AD-A065 644   VIRGINIA POLYTECHNIC INST AND STATE UNIV BLACKSBURG --ETC F/G 9/2
TRANSFER OF APL WORKSPACES: A USEFUL SOLUTION.(U)
SEP 78   K RAUTENKRANZ, R J ORGASS   AFOSR-79-0021
UNCLASSIFIED   CS78006-T   AFOSR-TR-79-0238   NL

OF

AD
A065644

END
DATE
FILMED
5-79
DDC

LEVEL (4)

ADA065644

Computer Science Department

D D C
RECEIVED
MAR 14 1979
C

*Virginia Polytechnic Institute
and State University
Blacksburg, Virginia 24061*

9 03 12 009

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY
PRACTICABLE. THE COPY FURNISHED
TO DDC CONTAINED A SIGNIFICANT
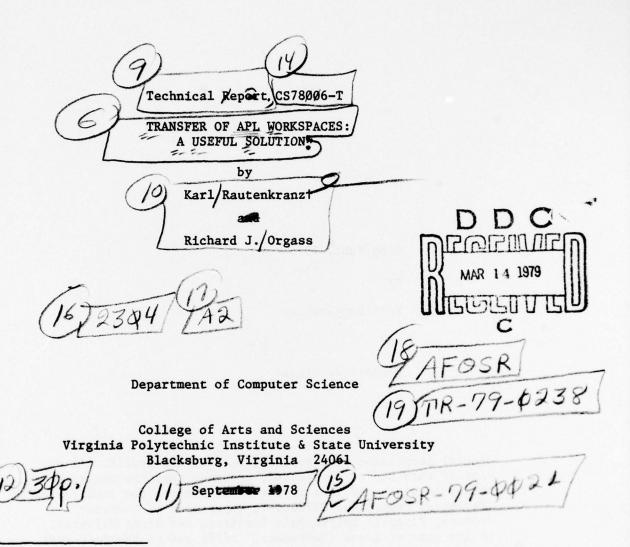NUMBER OF PAGES WHICH DO NOT
REPRODUCE LEGIBLY.

Technical Report CS78006-T

TRANSFER OF APL WORKSPACES: A USEFUL SOLUTION*

by

Karl/Rautenkranz†

and

Richard J./Orgass

16 2304   17 A2

18 AFOSR

Department of Computer Science

19 TR-79-0238

College of Arts and Sciences
Virginia Polytechnic Institute & State University
Blacksburg, Virginia 24061

12 30p.   11 September 1978   15 AFOSR-79-0021

DDC
RECEIVED
MAR 14 1979
C

411 098

ii

–A–

ABSTRACT

ABSTRACT. Most suppliers of *APL* have not
yet implemented the *STAPL* convention for trans-
mitting workspaces from one installation to
another. This report describes three workspace
representations which may be used on a
*DEC*system–10 for this purpose. Two are partial
implementations of the *STAPL* convention: one
for level 2 of the convention, the other for
level 3. The third representation is a terminal
transcript file which is to be used as an input
file. In addition, these representations may be
used to reduce the disk storage required for *APL*
workspaces on the *DEC*system–10.

1. Introduction

This report describes three representations of an APL workspace which can be used to move workspaces from or to a *DEC*system-10. The terminal transcript representation for moving from a *DEC*system-10 assumes that the destination machine is able to read input from a key paired *ASCII/APL* file as though the contents of the file were typed on a terminal. The second and third are implementations of levels 2 and 3 of the proposed *STAPL* convention for the interchange of workspaces [2]. Since the level 2 implementation is a key paired *ASCII/APL* representation, both the sending and receiving installations must be able to read *ASCII* files. For level 3 the workspace is converted to a bit stream and this representation is intended for exchanges between a *DEC*system-10 and a machine which does not support *ASCII* files.

Programs have been written for the *DEC*system-10 to produce the three representations and to create *APL* workspaces from the level 2 and level 3 representations of the *STAPL* convention. For the key paired *ASCII/APL* files, the programs described in [3] can be used to write the disk files on magnetic tape.

The terminal transcript and level 2 representations described here have been used to move workspaces from *DEC*system-10 to *APL/VS* on an *IBM 370/158* and to *MULTICS APL*. There is no claim, however, that the programs described in this report are completely correct and users are advised that there is no guarantee that their workspaces will be transferred correctly by any of the three methods described in this report. This caution is especially relevant to the procedures for the level 3 representation; these procedures have been tested on only a small sample of workspaces.

APLSF, as implemented on the *DEC*system-10, has several features which are not generally available on other *APL* implementations. It is desirable to remove most of these dependencies before transmitting the workspace. Section 2 describes some of the more important changes that should be made prior to converting the workspace.

Section 3 describes the terminal transcript representation and Section 4 the level 2 and level 3 canonical representations. Section 5 gives directions for using the workspace *GENTT* to produce a terminal transcript and Section 6 directions for using the workspace *GENCRV* to generate a workspace canonical representation at level 2 or level 3. Section 7 describes the workspace *GENWS* to create a workspace from a canonical representation file on the *DEC*system-10. Listings of these three workspaces are included as appendices.

2. Workspace Preparation

If there are locked functions in a workspace, these functions cannot be converted since it is impossible to obtain the character representations. Therefore, any locked functions which are to be transferred should be replaced by unlocked copies prior to the conversion.

APLSF differs from other _APL_ implementations in that two characters, carriage return and line feed, are used to separate lines; others use the single character: new line. For example, in _APLSF_:

```
'
        C←'
2       ρ,C
```
while in _APL/SV_:
```
'
        C←'
1       ρ,C
```

Since the character pair – carriage return, line feed – will not be converted into a single character by any of the three procedures discussed, it is desirable to replace all occurrences of carriage return-line feed with some other character. For the terminal transcript and level 2 representations, this must be done before a conversion is attempted; in fact, for both methods, all control characters must be replaced since the files generated are _ASCII_ character files.

APLSF has a number of system functions for communicating with the file system. These include □_ASS_, □_DAS_ and □_CLS_. These strings will cause errors in other implementations and should be replaced with user defined function names. More seriously in terms of transferring, the _APLSF_ input/output primitives (⊟, ⊟, ⊠) are not supported in this form in other implementations. These _APLSF_ characters will not be recognized as valid _APL_ characters and it is, therefore, important that these _APLSF_ primitives be replaced by user defined function names.

APLSF includes system commands in the domain of the execute function (⍕) but they are excluded in the IBM implementations. If such expressions remain in the texts of functions, they will result in an error when execution of the functions is attempted. Similarly, the _APLSF_ unquote function (∈ or ⊥) will cause errors and will need to be revised at some time.

The limited editing possible in _APL_ is not adequate for the sorts of modifications discussed above and the function _SOS_ described in [1] which permits entry into SOS from _APL_ is particularly useful.

3. Terminal Transcript Representation

Some _APL_ systems support a means of reading a file as though the lines of the file were typed on a terminal. The command)_INPUT_ in _APLSF_ is an example of this facility. If such a facility is available at the receiving site, it is possible to substantially reduce the computer resources required to transfer workspaces by simply writing a terminal transcript. There is, however, an important risk associated with the use of this representation. If the terminal transcript contains _APL_ characters not available in the destination system, a character error will occur during the reading of the file. If this occurs, the entire transcript file must be edited it may be necessary to start the workspace restoration from the beginning. Consequently,

if this representation is to be used, the preparation discussed in Section 2 is particularly important.

In this representation, the value of a variable is formatted as a character vector and the output to the file is a sequence of assignments to the variable, followed by commands to reshape and execute, if the variable is numeric. For a function the output is a sequence of assignments to a dummy variable of the raveled canonical representation of the function definition, followed by commands to reshape and to fix. The resulting file consists of a sequence of commands which, when read into an active workspace, result in the reconstruction of the original workspace. In addition, the file contains system commands, at the beginning, to clear and to rename the workspace and, at the end, to erase the dummy variable and to save the workspace.

As an illustration, consider the workspace *MUMBLE*:

```
      XX←2 5ρι10
      XX
   1   2   3   4    5
   6   7   8   9   10
      ∇Y
[1]   'MANY'
[2]   'SEVERAL'
[3]   ∇
```

For a file line length of 30, the terminal transcript of the workspace would be:

```
)CLEAR
)WSID MUMBLE
□PP←10
□PW←80
□CT←1.136364040407470365E¯13
□IO←1
□LX←''
□RL←0
∆∆RL←''
∆∆RL←∆∆RL,'30'
∆∆RL←≬∆∆RL
'∆∆RL'
XX←''
XX←XX,'1 2 3 4 5 6 7 8 9 10'
XX←≬XX
XX←2 5ρXX
'XX'
∆∆ET←''
∆∆ET←∆∆ET,'Y          ''MANY'''
∆∆ET←∆∆ET,'      ''SEVERAL'' '
∆∆ET←3 10ρ∆∆ET
∆∆ET←□FX ∆∆ET
)ERASE ∆∆ET
)SAVE
```

4.

4. STAPL Canonical Representation

The level 2 and level 3 representations described in this report are based on the *STAPL* proposed convention for the interchange of workspaces. A stream of canonical representation vectors, one for each of the individuals in the workspace, is generated and written to a file. For level 2 the stream consists of key paired *ASCII/APL* characters and for level 3 the individuals are encoded as a stream of binary numbers. This stream is appended to a stream identifier and a translation table. The translation table is used at the destination to convert the workspace stream of bits to *APL* characters and for level 3 the individuals are encoded as a stream of binary numbers. This stream is appended to a stream identifier and a translation table. The translation table is used at the destination to convert the workspace stream of bits to *APL* characters.

The *STAPL* proposal defines a canonical representation vector to be of the form:
    <length><type><name><space><rank><space><shape><space><elements>
In this report the types are restricted to:

    C   character variable
    N   numeric variable
    F   function
    P   pseudovariable used to describe the stream

For level 2 the stream which represents the workspace has the form:
    <wsid><crv$_1$>. . .<crv$_n$><end>

where   <wsid> is the canonical representation vector naming the work-
                    space
            <crv$_i$> are the canonical representation vectors of the individuals

            <end> is the stream termination vector

The level 2 canonical representation of the workspace *MUMBLE* would be:

```
16PWSID 1 6 MUMBLE9NⓊPP 0 109N
ⓊPW 0 808NⓊIO 0 130NⓊCT 0 1.13
6864040407470365‾139CⓊLX 1 0 8
NⓊRL 0 040FY 2 3 10 Y
'MANY'      'SEVERAL' 30NXX 2 2
5 1 2 3 4 5 6 7 8 9 108PEND 0
0
```

For level 3 the characters of the canonical representation of level 2 are further encoded as indices of a vector of the APL character set used by the sending maching. These indices are then expressed as binary numbers. For example, suppose the execute character (⍎) is the 110th element of this vector. Assuming the index origin to be zero, the execute character is represented or defined as the index 109 and would appear in the bit stream as 1101101. In this way, the *APL* characters of level 2 are expressed as binary numbers at level 3.

To reconstruct the *APL* characters from these binary numbers, the array *TRANSLATE* is defined. Each row of this array corresponds to an *APL* character available at the sending installation and the number of columns is the maximum number of overstrikes required to print the character (for characters requiring fewer than the maximum number of columns, the remaining columns contain the character *SPACE*). The array is then converted to a level 2 canonical representation vector and encoded as indices into the *ASCII/APL* transmission code vector. These indices are expressed in binary and this representation is attached to the beginning of the workspace bit stream. At the receiving site, this translation array is decoded using the *ASCII/APL* transmission code vector to a character string. This string is then converted to a character array which is used to construct an *APL* character vector. The *APL* character vector is then used to decode the remainder of the workspace stream to a level 2 representation.

5.   Generating a Terminal Transcript

The following directions for generating a terminal transcript of a workspace assume that the workspace has been prepared as described in Section 2. All of the directions use the workspace name *MUMBLE*, all occurrences of which should be replaced by the name of the workspace to be transferred.

After the workspace is prepared, load it as an active workspace:
)*LOAD MUMBLE*
Next copy the terminal transcript generator and begin execution as in the following transcript:
)*COPY APL:GENT*
      ΔΔINIT
DESTINATION WORKSPACE NAME: *MUMBLE*
OUTPUT LINE LENGTH: 80
The first prompt requests the name of the workspace to be used at the receiving site. If carriage return is entered, the name of the current workspace will be used (*MUMBLE*). The second prompt requests the maximum length of a file line. This will be the maximum number of *ASCII/APL* characters (the minimum should be at most 3 fewer). If carriage return is entered, the print width (□PW) of the active workspace will be used.

Since all output has been directed to the file, no reports are displayed on the terminal until the conversion is completed, at which time the following message is displayed:
   FILE CREATED: MUMBLE.TSP

This procedure is repeated for each workspace that is to be transmitted. If desired, the separate terminal transcript files may be combined into a single file. The disk files can be written on magnetic tape using the directions given in [3].

6.   Generating Canonical Representations

As in Section 5 the following directions assume that the workspace has been prepared for transmission. Again the directions use the workspace name *MUMBLE* which should be replaced by the workspace name to be transmitted.

After the workspace has been prepared, load with the command:
)*LOAD MUMBLE*
Next copy the canonical representation generator and begin execution:
)*COPY APL:GENVRV*
     Δ Δ*INIT*
DESTINATION WORKSPACE NAME: *MUMBLE*
LEVEL OF CONVERSION:
     ENTER 2(CHARACTER STREAM) OR 3(BIT STREAM):
The first prompt requests the name of the workspace to be used at the destination site. If a carriage return is entered, the name of the current workspace (*MUMBLE*) will be used. The second prompt requests whether level 2 or level 3 conversion is to be performed. If the number 3 is entered, the conversion begins. If the number 2 is entered, a third request is displayed
OUTPUT LINE LENGTH: 80
The third prompt requests the number of *APL* characters that are to be written on each output line. If a carriage return is entered, the print width ($\Box PW$) of the active workspace will be used. In deciding on the line length, allow for a substantial increase in the actual line length due to overstruck characters.

7.   Restoring Canonical Representations

The following directions assume that the canonical representation file to be converted is of a single workspace. Load the workspace generator and begin execution:
)*LOAD GENES*
     Δ Δ*INIT*
FILE TO BE CONVERTED: *MUMBLE*
TYPE OF FILE:
     ENTER 2(CHARACTER STREAM) OR 3(BIT STREAM):
The first prompt requests the name and extension (if any) of the file. The second prompt requests whether the file is an *ASCII/APL* character stream or a bit stream. If the number 2 is entered, a third request is displayed:
FILE LINE LENGTH:
The line length entered should be that used when the file was generated.

After this initial dialogue, the reconstruction of the workspace begins. For a bit stream, the initial phase is a check that the stream identifier is correct. If it is not, a message is displayed that the file does not conform to *STAPL* format conventions and the conversion is terminated. If it is correct, the translation table is reconstructed using the *ASCII/APL*

transmission vector and then written on a temporary file *MUMOnn.CRV*(where 'nn' is a two digit number). This file is then read to create the APL character vector. If there are *APL* characters which are not recognized by *APLSF*, a character error will occur at this point. Should this occur, the temporary file can be modified and execution resumed at ΔΔ*TCVT*[3].

Following this phase of the bit stream reconstruction, the workspace is converted to a character stream and written on the file *MUMOnn.CRV*.

The character stream is read in blocks of 10 lines and the individual canonical representation vectors are extracted. As the *APL* individuals are reconstructed from the canonical representation vectors, the names are displayed on the terminal.

When the individuals have been reconstructed, the conversion is terminated with the following messages:
THE ABOVE INDIVIDUALS HAVE BEEN RECREATED
TO COMPLETE THE WORKSPACE RECONSTRUCTION;
ENTER THE FOLLOWING:
)*WSID MUMBLE*
)*ERASE* ΔΔ*GENWS*
)*SAVE*
where ΔΔ*GENWS* is the group of global variables and functions of GENWS.

8.

APPENDIX A

WORKSPACE   GENNT

TERMINAL TRANSCRIPT GENERATOR

System Variables:

$\Box IO \leftrightarrow 1$

Variables:

$\Delta\Delta\underline{QS}$ is used in determining the number of *APL* characters to be written on an output line.

```
      ρΔΔQS
47
      ΔΔQS
A! ⍳↑↓AI⊖⊤⍉⍑⊕Φ⍟⍕▽⌈⌊⍅⍨⍫⍤ΔABCDEFGHIJKLMNOPQRSTUVWXYZ
```

```
      ∇  ΔΔINIT;ΔΔVS;ΔΔRL;⎕IO
[1]       ΔΔIO←⎕IO
[2]       ⎕IO←1
[3]       'DESTINATION WORKSPACE NAME: ⍙'
[4]       ΔΔVS←⍞
[5]       ΔΔVS←⍙ 5 ‾4[1+1≠ρΔΔVS,' ']↑')WSID ΔΔVS'
[6]       'TERMINAL TRANSCRIPT LINE LENGTH: ⍙'
[7]       ΔΔRL←⍞
[8]       ΔΔRL←⍙ 3 ‾5[1+1≠ρΔΔRL,' ']↑'⎕PW  ⍙ΔΔRL'
[9]       ΔΔGENW
      ∇
```

ΔΔGENW

```
      ∇  ΔΔGENW;ΔΔX;ΔΔF
[1]    ⍝  ΔΔGENW CONVERTS THE WORKSPACE ΔΔVS TO A TERMINAL
[2]    ⍝   TRANSCRIPT ON THE KEY PAIRED ACSII FILE ΔΔVS.TSP
[3]       ⍙')OUTPUT ',ΔΔVS,'.TSP'
[4]       ')CLEAR'
[5]       ')WSID ',ΔΔVS
[6]       '⎕PP←',⍕⎕PP
[7]       '⎕PW←',⍕⎕PW
[8]       ⎕PP←18
[9]       ⎕PW←ΔΔRL
[10]      '⎕CT←',⍕⎕CT
[11]      '⎕IO←',⍕ΔΔIO
[12]      '⎕LX←''',⎕LX,''''
[13]      '⎕RL←',⍕⎕RL
[14]   ⍝  CONVERT VARIABLES IN WORKSPACE
[15]      ΔΔX←⎕NL 2
[16]      ΔΔX←(∧/ΔΔXv.≠((1+ρΔΔX),5)↑⍒ 5 5 ρ'ΔΔVS ΔΔX   ΔΔCE ΔΔQS ΔΔIO ')⌿ΔΔX
[17]   ΔΔGENW1:→(0=ρ,ΔΔX)/ΔΔGENW2
[18]      ΔΔTTRV ΔΔTRIM ΔΔX[1;]
[19]      ΔΔX← 1 0 ↓ΔΔX
[20]      →ΔΔGENW1
[21]   ⍝  CONVERT FUNCTIONS IN WORKSPACE
[22]   ΔΔGENW2:ΔΔX←⎕NL 3
[23]      ΔΔF←'ΔΔGENW ΔΔTRIM ΔΔSTRP ΔΔTTRV ΔΔTTRF ΔΔDPLQ ΔΔQDCR ΔΔINIT'
[24]      ΔΔX←(∧/ΔΔXv.≠((1+ρΔΔX),8)↑⍒ 8 7 ρΔΔF)⌿ΔΔX
[25]   ΔΔGENW3:→(0=ρ,ΔΔX)/ΔΔGENW4
[26]      ΔΔTTRF ΔΔTRIM ΔΔX[1;]
[27]      ΔΔX← 1 0 ↓ΔΔX
[28]      →ΔΔGENW3
[29]   ΔΔGENW4:')ERASE ΔΔFT'
[30]      ')SAVE'
[31]      ⍙')OUTPUT'
[32]      'FILE CREATED: ',ΔΔVS,'.TSP'
      ∇
```

```
     ∇  ΔΔTTRV ΔΔN;ΔΔC;ΔΔS;ΔΔL
[1]   ⍝   ΔΔTTRV PRODUCES A TERMINAL TRANSCRIPT OF A
[2]   ⍝   VARIABLE ASSIGNMENT
[3]    ΔΔC←⍎ΔΔN
[4]    ΔΔS←▼⍴ΔΔC
[5]    ΔΔC←ΔΔDPLQ▼,ΔΔC
[6]   ⍝   ASSIGN TO VARIABLE NAME ITS VALUE REPRESENTED
[7]   ⍝   AS A CHARACTER VECTOR
[8]    ΔΔN,'←'''''
[9]   ΔΔTTRV1:→(0=⍴ΔΔC)/ΔΔTTRV2
[10]  ⍝   DETERMINE LENGTH OF CHARACTER STRING TO FILL FILE LINE
[11]   ΔΔW←¯1+((+\1+(ΔΔC∈ΔΔQS)×2)>(⍴ΔΔC)⌊⎕PW-4+2×⍴,ΔΔN)⍳1
[12]  ⍝   CHECK CHARACTER STRING DOES NOT
[13]  ⍝   CONTAIN ODD NUMBER OF QUOTES
[14]   ΔΔW←ΔΔW-0≠2|+/(ΔΔW↑ΔΔC)=''''
[15]   ΔΔN,'←',ΔΔN,',''',(ΔΔW↑ΔΔC),''''
[16]   ΔΔC←ΔΔW↓ΔΔC
[17]   →ΔΔTTRV1
[18]  ⍝   IF NUMERIC, CONVERT VALUE
[19]  ΔΔTTRV2:→(' '=1↑0⍴,⍎ΔΔN)/ΔΔTTRV3
[20]   ΔΔN,'←⍎',▼⍎ 4 ¯1[1+0=⍴,⍎ΔΔN]↑'ΔΔN 0'
[21]  ⍝   RESHAPE VARIABLE
[22]  ΔΔTTRV3:⍎(0≠⍴ΔΔS)/'ΔΔN,''←'',ΔΔS,''⍴'',ΔΔN'
[23]   '''',ΔΔN,''''
     ∇
```

```
                              ΔΔTTRF


     ∇  ΔΔTTRF ΔΔA;ΔΔW;ΔΔS
[1]   ⍝   ΔΔTTRF PRODUCES A TERMINAL TRANSCRIPT OF THE FUNCTION ΔΔA
[2]    ΔΔA←ΔΔQDCR ΔΔA
[3]    ΔΔS←⍴ΔΔA
[4]    ΔΔA←ΔΔDPLQ,ΔΔA
[5]    'ΔΔET←'''''
[6]   ΔΔTTRF1:→(0=⍴ΔΔA)/ΔΔTTRF2
[7]   ⍝   DETERMINE LENGTH OF CHARACTER STRING TO FILL FILE LINE
[8]    ΔΔW←¯1+((+\1+(ΔΔA∈ΔΔQS)×2)>(⍴ΔΔA)⌊⎕PW-12)⍳1
[9]   ⍝   CHECK THAT CHARACTER STRING DOES NOT
[10]  ⍝   CONTAIN ODD NUMBER OF QUOTES
[11]   ΔΔW←ΔΔW-0≠2|+/(ΔΔW↑ΔΔA)=''''
[12]   'ΔΔET←ΔΔET,''',(ΔΔW↑ΔΔA),''''
[13]   ΔΔA←ΔΔW↓ΔΔA
[14]   →ΔΔTTRF1
[15]  ⍝   RESHAPE FUNCTION DEFINITION AS A CANONICAL REPRESENTATION
[16]  ΔΔTTRF2:'ΔΔET←',(▼ΔΔS),'⍴ΔΔET'
[17]  ⍝   FIX THE FUNCTION
[18]   'ΔΔET←⎕FX ΔΔET'
     ∇
```

```
GENCRV                        ΔΔINIT

     ∇  ΔΔINIT;ΔΔWS;ΔΔRL;ΔΔCH;ΔΔLC;ΔΔFS;ΔΔIO
[1]     ΔΔIO←⎕IO
[2]     ⎕IO←0
[3]     'DESTINATION WORKSPACE NAME: ▲'
[4]     ΔΔWS←,⍞
[5]     ≢(1=ρΔΔWS,' ')/'ΔΔWS←≢'')WSID'''
[6]     'LEVEL OF CONVERSION:'
[7]     '          ENTER 2(CHARACTER STREAM) OR 3(BIT STREAM): ▲'
[8]     ΔΔLC← ¯2+≢⍞
[9]     →ΔΔLC/ΔΔINIT3
[10]  ⍝  INITIALIZATION FOR CHARACTER STREAM FILE
[11]    'FILE LINE LENGTH: ▲'
[12]    ΔΔRL←⍞
[13]    ΔΔRL←≢ 3 ¯5[1≠ρΔΔRL,' ']↑'⎕PW  ≢ΔΔRL'
[14]    ΔΔCH←⎕ASS ΔΔWS,'.CRV/AS'
[15]    ΔΔBV←''
[16]    →ΔΔINIT1
[17]  ⍝  INITIALIZATION FOR BIT STREAM FILE
[18]  ΔΔINIT3:ΔΔCH←⎕ASS ΔΔWS,'.BRV/BU'
[19]  ⍝  DETERMINE FRAMESIZE
[20]    ΔΔFS←⌈2⍟ρΔΔAV
[21]    ΔΔRL←⎕PW
[22]  ΔΔINIT1:ΔΔCCRV
[23]    ⎕DAS ΔΔCH
[24]    'CONVERSION COMPLETED:'
[25]    '   FILE CREATED: ',ΔΔWS, 7 ¯7[ΔΔLC]↑'.CRV/AS  .BRV/BU'
     ∇
```

12.

APPENDIX B

WORKSPACE   GENCRV

CANONICAL REPRESENTATION GENERATOR

System Variables:

$$\Box IO \leftrightarrow 0$$

Variables:

$\Delta\Delta\underline{A}\underline{V}$ is the *APL* character vector

```
      ρ∆∆AV
142
      ∆∆AV
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789,./[]{⊢←◊×¨¯<≤=≥>≠∨∧-÷
$→↑★○⍳↓↑~ρ∈ω?α⌈⌊_∇∆∘'⎕()}\:;|⊤⊥∪∩⊃⊆⍺!$∇∆⍁⊙⍣⍀⍥≢≠±⍂⊟⍞⊞⊟
∆ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

$\Delta\Delta\underline{B}\underline{V}$ is the level 3 representation of the stream identifier and the trans-
lation table. As described in the text, the translation table is an array,
the rows corresponding to the *APL* characters in $\Delta\Delta\underline{A}\underline{V}$. The number of col-
umns is the maximum number of overstrikes required to print the *APL* char-
acters. Thus

        row 2 would be *SPACE*,'*A*'

and   row 142 would be '_','*Z*'

$\Delta\Delta\underline{B}\underline{V}$ is formed by first constructing the level 2 canonical representation
vector

        $CRV \leftarrow ('O' \Delta\Delta PREP 'WSIS'),TT\Delta\Delta PREP 'TRANSLATE'$

            where TT is the translation table array

$\Delta\Delta\underline{BV}$ is then defined as

$,O(8\rho 2)T\Delta\Delta ASCII\Delta APL\iota CRV$

where $\Delta\Delta ASCII\Delta APL$ is the *ASCII/APL*

transmission code vector

($\Delta\Delta ASCII\Delta APL$ is defined in the workspace *GENWS*)

In generating the bit stream, the incoming character stream to the function $\Delta\Delta OUTF$ is processed in blocks of 512 characters. The size of this block may need to be modified.

```
GENCRV                          ΔΔCCRV


     ∇  ΔΔCCRV;ΔΔX;ΔΔY
[1]   ⍝  ΔΔCCRV GENERATES A WORKSPACE CANONICAL REPRESENTATION VECTOR
[2]   ⍝  CONFORMING TO THE STAPL PROPOSED CONVENTION FOR LEVEL 2
[3]   □IO←ΔΔIO
[4]   ΔΔX←(ΔΔVS ΔΔPREP 'WSID'),(ΔΔVREP '□PP'),(ΔΔVREP '□PW'),ΔΔVREP '□IO'
[5]   □PP←18
[6]   □PW←ΔΔRL
[7]   □IO←0
[8]   ΔΔQUTE ΔΔX,(ΔΔVREP '□CT'),(ΔΔVREP '□LX'),ΔΔVREP '□RL'
[9]   ⍝  GENERATION OF FUNCTION CANONICAL REPRESENTATION VECTORS
[10]  ΔΔX←□NL 3
[11]  ΔΔY←'ΔΔINIT ΔΔQUTE ΔΔTRIM ΔΔSTEP ΔΔPREP ΔΔVREP ΔΔPREP ΔΔCCRV ΔΔQDCR '
[12]  ΔΔX←(∧/ΔΔXv.≠((1↓ρΔΔX),9)↑� 9 7 ρΔΔY)⌿ΔΔX
[13]  ΔΔCCRV1:→(0=ρ,ΔΔX)/ΔΔCCRV2
[14]  ΔΔQUTE ΔΔPREP,ΔΔTRIM ΔΔX[0;]
[15]  'FUNCTION CONVERTED:  ',ΔΔX[0;]
[16]  ΔΔY←□EX ΔΔX[0;]
[17]  ΔΔX← 1 0 ↓ΔΔX
[18]  →ΔΔCCRV1
[19]  ⍝  GENERATION OF VARIABLE CANONICAL REPRESENTATION VECTORS
[20]  ΔΔCCRV2:ΔΔX←□NL 2
[21]  ΔΔY←'ΔΔY  ΔΔX  ΔΔRL ΔΔCH ΔΔAV ΔΔBV ΔΔFS ΔΔWS ΔΔLC ΔΔIO '
[22]  ΔΔX←(∧/ΔΔXv.≠((1↓ρΔΔX),10)↑� 10 5 ρΔΔY)⌿ΔΔX
[23]  ΔΔCCRV3:→(0=ρ,ΔΔX)/ΔΔCCRV4
[24]  ΔΔQUTE ΔΔVREP,ΔΔTRIM ΔΔX[0;]
[25]  'VARIABLE CONVERTED:  ',ΔΔX[0;]
[26]  ΔΔY←□EX ΔΔX[0;]
[27]  ΔΔX← 1 0 ↓ΔΔX
[28]  →ΔΔCCRV3
[29]  ΔΔCCRV4:ΔΔQUTE '0' ΔΔPREP 'END'
     ∇
```

GENCR/   ∆∆QUTE

```
    ∇  ∆∆QUTE ∆∆V;∆∆P;∆∆R;∆∆S;∆∆T;∆∆B
[1]    ⍝  IF CONVERSION COMPLETED, FLAG TO OUTPUT REMAINDER OF STREAM
[2]    ∆∆S←∧/'8PEND'≠5↑∆∆V
[3]    ⍝  ADD CURRENT VECTOR TO ACCUMULATED STREAM BUFFER
[4]    ⍝  (IF BIT STREAM, ITERATIVELY ADD BLOCKS OF 512
[5]    ⍝  CHARACTERS CONVERTED TO BIT VECTORS)
[6]    ∆∆B←512
[7]    ∆∆QUTE1:∆∆BV←∆∆BV,⍕ 3 ¯31[∆∆LC]↑'∆∆V ,⍉(∆∆FS⍴2)⊤∆∆AV⍳(∆∆E⌊⍴∆∆V)↑∆∆V'
[8]    ∆∆V←⍕ 2 ¯8[∆∆LC]↑''''' ∆∆B↓∆∆V'
[9]    ⍝  OUTPUT FULL LINES (CHARACTER STREAM)
[10]   ⍝  OR FULL WORDS (BIT STREAM) OF ACCUMULATED STREAM
[11]   ∆∆R←⍕ 4 ¯2[∆∆LC]↑'∆∆RL 36'
[12]   ∆∆T←(-∆∆R|⍴∆∆BV)↑∆∆BV
[13]   →(∆∆R>⍴∆∆BV)/∆∆QUTE2
[14]   ∆∆BV←(-∆∆R|⍴∆∆BV)↓∆∆BV
[15]   ∆∆BV←⍕ 36 ¯13[∆∆LC]↑'((((⍴∆∆BV)÷∆∆RL),∆∆RL)⍴∆∆BV)⌷[5]∆∆CH ∆∆BV⌷∆∆CH,0,2'
[16]   ∆∆QUTE2:∆∆BV←∆∆T
[17]   →(0≠⍴,∆∆V)/∆∆QUTE1
[18]   →∆∆S/0
[19]   ⍝  IF CONVERSION COMPLETED, OUTPUT REMAINDER OF STREAM
[20]   ∆∆BV←⍕ 12 ¯13[∆∆LC]↑'∆∆BV⌷[5]∆∆CH ∆∆BV⌷∆∆CH,0,2'
    ∇
```

```
GENCRV                              ΔΔVREP
```

```
    ∇   ΔΔR←ΔΔVREP ΔΔA
[1]    ⍝   ΔΔA IS THE NAME OF THE VARIABLE
[2]    ⍝   GET VALUE
[3]     ΔΔR←⍎ΔΔA
[4]    ⍝   CATENATE NAME, RANK, SHAPE VECTOR, AND VALUE
[5]     ΔΔR←ΔΔA,' ',(⍕(⍴⍴ΔΔR),⍴ΔΔR),' ',⍕,ΔΔR
[6]    ⍝   APPEND DATA TYPE
[7]     ΔΔR←'NC'[(⍳1)+' '=1↑0⍴⍎ΔΔA],ΔΔR
[8]    ⍝   APPEND VECTOR LENGTH
[9]     ΔΔR←(⍕⍴ΔΔR),ΔΔR
    ∇
```

```
                              ΔΔFREP
```

```
    ∇   ΔΔR←ΔΔFREP ΔΔA
[1]    ⍝   ΔΔA IS THE NAME OF THE FUNCTION TO BE CONVERTED
[2]    ⍝   GET CANONICAL REPRESENTATION
[3]     ΔΔR←ΔΔ⎕CR ΔΔA
[4]    ⍝   APPEND DATA TYPE, RANK, AND SHAPE VECTOR
[5]     ΔΔR←'F',ΔΔA,' ',(⍕(⍴⍴ΔΔR),⍴ΔΔR),' ',,ΔΔR
[6]    ⍝   APPEND VECTOR LENGTH
[7]     ΔΔR←(⍕⍴ΔΔR),ΔΔR
    ∇
```

```
                              ΔΔPREP
```

```
    ∇   ΔΔR←ΔΔV ΔΔPREP ΔΔA
[1]    ⍝   GENERATE PSEUDO-VARIABLE
[2]    ⍝   ΔΔA IS THE NAME OF THE PSEUDO-VARIABLE AND
[3]    ⍝   ΔΔV IS THE VALUE
[4]    ⍝   CATENATE NAME, RANK, SHAPE VECTOR, AND VALUE
[5]     ΔΔR←'P',ΔΔA,' ',(⍕(⍴⍴ΔΔV),⍴ΔΔV),' ',⍕,ΔΔV
[6]    ⍝   APPEND VECTOR LENGTH
[7]     ΔΔR←(⍕⍴ΔΔR),ΔΔR
    ∇
```

APPENDIX C

WORKSPACE   GENWS

WORKSPACE RECONSTRUCTION FROM

CANONICAL REPRENSTATION

System Variables:

$\Box IO \leftrightarrow 0$

Variables:

$\Delta\Delta ASCII\Delta APL$ is a partial $ASCII/APL$ transmission code vector containing the printable $ASCII/APL$ characters and the $ASCII/APL$ character NUL. If the incoming translation table contains control characters, $\Delta\Delta ASCII\Delta APL$ should be extended, using the standard decimal $ASCII$ representation to determine the indices.

The function $\Delta\Delta CRCVT$ extracts individual canonical representation vectors from the partial character stream in core. If the stream is too short to extract the next vector, $\Delta\Delta CFIN$ reads blocks of 5 lines of the $ASCII$ file until the stream is sufficiently long. The number of lines read by $\Delta\Delta CFIN$ may need to be modified.

GENWS   ΔΔINIT

```
     ∇   ΔΔINIT;ΔΔFN;ΔΔRL;ΔΔCH;ΔΔWS;ΔΔIO
[1]      'FILE TO BE CONVERTED: ▲'
[2]      ΔΔFN←⎕
[3]      'TYPE OF FILE:'
[4]      '     ENTER 2(CHARACTER STREAM) OR 3(BIT STREAM): ▲'
[5]      →('3'=⎕)/ΔΔINIT3
[6]      'FILE LINE LENGTH: ▲'
[7]      ΔΔRL←⍎⎕
[8]      ΔΔCH←⎕ASS ΔΔFN,'/AS'
[9]      →ΔΔINIT1
[10]  ΔΔINIT3:ΔΔBFIN
[11]     →ΔΔTRANS/0
[12]     ΔΔCH←ΔΔBCVT
[13]  ΔΔINIT1:ΔΔCRCVT ''
[14]     ⎕IO←ΔΔIO
[15]     ' '
[16]     'THE ABOVE INDIVIDUALS HAVE BEEN RECREATED.'
[17]     'TO COMPLETE THE WORKSPACE RECONSTRUCTION,'
[18]     'ENTER THE FOLLOWING:'
[19]     '      )WSID ',ΔΔWS
[20]     '      )ERASE ΔΔGENWS'
[21]     '      )SAVE'
     ∇
```

GENNS   ΔΔTRANS

```
    ∇  ΔΔERR←ΔΔTRANS;ΔΔBS;ΔΔCV;ΔΔTL
[1]    A  ΔΔTRANS CHECKS THE STREAM IDENTIFIER
[2]    A  AND EXTRACTS THE TRANSLATON TABLE
[3]    ΔΔCV←ΔΔASCIIΔAPL[(8⍴2)⊤⍘ 15 8 ⍴120↑ΔΔBF]
[4]    →('9PWSIS 0 0'=10↑ΔΔCV)/ΔΔTRANS1
[5]    ' FILE DOES NOT CONFORM TO STAPL FORMAT CONVENTIONS '
[6]    'TO INSPECT THE FILE,'
[7]    ΔΔFN,' HAS BEEN ASSIGNED TO THE VARIABLE ΔΔBF'
[8]    'AND THE ATTEMPTED CONVERSION OF THE FIRST 15 CHARACTERS'
[9]    'HAS BEEN ASSIGNED TO ΔΔCV'
[10]   ΔΔERR←1
[11]   →0
[12]   A  CONVERT THE TRANSLATION TABLE FROM BINARY
[13]   A   TO A CHARACTER VECTOR
[14]   ΔΔTRANS1:ΔΔL←⍕ΔΔCV[9+⍳1+(¯5↑ΔΔCV)⍳'P']
[15]   ΔΔL←8×(10+ΔΔL+⍴⍕ΔΔL)
[16]   ΔΔCV←ΔΔASCIIΔAPL[(8⍴2)⊤⍘((¯10+ΔΔL÷8),8)⍴80↓ΔΔL↑ΔΔBF]
[17]   ΔΔBF←ΔΔL↓ΔΔBF
[18]   A  CONSTRUCT THE TRANSLATION TABLE ARRAY
[19]   ΔΔCRCVT ΔΔCV
[20]   A   FOR EACH ROW OF THE TRANSLATION TABLE,
[21]   A   INSERT BACKSPACE BETWEEN EACH CHARACTER
[22]   ΔΔTT←,ΔΔTRANSLATE[;ΔΔC←0]
[23]   ΔΔBS←(⍴ΔΔTRANSLATE)[0]⍴⎕AV[98]
[24]   ΔΔTRANS2:→((⍴ΔΔTRANSLATE)[1]=ΔΔC←ΔΔC+1)/ΔΔTRANS3
[25]   ΔΔTT←ΔΔTT,ΔΔBS,,ΔΔTRANSLATE[;ΔΔC]
[26]   →ΔΔTRANS2
[27]   ΔΔTRANS3:ΔΔTCVT⍘((¯1+2×(⍴ΔΔTRANSLATE)[1]),(⍴ΔΔTRANSLATE)[0])⍴ΔΔTT
[28]   ΔΔERR←0
    ∇
```

ΔΔTCVT

```
    ∇  ΔΔTCVT ΔΔA;ΔΔCH;ΔΔI
[1]    A  ΔΔTCVT CONVERTS THE TRANSLATION TABLE
[2]    A   TO THE APL CHARACTER VECTOR
[3]    ΔΔCH←⎕ASS((4⌊⍴ΔΔFN)↑ΔΔFN),(¯2↑⍕⎕UL),'.CRV/AS'
[4]    ΔΔA←ΔΔAB[5]ΔΔCH
[5]    ⎕CLS ΔΔCH
[6]    ΔΔI←0
[7]    ΔΔTRANSLATE←''
[8]    ΔΔTCVT1:ΔΔTRANSLATE←ΔΔTRANSLATE,1↑⍷[5]ΔΔCH
[9]    →((⍴ΔΔA)[0]>ΔΔI←ΔΔI+1)/ΔΔTCVT1
[10]   ⎕DAS ΔΔCH
    ∇
```

GENVS   ΔΔCRCVT

```
      ∇   ΔΔCRCVT ΔΔW;ΔΔP;ΔΔL;ΔΔ/;ΔΔT;ΔΔN;ΔΔR;ΔΔS
[1]    ⍝   ΔΔCRCVT CREATES A WORKSPACE FROM THE
[2]    ⍝   CHARACTER STREAM OF THE WORKSPACE
[3]    ΔΔCRCVT1:⍎(10>ρΔΔW)/'ΔΔW←ΔΔCFIN ΔΔW'
[4]    ⍝   DETERMINE LENGTH OF AN INDIVIDUAL
[5]    ⍝   CANONICAL REPRESENTATION VECTOR
[6]     ΔΔL←(⌊/(10↑ΔΔW)⍳'NCFP')↑ΔΔW
[7]    ΔΔCRCVT2:→((ρΔΔW)≥⍎ΔΔL)/ΔΔCRCVT3
[8]     ΔΔW←ΔΔCFIN ΔΔW
[9]     →ΔΔCRCVT2
[10]   ⍝   EXTRACT INDIVIDUAL CANONICAL REPRESENTATION VECTOR
[11]   ΔΔCRCVT3:ΔΔW←((ρΔΔV←ΔΔW[(ρΔΔL)+⍳⍎ΔΔL])+ρΔΔL)↓ΔΔW
[12]    ΔΔP←(ΔΔV=' ')/⍳ρΔΔV
[13]   ⍝   GET TYPE OF INDIVIDUAL
[14]    ΔΔT←1↑ΔΔV
[15]   ⍝   GET NAME
[16]    ΔΔN←1↓ΔΔP[0]↑ΔΔV
[17]   ⍝   GET RANK
[18]    ΔΔR←⍎(1+ρΔΔN)↓ΔΔP[1]↑ΔΔV
[19]   ⍝   GET SHAPE VECTOR
[20]    ⍎'ΔΔS←', 2 ‾22[ΔΔR≠0]↑'⍳0 ⍎ΔΔP[1]↓ΔΔP[1+ΔΔR]↑ΔΔV'
[21]    ΔΔV←(1+ΔΔP[1+ΔΔR])↓ΔΔV
[22]    →(ΔΔT='P')/ΔΔCRCVT5
[23]   ⍝   IF NUMERIC VARIABLE, CONVERT VALUE TO NUMBER
[24]    ⍎(ΔΔT='N')/'ΔΔV←⍎4 ‾1[0=ρΔΔV]↑''⍎ΔΔV 0'''
[25]   ⍝   SAVE ⎕IO OF WORKSPACE
[26]    →('⎕IO'≠3↑ΔΔN)/ΔΔCRCVT4
[27]    ΔΔIO←ΔΔV
[28]    ΔΔN
[29]    →ΔΔCRCVT1
[30]   ⍝   ASSIGN VALUE TO VARIABLE OR FIX FUNCTION DEFINITION
[31]   ΔΔCRCVT4:⍎ 15 ‾11[ΔΔT='F']↑'⍎ΔΔN,''←ΔΔSρΔΔV''⎕FX ΔΔSρΔΔV'
[32]    ⍎(ΔΔT≠'F')/'ΔΔN'
[33]    →ΔΔCRCVT1
[34]   ⍝   PROCESS PSEUDOVARIABLES.
[35]   ΔΔCRCVT5:⍎(∧/'WSID'=4↑ΔΔN)/'ΔΔWS←ΔΔV'
[36]   ⍝   IF BIT STREAM, CONSTRUCT TRANSLATE TABLE
[37]    →('TRANSLATE'≠9↑ΔΔN)/ΔΔCRCVT6
[38]    ΔΔTRANSLATE←(⍎ΔΔS)ρΔΔV
[39]    →0
[40]   ΔΔCRCVT6:→('END'≠3↑ΔΔN)/ΔΔCRCVT1
      ∇
```

```
     ∇  ΔΔBFIN;ΔΔCH
[1]    ℝ  ΔΔBFIN READS THE BIT STREAM REPRESENTATION
[2]    ℝ  OF THE WORKSPACE
[3]    ΔΔCH←⎕ASS ΔΔFN,'/BU'
[4]    ΔΔBF←⍂ΔΔCH,0,2,(⎕FLS ΔΔCH)[2]×36
[5]    ⎕DAS ΔΔCH
     ∇
```

ΔΔBCVT

```
     ∇  ΔΔCH←ΔΔBCVT;ΔΔBL;ΔΔT;⎕PV;ΔΔFS
[1]    ℝ  ΔΔBCVT CONVERTS THE BINARY STREAM TO A CHARACTER
[2]    ℝ  STREAM USING THE APL CHARACTER VECTOR
[3]    ΔΔCH←⎕ASS((4⌊ρΔΔFN)↑ΔΔFN),(⁻2↑⍕⎕UL),'.CRV/AS'
[4]    ΔΔBL←⎕PV←128
[5]    ΔΔFS←⌈2⊛ρΔΔTRANSLATE
[6]    ΔΔBL←ΔΔFS×512
[7]  ΔΔBCVT1:ΔΔT←ΔΔTRANSLATE[(ΔΔFSo2)⊥⍉(512,ΔΔFS)ρΔΔBL↑ΔΔBF]
[8]    ΔΔT←(4 128 oΔΔT)⍂[5]ΔΔCH
[9]    ΔΔBF←ΔΔBL↓ΔΔBF
[10]   →(0≠ρΔΔBF)/ΔΔBCVT1
[11]   ⎕CLS ΔΔCH
     ∇
```

ΔΔCFIN

```
     ∇  ΔΔR←ΔΔCFIN ΔΔC;ΔΔL
[1]    ℝ  ΔΔCFIN READS A BLOCK OF 5 FILE LINES OF
[2]    ℝ  CHARACTER STREAM REPRESENTATION OF THE WORKSPACE
[3]    ΔΔR←ΔΔC
[4]    ΔΔL←0
[5]  ΔΔCFIN1:ΔΔV←⍂[5]ΔΔCH
[6]    →(0 75 =o,ΔΔV)/0
[7]    ΔΔR←ΔΔR,ΔΔRL↑ΔΔV
[8]    →(5>ΔΔL←ΔΔL+1)/ΔΔCFIN1
     ∇
```

22.

UTILITY FUNCTIONS

The following functions are used in the workspaces. $\Delta\Delta QDCR$ is included because, in the version of *APLSF* on which these programs were run, the system function $\Box CR$ did not perform correctly.

The four utility functions assume $\Box IO$ to be 1 and minor modifications are required if $\Box IO$ is 0.

ΔΔQDCR

```
     ∇    ΔΔR←ΔΔQDCR ΔΔA;ΔΔD;ΔΔL;ΔΔI
[1]    ⍝    ΔΔQDCR IS EQUIVALENT TO
[2]    ⍝     THE SYSTEM FUNCTION ⎕CR
[3]      ΔΔD←1↓⍜ΔΔA
[4]      ΔΔI←ΔΔD=⎕AV[102]
[5]      ΔΔL←⍳0
[6]   ΔΔQDCR1:→ΔΔQDCR2×⍳3≥ρΔΔI
[7]      ΔΔL←ΔΔL,(ΔΔI⍳1)-1
[8]      ΔΔI←((¯1↑ΔΔL)+2)↓ΔΔI
[9]      →ΔΔQDCR1
[10]  ΔΔQDCR2:ΔΔR←((ρΔΔL),⌈/ΔΔL)ρ' '
[11]     ΔΔI←1
[12]  ΔΔQDCR3:→(ΔΔI>ρΔΔL)/0
[13]     ΔΔR[ΔΔI;⍳ΔΔL[ΔΔI]]←ΔΔL[ΔΔI]↑ΔΔSTRP ΔΔL[ΔΔI]↑ΔΔD
[14]     ΔΔD←(ΔΔL[ΔΔI]+2)↓ΔΔD
[15]     ΔΔI←ΔΔI+1
[16]     →ΔΔQDCR3
     ∇
```


ΔΔSTRP

```
     ∇   ΔΔR←ΔΔSTRP ΔΔA
[1]    ⍝   ΔΔSTRP REMOVES LEADING BLANKS
[2]    ⍝    FROM THE CHACTER VECTOR ΔΔA
[3]     ΔΔR←,ΔΔA
[4]     ΔΔR←(((ΔΔR∈'      ')⍳0)-1)↓ΔΔR
     ∇
```


ΔΔTRIM

```
     ∇   ΔΔR←ΔΔTRIM ΔΔA
[1]    ⍝   ΔΔTRIM REMOVES TRAILING BLANKS
[2]    ⍝    FROM THE CHARACTER VECTOR ΔΔA
[3]     ΔΔR←⌽ΔΔSTRP⌽ΔΔA
     ∇
```


ΔΔDPLQ

```
     ∇   ΔΔR←ΔΔDPLQ ΔΔA;ΔΔI
[1]    ⍝   ΔΔDPLQ INSERTS A SECOND QUOTE AFTER EACH
[2]    ⍝    OCCURRENCE OF A QUOTE IN STRING ΔΔA
[3]     ΔΔR←ΔΔA
[4]     ΔΔI←¯1
[5]   ΔΔDPLQ1:→((ρΔΔR)<ΔΔI←ΔΔI+1+((ΔΔI+1)↓ΔΔR)⍳'''')/0
[6]     ΔΔR←(ΔΔI↑ΔΔR),'''',ΔΔI↓ΔΔR
[7]     →ΔΔDPLQ1
     ∇
```

24.

REFERENCES

[1]   R. J. Orgass, Transferring Files to an IBM Machine, Technical Memo-
       randum No. APLAD14, July 28, 1978, University of Arizona.

[2]   D. Cartwright, Proposed Standard for the Interchange of APL Work-
       spaces, Dec 1977, *APL* Quote Quad.

[3]   R. J. Orgass and G. M. Uhler, Using *SOS* in *APLSF*, Technical Memo-
       randum, September 14, 1977, University of Arizona.

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR. 79-0238	**2. GOVT ACCESSION NO.**	**3. RECIPIENT'S CATALOG NUMBER**
4. TITLE (and Subtitle) TRANSFER OF APL WORKSPACES: A USEFUL SOLUTION		**5. TYPE OF REPORT & PERIOD COVERED** Interim
		6. PERFORMING ORG. REPORT NUMBER CS78006-T
7. AUTHOR(s) Karl Rautenkranz and Richard J. Orgass		**8. CONTRACT OR GRANT NUMBER(s)** AFOSR 79-0021
9. PERFORMING ORGANIZATION NAME AND ADDRESS Virginia Polytechnic Inst. & St. Univ. Computer Science Department Blacksburg, Virginia 24061		**10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS** 61102F 2304/A2
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Office of Scientific Research/NM Bolling AFB, Washington, DC 20332		**12. REPORT DATE** September 1978
		13. NUMBER OF PAGES 27
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		**15. SECURITY CLASS. (of this report)** UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

APL Workspace Interchange, APL Workspace Transfer, Software Exchange

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Most suppliers of APL have not yet implemented the STAPL convention for transmitting workspaces from one installation to another. This report describes three workspace representations which may be used on a DECsystem-10 for this purpose. Two are partial implementations of the STAPL convention: one for level 2 of the convention, the other for level 3. The third representation is a terminal transcript file which is used as an input file. In addition, these representations may be used to reduce the disk storage required for APL workspaces on the DECsystem-10.

DD FORM 1 JAN 73 1473