute values. In the formulas for the standard deviation, (N represents the total number of arguments in the sample; $\iota$ is a subscript that varies from 1 to N.

Test ranges, where they do not cover the entire legal range of a function, were selected so that users may infer from the accuracy figures presented, the trend of errors as an argument moves away from the principal range. The accuracy of the answer deteriorates substantially as the argument approaches the limit of the permitted range in several of the functions. This is particularly true for trigonometric functions. An error generated by any of these functions, however, is at worst comparable in order of magnitude to the effect of the inherent rounding error of the argument.

# APPENDIX E. BATCH CONSIDERATIONS

VSPC provides a Job Entry facility by means of which you can submit batch jobs to the VS operating system under which VSPC is running. See *VS Personal Computing (VSPC) General User's Guide and Command Language* for a full description of the facilities available.

The VSPC command language provides a simple set of commands that allow you to submit jobs from your terminal to your computing center for conventional batch processing. While such jobs are being interpreted and executed, you can use your terminal for your interactive time-sharing applications.

Batch processing from your terminal is only slightly different from your interactive system. You use Job Control Language, any programming language, and data; all of which have been created as VSPC files. The 'host' system then executes the job exactly as if it had been submitted directly, instead of via VSPC.

Three requirements must be met for batch processing:

1. The Remote Job Control option must be installed with your VSPC subsystem.
2. Your profile must allow you to use Remote Job Control commands.
3. The VSPC operator must be allowing Remote Job Control at the time you submit your job.

See *VS Personal Computing (VSPC) General User's Guide and Command Language* for details about the job entry commands necessary to do this.

If you have a program which you have been running under VSPC FORTRAN, and you wish to SUBMIT it to run under one of the FORTRAN compilers available in the batch system, there are some things of which you should be aware. They are described below.

## Use of the OPSYS Subroutine

Not all FORTRAN libraries have an OPSYS subroutine. In those that do have an OPSYS, the functions supported are not the same as in VSPC FORTRAN, because they are operating system dependent.

If your program contains calls to OPSYS, and you need these functions in the batch system, you must provide a subroutine to perform the necessary function, and make it available in the linkage editor step of your batch job.

## Language Supported by VSPC FORTRAN

Not all FORTRAN compilers accept free-format source programs.

Not all FORTRAN libraries support list-directed I/O.

Not all FORTRAN compilers support FORTRAN II I/O statements.

Not all FORTRAN libraries handle the PAUSE statement in the same way.

# APPENDIX F. ERROR MESSAGES

Whenever VSPC FORTRAN detects an error or possible error in your program, it sends a message to your termiral.

All such messages have prefixes which do not appear unless you have issued the:

```
message id
```

command. However, all messages in this section have the prefix included.

The prefix consists of two parts: the letters AFP, which identifies the message as coming from VSPC FORTRAN, and a three-digit number which is unique within VSPC FORTRAN messages.

These messages are divided into three groups:

| Messages | Contents |
|---|---|
| 000-059 | Errors in your source program which are detected by the compiler. |
| 060-099 | Errors detected in initialization which do not allow compilation or execution to start. |
| 110-999 | Errors in the execution of your program which are detected by the library. |

## VSPC Error Messages

When VSPC finds an error in a command, it issues an error message and you need to retype the entire command. For example:

```
load 123
ASU280 INVALID FILENAME '123'
```

If there are errors in the operation of VSPC itself, you may get one of the following VSPC messages:

```
ASU200 SYSTEM PROBLEM 0
ASU207 SYSTEM PROBLEM n
```

where n is a nonzero number. Save this number as information for your system administrator.

Explanations of VSPC terminal messages are listed in a computer printout that can be obtained by the procedure described in the *VSPC General User's Guide and Command Language*. These explanations can also be displayed at your terminal using VSPC AID: enter ?ASU203 for an explanation of message ASU203, or ?n for an explanation of the nth preceding VSPC message.

## Compiler Messages

Whenever the compiler detects an error or possible error in your source program, it writes a message at the terminal. Errors fall into three classes: errors in the structure of a single statement; errors in the structure of a subprogram; and errors in the structure of the complete program.

When the compiler detects syntax or semantic errors in a statement, it notes the statement at your terminal, and for each error it inserts a '?' at the point the error was detected. Then it lists all the errors it could find in the statement. Thus:

```
00100       WRITE(6,10)(ARRAY1?2(i=?1,10)
AFP003 NAME LENGTH
AFP013 SYNTAX
```

When the statement is printed, non-significant blanks do not appear. If your program is protected NOREAD, only the line number is printed.

Some errors mean that the rest of the statement cannot be processed, making it possible to have some errors hidden by other errors occurring earlier in the statement.

When the compiler detects errors in the structure of a subprogram, it writes a message which includes the subprogram name, and unless your program is protected NOREAD, follows it with a list of variables or labels to which the message refers. The following is an example of what you could see printed:

```
AFP022   func UNDEFINED LABELS
01020
13490
```

When the compiler detects errors in the structure of your program, it writes messages indicating the problem to the terminal, such as:

```
AFP044   DSPQT SUBPROGRAM NOT IN LIBRARY
```

The compiler only checks for this type of error when there are no errors of code 8 or higher in the source program.

Messages appear subprogram by subprogram, with syntax errors preceding structure errors for each subprogram, and messages for the structure of the program appearing last.

If there are code 8 or higher errors detected in the source, the compiler does not allow execution to continue, and sends the message:

```
AFP000   SEVERITY CODE 8: EXECUTION INHIBITED
```

**The message prefixes, such as AFP000, only appear if they have been explicitly requested through the:**

```
message id
```

**command; otherwise the message appears as:**

```
SEVERITY CODE 8: EXECUTION INHIBITED
```

**The codes associated with compiler messages have the following meaning:**

| **Code** | **Meaning** |
|---|---|
| 0 | The error has been corrected, or has no effect on the interpretation of the source program. |
| 4 | The error has been corrected or bypassed, but it may cause errors or incorrect results during execution. |
| 8 | The error prevents execution of the program, and compilation continues. |
| 16 | The error is uncorrectable, and compilation of the subprogram is terminated. Compilation continues with the next subprogram. All further processing is inhibited. |

**AFP000 SEVERITY CODE *n*: EXECUTION INHIBITED**

*Explanation:* The compiler has detected severe errors in the program, the most severe of which was code *n*. Execution has been inhibited. Messages will have been previously written describing these errors.

Code 0

**AFP001 ILLEGAL TYPE**

*Explanation:* A constant or variable has been used as the argument to a FORTRAN-supplied function which requires arguments of a different type.

Code 8

**AFP002 LABEL**

*Explanation:* The statement follows an unconditional transfer of control, yet does not have a statement label. Thus, it can never be executed.

Code 0

**AFP003 NAME LENGTH**

*Explanation:* An identifier contains more than six characters. The first six are used.

Code 0

**AFP004 COMMA**

*Explanation:* A comma has been omitted, but the result is unambiguous.

Code 0

**AFP005 ILLEGAL LABEL**

*Explanation:* A label has been included on a statement which cannot be labeled, or a label reference is to an incorrect statement type.

Code 8

**AFP006 DUPLICATE LABEL**

*Explanation:* The label specified has been previously defined on another statement.

Code 8

**AFP007 ID CONFLICT**

*Explanation:* The use of a variable or subprogram name conflicts with a previous use or specification.

Code 8

**AFP008 ALLOCATION**

*Explanation:* Storage cannot be allocated for the specified variable or array because there is a conflict with a previous explicit or implicit specification.

Code 8

**AFP009 ORDER**

*Explanation:* The source statements of a subprogram are used in an improper sequence.

Code 8

**AFP010 SIZE**

*Explanation:* A number does not conform to the legal values for its use.

Code 8

**AFP011 UNDIMENSIONED**

*Explanation:* A subscripted reference has been made to a name which has not been previously dimensioned.

Code 8

**AFP012 SUBSCRIPT**

*Explanation:* The number of subscripts in an array reference conflicts with the number of dimensions of the array.

Code 8

**AFP013 SYNTAX**

*Explanation:* The statement does not conform to the syntax of the FORTRAN IV or FLINK language.

Code 8

**AFP014 CONVERTED**

*Explanation:* The mode of a variable in a specification statement differs from the mode of the constant which is its initial value. The constant is converted to the correct mode.

Code 0

**AFP015 NO END STMT**

*Explanation:* The source program does not end with an END statement. One is supplied.

Code 0

**AFP016 ILLEGAL STMT**

*Explanation:* The context in which the statement is used is illegal.

Code 8

**AFP017 WARNING: ILLEGAL STMT**

*Explantion:* A RETURN *i* statement has been used in a function subprogram. *i* is ignored.

Code 0

**AFP018 NUMBER OF ARGUMENTS**

*Explanation:* A library function name has been referenced with an incorrect number of arguments.

Code 8

**AFP019 FUNCTION VALUE UNDEFINED**

*Explanation:* No value has been assigned to the function name. If the function contains no ENTRY statements, it will return unpredictable values. Otherwise, the message is a warning as long as a value is always assigned to one of the entry point areas.

Code 4

**AFP020 /common name/ COMMON BLOCK ERRORS**

*Explanation:* An EQUIVALENCE statement has attempted to associate variables which are not in the same block, or extends the beginning of the block. This message may also be issued because of a COMMON variable alignment inefficiency. The variables in error are listed after the message.

The EQUIVALENCE is ignored, and the program will not execute correctly if it relies on the equivalence of the variables.

Code 4

**AFP021 subprogram UNCLOSED DO LOOPS**

*Explanation:* DO loops exist in the program for which there are no closing statements, or there is incorrect nesting of DO loops. The missing or incorrect closing statement numbers are listed after the message.

Code 8

**AFP022 subprogram UNDEFINED LABELS**

*Explanation:* Labels have been referenced but not defined. The missing labels are listed after the message.

Code 8

**AFP023 subprogram EQUIVALENCE ALLOCATION ERRORS**

*Explanation:* EQUIVALENCE sets contain conflicting specifications. The variables in error are listed after the message. The EQUIVALENCE is ignored and the program will not execute correctly if it relies on the equivalence of the variables.

Code 4

**AFP024 subprogram EQUIVALENCE DEFINITION ERRORS**

*Explanation:* An EQUIVALENCE set references an array element with an invalid subscript. The array names are listed after the message. The equivalence is ignored and the program will not execute correctly if it relies on the equivalence of the variables.

Code 4

**AFP025 subprogram DUMMY DIMENSION ERRORS**

*Explanation:* Dummy array dimensions are neither arguments to the subprogram nor in COMMON. A list of such variables follows the message.

Code 8

**AFP026 BLOCK DATA ERRORS**

*Explanation:* Variables have been defined which are not in COMMON. A list of such variables follows the message.

Code 4

**AFP027 line number—SOURCE STMT TOO LONG**

*Explanation:* A source statement contains more than 1320 characters, excluding the statement number and insignificant blanks. The remainder of the line, including a possible continuation character, is ignored.

Code 8

**AFP028 subprogram NO SPACE AVAILABLE—COMPILATION TERMINATED**

*Explanation:* The maximum size workspace is too small to allow the compiler to compile the specified program unit. Compilation of the program unit is terminated.

The problem can be solved by re-ordering program units within the source program, largest and most complex first, otherwise the program must be divided into sections and linked together using the VSPC FORTRAN CHAIN facility.

Code 16

**AFP029 SYSTEM PROBLEM: COMPILATION TERMINATED**

*Explanation:* The compiler has detected an unresolvable error in its communication with VSPC.

Code 16

**AFP030 subprogram COMPILER PROBLEM: COMPILATION TERMINATED**

*Explanation:* An unresolvable compiler error has occurred.

Code 16

**AFP031 subprogram TABLE OVERFLOW: COMPILATION TERMINATED**

*Explanation:* One of the compiler's work tables has exceeded its maximum size. The problem is due to either the size or the complexity of the program unit, and can be avoided by recoding some of the complex statements, or by splitting the program unit into two or more units.

Code 16

**AFP032 NO MAIN PROGRAM**

*Explanation:* There is no main program unit in the source program.

Code 8

**AFP033 OBJECT PROGRAM TOO LARGE**

*Explanation:* The completed object program is larger than the maximum workspace.

Code 8

**AFP034 ATTENTION INTERRUPT: COMPILATION (LINKING) TERMINATED**

*Explanation:* An attention interrupt has terminated all processing of the source program. If you were in the process of linking programs, the message uses the word LINKING instead of COMPILATION.

Code 20

**AFP035 PREVIOUSLY TYPED**

*Explanation:* The variable has been given a type in a previous specification statement. The first specified type is used.

Code 8

**AFP036 WARNING: ILLEGAL LABEL**

*Explanation:* A non-executable statement has been given a statement label. An execution error will result if it is referenced.

Code 4

**AFP037 PREVIOUSLY DIMENSIONED**

*Explanation:* An array has been given dimensions in more than one specification statement. The first specification is used.

Code 4

**AFP038 WARNING: SIZE**

*Explanation:* A variable has been initialized with a literal constant which is longer than the variable. The constant is truncated.

Code 4

**AFP039 RETURN**

*Explanation:* A RETURN statement has been omitted. One is supplied.

Code 0

**AFP040 COMMON ERROR IN BLOCK DATA**

*Explanation:* Either the BLOCK DATA subprogram references blank COMMON, or it contains no references to any named COMMONs.

Code 8

**AFP041 NO MAIN PROGRAM, 'LINK' OPTION USED**

*Explanation:* The STORE command has defaulted to the LINK operand, because no main program was found in the object program.

Code 4

**AFP042 TOO MANY EXTERNAL SYMBOLS**

*Explanation:* There are more than 255 external symbols in the program. External symbols include subprogram names, ENTRY names, and COMMON names.

Code 16

**AFP043 subprogram DUPLICATE SUBPROGRAM**

*Explanation:* The source program contains two program units with the same name. The first is used.

Code 4

**AFP044 subprogram SUBPROGRAM NOT FOUND**

*Explanation:* A reference has been made to a subprogram which is neither in the source program nor the FORTRAN library. Execution of the statement will result in an execution error. The reference may be resolved at execution time if FLINK is specified.

Code 4

**AFP045 OBJECT PROGRAM EMPTY**

*Explanation:* If none of the programs you include exist, the combined object program is empty.

Code 8

**AFP046 subprogram FILE CANNOT BE LINKED**

*Explanation:* The named program cannot be linked because it isn't a VSPC FORTRAN object file or it was not stored with the 'LINK' option.

Code 4

**AFP047 variable or subprogram ID CONFLICT IN subprogram**

*Explanation:* The use of a variable or subprogram name conflicts with a previous use or specification.

Code 8

**AFP048 subprogram PASSWORD ERROR**

*Explanation:* Where a password is attached to the named subprogram, either none or the wrong one has been issued.

Code 4

**AFP049 subprogram FILE TOO LARGE**

*Explanation:* The file specified is larger than the maximum workspace and cannot be linked.

Code 4

**AFP050 subprogram I/O ERROR**

*Explanation:* An I/O error has been encountered while attempting to link the named subprogram from the library.

Code 16

**AFP051 subprogram FILE NOT FOUND**

*Explanation:* The named subprogram to be linked cannot be found in the library.

Code 4

## Initialization Messages

Initialization messages are sent whenever VSPC FORTRAN encounters a situation in which it is unable to start compilation or execution.

Such messages always inhibit any processing by VSPC FORTRAN, and consequently all have code 20 severity.

**AFP060 MAXIMUM WORKSPACE SIZE TO SMALL**

*Explanation:* The user's maximum workspace is smaller than the minimum in which FORTRAN can compile.

Code 20

**AFP061 INCOMPATIBLE WORKSPACE**

*Explanation:* The workspace provided at the start of execution is not one which was produced by FORTRAN.

Code 20

**AFP062 NO SOURCE PROGRAM**

*Explanation:* The editable workspace to be compiled contains no lines.

Code 20

## Library Messages

Whenever the VSPC FORTRAN library encounters an error situation during execution, it sends a message to the terminal of the form:

```
AFP170   LINE number IN subprogram FILE 'f'
         DOES NOT EXIST
```

The message text is preceded by three pieces of information:

- the message prefix, which is only included if the command "MESSAGE ID" has been specified
- the line number in the program where the error occurred, which is included unless your program is protected by NOREAD
- the subprogram name containing the referenced line, which is always included

In the following messages, the message prefix is included, but the line number and subprogram name are not.

The following symbols have these meanings:

| Symbol | Meaning |
|---|---|
| f | A VSPC file name, including the library number if necessary. |
| c | A character or string of characters. |
| n | An integer. |
| r | A real number, written in 1PE11.4 format. |

Messages from the mathematical library functions (AFP250 and higher) are followed by a line of traceback information indicating the function which detected the error and the function which called it, if any. For example:

```
AFP251   ARG = -1.0000E+00,LT 0.
           FROM SQRT
```

The codes associated with the library messages have the following meaning:

| Code | Meaning |
|---|---|
| 0 | The error is corrected interactively, and so has no effect on subsequent execution. |
| 4 | The error is corrected by VSPC FORTRAN according to fixed rules, and so incorrect results may be produced. |
| 8 | Execution is terminated. |

**AFP130 PROGRAM INTERRUPT**

*Explanation:* An unexpected program interrupt has occurred. The error is probably a user error caused by current or previous use of out-of-range array subscripts.

Code 8

**AFP131 SYSTEM PROBLEM**

*Explanation:* An error has occurred in communication between the FORTRAN processor and VSPC.

Code 8

**AFP139 PROCESSING TIME LIMIT REACHED**

*Explanation:* Since the last terminal input request, CPU time has been used which exceeds the limit specified in the user profile. The user is prompted for terminal input and if the response is a null line (that is, carrier return only), execution is resumed. Any data entered, or an attention, will terminate execution.

Codes 0/8

**AFP140 ATTENTION INTERRUPT**

*Explanation:* An attention interrupt has been received. The user is prompted for input, and if the response is a null line (that is, carrier return only), execution is resumed. Any data entered, or another attention, will terminate execution.

Codes 0/8

**AFP141 INTEGER OVERFLOW**

*Explanation:* An integer overflow program check has occurred. Execution continues with the incorrect result unmodified.

Code 4

**AFP142 INTEGER ZERO DIVIDE**

*Explanation:* An integer zero divide program check has occurred. Execution continues with the division suppressed.

Code 4

**AFP143 OVERFLOW**

*Explanation:* An exponent overflow program check has occurred. Execution continues with the largest number which can be expressed in the machine with the same sign as the overflow result.

Code 4

**AFP144 UNDERFLOW**

*Explanation:* An exponent underflow program check has occurred. Execution continues with a zero result.

Code 4

**AFP145 ZERO DIVIDE**

*Explanation:* A floating point divide program check has occurred. Execution continues with the largest positive number which can be expressed in the machine.

Code 4

**AFP150 RECURSIVE ENTRY**

*Explanation:* A CALL or function reference has been made to a subprogram which is currently active.

Code 8

**AFP153 NO SPACE AVAILABLE FOR RECORD BUFFER**

*Explanation:* A record is being written or read from a non-DATA file, and the space required to create the record would require that the workspace exceed its maximum size. Either decrease the size of records being written, decrease the size of the program, divide the program into two or more programs and link them together using "call opsys('chain')," or close any unused file using ENDFILE or "call opsys('free')."

Code 8

**AFP156 function FUNCTION NOT AVAILABLE**

*Explanation:* The specified function is not available. Depending on the function, execution may or may not continue.

Codes 0/8

**AFP157 EXTERNAL FILE ACCESS NOT SUPPORTED**

*Explanation:* An attempt has been made to access an external file.

Code 8

**AFP158 INVALID INPUT CHARACTER**

*Explanation:* A character that has been received from the terminal is not supported by VSPC (this may be the result of a transmission line error). The user is prompted again for the input.

Code 0

**AFP159 INPUT TOO LONG**

*Explanation:* The user has entered a line from the terminal which exceeds the maximum length acceptable to FORTRAN. The user is prompted again for the input.

Code 0

**AFP160 *'c'* (COL *nn*) INVALID—RETYPE FROM BEGINNING**

*Explanation:* In response to a formatted READ statement, the user has entered a line which contains invalid or wrongly positioned data. The user is prompted again for the complete line.

Code 0

**AFP161 *'c'* (COL *nn*) INVALID—RETYPE FROM ITEM *nn***

*Explanation:* In response to a list-directed READ statement, the user has entered an item which contains invalid characters, or which conflicts with the receiving variable. The user is prompted again for input, starting from the item in error.

Code 0

**AFP162 *'nnnn'* OUT OF RANGE—RETYPE FROM BEGINNING**

*Explanation:* In response to a formatted READ statement, the user has entered a number which is not within the range valid for the data type. The user is prompted again for the complete line.

Code 0

**AFP163 NUMBER OUT OF RANGE—RETYPE FROM ITEM *nn***

*Explanation:* In response to a list-directed READ statement, the user has entered a number which is not within the range valid for the data type. The user is prompted again for input, starting from the item in error.

Code 0

**AFP164 *'c'* (COL *nn*) INVALID—RETYPE FROM NAME *ccc***

*Explanation:* In response to a namelist READ statement, the user has entered a name or subscripted name which contains invalid characters or subscripts. The user is prompted again for input, starting from the name in error.

Code 0

**AFP165 NUMBER OUT OF RANGE—RETYPE FROM NAME *ccc***

*Explanation:* In response to a namelist READ statement, the user has entered a subscripted name which contains a subscript which is not within the range valid for integers. The user is prompted again for input starting from the name in error.

Code 0

**AFP166 ILLEGAL BACKSPACE TO TERMINAL**

*Explanation:* A backspace request has been made to a unit which is allocated to the terminal. The request is ignored.

Code 4

**AFP167 ILLEGAL REWIND TO TERMINAL**

*Explanation:* A rewind request has been made to a unit which is allocated to the terminal. The request is ignored.

Code 4

**AFP168 ILLEGAL ENDFILE TO TERMINAL**

*Explanation:* An endfile request has been made to a unit which is currently allocated to the terminal. The request is ignored.

Code 4

**AFP170 FILE '*f*' DOES NOT EXIST**

*Explanation:* The specified file does not exist or, if another user's private library is specified, the file does not exist as a shareable file.

Code 8

**AFP171 UNIT *n* NOT ALLOCATED**

*Explanation:* No ALLOCATE command or "call opsys('alloc')" has been issued to associate the specified unit with a file name.

Code 8

**AFP172 FILE '*f*' PASSWORD ERROR**

*Explanation:* The required password was not correctly specified.

Code 8

**AFP173 FILE '*f*' NOT OWNED**

*Explanation:* An attempt has been made to write to a file which is not owned by the user.

Code 8

**AFP174 FILE '*f*' PROTECTED**

*Explanation:* The specified file is protected, and has either NOREAD or NOWRITE status, as applicable.

Code 8

**AFP175 FILE '*f*' ALREADY IN USE**

*Explanation:* An attempt has been made to access a file which is in use, when either the required or the current access is for output. A single attempt is made to reopen the file after a delay of approximately 15 seconds. The error occurs if this second attempt fails.

Code 8

**AFP176 FILE '*f*' ACCESS MODE CONFLICTS WITH FILE TYPE**

*Explanation:* The way in which a file has been accessed conflicts with the attributes of the file: specifically, an undefined file has been accessed with other than list-directed I/O statements.

Code 8

**AFP178 UNIT *n* NOT DEFINED**

*Explanation:* An attempt to reference the specified unit through the direct access I/O statements has been made, but there has been no preceding DEFINE FILE statement.

Code 8

**AFP179 FILE '*f*' DEFINE FILE CONFLICTS WITH FILE ATTRIBUTES**

*Explanation:* The specifications in the DEFINE FILE statement conflict with the attributes of an existing file: either the record sizes differ, or the file contains fewer records than required by the DEFINE FILE.

Code 8

**AFP180 FILE '*f*' INVALID FILE TYPE**

*Explanation:* An attempt has been made to access an OBJECT file.

Code 8

**AFP182 FILE '*f*' RECORD LENGTH CONFLICTS WITH FILE ATTRIBUTES**

*Explantion:* A direct file is being accessed sequentially, and an attempt has been made to write a record whose length differs from that of the remaining records in the file.

Code 8

**AFP183 FILE '*f*' INPUT DATA ITEM TOO LONG**

*Explanation:* The file is being accessed through list-directed or namelist READ statements, and contains a data item which exceeds 256 characters.

Code 8

**AFP184 FILE '*f*' RECORD NUMBER *n* TOO LARGE**

*Explanation:* Either the number of the record being written exceeds the system limit (8388607) or the number of a record being read from a non-DATA file exceeds the limit for editable files (99999).

Code 8

**AFP185 FILE '*f*' NO SPACE AVAILABLE FOR BUFFER**

*Explanation:* The space required for a new disk buffer (4096 bytes) would exceed the maximum workspace allowed. Either decrease the size of the program, split it into two programs, or close any unused files (using ENDFILE or "call opsys('free')").

Code 8

**AFP186 FILE '*f*' INVALID DATA ON UNDEFINED FILE**

*Explanation:* The contents of an undefined file do not conform to FORTRAN's expectations.

Code 8

**AFP187 INVALID FILE NAME**

*Explanation:* The file name specified in a "call opsys('alloc')" does not conform to VSPC syntax.

Code 8

**AFP190 TOO MANY ACTIVE FILES**

*Explantion:* More than 15 VSPC files are concurrently open, or more than 17 FORTRAN units are concurrently active.

Code 8

**AFP191 ILLEGAL *c* CONVERSION**

*Explanation:* The indicated format item corresponds to an element in the I/O list which cannot be converted according to that format.

Code 8

**AFP192 INVALID LINE NUMBER**

*Explanation:* A record being written to a non-DATA file does not begin with a valid line number—either the first non-blank character in the record is not a digit, or there are five leading blanks in the record.

Code 8

**AFP193 ($n_1$, $n_2$) SEQUENCE ERROR**

*Explanation:* Records being written to a non-DATA file are not in ascending sequence—the previous and the erroneous line numbers are included in the message.

Code 8

**AFP196 FILE '*f*' IS FULL**

*Explanation:* The file has reached its limiting size, as specified on the FILE command.

Code 8

**AFP197 LIBRARY SPACE LIMIT REACHED**

*Explanation:* The library containing the file being written has reached its limiting size, as specified in the users profile.

Code 8

**AFP198 SYSTEM LIBRARY IS FULL**

*Explanation:* There is no more space available for VSPC to use. Inform the VSPC operator of the problem.

Code 8

**AFP204 FILE '*f*' ITEM LENGTH EXCEEDS RECORD LENGTH**

*Explanation:* The number of characters needed to represent a data item in list-directed format is greater than the available record length, or the width of the terminal as specified by the LINESIZE command.

Code 8

**AFP206 FILE '*f*' INTEGER OUT OF RANGE**

*Explanation:* The file contains a number which cannot be converted because the result would be outside of the range, $\pm 2^{31}-1$, valid for INTEGER*4 variables, or the range $\pm 2^{15}-1$ valid for INTEGER*2 variables. The error may be in the file or the format with which it is read.

Code 8

**AFP211 '*c*' INVALID FORMAT CHARACTER**

*Explanation:* An object-time format statement contains an invalid format character.

Code 8

**AFP212 FILE '*f*' DATA LENGTH EXCEEDS RECORD LENGTH**

*Explanation:* Either the record being written is longer than the available record length, in which case the current record is written and a new one started, or the format statement requires more data than exists in the current record, in which case the remainder of the I/O list is ignored.

Code 4

**AFP215 FILE '*f*' INVALID DECIMAL CHARACTER '*c*' (COL *nn*)**

*Explanation:* The file contains a numeric field which contains an invalid decimal character.

Code 8

**AFP216 INVALID ARGUMENT LIST**

*Explanation:* The argument list in a CALL statement or function reference disagrees in number with that required by the subprogram; or in the case of library subroutines, one of the arguments is invalid.

Code 8

**AFP217 FILE '*f*' END OF INPUT DATA REACHED**

*Explanation:* A sequential READ statement has been issued, but there are no more records in the file to satisfy the request.

If the END= parameter is specified in the READ statement, the error message is not generated, but a branch is taken to the specified label.

Code 8

**AFP218 FILE '*f*' I/O ERROR**

*Explanation:* A physical I/O error has occurred while trying to access the indicated file.

If the ERR= parameter or the I/O statement is specified, the error message is not generated, but a branch is taken to the specified label.

Code 8

**AFP220 UNIT *n* INVALID**

*Explanation:* A reference has been made to an I/O unit which is not in the range 1 to 99.

Code 8

**AFP221 FILE '*f*' NAME *cccc* TOO LONG**

*Explanation:* The file is being accessed through a namelist READ statement, and contains a name which is more than six characters long.

Code 8

**AFP222 FILE '*f*' NAME *cccccc* NOT IN DICTIONARY**

*Explanation:* The file is being accessed through a namelist READ statement, and contains a name which is not in the namelist dictionary.

Code 8

**AFP223 FILE '*f*' END OF RECORD REACHED BEFORE '='**

*Explanation:* The file is being accessed through a namelist READ statement, and contains a name which is not in the same record as its associated "equals" sign.

Code 8

**AFP224 FILE '*f*' INVALID INDEX FOR NAME *cccccc***

*Explanation:* The file is being accessed through a namelist READ statement, and a subscripted array name contains an invalid subscript.

Code 8

**AFP225 FILE '*f*' INVALID HEX CHARACTER '*c*' (COL *nn*)**

*Explanation:* The file is being read using a formatted READ statement, and a field corresponding to a Z format item contains an invalid hexadecimal character.

Code 8

**AFP226 FILE '*f*' REAL NUMBER OUT OF RANGE**

*Explanation:* The file contains a real number which is outside the allowable range, which is $\pm(16^{63}, 16^{-65})$ and 0.

Code 8

**AFP227 FILE '*f*' ERROR IN REPEAT COUNT**

*Explanation:* The file is being read through a list-directed READ statement, and contains a repeated specification which has an invalid repeat count.

Code 8

**AFP230 SOURCE PROGRAM ERROR**

*Explanation:* An error in the source program has caused a situation in which the object program cannot continue execution.

If a statement is flagged with a code 4 error by the compiler, execution of that statement may result in this error.

Possible causes are: a CALL or function reference has been made to a subprogram which does not exist; an assigned GOTO statement has been executed in which the variable has not had a valid label ASSIGNed to it; a branch has been made to a label which has been flagged as ILLEGAL by the compiler—for instance, a label on a non-executable statement.

Code 8

**AFP231 FILE '*f*' IS SEQUENTIAL**

*Explanation:* A direct access I/O statement has been issued, and the file either does not have the DIRECT attribute, or the unit is currently being accessed sequentially.

Code 8

**AFP232 FILE '*f*' RECORD *n* OUT OF RANGE**

*Explanation:* A direct access I/O statement has referenced a record which is not in the file as specified in the DEFINE FILE statement.

Code 8

**AFP233 UNIT *n* RECORD LENGTH TOO LARGE**

*Explanation:* A DEFINE FILE statement specifies a record length which exceeds the VSPC limit (4058 bytes).

Code 8

**AFP235 FILE '*f*' IS DIRECT**

*Explanation:* A sequential I/O statement has been issued for a file which is currently being accessed directly.

Code 8

**AFP238 FILE '*f*' INVALID DELIMITER FOR LITERAL OR COMPLEX INPUT**

*Explanation:* A file is being read using a list-directed READ statement, and contains a quoted literal or complex constant which is not delimited by a closing quotation mark or parenthesis, respectively, followed by a valid delimiter.

Code 8

**AFP241 INTEGER BASE= 0, INTEGER EXPONENT= *n*, LE 0**

*Explanation:* An exponentiation of the form:

```
0 ** N
```

has occurred, and N is not positive. The result is mathematically undefined. A result of 0 is used.

Code 4

**AFP242 REAL BASE= 0., INTEGER EXPONENT= *n*, LE 0**

*Explanation:* An exponentiation of the form:

0. ** N

has occurred, and N is not positive. The result is mathematically undefined. A result of 0 is used if the exponent is zero, or the largest positive real number if it is negative.

Code 4

**AFP243 REAL BASE= 0., INTEGER EXPONENT= *n*, LE 0**

*Explanation:* An exponentiation of the form:

0D0 ** N

has occurred, and N is not positive. The result is mathematically undefined. A result of 0D0 is used if the exponent is 0, or the largest positive number if it is negative.

Code 4

**AFP244 REAL BASE= 0., REAL EXPONENT= *r*, LE 0.**

*Explanation:* An exponentiation of the form:

0. ** R

has occurred, where R is not positive. The result is mathematically undefined. A result of 0. is used.

Code 4

**AFP245 REAL BASE= 0., REAL EXPONENT= *r*, LE 0.**

*Explanation:* An exponentiation of the form:

0D0 ** D

has occurred, where D is not positive. The result is mathematically undefined. A result of 0D0 is used.

Code 4

**AFP246 COMPLEX BASE= (0., 0.), INTEGER EXPONENT= *n*, LE 0**

*Explanation:* An exponentiation of the form:

(0., 0.) ** N

has occurred, and N is not positive. The result is mathematically undefined. A result of (1., 0.) is used if the exponent is zero, and the largest positive real number if it is negative.

Code 4

**AFP247 COMPLEX BASE= (0., 0.), INTEGER EXPONENT= *n*, LE 0**

*Explanation:* An exponentiation of the form:

(0D0, 0D0) ** N

has occurred, and N is not positive. The result is mathematically undefined. A result of (1D0, 0D0) is used if the exponent is zero, and the largest positive number if it is negative.

Code 4

**AFP251 ARG= *r*, LT 0.**

*Explanation:* The argument to the SQRT function is negative. The result is undefined. The function returns the square root of the positive value of the argument.

Code 4

**AFP252 ARG= *r*, GE 174.673**

*Explanation:* The argument to the EXP function is greater than or equal to 174.673, and so would cause overflow. The function returns the largest real number.

Code 4

**AFP253 ARG= *r*, LE 0.**

*Explanation:* The argument to the ALOG or ALOG10 function is less than or equal to zero. The result is undefined. The function returns the largest negative real number.

Code 4

**AFP254 /ARG/= *r*, GE PI*2**18**

*Explanation:* The argument to the SIN or COS function is greater than PI*2**18, and would prevent the function from producing an accurate result. The function returns the value SQRT(2)/2.

Code 4

**AFP255 ARGS BOTH 0.**

*Explanation:* Both arguments to the ATAN2 function are zero. The result is undefined. The function returns a value of 0.

Code 4

**AFP256 /ARG/= *r*, GE 175.366**

*Explanation:* The argument to the SINH or COSH function is numerically greater than or equal to 175.366, and would cause overflow. The function returns the largest real number, with the same sign as the argument to SINH.

Code 4

**AFP257 /ARG/= *r*, GT 1.**

*Explanation:* The argument to the ARSIN or ARCOS function is numerically greater than 1., and so the result is undefined. The argument is replaced by 1. and the sign of the erroneous argument retained.

Code 4

**AFP258 /ARG/= *r*, GE PI*2**18**

*Explanation:* The argument to the TAN or COTAN function is greater than PI*2**18, and would prevent the function from producing an accurate result. The result is set to 1.

Code 4

**AFP259 /ARG/= *r*, APPROACHES SINGULARITY**

*Explanation:* The argument to the TAN or COTAN function is close to a singularity at which the function tends discontinuously to infinity. The singularities of TAN are at odd multiples of PI/2, and of COTAN are at multiples of PI. The function returns the maximum real number.

Code 4

**AFP261 ARG= *r*, LT 0.**

*Explanation:* The argument to the DSQRT function is negative, and so the result is undefined. The function returns the square root of the positive value of the argument.

Code 4

**AFP262 ARG= *r*, GE 174.673**

*Explanation:* The argument to the DEXP function is greater than or equal to 174.673, and so would cause overflow. The function returns the largest real number.

Code 4

**AFP263 ARG= *r*, LE 0.**

*Explanation:* The argument to the DLOG or DLOGIO function is less than or equal to zero, and so the result is undefined. The function returns the largest negative real number.

Code 4

**AFP264 /ARG/= *r*, GE PI*2**50**

*Explanation:* The argument to the DSIN or DCOS function is greater than PI*2**50, and would prevent the function from producing an accurate result. The function returns the value SQRT(2)/2.

Code 4

**AFP265 ARGS BOTH 0.**

*Explanation:* Both arguments to the DATAN2 function are zero, and so the result is undefined. The function returns a value of 0.

Code 4

**AFP266 /ARG/= *r*, GE 175.366**

*Explanation:* The argument to the DSINH or DCOSH function is numerically greater than or equal to 175.366, and would cause overflow. The function returns the largest real number, negative in the case of a negative argument to DSINH.

Code 4

**AFP267 /ARG/= *r*, GT 1.**

*Explanation:* The argument to the DARSIN or DARCOS function is numerically greater than 1., and so the result is undefined. The function returns 1, with the sign of the argument.

Code 4

**AFP268 /ARG/= *r*, GE PI*2**50**

*Explanation:* The argument to the DTAN or DCOTAN function is greater than PI*2**18, and would prevent the function from producing an accurate result. The result is set to 1.

Code 4

**AFP269 /ARG/= *r*, APPROACHES SINGULARITY**

*Explanation:* The argument to the DTAN or DCOTAN function is close to a singualrity, at which the function tends discontinuously to infinity. The singularities of DTAN are at the odd multiples of PI/2, and of DCOTAN are at multiples of PI. The function returns the maximum real number.

Code 4

**AFP271 REAL ARG= *r*, GE 174.673**

*Explanation:* The real part of the argument to the CEXP is greater than or equal to 174.673, and so would cause overflow. The function returns the largest real number, multiplied by CEXP (the imaginary part of the argument).

Code 4

**AFP272 /IMAG ARG/= *r*, GE PI*2**18**

*Explanation:* The imaginary part of the argument to the CEXP function is numerically greater than or equal to PI*2**18, and so would prevent the function from producing accurate results. The function returns CEXP (real part of the argument).

Code 4

**AFP273 ARG= (0.,0.)**

*Explanation:* The argument to the CLOG function is (0.,0.), and so the result is undefined. The function returns the largest negative real number.

Code 4

**AFP274 /REAL ARG/=** ***r*****, GE PI*2**18**

*Explanation:* The real part of the argument to the CSIN or CCOS function is greater than or equal to PI*2**18, and so would prevent the function from producing an accurate result. The function returns the value CSIN/CCOS (real part of the argument).

Code 4

**AFP275 /IMAG ARG/=** ***r*****, GE 174.673**

*Explanation:* The imaginary part of the argument to the CSIN or CCOS function is numerically greater than or equal to 174.673, and so would cause an overflow. The function takes 174.673 as the imaginary part.

Code 4

**AFP281 REAL ARG=** ***r*****, GE 174.673**

*Explanation:* The real part of the argument to the CDEXP function is greater than or equal to 174.673, and so would cause overflow. The function returns the largest real number, multiplied by CDEXP (the imaginary part of the argument).

Code 4

**AFP282 /IMAG ARG/=** ***r*****, GE PI*2**50**

*Explanation:* The imaginary part of the argument to CDEXP is numerically greater than or equal to PI*2**50, and so would prevent the function from producing an accurate result. The function returns the value CDEXP (real part of the argument).

Code 4

**AFP283 ARG= (0.,0.)**

*Explanation:* The argument to the CDLOG function is (0.,0.), and so the result is undefined. The function returns the largest negative real number.

Code 4

**AFP284 /REAL ARG/=** ***r*****, GE PI*2**50**

*Explanation:* The real part of the argument to the CDSIN or CDCOS function is greater than or equal to PI*2**50, and so would prevent the function from producing an accurate result. The function returns the value CDSIN/CDCOS (real part of the argument).

Code 4

**AFP285 /IMAG ARG/=** ***r*****, GE 174.673**

*Explanation:* The imaginary part of the argument to the CDSIN or CDCOS function is numerically greater than or equal to 174.673, and so would cause an overflow. The function takes 174.673 as the imaginary part.

Code 4

**AFP290 ARG=** ***r*****, LT 2**-252 or GE 57.5744**

*Explanation:* The argument to the GAMMA function is not in the indicated range, and so would cause overflow. The function returns the largest real number.

Code 4

**AFP291 ARG=** ***r*****, LT 0. OR GE 4.2937E+73**

*Explanation:* The argument to the ALGAMA function is not in the range indicated, and so would cause overflow. The function returns the largest real number.

Code 4

**AFP300 ARG=** ***r*****, LT 2**-252 or GE 57.5744**

*Explanation:* The argument to the DGAMMA function is not in the indicated range, and so would cause overflow. The function returns the largest real number.

Code 4

**AFP301 ARG= *r*, LT 0. or GE 4.2937E+73**

*Explanation:* The argument to the DLGAMMA function is not in the range indicated, and so would cause overflow. The function returns the largest real number.

Code 4

**Note: Explanations of VSPC terminal messages are listed in a computer printout that can be obtained by the procedure described in an appendix of *VSPC General User's Guide and Command Language*.**

# GLOSSARY

This glossary defines data processing terms used in this manual—including terms used only in VSPC as well as standard data processing terms. Terms not used in this manual are omitted.

American National Standard definitions are identified with asterisks (*).

IBM is grateful to the American National Standards Institute (ANSI) for permission to reprint its definitions from the *American National Standard Vocabulary for Information Processing* (Copyright © 1970 by American National Standards Institute, Incorporated), which was prepared by Subcommittee X3K5 on Terminology and Glossary of the American National Standards Committee X3.

**access control:** The alteration of normal VSPC file access conventions by use of the VSPC PROTECT and SHARE commands, and by the specification of passwords.

**accounting unit:** An installation-defined measure of VSPC system usage that is used for user accounting.

**administrative command:** A special command used by the VSPC administrator to process user profiles and perform other supervisory tasks.

**AID:** *See* VSPC AID.

**alphameric character:** The letters A through Z, digits 0 through 9, punctuation marks, and #, $, and @.

**alphanumeric:** *See* alphameric character.

**APL:** A programming language. A problem-solving programming language designed for use at terminals. It has capabilities for handling arrays and for performing mathematical functions. VS APL is an IBM program product that can be used with VSPC.

**attention key:** A key on some terminals that interrupts execution by the Central Processing Unit.

**attribute:** A descriptor of the kind of information contained in a user's workspace of files. A user profile is assigned a default attribute that applies to the user's workspace when he logs on to VSPC.

**background:** The portion of VSPC that provides batch processing capabilities.

**BASIC:** An algebra-like programming language used for problem solving by engineers, scientists, and others who may not be professional programmers. VS BASIC is an IBM program product that can be used with VSPC.

**batch job:** A serially processed job, as opposed to an interactive job.

**batch processing:** The processing of computer jobs serially, as opposed to interactive processing.

**blank:** A nonprinting graphic character used to separate words or other meaningful information.

**buffer:** An area of storage that is temporarily reserved for use in performing an input/output operation, into which data is read or from which data is written. Synonymous with I/O area.

**bulk input area:** In VIEW mode, the area of the screen in which several workspace lines can be entered before interacting with the VSPC system to have the lines placed in the user's workspace.

**byte:** (1)*A sequence of adjacent binary digits operated upon as a unit and usually shorter than a computer word. (2) The representation of a character.

**carrier return:** The operation that prepares for the next character to be printed or displayed at the first position on the next line.

**central processing unit (CPU):** The unit of a computer that includes the circuits controlling the interpretation and execution of instructions.

**chaining:** A method of defining VSPC command lists in such a way that execution of one command list automatically begins execution of another.

**character:** *A letter, digit, or other symbol that is used as part of the organization, control, or representation of data.

**character set:** The set of unique characters associated with a coding system.

**character string:** *A string consisting solely of characters.

**command:** An imperative statement used interactively, at a terminal, to tell VSPC to perform a specific action.

**command list:** A sequence of VSPC commands that can be saved for future execution as a group.

**command mode:** A VSPC processing mode that allows the user to enter general VSPC commands in the input area.

**communicating (online):** Pertaining to a terminal connected by a wire or a telephone circuit with the central processing unit.

**compilation:** The process by which a language processor source program is translated into a machine language object program.

**compile:** To prepare a machine language program from a computer program written in another programming language.

**compile time:** The time during which a language processor source program is translated into a machine language object program.

**compiler:** *A program that compiles.

**computer:** *See* computing system.

**computing system:** A central processing unit with main storage, input/output channels, control units, storage devices, and input/output devices connected to it.

**CONTINUE file:** A special VSPC file, available to a VSPC user, in which the contents of the workspace are automatically placed when a forced ending occurs or when the user requests it during logoff.

**conversational:** Describing a program or a system that carries on a dialog with a remote terminal user, alternately accepting input and then responding to the input quickly enough for the user to maintain his train of thought.

**current line pointer:** A line number within a workspace that is maintained by VSPC, that is set by commands that change the workspace, and that may be symbolically referenced in VSPC commands.

**cursor:** A movable spot of light on the cathode ray tube of a console or display unit that indicates where the next character will be entered.

**cursor control keys:** Special keys on the 3270 used to move the cursor vertically and horizontally.

**cursor wrap:** Movement of the cursor off one edge of the screen, "around the back," and appearing on the opposite edge.

**data:** *A representation of facts, concepts, or instructions in a formalized manner suitable for communication, interpretation, or processing by humans or automatic means.

**default:** The choice among exclusive alternatives made by VSPC when no explicit choice is specified by the user.

**dial line:** *See* switched line.

**dial-up terminal:** A terminal on a switched network.

**direct file:** In VSPC a file containing records of fixed length with line numbers beginning at one and incrementing by one. In such a file, specific records can be retrieved using the line number as a search key, but not by VS BASIC. A direct file must have the DATA attribute.

**directory:** A set of records in the VSPC library, each one of which identifies and describes a file in a user library. Each user library has its own directory.

**display station:** The 3270 group of screens, printers, keyboards, and auxiliary equipment.

**editable data:** One or more lines, each consisting of a line number followed by a character string; each line can be acted upon by the VSPC data editing commands. Editable data can consist of processor source programs, command lists, or simple data without any processing attribute.

**EOB:** End of block; a code that marks the end of a block of data.

**EOM:** End of message; the specific character or sequence of characters that indicates the termination of a message or record.

**external VSAM file:** A VSAM file stored outside of the VSPC library that is available to VSPC users through language processor input/output statements.

**file:** A unit of information stored in a library (a library member). it is the unit of information stored in a user library.

**file owner:** The VSPC user who created a file. In a private library, the file owner and the library manager are the same user; in project and public libraries, they may be either the same or different users.

**filename:** A name that identifies a file. A filename consists of a number (libnum), a name, and possibly a password.

**folding:** A technique used with the universal character set feature to allow each of the 256 possible character codes to print some character on a chain or train with few graphics. For example, it allows the printing of uppercase graphics when lowercase graphics are not available in the character array.

**forced ending:** A terminal session ending that is not under the control of the terminal user. A forced ending can occur when there is a detectable communication breakdown between the terminal and the computer system, when the system operator halts VSPC, or when the VSPC operator halts the user's session.

**foreground:** The portion of VSPC that provides interactive processing capabilities.

**foreground processor:** A computer program, such as a language processor, that runs interactively under the control of VSPC.

**FORTRAN:** FORmula TRANslating system. A programming language used primarily to write computer programs in arithmetic formulas. VSPC FORTRAN is an IBM program product that can be used with VSPC.

**general command area:** In VIEW mode, an area of the screen within which VSPC general commands may be entered with the ENTER key.

**hardcopy:** A printed representation of data.

**I/O:** *See* input/output.

**incr:** *See* increment.

**increment:** An increase in quantity or value.

**input/output:** A general term for the equipment used to communicate with a computer; also the data involved in such communication; commonly called I/O.

**installation:** A general term for a particular computing system, in the context of the overall function it serves and the individuals who manage it, operate it, apply it to problems, maintain it, and use the results it produces.

**integer:** An unsigned natural number.

**interactive processing:** Computer processing in which each entry from a terminal elicits a response from the computer.

**interpreter:** A language processor that translates and executes each source program statement before it translates and executes the next one. For example, VS APL.

**interpreter workspace:** A workspace that contains information produced by working under the control of an interpreter. Also a file that contains such information.

**invalid output:** Characters not recognized by VSPC.

**JCL:** job control language

**job card:** A job control language statement that begins and identifies a job within the job stream.

**job control language (JCL):** A programming language used to code job control statements. Abbreviated JCL.

**job control statement:** *A statement in a job that is used in identifying the job or describing its requirements to the operating system.

**job entry:** The submission of a batch computer job from a terminal.

**job stream:** On input, the sequence of job control statements and data submitted to the batch operating system; on output, the diagnostic messages and other output data produced when a batch job is executed.

**jobname:** An eight-character name that identifies a VSPC user's batch job. The first six characters are the user's preassigned VSPC job entry code; the last two characters are numeric and assigned by VSPC.

**keystroke:** A pressing of a terminal key.

**keyword:** A VSPC command name or operand that identifies a specific action to be taken.

**leased line:** A direct connection between a terminal and a computer.

**library:** A collection of files. (*See* private library, project library, public library.)

**library manager:** The VSPC user to whom a user library was assigned at enrollment.

**library member:** A unit of information stored in a library (a file).

**library number (libnum):** The user number that identifies a user's private, project, or public library.

**light pen:** A light-sensitive pen that can detect characters displayed on a 3270 cathode ray tube. By pressing the tip of the pen on control line fields, the user can request specific VSPC actions.

**line number:** A one- through five-digit number that begins a line of editable data.

**listing:** A display or printout of data.

**local command:** In VIEW mode, a command that is entered over a line number in the workspace display area and that acts upon a limited number of lines.

**logoff:** A user's procedure for terminating a session with VSPC.

**logon:** A user's procedure for beginning a session with VSPC.

**message:** A combination of characters and symbols transmitted from one point to another on a telecommunication network.

**message header:** A six-character identifier for a VSPC message in the form ASUnnn (VS BASIC, ICDnnn), where nnn is a three-digit number. VSPC message headers are not normally printed; the user can, however, request that they be printed with each message displayed.

**MTA (Multiple Terminal Access):** A feature of some terminals allowing more than one terminal on one communication line. The NCP recognizes terminals by analysis of characters transmitted from the terminal.

**NCP:** *See* network control program.

**network control program (NCP):** A program, transmitted to and stored in a communications controller, that controls the operation of the communications controller.

**nondial line:** *See* leased line.

**nonswitched line:** *See* leased line.

**null line:** A VSPC line that contains no characters. A null line is entered by pressing carrier return before any characters have been entered or after all characters have been erased.

**object program:** An executable computer program resulting from the compilation of a source program.

**offline:** Pertaining to a terminal not communicating with the CPU.

**online:** Pertaining to a terminal communicating with the CPU.

**operand:** Information entered after a command name that modifies the action of the command and/or defines the data on which the command operates. (*See* keyword, variable operand.)

**operating system:** *Software which controls the execution of computer programs and which may provide scheduling, debugging, input/output control, accounting, compilation, storage assignment, data management, and related services.

**operator:** A member of a data processing installation who is responsible for directing the overall operation of a computing system.

**option:** An operand you need not state and is so designated.

**parameter:** A variable that is given a constant value for a specific purpose or process.

**password:** A string of one through eight characters that limits access to VSPC (logon password) or to a VSPC file (filename password).

**PF key:** See program function key.

**predefined path:** A direct connection from a terminal to the CPU, making special identification from the 2741 and 1050 at logon unnecessary.

**private library:** A VSPC library containing files that are normally available only to the user with that library number (user number); if this user so specifies, however, such files may be made available on a read-only basis to other VSPC users.

**processing unit:** * A unit of a computer that includes circuits controlling the interpretation and execution of instructions. Synonymous with central processor, main frame.

**processor:** A computer program that runs either interactively under the control of VSPC (*foreground processor*), or as a separate batch job that communicates with VSPC (*auxiliary processor*).

**profile:** *See* user profile.

**program function key:** A key on the keyboard of a display device that passes a signal to a program to call for a particular program operation.

**program product:** A licensed IBM program that performs a function for the user and usually interacts with and relies upon system control programming or some other IBM-provided control program. For example, VS APL, VS BASIC, and VSPC FORTRAN are IBM program products that rely on VSPC.

**programming language:** A language used to write computer programs, such as VS APL, VS BASIC, or VSPC FORTRAN.

**project library:** A VSPC library containing files that are normally available only to a subset of VSPC users whose profiles specify that they may have access to the library on a read-only basis. Files in this library may be modified only by their owners. Files may be removed only by the owner or the library manager. Ownership of files may be transferred among project users.

**prompt:** A request from VSPC for certain information, such as a password.

**public library:** A VSPC library containing files that are normally available to all VSPC users on a read-only basis. Files in this library may be modified only by their owners. Files may be removed only by the owner or the library manager.

**remote terminal:** *See* terminal.

S/S mode: see start-stop mode

SDLC mode: see synchronous data link control mode

**selector light pen:** *See* light pen.

**sequential file:** A file whose records are organized on the basis of their successive physical positions, such as on magnetic tape. *Contrast with* direct file.

**session:** The activity that occurs at a terminal between logon and logoff.

**shared storage:** Virtual storage used by VSPC for data exchange among users and auxiliary processors.

**source program:** A sequence of computer program statements written in a programming language and requiring compilation or interpretation to be executable by the computer.

**special character:** *A graphic character that is neither a letter, nor a digit, nor a space character.

**start-stop mode (S/S mode):** Asynchronous transmission between a terminal and the computer in which each character (or group of code) is preceded by a start signal and followed by a stop signal. Abbreviated S/S mode.

**string:** *A linear sequence of entities such as characters or physical elements.

**stylus:** *See* light pen.

**switched line:** A communication line in which the connection between the computer and a remote station is established by dialing. Synonymous with dial line.

**synchronous data link control mode (SDLC mode):** Synchronous transmission between a terminal and the computer in which the sending and receiving instruments are operating continuously. Abbreviated SDLC mode.

**syntax:** *The structure of expressions in a language.

**terminal:** A device, usually equipped with a keyboard and some kind of display, capable of sending and receiving information over a communication channel.

**terminal user:** *See* user.

**time sharing:** A method of using a computing system that allows a number of users to execute programs concurrently and to interact with the programs during execution.

**typamatic:** Pertaining to typewriter keys having the ability to repeat their character or function when held down.

**type element:** A hardware feature on some terminals (2741, 1050) that is shaped like a ball and moves horizontally across the terminal on a carrier, printing characters on the paper as the user presses the keys.

**undefined file:** A VSPC processor output file that has records in a format unrecognizable by VSPC; such files cannot be edited using VSPC.

**user:** Under VSPC, anyone eligible to log on.

**user identification (usernum):** *See* user number.

**user number:** A one- to seven-digit symbol identifying each system user.

**user profile:** The definition of the characteristics associated with a specific end user, including: the user number, the optional password, whether or not this user is a project or public library manager, maximum storage allocation for this user's library and workspace, the default workspace contents, whether or not this user is able to use job entry commands, and whether or not this user is privileged to use administrative commands.

**usernum:** *See* user number.

**variable operand:** An operand that modifies the action of a VSPC command through a user-chosen value.

**VIEW mode:** A VSPC display terminal processing mode in which the user can modify a displayed workspace using the local facilities of the terminal, and using local and general commands, and can then copy changes back into the workspace.

**visual fidelity:** A technique that uses the position of the print head as a guide to where on the input line the next action will take place.

**VS APL:** An APL language running under VSPC.

**VS BASIC:** *See* BASIC.

**VS Personal Computing:** A program product that allows interactive or batch processing from a terminal (abbreviated VSPC).

**VSAM:** Virtual storage access method. The set of programs the operating system uses to transfer data between VSPC and direct-access storage.

**VSPC:** VS (Virtual System) Personal Computing

**VSPC administrator:** A VSPC user authorized to use the administrative commands for enrolling new VSPC users and processing their user profiles.

**VSPC AID:** A program contained in VSPC that, on request, prompts a user to specify a VSPC command correctly or provides an explanation of a VSPC message.

**VSPC FORTRAN:** A FORTRAN language running under VSPC.

**VTAM:** Virtual telecommunications access method. The set of programs the operating system uses to transfer data between terminals and VSPC.

**workspace:** The storage in which a VSPC user does his terminal work (such as entering data or source program statements and compiling and executing programs).

**workspace attribute:** The content classification of a portion of storage in VSPC.

**workspace display area:** An area of a display screen in VIEW mode in which part of the user's workspace is displayed and can be changed by use of the local facilities of the terminal.

**wrap:** *See* cursor wrap.

# INDEX

VSPC FORTRAN
Terminal User's Guide
SH20-9062-2

**Reader's Comment Form**

Note: Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

**List TNLs here:**

If you have applied any technical newsletters (TNLs) to this book, please list them here:

Last TNL ____________________

Previous TNL ____________________

Previous TNL ____________________

**Fold on two lines, tape, and mail.** No postage necessary if mailed in the U.S.A. (Elsewhere, any IBM representative will be happy to forward your comments.) Thank you for your cooperation.

SH20-9062-2

Reader's Comment Form

Fold and tape Please do not staple Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
P.O. Box 50020
Programming Publishing
San Jose, California 95150

Fold and tape Please do not staple Fold and tape

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

VSPC FORTRAN Terminal User's Guide Printed in U.S.A. SH20-9062-2

IBM Technical Newsletter

|  |  |
|---|---|
| This Newsletter No. | SN20-9303 |
| Date | July 10, 1979 |
| Base Publication No. | SH20-9062-2 |
| Prerequisite Newsletters | None |

# VSPC FORTRAN
# Terminal User's Guide

This technical newsletter, a part of Release 1 of VSPC FORTRAN, program number 5748-FO2, provides replacement pages for the subject publication. These replacement pages remain in effect for subsequent releases unless specifically altered. Pages to be inserted and removed are:

|  |  |
|---|---|
| Title page-14 | 83, 84 |
| 29 | 89-94 |
| 35-50 (40.1 added) | 107-110.1 (110.1 added) |
| 55-56.1 (56.1 added) | 115-118.5 (118.1-118.5 added) |
| 61-62.1 (62.1 added) | 137, 138 |
| 71,72 | 175-180 |

Each technical change is marked by a vertical bar to the left of the change.

**Summary of Amendments**

Technical changes are summarized under "Summary of Amendments" following the list of figures.

**Note:** Please file this cover letter at the back of the publication to provide a record of changes.

**IBM Corporation, P.O. Box 50020, Programming Publishing, San Jose, California 95150**

Printed in U.S.A.

SH20-9062-2

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

VSPC FORTRAN Terminal User's Guide Printed in U.S.A. SH20-9062-2