GH20-9064-4

Program Product

VS APL General Information

IBM

GH20-9064-4

Program Product

VS APL General Information

Program Numbers 5748-AP1 (VS APL)
5740-XR9 (VS TSIO)

IBM

Third Edition (June 1978)

This edition replaces the previous edition (numbered GH20-9064-2) and technical newsletter (GN20-9184) and makes that edition and technical newsletter obsolete.

This edition applies both to Release 3 of VS APL and Release 1 of VS TSIO, and to any subsequent releases of either program product 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 as of this publishing date are indicated by a vertical bar to the left of the text; these bars will be deleted at any subsequent republication of the page affected. Editorial changes that have no technical significance are not noted.

Information in this publication is subject to significant change. Any such changes will be published in new editions or technical newsletters. Before using the publication, consult the latest *IBM System/370 Bibliography,* GC20-0001, and the technical newsletters that amend the bibliography, to learn which editions and technical newsletters are applicable and current.

Requests for copies of IBM publications should be made to the IBM branch office that serves you.

Forms for readers' comments are provided at the back of the publication. If the forms have been removed, comments may be addressed to IBM Corporation, P. O. Box 50020, Programming Publishing, San Jose, California 95150. All comments and suggestions become the property of IBM.

PREFACE

This manual introduces the VS APL program product and the VS TSIO program product, and can be used as an aid in evaluating and planning for their use. The material is directed primarily to data processing management, and to system and application designers. Some chapters of this manual are also directed to other specific groups of data processing personnel as indicated below.

This publication has the following major divisions:

- "VS APL Overview" highlights the features and operation of VS APL, provides a general description of the APL language, and illustrates a VS APL commercial application.
- "VS APL Auxiliary Processors" describes how to gain access to special host-dependent services by exchanging information with another program.
- "Installation Planning for VS APL" describes the VS APL processor's requirements and storage estimates. This chapter is for people who plan for and install VS APL.
- "VS APL and Other IBM APL Implementations" describes the major differences between VS APL and APL\360. This chapter also provides information about how to convert from APL\360, APLSV, and APL/CMS to VS APL.
- "VS TSIO Overview and Installation Planning" highlights the features of VS TSIO and describes its operating requirements.
- "Supporting User Publications" describes the IBM publications that give additional information on VS APL.
- "Appendix A. Summary of APL Language Elements"
- "Appendix B. Summary of System Commands"
- "Index"

This publication refers to other IBM publications that contain related information:

- *VS Personal Computing (VSPC) for OS/VS and DOS/VS: General Information,* GH20-9070, describes the system configurations required and supported when installing VS APL under VSPC in OS/VS1, OS/VS2, or DOS/VS.
- *VM/370: Planning and System Generation Guide,* GC20-1801, describes the requirements for installing VM/370, which is a prerequisite to installing VS APL under the VM/370 Conversational Monitor System (CMS).
- *VM/370: CMS Command and Macro Reference,* GC20-1818, contains all reference information concerning CMS commands available to the VS APL user through auxiliary processors.
- *VM/370: CP Command Reference for General Users,* GC20-1820, contains all reference information on the VM/370 control program commands available to the VS APL user through auxiliary processors.
- *VM/370: CMS User's Guide,* GC20-1819, describes the basic use of CMS and the CMS file system.

- *Customer Information Control System/Virtual Storage (CICS/VS) Version 1, Release 3 General Information*, GC33-0066, describes the system configuration required and supported when installing CICS/VS.
- *Customer Information Control System/Virtual Storage (CICS/VS) Version 1, Release 3 Application Programmer's Reference (Command Level)*, SC33-0077, describes command syntax for CICS/VS commands referenced in this manual.

CONTENTS

FIGURES

SUMMARY OF AMENDMENTS

VS APL Release 3

VS APL under CICS/VS

Specification Change

VS APL can be run under CICS/VS. Several new auxiliary processors and distributed workspaces are available for use with VS APL under CICS/VS. Information has been added throughout this publication to reflect this support and to describe the new auxiliary processors and distributed workspaces. System configuration requirements and storage estimates for planning purposes are also included.

Storage Estimates for VS APL under VSPC

Service Change

The storage estimates for VS APL under VSPC have been updated.

Microcoded APL Assist Feature

Service Change

The list of System/370 models on which the APL Assist feature is standard or optional has been updated.

VS TSIO Release 1

VS TSIO

New Program Product

Information has been added to the publication describing VS TSIO, a new program product that operates with VS APL under VSPC using OS/VS1 or OS/VS2. VS TSIO includes an auxiliary processor which provides the VS APL user with access to OS/VS data sets supported by the BSAM, BDAM, and BPAM access methods. VS APL workspaces are also distributed as part of the program product; these workspaces assist in the use and operation of the VS TSIO auxiliary processor. VS TSIO is described in the chapter "VS TSIO Overview and Installation Planning."

VS APL Release 2

Full Screen Management Auxiliary Processor

New Programming Feature

VS APL under VSPC now includes the Full Screen Management Auxiliary Processor which provides for control of an IBM 3270 display device through a user application. A description of the auxiliary processor has been added to the chapter "VS APL Auxiliary Processors."

CMS VSAM Auxiliary Processor

New Programming Feature

VS APL under CMS now includes the CMS VSAM Auxiliary Processor which is used to perform file operations on entry-sequenced or key-sequenced VSAM files. A description of the auxiliary processor has been added to the chapter "VS APL Auxiliary Processors." The virtual machine requirements for VS APL under CMS have been changed to allow for the new auxiliary processor.

New Distributed Workspaces

New Programming Feature

Seven new workspaces are distributed with VS APL: HOWEDITS, SEDIT, MEDIT, SBIC, VSAPLFILE (VSPC only), FULLSCREEN (VSPC only) and PRINT (CMS only). A brief description of each workspace has been added to the section "Product Description" in the "VS APL Overview" chapter.

Supporting User Publications

Service Change

A new chapter entitled "Supporting User Publications" has been added to the text. The chapter lists and describes the IBM publications that give additional information on VS APL.

VS APL OVERVIEW

This chapter provides an overview of VS APL, a general description of the APL language, and an example of a VS APL application.

Highlights

VS APL is a program product that interprets the APL language and executes system commands and system functions to control the APL environment. VS APL is designed to operate with the VM/370 Conversational Monitor System (CMS), with VS Personal Computing (VSPC), or with Customer Information Control System/Virtual Storage (CICS/VS). VS APL can also operate as a batch processor under CMS using the CMS Batch Facility.

Use of VS APL gives installations a number of distinct programming and device features:

- It provides the language and enhancements of previous IBM APL implementations:

  –APL\360-OS
  –APL\360-DOS
  –APLSV
  –APL/CMS

- It allows users to communicate with programs called *auxiliary processors* outside the APL workspace. Installations can write their own processors or acquire them to provide specialized services for their users. Several auxiliary processors are included with VS APL. These processors provide services such as access to auxiliary storage files and are described in the chapter "VS APL Auxiliary Processors." The VS TSIO program product includes an auxiliary processor designed to operate with VS APL under VSPC using OS/VS1 or OS/VS2; it is described in the chapter "VS TSIO Overview and Planning."

- It supports, in conjunction with the host operating systems, new IBM terminals and I/O devices for which use is appropriate with APL. Additionally, installations that use leased-line terminals connected via the Virtual Telecommunications Access Method (VTAM) in other applications can now conveniently use these same terminals with VS APL.

- It affords an installation flexibility in the design of its APL applications. This is possible because VS APL is implemented in several host systems.

- It uses the APL Assist feature on certain System/370 models; use of the APL Assist feature can provide performance improvement in interpreting and executing APL statements. The chapter "Installation Planning for VS APL" gives specific System/370 models on which the feature is available.

- When used under CICS/VS, it allows a CICS/VS application program to use APL to do some of the processing without the terminal user initiating APL as a transaction.

Operation

Figure 1 relates VS APL to other parts of the system. The left-hand portion of the figure shows the terminals through which users enter APL statements. The right-hand portion of the figure shows VS APL, which runs under VSPC, CMS, or CICS/VS. The shared storage manager enables the user to gain access to auxiliary processors.

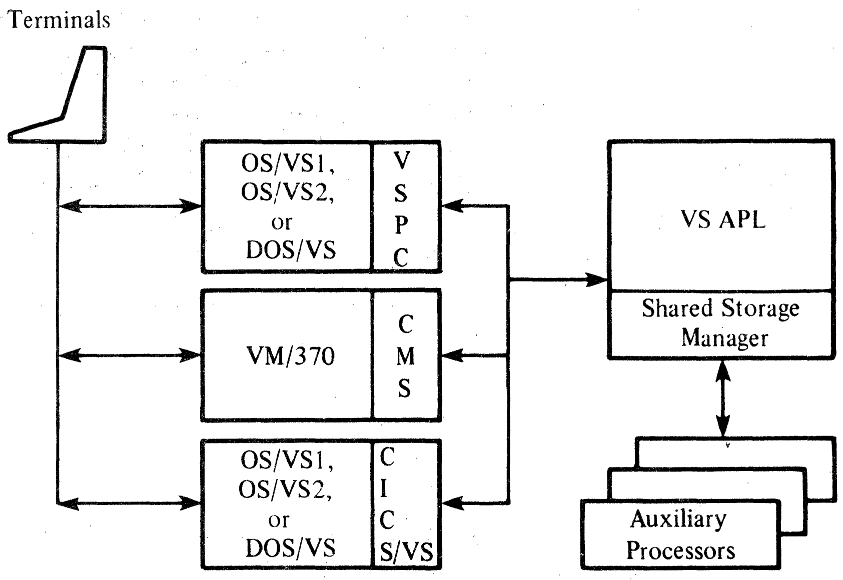

Figure 1. VS APL and Its Environment

VS APL is conversational. Users have the option of interacting with VS APL in either of two modes:

- Execution (or desk-calculator) mode, in which any statement the user enters is acted upon immediately.
- Definition mode, in which the user defines a function, which may consist of many statements, to be executed later.

The user can readily switch back and forth between modes, as required.

The APL character set consists of uppercase alphabetic characters, numerals, and APL special characters. The keyboard positions of some of these characters differ from conventional keyboards. The numerals, alphabetic characters, comma, and period appear in their standard keyboard positions. The uppercase shift is used to enter most APL special characters.

Terminals come in two varieties: typewriter-like devices, and display screen devices with attached keyboards. Figure 2 shows a typical APL keyboard for a display terminal.

Entry of APL special characters that are not available on the uppercase and lowercase keyboard varies depending on the terminal type; the procedures for entering these characters are in each of the VS APL user's guides.

A keyboard and display containing these special characters must be available if APL special characters are to be entered or displayed. Entering APL primitive functions, operators, and system variables from the keyboard requires special characters. Defined functions can be executed without using

APL special characters, if special characters are not part of the function name.

For most terminals supported by VS APL, the APL character set is available as a special feature.

As each new user is given access to VS APL, a unique profile is created to identify the user, the information he owns, and the information owned by others that he can access. The user is also given a *workspace.* A workspace is an area of storage that contains work in progress and can be thought of as a notebook in which the user can enter data, and define functions in his problem solving. A workspace can be saved for later use or erased. If the user saves the workspace for retrieval and use at a subsequent terminal session, it is put in a *library.* A library is simply a portion of storage that contains saved workspaces.

Libraries are identified as *private*, *public*, or *project* when they are created. A private library is generally available only to its owner. Project and public libraries are intended to be shared among users. Workspaces stored in a public library are available to all APL users. Workspaces stored in a project library are generally available to a limited group of APL users. Access to the library and workspaces varies slightly depending on whether VS APL is run under VSPC, CMS, or CICS/VS.

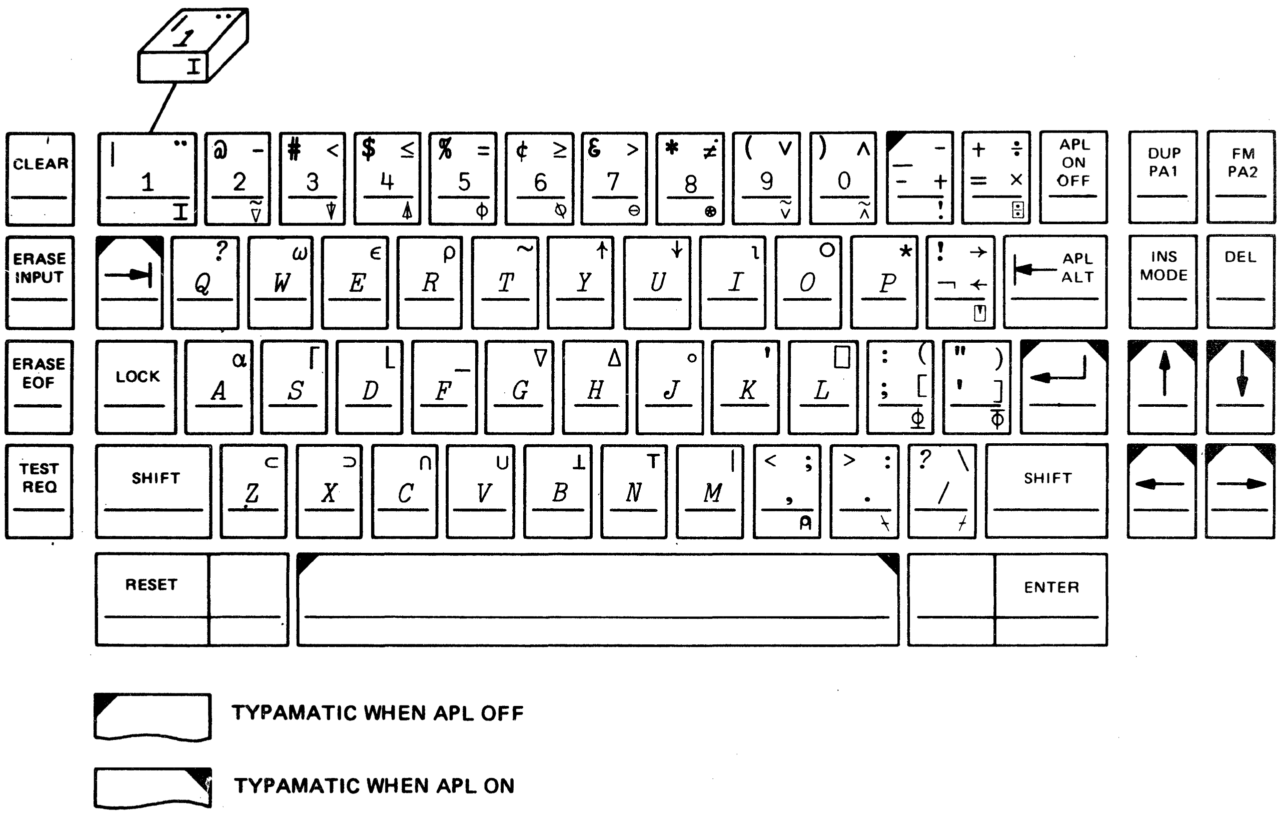

Figure 2. Typical APL Keyboard

In addition to the elements of the language, described under "Language" later in this chapter, VS APL recognizes system commands, system functions, and system variables. The user uses them in the following ways:

- System commands are used to send messages to other users, to save and retrieve workspaces, or to request reports about his work. System commands begin with a right parenthesis ") ".
- System functions and system variables are used to monitor and control the workspace environment. For example, one system variable is used to display the currently available area in the workspace. An important subclass of system functions comprises the shared-variable system functions which monitor and control the status of variables shared between a user and an auxiliary processor or another user. System functions and system variables begin with a quad "⎕".

Product Description

Installations that order the VS APL program product will receive:

- VS APL Interpreter—interprets and executes APL statements. The VS APL interpreter uses the APL microcode assist feature on certain System/370 models; the chapter "Installation Planning for VS APL" gives specific models on which the feature is available.
- Executors—one is provided for each environment in which VS APL can be installed: VSPC, CMS, or CICS/VS. The installation merely selects the executor required for its operating system.
- Auxiliary Processors— several are available for use under each environment in which VS APL can be installed. Again, the installation selects the auxiliary processors provided for use with its operating system. The functions of the processors are described in the chapter "VS APL Auxiliary Processors."
- Conversion Aids—facilitate migration of installations that wish to convert libraries and functions developed under other IBM APL implementations. The conversion aids are described in greater detail in the section "Conversion Aids."
- Distributed Workspaces—several workspaces containing functions that can be useful to terminal users are provided. The workspace names and brief descriptions follow:
  - ADMIN (CICS/VS only) provides functions to assist the APL administrator in maintaining, monitoring, and controlling the VS APL system. Using this workspace, the APL administrator can do operations such as define a new APL user, force a particular user to logoff, block a particular user from signing on to APL, modify library directory entries, list the information related to an APL library directory, list the current workspace names and internal file names for a user, delete a public library, user, or auxiliary processor, and display information about the status of the system.
  - APFNS (CMS only) makes the auxiliary processors provided for use under CMS easier to use.
  - 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.

- CONVERT assists in content conversion of workspaces developed under other IBM APL implementations.
- DLI (CICS/VS only) provides functions to assist the APL user in using the DL/I auxiliary processor distributed with VS APL.
- EXAMPLES illustrates APL coding techniques employing useful functions.
- FORMAT composes numeric output via a picture representation.
- FULLSCREEN (VSPC and CICS/VS only) assists in the use of the Full Screen Management Auxiliary Processor.
- HOWEDITS describes the SEDIT and MEDIT workspaces.
- MEDIT performs text editing operations; text is stored as a matrix.
- NEWS provides functions for the storage and display of bulletins to VS APL users.
- PLOT graphs values at the terminal.
- PRINT (CMS only) provides functions for the transmission of APL programs and data to an offline printer using an APL print train.
- QUEUE (CICS/VS only) provides functions to assist the VS APL user in using the Transient Data auxiliary processor distributed with VS APL.
- SBIC illustrates the use of APL in commercial data processing through a model set of sales, billing, and inventory control functions.
- SEDIT performs text editing operations; text is stored as a single string containing no blanks.
- TYPEDRILL tests the typing speed and accuracy of the terminal user.
- VSAPLFILE (VSPC and CICS/VS only) creates and manipulates files of APL arrays.
- VSFILES (VSPC and CICS/VS only) assists in the use of the APL Format, EBCDIC Format (VSPC only), and VSAM auxiliary processors.
- WSFNS assists in conversion to VS APL by providing comparable functions for certain system variables and system functions.

Language

APL represents a fresh approach to formulating and solving problems. Consistent rules and conversational interaction between user and computer make the language easy to learn and use. Nonprogrammers, such as businessmen, secretaries, financial analysts, educators, engineers, students, and mathematicians, can quickly become adept at using APL. Since its introduction, APL has grown in popularity as the result of increased efficiency in solving problems. Experiences of many users across the computer industry have demonstrated that productivity improvements can be obtained when programming in APL.

The elements of the APL language *(primitive functions, operators, system functions,* and *system variables)* include simple operations, such as addition, subtraction, multiplication, and division, as well as more specialized operations, such as reversing the order of a set of numbers or producing column totals from a table of quantities. The user can choose to use APL primitive functions or to define programs (called *defined functions*) of his own. He gradually learns to use more and more APL as the demands of his applications increase.

In this section, some basic APL concepts are introduced and explained through examples.

Primitive Functions

APL makes use of well-known mathematical functions (for example, +, -, ×, and ÷), relational functions used to compare values (for example, < for less than and ≠ for not equal to), and special symbols. One special symbol, the left arrow (←), is used for assigning a value. For example,

```
A←5
```

means assign the value 5 to the variable named A.

The rules for reading or writing an APL statement are simple. Expressions are executed from right to left. Expressions enclosed in parentheses are evaluated, as they are encountered, before the right-to-left execution of the statement proceeds. Thus, the value of C is 45 in:

```
C←(5+4)×2+3
```

APL functions handle single values (scalars), sets of values (vectors), tables (matrices), or n-dimensional arrays. A user can, for example, add several scalars:

```
5+10+10
```

for a total of 25. Or he can add a scalar to each element in a vector:

```
3+12 7 4 15
```

The scalar 3 is added to each element in the vector to the right of the +; the result is a vector:

```
15 10 7 18
```

Consider another example:

```
QUANTITY←7 11 2 4 13
PRICE←1.75 28.49 79.95 15.00 3.20
PAYMENT←QUANTITY×PRICE
```

QUANTITY and PRICE are each assigned five elements, and PAYMENT is assigned the value of QUANTITY multiplied by PRICE. The first element in QUANTITY is multiplied by the first element in PRICE, the second element in QUANTITY by the second in PRICE, and so on, to yield:

```
      12.25 313.39 159.9 60 41.6
```

Notice that no looping is needed.

QUANTITY and PRICE may consist of any like number of items. The result is a vector named PAYMENT containing a corresponding number of items.

Assume that a table containing the monthly budget for each of several departments has been entered into APL and assigned to the variable BUDGET. The APL statement to cause each month's budget for each department to be increased by 5% is:

```
      BUDGET←BUDGET×1.05
```

APL primitive functions available in VS APL are summarized in "Appendix A: Summary of APL Language Elements."

Defined Functions

A *defined function* is a named collection of APL statements. When the name is entered at the keyboard or used in another defined function, the collection of statements is executed. For example, a user may define a function to average a set of numbers, as follows:

```
      ∇AVERAGE SN
[1]   (+/,SN)÷ρ,SN
      ∇
```

The symbol ∇ opens function definition mode. Everything between it and the next ∇ forms the definition of the function. AVERAGE is the name given to the function. SN, which follows AVERAGE, indicates that AVERAGE will always require an argument. The expression (+/ , *SN*) means "sum the elements of the argument SN;" ÷ ρ , *SN* means "divide by the number of elements in the argument SN." The commas preceding SN in line [1] ensure that the elements of SN will be treated as a vector even if SN is given as a scalar or an n-dimensional array. Now, each time the user enters the word AVERAGE followed by a set of numbers, the result is the average of those numbers. For example:

```
      AVERAGE 2 3 4 5 6 7 8
```

yields 5. Or:

```
      AVERAGE 15 21 8 2 37 17 9 24
```

yields 16.625.

Similarly, another function called TRIAREA can be defined to calculate the area of a triangle:

```
      ∇TA←B TRIAREA H
[1]   TA←B×H÷2
      ∇
```

The arguments B and H on either side of TRIAREA indicate that TRIAREA will always require two arguments, the base and height of a triangle.

To find the area of a triangle with a base of 4 and a height of 5, the user would enter:

```
      4 TRIAREA 5
```

and receive a result of 10.

In the next example, the average area of several triangles is calculated by defining a third function, AVTRIAREA which uses the two previously defined functions AVERAGE and TRIAREA as building blocks:

```
      ∇B AVTRIAREA H
[1]   AVERAGE B TRIAREA H
      ∇
```

Thus if the user wished to calculate the average area of three triangles with bases of 6, 15, and 9 and heights of 7, 8, and 16, he would enter:

```
      6 15 9 AVTRIAREA 7 8 16
```

and receive the average area of these three triangles, which is 51.

Example

APL can be used in a wide variety of commercial and scientific applications.

In the example that follows, APL is used to compute compound annual interest:

```
      ∇ COMPINTEREST
[1]   P←⎕
[2]   R←⎕
[3]   'AT END OF YEAR:   AMOUNT AVAILABLE IS:'
[4]   A←P×(1+R÷100)*⍳20
[5]   4 0 22 2 ⍕(⍳20),[1.1]A
      ∇
```

The compound interest program operates as follows:

- Lines 1 and 2 request that values be entered from the terminal for P (principal) and R (annual interest rate).
- Line 3 provides a heading for the compound interest results.
- Line 4 computes the interest for each of 20 years. The ⍳20 means to generate consecutive whole numbers from 1 to 20; the parenthetical expression is subsequently raised to the power of 20 different values. These values are each multiplied by the principal to produce 20 different amounts.
- Line 5 formats and prints the table of results.

Note that it is not necessary to establish a counter and to increment and test the counter for each of the 20 years for which interest is being compounded.

Figure 3 shows the COMPINTEREST program as it is used. As the figure shows, after the user has defined COMPINTEREST, all he does is enter the name COMPINTEREST to cause it to be executed. The ⎕: indicates that he must provide some numerical information. In this case, the first piece of information required is the amount for which interest is to be compounded. The second piece of information is the annual interest rate at which the amount is invested. After the user has entered the principal and rate, VS APL calculates the compound interest for each of 20 years and prints the results under the headings provided in statement 3 of the definition of COMPINTEREST.

```
      COMPINTEREST
⎕:
      10000.009
⎕:
      7.25
AT END OF YEAR:      AMOUNT AVAILABLE IS:
   1                    10725.00
   2                    11502.56
   3                    12336.50
   4                    13230.89
   5                    14190.13
   6                    15218.92
   7                    16322.29
   8                    17505.66
   9                    18774.82
  10                    20135.99
  11                    21595.85
  12                    23161.55
  13                    24840.76
  14                    26641.72
  15                    28573.24
  16                    30644.80
  17                    32866.55
  18                    35249.37
  19                    37804.95
  20                    40545.81
```

Figure 3. Calculating Compound Interest with a User-Defined Function

VS APL AUXILIARY PROCESSORS

This chapter describes the auxiliary processors distributed with the VS APL program product. An auxiliary processor is a program that communicates with an APL program through shared variables and thus allows an APL user to request special host-dependent tasks such as external file input/output operations.

Other auxiliary processors may be written by system programmers to meet the needs of the installation.

An auxiliary processor designed to operate with VS APL under VSPC is the VS TSIO program product; it is described in the chapter "VS TSIO Overview and Installation Planning."

A user communicates with an auxiliary processor by sharing one or more variables with it. Once sharing is established, the APL user simply references or specifies values for each shared variable using the APL language in the same way he does for normal variables.

An APL user and an auxiliary processor operate separately; the only connection between them is the variables they share. Sharing variables as a means of linking independent processors can be used to build systems of any degree of complexity.

Auxiliary Processors Distributed with VS APL under VSPC

The auxiliary processors distributed with VS APL under VSPC provide data management services for both VSPC library and external VSAM files. They also provide for control of an IBM 3270 display screen through a user application.

APL Format Auxiliary Processor

This auxiliary processor allows VS APL users to read and write APL variables in the VSPC library. The auxiliary processor writes the variables in their VS APL internal form, and when it reads the variable, it expects the internal form. The files can be created in either sequential or direct organization. Sequential files must be read sequentially; variables of any size that can be manipulated within the user's workspace can be written. Files written in direct organization can be read sequentially or directly. They must be created and extended sequentially. The direct files use fixed-length records; variables of any length up to 4054 bytes can be written.

EBCDIC Format Auxiliary Processor

This auxiliary processor allows the VS APL user to write files that can be read by other processors, such as VS BASIC under VSPC or the VSPC data editing facilities. The VS APL user passes character vectors to this auxiliary processor, and the auxiliary processor translates the character vectors to EBCDIC and writes to a sequential or direct VSPC file. The VS APL user may also read VSPC files created by other processors under VSPC. The EBCDIC information in the VSPC files will be translated by the auxiliary processor to VS APL's internal form. The files can be created in either sequential or direct organization. Sequential files must be read sequentially; variable-length records up to 4058 bytes can be written. Files written in direct organization can be read sequentially or directly. They must be created and extended sequentially. The direct files use fixed-length records of any length up to 4058 bytes.

VSAM Auxiliary Processor

This auxiliary processor allows the VS APL user to read and write VSAM data sets that are maintained by the operating system, not by VSPC. Both key-sequenced and entry-sequenced data sets are supported.

None of the data transferred by this auxiliary processor is converted; however, functions are available in a VS APL distributed workspace to assist in the conversion.

The VSAM Access Method Services commands are used to define and allocate these data sets before starting VSPC.

Full Screen Management Auxiliary Processor

This auxiliary processor provides for control of an IBM 3270 display device through a user application. The application can use the processor to format the screen, read from or write to selected screen fields, erase fields, have program function and program attention keys read, and perform other operations generally available to an IBM 3270 display device, for instance, intensify a displayed field or prepare a field for light pen usage. Optionally, a formatted screen may be sent to a printer for permanent copy.

Auxiliary Processors Distributed with VS APL under CMS

Auxiliary processors distributed with VS APL under CMS allow the user to request specific services from CMS through the APL language.

CMS Command Auxiliary Processor

This processor gives the VS APL user the ability to issue some control program and CMS commands without leaving APL mode. Control program commands are described in *VM/370: CP Command Reference for General Users* and CMS commands are described in *VM/370: CMS Command and Macro Reference.*

CMS Stack Input Auxiliary Processor

This processor allows the VS APL user to add entries to the CMS console input stack. For example, he can (1) take checkpoints between the execution of APL functions and (2) load and execute applications that reside in more than a single workspace.

CMS File Auxiliary Processor

This processor provides sequential and direct access to the CMS disk files. This allows the VS APL user to read and write more data than would be available to him in only his own workspaces. The CMS disk file system is described in *VM/370: CMS User's Guide.*

CMS FILEDEF I/O Auxiliary Processor

This processor allows the VS APL user to sequentially read from or write to CMS data sets, non-CMS data sets supported by the Queued Sequential Access Method (QSAM), and unit record devices. A CMS FILEDEF command must be issued for the dataset or device before it is accessed.

CMS VSAM Auxiliary Processor

This processor allows the VS APL user to process VSAM data sets as defined through the CMS DLBL command. Key-sequenced and entry-sequenced data sets are supported. (Requests for keyed operations require the full VSAM key.)

None of the data transferred by this auxiliary processor is converted. However, defined functions are available in the distributed workspace APFNS to assist in converting various System/370 data formats (such as EBCDIC, packed decimal).

The VSAM Access Method Services commands are used to define and allocate data sets before processing by APL.

Auxiliary Processors Distributed with VS APL under CICS/VS

Auxiliary processors distributed with VS APL under CICS/VS allow the user to request a variety of services from CICS/VS through the APL language.

Command Auxiliary Processor

This auxiliary processor allows the VS APL user to issue a very limited set of CICS/VS commands from the VS APL transaction. It also can be used to display, override, or change the fields in the user's VS APL profile that control terminal output from VS APL.

Alternate Input Auxiliary Processor

This auxiliary processor allows the user to specify an APL command or an APL statement that can be executed at a later time when VS APL requests input. When VS APL requests you to enter a command or an APL expression, it first checks to see if such an expression or command has been specified. If so, it is used as if you had entered it; if not, input is requested from you at the terminal.

APL Format Auxiliary Processor

This auxiliary processor allows VS APL users to read and write APL variables in an auxiliary storage file. The file is part of an internal library maintained by VS APL, and its contents are accessible only by using this auxiliary processor. The auxiliary processor writes the variables in their VS APL internal form, and when it reads the variables, it expects the internal form. Files can be created in either sequential or direct organization. Sequential files must be read sequentially; variables of any size that can be manipulated within the user's workspace can be written. Files written in direct organization can be read sequentially or directly. Files must be created and extended sequentially. Direct files use fixed-length records; variables of any length up to 4054 bytes can be written.

VSAM Auxiliary Processor

This auxiliary processor allows the VS APL user to read and write VSAM and ISAM data sets. Both key-sequenced and entry-sequenced data sets can be accessed.

The data transferred by the auxiliary processor is not converted to APL internal form; however, functions are available in the VS APL distributed workspace, VSFILES, to assist in the conversion.

VSAM and/or ISAM data sets to be accessed must be defined and allocated before starting CICS/VS. VSAM Access Method Services are used to define and allocate VSAM data sets.

Full Screen Management Auxiliary Processor

This processor can be used within an APL function to control the format of display devices used by VS APL under CICS/VS, and is used if an alternative to the standard screen format is desired while running an application. An application can use the auxiliary processor to format the screen, read from or write to selected screen fields, erase fields, have program function and program attention keys read, and perform other operations available to the display device (for instance, intensify a displayed field or prepare a field for light pen usage). The current screen image can also be sent to the destination identified in the user's VS APL profile.

DL/I Auxiliary Processor

This auxiliary processor provides the VS APL user all the DL/I facilities available to a CICS/VS transaction. For example, a VS APL user can read from or write to a DL/I data base, insert, replace, or delete segments, as well as use the advanced functions such as path calls and last occurrence retrieval.

Transient Data Auxiliary Processor

This auxiliary processor allows the APL user to read and write to CICS/VS transient data destinations. A CICS/VS transient data destination may be defined as a sequential device such as a card reader, printer, or tape drive; or it may be used to pass data between CICS/VS tasks.

Writing to a transient data destination sends a record to that destination's queue; reading from a transient data destination gives the oldest entry in the destination's queue to the requestor and removes that entry from the queue.

Main Storage Access Auxiliary Processor

This auxiliary processor is used by the APL administrator through the functions provided in the administrative workspace, ADMIN, distributed with VS APL under CICS/VS. These functions are described in "Product Description" under distributed workspaces.

INSTALLATION PLANNING FOR VS APL

This chapter provides information needed to plan for the installation of VS APL under VSPC, under CMS, and under CICS/VS.

VS APL under VSPC

The topics that follow describe the system configuration requirements and storage estimates for installing and operating VS APL under VSPC.

System Configuration (VSPC)

VS APL can be run on System/370 Models 125 and above with VSPC under DOS/VS, OS/VS1 and OS/VS2. See *VS Personal Computing (VSPC) for OS/VS and DOS/VS: General Information* for a discussion of the configurations of systems that include VSPC.

The microcoded APL Assist feature is standard on System/370 Models 138, 148, 135-3, and 145-3, and is an optional special feature on System/370 Models 135-1 and 145-1. Its existence in the system is not apparent to the user, except that the user may notice improved performance.

The following terminals can be used with VS APL:

- IBM 1050 Data Communications System
- IBM 2741 Communications Terminal
- IBM 3270 Information Display System
- IBM 3767 Communications Terminal
- IBM 3770 Data Communications System
- CPT-TWX Models 33 and 35

The 3767 in SDLC mode and the 3270 require the APL feature to enable the terminal user to define, edit, or list functions. Without the APL feature, the terminal user can enter system commands, load existing workspaces, execute previously defined functions, and enter and display names that do not contain underlined alphabetic characters or special characters such as ⎕ or ∆.

The IBM 3770 Data Communications System is supported in execution mode only. Previously defined functions may be executed but new ones cannot be defined.

CPT-TWX Models 33 and 35 have a fixed and limited character set that prevents the full use of VS APL. The terminal user can load existing workspaces, execute functions, and enter names that do not contain the underlined alphabetic characters. The terminal user cannot edit or define APL functions, clear the stack of suspended functions, resume suspended functions or enter overstruck characters. Line deletion and character correction techniques are different than for other APL terminals. Therefore, these terminals are not recommended for use with APL.

Storage Estimates (VSPC)

In a virtual storage environment, the amount of real storage required depends on the level of performance desired. For VS APL under VSPC, the real storage requirements depend on a number of factors that vary from installation to installation. These factors include:

- System/370 CPU model and operating system
- Tuning parameters
- Characteristics of device used for paging and library storage
- Level and type of foreground activity
- Level and type of background activity
- Size of VS APL workspaces
- Shared-variable storage requirements
- Number and level of activity of auxiliary processors.

256K bytes of virtual storage are needed to install VS APL under VSPC. The virtual storage required to operate VS APL is 185K bytes, excluding VSPC, access methods, and workspaces.

The optional microcoded APL Assist feature available on the System/370 Models 135-1 and 145-1 requires the following control storage:

15K bytes on System/370 Model 135-1
20K bytes on System/370 Model 145-1

The microcoded APL Assist feature, standard on System/370 Models 138, 148, 135-3, and 145-3, requires no additional control storage.

VS APL under CMS

The topics that follow describe the system configuration requirements and storage estimates for installing and operating VS APL under CMS.

System Configuration (CMS)

VS APL can be run on System/370 Models 135 and above supported by VM/370 CMS. For detailed machine requirements, see *VM/370 Planning and System Generation Guide.*

The microcoded APL Assist feature is standard on System/370 Models 138, 148, 135-3, and 145-3, and is an optional special feature on System/370 Models 135-1 and 145-1. Its existence in the system is not apparent to the user, except that the user may notice improved performance.

VS APL under CMS supports the following terminals:

- IBM 1050 Data Communication System
- IBM 2741 Communication Terminal
- IBM 3767 Communication Terminal (in 2741 migration-aid mode)
- IBM 3270 Information Display System

Support for these terminals is provided by VM/370. VS APL uses the virtual console of the CMS virtual machine as the terminal device.

The 3767 and the 3270 require the APL feature to enable the terminal user to define, edit, or list functions. The APL feature for the 3270 is supported with Release 3 of VM/370 CMS. Without the APL feature, the terminal user can enter system commands, load existing workspaces, execute previously defined functions, and enter names that do not contain underlined alphabetic characters or special characters such as ⎕ or ∆.

Storage Estimates (CMS)

In the virtual machine environment of VM/370 CMS, the amount of real storage depends on the level of performance desired. For VS APL under CMS, the real storage requirements depend on a number of factors that vary from installation to installation. These factors include:

- System/370 CPU model
- Tuning parameters selected
- Characteristics of the devices used for paging and for library storage
- Size of VS APL workspaces
- Shared-variable storage requirements
- Number and level of activity of auxiliary processors
- Level and type of foreground activity
- Level and type of background activity
- Whether VS APL is running as a shared or non-shared system

A virtual machine of at least 400K bytes is required to install VS APL under CMS. VS APL under VM/370 Release 3 can be installed as a shared system, in which all users share one copy of the VS APL executor, interpreter, and shared storage manager. If VS APL is to be run as a shared system, the low address of the shared area is selected during system generation. To install VS APL as a shared system, a virtual machine of at least 400K bytes beyond that selected address is required.

To run VS APL as a non-shared system with no auxiliary processors requires a virtual machine of at least 400K bytes.

To run VS APL as a shared system with no auxiliary processors requires a virtual machine of at least 320K bytes, and 192K bytes of shared main storage.

To run all the auxiliary processors distributed with VS APL, except the VSAM auxiliary processor, increase the virtual machine size by 20K. This includes 4K of shared memory (used for sharing variables); if more than 4K is required for the application, increase the virtual machine size accordingly.

To run the VSAM auxiliary processor, increase the virtual machine size by 200K bytes, which includes 25K bytes of shared memory.

The virtual machine size requirements increase when the size of the workspace increases or auxiliary storage devices or files are added.

The optional APL Assist feature available on the System/370 Models 135-1 and 145-1 requires the following control storage:

15K bytes on System/370 Model 135-1
20K bytes on System/370 Model 145-1

The APL Assist feature, standard on System/370 Models 138, 148, 135-3, and 145-3, requires no additional control storage.

VS APL under CICS/VS

The topics that follow describe the system configuration requirements and storage estimates for installing and operating VS APL under CICS/VS.

System Configuration (CICS/VS)

VS APL under CICS/VS must be run on System/370 models equipped with the floating point feature and supported by CICS/VS Version 1 Release 3 (or later) under the DOS/VS, OS/VS1, or OS/VS2 (MVS) operating systems. The floating point feature is required as follows:

System/370 Model 125 or 145-1—feature 3910
System/370 Model 135-1—feature 3840 or feature 3900

See *Customer Information Control System/Virtual Storage (CICS/VS) Version 1, Release 3 General Information* for a discussion of system configurations under CICS/VS.

The microcoded APL Assist feature is standard on System/370 Models 138, 148, 135-3, and 145-3, and is an optional special feature on System/370 Models 135-1 and 145-1. If present, the APL Assist is used.

The Data Analysis-APL feature is required only if APL special characters must be entered or displayed at the terminal. When the Data Analysis-APL feature is not in use and an attempt is made to display a special APL character, it is replaced with double quotation marks (").

VSAM is required, and BTAM or VTAM is required.

The terminals that can be used with VS APL under CICS/VS are:

- IBM 3270 displays and printers attached to IBM 3270 control units.
- IBM 3270 displays and printers attached to IBM 3790 control units in compatability mode.

The *CICS/VS Version 1, Release 3 General Information* details on the configurations supported using BTAM and VTAM.

VS APL under CICS/VS requires the following CICS/VS components:

- The CICS/VS File Control Program with VSAM services.
- The Terminal Control Program with automatic transaction initialization.
- Interval Control Program
- Dynamic Open/Close Program
- Temporary Storage Program

Storage Estimates (CICS/VS)

The storage estimates presented in this section are for planning purposes only, and may vary from actual storage requirements.

VS APL under CICS/VS requires 200K bytes of virtual storage, plus workspace of up to 512K bytes of virtual storage per VS APL user. Also, 2K to 6K bytes are required per VS APL user (depending on the display screen size) for the 3270 interface. Additional virtual storage requirements for each auxiliary processor (when active), the shared storage manager, and hardcopy are as follows:

- CICS Command Auxiliary Processor—4K bytes plus 1K bytes per share request
- Alternate Input Auxiliary Processor—1K bytes
- APL Format Auxiliary Processor—4K bytes plus 2 times (buffer length per share request)
- VSAM Auxiliary Processor—8K bytes plus VSAM control interval size plus 2 times (buffer length per share request)
- Full Screen Manager Auxiliary Processor—2K bytes plus 2K to 8K bytes per VS APL user, depending on the screen size and usage.
- DL/1 Auxiliary Processor—4K bytes plus 2 times (buffer length per share request)
- Transient Data Auxiliary Processor—8K bytes plus 2 times (buffer length per share request)
- Main Storage Access Auxiliary Processor—1K bytes per share request
- Hardcopy—1K bytes for each user selecting screen image copy or continuous copy.
- Shared Storage Manager—8K bytes plus the shared storage size.

Auxiliary storage requirements for VS APL under CICS/VS are as follows:

- VS APL distributed workspaces, CICS/VS libraries, and system libraries—20 cylinders on an IBM 2314 or an IBM 3340, 10 cylinders on an IBM 3330, or 6 cylinders on an IBM 3350.
- Space allocations for the APL library and APL Format auxiliary processor are up to the administrator, and should be based on the number of users and the maximum space allowed for each.
- The amount of intrapartition and extrapartition transient data space which may be used for hardcopy and by the Transient Data auxiliary processor depends on the number of VS APL users authorized to send data to those destinations, the amount of data each user transmits and how long the data is kept, and the reuseability characteristics of those destinations.

The optional microcoded APL Assist feature available on the System/370 Models 135-1 and 145-1 requires the following control storage:

15K bytes on System/370 Model 135-1
20K bytes on System/370 Model 145-1

The microcoded APL Assist feature, standard on System/370 Models 138, 148, 135-3, 145-3, requires no additional control storage.

VS APL AND OTHER IBM APL IMPLEMENTATIONS

VS APL is designed for new and old users of APL. It is expected that users of APL\360-OS, APL\360-DOS, APLSV, and APL/CMS will find that VS APL offers an attractive alternative to their current APL product.

VS APL is an extension of APL\360 and incorporates the language features available in APLSV and APL/CMS. The following language features are provided:

- Variable size workspaces, which provide greater flexibility and convenience to the user.
- An execute function ⍎ that enables the user to execute an APL expression represented as a character vector. For example:

  ```
  A←'1 2 3 4'
  B←'5 6 7 8'
  (⍎A)+⍎B
  ```

  returns a numeric vector, as follows:

  ```
  6 8 10 12
  ```

- A format function ⍕ that enables the user to control the display of an array. This function also converts the numeric values of the array to characters. The left argument indicates the characteristics that the user wants applied to the right argument. For example:

  ```
  8 2⍕1.56 234.347 23.2 1.8976
  ```

  is formatted so that each element is allowed a total of eight characters, including the decimal point and any blanks to the left. Each element is rounded to two decimal places. The result is a character representation of the array as follows:

  ```
      1.56  234.35   23.20    1.90
  ```

- A scan operator (\) that, when used with another primitive function, returns a result with the same number of elements and of the same dimension as the argument, but with elements replaced by cumulative quantities. For example, the following add scan:

  ```
  +\1 2 3 4
  ```

  would result in:

  ```
  1 3 6 10
  ```

- A system variable (⎕*LX*, latent expression) that can be saved with a workspace; the expression is executed automatically whenever the workspace is loaded.
- A system function (⎕*CR*, canonical representation) that represents a function as a character matrix; this system function enables the user to store functions as data. As data, functions can be stored in a file, or manipulated by other APL functions.
- A system function (⎕*FX*, fix function) that enables the user to establish a function represented as a character array.
- The shared-variable system functions, which monitor and control the status of shared variables.

- Additional system functions and variables that enable the terminal user to inquire about and set the environment in which VS APL is running.

Conversion Aids

VS APL provides comprehensive conversion aids to assist the user in converting APL functions defined under other IBM APL implementations so that they can be used under VS APL. The conversion aids consist of off-line programs that provide for format and content conversion of APL\360, APLSV, and APL/CMS functions.

These same conversion programs can be used to transfer functions and data from other APL users who created their functions or entered their data under another APL implementation.

VS APL conversion aids distinguish between *format* and *content* conversion. Format conversion refers to changes needed in the internal format of directories and workspaces. Content conversion refers to changes required to allow for the effects of APL language differences within defined functions.

The VS APL conversion program gives each installation the option of performing only format conversion, or both format and content conversion. The installation can elect to convert only selected workspaces from an APL library or all of them. The conversion program also allows the renumbering of libraries and renaming of workspaces during conversion.

All format conversions are resolved unambiguously by the conversion program. However, some language differences that require content conversion may necessitate manual assistance. To identify these differences, the conversion program produces a report during content conversion; the report lists the line numbers and statements that may require attention. Language differences that are apparent only by execution-time evaluation of the arguments are not flagged by the conversion program.

Another type of content conversion is provided with VS APL in a distributed workspace named CONVERT. This workspace allows the terminal user to convert selected functions or workspaces individually. Generally, the same content conversions that are completely resolved or flagged by the off-line conversion program are resolved or flagged by the CONVERT workspace.

VS TSIO OVERVIEW AND INSTALLATION PLANNING

This chapter briefly explains VS TSIO: what it is, how it is used, and what it requires to operate.

VS TSIO Overview

VS TSIO is a program product, consisting of an auxiliary processor and supporting workspaces, that enables a VS APL user to access OS/VS sequential, direct, or partitioned data sets. (It is not designed to access data sets created by the Indexed Sequential Access Method, ISAM, or created by the Virtual Storage Access Method, VSAM.) VS TSIO operates with VS APL and VSPC in OS/VS1 or OS/VS2. It provides services similar to those provided by the TSIO facility of APLSV. VS TSIO can be used by APLSV-TSIO users when they have moved to VS APL and by current VS APL users as an additional file handling capability. Services provided by VS TSIO include:

- Creating and deleting data sets
- Reading and writing data sets (sequentially and directly)
- Renaming data sets
- Cataloging and uncataloging data sets
- Sending messages to the console operator
- Querying the operating status of VS TSIO
- Sharing data sets between users
- Synchronizing events among several users

VS TSIO also offers an installation of several mechanisms to control VS TSIO usage and restrict the services authorized to any one VS TSIO user. These mechanisms include the following:

- Controlling access to VS TSIO—The installation can specify which VS APL users are authorized to use VS TSIO.
- Controlling access to specific VS TSIO facilities—The installation can specify which selected subset (or subsets) of VS TSIO services is available to each authorized user.
- Reserving data sets—The installation can restrict data set access to only selected VS TSIO users.
- Establishing a command data set—The installation can authorize individual users to invoke specific VS TSIO command strings in data sets called *indirect command data sets.* These command strings may provide access to VS TSIO services and data sets which the user is otherwise not allowed to access.

Workspaces Distributed with VS TSIO

The workspaces distributed with VS TSIO assist in its use. The workspaces are:

- APLFILE, which creates or retrieves direct-access data sets containing files of APL arrays. (The workspace can retrieve new data sets or ones created using the APLFILE workspace under APLSV-TSIO.)
- CONVAPLFILE, which transfers data sets created or retrieved by the APLFILE workspace to the VSPC library. This prepares the data sets for access by the functions in VSAPLFILE, a workspace distributed with VS APL.
- TSIO, which simplifies the use of VS TSIO in creating and accessing data sets. This workspace also creates and maintains indirect command data sets.
- FEDIT, which transmits text such as program listings between auxiliary storage and the active workspace. (This workspace is intended for use with the MEDIT and SEDIT workspaces distributed with VS APL.)
- CONVERSION, which converts numbers and characters represented in VS APL internal format to and from standard System/370 representation or APLSV representation.
- TSIOPS, which controls VS TSIO user authorization and defines VS TSIO direct-access volumes. (This workspace is restricted to the VS TSIO controller and to users with the proper type of VS TSIO authorization.)
- UTILITY, which provides utility functions for managing operating-system data. (This workspace is restricted to users with the proper type of VS TSIO authorization.)

VS TSIO Operating Requirements

The VS TSIO auxiliary processor runs in its own partition (in OS/VS1) or region (in OS/VS2). VSPC must be active in a separate partition or region, and VS APL must be installed as a foreground processor of VSPC for a user to request services from the VS TSIO auxiliary processor. The workspaces distributed with VS TSIO are used in the VSPC partition or region with VS APL. A minimum of 128K bytes of virtual storage is needed to operate VS TSIO.

VS TSIO uses the Basic Sequential Access Method (BSAM), the Basic Direct Access Method (BDAM), and the Basic Partitioned Access Method (BPAM) to access its data sets. Devices supported by VS TSIO are those supported by BSAM, BDAM, and BPAM.

SUPPORTING USER PUBLICATIONS

VS APL is supported by the publications described below. For information on other publications relating to VS APL or its operating environments, refer to the *IBM System/370 Bibliography,* GC20-0001.

APL Language

Order No: GC26-3847

Audience: All users of VS APL.

This publication describes the elements of the APL language for users of VS APL as well as other APL systems. The initial two chapters, respectively, provide overview and basic information about APL while the remainder of the publication details the operation of each language element.

A VS APL Terminal User's Guide for the appropriate host environment should be used with this publication to identify environment-dependent features and limitations.

VS APL for VSPC: Terminal User's Guide

Order No: SH20-9066

Audience: Users of VS APL for VSPC.

This publication provides procedural and reference information about VS APL when it is operated under VSPC. It contains detailed information on the terminals that can be used with VS APL under VSPC and the procedures that must be followed in starting a terminal session. This publication also describes the system commands available with VS APL and the auxiliary processors and workspaces distributed with the product. General information is also provided on workspace conversion and the VS APL workspace conversion utility program.

No previous experience with VSPC is required for use of this publication. Familiarity with the APL language, however, is assumed.

VS APL for CMS: Terminal User's Guide

Order No: SH20-9067

Audience: Users of VS APL for CMS.

This publication provides procedural and reference information about VS APL when it is operated under control of CMS. It contains detailed information on the terminals that can be used with VS APL under CMS and the procedures that must be followed in starting a terminal session. This publication also describes the system commands available with VS APL and the auxiliary processors and workspaces distributed with the product. General information is also provided on workspace conversion and the VS APL workspace conversion utility program.

No previous experience with CMS is required for use of this publication. Familiarity with the APL language, however, is assumed.

VS APL for CICS/VS: Terminal User's Guide

Order No: (Available with VS APL Release 3)

Audience: Users of VS APL for CICS/VS.

This publication provides procedural and reference information about VS APL when it is operated under control of CICS/VS. It contains detailed information on the terminals that can be used with VS APL under CICS/VS and the procedures that must be followed in starting a terminal session. This publication also describes the system commands available with VS APL, and the auxiliary processors and workspaces distributed with the product. General information is also provided on workspace conversion and the VS APL workspace conversion utility program.

No previous experience with CICS/VS is required for use of this publication. Familiarity with the APL language, however, is assumed.

VS APL Reference Summary

Order No: SX26-3712

Audience: All users of VS APL.

This booklet is a handy digest to using VS APL. It summarizes the APL language elements and system commands available with VS APL, lists VS APL workspace attributes, and illustrates VS APL auxiliary procedures.

This booklet is a digest of information found in the *APL Language* manual and the VS APL terminal user's guides. It assumes that the reader is familiar with the language and system details described in these publications.

VS APL Installation Reference Material

Order No: SH20-9065

Audience: Installation managers and system programmers responsible for installing and maintaining VS APL.

This publication describes how to install VS APL and how to perform various system management functions such as defining VS APL users, creating and maintaining VS APL public and private libraries, and converting APL\360, APLSV, and APL/CMS workspaces to VS APL workspaces. This publication also describes how auxiliary processors may be defined to the host system and how VS APL workspaces may be transported between CMS, VSPC, and CICS/VS.

The reader is assumed to have a knowledge of VS APL and the host environment under which it runs.

VS Personal Computing (VSPC): Writing Processors

Order No: SH20-9074

Audience: System programmers who want to design, write, and implement auxiliary processors to be used with VS APL under VSPC.

This publication describes how to design, write, and implement auxiliary processors to be used with VS APL under VSPC. It provides information fundamental to the design of auxiliary processors. This includes information on the sharing of variables, data formats, register usage, return codes, macro instructions, and control blocks.

The reader is assumed to have a knowledge of assembler language.

VS APL for CMS: Writing Auxiliary Processors

Order No: SH20-9068

Audience: System programmers who want to design, write, and implement auxiliary processors to be used with VS APL for CMS.

This publication describes how to design, write, and implement auxiliary processors to be used with VS APL for CMS. It provides information fundamental to the design of auxiliary processors. This includes information on the sharing of variables, data formats, register usage, return codes, macro instructions, and control blocks.

The reader is assumed to have knowledge of assembler language.

VS APL for CICS/VS: Writing Auxiliary Processors

Order No: (Available with VS APL Release 3)

Audience: System programmers who want to design, write, and implement auxiliary processors with VS APL under CICS/VS.

This publication describes how to design, write, and implement auxiliary processors to be used with VS APL under CICS/VS. It provides information on design considerations, sharing variables, data formats, register usage, return codes, macro instructions, and control blocks.

The reader is assumed to have knowledge of assembler language, APL, and CICS/VS.

VS TSIO Guide and Reference

Order No: SH20-9107

Audience: People who plan for, install, operate, and use VS TSIO.

This publication provides an introduction to VS TSIO, and describes how to install and operate it. It also provides procedural and reference information about using VS TSIO.

The prerequisite knowledge expected of readers varies between chapters. A knowledge of APL is helpful to understand the introduction and overview; however anyone using VS TSIO is expected to be familiar with APL, and especially with the shared variable facility. The chapter on operating VS TSIO expects the reader to be familiar with system console concepts, operations, and functions. And the chapter on installing VS TSIO is directed to system programmers.

VS APL Program Logic

Order No: LY20-8032

Audience: System programmers and others who maintain VS APL.

This publication provides information about the method of operation, program organization, data areas, and control block formats of VS APL. It also provides information helpful in reading program listings, and information on determining and reporting problems.

Users of this manual should be familiar with the host operating environment.

APPENDIX A. SUMMARY OF APL LANGUAGE ELEMENTS

The summary of APL functions that follows is provided to illustrate the extent of APL's capability. APL distinguishes between a *monadic* function—a function with only one argument—and a *dyadic* function—a function with two arguments.

Primitive Functions

Scalar Monadic Functions

The following monadic functions return a scalar result when the argument is a scalar. These functions can also be applied on an element-by-element basis to an array (that is, a vector, matrix, or n-dimensional array).

+*Y*	Y (no change, that is, 0+Y)
-*Y*	Negative of Y (that is, 0-Y)
×*Y*	Signum of Y (the sign ¯1, 0, 1 of Y)
÷*Y*	Reciprocal of Y (that is, 1÷Y)
⋆*Y*	e to the Yth power (e is 2.71828...)
⌈*Y*	Ceiling of Y (next integer ≥ Y)
⌊*Y*	Floor of Y (next integer ≤ Y)
|*Y*	Absolute value of Y
○*Y*	PI times Y (PI is 3.14159...)
⍟*Y*	Natural logarithm of Y
!*Y*	Y factorial if Y is a positive integer; otherwise, the Gamma function of Y+1
?*Y*	Random equi-probable selection of an integer from 1 to Y
~*Y*	Not Y (logical arguments only)

Scalar Dyadic Functions

The following dyadic functions return a scalar result when their arguments are scalars. They may also be applied on an element-by-element basis to arrays (that is, vectors, matrices, or n-dimensional arrays) when one argument has only one element or both arguments have the same rank and length in every dimension. Any of these functions may be used in inner product, outer product, reduction, and scan operations, described later.

X+*Y*	X plus Y
X-*Y*	X minus Y
X×*Y*	X times Y
X÷*Y*	X divided by Y
X⋆*Y*	X to the Yth power
X⌈*Y*	Larger of X and Y

`X⌊Y`	Smaller of X and Y
`X|Y`	X residue of Y
`X⍟Y`	Logarithm of Y to the base X
`X○Y`	Trigonometric functions and inverse trigonometric functions of Y
`X!Y`	Number of combinations of Y things taken X at a time
`X<Y`	X less than Y (0 if false; 1 if true)
`X≤Y`	X less than or equal to Y (0 if false; 1 if true)
`X=Y`	X equals Y (0 if false; 1 if true)
`X≥Y`	X greater than or equal to Y (0 if false; 1 if true)
`X>Y`	X greater than Y (0 if false; 1 if true)
`X≠Y`	X not equal to Y (0 if false; 1 if true)
`X∧Y`	X and Y (logical arguments only)
`X∨Y`	X or Y (logical arguments only)
`X⍱Y`	Neither X nor Y (X NOR Y) (logical arguments only)
`X⍲Y`	Not both X and Y (X NAND Y) (logical arguments only)

Operators

Inner and Outer Product

The operations described below can be performed on arrays or scalars. The symbol ● (not an APL symbol) has been used to indicate any scalar dyadic function.

`X●.●Y`	Inner product of X and Y
`X∘.●Y`	Outer product of X and Y

Reduction

In the operations described below, the rank of the argument is reduced by one dimension. The specified dimension is eliminated by applying the scalar dyadic function (represented by ●) to the elements along the dimension. The effect is the same as placing ● between adjacent elements of the specified dimension and then executing the resultant statement.

`●/Y`	The reduction along the last dimension of Y
`●/[Z]Y`	The reduction along the Zth dimension of Y
`●⌿Y`	The reduction along the first dimension of Y

Scan

In the operations described below, the result has the same shape as the argument. The result is formed by reducing successive elements of the argument along the specified dimension using any scalar dyadic function (represented by ∘). For a given dimension of the result (R), *R*[1]← ∘/*Y*[1], *R*[2]←∘/*Y*[1 2], *R*[3]←∘/*Y*[1 2 3], etc.

∘\Y	The scan along the last dimension of Y
∘\\[*Z*]*Y*	The scan along the Zth dimension of Y
∘⍀*Y*	The scan along the first dimension of Y

Other Functions

The operations described below are generally helpful in manipulating elements by, for example, defining the shape of a matrix or array, catenating values, and inverting or rotating elements of a matrix or array.

X⍴*Y*	Reshape Y to have dimension X
⍴*Y*	Dimension of Y
X[*Y*]	The elements of X at locations Y
X⍳*Y*	Locations of Y within X
⍳*Y*	The first Y consecutive integers
X∊*Y*	Which elements of X are members of Y (0 if false; 1 if true)
X⊤*Y*	Representation of Y in number system X
X⊥*Y*	Value of the representation Y in number system X
X⌽*Y*	Rotation by X along the last dimension of Y
X⌽[*Z*]*Y*	Rotation by X along the Zth dimension of Y
X⊖*Y*	Rotation by X along the first dimension of Y
⌽*Y*	Reversal along the last dimension of Y
⌽[*Z*]*Y*	Reversal along the Zth dimension of Y
⊖*Y*	Reversal along the first dimension of Y
X⍉*Y*	Transpose by X of Y
⍉*Y*	Ordinary transpose of Y
X,*Y*	Y catenated to X along the last dimension
X,[*Z*]*Y*	Y catenated to X along the Zth dimension, or laminations of X and Y
,*Y*	Ravel of Y (make Y a vector)
X↑*Y*	Take the first X (or last X if X is negative) elements of Y
X↓*Y*	Drop the first X (or last X if X is negative) elements of Y
X?*Y*	X integers taken randomly without replacement from integers up to and including Y
⍋*X*	Indexes of elements of X in ascending order

⍒*X*	Indexes of elements of X in descending order
⌹*X*	Inversion of matrix X
Y⌹*X*	Division of matrix Y by matrix or vector X
⍎*Y*	Execute the character vector Y as an APL expression
⍕*Y*	Format Y to be a character array identical to the normal display of Y
X⍕*Y*	Format Y using X to determine the field width and precision
X/*Y*	X (logical) compression along the last dimension of Y
X/[*Z*]*Y*	X (logical) compression along the Zth dimension of Y
X⌿*Y*	X (logical) compression along the first dimension of Y
X\Y	X (logical) expansion along the last dimension of Y
X\\[*Z*]*Y*	X (logical) expansion along the Zth dimension of Y
X⍀*Y*	X (logical) expansion along the first dimension of Y

Other APL Symbols

The symbols described below are used with APL in the sense indicated.

( )	Parentheses. Expression within them is to be evaluated before being used as the argument of a function.
→*X*	Branch to X.
X←*Y*	X specified by Y: the name X receives the value of Y.
⎕←*X*	Print the value of X.
X←⎕	Request evaluated input. Value of X is the resulting value after expression entered is evaluated.
X←⍞	Request literal input. Value of X is entire input text as characters, up to but not including carrier return.
⍞←*X*	Print the value of X, and stop at the end of the line.
⍝	Comment. Is the first character of a line of comments.
⍬	Interrupt the function making an input request. (Symbol formed by entering *O* backspace *U* backspace *T*.) On an IBM 3270 display device, this is accomplished by using a PA key.
∇	Used to open and close function definition.
⍫	Used to lock a function definition.
[⎕]	Used to print function when in definition mode.
[∆n]	Used to delete function line "n" when in definition mode.

System Functions

System functions are used to manipulate defined functions, groups, variables, and labels. System functions all require one or two arguments and return a result .

`⎕CR A`	Canonical representation of function whose name is contained in A.
`⎕FX M`	Establish (fix) function represented by character array M.
`⎕EX A`	Erase (expunge) objects named in A.
`⎕NL N`	List names of objects in the dynamic environment as follows: for N=1,2, or 3, list all labels, variables, or functions, respectively.
`A ⎕NL N`	Same as `⎕NL N`, except list only those objects whose names begin with the character(s) specified in A.
`⎕NC A`	Provide class of each name in A.
`⎕DL S`	Delay processing for S seconds.

A subclass of system functions consists of the shared-variable system functions which are used to monitor and control the status of variables shared with processors outside of VS APL.

`⎕SVO A`	Request the offer status of variables named in the right argument; that is, have these variables been offered or accepted for sharing.
`A ⎕SVO B`	Offer to share variables named in the right argument with a processor named in the left argument.
`⎕SVC A`	Request access control information for the variables named in the right argument.
`A ⎕SVC B`	Request that the access control indicated in the left argument be associated with the variables named in the right argument.
`⎕SVR A`	Stop sharing the named variable. (Retract the offer.)
`⎕SVQ A`	Request the name of any variables offered by the processor(s) named in the argument.
`A ⎕SVQ B`	Inquire whether the variables named in the right argument are being offered for sharing or are being shared, as indicated in the left argument.

System Variables

System variables are used to monitor and control the VS APL environment.

⎕*CT*	Comparison tolerance (used in ⌈ ⌊ < ≤ = ≥ > ≠)
⎕*IO*	Index origin (either 0 or 1)
⎕*LX*	Latent expression, which is executed when the workspace in which it resides is activated
⎕*PP*	Printing precision—maximum number of digits printed
⎕*PW*	Printing width—longer lines are folded and indented on printing
⎕*HT*	Tab settings
⎕*RL*	Random link—used in ? function
⎕*AI*	Accounting information—userid, CPU time, connect time, keying time
⎕*AV*	A character vector that contains the 256 possible character representations
⎕*TC*	Terminal control characters—backspace, new line, linefeed
⎕*LC*	Line counter—line numbers of functions in execution—innermost first
⎕*TS*	Time stamp—year, month, day, hour, minute, second, and millisecond
⎕*WA*	Working area available, in bytes

APPENDIX B. SUMMARY OF SYSTEM COMMANDS

The topics that follow summarize the system commands available in VS APL.

System commands are used to:

- Sign off the system.
- Control workspaces.
- Request reports on the contents of workspaces and libraries.
- Send messages to the operator or to another terminal user.

These commands are summarized in the topics that follow.

Note: Brackets ([ ]) indicate optional items.

Signing Off

)*OFF* [:password]	End work session and, optionally under VSPC and CICS/VS, change a sign-on password. Under CICS/VS, this command has the same effect as)*OFF HOLD*.
)*OFF HOLD* [:password]	End APL work session and hold connection to VSPC, CMS, or CICS/VS. Optionally, under VSPC and CICS/VS, change a sign-on password.
)*CONTINUE* [:password]	End work session and store active workspace. Optionally, under VSPC and CICS/VS, change a sign-on password. Under CICS/VS, this command has the same effect as)*CONTINUE HOLD*.
)*CONTINUE HOLD* [:password]	End APL work session, store active workspace, and hold connection to VSPC or CMS. Optionally, under VSPC, change a sign-on password.

Controlling a Workspace

Note: Under VSPC and CICS/VS, passwords apply to individual workspaces; under CMS, passwords apply to all workspaces residing on a virtual disk. "Password" is abbreviated to "pass" in the following command summary.

)*CLEAR* [size]	Activate a clear workspace and optionally set the workspace size.
)*SYMBOLS* n	Change the number of permitted names in a clear workspace.

)*LOAD* [lib] wsname [:pass][size]	Activate a copy of a stored workspace and optionally set the workspace size.
)*COPY* [lib] wsname [:pass] obj1 obj2...	Copy one or more global objects from a stored workspace.
)*COPY* [lib] wsname [:pass]	Copy all global objects from a stored workspace.
)*PCOPY* [lib] wsname [:pass] obj1 obj2...	Copy one or more global objects from a stored workspace, protecting active workspace.
)*PCOPY* [lib] wsname [:pass]	Copy all global objects from a stored workspace, protecting active workspace.
)*GROUP* groupname [objects]	Gather object names into a group or disperse group.
)*ERASE* object(s)	Erase global objects.
)*WSID* [lib] wsname [:pass]	Give identification to active workspace and, optionally, assign a password.
)*STACK* n	Change size of that portion of the workspace used by the system for execution control.
)*SAVE*	Save a copy of active workspace using its current identification.
)*SAVE* [[lib] wsname [:pass]]	Save a copy of the active workspace assigning a new identification, and optionally include a password.
)*DROP* [lib] wsname	Under VSPC or CICS/VS delete a stored workspace.
)*DROP* [lib] wsname [:pass]	Under CMS, delete a stored workspace, giving the password, if any.

Requesting Reports

)*FNS* [letter(s)]	List names of defined global functions beginning with the specified letter.
)*VARS* [letter(s)]	List names of global variables beginning with the specified letter.
)*GRPS* [letter(s)]	List names of groups beginning with the specified letter.
)*GRP* name	List members of designated group.
)*SI*	List halted functions (state indicator).
)*SINL*	List halted functions and associated

	local names.
)*WSID*	Give identification of active workspace.
)*LIB* [number] [letters]	Under VSPC or CICS/VS, alphabetically list names of workspaces in the specified library beginning with the specified letters.
)*LIB* [number] [letters][:pass]	Under CMS, list the names of workspaces in the specified library, giving the password, if any.
)*SYMBOLS*	Give current maximum for number of names, and the number in use.
)*STACK*	Give current size in entries of that portion of workspace used by the system for execution control.
)*WSSIZE*	List size of active workspace in bytes.
)*QUOTA*	Give information about library space, workspace size, and shared variables.

Sending Messages (under VPSC and CMS only)

)*MSG* userid [text]	Address text to designated user and inhibit further entry pending a reply.
)*MSG OFF*	Prohibit incoming messages.
)*MSG ON*	Accept incoming messages.
)*OPR* [text]	Address text to system operator and inhibit further entry pending a reply.

INDEX

VS APL General Information
GH20-9064-4

Reader's Comment Form

Note: Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

Fold on two lines, tape, and mail. No postage necessary if mailed in the U.S.A. (Elsewhere, any IBM representative will be happy to forward your comments.) Thank you for your cooperation.

GH20-9064-4

Reader's Comment Form

Fold and tape | Please do not staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
P.O. Box 50020
Programming Publishing
San Jose, California 95150

Fold and tape | Please do not staple | Fold and tape

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

VS APL General Information Printed in U.S.A. GH20-9064-4

GH20-9064-4

VS APL General Information Printed in U.S.A. GH20-9064-4

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