THE SPECIAL CHARACTER SET

No. 7

APL SIG

DECUS

Printed in the U.S.A.

EDITORIAL

APL, POLITICS AND RELIGION

Douglas Bohrer
Bohrer and Co.
903 Ridge Road Suite 3
Wilmette, IL 60091

Most of you who are reading this are probably APL zealots. Your past experiences have convinced you that APL is the best thing since sliced bread. But many of you are probably very lonely in that belief. The way to spread your belief in APL to others is to engage in those two tricky areas of human activity, politics and religion.

The established religion of most data processing shops is probably Basic, Fortran or Cobol. However, unlike Jehovah, none of these languages is a jealous god. Your approach should be to convince the political powers of your shop that APL would make a nice minor diety in their religion.

While we all know APL is good for everything, it is probably threatening and impolite to say so. The established people in your shop will become alarmed if the great god Basic is threatened with dethroning. They will fight you to the corporate equivalent of death, employment termination, in order to preserve their positions as high priests of data processing. Your return-on-investment analysis will be disapproved at the very least.

The situation is far less hostile if you say APL is good for something the users want want but the established languages don't do well. For example, decision support management summaries or an information center for your economists or engineers are businesses that the great god Cobol is not too good at. The users requesting the support are bottomless pits eating resources that could be better used for more traditional things like transaction processing system maintenance. The power people in your company would be thrilled to support any suggestion that will slow the endless stream of requests and complaints from these hard-to-please users. You could be a political hero for suggesting an APL solution for this type of problem.

Once you have one APL application in house, you are in a good position to expand APL use. If you succeed with the first APL project, your credibility will be enhanced. Using APL for a new project will require no increase in spending because you already have paid for the language software. Saying APL is good for lots of things becomes less threatening and radical. Little by little the great god Fortran loses new projects and influence.

If you're going to use this strategy, don't tell anyone what you're doing. Let the power people of your shop, secure in their beliefs, remain undisturbed by the potential of an APL revolution. Once the benefits are obvious, they will begin to think expanding APL use was at least partially their own idea. The established religion will have been changed through politics, and results.

ADVERTISING?

Douglas Bohrer
Editor, The Special Character Set

In cooperation with the editors of the BASIC and UNIX newsletters, I have started an investigation of paid advertising in DECUS newsletters. In the last year I have been offered three paid ads, two from (gasp) DEC. I had to turn them down. The current DECUS commercialism policy does not allow commercials, paid or otherwise.

A change in policy might allow this publication to become free again, at least in the USA. The way this might work is that the Executive Board would grant an exception to the commercialism policy for newsletter ads. I would like to sell full page ads only for about $400-$600. The ads would be paid for in advance through the DECUS Library, if they agree. There would be a section in the back for paid ads. The rest of the newsletter would still adhere to the current guidelines excluding commercialism. Three or four full page ads would cover the cost of printing an issue.

A major problem with paid ads is the revenue split for foreign chapters. As far as I can tell, most if not all DECUS members with addresses outside the US are getting newsletters printed and mailed at their chapter's expense. This is how DECUS EUROPE is handling their APL subscriptions. There seem to be two possible solutions for the problem. Since the ads would be in a separate section at the end of the newsletter, they could be removed before printing outside the US. The other possibility is that the split could be based on each chapter's percentage of the total circulation. Currently, the US has about 780 subscribers. I think Europe has about 200, Canada 40 and GIA 40, but I am not too confident about these figures.

Obviously, nothing can be done without lots of consultation with everbody who's anybody in DECUS. The Executive Board would have to approve the policy change.

What do you think of the idea? To give me a short answer, fill in question 6 of the survey in this issue. Or write me a letter I can publish. Or write me a letter I can't publish.

THE SPECIAL CHARACTER SET
March 30, 1984 No.7

The Special Character Set is the official publication of the APL Special Interest Group of the Digital Equipment Computer Users Society. Unsolicited manuscripts are welcome.

For subscription information and an application, contact:

Membership Services
DECUS
249 Northboro Road BP02
Marlboro, MA 01752

STYLE SHEET

by Douglas Bohrer
Bohrer and Co.
903 Ridge Rd Suite 3
Wilmette, IL 60091

We look forward to your contribution to The Special Character Set. Your cooperation on a few matters of style would help get the newsletter out on time. The newsletter staff is all volunteer and the time you save us is appreciated.

Set-Up

Basically, your contibution should look like this article. Use black ink. Type your text single spaced. Use only one side of each sheet of paper. Skip one line between paragraphs. Set your margin at 4 3/8 inches wide. For type with 10 characters per inch, you would have 43 characters per line. For type with 12 characters per inch, you would have 52 characters.

Titles

Please start the title of your article on the left margin using all capital letters. Skip one line, then put your name, title, firm and mailing address. (You may omit your title, firm and mailing address for reasons of modesty, privacy, shame or whatever.) Sub-heads in your article should begin at the left margin with a blank line above and below them.

Letterhead will not be reproduced. Drawings should be either 4 3/8 or 9 inches wide.

Commercial Guidelines

The newsletter will follow strictly the guidelines related to the non-commercial nature of DECUS. If you have any questions on these, please contact the newsletter editor or the DECUS office.

CONTENTS

Local Variables

APL IN CINCINNATI

Susan M. Abercrombie
APL SIG Symposium Coordinator
Ventrex Laboratories, Inc.
217 Read Street
Portland, ME 04103

For the Cincinnati symposium, the APL SIG is short on quantity, but not on quality.

An introduction to APL will highlight the reasons for using APL and the application areas for which the language is especially well suited. This session will serve as a tutorial for novices, without regard to the particular system available.

A panel discussion on the various free APL interpreters available through DECUS (APL-11 for RSTS, RSX, and RT; SCI-APL for VAX) will describe these products and give users an opportunity to discuss them with experts. If you have questions about the features or the limitations of these APL versions, here is your chance to get answers (or, at least, commiseration).

A third session will offer a technical description of the effort now underway to write a new PDP-11 APL, using Whitesmith's C.

RSTS users should also note the session scheduled through the RSTS SIG, describing system management tools implemented in APL.

We expect to have a SIG suite, equipped with a portable hardcopy APL terminal, with access to APL on a VAX in the exhibit hall. Stop by to try out this uniquely useful language.

See you in Cincinnati!

[Editor's Note: The terminal for our hospitality suite is an LA12-AB with APL keycaps provided by Bruce W. Hunter, DEC's marketing guru for the Correspondent. The LA12 comes standard with APL; keycaps are slightly extra.]

HELP WANTED: APL SIG STEERING COMMITTEE

Douglas Bohrer
Editor, The Special Character Set

Nobody has applied to fill the open position in the APL SIG Steering Committee. The position is still open. It is particularly important to fill it. Since its founding, the APL SIG has been under pressure to merge with another SIG. The fewer visible people we have on the SIG Steering Committee, the more intense this pressure becomes. Our influence as a group on both DEC and DECUS will decline significantly if we loose our independent status. Please try to find a way to help us out. If you want to talk about it, call Doug Bohrer at 312-251-9449 or Larry LeBlanc at 312-860-8181 or Sue Abercrombie at 207-773-7231.

Until we can find someone, the Steering Committee has decided to reorganize. Sue Abercrombie will become Library Coordinator. She will also keep responsibility for overall coordination of symposia offerings. Scheduling and representation to the DECUS Symposia Committee will be pooled with two other SIGS.

NEWSLETTER PRICE FALLS

Douglas Bohrer
Editor, The Special Character Set

The APL SIG had over 70 percent of its US members subscribe to its newsletter. As a result, the SIG had a "paper profit" margin of over 50 percent of sales revenue. The movers and shakers of the DECUS publishing community, wowed by this unexpected result from the smallest SIG in DECUS, have decided to let us lower our yearly price to $5 starting next year. Most newsletters lost money this year and will be increasing their prices to cover the shortfall.

The problem of reprints of this years issues, numbers 6 and 7, has not yet been resolved. Hopefully, I can publish back issues through the DECUS Library and DECUS Store as I have the collection of issues 1-5. (The collection of issues 1-5 is still available from the DECUS Library for $5, order number "APL NEWSLETTERS".)

Language Enhancement

Proposal for a Dyadic Execute in APL

by Homer A. Hartung
Philip Morris,
USA Research Center
P.O. Box 26583
Richmond, Va. 23261

A few years ago I started using a simple function named DEFAULT. In APLSF on our TOPS-20 the listing is:

```
     ∇ ZK←XH DEFAULT YJ;⎕TRAP;⎕ERROR
[1]  ⎕TRAP←'ZK←YJ',⎕AV[102 100],'→0'
[2]  ZK←⍎XH
     ∇
```

In the first line a trap is defined such that the right argument is returned if an error is encountered. (Quad-AV 102 100 is the carriage return and line feed in APLSF.) In the second line an execute of the left argument is attempted and this will return a result unless an error occurs and the trap prevails.

To see how this works, consider the task of setting up formatted tables. Suppose we want to be able to put a title at the top but we don't want the program to stop on an error if we forget. This situation is handled very easily by using:

```
TL←'TITLE' DEFAULT 'UNTITLED'
```

where TL is localized in the formatting routine. By default TL is always defined for later use. If TITLE has been defined by the user, TL will take that value. If not, TL takes 'UNTITLED' by default.

DEFAULT is also useful when doing calculations where we want protection from inadvertant domain errors. In such cases, I have adopted the convention that the variable named 'NA' (not applicable) should have a numeric value which is easily recognized as such. The following line can be used:

```
LOGX←'10⍟X' DEFAULT 'NA' DEFAULT ¯9999
```

A program with this line in it will run smoothly even if X happens to have a zero or negative value. If the numeric value of NA was not defined in the workspace, LOGX will be assigned -999999 for NA.

The DEFAULT function has proved so useful that it is now extensively imbedded in many of our applications programs. This suggests that there should be an APL operator to do the same job. The "execute" is a logical choice since it is currently used only as a monadic operator. It would be consistent with the spirit of APL if it were defined for both monadic and dyadic use. In the dyadic case, the left hand argument would be the default. This is consistent with the current specifications for the monadic operation.

If the proposed dyadic operations were defined, my function that formats tables would use the line:

```
TL←'UNTITLED'⍎'TITLE'
```

Similarly, my functions to do logarithms would contain:

```
LOGX←(¯999999⍎'NA')⍎'10⍟X'
```

APPLICATIONS

APL APPLICATION IN INSURANCE

by James J. Skinner Jr.
Assistant Secretary - Treaty Systems
North American Reinsurance Corp.
100 E. 46th Street
New York, New York 10017

INTRODUCTION

Based on the quantitative nature of insurance, which evaluates and measures risks statistically in large numbers, APL becomes a perfect application because of its mathematical and interactive characteristics.

APPLICATION

From an underwriting and marketing point of view, APL can serve as a valuable management resource and tool. In insurance, statistical information is the backbone for the ratemaking, pricing, selection and evaluation of the business, and is utilized extensively by the Underwriting and Marketing areas. With utilization of APL, this information can be custom tailored and printed on final reports to fulfill the individual needs of these insurance areas. This is especially evident when a "what if?" or "as if" situation arises. With the dynamics of APL incorporated, management decision can be aided by the timeliness of the immediate turnaround of this statistical information.

Imagine the following scenario:

A data collection function has gathered loss information into a (10 x 5) matrix of data called MAT. (First dimension is ten losses for each year, second dimension is five loss years.) Another function has collected fives years of earned premiums into a (20 x 5) matrix of data called PREMS. (20 premium figures for each year.)

Variables are:

MAT — Losses
PREMS — Premiums

Management realizes this data is already available and then tries to analyze it. The question arises "What would the Total Incurred Loss be for each loss year if a certain deductible or retention existed?"

Functions and Idioms

The following interactive program would be written:

```
      ∇EXCESS[□]∇
    ∇  R←EXCESS MAT
[1]    'ENTER THE  RETENTION/DEDUCTIBLE ------ '
[2]    ,'CI13'$R←+/[1]MAT-□
    ∇
```

This program would subtract the deductible from each loss then add up all the excess amounts and summarize by loss year. Managers can now sit down and analyze data based on any deductible they choose and get on immediate summation on a terminal. Management now asks "What would be loss ratios look like when I select different retention levels?" Responding immediately, this function is written:

```
      ∇LOSSRATIO[□]∇
    ∇  FREMS LOSSRATIO MAT
[1]    ,'F13.2'$100×(EXCESS MAT)÷+/[1]FREMS
    ∇
```

As you can see, someone can now sit at a terminal, enter various deductible levels and evaluate total losses incurred and loss ratios just by entering a deductible. APL is quite a powerful tool!

This application was only two lines and one line respectively, obviously a simple task for APL. However, it is only the start of the thousands of applications which is only limited by the imagination of the User/Programmer. The beauty of this is that the User/Programmer may be the manager himself. This is one reason APL application can be so easily applied since the user requirements are programmed and/or used by the insurance managers themselves. They know the business so they use the applications.

Management can always get data, but to manipulate it themselves in any form provides an imaginative analysis of the insurance business. APL language *is* the instrument that makes this happen.


```
.R APL
TERMINAL..LA

WELCOME TO APL/11 V1.00
CLEAR WS
      )READ CJSW
      )FNS
CJSW DJSW
      )VARS

      ∇CJSW[□]∇
     ∇ I CJSW L ;R
[1]    23⊥36
[2]    R←ϕ,(16ρ2)⊤21⊥1
[3]    R[I+1]←L
[4]    23⊥36
[5]    22⊥(16ρ2)⊥ϕR
     ∇
      ∇DJSW[□]∇
     ∇ DJSW
[1]    23⊥36
[2]    (,(16ρ2)⊤21⊥1)/ϕ0,⍳15
     ∇
      DJSW
14 13 9
```

HACKING IN APL-11

Douglas Bohrer
Bohrer and Company
903 Ridge Road Suite 3
Wilmette, IL 60091

This article is a collection of the latest tricks I've found for APL-11 version 1. The first three may work in APL-11 version 2, but have not been tested. The fourth is version 1 only. The fifth is a TSX+(TM) trick. None of these features are documented in the APL manuals. Experiment before you depend on them.

Peek and Poke

There are three dyatic I-beam functions which implement peek and poke. For either peek or poke you must first position the peek/poke address pointer. Executing

```
   23 .IB N
```

sets the pointer at address N. Then either a peek or a poke at the address is possible.

To peek, execute

```
   21 .IB N
```

where M is the number of words to be looked at. After the peek, the peek/poke address pointer points to the address N+Mx2. If you want to poke location N after peeking, you must reposition the pointer before you poke.

Poking is much the same as peeking. First set the address pointer, as above, to the address N. Then do the poke with

```
  22 .IB IV
```

where IV is a vector of numbers representable as 16 bit integer values. After the peek the address pointer points to 1+N+2x⍴IV.

I have not tried this feature extensively. The two examples are CJSW and DJSW. The function CJSW changes bit settings in the RT-11 Job Status Word. Variable I is a vector of bit numbers to be changed. The right hand argument L is the new settings of the bits of I. For example

```
  0 15 CJSW 0 1
```

will turn off bit 0 and turn on bit 15, leaving other bits as they were.

Function DJSW displays the bit numbers of set bits.

I tried to use these functions to set up a single character activation mode for input inside APL. I couldn't get it to work. Maybe you can. If so, let me know.

Byte I/O for the Terminal

There are two features that can be used for byte I/O to the terminal. The first uses the quad-arrow with channel 0, the terminal channel. An equivalent method uses a dyadic I-beam function.

When the interpreter comes up in a clear workspace, channel 0 is open to the terminal. Output of arbitrary type can be sent to the terminal using channel 0. For example

```
DUMMY_0 .0Q[1]14
```

will send a shift in to the terminal. However, whatever is sent is followed by 6 spaces.

The equivalent I-beam output would be

```
20 .IB 14
```

As above, whatever is sent is followed by 6 spaces.

This feature can be used in a startup file under TSX+(TM) to set up smart terminals. Function TERSET sets up my Concept AVT(TM). QAV is a matrix because as a vector it will not survive a)WRITE)READ storage sequence correctly.

```
      ∇TERSET[□]∇
      ∇ TERSET ;I
[1]    I←15,ESC '←6[0U/'
[2]    I←I,ESC '←×101L/'
[3]    I←I,(ESC '←36[0U/'),ESC '←×101H/'
[4]    I←I,(ESC '←7[0U/'),ESC '←×103L/'
[5]    I←I,(ESC '←37[0U/'),ESC '←×103H/'
[6]    I←I,ESC '∨α'
[7]    I←I,ESC '∧2'
[8]    I←I,ESC '←×123L'
[9]    I←I,ESC '←12H'
[10]   20⌶I
      ∇
      ∇ESC[□]∇
      ∇ R←ESC C
[1]    R←27,31+(,QAV)⍳C
      ∇
      ⍴QAV
9 14
      QAV
 ¨)<≤=>]∨∧≠⍕,+
./0123456789([
;×:\¯⍺⊥∩⌊∈_∇Δ⍳
∘'□|⍤○*?⍴⌈~↓∪⍵
⊃↑⊂←⊢→≥-◇ABCDE
FGHIJKLMNOPQRS
TUVWXYZ{¬}$ ¨)
<≤=>]∨∧≠⍕,+./0
123456789([;×:
```

32 Bit Integers in Files

If you are reading or writing files for other languages, you may wish to use 32 bit integers. However, since 32 bit integers are not a storage type in the APL-11 file system, they must be faked using 16 bit integers.

To write 32 bit integers, you first convert an array of numbers to the equivalent pairs of 16 bit integer values. Then write them to the file as storage type 2 integers. If XX is a vector of integer values then

```
FP_0/8.0Q[2]C3216 XX
```

would write them to the file on channel 8 so that they can be read by another language as 32 bit integers. The function C3216 does the numeric conversion.

To read 32 bit integers, first read twice as many 16 bit integers. Then use function C1632 to convert them to the 32 bit values. To read fifteen 32 bit values from channel 8 into vector I32

```
I32_C1632 8.IQ[2]30
```

If you are doing this sort of thing, use double precision to get all the significant bits.

Localizing the Origin

The programs C1632 and C3216 are used in both origin 0 and origin 1 workspaces. In order to localize the origin in APL-11 version 1, you first save the result returned from executing an)ORIGIN command. In both C1632 and C3216 this is done on line 1. The returned value will be either 'WAS 1' or 'WAS 0'. At the end of the function you execute another)ORIGIN command to put the origin back

```
      ∇C3216[□]∇
      ∇ R←C3216 N ;CV
[1]    CV←∈')ORIGIN 1'
[2]    R←((⍴N←,N),2)⍴0
[3]    R[;1]←⌊N÷65536
[4]    R[;2]←N-R[;1]×65536
[5]    R[;2]←R[;2]+((32767<R[;2]),[1.5]¯32768>R[;2])+.×¯65536 65536
[6]    CV←∈')ORIGIN ',4↓CV
      ∇
      ∇C1632[□]∇
      ∇ R←C1632 N ;CV
[1]    CV←∈')ORIGIN 1'
[2]    N←((⌊0.5×⍴N),2)⍴N←,N
[3]    R←N[;2]+(¯65536×(N[;2]≥0)∧N[;1]<0)+65536×N[;1]+∨/N<0
[4]    CV←∈')ORIGIN ',4↓CV
      ∇
```

SETSIZ Under TSX+(TM)

Using the SETSIZ program can get you about 1K words extra workspace under TSX+ version 4.0. The dialogue is as follows:

```
R SETSIZ
*APL/T:n
*APLDP/T:n
^C
```

where n is the octal size that works for the hardware type of the APL-11 you're using. For APL06 I've used n=74 and for APL07 n=72. For other versions, experiment. If the number is too big you get SYSTEM ERROR messages when you try)READ commands or defining functions.

Mission Impossible

For all of the above tricks, the standard "Mission Impossible" warning applies: if any of your force is caught or killed, the APL SIG will disavow all knowledge of your actions.

new
11-631

APL-11/RT-11 Magtape/TSX Shared Files

Version: October 1983

Author: Doug Bohrer, et.al., Bohrer & Company, Wilmette, IL

Operating System: RT-11 V4, TSX-PLUS 2.2

Source Language: APL, C, FORTRAN IV, MACRO-11

Memory Required: 56KB

Special Hardware Required: FIS or FPP are recommended for APL.

This is a collection of several unrelated programs. The following is a brief description of the programs to be found on the tape. **The floppy diskettes include items three through six only.**

1. Very fast tape backup and restore system. Backup tape is blocked at 10kb per block and has its own directory. Files can be selectively backed up or restored. Tape writes are double buffered. Written in DECUS `C'. SAV files are included in the distribution.
2. Programs to read IBM and other foreign tapes using RT-11 V4 SYSLIB in FORTRAN and `C' with SAV files included.
3. APL-11 V1, considered more reliable than APL-11 V2. SAV files only. Sources not available.
4. APL utilities include file handling, fancy character bar graphs, print formatting aids and counting type computation functions. Multiple linear regression can use either workspace variables or files for data.
5. FORTRAN/C file handling filter programs to set up APL files, match records from two input files on a key field. SAV files included.
6. FORTRAN subroutines to handle TSX-PLUS shared files wih random access fixed length records. Buffering and locking/unlocking blocks is automatic. Records can span blocks.

Note: **Please note that the Floppy Diskettes (KB) contain a subset (items three through six) for floppy systems only.**

Restrictions: Shared file routines use TSXLIB (DECUS No. 11-490) which is not included with this package. The sources for APL V1 are not included.

Associated Documentation: FOR APL-11 documentation order the APL-11 V1 RSTS/E Digital manual: AA-5076A-TC from your Digital Sales Representative.

Complete sources not included. Documentation may or may not be included on the magnetic media.
Media (Service Charge Code): Floppy Diskettes (KB), 600' Magtape (MA)
Format: RT-11
S1176
830712/

Keywords: APL, Shared Files, Multiple Linear Regression
Category Index: 7, 17
Operating System Index: RT-11

STATE INDICATORS

WRITING A PDP-11 APL IN C

Douglas Bohrer
Bohrer and Company
903 Ridge Road Suite 3
Wilmette, IL 60091

The following is a copy of the slides I used for my talk in Las Vegas. I promised at that time to publish the slides in this issue. Since then some of the design details have changed. I will give an updated presentation in Cincinnati.

```
------
WRITING A PDP-11 APL IN C
       Preliminary questions
       Mapping workspace to disk
       Interpreter and its tables
       Calling foreign language routines
       Issuing EMT's directly from APL
------
WHY THE PDP-11?
       APL-11
------
WHY C?
       Portable
       High level language
       Fastest execution speed of the high level languages
------
WHY WHITESMITHS C?
       Commercially supported product
       Available for RT RSX RSTS/E PRO
       Issue EMT's direct from C
       Error handling _when _raise feature
------
WRITING A PDP-11 APL IN C
       Preliminary questions
>>>>   Mapping workspace to disk
       Interpreter and its tables
       Calling foreign language routines
       Issuing EMT's directly from APL
------
MEMORY MANAGEMENT ALTERNATIVES
       Map workspace to disk, demand page to memory
              High overhead
       Read objects to memory on demand
              Objects must fit in memory
       Entire workspace in memory
              Workspace size is limited by address space
              Interpreter is heavily overlaid to gain more workspace
------
MY CHOICE: MAP TO DISK WITH DEMAND PAGING
       No limit on object sizes
       Few or no overlays
       Can look like "big machine" APL
       Accept speed penalty
------
IMPLEMENTATION
       Workspace mapped to multiple disk files
              Each ws file is mapped to an address segment

              FILE1.AWS   8 blocks       0   0777
              FILE2.AWS   8 blocks   01000  01777

       Workspace addresses use 27 bits of a 31 bit address space
              Allows for a 134 million character workspace
------
ADVANTAGES OF MULTIPLE FILE ACTIVE WORKSPACE
       Use of multiple volumes to get more space/speed on small systems
       Shared read-only access of commonly used functions
------
MAP USER DISK FILES INTO SAME 31 BIT ADDRESS SPACE
       User file addresses range from 01 000 000 000 to 017 777 777 777

       channel 3        01 000 000 000   to    01 777 777 777
       channel 4        02 000 000 000   to    02 777 777 777
       channel 5        03 000 000 000   to    03 777 777 777
       .........
       channel 12      012 000 000 000   to   012 777 777 777
```

WHY MAP USER DISK FILES?
Same routines handles all disk accesses
Buffers shared by workspace and user disk files
Allows files to be treated as APL variables if desired

WRITING A PDP-11 APL IN C
Preliminary questions
Mapping workspace to disk
>>>> Interpreter and its tables
Calling foreign language routines
Issuing EMT's directly from APL

INTERPRETER INTERNALS

text entered

CHARACTER TRANSLATOR

internal character representation

LINE PARSER

internal code

CODE INTERPRETER

results

CHARACTER TRANSLATOR
Depends on the terminal character set
TT
Bit paired
Key paired

LINE PARSER
Manages conversion to op-codes and symbol table indexes
Makes symbol table entries for new names and literals

CODE INTERPRETER
Executes internal code
Stacks pending operations caused by)] ;
Maintains state indicator for function calls
Localizes variables

CODE INTERPRETER STATES
Looking for right hand argument
Looking for function
Looking for left hand argument
Looking for line end

ADVANTAGES OF APL OBJECTS AS INTERNAL INTERPRETER TABLES
Same routines manipulate both internal tables and user objects
Tables are part of the workspace saved
Tables may be interactively displayed and altered
Complex internal functions can be built in APL

⎕NAMES OBJECT NAMES
storage type: 8 bit character
matrix: one row per literal or named object
collumns:
1-16. 16 character object name. init to blanks
note: Literals have blank names

⎕SYMB SYMBOL TABLE, LINK BETWEEN ⎕NAMES AND ⎕STRUCT
storage type: 32 bit integer
matrix: one row per literal or named object
collumns:
1. Object type: nil fn, mon fn, dy fn, amb fn, var, label, lit
op code, value error
2. Index to ⎕STRUCT (nix if value error or op code)
3. Index to previous local/global entry in ⎕LOCAL
(nix if global entry)

⎕STRUCT OBJECT STRUCTURE TABLE
storage type: 32 bit integer
matrix: one row per object
collumns:
1. Storage type: 8 bit character, 32 bit real, 64 bit real;
signed/unsigned 8 bit or 16 bit integer;
32 bit integer; index object; function;
system/user reserved
2. Index to ⎕SPACE (nix if a scalar or null)
3. Index to ⎕SYMB (nix if not in ⎕SYMB)
4. Index to pending entry in ⍎OPSTAK
5. Number of elements
6. Rho rho
7-14. Rho if needed. Scalar value otherwise.

⍎SPACE ALLOCATED SPACE TABLE
storage type: 32 bit integer
matrix: one row for each block of workspace allocated
collumns:
1. Space used(+) or available(-)
2. Starting address of block
3. Index to ⍎STRUCT (nix if space available)

FUNCTION REPRESENTATION
storage type: 32 bit integer
vector elements:
1. Index to ⍎STRUCT for local variable list
local variable list is vector of indexes to ⎕SYMB
for result, left arg, right arg, other local variables
(result nix for no result, left arg nix for mon fn or
nil fn, right arg nix for nil fn)
2. Index to ⍎STRUCT for label list
label list is a matrix: one row per label
collumns are 1) index to ⍎SYMB, 2) line number
3. Index to ⎕STRUCT for function as entered
function as entered is a character matrix
4. Index to ⍎STRUCT for the line internal code list
line internal code list is a vector of indexes to ⎕STRUCT,
one for each line of code
each line is a vector of object and op codes
5. Index to ⎕STRUCT for stop vector (nix if no stops)
6. Index to ⎕STRUCT for trace vector (nix if no trace)
Note: Literals are entered in ⎕SYMB as global variables
each with a name of 16 spaces.

⍎LOCAL ⎕SYMB ENTRIES FOR OBJECTS SUPERSEDED BY LOCAL VARIABLES
storage type: 32 bit integers
matrix: one row per object
collumns:
1. Object type
2. Index to ⎕STRUCT (nix if value error)
3. Index to previous local/global entry in ⍎LOCAL
(nix if global entry)
4. Original position in ⎕SYMB
Note: Collumns 1-3 are copied from the original entry in ⎕SYMB

⍎OPSTAK STACK OF PENDING OPERATIONS
storage type: 32 bit integer
matrix: one row per pending operation
collumns:
1. Function negative if an op code, or non-negative
index to ⎕STRUCT if defined function
2. Axis index to ⍎STRUCT (nix if no axis)
3. Right argument index to ⎕STRUCT (nix if none)
4. Code interpreter state

⎕STATE STATE INDICATOR TABLE
storage type: 32 bit integer
matrix: one row per defined function call
collumns:
1. Function index to ⎕SYMB
2. Line of function at which suspension occurred
3. Point of line at which suspension occurred
4. Point of ⎕OPSTAK at which suspension occurred

⎕LITFREE LITERALS NO LONGER IN USE
storage type: 32 bit integer
vector:
each element is an index to ⎕SYMB

⎕SYADD SYSTEM ADDRESSES
storage type: 32 bit integer
vector: 3 elements
1. Starting address of ⎕SYMB
2. Starting address of ⎕STRUCT
3. Starting address of ⎕SPACE

Note: ⎕SYADD is always at addresses 0,1,2 of the workspace

WRITING A PDP-11 APL IN C
Preliminary questions
Mapping workspace to disk
Interpreter and its tables
>>>> Calling foreign language routines
Issuing EMT's directly from APL

ALTERNATIVES FOR CALLING FOREIGN LANGUAGE ROUTINES
Link (task build) the foreign routines into the interpreter
Chain to them

--

LINKING
Requires compatible calling interface
Imposes size constraints on foreign routines
Imposes limit on the number of routines that can be linked
Requires relinking for each change while debugging
Limits portability between installations

--

CHAINING
APL workspace must be "reentrant"
Slow execution compared to linked routines
Limited data exchange area (184 bytes)
All nonresident handlers released (RT-11)

--

CHAINING TO A FOREIGN ROUTINE FROM APL, RETURNING TO APL

IN APL: R←'file spec' ⎕CHAIN DATA
1. APL flushes all buffers and closes all files
2. APL stores the name of the first workspace file as an ascii string at the beginning of the chain memory area
3. APL puts DATA into the remaining chain memory area starting in the first full word after the string
4. APL chains to 'file spec'

--

IN FOREIGN PROGRAM:
1. Program extracts your data from the chain memory area
2. Program puts results into the chain memory area, leaving the workspace name string unaltered
3. Program chains to the APL interpreter

--

BACK IN APL
1. APL gets the workspace file name and loads the workspace
2. APL restores the workspace to exactly where it was before opening all files that were open
3. APL returns the data from the first full word following the workspace file name in the chain memory area
4. Execution resumes: R←filespec' ⎕CHAIN DATA

--

CHANGING STORAGE TYPE INSIDE APL

R←<type> ⎕RETYPE <variable>
       The result has storage type of <type> with bit-for-bit copy of
              <variable> as its contents. The rho of the result is
              unchanged from the <variable> except for the last
              dimension which reflects the new storage type.
              Length error if the length in bits of any dimension
              of <variable> is not a multiple of the length of
              storage type <type>

R←<type> ⎕CONVERT <variable>
       The result has storage type of <type> with a numerical conversion
              of <variable> as its contents. The rho of the result is
              the same as <variable>

R←⎕TYPE <variable>
       Returns the current storage type for <variable>

WRITING A PDP-11 APL IN C
       Preliminary questions
       Mapping workspace to disk
       Interpreter and its tables
       Calling foreign language routines
>>>>   Issuing EMT's directly from APL
--
SPECIAL VARIABLES NEEDED

⎕EMTDATA
       A variable always mapped to the same memory area when
              ⎕EMT is called

⎕EMTADD
       The base address of the area where EMTDATA will be put
--
R←⎕EMT ARG
       ARG is a vector of 16 bit integer arguments
       ARG[0] is the EMT code
       ARG[1] is the argument to be placed in regester 0
       ARG[2 ...] are arguments to be placed on the stack
--

```
      VDSTATUS DEV
[1]    ⎕EMTDATA←16 ⎕CONVERT 0 0 0 0,⎕RAD50 DEV
[2]    →(0≤⎕EMT 16 ⎕CONVERT _342,⎕EMTADD+0 4)/B1
[3]    'DEVICE NOT FOUND'
[4]   B1: ⎕EMTDATA←¯32 ⎕CONVERT ⎕EMTDATA
[5]    →(0≠⎕EMTDATA[2])/0
[6]    'HANDLER NOT LOADED'
      ∇
```

APLSF TO VAX APL MIGRATION TOOLS

Stan Whitlock
APL Development
Digital Equipment Corp

The following is the documentation for a set of tools to assist DEC APL users in moving APL workspaces and files from APLSF-10/20 to VAX APL. The tools are written in APL to allow easy modification by the user who has extraordinary migration circumstances. We plan to make this software available through the DECUS Library.

SF2VAX.MEM October, 1983

Revised December, 1983

1.0 PURPOSE

The purpose of this document is to explain the migration process from APLSF to VAX APL which is performed by the tool SFTOVX. An example is provided to illustrate the process of migrating APL workspaces and data files. This document also explains the possible error messages and lists the APLSF/VAX APL incompatibilities.

2.0 EXPLANATION

The migration of APL workspaces and data files from APLSF to VAX APL occurs through text files. MIGRATE, an APL function, builds a batch control stream which then executes APL functions that create text files from APL files. Note that the easiest way to perform the migration in an unattended mode is by building this batch control stream. MIGRATE interacts with the user to find out what should be transported and prompts the user for filenames. MIGRATE also allows the user to specify an optional filter workspace (containing functions such as FILTER, ASFILTER, and DAFILTER) which allows the user to modify the file before it is transported. Files which can be transferred include workspaces, and /AS, /IS, /DA and /BS data files.

Through MIGRATE, a batch stream is executed which invokes the DRIVER function to transfer workspaces, the ASDRIVER function to transfer /AS data files, etc. Note that when the driver function is invoked to transfer a data file, its name reflects the type of file it is transferring (for example, ASDRIVER, DADRIVER)

These driver functions dump the workspaces and data files as input scripts into text files which are then transferred by the user's own methods to the VAX. They are then used by VAX APL as)INPUT files. DRIVER transports variables, system variables, channel assignments, workspace identifications, groups, and unlocked functions within APLSF workspaces as text to the VAX. While the text file is being made, DRIVER examines the user-defined functions to perform source transformations that minimize the incompatibilities between APLSF and VAX APL.

Note that these transformations do not occur within literals (for example, arguments to execute). Any informational messages displayed for the user are comment lines in the file that is used as)INPUT to VAX APL.

Once DRIVER has generated the text file and the user has transferred it to the VAX, the user executes an APL function called SUBMIT. SUBMIT interacts with the user and builds a VMS command file that, when run indirectly or in batch, will execute VAX APL for each input script specified by the user. When these input scripts are processed by means of VAX APL, a workspace or a data file is created on the VAX. These created files are the migrated images of their counterparts on TOPS-20.

This is a general tool which can be overlaid by the user's software in certain instances (for example, the FILTER function).

NOTE

> DRIVER performs a)LOAD of each workspace being migrated. If the workspace contains a nonempty .BXLX, the .BXLX will be executed. This may interfere with the migration process. Note that APLSF has no facilities to suppress .BXLX when loading a workspace.

2.1 Files In The Migration Tool

Several files are contained in the migration tool.

SFTOVX contains files for both the -20 and the VAX. The tools themselves assume that they reside in a directory with the logical name APLT2V:, both on the VAX and the -20. The following files are contained in SFTOVX.

- SF2VAX.MEM

  SF2VAX.MEM is the documentation for the APLSF to VAX APL migration

tool. It contains only APL TTY mnemonics for the APL characters. It may be listed on an ASCII printer for easy access. All APL expressions in SF2VAX.MEM assume .BXIO is 1.

There are detailed examples listed in SF2VAX.MEM. The examples are always reproduced in APL TTY mnemonics, even though some of the files produced by the migration tool contain APL key-paired ASCII APL characters. Note that .KM and .KJ are the VAX APL TTY mnemonics for <carriage return> and <linefeed>. Since the files built for migration have lines longer than 132 characters, the examples listed in SF2VAX.MEM have their lines wrapped to fit on a printed line. When such wrapping occurs, the line ends with "<wrap>" to show that it was wrapped for expository reasons.

o BXAV.LA

This is a translation table from APLSF's .BXAV to VAX APL's .BXAV used by the migration tool. The file contains APL key-paired ASCII APL character set and therefore must be printed or displayed on a device that can display APL characters.

Note the following:

- All of the APLSF .BXAV characters that print as .SQ are translated into the VAX APL character .SS (.BXAV [210]).

- The following APLSF .BXAV characters do not exist in VAX APL's .BXAV and are translated into the VAX APL character .CO (.BXAV [211]): 38 (.FI), 121, 103, 113, 115, 116, 117, 122,123 (all internal use characters). See .LD.USAV below for an explanation of these internal use characters in APLSF.

- VAX APL allows illegal overstrikes in data, in quoted literals, and in .QQ input. The text files produced by this migration tool contain APL overstrikes. Therefore, an ASCII backspace cannot be migrated as a single character because VAX APL will assume it to be part of an overstrike. Therefore, APLSF .BXAV [99 265] are translated into the VAX APL character .CC (.BXAV [207]).

- There are three overstrikes in VAX APL's .BXAV that are not in APLSF's .BXAV. If these overstrikes exist as three characters in a row in the APLSF data being migrated, they will appear as a single character in VAX APL. These are the following:

```
.CC (, over -)       .BXAV [207]
.SS (.LU over .US)   .BXAV [210]
.CO (.RU over .US)   .BXAV [211]
```

o SF2VX0.APL

SF2VX0.APL is built using the text file SF2VX0.IN as a)INPUT file to APLSF. For example, the following terminal session builds SF2VX0.APL on the -20:

```
@APLSF
terminal..TT
<APLSF banner>
CLEAR WS
      )INPUT SF2VX0,IN
<messages>
```

SF2VX0.APL is an APLSF workspace containing the function MIGRATE. MIGRATE interacts with the user to build a control file that can be run in batch. The control file performs the time-consuming task of interacting with the DRIVER functions that transform APLSF workspaces and data files into text files.

On the -20, put SF2VX0.APL in APLT2V:.

o SF2VX1.APL

SF2VX1.APL is built from the APLSF)INPUT file SF2VX1.IN.

SF2VX1.APL is an APLSF workspace containing the function DRIVER. DRIVER runs inside an APLSF workspace to transform it into a text file. SF2VX1.APL transfers the workspace identification, the file channel assignments, the settable system variables, the user-defined groups, the user-defined functions, and the variables. It also performs certain source transformations inside user-defined functions in an effort to minimize incompatibilities between APLSF and VAX APL.

On the -20, put SF2VX1.APL in APLT2V:.

o SF2VX3.APL

SF2VX3.APL is built from the VAX APL)INPUT file SF2VX3.IN.

SF2VX3.APL is a VAX APL workspace containing numerous user-defined functions that simulate the behavior of certain APLSF primitives under VAX APL. The source transformations performed by DRIVER (in SF2VX1.APL) replace certain APLSF primitives with these user-defined APL functions:

- .LD.USEN uses .BXVR under VAX APL to simulate monadic .EN from APLSF.

- .LD.USDE uses monadic .EP under VAX APL to simulate monadic .DE from APLSF.

- .LD.USOM uses B/.ID.RO B under VAX APL to simulate .OM B from APLSF.

- .LD.USDQ uses a particular setting of .BXCT under VAX APL to simulate .DQ from APLSF.

- .LD.USAB sets .BXCT to 0 under VAX APL before simulating dyadic .AB from APLSF.

- .LD.USIB simulates certain monadic I-beams under APLSF on VAX APL. The ones that cannot be simulated or are not supported under APLSF signal an error.

- .LD.USAPPEND, .LD.USDEQ, .LD.USENQ, .LD.USFCM, and .LD.USMTP replace the APLSF system functions .BXAPPEND, .BXDEQ, .BXENQ, .BXFCM, and .BXMTP. These user-defined functions report functionality that is "NOT AVAILABLE IN VAX APL".

- .LD.USASCII replaces .BXASCII from APLSF. It also returns a vector of 128 characters from VAX APL's .BXAV which closely resemble the ASCII characters.

- .LD.USAV replaces .BXAV from APLSF. It also returns a vector of 512 characters from VAX APL's .BXAV which closely resemble APLSF's .BXAV. See the file BXAV.LA for the .BXAV translation table in APL character set.

  Note that all of the characters in APLSF's .BXAV that print as .SQ are simulated by VAX APL's .BXAV [251]. The following .BXAV codes in APLSF are not supported with .BXAV [211]:

APLSF .BXAV index	name
38	.FI
101	double linefeed
103	null
113	left bracket
115	right bracket
116	up arrow
117	left arrow
122	tilde
123	delete

- .LD.USASS replaces .BXASS from APLSF and reports an error if its argument contains an APLSF password, a "ppn" surrounded by square brackets, or the switches /BS or /DUMP, since none of these are supported in VAX APL.

- .LD.USCHS, .LD.USCLS, .LD.USDAS, and .LD.USFLS replace the APLSF system functions .BXCHS, .BXCLS, .BXDAS, and .BXFLS. While all of these system functions exist in VAX APL, their behavior in VAX APL on empty arguments and arguments greater than 100 is different from under APLSF. These user-defined functions simulate the behavior under APLSF in these cases.

- .LD.USCIQ, .LD.USCOQ, and .LD.USDVC replace the APLSF system functions .BXCIQ, .BXCOQ and .BXDVC. While all of these system functions exist in VAX APL, their behavior in VAX APL is different from under APLSF. These user-defined functions report an error that the specific system function is "IN VAX APL" but is "INCOMPATIBLE WITH APLSF".

- .LD.USQLD, .LD.USQCO and .LD.USQPC replace the APLSF system functions .BXQLD, .BXQCO, and .BXQPC. These user-defined functions check their argument for an APLSF-style password on the workspace being accessed and replace it with a VAX APL-style password.

- .LD.USRENAME replaces the APLSF system function .BXRENAME. This user-defined function is designed to simulate the behavior of APLSF's .BXRENAME by using the command ".EP')DO RENAME ..." in VAX APL.

- .LD.USTT replaces the unsettable APLSF system variable .BXTT. This user-defined function uses .BXTT in VAX APL to construct the correct value returned by .BXTT in APLSF for "terminal type".

- Because the user may not desire to perform all of the transformations, note that the primitives are in TAB3 and their replacements are in the rows of MAP3. The system function names are located in TAB4.

On the VAX, put SF2VX3.APL in APLT2V:.

o SF2VX4.APL

  SF2VX4.APL is built from the APLSF)INPUT file SF2VX4.IN.

  SF2VX4.APL is an APLSF workspace that contains the file drivers, DRIVER, ASDRIVER, ISDRIVER, BSDRIVER, and DADRIVER. These drivers are used to transform APLSF data files (/AS, /IS, /BS, and /DA files, respectively) into text files.

  On the -20, put SF2VX4.APL in APLT2V:.

o SF2VX5.APL

  SF2VX5.APL is built from the VAX APL)INPUT file SF2VX5.IN.

  SF2VX5.APL is a VAX APL workspace that contains the function SUBMIT. SUBMIT interacts with the user to build a command file that can be run in batch that will perform the time-consuming task of running VAX APL on all of the text files that have been migrated from the -20.

  On the VAX, put SF2VX5.APL in APLT2V:.

2.2 Files Built By The Migration Tool

Several files are built by the migration tool.

The APL functions in SFTOVX produce files that either run other tools or

consist of text that should be transferred from the -20 to the VAX.

- MIGRATE in SF2VX0.APL on the -20

  MIGRATE produces a batch control file named MIGnnn.CTL, where "nnn" is a number from 000 through 999, in the connected directory. It chooses "nnn" by finding the first such file name that does not exist.

  The workspaces and APL data files on the -20 that are to be migrated to the VAX can reside in any accessible directory, as long as the file is not protected against reading.

- DRIVER in SF2VX1.APL on the -20

  DRIVER produces a file named "wsid.W2V" where "wsid" is the workspace identifer of the workspace being transferred. The .W2V file contains APL key-paired ASCII APL characters.

- File drivers in SF2VX4.APL on the -20

  ASDRIVER, ISDRIVER, BSDRIVER, and DADRIVER each produce files named "file.x2V", where "file" is the name of the APL data file being transferred, and "x" is "A", "I", "B", or "D", indicating the type of DRIVER that produced the file. The .x2V file contains APL key-paired ASCII APL characters.

- SUBMIT in SF2VX5.APL on the VAX

  SUBMIT produces a command file named SUBnnn.COM, where "nnn" is a number from 000 through 999, in the default directory. It chooses "nnn" by finding the first such file name that does not exist.

  The VAX APL images of workspaces and APL data files migrated from the -20 will be constructed either in the default directory on the VAX or in a directory associated with the logical name APLnnn:, where "nnn" was chosen by the DRIVER that produced the text file on the -20. SUBMIT will prompt the user for the definitions for these logical names.

2.3 User-Defined Filters

The following describes user-defined filters which handle special migration difficulties.

MIGRATE allows the user to specify a "filter workspace" to be executed before the workspace or APL data file in question is processed:

- Migrating workspaces

  The user-specified filter workspace is copied, using)COPY, into the workspace and the function named FILTER is executed. This niladic user-defined function can perform whatever transformations it wishes on the workspace. To prevent this workspace from being migrated, it can circumvent further processing by exiting APLSF via)OFF HOLD. When FILTER finishes,it is erased, using)ERASE, so it will not be migrated with the rest of the workspace. The migration process assumes that FILTER will remove any objects that are not to be migrated from the workspace before it completes.

- Migrating APL data files

  The user-specified filter workspace is copied, using)COPY, into the workspace SF2VX4 and the function named "xxFILTER" is executed, where "xx" represents the first two characters of the DRIVER that will be executed. ISFILTER, BSFILTER, and DAFILTER are monadic (as are the associated drivers), where the right argument is the name of the file being processed. ASFILTER (and ASDRIVER) is dyadic, where the left argument is the character set to be used to read the file (1 if TTY, 2 if APL) and the right argument is the file name.

  Note that it is important for users to provide their own version of the xxREAD routine that the xxDRIVER calls to process the data file, especially if the file being processed is a /BS file with no headers on the records. In that case, users are the only ones who know what constitutes a record in the file. There is internal documentation of the xxREAD and xxDRIVER functions in SF2VX4.APL and SF2VX4.IN so users can determine if they must provide their own xxREAD to migrate a data file.

3.0 EXAMPLE

Below is an example that illustrates the interaction between the user and the migration tool. The example is divided into five parts:

1. MIGRATE running under APLSF
2. The TOPS-20 batch control file that results from running MIGRATE
3. DRIVER and the corresponding file drivers running under APLSF
4. Transfer of text files to VAX by the user's own methods
5. SUBMIT running under VAX APL

The input scripts, the result of executing a DRIVER, follow the running of each corresponding DRIVER function. Following each of these parts, there is a brief explanation of some lines of the script.

Note that terminal output is in uppercase, with APL input indented six spaces. Annotations are in lowercase.

1. The user runs MIGRATE from the workspace APLT2V:SF2VX0 and describes the files that are to be processed.

```
      @APLSF
      TERMINAL..TT
            )LOAD APLT2V:SF2VX0
      SAVED 10:41:05 9-SEP-83 7P
                        check to see what .CTL files are already on disk
            )LIB *,CTL
      DSK:
      MIG000.CTL
                        MIGRATE will create DSK:MIG001.CTL
            MIGRATE
      WHAT TYPE OF FILE DO YOU WISH TO MIGRATE?  <cr>
                        invalid input gets the help message
      VALID FILE TYPES ARE W (WORKSPACE), A (/AS), I (/IS), D (/DA), B (/BS)
      OR ANY SYSTEM COMMAND TO BE EXECUTED
      OR Q (TO QUIT)
                        and back to the prompt
      WHAT TYPE OF FILE DO YOU WISH TO MIGRATE?  )FOO
                        error 21 is INCORRECT (system) COMMAND
      .EP ')FOO' REPORTED ERROR 0 21
                        and back to the prompt
      WHAT TYPE OF FILE DO YOU WISH TO MIGRATE?  W
                        "revert" means go back to the previous question
      WHAT FILE SPEC?  [CR TO REVERT] <cr>
                        and back to the prompt
      WHAT TYPE OF FILE DO YOU WISH TO MIGRATE?  W
                        an illegal file spec
      WHAT FILE SPEC?  [CR TO REVERT] $
                        error 22 is INCORRECT PARAMETER (in a system command)
      ERROR 0 22 IN FILESPEC $,APL
                        file does not exist
      WHAT FILE SPEC?  [CR TO REVERT] XXX
      XXX,APL DOES NOT EXIST
      WHAT FILE SPEC?  [CR TO REVERT] <cr>
                        and back to the prompt
      WHAT TYPE OF FILE DO YOU WISH TO MIGRATE?  )LIB
                        output from .EP ')LIB'
      DSK:
           A
           B
                        and back to the prompt
                        user wants to migrate workspaces
      WHAT TYPE OF FILE DO YOU WISH TO MIGRATE?  W
                        migrate all workspaces in the connected directory
      WHAT FILE SPEC?  [CR TO REVERT] *
                        use FILTER in FOO:W
      USER DEFINED FILTER?  [CR IF NONE] FOO:W
                        workspaces DSK:A.APL and DSK:B.APL have been processed
      ! WORKSPACE A,APL SETUP COMPLETE
      ! WORKSPACE B,APL SETUP COMPLETE
                        user wants to do more
      DO YOU HAVE MORE FILES TO PROCESS?  [Y OR N] Y
                        user wants to migrate workspaces
      WHAT TYPE OF FILE DO YOU WISH TO MIGRATE?  W
                        wants to migrate DSK:WS.PD
      WHAT FILE SPEC?  [CR TO REVERT] WS,PD
                        no filter this time
      USER DEFINED FILTER?  [CR IF NONE] <cr>
                        the WS is locked
      WORKSPACE WS,PD IS LOCKED
                        if the user provides the password, the WS can be
                        processed
      WHAT IS THE PASSWORD?  [CR TO SKIP THIS WS] FOO
      ! WORKSPACE WS,PD SETUP COMPLETE
      DO YOU HAVE MORE FILES TO PROCESS?  [Y OR N] Y
      WHAT TYPE OF FILE DO YOU WISH TO MIGRATE?  W
      WHAT FILE SPEC?  [CR TO REVERT] WS,PD
      USER DEFINED FILTER?  [CR IF NONE] <cr>
      WORKSPACE WS,PD IS LOCKED
                        user wants to skip processing of WS.PD
      WHAT IS THE PASSWORD?  [CR TO SKIP THIS WS] <cr>
      ! SKIPPING WORKSPACE WS,PD
      DO YOU HAVE MORE FILES TO PROCESS?  [Y OR N] Y
      WHAT TYPE OF FILE DO YOU WISH TO MIGRATE?  )LIB WRK:*,ADA
           A.ADA
           B.ADA
                        user wants to process /DA files
      WHAT TYPE OF FILE DO YOU WISH TO MIGRATE?  D
      WHAT FILE SPEC?  [CR TO REVERT] WRK:*.ADA
                        use DAFILTER in FOO:DA
      USER DEFINED FILTER?  [CR IF NONE] FOO:DA
      ! DA FILE WRK:A,ADA SETUP COMPLETE
      ! DA FILE WRK:B,ADA SETUP COMPLETE
      DO YOU HAVE MORE FILES TO PROCESS?  [Y OR N] Y
                        user wants to process /AS files
      WHAT TYPE OF FILE DO YOU WISH TO MIGRATE?  A
      WHAT FILE SPEC?  [CR TO REVERT] A.AAS
                        an extra question for /AS files only
                        empty input causes a reprompt
      WHAT CHARACTER SET?  [TTY OR BIT OR KEY] <cr>
                        invalid input gets the help message
      WHAT CHARACTER SET?  [TTY OR BIT OR KEY] W
      VALID CHARACTER SETS FOR READING /AS FILES ARE T (TTY), B (BIT), K (KEY)
      WHAT CHARACTER SET?  [TTY OR BIT OR KEY] T
      USER DEFINED FILTER?  [CR IF NONE] FOO:TTY
      ! AS FILE A,AAS SETUP COMPLETE
      DO YOU HAVE MORE FILES TO PROCESS?  [Y OR N] Y
      WHAT TYPE OF FILE DO YOU WISH TO MIGRATE?  A
      WHAT FILE SPEC?  [CR TO REVERT] A.AAS
      WHAT CHARACTER SET?  [TTY OR BIT OR KEY] K
      USER DEFINED FILTER?  [CR IF NONE] FOO:KEY
      ! AS FILE A,AAS SETUP COMPLETE
      DO YOU HAVE MORE FILES TO PROCESS?  [Y OR N] Y
      WHAT TYPE OF FILE DO YOU WISH TO MIGRATE?  A
      WHAT FILE SPEC?  [CR TO REVERT] A.AAS
      WHAT CHARACTER SET?  [TTY OR BIT OR KEY] B
      USER DEFINED FILTER?  [CR IF NONE] FOO:BIT
      ! AS FILE A,AAS SETUP COMPLETE
      DO YOU HAVE MORE FILES TO PROCESS?  [Y OR N] Y
```

```
WHAT TYPE OF FILE DO YOU WISH TO MIGRATE?  )LIB *.AAS[4,244]
DSK:
     A.AAS
     B.AAS
WHAT TYPE OF FILE DO YOU WISH TO MIGRATE?  A
                    file spec may contain [ppn]
WHAT FILE SPEC?  [CR TO REVERT] *.AAS[4,244]
WHAT CHARACTER SET?  [TTY OR BIT OR KEY] T
USER DEFINED FILTER?  [CR IF NONE] FOO:AS
! AS FILE A,AAS[4,244] SETUP COMPLETE
! AS FILE B,AAS[4,244] SETUP COMPLETE
DO YOU HAVE MORE FILES TO PROCESS?  [Y OR N] Y
WHAT TYPE OF FILE DO YOU WISH TO MIGRATE?  )LIB WRK:*,AIS[4,244]
     A.AIS
     B.AIS
                    user wants to process /IS files
WHAT TYPE OF FILE DO YOU WISH TO MIGRATE?  I
WHAT FILE SPEC?  [CR TO REVERT] WRK:*.AIS[4,244]
USER DEFINED FILTER?  [CR IF NONE] FOO:IS
! IS FILE WRK:A,AIS[4,244] SETUP COMPLETE
! IS FILE WRK:B,AIS[4,244] SETUP COMPLETE
DO YOU HAVE MORE FILES TO PROCESS?  [Y OR N] Y
WHAT TYPE OF FILE DO YOU WISH TO MIGRATE?  )LIB *.ABI
DSK:
     A.ABI
     B.ABI
                    user wants to process /BS files
WHAT TYPE OF FILE DO YOU WISH TO MIGRATE?  B
WHAT FILE SPEC?  [CR TO REVERT] *.ABI
USER DEFINED FILTER?  [CR IF NONE] FOO:BI
! BS FILE A,ABI SETUP COMPLETE
! BS FILE B,ABI SETUP COMPLETE
                    user wants to quit
DO YOU HAVE MORE FILES TO PROCESS?  [Y OR N] Q
                    MIGRATE built MIG001.CTL all right
! MIG001,CTL [4,244]/AS COMPLETED
     )OFF HOLD
@
```

Note that ! means the message will be listed in the CTL file. Although users may insert a comma (,) or a period (.) in the filespec, only a comma is output in the messages.

Users can revert to the 'type of file' prompt even if they are at the file spec prompt level. This allows them to check what files are on disk through)LIB. Note the following behavior of)LIB in APLSF.

For R_.ep')LIB ' F, where R is the result and F is the argument, the following apply.

- F may have a comma or a period before the file type.
- If F is of the form f or f,apl or f,typ or DSK:f..., R's first line will be DSK:.
- If F has no *, R will be a vector.
- If F has a *, R is a matrix, .BXPW wide, with filenames right justified to column six and .filetype in column 7 if the type is not APL. If the file type is .APL, ,APL will not be present in R.
- R is 0 e .ro 0 if an error occurs.
- If 0 = .ro,R, F does not exist.
- If F is dev:f... and F does not exist, 0 = .roR.
- If F is f... or DSK:F... and F does not exist, R is DSK:
- ")LIB f*" does not work in APLSF.
- ")LIB F-X" yields 22 INCORRECT PARAMETER
-)LIB must have the device name (dev:) before the file specification and the [ppn] after the file specification.

2. MIGRATE creates a batch control file, MIG001.CTL, which contains the following:

- The files MIGRATE built and when the building occurred (.BXTS)

```
! BUILDING MIG001,CTL [4,244]/AS 1983 9 9 10 43 8 990
@NOERROR
```

- User errors that occurred while using MIGRATE

```
! MIGRATING $,APL TYPE ,APL
ERROR 0 22 IN FILESPEC $,APL
! MIGRATING XXX,APL TYPE ,APL
XXX,APL DOES NOT EXIST
```

- Workspace processing (with the user-defined filter that must be named FILTER in the specified workspace, for example, FOO:W)

```
! MIGRATING *,APL TYPE ,APL
! USING FILTER FOO:W
@APLSF
*TT
*)MAXCORE 352
*)LOAD A,APL
*)COPY FOO:W
*FILTER
*)ERASE FILTER
*)PCOPY APLT2V:SF2VX1 DRIVER
*DRIVER
```

```
*)OFF HOLD
! WORKSPACE A,APL SETUP COMPLETE
@APLSF
*TT
*)MAXCORE 352
*)LOAD B,APL
*)COPY FOO:W
*FILTER
*)ERASE FILTER
*)PCOPY APLT2V:SF2VX1 DRIVER
*DRIVER
*)OFF HOLD
! WORKSPACE B,APL SETUP COMPLETE
```

o Locked workspace processing (no user-defined filter)

```
! MIGRATING WS,PD TYPE ,APL
@APLSF
*TT
*)MAXCORE 352
*)LOAD WS,PD-FOO
*)PCOPY APLT2V:SF2VX1 DRIVER
*DRIVER
*)OFF HOLD
! WORKSPACE WS,PD SETUP COMPLETE
```

o Locked workspace processing - workspace skipped because user doesn't know password

```
! MIGRATING WS,PD TYPE ,APL
! SKIPPING LOCKED WORKSPACE WS,PD
```

o /DA files processing (with the user-defined filter, must be named DAFILTER in the specified workspace, for example, FOO:DA). Note the file specification contains a device name.

```
! MIGRATING WRK:*,ADA TYPE ,ADA
! USING FILTER FOO:DA
@APLSF
*TT
*)MAXCORE 352
*)LOAD APLT2V:SF2VX4
*)COPY FOO:DA
*    DAFILTER 'WRK:A,ADA'
*)ERASE DAFILTER
*    DADRIVER 'WRK:A,ADA'
*)OFF HOLD
! DA FILE WRK:A,ADA SETUP COMPLETE
@APLSF
*TT
*)MAXCORE 352
*)LOAD APLT2V:SF2VX4
*)COPY FOO:DA
*    DAFILTER 'WRK:B,ADA'
*)ERASE DAFILTER
*    DADRIVER 'WRK:B,ADA'
*)OFF HOLD
! DA FILE WRK:B,ADA SETUP COMPLETE
```

o /AS files processing (with the user-defined filter ASFILTER, which is dynamic). Note the TT/BIT response to the "terminal..." prompt for the /BIT character set.

```
! MIGRATING A,AAS TYPE ,AAS
! USING CHARACTER SET /TTY
! USING FILTER FOO:TTY
@APLSF
*TT
*)MAXCORE 352
*)LOAD APLT2V:SF2VX4
*)COPY FOO:TTY
* 1  ASFILTER 'A,AAS'
*)ERASE ASFILTER
* 1  ASDRIVER 'A,AAS'
*)OFF HOLD
! AS FILE A,AAS SETUP COMPLETE
! MIGRATING A,AAS TYPE ,AAS
! USING CHARACTER SET /KEY
! USING FILTER FOO:KEY
@APLSF
*TT
*)MAXCORE 352
*)LOAD APLT2V:SF2VX4
*)COPY FOO:KEY
* 2  ASFILTER 'A,AAS'
*)ERASE ASFILTER
* 2  ASDRIVER 'A,AAS'
*)OFF HOLD
! AS FILE A,AAS SETUP COMPLETE
! MIGRATING A,AAS TYPE ,AAS
! USING CHARACTER SET /BIT
! USING FILTER FOO:BIT
@APLSF
*TT/BIT
*)MAXCORE 352
*)LOAD APLT2V:SF2VX4
*)COPY FOO:BIT
* 2  ASFILTER 'A,AAS'
*)ERASE ASFILTER
* 2  ASDRIVER 'A,AAS'
*)OFF HOLD
! AS FILE A,AAS SETUP COMPLETE
```

o /AS files processing (with user-defined filter). Note the file specification contains [ppn].

```
! MIGRATING *,AAS[4,244] TYPE ,AAS
! USING CHARACTER SET /TTY
! USING FILTER FOO:AS
@APLSF
```

```
*TT
*)MAXCORE 352
*)LOAD APLT2V:SF2VX4
*)COPY FOO:AS
* 1   ASFILTER 'A,AAS[4,244]'
*)ERASE ASFILTER
* 1   ASDRIVER 'A,AAS[4,244]'
*)OFF HOLD
! AS FILE A,AAS[4,244] SETUP COMPLETE
@APLSF
*TT
*)MAXCORE 352
*)LOAD APLT2V:SF2VX4
*)COPY FOO:AS
* 1   ASFILTER 'B,AAS[4,244]'
*)ERASE ASFILTER
* 1   ASDRIVER 'B,AAS[4,244]'
*)OFF HOLD
! AS FILE B,AAS[4,244] SETUP COMPLETE
```

- /IS files processing (with user-defined filter). Note the file specification contains both a device name and [ppn].

```
! MIGRATING WRK:*,AIS[4,244] TYPE ,AIS
! USING FILTER FOO:IS
@APLSF
*TT
*)MAXCORE 352
*)LOAD APLT2V:SF2VX4
*)COPY FOO:IS
*    ISFILTER 'WRK:A,AIS[4,244]'
*)ERASE ISFILTER
*    ISDRIVER 'WRK:A,AIS[4,244]'
*)OFF HOLD
! IS FILE WRK:A,AIS[4,244] SETUP COMPLETE
@APLSF
*TT
*)MAXCORE 352
*)LOAD APLT2V:SF2VX4
*)COPY FOO:IS
*    ISFILTER 'WRK:B,AIS[4,244]'
*)ERASE ISFILTER
*    ISDRIVER 'WRK:B,AIS[4,244]'
*)OFF HOLD
! IS FILE WRK:B,AIS[4,244] SETUP COMPLETE
```

- /BS files processing (with the user-defined filter BSFILTER in FOO:BI).

```
! MIGRATING *,ABI TYPE ,ABI
! USING FILTER FOO:BI
@APLSF
*TT
*)MAXCORE 352
*)LOAD APLT2V:SF2VX4
*)COPY FOO:BI
*    BSFILTER 'A,ABI'
*)ERASE BSFILTER
*    BSDRIVER 'A,ABI'
*)OFF HOLD
! BS FILE A,ABI SETUP COMPLETE
@APLSF
*TT
*)MAXCORE 352
*)LOAD APLT2V:SF2VX4
*)COPY FOO:BI
*    BSFILTER 'B,ABI'
*)ERASE BSFILTER
*    BSDRIVER 'B,ABI'
*)OFF HOLD
! BS FILE B,ABI SETUP COMPLETE
! MIG001,CTL [4,244]/AS COMPLETED
```

Note that @NOERROR means that SFTOVX continues processing in batch no matter what messages it receives. The)ERASE that follows the copying of the filter workspace insures that the filter function will not be migrated.

3. The user runs DRIVER in APLT2V:SF2VX1 on DSK:TEST0.APL.

   - Workspace TEST0.APL contains the following:
     - channels 1, 2, 4, 6, and 8 assigned
     - various system variables settings different from a CLEAR WS
     - two groups
     - six functions (1 locked; 1 containing an error)
     - six variables

   - DRIVER runs under APLSF, producing messages.

```
!  MIGRATING TEST0,APL TYPE ,APL
@APLSF
terminal..TT
APL-20 DECSYSTEM-20 APLSF 2(552)
TTY226) 12:26:50 TUESDAY 27-SEP-83 USER [4,244]
CLEAR WS
      )MAXCORE 352
WAS 40P
      )LOAD TEST0,APL
SAVED 12:24:12 27-SEP-83 8P
      )PCOPY APLT2V:SF2VX1 DRIVER
SAVED 9:21:55 27-SEP-83 16P
      DRIVER
```

```
      " *** "[4,244]" REPLACED BY "APL000:" IN WSID
      " *** "FOO,ADA [4,244]/DA" REPLACED BY "APL001:FOO,ADA /DA" ON CHANNEL 1
      " *** "TWO,AAS [4,244]/AS" REPLACED BY "APL002:TWO,AAS /AS" ON CHANNEL 2
      " *** "DEV:FOUR,AIS [4,244]/IS" REPLACED BY "APL004:FOUR,AIS /IS" ON CHANNEL 4
      " *** "DEV:SIX,ADA [4,4]/DA" REPLACED BY "APL006:SIX,ADA /DA" ON CHANNEL 6
      " *** "EIGHT,ABI [4,4]/BS" REPLACED BY "APL008:EIGHT,ABI /BS" ON CHANNEL 8
      "1 QUAD NAMES WERE REMOVED FROM THE HEADER OF FF"
      "ASSIGNMENTS TO THE FOLLOWING QUAD NAMES WERE REPLACED WITH QUAD SINK IN FF
      AV
      "
      "THE FOLLOWING PRIMITIVE FUNCTIONS WERE REPLACED WITH USER FUNCTIONS IN FF
      .AB.OM.IB$
      "
      "THE FOLLOWING SYSTEM FUNCTIONS WERE REPLACED WITH USER FUNCTIONS IN FF
      AV
      "
      " *** LOCKED LOCKED
      " *** BIG INTEGER IN Y
            )OFF HOLD
      TTY226) 12:27:05 27-SEP-83
      CONNECTED 0:00:15 CPU TIME 0:00:08
      2233 STATEMENTS 9337 OPERATIONS

      EXIT
      @
```

o DRIVER produces a script in TEST0.W2V:. Note that APL000:, APL002:, APL004:, APL006:, and APL008: must be defined on the VAX for TEST.W2V to work on the VAX.

```
      " BUILDING TEST0,W2V [4,244]/AS 1983 9 27 12 26 53 675
      )CLEAR
      " *** "[4,244]" REPLACED BY "APL000:" IN WSID
      )WSID APL000:TEST0
      )COPY APLT2V:SF2VX3
      " *** "FOO,ADA [4,244]/DA" REPLACED BY "APL001:FOO,ADA /DA" ON CHANNEL 1
      .LD.USASS " 1 APL001:FOO.ADA /DA"
      " *** "TWO,AAS [4,244]/AS" REPLACED BY "APL002:TWO,AAS /AS" ON CHANNEL 2
      .LD.USASS " 2 APL002:TWO.AAS /AS"
      " *** "DEV:FOUR,AIS [4,244]/IS" REPLACED BY "APL004:FOUR,AIS /IS" ON CHANNEL 4
      .LD.USASS " 4 APL004:FOUR.AIS /IS"
      " *** "DEV:SIX,ADA [4,4]/DA" REPLACED BY "APL006:SIX,ADA /DA" ON CHANNEL 6
      .LD.USASS " 6 APL006:SIX.ADA /DA"
      " *** "EIGHT,ABI [4,4]/BS" REPLACED BY "APL008:EIGHT,ABI /BS" ON CHANNEL 8
      .LD.USASS " 8 APL008:EIGHT.ABI /BS"
      .BXRESET
      .BXAUS_0
      .BXCT_0
      .BXGAG_0
      .BXIO_0
      .BXPP_10
      .BXPW_79
      .BXRL_0
      .BXTIMELIMIT_0
      A.US_""
      A.US_A.US,".BX:.KM.KJ
```

```
.BXSF_10.ROA.US
A.US_''
A.US_A.US,'HI THERE - THIS IS .BXTRAP'
.BXTRAP_24.ROA.US
A.US_''
.BXLX_0.ROA.US'
A.US_''
A.US_A.US,')GROUP G1 A BBBBBBBBBBBBBBBBBBBBBBBBBBBBBB C '
.XQ A.US
A.US_''
A.US_A.US,')GROUP G2 XXXXXXXXXXXXXXXXXXXXXXXXXXXX Y ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ '
A.US_A.US,' AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA '
A.US_A.US,' '
.XQ A.US
A.US_''
A.US_A.US,'R_DUPSOUT X .KM.KJ'
A.US_A.US,'"REMOVES DUPLICATES FROM VECTOR .KM.KJ'
A.US_A.US,'R_((.IO.ROX)=X.IOX)/X_,X .KM.KJ'
A.US_.BX_.BXFX .BXBOX A.US .DM (0=1^0.RO A.US)/.BXERROR
A.US_''
A.US_A.US,'ERROR.KM.KJ'
A.US_A.US,'L:1 .KM.KJ'
A.US_A.US,'L:2 .KM.KJ''
A.US_.BX_.BXFX .BXBOX A.US .DM (0=1^0.RO A.US)/.BXERROR
A.US_''
A.US_A.US,'Z_A F B;T;.BXIO.KM.KJ'
A.US_A.US,'L:1 .KM.KJ'
A.US_A.US,'''1 2 3 4 5 6 7 8 9 0 +   .DM'' .KM.KJ'
A.USA.US,'''.DD .NG < .LE = .GE > .NE & .OR - % $'' .KM.KJ'
A.US_A.US,'''Q W E R T Y U I O P _ .GO'' .KM.KJ'
A.US_A.US,'''?  .OM .EP .RO ~ ^ .DA .IO .LO * ( )'' .KM.KJ'
A.US_A.US,'''A S D F G H J K L ( ) .RK'' .KM.KJ'
A.US_A.US,'''.AL .CE .FL .US .DL .LD .SO '''' .BX [ ] .LK'' .KM.KJ'
A.US_A.US,'''Z X C V B N M , .  /'' .KM.KJ'
A.US_A.US,'''LEFT .LU RIGHT .RU .DU .UU .DE .EN | ;  :  \'' .KM.KJ'
A.US_A.US,'''.ZA .ZB .ZC .ZD .ZE .ZF .ZG .ZH .ZI .ZJ .ZK .ZL .ZM'' .KM.KJ'
A.US_A.US,'''.ZN .ZO .ZP .ZQ .ZR .ZS .ZT .ZU .ZV .ZW .ZX .ZY .ZZ'' .KM.KJ'
A.US_A.US,'''.UD !  " .IB .XQ .FM .DQ .IQ .OQ .QQ .QD .GU .GD .PD'' .KM.KJ'
A.US_A.US,'''.NR .NN .LG .RV .TR .CR .CS .CB'' .KM.KJ'
A.US_.BX_.BXFX .BXBOX A.US .DM (0=1^0.RO A.US)/.BXERROR
A.US_''
'2 QUAD NAMES WERE REMOVED FROM THE HEADER OF FF'
A.US_A.US,'FF;.BXIO.KM.KJ'
A.US_A.US,' .LD.USOM ''.OM'' .KM.KJ'
A.US_A.US,'A .LD.USAB ''|'' .KM.KJ'
A.US_A.US,' .LD.USIB ''.IB'' .KM.KJ'
A.US_A.US,'.BXSINK_ .LD.USAV .KM.KJ'
A.US_A.US,'''$'' .BXFMT ''$'' .KM.KJ'
A.US_.BX_.BXFX .BXBOX A.US .DM (0=1^0.RO A.US)/.BXERROR
'ASSIGNMENTS TO THE FOLLOWING QUAD NAMES WERE REPLACED WITH QUAD SINK IN FF
AV
'
'THE FOLLOWING PRIMITIVE FUNCTIONS WERE REPLACED WITH USER FUNCTIONS IN FF
!.OM.IB$
'
```

```
'THE FOLLOWING SYSTEM FUNCTIONS WERE REPLACED WITH USER FUNCTIONS IN FF
AV
'
A.US_''
A.US_A.US,'HOWDUPSOUT .KM.KJ'
A.US_A.US,'''THIS FUNCTION RETURNS A VECTOR FORMED BY TAKING THE RIGHT ARGUMENT'' .KM.KJ'
A.US_A.US,'''AND REMOVING DUPLICATES FROM IT.'' .KM.KJ'
A.US_A.US,'''THE RIGHT ARGUMENT MAY BE A NUMERIC OR CHARACTER SCALAR OR VECTOR.'' .KM.KJ'
A.US_A.US,''' '' .KM.KJ'
A.US_A.US,''' .....TRY.....'' .KM.KJ'
A.US_A.US,'''.....RO.BX_DUPSOUT .IO12'' .KM.KJ'
A.US_A.US,'''.....RO.BX_DUPSOUT 12.RO .IO3 '' .KM.KJ'
A.US_A.US,'''NOTE THAT SCALARS BECOME VECTORS OF LENGTH 1'' .KM.KJ'
A.US_A.US,''' .RO.BX_DUPSOUT 1'' .KM.KJ'
A.US_A.US,'3 1 .RO'' '' .KM.KJ'
A.US_.BX_.BXFX .BXBOX A.US .DM (0=1^0.RO A.US)/.BXERROR
" *** LOCKED LOCKED
A.US_.IO0
A.US_A.US,0 1 2 3 4 5 6 7 8 9
A.US_A.US,10 11 12 13 14 15 16 17 18 19
A.US_A.US,20 21 22 23 24 25 26 27 28 29
A.US_A.US,30 31 32 33 34 35 36 37 38 39
A.US_A.US,40 41 42 43 44 45 46 47 48 49
A.US_A.US,50 51 52 53 54 55 56 57 58 59
A.US_A.US,60 61 62 63 64 65 66 67 68 69
A.US_A.US,70 71 72 73 74 75 76 77 78 79
A.US_A.US,80 81 82 83 84 85 86 87 88 89
A.US_A.US,90 91 92 93 94 95 96 97 98 99
X_100.ROA.US
A.US_.IO0
A.US_A.US,0.5 1.5 2.5 3.5 4.5 5.5 6.5 7.5 8.5
A.US_A.US,9.5 10.5 11.5 12.5 13.5 14.5 15.5 16.5 17.5
A.US_A.US,18.5 19.5 20.5 21.5 22.5 23.5 24.5 25.5 26.5
A.US_A.US,27.5 28.5 29.5 30.5 31.5 32.5 33.5 34.5 35.5
A.US_A.US,36.5 37.5 38.5 39.5 40.5 41.5 42.5 43.5 44.5
A.US_A.US,45.5 46.5 47.5 48.5 49.5 50.5 51.5 52.5 53.5
A.US_A.US,54.5 55.5 56.5 57.5 58.5 59.5 60.5 61.5 62.5
A.US_A.US,63.5 64.5 65.5 66.5 67.5 68.5 69.5 70.5 71.5
A.US_A.US,72.5 73.5 74.5 75.5 76.5 77.5 78.5 79.5 80.5
A.US_A.US,81.5 82.5 83.5 84.5 85.5 86.5 87.5 88.5 89.5
A.US_A.US,90.5 91.5 92.5 93.5 94.5 95.5 96.5 97.5 98.5
A.US_A.US,99.5
XX_100.ROA.US
A.US_.IO0
A.US_A.US,1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 <wrap>
0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 <wrap>
0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1
A.US_A.US,1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 <wrap>
1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1
XXX_200.ROA.US
A.US_''
A.US_A.US,'.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS!.RV.TR.IB.QQ <wrap>
.LG.NN.NR".GU.GD.CR.CS.CB.SS.PD.QD.IQ.OQ.DQ$.CO.XQ.FM.SS.SS.SS.SS.SS.SS <wrap>
.DD<.LE.GE>?.OM.EP.RO^.DA.IO.AL.CE.FL()[].LU.RU.UU.US.SS.SS.SS_.GO.BX.NE=.DU.DE&
```

```
A.US_A.US,'.OR~/*#%+-.LO|;,.SO.EN.DL:''.CC.KJ.CO.KM.CO@.KL.KI.SS..NG.SS@ <wrap>
.CO.LK.CO.CO.CO.DM{.RK}.CO.CO.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS <wrap>
.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS'

A.US_A.US,'.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS <wrap>
.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS <wrap>
.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS'

A.US_A.US,'.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.WN.KA.KB.KC <wrap>
.KD.KE.KF.KG.CC.KI.KJ.KK.KL.KM.KN.KO.KP.KQ.KR.KS.KT.KU.KV.KW.KX.KY.KZ.WE.WF <wrap>
.WG.WR.WU.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS0123456789.SS.SS.SS.SS.SS'

A.US_A.US,'.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS <wrap>
.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.LDABCDEFGHIJKLMNOPQRSTUVWXYZ.SS !.QU.PS$.PC.AP''()*+,-./01'

A.US_A.US,'23456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[]^_`abcdefghijklmnopqrstuvwxyz{|}~.WD.SS.SS'

A.US_A.US,'.SS.SS.UD.ZA.ZB.ZC.ZD.ZE.ZF.ZG.ZH.ZI.ZJ.ZK.ZL.ZM.ZN.ZO.ZP.ZQ.ZR.ZS.ZT.ZU.ZV.ZW.ZX.ZY.ZZ.SS.SS.SS.SS.SS'

XXXX_32 16.ROA.US
A.US_.IOO
" *** BIG INTEGER IN Y
A.US_A.US,8589934592 .NG8589934592 8589934592 .NG8589934592
Y_4.ROA.US
A.US_.IOO
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
```

```
.LD.LDRANK_300.ROA.US
A.US_.I00
A.US_A.US,2
YY_.LD.LDRANK.ROA.US
.BXSINK_.BXEX 2 6.RO 'A.US .LD.LDRANK'
)SAVE
```

o Using TEST0.W2V as input to VAX APL produces the workspace TEST0.APL. Note that TEST0.W2V is a /KEY file.

```
      )IN TEST0.W2V/KEY
      " BUILDING TEST0,W2V [4,244]/AS 1983 9 27 12 26 53 675
      )CLEAR
CLEAR WS
      " *** '[4,244]' REPLACED BY 'APL000:' IN WSID
      )WSID APL000:TEST0
WAS CLEAR WS
      )COPY APLT2V:SF2VX3
SAVED 26-SEP-1983 14:19:57 42 BLKS
      " *** 'FOO,ADA [4,244]/DA' REPLACED BY 'APL001:FOO,ADA /DA' ON CHANNEL 1
      .LD.USASS ' 1 APL001:FOO.ADA /DA'
1
      " *** 'TWO,AAS [4,244]/AS' REPLACED BY 'APL002:TWO,AAS /AS' ON CHANNEL 2
      .LD.USASS ' 2 APL002:TWO.AAS /AS'
2
      " *** 'DEV:FOUR,AIS [4,244]/IS' REPLACED BY 'APL004:FOUR,AIS /IS' ON CHANNEL 4
      .LD.USASS ' 4 APL004:FOUR.AIS /IS'
4
      " *** 'DEV:SIX,ADA [4,4]/DA' REPLACED BY 'APL006:SIX,ADA /DA' ON CHANNEL 6
      .LD.USASS ' 6 APL006:SIX.ADA /DA'
6
      " *** 'EIGHT,ABI [4,4]/BS' REPLACED BY 'APL008:EIGHT,ABI /BS' ON CHANNEL 8
      .LD.USASS ' 8 APL008:EIGHT.ABI /BS'
.BXASS FILE SPEC CONTAINING /BS IS UNSUPPORTED
      .BXRESET
      .BXAUS_0
      .BXCT_0
      .BXGAG_0
      .BXIO_0
      .BXPP_10
      .BXPW_79
      .BXRL_0
      .BXTIMELIMIT_0
      A.US_''
      A.US_A.US,'.BX:<cr><lf>
      .BXSF_10.ROA.US
      A.US_''
      A.US_A.US,'HI THERE - THIS IS .BXTRAP
      .BXTRAP_24.ROA.US
      A.US_''
      .BXLX_0.ROA.US
      A.US_''
      A.US_A.US,')GROUP G1 A BBBBBBBBBBBBBBBBBBBBBBBBBBBBB C
      .XQ A.US
      A.US''
      A.US_A.US,')GROUP G2 XXXXXXXXXXXXXXXXXXXXXXXXXXXX Y ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
```

```
      A.US_A.US,' AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA '
      A.US_A.US,' '
      .XQ A.US
      A.US_''
      A.US_A.US,'R_DUPSOUT X <cr><lf>'
      A.US_A.US,'"REMOVES DUPLICATES FROM VECTOR <cr><lf>'
      A.US_A.US,'R_((.IO.ROX)=X.IOX)/X_,X <cr><lf>'
      A.US_.BX_.BXFX .BXBOX A.US .DM (0=1^0.RO A.US)/.BXERROR
DUPSOUT

      A.US_''
      A.US_A.US,'ERROR<cr><lf>'
      A.US_A.US,'L:1 <cr><lf>'
      A.US_A.US,'L:2 <cr><lf>'
      A.US_.BX_.BXFX .BXBOX A.US .DM (0=1^0.RO A.US)/.BXERROR
2
  6 LABEL ERROR (DUPLICATE LABEL)
L:2
'
      A.US_''
      A.US_A.US,'Z_A F B;T;.BXIO<cr><lf>'
      A.US_A.US,'L:1 <cr><lf>'
      A.US_A.US,'''1 2 3 4 5 6 7 8 9 0 +   .DM'' <cr><lf>'
      A.USA.US,'''.DD .NG < .LE = .GE > .NE & .OR - % $'' <cr><lf>'
      A.US_A.US,'''Q W E R T Y U I O P _ .GO'' <cr><lf>'
      A.US_A.US,'''?  .OM .EP .RO ~ ^ .DA .IO .LO * { }'' <cr><lf>'
      A.US_A.US,'''A S D F G H J K L ( ) .RK'' <cr><lf>'
      A.US_A.US,'''.AL .CE .FL .US .DL .LD .SO '''' .BX [ ] .LK'' <cr><lf>'
      A.US_A.US,'''Z X C V B N M , .  /'' <cr><lf>'
      A.US_A.US,'''LEFT .LU RIGHT .RU .DU .UU .DE .EN | ;  :  \'' <cr><lf>'
      A.US_A.US,'''.ZA .ZB .ZC .ZD .ZE .ZF .ZG .ZH .ZI .ZJ .ZK .ZL .ZM'' <cr><lf>'
      A.US_A.US,'''.ZN .ZO .ZP .ZQ .ZR .ZS .ZT .ZU .ZV .ZW .ZX .ZY .ZZ'' <cr><lf>'
      A.US_A.US,'''.UD !  " .IB .XQ .FM .DQ .IQ .OQ .QQ .QD .GU .GD .PD'' <cr><lf>'
      A.US_A.US,'''.NR .NN .LG .RV .TR .CR .CS .CB'' <cr><lf>'
      A.US_.BX_.BXFX .BXBOX A.US .DM (0=1^0.RO A.US)/.BXERROR
F

      A.US_''
      '2 QUAD NAMES WERE REMOVED FROM THE HEADER OF FF'
2 QUAD NAMES WERE REMOVED FROM THE HEADER OF FF
      A.US_A.US,'FF;.BXIO<cr><lf>'
      A.US_A.US,' .LD.USOM ''.OM'' <cr><lf>'
      A.US_A.US,'A .LD.USAB ''|'' <cr><lf>'
      A.US_A.US,' .LD.USIB ''.IB'' <cr><lf>'
      A.US_A.US,'.BXSINK_ .LD.USAV <cr><lf>'
      A.US_A.US,'''$'' .BXFMT ''$'' <cr><lf>'
      A.US_.BX_.BXFX .BXBOX A.US .DM (0=1^0.RO A.US)/.BXERROR
FF

      'ASSIGNMENTS TO THE FOLLOWING QUAD NAMES WERE REPLACED WITH QUAD SINK IN FF
AV
'
ASSIGNMENTS TO THE FOLLOWING QUAD NAMES WERE REPLACED WITH QUAD SINK IN FF
AV
```

```
      'THE FOLLOWING PRIMITIVE FUNCTIONS WERE REPLACED WITH USER FUNCTIONS IN FF
].OM.IB$
*
THE FOLLOWING PRIMITIVE FUNCTIONS WERE REPLACED WITH USER FUNCTIONS IN FF
].OM.IB$

      'THE FOLLOWING SYSTEM FUNCTIONS WERE REPLACED WITH USER FUNCTIONS IN FF
AV
*
THE FOLLOWING SYSTEM FUNCTIONS WERE REPLACED WITH USER FUNCTIONS IN FF
AV

      A.US_''
      A.US_A.US,'HOWDUPSOUT <cr><lf>'
      A.US_A.US,'''THIS FUNCTION RETURNS A VECTOR FORMED BY TAKING THE RIGHT ARGUMENT'' <cr><lf>'
      A.US_A.US,'''AND REMOVING DUPLICATES FROM IT.'' <cr><lf>'
      A.US_A.US,'''THE RIGHT ARGUMENT MAY BE A NUMERIC OR CHARACTER SCALAR OR VECTOR.'' <cr><lf>'
      A.US_A.US,''' '' <cr><lf>'
      A.US_A.US,''' .....TRY.....'' <cr><lf>'
      A.US_A.US,'''.....RO.BX_DUPSOUT .IO12'' <cr><lf>'
      A.US_A.US,'''.....RO.BX_DUPSOUT 12.RO .IO3 '' <cr><lf>'
      A.US_A.US,'''NOTE THAT SCALARS BECOME VECTORS OF LENGHT 1'' <cr><lf>'
      A.US_A.US,''' .RO.BX_DUPSOUT 1'' <cr><lf>'
      A.US_A.US,'3 1 .RO'' '' <cr><lf>'
      A.US_.BX_.BXFX .BXBOX A.US .DM (0=1^0.RO A.US)/.BXERROR
HOWDUPSOUT

      '' *** LOCKED LOCKED
      A.US_.IO0
      A.US_A.US,0 1 2 3 4 5 6 7 8 9
      A.US_A.US,10 11 12 13 14 15 16 17 18 19
      A.US_A.US,20 21 22 23 24 25 26 27 28 29
      A.US_A.US,30 31 32 33 34 35 36 37 38 39
      A.US_A.US,40 41 42 43 44 45 46 47 48 49
      A.US_A.US,50 51 52 53 54 55 56 57 58 59
      A.US_A.US,60 61 62 63 64 65 66 67 68 69
      A.US_A.US,70 71 72 73 74 75 76 77 78 79
      A.US_A.US,80 81 82 83 84 85 86 87 88 89
      A.US_A.US,90 91 92 93 94 95 96 97 98 99
      X_100.ROA.US
      A.US_.IO0
      A.US_A.US,0.5 1.5 2.5 3.5 4.5 5.5 6.5 7.5 8.5
      A.US_A.US,9.5 10.5 11.5 12.5 13.5 14.5 15.5 16.5 17.5
      A.US_A.US,18.5 19.5 20.5 21.5 22.5 23.5 24.5 25.5 26.5
      A.US_A.US,27.5 28.5 29.5 30.5 31.5 32.5 33.5 34.5 35.5
      A.US_A.US,36.5 37.5 38.5 39.5 40.5 41.5 42.5 43.5 44.5
      A.US_A.US,45.5 46.5 47.5 48.5 49.5 50.5 51.5 52.5 53.5
      A.US_A.US,54.5 55.5 56.5 57.5 58.5 59.5 60.5 61.5 62.5
      A.US_A.US,63.5 64.5 65.5 66.5 67.5 68.5 69.5 70.5 71.5
      A.US_A.US,72.5 73.5 74.5 75.5 76.5 77.5 78.5 79.5 80.5
      A.US_A.US,81.5 82.5 83.5 84.5 85.5 86.5 87.5 88.5 89.5
      A.US_A.US,90.5 91.5 92.5 93.5 94.5 95.5 96.5 97.5 98.5
      A.US_A.US,99.5
      XX_100.ROA.US
      A.US_.IO0
```

```
      A.US_A.US,1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 <wrap>
0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 <wrap>
1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1
      A.US_A.US,1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 <wrap>
0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1
      XXX_200.ROA.US
      A.US_''

      A.US_A.US,'.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS!.RV.TR.IB.QQ.LG.NN.NR".GU.GD.CR.CS <wrap>
.CB.SS.PD.QD.IQ.OQ.DQ$.CO.XQ.FM.SS.SS.SS.SS.SS.SS.DD<.LE.GE>?.OM.EP.RO^.DA.IO.AL.CE.FL()[].LU.RU.UU <wrap>
.US.SS.SS.SS_.GO.BX.NE=.DU.DE&'

      A.US_A.US,'.OR~/*#%+-.LO!;,.SO.EN.DL:''.CC.KJ.CO.KM.CO@.KL.KI.SS..NG.SS@.CO.LK.CO.CO.CO.DM{ <wrap>
.RK}.CO.CO.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS <wrap>
.SS.SS.SS.SS.SS.SS'

      A.US_A.US,'.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS <wrap>
.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS <wrap>
.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS'

      A.US_A.US,'.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS <wrap>
.KM.KN.KO.KP.KQ.KR.KS.KT.KU.KV.KW.KX.KY.KZ.WE.WF.WG.WR.WU.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS <wrap>
.SS0123456789.SS.SS.SS.SS.SS'

      A.US_A.US,'.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS.SS <wrap>
.SS.SS.SS.LDABCDEFGHIJKLMNOPQRSTUVWXYZ.SS !.QU.PS$.PC.AP''()*+,-./01'

      A.US_A.US,'23456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[]^_`abcdefghijklmnopqrstuvwxyz{|}~.WD.SS.SS'

      A.US_A.US,'.SS.SS.UD.ZA.ZB.ZC.ZD.ZE.ZF.ZG.ZH.ZI.ZJ.ZK.ZL.ZM.ZN.ZO.ZP.ZQ.ZR.ZS.ZT.ZU.ZV.ZW.ZX.ZY.ZZ <wrap>
.SS.SS.SS.SS.SS'

      XXXX_32 16.ROA.US
      .BXSINK.BXEX 2 6.RO 'A.US    .LD.LDRANK'
      )SAVE
27-SEP-1983 13:01:49 62 BLKS
```

3a. The user runs ASDRIVER in APLT2V:SF2VX4 on DSK:A.AAS.

- A.AAS is created in TTY character set with []PW == 80 via the following statements:

```
[1] (.IO10) .OQ[2] chan
[2] (3 4.RO'ABCD') .OQ[2] chan
[3] (4 .RO .FL 2*33) .OQ[2] chan
[4] (5 5.RO 0 1 0 1 0) .OQ[2] chan
```

- ASDRIVER runs under APLSF, producing messages. Note that ASDRIVER is dyadic.

```
 1  ASDRIVER 'A,AAS'
" BUILDING A,A2V [4,244]/AS 1983 9 14 12 22 31 440
" ALL DONE WITH INPUT FILE:  A,AAS /AS
```

- DSK:A.A2V is a VAX APL)INPUT script that reconstructs A.AAS on the VAX.

```
" BUILDING A,A2V [4,244]/AS 1983 9 27 11 2 41 882
WS.USCHAN_.BXASS 'A.AAS/AS'
A.US_''
A.US_A.US,'1 2 3 4 5 6 7 8 9 10'
RECORD_20.ROA.US
RECORD.OQ[3]WS.USCHAN
A.US_''
A.US_A.US,'ABCD'
RECORD_4.ROA.US
RECORD.OQ[3]WS.USCHAN
A.US_''
A.US_A.US,'ABCD'
RECORD_4.ROA.US
RECORD.OQ[3]WS.USCHAN
A.US_''
A.US_A.US,'ABCD'
RECORD_4.ROA.US
RECORD.OQ[3]WS.USCHAN
A.US_''
A.US_A.US,'8589934592 8589934592 8589934592 8589934592'
RECORD_43.ROA.US
RECORD.OQ[3]WS.USCHAN
A.US_''
A.US_A.US,' 0 1 0 1 0'
RECORD_10.ROA.US
RECORD.OQ[3]WS.USCHAN
A.US_''
A.US_A.US,' 1 0 1 0 1'
RECORD_10.ROA.US
RECORD.OQ[3]WS.USCHAN
A.US_''
A.US_A.US,' 0 1 0 1 0'
RECORD_10.ROA.US
RECORD.OQ[3]WS.USCHAN
A.US_''
A.US_A.US,' 1 0 1 0 1'
RECORD_10.ROA.US
RECORD.OQ[3]WS.USCHAN
A.US_''
A.US_A.US,' 0 1 0 1 0'
RECORD_10.ROA.US
RECORD.OQ[3]WS.USCHAN
.BXDAS WS.USCHAN
" ALL DONE WITH INPUT FILE:  A,AAS /AS
```

3b. The user runs ISDRIVER in APLT2V:SF2VX4 on WRK:A.AIS.

- A.AIS is constructed via the following statements:

```
[1] (.IO10) .OQ chan
[2] (3 4.RO'ABCD') .OQ chan
[3] (4 .RO .FL 2*33) .OQ chan
[4] (5 5.RO 0 1 0 1 0) .OQ chan
[5] ((300 .RO 1).RO.NG77).OQ chan
```

- ISDRIVER runs under APLSF, producing messages. Note that APL002: must be defined on the VAX for A.I2V to work there.

```
          ISDRIVER 'WRK:A,AIS'
" BUILDING A,I2V [4,244]/AS 1983 9 14 12 22 31 440
" *** 'WRK:' REPLACED BY 'APL002:'
" *** BIG INTEGER IN RECORD
" ALL DONE WITH INPUT FILE:  WRK:A,AIS /IS
```

- DSK:A.I2V is a VAX APL)INPUT script that reconstructs A.AIS on the VAX. APL002: must be defined on the VAX for A.I2V to work there.

```
" BUILDING A,I2V [4,244]/AS 1983 9 27 11 2 44 572
" *** 'WRK:' REPLACED BY 'APL002:'
WS.USCHAN_.BXASS 'APL002:A.AIS/IS'
A.US_.IO0
A.US_A.US,1 2 3 4 5 6 7 8 9 10
RECORD_10.ROA.US
RECORD.OQWS.USCHAN
A.US_''
A.US_A.US,'ABCDABCDABCD'
RECORD_3 4.ROA.US
RECORD.OQWS.USCHAN
A.US_.IO0
" *** BIG INTEGER IN RECORD
A.US_A.US,8589934592 8589934592 8589934592 8589934592
RECORD_4.ROA.US
RECORD.OQWS.USCHAN
A.US_.IO0
A.US_A.US,0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
RECORD_5 5.ROA.US
RECORD.OQWS.USCHAN
A.US_.IO0
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
```

```
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
A.US_A.US,1 1 1 1 1 1 1 1 1 1
.LD.LDRANK_300.ROA.US
A.US_.IOO
A.US_A.US,.NG77
RECORD_.LD.LDRANK.ROA.US
RECORD.OQWS.USCHAN
.BXDAS WS.USCHAN
" ALL DONE WITH INPUT FILE:  WRK:A,AIS /IS
```

3c. The user runs BSDRIVER in APLT2V:SF2VX4 on A.ABI[4,244].

- o A.ABI was constructed via the following statements:

```
[1] (.IO10) .OQ chan, 1
[2] (3 4.RO'ABCD') .OQ chan, 1
[3] (4 .RO .FL 2*33) .OQ chan, 1
[4] (5 5.RO 0 1 0 1 0) .OQ chan, 1
```

- o BSDRIVER runs under APLSF, producing messages. Note that APL003: must be defined on the VAX for A.B2V to work there.

```
            BSDRIVER 'A,ABI[4,244]'
" BUILDING A,B2V [4,244]/AS 1983 9 14 12 22 31 440
" *** '[4,244]' REPLACED BY 'APL003:'
" *** BIG INTEGER IN RECORD
" ALL DONE WITH INPUT FILE:  A,ABI[4,244] /BS
```

- o DSK:A.B2V is a VAX APL)INPUT script that reconstructs A.ABI as a /IS file on the VAX (since /BS files do not exist in VAX APL). APL003: must be defined on the VAX for A.B2V to work there.

```
" BUILDING A,B2V [4,244]/AS 1983 9 27 11 2 46 285
" *** '[4,244]' REPLACED BY 'APL003:'
WS.USCHAN_.BXASS 'APL003:A.ABI/IS'
A.US_.IOO
A.US_A.US,1 2 3 4 5 6 7 8 9 10
RECORD_10.ROA.US
RECORD.OQWS.USCHAN
A.US_''
A.US_A.US,'ABCDABCDABCD'
RECORD_3 4.ROA.US
RECORD.OQWS.USCHAN
A.US_.IOO
" *** BIG INTEGER IN RECORD
A.US_A.US,8589934592 8589934592 8589934592 8589934592
RECORD_4.ROA.US
RECORD.OQWS.USCHAN
A.US_.IOO
A.US_A.US,0 1 0 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 1 0
RECORD_5 5.ROA.US
RECORD.OQWS.USCHAN
.BXDAS WS.USCHAN
" ALL DONE WITH INPUT FILE:  A,ABI[4,244] /BS
```

3d. The users runs DADRIVER in APLT2V:SF2VX4 on WRK:A.ADA[4,244].

- o A.ADA was constructed via the following statements:

```
[0] )CREATE 100 A.ADA
[1] (.IO10) .OQ[1] chan
[2] (3 4.RO'ABCD') .OQ[2] chan
[3] (4 .RO .FL 2*33) .OQ[3] chan
[4] (5 5.RO 0 1 0 1 0) .OQ[4] chan
```

- o DADRIVER runs under APLSF, producing messages. Note that APL004: must be defined on the VAX for A.D2V to work there.

```
            DADRIVER 'WRK:A,ADA[4,244]'
" BUILDING A,D2V [4,244]/AS 1983 9 14 12 22 28 949
" *** 'WRK:[4,244]' REPLACED BY 'APL004:'
" *** BIG INTEGER IN RECORD
" ALL DONE WITH INPUT FILE:  WRK:A,ADA[4,244] /DA
```

- o DSK:A.D2V is a VAX APL)INPUT script that reconstructs A.ADA on the VAX. APL004: must be defined on the VAX for A.B2V to work there.

```
" BUILDING A,D2V [4,244]/AS 1983 9 27 11 2 48 105
" *** 'WRK:[4,244]' REPLACED BY 'APL004:'
WS.USCHAN_.BXASS 'APL004:A.ADA/DA'
A.US_.IOO
A.US_A.US,1 2 3 4 5 6 7 8 9 10
RECORD_10.ROA.US
RECORD .OQ[1] WS.USCHAN
A.US_''
A.US_A.US,'ABCDABCDABCD'
RECORD_3 4.ROA.US
RECORD .OQ[2] WS.USCHAN
A.US_.IOO
" *** BIG INTEGER IN RECORD
A.US_A.US,8589934592 8589934592 8589934592 8589934592
RECORD_4.ROA.US
RECORD .OQ[3] WS.USCHAN
```

```
A.US_.IOO
A.US_A.US,0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
RECORD_5 5.ROA.US
RECORD .OQC4] WS.USCHAN
.BXDAS WS.USCHAN
" ALL DONE WITH INPUT FILE:  WRK:A,ADAC4,244] /DA
```

4. The text files created by each DRIVER are moved from the -20 to the VAX by some means. Facilities for moving files are not part of this migration tool. The user could, for example, use DECnet file transfer or ANSI labeled magnetic tape.

5. Once the text files reside on the VAX, the SUBMIT function in the VAX APL workspace, SF2VX5.APL, can be used to create a command procedure that processes each text file as a)INPUT (/KEY) file to VAX APL.

- The following files reside in the default directory:

A.A2V	text image of A.AAS on the -20
A.B2V	text image of A.ABI on the -20
A.D2V	text image of A.ADA on the -20
A.I2V	text image of A.AIS on the -20
TEST0.W2V	text image of TEST0.APL on the -20

- SUBMIT in SF2VX5.APL runs on the following text files:

```
      )LOAD SF2VX5
SAVED 28-SEP-1983 10:48:06 50 BLKS
      SUBMIT
ENTER FILE NAMES TO BE INCLUDED :C*.*;*] X:A.%2V

BUILDING DCL FILE SUB006.COM ***

$!  INCLUDING APLD$:CUSERJA.A2V;1

THE LOGICAL APL003 NEEDS TO BE DEFINED FOR FILE APLD$:CUSERJA.B2V;1
EQUIVALENCE NAME?:C<CR> IF NONE ] <cr>
$!  INCLUDING APLD$:CUSERJA.B2V;1

THE LOGICAL APL004 NEEDS TO BE DEFINED FOR FILE APLD$:CUSERJA.D2V;1
EQUIVALENCE NAME?:C<CR> IF NONE ] USRD$:CUSER]
THE LOGICAL APL004 WILL BE DEFINED AS USRD$:CUSER]
$!  INCLUDING APLD$:CUSERJA.D2V;1

THE LOGICAL APL002 NEEDS TO BE DEFINED FOR FILE APLD$:CUSERJA.I2V;1
EQUIVALENCE NAME?:C<CR> IF NONE ] <cr>
$!  INCLUDING APLD$:CUSERJA.I2V;1

DCL FILE SUB006.COM COMPLETED ***
      SUBMIT
ENTER FILE NAMES TO BE INCLUDED :C*.*;*] TEST0.W2V

BUILDING DCL FILE SUB007.COM ***

THE LOGICAL APL000 NEEDS TO BE DEFINED FOR FILE USRD$:CUSER.APLJTEST0
EQUIVALENCE NAME?:C<CR> IF NONE ] USRD$:CUSER]
THE LOGICAL APL000 WILL BE DEFINED AS USRD$:CUSER]
THE LOGICAL APL001 NEEDS TO BE DEFINED FOR FILE USRD$:CUSER.APLJTEST0
EQUIVALENCE NAME?:C<CR> IF NONE ] <cr>
THE LOGICAL APL002 NEEDS TO BE DEFINED FOR FILE USRD$:CUSER.APLJTEST0
EQUIVALENCE NAME?:C<CR> IF NONE ] <cr>
THE LOGICAL APL004 NEEDS TO BE DEFINED FOR FILE USRD$:CUSER.APLJTEST0
EQUIVALENCE NAME?:C<CR> IF NONE ] <cr>
THE LOGICAL APL006 NEEDS TO BE DEFINED FOR FILE USRD$:CUSER.APLJTEST0
EQUIVALENCE NAME?:C<CR> IF NONE ] <cr>
THE LOGICAL APL008 NEEDS TO BE DEFINED FOR FILE USRD$:CUSER.APLJTESTC
EQUIVALENCE NAME?:C<CR> IF NONE ] <cr>
$!  INCLUDING USRD$:CUSER.APLJTEST0.W2V;1

DCL FILE SUB007.COM COMPLETED ***
      )OFF
```

- The command files built by SUBMIT can be run indirectly or in batch. The text file is processed by VAX APL and the file being migrated is created. Note that answering "<cr>" to the request to define a logical name leaves the logical name as previously defined, either by a previous request in this SUBMIT execution or by the user externally to SUBMIT.

SUB006.COM built in the previous step looks like the following:

```
$!  BUILDING SUB006.COM/AS 1983 9 28 16 3 54 490
$SET NOON
$!  INCLUDING APLD$:CUSERJA.A2V;1
$ APL/TERM=TTY/NOSILENT
)OUTPUT SYS$OUTPUT:/TTY
)MAXCORE 1500
)INPUT APLD$:CUSERJA.A2V;1/KEY
)OFF HOLD
$!
$!  INCLUDING APLD$:CUSERJA.B2V;1
$ APL/TERM=TTY/NOSILENT
)OUTPUT SYS$OUTPUT:/TTY
)MAXCORE 1500
)INPUT APLD$:CUSERJA.B2V;1/KEY
)OFF HOLD
$!
$!  INCLUDING APLD$:CUSERJA.D2V;1
$ DEFINE APL004 USRD$:CUSER]
$ APL/TERM=TTY/NOSILENT
)OUTPUT SYS$OUTPUT:/TTY
)MAXCORE 1500
)INPUT APLD$:CUSERJA.D2V;1/KEY
)OFF HOLD
$!
$!  INCLUDING APLD$:CUSERJA.I2V;1
$ APL/TERM=TTY/NOSILENT
)OUTPUT SYS$OUTPUT:/TTY
)MAXCORE 1500
```

```
)INPUT APLD$:[USERJA.I2V;1/KEY
)OFF HOLD
$!
```

SUB007.COM built in the previous step looks like the following:

```
$!  BUILDING SUB007.COM/AS 1983 9 28 16 4 54 780
$SET NOON
$!  INCLUDING APLD$:[USERJTEST0.W2V;1
$ DEFINE APL000 USRD$:[USERJ
$ APL/TERM=TTY/NOSILENT
)OUTPUT SYS$OUTPUT:/TTY
)MAXCORE 1500
)INPUT APLD$:[USERJTEST0.W2V;1/KEY
)OFF HOLD
$!
```

4.0 INCOMPATIBILITES

Incompatibilities exist between APLSF and VAX APL. The action that the migration tool takes on each of these incompatibilties is listed below. "SFTOVX takes no action" means that the migration tool does not identify the incompatibility, nor does SFTOVX replace it to simulate APLSF behavior. Any transformations done by the migration tool are reported to the user when the tool runs.

NOTE

These transformations do not take place within quoted literals.

- Escape mode does not exist in VAX APL.

  "@" on input to VAX APL in TTY mode will remain an "@". "@R", for example, are not translated to .RO.

  SFTOVX takes no action.

- Tabs in data and in files will remain as tabs. They are not converted to spaces in VAX APL.

- .Z@ is not recognized by VAX APL as the character underscored delta.

  SFTOVX takes no action on .Z@. Underscored delta is output in APL characters by the migration tool. In VAX APL, .UD is underscored delta.

- In APLSF, .RU is a "U" with the feet pointing to the right; .LU is a "U" with the feet pointing to the left. In VAX APL, .RU is a "U" with the feet pointing to the left and .LU is a "U" with the feet pointing to the right (in other words, the directions of the feet are opposite what they are in APLSF).

  SFTOVX will output the "U" with the feet pointing to the right as "left U" (.LU) and the "U" with the feet pointing to the left as "right U" (.RU). The APL graphic from APLSF will be the same as the APL graphic inside VAX APL.

- Dyadic $ is not supported in VAX APL.

  SFTOVX converts a $ primitive function to .BXFMT.

- Monadic .EN, .DE and .OM are not supported in VAX APL.

  SFTOVX converts the primitive functions .EN, .DE, and .OM to the user-defined (DEC-provided) functions .LD.USEN, .LD.USDE, and .LD.USOM which simulate their behavior under APLSF.

- If A is an integer in A/B in APLSF, it is an error. In VAX APL, it is the function replicate.

  SFTOVX takes no action.

- Both monadic and dyadic .DQ are fuzzy in VAX APL but not in APLSF.

  SFTOVX replaces the primitive function .DQ with .LD.USDQ, which sets .BXCT to 0.222044605.NG15, APLSF's internal value for singularity testing (established in APLSF v2 in edit 257) before executing .DQ.

- Dyadic .AB is fuzzy in VAX APL but not in APLSF.

  SFTOVX replaces the primitive function .AB with .LD.USAB, which sets the local .BXCT to 0 if invoked dyadically.

- Floor and Ceiling apply .BXCT differently in VAX APL than in APLSF.

  SFTOVX takes no action.

o A negative number raised to a non-integer power by means of dyadic * is an error in VAX APL; it gives an answer in APLSF.

SFTOVX takes no action.

o Output from .FM format is different in VAX APL.

SFTOVX takes no action.

o VAX APL uses .BXCT of "absolute fuzz" to test for a near-integer.

SFTOVX takes no action.

o Monadic transpose of a scalar is a vector in APLSF; it is a scalar in VAX APL.

SFTOVX takes no action.

o Monadic and dyadic ! (Beta and Gamma functions) give different, more accurate answers in VAX APL than in APLSF.

SFTOVX takes no action.

o I-beams do not exist on the VAX.

SFTOVX replaces the primitive function .IB with .LD.USIB, which simulates most of the documented I-beams and none of the undocumented ones:

17	symbol table size, entries	not simulated
18	condition of the ws	always return 0 == ok
19	keying time	always return 0 == .BXAI[4]
20	time of day in 60th seconds	simulated
21	the CPU time in 60th seconds	simulated
22	WS available in words	return .BXWA % 4
23	system job number	return .BXUL
24	APL sign-on time in 60th seconds since midnight	something like .bxAI[3]
25	current date	.BXTSC2 3 1] reformatted
26	current line number	returns 1^1.DA.BXLC
27	all line numbers	returns 1.DA.BXLC
28	terminal character set	0 == APL; 1 == TTY
29	user's ppn	return .BXAI[1]
30	clear the SI stack	execute .BXRESET
31	number of statements executed	get info from)CHARGE
32	number of operations executed	get info from)CHARGE
33	TOPS-10 only kilo-core seconds	return 0

o .BXASCII is not supported in VAX APL.

SFTOVX replaces .BXASCII with a niladic function .LD.USASCII, which returns 128 VAX APL .BXAV codes that resemble ASCII.

o .BXAV is different in VAX APL than in APLSF.

SFTOVX replaces .BXAV with a niladic function .LD.USAV, which returns 512 VAX APL .BXAV codes that resemble .BXAV in APLSF.

o .BXRENAME may behave differently in VAX APL than it does in APLSF.

SFTOVX replaces .BXRENAME with the user-defined function .LD.USRENAME which attempts to simulate the behavior of .BXRENAME in APLSF.

o .BXENQ, DEQ, MTP, APPEND and FCM are not in VAX APL.

SFTOVX replaces these system names with user-defined functions, for example, .LD.USENQ, which executes .BXBREAK '.BXENQ not supported in VAX APL'.

o In APLSF, certain system variables and all system functions can be localized in function headers or assigned to. In VAX APL these system variables are niladic system functions so they cannot be localized or assigned to.

The system names involved are the following:

AV	ALPHA	ALPHAU	NUM	CTRL	LC
AI	TS	TT	UL	WA	

SFTOVX removes these and all system function names from function header local lists. It replaces assignment to them with assignment to '.BXSINK_'. Therefore the only quad names allowed in locals lists are the following:

AUS	IO	RL	CT	TIMELIMIT
SF	ERROR	PP	GAG	TIMEOUT
PW	TRAP	LX		

o Several system commands in APLSF are not in VAX APL.

The system commands involved are the following:

MODE	TABS	LIB and DROP switches	BLOT
C	CALL	CREATE	ECHO
R	RUN	SEAL	TIME

For any of these to be used, they would have to be executed, either with .XQ or .EP. Note that .EP returns 0 21 .ro 0 and .XQ

dies.

SFTOVX takes no action.

- In APLSF, the shape of the result of .EP')cmd' is a matrix of .BXPW columns (if the result contains more than 1 row). In VAX APL, the result is a vector with the rows delimited by CR-LF.

  SFTOVX takes no action. Note that .BXBOX in VAX APL converts a vector of lines delimited by CR-LF into a matrix.

- Passwords on workspaces and /DA files are spelled "-password" in APLSF. The syntax is "/PASSWORD=password" in VAX APL.

  Passwords may appear in)LOAD,)SAVE,)COPY,)PCOPY, .BXASS,)CREATE, .BXQLD, .BXQCO, .BXQPC.

  SFTOVX replaces .BXQLD, .BXQPC, .BXQCO, and .BXASS with, for example, .LD.USQLD which checks for a "-password" at run-time and change it to "/PASSWORD=password". SFTOVX takes no action on the system commands.

- .BXTT returns different answers in VAX APL than in APLSF.

  SFTOVX replaces the .BXTT system variable with the niladic user-defined function .LD.USTT. This produces the following results:

if VAX APL .BXTT is	then .LD.USTT returns
2 == TTY	0 == TTY
3 == GIGI	3 == KEY
4 == LA	2 == LA or Tektronix
5 == KEY	3 == KEY
6 == BIT	4 == BIT
8 == VT102-PA/RA	3 == KEY
11 == Tektronix 4013	2 == LA or Tektronix
12 == Tektronix 4015	2 == LA or Tektronix

  Note that VAX APL does not support TTYCOM, ONTEL or 2741 APL terminals, which APLSF does.

- .FM and $ both output "-" for negative numbers in TTY mode in APLSF. VAX APL outputs "-" only if .BXNG is 0.

  SFTOVX does not set .BXNG in the VAX APL workspace since it also affects .BXFI and .BXVI.

- T.LDf and S.LDf set the trace and stop vectors in VAX APL. These names are valid identifiers in VAX APL.

  SFTOVX leaves these as identifiers; the trap and stop settings for user-defined functions are not transferred.

- /BS, /BS* and /BU binary files are not supported in VAX APL.

  .BXASS is replaced by .LD.USASS, which executes .BXBREAK if the .BXASS fails. .BXASS reports a "not supported" error message for the following facilities: passwords, [ppn], /BS, /DUMP.

- .BXCLS, DAS, CHS, DVC and FLS on .IOO return information on all 12 channels.

  SFTOVX replaces each of these system names with user-defined functions. For example, .LD.USCLS checks to see if its argument is an empty vector: if it is, it executes .BXCLS .BXCHANS and returns .IOO (these functions are not quiet in APLSF). They also change channel numbers of the form 100+n to -n.

- .BXCIQ and COQ take different arguments and produce different results in VAX APL than in APLSF.

  .BXCIQ and COQ deal with internal representations, which are different in the VAX than in APLSF. Therefore SFTOVX replaces these system names with user-defined functions. For example, .LD.USCIQ executes .BXBREAK 'CIQ IN VAX APL INCOMPATIBLE WITH APLSF'.

- A trailing "lamp" on output to the terminal suppresses the next CR-LF in APLSF. It is output simply as a trailing "lamp" in VAX APL.

  SFTOVX takes no action.

- STOPSET can be trapped in VAX APL. It cannot in APLSF.

  SFTOVX takes no action.

- APLSF does very little shape-checking on indexed assignment.

  SFTOVX takes no action.

- The character representation of a number can overflow on the VAX.

  Numbers are transferred as character strings with .BXPP set to 18. Rounding may cause the string to be a number outside the range of

allowable numbers in VAX APL. A LIMIT ERROR is reported by VAX APL.

SFTOVX takes no action.

o Some functions are quiet in VAX APL, but not in APLSF.

The only function that is quiet that is not being transformed is .OQ. This function cannot be replaced with a user-defined function because it takes an axis argument, which user-defined functions can't do.

SFTOVX takes no action.

o .BXGAG values in APLSF are reversed in VAX APL.

SFTOVX takes no action.

o Valid value sets for certain system variables are different in VAX APL than they are in APLSF.

- .BXCT in an APLSF workspace is migrated to VAX APL as 2.328E.NG10 even if it is larger than this value.
- .BXPP in an APLSF workspace is migrated to VAX APL as 16 even if it is larger than this value.
- .BXRL in an APLSF workspace is migrated to VAX APL as .NG1+2*31 (2147483647).
- .BXTIMELIMIT in an APLSF workspace is converted from milliseconds to seconds and is migrated to VAX APL as 255 seconds even if .BXTIMELIMIT is bigger than that value.
- The internal null character (APLSF .BXAV [103]) is removed from .BXLX in an APLSF workspace before it is migrated to VAX APL.
- .BXPW in an APLSF workspace is migrated to VAX APL as 35 even if .BXPW is smaller than that value.

o The integer range in APLSF is from -2*35 to .NG1+2*35. In VAXAPL, the integer range is from -2*31 to .NG1+2*31. Any integer outside of the VAXAPL range will be stored as a floating point number under VAXAPL. The warning "BIG INTEGER IN name" is issued. When using base and represent to pack small values into integers, the packing may not work under VAXAPL.

5.0 ERROR MESSAGES

Error messages put out by the various pieces of SFTOVX:

o MIGRATE in SF2VX0

'MIG000,CTL THRU 999 IN USE'

MIGRATE cannot find a file spec of the form MIGnnn.CTL (where "nnn" is a 3 digit number) to use as the batch control file it wants to build. Execution will be suspended at this point - the user can take remedial action by getting rid of files named MIGnnn.CTL on DSK: and then typing _.GO .BXLC + 1 to restart MIGRATE.

The functions MIGRATE and PERFORM in SF2VX0 do not set .BXTRAP, which implies that software failures will display a message (only the 1st line since)ECHO is OFF) and suspend execution at the line in error. The user can then determine how to correct the failure.

o DRIVER in SF2VX1

'SI NOT EMPTY'

This is an informational message only. The SI stack in the workspace being migrated contains suspended and/or pendent functions. The workspace will be transferred but local copies of functions and variables will be dumped out to the input script.

DRIVER does not set .BXTRAP, which implies that software failures will display a message (only the 1st line since)ECHO is OFF) and suspend execution at the line in error. The user can then determine how to correct the failure.

o The file drivers, ASDRIVER, BSDRIVER, ISDRIVER and DADRIVER in SF2VX4

'ERROR ASSIGNING INPUT FILE: name'

all four drivers display this error if the file is not found

'ERROR n READING FROM FILE name COMPONENT i'
'ERROR n READING FROM FILE name RECORD i'

DADRIVER says COMPONENT; all others say RECORD, where "n" is an APLSF error number (see Appendix A in the VAX APL REFERENCE

MANUAL), "name" is the file the driver is trying to read, and "i" is the record or component number currently being read.

This message is then followed by the 1st line of .BXERROR, which will be the text for error "n".

```
'ERROR 69 READING FROM FILE name,ADA COMPONENT 0
FILE FORMAT NOT DIRECT ACCESS'

'ERROR 70 READING FROM FILE name,ais RECORD 1
FILE FORMAT NOT INTERNAL SEQUENTIAL'
```

Processing of the file stops when an error occurs and the output file (that is the VAX input script) being built is closed.

- All drivers check numeric data for integer values that are outside the integer range in VAX APL. The warning "BIG INTEGER IN name" means tnat such an integer was found in the variable name.

- While MIGRATE is interacting with the user, recoverable errors (such as FILE NOT FOUND) are reported and the user is prompted for alternate input. If tne error is not recoverable, the function suspends so the user can diagnose the failure.

- For functions that will most likely be executed in batch (for example,the drivers), errors are reported and APLSF then suspends execution. While running the control file built by MIGRATE from batch, the next command will be)OFF HOLD which returns the user to the "@" prompt. The user must examine the .LOG file from the batch job to see if the process was successful. If the function is run interactively, suspending the function gives the user the chance to diagnose the problem. Note that if a user-defined filter function is unsuccessful, the user can debug it if it is being run interactively.

30 March 1984

Dear APL Fanatic:

I'm writing you hoping that you'll write me back. As editor of the DECUS APL newsletter, The Special Character Set, I need articles to publish. If I can't get enough material to publish, DECUS probably will suggest our newsletter merge with COBOL, BASIC or even DATATRIEVE.

Almost anyone can write an article. It doesn't have to be long or heavy. Write an article about how your site uses APL or/and how it chose APL in the first place. Write an article about that handy function you just wrote. Write a testimonial about how APL changed your life.

Maybe you already have an article on an APL topic, but you published it somewhere else. Let me republish it if you can give me copyright permission.

I have printed a style sheet for any article you write. However, even if your article is in another format I'll take it. I can read RT-11 RX02 floppies. I have a volunteer typist. JUST GIVE ME SOMETHING TO PRINT!

The deadline for the next issue is 15 August 1984. However, if you have something now SEND IT NOW. I promise not to lose it, believe me.

Sincerely,

Doug Bohrer

Doug Bohrer
Bohrer and Company
903 Ridge Road, Suite 3
Wilmette, IL 60091 USA
Phone: 312-251-9449

SURVEY

In order to help the APL SIG serve your interests better, please fill out the following questionaire and return it to: Douglas Bohrer, 903 Ridge Road Suite 3, Wilmette, IL 60091, USA. Results will be published in the next issue.

1. I use the following computers (check all that apply):

___PDP-11/03 LSI-11 (Q-BUS)
___PDP-11/23 11/23+ (Q-BUS)
___Rainbow 100 series
___VAX-11
___DEC-20
___NONE OF THE ABOVE
___OTHER_____

___MINC
___Q-BUS 68000
___Professional 300 series
___Micro-VAX
___DEC-10

2. I have access to the following media (check all that apply):

___5.25 inch floppy
___8 inch RX01 floppy single density
___0.5 inch magnetic tape 800 BPI
___0.5 inch magnetic tape 6250 BPI
___Other portable media_____

___RL02
___8 inch RX02 floppy double density
___0.5 inch magnetic tape 1600 BPI

3. I use the following operating system(s) (check all that apply and fill in version):

___RT-11 version_____
___RSX-11 version_____
___RSTS/E version_____
___VAX/VMS version_____
___TOPS 20 version_____
___Others_____

___TSX+ version_____
___RSX-11M version_____
___UNIX version_____
___TOPS 10 version_____

4. I use the following APL's (check all that apply):

___APL-11 V1 (RT-11)
___APL-11 V2 (RSX-11)
___VAX-11 APL
___None of the above
___Others_____

___APL-11 V 2 (RT-11)
___APL-11 V 1 (RSTS/E)
___APL-SF (TOPS-10,TOPS-20)

5. I do most of my APL work using (pick one):

___APL special character set
___TTY mnemonics

6. This newsletter should accept paid advertising (circle one):

strongly agree strongly disagree

1 2 3 4 5 6 7

7. What do you want the APL SIG to do that it isn't?

8. What is the APL SIG doing that you think we should stop?

9. What should the APL SIG keep doing just the same?

==========

PLACE
POSTAGE
HERE

TO: Douglas Bohrer
903 Ridge Road Suite 3
Wilmette, IL 60091 USA