:IENCE AND ENGINEERING RESEARCH COUNCIL SUN/12.1

RUTHERFORD & APPLETON LABORATORIES
COMPUTING DIVISION

Starlink Project

S T A R L I N K U S E R N O T E 12.1

GKS Primer

15 June 1981

ıis is a primer for the GKS package on the VAX. It supercedes all versions of LSN10 (Chilton).

Apart from section 9, it is substantially VAX independent. The facilities in the GKS device drivers are limited at present; further facilities will be added later. Thus facilities like choice of marker types (4.2) and area filling (4.4) may not be present.

If more detail is required, spare copies of the GKS 6.2 reference document can be obtained from Julian Gallop. This is language-independent and the implementation document for the Starlink VAXs is in SUN/11.

The Starlink convention for GKS names is that the initial G of each subroutine name is replaced by GKS_ (4 characters). For example, on Starlink:

GOPKS becomes GKS_OPKS

1. GKS - THE GRAPHICAL KERNEL SYSTEM

GKS - the Graphical Kernel System - is a graphics package consisting of a set of Fortran callable routines. In its simplest form it is a low level package capable of producing output on a graphic display. However, when using its more complex facilities, it is possible for a user to work interactively with a number of devices at the same time. The primitives of GKS are designed to work for a variety of terminals ranging from simple plotters and storage tubes to high quality refresh display or colour raster systems. Facilities are provided for both temporary and long term graphical storage.

This primer will introduce the reader to some of the basic facilities of GKS and indicate those areas which are completely covered in the reference document.

GKS is divided into 6 levels called 0, 1, 2a, 2b, 3 and 4. Each layer includes all layers of a lower number together with additional facilities. The GKS system available to STARLINK is currently Level 3.

2. INITIALISATION

Just getting started with GKS is rather long winded if you only want to do simple things. First, you have to tell the system that you want to use GKS:

```
CALL GOPKS(22)
```

The parameter is an integer defining a Fortran unit on to which error messages and warnings are output when anything goes wrong. You can select any unit that you are not using for something else (see section 9). If this unit is connected to a file rather than a terminal, it is necessary to print out the file to see what has happened if anything goes wrong.

You next need to tell GKS which output device you want to use:

```
CALL GOPWK(4,0,3)
```

The first parameter (the 'workstation identifier') defines the number you want to call this output device. GKS calls output devices 'workstations' and it is possible to have a number of workstations open at the same time. Consequently, you can call GOPWK several times to tell the system which devices you intend to use.

The meaning of the second parameter, the 'connection identifier' depends on the device being used - see section 'USE ON THE VAX'.

The third parameter says what type of workstation this is. GKS needs to know whether it is outputting to a plotter or a refresh display as it will do different things in certain situations dependent on the type of device. The types 100, 200 and 300 are rather special in that they are used for saving graphical output. The possible output types are listed in the section "USE ON THE VAX". The parameter 3 in the example indicates (on the VAX) that the device we have connected is a GOC. Note that at present, for T4010 and GOC, the output goes to the terminal on which you are logged in.

The next thing we need to do is to activate the workstation. Once it is activated any output commands will cause output to appear on that device until you deactivate it. If you have opened several workstations, you can activate and deactivate them as you are outputting graphical information so that only a subset of the output appears on each device. As we only have a single output device, we might as well activate it immediately by:

```
CALL GACWK(4)
```

Later on, when we have finished all the graphics that we require, we need to deactivate the workstation, close it down, and close GKS itself. Thus, to tidy up before we finish executing the program, we should:

```
CALL GDAWK(4)
CALL GCLWK(4)
CALL GCLKS
```

If you are only ever outputting to one device, it is probably sensible to define a couple of small routines which do the setting up and closing down of GKS:

```
SUBROUTINE OPENGK
CALL GOPKS(22)
CALL GOPWK(4,0,3)
CALL GACWK(4)
RETURN
END

SUBROUTINE CLOSGK
CALL GDAWK(4)
CALL GCLWK(4)
CALL GCLKS
RETURN
END
```

3. USER COORDINATES

Before you can draw any lines, you need to know what coordinate system is being used. When you start, GKS defines the largest square that you can output on the workstation as having the coordinates (0,0) in the bottom left hand corner and (1,1) in the top right. If you are lazy, you can start drawing straight away using these coordinates. You are not allowed to draw outside this area so you have to remember that.

Normally, these are not the most appropriate units for your particular problem in which case you can redefine them by:

```
CALL GSW(-10.0, -5.0, 10.0, 5.0)
```

The first two arguments define the coordinates (X,Y) of the bottom left-hand corner, while the last two define the top right-hand corner. This is defined as the 'window' that you are going to output through and the coordinate system being used is called 'world coordinates'.

You can redefine your coordinate system any time you like. Once you become more proficient, you will find that you can place your window anywhere on the display surface of the device and even have it in different places on different devices. For the moment, we will leave it with the window being mapped on to the largest square on the display surface.

4. OUTPUT PRIMITIVES

4.1 Drawing Lines

Probably the simplest thing that you want to do is draw a line. Suppose we want to draw one from the bottom left hand corner (-10,-5) to the top right corner (10,5). Here is another surprise that you get when using GKS. There is not a routine for drawing a line! The GKS philosophy is that most of the time users are not drawing a single line but whole set of lines connected together. Consequently, the only routine for drawing lines is:

```
CALL GPOLYL(N,AX,AY)
```

This draws a set of lines connecting the points AX(1),AY(1) to AX(2),AY(2) and AX(2),AY(2) to AX(3),AY(3) right up to AX(N-1),AY(N-1) to AX(N),AY(N). For example, drawing a square around the boundary of our window would require:

```
      REAL BOUNDX(5),BOUNDY(5)
      DATA BOUNDX/-9.99, 9.99,9.99,-9.99,-9.99/
      DATA BOUNDY/ -4.99, -4.99, 4.99,  4.99, -4.99/
      . . . .
      CALL GPOLYL(5,BOUNDX,BOUNDY)
```

You could, of course, set the values to BOUNDX and BOUNDY in the program. However, in GKS you tend to use DATA statements more than you would in the more conventional graphical packages with a line drawing routine.

There are some good reasons why GKS does not have a routine for drawing a line and, if you want to output a connected set of lines, you should use the polyline routine, GPOLYL. Of course, if you always output separated single lines, you could define your own routine:

```
      SUBROUTINE VEC(X1,Y1,X2,Y2)
      DIMENSION X(2),Y(2)
      X(1)=X1
      X(2)=X2
      Y(1)=Y1
      Y(2)=Y2
      CALL GPOLYL(2,X,Y)
      RETURN
      END
```

However, if you want to output a connected set of lines, it is more efficient to use GPOLYL directly.

Getting back to where we started, if you want to draw a diagonal:

```
      CALL VEC(-10.0,-5.0,10.0,5.0)
```

or

```
      DIMENSION X(2),Y(2)
      DATA X/-10.0,10.0/
      DATA Y/ -5.0, 5.0/

      CALL GPOLYL(2,X,Y)
```

To draw a sine curve between 0 and 180 degrees would require:

```
      DIMENSION X(65),Y(65)
      CALL GSW(0.0,-1.0,3.142,1.0)
      DO 10 I=1,65
      X(I)=3.142*FLOAT(I-1)/64.
      Y(I)=SIN(X(I))
 10   CONTINUE
      CALL GPOLYL(65,X,Y)
```

4.2 Drawing Markers

Users who draw graphs like the one above often want to mark the points on the graph that are the starting points of the lines. Just as GKS does not have a line routine, it does not have a routine for plotting a single point but does have one for plotting a whole set of points:

```
CALL GPOLYM(N,AX,AY,MTYPE)
```

The points to be marked have coordinates AX(1),AY(1) up to AX(M),AY(M). The last parameter defines the marker type as follows:

GKS DEFINED MARKERS

MTYPE	Marker
1	.
2	+
3	*
4	O
5	X

If we want to mark with a dot all the points used in generating the sine curve above, we would:

```
CALL GPOLYM(65,X,Y,1)
```

4.3 Text Output

Before outputting text, it is necessary to define the size of characters to be output. This is done by:

```
CALL GSTXSP(CUPX,CUPY,CWIX,CWIY,SPX,SPY)
```

This routine allows you to output text with the characters at any angle, the text base line at any angle and a specified spacing between the characters! That is too complicated for us at the moment so let us consider the simple case of outputting characters which are 2 units in height (world coordinates), 3 units wide and no additional space between the characters:

```
CALL GSTXSP(0.0, 2.0, 3.0, 0.0, 3.0, 0.0)
```

To increase the spacing between characters, just change the second 3.0 parameter to 3.2 say. For the time being, keep the first, fourth, and sixth parameters as zero. You can play with them later on to get your more exotic text.

A string of text is output in GKS by:

```
CALL GTX(X,Y,N,CHARS)
```

The point (X,Y) is the bottom left hand corner of the first character. The parameter N defines the number of characters to be output. The characters are stored in the integer array CHARS from CHARS(1) onwards. For example:

```
INTEGER STRING(2)
DATA STRING/'STAR','LINK'/
 . . .
CALL GTX(2.0,2.0,8,STRING)
```

This will output the text STARLINK with the lower left corner of S at the point (2.0,2.0).

Note that the DATA statement here assumes 4 characters per word, as on the VAX. Unfortunately FORTRAN 77 character variables are not acceptable at esent. However:

```
CALL GTX(2.0,2.0,8,'STARLINK')
```

is interpreted correctly.

With many devices, you may find that after you have carefully chosen a size, spacing etc., the GTX routine ignores it all. This is because text on a workstation has a 'precision' associated with it. The default precision is zero, meaning text is produced as efficiently as possible (e.g. using workstation hardware if available) without caring too much about size. To guarantee implementing size and spacing correctly, you need to specify precision 2:

```
CALL GSTXRP(4,1,1,2)
```

The first argument is the workstation identifier, and the last is the precision. All subsequent output will be correct. The other arguments, which allow you to define a set of fonts and precisions, are not dealt with the Primer and can be left at 1.

4.4 Area Filling

If you happen to be working on certain raster displays or your plotter workstation has a sophisticated handler, it is possible to fill in areas. The area to be filled is defined by:

```
CALL GFILAR(N,LPX,LPY)
```

The parameters are similar to GPOLYL with LPX(1),LPY(1) to LPX(N),LPY(N) defining a set of points which form the boundary of the area. The boundary is closed by having LPX(N),LPY(N) connected to LPX(1),LPY(1). For example:

```
      DO 10 I=1,32
      THETA=FLOAT(I)*3.142/32.
      X(I)=R*SIN(THETA)
      Y(I)=R*COS(THETA)
 10   CONTINUE
      CALL GFILAR(32,X,Y)
```

This will generate a filled in regular polygon with radius R and centred at the origin.

If the fill area facility is not available on a workstation, only the boundary is drawn.

5. DRAWING ATTRIBUTES

So far we have assumed that all the line drawing is solid lines on a standard monochrome screen. However, you may be working on a plotter which allows you to use different colour pens, or a display with a variety of line styles and thickness, or even a raster display with a full range of colour possibilities.

In GKS, the method of catering for these differences is for each workstation to have a set of pens which are all differentiable. Some pens are predefined so do not have to be set up by the user. At least Pen Number 1 is guaranteed to exist on each workstation. Pen Number 1 is the standard default and is predefined to produce solid lines. Pen Number 2, on the other hand, may be different for different workstations. On a plotter, it may cause a red pen to be used while on a refresh display you might get brighter or dashed lines. Each installation will have its own set of predefined pens for each workstation.

If the predefined set is not suitable or sufficient, there is a routine "SET PEN REPRESENTATION" (described in the reference document) which can be used to alter the pen definitions.

If you just want to differentiate one piece of output from another, this can be done by selecting a different pen:

```
      CALL GPOLYL(N,XA,YA)
      CALL GSPN(2)
      CALL GPOLYL(M,XB,YB)
```

The first polyline will be output using the standard pen while the second polyline and all subsequent output until another call of GSPN will use Pen Number 2.

6. SEGMENTATION

In any complex graphical output on an interactive device, it is often useful to divide the output into specific entities or 'segments'. If you want to place an object on a display and manipulate it, GKS will allow this without the application program having to regenerate the output. For example:

```
CALL GCRSG(7)
CALL GPOLYL(N,XA,YA)
CALL GPOLYL(M,XB,YB)
CALL GCLSG
```

This will output the two polylines to the workstation but will also store them in a segment numbered 7. The first routine will create the segment while the last routine closes it.

Suppose we now want to transform the segment in some way. Mathematically, this is best done by defining the transformation matrix which we will store in ARMAT(6). However, it is not very easy to work out what the elements of the matrix should be for particular rotations and translations. Consequently, a routine is defined for setting up a transformation matrix:

```
CALL GSTRM(XO,YO,DX,DY,PHI,FX,FY,SW,ARMAT)
```

All the parameters are scalar real variables apart from SW which is LOGICAL and which is a real array. We will always set SW to .TRUE. to indicate that we are going to specify the transformation in world coordinates.

If you want to rotate the segment, you need to define the origin (XO,YO) of the rotation and the angle in radians (anti clockwise), PHI. If you want to scale it about a point (XO,YO), you need to specify the zoom factors in the X and Y directions, FX and FY. If you want to move the segment from its current position, you need to specify the displacement (DX,DY). For example:

```
CALL GSTRM(0.0,0.0, 0.0,0.0, 1.57, 1.0,1.0, .TRUE.,ARMAT)
```

This sets up a transformation matrix which can rotate the segment by about 90 degrees around the origin.

```
CALL GSTRM(0.0,0.0, 3.0,4.0, 0.0, 1.0,1.0, .TRUE.,ARMAT)
```

This can move the segment by 3 units in the X direction and 4 in the Y direction.

```
      CALL GSTRM(0.0,0.0, 0.0,0.0, 0.0, 2.0,2.0, .TRUE.,ARMAT)
```

This can double the size of the segment in both the X and Y directions.

Once the transformation matrix has been defined, it can be applied to a segment by:

```
      CALL GTRSG(7,ARMAT)
```

The first parameter is the number of the segment and the second is the transformation matrix. It will normally be necessary to make a call to GRSGWK to tell the system to draw all the segments:

```
      CALL GRSGWK(4)
```

where the argument is the workstation identifier. To rotate the segment continually about the origin:

```
      REAL ARMAT(6)
       . . .
      DO 10 I=1,1000
      PHI=3.142*FLOAT(I)/32.
      CALL GSTRM(0.0,0.0, 0.0,0.0, PHI, 1.0,1.0, .TRUE.,ARMAT)
      CALL GTRSG(7,ARMAT)
      CALL GRSGWK(4)
 10   CONTINUE
```

It is possible to insert segments within other segments so that a complex picture can be made up of a number of instances of a simple segment. We shall leave this to be described in the reference document.

When you have finished using a segment, it can be deleted by:

```
      CALL GDLSG(7)
```

This will now allow you to create an entirely different segment numbered 7.

7. NEW FRAME

To clear the screen or start a new frame on the plotter:

```
      CALL GRSGWK(4)
```

The argument is the workstation identifier. As previously mentioned, all segments defined for that workstation will be redrawn after the display screen has been cleared.

8. SUMMARY

This primer only describes about 20 of the routines in Level 3 of GKS. Consequently, it only touches on the complete set of facilities available in GKS. Even those routines described have not had their full implications listed. It is hoped that this primer has whetted the appetite of the reader to try GKS and to explore the more sophisticated facilities described in the reference document.

9. USE ON THE VAX

9.1 Linking a GKS program

GKS is in more than one library and some BLOCK DATA must be included. This is all listed in the link options file GKSLINK so that the required LINK command is simply:

```
$LINK GKSTEST,GKSLINK/OPT
```

This includes all the drivers. The library is HUGE and executable images of programs using it are at least 350 blocks. Attempts are being made to make the GKS library sharable in order to reduce the size of users' images.

9.2 Running

Before running a program, it is necessary to:

```
ASSIGN GKSWDT FOR030
```

This allows the package to read the workstation descriptions.

Error messages will be written to the stream defined in GOPKS. Errors can be made to come to the terminal by an appropriate assignment:

```
ASSIGN TT: FOR022
```

if Fortran unit 22 is specified in GOPKS. No other indication of error will be given and GKS executes subsequent routines as best it can.

9.3 Workstations on the VAX

The possible workstations are:

1 ARGS
2 Tektronix 4010
3 Sigma GOC

The number is the 3rd parameter of GOPWK (section 1). The value of the connection identifier (2nd parameter of GOPWK) is currently not used for any of these devices so can always be zero.

Other FORTRAN units in the range 31-34 are liable to be used by GKS.

ARGS, T4010 and GOC backends exist in a primitive form - lines at least can be drawn. Text at precision 2 is available.

Workstation type 100 is metafile storage, and 300 is segment storage.

An example of a GKS program, which has been tested on the VAX with the current devices, is given at the end of this document.

SUMMARY OF GKS FUNCTIONS

Control Functions

GOPKS(IEFILE)
Open GKS and set stream IEFILE for error messages.

GCLKS
Close GKS

GOPWK(IWSID,ICONID,IWSSQ)
Open Workstation number IWSID attached to Fortran unit ICONID. The workstation type is given by IWSSQ.

GCLWK(IWSID)
Close Workstation IWSID. No more output to go to this workstation.

GACWK(IWSID)
Activate Workstation IWSID ready for output.

DAWK(IWSID)
Deactivate workstation IWSID and clear screen. Output will not go to workstation until it is activated again.

Coordinates

GSW(WXMIN,WYMIN,WXMAX,WYMAX)
The world (user) coordinates from WXMIN to WXMAX in the X direction and from WYMIN to WYMAX in the Y direction are mapped on to the largest square area available on the output device.

Output Primitives

GPOLYL(N,ARX,ARY)
The set of lines joining the points ARX(1),ARY(1) through to ARX(N),ARY(N) are output.

GPOLYM(N,ARX,ARY,MTYPE)
Markers of type MTYPE are output at the set of points ARX(1),ARY(1) to ARX(N),ARY(N).

GTX(X,Y,N,ICHARS)
A text string of N characters stored in ICHARS(1) onwards is output at X,Y.

GSTXRP(IWSID,1,1,IPREC)
Defines default text output on workstation IWSID to have precision IPREC. IPREC=2 will give proper spacing and character sizes.

GSTXSP(0.0,HGT,WDTH,0.0,SPACE,0.0)

Defines text with height HGT and width WDTH. The space between the start point of neighbouring characters is SPACE.

GFILAR(N,ARX,ARY)
The area defined by the points ARX(1),ARY(1) to ARX(N),ARY(N) is filled in. The boundary is completed by connecting ARX(N),ARY(N) to ARX(1),ARY(1).

GSPN(IPEN)
Selects pen number IPEN for subsequent output.

Segmentation

GCRSG(ISGN)
Creates segment numbered ISGN.

GCLSG
Closes the segment currently being created.

GSTRM(XO,YO,DX,DY,PHI,FX,FY,.TRUE.,ARMAT)
Defines a transformation matrix ARMAT(1) to ARMAT(6) which has origin XO,YO, scaling by FX,FY, rotation by PHI radians anti clockwise, and a translation of DX,DY.

GTRSG(ISGN,ARMAT)
Outputs segment ISGN transformed by matrix ARMAT.

GDLSG(ISGN)
Deletes segment ISGN

New Frame

GRSGWK(IWSID)
Updates the display on workstation IWSID so that it is cleared and only the currently stored segments redrawn.

EXAMPLE

The following is a GKS program which has been tested on the VAX with the current device drivers.

```
C...  GKS TEST PROGRAM TO ROTATE THE TEXT 'STARLINK'
C...  IN A BOX AS THE RADIUS OF AN ELLIPSE
C
      PARAMETER WKSID=1
      PARAMETER ARGS=0
      INTEGER WKSTYP
C
      REAL BOUNDX(5),BOUNDY(5)
      DATA BOUNDX/-6.0, 9.9,9.9,-6.0,-6.0/
      DATA BOUNDY/ -0.5, -0.5, 0.5,  0.5, -0.5/
      REAL AXIS1X(2),AXIS1Y(2),AXIS2X(2),AXIS2Y(2)
      DATA AXIS1X/-6.0,-6.0/
      DATA AXIS1Y/10.0,-10.0/
      DATA AXIS2X/-10.0,10.0/
      DATA AXIS2Y/0.0,0.0/
C
      REAL TRANSF(6)
      INTEGER STRING(2)
      DATA STRING/'STAR','LINK'/
C
C
C...  DEFINE OUTPUT STREAM FOR ERRORS
      CALL GOPKS(10)
C
C...  WHICH OUTPUT DEVICE
      PRINT *, 'WK STATION (1 ARGS,2 4010,3 GOC)'
      READ *,WKSTYP
C
C...  OPEN AND ACTIVATE WORKSTATION
      CALL GOPWK(WKSID,ARGS,WKSTYP)
      CALL GACWK(WKSID)
C
C...  DEFINE VIEWPORT
      CALL GSW(-10.0,-10.0,10.0,10.0)
C
C...  SET TEXT REPRESENTATION - PRECISION 2 (LAST ARGUMENT)
      CALL GSTXRP(WKSID,1,1,2)
C
C...  DEFINE PARAMETERS FOR TEXT OUTPUT
      CALL GSTXSP(0.0,0.8,0.8,0.0,2.0,0.0)
```

```
C
C...  OPEN SEGMENT 7
      CALL GCRSG(7)
C
C...  TEXT  AT -5.5, 0   AND BOX ROUND IT
      CALL GPOLYL(5,BOUNDX,BOUNDY)
      CALL GTX(-5.5,-0.4,8,STRING)
C
C...  CLOSE SEGMENT 7
      CALL GCLSG
C
C...  LOOP TO ROTATE STRING BY ROTATING SEGMENT
C
      R=1.66667
      DO 20 I=1,128
      PHI=3.142*FLOAT(I)/32.0
      XZOOM=(R-1)/(R-COS(PHI))
C
C...  APPLY TRANSFORMATION TO SEGMENT 7
      CALL GSTRM(-6.0,0.0,0.0,0.0,PHI,XZOOM,1.0,.TRUE.,TRANSF)
      CALL GTRSG(7,TRANSF)
C
C...  REDRAW SEGMENTS
      CALL GRSGWK(WKSID)
C
C...  SELECT PEN 5 (FAINT LINE ON ARGS) AND DRAW AXES
      CALL GSPN(5)
      CALL GPOLYL(2,AXIS1X,AXIS1Y)
      CALL GPOLYL(2,AXIS2X,AXIS2Y)
C
 20   CONTINUE
C
C...  CLOSE EVERYTHING
      CALL GDAWK(WKSID)
      CALL GCLWK(WKSID)
      CALL GCLKS
      STOP
      END
```