RUTHERFORD & APPLETON LABORATORIES
COMPUTING DIVISION

Mr N. Metcalfe

Astrophys

Starlink Project

S T A R L I N K U S E R N O T E 10.1

ARGS Subroutine Library (ARGSLIB)

20 May 1981

The ARGS FORTRAN-callable subroutine library ARGSLIB provides access to some facilities in the ARGS.

ARGSLIB currently provides the following capabilities:

allocate the ARGS to the program and reset if required (SRINIT)
output image stored in array (packed or unpacked) (SRPXP, SRPXI2)
alter colour table (SRCOLS or SRCOL1)
draw filled rectangle (SRBLOC)
buffer control (SRSEND)

The intention is to provide further ARGS facilities through ARGSLIB and as many as is reasonable through the device-independent graphics system GKS.

For the purpose of this note, the ARGS has a coordinate system of 0 to 511 in each of x and y and at each point a value can be written. The least significant 8 bits of each value are used as an index to a colour table. The colour table can be altered at any time and any change has an effect on an image already present. Each entry in the table is a triad of intensities driving red, green and blue. Intensity values are in the range 0-255. The image is contained in the ARGS pixel store and this is scanned in conjunction with the colour table by a video processor, which refreshes a colour monitor.

An ARGS is selected in SRINIT through the VMS logical name ARGS_DEVICE. A default value for this exists when the user logs in, but can be altered (ASSIGN command). The device names themselves are set up by each site manager, but are normally IDA0:, IDB0: etc.

An ARGS is reserved when SRINIT is called. The ARGS becomes free again when the program stops, unless the program is broken into. If required, the ARGS can be reserved between programs by issuing the VMS command ALLOCATE before the first of them.

The routines are described in alphabetical order. The arguments are

described in the order in which they must be supplied by the caller. An argument marked as 'entry' is one whose value(s) is/are significant on entry to the routine. An argument marked as 'exit' might, at some time during the execution of the routine, have been altered.

SRINIT must be called before anything else in this package, otherwise all sorts of error messages will go to an obscure stream (probably zero).

Error messages from the package go to stream 6 if SRINIT is properly called. Error messages are of 3 sorts:

(a) Those resulting from mistakes in the application program begin:

    ARGSLIB error

    and are followed by an error code, a text message and a list of auxiliary values.

(b) Those resulting from some hardware or operating system event. These do not begin with a predefined text pattern, but each makes it obvious that it concerns the ARGS.

(c) Those resulting from internal errors in ARGSLIB. These begin:

    ARGSLIB INTERNAL ERROR

    and continue with an error code and a list of auxiliary values. Such errors should be reported to the author in full.

Error messages in classes (a) and (b) are listed in Appendix B.

The coordinate system is in ARGS device coordinates: currently 0 to 511 in each of X and Y. (0,0) is bottom left and (511,511) is top right.

The library can be searched by including the parameter

    ARGSLIB/LIB

in the LINK command.

NOTE The subroutine names have all been altered since an earlier description of this package. Argument lists have also changed - AFLUSH(-1) has become SRSEND and the FAIL argument has been dropped from several routines. The old names will remain in the library for a short overlap period, but users should change completely to the new names as soon as possible.

Whether or not the user is already using the new names, all programs should be relinked as soon as possible, since subsequent versions of the ARGS driver will not be compatible with old versions of the ARGS library.

SRBLOC - draw filled rectangle

Arguments

INTEGER X1,Y1 (entry) One corner of the rectangle to be drawn.

INTEGER X2,Y2 (entry) Diagonally opposite corner of rectangle.

INTEGER Z (entry) Z Data In value for this call (range 0 to 255 inclusive).

X1,Y1,X2 and Y2 should all be in the range 0 to 511 inclusive.

Description

SRBLOC fills a rectangle, whose sides are parallel to the axes, with one pixel value.

Example

Draw a red rectangle.

```
CALL SRCOL1(15, 255,0,0)
CALL SRBLOC(200,200, 300,300, 15)
CALL SRSEND
```

SRCOL1 - change one entry in the colour table

Arguments

INTEGER N (entry) Index of entry in colour table to be updated. Must be in range 0 to 255.

INTEGER RED,GREEN,BLUE (entry) Intensities of the 3 primary colours to be put into the table entry. Range of each intensity is 0 to 255.

Description

Puts red,green and blue intensities into one entry of ARGS colour table.

Example

The example for SRBLOC includes a example of SRCOL1 being used.

SRCOLS - change part or all the colour table

Arguments

INTEGER FIRST (entry) Index of first entry in ARGS colour table to be altered. Range is 0 to 255.

INTEGER N (entry) Number of entries in colour table to be altered.

INTEGER COLOUR(3,N) (entry) Colour array to be used to update the selected part of the ARGS colour table. COLOUR(1,x), COLOUR(2,x) and COLOUR(3,x) are the red, green and blue values respectively for colour table entry (FIRST-x+1).

Description

SRCOLS updates all or part of colour table.

Example

```
C DEFINE A GREY-SCALE.
      INTEGER COLOUR(3,256)
      DO 100 J=1,256
        K=J-1
        COLOUR(1,J)=K
        COLOUR(2,J)=K
 100    COLOUR(3,J)=K
      CALL SRCOLS(0,256,COLOUR)
```

SRINIT - allocate and optionally reset the ARGS

Arguments

INTEGER ARGSID (entry) Currently unused so any integer value can be put here. An ARGS is selected by the logical name ARGS_DEVICE.

LOGICAL RESET (entry) If and only if RESET is .TRUE., reset ARGS after allocating.

INTEGER FAIL (exit) Failure code. 0 is OK.

Description

Allocates the ARGS identified by ARGS_DEVICE to this program. If required, the ARGS is reset: the pixel store is cleared and the colour table is reset. If a failure occurs (FAIL.NE.0), the remainder of the ARGS package can still be obeyed, but will not produce any more ARGS output.

Example

```
C ALLOCATE ARGS BUT DO NOT RESET
```

```
INTEGER FAIL
CALL SRINIT(0,.FALSE.,FAIL)
IF( FAIL.NE.0 )WRITE(6,*)'FAIL CODE FROM ALLOCATING ARGS IS',FAIL
```

SRPXI2 - draw image using unpacked values

Arguments

INTEGER*2 PIXELS(D1,NY) (entry) Array containing pixel values unpacked (each pixel is contained in one array element at the least significant end). The first subscript increases as the x-coordinate increases (going toward the right). The second subscript increases with the y-coordinate (going upwards).

INTEGER D1 (entry) 1st dimension of PIXELS array.

INTEGER NX,NY (entry) Number of pixels in NX and NY directions.

INTEGER X,Y (entry) Position at which PIXELS(1,1) is placed.

INTEGER BITS (entry) Number of bits in pixel value to be sent to the ARGS (must be power of 2 and between 1 and 16 inclusive). Anything other than 16, causes packing to take place internally. In the current version of the package, 16 can always be used even if the spread of pixel values would suggest a much smaller bit value. The high I/O rate to the ARGS ensures that the time taken to send a 16-bit image is smaller than the time the VAX takes to pack.

LOGICAL OGICAL (entry) For the time being this should be .FALSE.

INTEGER*2 WORK(DWORK) (exit) Array for workspace. This is used for the pixel packing.

INTEGER DWORK (entry) Dimension of WORK. If no packing takes place (BITS=16), this can be 1. Otherwise it must be big enough to contain at least one row of pixels (i.e. at least NX/(16/BITS)+1).

Description

The pixel values in array PIXELS are delivered to the ARGS. The colour table is not altered.

Example

```
INTEGER*2 PIXELS(512,512),DUMMY(1)
INTEGER OX,OY,FUNC
  :
  :
NX=400
NY=450
```

```
C   CALCULATE PIXELS FOR  'FUNC' IS SOME EXTERNAL FUNCTION FOR
C   GENERATING A VALUE AT A GIVEN POINT.
      DO 100 JY=1,NY
        DO 90 JX=1,NX
 90       PIXELS(JX,JY)=MOD(FUNC(JX,JY),32768)
 100    CONTINUE
C   SET THE ORIGIN SO THE PICTURE IS CENTRED.
      OX=(SIZE-NX)/2
      OY=(SIZE-NY)/2
C   NOW SEND TO THE ARGS
      CALL SRPXI2(PIXELS,SIZE, NX,NY, OX,OY, 16, .FALSE., DUMMY,1)
```

SRPXI2D - draw image using unpacked values (fast but tramples)

This was a temporary alternative to SRPXI2 to gain speed but it is now not required and will be removed from the library. Calls to it should be replaced by SRPXI2.

Arguments

PIXELS array is therefore both 'entry' and 'exit'.

Description

Example
Similar to SRPXI2 example with the name changed.

SRPXP - output packed image

Arguments

INTEGER*2 BUFF(DIM) (entry) Array containing the closely packed pixel values. The values are written to the ARGS pixel store from the least significant end of each INTEGER*2 element first.

INTEGER NX,NY (entry) Number of pixels in X and Y directions respectively.

INTEGER X,Y (entry) X and Y coordinates of the bottom left corner of the rectangular array.

INTEGER BITS (entry) Number of bits in pixel value (1,2,4,8 or 16).

INTEGER DIM (entry) DIM dimensions BUFF.

Description

Example

SRPXPD - output packed image (fast but tramples)

This was a temporary alternative to SRPXP to gain speed, but is now not required and will be removed from the library. Calls to it should be replaced by SRPXP.

Arguments

BUFF is therefore both 'entry' and 'exit'.

Description

Example

SRSEND - output and clear buffer

Arguments

None.

Description
The ARGS buffer is built up by calls to the other routines, some of which output the buffer immediately after. Certain routines do not and a call to SRSEND guarantees that the buffer is output. The routines that are guaranteed to output the buffer are: SRCOLS,SRPXI2,SRPXI2D,SRPXP and SRPXPD.

Example

```
      N=127
      DO 200 J=1,N
 200    CALL SRCOL1(J, J*2,J,0)
      CALL SRSEND
```

Appendix A NAMES

The global names used in this package are as follows:

(a) public routines beginning with SR (described in this document)

(b) internal subprograms beginning with ARGS_

(c) internal subprograms beginning with GRP_

(d) COMMON blocks of the form CARGSn (where n is an integer).

Appendix B ERROR MESSAGES

Errors in calling sequence

These begin 'ARGSLIB error' and also contain an error code, a text message and a list of auxiliary values.

1.	SRBLOC	Rectangle corners are out of range.	Arguments X1,Y1,X2,Y2
2.	SRBLOC	Z value is out of range.	Argument Z
3.	SRCOL1,SRCOLS	Index of entry in colour table is out of range. For SRCOLS, the auxiliary values are the arguments FIRST and N. For SRCOL1, only the first auxiliary value is significant and this is the faulty index.	<-
4.	SRCOL1,SRCOLS	One or more of the R,G or B intensities for a particular table entry is out of range.	R,G and B
5.	SRPXI2,SRPXP	The number of bits is wrong.	Number if bits
6.	SRPXP	The length of the pixel array is insufficient to accomodate the information specified.	Arguments NX,NY, BITS,DIM
7.	SRPXI2	The length of the array provided by the caller to accomodate the workspace is not enough.	Argument DWORK & (NX-1)/(16/BITS)+1

Hardware and Operating System events

If an error is detected by hardware or operating system, a flag is set which prevents further system service calls to the ARGS until SRINIT is called again. This allows ARGSLIB routines to be executed without causing an encyclopaedia of error messages.

(a) ASSIGNING ARGS ARGS_DEVICE
This is mainly for information. ARGS_DEVICE is the logical name used and SRINIT causes this to be output before the VMS routine SYS$ASSIGN is called.

(b) ARGS ALREADY ALLOCATED TO ANOTHER USER
ARGS device driver prevents use by 2 users simultaneously. SRINIT causes this message to be output after SYS$ASSIGN has failed from this cause.

(c) ARGS OF THIS I.D. DOES NOT EXIST
Expansion of the logical name ARGS_DEVICE results in an unrecognised device name. Could be due to user error or due to device not being connected by the site manager. Device names normally used are _IDA0: or _IDB0:. This error could also be due to ARGS_DEVICE not having been set up as a logical name.

(d) FAILED TO ALLOCATE ARGS -- REPLY FROM SYS$ASSIGN IS <code>
The SYS$QIO(W) used to reset the ARGS has returned a failure code not recognised as corresponding to situations (b) or (c).

(e) FAILED TO RESET ARGS (followed by code)
The SYS$QIO(W) used to reset the ARGS has returned a failure code.

(f) FAILED TO SEND BUFFER TO ARGS (followed by code)
Failure code from SYS$QIO(W).

(g) RESULT OF ARGS SYSTEM SERVICE CALL IS .... (followed by code)
This is output after some of the above messages. The code is the result of the system service call when considered as a function.

SCIENCE AND ENGINEERING RESEARCH COUNCIL

RUTHERFORD & APPLETON LABORATORIES
COMPUTING DIVISION

Starlink Project

Addendum 1 to S T A R L I N K U S E R N O T E 10.1

Changes to ARGSLIB

issued by
J R Gallop

24 August 1981

Release 3 of the ARGS library (internally dated 20 August 1981) has some changes of which the most important are noted here:

(a) Linking has to be performed differently, because BLOCK DATA is used in the library. Use the parameter:

```
@ARGSLIB
```

in the link command. This automatically links the ARGS library, including the BLOCK DATA.

(b) The library gets round an addressing peculiarity in the VAX, which in some situations caused a machine crash.

(c) There are some trackerball/cursor routines. A demonstration program TBC is listed at the end of this document. If these routines are used, an additional link parameter ARGS_CODE_TBC should precede @ARGSLIB viz

```
ARGS_CODE_TBC,@ARGSLIB
```

(d) Some changes in the error messages, which should be obvious. It should now be less likely for user mistakes to cause ARGS INTERNAL ERROR: any occurrences should be reported as bugs.

(e) All error messages are reported through a single routine, which is replaceable by the application programmer.

(f) Old names (e.g. AINIT, ACOLS) used in documentation previous to SUN/10.1 are now removed.

Full details will appear in a revision of SUN/10. In the meantime, details may be obtained from Julian Gallop (RLVAD::JRG).

```
      PROGRAM TBC

*+

*  - - - - - -
*  :  T B C  :
*  - - - - - -
*
*
*  TEST ARGS TRACKERBALL/CURSOR PACKAGE
*
*
*  PTW/JUN-81
*  ALTERED BY JRG AUGUST 1981
*    REFERS ONLY TO SYSTEM CURSOR
*
*-


*  ALLOCATE ARGS
      CALL SRINIT(0,.FALSE.,JSTAT)
      IF (JSTAT.NE.0) GO TO 9000

*  ENABLE ONLY SYSTEM CURSOR
      CALL ARGS_CURS('+')

*  CENTRE CURSOR
      CALL ARGS_CURP(0,255,255)

*  SET CURSOR COLOUR
      CALL ARGS_CURC('W')

*  LOAD TRACKERBALL/CURSOR PROGRAM INTO ARGS
      CALL ARGS_TBCL(0)

*  SWITCH ON LAMPS
      CALL ARGS_LAMPS(0,0,1,1)

*  LOOP UNTIL BUTTON 4 PRESSED
      IB4=0
      NPOINT=0
      DO WHILE (IB4.EQ.0)

*     GET A CURSOR X,Y
         CALL ARGS_TBCX(IX,IY,IB1,IB2,IB3,IB4)

*     IF BUTTON 3 PRESSED DISPLAY X,Y
         IF (IB3.NE.0) THEN
            NPOINT=NPOINT+1
            PRINT *,NPOINT,IX,IY
         END IF
      END DO
```

```
*  SWITCH OFF LAMPS
      CALL ARGS_LAMPS(0,0,0,0)

   DISABLE CURSOR
      CALL ARGS_CURS('0')
      CALL SRSEND

*  EXIT
      GO TO 9999

*  ERRORS
 9000 CONTINUE
      PRINT *,'ARGS ERROR'
 9999 CONTINUE
      END
```