SH20-1461-0

APL SHARED VARIABLES (APLSV)

OPERATIONS GUIDE

SYSTEMS

Programming RPQ WE1191
Program Number 5799-AJF

This manual contains the information necessary to install, operate and maintain APLSV and TSIO. TSIO is an IBM-supplied auxiliary processor which provides access to Operating System data sets from APLSV terminals. The use of TSIO is optional.

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

IBM

CONTENTS

FIGURES

)UCTION

APLSV is a successor to APL\360-OS (Program Number 5734-XM6). ovides an enhanced language, access to Operating System files data through the auxiliary processor TSIO, and support of VS1 and OS/VS2 rather than MFT and MVT. APLSV, like APL\360, is interactive system designed to give rapid response to a large umber of terminal users.

System Description

APLSV consists of three main programs:

- The APLSV interpreter, which executes APL functions.
- The Shared-Variable Processor, which manages shared variables.
- The APLSV supervisor, which manages APLSV time sharing.

Certain service routines are also provided:

- An auxiliary processor, TSIO, which provides a shared-variable interface to the Operating System.
- A startup and initialization routine.
- The APLSV Utility, for library maintenance.

In addition, APLSV workspaces are provided in the APLSV distribution library to illustrate the use of new primitives, system variables and functions, and shared variables.

APLSV Operating Characteristics APLSV operation is controlled by the APLSV supervisor program which uses standard Operating System facilities for all services from the Operating System. A single type-1 SVC is used for communication between the APL interpreter and supervisor, and between all programs using shared variables and the shared variable processor.

When APLSV is started, a monitor program is loaded which uses either prespecified parameters or console input from the operator to determine the programs to be loaded and the allocation of storage to the various routines.

The APLSV interpreter, supervisor, shared-variable processor, and the initialization routine are then loaded, and control is transferred to the initialization routine which opens the swap and library files, formats the swap file, and initializes various tables associated with library control and terminal support. When initialization is complete, a timer event is enqueued and control is returned to the monitor, which remains inactive unless some event occurs which requires its action.

The expiration of the initial timer event causes the APLSV supervisor to be activated, and it proceeds to initialize typewriter buffers, overlay the initialization program, and enable the terminal ports. For the duration of the operating period, the system is driven entirely by interrupts passed to it by the Operating System supervisor, from the following sources:

Multiplexor Channel - appropriate action is taken depending on the condition. When an incoming call is answered and input has been received, the sign-on process is initiated by reading a user directory and passing control to the APL interpreter. The interpreter determines subsequent action, and issues requests to the supervisor for output or additional input as necessary.

Selector Channel - if an error has occurred, Operating System error recovery is invoked. If an I/O transfer has completed, the supervisor scheduling routine is notified.

Supervisor call - signals a request from the APLSV interpreter for library access, terminal I/O, or miscellaneous service.

Interval timer - signals that some event is due. Interval events include quantum end for time sharing, timed multiplexor retries, priority switching, and user-requested delays.

Program checks - in supervisor state, result in APLSV termination; in problem state, are passed to the APLSV interpreter for analysis.

Virtual memory support is enabled when APLSV has not been initiated with ADDRSPC=REAL. During initialization, the storage allocated to the APLSV supervisor and to terminal buffers is locked in real core. All code operates in relocation mode, and two additional supervisory routines are enabled:

A selector channel page-fix appendage which passes a list of addresses to be temporarily fixed for an I/O transfer.

CCW relocation routines, which translate multiplexor channel CCWs so that they contain real storage addresses during I/O transfer, and contain virtual addresses when the CCW chain has completed and is to be analysed.

The shared-variable processor operates as an independent subsystem. It consists of approximately 4,000 bytes of code and an arbitrary amount of storage (specified when APLSV is started) used for tables and for storing shared variables. The processor is designed and implemented as though it were an independent device analogous to a channel or a storage facility. An interface and associated protocol for submitting requests is defined, and can be accessed through the APLSV type-1 SVC using assembly-language macros provided with the distributed system.

The APLSV utility program provides a variety of accounting and library maintenance functions, and in addition, provides conversion between APL/360 workspaces and APLSV workspaces.

TSIO is an auxiliary processor which uses standard operating system routines for device and data-set allocation, and BSAM and BDAM for file access. A simple security scheme is provided.

Programming Systems

- The supported host operating systems are OS/VS1 and OS/VS2.
- The following programs are required for installation: Assembler, Linkage Editor, Utilities.
- APLSV is written entirely in Assembler Language.
- APLSV manages data as follows:
  Terminals: APLSV provides the same terminal support as APL\360, using its own access method.
  Swapping and Library Access: EXCP
- TSIO manages data as follows:
  APLSV user I/O: BSAM, BDAM. (Blocked sequential data sets are supported.)
  Logging data set: QSAM

System Requirements

The System/370 configuration chosen must include the considerations below:

Selector channel devices supported for swapping and library storage: 2314, 2319, 3330. An arbitrary number of devices may be used for swapping; these need not be of the same type. A maximum of 32 library extents; each of which may be an entire module, is supported; library devices must be of the same type. A minimum of one swap data set and one library data set is required.

Storage required: The recommended minimum OS/VS1 partition or OS/VS2 region for a ten-port APLSV system is 212k bytes. This includes space for ten terminal control and buffer areas of approximately 400 bytes each, the minimum of two workspace slots at the APL\360 standard size of 36000 bytes, and a 10000-byte shared variable storage. A separate partition or region of 80k bytes is required for TSIO. Since partitions and regions are actually allocated in multiples of 64k bytes, the minimum virtual storage requirements are 256k bytes and 128k bytes for APLSV and TSIO respectively. The recommended minimum amount of real storage which should be available when APLSV is operating is 100,000 bytes, assuming ten or fewer terminals are attached.

Transmission Control Units supported: 2702, 2703 or equivalent, with IBM line control. A minimum of one line position is required.

Terminal devices supported: IBM 1050, 2740 model 1, 2741, or equivalent. A minimum of one such terminal (which must have an interrupt feature) is required.

A minimum of one tape drive is required for installation and library maintenance.

Compatibility with APL\360

APL\360 programs will operate correctly in APLSV except for minor changes in residue, decode, and changes in literal input and output. Consult the APLSV User's Manual, SH20-1460, for details.

When APLSV is installed, APL\360 libraries must be converted by dumping them to tape with the APL\360 Utility, reallocating the library data sets, and then using the APLSV Utility to create APLSV libraries from the APL\360 dump tape.

The APLSV Utility program provides conversion between APL\360 workspaces and the APLSV workspace format. Although APLSV workspaces may be converted to the APL\360 format, functions which contain advanced language features such as execute, format, and system functions and variables, will not operate in APL\360.

PART I

APLSV AND TSIO INSTALLATION

The basic machine-readable material for APLSV consists of three tape files:

- The APLSV distribution library which contains a model APLSV user directory and a variety of APLSV workspaces. A listing of the workspaces supplied is given in Appendix B of this manual.
- APLSVS.GENLIB, a partitioned data set containing the installation macro library.
- APLSVS.PUNCH, a partitioned data set containing the object-module library. The modules in this library are linkedited with a locally assembled punch module to produce the various APLSV load modules.

The installer must determine the space necessary for and allocate the APLSV library and swap data sets, as described below. APLSV and TSIO can then be installed by copying the installation macro library to disk, then using it to assemble a set of configuration macros. This assembly punches four jobs:

- An installation job
- JCL to execute the APLSV Utility
- JCL to initiate APLSV
- JCL to initiate TSIO

The installation job constructs an APLSV system tailored to a particular installation and modifies the OS/VS1 or VS2 system to accept it by adding one type-1 user SVC. The remaining three jobs will only run with the modified OS/VS1 or VS2 nucleus, which will be a secondary nucleus until renamed by the installer.

The APLSV Utility is used during the installation process to initialize the APLSV user library data sets, and either to restore the APLSV distribution library to them or to convert and restore an existing APL\360 utility dump tape to them.

Figure 1, which immediately follows, represents a sample APLSV installation job. The description of APLSV installation which follows refers to this example.

```
//APLALLOC JOB (ACCOUNT),NAME
//* ALLOCATE APLSV SWAP AND LIBRARY DATA SETS
// EXEC PGM=IEFBR14
//APLLIB0 DD DSN=APLSVS.APLLIB0,
// SPACE=(CYL,(80),,CONTIG),DISP=(NEW,KEEP),
// UNIT=2314,VOL=SER=APLPAK
//APLSWAP0 DD DSN=APLSVS.APLSWAP0,
// SPACE=(CYL,(7),,CONTIG),DISP=(NEW,KEEP),
// UNIT=2314,VOL=SER=APLPAK

//APLSVGEN JOB (ACCOUNT),NAME
// EXEC PGM=IEHMOVE
//* RESTORE APLSV MACRO LIBRARY TO DISK
//SYSUT1 DD UNIT=2314,SPACE=(CYL,(4,4))
//SYSPRINT DD SYSOUT=A
//T DD UNIT=(2400,,DEFER),LABEL=(,NL),DISP=OLD,VOL=SER=APLSVT,
// DSN=APLSVT,DCB=(BLKSIZE=800,LRECL=80,RECFM=FB)
//D DD UNIT=2314,VOL=SER=APLPAK,DISP=OLD
//SYSIN DD *
 COPY PDS=APLSVS.GENLIB,FROM=2400=(APLSVT,4),TO=2314=APLPAK,FROMDD=T
/*

//ASMGEN EXEC ASMFC,PARM=DECK
//* GENERATE INSTALLATION AND RUN:DECKS
//SYSLIB DD DSN=APLSVS.GENLIB,VOL=SER=APLPAK,UNIT=2314
//SYSIN DD DATA
*
* SPECIFICATION MACROS
*
        APLSWAP UNIT=2314,VOL=APLPAK
        APLLIB UNIT=2314,VOL=APLPAK,DIRS=2
        APL UNIT=2314,VOL=APLPAK
        APLGEN UNIT=2314,VOL=APLPAK
        TSIO UNIT=2314,VOL=APLPAK
        APL SVC=255,JOBNAME=APL
        SYSRES UNIT=2314,VOL=SYSPAK,TYPE=VS1R2
        WORK UNIT=2314
        TAPE UNIT=2400-3
        SYSRES MACBLK=3520
        APLPORT UNIT='31',TYPE=2741
        APLPORT UNIT='33',TYPE=TS41
        APLPORT UNIT=X'30',TYPE=1050
        APLPORT UNIT=X'40',TYPE=2741
        APLPORT UNIT=(X'1E',X'23'),TYPE=AMBIG
```

```
 PRODUCTION MACROS

       REPRO
/APLINST JOB (ACCOUNT),NAME
       INSTALL
       REPRO
/APLUTIL JOB (ACCOUNT),NAME,TYPRUN=HOLD
       RUN UTILITY
       REPRO
/APLSV JOB (ACCOUNT),NAME,TYPRUN=HOLD
       RUN APL
       REPRO
/APLTSIO JOB (ACCOUNT),NAME,TYPERUN=HOLD
       RUN TSIO
       END
*
```

FIGURE 1: SAMPLE APLSV INSTALLATION

Installation Planning

One type-1 SVC number must be reserved by the SVCTABLE system-generation macro when generating the OS/VS1 or VS2 system which is to host APLSV. APLSV places no restrictions on the SVC number to be used.

The multiplexor channel addresses of the lines to be used for APLSV terminals and the kind of transmission control unit installed must be determined. If the control unit is an IBM 2702, the correct SAD command for the line adapters to be used for APLSV terminal ports must be determined. If any terminals are connected directly to the control unit (for example, by an IBM Limited Distance Four-Wire Modem) the kinds of terminals must be known.

The size of the APLSV user library and swap data sets must be determined.

Figure 2, at the end of the APLSV Installation section, illustrates the data sets required to install and run each APLSV system. The direct-access volumes which will contain these data sets must be chosen. Data sets specified in Figure 2 as allocation type 3 are allocated during the installation job. If it becomes necessary to rerun the installation job, these data sets must be deleted first.

The APLSV user library data sets must be allocated before the library maintenance utility is run. The APLSV swap data set must be allocated before the APLSV terminal system can be run.

Workspace Size The unit of storage allocated to APLSV users is a workspace which contains defined functions and variables. Each user may save workspaces in a private library, identified by the sign-on account number, and each active user has an active workspace containing functions and variables currently being manipulated. The workspace size, *WSSIZE*, is the amount of active storage that an APLSV user has available to him. A small number of areas (slots), each of *WSSIZE* bytes, is reserved in the APLSV partition or region to contain active workspaces. The number of slots is usually much smaller than the number of access ports. APLSV transfers workspaces between the slots in the region and a swapping data set as necessary. The minimum permissible number of slots is two. The number of slots is specified during installation by the SLOTS parameter of the APL macro, and can be varied during APLSV startup if necessary.

In order to aid in the transfer of workspaces between APL systems, a standard workspace size is established. All APL workspaces distributed by IBM are operable in a system which has been installed with a workspace size equal to or greater than the standard. A workspace of 36k bytes was the standard size for APL\360 systems, but the standard size of the APLSV system's workspace is 64400 bytes, representing 5 tracks of 3330 storage. In general, an increase in the workspace size results in an increase in the size of APL variables that can be manipulated, at a cost of increased storage requirements and an increase in the time required for swapping operations. The workspace size must not exceed the amount of real storage which will be available for page replacement when APLSV is running.

Swap and Library Data Sets A minimum of two data sets is required by APLSV. The first of these, a swap data set, is used to hold the active workspaces of users while other workspaces are being processed in the region or partition. The second is a library data set and is used to store user directories and workspaces saved by APLSV users in private and public libraries. The APLSV user directories contain all of the accounting information for the users of the system and the names and file locations of all saved workspaces. Two copies of each directory are maintained at the beginning of the first library data set.

APLSV can be configured to have up to 10 swap data sets and up to 32 library data sets.

Once the workspace size has been decided and the number of ports is known, the swap data set requirements can be calculated. Space is required in the swap file for three more workspaces than the number of ports configured. Workspaces are written in half-track blocks, 3520 bytes per block for the 2314 and 6440 bytes per block for the 3330. The number of blocks required per workspace is the workspace size divided by the number of bytes per block, rounded up.

For example, the 64400 byte standard workspace used with 2314 disk on a system with ten access ports requires

(10+3)×⌈(64400÷3520) or 13×19 = 247 blocks.

(⌈ is the APL symbol meaning "round up").

This is 123.5 tracks, or something less than 7 cylinders of 2314 space.

The requirements for the APLSV user libraries are more difficult to estimate, since workspaces saved by users in the libraries will occupy only as many blocks as they require, and will vary in size from a minimum of 3000 bytes to the full workspace size. In practice, library workspaces tend to be slightly more than half full, and the number of workspaces per user will generally increase to meet the limit imposed by the system manager. If a new system is being generated, three workspaces of one-half the workspace size per user is a reasonable estimate. If an existing *APL*\360 system is being replaced, the current APL library data set sizes can be used. In addition to user libraries, the first library data set contains the APLSV user directories, each one of which requires two times the next even number of blocks greater than or equal to that required for a full workspace.

Thus, if the 10 port system above had 2 directories, which is the minimum number allowed, the first library extent would require at least

2×2×20 blocks, or 40 tracks.

The number of user directories required for a new system can be estimated from the workspace size and the expected number of enrolled users, that is, the number of users who are permitted access to the system.

Multiply the number of users by 400 and divide by the workspace size minus an overhead of 10000, then round up to the next prime.

If one hundred users were allowed to access the ten port 2314 system mentioned above,

⌈(400×100)÷(64400-10000) or ⌈0.8

In other words, 1 directory is required, so the minimum directory allotment is sufficient.

If an existing *APL*\360 system is being replaced, simply copy the number of directories it has.

In either case, note that because the method used for assigning users to directories requires that the number of directories be a prime number, the next prime larger than or equal to the number calculated or copied above should be used. The primes up to 50, and the APL program which produced them, are given in appendix C.

An initial estimate of 3×100 or 300 saved workspaces of about 10 blocks each gives a total library space requirement of approximately 80 cylinders -- 40 tracks for directories, plus 1500 tracks for saved workspaces. Should it later become necessary, the APLSV Utility can be used to increase the size and number of the APLSV library data sets.

In summary, the example ten port 2314 system requires 7 cylinders for the swap data set and 80 cylinders for the APLSV user library. The two data sets must each be allocated in a single contiguous extent. The job shown below performs this allocation.

```
//APLALLOC JOB (ACCOUNT),NAME
//     EXEC PGM=IEFBR14
//APLLIB0 DD DSN=APLSVS.APLLIB0,
//     SPACE=(CYL,(80),,CONTIG),DISP=(NEW,KEEP),
//     UNIT=2314,VOL=SER=APLPAK
//APLSWAP0 DD DSN=APLSVS.APLSWAP0,
//     SPACE=(CYL,(7),,CONTIG),DISP=(NEW,KEEP),
//     UNIT=2314,VOL=SER=APLPAK
```

Figure 2 indicates that all the data sets required for generation and use of the 10 port APLSV system will easily fit on one 2314, so APLPAK will be the volume serial supplied to the installation assembly.

If the number of tracks required for swap or libraries exceeds the capacity of a single direct access volume, additional data sets on other volumes must be specified. Each data set used must consist of a single continuous extent.

Specifying Installation-Dependent Parameters

The APLSV installation is specified by a sequence of installation macro instructions which are summarized at the end of Part I. The specification macro instructions may be assembled in any order, but must precede the production macros which actually produce the necessary control statements in the Assembler SYSPUNCH data set. There is no requirement that all of the parameters for a macro be specified on a single call. For example,

```
APL UNIT=2314
APL SVC=255
```

is equivalent to

```
APL SVC=255,UNIT=2314
```

Referring to the sample installation contained in Figure 1, and examining the macros one at a time:

```
APLSWAP UNIT=2314,VOL=APLPAK
```

specifies that the APLSV swap data set will be on the 2314 volume APLPAK.

```
APLLIB UNIT=2314,VOL=APLPAK,DIRS=2
```

specifies that the APLSV user library will also be on APLPAK, and will contain 2 user directories. If the user libraries are to be contained on several volumes, each volume must be specified on an APLLIB statement. Note that the data set on the first volume specified must be at least large enough to contain the user directories.

```
APL UNIT=2314,VOL=APLPAK
```

indicates that the APLSV load-module data set should be built on APLPAK. The default WSSIZE of 64400 will be used, and this parameter is not explicitly specified.

```
APLGEN UNIT=2314,VOL=APLPAK
```

indicates that the generation macro data set has been restored to APLPAK, and also that the punch-module data set will be built there.

```
TSIO UNIT=2314,VOL=APLPAK
```

indicates that the TSIO catalog and the TSIO data set which will contain a table of valid users should also be allocated on APLPAK by the installation job.

Assuming that the host system is OS/VS1 release 2 and that the system generation has provided 255 as an available type 1 SVC, that the characters 'APL' will be used as the initial characters of acceptable APLSV, Utility, and TSIO job names, and that SYSPAK is the volume serial number of the volume containing SYS1.NUCLEUS, the APLSV installation assembly is supplied with two macros which contain the following:

```
APL SVC=255,JOBNAME=APL
SYSRES UNIT=2314,VOL=SYSPAK,TYPE=VS1R2
```

The work files could all be allocated on a specific volume, for instance WORK01, by specifying

```
WORK UNIT=2314,VOL=WORK01
```

However, in Figure 1 the allocation of these temporary data sets is left to the system by specifying

```
WORK UNIT=2314
```

The tape unit which will be used during APLSV installation is specified

```
TAPE UNIT=2400-3
```

Finally, the BLKSIZE parameter for the system macro library, SYS1.MACLIB, is necessary, so that the system macro library can be catenated with the APLSV macro library for the installation job. If not known, the blocksize of the system macro library can be obtained from the LISTVTOC FORMAT operation of the OS/VS utility IEHLIST.

```
SYSRES MACBLK=3520
```

The installation procedure expects the system macro library to be cataloged. It does not alter the library.

Suppose the terminal configuration of the 10-port system consists of six dial-up lines and four hard-wired terminals: two 2741s, a 1050 system, and a 2740-1; with the following multiplexor channel addresses:

```
1E,1F,20,21,22,23  The six dial-up lines
30                 The 1050 system
31                 One 2741
33                 The other
40                 The 2740-1
```

Suppose further that of the 2741s, the one on port 33 prints APL properly with the 988 typing element and the one on port 31 with the 987. Then the 2741 which uses the 987 is a Correspondence 2741.

```
APLPORT UNIT=X'31',TYPE=2741
```

The other is a PTTC/BCD 2741, coded

```
APLPORT UNIT=X'33',TYPE=TS41
```

The 2740, which uses the 987 element, is coded as though it were a 2741:

```
APLPORT UNIT=X'40',TYPE=2741
```

and the 1050 is coded:

```
APLPORT UNIT=X'30',TYPE=1050
```

Finally, the dial-up ports are coded:

```
APLPORT UNIT=(X'1E',X'23'),TYPE=AMBIG
```

Whenever two unit addresses are given in an APLPORT macro, they are assumed to define the limits of a sequence of addresses, all of which are to be used with APLSV.

The APLSV system has now been specified. The macro instruction INSTALL will produce a deck containing the JCL statements necessary to install APLSV and TSIO on APLPAK, create a secondary nucleus in SYS1.NUCLEUS on SYSPAK, and connect the main system catalog on SYSPAK to the TSIO catalog on APLPAK. However, the JOB card for this deck must be supplied. This can be done with an assembler REPRO statement, as follows:

```
         REPRO
//APLINST JOB (ACCOUNT),NAME
         INSTALL
```

The decks to run APLSV, the APLSV Utility, and TSIO are produced by the production macro RUN. Each requires a JOB card. The job names of the APLSV jobs must begin with the characters which were specified in the JOBNAME parameter of the APL macro. Thus,

```
         REPRO
//APLUTIL JOB (ACCOUNT),NAME,TYPRUN=HOLD
         RUN UTILITY
         REPRO
//APLSV JOB (ACCOUNT),NAME,TYPRUN=HOLD
         RUN APL
         REPRO
//APLTSIO JOB (ACCOUNT),NAME,TYPRUN=HOLD
         RUN TSIO
         END
/*
```

produces the decks. An END card terminates the assembly, and because the *//SYSIN DD DATA* statement was used, the *END* card must be followed by a delimiter.

Installing APLSV

The actual installation is performed by copying the installation macro library from the distribution tape to disk and then running the assembly just described. This assembly will produce four jobs in the SYSPUNCH data set which must be large enough to contain approximately 1,000 card images. The four jobs should be separated, and the job APLINST run to produce the APLSV system on APLPAK and create a second VS1 or VS2 nucleus on SYSPAK. SYS1.NUCLEUS must be large enough to contain a secondary nucleus.

Once the APLINST job has run to completion, the secondary nucleus, IEANUC02, in the SYS1.NUCLEUS data set on SYSPAK must be loaded. In order to do this the operator must set the rate switch on the CPU to instruction step before pressing the LOAD button. When the machine stops, he must enter alter/display mode and alter memory location 8 to hexadecimal F2. He must then set the rate switch back to process, and press start. If the system has been properly installed, the remainder of the IPL will be unchanged.

When the IPL procedure is complete, the APLUTIL job can be run. If this runs to completion, reading the first file on the distribution tape and producing a listing of the distribution libraries, the OS/VS modification has been successful, and the nucleus members can be renamed as follows:

```
//RENAME JOB (ACCOUNT),NAME
// EXEC PGM=IEHPROGM
//SYSPRINT DD SYSOUT=A
//A DD UNIT=2314,VOL=SER=SYSPAK,DISP=OLD
     RENAME DSNAME=SYS1.NUCLEUS,VOL=2314=SYSPAK,                    X
     MEMBER=IEANUC01,NEWNAME=IEANUC03
     RENAME DSNAME=SYS1.NUCLEUS,VOL=2314=SYSPAK,                    X
     MEMBER=IEANUC02,NEWNAME=IEANUC01
```

This will make the system nucleus which supports APLSV the primary one, and eliminate the need for the special IPL procedure to load an alternate nucleus.

If an existing *APL*\360 system is being replaced, a full dump tape taken with the *APL*\360 utility should be supplied to the APLUTIL job in place of the distribution tape. The distribution libraries may then be added to the new system with the APLSV utility operation SELREST.

The installation of APLSV and TSIO is now complete.

The APLSV job is used to start the APLSV system. If TYPRUN=HOLD was specified on the job cards, both APLSV and APLTSIO can be read in, APLSV can be released, and when the messages

```
DQCT05I APL HAS 2 SLOTS, 66 BUFFERS
DQCR01I SM SIZE 32200
```

appear, indicating that APLSV has started successfully, TSIO can be released. It should soon type

```
DQCB00I  TSIO 370 NOW ACTIVE
```

The operator should sign on by typing, at a terminal to be used with APLSV

```
)314159 (CR)
```

where the APL) character is the upshift of the typewriter key adjacent to the bottom of the return key, and (CR) indicates the return key, which must follow all keyboard entries.

When the APLSV operator has been successfully signed on, a

)LIB 1

command will produce a listing of the workspaces supplied with the system and also test the swapping and library data sets. The APLSV operator should

)LOAD OPFNS

and follow the instructions produced to verify that APLSV is operating correctly.

TSIO can then be checked for proper operation by

)LOAD TSIOPS
)QUOTA 314159 100 20

and then

FORMAT

which will initialize the TSIO user data set in preparation for enrolling users. The workspaces *OPFNS* and *TSIOPS* are explained in more detail in the operating sections of this manual.

Users may be enrolled in APLSV by use of the *)ADD* and, if necessary, *)QUOTA* commands; and in TSIO by use of the workspace *TSIOPS*. Specific instructions for the use of these commands are found in the section privileged system commands of PART II, and in the section TSIO operation, PART III.

rsion from APL\360

If the APLSV system is to replace an APL\360-OS system, the owing procedure should be followed:

Using the APL\360 Utility, DUMP the APL\360 user libraries to tape. Note that a fulldump, and not an INCDUMP, must be used.

2. Install APLSV as described.

3. Using the APL\360 dump tapes obtained in the first step as input for the APLSV Utility, perform a CREATE 1 operation to convert the APL\360 libraries to APLSV libraries.

   NOTE: This operation will clear the state indicators of all APL\360 workspaces. If logical damage is discovered in an APL\360 workspace, the workspace name will be written on SYSPRINT and an attempt will be made to correct the damage. Failure during damage correction results in a hexadecimal dump of the workspace on SYSPRINT. In some cases, workspaces may be recovered by starting the APL\360 system on a controlled basis, and performing the sequence)CLEAR)COPY N)SAVE N for each workspace so damaged, then taking a SELDUMP of the affected workspace with the APL\360 Utility and using the SELREST operation of the APLSV Utility to restore the seldump tape.

4. Use the APLSV Utility SELREST operation to restore the APLSV distribution library.

APLSV Installation Macros

APLSV Specification Macros

APL	UNIT= VOL=	The unit type and volume serial of the volume to contain the APLSV and TSIO load module data set, APLSVS.LOAD.
	SVC=	The APLSV type 1 SVC number, between 200 and 255, inclusive.
	WSSIZE=	The size of the APLSV workspaces. The default value is 64400. This size may be from 20480 to 211200 (2314) or 386400 (3330).
	SLOTS=	The number of areas in virtual storage reserved for workspaces. The minimum default is 2.
	JOBNAME=	0 to 6 characters which will begin the job names of APLSV, the APLSV Utility, and TSIO. The default is 'APL'.
APLLIB	UNIT=	The direct access device type for the APLSV user libraries. May be 3330 or 2314.
	VOL=	A list of up to 32 volume serials of volumes to be used for user libraries. NOTE: Library devices must all be of the same unit type.
	DIRS=	The number of user directories for accounting information. A 64400 byte directory will contain approximately 200 users. Should be a prime number greater than or equal to 2. If the APLLIB macro occurs more than once, each VOL entry adds to the list of volumes.
APLSWAP	UNIT= VOL=	The unit type and volume serial to be used for the APLSV swap data set. Repeated calls of APLSWAP add to the list of volume serials and unit types. Up to ten extents may be specified, but one will normally suffice. (See Swap and Library Data Sets, above).
APLPORT	UNIT=	The multiplexor address for a terminal access port. If the multiplexor channel is not channel 0, three-digit hexadecimal addresses must be specified. Note that APLSV supports only one multiplexor channel. If a list of two addresses is specified (see example installation), it is assumed to define the limits of a sequence of addresses, all of which are to be used by APLSV.
	TYPE=	The type of port: AMBIG indicates dialup, Hard wired terminals are specified as follows: TS41 - PTTC/EBCD 2741 or 2740-1 2741 - Correspondence 2741 or 2740-1 1050 - any 1050 system.
	SAD=	For a 2702, the SAD command to be used. Specify SAD0, SAD1, SAD2, or SAD3. The default is SAD1. The SAD command is not relevant for a 2703.

	EXPRESS=	NO or YES, default is NO. If YES, the port will be an express port when APLSV is started. Express ports are discussed in the APLSV operation section of this manual.

TSIO Specification Macros

TSIO	UNIT= VOL=	The unit type and volume serial for the TSIO user directory, TSIO.UTABLES, and catalog. The TSIO catalog will be connected to the main system catalog by the IEHPROGM utility, as INDEX=TSIO by the installation job.

General Specification Macros

SYSRES	UNIT= VOL= TYPE= MACBLK=	The unit type and volume serial on which the data sets SYS1.NUCLEUS resides. The APLSV type 1 SVC will be linkedited into the nucleus IEANUC01 to produce nucleus IEANUC02. VSxRy, where x is 1 for VS1 2 for VS2 and y is the release number. For example: VS1R1 Release 1.0 or 1.6 of VS1 VS1R2 Release 2.0 or 2.6 of VS1 VS2R1 Release 1.0 or 1.6 of VS2 The BLKSIZE of SYS1 MACLIB.
APLGEN	UNIT= VOL=	The unit type and volume serial to be used for the APLSVS.GENLIB and APLSVS.PUNCH data sets during installation.
WORK	UNIT= VOL=	The unit type and volume serial for scratch data sets. VOL= may be omitted, in which case a non-specific request for space is made.
TAPE	UNIT=	The type of tape unit to be used for APLSV installation. If not specified, 2400-3 is assumed.

Production Macros

INSTALL	blank APL TSIO	Install both APLSV and TSIO. Install APLSV only. Install TSIO only.
RUN	APL TSIO UTILITY	Produce APLSV run deck. Produce TSIO run deck. Produce Utility run deck.

LIBRARY	RETENTION	DSNAME	ALLOCATION TYPE	SPACE 2319	SPACE 3330	NUMBER OF DATA SETS	MACRO
APLSV user libraries	P	APLSVS.APLLIBX	1	SEE STORAGE REQ		1-32	APLLIB
APLSV swap data set	P	APLSVS.APLSWAP0	1			1	APLSWAP
APLSV installation macros	G	APLSVS.GENLIB	2	3 cylinders	2 cylinders	1	APLGEN
APLSV load module library	P	APLSVS.LOAD	3	17 cylinders	10 cylinders	1	APL
APLSV object module library	G	APLSVS.PUNCH	3	6 cylinders	4 cylinders	1	APLGEN
TSIO user table	P	TSIO.UTABLES	3	2 tracks	1 track	1	TSIO
TSIO local catalog	P	SYSCTLG	3	1 cylinder	1 cylinder	1	TSIO
WORK files for assembly,etc.	G	-	3	5 cylinders	5 cylinders	3	WORK

Allocation Types
1 User must perform
2 Allocated by IEHMOVE
3 Allocated during installation job

Retention
P permanent
G System installation Only

FIGURE 2: APLSV SYSTEM DATA SETS

PART II

APLSV OPERATION

Introduction

The responsibilities of the operator of a system which includes APLSV are somewhat different from those of an operator of a batch system, primarily because APLSV is interactive and users expect to communicate with the operator when they have questions. An APLSV system is a utility and should be constant and reliable. The user builds his opinion of the system on the basis of its reliability and quality of service as experienced at the terminal.

From the point of view of the host operating system, APLSV is a job which can be started by a start command, and which normally terminates by conventional return. The host system operator's console is used only for starting and stopping the system. APLSV itself is managed by APLSV system commands and operator functions entered at a particular APLSV terminal referred to as the recording terminal. The APLSV Utility, which is used to build and maintain the APLSV libraries, is run as a batch job, normally when the APLSV terminal system is not running.

The APL\360 User's Manual (IBM Publication GH20-0906), and the APLSV User's Manual (IBM Publication SH20-1460) contain instructions for signing on and off the APLSV system and for using system commands. Knowledge of APL by the APLSV Operator is not required but is useful, since it enables the operator to make more effective use of the operator functions and to answer user inquiries about system operation directly.

Starting APLSV

The date and time must be correctly set before APLSV is started. Changing the date or time settings while APLSV is running will invalidate APLSV accounting information.

The job to run APLSV punched by the RUN APL macro during APLSV installation can be entered by an OS/VS reader and initiated by an initiator, or it can be cataloged in SYS1.PROCLIB and started from the OS/VS operator's console by a start command of the form:

```
s aplsv
```

APLSV is made up of four modules whose names each begin with the APLSV component code, DQC, followed by the letter E denoting an executable module, followed by a letter which denotes the kind of module:

A an APLSV interpreter module
S an APLSV supervisor module
R a shared-variable processor module
T a supervisor initialization module

The remaining two characters in each name specify the version to be used. Normally, the names are DQCEA00, DQCES00, DQCER00 and DQCET00.

The APLSV region or partition is divided into two areas in addition to the area where the code resides. The first area, specified in parentheses after the supervisor name, contains buffer areas for terminals and workspace slots. This area must be large enough to contain at least two workspaces, and six buffers for each port configured. Each buffer is 32 bytes long. If the size of this area is not specified or if the area specified is too small, the minimum possible storage will be used. The second area is used by the shared variable processor. It must be at least twice as large as the largest value to be passed between sharing processors. The message *INTERFACE CAPACITY EXCEEDED* will be received by *APL* users who attempt to pass values larger than allowed by this parameter. If this parameter is not specified, 10K will be used.

Replying to APLSV Startup Messages

If the APLSV initialization routine detects an error in the parameter field or fails to find a module in the load library, a message is printed at the OS operator's console and a correction is requested. The possible actions are:

1. Cancel the job, by replying 'CANCEL' to the message.

2. Ignore the error, by replying 'U'.

3. Correct the error by specifying a replacement module in the same way it would be specified in the execute statement or start command. To obtain a listing of the modules to be loaded and the storage to be allocated, include ,* in the reply. All entries must be delimited by commas, and storage requests should be enclosed in parentheses following the appropriate module name.

Figure 3 shows a sample error message and the appropriate correction.

When initialization and verification have successfully completed, the messages

```
DQCT05I APL HAS yyy SLOTS, nnn BUFFERS
DQCR00I SM SIZE zzz
```

are sent to the OS operator, indicating the number of APLSV workspaces (SLOTS) which will reside in the APLSV region concurrently, the number of input/output buffers available for terminal operation, and the size of the storage area for shared variables (SM). When these messages are printed, APLSV is operating. Users may not use APLSV, however, until the APLSV recording terminal has been signed on as described in the next section.

```
s aplsv,,,(a11,s11)
DQCI00I MODULES NOT FOUND
      DQCEA11

*00 DQCI01W GIVE MODIFICATIONS OR U
r 00,'A00,*'
DQCI04I MODULE     SIZE
        DQCEA00
        DQCER11    10000
        DQCES11
        DQCET11
*00 DQCI01W GIVE MODIFICATIONS OR U
r 00,u

DQCT05I APL HAS 2 SLOTS, 1023 BUFFERS
DQCR00I SM SIZE 9240
```

FIGURE 3: SAMPLE STARTUP REPLY

The Recording Terminal

The terminal signed on with the account number 314159 automatically becomes the APLSV recording terminal. It behaves like any other APLSV terminal, with these exceptions:

1. Its keyboard is normally locked. Attention must be signalled before input can be entered.

2. Messages are logged on it whenever users sign on or off the APLSV system. Users can sign on and off only when its keyboard is locked.

3. It is always privileged. System management commands and functions which result in error messages when attempted at non-privileged terminals can be executed at the recording terminal.

4. Output interrupted by an attention signal is retransmitted.

The recommended recording terminal is an IBM 2741 Communications Terminal with the interrupt feature. Other terminal types can be used, but the recording terminal must have a mechanism for signalling attention.

The recording terminal may be signed on by typing)314159 (followed by a colon and password if one has been established). After the recording terminal has been signed on, there is a 30-second grace period before other users are allowed to sign on, giving the APLSV operator time to load a workspace of operator functions, set express ports if necessary, and establish an appropriate message to be sent to users as they sign on, using the *)HIPA* command as described in Privileged APLSV system commands.

)HIPA is one of a class of system commands which may be used only at a privileged terminal (for example, the recording terminal) and thus are not documented in the user's manuals. Privileged system commands are described in a later section of this manual. Similarly, the system management functions contained in the *OPFNS* workspace may be executed only at a privileged terminal.

Terminating APLSV

If APLSV cannot run for some reason detected by the APLSV initialization routine, execution terminates and one of the messages from Appendix A is printed on the OS operator's console.

APLSV, once initiated and running, requires no action from the OS operator's console; all communications except those associated with selector channel errors are directed to the recording terminal. APLSV may, in an emergency, be terminated by a stop command (for example, p aplsv). Users signed on at the time will lose their active work, and current accounting information will not be saved if this is done.

The following procedure, executed at the recording terminal, is recommended for the normal termination of APLSV:

1. Execute the *SHUTDOWN* function which is contained in the distributed workspace *OPFNS*. If this is unavailable, type 12⌶0.

2. Use the *)PA* command to advise users that system operation is about to terminate.

3. After allowing a suitable time for users to clean up their work and sign off, use *BOUNCE ALL* to force off any remaining signed-on users. Their active workspaces will be saved in their libraries under the name *CONTINUE*.

4. When all ports but that occupied by the recording terminal are signed off, sign off the recording terminal. APLSV will then terminate, using the standard OS return mechanism, about ten seconds after the accounting information has completed typing on the recording terminal. Users who have not signed off or have not been forced off when this occurs will not be signed off and will not have their active workspaces saved in *CONTINUE*.

Recording Terminal Messages

Messages are sent to the recording terminal when a user signs on or off, and when a user enters a *)OPR* or *)OPRN* command. Examples:

```
010) 15.35.23 09/17/72 ADFALKOFF 271828
004  15.37.02 09/17/72 LAM
017:RWHEN IS THE SYSTEM SHUTTING DOWN?  KEI
062: I'LL BE DOWN WITH COFFEE IN 5 MINUTES. RHL
```

The number in the left margin is the number of the access port from which the message was sent. The port number is followed, for sign-on messages, by a right parenthesis; for sign-off messages, by a blank; and for user-generated messages, by a colon. An underlined *R* following the colon indicates that the sender used the *)OPR* command, and is awaiting a reply. If no *R* follows, the command was *)OPRN* and no reply is expected.

Any message sent to the operator before a user has signed on causes his keyboard to lock, awaiting a reply. The user cannot proceed with his sign-on until the operator replies.

NOTE: Users who attempt to sign on while the APLSV operator's keyboard is unlocked are not allowed to proceed until the carriage is locked.

Privileged APLSV System Commands

The system commands described here are privileged, and may be used only by a user at a privileged terminal. Chapter 7 of the APL\360 User's Manual presents in more detail the mechanics of using APLSV system commands.

)ADD account-number user-name[:lock] quota-adjustment [cpu-limit]

Effect: Enroll a new user, create a new public library, or change a previously enrolled user's name, account-number lock, workspace quota, or cpu-limit. (See also the *)QUOTA* command)
Normal response: none.
Notes: Account numbers 1 through 999 designate public libraries. Numbers higher than 1000 designate individual users, and their private libraries. The user-name may contain from 1 to 11 alphanumeric characters, the first of which must be alphabetic. The first 3 characters of the user-name are used for identification in the *)PORTS* command and in operator functions such as *USER*. The quota-adjustment specifies the increase (or decrease if negative) in the largest number of workspaces a user may store, excluding the workspace named *CONTINUE*. A quota adjustment of zero does not change the quota. For a new user, the initial quota is zero. An account-number lock may be changed but not removed. If no new lock is given, the lock remains unchanged. The cpu-limit is an integer or decimal fraction that designates the number of seconds (rounded to tenths) of CPU time allowed a user for processing a single input line. If the cpu-limit parameter is omitted, it is unchanged from its previous value. A cpu-limit of zero is taken as infinity. The initial cpu-limit value is infinity. User-name and quota-adjustment must be given when adding a public library even though they have no effect.

Examples:

)*ADD* 1234 *JKTUTTLE:HERB* 4
add J. K. TUTTLE to the system with password HERB and a quota of 4 workspaces.

)*ADD* 1239 *PCBERRY* ¯2 .6
decrease P.C. Berry's workspace quota by 2 and limit him to 6 tenths of a second of CPU time per input line.

)*DELETE* account-number

Effect: Remove a public library or user account number from the APLSV directories, including any accumulated accounting information, and drop workspaces associated with the number.
Normal response: none.
Example:)*DELETE* 1212

)*HI* up to 120 characters of text
Effect: Prepare a message to be sent to each user as he signs on.
Normal Response: None
Example:)*HI*)*LOAD* 1 *NEWS PRINT* 1 *FOR SYSTEM CHANGES*.
Notes:)*HI* followed by a carriage return deletes any previous *HI* message.

)*PA* up to 120 characters of text

Effect: Send a public-address message to all users
Normal response: none.
Example:)*PA APL SHUTTING DOWN FOR MAINTENANCE AT* 23:55.
Notes: A *PA* message may spoil the appearance of users' output, and should be used only when an urgent message must be sent. The *PAWAIT* function in *OPFNS* can be used to determine the port numbers of any ports which have not received the most recent *PA*.)*PA* followed by a carriage return cancels the *PA*.

)*HIPA* up to 120 characters of text

Effect: Combined)*HI* and)*PA* commands.
Normal response: none.
Example:)*HIPA APL RUNNING UNATTENDED UNTIL* 6*AM APRIL* 4*TH*.
Note:)*HIPA* is normally used when starting the system to be sure that all users see the *HI* message.

)*LOCK* account-number

Effect: Prevent a user from signing on.
Normal response: none.
Notes: If the locked user attempts to sign on he will receive the trouble report *NUMBER LOCKED OUT*.)*LOCK* has no affect other than to prevent the user from signing on. The user's saved workspaces are not affected. Account numbers locked out are preceded by * in the APLSV Utility *ACCTG* listing.
Example:)*LOCK* 2223

)QUOTA account-number [library-quota] [shared-variable quota] [cpu-limit]

Effect: Set and print the quotas pertaining to a particular account.
Normal response: New values of the three quotas and the current number of saved workspaces are printed, followed by the user's sign-on identification.
Notes: This command differs from the *)ADD* command in that the library quota, rather than an adjustment, is specified. Entering the command with only the account number allows the operator to see the current values. The library quota specifies the total number of workspaces, exclusive of *CONTINUE*, which may be saved by the user. The shared-variable quota specifies the maximum number of variables a user may share at one time. Users who attempt to share more variables than allowed will receive the message *INTERFACE QUOTA EXHAUSTED*. The cpu-limit specifies, in tenths of seconds, the maximum amount of CPU time a user is permitted to process a single input line. Zero denotes no limit. (see also the *)ADD* command).
Examples:

```
      )QUOTA 1134
1134 10 5 .0 4 LEYOUNG
```

User 1134 has a quota of 10 workspaces and 5 shared variables, and currently four workspaces are saved.

```
      )QUOTA 1134 10 8
1134 10 8 .0 4 LEYOUNG
```

Increase user 1134's shared-variable quota to 8.

)UNLOCK account-number

Effect: Reinstate a previously locked-out user.
Normal response: none.
Example: *)UNLOCK* 2223

Use of Non-Privileged Commands

The non-privileged commands shown below have characteristics slightly different from the same commands executed at a normal APLSV terminal when executed from the recording terminal:

)OFF
)CONTINUE

The APLSV recording terminal can be moved from one 2741 to another while APLSV is running by simply signing off the first 2741 and signing on the second. Users will not be able to sign on or off while the operator is signed off. APLSV terminates following the operator's sign off only if the *SHUTDOWN* operator function (or 12I0) has been executed.

)LIB
)SAVE
)DROP

A privileged terminal may execute these commands for any public or private library. An attempt to save or drop a workspace belonging to another user is executed as if the user himself were saving or dropping the workspace.
Example: *)DROP* 1234 *CONTINUE*

)MSG
)MSGN

There is no distinction between these commands, since the recording terminal is ready to receive a reply any time the keyboard is locked. Messages from the recording terminal are never preceded by the character *R*.

)MSG OFF and *)MSG ON*

result in *INCORRECT COMMAND* when attempted from the recording terminal.

Passwords

Passwords on saved workspaces are inviolate. A privileged user can neither override nor learn a workspace password. If a user saves a workspace using a lock and then forgets the lock, his only recourses are: 1) to drop the workspace; or 2) have an earlier version, whose key is known, retrieved from an APLSV dump tape by the APLSV Utility. Account-number passwords can be changed at a privileged terminal by an *)ADD* command, but cannot be removed.

The APLSV Operator's Workspace, *OPFNS*

The workspace 314159 *OPFNS* (Operator Functions) contains a collection of APL programs which are specialized functions for controlling the running system. They enable the operator to do such things as set or remove express time limits, determine free port numbers, and forcibly sign off designated users. In addition to the specialized functions, the APLSV operator has available the full facilities of APL. The operator functions will often be used in combination with APL expressions and with each other.

Ports Each line available for use by users of the APLSV system is referred to as a port. Each port in the system has a port number, which is typed to the user when he signs on and off. Some functions in the *OPFNS* workspace take port numbers as arguments (for example, *BOUNCE* 1 4 23 would force users on ports 1, 4, and 23 off the system). Others return port numbers as results (for example, *SUSPECT* 20 returns the numbers of ports which, though signed-on, have been idle for 20 minutes.) The functions which take port numbers as arguments can also take as arguments the results of functions which return port numbers (thus, if *SUSPECT* 20 returned 4 29 38 as the port numbers of users who have been idle for 20 minutes, *BOUNCE SUSPECT* 20 would force the users on ports 4, 29, and 38 off the system).

Some functions in the operator's workspace may be used to combine port numbers (thus *BOUNCE (SUSPECT* 20) *EXCEPT* 4 would, if 4 29 and 38 were the port numbers of idle users, force users on port numbers 29 and 38 off the system, but would not affect the user on port 4.)

The operator functions all ignore invalid port numbers. Thus, *BOUNCE* 99999 would result in the report *OKAY* but no action.

Express Ports A system with many more enrolled users than ports may find it convenient to offer some ports on which an automatic limit to the length of a session has been imposed. When a user signs on such an express port, he receives the message *.APL.SV. EXPRESS.* After the express time-limit has expired, the user is forced off and the active workspace is automatically saved as the workspace *CONTINUE* in his private library. A suitable time limit is typically 5 to 15 minutes. On express ports the *)OFF HOLD* and *)CONTINUE HOLD* system commands behave like *)OFF* and *)CONTINUE* respectively.

Ports may be designated as express ports during system installation and while APLSV is running.

Operator Functions The following sections describe the principal functions included in the distribution workspace *OPFNS*. The variable *DESCRIBE* in *OPFNS* contain a brief summary of the functions, listed alphabetically. Functions in *OPFNS* not described are auxiliary functions which are used only by other functions.

The following functions report or alter system values that relate to all ports of the system.

NAME and SYNTAX DESCRIPTION

FREE

Returns the phone numbers of all ports not currently signed on. Phone numbers are associated with ports by means of the function *INITIALIZE*.

```
      FREE
2496600 2496603 2496608
```

SETLIMIT n

Sets the express time limit to n minutes, to take effect for any subseguent sign-on at any express port.

```
      SETLIMIT 15
8
                                  (returns the prior limit)
```

SHOWHI

Reports the current HI message to be printed with each sign-on.

```
      SHOWHI
OPR: PHILA APL RUNNING UNATTENDED UNTIL 0730EDT
WED, AUG. 5.
```

SHOWLIMIT Returns the current time limit for express ports, in minutes.

```
      SHOWLIMIT
8
```

SHOWPA Reports the last PA (public address) message sent.

```
      SHOWPA
                    (blank line means no current PA)
```

SHUTDOWN Conditions the system for APLSV shutdown. All ports not in use are disabled, and no subsequent sign-on will be accepted. After the operator signs off at the recording terminal, APLSV operation terminates. *SHUTDOWN* is not a reversible process

```
      SHUTDOWN
APL SYSTEM SHUTDOWN INITIATED
```

The following functions report or alter the status of ports of the system; vp stands for one or more port numbers. Some of the examples given refer to functions which are described later, but whose meaning is obvious.

NAME and SYNTAX DESCRIPTION

BOUNCE vp Forces the users on the designated ports (with the exception of the recording terminal) to be signed off, after saving the active workspaces in *CONTINUE*.

```
      BOUNCE 2
OKAY
```

DEFUSE vp Removes the time limit for the current user of each of the designated express ports. The time limit will be restored when the user signs off.

```
      DEFUSE 18 19 20
OKAY
```

DEPRIVILEGE vp Removes the users at the designated ports (with the exception of the recording terminal) from the privileged state.

```
      DEPRIVILEGE PRIVILEGED
OKAY
```

KBLOCK vp Makes the keyboards of the terminals on the designated ports "normally locked". This is the normal state of the recording terminal. In this state a terminal can receive messages any time the keyboard is locked

```
      KBLOCK USER 809380
OKAY
```

KBUNLK vp

Makes the keyboards of the terminals on the designated ports "normally unlocked". This is the normal state of a user's terminal. It is unwise to leave the recording terminal in this state when others are using APLSV because sign-on and sign-off attempts will be delayed until the keyboard is locked.

```
      KBUNLK USER 'JKT'
OKAY
```

LIMIT vp

Makes the designated ports express ports.

```
      LIMIT PHONE 2496600 2496601 2496602
OKAY
```

PORT vp

Reports the port number, phone number, multiplex address, user initials and account number for each of the designated ports.

```
      PORT ON
  2 2496601 012  LAM 267386
  3 2496602 013  XPB 4000000
 22 2496621 026  P4I 1234
 48 2496647 040  FCD 418965
128 2496727 098  OPE 314159
130 2496729 09A  APL 1776
```

PORTS vp

Reports in more detail than *PORT* the status of the designated ports. The additional information includes the connect-time, CPU-time, and sign-on time of the port.

PRIVILEGE vp

Places the users at the designated ports in the privileged state. This will remain in effect, unless revoked via the *DEPRIVILEGE* function, until the user signs off. In general, no one but the operator and APLSV system programmers should ever be privileged.

```
      PRIVILEGE 44
OKAY
```

UNLIMIT vp

Removes the express port status of the designated ports. (Note the difference between *UNLIMIT* and *DEFUSE*.)

```
      UNLIMIT PHONE 2496600 2496601 2496602
OKAY
```

The following functions return a vector of port numbers; vi stands for one or more integers.

NAME and SYNTAX DESCRIPTION

ALL
Returns the numbers of all ports in the system in ascending order.

```
      ALL
1 2 3 4 5 6 7 8 9
```

DEFUSED
Returns the numbers of the ports that are currently express but which have been defused by the *DEFUSE* function.

```
      DEFUSED
2 3
```

DOWN
Returns the numbers of the ports that are disabled

```
      DOWN

```

(empty line indicates no ports are disabled.)

EXPRESS
Returns the numbers of the ports currently in express status.

```
      EXPRESS
1 2 3
```

MPXAD 'xx'
Returns the number of the port corresponding to the hexadecimal multiplexor channel address xx, which must be enclosed in quotes.

```
      MPXAD '03'
6
```

OFF
Returns the numbers of the ports not currently signed on.

```
      OFF
3 6 7 9
```

ON
Returns the numbers of the ports currently signed on.

```
      ON
1 2 4 5 8
```

PAWAIT
Returns the numbers of the signed-on ports that have not received the most recent PA message.

```
      PAWAIT
2
```

PHONE vi
Returns the numbers of the ports corresponding to the designated phone numbers.

```
      PHONE 2496603
4
```

PRIVILEGED
Returns the numbers of the ports whose users are currently privileged.

```
      PRIVILEGED
8
```

SUSPECT m — Returns the numbers of the ports signed on but with no activity in the last m minutes

```
      SUSPECT 20
4
```

USER vi — Returns the numbers of the ports at which users with the designated account numbers are signed on.

```
      USER 3001 260419
4 0
```

(0 indicates not signed on.)

USER 'abc' — Returns the numbers of the ports at which users with the designated initials are signed on. (See also the)*ADD* command.)

```
      USER 'JAB'
5
```

The following functions produce groups of port numbers from groups of port numbers; vp1 and vp2 stand for one or more port numbers.

NAME and SYNTAX DESCRIPTION

vp1 *AND* vp2 — Returns the numbers of the ports that occur in both vp1 and vp2.

```
      1 2 3 AND 2 3 4
2 3

      PRIVILEGED AND NOT USER 314159
5
```

VP1 EXCEPT vp2 — Returns the numbers of the ports that occur in vp1 but not in vp2. (vp1 *EXCEPT* vp2 is the same as vp1 *AND NOT* vp2)

```
      1 2 3 EXCEPT 3
1 2

      PRIVILEGED EXCEPT USER 314159
5
```

NOT vp2 — Returns the numbers of the ports that do not occur in vp2, in ascending order. (*NOT* vp2 is the same as *ALL EXCEPT* vp2

```
      NOT ON
3 6 7 9
```

vp1 *OR* vp2 — Returns the numbers of the ports that occur in either vp1 or vp2, in ascending order.

```
      1 2 3 OR 2 3 4
1 2 3 4
```

Multiprogramming Control APLSV multiprogramming is metered by a unit of time called a quantum. The quantum is the maximum time which can pass before a scheduling decision is made.

When operating in a multiprogramming environment, APLSV ensures that other partitions receive frequent CPU service by alternating its own priority between high and low. When APLSV has low priority, other partitions will get CPU service if they are not quiescent.

The nominal proportion of time that APLSV has high priority is controlled by two system parameters, preset in the distributed system to 50 percent but modifiable at any time by the use of the *PRIORITY* function. Unless APLSV and some other partition both require heavy CPU service, the priority setting will have little or no effect on system behavior.

NAME and SYNTAX DESCRIPTION

PRIORITY a,b — Sets and reports the proportion of APLSV high-priority time. a is the proportion when no ports are in use, and b is the proportion when all ports are in use. The priority proportion varies approximately linearly with the number of ports in use. a and b are fractions, between 0 and 1. If only one value is given, a and b are taken to be equal. If APLSV is to be run by itself, system overhead may be lowered by setting *PRIORITY* to 1.

```
      PRIORITY 1
EMPTY WAS 0.5  IS 0.99
LOADED WAS 0.5  IS 0.99
```

QUANTUM x — The APLSV quantum (time-slice length) is set to x seconds. The initial value assumed by APLSV is .1 seconds. A good rule-of-thumb is that the quantum length should be approximately the same as the average time required to transfer a workspace to the swapping device.

```
      QUANTUM .05
0.1
```

(returns previous value)

Initialization Function This function can be used to establish a correspondence between APLSV port numbers and telephone numbers or other numeric identification. These can then be used as arguments to the functions *FREE*, *PHONE*, *PORT*, and *PORTS* described above.

NAME and SYNTAX DESCRIPTION

INITIALIZE

Shared Variable Processor Operator Functions The shared variable processor is the program which controls the flow of information between APLSV terminal users and auxiliary processors, such as TSIO or between two terminals that are communicating via shared variables. SVP runs without requiring action from the operator and the following functions will be of most use to system programmers.

SVMAP prints various statistics about the current state of SVP operation.

```
    SVMAP
 STARTED        OFFERS      WFS          GCOL      ⍴SVS        TRANSFERRED
07.21.45           102        0             0     29000             100256
     NOW         +/ON   1↑⍴PS      ⌈/+/ON   ⌈/1↑⍴PS           REQUESTS
22.48.30              3        2            13        22               1029
    PROCESSOR     VARS   PROCESSOR     VARS      PROCESSOR     VARS
          370        1          399        0         267386        2
```

STARTED is the time the APLSV system was initiated. *NOW* is the current time. *WFS* is the number of times users have had to wait for storage space in SVP before they could transfer a value. If this number is large, the size of SVS, the storage area in SVP, given here as *⍴SVS*, should be increased by changing the start procedure. *GCOL* is the number of times the storage area has been compacted. *TRANSFERRED* is the number of bytes transferred through SVP so far. *+/ON* gives the number of processors or APLSV users currently making use of SVP, and *⌈/+/ON* gives the largest number who have made use of SVP at any one time so far. *1↑⍴PS* gives the number of variables currently being shared or offered, and *⌈/1↑⍴PS* gives the maximum number shared or offered at any one time so far. *REQUESTS* gives the number of requests of any kind made to SVP, and *OFFERS* gives the number of variables shared so far. The remainder of the display gives the processor identification or sign-on identification numbers of all current users of the shared variable facility, and the number of shared variables each is currently using.

SVUSER id Returns a vector of identification numbers of the processors sharing variables with processor number id.

```
      SVUSER 370
267386
      SVUSER 267386
370
```

SVUSERS Lists the current users of SVP and the processors they are actively sharing variables with.

```
      SVUSERS
  370 WITH 267386
  399 WITH
267386 WITH 370
```

Service Aid Functions The functions below are intended for debugging the system, not for general use during APLSV operation. Improper use of the *REP* function can cause system failure.

'xxx' stands for a hexadecimal quantity of 1 to 8 digits, enclosed in quotes. Hexadecimal arithmetic is performed in 32-bit 2's complement. Integers without quotes may also be used with these functions, and are interpreted as decimal values.

NAME and SYNTAX	DESCRIPTION
'xxx' *AH* 'xxx'	Returns, in hexadecimal, the sum of the two values. *'3A' AH 2* 0000003C
ASUP	Returns, in decimal, the beginning address of the APLSV supervisor. *ASUP* 458362
ASUREL 'xxx'	Calculates the absolute address of relative offset 'xxx' from the beginning of the APLSV supervisor. *DISPLAY ASUREL '140'* 0000009D
DISPLAY 'xxx'	Returns, in hexadecimal, the contents of the four bytes at the designated storage address. *DISPLAY 16* 00006298
'xxx' *DISPLV* 'xxx'	Displays, in hexadecimal, the contents of the string of bytes whose length is given by the left operand, starting at the main storage address given by the right operand. *12 DISPLV ASUP* 06FF88 900FEC58 47F0E1EC 9500EAE5
DTH n	Returns the hexadecimal quantity equivalent of the integer n. *DTH 12* 0000000C
HTD 'xxx'	Returns the decimal integer equivalent of the hexadecimal quantity. *HTD '9C'* 156

'xxx' *REP* 'xxx' Replaces the four bytes of main storage designated by the left operand with the right operand. The function first lists the proposed change and waits for a reply. Typing anything except a period followed by a carriage return causes the patch to be skipped.

```
      (ASUREL '140') REP '0000009C'
0700C8 0000009D → 0000009C ?
. (entering a period permits replacement)
0700C8 0000009D→0000009C
```

SVCLEAR id Clears the Shared Variable Processor table entry for auxiliary processor number id. This function is supplied to allow writers of auxiliary processors to clear their processors from SVP in an emergency. It will never be required in normal operation of APLSV.

```
      SVCLEAR 399
0F0C20 00016D70 80000000 00003A6A...
399 CLEAR
```

'xxx' *SH* 'xxx' Returns, in hexadecimal, the difference between the two quantities.

```
      'A0' SH 10
00000096
```

'xxx' *VERIFY* 'xxx' Verifies that the four bytes of main storage designated by the left argument contain the right argument. If the verification fails, a return to immediate execution mode is forced. Most useful in writing patching functions.

```
      (ASUP AH '140') VERIFY '000000F0'
NOT VERIFIED: 0700C8 0000009C≠000000F0
```

APLSV Operating Suggestions

Usually, a user's only contact with the computer operation is through his remote terminal. During a period of system malfunction, or when schedule changes have been made too abruptly, or when APLSV does not behave as expected, the terminal can seem very remote indeed. The following suggestions are intended as a guide to running a reasonable APLSV service.

1. Publicize an Operator Assistance number for a telephone near the recording terminal.

2. Publicize and encourage the use of the workspace 1 *NEWS*. It contains functions for displaying notes to users, normal schedules, and schedule deviations. It also contains functions for the APLSV Operator's or system programmers' use in adding to or modifying these displays. Detailed instructions for

modifications are in 1 *NEWS* under the name *NEWSMAKING*. Instructions for the user are in the APL\360 User's Manual, Appendix B, and in 1 *NEWS* under the name *DESCRIBE*. The workspace 1 *NEWS* should contain adequate information regarding the auxiliary processors available at the central installation for use with the shared variable facility.

3. Make use of the)*HI* command to post last-minute notes and schedule changes (and admonitions to look at 1 *NEWS*.) If APLSV is left running with no one at the recording terminal, leave a)*HI* message saying *APL IS RUNNING UNATTENDED UNTIL* 2400, or some such. To post a message at the time you sign on, use)*HIPA* so that users signing on at the same time you do will also get the message.

4. Following a system failure, bring up APLSV as rapidly as possible. Remember that auxiliary processors must be stopped and restarted if APLSV fails.

5. Avoid arbitrary schedule changes. Remember that such a change may inconvenience many users.

6. A baffled user will sometimes request assistance from the operator via an)*OPR* message. Try to answer his question, put him in touch with an expert, or at least acknowledge receipt of the question, without undue delay.

7. Use the)*PA* command sparingly. It can interfere with a user's carefully-arranged output. Also, because there may be a significant delay between your sending a)*PA* and the user's receiving it, indicate all times by clock time -- for instance, say *SYSTEM SHUTTING DOWN AT* 21:30, not *SYSTEM SHUTTING DOWN IN* 5 *MINUTES*.

8. No more than five minutes before APLSV is scheduled to shut down, execute *SHUTDOWN* and notify users via a)*PA* command. Before signing off at the recording terminal, execute *BOUNCE ON* to forcibly sign off any remaining users. If they are not so signed off, their active workspaces and their CPU- and connect-time charges for that session will be lost.

9. Encourage users to change their account-number passwords occasionally. Always include a password when you)*ADD* a new user to the system (letting him know what it is, of course.) If an enrolled user gets the response *NUMBER NOT IN SYSTEM* when he tries to sign on, it is likely that he has inadvertently changed his account-number password. Two easy ways he could have done this are by signing off via)*OFF*: or via)*OFF*:*HOLD* . The first removes his password altogether; the second changes it to the word *HOLD*. If neither of these explanations fits, or if a user has simply forgotten his password, you can execute an)*ADD* to change his password to an agreed-upon word. Note that there is no similar method to recover a password-protected workspace.

10. Many important aspects of user security are at the APLSV Operator's discretion. The operator has the ability mentioned above to change account-number passwords; to make any other port privileged; to ascertain (via the APLSV Utility) the names of all workspaces in a library and the amount of CPU and connect time charged to a user. It is vital for user confidence and for acceptance of a time-sharing system that such facilities be used with care and discretion. For example:
    a. Require positive identification (a phone call, or a personal appearance, rather than just a terminal message) from anyone who asks to be privileged, to have his password changed, or who requests that a confidential or proprietary file be mounted.
    b. Protect your own account-number password by changing it regularly.
    c. Treat the output at the recording terminal and results of APLSV Utility accounting runs as confidential business records.

PART III

TSIO OPERATION

TSIO is an auxiliary processor supplied with the APLSV system which gives users at APL terminals access to OS/VS data sets. It is started after the APLSV terminal system is started, in a separate region from APLSV, and uses the shared variable processor for all communication with APLSV users. Details on the use of TSIO may be found in IBM publication, "TSIO Program Reference Manual" (SH20-1463).

TSIO should always be run at a higher priority than APLSV. In VS2, this is accomplished automatically by the DPRTY parameter in the EXEC card produced during installation. In VS1, however, the operator must initialize the partitions to be used for APLSV and TSIO with the VS1 DEFINE command so that the TSIO partition is higher in priority than the APLSV partition (partition 1 is higher in priority than partition 2). The APLSV partition should be defined as non-deactivatable, but the TSIO partition, if desired, may be deactivatable. For example, in response to
*06 IEE802A ENTER DEFINITION,

the reply

```
r 06,p0=(stp,128k,sn),p1=(g,128k,sa),
     p2=(g,384k,sn),p3=(ab,384k,sa,last),end
```

would initialize a VS1 system as follows:

```
P0 SYSTEM TASKS     128K NOT ELIGIBLE FOR DEACTIVATION
P1 TSIO PARTITION   128K ELIGIBLE
P2 APL PARTITION    384K NOT ELIGIBLE
P3 BATCH PARTITION  384K ELIGIBLE
```

When definition ends, the commands

```
s aplsv.p2
s apltsio.p1
```

will start the APLSV system and the TSIO processor in their correct partitions.

Note: A complete description of message IEE802A is contained in IBM Publication GC38-1001 "OS/VS Message Library: VS1 System Messages", providing more information on the DEFINE command.

Communicating with TSIO

TSIO normally does not require intervention from the operator unless an APLSV user requests access to a date or password protected data set, or requests use of a tape or unit record device. These requests, which take the form of operator messages on the OS/VS console, should not occur unexpectedly. If they do, the OS/VS operator should reply.

```
r xx,'M'
```

to deny the requests. If, however, the APLSV user has obtained permission to update the data set or use the tape or unit record device, the operator should reply following the instructions given in the TSIO messages portion of this section.

The operator can use the OS/VS modify command with TSIO. The modify command has the form

```
f procedure-name.identifier,subcommand
```

where procedure-name.identifier is the argument given to the OS/VS start command to initiate TSIO (for example, apltsio.tsio for VS2 (apltsio.p1 for VS1) and subcommand is one of the following TSIO subcommands, which cause the action given.

USERS	Lists the APLSV sign-on identification of each current TSIO user, and the unit address of the device he is using, if any.
SUPPRESS	Prevents further users from signing on to TSIO.
SHUTDOWN	Prevents further users from signing on to TSIO, and conditions TSIO so that, when all current users have signed off, the TSIO program will terminate.
ALLOW	Returns TSIO to normal from SHUTDOWN or SUPPRESS state.
DETACH	Forces all tape or unit record devices off of TSIO. Useful only in certain emergencies, such as when an APLSV user has requested a specific standard label tape, and the tape he has supplied has an incorrect label.

The stop command, p procedure-name.identifier, can be used to abnormally terminate TSIO with a dump. TSIO is an OS/VS subsystem, and the OS/VS cancel command may not be used against it.

Examples:

```
s apltsio.pl
DQCB001 TSIO 370 NOW ACTIVE
f apltsio.pl,users

DQCB071  TSIO USER  DEVICE
         267386     360
         1234       00C
         267386     182
f apltsio.pl,shutdown
f apltsio.pl,users
DQCB061 TSIO SHUTDOWN IN PROGRESS
DQCB071 TSIO USER DEVICE
         1234       00C
```

NOTE: When TSIO is started from a batch stream in VS1, the modify and stop commands must specify the JOBNAME only.

Maximum Number of Concurrent Users

The maximum number of users that can use TSIO concurrently is determined by the region or partition size and the number of DD statements in the TSIO procedure with DDNAME beginning with WKDD. The WKDD statements impose an upper bound, and users in excess of this number must wait until some other user retracts his shared-variable interface. If sufficient WKDD statements are available, but there is insufficient available space in the TSIO region or partition for an additional user's I/O buffers, the user will receive TSIO return code 14, indicating TSIO buffers full. The TSIO run deck produced during APLSV installation provides twenty WKDD statements.

The TSIO operator functions workspace, *TSIOPS*.

The use of TSIO is controlled by functions contained in the APLSV distribution workspace *TSIOPS*. These functions must be executed at the APLSV recording terminal.

Each APLSV user who wishes to use TSIO must be
1) Enrolled in the APLSV system by the)*ADD* command.
2) Given a shared variable quota (that is, allowed to use the shared variable facility) by means of a *QUOTA* command.
3) Enrolled in TSIO by means of the TSIO operator function, *ADD*.

For example, the following sequence of commands, entered at the APLSV recording terminal, would be necessary to add user 8003 to the APLSV terminal system and allow him access to TSIO.

```
      )LOAD TSIOPS
SAVED 23.24.15 6/27/73
      )ADD 8003 RWALLEN:A20 3
      )QUOTA 8003 3 3
8003 3 3 .0 0 RWALLEN
      ADD 8003
```

There are 5 functions in the TSIOPS workspace.

ADD n,d — Adds user n to TSIO with user level d or changes his user level. (See the description of user level discriminants below.) If d is not specified, zero is assumed.

DELETE n — Removes user n from TSIO

```
      DELETE 8003 8004
8003 0
8004 14
```

DISPLAY n — Displays the user level of user n, if n has been added previously.

```
      DISPLAY 8002 9001
8002 14
```

ULIST — Produces, as an explicit result, an array, containing in its first column the sign-on identification of all users enrolled in TSIO and in its second column the level associated with each user.

```
      ULIST
8002   14
8010   0
9005   10
314159 15
```

FORMAT — See "Initialization of TSIO", below.

TSIO User Volumes

TSIO users are normally allowed to create data sets only on the specific direct access volumes whose volume serials are contained in the TSIO user volume list, which is maintained by the functions in TSIOPS as follows.

ADD 'volser' — Adds a volume serial to the user volume list, which is initially empty.

```
      ADD 'APLPAK'
```

DELETE 'volser' — Removes a volume serial from the list. Returns those remaining.

```
      DELETE 'APL222'
APL111
APLPAK
```

DISPLAY PVOLS — Returns the current user volume list as an array.

```
      DISPLAY PVOLS
APL111
APLPAK
```

User Level Discriminants

Access to data sets in the host system from APLSV terminals is restricted by means of discriminants which may be supplied to the TSIOPS function *ADD*. The discriminants, and their effect on the behavior of TSIO towards the user to whom they are applied, are as follows. More details may be found in the TSIO user's manual.

DISCRIMINANT	CODE	EFFECT
SPACE	8	present: The TSIO user can allocate direct access space on any TSIO volume, if he does not have UNIT level authorization and on any OS/VS volume, if he does.
		absent The TSIO user cannot allocate space. A SYSTEM level user (such as the operator) must allocate any data sets he is to write. If he has the ACCESS discriminant, however, he can read and possibly update, but not rewrite (with *SW*), data sets created by other, non-SYSTEM level, users.
UNIT	4	present: The TSIO user can use the TSIO command parameters *UNIT* and *VOL* to access any direct access device, and, with the operator's intervention, any unit supported by BSAM on the system.
		absent: The TSIO user is restricted to the direct access volumes in the TSIO user volume list.
ACCESS	2	present: The TSIO user can read (using the TSIO command *SR*) other TSIO users' data sets, if he knows their identification. He can also use the TSIO commands *IR* and *IRW* against other user's data sets, if they are suitably formatted.
		absent: The TSIO user cannot access any data sets but his own.
SYSTEM	1	present: The TSIO user can access any data set in the OS/VS system by explicit data set name. He may also make use of certain restricted TSIO commands and parameters.

absent: The TSIO user may not access system data sets since all data set names are prefixed with TSIO followed by an encoding of his APLSV signon identification, or, if he has ACCESS permission, the signon identification of the owner of the data set.

Level is encoded for a given user by adding together the numeric code for each discriminant and suppling this sum to the *ADD* function. For example,

ADD 8003 10

would add user 8003 to TSIO with the discriminants SPACE and ACCESS, which are codes 8 and 2 respectively.

Initialization of TSIO

When the APLSV system is first installed, the following sequence must be entered at the operator's console to initialize TSIO so that the TSIOPS *ADD* function can be used:

```
      )LOAD TSIOPS
SAVED 23.24.25  6/07/74
      )QUOTA 314159 100 10
314159 100 10 .0 5 OPERATOR
      FORMAT
WARNING. TSIO.UTABLES BEING CREATED. ENTER "." TO CONTINUE.


(enter a period)
```

The typing element will nod when the operation is complete.

FORMAT formats the TSIO.UTABLES data set.

TSIO System Error Return Codes

If TSIO encounters a system error while performing some operation on the behalf of an APLSV user, the user will receive a two-element return code: 30 x, where x is one of the integers 1 through 8. (See SH20-1463, TSIO Program Reference Manual). These values of x have the following meanings:

x	Meaning
1	VTOC FULL - The volume table of contents of the volume on which TSIO attempted to allocate a data set cannot contain any further DSCBs.
2	Allocation failed - OS/VS DASD space allocation failed because of an OS/VS error.
3	DD Card missing - There is an error in the WKDD DD card sequence in the TSIO procedure or run deck.
4	System Queue Error - OS/VS experienced an unrecoverable I/O error while accessing SYS1.JOBQE.
5	System Queue Full - There were no free records on SYS1.JOBQE when required.
6	Directory Error - During the close of a member of a partitioned data set, STOW failed, either because the directory is full or because of an I/O error.
7	CATALOG failed - The OS/VS CATALOG operation failed during a CLOSE, RENAME, or explicit CATALOG. This can be caused by an inconsistent catalog structure or by an I/O error.
8	OPEN failed - An attempted OPEN was unsuccessful. This return code will be received if the OS/VS operator replies 'm' to a request to update a date-protected data set or uses the detach subcommand in a TSIO modify command.

In most cases, a system error return code will be correlated with a OS/VS message on the system operator's console, and the action, if any, associated with that message should be taken.

PART IV.

APLSV UTILITY OPERATION

Introduction

The APLSV Utility program provides maintenance for the APLSV Library such as formatting disk packs, copying workspaces from disk to tape and vice versa, verifying disk and tape readability, and printing and punching accounting information. Through the use of these Utility operations, an installation can, for instance,

1. Provide tape or disk backup copies of APLSV library disks.

2. Retrieve individual workspaces or libraries from backup copies.

3. Reallocate the APLSV Libraries to conform to changed extent boundaries.

4. Write selected workspaces onto tape for transmission to another APL installation.

5. Bill users for APLSV service, using installation accounting routines.

Terminology

APL dump tape or tape file refers to the one or more reels of tape written by a Utility operation.

APLSV Library refers to an installation's whole collection of directories and workspaces, as stored on disk. Library or user library refers to the collection of workspaces associated with a particular account number.

Card refers to a card or a card image, depending on device type.

Directories hold information concerning the enrolled users, including their names, accounting records, passwords, and lists of saved workspaces.

Library disk or library extent refers to the portion of a disk pack described in a VTOC as being allocated for APLSV Library storage. Library extents are numbered, beginning with 0, in the order in which the volume serial numbers are identified in the APLLIB installation macro calls.

Workspace selection card refers to a card containing a library number and, optionally, a workspace name. Workspace selection cards designate particular libraries or workspaces for Utility operations. A maximum of 100 workspace selection cards may be included in a run.

Wsid refers to the library number, and name, of a workspace.

A workspace transfer operation is any operation that writes workspaces to disk or tape.

Definitions of APLSV Utility Operations

Table 1, immediately below, summarizes the APLSV Utility Operation. A detailed description of each operation follows.

Operation Name	Parameter	Optional Listing	May Run with APLSV	Disk or Tape I/O	Purpose and Remarks
ACCTG	0 1	none	yes	disk	Lists users and, optionally, workspace names
	2,x	none	yes	disk	Users and workspace names for directory x
	3	none	yes	disk	Users and workspace names for workspaces saved since last full DUMP
BILLING	none	none	no	disk	Produces billing information using installation-defined formatting routines
CREATE	0	ws names written on disk	no	both	Allows changing number of directories without formatting library data sets
	1	ws names written on disk	no	both	Special-purpose form of RESTORE for sysgen
DISKFMT	library extent number	none	no	disk	Writes full-track records on a library extent
DUMP	tape record length	ws names written on tape	no	both	Writes all directories and workspaces (wss) to tape
INCDUMP	tape record length	ws names written on tape	no	both	Writes all directories and recently-saved wss to tape
LEVEL	0	none	yes	none	APL\360 compatibility
	1	none	yes	none	APLSV compatibility.
RESET	none	none	no	disk	Reset time accounting.
RESTORE	none	ws names written on disk	no	both	Writes directories and workspaces from tape to library disk

RETRIEVE	none	ws names written on disk	no	both	Searches dump tape for selected wss and adds them to library
SELDUMP	tape record length	ws names written on tape	yes	both	Writes selected workspaces to tape
SELREST	none	ws names written on disk	no	both	Adds all workspaces on tape to library
TESTBILL	none	none	yes	disk	Like BILLING, but no accounting reset
TVERIFY	none	ws names on tape	yes	tape	Readback check of dump tape
VERIFY	library extent number	none	yes	disk	verifies the readability of a library data set
WSLIST	none	none	yes	none	Makes other operations list ws names
WSDUMP	none	ws contents on printer	no	disk	Prints contents of damaged workspace on SYSLIST in hexadecimal

TABLE 1: APLSV UTILITY OPERATIONS

Utility operations are specified by free-field control cards containing an operation name and, in some cases, a numeric parameter. Parameters may not be omitted. The operation name and parameter, if any, must be separated by at least one blank. A completely blank card is ignored. Any number of operations may be performed in a single Utility run.

The parameter for operations that write tape designates the maximum tape record length. The upper limit is 32750; the lower limit is the greater of 500 and the workspace size divided by 190.

A workspace selection card contains a library number optionally followed by one or more blanks and a workspace name. The absence of a workspace name designates the entire library. Workspace passwords must not be present. In workspace names, A-Z and 0-9 are represented by their standard EBCDIC codes, $\underline{A}$-$\underline{Z}$ are represented by lowercase EBCDIC alphabetics, and Δ and $\underline{\Delta}$ are represented by EBCDIC - and =, respectively.

Examples of workspace selection cards:

314159 OPFNS	Refers to the workspace called OPFNS in library 314159.
500	Refers to all workspaces in library 500.

A group of workspace selection cards is terminated by a card containing, as the leftmost nonblank characters, the word END.

ACCTG n -- List user information

Prints on SYSPRINT the account numbers, names, accumulated connect and cpu times, counts of saved workspaces, and counts of occupied tracks, of all enrolled users. ACCTG 0 prints one line per user. ACCTG 1 prints, in addition, the names of all workspaces in each library, including common libraries. The library data set number, cylinder and head address, and number of tracks occupied is printed with each workspace name.

ACCTG 2,x prints information of the type provided by ACCTG 1, but only for directory $\underline{x}$. ACCTG 3 prints information of the type provided by ACCTG 1, but only for those workspaces saved since the last full DUMP operation. This is equivalent to a sorted TVERIFY on a tape generated by INCDUMP at any given time, and may be useful, if stored with output, for future RETRIEVE operations.

BILLING -- Produce usage information for billing

Outputs user billing information in installation-defined format. The billing operation reads cards containing account numbers (and, optionally, other data), locates and resets accounting information stored in an APLSV directory, and provides the accounting information to an installation-defined formatting

program as described below in "Adding Installation Billing Routines to the APLSV Utility". The use of card input to the billing program allows selective billing (since a user's accumulated cpu and connect time will be neither reset nor output unless his account number appears on an input card), and it also allows optional information on input cards to be combined with the accounting output. Because the billing program buffers up to two hundred input cards in order to minimize disk accesses, billing output will not usually appear in the same order as the input cards.

CREATE n -- Create APLSV Library

When CREATE 1 is specified, formats all library data sets, builds directories, and then, reading a tape file written by a DUMP operation, distributes enrolled users among the directories and restores all workspaces to disk. Any information previously residing on the library extents is destroyed. CREATE is needed only to initialize an APLSV Library or to change the number of directories. When a change in the number of directories is all that is required, CREATE 0 is specified. This omits the formatting of the library extents performed by CREATE 1.

DISKFMT n -- Format a library extent

Writes records of the correct length throughout library extent n. DISKFMT is used when adding a new library extent, or in conjunction with DUMP and RESTORE to change file names or build backup library packs.

DUMP n -- Write all directories and workspaces to tape

Writes all APLSV Library information to tape for system backup and individual workspace retrieval, and rewrites the current time and date on disk for future incremental-dump operations. Workspaces with internal damage are flagged. The DUMP operation is referred to below as a full dump. DUMP is used in conjunction with RESTORE, and when the amount of tape needed for an INCDUMP grows beyond reasonable bounds. n denotes the maximum tape record length in bytes.

INCDUMP n -- Write directories and recently-saved workspaces to tape

Writes to tape all directories, and those workspaces which have been saved since the time of the most recent full dump. Use of incremental dumps, rather than full dumps, reduces dump time and tape storage required for system backup. n denotes the maximum tape record length in bytes.

LEVEL n -- Set system level for tape output operations

APLSV workspaces written by this system are incompatible with workspaces produced by earlier versions of APL\360. The LEVEL control card determines whether conversion to APL\360 compatible workspaces is performed before the workspaces are written to tape.

LEVEL 1 specifies that no conversion is to be performed (APLSV is level 1). LEVEL 0 specifies that a tape in APL\360 compatible form is to be produced. The conversion performed in LEVEL 0 is equivalent to typing a right arrow at an APLSV terminal until 0=ρ⎕*LC*. The most noticeable effect is that only global variables, functions, and groups remain in the workspace. If a dump taken from APLSV without a preceding LEVEL 0 card is mounted as input to a restore operation in an APL\360 system, it will be rejected with the message "NOT AN APL DUMP TAPE".

RESET -- Reset Accounting Information

All accumulated CPU and connect time charges are reset to zero.

RESTORE -- Write directories and workspaces from tape to disk

Writes to disk all information from a full-dump tape file or from an incremental-dump file and its associated full-dump file, reordering workspaces to eliminate storage fragmentation. RESTORE is used to recover from damage to library packs, to condense disk storage, and to transfer the APLSV Library to alternate packs.

When restoring from a combination of full and incremental dumps, the incremental dump tape is mounted first. The utility will restore the user accounting to the time of the incremental dump and make for each user a list of all workspace names saved at that time. It will then read first the incremental dump tape and second the fulldump tape, saving only those workspaces which appear in each user's list, so that workspaces dropped since the full dump are not added back to the user's library. If a workspace occurs on both the full and incremental dump tapes (because a new version of it has been saved since the full dump), the first copy (the one found on the incremental dump tape) is the one saved. When the restore is complete, the utility scans all user accounts for workspaces not found on either the full or incremental dump tapes, and eliminates these workspaces from the user's list, printing a message for each missing workspace.

If an entire library data set is lost, because of machine failure or other calamity, an INCDUMP should be taken, ignoring the read errors which will be produced. This INCDUMP can be supplied as the first tape of a restore operation (assuming the library data set containing the directories was not the one lost). When the utility has read this INCDUMP, it will accept the previous INCDUMP tape and then the full dump tape. In this manner, only the workspaces saved on the offending library data set between the last INCDUMP and the calamity will be lost, and a list of the workspaces lost will be obtained at the end of the run. Note that this procedure may result in the restoration of an earlier version of a user's workspace, and that all of the utility output should be preserved as reference material at least until all users affected have been informed.

RETRIEVE -- Retrieve selected workspaces from tape

Reads workspace selection cards from SYSIN, then searches a tape file (written by DUMP, INCDUMP, or SELDUMP) for the selected workspaces and writes them to disk. Up to one hundred workspace selection cards can be processed in one RETRIEVE operation; excess cards are logged on SYSPRINT and ignored. RETRIEVE terminates when all workspaces have been retrieved, or when the end of the tape file is reached. Names of workspaces or libraries not found on tape are logged on SYSPRINT. See the discussion under SELREST.

SELDUMP n -- Dump selected workspaces to tape

Reads workspace selection cards from SYSIN and writes the selected workspaces and libraries to tape. Workspaces with internal damage are flagged. SELDUMP is used to transfer workspaces from one installation to another, or to provide archival storage for users. Any number of workspace selection cards may be processed by a SELDUMP operation. n denotes the maximum tape record length in bytes. Consult the description of the LEVEL command before sending workspaces to other installations.

SELREST -- Merge workspaces from tape into an APLSV Library

Reads workspaces from a tape file (usually written by a SELDUMP operation) and incorporates them into the APLSV Library. Workspaces processed by SELREST and RETRIEVE are handled by a mechanism similar to the APLSV *)SAVE* command. A workspace with a name and number matching that of another workspace on disk replaces the previously stored workspace; a workspace with a unique name is added to the list of workspaces in the corresponding library (and may cause an increase of the workspace quota, to prevent the count of saved workspaces from exceeding the quota); a workspace whose library number does not correspond to any library number in the system is logged on SYSPRINT and rejected. The state indicators of workspaces from APL\360 are cleared.

TESTBILL -- Execute the BILLING operation to test installation-defined routines

Executes like the BILLING operation, except that accounting information in the directories is not reset to zero. See discussion under BILLING.

TVERIFY -- Verify readability of APLSV tape file

Reads directories and workspaces from tape and ignores them, except for printing error messages if any workspaces cannot be read correctly. Any single DUMP, INCDUMP, or SELDUMP tape file can be checked with a TVERIFY operation. Verification is particularly important on tapes destined for archival storage or for shipment to another installation, or if tape drive malfunction is suspected.

VERIFY n -- Verify readability of library extent

Reads all tracks within library data set n and logs incorrect-length records and data checks. VERIFY is primarily useful in checking for a damaged disk, or for accidental overwriting of an APLSV library extent. It does not check the internal validity of workspaces. Recovery procedures are explained under the RESTORE operation.

WSLIST -- Request listing of workspace names by other operations

Causes all following CREATE, DUMP, INCDUMP, RESTORE, RETRIEVE, SELDUMP, SELREST, or TVERIFY operations to print on SYSPRINT the time and date of saving, and library number and name, of each workspace written to disk or tape.

WSDUMP -- Print contents of damaged workspace

Inclusion of a WSDUMP card preceding a CREATE, RESTORE, RETRIEVE, TVERIFY, DUMP, INCDUMP, or SELDUMP operation causes the contents of any damaged workspaces to be printed in hexadecimal on SYSPRINT.

Executing APLSV Utility Operations

Storage Requirements The region or partition size required for APLSV Utility operation is approximately 35000 bytes, plus twice the workspace size, plus space occupied by installation-defined billing programs. If additional storage equivalent to the workspace size is available, I/O overlap will allow workspace-transfer operations to be executed more rapidly.

Run Deck Setup The JCL used to execute the APLSV Utility is generated and punched by the RUN UTILITY macro during the installation of APLSV.

Notes:

1. The APLSV swap files are not used by the Utility.

2. A SYSPUNCH DD statement is required only when BILLING output is to be produced. The default is BLKSIZE=80,RECFM=FB,LRECL=80.

3. A SYSPRINT DD statement is required for all utility operations. Its default is BLKSIZE=131,RECFM=VBA,LRECL=125. The BLKSIZE of SYSPUNCH and SYSPRINT may be respecified on the DD statement.

4. The JOBNAME field must begin with the characters specified in the APL JOBNAME macro during APLSV installation.

Tape Mounting The operating system will normally allocate tape units and request that volumes be mounted during the allocation phase of initiation, and will issue a MOUNT message whenever a new volume is required on a drive. If the order or number of volumes to be mounted is not known, or if only one tape unit is available, the tape DD statements should specify deferred mounting. If tapes with standard volume labels are to be used, the TAPE1 and TAPE2 statements must specify LABEL=(1,BLP) and an OS/VS reader which permits the use of BLP must be used to read the utility JOB.

Note that the Utility does not support OS/VS standard tape label processing - it simply preserves the VOL1 Label.

Performing a DUMP, INCDUMP, or SELDUMP Execute the Utility with the appropriate control card. Follow a SELDUMP control card by workspace selection cards and an END card.

TAPE2 should never be DUMMY. If only one tape drive is available, make the VOL=SER= the same for TAPE1 and TAPE2.

Performing a CREATE, RETRIEVE, SELREST, or TVERIFY Mount an APLSV dump tape and execute the Utility with the appropriate control card. Follow a RETRIEVE control card by up to one hundred workspace selection cards and an END card. The descriptions of individual operations indicate the kinds of dump tapes accepted by each operation.

Performing a RESTORE Execute the Utility with a RESTORE control card. The tape mounting sequence depends on the kinds of tapes involved. To restore libraries as of the time of a full dump, mount the full-dump tape file. To restore libraries as of the time of an incremental dump, first mount the incremental-dump tape file. When the incremental-dump file has been processed, the Utility will request mounting of the associated full-dump tape file to complete the RESTORE process.

Performing a BILLING or TESTBILL Mount any tapes or disks needed by installation-written billing routines. Execute the Utility with a BILLING or TESTBILL control card followed by billing input cards for the users being billed, followed by an END card.

Performing other Utility Operations Include the appropriate control cards in the Utility job deck.

Reconfiguring APLSV System generation procedures, including designation of the number of directories and of library extents, may be found in PART I. The steps below must be performed if changes are being made to the number of directories or number of library extents.

1. Dump the APLSV Libraries to tape using the DUMP control card (see, however, 4a).

2. Make any necessary changes in the APLSV Installation and re-install APLSV as decribed in PART I. (The OS/VS nucleus need not be re-linkedited unless the type 1 SVC number is to be changed.)

3. If APLSV Library extents are being added or changed, or new disk packs are to be used for the APLSV Library, use the IEHPROGM utility program to allocate each new or changed extent.

4a. If the number of directories has not been changed, perform a DISKFMT operation on each new or changed extent, and RESTORE the APLSV Library from the tape written in step 1. In this case, step 1 can use the INCDUMP control card, and the INCDUMP tape created there, followed by its preceding DUMP tape, can be used as input to the RESTORE.

4b. If the number of directories has been changed, CREATE the APLSV Library from the tape written in step 1, using the CREATE 0 option to avoid reformatting all packs. In this case, an INCDUMP tape may not precede the DUMP tape.

Utility Operation Notes

It is possible, on a machine with sufficient storage, to execute certain Utility operations concurrently with APLSV. Table 1 indicates the operations that are permitted while APLSV is running.

The utility depends on the OS time and date for proper operation. Make sure that time and date settings are consistent with those set for APLSV operation.

The Utility preserves, but disregards, workspace and user number passwords. Knowledge of a workspace password is not required to copy it from disk to tape or vice versa.

Reasonable backup for the APLSV Library can be provided by taking a full dump at infrequent intervals and an incremental dump daily. A daily full dump provides no additional backup, so (after the first full dump) the appropriate time to take a full dump is when the incremental dump output approaches a full tape reel. To provide extensive backup, dump tapes should be cycled as follows: when all allocated tapes have been used for dumps, select for re-use every second reel (or group of reels, in case of a

multi-reel file) in chronological order, starting with the second oldest. Then, when these reels are used up, repeat this selection process on the entire set of reels. With two or three dozen reels, this procedure will give adequate backup over the life of a typical system, with the recent past covered most heavily. Note that an orphaned incremental-dump tape (one whose full-dump tape has been re-used) is still useful for retrieving individual workspaces.

Workspaces may be transferred by tape to a system with a different workspace size, as long as the space actually occupied by a workspace is not greater than the maximum size configured for the receiving system. The size conversion is done automatically when the Utility reads the workspace from tape. Workspaces too large to fit in the system being restored to will be logged and ignored.

A TVERIFY operation should be performed on any tape file intended for long-term backup or for shipment to another installation, and on any tape for which the OS tape error statistics indicate an unusual number of write errors.

Users tend to rely on the cumulative CPU and connect times APLSV prints at sign-off for an informal record of their level of APLSV use. These figures are considerably less useful if billing operations, which reset time records, are performed frequently. An ACCTG operation performed daily can provide an auditable record of usage; and the recording terminal's output holds a record of every sign-on and sign-off, though not of CPU time. CPU and connect time records for the session are updated each time a user saves a workspace in his own library or signs off.

A full or incremental dump operation should follow rather than precede billing, so that any subsequent RESTORE will not also restore accounting information for which the user has already been billed. Depending on local accounting practice, an ACCTG 0 before and after the billing may be advisable.

Tape files written by DOS versions of the APL\360 Utility distributed before July 1969 cannot be read by the APLSV Utility. APL\360 Libraries on disk packs used by earlier versions of APLSV cannot be read or written correctly by the current APLSV System or by the current Utility.

The SYSPRINT output from a Utility run holds a complete record of program output, and of SYSIN and operator input. Retain this output for installation records, and for analysis by system programmers.

When workspaces are being transmitted to another installation, it is good practice to do a WSLIST operation before the SELDUMP operation, and to send the SYSPRINT output along with the tape. Be sure to use a LEVEL 0 card before the SELDUMP card. A TVERIFY operation should also be performed on the tape to be sure it can be read.

Adding Installation Billing Routines to the APLSV Utility

This section sets forth the rules for writing and adding billing routines to APLSV. Billing routines may be added during system generation or at any later time; they need not be added at all if there is to be no billing of users.

The Utility BILLING or TESTBILL operation reads card images containing account numbers (and, optionally, other data), locates and resets accounting information stored in an APL directory, and provides the accounting information to an installation-written formatting program. The use of card image input allows selective billing (since a user's accumulated CPU and connect time will be neither reset not output unless his account number is on an input card), and it also allows optional information on input cards to be combined with the accounting output. Because the Utility buffers up to 200 input cards in order to minimize disk accesses, billing output will not usually appear in the same order as the input cards.

The Utility performs card image reading and buffering, directory reads, searches, and writes, and card punching. The routines to be written by the installation for input and output formatting are named APLUBILN and APLUBILF, and must be link-edited with the Utility. General registers 2-12 must be preserved by APLUBILN and APLUBILF.

Installation-written Routine Descriptions

APLUBILN

Purpose: To scan an input card for an account number, and return it as a fullword integer.

On entry, R1 points to a two-word parameter list. The first word is the address of an 80-character EBCDIC card image, and the second word is the address of a fullword in which to store the converted account number. A zero value returned for the account number causes the Utility to ignore the card, print an error message, and continue processing. A negative value causes the Utility to ignore the card with no error message. APLUBILN may modify the card image. The Utility calls APLUBILN once for each input card.

APLUBILF

Purpose: To prepare billing information for output, and optionally to output it.

On entry, R1 points to a two-word parameter list. The first word is the address of the same 80-character EBCDIC card image passed to APLUBILN, including any modifications made by APLUBILN. The second word is the address of a block of billing information defined by this DSECT:

```
BILINFO   DSECT              SECOND PARAM FROM BILLING
CONN      DS      F          CONNECTED TIME IN SEC/300
CPU       DS      F          COMPUTE TIME IN SEC/300
BILNAME   DS      CL12       SIGN ON NAME IN INTERNAL CODE
BILWSQ    DS      H          WORKSPACE QUOTA
BILWSA    DS      H          ACTUAL NUMBER OF WORKSPACES SAVED
WSTRACKS  DS      H          TOTAL TRACKS FOR ALL WS
          DS      5H         RESERVED
```

The track count in WSTRACKS includes tracks occupied by any workspaces saved by this user in common libraries. APLUBILF is called once for each billing input card that was not rejected by APLUBILN or by the BILLING program.

APLUBILF may prepare one or more output records or none. If output records are being punched, punching must be done by calling the subroutine APLUBILP with R1 pointing to a parameter list whose single entry is the address of the 80-character EBCDIC output record. APLUBILP outputs to SYSPUNCH.

Following the last billing output, APLUBILL calls APLUBILF one additional time with all fields of the second parameter set to -1. If output is done in the APLUBILF routine, this call may serve as a signal for APLUBILF to close any output files. It also provides an opportunity to output summary totals. As usual, APLUBILF must return to its caller.

Including APLUBILN and APLUBILF in the APLSV Utility

1. Assemble or compile APLUBILN and APLUBILF, preferably as a single CSECT.

2. Link-edit the APL Utility, including APLUBILN and APLUBILF routines.

3. Before performing a BILLING operation, make sure that APLUBILN and APLUBILF are debugged, by testing them using the TESTBILL operation. Because of the buffering done by the Utility, a program check during a BILLING execution is very likely to produce billing output without resetting system accounting information.

Sample Installation Routine The following pages show a sample COBOL program for APLUBILN and APLUBILF.

Sample COBOL

```
000100 IDENTIFICATION DIVISION.
000200   PROGRAM-ID.  APLUINST.
000300   REMARKS. INSTALLATION-DEFINED ROUTINES FOR APL BILLING.
000400     BILLING-OUTPUT-CARD AND BILLING-INPUT-CARD HAVE
000500     INSTALLATION-DEFINED FORMATS.  FORMATTING IS DONE BY APLUBILN
000600     AND APLUBILF.
000700 ENVIRONMENT DIVISION.
000800 DATA DIVISION.
000900   WORKING-STORAGE SECTION.
001000   01 BILLING-OUTPUT-CARD.
001100     02 FILLER PICTURE X, VALUE '0'.
001200     02 ACCOUNT-NO PICTURE X(11).
001300     02 FILLER PICTURE X, VALUE SPACE.
001400     02 PROJ-NO PICTURE X(4).
001500     02 FILLER PICTURE X(7), VALUE SPACE.
001600     02 DEPT-NO PICTURE X(3).
001700     02 NAME PICTURE X(15).
001800     02 TIME PICTURE 9(8).
001900     02 FILLER PICTURE X, VALUE '*'.
002000     02 CARDCODE PICTURE X.
002100     02 FILLER PICTURE X(28), VALUE SPACE.
002200   01 WORK-ACCOUNT PICTURE X(12), VALUE SPACE.
002300   01 CONCHG PICTURE S9(9) COMPUTATIONAL.
002400   01 CPUCHG PICTURE S9(9) COMPUTATIONAL.
002500   LINKAGE SECTION.
002600   01 BILLING-INPUT-CARD.
002700     02 ACCOUNT-NO PICTURE 9(12).
002800     02 FILLER PICTURE X(68).
002900   01 RETURN-ACCOUNT-NO PICTURE S9(9) COMPUTATIONAL.
003000   01 BILLING-INFO-CARD.
003100     02 FILLER PICTURE X.
003200     02 ACCOUNT-NO PICTURE X(11).
003300     02 FILLER PICTURE X.
003400     02 PROJ-NO PICTURE X(4).
003500     02 FILLER PICTURE X.
003600     02 DEPT-NO PICTURE X(3).
003700     02 FILLER PICTURE X.
003800     02 NAME PICTURE X(15).
003900     02 FILLER PICTURE X(42).
004000   01 BILLING-INFO-CHARGE.
004100     02 CONNECT PICTURE S9(9) COMPUTATIONAL.
004200     02 CPU PICTURE S9(9) COMPUTATIONAL.
004300     02 FILLER PICTURE X(12).
004400     02 WS-QUOTA PICTURE S9(4) COMPUTATIONAL.
004500     02 WS-ACTUAL PICTURE S9(4) COMPUTATIONAL.
004600     02 WS-BLOCKS PICTURE S9(4) COMPUTATIONAL.
004700     02 FILLER PICTURE X(10).
```

```
004800 PROCEDURE DIVISION.
004900     ENTRY 'APLUBILN' USING BILLING-INPUT-CARD RETURN-ACCOUNT-NO.
005000     MOVE ACCOUNT-NO IN BILLING-INPUT-CARD TO WORK-ACCOUNT.
005100     MOVE 0 TO RETURN-ACCOUNT-NO.
005200     EXAMINE WORK-ACCOUNT REPLACING LEADING SPACES BY ZEROES.
005300     IF WORK-ACCOUNT IS NUMERIC THEN MOVE ACCOUNT-NO IN
005400     BILLING-INPUT-CARD TO RETURN-ACCOUNT-NO ELSE MOVE -1 TO
005500     RETURN-ACCOUNT-NO.
005600     GOBACK.
005700     ENTRY 'APLUBILF' USING BILLING-INFO-CARD,
005800     BILLING-INFO-CHARGE.
005900     IF CPUCHG NEGATIVE GO TO NO-OUTPUT.
006000*    NOTE. WOULD CLOSE FILE INSTEAD HERE IF GOING TO TAPE OUTPUT.
006100       IF CONNECT ZERO GO TO NO-OUTPUT.
006200     DIVIDE 13000 INTO CONNECT GIVING CONCHG ROUNDED.
006300     DIVIDE 300 INTO CPU GIVING CPUCHG ROUNDED.
006400     IF CONCHG ZERO MOVE 1 TO CONCHG.
006500     IF CPUCHG ZERO MOVE 1 TO CPU.
006600     MOVE ACCOUNT-NO IN BILLING-INFO-CARD TO ACCOUNT-NO IN
006700       BILLING-OUTPUT-CARD.
006800     MOVE PROJ-NO IN BILLING-INFO-CARD TO PROJ-NO IN
006900       BILLING-OUTPUT-CARD.
007000     MOVE DEPT-NO IN BILLING-INFO-CARD TO DEPT-NO IN
007100       BILLING-OUTPUT-CARD.
007200     MOVE NAME IN BILLING-INFO-CARD TO NAME IN BILLING-OUTPUT-CARD
007300     MOVE '>' TO CARDCODE IN BILLING-OUTPUT-CARD.
007400     MOVE CONCHG TO TIME.
007500     CALL 'APLUBILP' USING BILLING-OUTPUT-CARD.
007600     MOVE '=' TO CARDCODE IN BILLING-OUTPUT-CARD.
007700     MOVE CPUCHG TO TIME.
007800     CALL 'APLUBILP' USING BILLING-OUTPUT-CARD.
007900 NO-OUTPUT.
008000     GOBACK.
```

PART V

INSTALLATION-WRITTEN AUXILIARY PROCESSORS

Introduction

An auxiliary processor is a program not written in APL which performs a service for APLSV users. The Shared Variable Processor, SVP, provides an interface from the APLSV system to such programs. This section is a guide to their implementation.

All macros and copy code required for writing auxiliary processors are in APLSVS.MACLIB, a partitioned data set contained in the APLSV optional machine-readable material. Familiarity with APLSV as described in the APLSV User's Manual, and with Assembly Language Programming in OS, is assumed. TSIO, primarily the main module, DQCVBTO, can be assembled as an example. It will be useful to print APLSVS.MACLIB(SVDEFN) for reference while reading the discussion below.

NOTE: An auxiliary processor is a system program. Safeguards have been built into SVP and into the Type 1 SVC to reduce the likelihood of an APLSV crash caused by an error in an auxiliary processor, but they are not foolproof. Extreme care should be taken if an auxiliary processor is to be tested while APLSV is providing regular service to users.

Control Blocks

The Shared-Variable Processor requires the use of two control blocks: a Processor Control Vector (PCV), and a Share Control Vector (SCV).

They are defined as follows:

Processor Control Vector

```
PCV       DSECT
PCVID     DS    2F                  PROCESSOR IDENTIFICATION
PCVECB    DS    A                   ADDRESS OF AN ECB
PCVTRAP   DS    A                   ADDRESS OF TRAP ROUTINE
```

By convention, the identification of an auxiliary processor, PCVID, consists of an integer between 0 and 1000 followed by an integer 0. For example, the identification of TSIO is defined as

```
          DC    F'370,0'            PCVID
```

PCVECB must contain the address of an Event Control Block in the auxiliary processor. This ECB will be posted when events relevant to the auxiliary processor occur.

PCVTRAP must contain the address of a routine in the auxiliary processor. The function of this routine is described below with the discussion of data transfer.

Share Control Vector

```
SCV       DSECT
SCVID     DS    2F          PROCESSOR IDENTIFICATION
SCVALUE   DS    A           ADDRESS OF VALUE DESCRIPTOR
SCVNO     DS    F           OFFER SEQUENCE NUMBER
SCVPSX    DS    F           PERSHARE INDEX
SCVOTHER  DS    2F          INTENDED SHARER'S IDENTIFICATION
SCVNAMEL  DS    C           NUMBER OF CHARACTERS IN NAME
SCVNAME   DS    0C          FIRST CHARACTER OF NAME
```

The meaning of these fields is described below with the discussion of SCAN and SHARE.

Return Codes

At the completion of any request to SVP, three separate results are returned.

1. The condition code is set:
   2 - indicates normal completion
   0 - indicates an error.
   This is tested as follows:

```
   PREREF PSX                EXAMPLE SVP REQUEST
   BE     ERROR              BRANCH ON ERROR
```

2. The ECB whose address was passed to SVP (see SIGNON) contains a return code. A complete list of return codes may be found in APLSVS.MACLIB(SVXDEFN).

   A return code of 0 or 2 indicates that the SVP request was rejected because of a temporary block. The condition code is 0 and the PCVTRAP routine is entered.

   A return code of 4 or 8 indicates that the operation completed. The condition code is 2.

   A return code greater than 8 indicates that the SVP request failed. Details of particular return codes are discussed with each request, below.

3. The request can have an explicit result which will be returned either in a general register or in storage, depending on the request.

Establishing a Connection with SVP

The relationship between an auxiliary processor and SVP is analogous to the relationship between an APL user and APLSV. A session begins with a sign-on procedure, consists of requests for services, and terminates with a sign-off.

Before signing on, the auxiliary processor must establish a valid STAE; the STAE must sign the processor off before terminating. The minimal STAE exit routine consists of the following:

```
TROUBLE  SIGNOFF                  SIGN OFF BEFORE STOPPING
         SR     15,15             INDICATE NO RESTART
         BR     14                RETURN TO OS
```

The STAE routine is enable by:

```
STAE  TROUBLE
```

A processor which terminates without signing off can cause two kinds of problems:

1. The processor id will remain in the SVP processor table, and it will be impossible to sign on another processor with the same id. When this situation occurs, the function *SVCLEAR* in *OPFNS* can be used to clear the table.

2. If the processor ABENDS when an APLSV user is actively sharing a variable with it, an attempt may be made to POST an RB which is no longer there, possibly causing an APLSV ABEND.

A special monitor program for testing auxiliary processors is supplied with the APLSV system. It should be used until the experimental auxiliary processor has been fully debugged to ensure that the processor is cleared from the system when it terminates. The APLSV operator function *SVCLEAR* should be used whenever an experimental processor terminates. It will report either that the processor signed off successfully, or that it has cleared the variable entry.

The connection with SVP is established by a SIGNON request which passes an appropriate PCV to SVP.

Suppose that the PCV for processor 666 is defined as follows:

```
MYPCV    DS    0A
         DC    F'666,0'          PROCESSOR ID 666
         DC    A(MYECB)          PCVECB ADDRESS
         DC    A(MYTRAP)         PCVTRAP ROUTINE ADDRESS
```

Then the signon request would be:

```
GO       SIGNON MYPCV            SIGN ON REQUEST
         BE     FAILED
*        CONTINUE PROCESSING
```

The branch following the SIGNON macro will be taken when the condition code has been set to zero because the request was rejected. The reason for the rejection will be found in MYECB unless the PCV was found to be invalid.

Equate statements for all possible return codes are found in APLSVS.MACLIB (SVXDEFN). The possible return codes from SIGNON and their meanings are:

SVZNE Normal end, *CC*=2, branch not taken.

SVZNIU Number in use: the identification contained in PCVID is already in use by another processor, or was in use by a processor which terminated without signing off.

SVZASO Already signed on: The auxiliary processor issuing the SIGNON request is already signed on with the PCVID given.

SVZNA SVP not active. This is returned by the type 1 SVC when SVP is not running. If the processor has been started before APLSV, but it is expected that APLSV will be started; the processor can delay for a few seconds (using, for example, STIMER WAIT), and then reissue the SIGNON request.

SVZPPF Processor table full. The maximum possible number of concurrent auxiliary processors is already signed on to SVP.

SVZARG Argument error: The PCV was not correctly formed or was not aligned on a word boundary.

Once a SIGNON request has been successfully completed, the PCV is no longer required.

Establishing a Shared Variable

A shared variable is established by a SHARE request, which requires a valid SCV as an argument. The information contained in the SCV can be obtained in two ways:

1. If the auxiliary processor is to wait until a variable is offered to it (TSIO operates this way), the information is obtained by a SCAN request; which also takes an SCV as its argument.

2. If the auxiliary processor is to extend an offer and then wait for a matching offer from another user, it defines the SCV itself.

Scanning for offers Before executing a SCAN to determine whether any variables have been offered to the processor id used in the SIGNON, the SCV is first initialized as follows:

SCVID must contain the id which was contained in PCVID at SIGNON.

SCVALUE ignored.

SCVNO SVP numbers each offer it processes sequentially. An auxiliary processor may wish to examine all offers made to it, but only accept certain ones. This field, which is set to binary zero by the auxiliary processor when it first begins examining offers, is set by SVP to the sequence number of the offer currently in the SCV, whenever the SCAN operation completes successfully. It is normally left unaltered so that the next SCAN issued will only consider offers with higher sequence numbers. If SCVNO is set to zero, all offers will be examined.

SCVPSX ignored.

SCVOTHER must contain the processor id of the intended sharer. If the auxiliary processor is to accept offers from any user, SCVOTHER must contain two zeros.

SCVNAMEL must contain a single-byte value which indicates the number of characters available to hold the name of the shared variable. If a specific variable name is used by the processor, SCVNAMEL must contain the number of characters in the name.

SCVNAME must reserve at least as many bytes of storage as the value in SCVNAMEL. If any name is acceptable to the auxiliary processor, then the first character of SCVNAME must be binary zero. If the processor requires a specific name, the name must be formed from the internal APLSV alphabet (see APLSVS.MACLIB(ZSYMBOLS)).

Illustrations:

Define MYSCV as follows:

```
MYSCV    DS    8F                    SHARE/SCAN CONTROL VECTOR
```

The SCV might be initialized as follows to begin scanning for an offer of any name for any user:

```
         USING SCV,8
         LA    8,MYSCV
         MVC   SCVID(8),=F'666,0'  MY PROCESSOR ID
         MVC   SCVNO(4),=F'0'      START SEARCH FROM OFFER ZERO
         MVC   SCVOTHER(8),=F'0,0' ACCEPT ALL USERS
         MVI   SCVNAMEL,7          MAXIMUM 7 CHARS IN NAME
         MVI   SCVNAME,0           ACCEPT ANY NAME
         DROP  8
```

Each successive SCAN would only clear the fields SCVOTHER and SCVNAME, and refresh SCVNAMEL.

The SCV might be initialized as follows to make a specific offer:

```
         USING SCV,8
         LA    8,MYSCV
         MVC   SCVID(8),=F'666,0'  MY PROCESSOR ID
         MVC   SCVNO,=F'0'         ACCEPT ANY OCCURRENCE
         MVC   SCVOTHER,=F'314159,0' SHARE WITH OPR ONLY
         MVI   SCVNAMEL,3          THREE-CHARACTER NAME
         MVC   SCVNAME(3),=AL1(ZV,ZA,ZR) NAME IS 'VAR'
         DROP  8
```

The SCAN is then executed:

```
        SCAN   MYSCV                SEARCH FOR OFFER
        BC     2,FOUNDONE           BRANCH IF OFFER FOUND
        CLI    MYECB+3,SVZNOF       CHECK FOR NONE FOUND
        BE     WAIT                 IF SO, BRANCH TO WAIT
*       AN ERROR OCCURRED
```

The SCV is unchanged if the condition code is 0. If an appropriate offer was found, the return code is set to SVZNE, the condition code is 2, and the SCV is completed as follows:

SCVID unchanged.
SCVALUE unchanged.
SCVNO contains the offer number of the offer found.
SCVPSX contains the PERSHARE index of the offer found (see discussion of SHARE, below).
SCVOTHER contains the id of the processor which made the offer.
SCVNAMEL contains the actual number of characters in the variable name.
SCVNAME contains the variable name, or a truncated version if the number of characters is greater than the initial value of SCVNAMEL.

Except in the case where the shared-variable name returned in SCVNAME has been truncated because it consisted of more characters than the initial value of SCVNAMEL, the completed SCV is an appropriate argument for SHARE.

Offering a shared variable Before offering to share a variable, the SCV must be properly initialized as follows:

SCVID must contain the id contained in the PCV at sign-on.
SCVALUE must contain the address of a value descriptor (described later), or must be -1.
SCVNO ignored.
SCVPSX must be zero if all offers are to be searched. If non-zero, it must be the pershare index of a matching offer, such as is returned by SCAN.
SCVOTHER must contain the processor id of the intended sharer. If any sharer is acceptable, it must be zero.
SCVNAMEL and *SCVNAME* must contain a valid shared-variable name, i.e., SCVNAMEL contains the number of characters and SCVNAME contains the SCVNAMEL characters, in the internal APL alphabet, of the name.

There are four results from SHARE:

1. The condition code is set to 2 (normal end) or 0 (error).

2. The return code is set in the ECB whose address was passed in PCVECB.

3. General register 0 contains an explicit result.

   0 indicates failure due to argument error.

   1 indicates that an offer to another processor was extended because no matching offer was found.

   2 indicates that a matching offer was found and that the connection has been established.

4. When the SHARE completes normally, and the explicit result in GR0 is 1 or 2, the SCV is completed as follows:

SCVNO contains the number of the shared-variable offer which has just been established or accepted.
SCVPSX contains the PERSHARE index of the shared-variable.

NOTE: The PERSHARE index must be saved, as it is required to identify the shared variable in all subsequent operations.

The possible return codes from SCAN and SHARE and their meanings are:

SVZNE the request completed normally (*CC*=2). The SCV and GPO only contain valid results from SVP when this return code is obtained.
SVZNSO The processor id contained in the SCV is not signed on to SVP.

NOTE: In the case of an auxiliary processor using SVP through the OS type-1 SVC, it is not possible to return this code to the auxiliary processor, since the location of the ECB is not known. To detect this case, initialize the ECB before issuing a request, and then test for a change when *CC*=0.

SVZSPE storage protection: The value descriptor addressed by SCVALUE did not describe a value contained in the caller's region or partition.
SVZNA SVP not available.
SVZPSF The shared-variable table is full, hence an offer cannot be extended.
SVZARG An error was detected in the SVC or in the value descriptor.
SVZNOF After SCAN, indicates that no matching offer was found.

Illustration:

Referring to the illustration of SCAN, above, an offer might be found and accepted as follows:

```
         SCAN  MYSCV                SEARCH FOR OFFER
         BC    2,FOUNDONE           BRANCH IF OFFER FOUND
         .
         .
         .

FOUNDONE SHARE MYSCV                ACCEPT OFFER
         BZ    FAIL                 BRANCH IF CC NOT 2
         C     0,=F'1'
         BL    FAIL                 BRANCH IF RESULT IS 0
         BE    OFFERED              BRANCH IF OFFER MADE
*        FALL THROUGH IF CONNECTION ESTABLISHED
         L     2,SCVPSX-SCV(1)      LOAD RETURNED PSX
         ST    2,MYPSX              SAVE FOR LATER USE
```

Although the SCV used as an argument to SHARE, above, was completed by the SCAN because an offer was found, the explicit result must be examined because the offerer may have retracted the offer between the SCAN and the SHARE.

A specific offer is made much in the same way, using an SCV which has been initialized as shown above for a SCAN for a specific offer. An additional requirement for SHARE is the SCVALUE field, which must be valid or be -1.

Terminating a Connection

The offer to share a variable is withdrawn, whether or not it is connected to another sharer, by a request to RETRACT. The argument for retract is a PERSHARE index obtained from a SHARE operation. Explicit results are returned in general registers 0 and 1 as follows:

Register 0:

0 indicates that the PSX did not denote a variable shared by the calling processor.

1 indicates that the variable was offered but was not connected to another processor. The shared variable has been deleted from Shared Variable Storage.

2 indicates that the shared variable was connected to another processor. At the completion of the retract, this variable remains in shared variable storage as a specific offer to the processor which retracted.

Register 1: When register 0 contains 2, contains the sequence number of the specific offer to the retracting processor.

Terminating a Session

Before an auxiliary processor terminates execution, it must notify SVP as follows:

```
SIGNOFF PCV                    or
SIGNOFF
```

When the Processor Control Vector which was used at sign-on is used as an argument, only the processor specified by PCVID is terminated. This is desirable when a particular auxiliary processor signs on with many identifications (e.g., APLSV). SIGNOFF with no argument causes all processors signed on by the program (determined by the JOB STEP TCB) to be signed off.

The condition code is set to 2 if the sign-off was successfully completed, or to 0 if the processor was not signed on.

Access Control

An access control vector (ACV) is associated in shared-variable storage with each shared variable. For a detailed description, see the APLSV User's Manual. The ACV may be set by either of the processors sharing a variable, and a temporary combined vector calculated by oring together the two settings is used when SVP determines whether an access is permissible.

The access control vector may be set:

```
ACV   MYPSX,B'1111'        SET ALL INHIBITIONS
```

The current combined ACV can be examined by:

```
ACV   PSX
```

At the completion (*CC*=2) of an ACV request, the current combined ACV is contained in general register 0.

Data Transfer

The shared variable processor appears to its users much like a storage box. Values of shared variables are set or read by a sequence of operations; an initial selection followed by the data transfer.

Initial Selection Initial selection is accomplished by issuing a PREREF, indicating a desire to read the value, or a PRESPEC, indicating a desire to write. Both of these operations require a PERSHARE index, returned by SHARE in SCVPSX, as an argument. SVP examines the Access Control Vector and the current state of the shared variable, and returns in one of three ways:

1. If the shared variable cannot be accessed, either because the sharing processor has control (i.e., has completed an initial selection, but has not completed the data transfer), the return code is set to SVZLOCK, register 15 is set to the address of the SVP request, and the PCVTRAP routine is entered.

2. If an error in the request is detected, an appropriate code is set in the ECB whose address was passed in PCVECB at sign-on, and the condition code is set to zero.

3. If the request is permissible, the shared variable is flagged as under the control of the caller, the ECB return code is set to SVZNE or SVZNOS (if no other processor is sharing the variable), and the condition code is zero.

At the successful completion of an initial selection, register 0 contains -1 if there is no value currently in shared-variable storage, or the size of the value in bytes. It is important to note that when a PREREF is successfully executed and GR0 contains a value other than -1, then completing the write operation will overwrite a value. The auxiliary processor can choose to abandon the write at this point in order to read the value (For an example, see TSIO).

Illustrations:

1.
```
READ    PREREF MYPSX            ATTEMPT TO GAIN CONTROL
        BZ     FAILURE          BRANCH IF ERROR
        C      0,=F'0'          CHECK LENGTH OF VALUE
        BL     NOVAL            BRANCH IF NONE
```

2.
```
WRITE   PRESPEC MYPSX           ATTEMPT TO GAIN CONTROL
        BZ     FAILURE          BRANCH IF ERROR
        C      0,=F'0'          CHECK FOR A VALUE
        BNL    WRITE2           GO COMPLETE WRITE IF NOT
        RELEASE MYPSX           OTHERWISE, RELEASE CONTROL
        B      READ             AND GO READ THE VALUE
```

The possible return codes after an error in initial selection (*CC*=0) are:

SVZNSO Processor not signed on (see note under SHARE).
SVZIVS Invalid sequence - the auxiliary processor already has control of the shared variable from a previous initial selection.
SVZNV PREREF - no value available.
SVZARG Argument error - the PERSHARE index does not denote a variable shared by the calling processor.

PCVTRAP Routines When an initial selection operation cannot be completed because the shared variable is under the control of the sharing processor or because the Access Control Vector inhibits the particular kind of request, the APLSV TYPE 1 SVC forces a return to the address passed in PCVTRAP at sign-on rather than to the instruction following the request. Register 15 contains the address of the SVC instruction generated by the PREPEF or PRESPEC macro. The ECB whose address was passed in PCVECB at sign-on will be posted whenever any shared-variable request by another processor affects this auxiliary processor, and the PCVTRAP routine can wait on this ECB.

Illustration:

```
MYTRAP   STM    15,1,MYSAVE          SAVE REGISTERS USED IN WAIT
         WAIT   ECB=MYECB            WAIT UNTIL SVP POSTS
         NI     MYECB,255-ECBPOST    CLEAR POST BIT
         LM     15,1,MYSAVE          RECOVER REGISTERS
         BR     15                    RE-ISSUE REQUEST
```

In most cases, like TSIO, it will wish to perform an operation for another user. The TRAP routine then becomes a scheduler. In general, this will take the form

```
MYTRAP   STM    0,15,REGSV(8)        SAVE REGS
         LA     8,PTABLEN(,8)        GET NEXT VARIABLE
         C      8,LASTVAR
         BH     RUN
         WAIT   ECB=MYECB            WAIT FOR ANY ACTION
         NI     MYECB,255-POSTBIT    TURN OFF POST BT
         L      8,FIRSTVAR           START WITH FIRST VARB

RUN      LM     0,15,REGSV(8)        RELOAD REGS
         BR     15                   GO TO INTERRUPT POINT
```

Data Transfer Once control of a shared variable has been obtained by a successful initial selection operation, the data transfer may take place. The value is described by a two-word value descriptor: the first word contains the address of the data area; the second contains the length, in bytes, of the value. If the value is to be passed to APLSV, it must be in the proper internal form, as follows:

```
MYTPE    DS     FL1                  DATA TYPE
*        1 - LOGICAL, 1 BIT PER ELEMENT
*        2 - INTEGER,4 BYTES PER ELEMENT
*        3 - FLOATING, 8 BYTES PER ELEMENT
*        4 - CHARACTER, 1 BYTE PER ELEMENT
         DS     FL1                  MUST BE ZERO
MRANK    DS     H                    4 TIMES NUMBER OF DIMENSIONS
MRHO     DS     F                    FIRST DIMENSION
```

The actual data elements begin following MRHO plus the number of bytes contained in MRANK. The elements of MRHO are fullword integers which denote the length along each dimension, with the columns last.

Illustrations:

1. Integer Matrix with 2 rows and 3 columns.

```
VALUE    DC     A(LOC,36)            VALUE DESCRIPTOR
LOC      DC     FL1'2'               INTEGER TYPE
         DC     FL1'0'
         DC     H'8'                 MRANK FOR MATRIX
         DC     F'2'                 MRHO - 2 ROWS
         DC     F'3'                 AND 3 COLUMNS
         DC     F'1,2,3,4,5,6'       DATA
```

2. Character vector, 8 elements

```
VALUE    DC    A(LOC,16)              VALUE DESCRIPTOR
LOC      DC    FL1'4'                 CHARACTER TYPE
         DC    FL1'0'
         DC    H'4'                   MRANK FOR VECTOR
         DC    F'8'                   MRHO - 8 ELEMENTS
         DC    AL1(ZA,ZB,ZC, ...      DATA  (ZSYMBOLS)
```

3. Logical scalar (value=1)

```
VALUE    DC    A(LOC,8)
LOC      DC    FL1'1'                 LOGICAL TYPE
         DC    FL1'0'
         DC    H'0'                   MRANK FOR SCALAR
         DC    X'80'                  VALUE=1
```

Note that, for a scalar, there are no dimensions and the data begins immediately following MRANK.

4. Floating-point array (rank=3), RHO = 1,2,2

```
VALUE    DC    A(LOC,END-LOC)
LOC      DC    FL1'3'                 FLOATING POINT
         DC    FL1'0'                 FILL
         DC    FL2'12'                RANK FOR 3 DIMENSIONAL
         DC    F'1,2,2'               MRHO
         DC    DL8'.3,.2,.1,0,-.1,-.2'
END      DS    0F
```

Note that doubleword floating point values are not necessarily aligned on a doubleword boundary.

When APLSV receives a value from SVP, it checks MTYPE, MRANK, the elements of MRHO, and the length for validity and consistency. If an error is found, the value is ignored and the APLSV user who is referencing the variable receives a VALUE ERROR.

Data are transferred by the successful execution of a data transfer operation: POSTREF to read a value from SVP, and POSTSPEC to write a value. These take the arguments:

```
POSTREF PSX,VALUE
POSTSPEC PSX,VALUE
```

where PSX is the PERSHARE index which was used in the initial selection operation, and VALUE is the address of a value descriptor. If, for any reason, a data-transfer operation is not to be executed, a RELEASE PSX must be issued to release control of the shared variable. The length contained in the value descriptor for a POSTREF (read) operation must be equal to the length returned by SVP at the completion of the PREREF request.

Illustrations:

1.

```
READ     PREREF MYPSX              ATTEMPT TO GAIN CONTROL
         BZ     FAILURE            BRANCH ON FAILURE
         C      0,=F,'0'           CHECK LENGTH
         BL     NOVAL              BRANCH IF NO VALUE
         C      0,MAXL             CHECK FOR TOO LARGE
         BH     LERROR             BRANCH TO ERROR RTN IF SO
         ST     0,VALUE+4          STORE LENGTH IN DESCRIPTOR
         POSTREF MYPSX,MYVALUE     MOVE DATA
         BE     FAILURE            BRANCH ON ERROR
```

2.

```
WRITE    PRESPEC MYPSX             ATTEMPT TO GAIN CONTROL
         BZ     FAILURE            BRANCH IF ERROR
         C      0,=F'0'            CHECK FOR A VALUE
         BNL    WRITE2             GO COMPLETE WRITE IF NOT
         RELEASE MYPSX             OTHERWISE, RELEASE CONTROL
         B      READ               AND GO READ THE VALUE
WRITE2   POSTSPEC PSX,VALUE        TRANSFER DATA
         BZ     FAILURE            BRANCH ON ERROR
```

The possible return codes following the completion of a data transfer operation are:

SVZSVSF (after POSTSPEC) - Shared variable storage is temporarily full; the PCVTRAP routine is entered. When storage becomes available, the ECB addressed by PCVECB will be posted and the POSTSPEC can then be retried.

SVZNE Normal end (*CC*=2). Data have been transferred.

SVZIVS Invalid sequence - the data-transfer operation was not preceded by an appropriate initial selection.

SVZSPE Storage protect exception: The value descriptor pointed to a location partially or wholly outside of the caller's region.

SVZVTL (after POSTSPEC) - Value too large for shared variable storage.

SVZARG An error was detected in one of the arguments.

Link Editing Auxiliary Processors

In order to include the number assigned to the APLSV Type 1 SVC at a particular installation, the auxiliary processor linkedit must include the JCL produced for step S7 of the installation procedure, plus a DD statement defining the object module. The load module data set, SYSLMOD, may be inappropriate and may have to be redefined. The following utility control statements can be used.

```
INCLUDE MYLIB(MYPROC)      INCLUDE OBJECT MODULE
INCLUDE APLPAK(DQCLRS)     REPLACE CARDS
INCLUDE APLLOAD(DQCEFOO)   CONFIGURATION
ENTRY MYENTRY
NAME MYPROC(R)
```

Use of the Test Monitor

The object module DQCGBX0 supplied with the APLSV system in APLSVS.PUNCH can be linkedited using the JCL produced for step S7 of the installation procedure with

```
INCLUDE APLPAK(DQCGBX0)
INCLUDE APLPAK(DQCLRS)
INCLUDE APLLOAD(DQCEF00)
ENTRY SENTRY
NAME DQCEBX0(R)
/*
```

Once link edited, the test monitor can be used by replacing the EXEC statement which would have been used without it, as follows:

If the user program to be tested would be executed without the test monitor by the statement

```
// EXEC PGM=MYPROG,PARM=MYOPTIC
```

it would be run under the test monitor by

```
// EXEC PGM=DQCEBX0,PARM='MYPROG/MYOPTIC'
```

Appendix A

SYSTEM MESSAGES

Messages produced by the APLSV system, the APLSV utility, and TSIO have a message identifier consisting of the system prefix *DQC* followed by a single character subcomponent identification, a two digit subcomponent message number, and a message type code.

The subcomponent identifications are

- U The APLSV library maintenance utility
- I The initial program loader for the APLSV terminal system
- R The shared variable processor
- B The auxiliary processor, TSIO
- S The APLSV supervisor
- T The APLSV supervisor initialization program

The message code character is

- I Message for information only
- W Wait for operator reply
- A Eventual operator action required

The message descriptions that follow have the following format:

DQCmnnt Message Text

DQCVssss Message Description

where m is the subcomponent code, nn is the message number, t is the message type code, Message Text is the text of the message produced on the OS/VS operator's console, ssss is the source module name for the module which produces the message, and Message Description is generally an expansion of the message text.

TSIO Messages

DQCB00I TSIO XXXXX NOW ACTIVE

DQCVBT0 TSIO has been successfully initialized, and is ready to accept users. The identification number for TSIO is xxxxx.

DQCB01I TSIO WAITING FOR SVP

DQCVBT0 TSIO was started before the APLSV terminal system, and will finish initialization when the system is available.

DQCB02I TSIO - XXXX NUMBER IN USE

DQCVBT0 The background processor identification selected for TSIO by the operator start command parameter (s apltsio.p1,,,370) is already in use by another background processor. TSIO terminates.

DQCB03I TSIO INITIALIZATION ERROR

DQCVBT0 Either the jobname was incorrect, a parameter in the start command was invalid, there were no work DD statements, or there was insufficient storage for TSIO to build its tables. TSIO terminates.

DQCB04I TSIO - INVALID COMMAND

DQCVBT0 The parameter field of a TSIO modify command does not contain a valid TSIO subcommand.

DQCB05I TSIO - NO ACTIVE USERS

DQCVBT0 TSIO had no users when a modify command with TSIO subcommand *USERS* was given.

DQCB06I TSIO SHUTDOWN IN PROGRESS

DQCVBT0 Produced in response to a modify command with TSIO subcommand *USERS*, this message indicates that TSIO will terminate when the current users have signed off.

DQCB07I TSIO USER DEVICE

DQCVBT0 This message and the following lines are produced in response to a modify command with TSIO subcommand users. Each TSIO user is identified by his APLSV signon identification. If he is currently using a device, its address is displayed.

DQCB08W TSIO USER XXXXXXXXXXX REQUESTS YYYYYYY

DQCVBT0 The APLSV user whose signon identification is xxxxxx has issued a TSIO command requesting use of *UNIT*=yyyyyy. The user should have previously arranged to use the unit. If he did not, reply 'm' to deny the request. If he has, and the request is for a specific unit (for example, 180), reply 'u' to permit use of that unit. If the request is for a generic name or unit type (for example, 2400-4) the operator can reply either 'u' to permit TSIO to select any free unit of that type or 'uuu', where uuu is a device address, to force TSIO to attempt to allocate a specific unit. Note that, in any case, TSIO checks after the operator reply has been given to be certain that the unit is free before allocating it.

DQCB09I TSIO SUBTASK ABEND XXXXXX

DQCVBT1 The portion of TSIO which opens and closes data sets has abnormally terminated with completion code xxxxxx. TSIO will restart the subtask if possible. Data sets of TSIO users running under the particular subtask will be forced close. This message will be produced in response to a modify commmand with TSIO subcommand *DETACH*, in which case all tape and unit record data sets will be closed.

Initial Program Load Messages

DQCI00I MODULES NOT FOUND

DQCVIAPL The modules listed after this message were not found in either APLSVS.LOAD or SYS1.LINKLIB. The incorrect modules were specified either in the PARM field of the EXEC statement, the parameter field of the start command or in response to message DQCI01W.

DQCI01W GIVE MODIFICATIONS OR U

DQCVIAPL This message will be produced if an error is detected in the module selection list given in the PARM field of the EXEC card, the parameter field of the start command, or in a previous reply to this message. APLSV load modules all begin with a standard prefix, *DQCE*. The user must specify only the last 3 characters of the module name for the four load modules which comprise the APLSV terminal system. Consult the APLSV OPERATION section of this manual for a complete description of the possible parameters.

DQCI02I INSUFFICIENT VIRTUAL STORAGE

DQCVIAPL The region or partition in which APLSV was started is too small to permit initialization.

DQCI03I APL STARTUP CANCELLED

DQCVIAPL This message is produced after a reply of "cancel" to the message DQCI01W. It is followed by a user completion code of 122. A dump of the initialization program will also be produced on SYSUDUMP.

DQCI04I MODULE SIZE

DQCVIAPL This message, which will be followed by message DQCI01W, lists the full module names of the 4 load modules which comprise the APLSV terminal system, and, for the supervision module (subcomponent identification S) the size of the area reserved for APLSV slots and terminal buffers; for the shared variable processor (subcomponent identification R) the size of the area reserved for the temporary storage of shared variables. See also DQCI01W.

DQCR00I STOP COMMAND ACCEPTED

DQCVRSVP APLSV may be terminated, in an emergency, by the command p aplsv.identifier, in which case this message is immediately produced to indicate that the system is terminating abnormally at the operator's request.

DQCR01I SM SIZE XXXXXXX

DQCVRSVP This message is produced after successful initialization of the shared variable processor. The APLSV trouble report *INTERFACE CAPACITY EXCEEDED* will be produced if a specification of a shared variable with an APLSV quantity which occupies more that .5xxxxxxx bytes is attempted. Each variable offered or in active sharing reduces xxxxxx by 16 bytes.

APLSV Supervisor Messages

DQCS00I APL DASD ERROR op,ccu,cchhr,status,sense,)account

DQCVSUPV APLSV encountered an uncorrectable error on a swap or library data set. ccu is the physical unit address of the device on which the volume containing the data set is mounted. cchhr is the direct access address (cylinder, head, and record) of the failing record. status and sense are the CSW status bytes and the first two sense bytes, respectively. account is the APLSV account experiencing the difficulty. The possible values for op, their meanings, and the reaction of the system to each one are as follows:

SW Write to swap data set. (1)
SR Read from swap data set. (2)
D1 Read of primary directory. (3)
D2 Read of alternate directory. (3,1)
A1 Read of primary directory. (4,3,1)
A2 Read of alternate directory. (1)
DW Rewrite of primary directory. (1)
AW Rewrite of alternate directory. (1)
LW Write to library data set during)*SAVE*. (1)
LR Read from library data set during)*LOAD* or)*COPY*. (5)

REACTIONS
(1) APLSV is terminated with completion code 000300, normally after producing message DQCS01I.
(2) If surplus swap slots are available in the swap data sets, the user whose account number appears receives an unexpected *CLEAR WS* report, and the swap slot whose dasd address is given is abandoned. If no extra swap slots remain in the swap data sets, action (1) is taken.
(3) The user directories are located in the first library data set. Two copies of each directory are maintained. Codes D1 and A1 indicate that the first copy of the directory at the dasd address given cannot be read, and that a read of the second copy will be attempted. Code A1 occurs only during a)*SAVE* or)*DROP* operation for a public library.

The APLSV system programmer should be notified immediately if any of the codes beginning with D or A are received. The user whose number appears with the message should also be contacted immediately, through the *)MSG* command, to determine what workspace he was accessing when the error occurred. If the op was D1 or A1, and the message with op D2 or A2 was not produced, the primary copy of the directory may have been corrected. The system programmer should repeat the operation performed by the user to see if the message is produced again. If it is, and the command is a *)LOAD* or *)COPY*, the system programmer should attempt to *)SAVE* a workspace in the library the user was attempting to access in order to force a rewrite of the primary directory. If this fails, or if the system programmer is unfamiliar with the library structure of APLSV, the system should be terminated with a stop command (p aplsv.identifier), and an incremental dump taken. Consult the description of recovery procedures under the APLSV Utility operation RESTORE for more information.

(4) This message will only occur during a *)SAVE* or *)DROP* operation for a workspace in a public library. It indicates that the error, which is otherwise identical to (3), occurred not in the (public) library number given explicitly in the *)SAVE* or *)DROP* command, but rather in the directory associated with the account number given in the error message.

(5) The APLSV user whose account number was given receives a *CLEAR WS* report in response to his *)LOAD* command.

DQCS01I APL ABEND XXXXXX

DQCVSOSI APLSV has terminated abnormally with completion code xxxxxx. The APLSV system programmer should be contacted. Some possible abend codes are

000300 Fatal DASD error
000500 Supervisor integrity check failure
00000C APLSV initialization failure.
yyy000 System completion code yyy.

APLSV Initialization Messages

DQCT00I APLSV INITIALIZATION FAILED.

DQCVTSIP could not be started for the reason given. The possible reasons are:

INSUFFICIENT VIRTUAL STORAGE: The size of the region or partition less the size specified for SVP (...,R00(xxK),...) was too small to permit allocation of APLSV slots and buffers.

CONFIGURATION ERROR: A configuration parameter specified by means other than normal APLSV generation is incorrect. For example, the workspace size is less than the 20K minimum.

NO PORTS: The transmission control unit for APLSV terminal ports is powered down or offline.

HOST NOT VS1 OR VS2: Initialization attempted on an unsupported host system.

INSUFFICIENT REAL STORAGE: The APLSV supervisor and typewriter buffers could not be fixed in real storage, presumably because not enough page frames were available.

TIMER ERROR: The time of day clock is invalid. Reset it and restart the system.

DQCT02I LIBRARY OPEN FAILED

DQCVTOPE The APLSV libraries could not be opened for the APLSV library maintenance utility or for the APLSV terminal system for the reason given. The reasons are:

CONFIGURATION ERROR: The allocation for the first library data set is too small to contain the user directories.

DEVICE TYPES DIFFER: The APLSV library data sets are not all allocated on the same device type (all 2314 or all 3330).

DQCT03I OPEN FAILED FOR XXXXXXXX.

DQCVTOPE One of the APLSV data sets could not be opened for the reason given. Reasons are:

DD STATEMENT MISSING: The procedure or run deck used to initiate APLSV or the APLSV Utility did not contain a DD statement with ddname *XXXXXXXX*.

INCORRECT ALLOCATION: Allocation for a swap or library data set is not in one contiguous extent.

UNSUPPORTED DEVICE: The APLSV swap or library data set given has been allocated on a device not supported by APLSV, such as a tape drive.

LIBRARY ACCESS DENIED: The jobname field of the JOB statement in use does not begin with the prefix selected during APLSV installation.

DQCT04I LIBRARY OPEN FAILED

DQCVTOPE See message DQCT02I.

DQCT05I APL HAS X SLOTS. XXXX BUFFERS.

DQCVTSIP When APLSV has been successfully initialized, this message will be produced. y is the number of areas reserved for workspaces in virtual storage during this run. xxxx is the number of terminal buffers.

DQCT06I INSUFFICIENT SWAP AREA

DQCVTOPE The allocation for the APLSV swap data sets is too small to hold the active workspaces of the system configured. Consult the APLSV installation portion of this manual.

DQCT07I DASD ERROR COMMAND=XX,CSW STATUS=XXXX,SENSE=XXXX,CCHHR=XXXXX

DQCVTOPE A direct access error was encountered while a swap or library extent was being formatted.

APLSV Utility Messages

DQCU01I DIRECTORY READ FAILURE

DQCVUDIS The utility was unable to read the first copy of an APLSV library directory. See message DQCS001, op D1.

DQCU02I ALTERNATE DIRECTORY READ FAILURE

DQCVUDIS The utility was unable to read the alternate copy of an APLSV library directory. The utility terminates. See also message DQCS001, op A1. Contact the local APLSV system programmer immediately if this message is received. Do not run APLSV until the problem is resolved, if necessary by an APLSV utility RESTORE operation.

DQCU03I APL BILLING TEST -- USE OUTPUT ONLY FOR DEBUGGING

DQCVUMAI Issued to indicate that a TESTBILL and not a BILLING operation is being performed. Since the directory accounting information is not reset during a TESTBILL, there is a danger that, if the output from TESTBILL were used to bill customers, the customers could be billed twice. Hence the message.

DQCU04I TRACK COUNTS FROM COMMON LIBRARIES ARE INCOMPLETE

DQCVUBIL Produced during a BILLING or TESTBILL operation to indicate that there exist public library workspaces whose saver is not enrolled in the system, and whose disk space can therefore not be charged to anyone.

DQCU05I USER NUMBERS NOT IN SYSTEM APPEAR ON SYSLST

DQCVUBIL Produced during a BILLING or TESTBILL operation to indicate that there exist cards in the input deck which specify APLSV account numbers not actually enrolled in the system, normally because they have been dropped since the last BILLING operation.

DQCU06I REJECTED BILLING APPEARS ON SYSLST

DQCVUBIL Produced during a BILLING or TESTBILL operation to indicate that accounts rejected by the user billing routine should be examined.

DQCU07I NO INSTALLATION ROUTINE - BILLING TERMINATED

DQCVUBIL Produced during a BILLING or TESTBILL operation when no user billing routines have been linkedited with the utility.

DQCU08I WARNING - - - APL LIBRARIES XX PERCENT FULL

DQCVUDIS Produced after the successful completion of an APLSV restore operation of any kind when the APLSV library data sets are 90 percent full or more. The additional message *XX PERCENT SALVAGED BLOCKS* which may occur indicates how much the libraries could be compressed by a dump and restore.

DQCU09I LIBRARY EXTENT PARAMETER OUT OF RANGE

DQCVUDIS Produced during the utility operations DISKFMT and VERIFY to indicate that the library data set selected (which must be origin-0; that is, 0 for the first extent, 1 for the second, etc.) is invalid for the configuration of the system.

DQCU10I WORKSPACE wsid *DISK READ ERROR. STATUS*=status dsname

DQCVUDIS The APLSV utility encountered a read error while attempting to access the library data set named. Utility operation continues.

DQCU11I WORKSPACE wsid *DISK WRITE ERROR. STATUS*=status data set name

DQCVUDIS The APLSV utility encountered a write error while attempting to write the workspace given. Utility operation terminates.

DQCU12I DIRECTORY WRITE ERROR

DQCVUDIS The APLSV utility encountered a write error while attempting to rewrite a user directory. Utility operation terminates.

DQCU13I NNNN WORKSPACES DUMPED

DQCVUDUM Produced after a DUMP or INCDUMP operation to indicate that the dump is complete.

DQCU14I NNNN WORKSPACES NNNNNN BLOCKS

DQCVUDUM Produced after a DUMP or INCDUMP operation to indicate the number of workspaces extant in the system for future reference.

DQCU15I INVALID RECORD LENGTH PARAMETER

DQCVUDUM Produced when any sort of dump operation has an incorrect record length (first parameter). See UTILITY OPERATION section of this manual.

DQCU16I ******** account number or workspace id *NOT FOUND*

DQCVUDUM Produced during a SELDUMP operation when a workspace selection card which specifies a nonexistent APLSV account number or workspace identification is encountered. The operatior can to the following message with either the correct workspace identification or a single blank (r 00,' ') to ignore the workspace. He may also reply 'cancel'.

DQCU17A DIRECTORY DAMAGED -- DO NOT RUN APL

DQCVUDVE Produced after a DVERIFY operation is internal damage is encountered in an APLSV user directory. See the utility operation DVERIFY.

DQCU18I SYSPRINT DDCARD NOT PRESENT

DQCVUMAI Produced during utility initialization if the SYSPRINT DD statement has inadvertently been omitted from the utility run deck.

DQCU19I INSUFFICIENT CORE STORAGE

DQCVUMAI Produced during utility initialization if there is not enough virtual storage to permit the utility to continue.

DQCU20W INCORRECT CONTROL CARD

DQCVUMAI Produced when a utility control card with an incorrect operation is supplied to the utility. The operator can reply to the succeeding message with a correct control card, the word 'cancel' or a blank line (to ignore the card).

DQCU21I NO BILLING - SYSPUNCH DDCARD NOT PRESENT

DQCVUMAI If the SYSPUNCH DD statement has been omitted from the utility run deck, the BILLING and TESTBILL operations will be treated as incorrect control cards.

DQCU22I INVALID LEVEL --- UTILITY CANCELLED

DQCVUMAI The parameter for the LEVEL command, which is normally placed before the SELDUMP command to force conversion of APLSV workspaces to APL\360 format, was neither 0, indicating that conversion should be performed for all future dump operations, or 1, indicating that it should not. The utility is terminated, since incorrect output might otherwise be created.

DQCU23W APL BILLING INFORMATION WILL BE CLEARED IF REPLY IS 'CLEAR'

DQCVUMAI If billing is done by other means than the APLSV utility BILLING operation, the command RESET may be used to reset all user accounting (connect time and cpu time) to zero. If in doubt, reply 'cancel'.

DQCU24I SELECTION TABLE OVERFLOW -- CARDS BELOW IGNORED

DQCVURST During a RETRIEVE or SELDUMP operation, more that 100 selection cards were supplied to the utility.

DQCU25I LIBRARY NUMBER NOT FOUND NNNNNNNN

DQCVURST During a restore operation of some kind, the library number indicated was not found in the APLSV user directories. An *)ADD* command must be performed before this library can be restored.

DQCU26I INCORRECT NUMBER OF DIRECTORIES ON TAPE

DQCVURST During a utility RESTORE operation, the number of user directories on the dump tape was found to differ from the number specified in the configuration. The libraries are now invalid. Consult the APLSV system programmer to determine if the correct restore tape was mounted. If the number of directories is to be changed, the utility operation CREATE 0 must be used with a complete full dump tape (one that has no accompanying INCDUMP tape).

DQCU27I LIBRARY RESTORE TO (DATE OF FULLDUMP)

DQCVURST Produced during a utility RESTORE operation as an historical record.

DQCU28A MOUNT YY.DDD FULL-DUMP TAPE FILE

DQCVURST Produced during a utility RESTORE operation after an incremental dump tape has been read, to indicate that the preceding full dump tape should be mounted. If recovering from a system failure, the operator may wish to mount a previous incdump tape at this point, in which case the message will be issued again when that incdump has been read. See the discussion of the APLSV utility operation RESTORE.

DQCU29I MISMATCH OF DUMP DATES,

DQCVURST The tape just mounted in response to message *DQCU27I* is not the DUMP tape which preceded the INCDUMP tape which has already been read. The correct tape must be mounted.

DQCU30I NOT A FULL-DUMP TAPE

DQCVURST Produced during a *CREATE* operation if an attempt is made to perform the *CREATE* with any tape except a full dump tape.

DQCU31I NOT FOUND ON TAPE workspace identification

DQCURST This message will be produced at the end of a RESTORE operation which began with an incdump tape for all workspaces which, though in the directory at the time of the incremental dump, were not found on either the incremental or full dump tape. See the description of the utility operation RESTORE.

DQCU32I DIRECTORIES FULL -- UTILITY CANCELLED

DQCVURST There is no more room in the APLSV user directories to hold the blocks which describe the names and locations of saved workspaces. The number of directories must be increased, following the instructions given at the end of the section APLSV INSTALLATION in this manual.

DQCU33I LIBRARY EXTENTS FULL -- UTILITY CANCELLED

DQCVURST All available space in the library data sets has been filled with saved workspaces. If possible, the number or size of the APLSV library data sets should be increased. If this is not possible, some users or some workspaces should be moved from the library data sets to archival storage by means of the APLSV utility operation SELDUMP and the APLSV operator commands *)DROP* and *)DELETE*.

DQCU34I NOT AN APL-DUMP TAPE

DQCVUTAP The tape mounted for a restore operation was not produced by the APLSV or APL\360 utility programs.

DQCU35I TAPE ERROR, WORKSPACE MAY HAVE BEEN LOST

DQCVUTAP During some type of restore operation, a tape read error prevented the workspace identified from being read.

DQCU36I UNEXPIRED FILE

DQCVUTAP The expiration date of the tape mounted for a dump operation of some sort is greater than today's date.

DQCU37W REPLY NEWTAPE, IGNORE OR CANCEL

DQCVUTAP This message is produced after messages such as DQCU36I to permit the operator to select whether he wishes the utility to restart with a new tape, ignore the error noted, or terminate.

DQCU38I VOLUME SEQUENCE ERROR

DQCVUTAP The tape mounted does not follow the previous tape, according to its volume sequence field. The message DQCU37W will be produced next. Sequence errors during the operation RESTORE should never be ignored.

DQCU39I WORKSPACES CONVERTED FROM NNNNNN BYTES TO NNNNNN

DQCVUTAP Produced during any sort of restore operation, this message indicates that the workspace size conversion function of the utility is being exercised.

DQCU40I NAMES OF OVERSIZE WORKSPACES APPEAR ON SYSLST

DQCVUTAP During a restore of a tape from a system with a workspace size larger than the current configuration, a workspace was found that was too large to be converted.

DQCU42I STATE INDICATOR CLEARED

DQCVUTAP An APL\360 dump tape is being read as input to an APLSV utility restore operation of some sort. This warning message indicates that the suspended execution state indicator is being cleared, and that all local variables are being deleted.

DQCU43I HDR1 data set name date *VOLID*=serial,*UNIT*=tape unit

DQCVUTAP This message is a formatted representation of the tape label read or produced during any APLSV utility operation which accesses tape.

DQCU44I EOF -- END CARD PROVIDED

DQCVUURE A set of workspace selection cards was not followed by an END card. One has been supplied.

DQCU45I INCORRECT SELECTION CARDS

DQCVUURE The selection card listed is incorrect. The operator may reply with the correct card, a single blank, or the word 'cancel'.

DQCU46I NAMES OF DAMAGED WORKSPACES APPEAR ON SYSLST

DQCVUVAL The workspace validity check routine, which operates during all utility operations which read or write tape, has detected internal damage in at least one workspace. The damaged workspaces should be corrected by use of the APLSV commands *)CLEAR*, *)COPY*, and *)SAVE*.

DQCU47I INVALID - - APL RUNNING

DQCVUMAI The utility operation listed cannot be performed while the APLSV system is running.

Appendix B

WORKSPACES DISTRIBUTED WITH APLSV

NEW WORKSPACES FOR USERS

The following workspaces are intended for APL users, and were not included in previous releases of APLSV. Each group of related workspaces is described in a paragraph below; the names of the workspaces are listed at the left margin, immediately beneath the relevent heading.

Array File

APLFILE

The workspace *APLFILE* provides functions for storing and retrieving APL arrays in 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

CONVERSION

This workspace 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 Functions

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 APL (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" contains additional functions intended to read a text (i.e. a program listing) from a data-set outside APL, 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 Workspace

TSIO

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. 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 interpet 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

FORMAT

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

WSFNS

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 functions which in APLSV are replaced by system variables having names such as *⎕FP* 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: *DFLAY*, *DIGITS*, *ORIGIN*, *SETFUZZ*, *SETLINK*, and *WIDTH*.

USER WORKSPACES DISTRIBUTED IN EARLIER APL RELEASES

The following workspaces are distributed essentially unchanged from the form in which they were included in earlier APL releases. They are described more fully in Part 4, "Library Functions," and Appendix B of the APL\360 User's Manual, GH20-0906.

Notice of System Changes, Schedules, etc.

NEWS

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

PLOTFORMAT

This workspace contains defined functions useful in formatting numeric data and in constructing at the typewriter graphs or histograms. 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 plotting and formatting are provided by the formatting workspace described above, and by the plotting functions contained in FDP 5708-AGL, "Graphs and Histograms in APL."

Instructional Workspaces

APLCOURSE
ADVANCEDEX
TYPEDRILL

The workspaces *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 evaluation 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.

WORKSPACES FOR THE SYSTEM OPERATOR

The following workspaces are provided for use of the APL operator or system personnel, and are distributed in library 314159.

Management of OS/VS Catalog Data Sets

CATALOG

This workspace is usable only by a system-level user of TSIO. It permits the user to list OS/VS catalog data set, and to search the system catalog for data sets belonging to a particular user and return their names and locations.

Sample Functions for Management of OS/VS Data Sets

UTILITY

The functions in this workspace demonstrate the use of APLSV and TSIO in managing operating system data. Display functions report the status of system resources; for instance, the volume serial numbers mounted on particular device classes may be displayed. Data-sets generated under TSIO may be distinguished from others, or transitory from permanent data sets. Management functions include the ability to catalog or uncatalog data sets, or to delete data sets. Combined with the display functions, these permit interactive control from the terminal of a number of system management functions.

Operator Functions

OPFNS

This workspace provides many functions useful to the system operator in displaying the status of the APL system and its users and for managing the APL system as it runs. The functions may be used only by a privileged APLSV user, normally the APLSV operator.

TSIO Management Workspace

TSIOPS

This workspace is used by the APLSV operator to enroll users of TSIO and to establish their access levels.

Test Workspace for Terminals

CETEST

This workspace may be used to exercise IBM terminals to test their adjustment, and facilitate diagnosis of malfunctions.

Appendix C

THE PRIMES UP TO 50

```
      (2=+⌿0=(⍳50)∘.|⍳50)/⍳50
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47
```