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.
Copyright, 1978

by

Karl Rautenkranz

and

Richard J. Orgass

General permission to republish, but not for profit, all or part of this report is granted, provided that the copyright notice is given and that reference is made to the publication (Technical Report No. CS78006-T, Department of Computer Science, Virginia Polytechnic Institute and State University, to its date of issue (September, 1978) and to the fact that reprinting privileges were granted by the author.
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 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 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 DECsystem-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 DECsystem-10. The terminal transcript representation for moving from a DECsystem-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 DECsystem-10 and a machine which does not support ASCII files.

Programs have been written for the DECsystem-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 DECsystem-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.

APLSP, as implemented on the DECsystem-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 GENCRT 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 DECsystem-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.
2. **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+1
2 ρ,C
```

while in **APL/SV**:

```
C+1
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 (**G**,  **S**,  **W** ) 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 (**e** or **I**) 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 '5p'10
XX
1 2 3 4 5
6 7 8 9 10
vy
[1] 'MANY'
[2] 'SEVERAL'
[3] v
```

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

```
)CLEAR
)WSID MUMBLE
)PP+10
)PW+80
)CT+1.13686404040747036E-13
)IO+1
)IX+''
)KRL+0
)ARL+'''
)ARL+ARL, '30'
)ARL+ARL
)'ARL'
)XX+''
)XX+XX, '1 2 3 4 5 6 7 8 9 10'
)XX+2XX
)XX+2 '5pXX'
)'XX'
)Def+''
)Def+'''DEF''', 'Y''''''''MANY''''''
)Def+'''DEF''', 'Y''''''''SEVERAL''''''''
)Def+3 '10pDEF'
)Def+IPK 'DEF'
)ERASE 'DEF'
)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:

```plaintext
<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:

```plaintext
<wsid><crv1>...<crv><end>
```

where

- `<wsid>` is the canonical representation vector naming the workspace
- `<crv1>` 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:

```plaintext
16FW$WID 1 6 NUMBE99999 0109N
UPW 0 608MIO 0 130NCT 0 1 13
6854040407470365139CLX 1 0 8
WAL 0 040FY 2 3 10 Y
'MANY' 'SEVERAL' 30Name 2 2
5 1 2 3 4 5 6 7 8 9 108PEND 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 machine. These indices are then expressed as binary numbers. For example, suppose the execute character (X) 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.
5.

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:

```apl
)LOAD MUMBLE
```

Next copy the terminal transcript generator and begin execution as in the following transcript:

```apl
)COPY APL:GENT

DEINIT

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 (UPW) 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.TSC
```

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 \texttt{MUMBLE} which should be replaced by the workspace name to be transmitted.

After the workspace has been prepared, load with the command:

\texttt{)LOAD MUMBLE}

Next copy the canonical representation generator and begin execution:

\texttt{)COPY APL:GENVRV}

\texttt{)INIT}

\texttt{DESTINATION WORKSPACE NAME: MUMBLE}

\texttt{LEVEL OF CONVERSION:}

\texttt{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 (\texttt{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

\texttt{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 (\texttt{DIPW}) 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:

\texttt{)LOAD GENES}

\texttt{)INIT}

\texttt{FILE TO BE CONVERTED: MUMBLE}

\texttt{TYPE OF FILE:}

\texttt{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 \texttt{ASCII/APL} character stream or a bit stream. If the number 2 is entered, a third request is displayed:

\texttt{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 \texttt{STAPL} format conventions and the conversion is terminated. If it is correct, the translation table is reconstructed using the \texttt{ASCII/APL}
transmission vector and then written on a temporary file \texttt{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 APLSR, a character error will occur at this point. Should this occur, the temporary file can be modified and execution resumed at \texttt{AACTY}[3].

Following this phase of the bit stream reconstruction, the workspace is converted to a character stream and written on the file \texttt{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:

\begin{verbatim}
THE ABOVE INDIVIDUALS HAVE BEEN RECREATED
TO COMPLETE THE WORKSPACE RECONSTRUCTION;
ENTER THE FOLLOWING:
 )WSSD MUMBLE
 )ERASE }AGENWS
 )SAVE
\end{verbatim}

where \texttt{AGENWS} is the group of global variables and functions of GENWS.
System Variables:

\[ \text{DI0} \rightarrow 1 \]

Variables:

\( \text{AAP0S} \) is used in determining the number of APL characters to be written on an output line.
DGINIT

1. DGINIT; DAWS; DTRL; D10
2. D10+1
3. 'DESTINATION WORKSPACE NAME: A'
4. DAWS=1
5. DAWS+2 5 '4[1+1 pDAWS,' ']')WSID DAWS'
6. 'TERMINAL TRANSCRIPT LINE LENGTH: A'
7. DTRL+1
8. DTRL+2 3 '5[1+1 pDTRL,' ']')PW DTRL'
9. DGENW

DGENW

1. DGENW; DAX; DAP
2. A DGENW CONVERTS THE WORKSPACE DAWS TO A TERMINAL
3. A TRANSCRIPT ON THE KEY PAIRED ASCII FILE DAWS.TSP
4. 1'')OUTPUT 'DAWS', 'TSP'
5. 1')CLEAR'
6. ')WSID ', DAWS
7. ')PP+', ')PP
8. ')PW+', ')PW
9. ')PP+18
10. ')PW+DTRL
11. ')CT+', ')CT
12. ')IO+', ')DIO
13. ')NL+', ')NL
14. A CONVERT VARIABLES IN WORKSPACE
15. DAX-2NL 2
16. DAX-((A/DAX,<5(1+DAX),5)+DAX 5 p'DAWS DAX DAXCR DAXSG DAXIO ')')DAX
17. DGENW1:=(D=0,DAX)/DGENW2
18. DTPFV DTPRM DAX[1;]
19. DAX+ 10 +DAX
20. +DGENW1
21. A CONVERT FUNCTIONS IN WORKSPACE
22. DGENW2: DAX-2NL 3
23. DAX-'DGENW DTPRM DTPRF DTPFV DTPRF DAPLQ DAPDR DAPCR DAPDT DGINIT'
24. DAX-((A/DAX,<5(1+DAX),8)+DAX 8 p'DAPE )')DAX
25. DGENW3:=(D=0,DAX)/DGENW4
26. DTPRF DTPRM DAX[1;]
27. DAX+ 10 +DAX
28. +DGENW3
29. DGENW4: ')ERASE DAPF'
30. ')SAVE'
31. 2')OUTPUT'
32. 'FILE CREATED: 'DAWS', 'TSP'
10.

**DETRAV**

A **DETRAV** PRODUCES A TERMINAL TRANSCRIPT OF A VARIABLE ASSIGNMENT

\[ \Delta f + \Delta f \]

\[ \Delta f + \Delta f \]

A **DETRAV** ASSIGN TO VARIABLE NAME ITS VALUE REPRESENTED AS A CHARACTER VECTOR

\[ \Delta f, '+', '...', '...

A **DETRAV** DETERMINE LENGTH OF CHARACTER STRING TO FILL FILE LINE

\[ \Delta f + 1 + ((\Delta f \times 2) + \Delta f) \times 2 + \Delta f \]

A **DETRAV** CHECK CHARACTER STRING DOES NOT CONTAIN ODD NUMBER OF QUOTES

\[ \Delta f + \Delta f + \Delta f - 0 = 2 \]

A **DETRAV** IF NUMERIC, CONVERT VALUE

\[ \Delta f + \Delta f + \Delta f + \Delta f \]

A **DETRAV** RESHAPE VARIABLE

\[ \Delta f + \Delta f + \Delta f \]

**DETRAV**

A **DETRAV** PRODUCES A TERMINAL TRANSCRIPT OF THE FUNCTION \( \Delta f \)

\[ \Delta f + \Delta f + \Delta f \]

\[ \Delta f + \Delta f + \Delta f \]

A **DETRAV** DETERMINE LENGTH OF CHARACTER STRING TO FILL FILE LINE

\[ \Delta f + 1 + ((\Delta f \times 2) + \Delta f) \times 2 + \Delta f \]

A **DETRAV** CHECK CHARACTER STRING DOES NOT CONTAIN ODD NUMBER OF QUOTES

\[ \Delta f + \Delta f + \Delta f - 0 = 2 \]

A **DETRAV** RESHAPE FUNCTION DEFINITION AS A CANONICAL REPRESENTATION

\[ \Delta f + \Delta f + \Delta f \]

A **DETRAV** FIX THE FUNCTION

\[ \Delta f + \Delta f \]
\texttt{\textbf{DAINIT}}

\begin{verbatim}
\texttt{\textbackslash{}DAINIT:DA\textbackslash{}WS:DA\textbackslash{}ARL:DA\textbackslash{}ACH:DA\textbackslash{}ALC:DA\textbackslash{}ASF:DA\textbackslash{}I0}
\end{verbatim}

\begin{verbatim}
[1] \texttt{\textbackslash{}DA\textbackslash{}ALC+:\textbackslash{}I0}
[2] \texttt{\textbackslash{}I0+=0}
[3] \texttt{'}\texttt{DESTINATION WORKSPACE NAME: a'}
[4] \texttt{\textbackslash{}DA\textbackslash{}WS+:\textbackslash{}U}
[5] \texttt{\textbackslash{}a(1=\textbackslash{}p\textbackslash{}DA\textbackslash{}WS, \textbackslash{}')+/\textbackslash{}DA\textbackslash{}WS+2')}\texttt{WSID'}\texttt{''}
[6] \texttt{'}\texttt{LEVEL OF CONVERSION:}
[7] \texttt{'}\texttt{ENTER 2 (CHARACTER STREAM) OR 3 (BIT STREAM): a'}
[8] \texttt{\textbackslash{}DA\textbackslash{}ALC+=2+2*\textbackslash{}a}
[9] \texttt{\textbackslash{}DA\textbackslash{}ALC/DA\textbackslash{}INIT3}
[10] \texttt{a INITIALIZATION FOR CHARACTER STREAM FILE}
[12] \texttt{\textbackslash{}DA\textbackslash{}ARL+=0}
[13] \texttt{\textbackslash{}DA\textbackslash{}ARL+2=5[1=\textbackslash{}p\textbackslash{}DA\textbackslash{}ARL, \textbackslash{}']}}\texttt{\textbackslash{}FW \textbackslash{}DA\textbackslash{}ARL'}
[14] \texttt{\textbackslash{}DA\textbackslash{}ACH+\textbackslash{\text{DASS}} \textbackslash{}DA\textbackslash{}WS, .'\textbackslash{}CRV/\textbackslash{}AS'}
[15] \texttt{\textbackslash{}DA\textbackslash{}BV+=0}
[16] \texttt{\textbackslash{}DA\textbackslash{}INIT1}
[17] \texttt{a INITIALIZATION FOR BIT STREAM FILE}
[18] \texttt{\textbackslash{}DA\textbackslash{}INIT3:DA\textbackslash{}ALC+\textbackslash{\text{DASS}} \textbackslash{}DA\textbackslash{}WS, .'\textbackslash{}BRV/\textbackslash{}BU'}
[19] \texttt{a DETERMINE FRAMESIZE}
[20] \texttt{\textbackslash{}DA\textbackslash{}ASF+12=\textbackslash{\text{Odd}}}
[21] \texttt{\textbackslash{}DA\textbackslash{}ALC+=0W}
[22] \texttt{\textbackslash{}DA\textbackslash{}INIT1:DA\textbackslash{}ACCRTV}
[23] \texttt{\textbackslash{}DA\textbackslash{}AS+DA\textbackslash{}CH}
[24] \texttt{'}\texttt{CONVERSION COMPLETED:}
[25] \texttt{'}\texttt{FILE CREATED: \textbackslash{}DA\textbackslash{}WS, \textbackslash{}7+/\textbackslash{}DA\textbackslash{}ALC+.'\textbackslash{}CRV/\textbackslash{}AS \textbackslash{}BRV/\textbackslash{}BU'}
\end{verbatim}
System Variables:

\[ \text{MO} \leftarrow 0 \]

Variables:

\[ \Delta\Delta V \text{ is the APL character vector} \]

142

\[ \text{\Delta\Delta V} \]

\[ \text{ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789.,/} \]

\[ \text{ABCDEFGHIJKLMNOPQRSTUVWXYZ} \]

\[ \Delta\Delta V \text{ is the level 3 representation of the stream identifier and the translation table. As described in the text, the translation table is an array, the rows corresponding to the APL characters in \Delta\Delta V. The number of columns is the maximum number of overstrike required to print the APL characters. Thus} \]

\[ \text{row 2 would be SPACE,'A'} \]

\[ \text{and row 142 would be ',','Z'} \]

\[ \Delta\Delta V \text{ is formed by first constructing the level 2 canonical representation vector} \]

\[ \text{CRV } \leftarrow (\text{\textasciitilde MO} \text{\textasciitilde \textasciitilde PREP 'WSIS'}, \text{TT}\Delta\text{PREP 'TRANSLATE'}} \]

\[ \text{where TT is the translation table array} \]
\Delta BV \text{ is then defined as }

O(8p2)\Delta ASCII\Delta APL\text{CRV}

where \Delta ASCII\Delta APL is the ASCII/APL
transmission code vector

(\Delta ASCII\Delta APL \text{ is defined in the workspace GENWS})

In generating the bit stream, the incoming character stream to the function \Delta OUTP is processed in blocks of 512 characters. The size of this block may need to be modified.
14.

GENCRV

\[ \Delta C N R V \; \Delta A X \; \Delta A X \]

[1] \[ \Delta C N R V \; \text{generates a workspace canonical representation vector} \]

[2] \[ \text{conforming to the STAPL proposed convention for level 2} \]

[3] \[ \Delta I O \; \Delta I O \]


[5] \[ \Delta I P + 1 8 \]

[6] \[ \Delta P W + \Delta A R L \]

[7] \[ \Delta I O + 3 \]

[8] \[ \Delta A U T E \; \Delta A X \; (\Delta A V R E E \; ' I C T ') \; (\Delta A V R E E \; ' I L X ') \; (\Delta A V R E E \; ' I R L ') \]

[9] \[ \text{generation of function canonical representation vectors} \]

[10] \[ \Delta A X + \Delta A N L \; 3 \]

[11] \[ \Delta A X + ' \Delta \text{INIT} \; \Delta A U T E \; \Delta A T R I M \; \Delta A S T R E \; \Delta A E R E P \; \Delta A V R E P \; \Delta A A R E P \; \Delta A C N R V \; \Delta A D C R V \]

[12] \[ \Delta A X + (A / \Delta A X . * ((1 + \rho \Delta A X ) \times 9) + 97 \rho \Delta A X ) + \Delta A X \]

[13] \[ \Delta A C N R V 1 + (0 = \rho , \Delta A X ) / \Delta A C N R V 2 \]

[14] \[ \Delta A U T E \; \Delta A A R E P \; \Delta A T R I M \; \Delta A X [0 ; ] \]

[15] \[ ' \text{function converted: } ', \Delta A X [0 ; ] \]

[16] \[ \Delta A X + \Delta A X \; \Delta A X [0 ; ] \]

[17] \[ \Delta A X + 1 0 + \Delta A X \]

[18] \[ \rightarrow \Delta A C N R V 1 \]

[19] \[ \text{generation of variable canonical representation vectors} \]

[20] \[ \Delta A C N R V 2 : \Delta A X + \Delta A N L \; 2 \]

[21] \[ \Delta A X + ' \Delta A X \; \Delta A X \; \Delta A R L \; \Delta A C H \; \Delta A A V \; \Delta A P V \; \Delta A E S \; \Delta A W S \; \Delta A L C \; \Delta I O ' \]

[22] \[ \Delta A X + (A / \Delta A X . * ((1 + \rho \Delta A X ) \times 10) + 105 \rho \Delta A X ) + \Delta A X \]

[23] \[ \Delta A C N R V 3 : (0 = \rho , \Delta A X ) / \Delta A C N R V 4 \]

[24] \[ \Delta A U T E \; \Delta A A R E P \; \Delta A T R I M \; \Delta A X [0 ; ] \]

[25] \[ ' \text{variable converted: } ', \Delta A X [0 ; ] \]

[26] \[ \Delta A X + \Delta A X \; \Delta A X [0 ; ] \]

[27] \[ \Delta A X + 1 0 + \Delta A X \]

[28] \[ \rightarrow \Delta A C N R V 3 \]

[29] \[ \Delta A C N R V 4 : \Delta A U T E \; ' O ' \; \Delta A A R E P \; ' E N D ' \]
\[ \Delta \text{QUOTE} \Delta \text{AY} ; \Delta \text{AZ} ; \Delta \text{AR} ; \Delta \text{AS} ; \Delta \text{AT} ; \Delta \text{AH} \]

1. IF CONVERSION COMPLETED, FLAG TO OUTPUT REMAINDER OF STREAM

2. \[ \Delta \text{AS} \times \\text{R} / \\text{SPEND} \times \Delta \text{AY} \]

3. \text{ADD CURRENT VECTOR TO ACCUMULATED STREAM BUFFER}

4. (IF BIT STREAM, ITERATIVELY ADD BLOCKS OF 512)

5. \text{CHARACTERS CONVERTED TO BIT VECTORS}

6. \[ \Delta \text{AB} = 512 \]

7. \[ \Delta \text{QUOTE} : \Delta \text{AY} = \Delta \text{AY} \]

8. \[ \Delta \text{AY} = 3 \cdot \text{31}(\Delta \text{ALC}) + \Delta \text{AY} = \text{\&}(\Delta \text{AF} \times \Delta \text{AY}) + \Delta \text{AY} \]

9. \[ \Delta \text{AY} = 2 \cdot \text{31}(\Delta \text{ALC}) + \Delta \text{AY} = \text{\&}(\Delta \text{AH} \times \Delta \text{AY}) + \Delta \text{AY} \]

10. \text{OUTPUT FULL LINES (CHARACTER STREAM)}

11. \text{OR FULL WORDS (BIT STREAM) OF ACCUMULATED STREAM}

12. \[ \Delta \text{AY} = 4 \cdot \text{2}(\Delta \text{ALC}) + \Delta \text{AY} = 36 \]

13. \[ \Delta \text{AY} = \text{\&}(\Delta \text{AE} \times \Delta \text{AY}) + \Delta \text{AY} \]

14. \[ \Delta \text{AY} = \text{\&}(\Delta \text{AE} \times \Delta \text{AY}) + \Delta \text{AY} \]

15. \[ \Delta \text{AY} = 2 \cdot 36 \cdot 13(\Delta \text{ALC}) + \text{5}(\Delta \text{ACH} \times \Delta \text{AY} \times \Delta \text{ALC}) \times 0.2 \]

16. \[ \Delta \text{QUOTE} : \Delta \text{AY} = \Delta \text{AY} \]

17. \[ \Delta \text{AY} \times \text{R} = \Delta \text{AY} \]

18. \[ \text{IF CONVERSION COMPLETED, OUTPUT REMAINDER OF STREAM} \]

19. \[ \Delta \text{AY} = 12 \cdot 31(\Delta \text{ALC}) + \Delta \text{AY} = 5 \times \Delta \text{ACH} \times \Delta \text{AY} \times \Delta \text{ALC} \times 0.2 \]
16.

**GENCRV**

\[ \Delta \Phi + \Delta C R E E \Delta \Delta \]

[1] \[ \Delta \Phi \text{ IS THE NAME OF THE VARIABLE} \]

[2] \[ \Delta \Phi \text{ GET VALUE} \]

[3] \[ \Delta \Phi \text{ CATENATE NAME, RANK, SHAPE VECTOR, AND VALUE} \]

[4] \[ \Delta \Phi \text{ APPEND DATA TYPE} \]

[5] \[ \Delta C R E E \text{ IS THE NAME OF THE FUNCTION TO BE CONVERTED} \]

[6] \[ \Delta \Phi \text{ GET CANONICAL REPRESENTATION} \]

[7] \[ \Delta \Phi \text{ APPEND DATA TYPE, RANK, AND SHAPE VECTOR} \]

[8] \[ \Delta \Phi \text{ APPEND VECTOR LENGTH} \]

\[ \Delta \Phi = (\psi \Delta \Phi), \Delta \Phi \]

\[ \Delta \Phi + \Delta C R E E \Delta \Delta \]

[1] \[ \Delta \Phi \text{ IS THE NAME OF THE FUNCTION TO BE CONVERTED} \]

[2] \[ \Delta \Phi \text{ GET CANONICAL REPRESENTATION} \]

[3] \[ \Delta \Phi \text{ CATENATE NAME, RANK, SHAPE VECTOR, AND VALUE} \]

[4] \[ \Delta \Phi \text{ APPEND DATA TYPE, RANK, AND SHAPE VECTOR} \]

[5] \[ \Delta \Phi \text{ APPEND VECTOR LENGTH} \]

\[ \Delta \Phi = (\psi \Delta \Phi), \Delta \Phi \]

\[ \Delta \Phi + \Delta C R E E \Delta \Delta \]

[1] \[ \Delta \Phi \text{ GENERATE PSEUDO-VARIABLE} \]

[2] \[ \Delta \Phi \text{ IS THE NAME OF THE PSEUDO-VARIABLE AND} \]

[3] \[ \Delta \Phi \text{ IS THE VALUE} \]

[4] \[ \Delta \Phi \text{ CATENATE NAME, RANK, SHAPE VECTOR, AND VALUE} \]

[5] \[ \Delta \Phi \text{ APPEND VECTOR LENGTH} \]

\[ \Delta \Phi = (\psi \Delta \Phi), \Delta \Phi \]
APPENDIX C

WORKSPACE GENWS

WORKSPACE RECONSTRUCTION FROM

CANONICAL REPRESENTATION

System Variables:

\( [N] \rightarrow 0 \)

Variables:

\( \Delta ASCII \Delta APL \) 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, \( \Delta ASCII \Delta APL \)
should be extended, using the standard decimal ASCII representation to
determine the indices.

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

\begin{verbatim}
V  \texttt{DLINIT;DLFN;DLRL;DLCH;DLSWS;DLIO}
[1]  'FILE TO BE CONVERTED: A'
[2]  \texttt{DLFN+1}
[3]  'TYPE OF FILE:'
[4]  ' ENTER 2 (CHARACTER STREAM) OR 3 (BIT STREAM): A'
[5]  \texttt{+( '3' = 1 )/DLINIT3}
[7]  \texttt{DLRL+1}
[8]  \texttt{DLCH+DASS DLFN,'/AS'}
[9]  \texttt{+DLINIT1}
[10] \texttt{DLINIT3;DLASPIN}
[11] \texttt{+DLTRANS/0}
[12] \texttt{DLCH+DLBCVT}
[13] \texttt{DLINIT1;DLCRCVT '}
[14] \texttt{DIO+DLIO}
[15] '
[16] 'THE ABOVE INDIVIDUALS HAVE BEEN RECREATED,'
[17] 'ENTER THE FOLLOWING:'
[18]  ' )WSID ',DLSWS
[19]  ' )ERASE \&GENWS'
[20]  ' )SAVE'
\end{verbatim}
GENS

\[ \Delta_{ERR} \rightarrow \Delta_{TRANS} ; \Delta_{EPS} ; \Delta_{ACV} ; \Delta_{ATL} \]

[1] r \ Delta_{TRANS} checks the stream identifier
[2] r AND EXTRACTS THE TRANSLATION TABLE
[3] \Delta_{ACV} = \Delta_{ASCII} \Delta_{APL} \ (802) 15 8 \ 0120 + \Delta_{BF} \]
[4] \rightarrow('9PSIS 0 0' = 10 + \Delta_{ACV}) / \Delta_{TRANS1}
[5] 'FILE DOES NOT CONFORM TO STAPL FORMAT CONVENTIONS,'
[6] 'TO INSPECT THE FILE,'
[7] \Delta_{EPN}, 'HAS BEEN ASSIGNED TO THE VARIABLE \Delta_{BF} '
[9] 'HAS BEEN ASSIGNED TO \Delta_{ACV} '
[10] \Delta_{ERR} \leftarrow 1
[11] \rightarrow 0

[12] r CONVERT THE TRANSLATION TABLE FROM BINARY
[13] r TO A CHARACTER VECTOR
[14] \Delta_{TRANS1}: \Delta_{AL} = 1 \Delta_{ACV} [9 + 1 + (-5 + \Delta_{ACV}) 'P']
[15] \Delta_{AL} = 6 \times (10 + \Delta_{AL} + 9 \Delta_{DL})
[16] \Delta_{ACV} = 1 \Delta_{ASCII} \Delta_{APL} [802] \overset{(802)\{8(-10 + \Delta_{AL} \div 8), 8) 080 + \Delta_{AL} + \Delta_{BF}}
[17] \Delta_{BF} + \Delta_{DL} + \Delta_{BF}
[18] r CONSTRUCT THE TRANSLATION TABLE ARRAY
[19] \Delta_{ACV} \Delta_{ACV}
[20] r FOR EACH ROW OF THE TRANSLATION TABLE,
[21] r INSERT BACKSPACE BETWEEN EACH CHARACTER
[22] \Delta_{ATT} \leftarrow \Delta_{TRANS} ; \Delta_{AC} + 0]
[23] \Delta_{BFS} \leftarrow \Delta_{TRANS} \ (0) \Delta_{AV} [98]
[24] \Delta_{TRANS2}: \rightarrow (\Delta_{TRANS} [1] = \Delta_{AC} + \Delta_{AC} + 1) / \Delta_{TRANS3}
[25] \Delta_{ATT} \rightarrow \Delta_{ATT}, \Delta_{BFS}, \rightarrow \Delta_{TRANS} [1] \Delta_{AC}
[26] \rightarrow \Delta_{TRANS2}
[27] \Delta_{TRANS3}: \rightarrow (\Delta_{TRANS} [1] = (\Delta_{TRANS} [1]) \rightarrow \Delta_{ATT} \leftarrow \Delta_{ERR} \leftarrow 0

\Delta_{ACV}

\[ \Delta_{ACV} \Delta_{AA} ; \Delta_{ACH} ; \Delta_{AI} \]

[1] r \ Delta_{ACV} converts the translation table
[2] r TO THE APL CHARACTER VECTOR
[3] \Delta_{ACH} = \Delta_{ASS (4 \div \Delta_{BF} \div \Delta_{BF})}, ('-2' \ WIL), ' ,CRV/AS' \]
[4] \Delta_{AI} \rightarrow \Delta_{ACH} [5] \Delta_{ACH}
[5] \rightarrow \Delta_{ACH} \Delta_{AI}
[6] \rightarrow \Delta_{AI} + 0
[7] \rightarrow \Delta_{TRANS} \leftarrow '
[8] \Delta_{ACVT} \leftarrow \Delta_{TRANS} \leftarrow \Delta_{TRANS} + 1 \div 5 \Delta_{ACH}
[9] \rightarrow ((\Delta_{AA}) [0] \leftarrow \Delta_{AI} + \Delta_{AI} + 1) / \Delta_{ACVT}
[10] \rightarrow \Delta_{ACH} \Delta_{ACH}
GENWS

ACRCVT

V

ΔACRCVT ΔNW;ΔΔP;ΔΔL;ΔΔV;ΔΔT;ΔΔN;ΔΔR;ΔΔS
[1]  A ΔACRCVT CREATERS A WORKSPACE FROM THE
[3]  ΔACRCVT1: 1(10>0ΔΔN)/'ΔΔN+ΔACFIN ΔΔV'
[4]  A DETERMINING LENGTH OF AN INDIVIDUAL
[6]  ΔΔL+(L0ΔΔN));'NCFP')4ΔΔN
[7]  ΔACRCVT2: 1(0ΔΔN)≥ΔΔL)/ΔACRCVT3
[8]  ΔΔW+ΔACFIN ΔΔN
[9]  +ΔACRCVT2
[10]  A EXTRACT INDIVIDUAL CANONICAL REPRESENTATION VECTOR
[11]  ΔACRCVT3: ΔΔW+(ΔΔV+(ΔΔN+(ΔΔL)+1ΔΔL))4ΔΔN
[12]  ΔΔP+(ΔΔV=1)];(1)4ΔΔN
[14]  ΔΔT+1ΔΔN
[15]  A GET NAME
[16]  ΔΔN+1+ΔΔN
[17]  A GET RANK
[18]  ΔΔN+2(1+ΔΔN)+ΔΔP(1)+ΔΔN
[19]  A GET SHAPE VECTOR
[20]  2'ΔΔS'=', 2+2LΔΔN)+10 ΔΔP(1)+ΔΔP(1+ΔΔR)+ΔΔN
[21]  ΔΔV+(1+ΔΔP(1+ΔΔR)+ΔΔN
[22]  +(ΔΔV=1)];ΔACRCVT5
[23]  A IF NUMERIC VARIABLE, CONVERT VALUE TO NUMBER
[24]  2ΔΔV=1');ΔΔV+84 ΔΔV=10ΔΔV+10ΔΔV 0';
[25]  A SAVE DIO OF WORKSPACE
[26]  +('DIO'=3+ΔΔN)/ΔACRCVT4
[27]  ΔΔV+ΔΔN
[28]  ΔΔN
[29]  +ΔACRCVT1
[30]  A ASSIGN VALUE TO VARIABLE OR FIX FUNCTION DEFINITION
[31]  ΔACRCVT4: 1 11[ΔΔV=12ΔΔV];'ΔΔN, 'ΔΔS+ΔΔV'
[32]  ΔΔN(ΔΔV=12ΔΔV)/ΔΔN
[33]  +ΔACRCVT1
[34]  A PROCESS PSEUDOVARAIABLES
[35]  ΔACRCVT5: 1'WSID'=4+ΔΔN)/ΔΔS+ΔΔN'
[36]  A IF BIT STREAM, CONSTRUCT TRANSLATE TABLE
[37]  +(TRANSFORM'=9+ΔΔN)/ΔACRCVT6
[38]  ΔΔTRANSLATE=(ΔΔS)ΔΔN
[39]  →0
[40]  ΔACRCVT6: 1('END'=3+ΔΔN)/ΔACRCVT1
READS THE BIT STREAM REPRESENTATION
OF THE WORKSPACE

ASS \( \text{AAPN} \), '/BY'

\( \text{AAPF} \rightarrow \text{AACCH} \), 0, 2, (\text{AFFL} \text{AACCH})[2] \times 36

\( \text{AACCH} \)

\( \text{AACCVT} \)

\( \text{AACCH} \rightarrow \text{AACCVT} ; \text{AAAL} ; \text{AAAT} ; \text{APW} ; \text{AAPF} \)

\( \text{AACCVT} \) CONVERTS THE BINARY STREAM TO A CHARACTER

\( \text{AACCH} \rightarrow \text{AACSV} ((4 \text{AAPF}) + \text{AAPF}), (2 \text{AAPUL}), '.CRV/AS' \)

\( \text{AAPR} \rightarrow \text{APW} + 128 \)

\( \text{AAPF} \rightarrow 2 \text{AAPAT} \text{TRANSLATE} \)

\( \text{AAPR} \rightarrow \text{ AAPSL } \times 512 \)

\( \text{AACVT1} \rightarrow \text{AACAT} \rightarrow \text{TRANSLATE} ((\text{AAPF} \times 2) \rightarrow (512, \text{AAPF}) \rightarrow \text{AAPAL} + \text{AAPF}) \)

\( \text{AACAT} + (4 \times 128 \text{AAPAT}) \rightarrow (5 \text{AACCH}) \)

\( \text{AAPF} \rightarrow \text{ AAPBL } + \text{ AAPF} \)

\( \rightarrow (0 \text{AAPAT}) / \text{AACVT1} \)

\( \text{AAPSL} \rightarrow \text{AACCH} \)

\( \text{AACFIN} \)

\( \text{AACFIN} \rightarrow \text{AACFNC} ; \text{AE} \)

\( \text{AACFIN} \) READS A BLOCK OF 5 FILE LINES OF

\( \text{AACCHARACTER STREAM REPRESENTATION OF THE WORKSPACE} \)

\( \text{AACCHARACTER} \)

\( \text{AADL} = 0 \)

\( \text{AACFIN1} ; \text{AADV} \rightarrow (5 \text{AACCH}) \)

\( \rightarrow (0 \rightarrow 75 = \text{A}, \text{AADV}) / 0 \)

\( \text{AADR} \rightarrow \text{AADR} + \text{AADV} \)

\( \rightarrow (5 \rightarrow \text{AADL} + \text{AADL} + 1) / \text{AACFIN1} \)
The following functions are used in the workspaces. $\Delta Q\Delta CR$ is included because, in the version of APLSF on which these programs were run, the system function $\Delta CR$ did not perform correctly.

The four utility functions assume $\Delta \Phi$ to be 1 and minor modifications are required if $\Delta \Phi$ is 0.
ΔΔQDCR

[1] ΔΔQDCR IS EQUIVALENT TO
[2] THE SYSTEM FUNCTION QCR
[3] ΔΔΔ+1+ΔΔΔ
[5] ΔΔΔ+10
[6] ΔΔQDCR1:→ΔΔQDCR2×13ζΔΔΔ
[7] ΔΔΔ+ΔΔΔ,(ΔΔΔ+1)-1
[8] ΔΔΔ-(1+ΔΔΔ)+2)→ΔΔΔ
[9] →ΔΔQDCR1
[10] ΔΔQDCR2:ΔΔΔ+((ζΔΔΔ,Γ/ΔΔΔ)ɤ’
[11] ΔΔΔ+1
[12] ΔΔQDCR3:→(ΔΔΔ>ζΔΔΔ)/0
[13] ΔΔΔ(ΔΔΔ+ΔΔΔ(ΔΔΔ)+ΔΔΔ(ΔΔΔ)+ΔΔSTRP ΔΔΔ(ΔΔΔ)+ΔΔΔ
[14] ΔΔΔ-(ΔΔΔ(ΔΔΔ)+2)+ΔΔΔ
[15] ΔΔΔ+ΔΔΔ+1
[16] →ΔΔQDCR3

ΔΔSTRP

[1] ΔΔSTRP REMOVES LEADING BLANKS
[2] FROM THE CHARACTER VECTOR ΔΔΔ
[3] ΔΔΔ+,ΔΔΔ
[4] ΔΔΔ+((ΔΔΔɤ’)10)-1)→ΔΔΔ

ΔΔTHIM

[1] ΔΔTHIM REMOVES TRAILING BLANKS
[2] FROM THE CHARACTER VECTOR ΔΔΔ
[3] ΔΔΔ+ΔΔSTRP+ΔΔΔ

ΔΔDPLQ

[1] ΔΔDPLQ INSERTS A SECOND QUOTE AFTER EACH
[2] OCCURRENCE OF A QUOTE IN STRING ΔΔΔ
[3] ΔΔΔ+ΔΔΔ
[4] ΔΔΔ+1
[5] ΔΔDPLQ:→((ζΔΔΔ<ΔΔΔ+ΔΔΔ+1(ΔΔΔ+1)+ΔΔΔ);’’’)/0
[6] ΔΔΔ+ΔΔΔ+ΔΔΔ),’’’,ΔΔΔ+ΔΔΔ
[7] →ΔΔDPLQ1
REFERENCES

