DISCLAIMER NOTICE

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

* 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 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.

APLSF, 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 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 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.
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\r\n2 \r \n\n```

while in APL/SV:

```
C\n1 \n\n```

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 OASS, OAS and OCLS. 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 (E, E, E) 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 (E) 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 E) 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 **HUMBLE**:

```
> XX+2 50 10
> XX
> 1 2 3 4 5
> 6 7 8 9 10
> vY
> [1] 'NAVY'
> [2] 'SEVERAL'
> [3] v
```

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

```
> CLEAR
> 14 ID MUBLE
> EFF+10
> EX+50
> CT=1.1365640407870355-13
> IO+1
> LRL=''
> RRL=0
> RRL= RRL,'30'
> RRL= RRL
> RRL=''
> XX+XX,'1 2 3 4 5 6 7 8 9 10'
> XX=XX
> XX+2 50 XX
> 'XX'
> DELT
> DELT=DELT,'Y
> DELT=DELT,'NAVY''
> DELT=DELT,'SEVERAL''
> DELT=3 10 DELT
> DELT=10 DELT
> SAV 35 DELT
> 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:

\[
<\text{length}><\text{type}><\text{name}><\space><\text{rank}><\space><\text{shape}><\space><\text{elements}>
\]

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:

\[
<\text{wsid}><\text{crv}><\text{crv}><\text{crv}><\text{end}>
\]

where

- **<wsid>** is the canonical representation vector naming the workspace
- **<crv>** 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:

```
16F\text{\textbackslash SID} 1 5 w u . 4 b 6 8 9\text{\textbackslash APP} 0 1 0 9\text{\textbackslash N}
\text{\textbackslash wp} 0 8 0 8 \text{\textbackslash \textbackslash \textbackslash } 0 1 3 0 \text{\textbackslash NUC} 0 1 . 1 3
0 8 0 4 0 4 0 4 7 4 0 3 8 3 1 3 9 \text{\textbackslash L W X} 1 0 8
\text{\textbackslash N AL} 0 0 0 0 \text{\textbackslash FY} 2 3 1 0 \text{\textbackslash Y}
\text{\textbackslash M NY} = \text\text{\textbackslash S E V I R AL} 3 0 4 \text{\textbackslash XX} 2 2
5 1 2 3 4 5 5 7 8 9 1 0 8 \text{\textbackslash P\textbackslash EN D} 0
\text{\textbackslash 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 (\$) 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 1011101. 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:

```
)LOAD MUMBLE
```

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

```
)COPY APL:GENT
)</INIT

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 (IPW) 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
```

```
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 (DPW) 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
```

```
DESTINATION WORKSPACE NAME: MUMBLE
LEVEL OF CONVERSION:
   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 \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 \texttt{APLSF}, a character error will occur at this point. Should this occur, the temporary file can be modified and execution resumed at \texttt{ATCVT[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:
)WSID MUMBLE
)ERASE dAGENWS
)SAVE
\end{verbatim}
where \texttt{dAGENWS} is the group of global variables and functions of \texttt{GENWS}.
8.

APPENDIX A

WORKSPACE GENNT

TERMINAL TRANSCRIPT GENERATOR

System Variables:

\[ YO \leftarrow 1 \]

Variables:

\[ \text{APL} \] is used in determining the number of APL characters to be written on an output line.
DAINIT

v
\[\text{DAINIT; } \text{DAX; } \text{DARL; } \text{IO}\]
[1] \text{DAI}+\text{IO}
[2] \text{IO}+1
[3] 'DESTINATION WORKSPACE NAME: a'
[4] \text{DAAS}+5
[5] \text{DAXS} \text{ = } 4(1+\text{DADA}), ')\text{WSID DAXS}'
[6] 'TERMINAL TRANSCRIPT LINE LENGTH: a'
[7] \text{DARL}+5
[8] \text{DARL} \text{ = } 5(1+\text{DADA}), ')\text{UPV } \text{DARL}'
[9] \text{DAENV}

\[\text{DAENV}\]

v
\[\text{DAENV; } \text{DAV; } \text{DAF}\]
[1] \text{DAENV3 CONVERTS THE WORKSPACE DAXS TO A TERMINAL}
[2] \text{TRANSCRIPT ON THE KEY PAIRED ASCII FILE DAXS.TSP}
[3] '')OUTPUT ',,DAXS',.TSP'
[4] ')CLEAR'
[5] ')WSID ',\text{DAXS}
[6] 'UPP+',\text{UPP}
[7] ')UPV+',\text{UPV}
[8] 'UPP+18
[9] 'PWT+DARL
[10] 'CT+',\text{CT}
[11] 'IO+',\text{DAINI}
[12] 'UX+',\text{UX}
[13] 'URL+',\text{URL}
[14] \text{CONVERT VARIABLES IN WORKSPACE}
[15] \text{DAV}+\text{NL }\text{2}
[16] \text{DAV}(A/\text{DAXV},((1+\text{DADA}),5)+W 5 5 \text{DAXS } \text{DAK } \text{DAV } \text{DAXS } \text{DAIL'}}\text{DAX}
[17] \text{DAENV1}:(0=0,\text{DA})/\text{DAENVW2}
[18] \text{DAENV2 } \text{DAENV3 } \text{DAENV4 }
[19] \text{DAENV2+10 } \text{DAK}
[20] \text{DAENVW1}
[21] \text{DAENV4+NL }\text{3}
[22] \text{DAENVW2: } \text{DAF+NL }\text{3}
[23] \text{DAENVW2 } \text{DAENV4 } \text{DARL4 } \text{DARTR } \text{DARTR } \text{DARTRF } \text{DARPLQ } \text{DAQCR } \text{DAINIT'}
[24] \text{DAENV2+DADA/\text{DAXV},(1+\text{DADA}),8}+W 8 8 \text{DADA}+\text{DAX}
[25] \text{DAENVW3}:(0=0,\text{DA})/\text{DAENVW4}
[26] \text{DAENVW4 } \text{DARTRF } \text{DAENV2 } \text{DAENV4 }
[27] \text{DAENVW4+10 } \text{DAK}
[28] \text{DAENVW3}
[29] \text{DAENVW4:')}\text{ERASE DAT'}
[30] ')SAVE'
[31] ')OUTPUT'
[32] 'FILE CREATED: ',,DAXS',.TSP'
V \Delta TTRV \Delta A; \Delta A; \Delta A; \Delta A

[1] \Delta TTRV PRODUCES A TERMINAL TRANSCRIPT OF A
[2] VARIABLE ASSIGNMENT
[3] \Delta A = \Delta A
[4] \Delta A = \Delta A
[5] \Delta A = \Delta A
[6] \Delta A = \Delta A
[7] \Delta TTRV1 \rightarrow (0=0\Delta A) / \Delta TTRV2
[8] \Delta TTRV2 \rightarrow (\pm = 100, 2\Delta A) / \Delta TTRV3
[9] \Delta A, \Delta A = \Delta A
[10] \Delta TTRV3 \rightarrow (0=0\Delta A) / \Delta A, \Delta A

\Delta TTRF

V \Delta TTRF \Delta A; \Delta A; \Delta A; \Delta A

[1] \Delta TTRF PRODUCES A TERMINAL TRANSCRIPT OF THE FUNCTION \Delta A
[2] \Delta A = \Delta A
[3] \Delta A = \Delta A
[4] \Delta A = \Delta A
[5] \Delta A = \Delta A
[6] \Delta TTRF1 \rightarrow (0=0\Delta A) / \Delta TTRF2
[7] \Delta TTRF2 \rightarrow (1=100, 2\Delta A) / \Delta TTRF3
[8] \Delta A = \Delta A
[9] \Delta TTRF3 \rightarrow (0=0\Delta A) / \Delta A

\Delta TTRF1

V \Delta TTRF1 \Delta A; \Delta A; \Delta A; \Delta A

[1] \Delta TTRF1 PRODUCES A TERMINAL TRANSCRIPT OF THE FUNCTION \Delta A
[2] \Delta A = \Delta A
[3] \Delta A = \Delta A
[4] \Delta A = \Delta A
[5] \Delta A = \Delta A
[6] \Delta TTRF1 \rightarrow (0=0\Delta A) / \Delta TTRF2
[7] \Delta TTRF2 \rightarrow (1=100, 2\Delta A) / \Delta TTRF3
[8] \Delta A = \Delta A
[9] \Delta TTRF3 \rightarrow (0=0\Delta A) / \Delta A
DATA STREAM

\[ \text{DDINIT: } \text{DDAS: } \text{DDAL: } \text{DDCH: } \text{DDLC: } \text{DDES: } \text{DDIO} \]

1. \[ \text{DDLG} - \text{WIO} \]
2. \[ \text{WIO} - \text{D} \]
3. \[ \text{DESTINATION WORKSPACE NAME: } \text{a} \]
4. \[ \text{DDAS} = 1 \]
5. \[ 2(1 \ast \text{DDAS}, ' ')/\text{DDWS} - 2 ' ' \text{VSID} '\]
6. \[ \text{LEVEL OF CONVERSION:} ' \]
7. \[ \text{ENTER 2 (CHARACTER STREAM) OR 3 (BIT STREAM): } \text{a} \]
8. \[ \text{DDAS} = 2 * 2 \]
9. \[ \text{DDAS} = \text{DDAS}/\text{DDINIT} \]
10. \[ \text{INITIALIZE CHARACTER STREAM FILE} \]
11. \[ \text{FILE LINE LENGTH: } \text{a} \]
12. \[ \text{DDAL} - \text{L} \]
13. \[ \text{DDAS} = 3 (5(1 \ast \text{DDAL}, ' ')) \text{UPW } \text{DDAL} ' \]
14. \[ \text{DDCH - WASS DDWS}, \text{CRV/AS} ' \]
15. \[ \text{DDAS} = - ' ' \]
16. \[ \text{DDINIT} \]
17. \[ \text{INITIALIZE BIT STREAM FILE} \]
18. \[ \text{DDINIT3: } \text{DDCH - WASS DDWS, CRV/BU} \]
19. \[ \text{DETERMINE FRAME SIZE} \]
20. \[ \text{DDCH = 2 END DDAL} \]
21. \[ \text{DDCH = UPW} \]
22. \[ \text{DDINIT1: DDCH} \]
23. \[ \text{DDAS DDCH} \]
24. \[ \text{CONVERSION COMPLETED:} ' \]
25. \[ \text{FILE CREATED: } \text{DDAS, } T - T(\text{DDCH}) + ' \text{CRV/AS } \text{CRV/BU} ' \]
System Variables:

\[ IO \rightarrow 0 \]

Variables:

\[ \Delta\Delta V \] is the APL character vector

\[ \rho \Delta\Delta V \]

\[ \Delta\Delta V \]

\[ ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789,./\{\)

\[ \Delta\Delta V \]

\[ Aa_{\ldots} \]

is formed by first constructing the level 2 canonical representation vector

\[ CRV + (\'O\' \Delta\Delta PREP \'WSIS\'), TT\Delta\Delta PREP \'TRANSLATE\' \]

where \( TT \) is the translation table array.
\( \Delta BY \) is then defined as
\[ 0(8p2)T\Delta ASCII\Delta APL\backslash CRV \]
where \( \Delta \text{ASCII}\Delta APL \) is the ASCII/APL
transmission code vector

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

In generating the bit stream, the incoming character stream to the func-
tion \( \Delta OUTF \) is processed in blocks of 512 characters. The size of this
block may need to be modified.
14.

\[ \Delta G C R Y: \Delta G C R Y \]

[1] \[ \Delta G C R Y \] \textbf{GENERATES} \textbf{A WORKSPACE CANONICAL REPRESENTATION VECTOR} \textbf{CONFORMING TO THE STAEL PROPOSED CONVENTION FOR LEVEL 2} \[ \Delta G C R Y \]

[4] \( \Delta G A = (\Delta A A P, \Delta A P P, \Delta A P V) \), \( \Delta G R E P \) \( \Delta G R D \) \[ \Delta G C R Y \]

[5] \[ \Delta G C R Y \]

[6] \[ \Delta C R Y \]

[7] \[ \Delta C R Y \]

[8] \( \Delta G R E P \) \( \Delta G R D \), \( \Delta G C R Y \) \( \Delta G R D \)

[9] \[ \Delta G R D \]

[10] \[ \Delta G R D \]

[11] \[ \Delta G R D \]

[12] \[ \Delta G R D \]

[13] \[ \Delta G R D \]

[14] \[ \Delta G R D \]

[15] \[ \Delta G R D \]

[16] \[ \Delta G R D \]

[17] \[ \Delta G R D \]

[18] \[ \Delta G R D \]

[19] \[ \Delta G R D \]

[20] \[ \Delta G R D \]

[21] \[ \Delta G R D \]

[22] \[ \Delta G R D \]

[23] \[ \Delta G R D \]

[24] \[ \Delta G R D \]

[25] \[ \Delta G R D \]

[26] \[ \Delta G R D \]

[27] \[ \Delta G R D \]

[28] \[ \Delta G R D \]

[29] \[ \Delta G R D \]
ΔΩΤΕ
(1) ΔΔΩΤΕ: ΔΔΩ: ΔΔΡ: ΔΔΩ: ΔΔΔ: ΔΔΩ
(2) ΔΔΩΔ ΔΔΩ: ΔΔΩ: ΔΔΩ: ΔΔΩ
(3) ΔΔΩΔ ΔΔΩ: ΔΔΩ: ΔΔΩ: ΔΔΩ
(4) ΔΔΩΔ ΔΔΩ: ΔΔΩ: ΔΔΩ: ΔΔΩ
(5) ΔΔΩΔ ΔΔΩ: ΔΔΩ: ΔΔΩ: ΔΔΩ
(6) ΔΔΩΔ ΔΔΩ: ΔΔΩ: ΔΔΩ: ΔΔΩ
(7) ΔΔΩΔ ΔΔΩ: ΔΔΩ: ΔΔΩ: ΔΔΩ
(8) ΔΔΩΔ ΔΔΩ: ΔΔΩ: ΔΔΩ: ΔΔΩ
(9) ΔΔΩΔ ΔΔΩ: ΔΔΩ: ΔΔΩ: ΔΔΩ
(10) ΔΔΩΔ ΔΔΩ: ΔΔΩ: ΔΔΩ: ΔΔΩ
(11) ΔΔΩΔ ΔΔΩ: ΔΔΩ: ΔΔΩ: ΔΔΩ
(12) ΔΔΩΔ ΔΔΩ: ΔΔΩ: ΔΔΩ: ΔΔΩ
(13) ΔΔΩΔ ΔΔΩ: ΔΔΩ: ΔΔΩ: ΔΔΩ
(14) ΔΔΩΔ ΔΔΩ: ΔΔΩ: ΔΔΩ: ΔΔΩ
(15) ΔΔΩΔ ΔΔΩ: ΔΔΩ: ΔΔΩ: ΔΔΩ
(16) ΔΔΩΔ ΔΔΩ: ΔΔΩ: ΔΔΩ: ΔΔΩ
(17) ΔΔΩΔ ΔΔΩ: ΔΔΩ: ΔΔΩ: ΔΔΩ
(18) ΔΔΩΔ ΔΔΩ: ΔΔΩ: ΔΔΩ: ΔΔΩ
(19) ΔΔΩΔ ΔΔΩ: ΔΔΩ: ΔΔΩ: ΔΔΩ
(20) ΔΔΩΔ ΔΔΩ: ΔΔΩ: ΔΔΩ: ΔΔΩ
16.

AV

\[ \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]

\[ \Delta \Delta \Delta \Delta \Delta \]
APPENDIX C

WORKSPACE GENWS

WORKSPACE RECONSTRUCTION FROM

CANONICAL REPRESENTATION

System Variables:

\[ \text{\( \Delta IO \leftrightarrow 0 \)} \]

Variables:

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

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

\textbf{\textit{DAINIT}}

\begin{verbatim}
  1. 'FILE TO BE CONVERTED: a'
  2. ADFN+1
  3. 'TYPE OF FILE:'
  4. 'ENTER 2(CHARACTER STREAM) OR 3(BIT STREAM): a'
  5. +(3='5')/DAINIT3
  6. 'FILE LINE LENGTH: a'
  7. ADFL+1
  8. AACH+ASS ADFN,'/AS'
  9. +DAINIT1
10. DAINIT3:DBBIN
11. +DTRANS/0
12. AACH+DAVCV
13. DAINIT1:DAVCV'T'
14. LI+DAIO
15. '
16. 'THE ABOVE INDIVIDUALS HAVE BEEN RECREATED,'
17. 'TO COMPLETE THE WORKSPACE RECONSTRUCTION,'
18. 'ENTER THE FOLLOWING:'
19. ''WSID ',DAWS
20. ''ERASE DAJENWS'
21. ''SAVE'
\end{verbatim}
\( \Delta \text{TRAN} \)

\[ \Delta \text{TRAN} = \Delta \text{TRAN} ; \Delta \text{BS} ; \Delta \text{CV} ; \Delta \text{TL} \]

1. \( \Delta \text{TRAN} \) checks the stream identifier.
2. \( \Delta \text{TRAN} \) extracts the translation table.
3. \( \Delta \text{CV} = \Delta \text{ASCII} \cdot \text{APL} [802] \cdot \text{18} \cdot 0 \cdot 120 + \Delta \text{BF} \)
4. \( + (\Delta \text{PSIS} \cdot 0 \cdot 0) = 10 \cdot \Delta \text{CV} / \Delta \text{TRAN} \)
5. "FILE DOES NOT CONFORM TO STAPL FORMAT CONVENTIONS"
6. "TO INSPECT THE FILE,"
7. \( \Delta \text{CV} \) has been assigned to the variable \( \Delta \text{BF} \)
8. "AND THE ATTEMPTED CONVERSION OF THE FIRST 15 CHARACTERS"
9. "HAS BEEN ASSIGNED TO \( \Delta \text{CV} \)"
10. \( \Delta \text{BS} + 1 \)
11. \( \to 0 \)
12. \( \Delta \text{CV} \) converts the translation table from binary.
13. \( \Delta \text{CV} \) to a character vector.
14. \( \Delta \text{TRAN} \cdot \Delta \text{CV} \cdot \Delta \text{BS} + (10 \cdot \Delta \text{CV} + 0 \cdot \Delta \text{BF}) \)
15. \( \Delta \text{CV} \cdot \Delta \text{ASCII} \cdot \text{APL} [802] \cdot \text{19} \cdot (10 \cdot \Delta \text{CV} + 8) + 0 \cdot \Delta \text{BS} + \Delta \text{BF} \)
16. \( \Delta \text{BS} + \Delta \text{CV} + \Delta \text{BF} \)
17. \( \Delta \text{CV} \) constructs the translation table array.
18. \( \Delta \text{CV} \) to \( \Delta \text{CV} \cdot \Delta \text{CV} \)
19. \( \text{FOR EACH ROW OF THE TRANSLATION TABLE,} \)
20. \( \text{INSERT \#BACKSPACE BETWEEN EACH CHARACTER} \)
21. \( \Delta \text{BS} = \Delta \text{TRANSLATE} ; \Delta \text{CV} + 0 \)
22. \( \Delta \text{CV} \) to the APL character vector.
23. \( \Delta \text{CV} = \Delta \text{TRANSLATE} [10] \cdot \text{CV} [98] \)
24. \( \Delta \text{TRAN} \cdot 1 \cdot 0 = \Delta \text{CV} \cdot \Delta \text{TRANSLATE} [10] \cdot \text{CV} [98] \)
25. \( \Delta \text{CV} \cdot \Delta \text{TRANSLATE} \cdot \Delta \text{BS} \cdot \Delta \text{TRANSLATE} [10] \cdot \text{CV} [98] \)
26. \( + \Delta \text{TRAN} \cdot 2 \)
27. \( \Delta \text{TRAN} \cdot 3 \cdot \Delta \text{CV} = (1 \cdot 0 \cdot 2 \cdot \text{TRANSLATE} [10]) \cdot \text{TRANSLATE} [10] \cdot \text{CV} [98] \)
28. \( \Delta \text{BS} + 0 \)

\( \Delta \text{CV} \)

\( \Delta \text{CV} \) converts the translation table.
1. \( \Delta \text{CV} \) to the APL character vector.
2. \( \Delta \text{CV} = \Delta \text{ASS} (4 \cdot 0 \cdot \text{DL} \cdot \text{BF}) \cdot (2 \cdot 0 \cdot \text{UL}) \cdot \text{CV} [\text{AS}] \)
3. \( \Delta \text{CV} = \Delta \text{ASS} [5] \cdot \Delta \text{CV} \)
4. \( \Delta \text{CV} \cdot \cl \cdot \text{CV} [\text{AS}] \)
5. \( \Delta \text{CV} \cdot 1 \cdot \text{CV} [\text{AS}] \)
6. \( \Delta \text{CV} \cdot 1 \cdot \text{CV} [\text{AS}] \)
7. \( \Delta \text{TRANSLATE} \cdot 2 \)
8. \( \Delta \text{CV} = \Delta \text{TRANSLATE} = \Delta \text{TRANSLATE} \cdot 1 \cdot 1 \cdot [5] \cdot \Delta \text{CV} \)
9. \( + (\text{TRANSLATE} [10] \cdot \Delta \text{CV} \cdot \Delta \text{CV} + 1) / \Delta \text{CV} \)
10. \( \Delta \text{BS} \cdot \Delta \text{CV} \)
DACRCVT

1. DACRCVT creates a workspace from the input.
2. Character stream of the workspace.
3. DACRCVT creates a vector from the workspace.
4. Determine length of an individual.
5. Canonical representation vector.
6. Extract individual canonical representation vector.
7. Get individual.
8. Get name.
10. Get shape vector.
11. If numeric variable, convert value to number.
12. Save lio of workspace.
13. Assign value to variable or fix function definition.
15. Assign value to variable or fix function definition.
16. If bit stream, construct translate table.
17. Translate vector.
ΔΔΔFIN

v ΔΔΔFIN; ΔΔCH
l 1] ΔΔΔFIN READS THE BIT STREAM REPRESENTATION
l 2] OF THE WORKSPACE
l 3] ΔΔCH=lassen ΔΔFN,’/ΔJ’
li] ΔΔΔF=ΔΔCH,0,2,(ΔΔLS ΔΔCH)[2]*36
l 5] ΔΔAS ΔΔCH

ΔΔΔCVT

v ΔΔCH=ΔΔCVT; ΔΔSL; ΔΔT; ΔΔPV; ΔΔFS
l 1] ΔΔΔCVT CONVERTS THE BINARY STREAM TO A CHARACTER
l 2] STREAM USING THE APL CHARACTER VECTOR
l 3] ΔΔCH=lassen((4lΔΔFN)+ΔΔFN),’/ΔT’,’ΔT’)
li] ΔΔRL=ΔΔSL-128
l 5] ΔΔAS=128*ΔΔTRANSLATE
l 7] ΔΔΔCVT:ΔΔT=ΔΔTRANSLATEi((ΔΔFS02)1Q(512,ΔΔFS)ΔΔSL+ΔΔBF)
lw] ΔΔΔF=ΔΔSL+ΔΔBF
li] +(0=0ΔΔBF)/ΔΔΔCVT1
l 11] ΔΔLS ΔΔCH

ΔΔΔFIN

v ΔΔR=ΔΔΔFIN ΔΔC; ΔΔL
l 1] ΔΔΔFIN READS A BLOCK OF 5 FILE LINES OF
l 2] A CHARACTER STREAM REPRESENTATION OF THE WORKSPACE
l 3] ΔΔR=ΔΔC
l 4] ΔΔL=0
l 6] +(0.75=0,ΔΔV)/0
l 7] ΔΔR=ΔΔR,ΔΔRL+ΔΔV
l 8] +(3>ΔΔL+ΔΔL+1)/ΔΔΔFIN1
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 QCR did not perform correctly.

The four utility functions assume □XO to be 1 and minor modifications are required if □XO is 0.
\[ \Delta QCR \]

\[ \Delta QCR \]

\[ \Delta QCR \]

\[ \Delta QCR \]

\[ \Delta A \]

\[ \Delta A \]

\[ \Delta A \]

\[ \Delta A \]

\[ \Delta A \]

\[ \Delta A \]

\[ \Delta A \]

\[ \Delta A \]

\[ \Delta A \]

\[ \Delta A \]

\[ \Delta A \]

\[ \Delta A \]

\[ \Delta A \]

\[ \Delta A \]

\[ \Delta A \]

\[ \Delta A \]

\[ \Delta A \]

\[ \Delta A \]

\[ \Delta A \]

\[ \Delta A \]

\[ \Delta A \]

\[ \Delta A \]

\[ \Delta A \]

\[ \Delta A \]

\[ \Delta A \]

\[ \Delta A \]
REFERENCES


**TRANSFER OF APL WORKSPACES: A USEFUL SOLUTION**

Karl Rautenkranz and Richard J. Orgass

Virginia Polytechnic Inst. & St. Univ.
Computer Science Department
Blacksburg, Virginia 24061

Air Force Office of Scientific Research/NM
Bolling AFB, Washington, DC 20332

Approved for public release; distribution unlimited.

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 DECSytem-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 DECSytem-10.