SCIENCE AND ENGINEERING RESEARCH COUNCIL
RUTHERFORD APPLETON LABORATORY
COMPUTING DIVISION

SUN/23.1

Starlink Project
Starlink User Note 23.1

Issued by

K F Hartley
09.10.81

ASPIC - A set of image processing programs

Contents

1.0 Introduction

ASPIC is a collection of programs which perform picture processing functions. It is not a monolithic package of the usual kind, nor even a tight group of interacting programs like SPICA. The common link is that they are all written using the first version of the Starlink Primitive Subroutine Interface (SUN/4.1) and are held in a single directory. They also share a common HELP/documentation library. Minimal effort is needed to add new or existing programs to the system.

They may be run using the RUNSTAR command. However this was found to be very tedious, and it offers no inter-application communication. Consequently an interim Command Language called DSCL has been developed. This will have to be replaced by the Starlink Command Language (SCL) eventually, but for the time being DSCL provides an on-line HELP facility both for ASPIC programs and DSCL itself, a procedure mechanism for creating new applications, an image stack for intermediate image storage and a convenient way of handling parameters by position, keyword or default which saves a lot of typing.

Communication between applications is also improved by having a database of information about the images displayed on the ARGS. This

makes it possible to separate the display function from manipulative functions.

It is essential to understand that ASPIC programs are perfectly ordinary Starlink programs - they are only part of ASPIC because of the directory in which they are stored. Likewise DSCL may be used to run any program. Thus any program held in any directory may be run by DSCL with no further action, except that the directory name is required if the program does not reside in the current default directory.

DSCL was developed by W F Lupton at RGO, and the ASPIC programs were written by J V Carey , K F Hartley , D J King , W F Lupton , C D Pike (RGO) and P T Wallace (RAL). Subsequent sections of this paper give further information, but the bulk of the documentation is held on-line where it may be accessed using the HELP facility or extracted ready for printing by using the ASPIC program FILEDOC.

2.0 An Introduction to DSCL

2.1 Basic Facilities

DSCL aims to prevent as much redundant program parameter specification as possible. The user can set the values of parameters, and the definitions remain in force until cancelled or until logout.

Bulk Data Frames may be stored on a stack and programs can be instructed to read or write their data from or to it. A set of stack-handling commands is provided.

Commands are provided with which the user can display and delete program parameter definitions and stack contents.

Most DCL commands (all except for symbol assignments, labels, IF and GOTO) can be used in DSCL. The procedure facility allows a file containing <u>any</u> DSCL or DCL commands to be executed, making full use of all DCL symbols, conditional tests etc. If you have a procedure called STARTUP, it is executed automatically when you enter DSCL.

Help on DSCL command syntaxes, DSCL procedures and ASPIC programs etc. can be obtained by using the HELP command, which operates just like DCL HELP and provides, in addition, help on DCL commands (sic).

All commands may be abbreviated to their briefest unambiguous form. DSCL commands will prompt for their parameters if they are omitted. (DCL ones won't).

2.2 Restrictions

It is a DCL restriction that command procedures cannot be nested more than eight deep. This will not cause problems unless user procedures are nested more than six deep!

In order to be able to communicate information to the environment, applications programs must have been linked with a Starlink library incorporating the features described in SUN/15.1. A call to a WRKEYx routine will then cause a DCL logical name to be set up, which can be translated by DSCL or used within a DSCL procedure. (The RGO local Starlink library has been modified so that, in addition, a DCL global symbol is set up. This is purely for the convenience of users who wish to manipulate parameters in DSCL procedures.)

2.3 Simple Use of DSCL

To enter DSCL from DCL, merely type the command :-

```
DSCL
```

The DSCL prompt is :-

```
Dscl>
```

This is an invitation to input :-

(i) The name of an application program ,or
(ii) The name of a DSCL procedure ,or
(iii) A DSCL or DCL command

with an optional argument list. In cases (i) and (ii), a search is then carried out for the image file or procedure file. The order of searching is :-

1 The current directory
2 The local DSCL / ASPIC directory
3 The global DSCL / ASPIC directory
4 Logical name tables

Thus, it will not normally be necessary to specify a disk or directory name. If one is specified, then no searching is done.

Note that, on receiving a command, DSCL assumes first of all that an application program is to be run. It is only when the search for an image file fails that it looks for a DSCL procedure file, and it is only when this second search fails that it tries to execute the command as a DSCL or DCL command. This obviously means that there is a possible ambiguity where, for example, a procedure has the same name as an application program. For this reason, it is possible to begin any DSCL command with a special character to indicate what sort of command it is. These special characters are :-

Application program	-	*
DSCL procedure	-	@
DSCL or DCL command	-	$

Clearly, the presence or absence of a "*" makes no difference. Note that these special characters affect only the "jump-in" point for the search. If the command starts "@" and no procedure is found, then an attempt is still made to execute it as a DSCL or DCL command.

2.4 Program Parameters

Suppose that you have a program ADD which adds two bulk data frames. It has three parameters :-

```
1     IN1   , the first input frame
2     IN2   , the second input frame
3     OUT   , the output frame
```

The normal way of running this program outside of DSCL, perhaps during its development, would be to type :-

```
RUNSTAR ADD/IN1=SALT/IN2=PEPPER/OUT=CRUET
```

(inserting your own values), or :-

```
RUNSTAR ADD
```

(responding to the prompts issued by the system).

When you communicate with DSCL, these parameters are known as ADD_IN1, ADD_IN2 and ADD_OUT and their values are of relevance only to program ADD.

The first time that you run ADD, none of the parameters are defined, so, if you merely type :-

```
ADD
```

the program will prompt you for them :-

```
IN1:=SALT          (note the convention that
IN2:=PEPPER         prompts are underlined)
OUT:=CRUET
```

Alternatively, you could have typed :-

```
ADD IN1=SALT IN2=PEPPER OUT=CRUET
```

These both have exactly the same effect, as, in fact, does :-

```
ADD SALT PEPPER CRUET
```

You could also type :-

```
ADD_IN1=SALT
ADD_IN2=PEPPER
ADD_OUT=CRUET
ADD
```

Here, we defined values for all three parameters, and DSCL will remember them for future use. DSCL never remembers parameter values unless you ask it to. Thus you can keep track of what is going on. You can always use the LOOK command in cases of doubt :-

```
LOOK ADD
```

In this case, the following output would result :-

```
ADD_IN1 = SALT
ADD_IN2 = PEPPER
ADD_OUT = CRUET
```

Typing "CLFAR ADD" will cancel all the above definitions. (They can be cancelled individually as well.)

All this would be wonderful, were ADD the only program in the world. This may be the case at present (it isn't in fact), but we can hope for a better state of affairs in the future. Suppose that we also have a program STATS which has a single parameter IMAGE which is an input frame whose size, mean data value etc. are to be calculated. We might want to run ADD and then use the output from it as the input to STATS . One way of doing this is :-

```
ADD_OUT=POWER
ADD REALSQ IMAGSQ
STATS ADD_OUT
```

In fact, we know that the value of ADD_OUT is POWER so we could just have typed "STATS POWER". However, in some circumstances we might know only the parameter name. Suppose we have a program LIMITS which takes a 1D image, displays it, and gets the user to define a left and right cutoff using the cursor. These values go into a parameter LIMS and, if they are ever needed again, can be retrieved as in the following example :-

```
LIMITS INPUT=M57.HIS
RESCALE INPUT=M57 INLIMS=LIMITS_LIMS OUTPUT=$ OUTLIMS=0,255
```

For an explanation of the mysterious "$", read on ...

2.5 The Image Stack

For many applications, it is far more convenient to use the stack than continually to have to think up new temporary names. A "$" as a parameter value (provided that it is a Bulk Data Frame) means that

the frame is to come from or go to the stack.

Note that because it is DSCL, not the application program, that handles the stack, it is not possible to respond "$" to a parameter prompt from an application program.

ADD is a good example of such an application. If we type :-

```
ADD $ $ $
```

then it is assumed that both input frames are to come from the stack and that the output frame is to go back to it. DSCL will ensure that the appropriate stack house-keeping is carried out. Thus, using the stack, DSCL behaves like an HP calculator.

Here is an example of a procedure which calculates a power spectrum from the real and imaginary parts of a Fourier Transform. It is self-explanatory and gives a taste of the power afforded through the use of DCL in DSCL procedures :-

```
!       procedure pspect
!
!       take real and imag ft frames from stack
!       and evaluate pspect on stack
!
        DUPE                ! duplicate top of stack
        MULT $ $ $          ! square one image
        SWAP                ! bring other to top of stack
        DUPE                ! duplicate top of stack
        MULT $ $ $          ! square other image
        ADD $ $ $           ! calculate the PS
```

3.0 Assistance

Assistance is provided in the form of an on-line HELP facility and as documents which may be printed on request. The appendices to this paper contain reference lists of DSCL commands and ASPIC programs.

3.1 HELP

Whilst running DSCL the command HELP gives access to a full library of on-line information using precisely the same mechanism as the standard VMS HELP facility. Outside DSCL exactly the same information may be obtained by replacing HELP in the following examples with ASPDOC. The HELP library contains information about ASPIC programs, DSCL procedures, DSCL commands and several general topics.

The following examples illustrate the use of HELP.

HELP shows how to use HELP and lists the topics.

HELP ADISP	gives information about the program ADISP.
HELP ADISP PARAMETERS	shows the parameters for that program. (Similar information could have been obtained from within DSCL by typing LOOK ADISP.)
HELP A*	gives information about all programs etc. starting with the letter A.

3.2 FILEDOC

This is an ASPIC program which may be used in two ways.

3.2.1 FILEDOC INFO=TRUE

This generates a file in the current default directory called INFO.LIS which contains a one line entry for every available ASPIC program. The file will be up-to-date, and may be printed in the usual way.

3.2.2 FILEDOC <progname>

This extracts documentation from the HELP library in a form which may be printed. It is stored in a file called <progname>.DOC in the current default directory. The same wild-card mechanism as for HELP may be used to collect separate documents for a whole class of programs.

3.3 FULLDSCL

The command PRINT ASPDIR:FULLDSCL.DOC may be used to print a more formal definition of DSCL than is contained here. There are also more examples and a description of the mechanism for creating and running procedures.

4.0 ARGS database

One of the design features of ASPIC has been the desire to separate the processing and manipulation functions from the display function. This is now possible because each time ADISP is used to display an image an entry is made in a small database. AZOOM, for example, can find out where the last image displayed was located on the ARGS screen and hence give sensible run-time defaults. Likewise routines which use the ARGS cursor "know" whether the point selected was inside an image, and can also return the co-ordinates of the selected point in "array" co-ordinates, not just in ARGS co-ordinates. Later it will be possible to use other co-ordinate systems, such as RA and DEC, to access the ARGS screen.

No programs have been implemented which read data back from the ARGS memory. Consequently programs which need to access the data, such as

SLICE, must prompt for the image name. Provided the image whose name is given is of the same size as the one displayed it need not be the one displayed. Thus a contrast enhanced or smoothed version may be displayed but a slice may be taken out of the raw data.

5.0 Demonstration

This is a simple demonstration of the use of DSCL to run a few of the ASPIC programs. It may also be used as a check that most aspects of the system are operating correctly.

First log-in to your own directory and ensure that there is some spare storage available. Then invoke DSCL by typing

DSCL

The system will continue to prompt with DSCL> until STOP is typed.

It is best first of all to reset the ARGS, in order to avoid the possibility of being confused by what somebody else has already displayed on it, so type

ARESET

Now put an image on the stack and duplicate it for later use.

PUSH ASPDIR:HORSE

DUPE

Note that ASPDIR is a logical name pointing the directory where all of ASPIC is stored. The status of the stack may be displayed by typing

LOOK STACK

The image may now be displayed by typing

ADISP $ = = = =

Note the use of $ for 'top of the stack' and '=' for 'accept the run time defaults provided'.

It may now be zoomed by using

AZOOM = = = =

Again defaults are accepted; in this case they were obtained from the ARGS database.

A standard colour table may be loaded by using the procedure

LUTCOL

The contents of this look-up-table may be seen by displaying a ramp

ABLOCK = = =

again allowing the database to suggest where it should be located.

It is possible to display a section through the image by typing

UNZOOM
LUTGREY
SLICE $ =

and using the the trackerball and buttons to select two points. (The UNZOOM and LUTGREY undo the zoom performed with AZOOM and load a greyscale lookup table.)

Now, if you like, you can clear the ARGS screen by typing

ACLEAR

The top of the stack may be replaced by a histogram-equalized version by using the command

HISTMATCH $ $

It may be worth typing

HELP HISTMATCH

to see what is being done, and

LOOK HISTMATCH

to see the current status of all the parameters. The stack now contains the original and processed images; this is why the DUPE command was used earlier. It is possible to put both images simultaneously into the ARGS by using the command

ABLINK $ $

The two centre trackerball buttons may the be used to flick between the two images. Exit from this program is by hitting the right hand button.

Find out to pan and zoom the image by typing

HELP APAN

and then run the program by typing

APAN

Having exited from APAN the demonstration is over. To exit from DSCL simply type

STOP

This restores control to VMS.

The whole of this demonstration may be invoked from VMS by typing

DSCL

followed by

DEMO/ECHO

Note the use of the ECHO qualifier to show what is going on.

Appendix 1 A summary of DSCL commands

command	format
application programs	[<commchar>]<progname>[<parval1>[<parval2>[...]]] <parvaln> is of the form [<param>][=][<value>]
procedures	<procname>[/EC[HO]][<param1>[<param2>[...]]]
DSCL commands	Behave as DCL commands
DCL commands	As normal
CLEAR	CL[EAR] [P[ARAM]]<progparam> CL[EAR] S[TACK] <progparam> is of the form <program>[_<param>]
COMPILE	COM[PILE] <procname>[/NOEC[HO]]
DUPE	DU[PE]
HELP	H[ELP][<item>[<subitem1>[<subitem2>[...]]]]
LASTX	LA[STX]
LET	[LE[T]]<progparam1>=<progparam2> In most circumstances "LET" can be omitted.
LOOK	LOO[K] [P[ARAM]]<progparam> LOO[K] S[TACK] <progparam> is of the form <program>[_<param>]
POP	PO[P]
PUSH	PUS[H] <filename>
RCL	RC[L] <filename>
STORE	STOR[E] <filename>
SWAP	SW[AP]

Appendix 2 A classified list of ASPIC programs

A2.1 Arithmetic operations

BITMASK	Takes the logical AND of a frame with a scalar.
ADD	Addition of two frames.
CADD	Addition of a scalar to a frame.
CDIV	Division of a frame by a scalar.
CMULT	Multiplication of a frame by a scalar.
CPOW	Raises each element of a frame to a power.
CSUB	Subtracts a scalar from a frame.
DIV	Divides a frame by a second frame.
DIVFF	Divides a frame by a frame, preserving the scaling.
EXP	Exponentiates each element of a frame.
LOG	Takes the natural logarithm of each element of a frame.
MULT	Multiplies two frames, element by element.
SUB	Subtracts the second frame from the first.

A2.2 Fourier techniques

COSBELL	Applies a cosine-bell function to a 1D or 2D frame.
FILDEF	Creates a 1D or an axi-symmetric 2D filter.
FFT	Applies a fast Fourier transform to a 1D or 2D frame.
PSFEST	Extracts a point-spread-function (star profile) from an image.
PSPEC	Computes the power spectrum from the Fourier transform.

A2.3 Non-Fourier filtering

BLURR	Generates a 5*5 frame containing a Gaussian profile.
CONV	Convolves an image with a second (smaller) one, as may have been generated by BLURR or PSFEST.

FINULS	Filters image noise using local statistics.
LAPLACE	Forms the difference between an image and some multiple of its Laplacian.
MEDIAN	Applies a median filter at each point in an image.
MODAL	Applies a modal filter to discrete boxes in an image, and then uses linear interpolation for intermediate values.
MODE	Applies the modal filter at each point of an image.
NITPIK	Removes small defects from an image.
SMOOTH	Smooths an image by Gaussian or top hat convolution.
USMASK	Forms an 'un-sharp' masked image by subtracting a smoothed version of an image from the original.

A2.4 <u>Extraction of frame parameters</u>

DESCR	Shows one or all the descriptors of a frame.
COG	Shows the centre of gravity (in X and Y) of an image.
MEANVAL	Finds the mean value of a frame.
SECTOR	Defines, displays and stores the radial profile in a cursor-defined sector of an image displayed on the ARGS.
SLICE	Defines, displays and stores a 1D cursor-defined slice through an image displayed on the ARGS.
STAR	Finds the location, size and intensity of a star which is assumed to occupy most of an image.
STATS	Shows dimensions, range of values and so on of a frame.
TBXY	Returns the location of a cursor-defined point in an image displayed on the ARGS.
TBXY2	Returns the location of a pair of cursor-defined points in an image displayed on the ARGS.
WRHIST	Forms and stores and/or writes out the histogram of a frame.

A2.5 Image expansion/contraction

CMPRS Compresses an image by integer factors in X and Y.

COMPAVE Compresses an image by averaging adjacent elements.

COMPICK Compresses an image by selecting elements in a regular grid.

EXPAND Expands part of an image by Sinc interpolation.

MANIC M and N image conversion - picks part of a 1D or 2D or 3D frame and converts it into a 1D or 2D or 3D frame.

PIXDUPE Expands an image by pixel duplication.

A2.6 Geometrical manipulations

DEPROJ Executes all the commands needed to de-project the image of a galaxy.

FLIP Inverts an image with respect to a horizontal or vertical axis.

GTAPLY Applies a polynomial geometrical transformation to an image.

GTCALC Computes a geometrical transformation from two lists of corresponding reference points.

GTLIST Lists a set of geometrical transformation coefficients.

GTLOAD Allows keyboard input of a set of reference points.

GTLOG Prints a log generated by GTCALC.

GTPROJ Allows cursor definition of the elliptical image of a circular galaxy displayed on the ARGS and stores two sets of reference points (major and minor axes).

GTSHOW Generates a plot for the Versatec, showing a set of reference points.

GTSTAR Allows cursor location of a set of stellar reference points in an image displayed on the ARGS. It also may be used to list the points or generate input to the ASTROM astrometric package.

MOVE Moves an image by a non-integral shift in X and Y.

A2.7 ARGS display functions

ABLINK Displays two images on the ARGS and allows them to be registered and blinked under trackerball control.

ABLOCK Displays a ramp on the ARGS.

ACLEAR Clears the ARGS display.

ADISP Displays an image on the ARGS, with automatic scaling of intensity values, if required.

AFRAME Displays a frame and graticule round the latest image displayed on the ARGS.

APAN Pans and zooms the ARGS display. Returns final position of cursor.

APANG Pans and zooms the ARGS display, with independent scaling in X and Y. Returns final position of cursor.

ARESET Resets all ARGS functions.

ATEXT Places text on the ARGS display.

AZOOM Zooms the ARGS about a defined position.

A2.8 Look-up-table operations

COLSEL Selects colours from a pre-defined palette. A standard palette is supplied by default.

HSICOL User definition of a palette for input to COLSEL, if the default is not acceptable.

LUTCOL Writes a standard colour LUT to the ARGS.

LUTCONT Fills the ARGS colour table with discrete values, to give a contour-like display.

LUTE Interactive tuning of a LUT using the push-buttons.

LUTFC Loads the 'false-colour' LUT. (As used in the opening demonstration at RAL).

LUTGREY Loads the standard grey LUT.

LUTLIN Interactive manipulation of a linear (or logarithmic) LUT using the args cursor.

LUTREAD Reads any pre-defined LUT and writes it to the ARGS.

LUTROT	Cycles a LUT using trackerball.
LUTSET	Fills part of the ARGS colour table with values linearly interpolated between two RGB sets.

A2. 9 False colour

FC	Performs the "standard" false colour and colour enhancement operations in sequence.
FCDISP	Dispalys a false colour image with the correct LUT and zoom.
FCPACK	Forms a single false colour image from three (R,G,B) input images.
FCSAT	Performs enhancement on the "saturation" image.
FCSCALF	Allous rescaling of three input images to a common background and exposure.
FCTHSI	Converts a set of R,G,B images to a set of H,S,I images. (Hue , Saturation and Intensity)
FCTRGB	Converts a set of H,S,I images back to R,G,B.
LOG	May be used for logarithmic intensity enhancement.

A2. 10 Other forms of display

BONW	Sets the ARGS background to white with black lines, when using the ARGS in vector rather than image mode.
CONTOUR	Draws a contour map of an image on one of several devices.
HIDE	Draws various forms of hidden-line plot.
LINPLOT	Draws the graph of a 1D frame on one of several devices.
LIMITS	Draws a graph of a 1D frame and asks for cursor selection of two points.
LIST	Lists part of a frame.
PEEP	Types a 9*9 section of an image.
PLOTIT	Plots the output generated by VERGREY on the Versatec.

VERGREY Generates pseudo-greyscale output for the Versatec.

WONB Sets the ARGS to white lines on a black background when it is used in vector mode.

A2.11 Image compression

CONFLEV Rescales an image using the background level and noise variance.

PACK Packs an image which has been scaled from 0 to 1 (thresholded) or 0 to 3 (confidence levels) by CONFLEV.

UNPACK Unpacks images created by PACK. (PACK and UNPACK may be used to compress images for transmission over the network)

A2.12 Data input

TOSTAR Converts RGODR-format images into Starlink frames.

TYPEIN Allows keyboard input to a 1D frame.

A2.13 Miscellaneous

FILEDOC Generates copies of the HELP available on any topic, in a form suitable for printing.

GRID Incorporates a black and white grid into an image.

HISTMATCH Rescales an image so that its histogram has a prescribed form.

LSEE A simple to use image display package.

PARDIF Generates a crude estimate of the partial derivative of an image.

PCT Generates the principle component transformation on a set of up to 4 frames.

RESCALE A general rescaling and format conversion program.

PATCH Replaces selected regions of an ARGS displayed image with a smooth or noisy patch.