SCIENCE AND ENGINEERING RESEARCH COUNCIL
RUTHERFORD AND APPLETON LABORATORIES
COMPUTING DIVISION

Starlink Project
Starlink User Note 15.1

1 June 81

Enhancements to Starlink Interim Environment

This note should be used in conjunction with SUN/4, "An Interim Environment for the Development of Starlink Applications Software". It describes the changes made to the environment since the last software release (08-JAN-81).

Users who wish to take advantage of the new facilities should re-link their applications programs with the Starlink Subroutine Library.

Contents

1. Double Precision Program Parameters

Two new routines have been added to the Starlink library to read and write program parameters as double precision floating point values. RDKEYD will read values from the environment, WRKEYD will set values. The subroutine CALL follows the RDKEYx/WRKEYx conventions, (with the obvious assumption that the array holding the values is of DOUBLE PRECISION type).

2. Symbolic Parameter Values

When running Starlink applications programs, a user will generally specify parameter values in their explicit form. e.g.

```
BSCALE=0.84453E+04
DATE=09-MAY-81
RESET=YES
COORDS=1,1,512,512
```

He can, however, pre-assign values to a DCL symbol, which is subsequently passed to a program on the RUNSTAR command line - symbol substitution is performed before the parameter is accessed by the program, e.g.

```
$ COORDS:="65,65,256,256"
$ RUNSTAR EXTRACT/SUBFRAME='COORDS'
```

This is particularly useful when a specific value is to be used by many programs during a login session. Note, however, that this mechanism can only be used for parameter values specified on the RUNSTAR command line - DCL symbol substitution is not performed on values supplied in response to system prompts or for connection-file defaults.

Bulk data frame filenames can also be passed to programs by this technique:

```
$ HH:="DBA0:[IMAGES]HORSEHEAD"
$ RUNSTAR ZOOM/IN='HH'
```

For frame filenames, however, users will generally prefer to use VMS logical names:

```
$ ASSIGN  DBA0:[IMAGES]HORSEHEAD  HH
$ RUNSTAR ZOOM/IN=HH
```

(Note, that as HH is not a DCL symbol in this case, the single quotes are <u>not</u> used).

The advantage of this technique is that the logical name can also be specified in response to a system prompt, or as a connection-file default - full translation is performed by the system.

This facility has now been extended to 'value-type' program parameters, e.g.

```
$ DEFINE COORDS "65,65,256,256"
$ RUNSTAR EXTRACT/SUBFRAME=COORDS
```

Note, that DEFINE has been used in this context, although the equivalent ASSIGN command would have the same effect:

```
$ ASSIGN "65,65,256,256" COORDS
```

However, the choice of command is not purely arbitary - ASSIGN is really intended for equivalencing logical names to VMS file specifications, whereas DEFINE is generally used for equivalencing to application-specific 'objects'.

The system will now treat <u>all</u> parameter values supplied to a Starlink program as 'potential' logical names and will attempt a <u>recursive</u> translation. If any value cannot be translated, then it remains unchanged - it is assumed to be in its explicit form.

Conflicts may arise, however, particularly for LOGICAL and CHARACTER type parameters which are normally specified as strings of alphanumeric characters. Consider the following example, where a user has assigned the name Y to a directory specification and subsequently run a Starlink program which expects a LOGICAL-type value for the parameter, ARGS :

```
$ ASSIGN DBA0:[IMAGES] Y
  .
  .
$ RUNSTAR SHOWY/ARGS=Y
```

The user has, quite rightly in his eyes, specified that the LOGICAL parameter, ARGS, is to take the value Y, meaning TRUE. However, as a result of the previous logical name assignment, the value will actually be translated to DBA0:[IMAGES] which, in this case, is not a valid LOGICAL-type constant. (The user will be prompted by the system to re-specify the value). To overcome this problem, users can prevent a translation operation from being performed by prefixing the parameter value with an underscore:

```
$ RUNSTAR SHOWY/ARGS=_Y
```

Any value specified in this manner, whether on the command line, in response to a system prompt, or in the connection-file, will be taken to be an explicit value of the type required and no translation will be attempted (the underscore is removed before the value is passed to the program).

Users should be aware that only his process logical name table is searched during translation - entries in the group and system tables are ignored.

3. WRKEYx and Inter-program Communication

The WRKEYx routines can be used to pass values from an applications program to the user environment and thus to other programs within it. The following subroutine specification is extracted from SUN/4 for convenience:

WRKEYx - Write value(s)

These routines are used to assign values to a parameter. The only effect within the applications program is that a subsequent call to RDKEYx will fetch these values. The real use, though, is as a general mechanism for communicating with the environment and, thus, with other programs within it.

There is a routine for writing each of the five main data types:

```
WRKEYC  -  Write CHARACTER
WRKEYI  -  Write INTEGER
WRKEYR  -  Write REAL
WRKEYD  -  Write DOUBLE PRECISION
WRKEYL  -  Write LOGICAL
```

Format of call:

```
CALL WRKEYx(name,values,nvals,status)
```

Input arguments:

name	CHARACTER	expression	Program parameter name.
values	CHARACTER	array or..	Array holding values to be assigned
	INTEGER	array or..	to the parameter.
	REAL	array or..	
	DOUBLE P.	array or..	
	LOGICAL	array	
nvals	INTEGER	expression	Number of values to be assigned.

Output argument:

status	INTEGER	variable	Status return.

(Note, that if a value is assigned to a parameter through these routines, which is subsequently read as an incompatible type, the value may be fetched again from the environment).

The passing of values from a program to the present environment is achieved through the use of VMS logical names.

As an illustration, consider the following program segment:

```
      PROGRAM MINMAX
*++
*     MINMAX - Find minimum and maximum of image
*--
      IMPLICIT  INTEGER(A-Z)
      INTEGER   MIN,MAX,RANGE(2)
      . . .
      . . .
      CALL RDIMAG('IMAGE',...)
      . . .
      . . .
      RANGE(1)=MIN
      RANGE(2)=MAX
      CALL WRKEYI('RANGE',RANGE,2,STATUS)
      . . .
      . . .
      END
```

The effect of this operation is to create the VMS logical name: MINMAX_RANGE. If the minimum and maximum values, say, were 17 and 254, respectively, then the 'equivalence name' assigned to MINMAX_RANGE would be "17,254". (the comma is used as a delimiter if more than one value is written). This can be verified at the DCL level by the command SHOW TRANSLATION, e.g. :

```
$ SHO TRANS MINMAX_RANGE
  MINMAX_RANGE = 17,254   (process)
```

These values can subsequently be used as input to other Starlink programs on the RUNSTAR command line, or in response to the system prompt:

```
$ RUNSTAR STRETCH/IMRANGE=MINMAX_RANGE
```

If the parameter is to be used subsequently by many programs, the user may find it more convenient to assign a shorter name through DCL:

```
$ DEFINE MR MINMAX_RANGE
```

The name MR can then be used in place of MINMAX_RANGE thus:

```
$ RUNSTAR STRETCH/IMRANGE=MR
```

Although this facility will primarily be used to pass parameter values between applications programs, users can manipulate the values within the DCL environment through use of the lexical function, F$LOGICAL. As

an illustration, the following DCL command procedure will display a centred image on the ARGS:

```
$ !
$ ! Program SIZE will determine the dimensions of a 2-dimensional
$ ! image and set the parameters NAXIS1 and NAXIS2 accordingly.
$ !
$   RUNSTAR SIZE/IMAGE='P1'
$ !
$ ! Calculate the X and Y origins for centering the image
$ !
$   X = (512-'F$LOGICAL("SIZE_NAXIS1")')/2
$   Y = (512-'F$LOGICAL("SIZE_NAXIS2")')/2
$   IF X.LT.0 THEN X = 0
$   IF Y.LT.0 THEN Y = 0
$ !
$ ! Program DISPLAY will display an image on the ARGS, given the
$ ! X and Y origin (default = 0,0).
$ !
$   RUNSTAR DISPLAY/IMAGE='P1'/ORIGIN='X','Y'
$ !
```

The logical name constructed by the WRKEYx routines is of the form:

<program name>_<parameter name>

The program name is that supplied by the user on the RUNSTAR command line. For example, all of the following will result in a program name of DISPLAY -

```
$ RUNSTAR DISPLAY
$ RUNSTAR [USEREXE]DISPLAY
$ RUNSTAR SAP:DISPLAY
$ ASSIGN  DBB1:[DJP]NEWDISP DISPLAY
$ RUNSTAR DISPLAY
```

The parameter name is that specified as input to the WRKEYx routine which will also be found in the program's connection file.

Programmers who intend to use this facility should be aware that VMS logical and equivalence names are restricted to a length of 63 characters.

4. Parameter Name Abbreviation

Parameter names on the RUNSTAR command line can be abbreviated by truncating characters on the right. It is the user's responsibility, however, to ensure 'uniqueness' between similar names - no validation is performed by the system. For example, given the following program segment:

```
CALL RDKEYI('ABCD',...)
CALL RDKEYI('AB',...)
```

and the command:

```
$ RUNSTAR PROG/AB=256
```

the system will assign 256 to the parameter ABCD and the user will be prompted for AB, i.e. the CALL sequence dictates which parameter is assigned the value.

For the moment, this is an optional facility which is controlled by the VMS logical name, STL$TRUNCATE, which must be set by the user (to any arbitary string) if abbreviation is to be used, e.g.

```
$ DEFINE STL$TRUNCATE "YES"
```

To 'turn off' the facility, the name must be de-assigned thus:

```
$ DEASSIGN STL$TRUNCATE
```

5. Commenting Connection-file Records

Records within a program connection file may contain explanatory comments. As with DCL and VAX FORTRAN, these are denoted by prefixing an exclamation mark (!). For example:

```
ARGS/VALUE/          ! LOGICAL: Specifies whether ARGS is required
LUT/FRAME(R)         ! BDF: Input look-up table
SIZE/VALUE/64,64     ! INTEGER(2): Size of output image (note default)
```

Users are reminded that they must relink their Starlink programs before any of these new facilities can be used.

User support

Any problems encountered by users should be reported to:

Dave Pearce,
STARLINK Project,
Atlas Centre,
Rutherford and Appleton Laboratories,
Chilton,
DIDCOT,
Oxon OX11 0QX

(Network mail address - RLVAD::DJP)

Refefences

[1] Starlink User Note 4 [1980]
"Interim Environment for Development of Starlink Software"
Rutherford and Appleton Laboratories

[2] "VAX/VMS Command Language User's Guide" [1980]
Digital Equipment Corporation

[3] "VAX/VMS Guide to Using Command Procedures" [1980]
Digital Equipment Corporation