SH20-9067-2

Program Product

VS APL for CMS: Terminal User's Guide

IBM

SH20-9067-2

Program Product

VS APL for CMS: Terminal User's Guide

Program Number 5748-AP1

IBM

Third Edition (December 1978)

This edition replaces the previous edition (numbered SH20-9067-1) and its technical newsletter (numbered SN20-9207) and makes them obsolete.

This edition applies to Release 3 of VS APL, program number 5748-AP1, and to any subsequent releases unless otherwise indicated in new editions or technical newsletters.

The changes for this edition are summarized under "Summary of Amendments" following the list of figures. Specific changes made 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 continually 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.

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 book describes the use of VS APL when it is under control of the Conversational Monitor System (CMS) component of the Virtual Machine Facility/370 (VM/370). It contains detailed information on the terminals that can be used with the product and the procedures that must be followed in contacting CMS and VS APL. This book also describes the commands presented by VM/370 and VS APL relevant to the needs of the VS APL user, and the auxiliary processors and workspaces distributed with the product. It is assumed that you are already familiar with the APL language, but that you have no prior experience with CMS.

The information in this publication is organized into seven sections:

- "Introduction," which introduces VM/370 and CMS and illustrates their relationship to the VS APL Program Product.
- "Terminal Procedures," which lists and describes the terminals that you may use in your work, details what conventions you must follow in making entries and correcting entries at these terminals, and discusses how the form of output at your terminal may be controlled.
- "The Work Session," which describes how to start and end a work session with CMS and VS APL.
- "Workspaces and Libraries," which covers the structure and attributes of the VS APL workspace and details the libraries supported by CMS.
- "System Commands," which illustrates the form and use of VS APL system commands. System commands are used to monitor and control the contents of APL workspaces and libraries, as well as to communicate messages to other users.
- "Auxiliary Processing," which describes the use of the auxiliary processors distributed with VS APL.
- "Sample Terminal Session," which shows a hypothetical terminal session with VS APL under CMS. This session shows many of the features presented in the first six sections of the publication. The session is designed so that you can sit at your terminal and work along as each entry and response is described.

Seven appendixes are also provided:

- "Appendix A," which lists and summarizes the predefined public library workspaces distributed with VS APL.
- "Appendix B," which introduces the VS APL Conversion Program and describes the VS APL conversion report.
- "Appendix C," which describes some special language considerations for VS APL.
- "Appendix D," which summarizes the operating procedures for all the terminals supported by CMS and discusses a number of APL considerations for these terminals.
- "Appendix E," which lists and explains the error messages that you may receive in interacting with CMS and VS APL, and suggests possible corrective action that you might take.

- "Appendix F: VS APL Batch Processing," which describes how you can submit your work to VS APL as a batch job using the CMS Batch Facility.
- "Appendix G: Character Translation," which shows the character translations between VS APL characters and EBCDIC characters done using the 192 conversion option.

Required Publications

This publication assumes that you are familiar with the APL language as described in the publication *APL Language,* GC26-3847.

Related Publications

In addition to *APL Language,* you might find the following publications helpful:

- *IBM Virtual Machine Facility/370: Terminal User's Guide,* GC20-1810, which describes the operating procedures for the terminals supported by VS APL under CMS.
- *IBM Virtual Machine Facility/370: System Messages,* GC20-1808, which describes the error messages reported by the control segment of VM/370 and CMS.
- *VS APL for CMS: Writing Auxiliary Processors,* SH20-9068, which describes how to write auxiliary processors to operate with VS APL. This book is directed primarily to system programmers.
- *VS APL Installation Reference Material,* SH20-9065, describes how to install VS APL and use the APL Conversion Utility Program; it should be consulted for a description of conversion procedures. This book is directed primarily to system programmers.
- *IBM Virtual Machine Facility/370: System Programmer's Guide,* GC20-1807. This publication is intended for system programmers. It describes how to debug VM/370 and how to modify, extend, or implement CP or CMS functions. This publication should be referenced as necessary to interpret and respond to VS APL executor messages.
- *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide,* GC26-3838, which contains information about using VSAM. It contains specific information on VSAM return codes generated in attempting to access VSAM files through the CMS VSAM Auxiliary Processor.
- *IBM Virtual Machine Facility/370: CMS User's Guide,* GC20-1819, which describes how to use the facilities of CMS to create, compile, debug and execute programs such as VS APL under CMS. This publication is a tutorial on the use of CMS commands, and EXEC, EDIT, and DEBUG subcommands.
- *IBM Virtual Machine Facility/370: CMS Command and Macro Reference,* GC20-1818, which contains reference information on CMS commands, subcommands, and macros.
- *IBM Virtual Machine Facility/370: CP Command Reference for General Users,* GC20-1820, which contains all reference information on CP privilege class G and any commands.

- *IBM Virtual Machine Facility/370: Operating Systems in a Virtual Machine,* GC20-1821. The first section of this publication describes how to use CP commands.
- *An Introduction to the IBM 3270 Data Analysis-APL Feature,* GA27-2788, which contains information pertinent to the use and understanding of the IBM 3270 Data Analysis-APL feature.
- Either *OS/VS1 Data Management Macro Instructions,* GC26-3872, or *OS/VS2 MVS Data Management Macro Instructions,* GC26-3873, which gives status information following an input/output operation.

Syntax Notation

Throughout this book an attempt is made to define as clearly as possible the statements and commands that you may enter at your terminal. Where an entry may take several forms, all the forms are indicated. When an entry is shown, a distinction is made between information that you must enter exactly as shown and information that you may supply, as appropriate. Furthermore, a distinction is made between APL entries and VM/370 entries. If an item is representative and must be replaced with an appropriate value, the item will appear in lowercase italics. If an APL entry is to be made exactly as shown, the entry will appear in APL font (an uppercase italicized font). For instance, in the following VS APL system command:

)*CLEAR size*

)*CLEAR* is entered at the terminal as shown, while *size* is only representative and is replaced by a suitable size when the actual entry is made. If a VM/370 entry is to be made exactly as shown, the entry will appear in lowercase Courier font. For instance, in the following VM/370 command, logon is entered exactly as shown, while *userid* must be replaced:

`logon` *userid*

CONTENTS

FIGURES

SUMMARY OF AMENDMENTS

Release 3 and Release 2.1, SH20-9067-2

CONTINUE Workspace not Dropped

Programming Change

Effective with Release 2.1 of VS APL, the *CONTINUE* workspace is not automatically dropped. There are no technical changes to VS APL under CMS resulting from Release 3 of VS APL.

Service Changes

Miscellaneous technical corrections and editorial changes have been made throughout the book.

TNL SN20-9207

SAVE command creates temporary file

Programming change

Describes the temporary file created when saving a workspace with the SAVE command.

Reconnecting to VS APL

Service Change

Describes procedure for reconnecting to VS APL after breaking the computer connection.

Storage Returned to CMS

Service Change

Describes how to determine the amount of storage returned to CMS when you contact VS APL.

New Messages

Service Change

Trouble report — NOT SAVED, NAME IN USE

Executor message — APL010I VIRTUAL MACHINE AND APL SHARED SYSTEM OVERLAP. APL ABORTED. (Replaces current message APL010I LOCATIONS xxxxxx THROUGH yyyyyy REPLACED BY SHARED APL SYSTEM.)

Executor message — APL159I FILE '*x* APLTMPWS' ALREADY EXISTS. WS NOT SAVED.

Canceling Output from CP and CMS Commands

Service Change

Describes how to cancel output from CP and CMS commands when using the CP/CMS Command Processor.

Character Translation Table

Service Change

A new table shows the character translation done with the 192 conversion option.

Workspace Conversion

Service Change

Provides information about character conversion for users converting to VS APL from other APL programs.

Release 2, SH20-9067-1

CMS VSAM Processor

New Programming Feature

VS APL under CMS now includes the CMS VSAM Processor which is used to perform file operations on entry-sequenced or key-sequenced VSAM files. A description of the processor has been added to the section "Auxiliary Processing" and new terminology associated with the processor has been added to the Glossary.

New VS APL Distributed Workspaces

New Programming Feature

Five new workspaces are distributed with VS APL under CMS: HOWEDITS, SEDIT, MEDIT, SBIC, and PRINT. A brief description of each workspace has been added to "Appendix A: Distributed Workspaces."

Indexed Specification

Specification Change

The previous VS APL restrictions on indexed specification have been relaxed. As a result, the VS APL Conversion Program no longer reports such specifications. The Conversion Report message that was previously produced has been removed from the list of messages in Appendix B.

Miscellaneous Changes

Auxiliary Processing

Service Changes

In addition to describing the new CMS VSAM Auxiliary Processor, several changes and additions have been made to the section "Auxiliary Processing" as follows:

- The description of the CP/CMS Auxiliary Processor has been generalized. The processor will accept any CMS command, however certain commands may terminate the auxiliary processor or VS APL.
- Corrections have been made to the table of data conversion for the 370 conversion option and to the description of conversion options for the CMS Disk I/O and FILEDEF I/O Auxiliary Processors.
- The table of auxiliary processor return codes has been updated.

Language Considerations

Service Changes

The APL Language manual (order number GC26-3847) contains information that previously appeared in "Appendix C: Language Considerations." The following discussions have therefore been removed from the appendix: "Outgoing Offer Query;" "System Variables ⎕*TT*, ⎕*UL*, ⎕*HT* and ⎕*TC*;" "Interrupting Input;" and "Indexing on the Left of an Assignment."

Display Terminal Considerations for APL

Service Changes

Various corrections and additions have been made to the discussion "IBM 3270 Display System Terminal Considerations for APL" in "Appendix D: CMS Terminals." In specific, the appendix now indicates that the 3270 Data Analysis-APL feature may be ordered with the standard APL keyboard or with an optional text keyboard.

Error Messages

Service Changes

A list of error reports relating to auxiliary processing has been added to the description of VS APL error reports in "Appendix E: Error Messages." Also, executor messages APL003I and APL004I have been added to the list of VS APL executor messages.

VS APL Batch Processing

Service Change

A new appendix has been added that describes how to execute VS APL batch jobs via the CMS Batch Facility.

INTRODUCTION

This book describes how you can use VS APL when it is operating under control of the Conversational Monitor System (CMS). CMS is a time-sharing system that operates under the Virtual Machine Facility/370 (VM/370).

Together, the *system* presented by the combination of VS APL and CMS allows you to interact with a computer conversationally through the powerful APL language. This book assumes that you already know the APL language as described in the publication *APL Language.*

In executing VS APL under control of CMS you are presented with the following features:

- A variety of different terminals that you can work from, including the IBM 2741 and 3767 terminals plus the IBM 3270 Information Display System Terminals.
- A set of instructions called *VS APL system commands* that allow you to monitor and control your work as well as send messages to other users.
- A number of programs called *auxiliary processors,* that allow you to perform file and other operations through shared variables.
- A collection of pre-defined or *distributed workspaces* that are helpful in learning APL, converting from other IBM APL systems, or using auxiliary processors.

Additionally, operating under control of CMS puts you in the *virtual environment* presented by VM/370. In this environment you operate as though you are in complete control of a simulated or *virtual computer.* You can work this way because VM/370 offers the central processing unit, storage and input/output devices of the real computer to each user of the system on a shared-time basis. Your terminal becomes the operator's console for the virtual computer, so that you are able to perform operations, like loading an operating system, that are normally beyond your control.

This book will specifically show you how to use the terminals, VS APL system commands, auxiliary processors and distributed workspaces supported by VS APL under CMS. It will also describe the aspects of the virtual environment that are pertinent to your work.

TERMINAL PROCEDURES

This section describes the terminals you can use with VS APL under CMS, some general conventions about making and correcting your terminal entries, and how output information is displayed at these devices.

Terminals

The terminals that you may use in working with VS APL under CMS are:

- IBM 2741 Communication Terminal
- IBM 3767 Communication Terminal
- IBM 3270 Interactive Display System Terminals (IBM 3275/3277)
- IBM 1050 Communication System Terminals (IBM 1052 Printer-Keyboard)
- CPT-TWX Model 33 and Model 35

All of these terminals have a typewriter-like keyboard for entering information and either a typewriter-like printer or display screen for recording your entries and displaying responses by the system. Most of them can be ordered with a number of different keyboards, and some can be used with different printing elements. As an APL user, you'll probably be working at an APL keyboard and, where printing elements are changeable, using a standard courier printing element to communicate with VM/370 and a standard APL printing element to communicate with VS APL. The elements required are listed in Figure 1.

		Standard Printing Element	
Terminal	**Keyboard**	**VM/370**	**VS APL**
2741	Correspondence	1167043	1167987
2741	PTTC/EBCD	1167963	1167988
2741	PTTC/BCD	1167938	1167988
1050	PTTC/EBCD	1167963	1167988
1050	PTTC/BCD	1167938	1167988

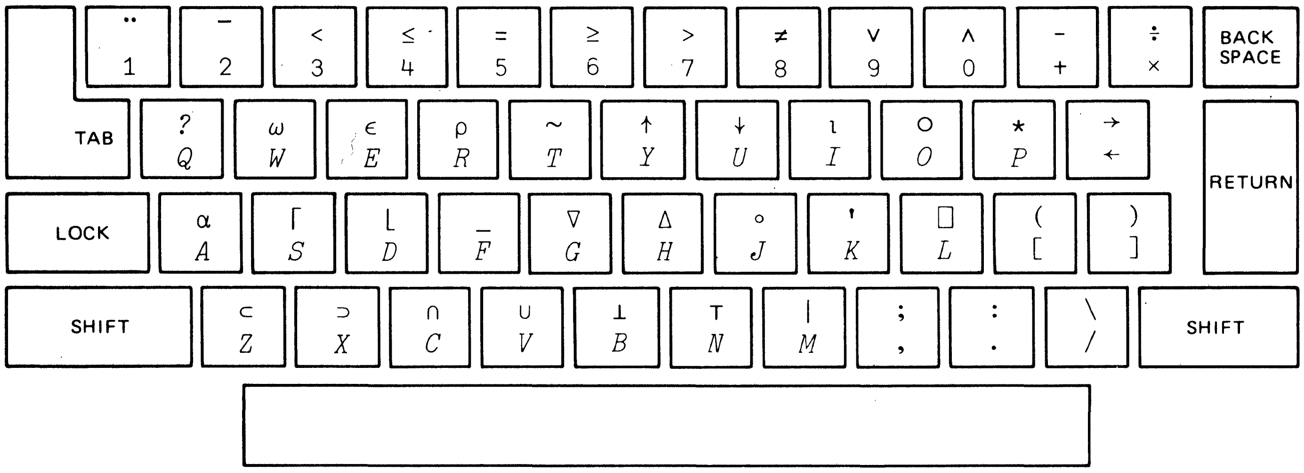
Figure 1. Standard Printing Elements for VM/370 Terminals

The IBM 3767 terminal uses a non-replaceable printing device whose set of printable characters is controlled by the switch marked EBCDIC (or Correspondence)/APL on the control panel. To communicate with VM/370 you should depress the EBCDIC part of the switch. To communicate with VS APL you should depress the APL part of the switch.

The operating procedures described in this book, unless otherwise indicated, assume that you are using an IBM 2741 terminal or an IBM 3767 terminal, with an APL keyboard. Existing IBM 2741 and IBM 3767 terminal keyboards can be modified by attaching *APL Characters* (order number GX20-1783) to the keys. The APL keyboards for these terminals are illustrated in Figure 2.

"Appendix D: CMS Terminals" summarizes the operating procedures for all the terminals that can be used with VS APL under CMS and describes restrictions and other items to be considered when using APL at these terminals.

IBM 2741 APL Keyboard



IBM 3767 APL Keyboard

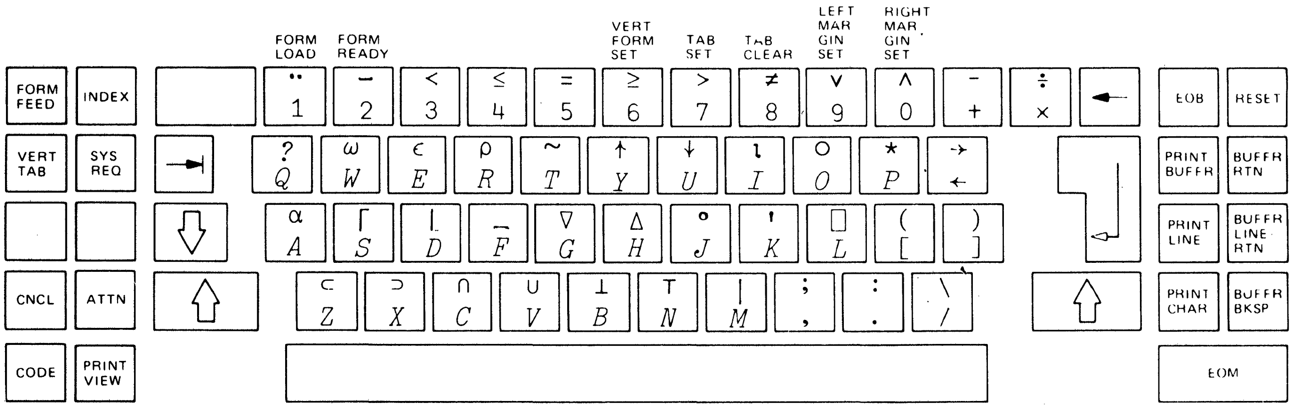


Figure 2. The IBM 2741 and 3767 APL Keyboards

Character Sets

The characters that you have available to create your entries depend on (1) the terminal you're working from, and (2) the part of the system you're communicating with.

When you communicate with VM/370, it assumes the non-APL character set associated with your terminal keyboard. These are the characters that can be printed using the standard VM/370 printing elements listed in Figure 1 or for the IBM 3767, the characters that can be printed when the EBCDIC (or Correspondence)/APL switch is set to EBCDIC or Correspondence.

When you establish contact with VS APL, the APL character set illustrated in Figure 3 is assumed. These are the characters that can be printed using the standard APL printing elements listed in Figure 1 or for the IBM 3767, the characters that can be printed when the EBCDIC (or Correspondence)/APL switch is set to APL.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A̲ B̲ C̲ D̲ E̲ F̲ G̲ H̲ I̲ J̲ K̲ L̲ M̲ N̲ O̲ P̲ Q̲ R̲ S̲ T̲ U̲ V̲ W̲ X̲ Y̲ Z̲

0 1 2 3 4 5 6 7 8 9

¨	dieresis	⍺	alpha	⍱	nor	~ ∨
¯	overbar	⌈	upstile	⍲	nand	~ ∧
<	less	⌊	downstile	⍒	del stile	∇ |
≤	not greater	_	underbar	⍋	delta stile	∆ |
=	equal	∇	del	⌽	circle stile	○ |
≥	not less	∆	delta	⍉	circle slope	○ \
>	greater	∘	null	⊖	circle bar	○ -
≠	not equal	'	quote	⍟	log	○ ⋆
∨	or	⎕	quad	⌶	I-beam	⊥ ⊤
∧	and	(	open paren	⍫	del tilde	∇ ~
-	bar	)	close paren	⍎	base null	⊥ ∘
÷	divide	[	open bracket	⍕	top null	⊤ ∘
+	plus	]	close bracket	⍀	slope bar	\ -
×	times	⊂	open shoe	⌿	slash bar	/ -
?	query	⊃	close shoe	⍝	cap null	∩ ∘
⍵	omega	∩	cap	⍞	quote quad	' ⎕
∊	epsilon	∪	cup	!	quote dot	' .
⍴	rho	⊥	base	⌹	domino	⎕ ÷
~	tilde	⊤	top			
↑	up (arrow)	|	stile			
↓	down (arrow)	;	semicolon			
⍳	iota	:	colon			
○	circle	,	comma			
⋆	star	.	dot			
→	right (arrow)	\	slope			
←	left (arrow)	/	slash			
			space			

Figure 3. The APL Character Set

Some of the characters shown in Figure 3, such as ⍞ are *compound characters* that are formed by entering a character, pressing the backspace key, and overstriking the first character.

Terminal Entry

Your dialogue with the system consists of entries made from the terminal and responses generated by the system. In general, an entry consists of a VM/370 command, a VS APL system command, an APL statement, or data.

You can never type more than one VS APL system command, APL statement, or data entry per line, and should not enter more than one VM/370 command per line.

All lines are completed by a carrier return. On the IBM 2741, the carrier return is the RETURN key, while on the IBM 3767, the carrier return is the key marked ⏎.

Once you complete a line, the terminal does not accept further information until the system has analyzed and executed the line. If it detects an error in your entry, it prints a message indicating the trouble. Otherwise, the system performs the operation and displays a response when called for. After acting on your request the system usually prompts you for your next line of input. It may indent six character positions or print input request characters such as

⎕:. The terminal then permits you to begin your next entry at that point. This method of entry and response continues until you sign off.

Figure 4 illustrates a typical dialogue between you and VS APL. The first line represents your entry while the second line represents the VS APL's response. The arrow indicates the starting point of the next entry.

```
      )CLEAR
CLEAR WS
      2×17 19
34 38
      ↑
```

Figure 4. A Typical Dialogue with VS APL

If a character is entered that is unrecognizable to VS APL, for instance ⍁, an *ENTRY ERROR* report is returned. The line you typed is then printed up to the unrecognizable character; the terminal then awaits further input. Any further characters that you enter are then substituted for the remainder of the entry.

Correcting Entries

There will be times when you'll have made a mistake before entering a carrier return and will want to make a correction.

If the error occurs in a VM/370 command, you have the option of (1) erasing the entire line by entering the character ¢; or (2) erasing the characters you just typed by entering an @ for each character to be erased. For example,

```
ipx a¢ ipl cms
iq@pl cms
ipl dm@@cms
```

all have the same effect, as if ipl cms had been typed without any errors.

If the error occurs in an entry to VS APL, you can erase all or part of the entry. To do so, backspace to the point of error and signal attention. On the IBM 2741 and IBM 3767, you press the ATTN key. The system responds by printing the inverted caret character (∨) under the character in error, advancing to the next line and awaiting further entry. This has the effect of erasing everything on the line from the point of the inverted caret to the end of the line, so that all characters from the point of error to the end must be retyped.

Suppose, for example, you typed the following statement at the IBM 2741 terminal:

```
      ADD2←76 24+10.5 1:.1
```

and you realized that you wanted to enter the number 7 instead of the character :. To make this correction, you simply backspace to the position of the character : and press the attention key. The system prints the inverted caret and advances to the next line. Now you enter 7.1.

In practice, the exchange would look like this:

```
      ADD2←76 24+10.5 1:.1
                        ∨
                        7.1
```

Tabs on Input

In communicating with VS APL, you may find it convenient to use the terminal's tab facility to save yourself the trouble of entering spaces. A tab stop is set by spacing or backspacing the carrier to the desired print position, then depressing the SET key on the IBM 2741, or holding the CODE key while pressing the numeral 7 key on the IBM 3767. Old tab stops should be cleared first.

When you set tab controls at the terminal, you must also communicate these settings to VS APL. You do this by assigning the settings (counting the left margin as position zero) to the APL system variable ⎕*HT*. A maximum of 26 settings can be assigned. The tab settings you indicate through ⎕*HT* are in effect until changed or until you sign off VS APL. When you next establish contact with VS APL, no tab settings are assumed and ⎕*HT* contains an empty vector. The value of ⎕*HT* may be changed by (1) respecifying it; (2) entering or completing a function where ⎕*HT* is localized (when execution of the function is complete, ⎕*HT* reassumes the value it had before the function was executed); or (3) issuing a)*LOAD* or)*CLEAR* command when ⎕*HT* is invalid (⎕*HT* will be assigned an empty vector).

For instance, if you physically set tab controls at positions 5, 10, 15, 20, 25, 30, 35 and 40 (counting the left margin as position zero) you must correspondingly set ⎕*HT* as follows:

```
        ⎕HT←5  10  15  20  25  30  35  40
```

Then, each time you press the tab key, the carrier will skip to the next tab setting, and VS APL will receive one or more space characters, exactly as if you had positioned the carrier by repeatedly pressing the space bar. On the IBM 2741, the tab key is marked TAB. On the IBM 3767, the tab key is marked ⇥.

Care should be taken that the value of ⎕*HT* actually reflects the physical tab settings, as misleading or unreadable output may otherwise be produced.

If no tab settings have been communicated to VS APL or if the carrier is beyond the largest tab setting, then pressing the tab key during keyboard entry will result in an entry error. This kind of entry cannot be corrected using the procedures described in "Correcting Entries." Instead, the line must be re-entered correctly.

Terminal Output and Display

One convenience of VS APL under CMS is that you can always follow the dialogue between it and yourself. The dialogue that is printed at the terminal consists of your printed entries and VS APL's responses. For instance, a typical dialogue between you and VS APL might include the following:

```
        ⍝ SEQUENCING
        NUMS←7 23 11 5 28
        NUMS[⍋NUBS]
VALUE ERROR
        NUMS[⍋NUBS]
                 ∧
        NUMS[⍋NUMS]
5 7 11 23 28
```

Messages and other printed responses sent to you by VS APL begin at the left-hand margin. This provides a handy way of distinguishing between your entries and VS APL's responses.

There may be occasions when VS APL attempts to print a character that is not representable at your terminal with the currently selected character set. When this happens, you'll see an error character. On the IBM 2741 terminal or IBM 3767, a blot (the character Z overstruck with the character N) will be printed instead of the unprintable character.

The examples illustrated in this book always indicate your entries and the system's responses as they would be printed at the terminal.

Tabs on Output

Terminal tab settings for VS APL output are established in the same way as they are for input. That is:

- Physical tab positions are set with the tab set (or SET) key.
- The physical tab positions must be communicated to VS APL through the system variable (⎕*HT*). Note that the value of ⎕*HT* may be changed as described in the discussion "Tabs on Input."

If tab settings have been communicated to VS APL, then it will use these tabs to reduce the time it takes to display information. The appearance of the printed output is not affected by the tab settings, only the time it takes VS APL to print it. For instance, the character string:

```
        A        B        C
```

is displayed the same whether the tabs are set for the positions of A, B, and C or not set but the printing time can be reduced if tabs are set for the positions of A, B, and C.

The value of ⎕*HT* and the physical tab settings must always correspond. Otherwise, misleading output may be produced.

Terminal Print Controls

Two system variables in VS APL have particular pertinence in controlling the format of output: these are ⎕*PW* (printing width) and ⎕*TC* (terminal control characters).

Printing Width

The system variable ⎕*PW* represents the maximum length of an output line—your printing width. The value assumed when you log on is 120 (or 79 for an IBM 3270 terminal). You can change this value by assigning a different value to ⎕*PW*. The value you assign can be any number between 30 and 255 (although a value greater than the physical capacity of the terminal is not advisable).

For example, the following assignment would limit printing to 40 characters per line:

```
      ⎕PW←40
```

so that an expression producing 40 output characters would print on one line:

```
      40ρ'A'
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

However, a printing width of 30 would force printing on two lines, the first line containing the first 30 characters while the remaining 10 characters would be printed, beginning in position 7, on the second line:

```
      ⎕PW←30
      40ρ'A'
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAA
```

The value you set for the printing width remains in effect throughout your communication with VS APL. When you next sign on, the printing width reassumes its default value. The value of ⎕*PW* may be changed by (1) respecifying it; (2) entering or completing a function where ⎕*PW* is localized (when execution of the function is complete, ⎕*PW* reassumes the value it had before the function was executed); or (3) issuing a)*LOAD* or)*CLEAR* command when ⎕*PW* is invalid (⎕*PW* will be restored to its last valid value).

Terminal Control Characters

The printing of output may also be controlled by the terminal control characters found in the system variable ⎕*TC*. The characters found in ⎕*TC* are set by the system and cannot be overridden by you. There are three terminal controls in ⎕*TC* as follows:

- ⎕*TC*[1]: backspace character
- ⎕*TC*[2]: new line character
- ⎕*TC*[3]: line feed character

When one of these elements is encountered by the system in the printing of output, the indicated terminal action is taken, so that the entry:

```
        'A',⎕TC[2],'BC',⎕TC[3],⎕TC[1],'DEF'
```

is a request to:

Symbol	**Meaning**
`'A'`	Print *A*
`⎕TC[2]`	Advance to the next line and reset to position 1
`'BC'`	Print *BC*
`⎕TC[3]`	Advance to the next line but stay in same relative position
`⎕TC[1]`	Backspace one position
`'DEF'`	Print *DEF*

In practice, the exchange would look like this:

```
        'A',⎕TC[2],'BC',⎕TC[3],⎕TC[1],'DEF'
A
BC
 DEF
```

Interrupting Output

You can interrupt the action that VS APL takes on your entries, and the possible printed responses it returns, by entering an interrupt signal at your terminal. On both the IBM 2741 and 3767 terminals, the interruption of output is controlled by the ATTN key. Pressing the ATTN key once, a *weak interrupt,* has the effect of terminating output immediately and interrupting execution at the end of the current line. Pressing the ATTN key twice, a *strong interrupt,* terminates output and terminates execution when the current primitive function is completed.

This ability to interrupt VS APL prevents you from being "locked out" while a long-running function is executing and particularly when a great deal of terminal printing is preventing you from efficiently doing your work.

If you press the ATTN key twice in quick succession you will be placed in contact with the control program segment of VM/370. To re-establish contact with VS APL, enter one of the following VM/370 commands, as appropriate: BEGIN, ATTN, or REQ. The BEGIN command directs VS APL to resume execution at the point it was last stopped. The ATTN or REQ command generates a weak interrupt signal in the current VS APL line.

Transmission Failures

Occasionally, the entry you make at your terminal or the output that the system returns may be garbled by certain failures in the transmission of data. If the failure occurs in your entry, the message `READ  ERROR` will be printed and you'll have to repeat the previous entry. If the failure occurs while the system is transmitting output, there may be one or more unexpected characters in the output line. The presence of these unexpected characters should be obvious in most contexts. Additionally, on the IBM 3767 the SYSTEM CHECK panel light should go on. However, on the IBM 2741, there is no absolutely certain way of detecting a transmission failure in output.

THE WORK SESSION

No matter what kind of work you're going to do at the terminal, there are some initial steps that you'll have to take to get things started. Similarly, there will be a number of things that you'll have to do to get things finished. In programming terms, these initial and final steps are said to start and end the *work session.* In this section you'll learn what these procedures are and how they may be affected by the equipment you're using.

Starting the Work Session

There are four steps involved in beginning a work session with VS APL. The first step is to establish a communications link between your terminal and the computer. Once a link has been established, the next step is to make contact with VM/370. The last two steps, which can be automatic, are to establish contact with CMS and VS APL.

The procedures described in this section are by no means all-inclusive. They only represent the most general ways of beginning a work session with VS APL. Certain requirements unique to your operations may alter the steps outlined in this section. To determine the additional actions you may have to take, see your system administrator.

Step 1: Connection

The way you make a connection between your terminal and the computer depends on the type of terminal equipment you're using. If your connection to the computer is through telephone lines, you will need to dial a telephone number. Some terminals are directly wired to the computer, in which case you'll have to set some switches, but you won't have to dial up.

The connection procedure for the IBM 2741 Communication Terminal and the IBM 3767 Communication Terminal will be covered here. For connection procedures related to other VM/370 terminals, see the *VM/370 Terminal User's Guide.*

Connecting the IBM 2741

An IBM 2741 can be permanently connected to a computer system through non-switched (leased) line, or temporarily connected through a switched (dial) line.

If you're working through a switched line, you have the additional option of using a telephone data set or a regular telephone with an acoustic coupler.

Figure 5 shows the procedure for connecting an IBM 2741. Steps 1 and 2 are the only steps that need to be carried out if you're using a non-switched line. Otherwise, steps 1 through 5 must be completed. Steps 3 and 5 additionally depend on whether you're using a telephone data set or an acoustic coupler to maintain the telephone connection; these steps do not apply to all acoustic coupler configurations, so check with your system administrator.

Switched and Non-Switched Lines

1. Find the COM/LCL switch on the left side of your terminal and switch it to COM.
2. For switched lines (excluding acoustic couplers) and non-switched lines, turn on the ON/OFF switch located at the right of the keyboard. (If the switch is on, turn it off, then on again.) For non-switched lines only, the message VM/370 ONLINE is displayed and the keyboard unlocks; at this point the terminal is connected to the computer and you are ready to establish contact with VM/370.

Switched Lines Only

3. **Telephone Data Set:** Press the TALK button.

   Acoustic Coupler: Make sure that the acoustic coupler is connected to the power supply, turned off, and connected to the terminal.
4. Remove the handset from the cradle and dial VM/370's telephone number. This number should be supplied to you by your system administrator.
5. **Telephone Data Set:** Wait for a high-pitched tone. When you hear it, push the DATA button and place the handset in the cradle. The DATA light will go on. Your keyboard will unlock; this means that the terminal is connected with the computer and you're ready to establish contact with VM/370. If the data light goes off at any time during the terminal session, start again from step 3.

   Acoustic Coupler: Wait for a high-pitched tone. When you hear it, place the handset face down in the coupler box, make sure the cord is in the slot, close the lid of the acoustic coupler and latch it. Turn on the acoustic coupler within 20 seconds. Turn on the ON/OFF switch located at the right of the keyboard. The keyboard will unlock, meaning the computer is ready to receive input data. If the keyboard does not unlock, start again from step 3.

Figure 5. Connecting the IBM 2741

Connecting the IBM 3767

Before a connection can be made between the IBM 3767 terminal and the computer system, certain switches on the terminal control panel must be set. Figure 6 lists the recommended switch settings for interaction with the system. On certain models of the IBM 3767, a security lock is provided. For these models, the key should be turned to ON before the POWER switch is turned on.

Like the IBM 2741, the IBM 3767 can be permanently connected to the computer through a non-switched (leased) line, or temporarily connected through a switched (dial) line.

If your terminal is operated through a non-switched line, once the switch settings in Figure 6 have been made, the connection to the computer is complete. The message VM/370 ONLINE is displayed and the PROCEED light on your terminal console will go on. At this point you are ready to establish contact with VM/370. If the light is not on, check to see that the security lock, if any, is in the ON position and the switches have been properly set.

To complete the connection procedure for a switched line, follow the instructions in Figure 7.

Switch	Setting
COMM/LOCAL	COMM
AUTO/OFF	AUTO
EDIT/OFF	OFF
AUTO VIEW/OFF	AUTO VIEW
DATA/TALK[1]	DATA
DIAL DISC/OFF[2]	OFF
SDLC/SS	SS.
EBCDIC(or Correspondence)/APL	EBCDIC or Correspondence as appropriate to your terminal keyboard. This switch is set to APL when communication with VS APL is to begin.
CALC/OFF	OFF
TEST/OFF	OFF
POWER/OFF	POWER

1 This switch appears on terminals supplied to World Trade countries except Germany.

2 This switch appears on terminals supplied to Germany only.

Figure 6. IBM 3767 Switch Settings

Connection Procedure

1. Press the TALK button, remove the handset from the cradle and dial VM/370's telephone number. This number should be supplied to you by your system administrator.
2. Wait for a high-pitched tone. When you hear it, switch from TALK to DATA and replace the handset. (This procedure depends on the type of telephone set you're using; check with your system administrator if necessary.)
3. The DATA SET READY light should go on followed by the PROCEED light. You are now ready to establish contact with VM/370. If the DATA SET READY light goes off at any time during the terminal session, try again from step 1.

Figure 7. Connecting the IBM 3767 (Switched Line)

Step 2: Contacting VM/370 (Logging On)

Once a connection has been made between your terminal and the computer, the next step is to contact the VM/370 system. This step is termed *logging on.* During the logon procedure it is important that you use the proper VM/370 printing element or for the IBM 3767, have the switch marked EBCDIC (or Correspondence)/APL set in the non-APL position.

To begin the logon procedure press the attention key. Unless you're connected to the computer via a leased line, the system responds by displaying one of the following messages:

```
vm/370 online

vm/370 online xxxxxx xxxxxx

xxxxxx xxxxxx vm/370 online
```

This lets you know that the system is ready to accept your logon command. (The xxxxxx xxxxxx portion of the message, if it appears, consists of meaningless characters and should be ignored.)

Now you're ready to enter the logon command. Press the attention key and type:

```
        logon   userid   m
```

where:

userid
  is the identification assigned by your system administrator.

m
  is an optional request to hide your entry password.

The entry logon can be abbreviated l, lo, log, and so on.

If you make an error, VM/370 will not accept your logon. Instead, it will display the reason the logon request was unsuccessful. You must then repeat the logon command correcting the indicated problem, if possible.

If you typed the logon command correctly, the system will respond by requesting your entry password:

```
ENTER PASSWORD
█████████
```

Once the blots are printed, you can enter your password over them, which keeps other users from learning what your entry password is. Like your identification number, your entry password is issued by your system administrator.

On some terminals, the function of the blots is handled by a print suppress feature. Under this arrangement, no blots are produced, and the password you enter is not displayed.

If the password you enter is not acceptable, VM/370 may repeat its prompt or ask you to re-enter the logon command.

Once your password is accepted, you are logged on to VM/370. At this point you are in contact with the *control program segment* of VM/370. You may now, if you wish, enter any of the commands acceptable to the control program segment. For a description of these commands, refer to the *VM/370: CP Command Reference for General Users.* You may now also see a logon message indicating the time and date of logon. For instance:

```
LOGON AT 17:37:55  PDT  SATURDAY   07/17/76
```

What you do next depends on how procedures have been established at your place of work. If certain requirements have been fulfilled by the personnel who have installed your system, a successful logon to VM/370 may automatically entail a connection to CMS and VS APL. If so, all you need to do is press the carrier return key. You will see the message:

```
        v s  a p l
```

At this point, you should change your printing element to a standard APL element or set the switch on the 3767 to APL. The "vs apl" message means that you have established contact with VS APL and that CMS has given you an area of virtual storage to make your APL entries. This area of storage is your *active workspace*. The active workspace you receive is either:

- Clear—a workspace where no information has been entered. You get a clear active workspace if you did not continue an active workspace from a previous session with VS APL.
- Continued—a workspace in which information has already been entered. You will receive a continued active workspace if you have a workspace named *CONTINUE* in your library when you log on.

You'll know that you have been given a clear workspace if VS APL displays the following message:

```
clear ws
```

If you receive a clear workspace, you'll be free to begin your entries. If you receive a continued workspace, the following message will be displayed:

`saved` *time date*

where:

time date
  is the time and date the continued workspace was saved.

You can then make your entries into the continued workspace.

Step 3: Contacting CMS

If a connection to CMS is not automatically generated by your logon, you'll have to establish a connection yourself. In VM/370 terms, you "load CMS into your virtual computer or machine." You do this by issuing the following command:

```
ipl cms
```

This command tells VM/370 that you will be doing your work in the CMS environment and makes your disk space available for use. The system administrator should have allocated some disk space for your permanent use when he gave you your identification and entry password. He should have also defined for you a sufficiently large virtual computer for executing VS APL.

In response to the IPL command, VM/370 displays a message such as:

```
CMS VERSION x.x PLC yy date
```

to indicate that the IPL command executed successfully and that CMS is loaded. (The x's and y's indicate a particular version of CMS.)

Once CMS is loaded you may enter any of the VM/370 commands acceptable to CMS. If this is the first time you are using VS APL under CMS, you may need to issue CMS commands to set up your disk space for VS APL work, that is *format your virtual disk.* For a description of how to format your virtual disk as well as a description of CMS commands, refer to the *VM/370: Users Guide* or the *VM/370: CMS Command and Macro Reference.*

Step 4: Contacting VS APL

If you're going to be doing your work with VS APL, it may be that execution (either explicitly by you or implicitly as a result of the logon command) of the IPL CMS command will automatically establish contact with VS APL and will prepare your virtual machine for APL entries. If so, to complete contact with VS APL press the carrier return key. If contact with VS APL is not automatically established, enter the following command:

`apl` *sharesize processors*

where:

sharesize
is an optional shared memory size, in bytes, that may be specified if you are going to use auxiliary processors. If a shared memory size is omitted and at least one auxiliary processor is specified in the command, a size of 4096 bytes is assumed. This is also the minimum size that can be used by the system, so that any specified size less than 4096 is taken as a request for 4096 bytes. If a size is specified, it may be immediately followed by the letter K or the letter M. A specification of K indicates that the shared memory size is to be multiplied by 1024. A specification of M indicates that the shared memory size is to be multiplied by 1,048,576. The value of sharesize also determines the amount of storage that is returned to CMS. After loading any auxiliary processors, VS APL obtains all remaining user storage in the virtual machine. It suballocates shared memory (the size is the value of sharesize), a 512-byte work area for each auxiliary processor (rounded to a 4K boundary), and a 4K area for communication between VS APL and CMS. After suballocation, the following amount, in bytes, is returned to CMS:

```
32768⌈16384+2×SHARESIZE
```

The remaining user storage is allocated for the active workspace.

processors
are the names of auxiliary processors. Each name may be followed by a parenthesized list which is used to pass parameters to the auxiliary processor. If distributed auxiliary processors are built into the APL system at your installation, do not specify their names when invoking APL.

The following command, for example, requests contact with VS APL, asks for a shared memory size of 10,240 bytes, indicates an auxiliary processor named APFILE, and provides a parameter list for APFILE.

```
apl 10k apfile(parma parmb parmc)
```

If the APL command is executed successfully, the system will return the message:

```
v s  a p l
```

and follow it with the message CLEAR WS or SAVED as appropriate. At this point, change the printing element to a standard APL printing element or set the switch on the IBM 3767 to APL. You can now begin your session with VS APL.

Example

Now that you have read through the description of logging on, let's look at an example. Let's assume that you are using a IBM 2741 terminal and that you have the proper courier printing element mounted. In addition, let's assume that the following has been supplied to you by your system administrator:

```
userid=123456
password=APLUSER
```

Here's what your logon procedure would look like. Your entries are in lowercase characters, and the system's responses are in uppercase characters:

```
logon 123456 m
ENTER PASSWORD
■■■■■■■■■
apluser
LOGON AT 14:04:33 PDT SATURDAY 12/11/76
ipl cms
CMS VERSION 2.0 PLC 13 12/11/75
apl
```

At this point, the system will return the message "vs apl" followed by CLEAR WS. Mount the APL typing element. Now you're ready to begin your session with VS APL.

Ending the Work Session

When your work with VS APL is finished, you have various options as to what you do next. You can simply end the session, thereby breaking all connection with VS APL, CMS and VM/370; you can end the session but maintain contact with CMS; or you can do either of these things and have your workspace stored at the same time. The way you tell the system is through one of four APL system commands:

```
)OFF
)OFF HOLD
)CONTINUE
)CONTINUE HOLD
```

Ending Contact with VS APL and CMS

The APL system commands)*OFF* and)*CONTINUE* end your contact with VS APL, CMS and VM/370.

The difference between these two commands is how they handle your active workspace. Once)*OFF* is executed, the information in your active workspace is lost. Once)*CONTINUE* is executed, a copy of your active workspace is saved, making its contents available for your next VS APL session. (The workspace is given the name *CONTINUE* and replaces any present workspace in your library named *CONTINUE*.)

In response to the *)OFF* command or the *)CONTINUE* command, the system displays the following message:

```
CONNECT=t1   VIRTCPU=t2   TOTCPU=t3
LOGOFF AT   t4   zone  day  date
```

where:

t1
: is the actual clock time spent in the current terminal session in hours:minutes:seconds.

t2
: is the virtual CPU time used in the current terminal session in minutes:seconds.hundredths of seconds.

t3
: is the total CPU time used in the current terminal session in minutes:seconds.hundredths of seconds.

t4
: is the time of logoff in hours:minutes:seconds

zone
: is the time zone, for instance PDT for Pacific Daylight Time.

day
: is the day the message is sent.

date
: is the date the message is sent in months/days/years.

When the *)CONTINUE* command is executed, the system precedes the sign-off message with a message showing the time and date the active workspace was saved.

Figure 8 illustrates a dialogue that ends your session with VS APL, CMS, and VM/370:

```
      )OFF
CONNECT=00:07:36 VIRTCPU=000:00.65 TOTCPU=000:02.05
LOGOFF AT 11:17:16 PDT FRIDAY 06/18/76
```

Figure 8.   Using the)OFF Command

In this example the workspace is not saved. If *)CONTINUE* were specified, a date and time message for the save operation would precede the sign-off messages.

Ending Contact with VS APL Only

The APL system commands *)OFF HOLD* and *)CONTINUE HOLD* end your contact with VS APL but maintain your contact with CMS.

When you specify *)OFF HOLD* the contents of your workspace are lost. When you specify *)CONTINUE HOLD*, a copy of your workspace is saved making its contents available for your next session with VS APL. (The workspace is given the name *CONTINUE* and replaces any present workspace in your library named *CONTINUE*.)

When the *)CONTINUE HOLD* command is executed, the system prints a message showing the time and date the active workspace was saved, followed by the characters ρ[(these are the characters R; assuming a Courier printing element). When the *)OFF HOLD* command is executed, only the ρ[characters are printed. Once these messages are displayed, change your typing element or on the IBM 3767, set the switch marked EBCDIC (or Correspondence)/APL to EBCDIC or Correspondence. You are now in contact with CMS.

Breaking the Computer Connection

The step that actually completes your interaction with the computer is a physical one. It may simply be turning the terminal off and, if you're using a telephone, hanging up.

Normally, to finish your work at the terminal, you should enter the APL system command *)OFF* or *)CONTINUE* as appropriate. The system will take the indicated action, including the printing of accounting information. You can then hang the phone up and turn the terminal off.

If you turn the terminal off before entering these commands, the system disconnects you from VM/370. You have 15 minutes to log on. If you log on within 15 minutes, you can re-establish contact with VS APL and recreate the status of your virtual computer by entering the following:

```
TERMINAL APL ON LINESIZE 255
SET EMSG OFF
SET IMSG OFF
BEGIN
```

In addition, if message blocking was in effect, you must also enter the following before entering `BEGIN`:

```
SET MSG OFF
SET WNG OFF
```

(Message blocking is described in "Communication Commands" in the section "System Commands.")

If the terminal is a typewriter terminal (that is, is not a 3270 display terminal) you must also enter the following before entering `BEGIN`.

```
TERMINAL ATTN OFF
```

If you break the computer connection and move to another terminal type during your VS APL session, you must ensure that VS APL changes its terminal handling by doing the following: enter the commands shown above, then save your workspace, end your VS APL session, and restart VS APL.

If you do not logon within 15 minutes, the status of your virtual computer plus the contents of your active workspace are lost.

Forced Endings

Sometimes, certain malfunctions in the computer, such as temporary losses of power, or failures in the telephone circuits may endanger the integrity of your workspace. To safeguard against this possibility, the system will break the connection. This is called a *forced ending.* The system handles a forced ending as if you turned the terminal off before entering an `)OFF` or `)CONTINUE` command.

WORKSPACES AND LIBRARIES

This section describes the content and attributes of VS APL workspaces and details the libraries supported by CMS. Workspaces are areas of virtual storage that contain your work. Libraries are areas where copies of workspaces are stored.

The Workspace

A workspace is an area of virtual storage that contains all the data, functions, variables, and groups that you define in your VS APL entries. In addition, a workspace contains certain system variables and system functions that monitor or control the nature of your work.

When you establish contact with VS APL, you are automatically given a workspace to use. This is called your *active workspace.* All interaction that you have with VS APL from this point until you end your work is made through the active workspace.

It is possible to request that a duplicate of a workspace be saved in a library. When that is done, a copy of the entire active workspace is saved, including all the data, functions, status and control information within it. When you subsequently ask to retrieve a saved workspace, you receive a duplicate of what was saved. This restores everything to the way it was at the moment the workspace was last used.

Workspace Attributes

Each workspace that you use is qualified by certain *workspace attributes.* These attributes can be changed without affecting the variables and defined functions you have in your workspace, but they may affect the results of APL statements that are being executed.

Unless you're continuing some previous work, the workspace initially presented to you when you establish contact with VS APL is a *clear workspace.* This is a workspace in which no data has been entered, no names defined, and in which the workspace attributes indicate standard initial values. As you make entries, the values of some of these attributes change. This may happen implicitly as functions are being executed or as more of your workspace is being filled up. The values of the attributes may also change explicitly through certain system variables or system functions.

Figure 9 lists the attributes of a VS APL workspace, indicating the possible range of values that they can assume and the standard initial values they have in a clear workspace.

Seven of the attributes (index origin, latent expression, line counter, printing precision, state indicator, comparison tolerance, and random link) are fully described in the publication *APL Language.* The remaining attributes involve workspace identification, size, and organization, topics that are covered in the following discussions.

Attribute	Possible Values	Value in a Clear WS	Can Be Changed By	Value Returned By
Name	See "Workspace Identification" in this section	*CLEAR WS*	)*WSID*)*SAVE*	)*WSID*
Password	See "Workspace Identification" in this section	None		
Size	See "Workspace Size" in this section	Depends on virtual machine size	)*LOAD*)*CLEAR*	)*WSSIZE*
Symbol Table Size	See "Workspace Organization" in this section	256	)*SYMBOLS* number	)*SYMBOLS*
Execution Control Area	See "Workspace Organization" in this section	512	)*STACK* number	)*STACK*
Available Work Area	See "Workspace Organization" in this section	Depends on virtual machine size	Changed via user action	⎕*WA*
State Indicator	See "Workspace Organization" in this section	Empty	Changed via user action	)*SI*)*SINL*
Index Origin	0 or 1	1	⎕*IO*←number	⎕*IO*
Latent Expression	Any character vector	Empty	⎕*LX*←character vector	⎕*LX*
Line Counter	Integer Vector	Empty	Changed via user action	⎕*LC*
Printing Precision	1 through 16	10	⎕*PP*←number	⎕*PP*
Comparison Tolerance	0 through 2*¯32	1*E*¯13	⎕*CT*←number	⎕*CT*
Random Link	integer 1 through (2*31)-1	7*5	⎕*RL*←number[1]	⎕*RL*

[1]The value of ⎕*RL* is changed implicitly by the ? primitive functions.

Figure 9. VS APL Workspace Attributes

Workspace Identification

Each workspace in VS APL is identified by a library number and a name. Library numbers are discussed in "The Library" later in this section.

A workspace name can be any combination of letters and numbers beginning with a letter. It cannot contain blanks, underscored letters, or special characters and should not be more than eight characters long. (Longer names are truncated on the right.)

Passwords

In addition to an identification, a workspace may have a read or write password associated with it. A read or write password is associated with a workspace if it resides on a virtual disk that is password-protected. This is only true for workspaces stored in a project library. Read and write passwords are assigned to virtual disks when your system is installed, and once assigned cannot be changed by you. (For further infomation on virtual disks, see "Workspaces and Libraries in the CMS Environment" later in this section.)

If a workspace is stored in a project library that is protected by a read password, any system command that requests reading from the project library (the commands)*LOAD*,)*COPY*,)*PCOPY* and)*LIB*) must include the proper read password. If a workspace is to be saved on or dropped from a virtual disk that is protected with a write password, the system commands)*SAVE* or)*DROP* must include the proper write password; for a save operation, if the workspace already is associated with the proper write password, no password is required in the)*SAVE* command.

For example, the following command drops a workspace from a project library that is protected with the write password SECRET:

```
)DROP 98226 TEST:SECRET
```

If you fail to specify the correct password when required, you will be prompted for one by the system.

To determine which workspaces, if any, require a password when referenced, ask your system administrator.

Workspace Size

Unless you change it, the size of your active workspace corresponds to the size of your virtual computer minus whatever is taken up by VS APL.

If you want to increase the size of your active workspace above this size, you must increase the size of your virtual computer. To do this, you must issue the VM/370 DEFINE command as described in the *VM/370: CP Command Reference for General Users.* The maximum workspace size that you can receive is 12 million bytes.

You can, however, directly control the size of your active workspace within the limits created by the virtual machine size through the system commands:

```
)CLEAR size
```

and

```
)LOAD wsid size
```

The command)*CLEAR size* activates a clear workspace whose size is the value of *size.* The command)*LOAD wsid size* activates a workspace whose identification is represented by *wsid* and whose size is the value of *size.* The system will add some area to your size specification for its own needs so the size allocated is somewhat greater than you specify. The minimum *size* that you may specify is 17,500.

You can determine the size provided by the system for your active workspace by issuing the system command:

```
)WSSIZE
```

You can also issue the command)*QUOTA* to determine the default size of your active workspace. Under CMS this is also the maximum size that you can specify.

Workspace Organization

A workspace is divided into a number of separate areas in which different types of information are contained. Aside from controlling the overall size of your active workspace, you also can control the size allotted to specific areas of the workspace. This is particularly useful when a lack of space in one or another workspace area halts the progress of your work.

Figure 10 illustrates how the active workspace is divided by the system and the means you have available to monitor and control the size of these areas. Three areas are shown: the symbol table with its accompanying command)*SYMBOLS*, the execution control area with its accompanying command)*STACK* and the dynamic storage area with its accompanying system variable ⎕*WA*.

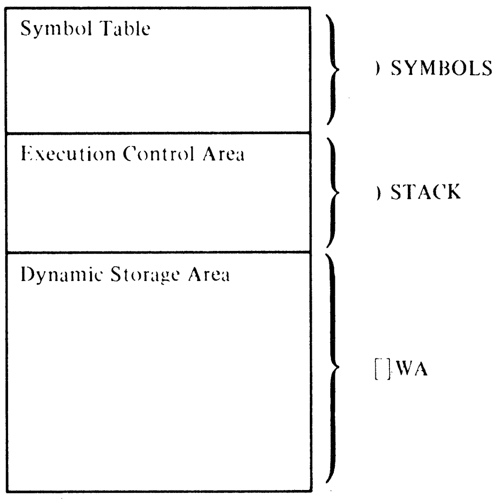


Figure 10. Organization of a Workspace

The *symbol table* is used by VS APL to keep track of names occurring in the workspace. In a clear workspace, the number of entries permitted in the table is 256.

If you want to change the number of symbols accommodated in the symbol table, you may issue the command:

```
        )SYMBOLS number
```

where *number* is the number of symbols you want accommodated in the table. You can issue the `)SYMBOLS` *number* command only while your active workspace is clear.

At any time during your session with VS APL, you can determine both the maximum number of entries that can be included in the symbol table and the current number of names in use, through the system command:

```
        )SYMBOLS
```

The *execution control area* is used to contain information that is generated during function execution. In particular, the execution control area contains the *state indicator* which indicates the progress of function execution. (In a clear workspace the state indicator is empty.)

A clear workspace contains an execution control area that can accommodate 512 entries. You can modify the number of entries permitted in the execution control area through the system command:

```
        )STACK number
```

where *number* is the number of entries to be permitted.

If the `)STACK` command is issued without a number, the currently allowable number of entries in the execution control area is displayed.

The remaining area, sometimes called the *dynamic storage area,* is the area in which data, function definitions, and group definitions are kept. Unlike the symbol table and the execution control area, the size of the dynamic storage area cannot be explicitly controlled. However the size of the dynamic storage area can be indirectly increased or decreased by respectively decreasing or increasing the size of the symbol table and execution control areas. Similarly, the size of the dynamic storage area will increase or decrease as the total size of the workspace is increased or decreased. As work progresses, you may find it useful to determine how much dynamic storage area is still unused. At any time during your work, you may determine the amount of dynamic storage remaining available by displaying the system variable ⎕*WA*.

The Library

Inactive workspaces that you or other users of VS APL have saved are stored in *libraries* which are identified by number. Libraries are classified as private, project, or public depending on their accessibility to users of the system.

Everyone who uses VS APL under CMS owns a private library, may have access to a project library and has access to all public libraries.

Private Libraries

As a user of VS APL under CMS, you own a private library which is available only to you. You can store copies of your active workspace there, retrieve copies of stored workspaces from it, display its contents, or drop workspaces from it when you no longer need them.

Any reference you have to a workspace in your private library is made using its workspace name. No library number need be supplied. You can, however, optionally specify the number 1001. For instance, either of the following commands could be used to retrieve a copy of a workspace named *GAMES* stored in your private library:

```
)LOAD GAMES
)LOAD 1001 GAMES
```

Project Libraries

A project library is intended for users who want to share workspaces. If you have full access to a project library, you can perform any of the operations that you can perform on your private library.

Any reference you have to a workspace in a project library is made using the library number of the project library and the workspace name. Additionally, the system administrator may have assigned separate read or write passwords to the virtual disk on which the project library resides. If so, any reference you have to a workspace in the project library must indicate the proper read or write passwords. (For further information, see "Passwords" earlier in this section.) For instance, the following command is used to copy the object *ITEM* from the workspace *ACCOUNTS* in project library 200; *ACCOUNTS* is protected with the password *PSW*:

```
)COPY 200 ACCOUNTS:PSW ITEM
```

Public Libraries

A public library is available to all users of VS APL under CMS. You may list the contents of a public library or retrieve copies of workspaces stored there. If you are not allowed to save or drop workspaces in a public library, you can request your system administrator to perform these operations for you.

Any reference you have to a workspace in a public library is made using the library number of the public library and the workspace name.

Workspaces and Libraries in the CMS Environment

Although your interaction with VS APL operates in terms of workspaces and libraries, CMS organizes these collections of information as files. To CMS, each workspace is a CMS file, so that a library is simply a collection of files.

For the most part, this internal representation will not be apparent to you. However, if you want to make use of the facilities of VM/370 to tailor your virtual computer, you should understand the CMS file concept. For further details, refer to the *VM/370: CMS User's Guide.*

CMS Files and Virtual Disks

In CMS, a *file* is a logically related group of records that is defined via CMS commands. Each CMS file resides on a *virtual disk,* that is a logical subdivision of a real disk. When you logon, VM/370 logically attaches one or more virtual disks to your virtual computer. Each disk is identified by a letter and an address. The A-disk, specifically, contains your private library and most of your file operations cause the system to read from or write to this disk.

Each CMS file on a virtual disk is identified by a filename, a filetype, and a filemode. The filename, which is limited to eight characters, simply names the file. The filetype indicates how the file functions in the CMS system. For instance a filetype can typically be a MODULE file containing object programs or a LISTING file containing program listings. The filemode indicates which disk the file is on.

VS APL Virtual Disk and File Assignments

CMS organizes VS APL workspaces in private, project and public libraries as follows:

- Private Libraries—Each workspace in your private library is a file on your A-disk. Each of these workspaces has a filename that is the same as its workspace name, a filetype of VSAPLWS, and a filemode of A.
- Project Libraries—Each workspace in a project library is a file on the G-disk having address 197. Each project library workspace has a filename that is the same as its workspace name, a filetype composed of the letter W and a seven-digit library number (for instance W0000101 for library 101), and a file mode of G.
- Public Libraries—Each workspace in a public library resides on a Y-disk (unless the library is defined on another disk by your installation). The filename and filetype of a public library workspace is the same as a project library workspace. The filemode of a public library workspace is Y.

SYSTEM COMMANDS

This section describes the system commands available when you use VS APL under control of CMS. Through system commands you can:

- Monitor and control the content and attributes of the active workspace.
- Monitor and control the contents of libraries.
- Communicate messages to other terminals.
- Sign off VS APL, CMS, and VM/370.

Using System Commands

To use system commands you simply enter them at your terminal just as you enter APL statements. System commands cannot be used in APL expressions and cannot be executed as part of or input to a function definition. Otherwise, a system command can be entered any time you are in contact with VS APL, in which case it is executed immediately. System command names must be entered as shown below, without truncation or extension.

System Command Types

System commands can be grouped into five types based on the kinds of operations they perform:

- Library Control Commands, which are used to control the contents of libraries.
- Workspace Control Commands, which are used to control the contents and attributes of the active workspace.
- Inquiry Commands, which are used to return information about libraries and the active workspace.
- Communication Commands, which are used to communicate information to other terminals.
- Sign-Off Commands, which are used to sign off the system.

Library Control Commands

Library control commands are used to control the contents of libraries. There are two system commands that belong in this category:)*SAVE*, which saves a copy of the active workspace in a library, and)*DROP*, which drops a workspace from a library.

Saving the Active Workspace: The)SAVE Command

You can save a copy of your active workspace by issuing the following command:

)*SAVE* *wsid*: *password*

where:

wsid
: is a workspace identification to be assigned to the active workspace before saving. If omitted, the workspace identification currently associated with the active workspace is retained. If the workspace does not have a name, *wsid* must be specified.

:
: is an optional separator, required only when a password is indicated.

password
: is required only if the workspace is to be saved in a project library; the password is the write password, if any, associated with the virtual disk on which the library resides. If the library is not a project library, any password specification is ignored. The system will prompt for the correct password if it is not specified when required.

When the)*SAVE* command is issued, a copy of the active workspace is stored in the library indicated by the current workspace identification. If *wsid* is specified, the current workspace identification is changed to the one indicated.

When you save a copy of your active workspace, all the attributes of the active workspace are retained by the stored copy. In particular, the library number and workspace name are stored with the copy, and the present size of the active workspace becomes the size of the copy. Most important, all the objects in the active workspace (functions, variables, and so on) retain their values in the stored copy (any shared variables are saved with their most recent values).

Saving a copy of your active workspace does not destroy the original. While a copy now exists in a library, the original is still available until you activate another workspace or sign-off.

After a successful save operation, the system responds with a message showing the time and date when the save operation was performed.

Figure 11 illustrates two instances of saving. In the first case, a copy of an active workspace named *TABLES* is saved with no change in its workspace identification. Notice that the response returned by the system includes the name of the saved workspace. In the second case, a copy of the active workspace is saved with a new workspace identification, a write password is required for the save operation.

```
      )SAVE
11:15:27 07/06/76 TABLES

      )SAVE 123456 NEWNAME:ABC
14:22:01 11/12/77
```

Figure 11. Using the)SAVE Command

Saving is not permitted when the workspace identification given in the command matches an identification of an existing saved workspace but does not match the identification of the active workspace. In other words, if you attempt to save a workspace named TEST with the command `)SAVE 75 NEWNAME`, and `NEWNAME` already exists in library `75`, the save operation will not be executed. This restriction prevents you from inadvertently overwriting one workspace with another. If you want to overwrite the workspace, you have to first change the identification with the `)WSID` command before you attempt to save it.

Because the `)CONTINUE` command saves a workspace named `CONTINUE` in your library, you should avoid using `CONTINUE` explicitly as a workspace name in a `)SAVE` command. Using `)SAVE CONTINUE` might result in overlaying a previous `CONTINUE` workspace that has been saved as the result of a forced ending.

You may not save a workspace in a public library directly. If you want a workspace saved in a public library, see your system administrator.

When a `)SAVE` command is issued for a workspace that already exists in the library, a copy of the active workspace is saved as a temporary file having the same filename, and a filetype of APLTMPWS. The old workspace filename is erased and the temporary file is renamed to the old workspace name. This procedure reduces the possibility that a workspace will be lost due to a system failure during a `)SAVE`.

Note: Enough library space must be available for both the old and temporary workspace files. If your library has insufficient space, you may `)DROP` the old workspace and then re-issue the `)SAVE`—at somewhat greater risk.

The `)LOAD`, `)LIB`, `)COPY`, and `)PCOPY` commands ignore the existence of a temporary file.

Dropping a Workspace from a Library: The)DROP Command

You can drop a workspace from a library by issuing the command:

`)DROP` *wsid*:*password*

where:

wsid
is the workspace identification of the workspace to be dropped.

:
is an optional separator, required only when a password is indicated.

password
is required only if the workspace to be dropped resides in a project library; the password is the write password, if any, associated with the virtual disk on which the library resides. If the library is not a project library, the password is ignored. The system will prompt you for the correct password if it is not specified when required. A temporary workspace file with the same workspace identification will also be dropped. See the `)SAVE` command for a description of the temporary workspace file.

The system responds with a message indicating the time and date the workspace was dropped. For example, to drop a workspace named `FINANCE` from your private library, you specify:

```
      )DROP FINANCE
14:22:27 07/10/77
```

If you want to drop the workspace *ACCOUNT* from project library 995053 associated with write password VS202, specify:

```
        )DROP 995053 ACCOUNT:VS202
14:07:13 07/08/77
```

You may not drop a workspace from a public library directly. To have a workspace dropped from a public library, see your system administrator.

Workspace Control Commands

Workspace control commands are used to change the contents and attributes of the active workspace. The workspace control commands are:

-)*LOAD* which retrieves a copy of a saved workspace and brings it into the active workspace.
-)*COPY* which copies global objects from a stored workspace into the active workspace; the active workspace is not protected.
-)*PCOPY* which copies global objects from a stored workspace into the active workspace; the active workspace is protected.
-)*GROUP* which gathers global objects in the active workspace into a group or disperses a group.
-)*CLEAR* which activates a clear workspace.
-)*ERASE* which erases global objects from the active workspace.
-)*WSID* which changes the workspace identification of the active workspace.
-)*SYMBOLS* which changes the number of entries allowed in the symbol table area of the active workspace.
-)*STACK* which changes the number of entries allowed in the execution control area of the active workspace.

Retrieving a Workspace from a Library: The)LOAD Command

To retrieve a saved workspace from a library, you specify the following system command:

```
        )LOAD  wsid:password  size
```

where:

wsid
is the workspace identification of the workspace to be retrieved.

:
is an optional separator, required only when a password is indicated.

password
is required only when the workspace to be retrieved resides in a project library; the password is the read password, if any, associated with the virtual disk on which the library resides. If the library is not a project library, the password is ignored. The system will prompt for the correct password if it is not specified when required.

size
is an optional size specification, required when you want to change the size of your active workspace.

When you retrieve a saved workspace, a complete copy of the saved workspace is brought into your active workspace. All the workspace attributes have the values they had when the saved workspace was last active. In addition, all the objects in the retrieved workspace (functions, variables, and so on) retain their previous values. Any shared variables in the previously active workspace are retracted.

In retrieving a workspace, you do not destroy the library copy. In effect, two copies now exist, one in your library and one in your active workspace. Unless you specifically overwrite the library copy, or the workspace is dropped from the library, the library copy always remains intact.

If the)*LOAD* command is successful, the system returns a message indicating the time and date the workspace was last saved. The system may also follow this with the message *WSSIZE IS* *size* if the retrieved workspace changes the size of your active workspace and you did not explicitly define a size in the)*LOAD* command. The new workspace size is indicated by *size*.

Figure 12 illustrates how you can use the)*LOAD* command to retrieve a copy of a stored workspace. The first example in Figure 12 shows how you can retrieve a workspace named *FINANCE* from your own library. In the second example, a password-protected workspace named *GAMES* is retrieved from project library 57. The final example indicates how *GAMES* can be retrieved and an explicit size set for the active workspace.

```
      )LOAD FINANCE
SAVED 11:15:27 07/06/77

      )LOAD 57 GAMES:FUN
SAVED 11:10:20 12/12/77
WSSIZE IS 152412

      )LOAD 57 GAMES:FUN 150000
SAVED 18:16:55 04/08/77
```

Figure 12. Using the)LOAD Command

If you specify a size in the)*LOAD* command that is larger than the maximum workspace size authorized to you, the command is rejected. If you don't specify a size, but the workspace you specify is larger than the maximum workspace size authorized to you, the system attempts to pare the workspace down to the allowable limit. If all the data in the workspace cannot fit into this smaller area, the)*LOAD* command is rejected. If you don't specify a size and the workspace size is less than the maximum workspace size authorized to you, it is expanded to the maximum size. If you specify a size that is insufficient to contain the workspace to be loaded, the)*LOAD* command is rejected.

Copying Objects into the Active Workspace: The)COPY and)PCOPY Commands

The system commands)*COPY* and)*PCOPY* are used to copy one or more objects from a stored workspace into your active workspace.

)*COPY* and)*PCOPY* differ in the way they handle a copied object when another object of the same name appears in the active workspace. The)*COPY* command replaces an existing object, while the)*PCOPY* command does not make the replacement.

The *)COPY* and *)PCOPY* commands are specified as follows:

```
        )COPY  wsid:password  objects

        )PCOPY  wsid:password  objects
```

where:

wsid
is the workspace identification of the workspace to be copied from.

:
is an optional separator, required only when a password is specified.

password
is required only when the workspace to be copied from resides in a project library; the password is the read password, if any, associated with the virtual disk on which the library resides. If the library is not a project library, the password is ignored. The system will prompt for the correct password if it is not specified when required.

objects
are the names of objects to be copied. Each name must be separated by at least one space. If no objects are listed, all global objects in the stored workspace are assumed.

Figure 13 shows how you can use the *)COPY* and *)PCOPY* commands to copy various objects from *STORED*, a workspace stored in your library, into *ACTIVE*, your active workspace. The upper portion of Figure 13 shows the contents of *STORED* and *ACTIVE* before copying. The names within braces are members of the group named *GROUP*. If you want to copy the objects in *STORED* named *A* and *GROUP* without protecting objects in *ACTIVE* from replacement, the command you issue is:

```
        )COPY STORED A GROUP
```

The system responds with the date and time *STORED* was last saved.

If you want to copy *A* and *GROUP* into *ACTIVE* but you also want to protect objects in *ACTIVE* from replacement, the command you issue is:

```
        )PCOPY STORED A GROUP
```

The middle portion of Figure 13 illustrates the contents of *ACTIVE* after execution of the *)COPY* command. Notice that the definition of *GROUP* in *STORED* has replaced the definition of *GROUP* in *ACTIVE*. Notice, too, that the values of *A* and *Z* are now the values of *A* and *Z* in *STORED* and that the groupname *C* is copied into *ACTIVE*. When a member of a copied group is a groupname, only its definition is brought into the active workspace, not its members.

The lower portion of Figure 13 illustrates the contents of *ACTIVE* after execution of the *)PCOPY* command. Notice that the definition of *GROUP* and the objects *A* and *Z* are protected from replacement. The objects *B* and *C* are copied in *ACTIVE*.

To copy all the objects from a stored workspace into the active workspace, you don't specify any object names. For example, the following command copies without protection all the objects from the workspace *OTHER* in library 6065:

```
        )COPY 6065 OTHER
```

STORED and *ACTIVE* Before Copying:

```
          STORED                        ACTIVE

NAMES          VALUES         NAMES          VALUES

A              5              A              10
GROUP⎧ B       20.2      ⎫    GROUP⎧ X       15 ⎫
     ⎨ C       groupname ⎬         ⎨ Y       25 ⎬
     ⎩ Z       37.7      ⎭         ⎩ Z       35 ⎭
```

ACTIVE After Issuing the Command *)COPY STORED A GROUP*:

```
          ACTIVE

NAMES          VALUES

A              5
X              15
Y              25
GROUP⎧ Z       37.7      ⎫
     ⎨ B       20.2      ⎬
     ⎩ C       groupname ⎭
```

ACTIVE After Issuing the Command *)PCOPY STORED A GROUP*:

```
          ACTIVE

NAMES          VALUES

A              10
GROUP⎧ X       15 ⎫
     ⎨ Y       25 ⎬
     ⎩ Z       35 ⎭
B              20.2
C              groupname
```

Figure 13. Using the)COPY and)PCOPY Commands

The following command copies all the objects in *OTHER* while protecting objects in the active workspace from replacement:

```
        )PCOPY 6065 OTHER
```

When you copy objects from a stored workspace you should be aware of the following:

- Shared variables that are replaced by copied objects are retracted.
- If there is not enough room in your workspace, some of the objects may not be copied.
- If copying creates more symbols than can be accommodated in the symbol table, some of the objects may not be copied.
- Any attempt to copy an object with the same name as a halted function (a function that has not completed execution) will cause the message *SI DAMAGE* to be printed.

Grouping Items Together: The)GROUP Command

You may find it convenient at times to group the names of related items together in your workspace. You can form a group (or redefine or extend one you have already created) through the following system command:

```
        )GROUP  groupname  names
```

where:

groupname
: is the name to be associated with a newly-created group or the name of an existing group to be modified.

names
: is an optional list of names to be associated with *groupname*. If specified, each name listed must be separated from a preceding name by at least one space.

The)*GROUP* command can be used in three ways:

- If you want to create a new group, you specify)*GROUP groupname names*. The names in *names* are grouped together under the name *groupname*. If a group named *groupname* already exists, the new group definition replaces any previous group definition.
- If you want to add one or more names to an existing group, you specify)*GROUP groupname groupname names*. The names in *names* are added to the definition of *groupname*.
- If you want to disperse a group, that is disassociate a groupname from a list of names, you specify)*GROUP groupname*.

Suppose, for example, you have three weather forecasting functions *PRECIP*, *BAROM*, and *TEMP*. At your option, you can group these names under a groupname *WEATHER* as follows:

```
        )GROUP WEATHER PRECIP BAROM TEMP
```

If you want to add the names *MEAN* and *AVG* to the existing definition of *WEATHER*, you specify:

```
        )GROUP WEATHER WEATHER MEAN AVG
```

When you want to disperse the group *WEATHER*, you simply specify:

```
        )GROUP WEATHER
```

The names you choose as a groupname cannot already be in use as a global variable or as a defined function name. If the name is already in use in this way, any attempt to specify the name as a groupname, is rejected.

Clearing the Active Workspace: The)CLEAR Command

The)*CLEAR* command is used to discard all the objects contained in your active workspace. The command is specified as follows:

```
        )CLEAR  size
```

where:

size
: is an optional size specification that is used to control the size of the cleared active workspace.

When you specify `)CLEAR`, all the objects in the active workspace are erased, any shared variables are retracted, and all the attributes of the active workspace are set to the standard values shown in Figure 9 in the section "Workspaces and Libraries." In particular, the size of the active workspace is set to a default value.

When you specify `)CLEAR` *size,* your active workspace is cleared and its size is changed to a value indicated by *size.* The system adds a certain amount of space to your size specification for its own needs. If the size you specify is larger than the maximum allowable size, the command is rejected. (You can determine this maximum by issuing the `)QUOTA` command.)

In response to the command `)CLEAR`, the system returns the message `CLEAR WS`. This is followed with the message `WSSIZE IS` *size,* if clearing the active workspace changes its size. The value *size* is the current size of the active workspace. If the `)CLEAR` command is specified with a size operand, the system returns the message `CLEAR WS`, only.

Figure 14 illustrates how the active workspace can be cleared. In the first example shown, no workspace size is explicitly specified. In the second example shown, the workspace size is explicitly specified.

```
      )CLEAR
CLEAR WS
WSSIZE IS 73000

      )CLEAR 100000
CLEAR WS
```

Figure 14. Using the)CLEAR Command

Erasing Objects in the Active Workspace: The)ERASE Command

You can erase any global variable, globally defined function, or group in your active workspace with the command:

```
      )ERASE objects
```

where:

objects
: are the names of one or more objects (global variables, defined functions, or groups) to be erased; each indicated object is separated from another by at least one space.

Suppose, for example, your active workspace contained a global variable *A*, a function *MATH*, and a group named *GRP1*, these objects would be erased by the specification:

```
      )ERASE A MATH GRP1
```

When a group is erased all the objects named in the group are erased with it. If one of the objects in a group is, in turn, the name of a group, its definition is erased but not its members. To erase a group without erasing its members, use the `)GROUP` command.

When you use the `)ERASE` command, be aware of the following:

- If you erase an object that has been offered for sharing, the share offer for the object is retracted.

- If you erase a suspended function (a function that has not completed execution) the execution of the suspended function or any function awaiting the suspended function's completion cannot be resumed.

Identifying the Active Workspace: The)WSID Command

You can assign an identification to your active workspace as follows:

```
        )WSID  wsid:password
```

where:

wsid
is the workspace identification to be assigned to the active workspace.

:
is an optional separator, required only when a password is specified.

password
is an optional password to be associated with the active workspace. If the workspace is to be later saved in a project library, this password must match the write password, if any, associated with the virtual disk on which the library resides.

When you specify the)*WSID* command, any current identification that your active workspace has, is replaced by the new identification. The system acknowledges the identification change by displaying the message:

WAS *wsid*

where *wsid* is *CLEAR WS* if the workspace had no identification, or is the previous identification that the workspace had.

If *password* is specified, the new password overrides any password associated with the active workspace. Otherwise, any current password is erased.

In the following example, a new identification and password are associated with the active workspace.

```
      )WSID REPORTS:UPDATE
WAS 123456 REPORTS
```

Controlling the Symbol Table: The)SYMBOLS Command

All the names that occur in your active workspace are maintained in an area of the workspace called the symbol table. In a clear workspace the maximum number of entries permitted in the table is 256. This includes entries for function and variable names, labels, names used in function definition, group names, and names appearing in a group list.

If you want to change the number of symbols permitted in the symbol table, you can do so only while your active workspace is clear. In that case, you can use the following system command:

```
        )SYMBOLS  number
```

where:

number
is the number of entries you want accommodated in the symbol table. The system may increase this number slightly for its own convenience. The *number* you specify must be between 10 and 8165, and the stack number plus twice the symbols number must not exceed 16384.

In response to the *)SYMBOLS number* command, the system returns the message:

WAS number

where *number* is the former number of entries permitted.

For instance, in a clear workspace you can change the number of symbols permitted as follows:

```
        )SYMBOLS 512
WAS 256
```

If your workspace is not clear, any attempt to assign a new size for the symbol table is rejected.

The maximum value acceptable in the *)SYMBOLS* command also depends on the amount of unused area in your workspace. The larger you permit the symbol table to be, the smaller you permit the dynamic storage area to be. If you indicate a value in *)SYMBOLS* so large that it doesn't leave enough room for a usable dynamic storage area, the command will be rejected.

Controlling the Execution Control Area: The)STACK Command

The execution control area is an area in your workspace that contains information that is generated during function execution and editing, and contains the state indicator, which tracks the progress of executing functions.

In a clear workspace, the number of entries that can be reserved for temporary names and the state indicator is 512. However, you have the option of directly specifying how many entries you want accommodated in the execution control area as follows:

```
        )STACK number
```

where:

number
: is the number of entries you want accommodated in the execution control area. The system may increase this number slightly for its own convenience. The *number* must be between 2 and 16312, and the stack number plus twice the symbols number must not exceed 16384.

In response to *)STACK number*, VS APL returns:

WAS number

where *number* is the previous number of entries permitted.

For instance, you can enlarge the execution control area of a workspace to 650 entries from 500 entries by specifying:

```
        )STACK 650
WAS 500
```

The maximum value acceptable in the *)STACK* command also depends on the amount of unused area in your workspace. The larger you permit the execution control area to be, the smaller you permit the dynamic storage area to be. If you indicate a value in *)STACK* so large that it doesn't leave enough room for a usable dynamic storage area, the command will be rejected.

Similarly, if you specify a value for the execution control area too small to contain the names generated by executing functions, the command will be rejected.

Inquiry Commands

Inquiry commands are used to display the current state of the active workspace, display the contents of libraries, and to indicate the availability of space in libraries and in the active workspace. Inquiry commands are also used to indicate the workspace, library, and shared variable quotas. The inquiry commands are:

-)*WSID*, which displays the identification of the active workspace.
-)*SYMBOLS*, which displays the number of entries currently in the symbol table and the maximum number of entries permitted.
-)*STACK*, which displays the number of entries currently permitted in the execution control area.
-)*WSSIZE*, which displays the size of the active workspace.
-)*LIB*, which lists the names of the workspaces in a designated library.
-)*QUOTA*, which lists the values in the user profile established for maximum and default workspace sizes, total library space, number of shared variables and shared variable size. This command also displays the remaining library space available for use.
-)*FNS*, which lists the names of the global defined functions in the active workspace.
-)*VARS*, which lists the names of the global variables in the active workspace.
-)*GRPS*, which lists the names of the groups in the active workspace.
-)*GRP*, which lists the members of a designated group.
-)*SI*, which displays the contents of the state indicator.
-)*SINL*, which displays the contents of the state indicator with local names.

Listing the Identification of the Active Workspace: The)WSID Command

At any time during your session with VS APL, you may request a display of the active workspace identification, by issuing the system command:

```
        )WSID
```

In return, VS APL displays the workspace identification currently associated with the active workspace, or the message:

```
IS CLEAR WS
```

if your active workspace has no identification.

Figure 15 indicates how a workspace identification is listed. In the first example, a workspace is given an identification of 1484 *SECRET*, which is then displayed via the)*WSID* command. In the second example, a workspace is saved with a new identification. When the workspace identification is that of a private workspace, no library number is displayed.

Assign an identification:

```
      )WSID 1484 SECRET
WAS CLEAR WS
```

What is the identification?

```
      )WSID
IS 1484 SECRET
      )SAVE UPDATE:WRITEPW
13:17:33 08/14/77
      )WSID
IS UPDATE
```

Figure 15. Using the)WSID Command

Monitoring the Symbol Table: The)SYMBOLS Command

If at some time during your work, you're not sure of the status of the symbol table, you can specify the following:

```
      )SYMBOLS
```

In reply, VS APL displays the message:

IS max; number IN USE

where *max* is the current number of entries permitted in the symbol table, and *number* is the number of entries that have been made so far. In a clear workspace *max* is 256 and *number* is 0.

For instance:

```
      )CLEAR
CLEAR WS
      )SYMBOLS
IS 256; 0 IN USE
      A←57
      )SYMBOLS
IS 256; 1 IN USE
      )CLEAR
CLEAR WS
      )SYMBOLS 350
WAS 256
      )GROUP ALPHA A B C D E
      )SYMBOLS
IS 350; 6 IN USE
```

Monitoring the Execution Control Area: The)STACK Command

If you issue the command:

```
      )STACK
```

with no number following it, VS APL will display the number of entries currently permitted in the execution control area. The value is specified as follows:

IS number

where *number* is the number of entries currently permitted.

In a clear workspace the value returned is 512:

```
      )CLEAR
CLEAR WS
      )STACK
IS 512
```

Listing the Workspace Size: The)WSSIZE Command

If you want, you can determine the size of your active workspace by issuing the command:

```
      )WSSIZE
```

For example:

```
      )WSSIZE
176988
```

In a clear workspace, if not otherwise specified, the value returned by `)WSSIZE` is dependent on the size of your virtual computer.

Listing the Workspaces in a Library: The)LIB Command

The `)LIB` command is used to generate an alphabetized list of workspaces in a given library. The format of the command is:

`)LIB` *libnum* *letters*:*password*

where:

libnum
is an optional library number, required only if the library is not your private library.

letters
is an optional series of letters, required when the list of workspace names is to alphabetically follow the indicated letters.

:
is an optional separator, required only when a password is indicated.

password
is for project libraries only and is the read password, if any, associated with the virtual disk on which the library resides. If the library is not a project library, the password is ignored. The system will prompt for the correct password if it is not specified when required.

Suppose, for instance, the following workspaces were in your library:

```
      INDEX
      FINANCE
      FACTORS
      ACCOUNT
```

If you specify `)LIB`, the list generated would be:

```
ACCOUNT
FACTORS
FINANCE
INDEX
```

If however, you specify:

```
        )LIB FI
```

the list would include only those workspaces whose names begin with *FI* or follow alphabetically:

```
        )LIB FI
FINANCE
INDEX
```

If the library to be displayed is a project library and the library resides on a virutal disk that is protected by a read password, the password must be specified in the *)LIB* command. For instance, the following command requests a display of project library 4065 residing on a password-protected virtual disk; the disk has a read password of X14C:

```
        )LIB 4065:X14C
```

Listing Workspace, Library, and Shared Variable Quotas: The)QUOTA Command

The *)QUOTA* command is used to list information about library, workspace, and shared variable availability. The command is specified as follows:

```
        )QUOTA
```

In response, the following message is displayed:

```
LIB  libquota   FREE  remaining
WS  default     MAX  max
SV  number      SIZE  size
```

where:

libquota
: is the total amount of space that you may use in your private library in bytes.

remaining
: is the remaining library space (in bytes) that you may use in libraries.

default
: is the default size of the active workspace in bytes.

max
: is the maximum workspace size you can request in bytes. Under CMS, this is always the same as the default, which varies with the virtual machine size, shared variable quota, shared memory size, and auxiliary processor size.

number
: is the maximum number of variables that can be shared. Under CMS, this number is 8 + (8 × number of auxiliary processors currently active), or 0 if no auxiliary processors are active.

size
: is the size of the user's shared memory. Under CMS, the maximum shared variable size is about 500 bytes less than the value shown.

A typical request for current library, workspace, and shared variable availability, might look like this:

```
          )QUOTA
LIB        500000  FREE      378000
WS         176988  MAX       176988
SV             48  SIZE        4096
```

Listing the Defined Functions in the Active Workspace: The)FNS Command

You can list the names of the globally defined functions appearing in your workspace by issuing the system command:

```
          )FNS  letters
```

where:

letters
: is a series of letters, required only when the list of function names is to alphabetically follow the indicated letters.

In response, VS APL displays all the functions in the active workspace in alphabetical order.

For example, if four functions, *MATH*, *GEOG*, *HIST*, and *LANG* were defined in your workspace, the command:

```
          )FNS
```

would return:

```
GEOG HIST LANG MATH
```

If you follow *)FNS* with one or more letters, VS APL displays only those functions whose names follow those letters in alphabetical order:

```
          )FNS L
LANG MATH

          )FNS GI
HIST LANG MATH

          )FNS P
```
(nothing is displayed)

Listing the Variables in the Active Workspace: The)VARS Command

At any time during your terminal session, you can determine the global variables that appear in your workspace, as follows:

```
          )VARS  letters
```

where:

letters
: is a series of letters, required only when the list of variable names is to alphabetically follow the indicated letters.

VS APL responds, by listing in alphabetical order, all the global variables in the active workspace; no local variables are listed.

Suppose, for example, your workspace contained the global variables, *AMT*, *VALUE*, *COST*, and *RATE*, then the command:

```
        )VARS
```

would return:

```
AMT COST RATE VALUE
```

If *)VARS* is followed by one or more letters, VS APL displays only the functions whose names follow those letters in alphabetical order:

```
        )VARS V
VALUE
        )VARS RAT
RATE VALUE
        )VARS RAV
VALUE
```

Listing the Groups in the Active Workspace: The)GRPS Command

The system command *)GRPS* lists the names of all the groups defined in your workspace in alphabetical order. The command is specified as follows:

```
        )GRPS  letters
```

where:

letters
  is a series of letters, required only when the list of groupnames is to alphabetically follow the indicated letters.

For example, in a workspace containing the groups *FINFNS*, *ACCTFNS*, and *TEMPVARS*, a request to list the groups could be stated as follows:

```
        )GRPS
ACCTFNS FINFNS TEMPVARS
```

Like *)FNS* and *)VARS*, the *)GRPS* command can be optionally followed by one or more letters which control the group names that are printed. VS APL displays only those group names that alphabetically follow the letters listed in the *)GRPS* command:

```
        )GRPS TEMP
TEMPVARS
        )GRPS F
FINFNS TEMPVARS
```

Listing the Members of a Group: The)GRP Command

Suppose you have a group in your workspace, but are not sure of its members. The way you can receive this information is by issuing the system command:

```
        )GRP  groupname
```

where:

groupname
  is the name of the group you are interested in.

In response, the system displays the members of the group you specify. There is no response if the group does not exist in your active workspace:

```
      )GRP FINANCE
GROSS NET
```

The group named *FINANCE* is defined with members *GROSS* and *NET*.

```
      )GRP MASTER
```
(nothing displayed)

The indicated group named *MASTER* does not exist in the workspace.

Displaying the State Indicator: The)SI and)SINL Commands

One of the items maintained in your active workspace is a state indicator which contains information on the progress of defined function execution. A display of the state indicator lists the names of halted functions in order, starting with the most recently halted first. The list may also include the symbols ⎕ or ⍎ if operations pertaining to these symbols are pending. For each function listed, the state indicator shows the line on which work was halted.

With this information available, you can either:

- Resume your work by entering → *n*, where *n* is a line number in the most recently halted function.
- Terminate all currently halted functions by entering → for each *suspended* function in the state indicator list. A suspended function is one that has not completed execution because of some error of its own or because you explicitly halted the function while it was executing.

A request to display the state indicator may be specified in two ways:

```
      )SI
      )SINL
```

If)*SINL* is specified, VS APL displays the state indicator with any names local to each function.

If no functions are currently halted,)*SI* or)*SINL* returns nothing.

For example, a typical use of the)*SI* command is shown below:

```
      )SI
FN3[4]  *
FN2[5]
FN1[2]
```

In this example, the function *FN*1 called the function *FN*2 which called the function *FN*3. The * after *FN*3 indicates that it is suspended. The functions *FN*2 and *FN*1 are *pendent*, that is, they are awaiting the completion of *FN*3 before completing their execution. The bracketed numbers following each function name indicate the statement to be executed in the function.

If the)*SINL* command were issued in the same situation, the results might appear as follows:

```
      )SINL
FN3[4]    *  A
FN2[5]       LABEL
FN1[2]
```

In this example, the name *A* is local to suspended function *FN*3, the name *LABEL* is local to the pendent function *FN*2, and the pendent function *FN*1 has no names local to it.

If the name of a function appears in the state indicator list, you should not attempt to erase it using the)*ERASE* command or attempt to replace it using a)*COPY* command. If you erase or copy a halted function, you make it impossible to resume its execution. A function that cannot be resumed is a *damaged* function.

A damaged function is displayed in the state indicator with a bracketed line number of ¯1:

```
      )SI
FN3[4] *
FN2[5]
FN1[2]
      )ERASE FN3
SI DAMAGE
      )SI
FN3[¯1] *
FN2[5]
FN1[2]
```

Sometimes editing a function in the state indicator list will also damage the function. If the function is suspended, damage will result if:

- The function header is modified.
- The order of its labels is changed.
- Labels are added or deleted.

If the function is pendent, damage to the function will result if any line is modified, moved, deleted, or inserted.

Communication Commands

Communication commands are used to send messages to other users of the system and to the VM/370 operator. The communication commands are:

-)*MSG*, which sends a message to another user.
-)*OPR*, which sends a message to the VM/370 operator.
-)*MSG OFF*, which suspends the reception of messages.
-)*MSG ON*, which restores the reception of messages.

Sending Messages: The)MSG and)OPR Commands

During your session at the terminal you may want to communicate with other users of the system or with the VM/370 operator.

You can send a message to another user by specifying the following command:

```
        )MSG  userid  message
```

where:

userid
is the VM user identification of the user you are directing the message to.

message
is a line of text to be transmitted. As many characters may be entered as will fit on the remainder of the line.

You can send a message to the VM/370 operator by specifying the following command:

```
        )OPR  message
```

where:

message
is a line of text to be transmitted. As many characters may be entered as will fit on the remainder of the input line.

As soon as you enter the)*MSG* or)*OPR* commands, your terminal will prevent you from making further entries, so that it can print any response that is sent to you.

The next thing you should see at your terminal is the system response:

```
SENT
```

This means that the system has posted your message for eventual delivery to its intended recipient. It does not necessarily mean that transmission has begun.

Once your terminal prevents you from making entries, it remains so until you enter a weak interrupt signal (by pressing the ATTN key once).

A message can be received by you any time your terminal is set to display output. In fact, you can receive a message during the execution of a function or between the displayed rows of an array.

When a message is displayed, it is preceded by the time the message was sent and the account number of the sender.

Figure 16 illustrates a typical exchange between two users of VS APL under CMS. The first user (123456) initiates communication by sending a message to user 713572. In response to his)*MSG* command, the sender's keyboard locks and the system displays *SENT*. Meanwhile, user 713572 receives the message preceded by a time indicator and sends a reply. In response to his reply, his keyboard locks and *SENT* is displayed. User 713572 now unlocks his keyboard by entering a weak interrupt signal.

After user 123456 receives the reply, he too unlocks his keyboard and continues his work.

User 123456

```
        )MSG 713572 YOUR PRINTING PRECISION?
SENT
15:53:25
MSG FROM 713572: I'M USING A PP OF 10
Ⓐ
```

User 713572

```
15:53:07
MSG FROM 123456: YOUR PRINTING PRECISION?
        )MSG 123456 I'M USING A PP OF 10
SENT
Ⓐ
```

Figure 16. Sending Messages

If for some reason, your message cannot be posted, the *SENT* message will not be displayed. You can then discard the message by entering a weak interrupt signal. You will then receive the report:

```
MESSAGE LOST
```

You can now try to resend your message if you want.

When you send a message to another user or to the VM/370 operator, he must be signed on to the system. If not, you will receive the report:

```
USER NOT LOGGED ON
```

You should try again later, or, if you have made a mistake in specifying the user's identification, you should try again now, with a corrected user identification.

If a user has blocked messages, or is in a VM/370 *disconnected* state, and you attempt to send a message to him, you will receive the report:

```
USER NOT RECEIVING
```

and your message will not be transmitted.

Blocking Messages: The)MSG OFF Command

If you do not want to be disturbed by incoming messages, you can tell the system to block them from reaching your terminal. You can do this by issuing the command:

```
        )MSG OFF
```

Once you issue this command, no user may direct a message to your terminal. If a user attempts to send a message your way, the report:

```
USER NOT RECEIVING
```

will be printed at his terminal and the message will not be transmitted.

If you need to communicate briefly with another user or with the VM/370 operator, you may issue the *)MSG* or *)OPR* commands. These commands will temporarily restore message acceptance at your terminal. While your terminal is locked in response to either command, you can receive messages from other users or the VM/370 operator. However, once you enter a weak interrupt signal, message blocking will be restored.

It is important to note that message blocking remains in effect for your entire terminal session and not simply for your session with VS APL. If you sign off of VS APL and maintain contact with CMS, message blocking will continue for your terminal.

Restoring Message Acceptance: The)MSG ON Command

If you want to restore message acceptance permanently after message blocking, you can issue the command:

```
        )MSG ON
```

No direct reply is made by the system.

Sign-Off Commands

Sign-off commands are used to end a session with VS APL, CMS, and VM/370 or to end the VS APL session only and remain under control of CMS. There are four commands in this category. Two of the commands save a copy of the active workspace as the session is ended while the remaining two discard the active workspace as the session is ended. The sign-off commands are:

- `)OFF`, which ends the session with VS APL, CMS and VM/370; the active workspace is lost.
- `)OFF HOLD`, which ends the session with VS APL only; the active workspace is lost.
- `)CONTINUE`, which ends the session with VS APL, CMS and VM/370; the active workspace is saved.
- `)CONTINUE HOLD`, which ends the session with VS APL only; the active workspace is saved.

For a detailed description of each of the sign-off commands, refer to "Ending the Work Session" in the section "The Work Session."

AUXILIARY PROCESSING

An auxiliary processor is a program that enables you to perform special host-dependent operations through shared variables. For instance, an auxiliary processor gives you the opportunity of creating and referencing files outside of your workspace. This chapter assumes that you are familiar with shared variables, as discussed in the *APL Language* manual.

Five auxiliary processors are available with the VS APL Program Product as follows:

Name	**Description**
APL100	CP/CMS Command Processor. This processor is used to enter CP or CMS commands.
APL101	Stack Input Processor. This processor is used to store data to be used at the next input request.
APL110	CMS Disk I/O Processor. This processor is used to read from or write to disk files under control of the CMS file system.
APL111	FILEDEF I/O Processor. This processor is used to read from or write to VM/370 devices supported by the Queued Sequential Access Method (QSAM).
APL123	CMS VSAM Processor. This processor is used to perform file operations on entry-sequenced or key-sequenced VSAM files.

To use an auxiliary processor (if your installation has not built it into the APL system), its name must first be specified in the APL command that initially establishes contact with VS APL. For a description of the APL command, see "Step 4: Contacting VS APL" in the section "The Work Session."

In addition to the auxiliary processors available with VS APL, you can write additional auxiliary processors to handle other operations. For a description of how you can create your own auxiliary processors, refer to *VS APL for CMS: Writing Auxiliary Processors.*

Communicating With an Auxiliary Processor

The following discussions describe general information for all the VS APL auxiliary processors except the CMS VSAM Processor. If you are using the CSM VSAM Processor, refer to the discussion "The CMS VSAM Processor" later in this section for general as well as detailed information.

There are generally five steps in communicating with one of the auxiliary processors distributed with VS APL:

1. Initialize a variable with information needed by the auxiliary processor.
2. Offer the initialized variable for sharing with the auxiliary processor.
3. Check the variable for a return code.
4. Send and/or retrieve information via the shared variable.
5. End the procedure by retracting the variable.

VS APL includes a public library workspace *APFNS* which contains, among other items, functions that perform the steps indicated for communication with an auxiliary processor. For further information on *APFNS*, see "VS APL Functions for Auxiliary Processing" later in this section.

Initializing a Variable

Before an auxiliary processor can begin operating, there are certain items of information it must have. For instance, the type of data conversion you need performed or the identification of a file you want to access. You communicate this kind of information through the following initializing assignment:

```
        x←'argument  (options)'
```

where:

x
: is the variable to be shared with the auxiliary processor.

argument
: indicates the source or destination of subsequent operations, for instance a filename to indicate the destination of an output operation.

(
: indicates options are to follow; it should be indicated only when options are specified.

options
: indicate various processing options, for instance data conversion or stacking order. If an option is specified more than once, the rightmost specification applies.

)
: is an optional delimiter that may be omitted if desired.

For example, the following assignment indicates a destination of CMS and that data is to be translated into an internal APL format:

```
        STACK←'CMS (APL'
```

Certain input/output operations require two variables to be initialized. The first variable is generally used to transmit data and is called a *record variable.* The second variable is used to control or monitor data transmission and is called a *control variable.*

Offering a Variable for Sharing

An offer to share a variable with an auxiliary processor is made as follows:

```
        num ⎕SVO 'var'
```

where:

num
: is the number of the auxiliary processor. This number is 100 for the CP/CMS Command Processor, 101 for the Stack Input Processor, 110 for the CMS Disk I/O Processor and 111 for the FILEDEF I/O Processor.

⎕SVO
: is a system function used to issue shared variable offers.

'*var*'
: is the (surrogate) name, in quotes, of the variable or variables to be shared. When more than one variable is to be shared, each (surrogate) name can be specified as a row in a character array. You can share up to 14 variables with each auxiliary processor.

When a variable is offered for sharing, the system responds with a status indication of the sharing operation. This status indicator is called the *degree of coupling* and consists of the numbers:

- 0 if the share offer has not been made
- 1 if the share offer has been made
- 2 if the share offer has been matched by the auxiliary processor, that is sharing is complete

If more than one variable is offered, the degree of coupling is a vector with one element for each variable offered.

As an example of a shared variable offer, the following expression offers two variables *X* and *CTL* to auxiliary processor 110:

```
      110 ⎕SVO 2 3⍴'X  CTL'
2 2
```

The response 2 2 indicates that sharing for both variables has been completed.

Checking for a Return Code

Once a variable has been offered and matched, the auxiliary processor inspects the argument and options specified in the variable. In response, the auxiliary processor assigns the following *return code* to the variable:

- 0 or a vector whose first element is 0—the initialization values are acceptable
- 1—the initialization values are not acceptable

If the return code is 1, correct the error by assigning a valid initial value to the variable.

Sending or Retrieving Information through the Shared Variable

Once a shared variable has been referenced for a return code, any subsequent reference of the variable retrieves information from the auxiliary processor. Any setting of the variable sends data, commands, or control information to the auxiliary processor.

For example, once a variable *COMMAND* has been shared with the CP/CMS Command Processor, it can be used to issue the CP command QUERY by specifying:

```
      COMMAND←'QUERY USERS'
```

Once a record variable, *REC* has been shared with the FILEDEF I/O Processor and checked for a return code, the next reference to *REC* reads a data record from the file:

```
      REC
```

(a record is returned)

Ending the Procedure

To end communication with an auxiliary processor, the shared variables are retracted. This is done through the following system function:

```
        ⎕SVR  'var'
```

where:

'var'
is the name in quotes of the shared variable to be retracted. Once a variable is retracted, it may be reinitialized and reoffered.

The system responds with the degree of coupling that the variable had prior to retraction.

The CP/CMS Command Processor

The CP/CMS Command Processor, auxiliary processor 100, is used to dynamically establish contact with the control program segment of VM/370 or CMS so that you can enter CP or CMS commands.

Once CP is contacted, any command acceptable to the control program segment of VM/370 can be issued. Once CMS is contacted, any CMS command can be entered. Note, however, that entry of any CMS command that requires program storage may cause an abrupt termination of the auxiliary processor or VS APL. CMS commands that require program storage include LOAD, LOADMOD, and START.

Output from CP commands may be canceled by signaling attention.

Output from CMS commands may be canceled by signaling attention once, and then entering HT. (If using a 3270, press the PA1 key twice, and then enter HT.) Entering HT cancels CMS and APL output from the current APL function until the next input from the terminal.

Initialization Values: The initialization arguments for communication with the CP/CMS Auxiliary Processor are:

- CP—contact the control program segment
- CMS—contact CMS

If no initial value is specified, or if the initial value is null, a default of CMS is assumed.

Return Codes: If the initial value is not CP, CMS, or null, a code of 1 is returned in the shared variable, otherwise 0 is returned.

Once the initial value is accepted, any subsequent reference of the variable will produce a return code set by the auxiliary processor, CP, or CMS. For further information on these codes, refer to "Auxiliary Processor Return Codes" later in this section.

Example: The following function *COPIES* uses the CP/CMS Command Processor to request multiple copies of any printed output:

```
      ∇ COPIES N;MODE
[1]     MODE←'CP'
[2]     100 ⎕SVO 'MODE'
[3]     MODE←'SPOOL PRT COPY ',⍕N
      ∇
```

In this example, the first statement indicates CP mode; the second statement offers a variable for sharing with auxiliary processor 100. The third statement requests the specified number of copies.

The Stack Input Processor

The stack input processor, auxiliary processor 101, is used to create a stack of data to be used when the system is next ready to accept input.

You build the stack by assigning to a shared variable each line of data as a character vector or scalar. Options specified when the shared variable is initialized indicate how you want the data stacked and what type of conversion you want applied to the stacked data.

If a strong interrupt signal is issued or a character error is detected as the stacked data is being used, the entire contents of the stack is deleted and the terminal opens for input.

It is important to note that the stack input processor will not stack CMS immediate commands like HT (halt terminal output) or RT (resume typing). These commands are executed when you enter them.

Initialization Values: The initialization values for communication with the Stack Input Processor are:

CMS(*order conversion*)

where:

CMS
indicates that data is to be placed on a stack maintained by CMS. If omitted, CMS is assumed.

(
indicates options are to follow; it should be specified only when options are specified.

order
indicates whether the processor places the data at the beginning of the stack (*BEG* or *LIFO*) or at the end (*END* or *FIFO*). The default is *FIFO*.

conversion
indicates the conversion to be applied to the stacked data. The conversion options are:

- 370—limited graphic character translation from APL to EBCDIC. Conversion is performed as indicated in Figure 17. Characters that cannot be translated directly are transmitted as blanks.

APL	EBCDIC
A̲ - Z̲	a - z
~	¬
∧	&
¨	"
α	@
÷	%
≠	$
∆	#
⊖	–0 (equivalent to })
⌽	+0 (equivalent to {)
←	= } only when translating from APL to EBCDIC
¯	– } only when translating from APL to EBCDIC

The following characters are common to APL and EBCDIC and are translated directly:

A through Z, 0 through 9, blank < = > + – . : ; , ? !) (/ \ | __ * ' , the backspace character, the new line character, and the line feed character

Figure 17. APL/EBCDIC Conversion via 370 Conversion Option

- 192—full character translation from APL to EBCDIC. Character mapping is provided for all 256 possible character representations in the EBCDIC character set as indicated in Figure 31 in Appendix G.
- APL—full graphic character translation from APL to EBCDIC using an internal code established by CMS. The characters put into the input stack are those that VS APL would expect if the characters were entered from the user's terminal. In other words, on a typewriter terminal compound characters are expanded to their constituent parts (A̲ becomes A, backspace, __), and on a display terminal compound characters are translated to single EBCDIC codes.

The default is 370.

If no initial value is specified, or the initial value is null, the defaults are used.

Return Codes: If the initial value is invalid, a code of 1 is returned in the shared variable, otherwise a 0 is returned. Once the initial value is accepted, any subsequent reference of the variable will produce a return code set by the auxiliary processor or CMS. For further information on these codes, refer to "Auxiliary Processor Return Codes" later in this section.

Example: Figure 18 illustrates a function *CHECKPOINT* that uses the Stack Input Processor to save a copy of the active workspace and then returns.

In execution, *CHECKPOINT* builds a two line stack. The first line to be executed is the)*SAVE* command, which saves a copy of the active workspace. The second line to be executed is a branch to *LABEL* which resumes execution. Notice that the HT and RT commands are not stacked, instead they are used to prevent typing while the stack is being used.

```
      ∇ CHECKPOINT;S
[1]    ⍝ USE CMS STACK,LAST IN,FIRST OUT
[2]     S←'CMS (APL BEG'
[3]    ⍝ SHARE S AND IGNORE RESULT
[4]     S←101 ⎕SVO 'S'
[5]    ⍝ HALT TERMINAL OUTPUT
[6]     S←'HT'
[7]    ⍝ RESUME EXECUTION
[8]     S←'→LABEL'
[9]    ⍝ SAVE THE ACTIVE WS
[10]    S←')SAVE TEMP'
[11]   ⍝ SET STOP VECTOR
[12]    S∆CHECKPOINT←LABEL
[13]   ⍝ RESUME TYPING WHEN RESTARTED
[14]   LABEL:S←'RT'
      ∇
```

Figure 18. Stack Input Processor Application

The CMS Disk I/O Processor

The CMS Disk I/O Processor, auxiliary processor 110, is used to read or write sequentially or randomly to disk files under control of the CMS file system.

If you want to process a file sequentially, access just one record at a time and not examine return codes, only a record variable is required. Once the initial value has been accepted, you can use the record variable to transmit data to and from the file. The first reference of the record variable reads the first record from the file; each successive reference reads the next sequential record from the file. Each time you set the record variable, a record is appended to the end of the file.

If you want to process a file randomly, access multiple records with one operation, or examine return codes, you must share two variables, a record variable and a control variable. The record variable must be offered first.

Using the Control Variable

The control variable is used to monitor and control each input/output operation. Once the initial return code is checked, each reference of the control variable returns a four-element vector that contains the return code of the previous input/output operation, the position of the next record to be read, the position of the next record to be written, and the blocking factor (the number of records to be accessed with the next operation). The auxiliary processor initially sets the read pointer to 1; the write pointer to n+1, where n is the number of records in the file; and the blocking factor to 1.

You have the option of respecifying the control variable before each input or output operation. If you specify a scalar or a one-element vector, the read pointer may be reset. If you specify a two-element vector, the read and write pointers may be reset. If you specify a three-element vector, both pointers and the blocking factor may be reset. If you specify a four-element vector, the last three elements are used to reset both pointers and the blocking factor. A subscripted assignment to the control variable can thus be used to change one or more of the values of the control vector. If you specify a 0 or a negative number as any element, the corresponding value in the control variable remains as is. If you specify an unacceptable value (that is, a non-integer, a vector with more than four elements, or an array), the entire control variable remains as is.

The read pointer can be set to any positive integer. If it exceeds the number of records in the file, an end-of-file condition occurs on the next input operation.

If the write pointer is set to the number of an existing record, that record is replaced by the next output operation. If the file contains variable-length records and you replace an existing record with one of a different length, the remainder of the file is damaged. If the conversion option is VAR, you must ensure that the record has not changed length. The record length can change if the variable being written has changed data type (for example, from logical to real) during execution of VS APL.

If the write pointer is set to one more than the number of existing records, the next output operation appends a record to the end of the file. If the write pointer is set beyond n+1 and the file contains fixed-length records, the next output operation appends one or more empty records followed by the specified record to the end of the file. If the write pointer is set beyond n+1 and the file contains variable-length records, an error (return code 7) occurs on the next output operation.

The blocking factor is always one for a file with variable-length records or when you use conversion option VAR or APL; if you specify a different value, it is ignored. If you specify a blocking factor too large for the auxiliary processor's buffer it is ignored. Note that the acceptance of a large blocking factor by the auxiliary processor does not ensure that a sufficient amount of shared storage will be available when the record variable is actually transmitted. You can decrease the time it takes to access a file with fixed-length records by requesting more than one record per operation. For records shorter than 800 bytes, the optimum blocking factor is the number of records contained in one 800-byte block.

Using the Record Variable

The record variable is used to transmit records between your workspace and the file. Once the initial return code is checked, each time you reference the record variable, one or more records are read from the file. If the conversion option is VAR, one record is transmitted; its shape is as described in the record. If the conversion option is other than VAR and the blocking factor is one, one record is transmitted as a vector. If the blocking factor is greater than one, that number of records are transmitted as a matrix; each row is one record. If the blocking factor is n and there are less than n records remaining, the remaining records are transmitted as a matrix with from 1 to n–1 rows.

When you reference the record variable and end-of-file is reached or a read error occurs, a null vector or matrix is transmitted. If you have shared a control variable, you can differentiate these cases by inspecting the return code in the control variable.

Each time you set the record variable, one or more records are written to the file. If the conversion option is VAR, one record is written; the data may be any shape. If the conversion option is other than VAR and the data is a scalar or vector, one record is written. If the blocking factor is greater than one and the data is a matrix, each row is written as one record; the number of rows cannot exceed the blocking factor.

Initialization Values: The initialization values for communication with the CMS Disk I/O Processor are:

fileid(`CTL` `FIX` *access* *conversion*)

where:

fileid
: specifies the filename, filetype and filemode of the CMS disk file to be accessed. The default filemode is A1. An asterisk (*) can be used as the filemode only for a file to be read. The default filetype is VMAPLcF where c is the first character of the indicated conversion option except for a conversion option of BYTE in which case c is Y.

(
: indicates options are to follow; it should be specified only when options are specified.

`CTL`
: is specified only if this variable is a control variable. If omitted, the variable is established as a record variable. When CTL is specified any additional initialization options are ignored. The control variable must be shared after the record variable is shared; it is associated with the most recently offered record variable for which an identical fileid was specified.

`FIX`
: if a file is to be created, this option indicates that the file will contain fixed length records; the record length is the length of the first record written to the file; all records subsequently written to the file must be this length. If omitted, the file is created with variable length records. If the file already exists, the existing record format is used.

access
: indicates the type of access to be associated with the file. The access options are:

- U—the file can be read or written.
- R—the file can only be read; any specification of the record variable except for its initial value is ignored. If you select this option, do not set the access control vector to control your specifications. If you do, and then specify the variable, the system is deadlocked.
- W—the file can only be written; any reference of the record variable is ignored. If you select this option, do not set the access control vector to control your references. If you do, and then reference the variable, the system is deadlocked.

The default is U.

conversion
: indicates the conversion to be applied to the processed data. The conversion options are:

- 370—limited graphic character translation from APL to EBCDIC. Conversion is performed as indicated in Figure 17. On output, the data part of the variable, which must be character, is written one byte per element; characters that cannot be translated are sent as blanks. On input, the record is transmitted as a character variable, one element per byte, characters that cannot be translated are accepted as the APL character (∘).

- 192—full character translation from APL to EBCDIC. Character mapping is provided for all 256 possible character representations in the EBCDIC character set as indicated in Figure 31 in Appendix G. On output, the data part of the variable, which must be character, is written one byte per element. On input, the record is transmitted as a character variable, one element per byte.
- APL—full graphic character translation from APL to EBCDIC using an internal code established by CMS. Compound characters are expanded to their constituent parts. For example, *A̲* is converted to A, backspace, _ on output and _, backspace, A is converted to *A̲* on input. On output, the data part of the variable, which must be character, is written. On input, the record is transmitted as a character variable.
- VAR—on output, the entire variable including its size, shape and type information is written with no conversion. On input, the entire record is transmitted.
- BIT—on output, the data part of the variable, which may be any data type, is written, one bit per element of data; each element of data must have a value of 0 or 1. On input, the record is transmitted as a logical variable, one element per bit.
- BYTE—on output, the data part of the variable, which must be character, is written with no conversion, one byte per element of data. On input, the record is transmitted as a character variable, one element per byte. The data conversion functions in the public library workspace *APFNS* can be used to convert this information. (See "VS APL Functions for Auxiliary Processing" later in this section.)

The default is VAR except if a filetype of VMAPLcF is specified. In this case, the default is the conversion option whose first letter matches c except if c is Y, in which case the conversion option is BYTE.

)

is an optional delimiter that may be omitted if desired.

Return Codes: If the initial value of the record variable is invalid, a code of 1 is returned in the record variable, otherwise the record variable contains a four-element vector whose first element is 0.

If the initial value of the control variable is invalid, a code of 1 is returned in the control variable, otherwise the control variable contains a 0.

After each read or write operation, the control variable, if any, contains a four-element vector, the first element of which indicates the return code of the previous operation. For further information on these codes, refer to "Auxiliary Processor Return Codes" later in this section.

Example: The following example illustrates how a CMS file containing fixed-length 80-byte records can be read or written randomly.

Initialize the record and control variables:

```
      REC←'OLDFILE(370 FIX'
      CTL←'OLDFILE VMAPL3F(CTL'
```

Offer the variables:

```
      110 ⎕SVO 2 3ρ'RECCTL'
2 2
```

Check for return codes:

```
        REC
0 1 4 1
        CTL
0
```

Check the current status:

```
        CTL
0 1 4 1
```

Write the fourth record:

```
        REC←80↑'TABLES'
```

Check the current status:

```
        CTL
0 1 5 1
```

Read two records starting at the third record:

```
        CTL←3 0 2
        REC
```

(third and fourth records are read)

The FILEDEF I/O Processor

The FILEDEF I/O Processor, auxiliary processor 111, is used to sequentially read from or write to any VM/370 device supported by the Queued Sequential Access Method (QSAM). The file representing the device and its characteristics must first be specified through the CMS command, FILEDEF. The FILEDEF command is described in the *VM/370: CMS Command and Macro Reference.*

All record processing using the FILEDEF I/O Processor is done through a record variable. Once sharing is completed and you check for a return code, the first reference of the record variable reads the first record from the file; each successive reference reads the next record from the file. The first setting of the record variable writes the first record; each successive setting writes the next record to the file.

You may optionally share a control variable. If you share one, the control variable can be used to check return codes for each read or write operation. If you set a value for the control variable, it is ignored.

Initialization Values: The initialization values for communication with the FILEDEF I/O Processor are:

ddname (*CTL* *conversion*)

where:

ddname
is the ddname of the device to be accessed. It must be the ddname defined by a FILEDEF command already issued to CMS.

CTL
is specified only if this variable is a control variable. If omitted, the variable is established as a record variable. If CTL is specified, any specified conversion option is ignored. The control variable must be shared after the

record variable is shared; it is associated with the most recently offered record variable for which an identical ddname was specified.

conversion

indicates the conversion to be applied to the processed data. The conversion options are:

- 370—limited graphic character conversion from APL to EBCDIC. Conversion is performed as indicated in Figure 17. On output, the data part of a character vector or scalar is written one byte per element, characters that cannot be translated are sent as blanks. On input, the record is transmitted as a character vector, one element per byte, characters that cannot be translated are accepted as the APL character (∘).
- 192—full character translation from APL to EBCDIC. Character mapping is provided for all 256 possible character representations in the EBCDIC character set as indicated in Figure 31 in Appendix G. On output, the data part of a character vector or scalar is written one byte per element. On input, the record is transmitted as a character vector, one element per byte.
- APL—full graphic character translation from APL to EBCDIC using an internal code established by CMS. Compound characters are expanded to their constituent parts. For example, $\underline{A}$ is converted to A, backspace, __ on output and __, backspace, A is converted to $\underline{A}$ on input. On output, the data part of a character vector or scalar is written. On input, the record is transmitted as a character vector.
- VAR—on output, the entire variable including its size, shape and type information is written with no conversion. On input, the entire record is transmitted.
- BIT—on output, the data part of a vector or scalar is written one bit per element; each element of data must have a value of 1 or 0. On input, the record is transmitted as logical vector, one element per bit.
- BYTE—on output, the data part of a character vector or scalar is written with no conversion, one byte per element. On input, the record is transmitted as a character vector, one element per byte. The data conversion functions in the public library workspace *APFNS* can be used to convert this information. (For a description of *APFNS*, see "VS APL Functions for Auxiliary Processing" later in this section.)

The default is VAR.

)

is an optional delimiter that may be omitted if desired.

Return Codes: If the initial value of the record variable is invalid, a code of 1 is returned in the record variable, otherwise, the record variable contains a scalar 0.

If the initial value of the control variable is invalid, a code of 1 is returned in the control variable, otherwise the control variable contains 0.

After each read or write operation, the control variable, if any, contains the return code of the previous operation. For further information on these codes, refer to "Auxiliary Processor Return Codes" later in this section.

Example: Figure 19 illustrates function *CTOP* that uses the FILEDEF I/O Processor to read a series of cards from the card reader and print each record on the printer.

The CMS VSAM Processor

The CMS VSAM Processor, auxiliary processor 123, is used to perform file operations on entry-sequenced or key-sequenced VSAM files. The VSAM files must be preallocated, and defined to CMS via a DLBL command before they can be accessed. The DLBL command is described in *VM/370: CMS Command and Macro Reference.*

The auxiliary processor does not convert any data transferred to or from a VSAM file. Each VSAM record brought into the workspace appears as a character vector. Data to be transferred from the workspace to the VSAM file must be in the form of a character vector. The data conversion functions in the public library workspace *APFNS* can be used to simplify conversion of various System/370 data types.

Communicating with the CMS VSAM Processor

Communication with the CMS VSAM Processor is made through a shared variable called the *control variable.* In certain operations, a second shared variable called the *data variable* is also required.

In general, there are four steps in the communications process:

1. A control variable is offered for sharing with the auxiliary processor. If data is to be transferred, a data variable is also offered.
2. The VSAM file is opened for processing.

```
     ∇ CTOP;CMS;SVPRINT;SVREAD;CARD
[1]    ⍝ PREPARE TO ISSUE FILEDEF COMMANDS
[2]     100 ⎕SVO 'CMS'
[3]     CMS←'FILEDEF CTOPOUT PRINTER (RECFM VA BLKSIZE 137'
[4]     CMS←'FILEDEF CTOPIN READER (RECFM F BLKSIZE 80'
[5]    ⍝ INITIALIZE A RECORD VARIABLE FOR PRINTER
[6]     SVPRINT←'CTOPOUT ( 370'
[7]    ⍝ INITIALZE A RECORD VARIABLE FOR CARD READER
[8]     SVREAD←'CTOPIN ( 370'
[9]    ⍝ OFFER RECORD VARIABLES
[10]    111 ⎕SVO 2 7 ρ'SVPRINTSVREAD '
[11]   ⍝ CHECK FOR INITIALIZATION ACCEPTANCE
[12]    →(SVPRINT∨SVREAD)/0
[13]   ⍝ CHECK FOR END OF FILE
[14]   LOOP:→(0=ρCARD←SVREAD)/0
[15]   ⍝ PRINT CARD
[16]    SVPRINT←' ',CARD
[17]    →LOOP
     ∇
```

Figure 19. FILEDEF I/O Processor Application

3. A request for a file operation is assigned to the control variable; any data to be processed is transmitted through the data variable. This step is repeated if additional data is to be transferred.

4. The VSAM file is closed.

Functions are included in the public library workspace *APFNS* to perform these steps. For further information on *APFNS*, see "VS APL Functions for Auxiliary Processing" later in this section.

Offering a Variable for Sharing with the CMS VSAM Processor

An offer to share a variable with the CMS VSAM Processor is made as follows:

```
        123 ⎕SVO 'var'
```

where:

123
  is the number of the CMS VSAM Processor.

'*var*'
  is the (surrogate) name, in quotes, of the variable(s) to be shared. When more than one variable is shared, each (surrogate) name can be specified as a row in a character matrix.

The control variable (surrogate) name must begin with the letters CTL; its total length is limited to eleven characters. The (surrogate) name of the data variable must begin with the letters DAT; it too is limited to eleven characters.

If data is to be transferred in a VSAM file operation, a control and data variable pair must be shared. Ignoring their first three characters, the (surrogate) names of the control and data variable pair must be identical. You can simultaneously open and access multiple VSAM files by sharing multiple control variables or multiple pairs of control and data variables with the auxiliary processor. Up to a maximum of 14 control variables or 7 pairs of control and data variables can be shared. The suffixes of the control and data variable (surrogate) names must be unique for each file which is open simultaneously.

The response to a shared variable offer is a degree of coupling:

- 0 if the offer has not been made
- 1 if the offer has been made
- 2 if the offer has been matched by the auxiliary processor, that is sharing is complete

If more than one variable is offered, the degree of coupling is a vector with one element for each variable offered.

The auxiliary processor also sets an access control vector for the control variable to 1 1 1 1. This means that two successive specifications of the control variable by you require an intervening reference by the auxiliary processor and that two successive references by you require an intervening specification by the auxiliary processor. If this sequence is not adhered to, you may be *interlocked,* that is, your terminal may be locked due to an access control violation. To break the interlock, enter a strong interrupt signal.

The access control vector for the data variable is set to 0 0 0 0 by the auxiliary processor. This setting allows more freedom in specifying and using the data variable.

Opening a VSAM File

A VSAM file is opened by assigning one of the following character vectors to the control variable:

'*OR* *fileid*'	Open the file for reading
'*OW* *fileid*'	Open the file for writing new records or for reading
'*OU* *fileid*'	Open the file for updating (read and update existing records), writing new records, reading, or erasing

The item *fileid* represents the file identification of the VSAM file as follows:

filename : *password*

where:

filename
is the ddname specified for the VSAM data set via a DLBL command during the session.

:
is an optional separator, that is specified only if a password is supplied.

password
is an optional VSAM password associated with the VSAM file. If you try to open a password-protected file but forget to supply the password or supply an incorrect password, the auxiliary processor will reply with an error message. (Passwords are ignored if specified for VSAM files residing on CMS disks.)

Requesting VSAM File Operations

Once a VSAM file has been opened, you can request various file operations to be performed on it. These requests, called *service requests,* are made by assigning an appropriate character scalar or vector to the control variable. If data is to be transferred, it is transmitted through the data variable. When the data is transferred, it may be in one of several System/370 data types such as EBCDIC or binary. To convert data to and from these data types, one of the functions in the public library workspace *APFNS* may be used.

Figure 20 lists the operations that can be performed on a VSAM file, the service request associated with the file operation, and the VSAM file type on which the operation may be performed.

The item *key* in Figure 20 represents the full key of a VSAM key-sequenced record. The key and the colon that precedes it are specified only when an operation is to be performed on a key-sequenced VSAM file. Any blanks between the colon and the ending quote are part of the key.

When a service request is issued, it results in a two element return code in the control variable. If both elements of the return code are zero, the requested operation was successful, otherwise the operation was not successful. Although it is not a requirement, it is good practice to examine the return code in the control variable after issuing each service request. For further information on return codes, refer to "Auxiliary Processing Return Codes" later in this section.

Operation	**Service Request**	**VSAM File Type**
Sequential Read	'*R*'	Entry-sequenced Key-sequenced
Sequential Read for Update	'*RU*'	Entry-sequenced Key-sequenced
Direct Read	'*R*:*key*'	Key-sequenced
Direct Read for Update	'*RU*:*key*'	Key-sequenced
Write a Record	'*W*'	Entry-sequenced Key-sequenced
Erase a Record	'*E*:*key*'	Key-sequenced
Position Record Pointer for Sequential Read	'*PO*[:*key*]'	Key-sequenced

Figure 20. VSAM File Operation Service Requests

The following discussions elaborate on the CMS VSAM service requests and cover items to be aware of when you request VSAM file operations through the CMS VSAM auxiliary processor. In these discussions, the item *cvar* is used to represent the control variable name and the item *key*, as before, represents the full key of a VSAM key-sequenced record. The actual control variable name and record key should be supplied when you issue your service requests.

Read: There are two types of read operations that can be performed through the CMS VSAM Processor: sequential and direct.

A sequential read operation directs the processor to return records in an ordered sequence. Sequential read operations are designed for entry-sequenced or key-sequenced files. Records in an entry-sequenced file are returned in the order they are encountered in the file. Key-sequenced records are returned in ascending order based on the value of a key embedded in each record.

When you read sequentially from a key-sequenced file, you have the option of explicitly indicating the first record (based on key) to be read in sequence. This is further described under "Position Record Pointer" later in this section.

In a direct read operation, there is no ordered sequence. Instead, each record is read from the file based on a record key supplied in the read request. Direct read operations can be performed only on key-sequenced VSAM files.

To read sequentially from a VSAM file, issue the following service request:

```
cvar← 'R'
```

To read directly from a VSAM file, issue the following service request:

```
cvar← 'R:key'
```

For either sequential or direct reading, the file can be open for reading, writing, or updating.

After a sequential or direct read operation is successfully completed, the transferred data is contained in the data variable.

Write: The write operation is designed for entry-sequenced or key-sequenced files. Records are written to an entry-sequenced file in the order they are specified. Key-sequenced records are written such that they can later be retrieved by key value; the key must be appropriately embedded in each record. (For a description of VSAM record keys, refer to the *VSAM Programmer's Guide.*)

To write to a VSAM file, issue the following request:

cvar← '*W*'

The file must first be opened for writing or updating and the data variable must be specified before a write request is issued.

When writing to a VSAM file, be aware of the following:

- When you write to an empty key-sequenced file (one that contained no records when you opened it) the records must be written in ascending key sequence. If the file is re-opened (and not empty) the records do not have to be written in ascending key sequence.
- Any attempt to write over an existing record in a VSAM file will cause an error. This can however be done as part of an update operation.

Update: Updating a record in a VSAM file is a two step process. Once the file is opened for updating, you must:

1. Read for update the record to modified.
2. Write a new record over the existing record.

A VSAM file can be read for updating either sequentially or directly. A sequential read for update can be requested for an entry-sequenced or key-sequenced VSAM file. A direct read for update can be requested for a key-sequenced VSAM file only.

To sequentially read a VSAM file for update, the following service request is issued:

cvar← '*RU*'

To directly read a VSAM file for update, the following service request is issued:

cvar← '*RU :key*'

When either read operation is completed, the transferred data is in the data variable.

To write the updated record, assign the record you want written to the data variable and issue the following service request:

cvar← '*W*'

The data to be written must be specified before the write request is issued.

When you update a VSAM file, be aware of the following:

- If the file to be updated is a key-sequenced file, the key of the new record must have the same value as that of the record to be written over.
- If you update an entry-sequenced file, you can't change the length of the record.

Erase: You may erase a record in a VSAM file only if the file is a key-sequenced file and you have previously opened the file for erasing. In that event, the service request you issue is the following:

cvar← '*E : key*'

Position Record Pointer: VSAM maintains an internal record pointer that points to the next record to be processed in a sequential read operation. You may explicitly position the internal pointer for sequential read operations on key-sequenced files. To position the record pointer, issue the following service request:

```
        cvar← 'PO:key'
```

The key is optional. If it is omitted in the service request, the record pointer is positioned at the beginning of the file.

Closing a VSAM File

Once a VSAM file operation is completed, the file should be closed. A VSAM file is closed by assigning the character 'C' to the control variable or by retracting the control variable.

Examples

The following examples illustrate how to use the CMS VSAM Processor. The examples assume that a DLBL command has been issued for the VSAM file to be accessed.

Example 1

The following example illustrates how you can sequentially read records from a VSAM file. The file is entry-sequenced and has an identification of *VFILE*:A1:

Offer a control and data variable for sharing:

```
      123 ⎕SVO 2 4⍴'CTLXDATX'
2 2
```

Open VFILE for reading:

```
      CTLX←'OR VFILE:A1'
```

Check for a return code:

```
      CTLX
0 0
```

Issue a sequential read request:

```
      CTLX←'R'
      CTLX
0 0
```

Assign the record to an unshared variable; the data is in the data variable:

```
      VALUE←DATX
```

Read the next record and assign to an unshared variable:

```
      CTLX←'R'
      CTLX
0 0
      VALUE←VALUE,DATX
```

Close the file:

```
        CTLX←'C'
        CTLX
0 0
```

At this point you may separate and convert the contents of *VALUE* to an APL format using an appropriate function in the *APFNS* workspace.

Example 2

Figure 21 illustrates a function, *KEYUPDT*, that updates a record in a key-sequenced file. The file identification is in the variable *FILEID*; the record to be updated has an EBCDIC key whose value is in the variable *NUM*; while the new record is in the variable *RECORD*.

```
    ∇ NUM KEYUPDT FILEID;CTLX;DATX
[1]  ⍝ OFFER A CONTROL AND DATA VARIABLE AND CHECK COUPLING
[2]   →(2≠123 ⎕SVO 2 4 ⍴ 'CTLXDATX' )/ERR
[3]  ⍝ OPEN FILE FOR UPDATE AND CHECK RETURN CODE
[4]   CTLX← 'OU ' ,FILEID
[5]   →(0≠CTLX)/ERR
[6]  ⍝ READ KEYED RECORD FOR UPDATE AND CHECK RETURN CODE
[7]  ⍝ FUNCTION ECO IN WS APFNS USED TO CONVERT KEY TO EBCDIC
[8]   CTLX← 'RU:' , ECO NUM
[9]   →(0≠CTLX)/ERR
[10] ⍝ WRITE NEW RECORD AND CHECK RETURN CODE
[11]  DATX←RECORD
[12]  CTLX←'W'
[13]  →(0=CTLX)/CLOSE
[14] ERR:  'KEYUPDT ERROR'
[15] ⍝ CLOSE THE FILE
[16] CLOSE:CTLX←'C'
    ∇
```

Figure 21. A Function That Updates a Key-Sequenced Record

Auxiliary Processor Return Codes

Figure 22 lists and describes the codes that may be returned by the CP/CMS Command Processor, the Stack Input Processor, the CMS Disk I/O Processor, and the FILEDEF I/O Processor. With the CP/CMS Command Processor and the Stack Input Processor, you reference the shared variable to obtain the return code. The input/output auxiliary processors return codes in the control variable.

In addition to the codes listed in Figure 22, other codes can be returned by the CP/CMS Command Processor and the FILEDEF I/O Processor. With the CP/CMS Command Processor, the return code is generally the one from the command that you previously assigned to the shared variable; these codes are described in *VM/370: CMS Command and Macro Reference.* If an I/O error occurs when using the FILEDEF I/O Processor, a code with a decimal value is returned. When this code is converted to its four-byte hexadecimal representation, the first two bytes are the sense bytes and the last two are the status bytes. Sense and status bytes are described in the "Status Information Following an Input/Output Operation" section of *OS/VS1 Data Management Macro Instructions* or *OS/VS2 MVS Data Management Macro Instructions.*

Code	Description
–3	Unknown CMS command.
–2	CMS command cannot be executed.
0	No error exists
1	Attempt to read a nonexistent file or unknown CP command.
3	Permanent read error.
4	First character of filemode is invalid.
5	Attempt to read more records than the maximum allowed by CMS.[1]
6	Attempt to write too many records in a CMS file.[1]
7	Attempt to write past the end of a variable-length file.
8	Attempt to read a record with incorrect record length.
10	Attempt to create a file when you already have the maximum allowed by CMS.[1]
12	End-of-file read or attempt to write on a read-only disk.
13	Attempt to write on a full disk.
14	Attempt to write on an unformatted disk.
15	Attempt to write a record with incorrect length into a file with fixed format.
17	Attempt to write a record that is too large into a variable length file.[1]
19	Attempt to write in a file already containing as many data blocks as CMS will allow.
440	Data set cannot be opened for output.
441	Data set cannot be opened for input.
443	Insufficient free storage for input/output buffers. Note that the amount of free storage allocated is based on the specified shared memory size.[2]
444	The value assigned to the shared variable is invalid. It is the wrong shape, size, or data type.
445	Attempt to read a record larger than shared memory.[2]

[1] See *IBM VM/370: CMS User's Guide* for these limits.

[2] If this code is returned, restart VS APL with more shared memory and try again. If the resultant workspace size is too small restart CMS with more virtual storage.

Figure 22. Processor Return Codes: CP/CMS, Stack Input, CMS Disk I/O, and FILEDEF I/O

Figure 23 lists and describes the codes that may be returned by the CMS VSAM Processor. This processor returns a two-element code in the control variable.

In addition to the codes listed in Figure 23, you may encounter other codes in response to CMS VSAM Processor service requests. These codes are generated by VSAM and have a first element of 4, 8, 12 or 16. For a description of these codes, refer to the *VSAM Programmer's Guide.*

Code	Description
0 0	Operation is successful.
1 12	Syntax error in request.
1 13	VSAM file already open.
1 15	Inappropriate open request issued for this operation.
1 16	Operation is not valid for this type of file.
1 17	Key length error.
1 18	VSAM MODCB error.[1]
1 19	VSAM SHOWCB error.[1]
1 20	A character vector is required.
1 21	Exceeded maximum record length for the file.
1 22	VSAM file is not open.
1 27	VSAM TESTCB error.[1]
1 32	Insufficient free storage for input/output buffers. Note that the amount of free storage allocated is based on the specified shared memory size.[2]
1 33	Data variable has not been specified or offered.
1 34	Incorrect password.
1 42	Data variable has not been referenced.
1 45	VSAM GENCB error.[1]
1 46	Shared memory full.[2]
1 47	Unrecognized VSAM file type.

[1] This error is caused by the auxiliary processor and should be reported to your local IBM representative.

[2] If this code is returned, restart VS APL with more shared memory and try again. If the resultant workspace size is too small, restart CMS with more virtual storage.

Figure 23. Processor Return Codes: CMS VSAM

VS APL Functions for Auxiliary Processing

VS APL includes a public library workspace named *APFNS* which can be used to simplify auxiliary processing. *APFNS* contains various types of functions. Some functions directly communicate with an auxiliary processor. Others can be used to prepare for auxiliary processing. Other functions still, can be used to help in converting data once an auxiliary processing operation has been executed.

Figure 24 lists the functions in *APFNS* summarizing the operations they perform. For detailed information on any one of these functions, display the variable *HOWfn* in *APFNS*, where *fn* is the name of the function. For example, display *HOWCMS∆* for information on the function *CMS∆*. For the functions *ECI*, *ECO*, *FI*, *FO*, *II*, *IO*, *LI*, *LO*, *PDI*, and *PDO*, display the variable *HOWDATACV*. For the VSAM file functions, display *HOWVSAM*.

Function Name	Description
CMSΔ	Allows you to issue certain CMS commands.
CPΔ	Allows you to issue CP commands.
DISCΔ	Disconnects the phone line but continues execution until the job is done.
ECI	Converts EBCDIC characters to APL characters.
ECO	Converts APL characters to EBCDIC characters.
FI	Converts System/370 floating-point numbers to APL format.
FO	Converts APL numbers to floating-point format.
FΔO	Issues a CMS FILEDEF command and shares two variables with auxiliary processor 111.
II	Converts System/370 binary values to APL format.
IO	Converts APL numbers to System/370 binary format.
LI	Converts System/370 logical data to APL format.
LO	Converts APL numbers with values of 0 or 1 to System/370 logical format.
PARSEΔ	Breaks up a character vector into tokens. Used by FΔO.
PDI	Converts System/370 packed-decimal numbers to APL format.
PDO	Converts APL numbers to System/370 packed-decimal format.
PRTΔCON	Prints a disconnected console file.
QTΔ	Asks for the time.
SLΔ	Locks the terminal so it can receive messages while using no CPU time.
USE	Selects a VSAM file for subsequent file operations. This function must be executed before the first VSAM file function in APFNS is executed. The selected file becomes the target file for these functions until a new file is selected.
VERASE	Erase a record from a key-sequenced VSAM file.
VGET	Reads directly from a key-sequenced VSAM file.
VGETHOLD	Reads directly for update from a key-sequenced VSAM file. Used in conjunction with VSET to update a VSAM file.
VPOSITION	Positions the internal record pointer in a key-sequenced or entry-sequenced VSAM file for subsequent record access.
VREAD	Reads sequentially from a key-sequenced or entry-sequenced VSAM file.
VREADHOLD	Reads sequentially from a key-sequenced or entry-sequenced VSAM file. Used in conjunction with VSET to update a VSAM file.
VSET	Writes a record to a key-sequenced or entry-sequenced VSAM file.

Figure 24. Functions in APFNS

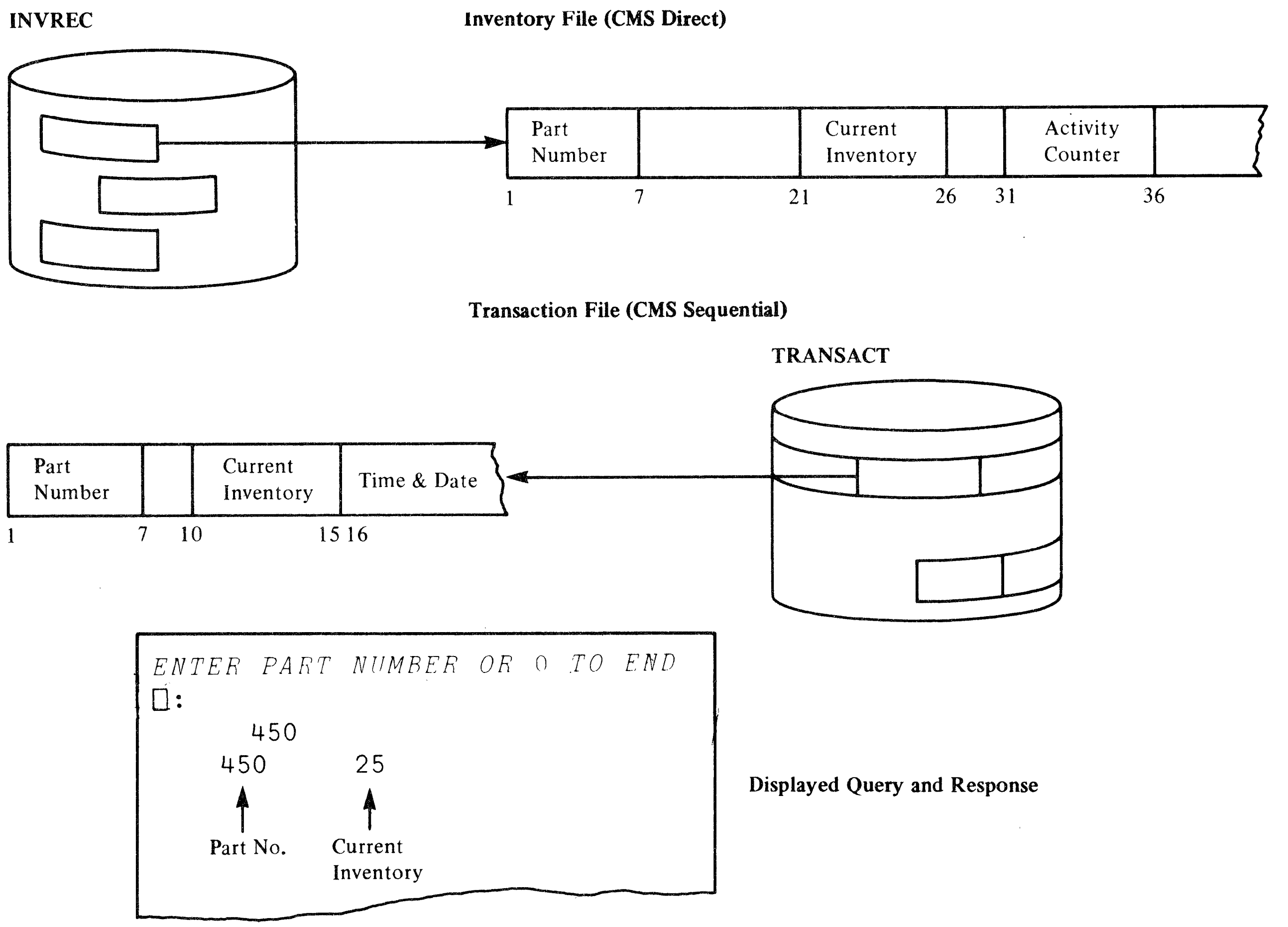
SAMPLE TERMINAL SESSION

This section contains a hypothetical terminal session in which a random CMS file of inventory records named *INVREC* is queried for information pertinent to a given part number. Once the information is displayed, the transaction is recorded sequentially on a CMS file named *TRANSACT*.

Figure 25 illustrates the format of the records in *INVREC* and *TRANSACT* and shows the information that is printed in response to a query.

The primary intent of this sample session is to demonstrate the use of VS APL system commands and to illustrate how file input/output operations can be performed through auxiliary processing.

Significant points in the session are identified by circled numbers in the left-hand margin; each number has a corresponding explanation at the end of the printed session.



```
ENTER PART NUMBER OR 0 TO END
⎕:
      450
     450        25
```

Figure 25. Inventory Record Application

```
   d'x38z irvyr;     vm/370 online

   logon 123456 m
❶  [redacted]
   LOGON AT 14:46:50 PDT TUESDAY 03/23/76
   ipl cms
   CMS VERSION 2.0 PLC 13 8/21/75
❷  apl apl110

         v s  a p l

   clear ws
❸        )LIB
   PAYROLL
   PRODCTL
   YEAREND
❹        )LOAD PRODCTL
   SAVED 10:01:57 6/14/75
   WSSIZE IS 103516
❺        )FNS
   GETINVENTORY INQUIRE NOTEACTIVITY
         )VARS
   PARTNUMBERS

❻        ∇INQUIRE[⎕]∇
❼      ∇ INQUIRE;INDEX;T;RECORD;TRANSACT;RECRD
   [1]  ⍝ INITIALIZE AND OFFER A RECORD VARIABLE FOR TRANSACT
   [2]   TRANSACT← 'TRANSACT(370'
   [3]   T←110 ⎕SVO 'TRANSACT'
   [4]  ⍝ INITIALIZE AND OFFER RECORD+CONTROL VARS FOR INVREC
   [5]   RECRD← 'INVREC FILE(370'
   [6]   INDEX← 'INVREC FILE(CTL'
   [7]   T←110 ⎕SVO 2 5⍴'RECRDINDEX'
   [8]   T←INDEX,RECRD,TRANSACT
   [9]  ⍝ INQUIRE FOR CURRENT INVENTORY
   [10] INQLOOP: 'ENTER PART NUMBER OR 0 TO END'
   [11]  →(0=INDEX←⎕)/0
❽  [12]  INDEX←PARTNUMBERS⍳INDEX
❾  [13]  →(INDEX[2]>⍴PARTNUMBERS)/INQLOOP
   [14]  SAVEPTR←INDEX[2]
   [15]  RECORD←RECRD
   [16] ⍝ PRINT PART NUMBER AND INVENTORY
   [17] ⍝ PART NUMBER IS IN 1-7
❿  [18]  ⎕←T←(7↑RECORD),'   ',GETINVENTORY
⓫  [19]  NOTEACTIVITY
   [20]  INDEX←0,SAVEPTR
   [21]  RECRD←RECORD
   [22] ⍝ RECORD TRANSACTION-ADD TIMESTAMP
⓬  [23]  TRANSACT←T,'   ',⍕⎕TS
   [24]  →INQLOOP
       ∇

         ∇GETINVENTORY[⎕]∇
       ∇ I←GETINVENTORY
   [1]  ⍝ INVENTORY IS IN 21-26
   [2]   I←RECORD[20+⍳6]
       ∇
```

```
          ∇NOTEACTIVITY
        ∇ NOTEACTIVITY
    [1]   ⍝ ACTIVITY COUNT IS IN 31-36
    [2]     RECORD[30+⍳6]←6 0⍕1+⍎RECORD[30+⍳6]
        ∇
13        INQUIRE
    ENTER PART NUMBER OR 0 TO END
    ⎕:
          125
         125      3040
    ENTER PART NUMBER OR 0 TO END
    ⎕:
          705
         705       127
    ENTER PART NUMBER OR 0 TO END
    ⎕:
14        0
15        )CONTINUE
16  14:52:52 08/12/75
    CONNECT=00:06:02 VIRTCPU=000:00.47 TOTCPU=000:01.33
    LOGOFF AT 14:52:54 PDT TUESDAY 3/23/76
```

1. At this point, a password is entered over the blots.
2. The CMS Disk I/O Processor (APL110) is loaded into the virtual machine.
3. The user displays the contents of his library.
4. A copy of the *PRODCTL* workspace is brought into the active workspace. This action changes the size of the active workspace to 103,516 bytes.
5. The global functions and the global variables in the active workspace are displayed.
6. *INQUIRE* asks the user to enter a part number. Based on the number entered, *INQUIRE* reads a record in the inventory file *INVREC*. It then prints the part number and the current inventory found in the record. With each display, *INQUIRE* updates an activity counter in the *INVREC* record and writes a transaction record to *TRANSACT*. If a part number of 0 is entered, execution of *INQUIRE* ends.
7. The initial part of *INQUIRE* prepares for input/output processing. Since *TRANSACT* is a sequential file, only a record variable is used. Since *INVREC* is a random file, a record variable and a control variable are used.
8. *PARTNUMBERS* is a numeric vector containing part numbers sequenced to match the part numbers in *INVREC*.
9. If the part number is valid, *INQUIRE* reads the record in *INVREC*. Before the read operation is performed, the read pointer is saved. It will be used later to update the write pointer.
10. *INQUIRE* prints the part number found in the record and uses the function *GETINVENTORY* to print the current inventory.
11. The function *NOTEACTIVITY* updates the activity counter found in the record. The updated record is then written to *INVREC*.

12. *INQUIRE* writes a record to *TRANSACT*. The written record contains the part number, current inventory, date and time of the write operation.
13. The user begins the transaction.
14. The user ends execution of *INQUIRE* by entering `0`.
15. A copy of the active workspace will be automatically activated as soon as contact is next made with VS APL.
16. The system responds with a summary of the connection and CPU time used during the session. The session is now over.

APPENDIX A: DISTRIBUTED WORKSPACES

Certain predefined workspaces are distributed with VS APL. These workspaces are very helpful if you're trying to learn APL, converting from some other IBM APL system, or intending to use auxiliary processors.

Figure 26 lists and summarizes the workspaces distributed with VS APL. Generally, distributed workspaces are found in public libraries 1 and 2, but it is up to your installation personnel to determine which public libraries will actually be used.

Like other public library workspaces, the workspaces in Figure 26 can be retrieved at will. For instance, the distributed workspace *NEWS* can be activated by entering:

```
      )LOAD 1 NEWS
SAVED 11:15:27 01/15/76
```

Similarly, if you are only interested in a particular function in one of the distributed workspaces, you can copy it into your active workspace:

```
      )COPY 1 NEWS SCHEDULE
SAVED 11:15:27 01/15/76
```

All of the workspaces distributed with VS APL are self-documenting. Specifically, each workspace contains a group named *DESCGP*, which can be used like a computer-stored textbook to describe the contents and use of the workspace. *DESCGP* includes the following variables:

Variable Name	Contents
ABSTRACT	Gives the purpose of the workspace in one or two sentences.
DESCRIBE	Gives the purpose of the workspace in detail. It also lists and details the names, syntax, and argument requirements of functions in the workspace.
HOW	Describes how the workspace is used. It also details the use of functions in the workspace.

Because distributed workspaces are self-documenting, no further discussion of their contents and use are provided here. You should reference *ABSTRACT*, *DESCRIBE* and *HOW* as appropriate for the information you need.

Workspace Name	Description
Library 1 workspaces:	
NEWS	Contains functions for the storage and display of bulletins to VS APL users.
CONVERT	Assists in workspace conversion from previous IBM APL systems to VS APL. It analyzes the content of each unlocked function and converts statements where possible. It also provides a report identifying statements that were converted and statements which may require manual correction.
WSFNS	Assists in workspace conversion to VS APL by providing comparable defined functions for the APL system variables ⎕*IO*, ⎕*PP*, ⎕*PW*, ⎕*CT*, and ⎕*RL*, and for the APL system function ⎕*DL*.
APLCOURSE	Tutors the APL user and tests his understanding of the elements of the APL language through a set of questions to which he responds.
TYPEDRILL	Tests the typing speed and accuracy of the terminal user.
EXAMPLES	Illustrates APL coding techniques employing useful functions.
PLOT	Prints graphs, histograms, and formatted numeric arrays and matrices.
FORMAT	Contains functions designed to aid in the formatting of numeric output. It allows the inclusion of commas as separators, provides for extension by leading and trailing zeros, permits arbitrary symbols to indicate negative numbers, and allows text (within limits) to be associated with each row of numeric output.
HOWEDITS	Describes the *SEDIT* and *MEDIT* workspaces.
SEDIT	Contains functions that are useful for editing line-type data such as program listings. Data is stored in a single string containing no blanks.
MEDIT	Functionally equivalent to *SEDIT* except that data is stored in a matrix.
SBIC	Contains functions for recording orders, maintaining an inventory, and preparing invoices. It is a skeletal system designed to illustrate the use of APL in commercial data processing.
Library 2 workspaces:	
APFNS	Contains functions that can be used with the VS APL auxiliary processors under CMS. For further details, refer to the section "Auxiliary Processing."
PRINT	Contains functions that can be used to transmit APL functions and data to an offline printer using the APL print chain. The functions in this workspace are organized so that the APL user can easily modify them to meet individual needs.

Figure 26. VS APL Distributed Workspaces

APPENDIX B: WORKSPACE CONVERSION

This appendix introduces the VS APL Conversion Program and describes the report generated by the program.

The VS APL Conversion Program

The VS APL Conversion Program converts APL\360, APLSV and APL/CMS workspaces to VS APL workspaces. The conversion program is designed to be executed by the personnel responsible for installing your system. Once the program has been run, you'll be given a conversion report which lists those items in each workspace which must be evaluated or modified to ensure proper execution of the workspace under VS APL.

Pre-Conversion Considerations

When a workspace is submitted for conversion, the conversion program uses the library number and workspace name of the input workspace to form a workspace identification for the converted workspace. Not all library numbers and workspace names are acceptable to VS APL. Specifically, the following are unacceptable:

- A library number greater than seven digits
- A workspace name containing the symbol delta (∆) or underscored characters

These unacceptable items should be modified before the conversion program is run.

APLSV workspaces should be dumped using LEVEL 1; dumping is the installation's responsibility.

Types of Conversions

There are two types of conversion that the VS APL Conversion Program performs: format and content.

In format conversion, all global objects (variables, functions, groups) and their names are converted into VS APL internal format.

In content conversion, all global objects are converted into VS APL internal format and all function statements are examined for items which require conversion in order to execute properly under VS APL. Such items are converted to their VS APL equivalent, if possible, and noted on the conversion report. Items with no VS APL equivalent are just reported.

Conversion of Character Variables

Character variables in APLSV workspaces are fully mapped, one-to-one, from 256 possible characters in APLSV to 256 possible characters in VS APL, provided the workspaces were dumped using LEVEL 1; dumping is your installation's responsibility. Similarly, character variables in APL/CMS workspaces are fully mapped into VS APL characters.

Character variables in APL/360 workspaces and in APLSV workspaces dumped using LEVEL 0 are mapped as follows:

- Terminal graphics described in the *APL \360 User's Manual* are mapped into corresponding VS APL characters.
- Terminal control characters, backspace, line feed, and carriage return are mapped into their equivalents in VS APL.
- All other characters are replaced by the blot character.

Conversion assumes that characters are used for their *logical* meaning as opposed to their *physical* meaning. The logical meaning of a character in a workspace is the terminal graphic or control character represented by its bit pattern in the workspace. The physical meaning is the absolute value of its bit pattern. Thus, those users who were using characters for their physical meaning will have to reconvert those characters to their original values. The function *CCO* in the distributed workspace *APFNS* can be used to reconvert APL/CMS characters. For APLSV characters, the distributed workspace *CONVERT* contains functions to reconvert the characters. Mappings from APL/CMS are documented in the *APFNS* workspace, and mappings from APLSV are documented in the *CONVERT* workspace.

The Conversion Report

The conversion report lists exceptional conditions found during format or content conversion. When format conversion is done, the report produced contains any exceptions found in workspace parameters or variables and any conversion errors. When content conversion is performed, the report additionally contains any exceptional functions. Figure 27 illustrates a sample conversion report.

```
LIBRARY        897574          CONVERSION SUMMARY REPORT
WORKSPACE: VIP

WS PARAMETER:
      PRINT PRECISION                 5
      RANDOM LINK                     1097971256
      COMPARISON TOLERANCE            1.136729599338082E-13

FUNCTION:      DELAY
      REPLACED

FUNCTION:      BET
*     TRANSPOSE                 1

FUNCTION:      KEYWORD
*     RESIDUE                   3
**    DYADIC IBEAM              1     2     3

FUNCTION:      HAD
      (LOCKED)

FUNCTION:      TIME
*     MIXED OUTPUT              1
*     ENCODE                    1
*     MONADIC IBEAM             1

FUNCTION:      IBE
      IBEAM SIMULATOR FUNCTION ADDED TO WORKSPACE

**    WARNING. WILL NOT EXECUTE UNDER VS APL
*     CAUTION. MAY NOT EXECUTE AS INTENDED UNDER VS APL
```

Figure 27. Sample Conversion Report

Workspace Parameters Reported

For each workspace to be converted, the name and value of the following workspace parameters are reported if the value in the input workspace is other than the VS APL default:

- Comparison tolerance
- Index origin
- Printing precision
- Random link
- Symbol table size

Variables Reported

If a variable is invalid, its name and one of the following exceptions are reported. Invalid variables are deleted.

** REJECTED, INVALID DIMENSIONS(S)

The element of a dimension is negative or exceeds the VS APL maximum, or the element count exceeds the VS APL maximum.

** REJECTED, INVALID RANK

The rank is less than zero or exceeds the VS APL maximum.

Functions Reported

Functions are examined and exceptional conditions are reported only when content conversion is done. For each function, the function name and a list of exceptions are reported. The format of each item in the list is:

flag exception line-number(s)

The line numbers are those in the function in which the exception occurs. The flag indicates the level of severity:

Flag	**Meaning**	**Description**
blank	Information	The item has been converted to its VS APL equivalent and will execute properly under VS APL.
*	Caution	The item may not execute as intended under VS APL.
**	Warning	The item will not execute under VS APL.

The exceptions are as follows. The information is parentheses indicates the source APL systems for which the exception should be evaluated and corrected where needed. The information in parentheses does not appear in the conversion report.

** DYADIC IBEAM (APL\360, APL/CMS, APLSV)

There is no VS APL equivalent for dyadic Ibeams. They are left unchanged. Execution results in a syntax error.

** LINE TOO LONG (APL\360, APL/CMS, APLSV)

If conversion causes a line to be expanded to more than 4093 bytes, the line is deleted and replaced with:

"THIS FUNCTION LINE WAS TOO LONG AND WAS DELETED BY CONVERSION" *item*

The term *item* will cause a syntax error in the converted statement.

** SYSTEM VARIABLE (APL/CMS, APLSV)

There is no VS APL equivalent for system variables ⎕*TT* and ⎕*UL*. They are left unchanged. Execution results in a syntax error.

** UNCONVERTIBLE (APL\360, APL/CMS, APLSV)

A function which localizes ⎕*TT* or ⎕*UL*, or a function whose header contains a syntax error cannot be converted. Such functions are deleted.

* AMBIGUOUS IBEAM (APL\360, APL/CMS, APLSV)

An ambiguous Ibeam is assumed to be monadic converted as described in "Monadic Ibeam" below.

* ELIDED SEMICOLS (APL\360, APLSV)

When mixed output is converted, contiguous semicolons are discarded (see "Mixed Output" below).

* ENCODE (APL\360)

The definition of encode for a left argument having one or more negative elements is different from the definition used in APL/360. Execution might have a result different from the one expected.

* INCOMPLETE LIST (APL\360, APL/CMS, APLSV)

The function contains more exceptions than the conversion program can record. Additional exceptions exist, but are not reported.

* MIXED OUTPUT (APL\360, APL/CMS, APLSV)

Mixed output is converted to an equivalent expression that uses format primitive function. Execution in rare cases may cause a rank or length error.

* QUAD AV (APL/CMS, APLSV)

The correspondence of particular symbols to elements of ⎕*AV* is implementation dependent. Reference to ⎕*AV* might have a different result from the one expected.

* RESIDUE (APL\360)

The definition of residue for a left argument having one or more negative elements is different from the definition used in APL\360. Dyadic and ambiguous residue are reported. Execution might have a different result from the one expected.

* TRANSPOSE (APL\360)

The definition of monadic transpose for an argument of rank greater than two is different from the definition used in APL\360. Monadic and ambiguous transpose are reported. Execution might result in a different result than expected.

CARRIER RETURN (APL\360, APL/CMS, APLSV)

A character array that contains carrier returns is converted to an equivalent expression that uses the second element of the system variable ⎕*TC*.

IBEAM SIMULATOR FUNCTION ADDED TO WORKSPACE (APL\360, APL/CMS, APLSV)

See "Monadic Ibeam" below.

* MONADIC IBEAM (APL\360, APL/CMS, APLSV)

A monadic or ambiguous Ibeam is replaced with a call to a monadic function; the argument of the function is that of the Ibeam. The function simulates all monadic Ibeams except I23 and I28, which have no equivalents in VS APL. Execution of the function with such arguments results in an error. The simulator function that is added to the workspace has a unique name; the name is <u>IBE</u> unless that name already exists in the workspace, in which case it is <u>IBF</u>. The function is locked so it will behave as a primitive function. The canonical representation of <u>IBE</u> is in the CONVERT distributed workspace and is named CR<u>IBE</u>.

REPLACED (APL\360, APLSV)

A WSFNS function (DELAY, DIGITS, ORIGIN, SETFUZZ, SETLINK, WIDTH) is replaced if the function is locked and its definition is exactly the same as that in APL\360 distributed library 1. The new function is unlocked; it performs the equivalent of the replaced function using the appropriate system variable.

(LOCKED)

The function is locked. Content conversion has been done, but exceptions are not reported. The flag indicates the highest severity level.

Conversion Errors Reported

The following error messages are displayed in the VS APL Conversion Report and should be reported to the personnel responsible for installing your system:

WORKSPACE FULL

SYSTEM ERROR WHILE CONVERTING WORKSPACE/DIRECTORY *number name* WORKSPACE/DIRECTORY UNCONVERTED AND PRESUMED DAMAGED

ERROR UNRECOVERABLE. CONVERSION ABORTED

*** WORKSPACE REJECTED, NOT CONVERTED, DUE TO I/O ERROR

*** HDR LABEL I/O ERROR. CONVERSION CANCELLED

*** TRLR LABEL I/O ERROR. CONVERSION CANCELLED

WRITE ERROR nn WHILE SAVING libnum name. CONVERSION CANCELLED.

NOT A CMS DUMP TAPE. CONVERSION CANCELLED.

filename filetype filemode NOT AN APL/CMS (PRPQ) WORKSPACE.

INADEQUATE SPACE TO RUN CONVERSION. CONVERSION CANCELLED.

filename filetype WORKSPACE TOO LARGE FOR VIRTUAL MACHINE SIZE. RERUN WITH LARGER MACHINE. WSSIZE IS xxxx NEED ABOUT yyyy BYTES TO CONVERT.

filename filetype WORKSPACE DAMAGED OR NOT VER2. NOT CONVERTED.

In addition, the conversion program can abend with user code 32 if more than 10 damaged workspaces are found.

Unreported Items

The following items may cause a converted workspace to execute differently under VS APL than it did under the source APL system. They are not reported by the conversion program.

Imbedded respecification may cause a result different than expected. In VS APL, the following statements yield a result of 15:

```
A←5
(A←3)×A
```

For APL/CMS and APLSV workspaces, system variables are given the corresponding workspace parameter value. The value of a system variable with an implicit error is not retained.

For APL/CMS workspaces, an underscore character _ in the name of an object is converted to a delta underscore ($\underline{\Delta}$) character. If the underscore character appears in a character array, the character is transmitted as is. The value of ⎕*LX* is not retained; it is given the default value of null.

Language differences that are apparent only by execution-time evaluation of the arguments are not flagged by the conversion program.

All APL/CMS workspaces have a filetype of VMAPLWS or Pxxxxxx after conversion. Such filetypes must be changed to VSAPLWS or Wxxxxxxx before use with VS APL. See your system administrator.

APPENDIX C: LANGUAGE CONSIDERATIONS

This appendix covers some special language considerations for VS APL.

Duplicate Names in a Defined Function

In VS APL, the name of a defined function, local names indicated in the function header, and labels contained within the body of the function are established in the following order:

1. Function name.
2. Other names in the function header taken from left to right.
3. Labels in the function body taken from top to bottom.

If duplicate names appear, then all but the first occurrence of each name taken in the above sequence are ignored.

Line Deletion in a Function Definition

During function definition, any entered statement *N* can be deleted by typing `[∆N]`, where *N* is a single statement number. After deleting the statement, the system awaits entry of statement *N*.

For example, the following illustrates the deletion of statement 1 in the definition of *FUNC*:

```
      ∇FUNC
[1]   ALPHA←2 3⍴⍳6
[2]   BETA←⎕
[3]   [∆1]
[1]
```

Maximum Sizes for VS APL Objects

An array can have no more than 2,097,117 elements. A function can have no more than 2048 lines (including the header), and its internal encoding must occupy no more than 65,536 bytes.

APPENDIX D: CMS TERMINALS

This appendix summarizes the sign-on/sign-off procedures and various operating characteristics for all the terminals supported by CMS and discusses a number of items to be considered when using APL at CPT-TWX or IBM 3270 terminals. Terminal operating characteristics are summarized in Figure 28. For further information concerning each of the terminals supported by CMS, refer to the *VM/370: Terminal User's Guide.*

Action	TERMINALS				CPT-TWX Mod 33/35
	3767	2741	3270	1050	
Set switches for computer connection	Keylock=ON COMM/LOCAL=COMM AUTO/OFF=AUTO EDIT/OFF=OFF AUTO VIEW/OFF=as desired DOUBLE/SINGLE SPACE=as desired DATA/TALK=DATA DIAL DISC/OFF=OFF SDLC/SS=SS EBCDIC (or Corr.)/APL= EBCDIC (or Corr.) CALC/OFF=OFF TEST/OFF=OFF POWER/OFF=POWER	COM/LCL=COM ON/OFF=ON	Pull out OFF/ PUSH knob	MAIN-LINE=ON SYSTEM=ATTEND MASTER=ON PRINTER 1=SEND/ REC PRINTER 2=HOME KEYBOARD=SEND PUNCH=NORMAL SYSTEM=PROGRAM EOB=AUTO SYSTEM=UP All others=OFF	Press ORIG button Model 35: Press K button On high-pitched tone, press CTRL, WRU
Terminal ready to accept logon	DATA SET READY, PROCEED lights on	Keyboard unlocks	SYSTEM AVAILABLE Cursor appears	POWER, PROCEED lights on	Key mode: paper advance optional ? printed
Logon	logon userid m	logon userid m	logon userid m	logon userid m	logon userid m
Enter a line	↵ (return)	RETURN	ENTER	RETURN ALTN CODING + EOT	RETURN
Terminal ready to accept input	ON LINE and PROCEED lights on	Unlocks keyboard	Cursor appears VM READ	Unlocks keyboard PROCEED light on	Paper advance Line number Stops noise Period
Correct character in current APL line	◄ (backspace), ATTN, retype, ↵(return)	BKSP, ATTN, retype, RETURN	Position cursor, type correct character	BACKSPACE, ATTENTION, retype, RETURN	Press BREAK and retype entire line, RETURN
Send signal to system	ATTN	ATTN	PA1	ATTENTION	BREAK
Interrupt input	O◄U◄T	O BKSP U BKSP T	PA2[1] ENTER	O BACKSPACE U BACKSPACE T	O CTRL H U CTRL H T
Cancel output	ATTN	ATTN	PA2[1,2]	ATTENTION	BREAK
Sign off	)OFF)CONTINUE	)OFF)CONTINUE	)OFF)CONTINUE	)OFF)CONTINUE	)OFF)CONTINUE
Turn off terminal	POWER=OFF Keylock=OFF	ON/OFF=OFF	Push OFF/PUSH knob	MAIN-LINE=OFF	CLR
Default printing width (⎕PW)	120	120	79	120	120

[1] VM/370 (Release 3) only

[2] While in MORE state

Figure 28. CMS Terminal Summary

CPT-TWX Terminal Considerations for APL

CPT-TWX models 33 and 35 have a fixed and limited character set and cannot be used in editing APL functions. Existing workspaces can be loaded and names which do not contain underscored alphabetic characters or delta (∆) can be entered and functions executed. There is no right arrow character on these terminals, so that suspended functions cannot be resumed, and the state indicator cannot be cleared. Similarly, there is no backspace character on these terminals. However, the combination of the CTRL and H keys is transmitted as a backspace.

IBM 3270 Display System Terminal Considerations for APL

The IBM 3270 terminals that can be used with VS APL are the IBM 3277 or the IBM 3275. The following discussions highlight some of the items to be considered when using VS APL at these terminals. For further details on the use of IBM 3270 terminals under VM/370, refer to the *VM/370 Terminal User's Guide.*

The IBM 3270 Data Analysis-APL Feature

The IBM 3270 Data Analysis-APL feature makes it possible to enter and display the full EBCDIC and APL character set at an IBM 3277 terminal. The feature is available on the IBM 3277 and may be used if VS APL is operating under VM/370 Release 3; it is not available on an IBM 3275 terminal.

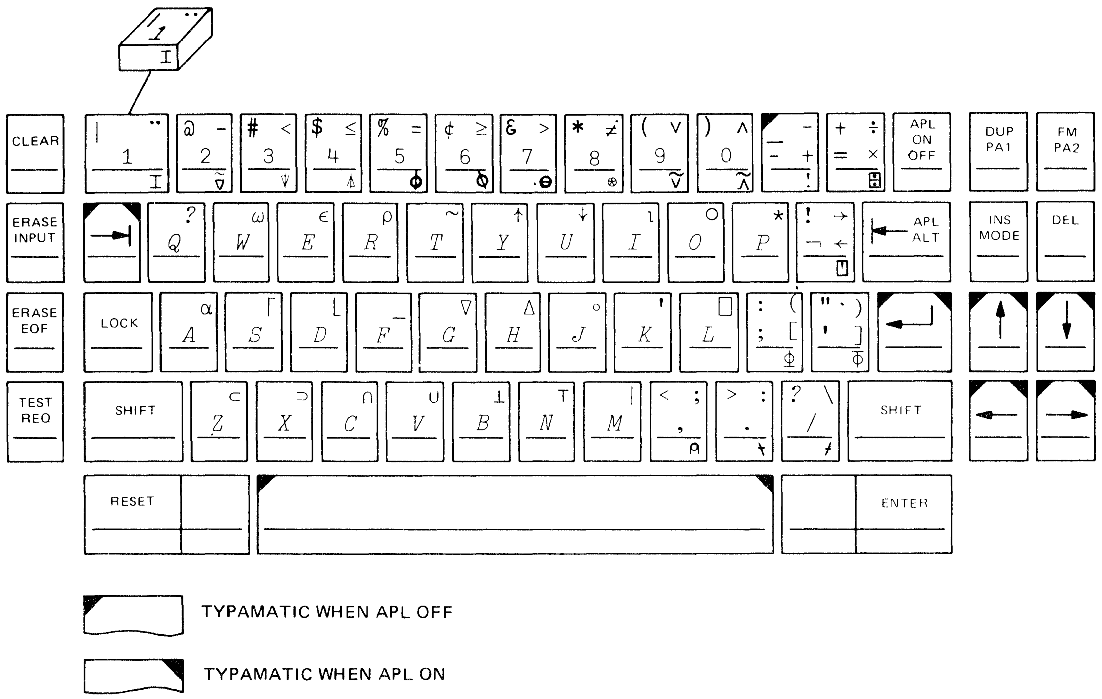
Figure 29 illustrates the keyboards that can be ordered with the Data Analysis-APL feature. The APL keyboard is standard, while the text keyboard is optional. If the feature is installed with the APL keyboard, all the operations normal to APL work are available. The text keyboard does not include all the APL characters and therefore cannot be used for all kinds of APL operations. (For the remainder of this discussion, unless otherwise indicated, an APL keyboard is assumed.)

When power is initially turned on, the keyboard acts as a regular 3270 keyboard. Pressing the APL ON/OFF key once, turns on the APL character set. All the APL graphic characters can be produced by pressing the appropriate key. Compound characters are produced by holding down the APL ALT key and pressing the key that has the appropriate compound character on its front. Underscored alphabetic characters are produced by holding the APL ALT key and pressing the alphabetic key. The APL ON/OFF key does not affect character display. A 3270 terminal equipped with the APL feature (either keyboard) can display all EBCDIC, APL, and text characters regardless of whether the APL ON/OFF key is on or off.

You must execute the CP command TERMINAL APL ON before you communicate APL characters to VS APL. Normally, the command TERMINAL TEXT ON, which is associated with the use of the text keyboard under VM/370, is incorrect for VS APL applications.

For further information on the APL feature, refer to the publication *An introduction to the IBM 3270 Data Analysis-APL Feature.* For further information on the TERMINAL APL ON command, refer to the publication *VM/370: CP Command Reference.*

APL KEYBOARD



TEXT KEYBOARD

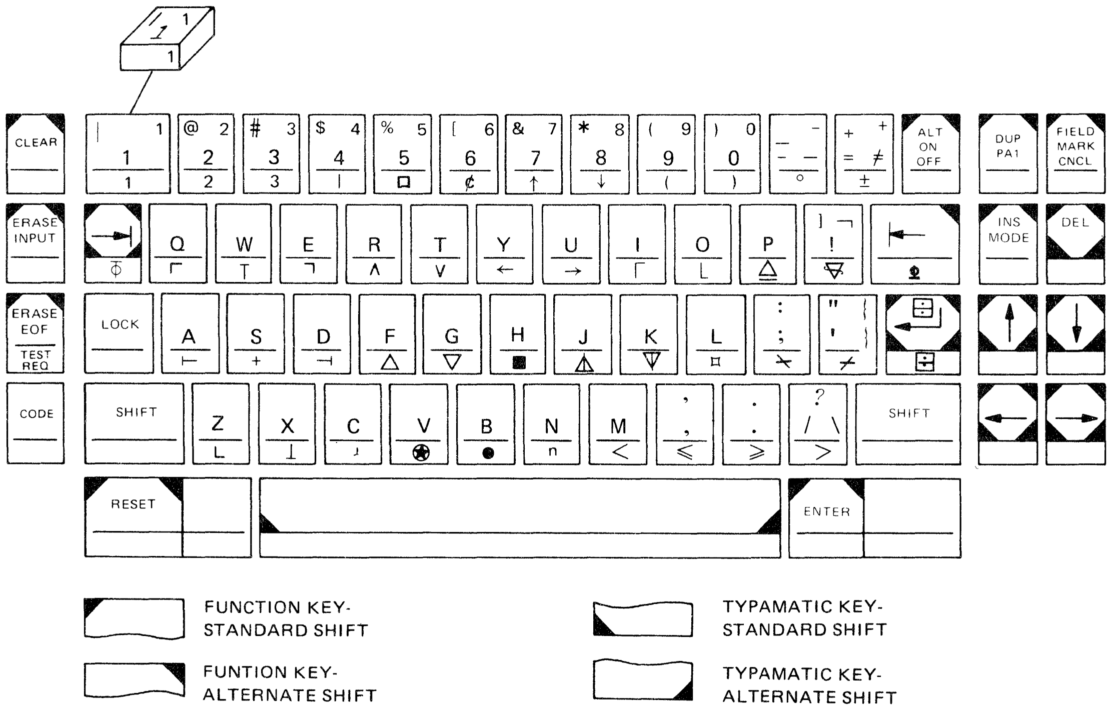

CLEAR
ERASE INPUT
ERASE EOF
TEST REQ
CODE
ALT ON OFF
DUP PA1
FIELD MARK CNCL
INS MODE
DEL
LOCK
SHIFT
RESET
ENTER
FUNCTION KEY-STANDARD SHIFT
FUNTION KEY-ALTERNATE SHIFT
TYPAMATIC KEY-STANDARD SHIFT
TYPAMATIC KEY-ALTERNATE SHIFT

Figure 29. APL and Text Keyboards for the IBM 3270 Data Analysis-APL Feature

Using the IBM 3270 without the Data Analysis-APL Feature

If the IBM 3270 is operated without an APL feature, the same restrictions in general discussed for the CPT-TWX terminals apply. Existing workspaces can be loaded and names which do not contain underscored alphabetic characters or delta can be entered and functions executed. There is no right arrow character on these terminals, so that suspended functions cannot be resumed, and the state indicator cannot be cleared.

In addition, if you do not have the APL feature installed, you will not be able to use APL characters to communicate with VS APL. APL characters directed to the screen by an executing function will be garbled and the screen format could be destroyed if certain characters are generated. (In the event that the screen format is destroyed, press the CLEAR key to reformat the screen.)

Screen Format

Figure 30 illustrates the format of the IBM 3270 display screen for interaction with VS APL. This screen format is established by VM/370 after a successful logon. The screen is separated into three areas:

- Output area, which displays your completed entries and VS APL's responses. The last character position in this area is reserved by the system and cannot be used to display output.
- Input area, which displays your input before it is entered to VS APL. Although the second input line can display as many as 59 characters, VM/370 will accept only 56 characters.

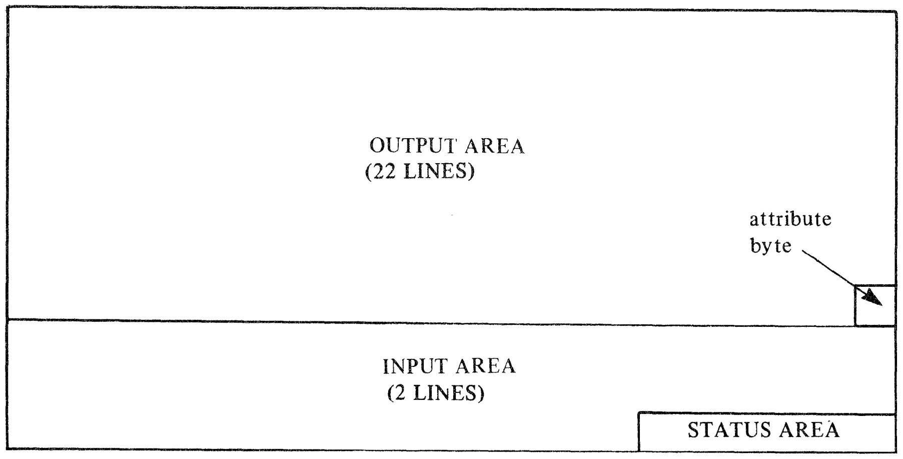


Figure 30. IBM 3270 Screen Format

- Status area, which displays the current status of the terminal. During interaction with VS APL terminal states are:
  - VM READ - VS APL is awaiting input.
  - CP READ - Input will be treated as a CP command. (This state is entered be pressing the PA1 key once.) Return to APL by entering the CP command BEGIN.

- MORE... - The output area is full and more lines must be displayed on the screen. Press the CLEAR key to clear the output area for more output or wait 60 seconds for automatic clearing.
- HOLDING - Same as MORE... but no automatic clearing is effected. Holding state is entered from MORE state or vice versa by pressing the ENTER key.
- RUNNING - VS APL is executing; do not enter input until VM READ is displayed.

For further information on terminal states, refer to the *VM/370: Terminal User's Guide.*

Entering Input

When VS APL asks for input it displays a prompt (such as six blanks) in the input area of the screen. It then positions the cursor in the same relative position that an IBM 2741 printing element would appear when the keyboard unlocks. You can then enter one or more APL characters. Each character entered appears in the input area. To correct a character, move the cursor back to the incorrect character using any of the four cursor control keys at the lower righthand position of the keyboard. Then type the correct character or use the INS MODE or DEL keys to otherwise alter the line. The INS MODE key allows insertion of one or more characters at the position of the cursor. The DEL key deletes the character at the position of the cursor without leaving a blank space. When the input line is complete, press the ENTER key to send the input line to VS APL. VM then displays the entered line in the output area of the screen.

Special Keys

In addition to the ENTER, INS MODE, and DEL keys, other keys have important uses:

Key	Use
CLEAR	Blanks screen to allow display of additional output when MORE... or HOLDING appears in the status area. The CLEAR key can also be used to clear the screen before entering additional input. Be aware, though, that this also erases any current prompt and moves the cursor to the first location in the input area.
ERASE INPUT	Blanks or erases the input area of the screen, erasing any prompt, and moves the cursor to the first location in the input area.
ERASE EOF	Replaces all characters from the cursor location to the end of the input area with nulls. (Nulls are displayed as blanks but are not transmitted as characters.)
RESET	Resets the terminal if INPUT INHIBITED is indicated and terminates insertion mode (initiated by the INS MODE key).

Key	Use
PA1	Used to signal attention. Pressing the PA1 key twice (or the ENTER key once) transmits a weak interrupt signal. Pressing the PA1 key four times (or the ENTER key twice) transmits a strong interrupt signal. If the PA1 key is pressed during input, it is ignored.
PA2	Used to interrupt the display of output. If the terminal is in VM READ state, pressing PA2 signals the character *O* backspace *U* backspace *T* and clears the output area. The ENTER key must be pressed to transmit the character. If the terminal is in MORE or HOLDING state and locally attached, pressing PA2 clears the output area and cancels output. If the terminal is in MORE state and remotely attached, pressing PA2 clears the screen but then proceeds to display the next screen of output. To clear the screen press the CLEAR key. Note that VS APL must be running under VM/370 Release 3 to effect this action of the PA2 key. In addition, the CP command TERMINAL APL ON must have first been invoked.

Output Characteristics

The maximum number of characters that can be displayed on a single line of the display screen is 80. An output line greater than 80 characters is folded, that is, characters beginning with the eighty-first appear on the following line at the first character position on the line.

Normal output lines, in addition, are subject to the value of the printing width (⎕*PW*) which has a default value of 79. Normal output lines longer than the current value of ⎕*PW* are folded.

Bare output is not affected by the value of ⎕*PW*, but is limited by the 80-character screen width.

If bare output (⍞ output) is followed by bare input (⍞ input) that includes positioning the cursor back into that bare output, then input characters that overlay identical output characters shown on the screen are replaced by blanks.

Using the Backspace Terminal Control Character

The backspace terminal control character (⎕*TC*[1]) is effected on the IBM 3270 only in bare output situations (⍞ output followed by ⍞ input) and only when it is the final output character. Trailing backspaces like these can be used to reposition the cursor in the input area for subsequent input. In all other situations, ⎕*TC*[1] produces a blot character (") on output. For example, the expression '*S*',⎕*TC*[1],'/' is displayed on a 3270 as:

```
S"/
```

Function Editing

Function editing, or for that matter function definition, is not available at a 3270 terminal that does not have the APL feature installed.

If the APL feature is installed, you can define and edit functions in the standard way. However, the separation of the screen input and output areas may impede the proper alignment of APL editing characters.

If you have the APL feature installed at your terminal, the easiest way to edit function lines is to enter:

```
[line number ⎕0]
```

This will cause the function line with the specified number to be displayed in the input area of the screen. You can then:

- Delete characters using the DEL key
- Insert characters using the INS MODE key. (Be sure to press RESET when you are finished.)
- Replace characters by positioning the cursor and entering replacement characters.

It is important that the printing width be greater than or equal to the length of the line to be edited. If the line is larger than the printing width, it will not be displayed in the input area.

APPENDIX E: ERROR MESSAGES

In your work at the terminal there will be occasions when certain conditions will prevent the system from successfully executing your entry and force the production of an error message instead. This appendix lists and discusses the error messages produced by VS APL that you may encounter at your terminal. For a description of the error messages produced by the control program segment of VM/370 and by CMS, refer to the *VM/370 System Messages* publication listed in the preface.

Error Messages Issued by VS APL

VS APL generates three kinds of error messages, each dealing with a specific type of error situation:

- Error reports, which are produced when VS APL encounters an unexecutable statement.
- Trouble reports, which are produced when VS APL encounters difficulty with a system command.
- Executor messages, which are produced for a variety of error conditions.

VS APL also issues a specific message when an error is encountered by a distributed auxiliary processor.

Error Reports

Error reports are produced by VS APL when it encounters an APL statement that it cannot execute. This may be because you've used incorrect syntax in an entry, specified inappropriate arguments to a primitive function, or otherwise misused names or symbols in a statement. Error reports are also produced when the execution of your statement requires certain resources of the system that the system does not have, like additional space for names.

When an error report is displayed, VS APL indicates the point at which it stopped execution by displaying your erroneous statement and printing the caret symbol (∧) below the point in the statement where the error was detected. You then have the option of correcting the statement or entering something new. For example:

```
      A←2×Z
VALUE ERROR
      A←2×Z
          ∧
```

Here a *VALUE ERROR* was detected because an attempt was made to use a name (*Z*) that had not been assigned a value. You now have the opportunity of assigning a value to *Z* and reentering the statement.

The following lists and describes the error reports issued by VS APL that relate to auxiliary processing. For each listed report, a cause for the report is indicated as well as a suggested action you might take to correct the error. For a description of other error reports generated by VS APL, refer to *APL Language*.

NO SHARES: NUMBER IN USE

Cause: An auxiliary processor has signed on to the shared storage manager with your user number.

Suggested Response: Notify your system administrator.

NO SHARES: SVP INACTIVE

Cause: The shared storage manager is not active.

Suggested Response: Sign off APL and then sign on again with at least one auxiliary processor specified in the APL command.

NOT OFFERED, SV QUOTA EXCEEDED

Cause: You offered more shared variables than your quota allows.

Suggested Response: Retract any unneeded shared variables. Otherwise, sign off APL and then sign on again with more auxiliary processors specified in the APL command.

SV SPACE QUOTA EXCEEDED

Cause: The value you assigned to the shared variable is larger than your quota allows.

Suggested Response: Sign off APL and then sign on again with a larger shared memory size specified in the APL command.

Trouble Reports

Trouble reports are produced by VS APL when it encounters a system command it cannot execute. When trouble is detected in a system command, the appropriate trouble report is immediately displayed. You then have the option of correcting your system command, changing some of the attributes of your workspace so that your command will work, or disregarding the offending entry and going on to something new.

The following lists and describes the trouble reports issued by VS APL. For each listed message, a cause for the message is indicated as well as a suggested action you might take to correct the error.

IMPROPER LIBRARY REFERENCE

Cause: You are not authorized to use the referenced library, or you referenced a non-existent library.

Suggested Response: Make the necessary change in your specification. Make sure that you are authorized to use the referenced library.

INCORRECT COMMAND

Cause: The command you issued was specified incorrectly.

Suggested Response: Check the proper form of the command in the section "System Commands" and make the appropriate changes.

LIBRARY NOT AVAILABLE

Cause: The referenced library is unavailable. Either the disk on which the library resides is not accessed, the private library is not accessed in read/write mode, or the link to a project library failed.

Suggested Response: Make sure the private library is accessed as a read/write A-disk and the public library is accessed as a Y-disk. Retry the operation. If the message persists, contact your system administrator.

MESSAGE LOST

Cause: You sent a message, but sent a weak interrupt signal immediately.

Suggested Response: Resend the message.

NO OBJECTS COPIED

Cause: A)*COPY* or)*PCOPY* command cannot be executed because there is insufficient disk space for temporary files.

Suggested Response: Drop any unneeded workspaces and reissue the command, or ask your system administrator to increase the size of your primary disk.

NOT A CLEAR WS

Cause: You issued the command)*SYMBOLS number,* but your active workspace was not clear.

Suggested Response: Save the workspace, issue a)*CLEAR* command and the *SYMBOLS number* command, and then copy the saved workspace.

NOT COPIED:list

Cause: The objects represented in **list** could not be copied either because they could not fit in your workspace or because you issued a protected copy command and the names already exist in the workspace. If the names of all objects not copied could not fit in your workspace, the list ends with the phrase (LIST INCOMPLETE).

Suggested Response: If appropriate, erase objects that are no longer needed and reissue the)*COPY* or)*PCOPY* command for the objects in **list**.

NOT ERASED:list

Cause: The items represented in **list** have not been erased as you requested; they may be undefined, or (if the state indicator is not empty) may be local rather than global names.

Suggested Response: Clear the state indicator (by typing →) and reissue the)*ERASE* command; or use the ⎕EX system function to erase local items.

NOT FOUND:list

Cause: The items represented in **list** are undefined or not global in the referenced workspace.

Suggested Response: Be sure your)*COPY* or)*PCOPY* command includes only defined global objects. The objects needed may exist in another workspace or may be named differently.

NOT GROUPED:list

Cause: The items represented in **list** have not been grouped as you requested, because they are not well-formed names.

Suggested Response: Correct the names as appropriate and recreate or extend the group to include the corrected names.

NOT GROUPED, NAME IN USE

Cause: The groupname you specified in the)*GROUP* command is already in use for a function or variable.

Suggested Response: Change groupname or erase object that already has that name.

NOT SAVED, LIBRARY IS FULL

Cause: Your attempt to save a copy of the active workspace will exhaust your quota of library space. The command has been rejected.

Suggested Response: Drop any unneeded workspaces, or request a larger quota from your system administrator.

NOT SAVED, NAME IN USE

Cause: A temporary file used for saving an existing workspace already exists. This message is preceded by message APL159I.

Suggested Response: Use the)*DROP* command to erase the old workspace file. This will also cause the temporary file to be erased. Or, use CMS commands to erase or rename the files. Then reissue the)*SAVE* command.

NOT SAVED, THIS WS IS CLEAR WS

Cause: Your active workspace has no wsid assigned to it.

Suggested Response: Assign a name to the workspace using a)*WSID wsid* or)*SAVE wsid* command.

NOT SAVED, THIS WS IS wsid

Cause: The name of your active workspace does not match the name specified in the)*SAVE wsid* command, and the name already exists in the library. The term **wsid** is the name of your active workspace.

Suggested Response: Change the name of your active workspace using the)*WSID wsid* command, change the name specified in the)*SAVE wsid* command, or drop the existing saved workspace.

SI DAMAGE

Cause: You edited or erased a pendent or suspended function, or you copied an object that has the same name as a pendent or suspended function. In consequence, one or more levels of function execution cannot be resumed.

Suggested Response: Issue a)*SI* or)*SINL* command to display the state indicator, and exit from the damaged levels as appropriate.

SPACE NOT AVAILABLE

Cause: The two causes of this message are: (1) the space indicated in the)*LOAD* or)*CLEAR* command cannot be allotted by the system; (2) bare output, bare output followed by bare input, or ordinary input is longer than 1024 characters. The message is displayed and a strong interrupt is generated by VS APL.

Suggested Response: The respective responses are: (1) change the size indication in the)*LOAD* or)*CLEAR* command; (2) make sure that bare output, bare output followed by bare input, or ordinary input is less than 1024 characters.

STACK FULL

Cause: You specified a command whose execution would exhaust the available space in the execution control area.

Suggested Response: Enlarge the execution control area via the)*STACK number* command.

SYMBOL TABLE FULL

Cause: Sufficient room does not exist in the symbol table area of your active workspace for all the new names created by the command.

Suggested Response: Save the workspace, load a clear workspace, define the required number of symbols, and copy the saved workspace.

SYSTEM BUSY, RETRY

Cause: The referenced project library is unavailable. This condition may be temporary as in the case where another user is currently writing to the designated library.

Suggested Response: Retry the command. If the message persists, contact your system administrator.

SYSTEM RESOURCE PROBLEM

Cause: A machine malfunction was encountered in attempting to read or write to a file.

Suggested Response: Reissue your previous command. If the report is still produced, see your system administrator.

USER NOT LOGGED ON

Cause: The user you indicated in the)*MSG* command is not signed-on to the system.

Suggested Response: Check to see that the account number you specified in the command is correct. Otherwise, contact your system administrator.

USER NOT RECEIVING

Cause: The user you designated in the)*MSG* command is blocking messages or is disconnected via a DISCONNECT command.

Suggested Response: If the message is important, contact the user directly. Otherwise, try again later.

WS FULL

Cause: The action you requested would overfill the available area in your active workspace.

Suggested Response: Erase unneeded objects in the workspace. If necessary, save the workspace and reload it with a larger size.

WS LOCKED

Cause: You failed to supply or supplied an incorrect read or write password for the project disk.

Suggested Response: Provide the correct password, as discussed in the "Passwords" section of the "Workspaces and Libraries" chapter. Otherwise see your system administrator.

WS NOT FOUND

Cause: You specified a workspace name that could not be found.

Suggested Response: If the workspace name was incorrectly specified, change it. Otherwise, the workspace you need may not exist.

WS TOO LARGE

Cause: The size you specified in a `)LOAD` command is too small to accommodate the contents of the referenced workspace, or your maximum allowable workspace size cannot accommodate the contents of the referenced workspace.

Suggested Response: If possible, enlarge the size in your `)LOAD` command or issue the command without a size specification.

Executor Messages

Executor messages are caused by a variety of error conditions that are encountered by the system, for instance untenable situations created while VS APL was initialized, internal errors in library designations, or insufficient resource allotments for APL work.

Executor messages are always preceded by an identifier beginning with the letters *APL*.

The following lists and describes the messages directed to your terminal by the VS APL executor. Each message is shown with its message identifier, its cause, and the action that the system takes in response. A suggested action that you might take to correct the error condition is also supplied for each message.

In a number of cases, the error condition described will have no specific meaning to you. For these cases, the user action will direct you to see your system administrator. He will know how to interpret the error situation with the information supplied.

APL001I ERROR WHILE GETTING SPACE FOR GLOBAL TABLE. OP=*x*, RC=*y*.

Explanation: During initialization, while virtual storage for the executor's global table was being obtained, an error return from a CMS DMSFREE or DMSFRET macro occurred; *x* is the request that failed:

- 1—Initial DMSFREE
- 2—DMSFREE used to adjust to a page boundary
- 3—DMSFRET used to adjust to a page boundary

y is the return code (see *IBM Virtual Machine Facility/370: System Programmer's Guide*).

System Action: Execution is terminated with an abend.

User Action: Restart VS APL. If error persists, use the CP DEFINE STORAGE command to increase the size of the virtual machine.

APL002I ERROR *x* INITIALIZING APL ASSIST. ASSIST NOT IN USE.

Explanation: During initialization, when a test was made for the presence of the VS APL Microcode Assist, an unexpected condition code *x* was returned. Probable cause is a program error either in the Microcode Assist or in module APLSCINI.

System Action: Execution continues without use of the VS APL Microcode Assist.

User Action: Report the problem to the system administrator.

APL003I CONDITION CODE 1 FROM PURGESYS OF SHARED APL.

Explanation: During shutdown, while the shared address space occupied by part of the VS APL Processor was being purged, the PURGESYS request failed with condition code 1 (no shared program). Probable cause is a program error in module APLSCINI.

System Action: Execution is terminated with an abend.

User Action: Report the problem to the system administrator.

APL004I CONDITION CODE 2 FROM PURGESYS OF SHARED APL, RC=*x*.

Explanation: During shutdown, while the shared address space occupied by part of the VS APL Processor was being purged, the PURGESYS request failed with condition code 2 and return code *x*. (See *IBM Virtual Machine Facility/370: System Programmer's Guide.*) Probable cause is a program error in module APLSCINI.

System Action: Execution is terminated with an abend.

User Action: Report the problem to the system administrator.

APL005I LIBRARY TABLE FILE NOT FOUND. NO PUBLIC OR PROJECT LIBS.

Explanation: During initialization, the library table file (filename and filetype are APLIBTAB) was not found with the standard CMS search order.

System Action: Execution continues with no access to public or project libraries.

User Action: If access to public or project libraries is required, (1) terminate the VS APL session, (2) use CMS and CP commands to access the disk that contains the APLIBTAB file, and (3) restart VS APL.

APL006I ERROR *x* ON FSREAD OF LIB TABLE FILE. NO PUBLIC/PROJECT LIBS.

Explanation: During initialization, an error occurred while reading the library table file (filename and filetype are APLIBTAB). *x* is the return code from the CMS FSREAD macro (see *IBM Virtual Machine Facility/370: CMS Command and Macro Reference*).

System Action: Execution continues with no access to public or project libraries.

User Action: Report the problem to the system administrator.

APL007I SYNTAX ERROR *x* ON CARD *y* OF LIB TABLE FILE. CARD IGNORED.

Explanation: There is a syntax error in one of the cards in the library table file (filename and filetype are APLIBTAB); *x* is the error code:

- 1—Card is blank.
- 2—First two letters of keyword are neither PU nor PR.
- 3—Library number is omitted.
- 4—Syntax of library number or range of library numbers is invalid.
- 5—Second library number of range is less than first.
- 6—On PRoject card, userid and disk address are omitted.
- 7—On PRoject card, disk address is omitted.
- 8—On PRoject card, extra field follows disk address.
- 9—On PUblic card, extra field follows library numbers or filemode.

y is the position of the card in the file relative to 1.

System Action: Card in error is ignored. Execution continues.

User Action: Report the problem to the system administrator.

APL008E APL ABEND DURING INITIALIZATION.

Explanation: During initialization, an abend was forced either by VS APL (user abend) or by CMS (system abend). A prior message explains the reason for the abend. In message DMSABN155T issued by CMS, the meaning of the reason code is:

- 1—Internal error; prior message explains error.
- 2—Internal error; unable to print prior message.
- 1*xx*—Program check; *xx* is program check code in decimal.

System Action: Message APL103D is issued to user to permit an optional dump of virtual storage prior to exiting from APL.

User Action: Report the problem to the system administrator. Include the terminal printout and the dump.

APL009I ERROR RETURN FROM STAE MACRO DURING INITIALIZATION. NO STAE EXIT.

Explanation: During initialization, an error occurred when an attempt was made to establish an abend exit using the STAE macro.

System Action: Execution continues with no STAE exit. Subsequent abends will be handled incorrectly.

User Action: Report the problem to the system administrator.

APL010I VIRTUAL MACHINE AND APL SHARED SYSTEM OVERLAP. APL ABORTED.

Explanation: During startup, VS APL discovered that the shared virtual storage to be used for the VS APL processor will overlap the addressability of the user's virtual machine. This could cause shared VS APL code to overlay CMS control blocks.

System action: VS APL terminates without loading the shared virtual storage.

User action: Define a smaller virtual machine using the CP DEFINE STORAGE command. If the problem persists, see the system administrator.

APL011I ERROR *x* LOADING APL MODULE. APL ABORTED.

Explanation: During initialization, an attempt was made to obtain addressibility to shared virtual storage for the VS APL Processor. The CP FINDSYS or LOADSYS request failed with condition 2 and return code *x* (see *IBM Virtual Machine Facility/370: System Programmer's Guide*).

System Action: Execution is terminated with an abend.

User Action: Report the problem to the system administrator.

APL012I AP NAME *'xxxxxx'* INVALID OR TEXT FILE NOT FOUND.

Explanation: Either (1) an auxiliary processor name in the APL command is invalid (first character is not alphabetic), or (2) a file with the specified name and a filetype of TEXT was not found with the standard CMS search order.

System Action: Execution is terminated.

User Action: Use the CMS LISTFILE command to obtain the names of text files. Type the APL command with the correct auxiliary processor name.

APL013I ERROR *x* IN DMSFRET DURING YYOFF, RC=*y*.

Explanation: During shutdown, while virtual storage was being freed, an error return from the CMS DMSFRET macro occurred; *x* is the request that failed:

- 1—DMSFRET of area that includes workspace, shared memory, and auxiliary processor work areas.
- 2—DMSFRET of executor's global table.

y is the return code (see *IBM Virtual Machine Facility/370: System Programmer's Guide*).

System Action: Execution is terminated with an abend.

User Action: Report the problem to the system administrator.

APL014I ERROR *x* FROM DMSFRET RETURNING STORAGE.

Explanation: During initialization, while part of available virtual storage was being returned to CMS, an error return from the CMS DMSFRET macro occurred; *x* is the return code (see *IBM Virtual Machine Facility/370: System Programmer's Guide*).

System Action: Execution is terminated with an abend.

User Action: Use the CP DEFINE STORAGE command to increase the size of the virtual machine; then restart VS APL. If the problem persists, report it to the system administrator.

APL015I INSUFFICIENT STORAGE FOR MINIMUM-SIZED WORKSPACE.

Explanation: After storage is allocated for the executor's global table, shared memory, and auxiliary processors, the remaining storage is allocated to the workspace. The resulting workspace size is less than the minimum allowed.

System Action: Execution is terminated.

User Action: Either (1) use the CP DEFINE STORAGE command to increase the size of the virtual machine; (2) if a shared memory size was specified in the APL command, specify a smaller value; or (3) if auxiliary processors were named in the APL command, specify fewer or no names.

APL016I NO BALANCING RIGHT PARENTHESIS ON SUBLIST.

Explanation: In the APL command, the right parenthesis that delimits a sublist is omitted.

System Action: Execution is terminated.

User Action: Type the APL command correctly.

APL017I LEFT PARENTHESIS WITHIN SUBLIST IS ILLEGAL.

Explanation: In the APL command, a left parenthesis occurs within a sublist.

System Action: Execution is terminated.

User Action: Type the APL command correctly.

APL018I PROGRAM CHECK (CODE *x*) AT LOCATION *yyyyyy* DURING INITIALIZATION.

Explanation: During initialization, a program check occurred in the VS APL Processor, in the Shared Storage Manager, or in an auxiliary processor; *x* is the program interrupt code from the program old PSW; *yyyyyy* is the address from the program old PSW.

System Action: Execution is terminated with an abend.

User Action: Report the problem to the system administrator.

APL019I ERROR IN SPECIFIED SHARED MEMORY SIZE.

Explanation: In the APL command, the shared memory size is either (1) greater than 16 megabytes, or (2) requested in units other than K or M.

System Action: Execution is terminated.

User Action: Type the APL command correctly.

APL020I ERROR CODE *x* FROM SSM DURING INITIALIZATION. NO SHARED VARS.

Explanation: During initialization, an error return from the Shared Storage Manager occurred; *x* is the return code:

- 4—Invalid parameters passed to Shared Storage Manager from module APLSCINI.
- 8—Error in an auxiliary processor.

System Action: Execution continues with no shared variable facility.

User Action: Report the problem to the system administrator.

APL021I APL ASSIST INCOMPATIBLE WITH APL PROCESSOR. ASSIST NOT USED.

Explanation: During initialization, when a test was made for the presence of the VS APL Microcode Assist, it was discovered that the Microcode Assist is not at the same level as the APL Processor.

System Action: Execution continues without use of the VS APL Microcode Assist.

User Action: Report the problem to the system administrator.

APL022I NAME OR ENTRY POINT FOR AUX PROCESSOR *'xxxxxx'* IS DUPLICATED.

Explanation: In the APL command, the noted auxiliary processor either (1) is named more than once, or (2) has a different name but the same entry point address as another specified auxiliary processor.

System Action: Execution continues.

User Action: Type the APL command with no duplicate or synonymous auxiliary processor names. Note that your installation may have built your APL system with some distributed auxiliary processors included, in which case you should not explicitly name them in the APL command.

APL023I CANNOT LOAD AP—A-DISK IS NOT READ/WRITE

Explanation: The user named an auxiliary processor in the APL startup command, however its TEXT file cannot be loaded because a temporary file required by the CMS loader cannot be written on the user's A-disk.

System Action: Execution is terminated.

User Action: Use the CMS ACCESS command to gain READ/WRITE access to an A-disk, or restart APL without specifying an auxiliary processor.

APL101I PGM INTERRUPT LOOP IN APL PROCESSOR.

Explanation: During processing of a program check, a second program check occurred.

System Action: Messages APL104I, APL105I, and APL106I, which show the PSW and general registers at the time of the second program check, are issued. Execution is terminated with an abend.

User Action: Report the problem to the system administrator.

APL102I PGM INTERRUPT IN EXECUTOR.

Explanation: A program check occurred in the VS APL Executor.

System Action: Messages APL104I, APL105I, and APL106I, which show the PSW and general registers at the time of the program check, are issued. Execution is terminated with an abend.

User Action: Report the problem to the system administrator.

APL103D TYPE 'DUMP 0-END' FOR DUMP, OR 'BEGIN' TO CONTINUE

Explanation: VS APL has abended. At the user's option, a dump of virtual storage will be sent to the user's virtual printer.

System Action: The virtual machine is stopped with an address-stop, which allows the user to enter CP commands.

User Action: To take a dump and continue abend processing, enter the following CP commands, in the order indicated:

```
DUMP 0-END

CLOSE PRINTER

BEGIN
```

To continue abend processing with no dump, enter:

```
BEGIN
```

Report the problem to the system administrator. Include the terminal printout and the dump, if one was taken.

APL104I PSW= *xxxxxxxx xxxxxxxx*

Explanation: VS APL is about to abend because of a program check. The message shows the contents of the program check old PSW.

System Action: Abend processing continues.

User Action: Include this information when reporting the problem to the system administrator.

APL105I R0-7= *xxx*

Explanation: VS APL is about to abend because of a program check. The message shows the contents of general registers 0 through 7 at the time of the program check.

System Action: Abend processing continues.

User Action: Include this information when reporting the problem to the system administrator.

APL106I R8-15= *xxx*

Explanation: VS APL is about to abend because of a program check. The message shows the contents of general registers 8 through 15 at the time of the program check.

System Action: Abend processing continues.

User Action: Include this information when reporting the problem to the system administrator.

APL107I SAVE AREA OVERFLOW. CALLEE=*xxxxxx*, CALLER=*yyyyyy*.

Explanation: The executor is designed so that routine calls are nested to a maximum depth of three. An attempt to call a fourth-level routine was made, and there is no fourth register save area; *xxxxxx* is the address of the called entry point; *yyyyyy* is the address of the instruction following the call.

System Action: Execution is terminated with an abend.

User Action: Report the problem to the system administrator.

APL108I SYSTEM ERROR IN APL PROCESSOR.

Explanation: VS APL encountered an internal error. The error will not terminate processing.

System Action: Messages APL109I and APL110D are printed. System error processing continues, and the active workspace is cleared.

User Action: Report the problem to the system administrator.

APL109I TYPE 'BEGIN *xxxxxx*' TO TAKE WORKSPACE DUMP ON PRINTER.

Explanation: VS APL encountered an internal error. The error will not terminate processing. At the user's option, a dump of the workspace and perterm header will be sent to the user's virtual printer.

System Action: Message APL110D is issued.

User Action: See message APL110D.

APL110D TYPE 'BEGIN' TO SKIP WORKSPACE DUMP.

Explanation: VS APL encountered an internal error. The error will not terminate processing. At the user's option, a dump of the workspace and perterm header will be sent to the user's virtual printer.

System Action: The virtual machine is stopped with an address-stop, which allows the user to enter CP commands. When control is returned following the CP BEGIN command, (1) diagnostic information is printed at the terminal, (2) a clear workspace is loaded, and (3) execution continues.

User Action: If a dump is wanted, type the CP command noted in message APL109I. If a dump is not wanted, type the CP command noted in message APL110D.

APL111I WORKSPACE AND PERTERM DUMP NUMBER *x*

Explanation: In response to message APL109I, the user requested that a dump be printed. This message appears as a heading in the dump; *x* is the dump number corresponding to the one that appears in the terminal printout.

System Action: System error processing continues.

User Action: None.

APL112I THIS IS DUMP OF ACTIVE WORKSPACE AREA

Explanation: In response to message APL109I, the user requested that a dump be printed. This message appears in the dump preceding the printout of the active workspace.

System Action: System error processing continues.

User Action: None.

APL113I THIS IS A DUMP OF THE PERTERM HEADER

Explanation: In response to message APL109I, the user requested that a dump be printed. This message appears in the dump preceding the printout of the Perterm Header.

System Action: System error processing continues.

User Action: None.

APL114I PROGRAM INTERRUPT IN SSM OR AUXILIARY PROCESSOR.

Explanation: A program interrupt occurred in the Shared Storage Manager or in an auxiliary processor. An APL processor system error is simulated and the program check old PSW and registers are printed.

System Action: Messages APL109I and APL110D, which allow the user to request a dump, are issued.

User Action: Report the problem to the system administrator.

APL115I APL HAS ABENDED.

Explanation: VS APL is about to abend. A prior message explains the reason for the abend.

System Action: Message APL103D, which allows the user to request a dump, is issued.

User Action: Report the problem to the system administrator.

APL151I UNKNOWN RETURN CODE FROM CMS, OP=*x*, RC=*y*.

Explanation: An unknown error return from a CMS operation or a CP LINK command occurred; *x* is the request that failed:

- 1—FSWRITE of main part of workspace file during YYSAVE.
- 2—FSWRITE of first record of workspace file during YYSAVE.
- 4—FSERASE of workspace file during YYDROP.
- 5—FSREAD of first of workspace file during YYLOAD.
- 6—FSREAD of main part of workspace file during YYLOAD.
- 7—CP LINK to project disk during YYDROP, YYLIB, YYSAVE, or YYLOAD.
- 8—Simulated CMS ACCESS command.
- 9—GETFST to test for existence of workspace file during YYSAVE.

y is the return code (see *IBM Virtual Machine Facility/370: CMS Command and Macro Reference*).

System Action: Execution is terminated with an abend.

User Action: Report the problem to the system administrator.

APL152I CMS FILE ERROR, OP=*x*, RC=*y*.

Explanation: An error return from a CMS operation or a CP LINK command occurred; *x* is the request that failed:

- 1—FSWRITE of workspace or temporary workspace during YYSAVE.
- 2—FSERASE of old workspace file after writing a temporary workspace file during YYSAVE.
- 3—Rename of temporary workspace file to workspace file during YYSAVE.
- 4—FSERASE of workspace file during YYDROP.
- 5—FSREAD of first record of workspace file during YYLOAD.
- 6—FSREAD of main part of workspace file during YYLOAD.
- 7—CP LINK to project disk during YYDROP, YYLIB, YYSAVE, or YYLOAD.
- 8—Simulated CMS ACCESS command.

y is the return code (see *IBM Virtual Machine Facility/370: CMS Command and Macro Reference*).

System Action: Execution is terminated with an abend.

User Action: Report the problem to the system administrator.

APL153I ERROR MSG DMKLNK*nnn*X FROM LINK COMMAND, RC=*nnn*.

Explanation: While a project library was being accessed, an error return from the CP LINK command occurred. *nnn* is the return code and CP error message number.

System Action: Message LIBRARY NOT AVAILABLE is issued. Execution continues with no access to the project library.

User Action: Report the problem to the system administrator.

APL154I RETURN CODE *x* FROM CMS 'ACCESS' COMMAND,RC=*x*.

Explanation: While a project library was being accessed, an error return from the CMS ACCESS command occurred; *x* is the return code (see *IBM Virtual Machine Facility/370: CMS Command and Macro Reference*).

System Action: If the return code is 24 (invalid parameters), APL will terminate with an abend. If the return code is 100 (disk 197 not found), execution continues but project library is not accessed.

User Action: Report the problem to the system administrator.

APL155I ERROR *x* FROM DMSFREE FOR)LIB NAME TABLE (RC=*x*). REQUEST IGNORED.

Explanation: When a)LIB command is processed, virtual storage for a table of workspace names is obtained. The CMS DMSFREE macro issued to obtain the space failed; *x* is the return code (see *IBM Virtual Machine Facility/370: System Programmer's Guide*). The problem may be due to the presence of an auxiliary processor that has obtained a large amount of space.

System Action: The command is rejected. Execution continues.

User Action: Issue the)LIB command when no auxiliary processors are accessed. If the problem persists, report it to the system administrator.

APL156I ERROR *x* FROM FST LOOKUP DURING YYLIB (RC=*x*).

Explanation: While the File Status Table (FST) entries for workspace files were being obtained, an error occurred; *x* is the return code from the FST lookup routine FSTLKP.

System Action: Execution is terminated with an abend.

User Action: Report the problem to the system administrator.

APL157I ERROR *x* FROM DMSFRET FOR)LIB NAME TABLE (RC=*x*).

Explanation: When a)LIB command is processed, virtual storage for a table of workspace names is obtained. The CMS DMSFRET macro issued to free the space failed; *x* is the return code (see *IBM Virtual Machine Facility/370: System Programmer's Guide*).

System Action: Execution is terminated with an abend.

User Action: Report the problem to the system administrator.

APL158I INTERNAL ERROR *x* LOADING WS FILE (OP=*x*).

Explanation: While a)LOAD or)COPY command was being processed, the workspace file was found to be invalid. *x* is the error code:

- 1—Logical record length is not 800.
- 2—Workspace check word (defined in WSMCHECK in the workspace header) is incorrect.
- 3—The size of the workspace, as indicated in WSMFREEA, is larger than the size indicated by the number of records in the file.

System Action: The command is rejected. Message WS NOT FOUND is issued. Execution continues.

User Action: If error code 1 or 3, use the CMS LISTFILE command with the DATE option to determine the record length and size of the file. Report the problem to the system administrator.

APL159I FILE '*x* APLTMPWS' ALREADY EXISTS. WS NOT SAVED.

Explanation: Due to a failure during a prior save command, the temporary workspace file was not renamed to become the new copy of the workspace. The old workspace file may also exist. The description of the)*SAVE* command contains an explanation of the use of temporary files.

System Action: The active workspace is not saved. The workspace and temporary files are not changed.

User Action: Use the)*DROP* command to erase the old workspace file. This will also cause the temporary file to be erased. CMS commands may also be used to erase or rename the files.

APL201I ERROR DURING COPY, OP=*x*, RC=*y*.

Explanation: While a)COPY or)PCOPY command was being processed, a file operation failed; *x* is the operation that failed:

- 1—FSWRITE of COPYDATA file during YYCOPO.
- 2—FSREAD of COPYDATA file during YYCOPI.
- 3—FSWRITE of record 1 of part A of SINKWS file during YYCOPA.
- 4—FSWRITE of rest of part A of SINKWS file during YYCOPA.
- 5—FSWRITE of part B of SINKWS file during YYCOPA.
- 6—YYCOPZ issued and either (1) both copy source and copy sink status are indicated, or (2) neither copy source nor copy sink status is indicated.
- 7—FSCLOSE of SINKWS file during YYCOPA.
- 8—FSCLOSE of COPYDATA file during YYCOPZ for source.
- 9—FSREAD of record 1 of part A of SINKWS file.
- 10—FSREAD of rest of part A of SINKWS file.
- 11—FSREAD of part B of SINKWS file.

y is the return code from CMS macro (see *IBM Virtual Machine Facility/370: CMS Command and Macro Reference*).

System Action: If operation that failed is code 2, 6, 9, 10, or 11, execution is terminated with an abend. If operation that failed is code 1, 3, 4, 5, 7, or 8, the command is rejected with the message NO OBJECTS COPIED; execution continues.

User Action: Report the problem to the system administrator.

Auxiliary Processor Messages

The following message is reported when an auxiliary processor distributed with VS APL receives an unexpected return or reason code as the result of a shared variable operation:

id ABENDED AT *address,* RETURN CODE IS *code*$_1$,
REASON CODE IS *code*$_2$

where:

id
identifies the auxiliary processor that encountered the unexpected code (AP100, AP101, AP110, AP111, and AP123 for auxiliary processor numbers 100, 101, 110, 111, and 123 respectively.)

address
is the address at which the unexpected return or reason code was discovered.

code$_1$
is the return code as described in the macro ASUSHSVP.

code$_2$
is the reason code as described in the macro ASUSHSVP.

In response to the unexpected code, the auxiliary processor terminates. You should immediately communicate the displayed information to your system administrator.

System Errors

A system error is reported any time during your terminal session an unrecoverable internal error is encountered by VS APL. The format of a system error report is:

time date SYSTEM ERROR *code*

where *code* indicates the reason for the error. The code displayed will have no specific meaning to you but is useful if communication with your system administrator is necessary.

APPENDIX F: VS APL BATCH PROCESSING

There may be occasions when you don't want your APL work to be done interactively. For instance, your work is excessively time consuming or a terminal is unavailable. At times like these you can submit your work to VS APL as a batch job using the CMS Batch Facility (CMSBATCH). While doing this, you have the option of continuing to use VS APL interactively for other APL work. This appendix illustrates the CMS Batch Facility as it pertains to VS APL. A detailed description of CMSBATCH is provided in the *CMS User's Guide.*

CMS Batch Facility Input

There are two ways to submit input to CMSBATCH:

- Provide real punched card input in the system card punch.
- Provide virtual card input by *spooling* (using the CMS SPOOL command to process virtual output) from your virtual machine to a CMSBATCH virtual card reader.

If you use real punched cards, the first card in the deck must be:

ID *batchid*

where *batchid* is the identification of CMSBATCH. (Get this from your system administrator.)

In either case, the input must include:

- Control statements for CMSBATCH.
- VM/370 control program segment (CP) or CMS commands to give CMSBATCH access to virtual disks required by VS APL.
- CMS statements to load and start VS APL (such as LOADMOD and START).
- VS APL input statements.

There are three CMSBATCH control statements: /JOB, /*, and /SET. /JOB identifies you to CMSBATCH; /* indicates the end of a CMSBATCH job; and /SET sets limits on a system's time, printing, and punching resources during execution of a CMSBATCH job. For further details on these statements, refer to the *CMS User's Guide.*

Which CP and CMS commands you use and what information you specify, depends on your installation's requirements; see your system administrator.

As an example of a VS APL job submitted to CMSBATCH, examine the following:

```
/JOB MYUSERID 1234 APLJOB
CP LINK MYUSERID 191 191 MW DISKPASS
ACCESS 191 B/A
APL APL100 APL101 APL110 APL111
```

```
)LOAD MYWS
FUNCTION1
)SAVE
)OFF HOLD
/*
```

Here, user MYUSERID uses CMSBATCH to load a workspace (MYWS) and execute a function (FUNCTION1). He then saves MYWS and returns to CMS. This job is designed to be submitted to CMSBATCH via spooling; notice there is no ID statement. You can prepare this job as a CMS file through the CMS Editor; you then spool the file to the CMSBATCH virtual card reader. One way to do this is shown below:

```
CP SPOOL PUNCH TO CMSBATCH NOCONT
PUNCH APLFILE EXEC (NOHEADER)
```

CMS Batch Facility Output

CMSBATCH output consists of all the input statements submitted and output information generated as part of the batch job. Unless you take special action, all CMSBATCH output is directed to the system printer.

To redirect CSMBATCH output to your terminal, do the following:

1. Spool the output to your virtual machine by issuing the CP command:

   CP SPOOL CONS *userid*

   where *userid* is your user identification. Once spooled to your virtual machine, the output file appears in your virtual card reader and has a filename of BATCH, a filetype of CONSOLE, and a reader class of T.

2. Read the file onto your A disk by entering the commands:

   ```
   CP SPOOL RDR CLASS T
   READ CARD BATCH CONSOLE
   CP SPOOL RDR CLASS A
   ```

3. Display all or part of the file at your terminal by issuing the CMS TYPE command or by using the CMS Edit Facility.

VS APL Restrictions

The following restrictions apply to CMSBATCH jobs using VS APL:

- The CP/CMS Command Processor, auxiliary processor 100, cannot be used to submit CP or CMS commands that are prohibited by CMSBATCH. Refer to the *CMS User's Guide* for a discussion of CP and CMS command restrictions.

- Input statements may not include APL-only characters (such as ⍳ and ⍴). These characters cannot be properly translated by CMSBATCH.

- If character input (⍞ input) follows bare output (⍞ output), the character input must contain leading blanks to skip over the bare output.

- If a password is required in a VS APL system command that accesses a password-protected project library, the password must be specified; CMSBATCH will not prompt for the password if it is missing.

APPENDIX G: CHARACTER TRANSLATION

EBCDIC		VS	APL		EBCDIC		VS	APL	
DEC	HEX	DEC	HEX		DEC	HEX	DEC	HEX	
0	00	255	FF		50	32	241	F1	
1	01	0	00		51	33	242	F2	
2	02	1	01		52	34	243	F3	
3	03	2	02		53	35	244	F4	
4	04	3	03		54	36	245	F5	
5	05	4	04		55	37	246	F6	
6	06	5	05		56	38	247	F7	
7	07	6	06		57	39	248	F8	
8	08	7	07		58	3A	249	F9	
9	09	203	CB		59	3B	250	FA	
10	0A	204	CC		60	3C	251	FB	
11	0B	205	CD		61	3D	252	FC	
12	0C	206	CE		62	3E	253	FD	
13	0D	207	CF		63	3F	254	FE	
14	0E	208	D0		64	40	64	40	Blank
15	0F	209	D1		65	41	92	5C	A
16	10	210	D2		66	42	93	5D	B
17	11	211	D3		67	43	94	5E	C
18	12	212	D4		68	44	95	5F	D
19	13	213	D5		69	45	96	60	E
20	14	214	D6		70	46	97	61	F
21	15	202	CA	New line	71	47	98	62	G
22	16	200	C8	Backspace	72	48	99	63	H
23	17	215	D7		73	49	100	64	I
24	18	216	D8		74	4A	22	16	
25	19	217	D9		75	4B	130	82	.
26	1A	218	DA		76	4C	158	9E	<
27	1B	219	DB		77	4D	194	C2	(
28	1C	220	DC		78	4E	144	90	+
29	1D	221	DD		79	4F	30	1E	
30	1E	222	DE		80	50	9	09	&
31	1F	223	DF		81	51	101	65	J
32	20	224	E0		82	52	102	66	K
33	21	225	E1		83	53	103	67	L
34	22	226	E2		84	54	104	68	M
35	23	227	E3		85	55	105	69	N
36	24	228	E4		86	56	106	6A	O
37	25	201	C9	Line feed	87	57	107	6B	P
38	26	229	E5		88	58	108	6C	Q
39	27	230	E6		89	59	109	6D	R
40	28	231	E7		90	5A	23	17	
41	29	232	E8		91	5B	37	25	
42	2A	233	E9		92	5C	148	94	*
43	2B	234	EA		93	5D	195	C3	)
44	2C	235	EB		94	5E	196	C4	;
45	2D	236	EC		95	5F	31	1F	
46	2E	237	ED		96	60	145	91	-
47	2F	238	EE		97	61	174	AE	/
48	30	239	EF		98	62	110	6E	S
49	31	240	F0		99	63	111	6F	T

Figure 31 (Part 1 of 3). APL/EBCDIC Conversion via 192 Conversion Option

EBCDIC		VS APL			EBCDIC		VS APL		
DEC	HEX	DEC	HEX		DEC	HEX	DEC	HEX	
100	64	112	70	U̲	150	96	52	34	o
101	65	113	71	V̲	151	97	53	35	p
102	66	114	72	W̲	152	98	54	36	q
103	67	115	73	X̲	153	99	55	37	r
104	68	116	74	Y̲	154	9A	142	8E	⊃
105	69	117	75	Z̲	155	9B	141	8D	⊂
106	6A	29	1D		156	9C	131	83	
107	6B	181	B5	,	157	9D	149	95	○
108	6C	10	0A	%	158	9E	12	0C	±
109	6D	133	85	_	159	9F	186	BA	←
110	6E	162	A2	>	160	A0	129	81	¯
111	6F	182	B6	?	161	A1	33	21	
112	70	14	0E		162	A2	56	38	s
113	71	154	9A	∧	163	A3	57	39	t
114	72	134	86	¨	164	A4	58	3A	u
115	73	15	0F		165	A5	59	3B	v
116	74	16	10		166	A6	60	3C	w
117	75	17	11		167	A7	61	3D	x
118	76	18	12		168	A8	62	3E	y
119	77	19	13		169	A9	63	3F	z
120	78	155	9B	∨	170	AA	139	8B	∩
121	79	34	22		171	AB	140	8C	∪
122	7A	197	C5	:	172	AC	169	A9	⊥
123	7B	36	24		173	AD	192	C0	[
124	7C	35	23		174	AE	161	A1	≥
125	7D	198	C6	'	175	AF	191	BF	∘
126	7E	160	A0	=	176	B0	137	89	⍺
127	7F	11	0B	"	177	B1	168	A8	∊
128	80	164	A4	~	178	B2	167	A7	⍳
129	81	38	26	a	179	B3	166	A6	⍴
130	82	39	27	b	180	B4	138	8A	⍵
131	83	40	28	c	181	B5	20	14	
132	84	41	29	d	182	B6	146	92	×
133	85	42	2A	e	183	B7	176	B0	\
134	86	43	2B	f	184	B8	147	93	÷
135	87	44	2C	g	185	B9	13	0D	
136	88	45	2D	h	186	BA	135	87	∇
137	89	46	2E	i	187	BB	91	5B	∆
138	8A	183	B7	↑	188	BC	170	AA	⊤
139	8B	184	B8	↓	189	BD	193	C1	]
140	8C	159	9F	≤	190	BE	163	A3	≠
141	8D	151	97	⌈	191	BF	153	99	|
142	8E	152	98	⌊	192	C0	24	18	
143	8F	185	B9	→	193	C1	65	41	A
144	90	187	BB	⎕	194	C2	66	42	B
145	91	47	2F	j	195	C3	67	43	C
146	92	48	30	k	196	C4	68	44	D
147	93	49	31	l	197	C5	69	45	E
148	94	50	32	m	198	C6	70	46	F
149	95	51	33	n	199	C7	71	47	G

Figure 31 (Part 2 of 3). APL/EBCDIC Conversion via 192 Conversion Option

EBCDIC		VS APL		
DEC	HEX	DEC	HEX	
200	*C*8	72	48	*H*
201	*C*9	73	49	*I*
202	*CA*	156	9*C*	⍲
203	*CB*	157	9*D*	⍱
204	*CC*	28	1*C*	
205	*CD*	171	*AB*	⌽
206	*CE*	27	1*B*	
207	*CF*	172	*AC*	⍉
208	*D*0	25	19	
209	*D*1	74	4*A*	*J*
210	*D*2	75	4*B*	*K*
211	*D*3	76	4*C*	*L*
212	*D*4	77	4*D*	*M*
213	*D*5	78	4*E*	*N*
214	*D*6	79	4*F*	*O*
215	*D*7	80	50	*P*
216	*D*8	81	51	*Q*
217	*D*9	82	52	*R*
218	*DA*	143	8*F*	⌶
219	*DB*	165	*A*5	!
220	*DC*	180	*B*4	⍒
221	*DD*	179	*B*3	⍋
222	*DE*	188	*BC*	⍞
223	*DF*	199	*C*7	⍝
224	*E*0	32	20	
225	*E*1	21	15	
226	*E*2	83	53	*S*
227	*E*3	84	54	*T*
228	*E*4	85	55	*U*
229	*E*5	86	56	*V*
230	*E*6	87	57	*W*
231	*E*7	88	58	*X*
232	*E*8	89	59	*Y*
233	*E*9	90	5*A*	*Z*
234	*EA*	175	*AF*	⌿
235	*EB*	177	*B*1	⍀
236	*EC*	26	1*A*	
237	*ED*	173	*AD*	⊖
238	*EE*	178	*B*2	⌹
239	*EF*	190	*BE*	⍕
240	*F*0	119	77	0
241	*F*1	120	78	1
242	*F*2	121	79	2
243	*F*3	122	7*A*	3
244	*F*4	123	7*B*	4
245	*F*5	124	7*C*	5
246	*F*6	125	7*D*	6
247	*F*7	126	7*E*	7
248	*F*8	127	7*F*	8
249	*F*9	128	80	9
250	*FA*	132	84	
251	*FB*	136	88	⍫
252	*FC*	118	76	⍙
253	*FD*	150	96	⍟
254	*FE*	189	*BD*	⍎
255	*FF*	8	08	

Figure 31 (Part 3 of 3). APL/EBCDIC Conversion via 192 Conversion Option

GLOSSARY

This glossary defines words and phrases used in the text. The definitions are primarily in terms of VS APL and its use; the definitions are not intended to apply generally to all of data processing.

auxiliary processor: A program that communicates with another program through shared variables.

clear workspace: A workspace in which no data has been entered, no names defined, and in which the workspace attributes indicate standard initial values.

CMS Disk I/O Processor: An auxiliary processor distributed with VS APL that can be used to read from or write to CMS disk files.

CMS file: (see *file*)

CMS file identification: (see *file identification*)

CMS VSAM Processor: An auxiliary processor distributed with VS APL that can be used to perform operations on entry-sequenced or key-sequenced VSAM files.

communication command: A system command used to communicate information to other terminals.

continued workspace: A workspace that is stored in a library via the)CONTINUE or)CONTINUE HOLD commands. A continued workspace has the name CONTINUE.

control variable: A shared variable used in auxiliary processing operations to monitor and in some instances control data transmission.

CP/CMS Command Processor: An auxiliary processor distributed with VS APL that can be used to dynamically enter a CP or CMS command.

data variable: A shared variable in CMS VSAM Processor operations used to transmit data.

distributed workspace: A public library workspace distributed with VS APL. Generally distributed workspaces are installed as part of libraries 1 or 2.

dynamic storage area: An area of the workspace in which data, function definitions, and group definitions are kept.

entry-sequenced file: A VSAM file whose records are organized sequentially. In general, as each record is written to the file, it is appended to the last record written. (Contrast with *key-sequenced file.*)

error report: A message produced by VS APL when it encounters an unexecutable statement.

execution control area: An area of the workspace used by VS APL to contain information generated during function execution and to contain the state indicator.

executor message: A message produced by the executor portion of VS APL. Executor messages are caused by a variety of error conditions such as erroneous library designations or initialization errors.

file: A logically related group of records defined via CMS commands. VS APL workspaces and libraries are treated as files by CMS.

file identification: The filename, filetype, and filemode that uniquely identify a CMS file.

FILEDEF I/O Processor: An auxiliary processor distributed with VS APL that can be used to read from or write to any device supported by VM/370.

filemode: The part of a CMS file identification that indicates the disk on which the file resides. VS APL workspaces stored in private libraries are associated with a filemode of A, project library workspaces are associated with a filemode of G, while public library workspaces are associated with a filemode of Y.

filename: The part of a CMS file identification that names the file. VS APL workspaces are associated with filenames that are the same as their workspace names.

filetype: The part of a CMS file identification that indicates the function of a file. VS APL workspaces stored in private libraries are associated with a filetype of VSAPLWS, while project and public library workspaces are associated with a filetype composed of the letter W followed by a seven-digit library number.

forced ending: An abrupt break of your link to the computer caused by machine malfunctions such as failures in the telephone circuits.

inquiry command: A system command that returns information about the contents and attributes of the active workspace and libraries.

interlock: An inoperative state in which the terminal is locked due to an access control violation during auxiliary processing.

key-sequenced file: A VSAM file whose records are organized according to the value of a key field contained in each record. (Contrast with *entry-sequenced file.*)

library control command: A system command used to control the contents of libraries.

logoff: The procedure by which you end contact with VM/370.

logon: The procedure by which you establish contact with VM/370.

message: A communication sent by the system in response to an entry, or an indication of an encountered or impending problem. A line of text sent from one terminal to another. (see also *error report, trouble report,* and *executor message*)

message blocking: The act of halting incoming terminal messages. You can instruct the system to block messages through the system command)MSG OFF.

print suppression: A feature of certain terminals that suppresses the printing of sensitive information such as your logon password.

private library: A set of stored workspaces available to a specific VS APL user.

project library: A set of stored workspaces that can be shared among a selected set of VS APL users.

public library: A set of stored workspaces available to all VS APL users.

record variable: A shared variable used in auxiliary processor operations to transmit data.

return code: A value placed in the record or control variable that indicates the acceptance or rejection of an auxiliary processing request.

service request: A character vector that indicates a file operation to be performed by the CMS VSAM Processor.

sign off: The procedure by which you end contact with VS APL.

sign-off command: A system command used to sign off the system.

sign on: The procedure by which you establish contact with VS APL.

spooling: A CMS operation that directs output files to auxiliary storage devices in preparation for later input/output operations.

Stack Input Processor: An auxiliary processor distributed with VS APL that can be used to create a stack of data for subsequent input to CMS.

state indicator: An object in the workspace that tracks the progress of executing defined functions. If function execution is halted, the state indicator shows all the halted functions in order, the most recently active function first, and the line in each function that is in the process of being, or the next to be, executed.

symbol table: An area of the workspace used by VS APL to keep track of names.

system administrator: The person who is responsible for the system and who assists you in your use of the system.

system command: (see *VS APL system command*)

system error: An internal error condition detected by VS APL.

system error report: A message indicating that a system error has been encountered by VS APL. The message indicates the time and date the error was encountered and a code representing the nature of the error.

trouble report: A message produced by VS APL when it encounters an unexecutable statement.

virtual computer: (see *virtual machine*)

virtual disk: The virtual equivalent of a real disk. Each disk is identified by a letter and an address.

virtual machine: The functional equivalent of an IBM System/370 computing system.

VM/370 Operator: The VM/370 user who is designated to receive VM/370 operator messages.

VS APL system command: An instruction to VS APL that allows you to monitor and control the contents of workspaces and libraries.

work session: The time elapsed between your VM/370 logon and logoff.

workspace control command: A system command used to control the content and attributes of the active workspace.

workspace identification: The library number and workspace name that uniquely identify a workspace.

INDEX

Where more than one page reference is given, the major reference is first.

SH20-9067-2

VS APL f S: Terminal User's Guide Printed in U.S.A. SH20-9067-2

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