GFM

Graphics Flow Mark

Language Reference Guide

Version 1

Graphics Flow Mark Language Reference 1.0

PUBLISHED BY

Graphic Prepress Solutions

This document is provided for developers who wants to write GFM programs and scripts or to use GFM as a part of their project.

INTRODUCTION

This guide is an alphabetical reference for the GFM script language. It should be used together with the *GFM Programmer's Guide*. It also contains a reference over the Baseline properties when conneting to modules with GFM and the procedures and variables defined in the library STANDARD.LIB.

Note that all commands, variables, procedures and properties are case sensitive in GFM.

!Action property

Syntax *value* **!Action**

Applies to Standard and Advanced Edition

Description Sets the Action property of the connected module.

Remarks The type of action to be done is determined by the *value* according to the following table:

Value	Meaning
1	Show configuration menu. The configuration file to load must be defined as SourceFileName. If SourceFileName is defined as 'New File' will a new empty configuration be created. In that case must the file name to create be DestinationFileName.
2	Not used as any default action. (Convert to OIF or Start Command.)
3	Not used as any default action. (Convert from OIF or Stop Command)
4	Not used as any default action. (Do a filecheck)
5	Run the configure module command. Will load a configuration file as defined in the SourceFileName property.

The command depends on a set of other commands and the type of module. For more information about this topic see *Chapter 6 Programming Modules With GFM* in the *GFM Programmer's Guide*.

Note! User Defined Modules (UDM) can have any setting for the !Action property. How to use the !Action property for UDM:s see the documentation for that module.

See also connectmodule, !AllowConfigure, !Direction, !SourceFileName, !DestinationFileName, >Moduletype

Example This example connects to the TIFF module and converts an image called C:\IMAGE1.TIF to an image called C:\IMAGE1.OIF

```
(GFMTIFFModule.TIFF) connectmodule
1 !Direction
(C:\IMAGE1.TIF) !SourceFileName
(C:\IMAGE1.OIF) !DestinationFileName
2 !Action
```

Errors no module connected, stack underflow, typecheck

add command

Syntax *value1 value2* **add** *result*

Applies to All Edtions

Description Adds the two top values on the stack and pushes the result back to the stack. *Value1* and v*alue2* will be removed from the stack

Remarks The two top values can be any number. If any of the two top values are a string, the Typecheck error will occur.

See also div, mul, sub

Example The following example pushes two values to the stack and adds them.

```
10 20 add
```

Errors stack underflow, typecheck

>AllowConfigure property

Syntax >**AllowConfigure**

Applies to Standard and Advanced Edition

Description Returns an integer value indicating if the connected module can be configured or not.

Remarks The value returned will be on top of the stack after the command has been sent to the module. The value has the following meaning:

Value	Meaning
0	The module cannot be configured
1	The module can be configured

The command depends on a set of other commands and the type of module. For more information about this topic see *Chapter 6 Programming Modules With GFM* in the *GFM Programmer's Guide*.

See also connectmodule, !AllowConfigure, !Direction

Example This example connects to the TIFF module and checks if the module can be configured

```
(GFMTIFFModule.TIFF) connectmodule
>AllowConfigure
```

Errors stack underflow, typecheck

>AllowedFileType property

Syntax >**AllowedFileType**

Applies to Standard and Advanced Edition

Description Returns an integer value indicating if the connected module can handle a specified file or not.

Remarks The value returned will be on top of the stack after the command has been sent to the module. Before the command can be used the filename must be in !**SourceFileName** and the !**Action** must be set to 4. The value returned has the following meaning:

Value	Meaning
0	The module cannot handle the file.
1	The module can handle the file

The command depends on a set of other commands and the type of module. For more information about this topic see *Chapter 6 Programming Modules With GFM* in the *GFM Programmer's Guide*.

See also connectmodule, !SourceFileName, !Action

Example This example connects to the TIFF module and checks if the file C:\FLOWER.TIF can be handled by the module.

```
(GFMTIFFModule.TIFF) connectmodule
(C:\FLOWER.TIF) !SourceFileName
4 !Action
>AllowedFileType
```

Errors no connected module

---

aload command

Syntax *array* **aload** $array_{n-1}$ … $array_0$

Applies to All editions

Description Pushes all *n* ellements of *array* to the stack. *array* will be removed from the stack.

See also astore, get

Example The following examples shows how the use the aload command:

```
[1 2 3] -> 3 2 1
[(GFM) 1 (aload)] -> (aload) 1 (GFM)
```

Errors stack underflow, typecheck

---

and command

Syntax *value1 value2* **and** *result*

Applies to All editions

Description Performs a logical AND operation of the two top values on the stack. The *value1* and v*alue2* will be removed from the stack

Remarks The two top values can be any number. If any of the two top values are a string, will the typecheck error occur.

See also not, or, xor

Example The following examples does a logical AND comparsion:

```
99 1 and -> 1
52 7 and -> 4
```

Errors stack underflow, typecheck

---

appendtofile command

Syntax *filename string* **appendtofile**

Applies to All editions

Description Appends the string *string* to the file *filename*.

Remarks If the file *filename* doesn’t exist, it will automatically be created.

See also deletefile

Example This example appends the string %%Creator: Graphics Flow Mark to the file C:\GFM\TEST.PS.

```
(C:\GFM\TEST.PS)
(%%Creator: Graphics Flow Mark)
appendtofile
```

Errors stack underflow, typecheck

AssignHeaderToVariables procedure

Syntax *imageheader* **AssignHeaderToVarables**

Applies to All editions

Description Sets the *imageheader* to variables that will be stored in GFMDict.

Remarks The procedure is in the STANDARD.LIB library and should always be used since many commands uses these varibles.

See also PushImageHeaderToStack

Example This example saves the image in the PixelArea to disk using the **PushImageHeaderToStack** procedure as C:\IMAGE1.OIF and closes the file.

```
(C:\GFM\LIBS\STANDARD.LIB) run
(C:\IMAGES\IMAGE1.OIF) dup openimagefile getimageheader
AssignHeaderToVariables
```

BitsPerSample variable

Syntax **BitsPerSample**

Applies to Standard and Advanced Edition

Description Contains the number of bits per sample for the current image in the PixelArea.

Remarks The *BitsPerSample* can in this version be 1 or 8. The PixelArea is already prepared for 16 bits per sample but this has not been implemented fully in the GFM language in this version.

---

char command

Syntax *value* **char**

Applies to All editions

Description Returns the character from the ASCII code value to the top of the stack.

Example This will return the string A to the top of the stack since 65 is the character code for A.

```
65 char
```

Errors stack underflow, typecheck

---

closeallfiles command

Syntax **closeallfiles**

Applies to All editions

Description Closes all opened files.

See also closeimagefile

Example This example saves the image in the PixelArea to disk using the **PushImageHeaderToStack** procedure as C:\IMAGE1.OIF and closes the file.

```
C:\IMAGE1.OIF)
PushImageHeaderToStack
writeimagefile
closeallfiles
```

---

closedb command

Syntax **closedb**

Applies to Standard and Advanced Edition

Description Closes the current database.

See also opendb

Example This example closes the current database.

```
closedb
```

Errors no current database

---

closefile command

Syntax *filehandle* **closefile**

Applies to All editions

Description Closes the file refered to as *filehandle*. The closefile will write all bufferered data to the file before closing it

See also closeallfiles, file

Example This example opens the file C:\IMAGES\STATUS.LOG, reads the first line and then closes the file.

```
C:\IMAGES\STATUS.LOG) (r) file
/F exch def
F readline /FirstLine exch def
F closefile
```

---

closeimagefile command

Syntax **closeimagefile**

Applies to Standard and Advanced Edition

Description Closes the current image file.

See also closeallfiles

Example This example saves the image in the PixelArea to disk using the **PushImageHeaderToStack** procedure as C:\IMAGE1.OIF and closes the file.

```
C:\IMAGE1.OIF)
PushImageHeaderToStack
writeimagefile
closeimagefile
```

---

cmd command

Syntax *string* **cmd**

Applies to All editions

Description Runs the *string* operating system command.

Remarks This command is not very well implemented in GFM. There is no error check on this command. The command is also asynchronous, which means that you have no control when starting this command.

See also closeallfiles

Example This example run windows notepad from a GFM script.

```
(notepad) cmd
```

ColorMode variable

Syntax **ColorMode**

Applies to Standard and Advanced Edition

Description Contains the color mode, or color space, for the current image in the PixelArea.

Remarks The *ColorMode* variable has the following meaning:

Value	Meaning
0	Bitmap
1	Monochrome
2	Palette (Not supported in this version)
3	RGB
4	CMYK
5	Multi-Channel (Not supported in this version)

Compression variable

Syntax **Compression**

Applies to Standard and Advanced Edition

Description Contains the compression for the current image in the PixelArea.

Remarks The *Compression* variable has the following meaning:

Value	Meaning
0	No compression
1	LZW
2	CCITT Group 3
3	CCITT Group 4
4	JPEG
5	Run-Length (Only if ColorMode is set to 0)

Most images do not need any compression since OIF is only an interchange format and it is faster not to use any compression.

---

connectmodule command

Syntax m*odulename* **connectmodule**

Applies to Standard and Advanced Edition

Description Connects to the module *modulename*.

Remarks You must always connect to a module before you can use it. The command starts the module as a process and creates a thread to the process. After the command has been executed successfully, the module will be available for the GFM engine.

The command will also change a string variable in GFMDict called **ConnectedModule**. This variable will contain the name of the connected module. If there is no module connected, the variable will be an empty string. The variable will automatically be created when the GFM engine starts.

You can only have one module connected at the same time.

For more information about this topic see *Chapter 6 Programming Modules With GFM* in the *GFM Programmer's Guide*.

See also disconnectmodule

Example This example connects to the TIFF module and checks if the module can be configured.

```
(GFMTIFFModule.TIFF) connectmodule
```

Errors stack underflow, typecheck, module already connected

ConnectedModule variable

Syntax **ConnectedModule**

Applies to Standard and Advanced Edition

Description Contains the name of the connected module.

Remarks The name of the connected module will be stored in this variable. If no module has been connected, the variable will be an empty string. The variable is defined in STANDARD.LIB. After the library has been loaded, the variable will be an empty string.

---

converttogray command

Syntax **converttogray**

Applies to Advanced Edition

Description Converts the image loaded into the Pixel Area to a GrayMode image. I.e. the RGBMode or CMYKMode image that is loaded before the command will be erased from PixelArea and replaced by a GrayMode image.

Remarks The loaded image must be an RGBMode or CMYKMode image. The relation between how colours are converted to GrayMode are determined by the **graycomponents** command.

The command also changes the ColorMode parameter in GFMDict. This means that ColorMode MUST be present in GFMDict. This is done by loading the STANDARD.LIB library.

The command does not change the resolution or the image dimension, i.e. the only parameter that will be changed is the ColorMode.

See also graycomponents, loadimage

Example This example loads an image called C:\IMAGES\IMAGE1.OIF to the Pixel Area, converts the image to GrayMode and saves the new image as C:\IMAGES\IMAGE1GRAY.OIF.

```
(C:\GFM\LIBS\STANDARD.LIB) run
(C:\IAMEGS\IMAGE1.OIF) dup openimagefile getimageheader
AssignHeaderToVariables
(Processing Image: ) exch mergestrings print
ViewColorMode
ColorMode 1 {(Image is already GrayMode) print endjob} if
ColorMode 3 {ImageHeight ImageWidth 3 mul mul} if
ColorMode 4 {ImageHeight ImageWidth 4 mul mul} if
(Loading Image...) print
loadimage
(Converting to GrayMode) print
converttogray
PushImageHeaderToStack
(C:\IMAGES\IMAGE1GRAY.OIF)
(Writing image file) print
writeimagefile
(Closing files) print
closeimagefile
```

Errors cannot convert. the image is not an RGB or CMYK image, no image loaded.

---

converttocmyk command

Syntax **converttocmyk**

Applies to Advanced Edition

Description Converts the image loaded into the Pixel Area to a CMYKMode image. I.e. the RGBMode or GrayMode image that is loaded before the command will be erased from memory and replaced by a CMYKMode image.

Remarks The loaded image must be an RGBMode or GrayMode image. The relation between how colors are converted to CMYKMode are determined by the **setblackgeneration, setmaxblackink** and the **setundercolorremoval** commands.

The command also changes the ColorMode parameter in GFMDict. This means that ColorMode MUST be present in GFMDict. This is done by using the STANDARD.LIB library.

The command does not change the resolution or the image dimension, i.e. the only parameter that will be changed is the ColorMode.

See also setmaxblackink and setundercolorremoval

Example This example loads an image called C:\IMAGES\IMAGE1.OIF to the Pixel Area, converts the image to CMYKMode and saves the new image as C:\IMAGES\IMAGE1CMYK.OIF.

```
(C:\GFM\LIBS\STANDARD.LIB) run
(C:\IAMEGS\IMAGE1.OIF) dup openimagefile getimageheader
AssignHeaderToVariables
```

```
(Processing Image: ) exch mergestrings print
ViewColorMode
ColorMode 1 {ImageHeight ImageWidth 1 mul mul} if
ColorMode 3 {ImageHeight ImageWidth 3 mul mul} if
ColorMode 4 {(Image is already CMYKMode) print endjob} if
(Loading Image...) print
loadimage
(Converting to CMYKMode) print
converttocmyk
PushImageHeaderToStack
(C:\IMAGES\IMAGE1GRAY.OIF)
(Writing image file) print
writeimagefile
(Closing files) print
closeimagefile
```

Errors cannot convert. the image is not an RGBMode or GrayMode image.

---

converttorgb command

Syntax **converttorgb**

Applies to Advanced Edition

Description Converts the image loaded into the Pixel Area to an RGBMode image. I.e. the GrayMode or CMYKMode image that is loaded before the command will be erased from PixelArea and replaced by a RGBMode image.

Remarks The loaded image must be an CMYKMode or GrayMode image.

The command also changes the ColorMode parameter in GFMDict. This means that ColorMode MUST be present in GFMDict. This is done by using the STANDARD.LIB library.

The command does not change the resolution or the image dimension, i.e. the only parameter that will be changed is the ColorMode.

See also converttocmyk, converttogray

Example This example loads an image called C:\IMAGES\IMAGE1.OIF to the Pixel Area, converts the image to RGBMode and saves the new image as C:\IMAGES\IMAGE1CMYK.OIF.

```
(C:\GFM\LIBS\STANDARD.LIB) run
(C:\IAMEGS\IMAGE1.OIF) dup openimagefile getimageheader
AssignHeaderToVariables
(Processing Image: ) exch mergestrings print
ViewColorMode
ColorMode 1 { ImageHeight ImageWidth 1 mul mul} if
ColorMode 3 {(Image is already RGBMode) print endjob } if
ColorMode 4 {ImageHeight ImageWidth 4 mul mul} if
(Loading Image...) print
loadimage
(Converting to GrayMode) print
converttorgb
PushImageHeaderToStack
(C:\IMAGES\IMAGE1GRAY.OIF)
(Writing image file) print
writeimagefile
(Closing files) print
closeimagefile
```

Errors cannot convert. the image is not a GrayMode or CMYKmode image, no image loaded

copyfile command

Syntax *sourcefilename destinationfilename* **copyfile**

Applies to All editions

Description Copies the *sourcefilename* to *destinationfilename*.

Remarks If you try to use the **copyfile** command on a currently opened file, an error occurs. An error will also occur if the *sourcefilename* does not exist or if the *destinationfilename* exists. If the destinationpath, the *destinationfilename* without the file name, doesn't exist, an error will also occur.

See also deletefile, movefile

Example This example copies a file C:\IMAGES\IMAGE1 to the D:\IMAGEBACKUP\IMAGE1

```
(C:\IMAGES\IMAGE1) (D:\IMAGEBACKUP\IMAGE1) copyfile
```

Errors stack underflow, typecheck, file not found.

count command

Syntax **count**

Applies to All editions

Description Counts the number of items on the stack and pushes the result on top of the stack.

Example This example pushes three values onto the stack and counts them.

```
100 200 300 count
```

createdirectory command

Syntax n*ame* **createdirectory**

Applies to All editions

Description Creates the directory *name* at specified path.

Remarks The n*ame* must be a valid path and an absolute path.

See also deletedirectory

Example This example creates a directory called IMAGES in in the C:\GFM directory.

```
(C:\GFM\IMAGES) createdirectory
```

cval command

Syntax *string* **cval** *number*

Applies to All editions

Description Converts the top string value on the stack to number.

Remarks The cval command stops reading the string at the first character it can't recognize as part of a number. Symbols and characters that are often considered parts of numeric values, such as dollar signs and commas, are not recognized. However, the function recognizes radix prefixes &O (for octal) and &H (for hexadecimal). Blanks, tabs, and linefeeds are stripped from the argument.

The cval command recognises only the period (.) as a valid decimal separator.

See also abs, cvs

Example This example converts the result of a division to an integer.

```
(200) cval
```

Errors stack underflow, typecheck

---

cvi command

Syntax *real* **add** *int*

Applies to All editions

Description Converts the top value on the stack to an integer

Remarks The top value can be an integer or real.

See also abs, cvs

Example This example converts the result of a division to an integer.

```
100 283 div cvi
```

Errors stack underflow, typecheck

---

cvs command

Syntax *value* **add** *string*

Applies to All editions

Description Converts the top value to a string.

Remarks The top value can be an integer or real. The command can be useful when printing a text in the system monitor using the **print** command and printing numbers.

See also cvi

Example This example converts a number to a string and prints the string in the system monitor.

```
100 283 div cvs print
```

Errors stack underflow, typecheck

---

cvx command

Syntax *string* **cvx** *executable*

Applies to All editions

Description Converts the top string on the stack to an executable.

Remarks The top value must be a string.

Example This example converts the string (/Loop (2:1) def) to a executable.

```
(/Loop (2:1) def) cvx
```

Errors stack underflow, typecheck

---

date command

Syntax **date**

Applies to All editions

Description Pushes the value of the current date to the top of the stack. If the date is below 10 will it contain leading zeros. Note that the date will be a string value.

See also month, time, year

---

dbaddrecord command

Syntax **dbaddrecord**

Applies to Standard and Advanced Edition

Description Prepares the current table to add a new record.

Remarks If a current table or a current record doesn't exist, an error will occur.

See also opendb, dbsettable, dbnewfield

Example This example adds a new field to the tblPage table in the C:\PROGRAM\PAGEPAIR.MDB database.

```
(C:\PROGRAM\PAGEPAIR.MDB) opendb
(tblPage) dbsettable
dbaddrecord
(sProductName) (Aftonbladet) dbnewfield
(sPageName) AFileName dbnewfield
(iPageNumber) 9 dbnewfield
(iStatus) 2 dbnewfield
(sColor) (B) dbnewfield
(iColor) 1 dbnewfield
(dateArrival) date dbnewfield
(timeArrival) time dbnewfield
dbupdate
closedb
```

Errors no current database, no current record, no current table, stack underflow, typecheck.

---

dbfind command

Syntax *value* **dbfind**

Applies to Standard and Advanced Edition

Description Searches for the value in the current recordset index.

Remarks If a current table or a current record doesn't exist, an error will occur.

See also opendb, dbsettable, dbsetindex, dbfind

Example This example searches for the value C:\IMAGE1.TIF in the current table.

```
(C:\IMAGE1.TIF) dbfind
```

Errors no current database, no current record, no current table, stack underflow, typecheck.

---

dbgetfield command

Syntax *fieldname* **dbgetfield** *value*

Applies to Standard and Advanced Edition

Description Reads the value of *fieldname* of the current table to top of the stack.

Remarks If a current table or a current record doesn't exist, an error will occur.

See also opendb, dbsettable, dbsetindex, dbfind

Example This example reads the sPageName field of the current record.

```
(sPageName) dbgetfield
```

Errors no current database, no current record, no current table, stack underflow, typecheck.

---

dbnewfield command

Syntax *fieldname value* **dbnewfield**

Applies to Standard and Advanced Edition

Description Adds a new *value* to the *fieldname* of the current table.

Remarks The new field must have been created with the dbaddrecord command before this command can be executed.

See also opendb, dbsettable, dbaddrecord, dbupdate

Example This example adds a new field to the tblPage table in the C:\PROGRAM\PAGEPAIR.MDB database.

```
(C:\PROGRAM\PAGEPAIR.MDB) opendb
(tblPage) dbsettable
dbaddrecord
(sProductName) (Aftonbladet) dbnewfield
(sPageName) AFileName dbnewfield
(iPageNumber) 9 dbnewfield
(iStatus) 2 dbnewfield
(sColor) (B) dbnewfield
(iColor) 1 dbnewfield
(dateArrival) date dbnewfield
(timeArrival) time dbnewfield
dbupdate
closedb
```

Errors no current database, no current record, no current table, stack underflow, typecheck.

dbsetindex command

Syntax *fieldname* **dbsetindex**

Applies to Standard and Advanced Edition

Description Sets the index for the current table.

Remarks The command is used in combination with the dbfind command to locate recordsets.

See also opendb, dbsettable, dbfind

Example This example sets the index to sPageName.

```
(sPageName) dbsetindex
```

Errors no current database, no current record, no current table, stack underflow, typecheck.

dbsettable command

Syntax *tablename* **dbsettable**

Applies to Standard and Advanced Edition

Description Sets the *tablename* as the current table.

See also opendb

Example This example sets the current table to tblPage.

```
(tblPage) dbsettable
```

Errors no current database, stack underflow, table not found, typecheck.

dbupdate command

Syntax **dbupdate**

Applies to Standard and Advanced Edition

Description Updates the current table after a new record has been added.

Remarks If a current table or a current record doesn't exist, an error will occur.

See also opendb, dbsettable, dbnewfield, dbaddrecord

Example This example adds a new filed to the tblPage table in the C:\PROGRAM\PAGEPAIR.MDB database.

```
(C:\PROGRAM\PAGEPAIR.MDB) opendb
(tblPage) dbsettable
dbaddrecord
(sProductName) (Aftonbladet) dbnewfield
(sPageName) AFileName dbnewfield
(iPageNumber) 9 dbnewfield
(iStatus) 2 dbnewfield
(sColor) (B) dbnewfield
(iColor) 1 dbnewfield
(dateArrival) date dbnewfield
(timeArrival) time dbnewfield
dbupdate
```

```
closedb
```

Errors no current database, no current record, no current table, stack underflow, typecheck.

---

dbupdatefield command

Syntax *fieldname value* **dbupdatefield**

Applies to Standard and Advanced Edition

Description Changes the value of the current field of the current table.

Remarks If a current table or a current record doesn't exist, an error will occur.

See also opendb, dbsettable, dbsetindex, dbfind

Example This example changes the iStatus field of the current record.

```
(iStatus) 1 dbupdatefield
```

Errors no current database, no current record, no current table, stack underflow, typecheck.

---

def command

Syntax *name value* **def**

Applies to All editions

Description Associates the *name* with *value* in the GFMDict. The value can also be a procedure. If the value already exists in GFMDict, the value will be changed.

Remarks If you are defining a procedure, the syntax of the code in the procedure will not be checked when you define the procedure. If there is an error in the procedure, the error will occur at the first call to the procedure.

Example This example defines the name NewResolution to the value 72.

```
/NewResolution 72 def
```

This example defines the name Add3 to be a procedure.

```
/Add3 {3 add} def
```

Errors stack underflow, typecheck

>DefaultExtension property

Syntax >**DefaultExtension**

Applies to Standard and Advanced Edition

Description Returns a string variable indicating the current modules default extension.

Remarks This command is useful when the direction of the module is set to be an OUT module and you don't have any extension for the filename.

The command depends on a set of other command and the type of module. For more information about this topic see *Chapter 6 Programming Modules With GFM* in the *GFM Programmer's Guide*.

See also !Action, connectmodule, !Direction, !SourceFileName, !DestinationFileName

Example This example connects to the TIFF module and sets the direction to be an OUT module, uses the default extension for the TIFFModule (.TIF) and converts the file to a TIFF file.

```
(GFMTIFFModule.TIFF) connectmodule
2 !Direction
1 getglobalstring !SourceFileName
>DefaultExtension
(C:\FLOWER) exch mergestrings !DestinationFileName
3 !Action
```

Errors stack underflow, typecheck

---

deletefile command

Syntax *filename* **deletefile**

Applies to All editions

Description Deletes the file *filename*.

Remarks There is no warning that a file will be deleted. If *filename* can't be found, an error will occur. An error will also occur if *filename* is opened or write proteced. **deletfile** also supports wildcharacters as * and ?.

See also copyfile, movefile

Example This example deletes a file called C:\FLOWER.OIF.

```
(C:\FLOWER.OIF) deletefile
```

Errors stack underflow, typecheck, file *filename* not found, file *filename* is write protected

---

!DestinationFileName property

Syntax **!DestinationFileName** or **>DestinationFileName**

Applies to Standard and Advanced Edition

Description Sets or reads the modules destination filename.

Remarks The meaning of the **!DestinationFileName** depends on **!Action**, **>ModuleType** and **!Direction** properties of the connected module. If the module has been configured as an IN module, the **!DestinationFileName** will be the .OIF filename. If the module has been configured as an OUT module, the **!DestinationFileName** will be the modules filetype filename.

The command depends on a set of other command and the type of module. For more information about this topic see *Chapter 6 Programming Modules With GFM* in the *GFM Programmer's Guide*.

See also !Action, connectmodule, !Direction, >ModuleType, !SourceFileName

Example This example connects to the TIFF module and sets the direction to be an OUT module uses the default extension for the TIFFModule (.TIF) and converts a file from GlobalString 1 to a file called C:\FLOWER.TIF.

```
(GFMTIFFModule.TIFF) connectmodule
2 !Direction
```

```
1 getglobalstring !SourceFileName
>DefaultExtension
(C:\FLOWER) exch mergestrings !DestinationFileName
3 !Action
```

Errors stack underflow, typecheck

---

dev_getstack command

Syntax **dev_getstack**

Description Returns the contents of the stack to the GFM Developer Tool.

Remarks Never use this command in a GFM script.

See also dev_getGFMDict

---

dev_getGFMDict command

Syntax **dev_GFMDict**

Description Returns the contents of the GFMDict to the GFM Developer Tool.

Remarks Never use this command in a GFM script.

See also dev_getstack

!Direction, >Direction property

Syntax *value* **!Direction** or **>Direction**

Applies to Standard and Advanced Edition

Description Sets or reads the modules direction.

Remarks With the direction, you will set the module as an IN or OUT module.

Value	Meaning
1	The module is set as an IN module.I.e the module shall convert to an OIF file
2	The module is set as an OUT module.I.e the module shall convert from an OIF to the modules internal format.

The command depends on a set of other command and the type of module. For more information about this topic see *Chapter 6 Programming Modules With GFM* in the *GFM Programmer's Guide*.

See also connectmodule, !AllowConfigure, !Direction, !SourceFileName, !DestinationFileName

Example This example connects to the TIFF module and sets the direction to be an IN module.

```
(GFMTIFFModule.TIFF) connectmodule
1 !Direction
```

Errors stack underflow, typecheck

disconnectmodule command

Syntax **disconnectmodule**

Applies to Standard and Advanced Edition

Description Disconnects the currently connected module.

Remarks You must always disconnect a module before connecting to another. You cannot have two modules connected at the same time.

For more information about this topic see *Chapter 6 Programming Modules With GFM* in the *GFM Programmer's Guide*.

See also connectmodule

Example This example connects to the TIFF module and immediate disconnects.

```
(GFMTIFFModule.TIFF) connectmodule disconnectmodule
```

Errors no module connected, stack underflow, typecheck

div command

Syntax *value1 value2* **div** *result*

Applies to All editions

Description Divides the two top values on the stack and pushes the result back to the stack. *Value1* and *value2* will be removed from the stack.

Remarks The two top values can be any number. If any of the two top values are a string a typecheck error will occur. The division is *value2/value1*.

See also add, mul, sub

Example This example divides 100 with 200, i.e. the stack will contain 0.5

```
100 200 div
```

Errors stack underflow, typecheck

dup command

Syntax **dup**

Applies to All rditions

Description Duplicates the top value on the stack.

Example This example pushes the string "This is a string" to the stack and duplicates it.

```
(This is a string) dup
```

Errors stack underflow

endjob command

Syntax **endjob**

Applies to All editions

Description Abort the current GFM script uncondional.

Remarks Can be used in **if** structures to do condional breaks.

Example This example is an **if** structure with an **endjob** command.

```
ColorMode 1 {(Image is already GrayMode) print endjob} if
```

eq command

Syntax *value1 value2* **eq** *boolean*

Applies to All editions

Description Checks if the two top values on the stack are equal or not.

Remarks The command will pop two objects from the stack and push the value True to the stack if they are equal, False if not. The two top values can be any number or string.

See also ge, gt, le, lt

Example This example will check if a value is odd or even. If the value is odd will it return False if even True.

```
3 2 div
3 2 div cvi
eq
```

Errors stack underflow

exch command

Syntax *value1 value2* **exch** *value2 value1*

Applies to All editions

Description Exchanges the two top values on the stack.

Remarks The two top values can be any number or string.

See also dup, pop

Example This example pushes one number and one string to the stack and changes places on them.

```
1020
(This is a string)
exch
```

Errors stack underflow

False variable

Syntax **False**

Applies to All editions

Description Contains the value 0.

Remarks The variable is generated internally each time a script starts.

file command

Syntax *filename access* **file** *filehandle*

Applies to All editions

Description Creates a file handle for the file identified as *filename*, accessing it as defined by *access*. Both parameters are strings. *filename* has to be a valid path including disk and pathway. Access has the following meaning:

Value	Meaning
a	The file is opened for writing only and all data sent to the file will be appended to the end of the file. If the file doesn't exist will it automatically be created.
r	The file is opened for reading only. If the file doesn't exist will an error occur.
w	The file is opened for writing only. If the file doesn't exist will it automatically be created.

Access may be upper or lower case.

Remarks The command will return a file handle. A file handle is simply an integer refering to the file that is used by the GFM Engine to access the file later. To read more about file handles and how to work with files see Section 2 in the *GFM Programmer's Guide*.

See also closefile, closeallfiles

Example This example creates a file called STATUS.LOG in the directory Images on disc C.If the file exists will the existsing file be removed and overwritten by the new data. If the file doesn't exist will be created..

```
(C:\Images\STATUS.LOG) (w) file
/F exch def
F (File has been updated) writestring
F closefile
```

Errors stack underflow, typecheck

findfile command

Syntax *filename* **findfile**

Applies to All editions

Description Searches for the file *filename*. Filename must be the absolute path including all directory names separated by backslashes.

Remarks If the file can be found will the value True be left on the stack. If the file can't be found, or if the path is incorrect will the stack contain the value False.

See also copyfile, findfolder, movefile

Example This example checks if the file called FLOWER.OIF exists in the directory Images on disc C. If the file exists will the top value on the stack be True otherwise will the top value be False.

```
(C:\Images\FLOWER.OIF) findfile
```

Errors stack underflow, typecheck

---

findfolder command

Syntax *pathname* **findfolder**

Applies to All editions

Description Searches for the folder *pathname*. pathname must be the absolute path including all directory names separated by backslashes.

Remarks If the folder can be found will the value True be left on the stack. If the folder can't be found, or if the path is incorrect will the stack contain the value False.

See also findfile

Example This example checks if the folder called System32 exists in the directory WINNT on disc C. If the file exists will the top value on the stack be True otherwise will the top value be False.

```
(C:\WINNT\System32) findfolder
```

Errors stack underflow, typecheck

---

ge command

Syntax *value1 value2* **ge** *boolean*

Applies to All editions

Description Pops *value1* and *value2* and returns True if *value2* is greater than or equal to *value1*, else will the command return False to the stack.

Remarks The values must be number. If any value is a string will the typecheck error be executed.

See also eq, gt, le, lt

Errors stack underflow, typecheck

---

getcmdwindowstyle command

Syntax **getcmdwindowstyle**

Applies to All editions

Description Pushes the command window style to the stack.

The default value is 2, i.e. Window is displayed as an icon with focus.

See also cmd, setcmdwindowstyle

Example This example pushes the current command window style to the stack.

```
getcmdwindowstyle
```

---

getcolorcomponents command

Syntax **getcolorcomponents**

Applies to Advanced Edition

Description Gets the color componts values used when converting images to RGBMode or CMYKMode. This topic is discussed in *Appendix D Images, Colors And Halftoning* in the *GFM Programmer's Guide*.

See also converttocmyk, getundercolorremoval, setmaxblackink, setundercolorremoval

Example This example gets the color components to the stack.

```
getcolorcomponents
```

---

getfileextensionname command

Syntax *filename* **getfilextensionname**

Applies to All editions

Description Returns a string containing the extension name for the last component in *filename*.

Remarks For network drives, the root directory (\) is considered to be a component. The **getfilextensionname** command returns a zero-length string () if no component matches the *filename* argument.

Example This example will push the string (exe) to the stack.

```
(C:\WINNT\System32\Notepad.exe) getfileextensionname
```

---

getfileattr command

Syntax *filename* **getfiledatetime**

Applies to All editions

Description Pushes the the file atrributes information for *filename* to the stack.

Remarks The value returned by **getfileattr** is the sum of the following attribute values

Value	Description
0	Normal
1	Read-only
2	Hidden
4	System file
16	Directory or folder

32 Archive. File has been changed since last backup.

To determine which attributes are set, use the **and** operator to perform a bitwise comparison of the value returned by the **getfileattr** function and the value of the individual file attribute you want. If the result is not zero, that attribute is set for the named file.

If *filename* can't be found will an error occur.

See also setfileattr

Example This example pushes the file attribute information for the file C:\FILE.TMP to the stack.

```
(C:\FILE.TMP) getfileattr
```

---

getfilecount command

Syntax *dirname* **getfilecount**

Applies to All editions

Description Pushes the the number of files in the *dirname* directory to the stack. Only files with attribute normal will be counted.

See also getfiledatetime, getfilesindirectory

Example This example counts the number of files in the C:\WINNT directory to the stack.

```
(C:\WINNT\*.*) getfilecount
```

---

getfiledatetime command

Syntax *filename* **getfiledatetime**

Applies to All editions

Description Pushes the the date and time when *filename* was created or last modified to the stack. The command will push two values (both string values) to the stack. The top value will always be the date and the second value will be the time.

See also getfilecount

Example This example pushes the date and time for the file C:\FILE.TMP to the stack.

```
(C:\FILE.TMP) getfiledatetime
```

---

getfilesindirectory command

Syntax *directorypath* **getfilesindirectory**

Applies to All editions

Description Pushes all files in the *directorypath* to the stack. The command will remove the *directorypath* from the stack before executing the command. Only files with file attribute Archive set. Hidden, system and read-only files will not be pushed to the stack.

Remarks The directorypath may include file extensions to search for. If you wants to retrieve all files in a directory you must use the * extension

See also getfilecount, getfiledatetime

Example This example pushes all files in the C:\WINNT directory to the stack.

```
(C:\WINNT\*.*) getfilesindirectory
```

This example pushes all files with the extension .TMP in the C:\WINNT directory to the stack.

```
(C:\WINNT\*.TMP) getfilesindirectory
```

---

getglobalstring command

Syntax *value* **getglobalstring**

Applies to All Editions

Description Pushes the contents of GlobaString(*value*) to the top of the stack.

See also setglobalstring

Example This example gets the globalstring 4 to the stack.

```
4 getglobalstring
```

Errors stack underflow, typecheck

---

getgraycomponents command

Syntax **getgraycomponents**

Applies to Advanced Edition

Description Gets the gray componts values used when converting images to GrayMode. This topic is discussed in *Appendix D Images, Colors And Halftoning* in the *GFM Programmer's Guide*.

See also converttocmyk, getundercolorremoval, setmaxblackink, setundercolorremoval

Example This example gets the gray components to the stack.

```
getgraycomponents
```

---

getimageheader command

Syntax **getimageheader**

Applies to Standard and Advanced Edition

Description Pushes the header of the current OIF file to the stack.

Remarks Before you can execute the command, the file must be opened with the **openimagefile** command. After the command, you should always run the procedure **AssignHeaderToVariables** included in the STANDARD.LIB library since many command uses the variables created in this procedure.

See also closeimagefile, loadimage, openimagefile, writeimagefile

Example This example opens an .OIF file called C:\FLOWER.OIF, reads the header and stores the header in GFMDict according to the AssignHeaderToVariables procedure..

```
200 loadimage
```

Errors no current image, stack underflow, typecheck

---

getmaxblackink command

Syntax **getmaxblackink**

Applies to Advanced Edition

Description Pushes the percentage of max black ink used when separating an RGBMode image into CMYKMode image to the stack. The value must be in the range 0 to 255, where 0 is 0% ink and 255 is 100% black ink. This topic is discusssed in *Appendix D Images, Colors And Halftoning* in the *GFM Programmer's Guide*.

The default value is 255, i.e. 100% black ink will be allowed when generating the black channel in a CMYKMode image.

See also converttocmyk, getundercolorremoval, setmaxblackink, setundercolorremoval

Example This example pushes the max black ink amount to the stack.

```
getmaxblackink
```

---

getinterpolationmethod command

Syntax **getinterpoaltionmethod**

Applies to Advanced Edition

Description Pushes the current interpolation method to the stack.

The default value is 1, i.e. the interpolation method will be nearest pixel.

See also setinterpolationmethod

Example This example pushes the current interpolation method to the stack.

```
getinterpolationmethod
```

---

getoverwriteonmove command

Syntax **getoverwriteonmove**

Applies to All editions

Description Pushes the mode how to overwrite an existing file when using the **movefile** command to the top of the stack.

The default value is 0, i.e. files will not be overwritten when using the **movefile** command.

See also movefile, setoverwriteonmove

Example This example pushes the current overwrite mode to the stack.

```
getoverwriteonmove
```

getGFMDict command

Syntax **getGFMDict**

Applies to All editions

Description Pushes the contents of all names in GFMDict to the stack.

See also init

Example This example gets all names in GFMDict to the stack.

```
getGFMDict
```

gt command

Syntax *value1 value2* g**t** *boolean*

Applies to All editions

Description Pops *value1* and *value2* and returns True if *value2* is greater than *value1*, else will the command return False to the stack.

Remarks The values must be number. If any value is a string will the typecheck error be executed.

See also eq, ge, le, lt

Example

```
3 2 gt -> False
2 3 gt -> True
```

Errors stack underflow, typecheck

if command

Syntax *value1 value2 proc* **if**

Applies to All editions

Description If *value1* and *value2* are equal *proc* will be executed.

Remarks If *value1* and *value2* are equal and *proc* has been executed, *value1* and *value2* will be removed from the stack.

See also ifelse, ifnot

Example In the example below , the procedure will be executed if ColorMode is 4.

```
ColorMode 4 {ImageHeight ImageWidth 4 mul mul} if
```

Errors stack underflow, typecheck

ifelse command

Syntax *value1 value2 proc1 proc2* **ifelse**

Applies to All editions

Description If *value1* and *value2* are equal *proc1* will be executed if *value1* and *value2* aren't equal will *proc2* be executed.

Remarks The command will remove all four items from the stack and not by itself push anything back to the stack, but the procedures it execute may do so.

See also if, ifnot

Errors stack underflow, typecheck

---

ifnot command

Syntax *value1 value2 proc* **ifnot**

Applies to All editions

Description If *value1* and *value2* are not equal, *proc* will be executed.

Remarks If *value1* and *value2* are not equal and *proc* has been executed, *value1* and *value2* will be removed from the stack.

See also if, ifelse

Example In the example below, the procedure will be executed if ColorMode is not 4.

```
ColorMode 4 {ImageHeight ImageWidth 4 mul mul} ifnot
```

Errors stack underflow, typecheck

ImageHeight variable

Applies to Standard and Advanced Edition

Syntax **ImageHeight**

Description Contains the image height in pixels for the current image in the PixelArea.

ImageWidth variable

Applies to Standard and Advanced Edition

Syntax **ImageWidth**

Description Contains the image width in pixels for the current image in the PixelArea.

---

init command

Syntax *value* **init**

Applies to All editions

Description Initiates the Stack, GFMDict and GrayComponents to different states according to the following table:

Value	Description
1	Clears the Stack
2	Clears the GFMDict
3	Clears the GFMDict and the Stack
4	Clears the Stack and the GFMDict. Sets GrayComponents to its default values. This init command also disconnects any connected modules and closes any open file including databases. For more information about the GrayComponets see *Appendix D Notes on Colour Mode Changes* in the *GFM Specification Manual*.

See also graycomponents, pop

Example This example clears the GFMDict.

```
2 init
```

Errors invalidparameter, stack underflow, typecheck

---

instring command

Syntax *string seek start* **instring**

Applies to All editions

Description Looks for the first occurrence of the string *seek* within *string* starting from character *start*. The command will return the first character position if *seek* can be found. If *seek* cannot be found will the return value be 0

Example This example searches for the string als in the string Salsa and will return 2 as this is the first occurrence of als.

```
(Salsa) (als) 1 instring
```

Errors stack underflow, typecheck

---

lcasestring command

Syntax *string* **lcasestring**

Applies to All editions

Description Converts a string to lowercase.

See also ucasestring

Remarks Only uppercase letters are converted to lowercase, all lowercase letters and nonletter characters remain unchanged.

Example This example converts a string to lowercase.

```
(PRINT) lcasestring
```

Errors stack underflow, typecheck

le command

Syntax *value1 value2* **le** *boolean*

Applies to All editions

Description Pops *value1* and *value2* and returns True if *value2* is less than or equal to *value1*, else will the command return False to the stack.

Remarks The values must be number. If any value is a string will the typecheck error be executed.

See also eq, ge, gt, lt

Errors stack underflow, typecheck

leftstring command

Syntax *string value* **leftstring**

Applies to All editions

Description Removes the *value* number of characters from the string *string*, starting from the leftmost character.

See also lengthstring, midstring, rightstring, trimstring

Example This example removes the first character, the file number, from a file called 1FLOWER.TIF so that the file is named FLOWER.TIF.

```
(1FLOWER.TIF) 1 leftstring
```

Errors stack underflow, typecheck.

length command

Syntax *array* **length** *int*

Applies to All editions

Description Returns the number of elements in the array *array* as *int*.

Remarks The top value on the stack must be an array.

Example This example pushes one array to the stack and the length command will push the number of elements to the stack.

```
[1 2 3 4 5] length -> 5
```

Errors stack underflow, typecheck.

lengthstring command

Syntax *string* **lengthstring**

Applies to All editions

Description Return the number of characters in the string *string*. *string* will be removed from the stack.

See also trimstring

Example This example pushes one string to the stack and the lengthstring command will push the number of characters to the stack.

```
(This is a string) lengthstring
```

Errors stack underflow, typecheck.

---

loadcolortable command

Syntax *filename* **loadcolortable**

Applies to Standard and Advanced Edition

Description Reads the color table *filename* to the active colortable.

Remarks The color table can be in any ColorMode. The important is that the image in the PixelArea has the same ColorMode as the color table when executing the **runcolortable** command.

See also runcolortable, savecolortable

Example This example reads the colortable C:\GFM\ColorTables\.ReduceRed.CTB and runs it on the current image in the PixelArea.

```
(C:\GFM\ColorTables\ReduceRed.CTB) loadcolortbale
runcolortable
```

Errors no current image, not a valid colortable, stack underflow, typecheck.

---

loadimage command

Syntax *value* **loadimage**

Applies to Standard and Advanced Edition

Description Loads the *value* number of samples to the Pixel Area from the currently opened OIF file.

Remarks The OIF file must have been opened with the **openimagefile** command before **loadimage** can be run. Note that the command loads samples, not pixels. In a GrayMode, samples are the same as pixels, but in an RGB or CMYK image you must add ColorMode with samples. For example: if the image is an RGB image and you shall load 200 pixels, you must use the command syntax: **200 3 mul** to get the number of samples to load. If the image is a CMYK image you must use the syntax: **200 4 mul**.

See also openimage, writeimagefile, closeimagefile.

Example This example loads 200 samples from the currently opened OIF file.

```
200 loadimage
```

Errors no current image, stack underflow, typecheck.

---

logfile command

Syntax *filename* **logfile**

Applies to All editions

Description Starts logfile monitoring of the System Monitor display to the file *filename*.

Remarks At start-up there is no logfile monitoring. After this command everything that is viewed in the System Monitor is also logged into the file *filename*. If the file doesn't exist, it will automatically be created. If the file exists, all logging will be appended to the file.

See also openimage, writeimagefile, closeimagefile

Example This example starts logfile monitoring to the file C:\PAGEPAIR.LOG.

```
(C:\PAGEPAIR.LOG) logfile
```

Errors stack underflow, typecheck.

---

lt command

Syntax *value1 value2* **lt** *boolean*

Applies to All editions

Description Pops *value1* and *value2* and returns True if *value2* is less than *value1*, else will the command return False to the stack.

Remarks The values must be number. If any value is a string will the typecheck error be executed.

See also ge, gt, le, lt

Example

```
3 2 lt -> True
2 3 lt -> False
```

Errors stack underflow, typecheck

---

mergestrings command

Syntax *string1 string2* **mergestrings** *string3*

Applies to All editions

Description Merges the two top strings on the stack to one string.

Remarks The two top values on the stack must be strings.

See also trimstring

Example This example pushes two strings to the stack and merges them into one.

```
(This is )
(a string)
mergestrings
```

Errors stack underflow, typecheck.

---

midstring command

Syntax *string1 startpos length* **midstring** *string2*

Applies to All editions

Description Returns the *length* number of characters from the string *string1* starting with character *startpos*.

Remarks If *startpos* is greater than the length of the string an error will occur. If *length* is greater than remaining of the string *string1* the remaining of *string1* will be returned.

See also leftstring, rightstring, lengthstring

Example This example will return the string BCD from the string ABCDEFG.

```
(ABCDEFGH) 2 3 midstring
```

Errors stack underflow, string to short, typecheck

>ModuleType property

Syntax **>ModuleType**

Applies to Standard and Advanced Edition

Description Returns an integer value indicating the moduletype of the connected module.

Remarks The value returned after the **>ModuleType** will be on top of the stack and has the following meaning:

Value	Meaning
1	Standard Image Module (SIM). The module can read it's internal format, for example the TIFF format, and convert the image format to OIF (Open Interchange Format). It can also convert back the image from OIF to it's internal format.
2	Input Image Module (IIM). The module can only read it's internal format and convert the format to OIF.
3	Output Image Module (OIM). The module can only read OIF and convert OIF to it's internal format.
4	OIF Application Module (OAM). These modules can read and manipulate OIF files. For example can one of these modules be an RGB to CMYK converter. You have these functions also in GFM but if you want to fine tune this conversion, it could be useful to have this module.
5	Stand-alone Modules(SAM). These modules can be seen as separate programs and can be the only one enabled in the Task List. For example could a Page-Pairing module be a Stand-Alone Module. Another example is a Blend module, that is a part of an OPI function, and can also be a Stand Alone Module.
255	User Defined Modules (UDM). These modules are non-standard and must be managed from GFM code. There are actually no guidelines about how to write an UDM. You can for example write an UDM that reads and changes the contents in the database or an UDM that becomes an OLE-client and uses other modules to achieve a specific task.

The command is very important for how other commands shall be used. For more information about this topic see *Chapter 8 Programming Modules With GFM* in the *GFM Programmer's Guide*.

See also connectmodule, !Action >AllowConfigure, !Direction, !SourceFileName, !DestinationFileName

Example This example connects to the TIFF module and reads the moduletype.

```
(TIFFModule) connectmodule
>ModuleType
```

Errors Stack underflow, Typecheck

---

month command

Syntax **month**

Applies to All editions

Description Pushes the value of the current month to the top of the stack. If the month is below 10 will it contain leading zeros. Note that the month will be a string value.

See also date, time, year

---

movefile command

Syntax *sourcefilename destinationfilename* **movefile**

Applies to All editions

Description Moves the *sourcefilename* to *destinationfilename*.

Remarks You can only use the movefile command on the same drive. You cannot move files between drives.

If you try to use the movefile command on a currently open file, an error occurs. An error will also occur if the *sourcefilename* does not exist or the path for the *destinationfilname* is incorrect.

An error will also occur if the *destinationfilname* already exists unless you have used the **setoverwriteonmove** command and set the overwriteonmove to 1.

movefile also supports wildcharacters as * and ? when moving files.

See also copyfile, setoverwriteonmove

Example This example moves a file C:\IMAGES\IMAGE1 to the C:\IMAGEBACKUP\IMAGE1

```
(C:\IMAGES\IMAGE1) (C:\IMAGEBACKUP\IMAGE1) movefile
```

Errors file already exists, file not found, Stack underflow, Typecheck, $syserror

---

mul command

Syntax *value1 value2* **mul** *result*

Applies to All editions

Description Multiplies the two top values on the stack and pushes the result back to the stack. V*alue1* and *value2* will be removed from the stack.

Remarks The two top values can be any number. If any of the two top values are a string, the typecheck error will occur.

See also div, mul, sub

Example 10 20 mul -> stack contains 200

Errors stack underflow, typecheck

---

newresolution command

Syntax *resolution* **newresolution**

Applies to Standard and Advanced Edition

Description Changes the resolution of an image loaded to the PixelArea. The new resolution (that must be in dpi (dots per inch)) must be on top of the stack.

Remarks The width and height of the image will be retained when using the command. That is, if the resolution is increased, the file size will be increased. The interpolation method used is very important when increasing the resolution of an image. The interpolation method used is determined with the **setinterpolationmethod** command. If the new resolution is lower than the resolution of the image in the PixelArea, the interpolation method will have no meaning at all.

See also setinterpolationmethod

Example This example creates a 72 dpi image in the current PixelArea.

```
72 newresolution
```

Errors not a vaild resolution , stack underflow, typecheck .

---

not command

Syntax *value1* **not** *result*

Applies to All editions

Description Performs a logical NOT operation of the top value on the stack. The *value1* will be removed from the stack

Remarks The top value can be any number. If the top value is a string, will the typecheck error occur.

See also and, or ,xor

Example The following example does a logical NOT comparsion:

```
52 not -> -53
```

Errors stack underflow, typecheck

---

opendb command

Syntax *filename* **opendb**

Applies to Standard and Advanced Edition

Description Opens a Microsoft Access database.

See also closedb

Example This example opens a database named C:\DATABASE.MDB.

```
(C:\DATABASE.MDB) opendb
```

Errors file not found, stack underflow, typecheck

---

openimagefile command

Syntax *filename* **openimagefile**

Applies to Standard and Advanced Edition

Description Opens the .OIF *filename* file for processing.

Remarks This command must be executed before the **getinmageheader** or **loadimage** command can be executed. The command only opens the file and creates a handle internally for the file.

See also closeimagefile getimageheader, loadimage, writeimagefile.

Example This example opens an OIF file named C:\IMAGE1.OIF.

```
(C:\IMAGE1.OIF) openimagefile
```

Errors file not found, stack underflow, typecheck.

---

or command

Syntax *value1 value2* **or** *result*

Applies to All editions

Description Performs a logical OR operation of the two top values on the stack. The *value1* and v*alue2* will be removed from the stack

Remarks The two top values can be any number. If any of the two top values are a string, will the typecheck error occur.

See also and, not, xor

Example The following example does a logical OR comparsion:

```
17 5 or -> 21
```

Errors stack underflow, typecheck

---

pdfbleedbox command

Syntax *x1 y1 x2 y2* **pdfbleedbox**

Applies to Standard and Advanced Edition

Description Rectangle specifying the region to which all page content should be clipped if the page is being output in a production environment. In such environments, a “bleed area” is desired, to accommodate physical limitations of cutting, folding, and trimming equipment. The actual printed page may include printer’s marks that fall outside the bleed box. The default is the value of the crop box as defined by **pdfcropbox**.

See also pdfcropbax, pdfmediabox, pdftrimbox

Example The following example sets the bleed box to A4.

```
0 0 595.276 841.89 pdfbleedbox
```

pdfclosefile command

Syntax **pdfclosefile**

Applies to Standard and Advanced Edition

Description Closes the current opened PDF file.

See also pdfopenfile

Example The following example closes the currently opened PDF file.

```
pdfclosefile
```

pdfcolor command

Syntax *cyan mangenta yellow black* **pdfcolor**

Applies to Standard and Advanced Edition

Description Sets color for next text or shape that will be inserted in a page.

Remarks The color must be in the range 0 to 1 where 1 equals max ink. In the case of an grayscale image will only the *black* value be used.

Default values are 0 0 0 1 which equals solid black.

See also pdfinserttext

Example The following example sets the current color to 100% (solid) black.

```
0 0 0 1 pdfcolor
```

pdfcropbox command

Syntax *x1 y1 x2 y2* **pdfcropbox**

Applies to Standard and Advanced Edition

Description Rectangle specifying the default clipping region for the page when displayed or printed. The default is the value of the media box as defined with **pdfmediabox**.

See also pdfbleedbox, pdfmediabox, pdftrimbox

Example The following example sets the crop box to A4.

```
0 0 595.276 841.89 pdfcropbox
```

pdffontname command

Syntax *font* **pdffontname**

Applies to Standard and Advanced Edition

Description Sets the current font used when appending text on a page.

Remarks The command should be used to position text, images or shapes that shall be inserted on a page.

See also pdffontsize, pdfmoveto, pdfinserttext

Example The following example sets the current font to Courier.

```
(Courier) pdffontname
```

---

pdffontsize command

Syntax *fontsize* **pdffontsize**

Applies to Standard and Advanced Edition

Description Sets the size for the current font used when inserting text on a page.

Remarks The *fontsize* must always be expressed in points.

See also pdffontname, pdfmoveto, pdfinserttext

Example The following example sets the current font size to 12 points.

```
12 pdffontsize
```

---

pdfgetcatalogdict command

Syntax **pdfgetcatalogdict**

Applies to Standard and Advanced Edition

Description Reads the Catalog (Root) dictionary from the currently opened PDF file.

Remarks Before the command can be used must the **pdfgettrailer** command been executed. Will read a number of variables that must be present before starting working wih the PDF.

The command will push the variable **PDFPageCount** onto the GFMDict. The value holds the number of pages in this PDF file.

See also pdfgettrailer, pdfgetpages

Example The following example reads the catalogdict of the currently opened PDF file.

```
(C:\PDF\PDFFile.ps) pdfopenfile
pdfgettrailer
pdfgetcatalogdict
```

---

pdfgetinfodict command

Syntax **pdfgetinfodict**

Applies to Standard and Advanced Edition

Description Reads the Info dictionary from the currently opened PDF file.

Remarks Before the command can be used must the **pdfgettrailer** command been executed. Will create a number of variables in GFMDict as described below:

`PDFAuthor`

```
PDFCreationDate

PDFModDate

PDFCreator

PDFProducer

PDFTitle

PDFSubject

PDFKeywords
```

Note that empty variables will also be created in the GFMDict.

See also pdfgettrailer, pdfwriteinfodict

Example This example opens a PDF file, reads the Info dict, changes the PDFTitle in the PDF file and saves the updated PDF file

```
(C:\PDF\PDFFile.pdf) pdfopenfile
pdfgettrailer
pdfgetinfodict
/PDFTitle (A new title) def
pdfwriteinfodict
pdfclosefile
```

---

pdfgetpages command

Syntax **pdfgetpages**

Applies to Standard and Advanced Edition

Description Reads the pages tree from the currently opened PDF file.

Remarks Before the command can be used must the **pdfcatalogdict** command been executed. **pdfgetpages** will read the complete pages tree and save the structure of all pages so that the GFM engine can access each page.

See also pdfgettrailer, pdfgetpages

Example The following example loads the page tree the currently opened PDF file.

```
(C:\PDF\PDFFile.ps) pdfopenfile
pdfgettrailer
pdfgetcatalogdict
pdfgetpages
```

---

pdfgettrailer command

Syntax **pdfgettrailer**

Applies to Standard and Advanced Edition

Description Reads the trailer(s) and the XREF table(s) of the currently opened PDF file.

Remarks The command must be used if the current PDF file should be read, changed or updated. The only sitiation where the command won't be needed is when working with the Info dict.

Will create a variable called **PDFSize** in GFMDict. The variable will hold the number of entries in the PDF file's cross-reference table.

Other variables that will be created in GFMDict is the following:

PDFID

PDFInfo

PDFRoot

See also pdfopenfile, pdfgetcatalogdict

Example The following example reads the trailer of the currently opened PDF file.

```
pdfgettrailer
```

---

pdfinserttext command

Syntax *string* **pdfinserttext**

Applies to Standard and Advanced Edition

Description Inserts the text *string* at the position given by **pdfmoveto** in font given by **pdffontname** at the size given by **pdffontsize**.

Remarks The text will not be saved to the PDF file until the **pdfsavepage** command has been executed. This means that you have to issue the **pdfsavepage** command for each text that should be inserted on the same page.

The text will always be interpretated as WinAnsiEncoding.

Example The following example will insert the text ABCDEFG on the current page in font Courier, 36 points, at the position 100 100 (x,y) in the color black.

```
(Courier) pdffontname
0 0 0 1 pdfcolor
36 pdffontsize
100 100 pdfmoveto
(ABCDEFG) pdfinserttext
```

---

pdfloadpage command

Syntax *pagenumber* **pdfloadpage**

Applies to Standard and Advanced Edition

Description Loads the page *pagenumber* and makes the page the current page.

Remarks **pdfloadpage** will only load the page, i.e. no changes will be applied to the page until the next **pdfsavepage** is executed. **pdfloadpage** will not restore any of the pdf variables. For example if the current font has been set with **pdffontname** will **pdfloadpage** not change the font name.

The variable **PDFPageCount** will hold the number of pages in the current PDF file.

See also pdfsavepage

Example The following example loads page number 1 in a PDF file and makes it the current page.

```
(C:\PDF\PDFFile.ps) pdfopenfile
pdfgettrailer
pdfgetcatalogdict
pdfgetpages
1 pdfloadpage
```

pdfmediabox command

Syntax *x1 y1 x2 y2* **pdfmediabox**

Applies to Standard and Advanced Edition

Description Rectangle specifying the "natural size" of the page, for example the dimensions of an A4 sheet of paper. The coordinates are measured in points where 1 point equals 1/72 inch. This rectangle includes any extended area surrounding the finished page for bleed, printing marks, or other similar purpose.

See also pdfcropbox, pdfbleedbox, pdftrimbox

Example The following example sets the media box to A4.

```
0 0 595.276 841.89 pdfmediabox
```

pdfmoveto command

Syntax *x y* **pdfmoveto**

Applies to Standard and Advanced Edition

Description Sets the current point on a page *(x,y)* without adding anything to the page.

Remarks The command should be used to position text, images or shapes that shall be inserted on a page.

The position must always be expressed in points where 1 point is 1/72 inch. You can although define a procedure that will recaclulate fron any measurment system to points

See also pdfaddtext

Example The following example sets the current point to 100 (x) and 200 (y).

```
100 200 pdfmoveto
```

pdfopenfile command

Syntax *string* **pdfopenfile**

Applies to Standard and Advanced Edition

Description Opens the file *string* as a PDF file.

Remarks If the file *string* doesn't exist will an error occur. If file isn't a PDF file will an error occur. If the file is a valid PDF file will a string variable called **PDFVersion** be created in GFMDict. The variable will hold **%PDF-1.1**, **%PDF-1.2** or **%PDF-1.3** depending on the version of the PDF file.

See also pdfclosefile

Example This example opens a PDF file.

```
(C:\PDF\PDFFile.ps) pdfopenfile
```

Errors stack underflow, typecheck

pdfrotatepage command

Syntax *angle* **pdfrotatepage**

Applies to Standard and Advanced Edition

Description Rotates the currently loaded page *angle* degrees.

Remarks Any previous rotation will be overwritten. The rotation must be 0, 90, 180 or 270 degrees.

Example The following example rotates the current page 180 degrees.

```
180 pdfrotatepage
```

pdfsavepage command

Syntax **pdfloadpage**

Applies to Standard and Advanced Edition

Description Saves the current page and writes all changes done.

Remarks **pdfloadpage** will only load the page, i.e. no changes will be applied to the page until the next **pdfsavepage** is executed. Pdfloadpage will not restore any of the pdf variables. For example if the current font has been set with **pdffontname** will pdfloadpage

The variable **PDFPageCount** will hold the number of pages in the current PDF file.

See also pdfloadpage

Example The following example loads page number 1 in a PDF file and makes it the current page.

```
(C:\PDF\PDFFile.ps) pdfopenfile
pdfgettrailer
pdfgetcatalogdict
pdfgetpages
1 pdfloadpage
```

pdftrimbox command

Syntax *x1 y1 x2 y2* **pdftrimbox**

Applies to Standard and Advanced Edition

Description Rectangle specifying the intended “finished size” of the page (for example, the dimensions of an A4 sheet of paper). In some cases, the media box will be a larger rectangle, which includes printing instructions, cut marks, or other content. The default is the value of the crop box as defined with the **pdfcropbox**.

See also pdfcropbax, pdfbleedbox, pdfmediabox

Example The following example sets the trim box to A4.

```
0 0 595.276 841.89 pdftrimbox
```

pdfviewprogress command

Syntax *boolean* **pdfviewprogress**

Applies to Standard and Advanced Edition

Description Determines if a progress bar should be visible or not as in the following table:

Value	Meaning
False	Do not show a progress bar. Default value.
True	Show a progress bar as the PDF file is parsed.

Example The following example activates the progress bar.

```
True viewprogress
```

pdfwriteinfodict command

Syntax **pdfwriteinfodict**

Applies to Standard and Advanced Edition

Description Saves the Info dict to the currently opened PDF file.

Remarks The command will write down all of the following variables defined in GFMDict.

`PDFAuthor`

`PDFCreationDate`

`PDFModDate`

`PDFCreator`

`PDFProducer`

`PDFTitle`

`PDFSubject`

`PDFKeywords`

Under normal circumstances would you open a PDF file, read the Info dict, do your changes, write down your changes and finally close the PDF file. See the example below.

Note that the command will write a new cross-reference table and a new trailer to the PDF file allowing you to only read the Info dict, do any changes and then save the Info dict without loading any pages since the Info dict is independent of the rest of the PDF file.

See also pdfgetinfodict

Example This example opens a PDF file, reads the Info dict, changes the PDFTitle in the PDF file and saves the updated PDF file

```
(C:\PDF\PDFFile.ps) pdfopenfile
pdfgettrailer
pdfgetinfodict
/PDFTitle (A new title) def
pdfwriteinfodict
pdfclosefile
```

Errors stack underflow, typecheck

PixelConfiguration variable

Syntax **PixelConfiguration**

Applies to Standard and Advanced Edition

Description Contains the pixel configuration for the current image in the PixelArea.

Remarks The *PixelConfiguration* indicates how the image data has been organised and has the following meaning:

Value	Meaning
1	Pixel Interleave. For example RGB data will be stored as RGBRGBRGBRGB
2	Pixel Planar. The colours are stored in separate planes. For example RGB data will be stored as RRRRRGGGGGBBBBB.

The default value is 1.

pop command

Syntax **pop**

Applies to All editions

Description Removes the top item from the stack.

See also init

Example This example pushes three values to the stack and removes one.

```
1 2 3 pop
```

Errors stack underflow.

print command

Syntax *string* **print**

Applies to All editions

Description Prints the string *string* in the system monitor.

See also mergestring, trimstring

Example This example prints the string ”This is a string” in the system monitor.

```
(This is a string)
print
```

Errors stack underflow, typecheck.

psboundingbox command

Syntax **psboundingbox**

Applies to Standard and Advanced Edition

Description Returns the %%BoundingBox parameters from the opened PostScript file.

See also psopenfile, pstitle

Example This example opens a PostScript file and returns the %%BoundingBox parameters to the stack.

```
(C:\PS\Postscript0.ps) psopenfile
psboundingbox
```

Errors no current postscript file

---

psclosefile command

Syntax **psclosefile**

Applies to Standard and Advanced Edition

Description Closes the currently opened PostScript file.

See also psopenfile

Example This example opens a closes a PostScript file.

```
(C:\PS\Postscript0.ps) psopenfile
psclosefile
```

---

psopenfile command

Syntax *string* **psopenfile**

Applies to Standard and Advanced Edition

Description Opens the file *string* as a PostScript file.

See also psclosefile

Example This example opens a PostScript file.

```
(C:\PS\Postscript0.ps) psopenfile
```

Errors Stack underflow, Typecheck

---

pstitle command

Syntax **pstitle**

Applies to Standard and Advanced Edition

Description Pushes the name of a PostScript file to the stop of the stack.

See also psboundingbox, psopenfile

Example This example prints the string (This is a string) in the system monitor.

```
(C:\PS\Postscript0.ps) psopenfile
pstitle
```

Errors no current postscript file.

PushImageHeaderToStack procedure

Syntax **PushImageHeaderToStack** *imageheader*

Applies to Standard and Advanced Edition

Description Pushes the imageheader of the current image in the PixelArea to the stack.

Remarks The procedure is in the STANDARD.LIB library and should always be executed before saving an image file.

See also AssignImageHeaderToStack

Example This example saves the image in the PixelArea to disk using the **PushImageHeaderToStack** procedure as C:\IMAGE1.OIF and closes the file.

```
(C:\IMAGE1.OIF)
PushImageHeaderToStack
writeimagefile
closeallfiles
```

---

rand command

Syntax *lowerbound upperbound* **rand** *real*

Applies to All editions

Description Pushes a random value in the range *lowerbound* to *upperbound* to the top of the stack.

Example This example creates a random value in the range 65 to 77 and converts the value to an integer and the integer value to a character.

```
65 77 rand cvi char
```

Errors stack underflow, typecheck.

---

readline command

Syntax *filehandle* **readline** *substring*

Applies to All editions

Description Reads a line of characters teminated by a newline from the file refered to as *filehandle* into the string *substring*.

The newline character will not be stored in substring.

See also file

Example This example opens the file C:\IMAGES\STATUS.LOG, reads the first line and then closes the file.

```
C:\IMAGES\STATUS.LOG) (r) file
/F exch def
F readline /FirstLine exch def
F closefile
```

Errors stack underflow, typecheck.

repeat command

Syntax *int proc* **repeat**

Applies to All editions

Description Executes *proc int* times, where *int* must be a non-negative integer. The **repeat** leaves no result on its own on the stack but *proc* may do so. If *proc* executes the **endjob** command will the script be terminated.

Example 4 {(Hello World!) print} repeat

Errors stack underflow, typecheck.

rightstring command

Syntax *string value* **rightstring**

Applies to All editions

Description Removes the *value* number of characters from the string *string*, starting from the rightmost character.

Remarks Can be useful when, for example, removing a file extension from a filename.

See also trimstring, leftstring, midstring

Example This example removes four characters, the extension, from a file called C:\IMAGE1.TIF so that the file is named C:\IMAGE1 and then adds the default extension from the current module.

```
(C:\IMAGE1.TIF) 4 rightstring
>DefaultExtension mergestrings
```

Errors stack underflow, typecheck.

run command

Syntax *filename* **run**

Applies to All editions

Description Runs a GFM file from GFM code.

Remarks The file *filename* must exist. This command is premaliary used for loading libraries. You cannot use the **endjob** command in a GFM file started from another file.

Example This example loads the STANDARD.LIB library to the GFMDict.

```
(C:\GFM\LIBS\STANDARD.LIB) run
```

Errors file not found, Stack underflow, Typecheck.

runcolortable command

Syntax **runcolortable**

Applies to Standard and Advanced Edition

Description Runs the active ColorTable with the image loaded into the PixelArea.

Remarks It is important that the image in the PixelArea has the same ColorMode as the color defined in the ColortTable when executing the command.

See also loadcolorcolortable, savecolortable

Example This example reads the colortable C:\GFM\ColorTables\.ReduceRed.CTB and runs it on the current image in the PixelArea.

```
(C:\GFM\ColorTables\ReduceRed.CTB) loadcolortbale
runcolortable
```

Errors color mode mismatch, no current image

---

savecolortable command

Syntax *filename ColorMode* **savecolortable**

Applies to Standard and Advanced Edition

Description Saves the active color table as *filename* with ColorMode.

Remarks For more information about colour tables see *Chapter 10 The Use of ColorTables* in the *GFM Programmer's Guide*.

See also loadcolortable, runcolortable

Example This example saves the active colortable as an RGB the colortable as C:\GFM\ColorTables\.ReduceRed.CTB.

```
(C:\GFM\ColorTables\ReduceRed.CTB) 3 savecolortable
```

Errors no current image, not a valid colortable, stack underflow, typecheck

---

setblackgeneration command

Syntax *value* **setblackgeneration**

Applies to Advanced Edition

Description Sets the values used when converting an RGBMode image to CMYKMode. max black ink used when separating an RGBMode image into CMYKMode image. The value must be in the range 0 to 255, where 0 is 0% ink and 255 is 100% black ink. The value is used according to a formula that is discussed in *Appendix D Images, Colors And Halftoning* in the *GFM Programmer's Guide*.

See also converttocmyk, getundercolorremoval, getmaxblackink, setundercolorremoval

Example This example stops the interpretation in 5 seconds.

```
255 setblackgeneration
```

Errors stack underflow, typecheck

setcmdwindowstyle command

Syntax *value* **setcmdwindowstyle**

Applies to All editions

Description Sets the current window style used when starting the **cmd** command. The window style determines how the new **cmd** process will appear on your monitor. The following table discusses the different window style values.

Value	Meaning
0	Window is hidden and focus is passed to the hidden window.
1	Window has focus and is restored to its original size and position.
2	Window is displayed as an icon with focus.
3	Window is maximized with focus.
4	Window is restored to its most recent size and position. The currently active window remains active.
6	Window is displayed as an icon. The currently active window remains active.

The default value is 2, i.e. Window is displayed as an icon with focus.

Remarks This command could be very important when using the **cmd** command since it could totally change the apperance of your display.

See also cmd, getcmdwindowstyle

Example This example sets the command window style to 3.

```
3 setcmdwindowstyle
```

Errors not an allowed value, stack underflow, typecheck

setcolorcomponents command

Syntax v*alue1 value2 value3 value4* **setcolorcomponents**

Applies to Advanced Edition

Description Changes the default values how an image is converted to CMYKMode or RGBMode.

Remarks When converting from one ColorMode to another, some sort of algorithm must be used. In GFM you can affect this algorithm by changing different values. When converting an image into RGBMode or CMYKMode, you can use the **setcolorcomponents** command.

Conversion from RGBMode to GrayMode

When converting from RGBMode to GrayMode, the gray value is computed according to the NTSC video standard. This standard determines how a colour television signal is rendered on a black and white television.

gray = 0.3 x red + 0.59 x green + 0.11 x blue

Conversion from CMYKMode to GrayMode

To convert a CMYK value to gray you will use the following algorithm:

gray = 1.0 - min(1.0, 0.3 x cyan+ 0.59 x magenta + 0.11 x yellow + black)

Now, with the **graycomponents** command you can change the values 0.3, 0.59 and 0.11 in the formulas above where 0.3 value must be on top of the stack and 0.11 value must be the third value on the stack. This command should of course be used with care. The default values will do for most situations. If you change any of these values, be sure that you really know what you are doing

For more information about colours and colour conversion see *Chapter 6 Image And ColorMode Conversions* and *Chapter 7 Changing Resolutions* in the *GFM Programmer's Guide*.

See also converttogray, getgraycomponets, init, loadimage

Example This example loads an image called C:\IMAGES\IMAGE1.OIF to the Pixel Area, changes the gray components, converts the image to GrayMode and saves the new image as C:\IMAGES\IMAGE1GRAY.OIF.

```
(C:\GFM\LIBS\STANDARD.LIB) run
(C:\IAMEGS\IMAGE1.OIF) dup openimagefile getimageheader
AssignHeaderToVariables
(Processing Image: ) exch mergestrings print
ViewColorMode
ColorMode 1 {(Image is already GrayMode) print endjob} if
ColorMode 3 {ImageHeight ImageWidth 3 mul mul} if
ColorMode 4 {ImageHeight ImageWidth 4 mul mul} if
(Loading Image Stripe...) print
loadimage
(Converting to GrayMode) print
0.35 0.59 0.13 graycomponents
converttogray
PushImageHeaderToStack
(C:\IMAGES\IMAGE1GRAY.OIF)
(Writing image file) print
writeimagefile
(Closing files) print
closeimagefile
```

Errors stack underflow, typecheck.

---

setfileattr command

Syntax *filename attributes* **settfiledatetime**

Applies to All editions

Description Sets the the file atrributes *attributes* for *filename* to the stack.

Remarks *attributes* is the sum of the following values:

Value	Description
0	Normal
1	Read-only
2	Hidden
4	System file
32	Archive. File has been changed since last backup.

If *filename* can't be found will an error occur.

See also setfileattr

Example This example sets the file attribute information for the file C:\FILE.TMP Read-only and archive.

```
(C:\FILE.TMP) 33 setfileattr
```

---

setglobalstring command

Syntax *string value* **setglobalstring**

Applies to All editions

Description Sets the top value on the stack as GlobalString(*value*).

See also getglobalstring

Example This example sets the top value on the stack as globalstring 4.

```
(C:\IMAGE1.OIF)
4 setglobalstring
```

Errors stack underflow, typecheck

---

setgraycomponents command

Syntax *Value1 Value2 Value3* **setgraycomponents**

Applies to Advanced Edition

Description Changes the default values how an image is converted from RGBMode or CMYKMode to GrayMode.

Remarks When converting from one colorspace to another, some sort of algorithm must be used. In GFM you can affect this algorithm by changing different values. When converting an RGBMode or CMYKMode image to GrayMode, you can use the **graycomponents** command.

Conversion from RGBMode to GrayMode

When converting from RGBMode to GrayMode, the gray value is computed according to the NTSC video standard. This standard determines how a colour television signal is rendered on a black and white television.

gray = 0.3 x red + 0.59 x green + 0.11 x blue

Conversion from CMYKMode to GrayMode

To convert a CMYK value to gray you will use the following algorithm:

gray = 1.0 - min(1.0, 0.3 x cyan+ 0.59 x magenta + 0.11 x yellow + black)

Now, with the **graycomponents** command you can change the values 0.3, 0.59 and 0.11 in the formulas above where 0.3 value must be on top of the stack and 0.11 value must be the third value on the stack. This command should of course be used with care. The default values will do for most situations. If you change any of these values, be sure that you really know what you are doing

For more information about colours and colour conversion see *Chapter 6 Image And ColorMode Conversions* and *Chapter 7 Changing Resolutions* in the *GFM Programmer's Guide*.

See also converttogray, getgraycomponets, init, loadimage

Example This example loads an image called C:\IMAGES\IMAGE1.OIF to the Pixel Area, changes the gray components, converts the image to GrayMode and saves the new image as C:\IMAGES\IMAGE1GRAY.OIF.

```
(C:\GFM\LIBS\STANDARD.LIB) run
(C:\IAMEGS\IMAGE1.OIF) dup openimagefile getimageheader
AssignHeaderToVariables
(Processing Image: ) exch mergestrings print
ViewColorMode
ColorMode 1 {(Image is already GrayMode) print endjob} if
ColorMode 3 {ImageHeight ImageWidth 3 mul mul} if
ColorMode 4 {ImageHeight ImageWidth 4 mul mul} if
(Loading Image Stripe...) print
loadimage
(Converting to GrayMode) print
0.35 0.59 0.13 graycomponents
converttogray
PushImageHeaderToStack
(C:\IMAGES\IMAGE1GRAY.OIF)
(Writing image file) print
writeimagefile
(Closing files) print
closeimagefile
```

Errors stack underflow, typecheck.

---

setinterpolationmethod command

Syntax **settinterpoaltionmethod**

Applies to Standard and Advanced Edition

Description Sets the current interpolation method used. The interpolation method determines how accurate pixels will be added to an image loaded into the PixelArea when using the **newresolution** command. The following table discusses the currently supported methods. Note that when reducing pixels in an image, the interpolation method has no meaning.

Value	Meaning
1	**Nearest Pixel**. The new pixel that will be generated is just a copy of the pixel next to the pixel. This method is fast but will not do under most circumstances.
2	**Average Line**. With this method , the new pixel will be an average of both pixels surrounding the new pixel. This method will generate much better results than Nearest Pixel and will do in many cases.
3	**Average Box**. If interpolation method is set to 3, the new pixel will be an average of the eight pixel surrounding the new pixel in all directions. This method will generate the best results, and will do in most cases. This is also the most time-consuming method when generating new pixels.

The default value is 1, i.e. the interpolation method will be nearest pixel.

Remarks This command is very important when using the **newresolution** command. It is recommended that you should set the interpolation method to 2 or 3 since most images will not do when using nearest pixel method. The nearest pixel method will only do when you don't care about the quality of the image.

See also getinterpolationmethod, newresolution

Example This example sets the interpolation method to 3.

```
3 setinterpolationmethod
```

Errors not an allowed value, stack underflow, typecheck

---

setmaxblackink command

Syntax *value* **setmaxblackink**

Applies to Advanced Edition

Description Sets the percentage of max black ink used when separating an RGBMode image into CMYKMode image. The value must be in the range 0 to 100, where 0 is 0% ink and 100 is 100% black ink. This topic is discusssed in *Appendix D Images, Colors And Halftoning* in the *GFM Programmer's Guide*.

The command is also used when converting from CMYKMode to RGBMode and sets the amount of black used when creating RGBMode. For more information see *Appendix D Images, Colors And Halftoning* in the *GFM Programmer's Guide*.

The default value is 100, i.e. 100% black ink will be allowed when generating the black channel in a CMYKMode image.

See also converttocmyk, converttocmyk, getundercolorremoval, getmaxblackink, setundercolorremoval

Example This example sets the max black ink to 95%.

```
95 setmaxblackink
```

Errors not an allowed value, stack underflow, typecheck

---

setoverwriteonmove command

Syntax *value* **setoverwriteonmove**

Applies to All editions

Description Sets the mode whether to overwrite an existing file when using the **movefile** command.or not. The following table discusses the different overwrite modes.

Value	Meaning
0	Files are not overwritten when using the **movefile** command.
1	Files are overwritten automatically when using the **movefile** command.

The default value is 0, i.e. files will not be overwritten when using the **movefile** command.

Remarks This command could be very important when using the **movefile** command since it can overwrite files that you want to keep on your harddisk.

See also getoverwritemode, movefile

Example This example sets the overwrite mode to 1.

```
1 setoverwritemode
```

Errors not an allowed value, stack underflow, typecheck

setundercolorremoval command

Syntax *value* **setundercolorremoval**

Applies to Advanced Edition

Description Sets the values used when converting an RGBMode image to CMYKMode. max black ink used when separating an RGBMode image into CMYKMode image. The value must be in the range 0 to 255, where 0 is 0% ink and 255 is 100% black ink. The value is used according to a formula that is discussed in *Appendix D Images, Colors And Halftoning* in the *GFM Programmer's Guide*.

See also converttocmyk, getundercolorremoval, getmaxblackink, setmaxblackink

Example This example stops the interpretation in 5 seconds.

```
176 setundercolorremoval
```

Errors not an allowed value, stack underflow, typecheck

showprogressbar command

Syntax *value* **showprogressbar**

Applies to All editions

Description Sets if the progressbar shall be visible or not. The progressbar will show the status of some commands that can take a while to complete.

Value	Meaning
False	The progressbar will not be visisble.
True	The progressbar will be visible.

The default value is False, i.e. the progressbar will not be visible.

Remarks Some commands will not be affected at all by this command. However, some commands will be affected in the way that a menu with a progressbar and a cancel button will be visible.

Example This example sets the interpolation method to 3.

```
True showprogressbar
```

Errors not an allowed value, stack underflow, typecheck

!SourceFileName, >SourceFileName property

Syntax **!SourceFileName**

Applies to Standard and Advanced Edition

Description Sets or reads the modules source filename.

Remarks The meaning of the **!SourceFileName** depends on **!Action**, **>ModuleType** and **!Direction** properties of the connected module. If the module has been configured as an IN module, the **!SourceFileName** will be the modules filetype filename. If the module has been configured as an OUT module, the **!SourceFileName will** be the .OIF filename.

The command depends on a set of other command and the type of module. For more information about this topic see *Chapter 6 Programming Modules With GFM* in the *GFM Programmer's Guide*.

See also !Action, connectmodule, !DestinationFileName, !Direction, >ModuleType,

Example This example connects to the TIFF module and sets the direction to be an OUT module uses the default extension for the TIFFModule (.TIF) and converts a file from GlobalString 1 to a file called C:\FLOWER.TIF.

```
(GFMTIFFModule.TIFF) connectmodule
2 !Direction
1 getglobalstring !SourceFileName
>DefaultExtension
(C:\FLOWER) exch mergestrings !DestinationFileName
3 !Action
```

Errors stack underflow, typecheck

---

sub command

Syntax *value1 value2* **sub** *result*

Applies to All editions

Description Returns the subtraction of *value2* from *value1*.

Remarks The two top values can be any number. If any of the two top values are a string, the typecheck error will occur.

See also add, div, mul

Example The following example pushes two values to the stack and subtracts them.

```
10 20 sub
```

Errors stack underflow, typecheck

---

xor command

Syntax *value1 value2* x**or** *result*

Applies to All editions

Description Performs a logical XOR operation of the two top values on the stack. The *value1* and v*alue2* will be removed from the stack

Remarks The two top values can be any number. If any of the two top values are a string, will the typecheck error occur.

See also and, not, or

Example The following example does a logical XOR comparsion:

```
7 3 xor -> 4
12 3 xor -> 15
```

Errors stack underflow, typecheck

tiffclosefile command

Syntax **tiffclosefile**

Applies to Standard and Advanced Edition

Description Closes the current opened TIFF file.

Remarks This command should be used when a TIFF file has been opened and read.

See also tiffgetheader, tiffopenfile

Example The following example closes the currently opened TIFF file.

```
tiffclosefile
```

TIFFBitsPerSample variable

Syntax **TIFFBitsPerSample**

Applies to Standard and Advanced Edition

Description Holds the number of bits per component for the currently loaded TIFF file.

Remarks In most cases will this variable be 1 for bilevel images and 8 for grayscale and color images.

See also tiffgetheader, tiffopenfile

TIFFCompression variable

Syntax **TIFFCompression**

Applies to Standard and Advanced Edition

Description Holds the compression method for the currentlty loaded TIFF file.

Remarks The TIFFCompression value is a copy of the Compression tag in the currently opened TIFF file

Value	Meaning
1	Uncompressed
2	CCITT 1D
3	Group 3 Fax
4	Group 4 Fax
5	LZW
6	JPEG
32773	PackBits

See also tiffgetheader, tiffopenfile

tiffgetheader command

Syntax **tiffgetheader**

Applies to Standard and Advanced Edition

Description Reads the header of the currently opened TIFF file.

Remarks Before the command the **tiffopenfile** command must be executed. The **tiffheader** command will create the following values on GFMDict:

```
TIFFImageWidth
TIFFImageLength
TIFFCompression
TIFFSubFileType
TIFFNewSubFileType
TIFFBitsPerSample
TIFFPhotometricInterpretation
TIFFSamplesPerPixel
TIFFRowsPerStrip
TIFFXResolution
TIFFYResolution
TIFFPlanarConfiguration
TIFFResolutionUnit
TIFFStripOffsets (Only single strip files)
TIFFStripByteCounts (Only single strip files)
```

All other tags will be ignored. Tags that doesn't exist will not be saved in GFMDict.

See also tiffclosefile, tiffopenfile

Example The following example opens a TIFF file named C:\FILE1.TIF and makes it accesible to the GFM Engine.

```
(C:\FILE1.TIF) tiffopenfile
tiffgetheader
```

Errors this is not a tiff file

TIFFImageLength variable

Syntax **TIFFImageLength**

Applies to Standard and Advanced Edition

Description Holds the number of rows of pixels for the currently loaded TIFF file.

See also tiffgetheader, tiffopenfile

TIFFImageWidth variable

Syntax **TIFFImageWidth**

Applies to Standard and Advanced Edition

Description Holds the number of columns, i.e. the number of pixels per row for the currently loaded TIFF file.

See also tiffgetheader, tiffopenfile

---

tiffopenfile command

Syntax *string* **tiffopenfile**

Applies to Standard and Advanced Edition

Description Opens the file defined as *string* as a TIFF file.

Remarks This command must be used before any manipulation can be done with a TIFF file. If the file defined as *string* can't be found will an error occur.

See also tiffclosefile, tiffgetheader

Example The following example opens a TIFF file named C:\FILE1.TIF and makes it accesible to the GFM Engine.

```
(C:\FILE1.TIF) tiffopenfile
```

Errors stack underflow, typecheck

TIFFPhotometericInterpretation variable

Syntax **TIFFPhotometericInterpretation**

Applies to Standard and Advanced Edition

Description Holds the color space for the currently loaded TIFF file.

Remarks The TIFFPhotometericInterpretation variable is a copy of the PhotometericInterpretation tag in the currently opened TIFF file

Value	Meaning
0	WhiteIsZero
1	BlackIsZero
2	RGB
3	Palette Color
4	Transparency Mask
5	Separated – usually CMYK

See also tiffgetheader, tiffopenfile

TIFFRowsPerStrip variable

Syntax **TIFFRowsPerStrip**

Applies to Standard and Advanced Edition

Description Holds the number of rows per strip for the currently loaded TIFF file.

Remarks TIFF image data is organized into one or several strips

See also tiffgetheader, tiffopenfile

TIFFSamplesPerPixel variable

Syntax **TIFFSamplesPerPixel**

Applies to Standard and Advanced Edition

Description Holds the number of components for the currently loaded TIFF file.

Remarks TIFFSamplesPerPixel is usually 1 for bilevel, grayscale and palette-color images. TIFFSamplesPerPixel is usually 3 for RGB images and 4 for CMYK images.

See also tiffgetheader, tiffopenfile

---

time command

Syntax **time**

Applies to All editions

Description Pushes the current time to the top of the stack. Note that the time will be a string value in the format HHMMSS.

See also date

---

trimstring command

Syntax *string* **trimstring**

Applies to All editions

Description Removes spaces at the beginning and end of a string.

See also leftstring, rightstring

Example This example trims a string.

```
( Image name is: ) trimstring
```

Errors stack underflow, typecheck

True variable

Syntax **True**

Applies to All eitions

Description Contains the value -1.

Remarks The variable is generated internally each time when a script starts.

---

ucasestring command

Syntax *string* **ucasestring**

Applies to All editions

Description Converts a string to uppercase.

See also lcasestring

Remarks Only lowercase letters are converted to uppercase, all uppercase letters and nonletter characters remain unchanged.

Example This example converts a string to uppercase.

```
(flower.tif) ucasestring
```

Errors stack underflow, typecheck

---

verbosity command

Syntax *value* **verbosity**

Applies to All editions

Description Sets the verbosity level. I.e. the level of messages returned by the GFM engine to the SystemMonitor. The meaning of each value for the verbosity is according to the following table:

Value	Description
0	Normal messaging. The default value.
1	Show all commands in the GFM script as they are executed.
2	Show all commands and the stack contents at each command.
3	Show all commands, the stack contents and the GFMDict contents at each command.
4	Shows all commands and extra comments to each command. Usefull for fault finding in scripts. Note: can create a lot of text in the system monitor.

Remarks At start-up, the verbosity is set to 0 which means normal messaging. It means that only error messages will be displayed in the SystemMonitor. You can of course display more information via the GFM script and the **print** command.

See also logfile, print

Example This example sets the verbosity level to 2.

```
2 verbosity
```

Errors stack underflow, invalid parameter, typecheck

Version variable

Syntax **Version**

Applies to All editions

Description Contains the version number of OIF for the current image in the PixelArea.

Remarks Version is currently always 1. If the version number is anything but 1, we advise you to reject the image.

>Version property

Syntax >**Version**

Applies to Standard and Advanced Edition

Description Returns a string value indicating the current version of the module.

Remarks The value returned will be on top of the stack after the command has been sent to the module.

Example This example connects to the TIFF module and returns the version of the module to the stack.

```
(GFMTIFFModule.TIFF) connectmodule
>Version
```

Errors no connected module

ViewColorMode procedure

Syntax **ViewColorMode**

Applies to Standard and Advanced Edition

Description Displays the color mode for the current image in the PixelArea in the system monitor.

Remarks The procedure is in the STANDARD.LIB library. The value returned by the procedure is the same as the value stored in the variable *ColorMode*.

ViewCompression procedure

Syntax **ViewCompression**

Applies to Standard and Advanced Edition

Description Displays the compression used for the current image in the PixelArea in the system monitor.

Remarks The procedure is in the STANDARD.LIB library. The value returned by the procedure is the same as the value stored in the variable *Compression*.

ViewNumberofPixels procedure

Syntax **ViewNumberofPixels**

Applies to Standard and Advanced Edition

Description Displays the number of pixels in width and height for the current image in the PixelArea in the system monitor.

Remarks The procedure is in the STANDARD.LIB library.

ViewResolution procedure

Syntax **ViewResolution**

Applies to Standard and Advanced Edition

Description Displays the resolution of the current image in the PixelArea in the system monitor.

Remarks The procedure is in the STANDARD.LIB library. It uses the xResolution variable when displaying the resolution. If you know that you will have different resolutions in height and width, you must write your own procedures for displaying the resolutions.

writeimagefile command

Syntax *filename imageheader* **writeimagefile**

Applies to Standard and Advanced Edition

Description Writes the contents of the PixelArea to disk as the file *filename*.

Remarks Before this command can be executed, the new *filename* must be pushed to the stack followed by the complete *imageheader*. The easiest way to have the complete *imageheader* is to use the procedure **PushImageHeaderToStack** in the STANDARD.LIB library.

See also closeimagefile, getimageheader, loadimage, openimagefile

Example This example saves the image in the PixelArea to disk using the **PushImageHeaderToStack** procedure as C:\IMAGE1.OIF.

```
C:\IMAGE1.OIF)
PushImageHeaderToStack
writeimagefile
```

Errors no current image, stack underflow, typecheck

writestring command

Syntax *filehandle string* **writeimagefile**

Applies to All Editions

Description Writes the string *string* to the file refered to as *filehandle*.

Remarks *string* will be written at the current file position which normally is at the end of the file. The string will be ended by a linefeed and carriage return causing the file to have a new line at the end of the file and this will be the new current file position.

See also file

Example This example creates a file called STATUS.LOG in the directory Images on disc C and writes the string ‘File has been updated’ to the file.If the file exists will the existsing file be removed and overwritten by the new data. If the file doesn’t exist will be created.

```
(C:\Images\STATUS.LOG) (w) file
/F exch def
F (File has been updated) writestring
F closefile
```

Errors no current image, stack underflow, typecheck

xResolution variable

Syntax x**Resolution**

Applies to Standard and Advanced Edition

Description Contains the resolution in pixels per meter in the width direction for the current image in the PixelArea.

year command

Syntax **year**

Description Pushes the value of the year to the top of the stack. Note that the year will be a string value.

See also month, time, year

yResolution variable

Syntax **yResolution**

Applies to All editions

Description Contains the resolution in pixels per meter in the height direction for the current image in the PixelArea.

$OnError procedure

Syntax **$OnError**

Applies to All editions

Description Contains the procedure that will be executed if the GFM Engine enters Error mode, i.e. if there is an error in a script.

Remarks By entering the correct code in the **$OnError** procedure can you for example start the GFM Mailer module and send an e-mail to a predefined number of addresses.

By default is the **$OnError** procedure empty.

$SysPath variable

Syntax **$SysPath**

Applies to All editions

Description Contains the GFM Main directory.

Remarks The GFM Main directory is usually C:\GFM. It is set during the installation of the GFM engine.

$WinSysPath variable

Syntax **$SysPath**

Applies to All editions

Description Contains the Windows Main directory.

Remarks The Windows Main directory is usually C:\WINNT.