**SUN/85.6**

Starlink Project
Starlink User Note 85.6

P T Wallace & D L Terrett

10 July 1995

---

# SGS — Simple Graphics System
# v1.1
# Programmer's Manual

---

## Abstract

SGS is a set of subroutines for performing low-level graphics input/output. SGS is implemented above the GKS package (described in SUN/83 the RAL GKS Guide and the ISO document GKS 7.4) which is a device independent graphics system designed to be the kernel of a wide variety of graphics systems. GKS, which is very comprehensive, does not itself set out to provide the most convenient interfaces for all applications, and SGS allows easy access to many of its more straightforward features.

# Contents

# List of Figures

# 1 INTRODUCTION

SGS is a set of subroutines for performing low-level graphics input/output. SGS is implemented above the GKS package (described in SUN/83 the RAL GKS Guide and the ISO document GKS 7.4) which is a device independent graphics system designed to be the kernel of a wide variety of graphics systems. GKS, which is very comprehensive, does not itself set out to provide the most convenient interfaces for all applications, and SGS allows easy access to many of its more straightforward features.

SGS, while preserving GKS concepts and indeed allowing direct GKS calls to be interspersed with SGS calls, is optimised for convenience in simple cases. Many of its features are low level (for example: open, draw a line, draw a text string, close), but there are some routines of a slightly higher level (drawing arcs and formatted numbers etc). SGS does not include routines for high level operations like drawing annotated axes or complete graphs.

Many plotting programs will be written entirely with SGS or higher level calls. There will, however, be occasions when GKS routines are used as well, usually because a specialised feature of GKS is needed. Therefore this user manual uses GKS concepts to describe SGS, and Appendix C describes SGS routines in terms of their effect on GKS. This will allow programmers to mix GKS and SGS calls in safety. The unadventurous user may safely ignore the appendices.

Throughout this manual the normal FORTRAN variable naming conventions are adhered to: i.e. all integers start with the letters I–N, all reals with letters A–H and O–Z. Character and logical variables have their type explicitly referenced in the text.

# 2 SUMMARY OF SGS CALLS

### Control

SGS_OPEN(WKSTN, IZONID, ISTAT)
: Open SGS & GKS, opening and activating one workstation.

SGS_INIT(LUN, ISTAT)
: Open SGS & GKS, without opening a workstation.

SGS_CLOSE
: Shut down SGS & GKS.

SGS_OPNWK(WKSTN, IZONID, ISTAT)
: Open a new workstation and select it.

SGS_CLSWK(IZONID, ISTAT)
: Close a workstation.

SGS_WLIST(LUN)
: Output a list of available workstations.

SGS_SPEN(NPEN)
: Select pen for line and text drawing.

SGS_IPEN(NPEN)
Inquire current pen number.

SGS_FLUSH
Complete all pending output.

SGS_ISLER(BLKER)
Inquire whether device has selective erase capability.

SGS_CLRFG(ISTATE)
Set or clear 'clear display on open' flag.

## Zones

SGS_ZONE(X1, X2, Y1, Y2, IZONID, ISTAT)
Create and select a zone of the specified extent.

SGS_ZSHAP(AR, POS, IZONID, ISTAT)
Create and select a zone of the specified aspect ratio.

SGS_ZSIZE(XM, YM, POS, IZONID, ISTAT)
Create and select a zone of the specified size.

SGS_ZPART(NX, NY, IZONES, ISTAT)
Partition current zone into NX by NY pieces.

SGS_SW(X1, X2, Y1, Y2, ISTAT)
Set window of current zone to given bounds.

SGS_SELZ(IZONID, ISTAT)
Select another zone.

SGS_RELZ(IZONID)
Release the specified zone.

SGS_CLRZ
Clear the current zone (even if this means clearing the whole display surface).

SGS_ICURZ(IZONID)
Inquire ID of current zone.

SGS_IZONE(X1, X2, Y1, Y2, XM, YM)
Inquire window bounds for the current zone, and its size on the display surface.

SGS_IDUN(DXW,DYW)
Inquire the plotting resolution of the current zone.

SGS_TPZ(IZIN, XIN, YIN, IZOUT, XOUT, YOUT, ISTAT)
Transform position from one zone to another.

SGS_BZNDC(X1, X2, Y1, Y2, POS, ISTAT)
Set the NDC extent of a base zone.

## Plotting Lines

SGS_BPOLY(X, Y)
: Begin a new polyline.

SGS_APOLY(X, Y)
: Append a new line to the polyline.

SGS_OPOLY
: Output the polyline.

SGS_IPLXY(X, Y)
: Inquire end of current polyline.

SGS_LINE(X1, Y1, X2, Y2)
: Begin a new polyline with a single line.

SGS_BOX(X1, X2, Y1, Y2)
: Draw a rectangle.

SGS_CIRCL(X, Y, R)
: Draw a circle.

SGS_ARC(X, Y, R, THETA1, THETA2)
: Draw an arc of a circle.

SGS_CLRBL(X1, X2, Y1, Y2)
: Clear a rectangular area (if this can be done without affecting the rest of the display surface).

## Plotting Text

SGS_BTEXT(X, Y)
: Begin a new text string.

SGS_ATEXT(STRING)
: Append text to the current string.

SGS_OTEXT
: Output the text string.

SGS_ATXI(I, NFI)
: Format an integer onto the current text string.

SGS_ATXR(R, NFI, NDP)
: Format a real number onto the current text string.

SGS_ATXL(STRING)
: Append text to the current text string omitting trailing blanks.

SGS_ATXB(STRING, NSPACE)
: Append right justified field to the current text string.

SGS_TX(X, Y, STRING)
: Begin a new text string with a string.

SGS_TXI(X, Y, I, NFI)
Begin a new text string with a formatted integer.

SGS_TXR(X, Y, R, NFI, NDP)
Begin a new text string with a formatted real number.

SGS_ITXB(X, Y, N, DX, DY)
Inquire status of current text string.

### Plotting Markers

SGS_MARK(X, Y, MTYPE)
Draw a single marker.

SGS_MARKL(MTYPE)
Draw marker at current end of polyline.

### Attributes of Characters

SGS_SFONT(NF)
Select text font.

SGS_SPREC(NPR)
Specify text precision.

SGS_SHTX(HT)
Specify character height.

SGS_SARTX(AR)
Specify text aspect ratio.

SGS_SUPTX(XU, YU)
Specify character orientation.

SGS_SSPTX(SP)
Specify character spacing.

SGS_STXJ(TXJ)
Specify text alignment.

SGS_ITXA(NF, NPR, HT, AR, XU, YU, SP, TXJ)
Inquire text attributes.

### Input

SGS_CUVIS(VIS)
Set visibility of cursor.

SGS_DEFCH(CHOSTR)
Define valid choice keys.

SGS_DISCU
: Disable sample mode for cursor.

SGS_ENSCU
: Enable sample mode for cursor.

SGS_ICUAV(AVAIL)
: Inquire cursor availability.

SGS_INCHO(NCHDEV, N)
: Inquire number of choices.

SGS_REQCH(N)
: Request choice.

SGS_REQCU(X, Y, N)
: Request cursor position.

SGS_SAMCU(X, Y)
: Sample cursor position.

SGS_SELCH(NCHDEV)
: Select choice device.

SGS_SETCU(X, Y)
: Set cursor position.

**GKS Inquiries** (described in Appendix B)

SGS_ICURW(IWKID)
: Inquire workstation ID for the current zone.

SGS_WIDEN(WKSTN, IWTYPE, ICONID, ISTAT)
: Translate workstation name to GKS type and connection ID.

SGS_WNAME(ROUTIN, I, ISTAT)
: Get list of workstation names.

# 3 CONTROL FUNCTIONS

## 3.1 Opening and closing SGS & GKS

*If you are writing an ADAM application you should use the routines described in the ADAM Graphics Programmer's Guide (SUN/113) to open and close workstations and not the routines described here.*

Most programs that are to perform graphics I/O via SGS begin by calling SGS_OPEN and end by calling SGS_CLOSE:

```
CALL SGS_OPEN (WKSTN, IZONID, ISTAT)
         :
         :
CALL SGS_CLOSE
```

The arguments of the OPEN call are as follows. WKSTN is the workstation name, a character string specifying the required graphics workstation. IZONID receives from SGS the 'zone identifier', a number allocated by SGS which we may need later to refer back to this workstation. ISTAT is the status, which will be zero if the OPEN has succeeded.

Friendly *workstation names* (e.g. 'PLOTTER') can be given to graphics workstations, either by the user or by the system manager, and then specified in the SGS_OPEN call. More details can be found in Appendix A; consult your system manager about local conventions. Alternatively, the two numbers needed to specify the workstation—the workstation type (what sort of device) and the connection identifier (which one)—can be given directly, as a character string consisting of 2 numbers separated by a comma. A list of available workstation types can be obtained by running the program `/star/bin/examples/sgs_workstations`[1]. If the program just prints a message stating that the database could not be opened you should inform your system manager.

Broadly speaking, the *connection identifier* specifies which of several workstations of the nominated type is to be opened. Exactly how connection IDs are mapped onto real devices depends on the GKS implementation being used. For example, in RAL GKS, for devices that are terminals, 5 will be the terminal which you have logged on to. For the metafile workstations the connection identifier is the Fortran unit to which the output will be sent for temporary storage on disk.

As an example, if we are working on a Tektronix 4010 terminal, the mandatory open call might be:

```
CALL SGS_OPEN ('201,5', ITEK, ISTAT)
```

If, instead, we wish to create a metafile on disk we could use:

```
CALL SGS_OPEN ('50,9', IMETA, ISTAT)
```

The purpose of the *zone identifier* will become clearer later on; suffice it to say at this stage that this number (which is allocated by SGS and cannot be set or otherwise processed by the user's program) allows the program to refer back to the full display surface of the workstation that has been opened. A zone set up by SGS_OPEN or SGS_OPNWK is called a 'base zone'.

Should any error messages be output by the SGS or GKS routines, they will be reported through the Starlink error reporting system (EMS - see SUN/104)

A formatted table of the workstation names known to SGS can be output by calling SGS_WLIST:

```
CALL SGS_WLIST (LUN)
```

where LUN is the FORTRAN logical unit on which the table should be written. Each name is accompanied by a one-line description of the workstation.

If you want the table to appear on the user's terminal, and you have not set up your own logical units, specify a unit number of 6, which is by default directed to the terminal. For example:

[1] if you are using this software on a non-Starlink machines you mave to replace `/star` by some other path name.

```
CALL SGS_WLIST (6)
```

If you want to perform your own formatting of the list, or manipulate the names in some other way, this can be done with SGS_WNAME. See section 10 for more information. SGS_WLIST and SGS_WNAME are the only SGS routines that can be called before SGS has been opened.

The appearance of a name in the list does not guarantee that the workstation can be opened (it could, for example, be in use by someone else), nor does it mean that no other valid names exist. However, it does provide a mechanism for programs to help users with workstation names which is always up-to-date, even when programs are copied from one computer to another. If the graphics devices on a system change, it only requires the system manager to edit a table, and all SGS programs not only have access to new devices but can also reflect the changed situation in their help information.

> *Programs should not make assumptions about the properties of workstations given the workstation name or type, both of which are subject to change. If special features of a device are to be exploited, their existence must be established via the appropriate SGS and GKS inquiry calls.*

### 3.2 Multiple workstations

It is a central philosophy of SGS that only one workstation should be *active* at once. However, multiple workstations may be *open* at once, and plotting may switch between them. Successive workstations may be opened with calls to the routine SGS_OPNWK:

```
CALL SGS_OPNWK (WKSTN, IZONID, ISTAT)
```

WKSTN is the workstation name, IZONID receives the zone identifier for the base zone (see section 4) and ISTAT is as before.

On each occasion that SGS_OPNWK is called, previously active workstations are deactivated before plotting switches to the *new* workstation. Thus, if the program is to plot first on a VDU then later on a plotter, it is best to open the plotter in the SGS_OPEN call and then to use SGS_OPNWK to open the VDU. Plotting on the latter can then begin.

Multiple SGS_OPEN calls to the same workstation are permitted; on each occasion a new base zone is created.

Individual workstations can be closed by means of:

```
CALL SGS_CLSWK (IZONID, ISTAT)
```

where IZONID must be a base zone; if there are other base zones on the workstation the specified zone is released but the workstation is not closed.

Switching between multiple workstations is accomplished by the ‘select zone’ routine which will be described in the next section.

If we want more flexible control—for example the ability to have several workstations active at once—this is available directly through the GKS activate/deactivate routines:

```
CALL GACWK(IWKID)
      :
CALL GDAWK(IWKID)
```

A final call to SGS_CLOSE will always leave the system correctly terminated.

### 3.3 Pens

It is sometimes necessary to distinguish between different parts of a graph. There are a variety of ways this might be achieved, depending on the characteristics of the particular graphics device concerned: for example by plotting in different colours, or by using dotted lines, or by varying the line width. Individual control over all of these options is possible within GKS; the SGS package offers instead a simplified facility where one of a small number of clearly distinguishable line qualities may be selected, each specified by an SGS 'pen number'. The call is:

```
CALL SGS_SPEN (NPEN)
```

where NPEN is the number of the SGS pen—an integer greater than 0.

SGS pen number 1 is the default, and is the 'normal' pen for the device—a solid white line on a VDU, for example, or a black line of ordinary width on a plotter. SGS pens 1 to 5 may be assumed to be available and will correspond—depending on the device—to different colours or different styles of dashed line. Pen numbers greater than 5 may be available depending on the device. New pens may be defined or existing ones redefined with the GKS set polyline representation function (GSPLR) but should only be done immediately after opening the workstation to avoid causing implicit regeneration of the picture.

Appendix B should be consulted before using GKS facilities directly.

The last requested SGS pen number can be obtained by means of:

```
CALL SGS_IPEN (NPEN)
```

### 3.4 Buffering

For efficiency, both SGS and GKS save up output in memory and send it to the graphics device in batches. Occasionally this may not be appropriate. For example an application program may plot a graph and then ask the user if the plot is satisfactory; it is clearly important that the whole plot has been completed before user input is solicited. In such cases the call:

```
CALL SGS_FLUSH
```

causes any pending output to be sent immediately (to all active workstations). Unnecessary calls to SGS_FLUSH will have no significant ill effects (other than defeating any benefits from the buffering of output)[2].

It is particularly important that SGS_FLUSH be called before any sequence of direct calls to GKS routines. For the same reason SGS_CLOSE must be called before your program exits, otherwise the plot may be incomplete or even absent.

[2]Except when plotting dashed or dotted lines as SGS_FLUSH will cause the dash pattern to be restarted; if the line is being plotted with points very close together and SGS_FLUSH is called after each point the line may appear to be solid.

### 3.5 Special control functions

In some sophisticated applications it may be useful or even necessary to separate the opening of SGS/GKS, and the opening of the first workstation, for example in order to make inquiries about the capabilities of a workstation before opening it. SGS/GKS can be opened with:

```
CALL SGS_INIT (LUN, ISTAT)
```

where ISTAT is as for SGS_OPEN. LUN is the logical unit passed to GOPKS but this is not used by the GKS implementation distributed by Starlink. GKS is then open and GKS functions may be called (at least one workstation must be opened and activated before any plotting can be done of course). Workstations can then be opened with calls to SGS_OPNWK in the normal way.

The GKS standard specifies that when a workstation is opened its display surface must be cleared. This is usually exactly what is required, but for some system architectures this is a problem as it prevents one program adding to, or interacting with, a picture drawn by another. An escape function has therefore been implemented in Starlink's version of RAL GKS to suppress the screen clear when a workstation is opened. This function is only available on interactive workstations, not on hard copy devices, and futhermore this feature will not be available in other GKS implementations. It should therefore only be used when no other solution is possible.

```
CALL SGS_CLRFG (1)
```

will suppress the clearing of the display surface when the next workstation is opened *provided that the workstation supports this feature*. If it does not, no error message is output and the display surface is cleared as usual. After SGS_OPNWK has been called, the normal GKS behaviour will have been restored (because SGS_OPNWK explicitly resets the flag), and so SGS_CLRFG has to be called again each time a new workstation is opened. This is not true if GOPWK is used to open the workstation, in which case a return to normal behaviour requires:

```
CALL SGS_CLRFG (0)
```

## 4 ZONES

An SGS *zone* is a block of display surface on a particular workstation which behaves in many respects as if it were an independent plotting device.

A zone has an extent (in world coordinates), a position and size on the display surface (or, more correctly, in normalised device coordinates—see Appendix B) and a workstation. Clipping occurs at the zone boundary (unless disabled via GKS).

There are routines to create new zones and to switch plotting from the current one to another. When a zone is created it is allocated a *zone identifier* by SGS which is thereafter used by the program to refer to that zone. When a workstation is opened (in either SGS_OPEN or SGS_OPNWK), a 'base' zone is created which occupies the whole display surface. This base zone has a world coordinate extent such that the square (0,0) to (1,1) spans the zone in at least one coordinate and appears on the display surface as a square; the bottom left hand corner is

(0,0). A similar default coordinate system is set up whenever a zone is created. It is possible to have more than one base zone on a given workstation.

One important use of the zone facility is to enable the allocation of plotting resources to be separated from the plotting itself. Thus the user might decide that the three graphs he is about to plot should, on one occasion, go to three regions of his VDU screen, and on another occasion to two regions of his screen and to a pen plotter respectively; by using zones the plotting program can operate as if the three plots were always on separate workstations, and switch freely between them during plotting. Activation and deactivation of workstations as well as control of the various windows and viewports is handled automatically by SGS (see Appendix B).

Another occasion on which SGS zones can be useful is when it is convenient to operate in several world coordinate systems within one graph. For example, a program might create one zone for plotting axes and annotation, and create one or more smaller zones within that zone for plotting data in world coordinates.

There is always a *current zone*, within which plotting occurs. Zones are created in terms of the current zone, with no restriction on the number of levels deep. Overlapping zones are permissible, but a new zone must lie entirely within the current one. There is a maximum number of zones that can exist at any one time, and a routine is available for dispensing with a zone once it is no longer required. There is no hierarchy of zones; once a zone has been created within the current zone and then made current, the parent zone can be released.

### 4.1 Zone creation

In all the create zone routines the new zone is automatically 'selected', so that further plotting is to the new zone. The IZONID argument is the zone identifier, supplied by SGS. ISTAT is the status return, which will be zero if the routine succeeds. The world coordinate system that the new zone is born with is such that (0,0) is at the bottom left hand corner, the top right corner of the zone has one coordinate unity and the other unity or more, and a square in world coordinates appears square on the display surface.

A zone can be created within a nominated region of the current zone by means of:

```
CALL SGS_ZONE (X1, X2, Y1, Y2, IZONID, ISTAT)
```

X1, X2, Y1, Y2 are the bounds of the new zone in the world coordinate system of the current zone. Note that the new zone will have the default coordinate system, which will, in general, be different from that of the parent. To create a new zone the coordinates of which match those of the parent, follow the SGS_ZONE call with a call to SGS_SW and specify the same X1, X2, Y1, Y2 in each case. SGS_SW is described below.

A zone of nominated shape (on the display surface) can be fitted into the current zone as follows:

```
CALL SGS_ZSHAP (AR, POS, IZONID, ISTAT)
```

AR is the aspect ratio x/y. Thus a normal TV raster has an AR of about 1.33. The new zone has this aspect ratio, and fills the parent zone in at least one dimension. POS is a character string specifying the position of the new zone within the parent zone. The first character is 'B', 'C' or 'T', to position the new zone at the bottom of the parent zone, centred vertically, or at the top. The second character is 'L', 'C' or 'R', for left, centre or right respectively. Thus a POS value of

'BL' would set up the new zone at the bottom left of the parent zone; 'CC' would centre the new zone in the parent.

For a zone of nominated absolute size:

```
CALL SGS_ZSIZE (XM, YM, POS, IZONID, ISTAT)
```

XM,YM are the dimensions of the new zone in metres. POS is as for SGS_ZSHAP.

The current zone can be partitioned into a number of equally sized pieces by:

```
CALL SGS_ZPART (NX, NY, IZONES, ISTAT)
```

where NX is the number of new zones horizontally and NY is the number of new zones vertically. A total of NX x NY new zones will be created and the zone identifiers returned in the array IZONES, starting with the bottom left hand corner zone in the first element, the second zone in the bottom row in the next element, and so on. The last element of the array will contain the zone identifier of the top right had corner zone. Unlike the other zone creation routines, SGS_ZPART leaves the current zone unchanged, and to plot in one of the new zones it must be explicitly selected (see section 4.2).

Once a zone has been created (by SGS_OPEN, SGS_OPNWK or one of the above routines), it can be given a suitable coordinate system as follows:

```
CALL SGS_SW (X1, X2, Y1, Y2, ISTAT)
```

X1, X2, Y1, Y2 are the bounds of the current zone in the new world coordinate system. Note that it is a fundamental property of GKS that the world coordinate x and y values always increase from left to right and from bottom to top respectively.

## 4.2 Zone selection and release

To start plotting on a new zone:

```
CALL SGS_SELZ (IZONID, ISTAT)
```

Plotting is suspended on the current zone and the new zone, specified by the zone identifier IZONID, is made the current one. Subsequent plotting goes to the new zone. The selection may imply a change of workstation; this is handled by SGS.

Once a zone is no longer required it is wise to release it:

```
CALL SGS_RELZ (IZONID)
```

This liberates workspace within SGS, thereby making room for one more zone later on. Neither the current zone nor a base zone can be released, but if this is attempted no error message is produced.

### 4.3 Clearing zones

The current zone may be cleared by:

```
CALL SGS_CLRZ
```

This will ensure that the area covered by the zone is empty. On some devices only the area covered by the zone will be cleared but on others the whole display surface may be cleared (because the hardware is not capable of clearing selected areas). On a hardcopy device a call to SGS_CLRZ will cause the paper to be advanced to a new frame except when the whole plotting surface is already empty. If any part of the surface has has been plotted on, even if the current zone is empty, a new frame will be started. This behaviour should be contrasted with that of SGS_CLRBL (see section 5.4) which never causes a frame advance. Whether SGS_CLRZ will clear the whole display surface can be inquired by:

```
CALL SGS_ISLER (BLKER)
```

If BLKER (LOGICAL) is .TRUE. then the workstation is capable of clearing selected areas and the whole display surface will not be cleared (unless the current zone fills the entire area). The whole display surface can always be cleared by clearing the base zone.

### 4.4 Inquiries

The zone identifier for the current zone is available via:

```
CALL SGS_ICURZ (IZONID)
```

To find out the size of the current zone:

```
CALL SGS_IZONE (X1, X2, Y1, Y2, XM, YM)
```

X1, X2, Y1, Y2 are the world coordinate bounds. XM, YM are the dimensions in metres.

When two zones overlap, the same point on the display surface corresponds to different positions in world coordinates in the two zones. A position in one zone can be converted to the corresponding position in another by:

```
CALL SGS_TPZ (IZIN, XIN, YIN, IZOUT, XOUT, YOUT, ISTAT)
```

where XIN, YIN is a point in the zone IZIN, and IZOUT is the ID of the zone to which the position is to be converted. The position in this zone is retured in XOUT, YOUT. ISTAT will be zero if both zone IDs are valid. Although all plotting is clipped at the boundary of the current zone, the world coordinates are still meaningful outside the zone, and so the point to be converted can be outside one or both zones and the conversion will still succeed.

To optimise plotting efficiency it is sometimes necessary for a program to find out what the plotting resolution is. The inquiry:

```
CALL SGS_IDUN (DXW, DYW)}
```

returns the plotting resolution of the current zone, in world coordinates, in x and y separately.

### 4.5 Controlling NDC

This section is only of interest if you are mixing SGS calls with use of other high level graphics packages in the same program.

Some high level packages make make assumptions about what area of Normalized Device Coordinates (NDC—see appendix B) is visible on the device, for example that the whole of the NDC unit square is visible. When SGS opens a workstation it arranges that the whole display surface is available for plotting and on a workstation whose display surface is not square this inevitably results in part of the NDC unit square not being visible.

The routine SGS_BZNDC allows the area of NDC occupied by a base zone to be manipulated.

```
CALL SGS_BZNDC (X1, X2, Y1, Y2, POS, ISTAT)
```

sets the NDC limits of the current zone, which must be a base zone, to be X1 to X2 in x and Y1 to Y2 in y. All four limits must be between 0.0 and 1.0, and X2 and Y2 must be greater than X1 and Y1 respectively. Since the modified zone will, in general, not be the same aspect ratio as the display surface, the display surface will not be filled in one of either x or y. The character argument POS specifies how the zone should be positioned (c.f. SGS_ZSHAP above). ISTAT is the usual status argument.

In order to achieve its effect SGS_BZNDC has to set the workstation transformation, which would both alter the properties of any other zones on the workstation and cause the picture to require regeneration. Therefore it can only be used when the workstation is empty and when no other zones exist on the workstation. It will typically be used immediately after opening the workstation.

## 5 PLOTTING LINES

### 5.1 Polylines

Many graphics packages have line drawing routines based on a ‘current position’ concept. A single straight line is drawn either by calling first a ‘set current position’ routine then calling a ‘draw line from current position to new position’ routine, or by calling a general ‘move’ routine twice with a ‘pen up/down’ argument suitably set each time. Though familiar and convenient, this approach is prone to various subtle problems, for example where the coordinate system is changed during plotting. The GKS line drawing primitive, on the other hand, does not use a current position but instead allows a series of connected straight lines, called a *polyline*, to be plotted as one object. The SGS package supports both concepts, and includes ‘current position’ style routines, called SGS_BPOLY, SGS_APOLY and SGS_OPOLY, which construct and output GKS polylines.

The SGS_BPOLY routine is used to begin a polyline:

```
CALL SGS_BPOLY (X, Y)
```

where X,Y are the starting coordinates for the polyline. (This is the equivalent of 'move to x,y with pen up' in other plotting packages.)

The polyline is then built up by calls to SGS_APOLY, each of which appends a single line from the current end of the polyline to the new X, Y:

```
CALL SGS_APOLY (X, Y)
```

(This is the equivalent of 'move to x,y with pen down'.)

Finally, when the polyline is complete, it can be output using:

```
CALL SGS_OPOLY
```

Thus the following code would plot a triangle, with vertices at (0,0),(3,0) and (0,4):

```
CALL SGS_BPOLY (0.0, 0.0)
CALL SGS_APOLY (3.0, 0.0)
CALL SGS_APOLY (0.0, 4.0)
CALL SGS_APOLY (0.0, 0.0)
CALL SGS_OPOLY
```

In most cases the call to SGS_OPOLY may be omitted; any polyline awaiting output is automatically plotted if a new polyline is begun (via SGS_BPOLY or SGS_LINE) or at various other critical places within SGS. It is, however, good practice to issue SGS_OPOLY if there is any uncertainty—redundant calls to SGS_OPOLY are harmless. For example, at the end of a general purpose subroutine is a good place, as the subroutine might be called in a case where direct access to GKS routines occurs before any subsequent SGS line-drawing takes place:

```
      SUBROUTINE CROSS (X1, X2, Y1, Y2)
*  Draw a cross
      REAL X1, X2, Y1, Y2

      CALL SGS_BPOLY (X1, Y1)
      CALL SGS_APOLY (X2, Y2)
*  (could call SGS_OPOLY here but is unnecessary)
      CALL SGS_BPOLY (X1, Y2)
      CALL SGS_APOLY (X2, Y1)
*  (this call to OPOLY is advisable)
      CALL SGS_OPOLY

      END
```

At any time, the current end coordinates of the polyline being built can be obtained by:

```
CALL SGS_IPLXY (X, Y)
```

(Only a limited amount of space is reserved for building the polyline; however, when this space is filled the polyline is automatically plotted and a new one, starting from the last point given, is begun. Thus polylines of arbitrary length may be plotted via the SGS routines. Only if a dotted linetype are in use will the joins show, an occasion on which direct use of the GKS polyline routine might be preferable.)

When the polyline to be drawn is merely a single line, it may be most convenient to use the routine SGS_LINE, which opens a new polyline consisting of a single line:

```
CALL SGS_LINE (X1, Y1, X2, Y2)
```

The polyline begins at (X1,Y1) and ends at (X2,Y2). Output of the polyline is not forced, so the LINE routine can be used to begin a polyline of any length.

Notes:

(1) All of the SGS calls which affect the coordinate transformation, change pen, etc, automatically arrange for any pending polyline to be plotted before the change occurs.

(2) Improperly begun polylines (where no call to SGS_BPOLY or to SGS_LINE has been made) are nevertheless plotted. The starting point is either the end of the previous polyline or—immediately upon opening or after calling one of the routines which affect the coordinate transformation—the first X, Y to be appended to the new polyline. This property of SGS should not be exploited.

(3) There will be occasions when the GKS polyline routine offers a more convenient facility than the SGS routines. The call is:

```
CALL GPL(LENGTH, XARRAY, YARRAY)
```

where LENGTH is the size of the one-dimensional arrays XARRAY and YARRAY, which contain the X and Y coordinates of the vertices of the polyline, in world coordinates.

The triangle plotted in the example given earlier can be produced with the following code:

```
REAL XA(4), YA(4)
 :
DATA XA/0.0, 3.0, 0.0, 0.0/
DATA YA/0.0, 0.0, 4.0, 0.0/
 :
 :
CALL GPL(4, XA, YA)
```

## 5.2 Plotting rectangular boxes

Rectangular boxes, with sides parallel to the axes, can be plotted by means of the routine SGS_BOX:

```
CALL SGS_BOX (X1, X2, Y1, Y2)
```

The arguments are the x and y extents of the box.

## 5.3 Plotting arcs and circles

The SGS package includes facilities for outputting both complete circles and arcs of circles:

```
CALL SGS_CIRCL (X, Y, R)

CALL SGS_ARC (X, Y, R, THETA1, THETA2)
```

The arc or circle is centred on X, Y and has radius R. The start and finish angles of the arc are THETA1 and THETA2 respectively; they are expressed in radians and have the conventional zero point and direction (e.g. an arc from 0 to $\pi/2$ would begin at [X+R,Y] and end at [X,Y+R]).

The arcs and circles are, of course, plotted as such in world coordinates, and will appear distorted if the window has been set so that the window-to-display-surface scales in x and y are different.

### 5.4 Clearing rectangular areas

A rectangular area of the display surface can be cleared with:

```
CALL SGS_CLRBL (X1, X2, Y1, Y2)}
```

where the arguments are the x and y extents of the area. This routine never affects anything outside the specified area and so on some devices may not have any effect at all. It never causes a frame advance on hard copy devices.

The routine might be used, for example, to clear the area in which some text was to be plotted in order to avoid the text being obscured by existing plotting. On a device that cannot clear selected areas no clearing would take place and the end result would be the best that could be achieved without re-plotting the entire picture. If you wish to guarantee that an area is clear SGS_CLRZ should be used instead.

## 6 PLOTTING TEXT

There are several SGS routines for creating and plotting strings of characters. The most general, low level routines will be described first, although in many instances it will be more convenient to use the higher level routines SGS_TX, SGS_TXI and SGS_TXR, which allow single strings and formatted numbers to be plotted; these will be described later.

A text string can be plotted by means of the routines SGS_BTEXT, SGS_ATEXT and SGS_OTEXT. SGS_BTEXT begins a new text string:

```
CALL SGS_BTEXT (X, Y)
```

where X,Y is the 'position' of the string (more about that shortly). ATEXT appends more text onto the string:

```
CALL SGS_ATEXT (TEXT)
```

(TEXT is a character string). SGS_OTEXT outputs a completed string:

```
CALL SGS_OTEXT
```

The call to SGS_OTEXT can usually be omitted; any existing string is output automatically when SGS_BTEXT or SGS_CLOSE is called (and at various other critical places within SGS). (n.b. The maximum length of string that can be plotted is set by an internal workspace. Unreasonably long strings will suffer truncation to this maximum.)

There are several higher level append routines. The routine SGS_ATXL appends a field to the text string after stripping any trailing blanks; thus if:

```
INITAL='S MCN   '
```

the two calls:

```
CALL SGS_ATXL (INITAL)

CALL SGS_ATEXT (INITAL(:5))
```

produce identical results.

The routine SGS_ATXB, in contrast, appends a right justified field to the text string, but replaces any leading blanks with a specified number of blanks. Thus, if:

```
NUM='    -0.035'
```

the two calls:

```
CALL SGS_ATXB (NUM,1)

CALL SGS_ATEXT (NUM(4:))
```

produce identical results.

Routines are provided for conveniently plotting numbers; SGS_ATXI formats and appends an integer, and SGS_ATXR does the same for a real.

Integers may be plotted by means of:

```
CALL SGS_ATXI (I, NFI)
```

where I is the number and NFI controls the formatting. If the number is to be left justified (for example within a message), an NFI value equal to the number of leading spaces should be specified. For example:

```
CALL SGS_ATEXT ('THERE WERE')
CALL SGS_ATXI (25, 1)
CALL SGS_ATEXT (' SAMPLES')
```

would produce the string 'THERE WERE 25 SAMPLES'. If, on the other hand, right alignment is required (if a scale were to be marked on the left of a vertical axis, for example) an NFI value of *minus* the required field width should be specified.

Similarly, real numbers can be plotted with the routine SGS_ATXR:

```
CALL SGS_ATXR (R, NFI, NDP)
```

where R is the number and NFI is as for the ATXI routine. NDP is the number of decimal places to be plotted. $-1$ (or any other negative value) causes only the integer part of the number to appear; if 0 is specified the decimal point appears as well; positive values result in the specified number of decimal places being plotted. (N.B. Both SGS_ATXI and SGS_ATXR are limited in the width of fields they can handle, and only sensible values should be used.)

Details of the text string under construction can be inquired through the routine SGS_ITXB:

```
CALL SGS_ITXB (X, Y, N, DX, DY)
```

For the current zone, the SSG_ITXB routine returns the reference position (X, Y), the number of characters in the string (N) and the string extent DX, DY is such that X+DX, Y+DY is the concatenation point for subsequent strings. In the common case where the strings are being drawn left justified, a new string drawn at the concatenation point will follow on from the old one. If, however, right justification has been specified, each successive string will appear right justified against the previous one. This may not be what was expected; for example, to draw 'aBc' where 'B' is in a different font or requires a different pen and thus causes the string to be flushed before and after, it will be necessary if using right justification to plot the 'c' first, then to change font or pen and plot the 'B', and finally to plot the 'a'. In the case of text which is centred horizontally (viewing the string in the normal orientation), concatenation is not appropriate, and zero DX and DY are returned.

When the string to be output has already been formatted, or where a single number is to be plotted, it is convenient to use the routines SGS_TX, SGS_TXI and SGS_TXR:

```
CALL SGS_TX (X, Y, STRING)
CALL SGS_TXI (X, Y, I, NFI)
CALL SGS_TXR (X, Y, R, NFI, NDP)
```

In each case X, Y is the position of the string. STRING is the character string to be output; I and NFI are the integer to be formatted and the format indicator (exactly as for the ATXI routine); R, NFI and NDP are the real number to be formatted, the format indicator, and the number of decimal places (likewise exactly as for the SGS_ATXR routine).

These routines merely open a new text string and append the requested field to it; immediate output is not implied and more fields can be appended if desired.

Notes:

(1) All of the routines which affect the coordinate transformation automatically arrange for any pending text string to be output before the change occurs. The same applies to changes in the SGS pen selection.

(2) If a text string is improperly begun without a call (explicit or implicit) to SGS_BTEXT, nothing is ever plotted for that string.

(3) A consequence of the previous two points is that any text appended after the current string has been flushed—for example by requesting a change of pen—will be lost.

# 7 TEXT ATTRIBUTES

The manner in which the text is plotted is controlled by a set of attributes, the most important of which are size, orientation and alignment. In addition, control over precision, font, aspect ratio and spacing is possible.

The size is specified by means of the routine SGS_SHTX:

```
CALL SGS_SHTX (HT)
```

The argument HT is the height of the character box, in world coordinates. The size, shape and orientation of characters are defined in world coordinates and are mapped onto the display surface in the same way as lines and markers. Therefore, if the world coordinates are altered without making a corresponding change to the character height the character size on the display surface (in mm) will change.

The orientation is specified by means of the routine SGS_SUPTX:

```
CALL SGS_SUPTX (XU, YU)
```

where XU,YU is a vector specifying the 'up' direction. The standard orientation is (0.0,1.0); for text on its side reading upwards an up vector of (-1.0,0.0) could be used; and so on. Only the direction of the vector, not its magnitude, is significant. Remember that this vector is a vector in world coordinates and that if plotting scale is different in X and Y and it is not parallel to either axis, its direction on the display surface will not be the same. Furthermore the character boxes will be transformed from rectangles into parallelograms. There is some advice on plotting text in non-uniform coordinate systems at the end of this section.

The disposition of the plotted string with respect to the reference x,y specified in the SGS_BTEXT call may be controlled using the routine SGS_STXJ:

```
CALL SGS_STXJ (TXJ)
```

TXJ is a character string of length 2. The first character of TXJ is B, C, or T, and specifies whether the reference position is to lie on the bottom, centre, or top horizontal of the string. The second character is L, C, or R, and specifies whether the reference position is to lie on the left, centre, or right vertical of the string. Thus a TXJ of 'BL' (the default) will cause all strings to be plotted with the given X,Y at the bottom left corner. If 'CC' is specified, all strings will be plotted centred on the given positions. (Notes: (1) The SGS_STXJ routine may be called at any time before the string is actually plotted and applies to all strings from then on. (2) The directions bottom, left, etc., refer to the string seen in its conventional orientation.)

The routine SGS_SPREC allows the text precision to be specified:

```
CALL SGS_SPREC (NPR)
```

A precision value of NPR=2 (the default) indicates that no avoidable discrepancy between the characters as defined by the font and as they appear on the display device is tolerable; in effect, software generated characters will be plotted which look the same on all devices. A precision value of 0 allows the use of hardware characters, which will be faster but may compromise appearance especially if more than one device is used.

Different fonts may be selected by means of the routine SGS_SFONT:

```
CALL SGS_SFONT (NF)
```

where NF is the font number. The default font (number 1) is ordinary Roman letters and Arabic numbers. The GKS User Guide lists the available fonts.

Control of the aspect ratio of the character box is available via the routine SGS_SARTX:

```
CALL SGS_SARTX (AR)
```

AR is the aspect ratio (width/height). Again the aspect ratio is in world coordinates and will need to be modified if a non-uniform coordinate system is being used.

The routine SGS_SSPTX allows the spacing between characters to be changed:

```
CALL SGS_SSPTX (SP)
```

SP is the distance between one character box and the next as a fraction of the width of the box. The default spacing is 0 which gives no extra space between characters and results in the text being of normal appearance. The spacing can be negative, which results in the characters overlapping.

The aspect ratio and spacing are defined for a 'nominal' standard character. The actual values may differ from those set for proportional fonts (the default font is proportional) and for fonts of unusual design. Do not rely on the spacing, orientation and aspect ratio to define the exact length of a text string. The GKS routine GQTXX (inquire text extent) should be used.

The routine SGS_ITXA allows all the above text attributes to be inquired:

```
CALL SGS_ITXA (NF, NPR, HT, AR, XU, YU, SP, TXJ)
```

If different plotting scales are in use in X and Y, plotting text requires considerable care. Not only must an appropriate height be used but the aspect ratio must also be adjusted and if the direction of the up vector is changed both will need to be altered to maintain the same character size. Futhermore text of normal appearance cannot be plotted with the up vector other than parallel with one of the axes. It is therefore recommended that you use a uniform coordinate system for plotting text and the SGS zone facility provides a convenient way of doing this. A new zone, covering the same area as the current one, can be created by:

```
CALL SGS_IZONE (X1, X2, Y1, Y2, XM, YM)

CALL SGS_ZONE (X1, X2, Y1, Y2, IZTXT, ISTAT)
```

The new zone will have a uniform coordinate system (c.f. section 3.1) and is therefore suitable for plotting text. If the text has to be plotted at a particular position X, Y in world coordinates (for example, to label tick marks on an axis) the corresponding position XT, YT in new zone can be obtained by:

```
CALL SGS_TPZ (IZ, X, Y, IZTX, XT, YT, ISTAT)
```

where IZ is the zone ID of the original zone.

# 8 MARKERS

A set of centred symbols, called markers, is supported. The symbol to be plotted is selected via a number called the marker type; the following set of markers is currently available:

| type | symbol |
|---|---|
| 1 | · |
| 2 | + |
| 3 | * |
| 4 | ○ |
| 5 | × |

To plot a single marker, use:

```
CALL SGS_MARK (X, Y, MTYPE)
```

where X and Y are now single values.

Markers may be plotted at intervals along a polyline by means of the routine SGS_MARKL:

```
CALL SGS_MARKL (MTYPE)
```

which plots the specified symbol at the current end of the polyline being built via SGS_APOLY. For example, the following code will plot a triangle with a star marker at corner A, an open symbol at corner B, and an X at corner C:

```
CALL SGS_BPOLY (XA, YA)
CALL SGS_MARKL (3)
CALL SGS_APOLY (XB, YB)
CALL SGS_MARKL (4)
CALL SGS_APOLY (XC, YC)
CALL SGS_MARKL (5)
CALL SGS_APOLY (XA, YA)
CALL SGS_OPOLY
```

# 9 INPUT

SGS provides two ways in which a program can interact with the person running it:

**a graphics cursor** (locator in GKS terminology) which the user can move around the screen with, for example, a joystick or trackerball, and whose position in world coordinates can be read by the program, and

**a choice device** in which the user selects one of a number of choices, usually by pressing a button on a control box or a key on a keyboard.

The most familiar form of interaction is where the cursor is displayed and the program waits for the user to move it to the desired position and press one of the choice keys. The cursor position (a position in world coordinates) and the choice selected (a positive integer) are then returned to the program.

This type of interaction is achieved by:

```
CALL SGS_REQCU (X, Y, N)
```

If this routine is called repeatedly the cursor will reappear at the position that it was left at by the previous call. The cursor's position can be explicitly set by:

```
CALL SGS_SETCU (X, Y)
```

provided that the hardware allows the position of the cursor to be controlled by the computer. Not all devices have cursors, e.g. pen plotters, and so a program can find out if the workstation of the current SGS zone has a cursor (all SGS input routines operate on the currently selected SGS zone) with:

```
CALL SGS_ICUAV (AVAIL)
```

where AVAIL is set to the logical value .TRUE. or .FALSE.. The cursor can also be used in a mode (sample mode) in which the cursor can be moved around by the user but the position is read by the program without waiting for the use to press a choice key. This mode of operation has to be explicitly enabled with:

```
CALL SGS_ENSCU
```

but may not be available on all devices or at all in some GKS implementations **(at present this includes the RAL implementation used by Starlink)** and if it is not an error message will be generated; the cursor position can then be sampled with:

```
CALL SGS_SAMCU (X, Y)
```

typically in a loop. Sample mode is disabled with:

```
CALL SGS_DISCU
```

It is sometimes desirable to use this mode without the cursor being visible, its position being indicated in some other way, such as continuously drawing a line to the cursor position from its position when last sampled. In order to allow this, the cursor can be made invisible by:

```
CALL SGS_CUVIS (.FALSE.)
```

and made visible again with:

```
CALL SGS_CUVIS (.TRUE.)
```

The cursor will, of course, only actually appear if sample mode is enabled or when SGS_REQCU is called.

The choice device, which can be invoked either in conjunction with the cursor or on its own, can be one of two devices. By default it is a choice device associated with the graphics workstation such as the buttons on a mouse or the numeric keys on a graphics terminal and is characterised by having a small number of choices. SGS provides an alternative device which is the keyboard of the terminal from which the program is being run. This has the advantage of being always available and of having a much larger number of choices. However, since it is potentially a different device from the graphics device, two separate read operations are required to get the choice and the cursor information, and on a very busy system or over a network these operations may be separated by an appreciable length of time.

The choice device you wish to use is selected by:

```
CALL SGS_SELCH (NCHDEV)
```

where NCHDEV is 1 or greater for the normal GKS choice devices and 0 for the command terminal keyboard. The number of choices on a choice device can be inquired with:

```
CALL SGS_INCHO (NCHDEV, N)
```

If the choice device does not exist, N will be returned as zero.

For the terminal keyboard (choice device 0) only, the mapping of the keyboard keys onto the integers returned by SGS_REQCH is controlled by:

```
CALL SGS_DEFCH (CHOSTR)
```

where CHOSTR is a character string specifying which keys are to be considered to be valid choices. If the key specified by the first character in the string is pressed an integer 1 will be returned, if the second a 2 will be returned, and so on. If the key is not included in the string a zero will be returned. There is no restriction on which characters can be specified but upper and lower case characters are not distinguished and you are advised only to use the FORTRAN character set (`A-Z 0-9 =+-*'/(),.:$` and blank) as other characters cannot be guaranteed to be present on a terminal. For this reason, if the number of choices on device 0 is inquired 49 is returned.

A typical use of this facility might be a program which plots a star field and then asks the user to identify each object on the plot as either a star a galaxy or a plate defect by positioning the cursor on each object in turn and pressing the S G or D keys to indicate the nature of the object or the '.' key to terminate the sequence. The interactive part of this program might be:

```
      INTEGER N
      REAL X,Y

*     Choice device number for keyboard
      INTEGER KB
      PARAMETER (KB=0)
```

```
      :
      :
*   Select terminal keyboard as choice device
     CALL SGS_SELCH (KB)

*   Define valid keys
     CALL SGS_DEFCH ('SGD.')
100   CONTINUE
     CALL SGS_REQCU (X, Y, N)
     GO TO (200, 300, 400, 500) N

*   Not a valid choice (N is 0) - try again
     GO TO 100

*   A Star
200   ...
     GO TO 100

*   A Galaxy
300   ...
     GO TO 100

*   A Defect
400   ...
     GO TO 100

*   No more objects ..
 500 CONTINUE
```

It is a Starlink recommendation that you use the "." character to terminate a sequence of operations such as this.

# 10 PROCESSING WORKSTATION NAMES

The routine SGS_WNAME allows the list of workstation names known to SGS, and their associated comments, to be retrieved for processing by your program.

```
CALL SGS_WNAME (ROUTIN, I, ISTAT)
```

where ROUTIN is the name of a subroutine which is to process the names, I is an integer which is also passed your routine, and ISTAT is a status. ROUTIN (which must of course be declared as EXTERNAL) is called once for each workstation in the list with the following arguments:

| | | |
|---|---|---|
| NAME | (CHARACTER) | Workstation name |
| COMMNT | (CHARACTER) | Associated comment string |
| I | (INTEGER) | The value specified in the call to SGS_WNAME |
| ISTAT | (INTEGER) | Status return |

The character arguments should be declared as assumed length variables and can be up to 255 characters long. The integer argument I can be used for any purpose you wish; a typical use would be to pass a logical unit number to which the routine is to write the workstation names. ISTAT should be set to zero if the routine completes successfully; if any other value is returned SGS_WNAME will abandon processing the list and return immediately to its caller.

# 11 EXAMPLE PROGRAM

This section consists of a complete program using SGS. This and another (longer) program are available on disc in the files `/star/share/sgs/sgsx1.f` and `/star/share/sgs/sgsx2.f`[3].

```
      PROGRAM SGSX1

*-
*
*     - - - - - - - - -
*        S G S X 1
*     - - - - - - - - -
*
*     USES SGS PACKAGE TO DRAW BOX WITH 'STARLINK' WRITTEN IN IT
*
*-

      CHARACTER*20 WKSTN
      INTEGER IZONID, IZ, J

*  Print list of workstation names
      CALL SGS_WLIST (6)

*  Get workstation type & number
      PRINT *, 'WORKSTATION?'
      READ (*,'(A)') WKSTN

*  Open
      CALL SGS_OPEN (WKSTN, IZONID, J)

*  Declare a square zone
      CALL SGS_ZSHAP (1.0, 'BL', IZ, J)

*  Box
      CALL SGS_BOX (0.1, 0.9, 0.4, 0.6)

*  Message
      CALL SGS_SHTX (0.1)
      CALL SGS_STXJ ('CC')
      CALL SGS_BTEXT (0.5, 0.5)
      CALL SGS_ATEXT ('STARLINK')
```

[3]If you are using this software on a non-Starlink machines you may have to replace `/star` by some other path name.

```
*  Wrap up
      CALL SGS_CLOSE

      END
```

# A LINKING AND RUNNING SGS

A program can be linked with SGS by:

```
f77 prog.f -L/star/lib `sgs_link`
```

Adam application are linked with SGS by putting `` `sgs_link_adam` `` on the alink command line.

Workstation names are translated to their GKS equivalents using the 'Graphics Name Service' (xrefSUN/57sun57). A list of those names defined on your system can be output to your terminal by running the program `/star/bin/examples/sgs_workstations`[4]. In addition to the system defined names an explicit GKS workstation type and connection identifier can be used. Examples of valid strings are:

```
"GKS_102_0"
"201"
"201,1"
"103 0"
```

SGS can be opened without further ado by supplying SGS_OPEN with such a string, but more friendly names can be created with environment variables. For example after executing the C shell command:

```
% setenv t4010 GKS_201
```

the name 't4010' can be used to open a Tektronix 4010 as a workstation. When defining logical names you are advised to use the GKS_ prefix to distinguish them other environment variables and to use an underscore or comma as the separator as the space is used as a separators by most shells.

The connection identifier indicates which of several devices of the same type to use. For devices which are terminals, 5 indicates your own terminal. For other devices 0 will be a 'default' device of some sort. For connection identifiers other than zero, GKS will attempt to translate the environment variable GKS_n_m (where n is the workstation type and m is the connection identifier) in order to find the name of the device to use.

You should refer to the GKS documentation for precise details of the mapping of connection identifiers to device names for any particular device type.

[4] if you are using this software on a non-starlink machines you may have to replace `/star` by some other path name.

# B SGS–GKS INTERACTION

SGS does not aim to address all the graphics facilities that are available through GKS, and although the majority of graphics programs will not call GKS routines directly, some more sophisticated programs will have to. SGS has been designed to make this as easy as possible, but in order to mix SGS and GKS successfully it is necessary to have some understanding of how SGS controls the state of GKS and how changing the state of GKS may affect the result of SGS calls.

Two routines are provided for obtaining the numbers that GKS uses to identify workstations, and which are often required as inputs to GKS routines.

```
CALL SGS_ICURW (IWKID)}
```

will return the GKS workstation identifier of the current SGS zone, and:

```
CALL SGS_WIDEN (WKSTN,ITYPE,ICONID,ISTAT)}
```

will translate the SGS workstation name WKSTN to a GKS workstation type and connection identifier. ISTAT is set to zero if WKSTN is a valid SGS name, but this does not guarantee that the workstation can be opened. This routine may be called before SGS_OPEN.

SGS makes various assumptions about the state of GKS but rarely enforces them. Therefore changing GKS's state (for example the aspect source flag settings) can be used to alter the behaviour of SGS in subtle and creative ways, but of course if used unwisely can produce bizarre and unexpected effects.

### Coordinate systems

In the discussion that follows, (i) all the coordinate systems mentioned will be 2-D and rectangular, (ii) official GKS terms will be given in capitals when they are being defined, and (iii) only rather ordinary cases are described. It should be read in conjunction with the diagram, which follows the discussion.

The **workstation** is the complete graphics station, consisting of at least a display device and perhaps some input devices as well. An example of a workstation is a VDU with keyboard and joystick; the VDU is the display device in this case. The region of a display device on which images can be displayed is called the **display surface**; for the example of the VDU the display surface is the addressable area of the screen.

Moving now to the other end of the chain, **world coordinates** are those in which the user (i.e. the programmer) wishes to work. The units may be natural for the information being plotted (counts versus Ångstroms for example) but equally often will simply be convenient units for planning the picture. (Note that the world coordinate x and y always increase from left to right and from bottom to top; this is a fundamental property of GKS.)

The window is mapped onto the display surface by means of a sequence of two independently controllable transformations. Each of the two transformations proceeds from a 'window' to a 'viewport'.

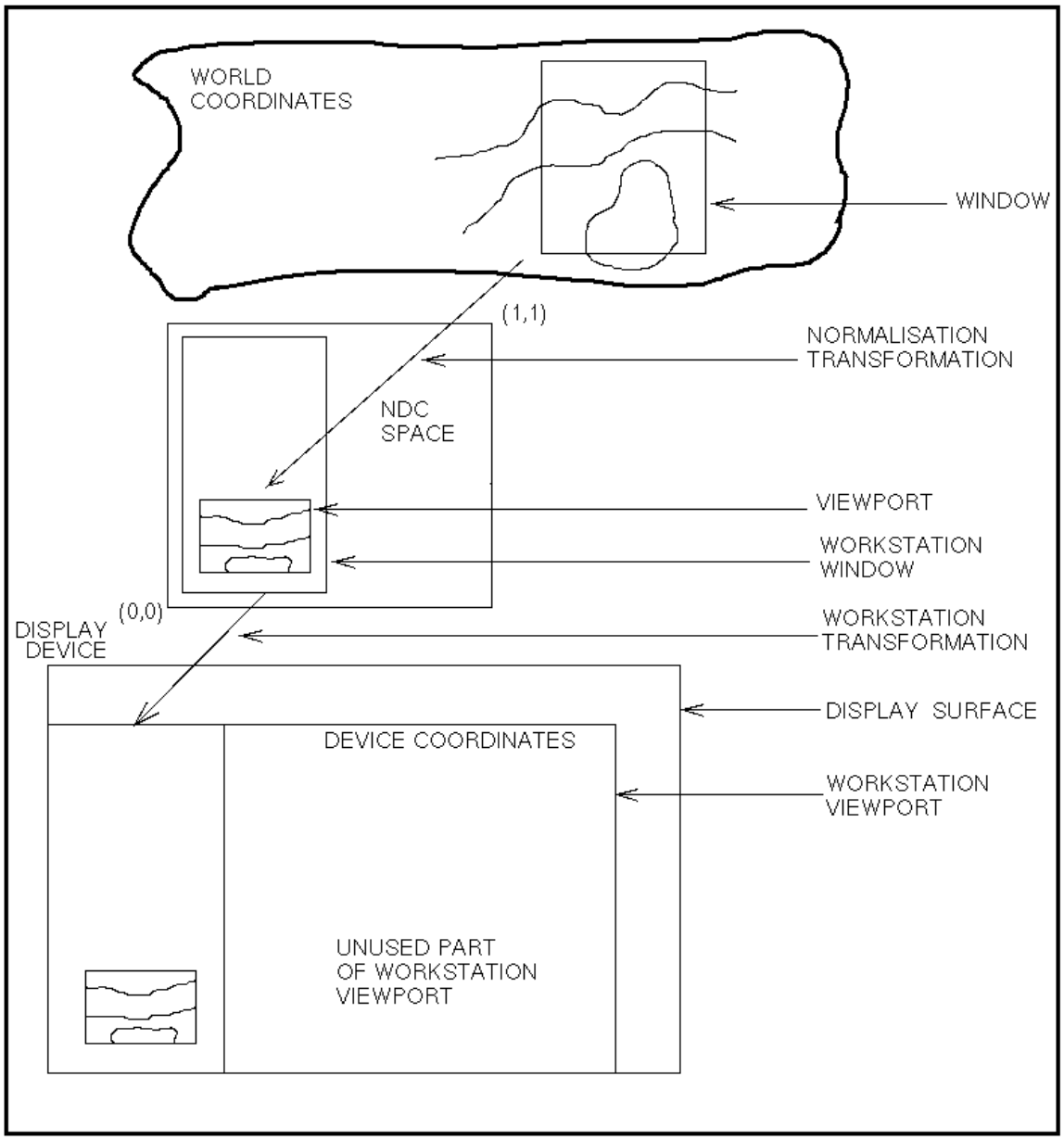


Figure 1: GKS Coordinate Systems

The first transformation, called the **normalisation transformation**, starts with the user's region of interest in world coordinates which is called simply the **window**. This maps onto an intermediate system, called **normalised device coordinates** (NDC); the NDC space consists of a square region whose bottom left and top right corners are respectively (0,0) and (1,1). The region within NDC space onto which the world coordinate window maps is called the **viewport**.

The second transformation is called the **workstation transformation**. For each workstation, there is a region in NDC space called the **workstation window**, and this maps onto part or all of a region of the display surface called the **workstation viewport**.

The NDC space is an idealised model of the different plotting devices; in particular, the things plotted in NDC space appear on the various devices without change of shape—a circle in NDC space will appear circular on all display devices, for example. Because of this conservation of aspect ratio, the workstation window may not map onto the whole of the workstation viewport, although this will commonly be contrived.

The normalisation transformation allows the window to be positioned on the display surface without needing to refer to any device units, and the workstation transformation allows different workstations to be treated separately.

Each workstation has its own workstation viewport and window and there can be many window (world coordinates) to viewport (NDC) transformations in existence, the one used to transform output coordinates being selected by GSELNT.

### Buffering

Polylines and text strings are buffered within SGS and so at any time there may be lines and text which have been 'plotted' by SGS calls but have not yet been passed to GKS (let alone output to the workstation). Therefore any GKS calls should normally be preceded by a call to SGS_FLUSH to ensure that all graphics activities are presented to GKS in the correct order.

### Transformations

SGS only sets normalization transformation number 1. Selecting a new zone is the only SGS activity that selects this to be the current transformation. Transformation 1 is used by SGS_SETCU when it sets the locator position.

SGS_SELZ also sets the input priority of transformation 1 to be greater than that of transformation 0 and the locator input routines expect transformation 1 to have the highest priority of all transformations.

### Aspect source flags

SGS_OPEN and SGS_INIT set the following aspect source flags:

**POLYLINE** — bundled
**POLYMARKER** — individual
**TEXT** — individual

SGS assumes these settings when it manipulates the polyline, marker and text attributes.

**Polylines**

Setting the SGS pen number simply sets the polyline index. If the ASF for any attribute has been changed to individual then that attribute will be unaffected by changing the SGS pen. Pen numbers greater than 4 may be used provided that the corresponding polyline index has been defined (either explicitly or by default). If any polyline indices (or any other primitive index) are set explicitly then this must be done before any drawing takes place, otherwise GKS will consider that the picture needs to be regenerated. When this happens, any drawing done outside a segment will be lost. Since Close Workstation forces regeneration if it is required, the net effect is for the workstation to be cleared when it is closed!

**Markers**

When the SGS pen number is changed, the colour index of the new pen on the current workstation is inquired and the polymarker colour index set to the same value. This results in markers being plotted in the same colour as lines *on the current workstation* but not necessarily on any other workstations that may be active. Activating a new workstation by selecting a new SGS zone modifies the colour index to match the polylines on the new workstation. If either the polyline colour index ASF has been set to individual or the polymarker colour index ASF has been set to bundled, this mechanism will fail.

The SGS marker routines set the current marker type. If the marker type ASF has been set to bundled, the marker type argument will have no effect.

**Text**

Text colour works in exactly the same way as marker colour and the same remarks apply. If any other text ASF is set to bundled, setting the corresponding SGS text attribute will have no effect.

# C SUBROUTINE SPECIFICATIONS

The parameter descriptions have the following format:

*par* type description

The 'type' specifications use the following codes with the meanings shown:

| | |
|---|---|
| C | CHARACTER∗(∗) |
| I | INTEGER |
| R | REAL |
| Cn | CHARACTER∗n |
| L | LOGICAL |

## SGS_APOLY
## Append new line to polyline

**ACTION:** Adds a new line, whose end-point is (*x,y*), to an existing buffered polyline. It is possible that a partial polyline will be output automatically because of finite buffer space (which however will be large). SGS will, in this case, record that the polyline has to be continued but the join might not be perfect in the case of dotted and dashed line-styles.

**CALL:** `SGS_APOLY(x,y)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *x,y* | R | The end point of the line. |

**GKS EFFECT:** If the current polyline buffer becomes full, then the polyline so far is sent to GKS. The specified point is added to the polyline buffer.

## SGS_ARC
## Draw arc of a circle

**ACTION:** Draws an arc of a circle, centre (*x,y*), radius *r*, starting at angle *theta1* and finishing at angle *theta2* in an anti-clockwise direction.

**CALL:** `SGS_ARC(x,y,r,theta1,theta2)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *x,y* | R | Centre of complete circle in world coordinates. |
| *r* | R | Radius of complete circle in world coordinates. |
| *theta1,theta2* | R | Start and finish angle of the arc in radians. The angle increases anti-clockwise and 0 is at (x+r,y). |

**GKS EFFECT:** Starts a new polyline (equivalent of SGS_BPOLY). Adds lines to the polyline, approximating the arc. Finally, the polyline is sent to GKS. The line length is chosen to correspond to the addressability of the finest active workstation (inquiry is made of GKS to determine this).

## SGS_ATEXT
## Append a field to the text buffer

**ACTION:** Appends to the SGS buffered text the entire specified CHARACTER expression (all spaces included). If the buffer would overflow, excess characters are lost.

**CALL:** `SGS_ATEXT(string)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *string* | C | Text string to be appended entirely. |

**GKS EFFECT:** None.

## SGS_ATXB
## Append to text with blanks

**ACTION:** Appends to the SGS buffered text *nspace* spaces followed by the specified CHARACTER expression with all its leading spaces removed. If the buffer would overflow, excess characters are lost.

**CALL:** `SGS_ATXB(string,nspace)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *string* | C | Text string to be appended. |
| *nspace* | I | The exact number of spaces that are to be put in front of the first non-space character appended to the buffer. |

**GKS EFFECT:** None.

## SGS_ATXI
## Append to text an integer

**ACTION:** Converts an integer to a character string, which is then placed in a field, using *nfi* as an indication of field width. Appends the resulting field to the SGS buffered string.

**CALL:** `SGS_ATXI(i,nfi)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *i* | I | Integer to be converted to characters. |
| *nfi* | I | Indicates what is to be done with the integer. |

**GKS EFFECT:** None.

**NOTES:** If $nfi \geq 0$, *nfi* spaces are appended to the SGS buffered string, followed by the characters representing the number. If $nfi < 0$, the characters representing the number are put right-justified into a field which is $-nfi$ characters wide. If the field width is not sufficient, leading characters are lost.

## SGS_ATXL
## Append to text left justified

**ACTION:** Appends to the SGS buffered text the specified CHARACTER expression with trailing spaces removed. If the buffer would overflow, excess characters are lost.

**CALL:** `SGS_ATXL(string)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| string | C | Text string to be appended. |

**GKS EFFECT:** None.

## SGS_ATXR
## Append to text a real number

**ACTION:** Converts a real number to a character string, using *ndp* to determine the number of decimal spaces. Places the string in a field, using *nfi* as an indication of field width. Appends the field to the SGS buffered string.

**CALL:** `SGS_ATXR(r,nfi,ndp)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *r* | R | Real number to be converted to characters. |
| *nfi* | I | Similarly to routine ATXI, this parameter indicates how the converted real number is to be put in the field. |
| *ndp* | I | Indicates the number of digits after the decimal point. |

**GKS EFFECT:** None.

**NOTES:** If $ndp > 0$, then *ndp* digits are included. If $ndp < 0$, then only the integer part appears. If $ndp = 0$, then only the integer part and the decimal point appears.

---

## SGS_BOX

## Draw a box SGS_BOX

---

**ACTION:** Draws a rectangular box with sides parallel to the axes.

**CALL:** `SGS_BOX(x1,x2,y1,y2)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *x1,x2,y1,y2* | R | Extent of box in world coordinates. |

**GKS EFFECT:** A polyline is drawn (via SGS_BPOLY and SGS_APOLY).

---

## SGS_BPOLY
## Begin polyline

---

**ACTION:** Indicates that a new chain of lines (could be only one line) is to begin at the specified point. Further end points are added by SGS_APOLY. Any existing buffered polyline is output first.

**CALL:** `SGS_BPOLY(x,y)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| x,y | R | Starting co-ordinates for polyline. |

**GKS EFFECT:** Any existing buffered polyline is output first by calling the GKS polyline routine. Then the starting position of the new polyline is saved.

## SGS_BTEXT
## Begin a new text string

**ACTION:** Flushes the SGS buffered string if one exists, and sets up a new one with zero characters. The reference point is recorded.

**CALL:** `SGS_BTEXT(x,y)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *x,y* | R | Reference point in world coordinates. This is used when the buffer is flushed. |

**GKS EFFECT:** The current text string (if it exists) is sent to GKS.

## SGS_BZNDC
## Set a base zone extent in NDC

**ACTION:** Sets the extent in NDC of a base zone.

**CALL:** `SGS_BZNDC(x1,x2,y1,y2,pos,istat)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *x1,x2,y1,y2* | R | Extent of zone in normalised device coordinates. |
| *pos* | C2 | Determines the positioning of the zone on the display surface. The first character is 'B', 'C' or 'T' (standing for bottom, centre and top), and the second is 'L', 'C' or 'R' (standing for left, centre and right). |

**OUTPUT PARAMETERS:**

| | | |
|---|---|---|
| *istat* | I | Status. |

**GKS EFFECT:** The viewport and workstation window are set to the requested NDC values, and the workstation viewport set to the largest rectangle of the same aspect ratio that will fit on the display surface, and positioned as specified.

## SGS_CIRCL
## Draw circle

**ACTION:** Draws a complete circle, centre (*x,y*), radius *r*.

**CALL:** `SGS_CIRCL(x,y,r)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *x,y* | R | Centre of circle in world coordinates. |
| *r* | R | Radius of circle in world coordinates. |

**GKS EFFECT:** Similar to ARC routine except that complete circle is drawn.

## SGS_CLOSE
## Close graphics

**ACTION:** Puts out the SGS text and polyline buffers. Closes GKS.

**CALL:** `SGS_CLOSE`

**GKS EFFECT:** All pending output dispatched (via SGS_FLUSH). All active workstations are deactivated. All open workstations are closed. GKS is closed.

## SGS_CLRBL
## Clear block

**ACTION:** Clears specified area of screen if this can be done without affecting the rest of the display surface.

**CALL:** `SGS_CLRBL(x1,x2,y1,y2)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *x1,x2,y1,y2* | R | Extent of area in world coordinates. |

**GKS EFFECT:** The specified area is cleared. The technique used depends on the workstation type.

## SGS_CLRFG
## Set clear screen flag

**ACTION:** Sets or clears the 'suppress clear screen flag on open workstation' according to the *iflag* argument. N.B. This function is specific to RAL GKS and is not implemented on all workstations.

**CALL:** `SGS_CLRFG(iflag)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *iflag* | I | 1 disables screen clearing on open; 0 restores the normal GKS behaviour. |

**GKS EFFECT:** GESC is called with escape number 1000.

## SGS_CLRZ
## Clear zone

**ACTION:** Clears the current zone. The entire display surface may be cleared on some devices.

**CALL:** `SGS_CLRZ`

**GKS EFFECT:** If the device is capable of selective erasure SGS_CLRBL is called, otherwise the entire display surface is cleared.

## SGS_CLSWK
## Close workstation

**ACTION:** Puts out the SGS text and polyline buffers. If there are no other base zones on the workstation, releases all zones associated with the workstation, deactivates and closes the workstation. Otherwise, the specified zone is released.

**CALL:** `SGS_CLSWK(izonid,istat)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *izonid* | I | Zone identifier for a base zone on this workstation. |

**OUTPUT PARAMETERS:**

| | | |
|---|---|---|
| *istat* | I | If workstation is closed OK, then this value is zero. |

**GKS EFFECT:** All pending output dispatched (via SGS_FLUSH). The appropriate workstation may be deactivated and closed.

## SGS_CUVIS
## Set cursor visibility

**ACTION:** Sets the visibility of the cursor on the current SGS workstation.

**CALL:** `SGS_CUVIS(vis)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *vis* | L | The visibility of the cursor. |

**GKS EFFECT:** The echo switch for locator device 1 is set.

## SGS_DEFCH
## Define valid choice keys

**ACTION:** Defines the valid keys for choice input from the command terminal keyboard (SGS choice device type 0).

**CALL:** `SGS_DEFCH(chostr)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *chostr* | C | A character expression of up to 49 characters which defines the keys that are valid for choice input. The order of the characters determines the number returned when a key is pressed. The first key in the string will return 1, the second 2, etc. If a key which has not been defined is pressed, a 0 is returned. |

**GKS EFFECT:** None

## SGS_DISCU
## Disable sample cursor

**ACTION:** Disables sample mode for the cursor.

**CALL:** `SGS_DISCU`

**GKS EFFECT:** The mode of locator device 1 on the current SGS workstation is set to REQUEST.

## SGS_ENSCU
## Enable sample cursor

**ACTION:** Enables sample mode for the cursor.

**CALL:** `SGS_ENSCU`

**GKS EFFECT:** The mode of locator device 1 on the current SGS workstation is set to SAMPLE.

## SGS_FLUSH
## Flush buffers

**ACTION:** Flushes all buffers.

**CALL:** `SGS_FLUSH`

**GKS EFFECT:** Flushes the SGS text and line buffers before calling GUWK for all active workstations to flush the GKS buffers.

## SGS_ICUAV
## Inquire cursor availability

**ACTION:** Sets *avail* to .TRUE. if a cursor is available on the current SGS device, and to .FALSE. otherwise.

**CALL:** `SGS_ICUAV(avail)`

**OUTPUT PARAMETERS:**

| | | |
|---|---|---|
| *avail* | L | .TRUE. or .FALSE., depending on the availability of a cursor. |

**GKS EFFECT:** Inquires only.

## SGS_ICURW
## Inquire current workstation

**ACTION:** Returns workstation ID for current zone.

**CALL:** `SGS_ICURW(iwkid)`

**OUTPUT PARAMETERS:**

| | | |
|---|---|---|
| *iwkid* | I | The workstation ID for the SGS zone last selected. |

**GKS EFFECT:** None.

## SGS_ICURZ
## Inquire current zone

**ACTION:** Returns ID of current SGS zone.

**CALL:** `SGS_ICURZ(izonid)`

**OUTPUT PARAMETERS:**

| | | |
|---|---|---|
| *izonid* | I | Identifier of last selected SGS zone. |

**GKS EFFECT:** None.

## SGS_IDUN
## Inquire device units

**ACTION:** Obtains the length of device units in each of x and y for the current SGS zone.

**CALL:** `SGS_IDUN(dxw,dyw)`

**OUTPUT PARAMETERS:**

| | | |
|---|---|---|
| *dxw,dyw* | R | Size of a raster unit in world coordinates in each of the x and y directions. |

**GKS EFFECT:** Inquires only.

## SGS_INCHO
## Inquire number of choices

**ACTION:** Inquires the number of choices available on the currently selected workstation.

**CALL:** `SGS_INCHO(nchoic,n)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *nchoic* | I | The SGS choice device number: 1 or greater is a GKS choice device, 0 is the command terminal. |

**OUTPUT PARAMETERS:**

| | | |
|---|---|---|
| *n* | I | The number of choices available on the selected device. If no such device exists, zero will be returned. |

**GKS EFFECT:** Inquires only.

## SGS_INIT
## Initialize SGS

**ACTION:** Initializes SGS and (if necessary) opens GKS.

**CALL:** `SGS_INIT(lun,istat)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *lun* | I | Logical unit number for error messages. |

**OUTPUT PARAMETERS:**

| | | |
|---|---|---|
| *istat* | I | Status; set to 0 if SGS is successfully initialized. |

**GKS EFFECT:** If GKS is not already open GOPKS is called with the specified logical unit number as the error channel.

## SGS_IPEN
## Inquire pen number

**ACTION:** Returns current SGS pen number.

**CALL:** `SGS_IPEN(npen)`

**OUTPUT PARAMETERS:**

| | | |
|---|---|---|
| *npen* | I | The current SGS pen number as presented via SGS_SPEN. |

**GKS EFFECT:** None.

## SGS_IPLXY
## Inquire polyline x and y

**ACTION:** Returns the coordinates of the last line added to the polyline buffer, if there is one.

**CALL:** `SGS_IPLXY(x,y)`

**OUTPUT PARAMETERS:**

| | | |
|---|---|---|
| *x,y* | R | *x* and *y* of the last line added to the polyline buffer, if there is such a line. If not, *x* and *y* are unchanged. |

**GKS EFFECT:** None.

## SGS_ISLER
## Inquire selective erase capability

**ACTION:** Sets *blker* to .TRUE. if the current workstation is capable of clearing selected areas of the display; if not it is set to .FALSE..

**CALL:** `SGS_ISLER(blker)`

**OUTPUT PARAMETERS:**

| | | |
|---|---|---|
| *blker* | L | .TRUE. or .FALSE. depending on whether device has selective erase capability. |

**GKS EFFECT:** None.

## SGS_ITXA
## Inquire text attributes

**ACTION:** Returns the values of all the SGS text attributes.

**CALL:** `SGS_ITXA(nf,npr,ht,ar,xu,yu,sp,txj)`

**OUTPUT PARAMETERS:**

| | | |
|---|---|---|
| *nf,npr* | I | Font number and text precision. |
| *ht,ar* | R | Character height, aspect ratio. |
| *xu,yu* | R | Text orientation direction cosines. |
| *sp* | R | Text spacing. |
| *txj* | C2 | Text alignment coding. |

**GKS EFFECT:** None.

## SGS_ITXB
## Inquire text buffer

**ACTION:** Returns state of SGS buffered string.

**CALL:** `SGS_ITXB(x,y,n,dx,dy)`

**OUTPUT PARAMETERS:**

| | | |
|---|---|---|
| *x,y* | R | Reference position of SGS buffered string. |
| *n* | I | Number of characters in buffered string. |
| *dx,dy* | R | The extent of the current buffered string (expressed as a displacement from its starting point). |

**GKS EFFECT:** Inquiry is made of GKS to determine the concatenation point, from which *x* and *y* are subtracted to give the SGS extent.

## SGS_IZONE
## Inquire zone attributes

**ACTION:** Returns bounds and size of current SGS zone.

**CALL:** `SGS_IZONE(x1,x2,y1,y2,xm,ym)`

**OUTPUT PARAMETERS:**

| | | |
|---|---|---|
| *x1,x2,y1,y2* | R | Bounds of current zone. |
| *xm,ym* | R | Size of zone in metres. |

**GKS EFFECT:** The workstation transformation for the workstation of the current SGS zone is inquired via GQWKT.

## SGS_LINE
## Begin polyline with single line

**ACTION:** Starts a new polyline with the specified line. If subsequent lines are added, the first 2 points will be (*x1,y1*) and *(x2,y2)*.

**CALL:** `SGS_LINE(x1,y1,x2,y2)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *x1,y1* | R | Coordinates of the first point on the polyline. |
| *x2,y2* | R | End point of the line. |

**GKS EFFECT:** Effect is same as `CALL SGS_BPOLY(`*x1*`,`*y1*`)` then `CALL SGS_APOLY(`*x2*`,`*y2*`)`.

## SGS_MARK
## Draw marker

**ACTION:** Draws a single marker, centred on the given position (*x,y*).

**CALL:** `SGS_MARK(x,y,mtype)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *x,y* | R | Coordinates of the marker. |
| *mtype* | I | Marker type. |

**GKS EFFECT:** The current marker type is set to *mtype* and a marker is output using the GKS polymarker routine GPM. No SGS buffers are affected.

**NOTES:** The correspondence between marker types and marker shapes is shown in section 8.

## SGS_MARKL
## Draw marker at end of polyline

**ACTION:** If there is an SGS buffered polyline, it is output, a marker is output at its end position and a new buffered polyline is begun.

**CALL:** `SGS_MARKL(mtype)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *mtype* | I | Marker type. |

**GKS EFFECT:** Current buffered polyline is output via GKS; the current marker type is set to *mtype* and a marker is output at the end position using the GKS polymarker routine GPM; a new SGS buffered polyline is begun.

**NOTES:** Same as SGS_MARK.

## SGS_OPEN
## Open graphics

**ACTION:** Performs all GKS calls necessary to get the specified workstation open and active. Also sets defaults for various attributes and should therefore be used if other SGS routines are called. It is intended to be used at the beginning of graphics.

**CALL:** `SGS_OPEN(wkstn,izonid,istat)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *wkstn* | C | Workstation name: a character string which translates ultimately into GKS workstation sequence number and connection identifier. See SUN/57 for full details of formats supported; simplest is the two numbers as decimals separated by a comma. |

**OUTPUT PARAMETERS:**

| | | |
|---|---|---|
| *izonid* | I | Zone identifier for the base SGS zone on this workstation; allocated by SGS. |
| *istat* | I | If workstation is opened OK, then this value is zero. |

**GKS EFFECT:** If GKS is already open it is left open; otherwise it is opened with logical unit 22 as the error channel. The specified workstation is opened and activated. Maximum deferral is requested. An SGS base zone filling the display surface is set up. The polyline aspect source flags are set to bundled and the marker and text aspect source flags set to individual.

## SGS_OPNWK
## Open workstation

**ACTION:** Opens an additional workstation, and sets up the base SGS zone, which is then selected.

**CALL:** `SGS_OPNWK(wkstn,izonid,istat)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *wkstn* | C | Workstation type/connection identifier pair (see SGS_OPEN). |

**OUTPUT PARAMETERS:**

| | | |
|---|---|---|
| *izonid* | I | Zone identifier—number allocated by SGS. |
| *istat* | I | If workstation is opened OK, this value is 0. |

**GKS EFFECT:** Unless already opened by SGS the specified workstation is opened; the workstation viewport is set to cover the full display surface. A base zone is set up and selected via SGS_SELZ.

## SGS_OPOLY
## Output buffered polyline

**ACTION:** Outputs buffered polyline. SGS_OPOLY is automatically invoked, at appropriate times, by other routines and only rarely requires calling directly.

**CALL:** `SGS_OPOLY`

**GKS EFFECT:** The buffered polyline is sent to GKS.

## SGS_OTEXT
## Output buffered text

**ACTION:** Outputs existing SGS buffered text. The point stored by SGS_BTEXT is used as the reference point. Current text attributes are used, including the SGS text alignment. SGS_OTEXT is automatically invoked, at appropriate times, by other routines and rarely needs to be called directly.

**CALL:** `SGS_OTEXT`

**GKS EFFECT:** The current text string is sent to GKS.

## SGS_RELZ
## Release zone

**ACTION:** Offers the specified SGS zone for release.

**CALL:** `SGS_RELZ(izonid)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *izonid* | I | The identifier for the SGS zone which the program no longer requires. |

**GKS EFFECT:** None.

## SGS_REQCH
## Request choice

**ACTION:** Waits for the user to press a key on the current SGS choice device and returns the number of the key selected. When choice device 0 is being used, the number is the position in the define keys string passed to SGS_DEFCH. If the key is not included in the string, 0 is returned.

**CALL:** `SGS_REQCH(n)`

**OUTPUT PARAMETERS:**

| | | |
|---|---|---|
| *n* | I | The number of the key pressed. |

**GKS EFFECT:** All buffered output is plotted and for device 1 GKS request choice is called.

## SGS_REQCU
## Request cursor position

**ACTION:** Waits for the user to press a key on the current SGS choice device and then returns the position of the cursor and the number of the choice key.

**CALL:** `SGS_REQCU(x,y,n)`

**OUTPUT PARAMETERS:**

| | | |
|---|---|---|
| *x,y* | R | The cursor position in world coordinates. |
| *n* | I | The number of the key pressed. |

**GKS EFFECT:** GKS request locator is called. If the current choice device is not 0, GKS request choice is called.

## SGS_SAMCU
## Sample cursor

**ACTION:** Returns the cursor position without waiting for the user to press any key.

**CALL:** `SGS_SAMCU(x,y)`

**OUTPUT PARAMETERS:**

| | | |
|---|---|---|
| *x,y* | R | The cursor position in world coordinates. |

**GKS EFFECT:** All output buffers are plotted and locator device 1 is sampled on the current SGS workstation.

## SGS_SARTX
## Set aspect ratio of text

**ACTION:** Sets aspect ratio (width/height) of text. A usual figure is between 1/1.3 and 1/1.5; the default is 2/3.

**CALL:** `SGS_SARTX(ar)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *ar* | R | Aspect ratio. |

**GKS EFFECT:** Outputs any SGS buffered text, sets SGS aspect ratio and resets all GKS text attributes to follow current SGS settings.

## SGS_SELCH
## Select choice device

**ACTION:** Selects the current SGS choice device.

**CALL:** `SGS_SELCH(nchdev)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *nchdev* | I | The SGS choice device to be used for subsequent input: 1 or greater is a GKS choice device number, 0 is the command terminal. |

**GKS EFFECT:** None.

## SGS_SELZ
## Select zone

**ACTION:** Selects the nominated SGS zone.

**CALL:** `SGS_SELZ(izonid,istat)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *izonid* | I | The zone identifier supplied by SGS when the zone was set up. |
| *istat* | I | If the zone is selected successfully, then this value is 0. |

**GKS EFFECT:** If the new zone is on a different workstation from the current zone, the current workstation is deactivated and the new one activated and the marker and text indices are set to those of the current SGS pen. Normalization transformation number 1 is selected and a new window/viewport is set up.

## SGS_SETCU
## Set cursor position

**ACTION:** Sets the cursor to the specified position, provided that the hardware is capable of this. If the cursor is not visible, it will appear at the requested position when it next becomes visible. If the requested position is outside the workstation window, no action is taken.

**CALL:** `SGS_SETCU(x,y)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *x,y* | R | The desired cursor position in world coordinates. |

**GKS EFFECT:** Initialize Locator called with (*x,y*) as the new initial position and transformation 1 as the initial transformation. Other attributes are unchanged.

## SGS_SFONT
## Set font of text

**ACTION:** Sets font number.

**CALL:** `SGS_SFONT(nf)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *nf* | I | Font number. The default is 1. See the GKS user guide for details of other available fonts. |

**GKS EFFECT:** Outputs any SGS buffered text, sets SGS font number, selects GKS font and precision.

## SGS_SHTX
## Set height of text

**ACTION:** Sets character height. Default is 0.02.

**CALL:** `SGS_SHTX(ht)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *ht* | R | Character height in world coordinates. |

**GKS EFFECT:** Outputs any SGS buffered text, sets SGS character height and sets GKS text attributes from the SGS settings.

**NOTES:** If the coordinate system changes, then the displayed size will alter. The program would then have to supply a new value to maintain the same display size.

## SGS_SPEN
## Select pen

**ACTION:** Selects SGS pen. Any polyline or text string is flushed first. The initial pen is number 1.

**CALL:** `SGS_SPEN(npen)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *npen* | I | Required SGS pen number. |

**GKS EFFECT:** The GKS pen number is set to the SGS pen number given. All available workstations are assumed to support at least 4 predefined pens, giving readily distinguishable lines. The marker and text colour indices are set to the same value as for polylines on the current SGS workstation.

## SGS_SPREC
## Set precision of text

**ACTION:** Sets text precision. This indicates the fidelity with which the text attributes are followed by GKS. Precision 2 (the SGS default) is the most accurate and is normally achieved by a software font.

**CALL:** `SGS_SPREC(npr)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *npr* | I | Text precision. |

**GKS EFFECT:** Outputs any SGS buffered text, sets SGS precision number, sets GKS font and precision.

## SGS_SSPTX
## Set spacing of text

**ACTION:** Sets text spacing. Default is 0.0.

**CALL:** `SGS_SSPTX(sp)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *sp* | R | Spacing between the end of one character and the start of the next. |

**GKS EFFECT:** Outputs any SGS buffered text. Sets GKS text attributes from the SGS settings.

**NOTES:** Spacing in SGS is expressed as a fraction of the nominal width of the character box.

## SGS_STXJ
## Set text justification

**ACTION:** Sets the alignment options that are subsequently used by SGS_OTEXT and by implicit and explicit flushes of the SGS text buffer.

**CALL:** `SGS_STXJ(txj)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *txj* | C2 | A 2 letter option string. |

**GKS EFFECT:** Outputs SGS buffered text. Sets GKS text attributes from SGS settings.

**NOTES:** The 1st letter is the vertical alignment: B(OTTOM), C(ENTRE) or T(OP). The 2nd is the horizontal alignment: L(EFT), C(ENTRE) or R(IGHT). In either case, space implies no change. The default is 'BL'. The alignment refers to where in the character string the nominated X,Y lies; thus a code of 'BL' would place the nominated X,Y at the bottom left of the string. The terms 'top', 'left' etc. refer to the character string as seen in its conventional orientation, rather than as actually displayed.

## SGS_SUPTX
## Set up vector of text

**ACTION:** Specifies character orientation in terms of a vector. Default is (0.0,1.0).

**CALL:** `SGS_SUPTX(xu,yu)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *xu,yu* | R | A vector specifying the up direction of subsequent text. |

**GKS EFFECT:** Outputs any SGS buffered text. Sets GKS text attributes from SGS settings.

**NOTES:** The vector is the direction of the vertical axis of each character. The standard orientation would be (0.0,1.0). Only the direction of the vector, not its magnitude, is significant.

## SGS_SW
## Set window

**ACTION:** Sets the bounds of plotting in world coordinates.

**CALL:** `SGS_SW(x1,x2,y1,y2,istat)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *x1,x2* | R | Left and right bounds in X. |
| *y1,y2* | R | Bottom and top bounds in Y. |
| *istat* | I | Status. |

**GKS EFFECT:** Sets the window bounds for normalization transformation number 1 according to the parameters provided. The *x1,x2* and *y1,y2* values are sorted so that $x2 > x1$ and $y2 > y1$ before the GKS transformation is set up.

## SGS_TPZ
## Transform position to new zone

**ACTION:** Translates a position in zone IZIN to its corresponding position in zone IZOUT.

**CALL:** `SGS_TPZ(izin,xin,yin,izout,xout,yout,istat)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *izin* | I | ID of zone of input position. |
| *xin,yin* | R | Position to be transformed. |
| *izout* | I | ID of zone to which the position is to be transformed. |

**OUTPUT PARAMETERS:**

| | | |
|---|---|---|
| *xout,yout* | R | Output position. |
| *istat* | I | Status. |

**GKS EFFECT:** None.

## SGS_TX
## Begin a new text string with a string

**ACTION:** Starts a new SGS buffered text string with the specified text.

**CALL:** `SGS_TX(x,y,string)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *x,y* | R | Reference point for new SGS buffered string. |
| *string* | C | Text string to be put into emptied text buffer. |

**GKS EFFECT:** A new SGS buffered string is begun: equivalent to SGS_BTEXT(x,y); the specified string is appended to it: equivalent to SGS_ATEXT(string)

## SGS_TXI
## Begin new text with integer

**ACTION:** Starts a new SGS buffered text string with text derived from the specified integer.

**CALL:** `SGS_TXI(x,y,i,nfi)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *x,y* | R | Reference point of new SGS buffered string. |
| *i* | I | Integer to be converted to characters. |
| *nfi* | I | Indication of field width: similar to SGS_ATXI. |

**GKS EFFECT:** A new SGS buffered string is begun: equivalent to SGS_BPOLY(x,y). The converted integer is appended to it: equivalent to SGS_ATXI(i,nfi).

## SGS_TXR
## Begin new text with real

**ACTION:** Starts a new SGS buffered string with text derived from the specified real number.

**CALL:** `SGS_TXR(x,t,r,nfi,ndp)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *x,y* | R | Reference point of new SGS buffered string. |
| *r* | R | Real number to be converted to characters. |
| *nfi,ndp* | I | Conversion parameters, similar to SGS_ATXR. |

**GKS EFFECT:** A new SGS buffered string is begun: equivalent to SGS_APOLY(x,y). The converted real number is appended: equivalent to SGS_ATXR(r,nfi,ndp).

## SGS_WIDEN
## Translate SGS workstation name to GKS

**ACTION:** Translates the SGS workstation name to its equivalent GKS workstation type and connection identifier. May be called before SGS_OPEN.

**CALL:** `SGS_WIDEN(wkstn,itype,iconid,istat)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *wkstn* | C | SGS workstation name. |

**OUTPUT PARAMETERS:**

| | | |
|---|---|---|
| *itype* | I | GKS workstation type. |
| *iconid* | I | GKS connection identifier. |
| *istat* | I | Status. Set to zero if translation was possible. |

**GKS EFFECT:** None.

## SGS_WLIST
## List available workstations

**ACTION:** Outputs a list of the SGS workstation names that have been defined on the specified FORTRAN logical unit. This routine may be called before SGS_OPEN.

**CALL:** `SGS_WLIST(LUN)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *lun* | I | FORTRAN logical unit number. |

**GKS EFFECT:** As for SGS_WNAME

---

## SGS_WNAME
## Generate list of workstation names

---

**ACTION:** Passes a list of the SGS workstation names that have been defined to the specified action routine. May be called before SGS_OPEN.

**CALL:** `SGS_WNAME(actrou,iarg,istat)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *actrou* | | The name of the subroutine which will process the names (see user manual for the specification of this routine). |
| *iarg* | I | Integer argument passed unaltered to the action routine. |

**OUTPUT PARAMETERS:**

| | | |
|---|---|---|
| *istat* | I | Status. |
| Set to zero if the routine completes successfully. | | |

**GKS EFFECT:** If not already open, GKS is opened with logical unit 22 as the error channel.

---

## SGS_ZONE
## Create a zone

---

**ACTION:** Creates a new SGS zone occupying the given bounds within the world coordinate system of the current zone. It is then selected.

**CALL:** `SGS_ZONE(x1,x2,y1,y2,izonid,istat)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *x1,x2,y1,y2* | R | The extent of the new zone in the world coordinates of the old zone. |

**OUTPUT PARAMETERS:**

| | | |
|---|---|---|
| *izonid* | I | The zone identifier for the new zone—supplied by SGS. |
| *istat* | I | Status. |

**GKS EFFECT:** Normalization transformation number 1 set and selected and its input priority set to be greater than that of transformation 0.

## SGS_ZPART
## Partition zone

**ACTION:** Partitions the current zone into *nx* by *ny* equal sized zones. The current zone is unchanged.

**CALL:** `SGS_ZPART(nx,ny,izones,istat)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *nx,ny* | I | The number of zones in X and Y. |

**OUTPUT PARAMETERS:**

| | | |
|---|---|---|
| *izones(nx*ny)* | I | A list of the zone IDs of the new zones. |
| *istat* | I | Status. |

**GKS EFFECT:** None.

## SGS_ZSHAP
## Create a zone of given shape

**ACTION:** Creates an SGS zone of the specified shape (aspect ratio) within the current zone and then selects it.

**CALL:** `SGS_ZSHAP(ar,pos,izonid,istat)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *ar* | R | The absolute aspect ratio x/y of the new zone. |
| *pos* | C2 | Determines the positioning of the new zone within the current one. The first character is 'B', 'C' or 'T' (standing for bottom, centre and top), and the second is 'L', 'C' or 'R' (standing for left, centre and right). |

**OUTPUT PARAMETERS:**

| | | |
|---|---|---|
| *izonid* | I | The zone identifier for the new zone—supplied by SGS. |
| *istat* | I | Status. |

**GKS EFFECT:** As for SGS_ZONE.

## SGS_ZSIZE
## Create a zone of given size

**ACTION:** Creates an SGS zone of the specified absolute size within the current zone, then selects it.

**CALL:** `SGS_ZSIZE(xm,ym,pos,izonid,istat)`

**INPUT PARAMETERS:**

| | | |
|---|---|---|
| *xm,ym* | R | The absolute size of the new zone in metres. |
| *pos* | C2 | As for SGS_ZSHAP. |

**OUTPUT PARAMETERS:**

| | | |
|---|---|---|
| *izonid* | I | The zone identifier for the new zone—supplied by SGS. |
| *istat* | I | Status. |

**GKS EFFECT:** As for SGS_ZONE.