SUN/113.5

Starlink Project
Starlink User Note 113.5

P M Allan
A J Chipperfield
D L Terrett

19 July 1995

ADAM Graphics Programmer's Guide 1.1-6 Programmers' Manual

Abstract

This guide outlines the options available to a programmer writing an ADAM applications program who wishes to incorporate graphics into the program and describes the routines that handle the necessary interaction with the ADAM parameter system when a graphics device is opened for a GKS based graphics package.

Contents

1 Introduction

This guide outlines the options available to a programmer writing an ADAM applications program who wishes to incorporate graphics into the program and describes the routines that handle the necessary interaction with the ADAM parameter system when a graphics device is opened for a GKS based graphics package.

Where a suite of graphics programs are being written or where a program is to interact with graphics produced by another package (e.g. KAPPA), the Applications Graphics Interface (AGI—SUN/48) should be used. AGI provides facilities for storing and restoring information about pictures such as extent and world coordinates. AGI is layered on top of the routines described in this document.

2 Application

For most ADAM applications, PGPLOT (SUN/15) is the graphics package of choice; it provides an easy to use yet powerful interface for plotting 2D axes, contour maps, grey-scale, colour images and many other styles of plot. The NCAR package (SUN/88) provides similar facilities and some additional styles of plot (e.g. stream-line diagrams) and is in general more powerful (for example, NCAR can draw an X,Y plot with different scales on the right and left hand axes) but is somewhat harder to use. NCAR also contains a 3D drawing package. It is possible to combine NCAR graphics with calls to SGS and GKS by using the Starlink Extensions to NCAR (SNX—SUN/90) package.

The NAG graphics package can also be used but it is forbidden for software that is to become part of the Starlink software collection because its availability on all hardware platforms is not guaranteed (particularly in its single precision form) and non-Starlink sites may not have a licence to use it.

NCAR and the NAG graphics do not have their own mechanisms for opening graphics devices and SGS or GKS (see below) must be used to open a device before plotting can begin.

For line graphics which requires a style not catered for by PGPLOT or NCAR, the Simple Graphics System (SGS) should be used (SUN/85). This package is “lower level” than PGPLOT or NCAR (there are no facilities for drawing axes for example) but it does have excellent facilities for manipulating transformations and “organising” the display surface. SGS programs are also permitted to make direct calls to GKS (with a few documented restrictions) so the full power of GKS is available if SGS’s facilities are not adequate.

Programming purely with GKS is not advisable because getting even simple things done (e.g. opening a workstation) can be very long-winded and more easily achieved with SGS. Furthermore, achieving device independence is far from straight-forward; all the necessary facilities are available but making proper use of them requires an in-depth knowledge of GKS. Again, this is better left to SGS or higher level packages.

Programs requiring intimate interaction with an image display device should use the Image Display Interface (IDI—SUN/65) which is not based on GKS and offers access to facilities that are outside the GKS model of a graphics device. Currently the only device supported by IDI

is X-windows. The routines for opening and closing an IDI device in an ADAM program are described in SUN/65.

2.1 PGPLOT

All PGPLOT routines may be used in ADAM applications with the exception of the following routines, which must *never* be called in ADAM programs :

- PGBEG
- PGEND
- PGASK

The functions of `PGBEG` (and the equivalent obsolete routine `PGBEGIN`) are performed by the environment routine `PGP_ASSOC`. The environment routines `PGP_ANNUL`, `PGP_CANCL` and `PGP_DEACT` all call `PGEND`, but should be used as described in section 3.2 in order to be compatible with any future enhancements to PGPLOT which might separate the action of closing a workstation and closing down PGPLOT itself.

`PGASK` controls whether PGPLOT prompts the user for confirmation before clearing the screen of a graphics device; if used in an ADAM program PGPLOT will bypass the parameter system and attempt to read directly from the users terminal. So the "prompt state" must never be changed from its default state (`'OFF'` in ADAM programs). If prompting is required it is up to the application to keep track of when `PGPAGE` will clear the screen and issue its own prompt.

2.2 SGS

All SGS routines may be used in ADAM applications with the exception of the following routines, which must *never* be called in ADAM programs :

- SGS_INIT
- SGS_OPEN
- SGS_OPNWK
- SGS_CLOSE

The functions of `SGS_INIT`, `SGS_OPEN` and `SGS_OPNWK` are performed by the environment routine `SGS_ASSOC`. The environment routines `SGS_ANNUL` and `SGS_CANCL` dispose of resources used in SGS programs. `SGS_CLOSE` will be called by the environment level SGS de-activation routine `SGS_DEACT`.

2.3 GKS

Subroutines in the 'environment level' of the ADAM GKS system have names of the form GKS_x whereas true GKS routines have names of the form Gx.

All GKS routines may be used in ADAM applications with the exception of the following routines, which must *never* be called in ADAM applications:

- GOPKS
- GOPWK
- GCLWK
- GCLKS

The functions of GOPKS and GOPWK are performed by the environment level routine GKS_ASSOC. The function of GCLWK is performed by the routines GKS_ANNUL or GKS_CANCL. The function of GCLKS is performed by the routine GKS_DEACT.

3 Environment Level Routines

As with all other packages in ADAM, the only access to objects outside of application programs is via ADAM program parameters. The connection between graphics devices and the application program is controlled by means of a few environment level routines.

3.1 SGS routines

SGS has four environment level subroutines (the SGS 'parameter' or SGSPAR routines) which provide the necessary interaction with the outside world. They are :

Subroutine	Function
SGS_ASSOC	Associate a graphics workstation with a parameter and open it.
SGS_ANNUL	Close a graphics workstation without cancelling the parameter.
SGS_CANCL	Close a graphics workstation and cancel the parameter.
SGS_DEACT	De-activate ADAM SGS.

Here is a skeletal example of a program using SGS :

```
      SUBROUTINE SGSTEST( STATUS )
      ..
      INTEGER STATUS
      INTEGER ZONE

      ..
*     Obtain Zone on a graphics workstation
      CALL SGS_ASSOC( 'DEVICE', 'WRITE', ZONE, STATUS )
      IF( STATUS .EQ. SAI__OK ) THEN

         ..

*        Perform graphics operations on the zone
```

```
          ..

*       Release Zone
          CALL SGS_ANNUL( ZONE, STATUS )
        ENDIF

        ..

        CALL SGS_DEACT( STATUS )
```

SGS_ASSOC should be the first SGS routine to be called in the application. It obtains (via the parameter system) the name of the graphics workstation to be used, creates an initial SGS zone on the workstation, and returns an SGS zone identifier which can be used in subsequent SGS subroutine calls.

The first argument of SGS_ASSOC is an ADAM program parameter. It should be defined to be a graphics device parameter in the interface module for the application (see the example interface file in section4). The value of this parameter should be the name of a workstation as defined by GNS (see SUN/57).

The second argument is the access mode required. This can be one of :

'READ' The application is only going to 'read' from the workstation (*i.e.* perform cursor or similar operations). Screens will not be cleared when vdu workstations are opened.

'WRITE' The application is going to 'write' on the workstation, *i.e.* actually do some graphics. Workstations are completely initialized when they are opened.

'UPDATE' The application will modify the graphics on the workstation. Screens will not be cleared when vdu workstations are opened.

Note that the facility to prevent screen clearing is specific to RAL GKS and is not implemented for all workstations.

The third argument is the SGS zone identifier returned to the application.

The fourth argument is the usual status value. It follows the ADAM error strategy as described in SUN/104.

When the application has finished using the workstation, it should be closed using SGS_CANCL unless it is required to keep the parameter active (to update a global parameter for example), in which case SGS_ANNUL should be used.

When the application has finished using SGS, it should be de-activated by calling SGS_DEACT.

3.2 PGPLOT routines

PGPLOT has four environment level subroutines (the PGPLOT 'parameter' or PGPPAR routines) which provide the necessary interaction with the outside world. They are :

Subroutine	Function
PGP_ASSOC	Associate a graphics workstation with a parameter and open it.
PGP_ANNUL	Close a graphics workstation without cancelling the parameter.
PGP_CANCL	Close a graphics workstation and cancel the parameter.
PGP_DEACT	De-activate ADAM PGPLOT.

Here is a skeletal example of a program using PGPLOT :

```
      SUBROUTINE PGPTEST( STATUS )
      ..
      INTEGER STATUS
      INTEGER UNIT

      ..
*     Obtain Zone on a graphics workstation
      CALL PGP_ASSOC( 'DEVICE', 'WRITE', 1, 1, UNIT, STATUS )
      IF( STATUS .EQ. SAI__OK ) THEN

         ..

*        Perform graphics operations on the device

         ..

*        Release Device
         CALL PGP_ANNUL( UNIT, STATUS )
      ENDIF

      ..

      CALL PGP_DEACT( STATUS )
```

Note the close similarity with the SGS skeleton program. `PGP_ASSOC` should be the first PGPLOT routine called in the application. It obtains (via the parameter system) the name of the graphics workstation to be used, opens the workstation, and returns a PGPLOT unit number. PGPLOT only supports one graphics device open at one time so this unit number is always returned as one[1].

The first, second and last arguments of `PGP_ASSOC` are the same as the corresponding arguments of `SGS_ASSOC`.

The third and fourth arguments are the number of sub-plots per page in X and Y (c.f. `PGBEG`).

The fifth argument is the unit identifier returned to the application.

When the application has finished using the workstation, it should be closed using `PGP_CANCL` unless it is required to keep the parameter active (to update a global parameter for example), in which case `PGP_ANNUL` should be used.

When the application has finished using PGPLOT, it should be de-activated by calling `PGP_DEACT`.

[1] `PGBEG` has a similar redundant argument so that if, one day, PGPLOT is extended to support multiple devices, existing programs would not have to be changed.

3.3 GKS routines

GKS has five environment level subroutines (the GKS 'parameter' or GKSPAR routines). Four provide the necessary interaction with the outside world and one gives access to the GKS internal status. They are :

Subroutine	Function
GKS_ASSOC	Associate a graphics workstation with a parameter and open it.
GKS_ANNUL	Close a graphics workstation without cancelling the parameter.
GKS_CANCL	Close a graphics workstation and cancel the parameter.
GKS_DEACT	De-activate ADAM GKS.
GKS_GSTAT	Inquire whether GKS has reported an error (see Section 5).

Here is a skeletal example of a program using GKS :

```
      SUBROUTINE GKSTEST( STATUS )
      ..
      INTEGER STATUS
      INTEGER WKID

      ..
*     Obtain Workstation
      CALL GKS_ASSOC( 'DEVICE', 'READ', WKID, STATUS )
      IF( STATUS .EQ. SAI__OK ) THEN

         ..

*     Perform graphics operations on the workstation

         ..

*       Release Workstation
         CALL GKS_ANNUL( WKID, STATUS )
      ENDIF

      ..

      CALL GKS_DEACT( STATUS )
```

Note the close similarity with the SGS skeleton program. `GKS_ASSOC` should be the first GKS routine to be called in the application. It obtains (via the parameter system) the name of the graphics workstation to be used, opens it, and returns a GKS workstation identifier which can be used in subsequent GKS subroutine calls.

The notes in the previous section on the arguments of `SGS_ASSOC` apply equally well to `GKS_ASSOC` with the exception that the third argument is a GKS workstation identifier rather than an SGS zone identifier.

The workstation is closed using `GKS_ANNUL` or `GKS_CANCL` when the application has finished using it.

When the application has finished using GKS, it should be de-activated by calling GKS_DEACT.

4 Interface Files

A simple interface file for the above SGS example would be:

```
## SGSTEST - Test some SGS, PGPLOT or GKS graphics functions

interface SGSTEST

   parameter      DEVICE
      ptype       'DEVICE'
      position    1
      type        'GRAPHICS'
      access      'read'
      vpath       'prompt'
      help        'Name of workstation to be used'
      default     xwindows
   endparameter

endinterface
```

The same interface file can be used with the PGPTEST and GKSTEST programs if the line saying 'interface SGSTEST' is replaced by the line 'interface PGPTEST' or 'interface GKSTEST.

5 Error Handling

The PGPLOT, SGS and GKS parameter routines conform to the Starlink error reporting and inherited status strategy described in SUN/104. Error reports will generally be of the form:

```
ROUTINE_NAME: Text
```

but where there routine name is not useful, it will be omitted.

If it is required to test for particular status values, symbolic names should be used. They can be defined by including in the application, the statements:

```
INCLUDE 'SAE_PAR'   ! To define SAI__OK etc.
INCLUDE 'SGS_ERR'   ! To define the SGS__ error values
```

for SGS program, or the statements:

```
INCLUDE 'SAE_PAR'   ! To define SAI__OK etc.
INCLUDE 'PGP_ERR'   ! To define the GKS__ error values
```

for PGPLOT programs

```
      INCLUDE 'SAE_PAR'   ! To define SAI__OK etc.
      INCLUDE 'GKS_ERR'   ! To define the GKS__ error values
```

for GKS programs. It is of course possible to include PGPLOT, SGS and GKS status values in a single program.

The symbolic names are listed in Section 7.

A substitute GKS error handling routine is linked with ADAM applications so that error messages are reported in conformance with the Starlink error reporting strategy, however, GKS stand-alone routines do not operate the inherited status strategy. Thus programs which use GKS must be very careful about handling error conditions if they are to avoid generating streams of unhelpful error messages.

Subroutine `GKS_GSTAT` has been provided by the environment to help with this problem. If this routine is called after every non-environment GKS routine (or sequence of routine calls) then the net effect is as if the GKS routine(s) did have an inherited status argument.

Note that for stand alone GKS routines which do have a status argument, the environment status variable should *not* be used. Use a local variable, ignore its value and use `GKS_GSTAT` to get an environment status value for the routine. Liberal use of IF .. THEN .. ELSE .. ENDIF structures testing the environment status value is recommended to prevent invalid output from GKS and a large number of error messages.

Note also that the GKS enquiry routines do not report an error when they fail, they just return a status value (this is to allow the GKS error handling routine to make enquires about the state of GKS when handling an error without fear of triggering another error report and hence being called recursively). This means that an application that detects an error from an enquiry routine must compose and report a suitable message as well as setting the ADAM status

Example :

```
      SUBROUTINE GKSTEST(STATUS)

* Test GKS in ADAM.
* Draw a box and write some text in the box.

      INCLUDE 'SAE_PAR'
      INCLUDE 'GKS_PAR'
      INTEGER STATUS
      INTEGER WKID
      INTEGER N
      PARAMETER (N = 5)
      REAL X(N), Y(N)
      DATA X/0.0,0.0,1.0,1.0,0.0/,
     :     Y/0.0,1.0,1.0,0.0,0.0/

* Check STATUS on entry.
      IF ( STATUS .NE. SAI__OK ) RETURN

* Open GKS, open and activate workstation.
      CALL GKS_ASSOC( 'DEVICE', 'WRITE', WKID, STATUS )
      IF ( STATUS .EQ. SAI__OK ) THEN
```

```
* End of standard opening sequence.
*-----------------------------------------------------------------------
* Draw a box and write in it.
*   Polyline.
          CALL GPL( N, X, Y )
*   Set text alignment.
          CALL GSTXAL( GACENT, GACENT )
*   Set character height.
          CALL GSCHH( 0.04 )
*   Text.
          CALL GTX( 0.5, 0.5, 'Successful test of GKS 7.2' )
*-----------------------------------------------------------------------
* Check for error in GKS call sequence.
          CALL GKS_GSTAT( STATUS )

* Cancel parameter and annul the workstation id.
          CALL GKS_CANCL( 'DEVICE', STATUS )
       ENDIF

       CALL GKS_DEACT( STATUS )
       END
```

6 Compiling and Linking

Graphics programs are linked by including ‘`pgp_link_adam`‘, ‘`sgs_link_adam`‘ or ‘`gks_link_adam`‘ as appropriate on the alink command line.

7 Status Values

The following status values may be returned by the SGS parameter routines.

SGS__ILLAC	Illegal access mode
SGS__TOOZD	No more available zone descriptors
SGS__UNKPA	Parameter not found

The following status values may be returned by the PGPLOT parameter routines.

PGP__TOOZD	No more available unit descriptors
PGP__ILLAC	Illegal access mode
PGP__UNKPA	Parameter not found
PGP__ISACT	Parameter currently active
PGP__ERROR	An error has been reported from GKS itself
PGP__DVERR	Workstation name not defined by GNS

The following status values may be returned by the GKS parameter routines.

GKS__TOOZD	No more available graphics descriptors
GKS__ILLAC	Illegal access mode
GKS__UNKPA	Parameter not associated with workstation
GKS__ISACT	Parameter currently active
GKS__ERROR	An error has been reported from GKS itself
GKS__DVERR	Workstation name not defined by GNS

8 Routine descriptions

GKS_ANNUL
Close a graphics workstation without cancelling the parameter

Description:

De-activate and close the graphics workstation whose Workstation Identifier was obtained using GKS_ASSOC, and annul the Workstation Identifier. Do not cancel the associated parameter.

Invocation:

```
CALL GKS_ANNUL( WKID, STATUS )
```

Arguments:

WKID = INTEGER (Given)
A variable containing the Workstation Identifier.

STATUS = INTEGER (Given and returned)
The global status.

Notes:

On entry, the STATUS variable holds the global status value. If the given value of STATUS is SAI__OK and the routine fails to complete, STATUS will be set to an appropriate error number. If the given value of STATUS is not SAI__OK, then the routine will still attempt to execute and will return with STATUS unchanged.

GKS_ASSOC
Associate a graphics workstation with a parameter, and open it

Description:

Associate a graphics workstation with the specified Graphics Device Parameter and return a GKS Workstation Identifier to reference it. In GKS terms, the following actions occur – If GKS is not already open, it is opened and, if the workstation associated with the device parameter is not already open, that too is opened and activated.

Invocation:

```
CALL GKS_ASSOC ( PNAME, MODE, WKID, STATUS )
```

Arguments:

PNAME = CHARACTER∗(∗) (Given)
Expression specifying the name of a Graphics Device Parameter.

MODE = CHARACTER∗(∗) (Given)
Expression specifying the access mode - 'READ', 'WRITE' or 'UPDATE', as appropriate.

WKID = INTEGER (Returned)
A Variable to contain the Workstation Identifier.

STATUS = INTEGER (Given and returned)
The global status.

Notes:

The facility whereby update mode does not to clear the workstation is specific to RAL GKS and is not implemented for all workstations.

GKS_CANCL
Close graphics workstation and cancel the parameter

Description:

De-activate and close the graphics workstation associated with the specified Graphics Device Parameter and cancel the parameter. The workstation must have been opened using GKS_ASSOC.

Invocation:

```
CALL GKS_CANCL ( PNAME, STATUS )
```

Arguments:

PNAME = CHARACTER∗(∗) (Given)

Expression specifying the name of a Graphics Device Parameter.

STATUS = INTEGER (Given and returned)

The global status.

Notes:

On entry, the STATUS variable holds the global status value. If the given value of STATUS is SAI__OK and the routine fails to complete, STATUS will be set to an appropriate error number. If the given value of STATUS is not SAI__OK, then the routine will still attempt to execute and will return with STATUS unchanged.

GKS_DEACT
De-activate ADAM GKS

Description:

De-activate ADAM GKS after use by an application.

Invocation:

```
CALL GKS_DEACT ( STATUS )
```

Arguments:

STATUS = INTEGER (Given and returned)

The global status.

Notes:

On entry, the STATUS variable holds the global status value. If the given value of STATUS is SAI__OK and the routine fails to complete, STATUS will be set to an appropriate error number. If the given value of STATUS is not SAI__OK, then the routine will still attempt to execute and will return with STATUS unchanged.

GKS_GSTAT
Inquire whether GKS has reported an error

Description:

Check whether a GKS error has been reported since the last time this routine was called. If it has, set STATUS to GKS__ERROR, and cancel the GKS error state.

Invocation:

```
CALL GKS_GSTAT ( STATUS )
```

Arguments:

STATUS = INTEGER (Given and returned)

The global status.

Notes:

The global status will be set to GKS__ERROR if GKS has reported an error; otherwise it will be unchanged. If the given value of STATUS is not SAI__OK, then the routine will return without action.

PGP_ANNUL
Close a graphics workstation without cancelling the parameter

Description:

Close PGPLOT, and annul the unit identifier. Do not cancel the associated parameter.

Invocation:

```
CALL PGP_ANNUL ( UNIT, STATUS )
```

Arguments:

UNIT = INTEGER (Given and returned)
A variable containing the PGPLOT unit number

STATUS = INTEGER (Given and returned)
The global status.

Notes:

On entry, the STATUS variable holds the global status value. If the given value of STATUS is SAI__OK and the routine fails to complete, STATUS will be set to an appropriate error number. If the given value of STATUS is not SAI__OK, then the routine will still attempt to execute and will return with STATUS unchanged.

PGP_ASSOC
Associate a graphics workstation with a parameter, and open it

Description:

Associate a graphics workstation with the specified Graphics Device Parameter and return an PGP unit identifier

Invocation:

```
CALL PGP_ASSOC ( PNAME, MODE, NX, NY, UNIT, STATUS )
```

Arguments:

PNAME = CHARACTER∗(∗) (Given)
Expression specifying the name of a graphics parameter.

MODE = CHARACTER∗(∗) (Given)
Expression specifying the access mode. If it is 'READ' or 'UPDATE', a vdu workstation screen will not be cleared when the workstation is opened. If it is 'WRITE', the screen will be cleared as usual.

NX = INTEGER (Given)
Number of sub-plots in X.

NY = INTEGER (Given)
Number of sub-plots in Y.

UNIT = INTEGER (Returned)
A Variable to contain the unit identifier.

STATUS = INTEGER (Given and returned)
The global status.

Notes:

PGPLOT currently only supports one workstation so the unit is always returned as one.

PGP_CANCL
Close a graphics workstation and cancel the parameter

Description:

Close the graphics workstation associated with the specified graphics device parameter and cancel the parameter. The workstation must have been opened using PGP_ASSOC.

Invocation:

```
CALL PGP_CANCL ( PNAME, STATUS )
```

Arguments:

PNAME = CHARACTER∗(∗) (Given)

Expression specifying the name of a Graphics Device Parameter.

STATUS = INTEGER (Given and returned)

The global status.

Notes:

On entry, the STATUS variable holds the global status value. If the given value of STATUS is SAI__OK and the routine fails to complete, STATUS will be set to an appropriate error number. If the given value of STATUS is not SAI__OK, then the routine will still attempt to execute and will return with STATUS unchanged.

PGP_DEACT
De-activate ADAM PGPLOT

Description:

De-activate ADAM PGPLOT after use by an application.

Invocation:

```
CALL PGP_DEACT ( STATUS )
```

Arguments:

STATUS = INTEGER (Given and returned)

The global status.

Notes:

On entry, the STATUS variable holds the global status value. If the given value of STATUS is SAI__OK and the routine fails to complete, STATUS will be set to an appropriate error number. If the given value of STATUS is not SAI__OK, then the routine will still attempt to execute and will return with STATUS unchanged.

SGS_ANNUL
Close a graphics workstation without cancelling the parameter

Description:

De-activate and close the graphics workstation whose base zone (ZONE) was obtained using SGS_ASSOC, and annul the zone identifier. Do not cancel the associated parameter.

Invocation:

```
CALL SGS_ANNUL ( ZONE, STATUS )
```

Arguments:

ZONE = INTEGER (Given and returned)
A variable containing the base zone identifier.

STATUS = INTEGER (Given and returned)
The global status.

Notes:

On entry, the STATUS variable holds the global status value. If the given value of STATUS is SAI__OK and the routine fails to complete, STATUS will be set to an appropriate error number. If the given value of STATUS is not SAI__OK, then the routine will still attempt to execute and will return with STATUS unchanged.

SGS_ASSOC
Associate a graphics workstation with a parameter, and open it

Description:

Associate a graphics workstation with the specified Graphics Device Parameter and return an SGS zone identifier to reference the base zone of the workstation.

Invocation:

```
CALL SGS_ASSOC ( PNAME, MODE, ZONE, STATUS )
```

Arguments:

PNAME = CHARACTER∗(∗) (Given)

Expression specifying the name of a graphics parameter.

MODE = CHARACTER∗(∗) (Given)

Expression specifying the access mode. If it is 'READ' or 'UPDATE', a vdu workstation screen will not be cleared when the workstation is opened. If it is 'WRITE', the screen will be cleared as usual.

ZONE = INTEGER (Returned)

A Variable to contain the Zone identifier.

STATUS = INTEGER (Given and returned)

The global status.

Notes:

The facility whereby update mode does not to clear the workstation is specific to RAL GKS and is not implemented for all workstations.

SGS_CANCL
Close a graphics workstation and cancel the parameter

Description:

De-activate and close the graphics workstation associated with the specified graphics device parameter and cancel the parameter. The workstation must have been opened using SGS_ASSOC.

Invocation:

```
CALL SGS_CANCL ( PNAME, STATUS )
```

Arguments:

PNAME = CHARACTER∗(∗) (Given)

Expression specifying the name of a Graphics Device Parameter.

STATUS = INTEGER (Given and returned)

The global status.

Notes:

On entry, the STATUS variable holds the global status value. If the given value of STATUS is SAI__OK and the routine fails to complete, STATUS will be set to an appropriate error number. If the given value of STATUS is not SAI__OK, then the routine will still attempt to execute and will return with STATUS unchanged.

SGS_DEACT
De-activate ADAM SGS

Description:

De-activate ADAM SGS after use by an application.

Invocation:

```
CALL SGS_DEACT ( STATUS )
```

Arguments:

STATUS = INTEGER (Given and returned)

The global status.

Notes:

On entry, the STATUS variable holds the global status value. If the given value of STATUS is SAI__OK and the routine fails to complete, STATUS will be set to an appropriate error number. If the given value of STATUS is not SAI__OK, then the routine will still attempt to execute and will return with STATUS unchanged.