Technical Report CS78006-T

TRANSFER OF APL WORKSPACES: A USEFUL SOLUTION*

by

Karl Rautenkranz†

and

Richard J. Orgass

Department of Computer Science

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

September 1978

* Research Sponsored by the Air Force Office of Scientific Research, Air Force Systems Command, Under Grant No. AFOSR-78-3499 to the University of Arizona. The United States Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation hereon.

† Current address: Trent University, Peterboro, Ontario K9J 7B8, Canada.

ABSTRACT

ABSTRACT. 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 *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.

Key Words and Key Phrases: *APL* Workspace Interchange
APL Workspace Transfer,
Software Exchange

CR Category: 4.49

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 (ϵ 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 ~~i~~s 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 5⍴⍳10
      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←1.136864040747036E¯13
⎕IO←1
⎕LX←''
⎕RL←0
ΔΔRL←''
ΔΔRL←ΔΔRL,'30'
ΔΔRL←⍎ΔΔRL
'ΔΔRL'
XX←''
XX←XX,'1 2 3 4 5 6 7 8 9 10'
XX←⍎XX
XX←2 5⍴XX
'XX'
ΔΔF̲T̲←''
ΔΔF̲T̲←ΔΔF̲T̲,'Y         ''MANY'''
ΔΔF̲T̲←ΔΔF̲T̲,'    ''SEVERAL'' '
ΔΔF̲T̲←3 10⍴ΔΔF̲T̲
ΔΔF̲T̲←⎕FX ΔΔF̲T̲
)ERASE ΔΔF̲T̲
)SAVE
```

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 workspace

<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
686404040747036E¯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
     ΔΔINIT
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 (⎕*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
     ΔΔINIT
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 *MUMO*nn.*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 *∆∆TCVT*[3].

Following this phase of the bit stream reconstruction, the workspace is converted to a character stream and written on the file *MUMO*nn.*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 ∆∆GENWS
      )SAVE
```

where *∆∆GENWS* is the group of global variables and functions of GENWS.

APPENDIX A

WORKSPACE GENNT

TERMINAL TRANSCRIPT GENERATOR

System Variables:

⎕*IO* ↔ 1

Variables:

∆∆O̲S̲ is used in determining the number of *APL* characters to be written on an output line.

```
      ⍴∆∆O̲S̲
47
      ∆∆O̲S̲
⍝!$⍒⍋⌶⍟⍱⍲⍀⌿⊖⌽⍉⍎⍕⌹⍞⍈⍇⍙A̲B̲C̲D̲E̲F̲G̲H̲I̲J̲K̲L̲M̲N̲O̲P̲Q̲R̲S̲T̲U̲V̲W̲X̲Y̲Z̲
```

ΔΔINIT

```
∇  ΔΔINIT;ΔΔW̲S̲;ΔΔRL;⎕IO
    ΔΔI̲O̲←⎕IO
    ⎕IO←1
    'DESTINATION WORKSPACE NAME: ⍝'
    ΔΔW̲S̲←⍞
    ΔΔW̲S̲←⍎ 5 ¯4[1+1≠ρΔΔW̲S̲,' ']↑')WSID ΔΔW̲S̲'
    'TERMINAL TRANSCRIPT LINE LENGTH: ⍝'
    ΔΔRL←⍞
    ΔΔRL←⍎ 3 ¯5[1+1≠ρΔΔRL,' ']↑'⎕PW  ⍎ΔΔRL'
    ΔΔGENW
∇
```

ΔΔGENW

```
∇  ΔΔGENW;ΔΔX̲;ΔΔF̲
   ⍝  ΔΔGENW CONVERTS THE WORKSPACE ΔΔW̲S̲ TO A TERMINAL
   ⍝  TRANSCRIPT ON THE KEY PAIRED ACSII FILE ΔΔW̲S̲.TSP
    ⍎')OUTPUT ',ΔΔW̲S̲,'.TSP'
    ')CLEAR'
    ')WSID ',ΔΔW̲S̲
    '⎕PP←',⍕⎕PP
    '⎕PW←',⍕⎕PW
    ⎕PP←18
    ⎕PW←ΔΔRL
    '⎕CT←',⍕⎕CT
    '⎕IO←',⍕ΔΔI̲O̲
    '⎕LX←''',⎕LX,''''
    '⎕RL←',⍕⎕RL
   ⍝  CONVERT VARIABLES IN WORKSPACE
    ΔΔX̲←⎕NL 2
    ΔΔX̲←(∧/ΔΔX̲∨.≠((1↓ρΔΔX̲),5)↑⍉ 5 5 ρ'ΔΔW̲S̲ ΔΔX̲  ΔΔC̲R̲ ΔΔO̲S̲ ΔΔI̲O̲ ')⌿ΔΔX̲
   ΔΔGENW1:→(0=ρ,ΔΔX̲)/ΔΔGENW2
    ΔΔTTRV ΔΔTRIM ΔΔX̲[1;]
    ΔΔX̲← 1 0 ↓ΔΔX̲
    →ΔΔGENW1
   ⍝  CONVERT FUNCTIONS IN WORKSPACE
   ΔΔGENW2:ΔΔX̲←⎕NL 3
    ΔΔF̲←'ΔΔGENW ΔΔTRIM ΔΔSTRP ΔΔTTRV ΔΔTTRF ΔΔDPLQ ΔΔQDCR ΔΔINIT'
    ΔΔX̲←(∧/ΔΔX̲∨.≠((1↓ρΔΔX̲),8)↑⍉ 8 7 ρΔΔF̲)⌿ΔΔX̲
   ΔΔGENW3:→(0=ρ,ΔΔX̲)/ΔΔGENW4
    ΔΔTTRF ΔΔTRIM ΔΔX̲[1;]
    ΔΔX̲← 1 0 ↓ΔΔX̲
    →ΔΔGENW3
   ΔΔGENW4:')ERASE ΔΔF̲T̲'
    ')SAVE'
    ⍎')OUTPUT'
    'FILE CREATED: ',ΔΔW̲S̲,'.TSP'
∇
```

ΔΔTTRV

```
∇  ΔΔTTRV ΔΔN̲;ΔΔC̲;ΔΔS̲;ΔΔL̲
     ⍝  ΔΔTTRV PRODUCES A TERMINAL TRANSCRIPT OF A
     ⍝  VARIABLE ASSIGNMENT
      ΔΔC̲←⍎ΔΔN̲
      ΔΔS̲←⍕ρΔΔC̲
      ΔΔC̲←ΔΔDPLQ⍕,ΔΔC̲
     ⍝  ASSIGN TO VARIABLE NAME ITS VALUE REPRESENTED
     ⍝  AS A CHARACTER VECTOR
      ΔΔN̲,'←'''''
    ΔΔTTRV1:→(0=ρΔΔC̲)/ΔΔTTRV2
    ⍝  DETERMINE LENGTH OF CHARACTER STRING TO FILL FILE LINE
     ΔΔW̲←¯1+((+\1+(ΔΔC̲∊ΔΔQ̲S̲)×2)>(ρΔΔC̲)⌊⎕PW-4+2×ρ,ΔΔN̲)⍳1
    ⍝  CHECK CHARACTER STRING DOES NOT
    ⍝  CONTAIN ODD NUMBER OF QUOTES
     ΔΔW̲←ΔΔW̲-0≠2|+/(ΔΔW̲↑ΔΔC̲)=''''
     ΔΔN̲,'←',ΔΔN̲,',''',(ΔΔW̲↑ΔΔC̲),''''
     ΔΔC̲←ΔΔW̲↓ΔΔC̲
     →ΔΔTTRV1
    ⍝  IF NUMERIC, CONVERT VALUE
    ΔΔTTRV2:→(' '=1↑0ρ,⍎ΔΔN̲)/ΔΔTTRV3
     ΔΔN̲,'←⍎',⍕⍎ 4 ¯1[1+0=ρ,⍎ΔΔN̲]↑'ΔΔN̲ 0'
    ⍝  RESHAPE VARIABLE
    ΔΔTTRV3:⍎(0≠ρΔΔS̲)/'ΔΔN̲,''←'',ΔΔS̲,''ρ'',ΔΔN̲'
     '''',ΔΔN̲,''''
  ∇
```

ΔΔTTRF

```
∇  ΔΔTTRF ΔΔA̲;ΔΔW̲;ΔΔS̲
     ⍝  ΔΔTTRF PRODUCES A TERMINAL TRANSCRIPT OF THE FUNCTION ΔΔA̲
      ΔΔA̲←ΔΔQDCR ΔΔA̲
      ΔΔS̲←ρΔΔA̲
      ΔΔA̲←ΔΔDPLQ,ΔΔA̲
      'ΔΔF̲T̲←'''''
    ΔΔTTRF1:→(0=ρΔΔA̲)/ΔΔTTRF2
    ⍝  DETERMINE LENGTH OF CHARACTER STRING TO FILL FILE LINE
     ΔΔW̲←¯1+((+\1+(ΔΔA̲∊ΔΔQ̲S̲)×2)>(ρΔΔA̲)⌊⎕PW-12)⍳1
    ⍝  CHECK THAT CHARACTER STRING DOES NOT
    ⍝  CONTAIN ODD NUMBER OF QUOTES
     ΔΔW̲←ΔΔW̲-0≠2|+/(ΔΔW̲↑ΔΔA̲)=''''
     'ΔΔF̲T̲←ΔΔF̲T̲,''',(ΔΔW̲↑ΔΔA̲),''''
     ΔΔA̲←ΔΔW̲↓ΔΔA̲
     →ΔΔTTRF1
    ⍝  RESHAPE FUNCTION DEFINITION AS A CANONICAL REPRESENTATION
    ΔΔTTRF2:'ΔΔF̲T̲←',(⍕ΔΔS̲),'ρΔΔF̲T̲'
    ⍝  FIX THE FUNCTION
     'ΔΔF̲T̲←⎕FX ΔΔF̲T̲'
  ∇
```

ΔΔINIT

```
     ∇  ΔΔINIT;ΔΔW̲S̲;ΔΔR̲L̲;ΔΔC̲H̲;ΔΔL̲C̲;ΔΔF̲S̲;ΔΔI̲O̲
[1]     ΔΔI̲O̲←⎕IO
[2]     ⎕IO←0
[3]     'DESTINATION WORKSPACE NAME: ⍝'
[4]     ΔΔW̲S̲←,⍞
[5]     ⍎(1=ρΔΔW̲S̲,' ')/'ΔΔW̲S̲←⍎'')WSID'''
[6]     'LEVEL OF CONVERSION:'
[7]     '              ENTER 2(CHARACTER STREAM) OR 3(BIT STREAM): ⍝'
[8]     ΔΔL̲C̲←¯2+⍎⍞
[9]     →ΔΔL̲C̲/ΔΔINIT3
[10]   ⍝  INITIALIZATION FOR CHARACTER STREAM FILE
[11]    'FILE LINE LENGTH: ⍝'
[12]    ΔΔR̲L̲←⍞
[13]    ΔΔR̲L̲←⍎ 3 ¯5[1≠ρΔΔR̲L̲,' ']↑'⎕PW  ⍎ΔΔR̲L̲'
[14]    ΔΔC̲H̲←⎕ASS ΔΔW̲S̲,'.CRV/AS'
[15]    ΔΔB̲V̲←''
[16]    →ΔΔINIT1
[17]   ⍝  INITIALIZATION FOR BIT STREAM FILE
[18]   ΔΔINIT3:ΔΔC̲H̲←⎕ASS ΔΔW̲S̲,'.BRV/BU'
[19]   ⍝  DETERMINE FRAMESIZE
[20]    ΔΔF̲S̲←⌈2⍟ρΔΔA̲V̲
[21]    ΔΔR̲L̲←⎕PW
[22]   ΔΔINIT1:ΔΔC̲C̲R̲V̲
[23]    ⎕DAS ΔΔC̲H̲
[24]    'CONVERSION COMPLETED:'
[25]    '    FILE CREATED: ',ΔΔW̲S̲, 7 ¯7[ΔΔL̲C̲]↑'.CRV/AS  .BRV/BU'
     ∇
```

APPENDIX B

WORKSPACE GENCRV

CANONICAL REPRESENTATION GENERATOR

System Variables:

```
            ⎕IO ↔ 0
```

Variables:

ΔΔA̲V̲ is the *APL* character vector

```
      ρΔΔA̲V̲
142
      ΔΔA̲V̲
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789,./[]{⍞⊢↔⋄×¨¯<≤=≥>≠∨∧-÷
      $⊣→⋆○ι↓↑~ρ∈ω?α⌈⌊_∇Δ∘'⎕()}\:;|⊤⊥∪∩⊃⊂⍝!$⍒⍋⌶⍟⍱⍲⍀⌿⍎⍕⌹⍞⍈⍇
      Δ̲A̲B̲C̲D̲E̲F̲G̲H̲I̲J̲K̲L̲M̲N̲O̲P̲Q̲R̲S̲T̲U̲V̲W̲X̲Y̲Z̲
```

ΔΔB̲V̲ is the level 3 representation of the stream identifier and the translation table. As described in the text, the translation table is an array, the rows corresponding to the *APL* characters in ΔΔA̲V̲. The number of columns is the maximum number of overstrikes required to print the *APL* characters. Thus

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

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

ΔΔB̲V̲ is formed by first constructing the level 2 canonical representation vector

```
      CRV ← ('O' ΔΔPREP 'WSIS'),TTΔΔPREP 'TRANSLATE'
```

where *TT* is the translation table array

∆∆*BV* is then defined as

```
,O(8ρ2)T∆∆ASCII∆APLιCRV
```

where ∆∆*ASCII*∆*APL* is the *ASCII/APL*

transmission code vector

(∆∆*ASCII*∆*APL* is defined in the workspace *GENWS*)

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

GENCRV   ∆∆CCRV

```
    ∇  ∆∆CCRV;∆∆X;∆∆Y
[1]    ⍝  ∆∆CCRV GENERATES A WORKSPACE CANONICAL REPRESENTATION VECTOR
[2]    ⍝  CONFORMING TO THE STAPL PROPOSED CONVENTION FOR LEVEL 2
[3]     ⎕IO←∆∆IO
[4]     ∆∆X←(∆∆WS ∆∆PREP 'WSID'),(∆∆VREP '⎕PP'),(∆∆VREP '⎕PW'),∆∆VREP '⎕IO'
[5]     ⎕PP←18
[6]     ⎕PW←∆∆RL
[7]     ⎕IO←0
[8]     ∆∆OUTE ∆∆X,(∆∆VREP '⎕CT'),(∆∆VREP '⎕LX'),∆∆VREP '⎕RL'
[9]    ⍝  GENERATION OF FUNCTION CANONICAL REPRESENTATION VECTORS
[10]    ∆∆X←⎕NL 3
[11]    ∆∆Y←'∆∆INIT ∆∆OUTE ∆∆TRIM ∆∆STRP ∆∆FREP ∆∆VREP ∆∆PREP ∆∆CCRV ∆∆QDCR '
[12]    ∆∆X←(∧/∆∆X∨.≠((1↓⍴∆∆X),9)↑⍉ 9 7 ⍴∆∆Y)⌿∆∆X
[13]   ∆∆CCRV1:→(0=⍴,∆∆X)/∆∆CCRV2
[14]    ∆∆OUTE ∆∆FREP,∆∆TRIM ∆∆X[0;]
[15]    'FUNCTION CONVERTED:  ',∆∆X[0;]
[16]    ∆∆Y←⎕EX ∆∆X[0;]
[17]    ∆∆X← 1 0 ↓∆∆X
[18]    →∆∆CCRV1
[19]   ⍝  GENERATION OF VARIABLE CANONICAL REPRESENTATION VECTORS
[20]   ∆∆CCRV2:∆∆X←⎕NL 2
[21]    ∆∆Y←'∆∆Y  ∆∆X  ∆∆RL ∆∆CH ∆∆AV ∆∆BV ∆∆FS ∆∆WS ∆∆LC ∆∆IO '
[22]    ∆∆X←(∧/∆∆X∨.≠((1↓⍴∆∆X),10)↑⍉ 10 5 ⍴∆∆Y)⌿∆∆X
[23]   ∆∆CCRV3:→(0=⍴,∆∆X)/∆∆CCRV4
[24]    ∆∆OUTE ∆∆VREP,∆∆TRIM ∆∆X[0;]
[25]    'VARIABLE CONVERTED:  ',∆∆X[0;]
[26]    ∆∆Y←⎕EX ∆∆X[0;]
[27]    ∆∆X← 1 0 ↓∆∆X
[28]    →∆∆CCRV3
[29]   ∆∆CCRV4:∆∆OUTE '0' ∆∆PREP 'END'
    ∇
```

ΔΔOUTF

```
     ∇  ΔΔOUTF ΔΔV;ΔΔP;ΔΔR;ΔΔS;ΔΔT;ΔΔB
[1]     ⍝  IF CONVERSION COMPLETED, FLAG TO OUTPUT REMAINDER OF STREAM
[2]      ΔΔS←∧/'8PEND'≠5↑ΔΔV
[3]     ⍝  ADD CURRENT VECTOR TO ACCUMULATED STREAM BUFFER
[4]     ⍝  (IF BIT STREAM, ITERATIVELY ADD BLOCKS OF 512
[5]     ⍝  CHARACTERS CONVERTED TO BIT VECTORS)
[6]      ΔΔB←512
[7]     ΔΔOUTF1:ΔΔBV←ΔΔBV,⍎ 3 ¯31[ΔΔLC]↑'ΔΔV ,⌽(ΔΔFSρ2)⊤ΔΔAV⍳(ΔΔB⌊ρΔΔV)↑ΔΔV'
[8]      ΔΔV←⍎ 2 ¯8[ΔΔLC]↑''''' ΔΔB↓ΔΔV'
[9]     ⍝  OUTPUT FULL LINES (CHARACTER STREAM)
[10]    ⍝  OR FULL WORDS (BIT STREAM) OF ACCUMULATED STREAM
[11]     ΔΔR←⍎ 4 ¯2[ΔΔLC]↑'ΔΔRL 36'
[12]     ΔΔT←(-ΔΔR|ρΔΔBV)↑ΔΔBV
[13]     →(ΔΔR>ρΔΔBV)/ΔΔOUTF2
[14]     ΔΔBV←(-ΔΔR|ρΔΔBV)↓ΔΔBV
[15]     ΔΔBV←⍎ 36 ¯13[ΔΔLC]↑'((((ρΔΔBV)÷ΔΔRL),ΔΔRL)ρΔΔBV)⍈[5]ΔΔCH ΔΔBV⍈ΔΔCH,0,2'
[16]    ΔΔOUTF2:ΔΔBV←ΔΔT
[17]     →(0≠ρ,ΔΔV)/ΔΔOUTF1
[18]     →ΔΔS/0
[19]    ⍝  IF CONVERSION COMPLETED, OUTPUT REMAINDER OF STREAM
[20]     ΔΔBV←⍎ 12 ¯13[ΔΔLC]↑'ΔΔBV⍈[5]ΔΔCH ΔΔBV⍈ΔΔCH,0,2'
    ∇
```

ΔΔV̲R̲E̲P̲

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

ΔΔF̲R̲E̲P̲

```
     ∇  ΔΔR̲←ΔΔF̲R̲E̲P̲ ΔΔA̲
[1]     ⍝  ΔΔA̲ IS THE NAME OF THE FUNCTION TO BE CONVERTED
[2]     ⍝  GET CANONICAL REPRESENTATION
[3]      ΔΔR̲←ΔΔQ̲D̲C̲R̲ ΔΔA̲
[4]     ⍝  APPEND DATA TYPE, RANK, AND SHAPE VECTOR
[5]      ΔΔR̲←'F',ΔΔA̲,' ',(⍕(⍴⍴ΔΔR̲),⍴ΔΔR̲),' ',,ΔΔR̲
[6]     ⍝  APPEND VECTOR LENGTH
[7]      ΔΔR̲←(⍕⍴ΔΔR̲),ΔΔR̲
     ∇
```

ΔΔP̲R̲E̲P̲

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

APPENDIX C

WORKSPACE GENWS

WORKSPACE RECONSTRUCTION FROM
CANONICAL REPRENSTATION

System Variables:

⎕*IO* ↔ 0

Variables:

∆∆*ASCII*∆*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, ∆∆*ASCII*∆*APL* should be extended, using the standard decimal *ASCII* representation to determine the indices.

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

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'
     ∇
```

ΔΔTRANS

```
    ∇  ΔΔERR←ΔΔTRANS;ΔΔBS;ΔΔCV;ΔΔTL
         ⍝  ΔΔTRANS CHECKS THE STREAM IDENTIFIER
         ⍝  AND EXTRACTS THE TRANSLATON TABLE
         ΔΔCV←ΔΔASCIIΔAPL[(8⍴2)⊥⍉ 15 8 ⍴120↑ΔΔBF]
         →('9PWSIS 0 0'=10↑ΔΔCV)/ΔΔTRANS1
         ' FILE DOES NOT CONFORM TO STAPL FORMAT CONVENTIONS '
         'TO INSPECT THE FILE,'
         ΔΔFN,' HAS BEEN ASSIGNED TO THE VARIABLE ΔΔBF'
         'AND THE ATTEMPTED CONVERSION OF THE FIRST 15 CHARACTERS'
         'HAS BEEN ASSIGNED TO ΔΔCV'
         ΔΔERR←1
         →0
        ⍝  CONVERT THE TRANSLATION TABLE FROM BINARY
        ⍝  TO A CHARACTER VECTOR
        ΔΔTRANS1:ΔΔL←⍎ΔΔCV[9+⍳1+(¯5↑ΔΔCV)⍳'P']
         ΔΔL←8×(10+ΔΔL+⍴⍕ΔΔL)
         ΔΔCV←ΔΔASCIIΔAPL[(8⍴2)⊥⍉((¯10+ΔΔL÷8),8)⍴80↓ΔΔL↑ΔΔBF]
         ΔΔBF←ΔΔL↓ΔΔBF
        ⍝  CONSTRUCT THE TRANSLATION TABLE ARRAY
         ΔΔCRCVT ΔΔCV
        ⍝  FOR EACH ROW OF THE TRANSLATION TABLE,
        ⍝  INSERT BACKSPACE BETWEEN EACH CHARACTER
         ΔΔTT←,ΔΔTRANSLATE[;ΔΔC←0]
         ΔΔBS←(⍴ΔΔTRANSLATE)[0]⍴⎕AV[98]
        ΔΔTRANS2:→((⍴ΔΔTRANSLATE)[1]=ΔΔC←ΔΔC+1)/ΔΔTRANS3
         ΔΔTT←ΔΔTT,ΔΔBS,,ΔΔTRANSLATE[;ΔΔC]
         →ΔΔTRANS2
        ΔΔTRANS3:ΔΔTCVT⍉((¯1+2×(⍴ΔΔTRANSLATE)[1]),(⍴ΔΔTRANSLATE)[0])⍴ΔΔTT
         ΔΔERR←0
    ∇
```

ΔΔTCVT

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

GENWS

ΔΔCRCVT

```
     ∇  ΔΔCRCVT ΔΔW;ΔΔP;ΔΔL;ΔΔV;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

```
    ∇  ΔΔBFIN;ΔΔCH
        ⍝  ΔΔBFIN READS THE BIT STREAM REPRESENTATION
        ⍝  OF THE WORKSPACE
         ΔΔCH←⎕ASS ΔΔFN,'/BU'
         ΔΔBF←⍈ΔΔCH,0,2,(⎕FLS ΔΔCH)[2]×36
         ⎕DAS ΔΔCH
    ∇
```

ΔΔBCVT

```
    ∇  ΔΔCH←ΔΔBCVT;ΔΔBL;ΔΔT;⎕PW;ΔΔFS
        ⍝  ΔΔBCVT CONVERTS THE BINARY STREAM TO A CHARACTER
        ⍝  STREAM USING THE APL CHARACTER VECTOR
         ΔΔCH←⎕ASS((4⌊ρΔΔFN)↑ΔΔFN),(¯2↑⍕⎕UL),'.CRV/AS'
         ΔΔRL←⎕PW←128
         ΔΔFS←⌈2⍟ρΔΔTRANSLATE
         ΔΔBL←ΔΔFS×512
       ΔΔBCVT1:ΔΔT←ΔΔTRANSLATE[(ΔΔFSρ2)⊥⍉(512,ΔΔFS)ρΔΔBL↑ΔΔBF]
         ΔΔT←(4 128 ρΔΔT)⍈[5]ΔΔCH
         ΔΔBF←ΔΔBL↓ΔΔBF
        →(0≠ρΔΔBF)/ΔΔBCVT1
        ⎕CLS ΔΔCH
    ∇
```

ΔΔCFIN

```
    ∇  ΔΔR←ΔΔCFIN ΔΔC;ΔΔL
        ⍝  ΔΔCFIN READS A BLOCK OF 5 FILE LINES OF
        ⍝  CHARACTER STREAM REPRESENTATION OF THE WORKSPACE
         ΔΔR←ΔΔC
         ΔΔL←0
       ΔΔCFIN1:ΔΔV←⍈[5]ΔΔCH
        →(0 75 =ρ,ΔΔV)/0
        ΔΔR←ΔΔR,ΔΔRL↑ΔΔV
        →(5>ΔΔL←ΔΔL+1)/ΔΔCFIN1
    ∇
```

APPENDIX D

UTILITY FUNCTIONS

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

The four utility functions assume ⎕*IO* to be 1 and minor modifications are required if ⎕*IO* is 0.

ΔΔQDCR

```
    ∇  ΔΔR̲←ΔΔQDCR ΔΔA̲;ΔΔD̲;ΔΔL̲;ΔΔI̲
[1]     ⍝  ΔΔQDCR IS EQUIVALENT TO
[2]     ⍝  THE SYSTEM FUNCTION ⎕CR
[3]      ΔΔD̲←1↓⊤ΔΔA̲
[4]      ΔΔI̲←ΔΔD̲=⎕AV[102]
[5]      ΔΔL̲←⍳0
[6]     ΔΔQDCR1:→ΔΔQDCR2×⍳3≥ρΔΔI̲
[7]      ΔΔL̲←ΔΔL̲,(ΔΔI̲⍳1)-1
[8]      ΔΔI̲←((¯1↑ΔΔL̲)+2)↓ΔΔI̲
[9]      →ΔΔQDCR1
[10]    ΔΔQDCR2:ΔΔR̲←((ρΔΔL̲),⌈/ΔΔL̲)ρ' '
[11]     ΔΔI̲←1
[12]    ΔΔQDCR3:→(ΔΔI̲>ρΔΔL̲)/0
[13]     ΔΔR̲[ΔΔI̲;⍳ΔΔL̲[ΔΔI̲]]←ΔΔL̲[ΔΔI̲]↑ΔΔSTRP ΔΔL̲[ΔΔI̲]↑ΔΔD̲
[14]     ΔΔD̲←(ΔΔL̲[ΔΔI̲]+2)↓ΔΔD̲
[15]     ΔΔI̲←ΔΔI̲+1
[16]     →ΔΔQDCR3
    ∇
```

ΔΔSTRP

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

ΔΔTRIM

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

ΔΔDPLQ

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

REFERENCES

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

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

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