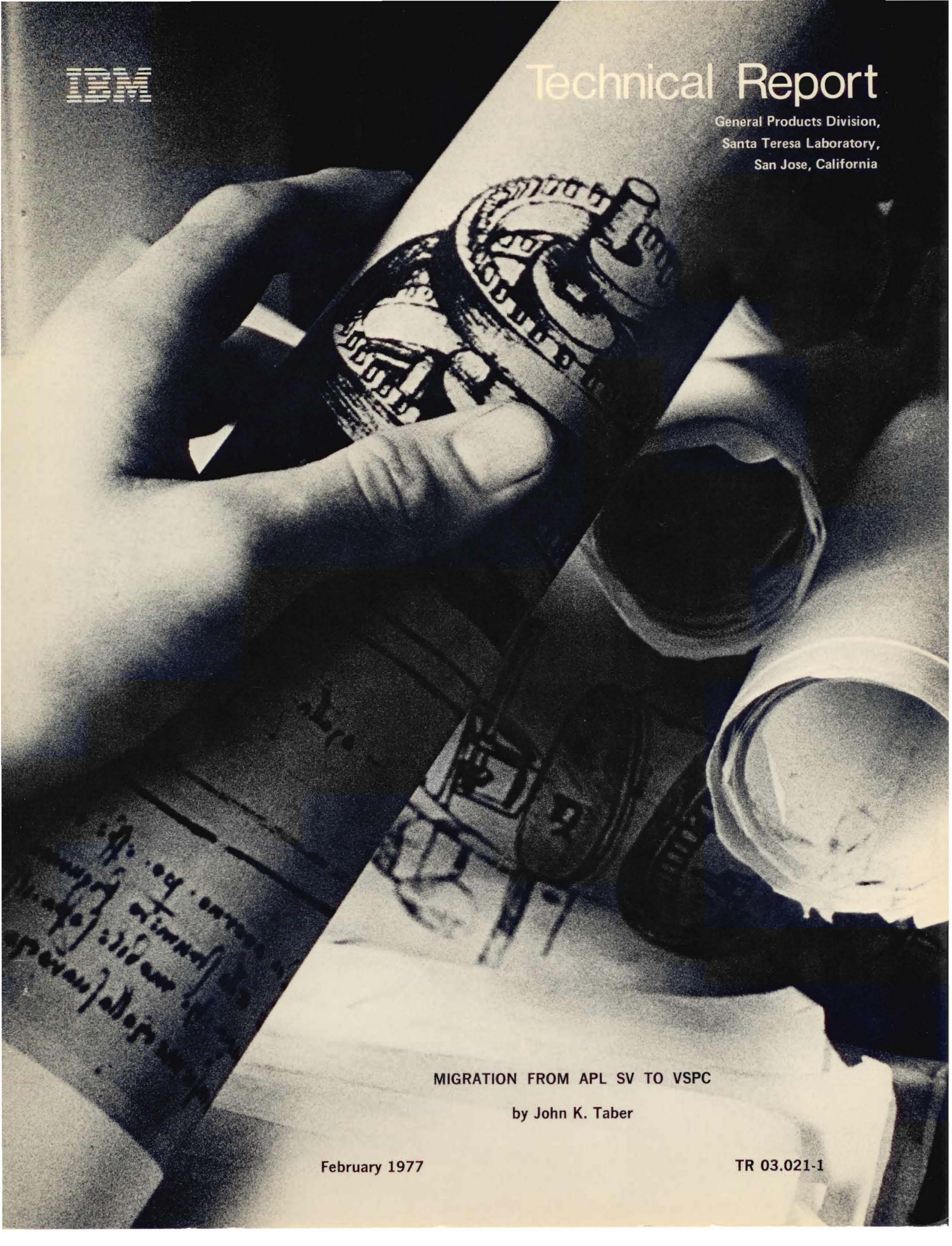
IBM

Technical Report

General Products Division,
Santa Teresa Laboratory,
San Jose, California



MIGRATION FROM APL SV TO VSPC

by John K. Taber

February 1977

TR 03.021-1

TR03.021-1
February 1977

MIGRATION FROM APL SV TO VSPC

by

John K. Taber

International Business Machines Corporation
General Products Division
Santa Teresa Laboratory
San Jose, California

ABSTRACT

This report discusses system considerations necessary to smoothly migrate an APL SV installation to VS APL under VSPC. It outlines migration strategies and procedures. It also describes what the Conversion program does.

ACKNOWLEDGMENTS

I wish to thank Harry Saal, William McCain, Vern Griffith, Joanna Woodrow, and Alex Morrow for their helpful suggestions and review of this report.

CONTENTS

MIGRATION FROM APL SV TO VSPC
by
JOHN K. TABER

INTRODUCTION

Migration of an installation cannot be painless; it requires planning and the cooperation of the installation and its users if disruption is to be minimized. To keep the task manageable for an installation of any size, migration will usually be segmented, and each segment will need several iterations. Do not throw away the old APL SV installation as soon as VS APL is installed under VSPC; you will have to keep it around to accommodate recalcitrant conversion problems and stragglers. The Conversion program will prove an invaluable tool in migrating your users, but it is only a tool, not a substitute for intelligent, human planning. The following pages describe migration problems that we are aware of and give advice on how to cope with them using the available tools. They are addressed to both the installation and the user. The user and installation people should be familiar with the following manuals:

APL Language (GC26-3847)
VS APL for VSPC: Terminal User's Guide (SH20-9066)
VS TSIO Guide and Reference (SH20-9107)

Installation people should also be familiar with:

VS APL Installation Reference Material (SH20-9065)
APL Shared Variables (APL SV) Operations Guide (Version 2) (SH20-1461)
APL Shared Variables (APL SV) Operations Guide (Version 3) (SH20-9088)

SYSTEM PLANNING

This section describes how to allocate VSPC resources to accommodate APL SV users for workspace libraries and shared variables. It also discusses differences in handling libraries between VSPC and APL SV. It concludes with a brief summary of TSIO migration considerations.

Library Space

Given that we know how much disk space the APL SV workspaces occupy for an installation, we would like to determine how much space we should allocate for the VSAM data sets. This cannot be computed exactly, but the following discussion will help in making an intelligent guess. On the average, it is slightly more space.

There are major differences in how the libraries file workspaces between APL SV and VSPC. In APL SV, all workspaces for a given installation are fixed length, but the disk space occupied by workspaces differs depending on how full the workspaces are. In general, free space is not filed. The R13 stack (1000 bytes), however, is filed. In addition, each workspace requires a PERSAVEW entry in a directory, and each account requires a PERLIB entry in a directory. The workspace is filed in two segments, each of which consists of one or more physical records. The physical record length is fixed and dependent on device type; it is one-half the track length of the device for most devices. The last record may have any number of pad bytes to fill it out to the record length. On the average, half of one record per workspace is waste. The first segment includes the beginning of the workspace through the last m-entry, rounded up to a double word boundary. The second segment includes the execution stack (if any) through the R13 stack rounded up to the length of a physical record. In VSPC, all workspaces are variable length, and are filed in one logical segment, which consists of one or more VSAM control intervals. The segment begins 1,956 bytes after the start of the workspace and goes through the m-entries. Free space, the R13 stack, and the first 1,956 bytes are not saved. However, there are dead bytes filed in VSPC as in APL SV, because the minimum segment filed is a 16K control interval, padded as necessary to fill out the interval length. On the average, one half of one control interval per workspace, or 8K, will be waste. The control interval itself consists of:

16,352 bytes of workspace and pad
32 bytes of appended trailer data
(including 7 bytes of VSAM overhead)

The workspace is filed in one or more of these records. In addition to library space to contain the workspaces, 112 bytes is required for each user for his User Profile Record (UPR – the equivalent of the PERLIB) in the directory and 64 bytes for each workspace for the Directory Entry Record (DER – the equivalent of the PERSAVEW).

There is a further complication; although less of the VS APL workspace is filed than the APL SV, the VS APL workspace may be some unknown percentage larger. Internal objects (variables, group lists, functions, etc.) occupy doubleword blocks, and hence are padded up to doubleword boundaries in VS APL as opposed to full word blocks in APL SV. Also, objects have more description overhead. And finally, function tokens (the primitive functions of the language in defined functions) require half word representation as opposed to byte representation. But, VS APL saves space because functions do not have directory blocks and it uses AP vectors and synonym links to save space for variables. The Conversion program, however, does not convert variables to synonym links or AP vectors; this savings will not be realized until the converted workspace is loaded, executed and resaved. Hence, it is exceedingly difficult to express the size of a VS APL workspace exactly in terms of its APL SV workspace equivalent. It gains space here and loses space there. The Conversion program uses 10 percent as a growth figure going from APL SV to VS APL, so far, successfully. It may be too high.

For the VSAM directory data set, we may calculate

112 x no. of users for UPR
64 x no. of workspaces for DER
not including VSAM and VSPC overhead
(about 12 bytes per record)
not including desired growth

Some percentage for growth is recommended because UPR's are not easily reclaimable for 'deleted' accounts as are PERLIB's in APL SV. Locking an account in VSPC does not free the UPR for adding an account. Accounts are deleted only by copying the libraries, scratching then redefining the VSAM libraries, and, finally, copying the files back in with a REMOVE option. If your accounts are volatile, that is, there is a lot of adding and deleting of accounts, you can minimize these library compressions by allowing for sufficient growth.

You may determine the number of users and workspaces from the last page of an APL SV ACCTG 1 listing. Total the number of current and remaining libraries for all directories. This is the number of UPR's that you will need. Do the same for current and remaining workspaces; this is the number of DER's that you will need. These totals include actual users and workspaces and potential new users and workspaces provided for in the APL SV installation. If these potentials (remaining libraries and workspaces) seem unreasonable, adjust accordingly. The unbooked space columns on the ACCTG 1 listing can be ignored if the numbers are positive. Unbooked space means the left over directory space if the entire installation quota were indeed used. If negative, it means that the directory allocation is not big enough for the installation quota. If negative, add their absolute values to the library or workspace totals as appropriate.

Computing the minimum VSAM library allocation for workspaces is more difficult. The actual space occupied in APL SV would not give a satisfactory figure because it would not allow for workspaces which expand in size. On the other hand, using the slot size of the APL SV installation (a worst case figure) would give an unrealistically large figure. Indeed, APL SV installations do not allocate worst cast library extents. They figure (from experience) that half the workspace quota for the installation will be saved, and of that half, the workspaces will be half full. Thus, if the installation workspace quota is 10,000 and the slot size is 90,000, the library extent in bytes is 5,000 x 45,000. We suggest the following procedure to determine a reasonable VSAM library allocation for the workspaces.

1. Determine the actual bytes used and the actual extent of the library. This can be figured from a recent ACCTG 1 run and a recent INCDUMP or FULL DUMP run. The accounting run listing on the first page lists the slot size and on the second to last page lists the number of saved workspaces and the number of blocks occupied by the saved workspaces. On the last page, the dumps list the percentage of allocated blocks used. The APL SV installation manual gives the block size in bytes for the library device. Compute the average saved workspace size (WSSIZE) as follows:

   WSSIZE ← (*BLOCKCOUNT* x *BLOCKSIZE*) ÷ *WSCOUNT*

   WSCOUNT is the number of saved workspaces

2. Compute the growth factor, a figure between 1 and 2 as follows:

   GROWTH ← 1 + (100 – *PERCENTFULL*) ÷ 100

   PERCENTFULL refers to the percent of library used in the dump listing. Trim or add to the decimal part as desired.

3. Now compute the VSAM library extent in bytes by the following formula

   GROWTH x *WSCOUNT* x 16384 x ⌈ (1.1 x *WSSIZE* – 3000) ÷ 16352

The result is the minimum VSAM library size for workspaces ignoring any system overhead.

Workspace Size

There is a minimum VS APL workspace size below which the workspace is not viable, that is, there would not be enough free space for interpreter demands. This figure is 20K. Conversion forcibly defines small APL SV workspaces to be a minimum of 20K. Otherwise, Conversion defines the workspace size to be

6144 + 1.1 x *APLSVSLOTSIZE*

The increase of 10 percent is to avoid workspace full problems during conversion. The 6K allows for the terminal buffer and VSPC/VS APL interface requirements.

Shared Variable Space

In VSPC, each user must be given adequate space for his shared variables. To determine space needs, you will have to consider the user's current TSIO needs and his future use of VSPC auxiliary processors. The following discussion gives you a simple, gross way to provide a shared variable size quota and a refined, precise way to calculate a size quota based on TSIO usage.

VSPC limits shared variable resources by both a quota and a size limitation. The quota is simply the shared variable quota, as in APL SV. The size limitation must be specified for each user because the default is zero bytes. Size limitation means that the user's largest single shared variable that he is allowed to specify cannot exceed a certain specified size, entered in his user profile record. Although there is no similar limitation in the APL SV PERLIB for a user, there is nevertheless an installation determined limit. This is one-half the shared memory, which is an APL SV startup parameter. In other words, in APL SV, a user cannot specify a shared variable larger than this figure even though this figure is not an explicit limitation for his account. There is a similar startup parameter for VSPC which you should have changed if necessary so that VSPC can provide adequate space for the APL shared variable users. VSPC uses the smaller of half the shared memory size or the user's shared variable size maximum in allocating space for a shared variable. To be more accurate, the size maximum of both share partners, or one half of shared memory, determine the maximum size variable which can be shared. The largest variable the user can *specify* is the smaller of half shared memory or the user's size maximum. The largest variable the user can *reference* is the smaller of half shared memory or his share partner's size maximum. Thus the share size in the UPR gives you a further refinement for resource allocation. Since a shared variable is transient, that is requires system resources for a short time, you may make the user's share size in VSPC be one-half the shared memory size commonly used in the APL SV installation. This figure may be provided to the Conversion program, or entered by an administrator for each user in VSPC. This is the simple, gross way to set a size maximum, but the price for this could be slower system throughput (if variables this large are actually shared) since shared variables must wait for space to be made available. It is better to determine a reasonable share size for the most users (say 75 to 90 percent) and supply this figure to Conversion, then use VSPC ALTER to change share size for the few users for whom the reasonable figure is inadequate. To do this, determine share size for all or most users, then pick one that will satisfy the most.

Reasonable Share Size for TSIO

Reasonable share size for an individual user depends on TSIO code and record format used to access the file. If the record format is undefined (U), or fixed blocked standard (FBS), or fixed blocked (FB) but accessed as FBS, then his reasonable share size is the largest blocksize of all such files plus an APL overhead explained later. If the record format is variable (V), or variable blocked (VB), or unblocked (F), or fixed blocked (FB) and accessed as FB, then his reasonable share size is his largest logical record for such files plus an APL overhead. If the user has a mix of format types, then use the largest figure obtained.

The APL overhead is 16 bytes for all TSIO files except FBS record format, for which it is 28 bytes. Get the data set descriptions (record length, blocksize, format) by running System Utility IEHLIST (LISTVTOC) for your TSIO packs. You will have to survey your users to determine if they access FB format files as FBS. You should be aware that for the rare TSIO user, this determination of share size will not be adequate because VS APL will double his blocksize. This occurs where the type of the result of an APL expression is floating point in VS APL but is integer in APL SV. See the discussion of TSIO considerations.

Share Size for VS APL Auxiliary Processors

You should count on your users eventually using VSPC library files or VSAM external files, especially those users who will be writing new applications. For the auxiliary processors distributed with VS APL, share size is determined only by the size of the variable to be shared, not by record length or blocksize of the file. To the variable size you must also add an APL overhead, which depends on the rank of the variable. It is about 16 bytes plus 4 bytes times the rank of the variable. You will have to canvass your users to determine the maximum sized variable they expect to write

File Buffers

For VS APL Auxiliary Processors, but not for TSIO migration, the user will require an adequate number of file I/O buffers. Usually, one 4K buffer is needed for each opened file. However, VSAM data sets in rare cases may require more than one buffer per file. If a user plans to use the full screen management auxiliary processor (AP 124), he will require one and a half buffers (6K) and perhaps more depending on the complexity of his screen format just for the full screen manager. If the screen format is 10 fields or less, a 6K buffer should be adequate. The screen manager requires more buffer space in 2K increments for more complex formatting. VSPC subtracts the buffer space for each opened file from the user's maximum workspace size quota while the file is opened. It is available for the active workspace again when the file is closed. The maximum workspace size is a VSPC parameter in the user's profile which must not be confused with the size of the workspace. The maximum workspace size is the largest size VSPC workspace slot which the user's workspaces will need. In this slot will fit the terminal buffer, the file buffers, the full screen buffer, and the VS APL workspace itself with all its objects, free space, and R13 stack. For example, if a user's APL SV workspace size is 80K, but its shared variables are managing four simultaneously opened files, his VSPC maximum size will be about 90K for his original workspace plus 16K for file buffers, plus about 10K for the full screen manager. If buffer need is not allowed for, the user's shared variables will not be able to communicate with the user's files. Conversion automatically gives each user a workspace maximum large enough for two file buffers, and about 10K for the full screen manager buffer. If these are not adequate, the VSPC administrator will have to alter the user's maximum workspace size. The smallest user maximum workspace size that Conversion will define is 30,000 bytes, regardless of the original workspace size. In other words, for small APL SV workspaces (for example, 16K), Conversion forces the VSPC slot size to be 30,000 bytes. The minimum viable slot size to satisfy implicit resource demands is at present about 25,000 bytes, but Conversion uses 30,000 because these requirements may change in the future.

Library Differences

Many aspects of library difference could be discussed here, but we shall limit ourselves to a few significant points which might confuse the user due to his APL SV expectations

Library Types

VSPC distinguishes three types of libraries: public, project, and private. A library's type is not determined by the account number as in APL SV, but instead by flags. Once type has been defined, it cannot be changed. The Conversion program automatically flags accounts under 1,000 as public libraries and accounts greater than 999 as private. A VSPC command issued by an administrator must be used to mark an account as project. The project library is new for APL SV users. In effect, a project library is a public library for those accounts which are members of the project, and is a private library for non-members of the project.

Since type cannot be changed, be sure that the conversion definition is acceptable. If unacceptable, predefine the accounts in VSPC before the Conversion produced 'copy' tape is copied into the VSPC library. Do consider making some of your hitherto public libraries into project libraries. Apt candidates are specialized public libraries devoted to the interests of a department, group, or project, and otherwise not of interest to the installation as a whole. To create a project library, predefine the account in VSPC as a project library, then after migration, alter the UPR's of members to make them project members of the project account.

Shareable Access

By default, files in VSPC are non-shareable. This means that normally a file cannot be loaded by any user other than its owner. An APL workspace, however, is normally made shareable by VS APL to preserve a traditional characteristic of APL. Nevertheless, a VS APL workspace can become non-shareable if the owner explicitly makes it non-shareable through the VSPC SHARE command or if the workspace is imported into the VSPC library through the VSPC Service Program IMPORT command. An imported workspace must be explicitly made shareable with the SHARE command if that is what the owner wants. The non-shareable characteristic of a workspace is ignored if it is in a public library (that is, it is loadable by non-owners), but it retains its non-shareable status. If renamed and saved in a private library, it is non-loadable from the private library by other users. Also, a non-shareable project library workspace keeps its status in a similar way; however, non-members cannot load a non-shareable workspace (they can load a shareable workspace if they know its name and password). Note well that the distributed workspaces are *imported* into VSPC from the distribution tape and are therefore non-shareable. A system administrator should make them shareable. Conversion marks all workspaces as shareable (in the DER), thus preserving the characteristics of APL SV workspaces. The user should be made aware that he can further protect his workspaces by using VSPC commands to make his workspaces non-shareable. Non-shareable status is especially recommended for project library workspaces. The project library owner should use the VSPC SHARE command to make these workspaces non-shareable. Workspaces imported into VSPC (by the import command) are non-shareable and must be specifically made shareable by their owners, if that is what the owner wants.

Passwords and Workspace Names

Passwords and workspace names in VSPC must be enterable by non-APL keyboards. Practically, this means that passwords and workspace names cannot contain underscored characters and delta. The Conversion program preserves all passwords as is. This means that if an APL SV password contains VSPC forbidden characters, the user will not be able to log on to his password protected account or load his password protected workspace. The account password can be changed by an administrator The illegal workspace password can never be changed by anybody and the workspace is forever inaccessible. The Conversion program rejects VSPC forbidden workspace names. Users must change such workspace passwords while still using APL SV.

Matching Content Attributes

Each processor in VSPC has an internal name, called the content attribute, which uniquely identifies the processor the VSPC. The VS APL interpreter's customary content attribute is 'A0' (external name APL). However, the installation may assign any content attribute desired from 'A0' to 'AF'. Each workspace saved by a processor is given a content attribute which matches its processor, and normally, an interpreter cannot load a workspace with a different content attribute. Conversion automatically marks each workspace with the content attribute 'A0'. A provision in VSPC, however, allows a higher level interpreter to access the workspaces of its predecessor. A compatibility bit map, assembled into the interpreter, describes which 'An' workspaces are loadable. Compatibility may be in either or both directions depending on the bit maps in each interpreter. Once the back level workspace is saved by the higher level interpreter, it is given the content attribute of the higher level. This could mean that it cannot now be loaded by the back level interpreter. Thus, if a VSPC installation is running two different versions of essentially the same interpreter, but the bit maps specify compatability in only one direction, the higher level user should avoid preempting his back level workspace. If he loads a back level workspace he should rename it when saving the higher level version. Finally, if the VSPC installation installs VS APL with a content attribute other than 'A0', the area mapped by the ASUPAT macro in the interpreter module APLPCOEX should be reassembled to mark A0 workspaces as compatible to use Conversion's results. Note also that you can change the content attribute of a workspace by exporting it, then importing it. The)LIB command does not list workspaces with a content attribute different from the interpreter, whether compatible or not.

Keeping CONTINUE Workspaces

In previous implementations of APL, the CONTINUE workspace has been maintained in the library by the installation but not counted against a user's quota. Many users save an extra workspace by saving it as CONTINUE, taking the chance, of course, that they may lose it with a genuine line drop. In VSPC, CONTINUE workspaces are dropped with)OFF and)OFF HOLD, and CONTINUE files of all types are dropped by the VSPC OFF command. Conversion preserves CONTINUE workspaces, but if the user wants to keep it in VSPC, he must resave it with a)CONTINUE command at the end of each session, or save it under a new name before he logs off from VSPC. Otherwise, when he logs off, he will lose the workspace. When he first logs on to VSPC, his APL SV CONTINUE workspace will be automatically loaded for him, unless it is password protected. As in APL SV, the VSPC CONTINUE workspace disk space is not counted against the maximum library quota which VSPC checks when a workspace is saved, nor is it included in the disk space totals recorded for accounting purposes.

TSIO Considerations

Migration requirements for TSIO users and their data files are discussed in considerable detail in VS TSIO Guide and Reference (SH20-9107). This manual should be read by both the installation and the users before migration. Essentially, VS TSIO is an auxiliary processor which closely duplicates the functions of APL SV's TSIO so that almost all users will be accommodated with no conversion effort on their part. Some users, however, will have to either convert their data files, or modify their workspace functions, or do both. After migration, other users may choose to convert their files to VSPC library files or to external VSAM data sets.

There are three potential problems that may be encountered by a few users: one is CODE=A character data which the user intends as byte integers; second is CODE=C character data which the user intends as APL graphics; and third is the expansion of record sizes in rare cases caused by VS APL type differences from APL SV.

VS TSIO always assumes that the meaning of character data in CODE=A files is their APL graphic representations. It assumes that character data in the file is the APL SV encoding of graphics. Thus, reading from the file, VS TSIO translates character data to VS APL graphic equivalents. Writing to the file from the VS APL workspace, VS TSIO translates to EBCDIC equivalents to enforce a standard graphic code for APL. This assumption is not valid for all users. For a few users, character data are byte integers (have physical meaning instead of logical meaning — see discussion of variables in workspace conversions later in this report). These users should convert their character data to EBCDIC equivalents as follows.

Read each character record in the file. In the workspace, the characte.ᶜ vill be translated to their VS APL graphic equivalents by VS TSIO.

Using the translation functions and index variables in the CONVERSION distributed workspace, backward translate to APL SV byte integers.

Write the backward translated characters to the file.

If the user does not update his file, he may modify his read functions to backward translate instead of converting the file.

Note that APL SV TSIO cannot read CODE=A character data once it has been written to the file by VS TSIO.

For CODE=C files, VS TSIO always assumes that,the data is byte integers (even though they may have graphic meaning outside of APL). There is no translation either way, between the workspace and the file. But a few users may have intended APL graphic meaning for CODE=C data. These users should convert their data set as follows:

Read each record.

Using the functions and index variables in the CONVERSION distributed workspace, forward translate from APL SV characters to VS APL characters.

Rewrite the forward translation into the file.

These users may, if they choose, not convert the data set, but modify their read functions to forward translate, and their write functions to backward translate.

The third potential problem is that in rare cases, VS APL, in writing to a TSIO file via VS TSIO, may increase the record size (and hence, the blocksize). This can occur only with CODE=A files. An APL expression may store results in a different internal type in VS APL than in APL SV (for example, floating point instead of fixed point) and hence generate larger records. One example is

A[3;4] ← 1 ↑ 6 3.2

In this example, the result is forced to floating point in VS APL while in APL SV the result is fixed point if A is fixed point. In such cases, the user may use the *SIZEGP* functions in the TSIO distributed workspace to force the results to be the right type. Or he may redefine his file for the larger record and blocksizes. He can do this simply by copying the original data set to a new data set, using VS TSIO.

The installation should back up the TSIO files before migration in such a way that an individual file can be retrieved later. One suggestion is to use IEHMOVE with COPY VOLUME from the TSIO packs to tape. Later, an individual file can be retrieved by IEHMOVE COPY DSNAME and the right tape sequence number (provided by the COPY VOLUME listing). However, IEHMOVE does not work for partitioned data sets with fixed blocked standard (FBS) record formats, unless your system is MVS release 3 or later. To successfully archive these data sets, redefine them as fixed block (FB) instead of FBS. Another suggestion is to use IEBCOPY for a pack-to-pack copy. This does not have a problem with FBS partitioned data sets because in the DCB parameter of the DD card for the data set, you can specify RECFM=FB.

If migration involves the renumbering of accounts, TSIO files that use the account number to generate part of the data set name will have to be renamed to match the new account number. User TSIO data sets have qualified OS names, the second qualifier being an alphabetic encryption of the user's account number. The second qualifier encryption is

'ABCDEFGHIJKLMNOP' [⎕*IO* + (8⍴16)⊤*LIBNO*]

The second qualifier will have to be changed by the VS TSIO administrator. Note that LIBNO may be negative, which indicates a restricted data set, accessible only by the owner (or a system level user).

THE CONVERSION PROGRAM

This section discusses significant points about the input to the Conversion program, the output, and finally conversions to the workspace itself. In brief, Conversion both converts workspaces, and enrolls users in VSPC. It runs as an independent batch job.

Input and Output

Input APL SV Dump Tapes

Input to Conversion is an APL SV selective dump tape, or incremental dump tape, or full dump tape. The full dump tape contains all the directories followed by all the workspaces of an installation. The incremental dump tape contains all the directories followed by only those workspaces saved since the last full dump. The selective dump tape contains only specific workspaces, and no directories. Only the directory contains a user's workspace quota, shared variable quota, and CPU time limit. Therefore, for best results, dump tapes with directories should be used for Conversion. If directories are not input to Conversion, Conversion uses defaults, which are:

500,000 bytes library space
0 shared variable quota
0 shared variable space
infinity cpu time limit
no password for user account

If directories are not input, only the library space parameter may be overridden with a Conversion control card, however, the figure supplied will apply to all users converted.

Conversion Control Cards

Optional input is control cards which specify Conversion options and parameters and specific workspaces for selective conversion. Full conversion is the default option. It does not mean that the input tape is a full dump tape. It means that Conversion will attempt to convert all the workspaces on the input tape(s) whether full dump, incremental, or selective dump. Select conversion means that Conversion will convert only the specified workspaces if they are found on the input tape, which may be any sort of dump. Resume conversion means that Conversion will resume a full conversion once it has found the specified workspace on the input tape(s). Workspaces before the target workspace are ignored. Resume is used to segment the migration task, to skip a bad spot on the tape, and to restart conversion if Conversion shuts itself off due to an excessive number of damaged workspaces. (Conversion assumes that it itself has been damaged if more than ten damaged workspaces have caused program checks.)

Output VSPC 'COPY' Tape

The output of Conversion is a VSPC 'copy' tape, suitable for a copy input to the VSPC Service Program. The 'copy' tape consists of a User Profile Record (UPR) for each account, a Directory Entry Record (DER) for each workspace, and the workspaces represented as VSAM 16k control intervals. The UPR enrolls the user in VSPC if he is not already enrolled. It is ignored by VSPC if the user is already enrolled. The Conversion produced 'copy' tape is not a standard VSPC copy tape. Its significant difference is the sequence of accounts and workspaces, which are written on tape in APL SV directory order instead of collating order as for a VSPC produced copy tape. Directory order means that all the workspaces for directory 0 are followed by all the workspaces for directory 1, and so on until directory n. Furthermore, accounts within a given directory are not in collating sequence by signon number; they are entry sequenced by time (barring account deletions, changes, etc. which reclaim freed PERLIBS). For an APL SV account, the workspaces are in collating sequence by workspace name. This is not true for an APL/360 account, however, where the workspaces for an account are dumped in entry sequence by time. The practical significance of this is that the VSPC Service Program selection functions cannot be used on a Conversion produced 'copy' tape. There are minor differences as well, concerning the blocking of records on the tape.

Note that under DOS, the Conversion 'copy' tape must have standard DOS labels to be acceptable to DOS VSPC. In OS, the tape may be labelled or unlabelled. VSPC can read a DOS copy tape in OS by using the Bypass Label Processing option. An OS copy tape cannot be read in DOS.

Output Conversion Report

A second output of Conversion is the conversion report, which is a summary of exceptional conditions for each workspace grouped by account. The report is to be burst apart by account and sent to each user. Users are urged to at least scan the report. It is our experience that users tend to ignore the report. Apparently, most users are able to use their converted workspaces scarcely aware that Conversion has modified many lines of their functions. Where an idiomatic translation fails, users tend to correct the translation during execution without ever consulting the report, which is rather voluminous even though it is a summary of exceptions. Nevertheless, the user should scan the report for disasterous failures. Some workspaces are damaged and unconvertible and are deleted by Conversion. Functions with unacceptable headers to VS APL are unconvertible and are deleted. The algorithm for computing the VS APL workspace size can fail so that some of the APL SV workspace objects cannot be copied into the VS APL workspace due to workspace full. A function line too long for conversion is replaced in its entirety by an execution time warning message containing a deliberate syntax error. This can be painful because the figure is quite large (about 4000 bytes) meaning that a lot of data is lost. This is likely to occur where a user enters a large amount of text in a character constant without closing quotes between carrier returns.

APL SV conversion reports can be shortened considerably by one simple precaution, which is to have all users define comparison tolerance in all their workspaces where the default tolerance is appropriate to be 1E-13. Due to a bug in APL SV, the default comparison tolerance is 1.136729599338082E-13. This is a reportable exception for Conversion and can waste a whole page.

Workspace Conversions

Essentially, workspace conversion is a)COPY of the APL SV workspace into an initialized clear VS APL workspace. Only global objects and some workspace parameters are converted; temporaries and local objects are ignored.

Workspace Parameters

A clear VS APL workspace is initialized by setting global workspace parameters to their APL SV values as follows:

> ⎕*IO* = index origin
> ⎕*CT* = fuzz
> ⎕*RL* = seed
> ⎕*PP* = print precision
>)*SYMBOLS* =)*SYMBOLS* + 6

These system variables are set only if their APL SV values are valid (implicit errors are excluded) and only if their values are other than clear workspace defaults. The symbol table size is always set to 6 greater than the APL SV size, but is reported only if the APL SV size is different from 256. The reason for increasing the symbol table size is that Conversion may add six new names to the workspace. Note that ⎕*HT* and ⎕*PP* are not converted. This is because in VSPC tabs and print width are session parameters, rather than workspace parameters. In VSPC, tab and print width settings are valid only for one terminal session and apply to all workspaces loaded in that session. Also ⎕*HT* is origin 1 in VS APL and origin 0 in APL SV. This difference is not handled by Conversion. Latent expression is ignored by Conversion ⎕*LX* is always null. If the user wants to preserve ⎕*LX*, he should assign its value to a workspace variable, making it an easy matter to reassign the value to ⎕*LX* in the converted workspace. ⎕*TT* and ⎕*UL* do not exist in VS APL. The workspace identification (library number and name) is preserved unless invalid for VSPC or renamed by a selective conversion. It is invalid if the library number is greater than seven digits, in which case all workspaces in that library are rejected, or if the name contains delta or underscored characters, in which case the workspace only is rejected. Workspace passwords are preserved unchanged without examination. They had better be valid for VSPC or else the workspace will be inaccessible.

Global Objects

After initialization, global objects are converted in symbol table order. Conversion consists of changing internal formats from APL SV to VS APL formats, and in the case of functions, changing language idioms as well. (The format only option of Conversion prevents idiom conversions; however, in our experience, this is a rarely used option.) An idiom is a usage of the language peculiar to an implementation: mixed output, for example, or carrier return imbedded in character literals, both of which are grammatical in APL SV but are not permitted in VS APL. Users who keep APL functions on a file and read them into their workspace and fix them should be aware of potential conversion problems that Conversion cannot attempt to address.

Group Lists. Global objects converted are group lists, variables, and functions. Group list conversion consists of entering the group name and the name of each group member into the VS APL symbol table, then creating a corresponding group list in the VS APL workspace. The objects named in the group list are not converted at this time, but as they are encountered in the APL SV symbol table.

Variables. Variables are converted by entering their names into the VS APL symbol table and reformatting their values for VS APL. Conversion does not attempt to reformat variables into VS APL synonym links or integer vectors into arithmetic progression vectors (AP vectors). Hence, Conversion is not so frugal of space as it might be. The user can compact his workspace in VSPC by executing its functions, then resaving the workspace. This will take care of synonym links and AP vectors specified within functions. Global variables generated by ι and specified from the keyboard should be respecified by the user to set them up as AP vectors

The characters of a character variable are translated according to whether Conversion thinks the workspace is APL/360 or APL SV. If the workspace is from APL/360, or if the workspace is APL SV but is dumped LEVEL 0, the dump tape appears to be APL/360. In these cases, Conversion translates the terminal control characters backspace, linefeed, and carrier return, and terminal graphics specified in the APL/360 Users Manual into their equivalents in VS APL. All other characters are translated into blot, the illegal graphic character. Loss of meaningful data can result. If the workspace is from APL SV and is dumped LEVEL 1, the dump tape appears to be APL SV. In this case, all possible 256 characters are fully translated into all possible 256 VS APL characters. There is no loss of meaningful data. Therefore it is important that the APL SV installation use LEVEL 1 when it prepares dump tapes for Conversion. (LEVEL 0 is a backward compatability feature of APL SV which allows an APL SV workspace to be transported to an APL/360 installation and vice versa. It mainly concerns a slight difference in the execution stack between the two implementations which is utterly transparent to Conversion. It also causes the writing of an APL/360 tape label, which is important to Conversion. LEVEL 1 writes an APL SV tape label.) The full translation is contained in the CONVERSION workspace distributed with VS TSIO, and in the CONVERT workspace distributed with VS APL. The three terminal control characters listed above and the terminal graphics are translated into their corresponding characters in VS APL. Other characters are translated more or less arbitrarily, but consistently into the remaining VS APL characters. Translation is one for one, and hence, reversible. The important point to be made here is the implicit assumption of Conversion that the user always intends for characters to have their logical meaning and never their physical meaning. By logical meaning, we mean the graphic representation intended by the bit configuration of a character. By physical meaning, we mean the absolute value of the bit configuration. This assumption is not valid for all users. For some, three bytes of characters are the letters 'ABC'. For others, the same three bytes are the core address X'565758'. Translated to its logical meaning, the three bytes become X'414243', obviously a problem where physical meaning was intended. But these translations can be reversed by using the CONVERSION or the CONVERT workspaces.

Conversion does not distinguish between canonical representations and other uses of character variables. Therefore, canonical representation of functions are not examined for idioms. If fixed and executed, they may not work. The user must manually check canonical representations for VS APL validity, or he may use the CONVERT workspace to check them after migration, or he may fix these functions before the workspace is dumped in APL SV to allow Conversion to translate idioms.

Conversion rejects variables if their rank exceeds 63, if any dimension exceeds 2,097,118, or if the product reduction of the dimension vector exceeds 2,097,118, or if any of these parameters is negative.

Functions and Idioms. In format conversion, only the format of functions is converted. For content conversion every line of a function is examined and idioms are converted as well as format. In both options of Conversion, a function is unconvertible if the header is unacceptable to VS APL. Unacceptable headers are those which contain a syntax error or which specify system variables as their result or argument. Examples of header syntax errors are headers containing any primitive function other than the result assignment or the locals list separator semicolon. This includes localized ⎕*TT* or ⎕*UL*, which do not exist in VS APL, and would be tokenized as primitive ⎕, name *TT* or *UL*. Unconvertible functions are reported by Conversion regardless of option. Also, lines too long for conversion are reported regardless of option. The following paragraphs discuss idiom conversions.

I-Beams. Monadic and ambiguous i-beams are replaced by a function name, nominally *IBE*. Conversion adds the function *IBE* to the VS APL workspace before writing it to tape. At execution time, *IBE* operates on the evaluated right argument and simulates i-beams 19 through 29. I-beams 23 and 28 have no equivalent in VS APL, and will cause an *IBE* function error message. An i-beam is ambiguous if its left argument's global value is a function, or is syntactically invalid. It is ambiguous because at execution time, the function name could be shadowed by a local variable of the same name at some unforeseeable level of the execution stack. Conversion assumes that most likely the i-beam really is monadic. If the left argument is a variable, Conversion assumes that the i-beam is dyadic even though the variable could be a function due to fixing at execution time. Ambiguous and monadic i-beam substitutions are reported. Dyadic i-beams have no equivalents and are format converted only. They are reported with a warning flag. Conversion gives the i-beam simulator function a unique name that does not exist in the APL SV workspace. The first choice is *IBE*. If that name exists in the APL SV symbol table, Conversion generates *IBF*, the next name in alphabetical order. If that name too exists, Conversion generates *IBG*, and so on in alphabetical order until an unused name is found. The unique name prevents an accidental replacement of data with the simulator function or an accidental syntax error at execution time. The function is locked, not for proprietary reasons, but so that at execution time, it will behave like a primitive. But this is a problem for users who want to edit the function to supply a value for i-beams 23 and 28. To get around the problem, in the CONVERT distributed workspace is the canonical representation of *IBE*, named *CRIBE*. The user should erase *IBE* in his workspace, copy *CRIBE*, fix it, edit it, then lock it, being careful to give the new function the same unique name.

APL/360 Idioms. Monadic transpose, encode and residue are flagged by conversion. These flags can normally be ignored for an APL SV workspace where the definitions are the same as in VS APL. However, some APL SV workspaces are really APL/360 workspaces migrated to APL SV and have not been edited for idiomatic differences. Owners of these workspaces may want to examine occurrences of these idioms. The differences are:

1. Monadic transpose reverses the coordinates of an array. In APL/360, it reverses only the last two dimensions of an array.

2. The definition of residue now includes negative values in the left argument. In APL/360, only the absolute values of the left argument are used.

3. The definition of encode now includes negative values in the left argument, since encode is defined in terms of residue.

The CONVERT workspace has functions which simulate the APL/360 definitions; these are *OLDENCODE*, *OLDMONTRANS*, and *OLDRESIDUE*.

Mixed Output. Mixed output is replaced with an expression using the format function, according to which paradigm the mixed output statement resembles. Paradigms are:

A; B; C	simple print lists
(⍕*A*), (⍕*B*), ⍕*C*	
(*A; B; C*)	redundant parentheses
((⍕*A*), (⍕*B*), ⍕*C*)	
A; ;B	empty, nonempty print lists
(⍕*A*), ⍕*B*	
A;	final empty print list
⍕*A*	
;A	beginning empty print list
⍕*A*	
;	empty print lists
	(empty line)
(*;A*)	empty, nonempty lists, redundant parens
(⍕*A*)	
(*A;*)	empty, nonempty lists, redundant parens
(⍕*A*)	
(*;*)	empty print lists, redundant parentheses
	(empty line, redundant parens suppressed)

This translation results in an execution time length error or unintended formatting if one print list is an array and the other is a vector or scalar. Therefore, Conversion reports this idiom with a caution flag.

Imbedded Carrier Returns. In APL SV, it is possible to enter carrier returns as characters in a character literal in defined functions by hitting carrier return before closing the quote. This is not possible in VS APL, where the carrier return always signifies the end of the input whether or not the quote is closed. In VS APL, the system variable ⎕*TC* must be used to control the terminal. Conversion translates this idiom to an expression using ⎕*TC* according to the following paradigms (let ⁎ be carrier return):

'⁎'	single character is carrier return
⎕*TC*[1 + ⎕*IO*]	
'. . . ⁎ . . .'	intervening carrier return
('. . .', ⎕*TC*[1 + ⎕*IO*], '. . .')	
'. . . ⁎⁎ . . .'	successive carrier returns
('. .',(n⍴⎕*TC*[1 + ⎕*IO*]),'. .')	n = no. of successive carrier returns

'* . . . ' (⎕TC[1 + ⎕IO],'. . .')	initial carrier return
'. . . *' ('. . .',⎕TC[1 + ⎕IO])	final carrier return

System Variables. ⎕TT and ⎕UL are translated to ⎕ TT and ⎕ UL. They are reported with warning flags because execution will result in a syntax error. ⎕AV is reported with a caution flag. Since the elements of ⎕AV represent different characters, reference to it might have unintended results at execution time.

WSFNS Functions (*DELAY, DIGITS, ORIGIN, SETFUZZ, SETLINK, WIDTH*). These functions are replaced provided that they are locked, two line functions, and match bit for bit the same functions in the APL/360 distributed workspace WSFNS. The replacement performs the equivalent function with the appropriate system variable.

Locked Functions. Conversion idiomatically translates locked functions, but reports severity only. It does not report idioms found to preserve the proprietary nature of locked functions. These functions can represent a special problem because a locked function cannot be edited. The APL conversant user should have an unlocked copy of his function available so that he can correct idiomatic problems. However, there are many APL locked functions which form application packages used by people not conversant in APL. Due to idiomatic differences, these applications can fall apart, leaving the user without recourse. The disruption can be severe. Often, the application writer is no longer available to update the application. The installation should plan to accommodate these users. A suggestion, if the users will permit, is for the APL SV installation to make a copy of these locked functions into special libraries, then from a privileged terminal, unlock the functions. Later, after conversion, a systems programmer skillful in APL can edit these functions in VS APL to correct idioms. After relocking them, the concerned user can copy the cor. ted functions into his workspace. The Appendix contains UNLK, a privileged function, which unlocks ' functions in a workspace. LK locks all functions in a workspace. These functions can be executed only from a privileged terminal.

Damaged Workspaces

An APL SV WSLIST listing will show damaged workspaces, but it will not always agree with Conversion's report of damaged workspaces. In particular, the WSLIST may report a workspace as damaged, while Conversion will accept the same workspace with no report of damage. This result usually occurs where forward-backward pointers in the workspace do not agree. The APL SV utilities are sensitive to conflicting forward-backward pointers whereas Conversion is not. On the other hand, Conversion can reject a workspace as damaged while the WSLIST listing does not mark the workspace as damaged. This usually occurs where the printname of a group member, which has been erased, is damaged. The APL SV utilities are not sensitive to this sort of damage whereas Conversion is quite sensitive. We recommend that users pay special attention to groups in their workspaces to avoid this confusion.

PROCEDURES

The installation must select a suitable strategy for migration. Also, it must plan for several iterations to migrate the bulk of the installation, the first iteration rejectees, the recalcitrant, and finally the stragglers. The old installation must be kept available until migration is complete.

You can let Conversion enroll users as they existed in APL SV or you can use select conversion to reassign account numbers and enroll users, or you can manually enroll users and use Conversion to merely convert workspaces. If the VSPC installation is replacing an APL SV installation, use the first strategy. If the APL SV installation is being added to a VSPC installation that already has other users, the second will have to be used because account numbers may conflict. Finally, if you manually define users, you can simplify your task by taking advantage of VSPC's 'model' profile facilities. If you renumber accounts, you will also have to rename the users' associated TSIO data sets.

Conversion Definition

1 Canvass the users. Publish the shared variable size that you plan to establish and the default number of file buffers. Using the VSPC DEFINE command, predefine those users for whom Conversion size and buffers are inadequate.

2. Clean up the APL SV installation. Have the users do the following:

   1. set default comparison tolerance to 1E-13 in all appropriate workspaces.

   2. change their logon and workspace passwords to VSPC usable passwords.

   3. change workspace names to VSPC acceptable names.

   4. get rid of damaged workspaces by the following sequence:

      a. Check all groups to ensure that all printnames are valid (no garbage displayed). If a member has an invalid name (usually it will be of an erased object, no longer in the workspace) reform the group either supplying a valid name or omitting the invalid member name. Resave the workspace.

      b.)*CLEAR*
)*COPY* workspace
)*WSID* workspace
)*SAVE*

      Notes: Preserve the source workspace's index origin, print precision, symbol table size, random link (where it matters), and comparison tolerance. The installation may perform this cleanup from a privileged terminal, working from a WSLIST.

   5. verify their function headers for VS APL acceptability.

   6. repair functions with long lines apt to be deleted.

   7. if ⎕*LX* is not null, assign its value to a variable so that ⎕*LX* can be easily reestablished in the VS APL workspace.

The installation should reassign account (signon) numbers for users whose signon numbers exceed seven digits. This cleanup will save Conversion iterations.

3. Take a full dump of the installation being sure to use LEVEL 1. Also, use WSLIST option so that you can identify the workspaces on each reel. Back up the TSIO files so that they can be selectively restored later in case of user error. System utility IEHMOVE (with COPY VOLUME) is suggested.

4. At this point, you have the choice of segmenting the Conversion-Copy runs or trying to do the whole job in one iteration. If the dump produces more than two reels, we recommend segmenting the task, reel by reel. The disadvantage of segmenting is that the first reel must be run first before each reel to be converted (the first reel has the directories, which Conversion saves on a temporary data set until the target workspaces are encountered). To segment the task, run a fill conversion on the first volume. Reply cancel to Conversion's request for the next volume. Then run the VSPC Service Program using COPY command with NOREPLACE option on the Conversion produced 'copy' tape. This will enroll those users on the first dump reel in VSPC. It is important to use NOREPLACE instead of REPLACE because VS APL is installed in VSPC with distributed workspaces in libraries 1 and 2 that could be replaced with identically named workspaces from APL SV. However, you may want to replace 1 NEWS in VS APL with your current 1 NEWS in APL SV. If so, this will have to be done in an independent selective conversion-replace copy run. If you are migrating to a DOS VSPC installation, be sure the 'copy' tape has a DOS label. Migrate the next batch by running resume conversion specifying the first workspace on the second reel of the dump tape. Mount the first reel so that Conversion can retrieve the directories, then the second when requested by Conversion. Run copy with NOREPLACE on this 'copy' tape. Repeat for all reels.

5. This completes the first iteration. Now, migrate the rejected users or workspaces from the first iteration. The conversion report distributed to the users lists rejected workspaces and accounts with reasons for the rejection. Collect a list of rejected workspaces with reasons from the users. If rejected because of invalid WSID, run selective conversion using rename facilities on those workspaces. The directory reel will not be needed if the first iteration successfully defined the rejected users. You need input to Conversion only those reels containing the selected workspaces. You can avoid mounting the directory reel by predefining users whose accounts were rejected. If a workspace was rejected because of damage or tape I/O error, selectively dump the APL SV workspace (after cleaning up damaged workspaces). Run full conversion on the seldump tape.

6. Migrate the recalcitrant workspaces. You will have to analyze the reasons for rejection and correct them. If locked functions are the problem, copy the user's workspace into a special library in APL SV, then from a privileged terminal, unlock the problem functions. Dump and convert the workspace again. Then, most likely with the user's data, procedures, and cooperation, edit the unlocked functions to get them working right. Lock the functions and copy them into the user's workspace. Other problems will have to be solved on an ad hoc basis.

7. Using the VSPC ALTER command, modify the profiles of those users for whom Conversion definition was inadequate.

Renumbering of Accounts

In this strategy, selective conversion is always used to renumber accounts. The Installation Reference Manual says that up to 100 workspaces or accounts may be selected for conversion in one run. This is not quite true; it is a worst case figure. Conversion uses an internal selection table with variable length entries.

If selection involves only the renumbering of libraries, with no selection of specific workspaces, up to 366 libraries with all their workspaces may be selected and renumbered in one run. Procedure remains the same as above, except that you never use full and resume options.

You will need to keep a cross reference of old to new numbers so that the TSIO data sets can be renamed to match the new accounts. Renaming is a task for the VS TSIO administrator.

Manual Definition

In this strategy, all users are enrolled in VSPC manually by a system administrator using VSPC's DEFINE command. The UPR's created by Conversion will be ignored by VSPC. First, define one or more model users. Then reference the appropriate model profile when defining the subsequent users. If account numbers differ, use selective conversion to renumber accounts to match the assigned VSPC account numbers.

Use of Select Cards

Select cards are used to select specific workspaces for conversion and to renumber and rename workspaces. With one exception, explained later, select cards may be entered in any order, however, we recommend that they be entered in the order shown on the WSLIST listing for better efficiency of the Conversion program. Also, be careful to avoid specifying a workspace (or library) that does not exist on the dump tape. This error causes Conversion to sift through the entire dump (which may be many volumes) looking for the nonexisting workspace before it quits. Conversion does not report unfound workspaces. If there are no errors in specifying workspaces, Conversion quits as soon as all the specified workspaces have been encountered on the tape.

How to write select cards is explained in the Installation Reference Manual for VS APL. As mentioned before, considerably more than 100 select cards can be entered in one run, depending on the complexity of each select card. Not explained in the Installation Reference Manual is how to select all the members (workspaces) of a library and rename only some of the members. Suppose, for example, that library 5 contains in WSLIST order:

```
WS1
WS2
WS3
```

WS3 must go to library 2 in VSPC and be renamed OLDWS3 to avoid replacing a WS3 already in library 2. This can be done with two select cards as follows:

```
SELECT 5 (2) WS3 (OLDWS3)
SELECT 5
```

The trick is to place the renaming/renumbering cards ahead of the library selection card. Use this trick carefully, if the library selection card is placed before the rename cards, the renaming will be ignored. Further, the renaming cards will act like an erroneous selection card.

GLOSSARY

AP Vector. Object in VS APL workspace generated by any expression using ιn and consisting of integers. Consists of three elements; the initial value, the increment, and the number of integers.

Byte Integer. A byte of character data that is not intended to have graphic or terminal control character meaning within the APL workspace. It may have graphic meaning outside APL, for example, a printable EBCDIC character such as hexadecimal byte F1.

Control Interval. Unit of data used by VSAM to transfer data between programs and physical storage devices. It may contain one or more logical records. A virtual physical record.

COPY. In APL, refers to action or result of the)COPY command. In VSPC, refers to action or result of the VSPC Service Program COPY command.

Directory. A data set which contains definitions of accounts and pointers to other data sets which contain workspaces and files belonging to each account and/or definitions of the workspaces and files. An index of accounts and files.

Execution Stack. A section of a workspace in which APL statements are temporarily stored for analysis and execution.

Free Space. Space available in a workspace for temporary and permanent objects. If used for permanent object, it is no longer part of free space.

Format. The organization and contents of an object in the workspace.

Idiom. A usage of language peculiar to an implementation of the language.

M-Entry. Storage in a workspace allocated to a workspace object or printname.

NOREPLACE. A VSPC COPY option (and the default). It means that a workspace will be added during a COPY to the VSPC library only if its identification does not match any existing workspace identification. Prevents inadvertent replacement of workspaces.

Object. The value or definition in an APL workspace which defines an APL item, as opposed to its name. For example, a variable consists of a name and an object pointed to by its name which contains the value(s) for the variable.

Paradigm. A model statement illustrating proper grammar or idiom.

Predefinition. Manual enrollment of an account in VSPC before conversion of the account to circumvent account definition by the Conversion program.

REMOVE. A VSPC COPY option. If the COPY source account is locked (logically deleted) this option prevents the account from being copied to the output VSPC library.

REPLACE. A VSPC COPY option. Permits a workspace to replace a workspace with the same identification in the output VSPC library.

R13 Stack. A section of an APL workspace, at the high address end, reserved for temporary storage by reentrant interpreter routines. So called because it is addressed by general register 13 in interpreter routines.

Symbol Table. A section of an APL workspace which contains a pointer to every name and a pointer to every object in the workspace.

Synonym Link. A VS APL workspace object which represents the values of another workspace object. For example, if A has a value, then assigning A to B generates a synonym link as the object for B, which contains a pointer to the object for A.

Terminal Control Character. A byte representing a terminal function, such as carrier return, backspace, or line feed. Opposed to terminal graphic, q. v..

Terminal Graphic. A byte representing one of the elements of the APL character set.

Token. An element within or associated with an object which syntactically classifies or represents the object or element of the object. For example, each primitive function is represented in an APL statement object by a unique token which both represents the function and classifies the syntax of the function.

APPENDIX

```
UNLOCK FNS )1977 11 7 15 22 22 DUMPED BY USER 538852



     ∇ LK;I;J
[1]    I←2⊥ 8 128
[2]   L1:→(18 19 ∧.≠1↑J←, 256 16777216 ⊤2⊥4,1↓I)/L2
[3]    16⊥4,1↓J,536870912
[4]   L2:→(>/I←I+ 0 8)/L1
     ∇

     ∇ UNLK;I;J
[1]    I←2⊥ 8 128
[2]   L1:→(18 19 ∧.≠1↑J←, 256 16777216 ⊤2⊥4,1↓I)/L2
[3]    3⊥4,1↓J,2⊥(,(32⍴2)⊤2⊥4,1↓J)∧¯2⌽¯32↑31⍴1
[4]   L2:→(>/I←I+ 0 8)/L1
     ∇
```