SH20-1460-1

Systems

APL Shared Variables (APLSV) User's Guide (Programming RPQ WE1191)

Program No. 5799-AJF

IBM

Second Edition (March 1975)

The programming RPQ described in this manual, and all licensed material available for it, are provided by IBM on a special quotation basis only, under terms of the License Agreement for IBM Program Products. Your local IBM branch office can advise you regarding the special quotation and ordering procedures.

This edition replaces the previous edition (Order No. SH20-1460-0) and makes that edition obsolete. This manual also obsoletes and replaces the APL Shared Variable TSIO Program Reference Manual (SH20-1463-0).

This edition, including changes released in Technical Newsletter SN26-0800, applies to Version 2.1 of the APL Shared Variable System (APLSV) (programming RPQ number WE1191; program number 5799-AJF) and to all subsequent versions unless otherwise indicated in new editions or Technical Newsletters.

Changes are periodically made to the specifications herein; any such changes will be reported in subsequent revisions or Technical Newsletters. Before using this publication, consult the latest IBM System/360 and System/370 Bibliography, Order No. GA22-6822, for the editions that are applicable and current.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Department J04, 1501 California Avenue, Palo Alto, California 94304.

PREFACE

This publication is intended for APLSV users and application programmers. It complements the description of APL given in the APL Language Manual (GC26-3847), and describes the TSIO auxiliary processor, a program which gives the user at an APL terminal interactive control of OS/VS Data Management facilities through Shared Variables.

This publication consists of the following sections:

Use of APL - Describes APL features particular to the APLSV system such as: number and character representations; values of system variables; bounds on the lengths of names and on the ranks of arrays; and additional system commands.

Use of TSIO Auxiliary Processor - Describes the auxiliary processor that provides for interactive use of many of the data management facilities of the OS/VS Operating System by the APL user. In order to take full advantage of all the facilities of the TSIO Auxiliary Processor, users may require further information on OS/VS data management. This information may be found in the publication OS/VS Data Management Services Guide (GC26-3783). In addition, the following publications should be consulted for information specific to the OS/VS operating system being used:

OS/VS1 JCL Reference (GC24-5099)

OS/VS2 JCL (GC28-0692)

Additional information about the use of magnetic tape can be found in the publications:

System/360 Component Description 2400-Series Magnetic Tape Units (GA22-6866).

3803/3420 Magnetic Tape Subsystems Component Description (GA32-0020).

Use of Distributed Workspaces - Summarizes the contents and the application of the workspaces distributed with APLSV.

SECTION 1: INTRODUCTION

The APL Shared Variable (APLSV) system is a program which runs on an IBM System/370 computer under one of the IBM Operating Systems OS/VS1 or OS/VS2. It provides facilities for the execution of programs written in APL and for communication between APL programs and other subsystems by means of shared variables. The program consists of:

- An APL Interpreter which executes the APL Language described in the APL Language Manual, GC26-3847
- An APLSV supervisor, which maps host-independent service requests made by the APL interpreter into calls for OS/VS system services, and distributes the central computer resources allocated to APLSV among the current APL users.
- The Shared Variable Processor, which provides the host independent services necessary to implement shared variables as described in the APL Language Manual. These services are available both to the APL programmer and to the system programmer who wishes to permit access from APLSV to portions of the computer system external to the APLSV facility.
- The Time Shared Input/Output (TSIO) auxiliary processor, which accepts commands through shared variables and permits users at APL terminals interactive control of standard OS/VS operating system data management facilities.
- A library management facility, used to make copies on tape of workspaces saved in APL libraries, either for archival storage or for export to other systems.

SECTION 2: USE OF APL

ACCESS TO THE SYSTEM

Each user of the APLSV system is assigned, by a system manager, an account number used to identify his data on external storage and his charges for use of the system. This account number is required in order for the user to sign on.

Once a connection to the computer has been established any input other than a sign-on is rejected with the report *INCORRECT SIGN-ON*.

The sign-on command consists of a right parenthesis followed by an account number, a colon, and a password, for example:

```
)8888:AGBDF
```

Note that the APL right parenthesis and the APL colon, whose positions are given in the APL Language manual, must be used. The system manager assigns an initial password. The user should change his password frequently.

Successful sign-on is acknowledged by a message showing the port number, the date and time, the user's name, and the system identification. This may be preceded by a broadcast message from the system operator.

At sign-on, the accumulation of time charges begins. A clear workspace is activated, except in the case that the preceding session was terminated by the command *)CONTINUE* resulting in the saving of an unlocked *CONTINUE* workspace. In that case, the workspace *CONTINUE* is activated and its latent expression executed.

An attempt to sign-on may be rejected, with one of the following reports:

INCORRECT COMMAND: The terminal is in use, and must first be signed off before a new sign-on can be accepted.

NUMBER IN USE: The indicated account number is in use at another port. This may arise either because another user is in fact using the account, or occasionally (perhaps because of an equipment problem) because the system did not complete disconnection at an earlier work session.

NUMBER LOCKED OUT: While the indicated account number and password are valid, authorization for use of the account has been withdrawn. This also prevents workspaces stored in the library of this account from being loaded or copied by other users.

REPRESENTATION OF DATA

CHARACTERS

The elements of the system variable ⎕*AV* are represented in APLSV as 8 bit bytes. The correspondence between the base 2 values of the bytes (0-origin indices to ⎕*AV*) and the APLSV characters, shown in Figure 1, was chosen to simplify implementation matters (for example, the letters are grouped together so that the test for alphabetic is performed with two comparisons). Certain elements of ⎕*AV* cause special actions, as their names in Figure 1 indicate, when transmitted to certain terminals. Elements 1 to 13 and 168 to 255 have no visible

effect on display. Functions that index □*AV* to generate specific characters may require modification for transfer to other APL systems.

NUMBERS

A number may be represented in an APLSV workspace as 8-byte floating-point, 4-byte integer, or 1-bit boolean.

Any number between ⌈/⍳0↔¯7.237*E*75 and ⌊/⍳0↔7.237*E*75 may be represented as floating-point. If the number is integral of magnitude less than 2*31 it may also be represented as an integer. Finally, if it is either 0 or 1, it may in addition be represented as boolean.

Non-integers or integers of magnitudes exceeding ¯1+2*31 are represented as floating-point.

The way a number is represented in the workspace depends on the way it was entered or which primitive function computed it. For example, if *X*←0*0 1 , then *X* and ⌊*X* and 1=*X* have the same value 1 0 , but are represented in floating point, integral, and boolean forms, respectively.

Numbers may be entered or displayed in decimal or scaled forms. Display form may be explicitly controlled by the use of the dyadic format function. If not specified by the user, the form chosen for the display depends on the representation of the number in the workspace. Numbers stored as integers are displayed in decimal form, and their display is independent of the setting of the printing precision □*PP*. On the other hand, the display of numbers stored in floating-point is dependent on □*PP*. These are displayed either in scaled form, or,

bs	backspace
ce	card eject*
cr	carrier return
ht	horizontal tab
il	idle
lf	line feed
rhlf	reverse half-lf*
tl	track link*
uc	upper case

LEGEND

*Magnetic Card Selectric Typewriter controls

0	il	32	⌊	64	∘	96	*K*	128	*P*	160	
1		33	|	65	□	97	*L*	129	*Q*	161	
2		34	∧	66	⍞	98	*M*	130	*R*	162	ht
3		35	∨	67	⊛	99	*N*	131	*S*	163	uc il
4		36	<	68	⍲	100	*O*	132	*T*	164	bs
5		37	≤	69	⍱	101	*P*	133	*U*	165	rhlf
6		38	=	70	⍝	102	*Q*	134	*V*	166	tl
7		39	≥	71	⍙	103	*R*	135	*W*	167	ce
8		40	>	72	⍒	104	*S*	136	*X*		
9		41	≠	73	⍳	105	*T*	137	*Y*		
10		42	α	74	⍀	106	*U*	138	*Z*		
11		43	∈	75	⍉	107	*V*	139	Δ		
12		44	⍴	76	⌹	108	*W*	140	0		
13		45	ρ	77	⍕	109	*X*	141	1		
14	[	46	ω	78	⍎	110	*Y*	142	2		
15	]	47	,	79		111	*Z*	143	3		
16	(	48	!	80		112	Δ	144	4		
17	)	49	⌽	81		113	*A*	145	5		
18	;	50	⊥	82		114	*B*	146	6		
19	/	51	⊤	83		115	*C*	147	7		
20	\	52	○	84		116	*D*	148	8		
21	←	53	?	85		117	*E*	149	9		
22	→	54	~	86	*A*	118	*F*	150	.		
23		55	↑	87	*B*	119	*G*	151			
24		56	↓	88	*C*	120	*H*	152	space		
25	¨	57	⊂	89	*D*	121	*I*	153	'		
26	+	58	⊃	90	*E*	122	*J*	154	:		
27	-	59	∩	91	*F*	123	*K*	155	∇		
28	×	60	∪	92	*G*	124	*L*	156	cr		
29	÷	61	_	93	*H*	125	*M*	157			
30	*	62	⍵	94	*I*	126	*N*	158	uc bs		
31	⌈	63	⍺	95	*J*	127	*O*	159	lf		

Figure 1: Atomic Vector (□*AV*)

provided they fall within the range from 10*¯4 to 10*⎕PP, in decimal form.

For example:

```
      ⎕PP←5
      ⌊¯1+2*31
2147483647
      ⌊2*31
2.1475E9
      .0001
0.0001
      .00001
1E¯5
      1E4
10000
      1E5
1E5
```

ARRAYS

An array *A* occupies a sequence of adjacent words in the workspace. These words contain, in particular, the rank of *A*, its size, and its elements in row-major order. The number of words (4 bytes) required to store an array *A* is given by the expression:

$$3+(\rho\rho A)+\lceil T\times\times/\rho A$$

where *T* is 2, 1, .25 or .03125 according to whether the elements of *A* are represented in floating-point, integer, character or Boolean form, respectively. The maximum rank of an array is 64. Every element of the array is of the same type.

If *A* is a numeric array of rank no less than 2, and has some element that would normally be displayed in scaled form, then the entire array *A* is displayed in scaled form. In addition, if any pair of elements of *A* differ by 10*4 or more in magnitude, then *A* is displayed in scaled form. If the length of the display of a vector, or of a row of a higher dimensional array, exceeds the printing width ⎕PW, the display is continued on subsequent lines, each indented by 6 positions.

SCANS OF ASSOCIATIVE FUNCTIONS

For any dyadic function α and any vector *X*, the α-<u>scan</u> of *X* (denoted by α*X*) yields a result *R* of the same shape as *X* such that *R*[*I*] is equal to α/*I*↑*X*. For example:

```
        +\1 2 3 4 5 6
1 3 6 10 15 21
```

For any associative function α the following definition of *R*←α*X* is <u>formally</u> equivalent to the definition *R*[*I*]=α/*I*↑*X*:

```
R[1]=X[1]
R[I]=R[I-1]αX[I] for I∈1↓⍳ρX
```

This definition requires only ¯1+ρ*X* applications of α (as compared to .5×(ρ*X*)×¯1+ρ*X*), and is the one actually used for associative functions. Because of the finite precision used in machine arithmetic the results of the two definitions may differ, and differ significantly, if the elements of *X* differ by many orders of magnitude. For example, compare

the last element of the scan with the corresponding reductions in the following case:

```
      ⎕PP←5
      X←1E6 ¯1E6 1E¯16
      +\X
1E6 0 1E¯16
      +/X
0
      +/⌽X
1E¯16
```

Therefore the scan (as well as reduction), specifically with respect to + and ×, should be used with care in work requiring high precision.

SYSTEM PARAMETERS AND LIMITATIONS

SYSTEM VARIABLES

The horizontal tab, ⎕*HT*, facilitates both entry and display at a terminal with tab stops. The meaningful values of ⎕*HT* are non-negative integer vectors and scalars which denote distances from the left margin. The value of ⎕*HT* in a clear workspace is the empty vector. Inappropriate values of ⎕*HT* result in the report ⎕*HT IMPLICIT ERROR* when ⎕*HT* is used by the system; that is, at the beginning of any display or when a tab is entered on input.

Effect of ⎕*HT* on entry:

While the carriage is within the range of positions indicated by ⎕*HT*, each tab entered is converted to a number of spaces equal to the difference between the carriage position and the next larger element of ⎕*HT*. Tabs entered while the carriage is positioned outside the range indicated by ⎕*HT* are not converted to spaces. In particular, when ⎕*HT* is empty, no tab conversion takes place.

Note that ⎕*HT* does not control the physical tab stops, nor does the system sense the actual tab stop positions set. Therefore, it is possible, when ⎕*HT* does not correspond to the actual tab stop positions, to lose visual fidelity.

Effect of ⎕*HT* on display:

Any sequence of spaces is converted to that combination of tabs, spaces and backspaces that that will minimize the display time. Spaces beyond the range of positions indicated in ⎕*HT* are not converted.

Regardless of the value of ⎕*HT*, the six spaces provided by the system to indent before the keyboard unlocks are never tabbed.

Figure 2 gives, for certain system variables, their value in a clear workspace and the range of values that will not result in an *IMPLICIT ERROR* report upon their use. (⎕*AV* is listed separately in Figure 1.)

Name	Value in a clear ws	Meaningful Values
□*CT*	1*E*¯13	not less than 0 and less than 1
□*IO*	1	0or 1
□*LX*	''	character scalar or vector
□*PP*	10	integers 1 through 16
□*PW*	120	integers 30 through 390
□*RL*	16807	integers 1 through (2*31)-1
□*HT*	⍳0	non-negative integers

Figure 2. System Variable Values

NAME LENGTHS

Names of workspaces, functions, variables and groups are formed of alphabetic (*A* through *Z*, *a* through *z*) and numeric (0 through 9) characters, and start with an alphabetic. Only the initial 11 characters of a workspace name, and the initial 77 characters of other names, are significant. Longer names may be entered, but the additional characters beyond these limits are not retained, and will be missing from subsequent display. The surrogate names of variables shared with TSIO (or of the variables if they are their own surrogates) may not exceed 15 characters.

User account numbers and workspaces are protected from unauthorized use by association with a password. Passwords are formed from alphabetic and numeric characters and may begin with either an alphabetic character or a number. Only the initial 8 characters of a password are significant.

The special rules of naming the OS data sets manipulated by the TSIO auxiliary processor are described in the section Data Set Identification and Location.

DOMAINS OF PRIMITIVE FUNCTIONS

The domains of the arithmetic functions are restricted by the constraints imposed by System/370.

For example, the upper limit of the argument to the exponential is ⍟⌊/⍳0, that is, approximately 174.673, and the upper limit of the argument to the factorial function is approximately 56.5452.

The domains of rank-increasing mixed functions and operators are restricted by the requirement that the ranks of their results fall within the APLSV range (not more than 64). For example, the sums of the ranks of the arguments to an outer or to an inner product operator are 64 or 65 respectively; the maximum sizes of the left arguments to the reshape function and the take function are 63 and 9 respectively; the maximum rank of an argument to the lamination function is 24; the maximum rank of an array to be indexed is 14.

The processor identification, which is the argument to *⎕SVQ* and the left argument to *⎕SVO*, is a nonnegative integer less than 2*31. The delay, argument to *⎕DL*, must be less than (2*31)÷300.

WORKSPACE ATTRIBUTES

The number of names the symbol table can accomodate is at least 26, and is 256 in a clear workspace. Each name occupies 8 bytes in the table.

The size of a workspace in an APLSV system is the same for all users and is determined at the time the system is generated.

USER ACCOUNT ATTRIBUTES

Features that vary according to the individual user, such as library quota, shared variable quota, and the CPU limit (maximum time allotted to a single input line) can be changed, on request, by the system operator.

ADDITIONAL SYSTEM COMMANDS

Private APLSV libraries have user account numbers greater than 1000. Libraries with account numbers smaller than 1000 are public.

In addition to the system commands listed in the APL Language publication, APLSV has commands for communication between users or between a user and the operator; and for reporting and modifying the environment.

In the text that follows, each system command is shown in a sample form. In use, the appropriate names or numbers should be substituted in the samples for the following:

A	A letter of the alphabet
N	A number
USERCODE	An abbreviation of user's name
OBJNAME	A name of an object within a workspace
[ ]	Items enclosed in brackets may be omitted

)MSG N [TEXT]

Transmits one line of text to the port indicated by *N*. If *N* does not indicate an active port, the message is reflected to the port from which it originated. When the text has been dispatched, the system responds *SENT*. The sending terminal accepts no further input, except for incoming messages, until either a reply is received or the keyboard is freed. The message displayed at port *N* shows the number of the port from which it was sent and the symbol *R*, meaning "reply expected."

If no reply is received, the keyboard may be freed for new input by signalling a weak interrupt. However, if the user signals before the message has been sent, the message is lost, and the system reports *MESSAGE LOST*.

)MSGN N [TEXT]

Sends a message to port *N*, but does not inhibit further entry while awaiting a reply.

The message received at port *N* shows the number of the port from which it was sent, but without the symbol *R*.

)OPR [TEXT]
)OPRN [TEXT]

Sends a message to the system operator (recording terminal) in the same way as *)MSG* or *)MSGN* (described above). These commands may also be used from a terminal before the sign-on to APL is completed.

)MSG OFF

Prevents the delivery of messages, except for a reply to a previous *)MSG* or *)OPR* command. Following use of *)MSG OFF*, the user's port appears inactive to those attempting to send messages to it. That is, messages are reflected back from this port to their sources.

)MSG ON

Restores the acceptance of messages suspended during the previous *)MSG OFF* command, and delivers the last broadcast message, if any, sent by the system operator.

)PORTS [N]

Lists in order the numbers of ports signed on to the same APL system as the user [optionally, starting at port *N*], with the first three letters of the account name of each.

)PORT USERCODE

Lists the port number(s) of the user whose account name starts with *USERCODE*.

ORIGIN, *WIDTH* and *DIGITS*

)ORIGIN N

Sets the global value of the index origin (□*IO*) as indicated. Valid use of the command results in the message *WAS ...* showing the former value. An argument other than 0 or 1 causes the response *INCORRECT COMMAND*.

)WIDTH N

Sets the global value of the printing width (□*PW*). Valid use of the command results in the message *WAS ...*, showing the former value. An attempt to set a value that is not a positive integer, or a value less than 30 or greater than 390 results in the message *INCORRECT COMMAND*.

)DIGITS N

Sets the global value for the number of significant digits appearing in unformatted numeric output (□*PP*). This has no effect on the precision of internal calculations, nor on the display of function definitions. An attempt to set a value which is not an integer from 1 through 16 results in the message *INCORRECT COMMAND*. Valid use of the command results in the message *WAS ...*, showing the former value.

FUNCTION EDITING

STATEMENT DELETION

A statement is deleted by following the bracketed statement number by attention (or linefeed) and a carrier return, and nothing else: a carrier return alone has no effect.

DUPLICATE LOCAL NAMES

Duplication of labels and of the names within a function header is not checked and will generally produce undesired results.

PENDENT OR SUSPENDED FUNCTIONS

A pendent function cannot be edited. An attempt to edit the header line of a suspended function, whether it results in an actual change or not, produces the error report *SI DAMAGE* immediately. An attempt to edit a label in a suspended function produces the *SI DAMAGE* report when function editing mode is ended. An *SI DAMAGE* report is issued only once for each editing session of the function.

SECTION 3: USE OF TSIO

PRINCIPLES OF OPERATION

The TSIO auxiliary processor makes available to the APLSV user, through the use of shared variables, many of the data management facilities of the OS/VS operating system. This section describes how to make effective use of these facilities for operations such as the construction and use of files outside of an APL workspace, and certain types of high speed input and output.

In the host system, all operations involving input and output or auxiliary storage are said to concern data sets. Initially names are assigned to collections of data that are to be created on a storage device. The resulting data set can be accessed at a later time by referring to its assigned name. A data set is said to be opened when the necessary preparation for accessing it is complete. This usually involves the deployment of programs and storage facilities for managing data transfer between the storage device and the computer memory. When transfer operations are finished, the facilities are made available for other uses and the data set is then said to be closed.

Data sets are composed of records grouped in blocks, which are the units of information actually transferred between the storage device and the computer memory. A block may consist of only one record or of many. When a data set is constructed exclusively with blocks containing only a single record, the data set is said to be unblocked.

The information required by the system to open a data set varies, depending chiefly on whether the data set is new or old, the complexity of its format, and the acceptability of certain default options provided by the system. In the APL environment, this information is provided by the user in the form of a character vector, which TSIO translates into a form understandable by the operating system. Responses by the operating system are accepted by TSIO, which sends them on to APL. A single shared variable, called a control variable, is used for this two-way communication.

Once the data set has been opened, data transfer can start, using either the control variable or, in certain cases, an associated shared data variable. Data transmission is mediated by TSIO, which may remove or supply the header - the descriptive information that is part of the internal representation of all APL data. TSIO also monitors the control variable for the appearance of an empty vector, which is used to signal the end of data transmission and the closing of the data set. A retraction of the control variable for any reason will also end the operation in progress and force the data set to be closed.

COMMUNICATION WITH TSIO

Communication with the TSIO processor is established through a shared variable whose (surrogate) name must begin with the characters *CTL*. This variable is called the control variable. In indexed reading and writing, a second shared variable is required and its surrogate name must begin with *DAT* and conclude with the same suffix used in the control variable name. This variable is referred to as the data variable. Several different pairs of data and control variables may be shared with TSIO, the pairing indicated by the name suffixes as in *CTLB* and *DATB*, or *CTL*5 and *DAT*5. Each pair is treated independently by TSIO.

The identification number assigned to the TSIO processor may differ with each installation. However, as with all auxiliary

processors it must be identified by a positive number. Using the variable *T* to indicate this assigned value, the control variable can be shared as follows:

```
      T ⎕SVO 'CTL'
1
```

From here on the term interlock will be used for the access control imposed on a shared variable. Even though TSIO places a full interlock 1 1 1 1 on *CTL* as soon as it accepts the offer, this acceptance may not be immediate. It is desirable that this acceptance and interlock be established by the APL user before further action is taken. This can be insured as follows:

```
      1 1 1 1 ⎕SVC 'CTL'
1 1 1 1
```

Except for the explicit synchronization provided by a full interlock on the control variable, the operations of TSIO and the data management system are independent of, and not synchronized with, the execution of any APL program which communicates with it.

CONTROL VARIABLE MODES

Since a control variable may be used for data transfer as well as for the initiation of operations, a protocol is defined which allows TSIO to determine which type of information is being communicated. The value of the control variable will be interpreted by TSIO to be a command only when the control variable is in command mode, which occurs:

- Immediately after the user's offer to share the control variable has been matched by TSIO.
- Immediately after the execution of commands which request a one-step operation, such as renaming or deleting a data set.
- After the termination of data access, as indicated by the user or by TSIO while in data transfer mode.

The value of the control variable will be interpreted by TSIO to be data to be transferred only when the control variable is in a data transfer mode. Data transfer mode is established by the successful execution of a command which requests data set access. Data transfer is ended as follows:

- Termination of data access is signalled by assigning an empty vector to the control variable, or by explicit or implicit retraction of that variable.
- TSIO will normally terminate data transfer only in a sequential reading operation. In this case, the next use of the control variable after the last successive record has been read yields an empty vector; the following use yields the numeric return code resulting from the end of the read operation; and the control variable is in command mode.
- TSIO will also terminate data transfer during a sequential write operation if certain errors are encountered. In this case the control variable yields a numeric return code immediately, and the control variable is in command mode. When the data set is closed, TSIO retracts the control variable.

DATA MANAGEMENT COMMANDS

A command is a character vector that begins with a word (or abbreviation) designating the operation to be performed, followed by a

space and a sequence of one or more items separated by commas. The items in a command can appear in any order. They are used to establish the parameters which govern the attributes or the usage of the data set. In the simplest case, an item consists of a keyword followed by an equal sign followed by a value, which may be a name, number or another keyword. The commas and the equals sign may have any number of spaces on either side. For example:

```
CTL←'SW DSN=A1, DISP=NEW, BLKSIZE = 550'
```

One item must always identify the data set (for example, *DSN=A1*), and other items either describe physical attributes of the data set (for example, *BLKSIZE=550*), or further describe the function to be performed on it (for example, *DISP=NEW*).

When a parameter value has multiple fields, the significance of each field is determined by its position. A convention involving both commas and parentheses is used for representing the several values. Since only four parameters are affected by these conventions, the details are given for each as its use is explained in the text.

Eleven data management operations are provided by TSIO, of which four (*CATALOG, UNCATALOG, READVTOC, MSG*), are restricted to those users identified to TSIO as system level users, to be described in a later section. The seven data management operations available to all users of TSIO are shown in Figure 3.

ABBREV.	OPERATION	MEANING AND USAGE
SW	Sequential Write	Create a new data set, and rewrite or append to an existing data set.
SR	Sequential Read	Read records sequentially from an existing data set.
IRW	Indexed Read and Write	Read and write records in arbitrary sequence from or to an existing data set.
IR	Indexed Read	Read records in arbitrary sequence from an existing data set.
RENAME		Change the name of an existing data set.
DELETE		Delete an existing data set.
IC	Indirect Command	Execute a prepared command from a command data set.

Figure 3. Data Management Operations

RESPONSE CODES

The assignment of a command to the control variable in command mode results in either the successful performance of the indicated operation or in failure. In either case, the value of the control variable is a response code which indicates either that the command has been successfully executed or the reason for failure. A zero indicates success; the significance of other codes is given in Figure 4. Response codes are also used during data transfer as shown in Figure 5.

	CODE	NAME	RESPONSE TO	CAUSE, RESULT, AND CORRECTIVE ACTION
	0	Normal	Anything	Success
	1*	Imparsible command	Anything	Parse of command failed.
	2*	Restricted command	Anything	Operation, parameter, or parameter value requires different level authorization.
PARAMETERS	3	*DSNAME* error	Command	*DSN* missing or bad name.
	4	*BLKSIZE* error	*S I*	*BLKSIZE* missing or too large.
	5	*LRECL* error	*S*	*FB*, *VB* need *LRECL*; or *LRECL>BLKSIZE*.
	6	*DISP* error	*SR I*	*DISP=NEW*
	7	*RECFM* error	*I*	*RECFM≠F* (In *IRW*, *RECFM* is determined by the data set.)
	8	*UNIT* error	Command	*CATALOG* needs *UNIT*, or data set not cataloged, or no such *UNIT*.
	9	*VOL=* required	Command	The combination *DSN*, *UNIT*, and *DISP≠NEW* needs *VOLUME*.
	10	*NEWNAME* error	*REN*	*NEWNAME* missing or *≠DSN* for PDS.
	11	Duplicate name	*DISP=NEW*	*DISP=NEW* and name already used.
	12	Dataset not found	Command	*DISP≠NEW* and name not found.
	13	Member not found	*S D REN*	Member of a PDS not found.
RESOURCES	14	I/O buffers full	*S I*	Available space in TSIO region smaller than buffer. See Note 1.
	15	Data set in use	Command	Data set in use with *DISP≠SHR*.
	16	Volume full	*SW*	Insufficient space for primary allocation.
	17	*PDS* directory full	*SW*	Attempting to write a new member when directory of a PDS is full.
	18	Volume unavailable	Command	DASD volume not mounted or request to allocate unit record device denied.
	19	Already cataloged	*SW REN*	(With *DISP=NEW*) Name already cataloged or name conflict as in *A.B.C* and *A.B*
	20	Control variable in use	Command	Last used with *DISP* of *LEAVE* or *REREAD*. Use new variable or retract and reshare.

LEGEND AND NOTES

D DELETE *I IR* or *IRW* *S SR* or *SW* *REN RENAME*	PDS Partitioned Data Set DASD Direct Access Storage	* These codes have a second element that shows the point of difficulty in the command.	1. Try later or ask operator to allocate more space.

Figure 4: Response in Command Mode

CODE	NAME	DURING	CAUSE ACTION (None unless stated)
0	Normal	*S I*	Success.
21	Data type error	*S I*	Data type not appropriate to *CODE*. *SR*: Data set closed; to command mode.
22	Data length error	*S I*	*RECFM=F,FB* and *RL*≠ (for *CODE*≠*A*) or <(for *CODE=A*) *BLKSIZE* or *LRECL*, respectively, or *RL>BLKSIZE*. *SR*: Data set closed; to command mode.
23	Data rank error	*W*	*CODE*≠*A* and data not a vector.
24	File index error	*I*	Index ≥ number of records in data set.
25	*CTL* domain error	*I*	*CTL* not a 2-element non-negative integer vector with leading 0 1 2 3 4 or 5.
26	*DAT* variable required	*I*	*DAT* variable (with suffix = *CTL* suffix) not shared on first data transfer attempt.
27	Variable too large for SVS	*R*	Shared variable storage area too small. *SR*: Data set closed; to command mode. *CODE*≠*A*.
28	I/O error	*S I*	Physical error in data transfer. *SR*, *SW*: Data set closed; to command mode *IRW* (writing): Record may have been destroyed.
29	Data set full	*S W*	16 extents have been allocated and filled, or no space on volume for a needed secondary allocation, or primary allocation filled and no secondary allocation specified. Data set closed; to command mode.
30	System error	*S I*	See legend for *. All relevant terminal output should be given to the system manager.

CTL Control Variable
DAT Data Variable
RL Record length
S *SR* or *SW*
I *IR* or *IRW*
R Reading
W Writing

* Second element of response code (for code 30) has the following significance:

1. VTOC full
2. Allocation failed
3. DD card missing
4. System queue error
5. System queue full
6. Directory error
7. *CATALOG* failed
8. *OPEN* failed

(Also see APLSV Operations Guide, IBM Publication SH20-1461.)

Figure 5: Response in Data Transfer Mode

SEQUENTIAL READING AND WRITING

Data sets can be created and extended by sequential writing only. They may be accessed sequentially by the commands *SW* and *SR* (see Figure 7). The following example shows the creation, reading, extension, and rewriting of a data set:

```
      T □SVO 'CTL'          Share control variable with processor T
1
```

```
      1 1 1 1 ⎕SVC 'CTL'      Impose access control on (i.e. interlock) the
1 1 1 1                         shared variable

      CTL←'SW DSN=A1,BLKSIZE=550,DISP=NEW'
                              Issue command to create data set
```

The value associated with *DSN* is the name of the data set. The *BLKSIZE* parameter value specifies the largest size of a block within the data set. The value of *DISP* indicates that the data set is to be created. This is the minimum information required for the creation of a data set.

```
      CTL
0                             A zero response indicates that the command
                                has been successfully completed.
```

Data transfer can now take place:

```
      CTL←⍳8                  Value for first record is transferred.
      CTL
0                             Response indicates record has been
                                successfully written.
      CTL←2 3⍴'ABCDEF'        Value for second record is transferred.
      CTL                     Display of response.
0
      CTL←''                  Empty vector requests return to command mode.
      CTL
0                             Response indicates that writing terminated
                                normally and the data set is closed.
```

The display of responses from TSIO can be omitted without altering the essential behavior of TSIO; output data transfer is initiated as soon as *CTL* is set, and input as soon as it is used. However, it is good practice to read and check each response, so that error indications can be used to identify trouble, and meaningful continuation will be assured.

The records in the data set *A1* created in the previous example can now be read sequentially as follows:

```
      CTL←'SR DSN=A1'         Open A1 for sequential reading.
      CTL                     Response to command.
0
      CTL                     First record is read and displayed.
1 2 3 4 5 6 7 8
      X←CTL                   Second record is read.
      X
ABC
DEF
      CTL                     Response from TSIO is an empty vector,
                                indicating that it has been returned to
                                command mode by the reading of the final
                                item in the data set.
      CTL
0                             Response indicates that the reading
                                terminated normally and the data set is
                                closed.
```

Further records can now be appended to the end of the data set, as follows:

```
      CTL←'SW DSN=A1,DISP=MOD'    Open A1 in such a way that access
0                                   starts after the last record.

      CTL←⍳3                      Append a record
      CTL
0
      CTL←''                      Request return to command mode
      CTL
0
```

At this point *A1* has three records, each containing an APL value. To rewrite the data set from its beginning, the *SW* command is used omitting the *DISP* parameter:

```
      CTL←'SW DSN=A1'             DISP default value is OLD
      CTL
0
```

The above command erases the records in *A1*, while still retaining the identity of *A1* as a data set. The next data item transferred will become the first record in *A1*. For example:

```
      CTL←'A FRESH START'
      CTL
0
      CTL←''
      CTL
0
```

The data set *A1* now contains one record only.

INDEXED READING AND WRITING

Data sets can be created and extended by sequential writing only, but records of existing data sets can also be accessed non-sequentially, by means of the commands *IR* and *IRW*. Indexed access makes use of two variables shared with TSIO: the control variable, whose (surrogate) name must begin with *CTL*, and a data variable, whose (surrogate) name must begin with *DAT*.

This method of access can be used only for data sets containing uniform length, unblocked records, which are produced by an *SW* command in which the *RECFM* parameter is *F* (fixed and unblocked) (but note that the workspace 1 *APLFILE*, described later, permits indexed access of APL variables of arbitrary rank and shape).

The following example shows the sequential creation of a data set and its indexed use:

```
      T ⎕SVO 'CTL'                Share control variable with processor T
1
      1 1 1 1 ⎕SVC 'CTL'          Impose access control on the shared variable
1 1 1 1
      CTL←''                      Enter command mode

      CTL←'SW DSN=A2,BLKSIZE=550,DISP=NEW,RECFM=F'
                                  Create RECFM=F data set
      CTL                         Test for successful opening.
0
      CTL←100                     Write first record
      CTL
0
```

```
      CTL←200                 Write second record
      CTL
0
      CTL←300                 Write third record
      CTL
0
      CTL←''                  Terminate writing.
      CTL
0
      T ⎕SVO 'DAT'            A data variable must be shared sometime
                                before indexed access actually begins.
1
      CTL←'IRW DSN=A2'        The data set is opened for indexed access.
      CTL
0
```

Once the data set has been opened, reading or writing is signalled by assigning a two-element control vector to *CTL*. The first element of this control vector is the operation code, which may have a value of 0, 1, 2, 3, 4, or 5. When the operation code is 0 or 1, signifying a read or write, the second element of the control vector specifies the index of the record to be accessed -- always in 0-origin.

The use of the operation codes for read and write is illustrated in the example here; the significance of control vectors using other operation codes is discussed later, in the section on Shared Data Sets.

During indexed access all synchronization is managed by the interlock on *CTL*. When reading, *CTL* must be referenced after being set, to ensure that *DAT* has the proper value. When writing, *CTL* must be referenced before the new value is placed in *DAT*, in order to be sure that the last operation has been completed; and only after *DAT* has been specified should *CTL* be set.

```
      CTL←0 2                 Read record number 2.
      CTL
0                             Response indicates that the operation is
                                completed, and DAT has the proper value.
      DAT
300
```

To assign the value '*X*' to record number 2 of *A2*:

```
      DAT←'X'
      CTL←1 2
      CTL
0
```

It should be noted that a continuation such as:

```
      CTL←1 1
      CTL
0
```

would assign the same value '*X*' to record number 1 as well, since *DAT* remains shared (and not interlocked) and TSIO uses the current value whenever *CTL* indicates a record is to be written.

To open the data set for indexed access for reading only, for example to protect it against accidental revision or to allow several users to read it concomitantly, the sequence might be:

```
      CTL←''
      CTL
0
```

```
      CTL←'IR DSN=A2'
      CTL
0
      CTL←0 2
      CTL
0
      DAT
300
```

and so on.

Access by *IRW* and with parameters *BLKSIZE*, *RECFM* or *LRECL* different from the ones with which the data set has been created will alter the values of these parameters, whereas access by *IR* will not. Further, if a data set created for use with indexed read and write is to be extended by a sequential write operation with *DISP=MOD*, the parameters *BLKSIZE*, *LRECL*, and *RECFM* should be omitted, so that those already in existence will remain in effect.

USER LEVELS

Any APLSV user having a non-zero shared-variable quota can offer to share variables with TSIO if the user knows its identification number, but TSIO will accept offers only from users whose account numbers are known to it. Each user thus authorized to use the processor is permitted usage qualified further by a combination of the four discriminants detailed in Figure 6 below. The set of discriminants associated with a given user is referred to as the user's level.

DISCRIMINANT	USER QUALIFICATION
SPACE	Allowed to create new direct access data sets, which implies the allocation of storage space.
DEVICE	Allowed to use the *UNIT* and *VOLUME* parameters, which implies the allocation of specific devices or storage units.
ACCESS	Allowed indexed access or sequential reading of other TSIO users' non-reserved data sets, given a knowledge of their identification.
SYSTEM	Has use of commands beyond the seven available to all users, and access to any data set, including those vital to system operation, within the constraints of the operating system security provisions.

Figure 6. User Levels

These qualifications are used to control the use of system resources, assure the integrity of the system, and serve the interests of privacy.

DATA SET ORGANIZATION

TSIO directly supports OS/VS data sets organized in three ways:

Sequential in which records can be accessed only in the sequence of their (linear) physical relationship. This organization is available on all storage devices, and is mandatory for tapes and unit record devices.

Direct in which records may be accessed in arbitrary sequence (once they have been sequentially established). Available only on direct-access devices.

Partitioned in which independently accessible groups of sequentially organized records are identified by compound names with a common first part. Available only on direct-access devices.

DATA SET PRIVACY

Provision for enhanced security of data sets created under TSIO is embodied in the use of reserved data sets with controlled access. A data set is said to be reserved to a user when the name of the data set is qualified by the negative of the user's account number. This user can access the data set freely, but other users can access it on a restricted basis only, as follows:

1. A data set reserved to a particular user can be accessed by other users directly, but only for the purpose of obtaining access to another data set reserved to that user. It is convenient to refer to a reserved data set used in this way as a command data set.

2. A data set reserved to a particular user can be accessed by other users for reading and writing only indirectly, through the mediation of a command data set reserved to that user.

However, batch jobs, which always execute at the system level of privilege, can access any TSIO data set.

Consider the case in which user 1234 wishes to authorize user 5678 for indexed reading of the reserved data set ¯1234 *SECRETS*. To accomplish this, user 1234 stores an image of the command *IR DSN=¯1234 SECRETS* as, say, the eleventh member of the reserved data set ¯1234 *GATEKEEP*. Subsequently, after sharing a control variable with TSIO in the normal way, user 5678 may submit a request for an indirect operation, using an *IC* command:

```
CTL←'IC DSN=¯1234 GATEKEEP (11)'
```

On receipt of this command, TSIO will first determine that ¯1234 *GATEKEEP* has the proper format for a command data set, then check the authorization matrices within it to determine whether user 5678 is permitted to utilize command form 11. Only if both of these checks are satisfactory will TSIO execute the read command on behalf of user 5678, who may then proceed with normal reading of ¯1234 *SECRETS*.

If the data set ¯1234 *GATEKEEP* does not exist, or if it is not properly constructed as a command data set, or if user 5678 has not been authorized to use the eleventh command, then TSIO issues the return code 2 1 to user 5678.

A command data set must have *CODE=A*, *RECFM=F*, *BLKSIZE=320*.

Record 0 of the command data set must be an integer vector with the following elements:

```
[0]  First authorization matrix record number (typically 97)
[1]  Last authorization matrix record number
[2]  Maximum number of rows in authorization matrices (maximum 19)
[3]  Last block allocated for authorization matrices
[4]  Number of authorized user entries, total for all
     authorization matrices
[5]  Creator's user identification
[6 7 8 9 10 11]  Year, month, day of month, hour, minute, second
     of last update of the data set
```

Records 1 to 96 of the data set provide for the indirect commands. From record 97 on the records are authorization matrices of 128 columns and of a number of rows no greater than element [2] of record 0.

Columns 0 to 31 of these matrices contain the (32 bit) twos-complement representations of user identification numbers. Columns 32 to 127 correspond to records 1 to 96 of the data set: if the element in row *I*, column 31+*J* is 1, the command in record *J* may be executed on behalf of the user identified in columns 0 to 31 of row *I*.

An entry for user 0 represents a general authorization for users other than those listed in the authorization matrices. A 1 in column 31+*J* of this row permits such users to execute the *J*-th command.

Each authorization matrix except the last must have the maximum number of rows as given by the element [2] of record 0. The rows of all the authorization matrices, considered as one matrix, must be in order by the base-two values of columns 0 to 31.

Command data sets are created and maintained by means of APL functions distributed in the workspace 1 *TSIO*. The necessary instructions for their use are found under *ICDESCRIBE* in that workspace.

SHARED DATA SETS

There are many applications in which the same data set must be used asynchronously by a number of different users. To a certain extent this can be managed without conflict by appropriate use of the disposition parameter, but this alone may not be completely satisfactory. On the one hand, if the data set is opened for exclusive use by one user, no matter how small the portion of the records actually involved, no other user can access any part of it until the data set has been closed and reopened. On the other hand, if the data set has been opened for shared use, except for the simplest case, in which all the sharers are only reading, some further mechanism is necessary to provide more dynamic control and finer resolution of the shared use.

Such a mechanism is provided by the operation codes 2 3 4 and 5, which may be used in the control vector (*CTL*) when a data set has been opened for indexed access with *DISP=SHR*. When one of these operation codes is used, the second element of the control vector may have any non-negative integer value. Except for zero, the significance of this value is completely arbitrary, although commonly it may be a record number or a coded identification of a set of records.

The operation codes 2 and 3 signify that the user wishes to have exclusive use of the facility denoted by the second element of the control vector, either immediately (2) or whenever it becomes available (3). The operation codes 4 and 5 signify that the user wishes to have non-exclusive use of the facility, -- on a shared basis, simultaneously with other users -- either immediately (4) or whenever it becomes available for such use (5). In any case, once use of a facility has been acquired, the user is said to have an exclusive hold or a shared hold on that facility.

If more than one control variable has been attached to a data set, each one may independently have a hold on a facility associated with that data set. An existing hold associated with a control variable is automatically cancelled by any subsequent use of one of the operation codes 2 3 4 5, whether or not the new request is fulfilled. An existing hold can also be cancelled explicitly by using a zero for the second element of the control vector with any of these operation codes.

Consider for example the following function, several instances of which are assumed contending to update records 100-120 of *DATASET*:

```
      ∇ F;I
[1]   TCTL 'IRW DSN=DATASET'
[2]   TOCTL 3 7
[3]   I←0
[4]  L1:→(120<99+I←I+1)/L2
[5]   DAT←RECORD I
[6]   TOCTL 1,I
[7]   →L1
[8]  L2:TOCTL 3 0
      ∇

      ∇ TOCTL X;E
[1]   CTL←X
[2]   →(0=E←CTL)/0
[3]   'TSIO RESPONSE: ',▼E
[4]   ⍬
      ∇
```

The updating is performed by the function *RECORD*. The contender that achieves exclusive hold on facility 7 of *DATASET* will next update records 100-120.

If a request for an immediate hold on a facility cannot be fulfilled because the facility is being held by another user, the control variable is returned with the value 15. Note that a request for exclusive use may be denied either because another user has an exclusive hold or a shared hold, and that a request for a shared hold may be denied only if another user has an existing exclusive hold. A request for a hold with codes 3 or 5, indicating that the user can wait, will remain in force until the facility becomes available for the type of hold requested, unless cancelled by retracting the variable. The fulfillment of the request is sgnified by the value 0 in the control variable, which is, as usual, interlocked against reading until TSIO respecifies its value following receipt of the request.

Two points are worth noting: First, the use of these operation codes does not automatically guarantee protection against conflicting use or other problems associated with shared facilities. It merely provides the potential for such protection to be incorporated in an application program. Thus, for example, there is no built-in constraint against the use of the 0 and 1 operation codes for reading or writing into a record which is otherwise part of a facility held by another user.

Second, because of the absence of such built-in constraints, the definition of the facilities denoted by the second element of the control vector when using an operation code of 2 3 4 or 5 is completely up to the application program, and need not refer in any way to the data set in use. In such a case the data set acts as a communication device with certain useful properties. The application will be making use of the built-in queuing facilities of TSIO, as well as the properties of the operation codes, to control the use of some common resource.

TSIO COMMAND PARAMETERS

Figure 7 shows parameter usage for each of the generally available operations, for parameters of general interest. Usage of parameters applicable only to tapes, or whose use is restricted, is given in appropriate sections. Where the table entry is C (for caution), the use of the parameter is not required, may sometimes be useful for a special purpose, but could result in damage to the data set or its label. In case of doubt, the publications mentioned in the preface should be consulted.

	DSN	*DISP* (2)	*NEWNAME*	*CODE*	*RECFM* (1)	*LRECL* (1)	*BLKSIZE* (1)	*SPACE* (2)	*SYSOUT* (2)
SW New	R	R		O *A*	O *U*	R for blocked data	R	O (4)	E
Not New	R	O *OLD*		O *A*	C L	C L	C L		E
Tran-sitory	E			O *E*	O *VB*	O 137	O 689	O (4)	R
SR	R	O *SHR*		Q *A*	O L	O L	O (3) L		E
IR	R	O *SHR*		Q *A*	C	C	C		E
IRW	R	O *OLD*		Q *A*	C	C	C		E
RENAME	R	C *OLD*	R						
DELETE	R	C *OLD*							
IC	R	E	E	E	E	E	E	E	E

LEGEND

USE

C With caution
E Error
O Optional
R Required
Q Required for *CODE≠A*
Blank-Ignored

DEFAULT

L From label or record Parameter values (in APL font)

OPERATIONS

SW Write sequentially
SR Read sequentially
IR Read with index
IRW Read and write with index
RENAME Change the data set name
DELETE Expunge the data set
IC Command indirectly

NOTES

1. See Figure 5.
2. Parameter value may be compound.
3. *BLKSIZE* is required for data sets that do not have standard labels.
4. (1000,(100,0)) for sequential data set

   (1000,(100,0,5)) for partitioned data set

(See text for *DEN*, *EXPDT*, *LABEL*, *TRTCH*, *UNIT*, and *VOLUME*)

Figure 7. Data Set Operations, Parameter Usage, and Default Values

Figure 8 summarizes the command parameters in the style of the OS/VS JCL Reference Manual, so that the reader may compare TSIO keywords to JCL keywords.

PARAMETER USAGE	COMMENTS
[*BLKSIZE*=block size]	Required with *DISP=NEW* or *LABEL=NL* or *BLP*
[*CODE*= { *A* / *B* / *C* / *E* / *F* / *I* }]	Required when accessing non-APL data sets
[*DEN*= { 0 / 1 / 2 / 3 / 4 }]	Used only with *UNIT*=tape and *DISP=NEW*
[*DISP*=({ *NEW* / *OLD* / *MOD* / *SHR* } [,{ *LEAVE* / *REREAD* }])]	See Figure 7 for defaults
{ *DSN* / *DSNAME* }=[[-]user no.]dsname[(member)]	Required except for *READVTOC* or *SYSOUT*
[*EXPDT*=yyddd]	Only with *SW DISP=NEW*
[*KEYLEN*=keylength]	Only with *DISP=OLD* or *SHR* and *UNIT*=direct access
[*LABEL*=(data set seq no.[,{ *BLP* / *NL* / *SL* }])]	Only with *UNIT*=tape
[*LRECL*=record size]	With *RECFM=FB* and *DISP=NEW* or *LABEL=BLP* or *NL*
[*NEWNAME*=[[-]user no.]dsname[(member)]	With *RENAME* command
[*RECFM*= { *U* / *F* / *V* / *FB* / *VB* / *FBS* } [*A* / *M*]]	Only with *DISP=NEW* or *LABEL=BLP* or *NL*
[*SPACE*=(blocklength,(primary[,secondary[,directory]]))]	
[*TRTCH*={ *C* / *E* / *T* / *ET* }]	Seven track tape only Normally not required
[*UNIT*= { device type / group name / unit address }]	
[{ *VOL* / *VOLUME* }=[*SER* volume serial	Use prevents cataloging a non-TSIO data set
[*SYSOUT*=(output class[,[pgm name] [,form number]])	Consult local System Administrator

Figure 8. TSIO Command Parameters.

DATA SET IDENTIFICATION AND LOCATION (*DSN, UNIT, VOLUME*)

When a data set is created on a direct access device, information about it is automatically stored in a label on the same volume of storage - that is, on the disk pack or drum - that contains it. Label data includes control information such as space allocation and data set format. For some formats (*F, FB*), all the control information is in the label; for others it is partly distributed throughout the data set, either explicitly (*V, VB*) or implicitly (*U*).

TSIO provides, for each new data set, an entry in the system catalog, a data set which lists the names and locations of data sets available in its particular installation. Locations are designated by volume serial number and unit, a reference to the type of device on which the volume containing the data set may be mounted.

Data Set Names

New data set names are composed of one or more simple names joined by periods, as in *DEPT58.SMITH.DATA3*. Each simple name must begin with an alphabetic or national character (see Figure 9), and may continue with at most seven more characters which may also include numerals. Compound names are treated by the system as a succession of qualifiers or indices in a tree-like structure leading from the major index, or root, on the left, to the name itself, or leaf, on the right. The maximum length of a (qualified) name is 44 characters, including periods.

Except for system level users, TSIO prefixes the name supplied in the *DSN* parameter with a two-level qualifier composed of the name *TSIO* and a character representation of the (positive or negative) account number *U* supplied:

```
'TSIO.','ABCDEFGHIJKLMNOP'[⎕IO+(8⍴16)⊤U],'.'
```

For example, if the name above used account number 1234, the fully qualified name transmitted to the system by TSIO would be *TSIO.AAAAAENC.DEPT58.SMITH.DATA3*, while if it used the negative account number ¯1234 the fully qualified name would become *TSIO.PPPPPLCO.DEPT58.SMITH.DATA3*.

In the case that an account number is not provided, the system supplies the (positive) account number of the user. For an alien data set or for a reserved data set the qualifier must be supplied explicitly. Thus, for user 5678 to access the data set in the example, he would have to use the form, *DSN=1234 DEPT58.SMITH.DATA3*. (User 1234 could use exactly the same form, but need not because his account number is supplied by default, much as for APL system commands. The first qualifier, *TSIO.*, is never supplied explicitly.)

A data set name may be further qualified in a different sense by appending to it another name enclosed in parentheses, as in *DSN=ABC(PART1)*. A set of such names (such as *ABC(PART1)* and *ABC(PART2)*) constitute a partitioned data set. Casual use of such data sets is not recommended; the interested user should consult the OS/VS references or a knowledgeable programmer before planning their use.

Data Set Locations

Unless otherwise specified, TSIO will allocate space for new data sets on a volume of a direct access device, selecting the particular volume from a more general allocation made previously by the system managers. Creation of a data set on other volumes, or access to non-cataloged data sets, requires explicit use of the *UNIT* and *VOLUME* parameters in the command.

Since specific device allocation (for old as well as new data sets) implies the arbitrary disposition of system resources in terms of both storage space and active use of devices, use of the *UNIT* and *VOLUME* parameters is restricted to users at the device level. Moreover, for all tape volumes, and certain direct access volumes, active cooperation of the system operator is required, and the operator should be consulted before these parameters are used.

If the *UNIT* parameter must be used - for example, to input data from a card reader - its value is chosen, according to the circumstances, to be either a specific device unit address, the designation of a device type (such as 3330), or an installation-dependent class name that denotes a collection of devices of possibly divergent types. The *VOLUME* parameter is always given as a serial number that identifies a disk pack or tape reel.

DISPOSITION (*DISP*)

The disposition parameter, *DISP*, is used primarily to position the device with respect to the data set. Secondarily, it determines whether or not a data set can be shared during an operation, and also, through a second field in its value, allows the data set to remain allocated with the device in position after the data set is closed. The latter facility is of most interest in tape operations.

The values for the first field of *DISP* are *NEW*, *OLD*, *MOD*, and *SHR*. The first three are explicitly positional, calling respectively for the device to be positioned, when opened, at the beginning, beginning, or end of the data set for a sequential operation, but imply exclusive control by one user. The fourth value explicitly allows more than one user concomitant access, but implies that the operation is to start at the beginning of the data set.

The second field may have the value *LEAVE* or *REREAD*, which positions the device at the end or beginning of the data set when it is closed. In either case the data set remains allocated to the user who opened it, until that user either reopens it for further activity with a different disposition or retracts the control variable. When used with a tape, this parameter permits the user to read the same data set several times (*REREAD*), or to read successive data sets on the same tape volume (*LEAVE*), without intervention by the system operator.

A complete specification of the *DISP* parameter has the form *DISP=(MOD,LEAVE)*. If the second field is not used, the simple form *DISP=MOD* suffices.

DATA REPRESENTATION (*CODE*)

In transferring data between an APL workspace and the operating system, TSIO transforms the internal representation of APL values according to the value of the *CODE* parameter. In the default case, *CODE=A*, the transformation is minor: TSIO merely replaces the first eight bytes of the header (which are used for internal addressing) by four bytes giving the length of the representation. For the other code values the transformation involves the total removal or replacement of the header. The values of the *CODE* parameter are:

A	APL representation	*I*	integer
B	boolean	*F*	floating-point
C	character	*E*	character with APL-EBCDIC translation

When the same code value is used for both writing and reading a data set the transformations are not perceptible to the APL user, except that code *A* requires slightly larger records. If different codes are used for reading and writing, a variety of effects can be achieved.

A variable *X* written with *CODE=A* requires four bytes less space than the amount it takes up within the workspace (see formula under the heading Arrays), rounded to the nearest word (four bytes). A variable stored with *CODE=A* and retrieved with some other code will start with an approximate representation of its original header. It may also end with meaningless fill, especially when *RECFM=F* is used with code *A*, since TSIO then provides fill to complete the specified record size.

When written with any code but *A*, the array to be passed from APL to TSIO must meet shape and size requirements imposed by the *RECFM*, *BLKSIZE*, and *LRECL* parameters. These are detailed in the section DATA SET FORMATS below.

When reading with any code but *A*, TSIO provides the indicated personality by appending an appropriate header during the transfer. If the number of bytes in the record is not an integral multiple of the unit of internal representation, the record is truncated. When reading with code *A*, the first part of the record is interpreted as a header and, if invalid, may prevent reading of the record, so that the data transfer variable has no value.

Writing a data set with *CODE = E* causes a translation from APL characters to EBCDIC characters, and reading causes a translation in the reverse direction. These translations are specified in Figure 9.

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*
EBCDIC	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
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*
EBCDIC	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
APL	0 1 2 3 4 5 6 7 8 9 space
EBCDIC	0 1 2 3 4 5 6 7 8 9 space
APL	- ~ * ¨ ' , . ; : ( ) / ÷ < > ≠ ⍺ Δ = ∧ ? + _ | ⌽ ⊖
EBCDIC	- ¬ * " ' , . ; : ( ) / % < > $ @ # = & ? + _ | +0 -0

Notes:

1. All other APL characters map into EBCDIC space, except APL ← and ¯ which map, respectively, into EBCDIC = and -.
2. All other EBCDIC characters map into APL ∘.
3. The characters $ @ # are referred to as national characters and may be represented by different graphics in different countries.
4. The +0 and -0 are obtained by overpunching. These values occur in the zoned decimal numeric representation used in keypunch data.

Figure 9. APL-EBCDIC and EBCDIC-APL Translation

DATA SET FORMATS (*RECFM*, *BLKSIZE*, *LRECL*)

The OS/VS record formats supported by TSIO are given in Figure 10, which describes their structure and the requirements imposed on the block size and record length parameters in the use of each.

RECFM	Description, transmission form when *CODE* is not *A*
F	Fixed length blocks, containing only one record, transmitted as a vector. Only format available for creating datasets for use with *IR* and *IRW*.
FB	Variable length blocks, of length up to *BLKSIZE*, each containing some number of fixed length records; each record transmitted as a vector.
FBS	Fixed length blocks, each containing a fixed number of records; each block transmitted as a matrix of size *M N* where *M* is *BLKSIZE* ÷ *LRECL* and *N* is given below. May not be used with code *A*. *N* is *LRECL* ÷ .125 1 1 4 8 when *CODE* is *B* *C* *E* *I* *F* Final record may be of size *L N* where *L* < *M*; next return code is from dataset close.
VB	Variable length self describing blocks, containing variable length self describing records, which are transmitted as a sequence of vectors whose length, when writing, must be less than *LRECL*-4, to accomodate the descriptors. *BLKSIZE* must be at least *LRECL* + 4.
V	Exactly the same as *VB* except that, when writing, a block is sent to the storage medium each time a record is transmitted, rather than only when the next record to be transmitted will not fit in the current block.
U	Variable length blocks each consisting of one record, transmitted as a vector. Can be used to read data in any other format, so long as *BLKSIZE* is large enough.

Figure 10: Parameters Related to Format

Format

The value of the parameter *RECFM* is recorded as part of the data set label when a data set is created. Each time the data set is subsequently opened, this value is used as the default. If an explicit value is given in the command it overrides the stored value, at least for the duration of the access, and replaces it if the access is for sequential writing.

When reading blocked records with *RECFM=U*, each block is returned as a single record, regardless of the structure originally imposed by blocking.

Any *RECFM* value except *U* can be extended by the letters *A* or *M* (for example, *VBA* or *VBM*), which means that the first character in the record is to be treated by the system output writers as a device control character according to the ANSI or IBM code convention given in Figure 11.

Record Length

The parameter *LRECL* sets the record size, expressed in bytes, in case of fixed format records, and indicates a maximum size for variable format records.

The four-byte length indicator automatically prefixed to the record in writing with *RECFM=V* or *VB* includes itself in its value, and the record length is therefore four more than the length of the data to be stored. The latter is often referred to as the logical record length.

Block Size

The general considerations that go into a choice of block size involve a balance between the need to limit buffer areas in the computer memory and the need to conserve external storage space by the use of larger blocks or the need to reduce retrieval time by having more records per block.

The block size parameter always has the form *BLKSIZE*=n, where n is expressed in bytes. Except for fixed formats it indicates an upper limit. When reading with *RECFM=U*, oversize records are truncated to the indicated block size.

ACTION (1)	CONTROL CHARACTERS (2)		
	A-ANSI		M--MACHINE
	For *CODE=C* use ⎕*AV* indexed by	For *CODE=E* use APL character	For *CODE=C* use ⎕*AV* indexed by
Suppress line spacing	78 (5)	+	1 (5)
Space one line	64	space	9
Space two lines	240	0	17
Space three lines	96	-	25
Skip to channel 1	241	1	137
Skip to channel 2	242	2	145
Skip to channel 3	243	3	153
Skip to channel 4	244	4	161
Skip to channel 5	245	5	169
Skip to channel 6	246	6	177
Skip to channel 7	247	7	185
Skip to channel 8	248	8	193
Skip to channel 9	249	9	201
Skip to channel 10	193	*A*	209
Skip to channel 11	194	*B*	217
Skip to channel 12	195	*C*	225
Select punch pocket 1 (3)	229	*V*	1
Select punch pocket 1, CB (4)			33
Select punch pocket 2	230	*W*	65
Select punch pocket 2, CB			97

Notes:
1. Action takes place before line is printed for ANSI, and after line is printed for machine encoding.
2. Characters other than these default to space one line or select punch pocket 1, as appropriate.
3. Characters shown for 2520 Punch. See OS/VS references for others.
4. CB: column binary.
5. Zero-origin indices.

Figure 11: Unit Record Output Control Characters

Figure 12 shows the relationship of block size to track length for three direct access devices. Formulas for these values under different conditions can be found in the OS/VS reference publications.

Device	2314	3330	3340
Bytes/Track (Unit Size)	7294	13,030	8368
Tracks/Cylinder	20	19	12
Bytes/Cylinder	145,880	247,570	100,416
Cylinders/Volume	200	404	348/696
Bytes/Volume (millions)	29	100	35/70

Figure 12. Disk Storage Device Capacities

SPACE ALLOCATION (*SPACE*)

Direct access devices are comprised of storage units called tracks; the storage capacity of a track varies with the type of device, for example, the maximum on a 2314 disk is 7294 bytes per track, but on a 3330 is 13,030. Space is always allocated in increments of tracks - that is, one or more whole tracks are allocated even though a portion of the last track may not be needed. Each connected piece of allocated storage is called an extent.

The storage space on each track may be subdivided into one or more blocks (or physical records), separated by inter-record gaps. The size of the gaps varies with the type of device and in some cases with the size and number of blocks stored on the track.

The *SPACE* parameter has a compound value which in its complete form has the appearance *SPACE*=(1000,(100,20,13)). The fields give, in order, a nominal block size, the number of blocks to be allocated to the data set initially, an incremental number of blocks for secondary allocation if required (up to a maximum of 15 times), and the number of records that are to be contained in the directory of a partitioned data set. The parameter can be limited to two or three fields with the following alternatives:

SPACE=(1000,(100,20)) - normal form for non-partitioned data set

SPACE=(1000,(100)) - simplest form of parameter specification, disallows secondary allocations

SPACE=(1000,(100,,13)) - as above, but with a directory space requirement specified

The number of units (tracks) of space allocated is approximated from the block-size field B and the number of blocks N according to the expression $\lceil N \div \lfloor TL \div B$, where TL is track length for the device. A more exact computation makes use of a table such as Figure 13.

Figure 13 shows the possible division of a single track into blocks of storage space for the 2314, 3330, and 3340 disk devices. For each device, one column shows the maximum number of bytes that each block could contain if the track contained from one to 30 blocks; a second column shows the percentage of the track that would actually be used for storage of data, the rest being lost to inter-record gaps.

Blocks per Track	2314 Bytes per Block	2314 % Utili-zation	3330 Bytes per Block	3330 % Utili-zation	3340 Bytes per Block	3340 % Utili-zation
1	7294	100	13030	100	8368	100
2	3520	97	6447	99	4100	98
3	2298	95	4253	98	2678	96
4	1693	93	3156	97	1966	94
5	1332	91	2498	96	1540	92
6	1092	90	2059	95	1255	90
7	921	88	1745	94	1052	88
8	793	87	1510	93	899	86
9	694	86	1327	92	781	84
10	615	84	1181	91	686	82
11	550	83	1061	90	608	80
12	496	82	962	89	544	78
13	450	80	877	87	489	76
14	411	79	805	86	442	74
15	377	78	742	85	402	72
16	347	76	687	84	366	70
17	321	75	639	83	335	68
18	298	74	596	82	307	66
19	276	72	557	81	282	64
20	258	71	523	80	259	62
21	241	69	491	79	239	60
22	226	68	463	78	220	58
23	211	67	437	77	204	56
24	199	65	413	76	188	54
25	187	64	391	75	174	52
26	176	63	371	74	161	50
27	166	61	352	73	149	48
28	157	60	335	72	137	46
29	148	59	318	71	127	44
30	139	57	303	70	117	42

Figure 13. Track Length and Block Size

TAPE DATA SETS (*DEN*, *LABEL*, *TRTCH*)

It is often necessary to introduce into APLSV data from other sources, such as a physical-data collection device or an existing time-series data base. TSIO permits the direct use of magnetic tape devices for this purpose. The prospective tape user must consult the system operator to schedule use of a magnetic tape drive. When submitting a magnetic tape reel, the user should ensure that the tape is clearly identified, by its volume serial and APL user's account number, since these will be displayed to the operator from TSIO. A tape having standard labels can be read by a command such as:

```
CTL←'SR DSN=NNN,CODE=C,UNIT=2400,VOLUME=ABC02P'
```

where the volume serial number, used in the *VOLUME* parameter, and the data set name must match those recorded on the tape.

The standard data set label carries information on format, block size, and record length, so that the corresponding parameters need not be given explicitly. (The manual "OS/VS Tape Labels" (GC26-3795) gives detailed information about the content and use of standard magnetic tape labels.) The use of code *C* assures that a complete image of the record will be transmitted to the workspace without loss of information, but if a code other than *C* is known to be appropriate for the data, its use may reduce the requirement for subsequent processing.

The *UNIT* parameter must be given explicitly, since the default value is a subset of the direct access devices at the installation. The value for this parameter will vary with the installation, but 2400 is often used as a generic designation for tape drives.

Tapes without labels, or with non-standard labels, can also be read, using variations in the reading command with parameters such as *LABEL* and *TRTCH* (tape recording technique). The parameters *RECFM*, *LRECL*, and *BLKSIZE* must be given explicitly for tapes without standard labels. Some experimentation will usually be required, but this can be conducted at the APL terminal.

The reading and writing of data sets on tapes involves the use of density, label, and tape recording technique parameters. The density determines the recording density (bytes per inch) and applies to seven-track and nine-track tapes as follows:

DEN	Track	Density
0	7	200
1	7	556
2	7	800
2	9	800
3	9	1600
4	9	6250

The default value is the largest applicable to the particular tape unit allocated.

The *LABEL* values *SL*, *NL*, and *BLP* denote standard labels, no labels, and bypass label processing, respectively. The values may be prefaced by data set sequence numbers, as in *LABEL*=(12,*NL*), or may appear with the number omitted, as in *LABEL*=(,*NL*). An omitted number implies 1, that is, the first data set on the tape. See OS/VS Tape Labels (GC26-3795) for further information.

The *TRTCH* parameter, which applies to 7-track tapes, specifies the type of parity check (odd or even) applied in reading or recording each byte, and the application of the data conversion feature, or EBCDIC-BCD translation, as follows:

TRTCH	Parity	Translation	Conversion
C	Odd	None	Yes
E	Even	None	No
T(write)	Odd	EBCDIC→BCD	No
T(read)	Odd	BCD→EBCDIC	No
ET(write)	Even	EBCDIC→BCD	No
ET(read)	Even	BCD→EBCDIC	No

The data conversion feature provides the encoding of 8-bit characters in the 6-bit configuration of a 7-track tape, using four frames for three characters. The default case (when TRTCH is not specified) provides odd parity, no translation, and no conversion.

Detailed information about 7-track tape drives may be found in the System/370 Component Description publication for the tape unit to be used.

SYSTEM LEVEL OPERATIONS (*EXPDT*, *KEYLEN*, *SYSOUT*)

System level users may employ four additional data management commands (*CATALOG*, *UNCATALOG*, *READVTOC*, and *MSG*), and two further parameters (*EXPDT* and *KEYLEN*). They must, however, provide a fully qualified OS/VS data set name for all operations, and explicitly catalog

and uncatalog their data sets whose names do not begin with *TSIO*, unless the name is given with an associated account number, as in *DSN=45273 DATA*.

A catalog operation command relates a data set name to a volume and has the form:

```
CTL←'CATALOG DSN=ABC.DE,VOLUME=QRS012,UNIT=3330'
```

The user should be careful to use only device class designations in the unit field (3330, 2314, and so forth).

The uncatalog operation requires only a data set name:

```
CTL←'UNCATALOG DSN=OBSO.LETE'
```

A direct access volume carries a table of contents which comprises the labels for the data sets it contains. This can be read by the following command form:

```
CTL←'READVTOC VOLUME=QRS012,CODE=C'
```

after which the VTOC is treated as a sequential file. The publication OS/VS System Data Areas (VS1: SY28-0605; VS2: SYB8-0606) gives a complete description of the VTOC.

A system level user can send a message to the OS/VS console by using the form

```
CTL←'MSG PLEASE DO SOMETHING FOR ME'
```

The next reference to *CTL* reads the operator's reply, and *CTL* is returned to command state. If the operator does not reply the variable *CTL* remains unusable unless retracted and reshared.

Date Protection

An important data set can be protected against deletion or over-writing by associating an expiration date with it. The *EXPDT* parameter specifies the date before which the data set can not be deleted or overwritten without operator intervention. The form is *EXPDT=75001*, which has a two-digit year number and a three-digit day number. It can be used in a sequential write command with *DISP=NEW*.

Key Length

The data sets created by TSIO users are recorded without keys, although the use of keyed records is an option in OS/VS. To provide access to existing direct access data sets that may be keyed, system level users can use the *KEYLEN* parameter to read such data sets. The key will appear as the first *KEYLEN* bytes of the record.

The parameter specifies the length n, in bytes, of the keys used in the data set, and has the form *KEYLEN=n*. The largest value allowed for this parameter is 255.

Sysout Data Sets

The *SYSOUT* parameter identifies a temporary data set and enqueues it for recording on a unit of the class specified by the single character associated with it. The associated character varies from installation to installation, but the use of *SYSOUT=A* for the high-speed printer and of *SYSOUT=B* for the card punch is common practice.

In addition to the class character, the *SYSOUT* parameter may give a program name and form number, as follows:

```
SYSOUT=(A,PRINT,AR2)
SYSOUT=(A,PRINT,)
SYSOUT=(A,,AR2)
```

The program name identifies the program from the system library to be used instead of the standard system output writer. The form number (limited to at most four alphanumeric characters) specifies the paper or card form to be used in printing or punching respectively.

Following the opening of the data set by, say:

```
CTL←'SW SYSOUT=A'
CTL
0
```

CTL is specified successively by each line of printed output, until the entire text has been transmitted. Since an empty vector would signal the close of the data set, blank lines must be transmitted as consisting of at least one space. After the transmission, the data set must be closed before any actual output can take place:

```
CTL←''
CTL
0
```

Jobs may be submitted for execution to the operating system by a system level user, or via an *IC* command that accesses a command data set of a system level user. A job is given by a data set with parameters *SYSOUT=(A,INTRDR)*, where *A* stands for a job class and *INTRDR* stands for an installation- or system-defined reader procedure. The records of the data set are card images. The job is submitted for execution when the data set is closed.

SECTION 4: USE OF DISTRIBUTED WORKSPACES

Groups of related workspaces are described under each heading below. In general, if there is no workspace of instructions associated with the application workspace, the instructions will be contained in either a function or a variable *DESCRIBE* within the workspace. As distributed, these workspaces are in library 1.

ARRAY FILE

The workspace *APLFILE*, which makes use of TSIO, provides functions for storing and retrieving APL arrays in indexed files outside the workspace. The arrays may differ in size, and may be of any length that will fit within the total space allocated, including lengths that span several tracks of the storage device. The user stores or retrieves an array by means of the functions *SET* and *GET*, specifying in their arguments the name of the file and an index number. For example, the instruction:

```
('FILE' AT I) SET X
```

stores the array *X* as the *I*th array in a file called *FILE*. Similarly, the *I*th array of the file called *FILE* is retrieved by the expression:

```
GET 'FILE' AT I
```

When repeated references are made to the same file, instructions after the first need refer only to the index number; the file name is presumed to be the one last named. The foregoing instructions become:

```
I SET X      and      GET I      respectively.
```

Other functions supplied in the workspace may be used to create a new file, to initiate or terminate use of a file, to delete an entire file or to erase indicated members within a file. To accomodate files shared between several users, the file includes a record of the identity of the last user to update it and the time at which updating occurred.

NUMERIC DATA CONVERSION BETWEEN APLSV AND S/370

The workspace *CONVERSION* provides functions to convert to and from the number representations used by APLSV and alternative schemes used by other programs. Each of the conversion functions has a two-letter name of which the last letter is either *I* or *O* (indicating whether data is to be transferred into or out of an APLSV environment) and starting with one of the letters *C*, *F*, *I*, or *B* (indicating whether the data format outside APLSV is character, floating point, integer, or Boolean).

In addition, the workspace contains the variables *CD*, *FD*, *ID*, and *BD*, each of which provides a description of the pair of conversion functions for characters, floating point, integers, or Boolean, respectively.

LINE EDITING

MEDIT
SEDIT
FEDIT
HOWEDITS

Three related workspaces are provided to facilitate editing text material that is organized into lines; these are useful in preparing

text that will subsequently be treated as a program to be submitted to a language processor outside APLSV (such as an assembler or compiler) or as an APL function definition. Functions are provided to display, delete, or edit the existing lines of a program, and to append or insert new lines.

The workspaces *MEDIT* and *SEDIT* (for "matrix edit" and "string edit") are functionally identical, but differ in the way in which the stored program is represented: as a single string containing no blanks, or as a matrix. *MEDIT*, using matrix representation, is faster but requires more space than *SEDIT*.

The workspace *FEDIT* (for "file edit"), which makes use of TSIO, contains additional functions intended to read a text (that is, a program listing) from a data-set outside APLSV, or return the resulting program from the APL workspace to an external data-set, for storage or for subsequent submission to an external language processor.

Instructions for use of the three line-editing workspaces are contained in the workspace *HOWEDITS*.

TSIO UTILITY

The workspace *TSIO* contains functions intended to facilitate the use of TSIO to manage files and to transmit data between files and the APL workspace. It also contains functions for creating, manipulating, and displaying the contents of command data sets, that is, of secured data sets intended for control of indirect command execution.

The function *TRY* takes as its left argument the names of the variables shared with TSIO, and as its right argument the TSIO command to be executed. The function reports error conditions as appropriate, or may be embedded in a user function containing provision to branch, exit, retry, etc. as appropriate.

The function *CHK* is used to interpret the return values of TSIO control variables. It is called by *TRY* to interpret the return from TSIO commands, and may also be embedded in user-written functions for data transfer. It permits the user (at his option) to have an error message printed, to branch to another statement in the user's function, or to suspend execution for debugging. The workspace contains tables for interpreting TSIO return codes, and provision to point to the defective portion of a TSIO command.

FORMATTING NUMERIC DATA

The workspace *FORMAT* contains the definition of the function *FMT*, used to format numeric data. The left argument is a character string depicting a sample output, including non-numeric text or decorations both between and within the numeric fields, and indicating characters that will mark negative values. These permit the user to insert commas, to manage leading or trailing zeros, or to select his own symbol or notation for negative values.

The result of *FMT* is an array in which each line reproduces the picture provided in the left argument, substituting appropriate numeric characters for those in the sample, and including embedded decorations and the negative indicators where needed.

This function may be used as a convenience in generating specialized formats beyond those available from the primitive format function ⍕. Several illustrative examples are provided in the workspace.

SUBSTITUTE DEFINITIONS FOR WORKSPACE FUNCTIONS

Earlier APL systems included a workspace *WSFNS* containing the locked definitions of several functions useful in controlling certain workspace parameters such as the printing precision, printing width, or index origin. These were implemented using ⌶ (I-beam) functions, which in APLSV are replaced by system variables with names such as ⎕*PP* for printing precision, ⎕*IO* for index origin, and so on. All new user functions should make use of the system variables for this purpose.

However, programs brought into APLSV but originating in older systems may include functions from the workspace functions formerly distributed. These will no longer work. To facilitate conversion of those old workspaces, the new workspace *WSFNS* contains definitions having the same name and effect as the former set, but now defined in terms of system variables. Copying the workspace *WSFNS* into the old workspace will replace the former workspace functions with the new ones. The functions affected are: *DELAY*, *DIGITS*, *ORIGIN*, *SETFUZZ*, *SETLINK*, and *WIDTH*.

NOTICE OF SYSTEM CHANGES AND SCHEDULES

The workspace *NEWS* permits users to display notices regarding the system operation posted after the date they indicate. Users may also obtain a brief summary of the notices and then indicate which they wish to see in full.

The workspace contains functions that assist the system managers to insert or revise the notices displayed.

The *NEWS* workspace distributed with APLSV is initialized to contain a description of the new features of APLSV.

PLOTTING AND FORMATTING FUNCTIONS

The workspace *PLOTFORMAT* contains defined functions useful in formatting numeric data and in constructing graphs or histograms at the terminal. Functions have been redesigned to use new features of APLSV, and will generally be more efficient than APL/360 counterparts but less efficient than the direct use of new primitive functions. More comprehensive functions for output display are provided by the formatting workspace described above, and by the plotting functions contained in FDP 5798-AGL, "Graphs and Histograms in APL."

INSTRUCTIONAL WORKSPACES

APLCOURSE
ADVANCEDEX
TYPEDRILL
SBIC

The workspace *APLCOURSE* contains two functions which generate expressions which the user is asked to evaluate. They report his success, supply correct answers on request, and keep a record of his score. The function *EASYDRILL* limits the problems posed, but is otherwise the same as the function *TEACH*.

The workspace *ADVANCEDEX* contains the definitions of illustrative functions for a number of applications, including conversion of number representations, data entry, string matching, the evaluations of

polynomials, finding roots of polynomials, permutations, combinations, matrix inversion, and so on. Each function is accompanied by a character vector whose name consists of the letter *D* followed by the name of the function. When displayed, these provide a description and discussion of the programming examples.

It should be noted that some of the examples have been made obsolete by extensions to APL. For instance the discussion of matrix inversion, while valid, duplicates the effect of the primitive function ⌹.

The workspace *TYPEDRILL* permits a student of typing to enter a typewriter keyboard exercise, and records the speed and accuracy with which he is able to repeat it.

The workspace *SBIC* contains a model set of functions to perform sales, billing and inventory control. Their use and limitations are described in the APL Language Manual (GC26-3847).

A

B

C

D

E

F

G

H

I

J

K

Z

APL Shared Variables (APLSV) User's Guide
(Programming RPQ WE1191)
SH20-1460-1

Reader's Comment Form

Your comments about this publication will help us to improve it for you. Comment in the space below, giving specific page and paragraph references whenever possible. All comments become the property of IBM.

Please do not use this form to ask technical questions about IBM systems and programs or to request copies of publications. Rather, direct such questions or requests to your local IBM representative.

If you would like a reply, please provide your name, job title, and business address (including ZIP code).

Fold on two lines, staple, 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.

Fold and Staple

[redacted]

First Class Permit
Number 439
Palo Alto, California

Business Reply Mail

No postage necessary if mailed in the U.S.A.

[redacted]

Postage will be paid by:

IBM Corporation
System Development Division
LDF Publishing—Department J04
1501 California Avenue
Palo Alto, California 94304

Fold and Staple

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

APL Shared Variables (APLSV) User's Guide (Programming RPQ WE1191) Printed in U.S.A. SH20-1460-1

IBM **Technical Newsletter**

This Newsletter No.	SN20-9083
Date	March 31, 1975
Base Publication No.	SH20-1460-1
Previous Newsletters	None

APL Shared Variables (APLSV) User's Guide
(Programming RPQ WE1191)

The information in this manual applies to Version 2.0 of the APLSV subsystem, which operates under OS/VS2, Release 2.

The references to OS/VS1 in this manual are provided for planning purposes only until the release of APLSV Version 2.0 under that operating system. In particular, the sections titled "Data Set Privacy," "Shared Data Sets," and "SYSOUT Data Sets," all in "Section 3: Use of TSIO," apply only to Version 2.0 of APLSV.

APLSV Version 2.0 runs under only OS/VS2, Release 2. At some future date, these releases of OS/VS1 and OS/VS2 will support this version of APLSV:

OS/VS1, Release 3.1 and Release 4.0

OS/VS2, Release 1.7 and Release 3.0.

Note: Please file this letter at the back of the manual to provide a record of changes.

IBM Corporation, LDF Publishing, 1501 California Avenue, Palo Alto, California 94304

Printed in U.S.A.

IBM **Technical Newsletter**

This Newsletter No. SN26-0800
Date June 30, 1975

Base Publication No. SH20-1460-1

Previous Newsletters SN20-9083
(see below)

APL Shared Variables (APLSV)
User's Guide
(Programming RPQ WE1191)

Program Number 5799-AJF

This technical newsletter, a part of Version 2.1 of APLSV, provides replacement pages for the subject publication. These replacement pages remain in effect for subsequent versions of APLSV unless specifically altered. With this version, APLSV becomes available under OS/VS1, Release 3.1 and Release 4.0, and under OS/VS2, Release 1.7, Release 2.0, and Release 3.0. Pages to be replaced or inserted are:

Title page/edition
1-6
6.1 (added)
9, 10
10.1 (added)
17-20
20.1 (added)
31, 32
41, 42
43 (added) .

A change to the text or to an illustration is indicated by a vertical line to the left of the change.

This technical newsletter obsoletes previous technical newsletter SN20-9083.

Note: Please file this cover letter at the back of the manual to provide a record of changes.

IBM Corporation, LDF Publishing, 1501 California Avenue, Palo Alto, California 94304

Printed in U.S.A.

APL Shared Variables (APLSV) User's Guide (Programming RPQ WE1191) Printed in U.S.A. SH20-1460-1

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)