OpenPrinting

Vector Printer Driver Application Program Interface (PDAPI) Specification

Version-1.0 RC2

(2006-6-29)

OpenPrinting Vector Printer Driver Application Interface Specification

Table of Contents

1

I.Introduction

This specification defines Application Program Interface (API) which is used in printing environment offered by OpenPrinting. Here, the word “printer driver” refers software library which translate printing data generated by application programs into data stream which can be interpreted by printers. The driver offers functions for translating each drawing commands.

This specification tries to make abstraction of graphics drawing functions which is supported by printer languages and offers them as API, and make enable to access printer graphics drawing functions without printer model specific knowledge by render.

This specification covers from ink jet printers which handles raster data only to high-end laser printers which have high-level graphics functions. Specifically, by providing access method to high level graphics drawing function, it improves printing performance. The specification also covers black and white to full color printers.

As examples of render, there exist Ghostscript and X Print Server. The printer driver based on this specification does not assume any specific type of render.

II.Linking with Render

Loading and Calling Printer Driver

Printer drivers can be provided in forms of static or dynamic library. Liking to the library module will be followed by methods prepared by operating systems, therefore, it is not specified in this document.

Printer drivers can be loaded as a library code into the same memory space of render. It is also executed as server process which communicate with render via RPC call (the driver is loaded into different memory space of render). In case of server type, the server process links printer drivers. The specification for RPC, other documents should care of that.

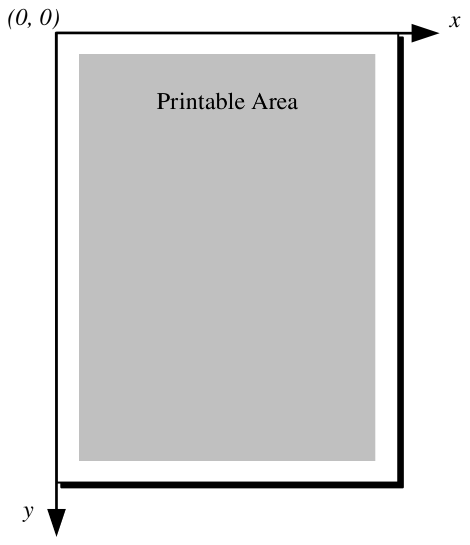
Printing data stream, which is generated by printer driver corresponding to each drawing APIs, are send back to render through file descriptor, which is designated by fsgpdOpenPrinter() call. The printer driver does not need to generate data stream API by API, rather than, it may generates data stream asynchronously to API call or generates whole data at fsgpdEndPage() call. In order to receive the steam data properly, the render should handle input stream from the driver in any timing between fsgpdOpenPrinter() and fsgpdClosePrinter() calls.

Notes about Function Parameters

In case that a driver is linked as a library, data area which is designated by pointers of arguments of each functions are possibly referred until the process of the page on which drawing functions are executed. Therefore, the caller should keep the area until it calls fsgpdEndPage() operation. In addition, deallocation from memory of such area should be done by the caller after fsgpdEndPage() operation.

III.Graphic Model

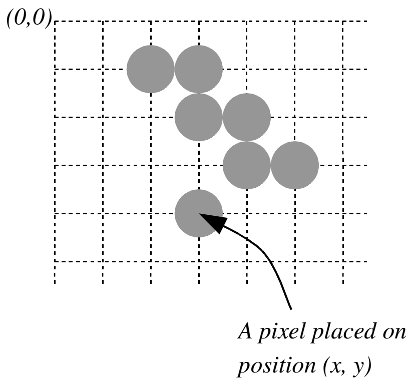
Coordinate System



It takes the coordinate system such that the origin takes physical upper left corner of media and the unit is device pixel. Positive value in abscissa moves origin to right direction, positive value in ordinate moves origin to lower direction.

The type of coordinate takes singed, fixed point 32 bits value, where integer takes 24 bits and decimal takes 8 bits (type "FSGPD_FIX").

A coordinate of a pint shows the horizontal and vertical distance from origin to intersection of grid. Pixels are assumed to be placed on intersections of grid (Grid Intersection Model). The relationship between coordinate grid and pixels are shown in the right figure.



Objects

Following kinds of graphics objects are recognized and handled under this specification.

- Path
- Bitmap Image
- Scanline
- Raster Image
- Text

The printer driver maintains "Graphics State", in which contains properties for drawing graphics. Only one graphics state is effective at one time, however, it can be switched by fsgpdSaveGS() and fsgpdRestoreGS().

CTM

Coordinate Translation Matrix (CTM), which is used to translate coordinate system to render coordinate to device coordinate, is maintained in Graphics State. CTM is presented in following matrix form.

$$\begin{bmatrix} x_{dev} & y_{dev} & 1 \end{bmatrix} = \begin{bmatrix} x_{ren} & y_{ren} & 1 \end{bmatrix} \times \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ e & f & 1 \end{bmatrix}$$

For detail of CTM, refer description in fsgpdResetCTM/fsgpdSetCTM/fsgpdGetCTM.

Color

As Color mapping, FSGPD_CMAP_DIRECT (direct color) and FSGPD_CMAP_INDEXED (indexed color) will be supported. At this time, FSGPD_CMAP_INDEXED is pending.

Following color space can be used.

Color Space Macro	Color Space
FSGPD_CSPACE_BW	Black and White
FSGPD_CSPACE_DEVICEGRAY	Grayscale
FSGPD_CSPACE_DEVICECMY	CMY
FSGPD_CSPACE_DEVICECMYK	CMY and Black
FSGPD_CSPACE_DEVICERGB	Device RGB
FSGPD_CSPACE_STANDARDRGB	sRGB
FSGPD_CSPACE_STANDARDRGB64	scRGB

At this time FSGPD_CSPACE_BW, FSGPD_CSPACE_DEVICEGRAY and FSGPD_CSPACE_STANDARDRGB will be supported.

Alpha Composition

This specification supports Alpha Composition. Alpha value which is saved in the Graphics State and is effective to whole drawing and alpha for pixel by pixel of bitmap data which is represented by aRGB format.

(However, alpha is not supported at this moment – we need more consideration on this)

Scan Rule

Pixel placement on painting of path and scan line operation should be done without painting problem which is sometimes occurred as a result of raster operation when painting a region by dividing into small polygon regions. As an example of scan rule, there is right-bottom exclusive method. For actual implementation, it depends on page description language, therefore this specification does not determine specific implementation.

IV.Operations

Creating and Managing Print Contexts

Functions to create and delete printer driver contexts. These functions MUST be supported by all drivers.

fsgpdOpenPrinter

Name

fsgpdOpenPrinter – Create printer context.

Synopsis

```
FSGPD_DC fsgpdOpenPrinter(
      FSGPD_INT outputFD,
      FSGPD_CHAR *printerModel,
      FSGPD_INT apiVersion[2],
      FSGPD_INT *nApiEntry,
      FSGPD_INT (**apiEntry[])());
```

Arguments

outputFD – file descriptor to write print data stream

printerModel – Printer Model Name in UTF-8. If NULL, it assumes default model by the printer driver.

apiVersion – Vector Printer Driver Specification Version number which the driver supports. ApiVersion[0] is the major and apiVersion[1] is the minor value of the version respectively.

nApiEntry – array size of apiEntry.

apiEntry – address of function pointer array to each API entry.

Description

This function initializes printer driver and designates the file descriptor for writing printing data stream. Printing data generated by drawing function and others are written through this file descriptor. For printerModel, the printer model which is defined in printer model data base.

The driver writes debug messages to stderr, so stderr cannot be designated

fsgpdOpenPrinter() stores function pointers for each API entry to apiEntry and returns. NULL will be stored for functions not supported by the driver. Array size is set to nApiEntry.

Only fsgpdOpenPrinter() is exported by printer driver library. For other APIs, they are referred by entry addresses stored in apiEntry.

When it finished normally, printer context is returned as return value. The printer context is a index number or pinter value to manager opened printer and is designated as first parameter for APIs thereafter. So, the caller should remember this value.

Return Value

If successful, this function returns printer context value (positive value). This function returns -1 when error is occurred and error code is stored in fsgpdErrorNo.

fsgpdClosePrinter

Name

fsgpdClosePrinter – Delete printer context.

Synopsis

```
FSGPD_RESULT fsgpdClosePrinter(
        FSGPD_DC printerContext);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

Description

It finished printing process and deletes printer context. Closing of outputFD, which is designated at fsgpdOpenPrinter(), should be closed by caller side.

If this function is called when printing job is not completed, such as a case no fsgpdEndJob() is called, data in process is discarded, but it does not guarantee that printing data which is cancel printing job normally is generated.

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

Job Control Operations

These functions operate job control. These functions MUST be supported by all drivers. One print job consist of one or more documents. One document consist of one or more pages. If a job consist of only one document, fsgpdStartDoc() and fsgpdEndDoc() functions can be omitted. fsgpdStartJob(), fsgpdEndJob(), fsgpdStartPage() and fsgpdEndPage() functions cannot be omitted.

fsgpdStartJob

Name

fsgpdStartJob – Information that start print job.

Synopsis

```
FSGPD_RESULT fsgpdStartJob(
        FSGPD_DC printerContext,
        FSGPD_CHAR *jobInfo);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

jobInfo – Set property of print job. If NULL, default values by printer driver are set. Format and contents are described later.

Description

fsgpdStartJob() informs the start of print job.

Property of the print job is set by jobInfo.

It is depend on printer device characteristics whether job nesting (fsgpdStartJob() - fsgpdEndJob() are called between fsgpdStartJob() - fsgpdEndJob()) is possible. If the printer does not allow job nesting, the driver returns error for this call.

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdEndJob

Name

fsgpdEndJob –Information that end print job.

Synopsis

```
FSGPD_RESULT fsgpdEndJob(
        FSGPD_DC printerContext);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

Description

fsgpdEndJob() informs the end of print job.

Job property values are reset to initial values.

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdAbortJob

Name

fsgpdAbortJob – Cleanup for abort print job.

Synopsis

```
FSGPD_RESULT fsgpdAbortJob(
        FSGPD_DC printerContext);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

Description

fsgpdAbortJob() designates clearup operation to abort print job. It may perform termination of print job and create print data to clearnup printer state. It is depends on the timing when this function is called whether the print job are sent to target printer or not, so calling this function does not guarantee that nothing is printed on the printer, and papers may be wasted. However, after this function is executed, the printer should be initial state and next job has to be accepted normally.

Job property values are reset to initial values.

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdStartDoc

Name

fsgpdStartDoc – Start document.

Synopsis

```
FSGPD_RESULT fsgpdStartDoc(
        FSGPD_DC printerContext,
        FSGPD_CHAR *docInfo);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

docInfo – Set property of print document. If NULL, default values by printer driver are set. Format and contents are described later.

Description

fsgpdStartDoc() starts printing of a document.

Document property is set through docInfo. The document property is effective until fsgpdEndDoc() is called. When fsgpdStartDoc() is called again after the fsgpdEndDoc() call, the docInfo at this call is effective and properties not in docInfo are reset to initial value.

It is depend on printer device characteristics whether document nesting (fsgpdStartDoc() - fsgpdEndDoc() are called between fsgpdStartDoc() - fsgpdEndDoc()) is possible. If the printer does not allow document nesting, the driver returns error for this call

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdEndDoc

Name
fsgpdEndDoc – End document

Synopsis
```
FSGPD_RESULT fsgpdEndDoc(
        FSGPD_DC printerContext);
```

Arguments
printerContext – printer context value returned by fsgpdOpenPrinter()

Description
fsgpdEndDoc() ends document printing.

Return Value
If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdStartPage

Name
fsgpdStartPage – Start print page.

Synopsis
```
FSGPD_RESULT fsgpdStartPage(
        FSGPD_DC printerContext,
        FSGPD_CHAR *pageInfo);
```

Arguments
printerContext – printer context value returned by fsgpdOpenPrinter()

pageInfo – Page property. If NULL, default values by printer driver are set. Format and contents are described later.

Description
fsgpdStartPage() starts printing of a page.

Return Value
If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdEndPage

Name
fsgpdEndPage – End print page.

Synopsis
```
FSGPD_RESULT fsgpdEndPage(
        FSGPD_DC printerContext);
```

Arguments
printerContext – printer context value returned by fsgpdOpenPrinter()

Description
fsgpdEndPage() ends a page printing. After this function is called, memory area which is allocated by caller can be released.

Return Value
If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdQueryDeviceCapability

Name
fsgpdQueryDeviceCapability – Query device capabilities.

Synopsis
```
FSGPD_ERRROR fsgpdQueryDeviceCapability(
        FSGPD_DC printerContext,
        FSGPD_FLAG queryflag,
        FSGPD_INT buflen,
        FSGPD_BYTE *infoBuf);
```

Arguments
printerContext – printer context value returned by fsgpdOpenPrinter()

queryflag – flags showing device capability queried.

buflen – buffer length which is prepared as *infoBuf

infoBuf – a buffer to put device capability information

Description
This function is used to query capabilities which is supported by printer device or driver.

If infoBuf is set to NULL, the printer driver returns buffer length to save information in int format In this case, the caller should queryflag and buflen correctly.

Queryflag is used to set bit flag to be queried. More than two flags can be set. Following macro should be used.

```
typedef enum _FSGPD_QUERYINFOFLAGS {
  FSGPD_QF_DEVICERESOLUTION =    0x00000001,
  FSGPD_QF_MEDIASIZE =           0x00000002,
  FSGPD_QF_PAGEROTATION =        0x00000004,
  FSGPD_QF_MEDIANUP =            0x00000008,
  FSGPD_QF_MEDIADUPLEX =         0x00000010,
  FSGPD_QF_MEDIASOURCE =         0x00000020,
  FSGPD_QF_MEDIADESTINATION =    0x00000040,
  FSGPD_QF_MEDIATYPE =           0x00000080,
  FSGPD_QF_MEDIASIZE =           0x00010000,  /* only for fsgpdQueryDeviceInfo */
  FSGPD_QF_PRINTREGION =         0x00020000   /* only for fsgpdQueryDeviceInfo */
} FSGPD_QUERYINFOFLAGS;
```

The driver returns queried information in ASCII text format. If buffer doesn't have enough length to write, the written information is truncated. In this case, the driver doesn't set error code and returns without error.

The format of information written in infoBuf follows the property format which is used by fsgpdStartJob() and other functions. For example, if resolution is queried, the resolution list supported is written in following format. The first one of the list is default setting.

```
"updf:DeviceResolution=deviceResolution_600x600,deviceResolution_1200x1200"
```

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdQueryDeviceInfo

Name

fsgpdQueryDeviceInfo – Query device information.

Synopsis

```
FSGPD_RESULT fsgpdQueryDeviceInfo(
        FSGPD_DC printerContext,
        FSGPD_FLAG queryflag,
        FSGPD_INT buflen,
        FSGPD_CHAR *infoBuf);
```

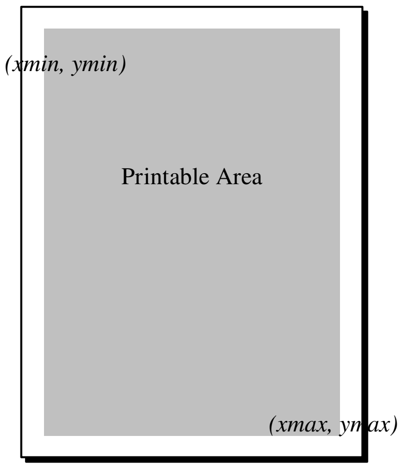
Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

queryflag – flags showing device capability queried.

buflen – buffer length which is prepared as *infoBuf

infoBuf – a buffer to put device capability information



Description

fsgpdQueryDeviceInfo() is used to query current setting of printer device or driver.

If infoBuf is set to NULL, the printer driver returns buffer length to save information in int format In this case, the caller should queryflag and buflen correctly.

Queryflag is used to set bit flag to be queried. More than two flags can be set. Same macro which is used in fsgpdQueryDeviceCapability() should be used.

The driver returns queried information in ASCII text format. If buffer doesn't have enough length to write, the written information is truncated. In this case, the driver doesn't set error code and returns without error.

The format of information written in infoBuf follows the property format which is used by fsgpdStartJob() and other functions.

Query about printable area is respond in current printing resolution and format is shown as follow (see the figure below.) These values follows in orientation setting.

PrintRegion=xmin,ymin,xmax,ymax

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

Attribute of Job, Document and Page for Job Control Operations

Properties for job, document and page are set as parameter of fsgpdStartJob(), fsgpdStartDoc(), fsgpdStartPage(). Supported parameters and their values are provided by printer driver database file distributed with drivers.

The property values are set in ASCII strings, thus follows the format below.

<*scheme*>:<*key*>=<*value*>{,<*value*>}*{;<*key*>=<*value*>{,<*value*>}*}*

- <*scheme*>: The specification which defines name space.
- <*key*>: Name of the property.
- <*value*>: Value of the property.

Several <*value*> can be designated separating by ”,” (comma). If several values are designated, the printer driver searches it from begging of the string and takes possible value.

Several pairs - <*key*>=<*value*>{,<*value*>}* can be designated separating by ”;” (semi-colon)

For <*scheme*>, “updf” can be designated.

If scheme is updf, the format follows the definition by UPDF (Universal Printer Driver Description File) by Printer Working Group.

It if preferable that the driver ignores unknown property, for future extensions.

Major properties are shown in the table below.

Property	Name	Value	Effective in		
			Job	Doc	Page
Orientation	MediaPageRotation	landscape portrait reverse-landscape reverse-portrait	✔	✔	✔
Page Size	MediaSize	iso_a4_210x297mm iso_a3_297x420mm jpn_hagaki_100x148mm ...	✔	✔	✔
Number Up	MediaNUp	nup-1x1 nup-2x1 nup-2x2 ...	✔	✔	✔
Duplex	MediaDuplex	simplex duplex-long-edge duplex-short-edge	✔	✔	✔
Resolution	DeviceResolution	deviceResolution_1200x1200 deviceResolution_600x600 ...	✔	✔	✔
input-bin	MediaSource	manual continuous roll	✔	✔	✔

Property	Name	Value	Effective in		
			Job	Doc	Page
		cut-sheet			
		proprietary-value			
		device-setting			
output-bin	MediaDestination	standard	✔	✔	✔
		proprietary-value			
		device-setting			
Media Type	MediaType	cardstock	✔	✔	✔
		continuous			
		stationery			
		stationery-fine			
		...			
Media Copy	MediaCopy	1, 2, ...			
Print Quality	PrintQuality	draft	✔	✔	✔
		high			
		normal			

The values are always effective in order of Job < Doc < Page. For example, in a job which is set to landscape, if a page is set to portrait, the page is set to portrait and other pages are set to landscape.

Graphics State Object Operations

These functions operate graphics state object. All of these functions are OPTIONAL.

fsgpdResetCTM

Name

fsgpdResetCTM – Initialize CTM of the Graphics State Object.

Synopsis

```
FSGPD_RESULT fsgpdResetCTM(
      FSGPD_DC printerContext);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

Description

fsgpdResetCTM() initializes CTM of Graphics State Object. The initial setting of CTM is as follow.

$$.\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdSetCTM

Name

fsgpdSetCTM – Set CTM to the Graphics State Object.

Synopsis

```
FSGPD_RESULT fsgpdSetCTM(
      FSGPD_DC printerContext,
      FSGPD_CTM *pCTM);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

pCTM – Pointer to the CTM structure. It contains 6 FSGPD_FLOAT elements - a, b, c, d, e and f.

Description

fsgpdSetCTM() sets a CTM to the Graphics State Object.

Device coordinate system value [xdev, ydev] is represented by CTM and render coordinate system value [xren, yren] in following equation.

$$\begin{bmatrix} x_{dev} & y_{dev} & 1 \end{bmatrix} = \begin{bmatrix} x_{ren} & y_{ren} & 1 \end{bmatrix} \times \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ e & f & 1 \end{bmatrix}$$

Structures

```
typedef struct _FSGPD_CTM {
      FSGPD_FLOAT a, b, c, d, e, f;
} FSGPD_CTM;
```

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdGetCTM

Name

fsgpdGetCTM – Get CTM from the Graphics State Object.

Synopsis

```
FSGPD_RESULT fsgpdGetCTM(
      FSGPD_DC printerContext;
      FSGPD_CTM *pCTM);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

pCTM – Pointer to the CTM structure. It contains 6 FSGPD_FLOAT elements - a, b, c, d, e and f.

Description

fsgpdGetCTM() retrieves a CTM from Graphics State Object.

Structures

```
typedef struct _FSGPD_CTM {
      FSGPD_FLOAT a, b, c, d, e, f;
} FSGPD_CTM;
```

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdInitGS

Name

fsgpdInitGS – Initialize the Graphics State parameters.

Synopsis

```
FSGPD_RESULT fsgpdInitGS(
      FSGPD_DC printerContext);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

Description

fsgpdInitGS() initializes parameters in Graphics State Object. Graphics state is a set of values which is managed by printer driver and contains parameters for drawing mode and others. There is always one Graphics state object which is effective as current graphics state and used as current setting. Graphics state object can be pushed or popped on/from driver managed stack by fsgpdSaveGS() and fsgpdRestoreGS() operations described later in this document. These operations are used when changing graphics state temporary and restores later.

Graphics state object keeps its setting values between fsgpdStartJob() and fsgpdEndJob(), unless fsgpdInitGS() is called. At fsgpdStartJob(), same values as fsgpdInitGS() is called are set.

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdSaveGS

Name

fsgpdSaveGS – Save the Graphics State parameters.

Synopsis

```
FSGPD_RESULT fsgpdSaveGS(
        FSGPD_DC printerContext);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

Description

fsgpdSaveGS() pushes graphics state object parameters onto the driver managed stack.

The size of the stack is limited, but at least n times of this operation can be called. If more than n times this function is called without calling fsgpdRestoreGS(), error is returned.

After n times fsgpdSaveGS() is called, same times fsgpdRestoreGS() can be executed. More than n time fsgpdRestoreGS() is called, it results in error (BADREQEST).

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdRestoreGS

Name

fsgpdRestoreGS – Restore the Graphics State parameters.

Synopsis

```
FSGPD_RESULT fsgpdRestoreGS(
        FSGPD_DC printerContext);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

Description

Saved graphics state object is restored to current value.

After n times fsgpdSaveGS() is called, same times fsgpdRestoreGS() can be executed. More than n time fsgpdRestoreGS() is called, it results in error (BADREQEST).

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdQueryColorSpace

Name

fsgpdQueryColorSpace – Obtain the color space which can be used by the driver.

Synopsis

```
FSGPD_RESULT fsgpdQueryColorSpace(
        FSGPD_DC printerContext,
        FSGPD_CSPACE *pcspace
        FSGPD_INT *pnum);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

pcspace –Pointer to the Color Space enum. array.

pnum – Pointer to the number of the Color Space enum. array elements.

Description

Retrieves list of color space which can be treated by the driver from graphics state object.

The caller should allocate at least n as the number of elements of an array pcspace[] which stores color space. Also the caller should call this function by setting it's maximum number of elements in *pnum. The number of elements retrieved is set to *pnum. If the number of elements exceeds the number of elements designated, this function returns necessary number in *pnum.

The driver returns the color space into array in preferable order.

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdSetColorSpace

Name

fsgpdSetColorSpace – Set the current color space which can be dealt by the driver to the Graphics State Object.

Synopsis

```
FSGPD_RESULT fsgpdSetColorSpace(
        FSGPD_DC printerContext,
        FSGPD_CSPACE cspace);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

cspace –Color Space enum.

Description

fsgpdSetColorSpace() sets color space handled by the driver into graphics state object.

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdGetColorSpace

Name

fsgpdGetColorSpace – Get the current color space from the Graphics State Object.

Synopsis

```
FSGPD_RESULT fsgpdGetColorSpace(
        FSGPD_DC printerContext,
        FSGPD_CSPACE *pcspace);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

pcspace – Pointer to the Color Space enum.

Description

fsgpdGetColorSpace() retrieves color space treated by the driver from graphics state object.

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdQueryROP

Name

fsgpdQueryROP – Obtain the ROP which can be used by the driver.

Synopsis

```
FSGPD_RESULT fsgpdQueryROP(
        FSGPD_DC printerContext,
        FSGPD_INT *pnum,
        FSGPD_ROP *prop);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

pnum – Pointer to the number of the ROP mode array elements.

prop –Pointer to the ROP mode. array.

Description

fsgpdQueryROP retrieves a list of ROP (Raster Operations) which can be treated by the driver from graphics state object.

If NULL is set in *prop, the number of ROP supported is returned in *pnum.

So, at least the number of ROP supported by the operation above should be allocated for the array prop[].

The number of elements is returned in *pnum. If the number of ROP exceeds the size of array designated, it returns an error (FSGPD_PARAMERROR) and necessary number of elements is returned in *pnum.

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdSetROP

Name

fsgpdSetROP – Set the ROP mode to the Graphics State Object.

Synopsis

```
FSGPD_RESULT fsgpdSetROP(
        FSGPD_DC printerContext,
        FSGPD_ROP rop);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

rop –ROP mode (ROP3 code)

Description

fsgpdSetROP() sets raster operation code designated by rop to graphics state object.

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdGetROP

Name

fsgpdGetROP – Get the ROP mode from the Graphics State Object.

Synopsis

```
FSGPD_RESULT fsgpdGetROP(
        FSGPD_DC printerContext,
        FSGPD_ROP *prop);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

prop – Pointer to he ROP mode.

Description

fsgpdGetROP() retrieves raster operation code currently set in graphics state.

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdSetFillMode

Name

fsgpdSetFillMode – Set the filling mode.

Synopsis

```
FSGPD_RESULT fsgpdSetFillMode(
        FSGPD_DC printerContext,
        FSGPD_FILLMODE fillmode);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

fillmode – Filling mode enum. Either of FSGPD_FILLMODE_EVENODD, FSGPD_FILLMODE_WINDING can be designated.

Description

fsgpdSetFillMode() sets a fill mode to graphics state object.

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdGetFillMode

Name

fsgpdGetFillMode – Get the filling mode.

Synopsis

```
FSGPD_RESULT fsgpdGetFillMode(
        FSGPD_DC printerContext,
        FSGPD_FILLMODE *pfillmode);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

pfillmode – Pointer to the filling mode enum.

Description

fsgpdGetFillMode retrieves the fill mode currently set in graphics state object.

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdSetAlphaConstant

Name

fsgpdSetAlphaConstant – Set the alpha blending constant.

Synopsis

```
FSGPD_RESULT fsgpdSetAlphaConstant(
        FSGPD_DC printerContext,
        FSGPD_FLOAT alpha);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

alpha – Alpha blending constant. Between 0.0 and 1.0

Description

Set alpha constant, which is transparent ratio, to the graphics state object. The value should be in range of 0.0 to 1.0. If a value extends this range, it will be truncated to 0.0 or 1.0.

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdGetAlphaConstant

Name

fsgpdGetAlphaConstant – Get the alpha blending constant.

Synopsis

```
FSGPD_RESULT fsgpdGetAlphaConstant(
      FSGPD_DC printerContext,
      FSGPD_FLOAT *palpha);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

palpha – Pointer to the alpha constant value.

Description

fsgpdGetAlphaConstant() retrieves alpha constant setting from graphics state object.

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdSetLineWidth

Name

fsgpdSetLineWidth – Set the line width.

Synopsis

```
FSGPD_RESULT fsgpdSetLineWidth(
      FSGPD_DC printerContext,
      FSGPD_FIX width);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

width – Line width value.

Description

fsgpdSetLineWidth() sets line width for stroke operations to graphics state object. The line width should be set by units in device coordinate system.

The treatment of a line width less than one depends on the device or driver implementation.

The maximum line width depends on device.

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdGetLineWidth

Name

fsgpdGetLineWidth – Get the line width.

Synopsis

```
FSGPD_RESULT fsgpdGetLineWidth(
        FSGPD_DC printerContext,
        FSGPD_FIX *pwidth);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

width – Pointer to the Line width value.

Description

fsgpdGetLineWidth() retrieves line width for stroke operations to graphics state object. The line width should be set by units in device coordinate system.

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdSetLineDash

Name

fsgpdSetLineDash – Set the line dash pattern.

Synopsis

```
FSGPD_RESULT fsgpdSetLineDash(
        FSGPD_DC printerContext,
        FSGPD_FIX pdash[],
        FSGPD_INT num);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

pdash – Pointer to the line dash pattern array.

num – Number of the line dash pattern array elements.

Description

fsgpdSetLineDash() sets a dash pattern to graphics state object.

When the painting mode is paintOpaque, the first element in the array pdash[] is the length for foreground color, the next element is a length for background color, and so on.

When the painting mode is paintTransparent, the first element in the array pdash[] is the length for foreground color, the next element shows the length of a line segment which is NOT painted, and so on.

The length designated in the array pdash[] is designated by the unit in the device coordinate system.

If the number of elements of the array pdash[] is odd, the dash pattern is created as if the number of element is double the number of element of the array. In the second cycle of this case, the first element of the array is treated as the length for background color in case of paintOpaque mode and for not-painted length in case of paintTransparent mode.

The maximum number of elements depend on the device.

If num is zero, solid lines are drawn.

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdGetLineDash

Name

fsgpdGetLineDash – Get the line dash pattern.

Synopsis

```
FSGPD_RESULT fsgpdGetLineDash(
      FSGPD_DC printerContext,
      FSGPD_FIX pdash[],
      FSGPD_INT *pnum);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

pdash – Pointer to the line dash pattern array.

pnum – Pointer to the number of the line dash pattern array elements.

Description

fsgpdGetLineDash() retrieves dash pattern for stroke from graphics state object.

The caller should allocate at least n elements for array pdash[]. Also, the caller should set the maximum number of element of pdash[] to *pnum. The number elements returned is set in *pnum. If the number of elements exceeds the *pnum set by the caller, it returns error and necessary number of elements are set to *pnum.

If odd elements are set as dash pattern, odd number of elements are returned. If even elements are set, even number of elements are returned.

If NULL is set to pdash, it returns necessary number of elements for pdash[] in *pnum.

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdSetLineDashOffset

Name

fsgpdSetLineDashOffset – Set the line dash pattern offset.

Synopsis

```
FSGPD_RESULT fsgpdSetLineDashOffset(
      FSGPD_DC printerContext,
      FSGPD_FIX offset);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

offset – Offset value for applying the line dash pattern.

Description

fsgpdSetLineDashOffset() sets offset for dash pattern of stroke operations in units of device coordinate system.

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdGetLineDashOffset

Name

fsgpdGetLineDashOffset – Get the line dash pattern offset.

Synopsis

```
FSGPD_RESULT fsgpdGetLineDashOffset(
      FSGPD_DC printerContext,
      FSGPD_FIX *poffset);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

poffset – Pointer to the offset value for applying the line dash pattern.

Description

fsgpdGetLineDashOffset() retrieves offset value of dash pattern from graphics state object.

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdSetLineStyle

Name

fsgpdSetLineStyle – Set the line style.

Synopsis

```
FSGPD_RESULT fsgpdSetLineStyle(
      FSGPD_DC printerContext,
      FSGPD_LINESTYLE linestyle);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

linestyle – Line style enum. FSPGD_LINESTYLE_SOLID or FSGPD_LINESTYLE_DASH can be designated.

Description

fsgpdSetLineStyle() sets line style to graphics state object.

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdGetLineStyle

Name

fsgpdGetLineStyle – Get the line style.

Synopsis

```
FSGPD_RESULT fsgpdGetLineStyle(
      FSGPD_DC printerContext,
      FSGPD_LINESTYLE *plinestyle);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

plinestyle – Pointer to the line style enum.

Description

fsgpdGetLineStyle() retrieves line style for stroke from graphics state object.

$$Miter\ Limit = \frac{Miter\ Length}{Line\ Width} = \frac{1}{\sin(\frac{\phi}{2})}$$

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdSetLineCap

Name

fsgpdSetLineCap – Set the line cap style.

Synopsis

```
FSGPD_RESULT fsgpdSetLineCap(
      FSGPD_DC printerContext,
      FSGPD_LINECAP linecap);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

linecap – Line cap style enum. Either of FSGPD_LINECAP_BUTT, FSGPD_LINECAP_ROUND, FSGPD_LINECAP_SQUARE can be set.

Description

fsgpdSetLineCap() sets the style for line cap to graphics state object.

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdGetLineCap

Name

fsgpdGetLineCap – Get the line cap style.

Synopsis

```
FSGPD_RESULT fsgpdGetLineCap(
      FSGPD_DC printerContext,
      FSGPD_LINECAP *plinecap);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

plinecap – Pointer to the line cap style enum.

Description
fsgpdGetLineCap() retrieves the style for line cap from graphics state object.

Return Value
If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdSetLineJoin

Name
fsgpdSetLineJoin – Set the line join style.

Synopsis
```
FSGPD_RESULT fsgpdSetLineJoin(
        FSGPD_DC printerContext,
        FSGPD_LINEJOIN linejoin);
```

Arguments
printerContext – printer context value returned by fsgpdOpenPrinter()

linejoin – Line join style enum. Either of FSGPD_LINEJOIN_MITER, FSGPD_LINEJOIN_ROUND and FSGPD_LINEJOIN_BEVEL can be set.

Description
fsgpdSetLineJoin() sets the line joining method to graphics state object.

Return Value
If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdGetLineJoin

Name
fsgpdGetLineJoin – Get the line join style.

Synopsis
```
FSGPD_RESULT fsgpdGetLineJoin(
        FSGPD_DC printerContext,
        FSGPD_LINEJOIN *plinejoin);
```

Arguments
printerContext – printer context value returned by fsgpdOpenPrinter()

plinejoin – Pointer to the line join style enum.

Description
fsgpdGetLineJoin() retrieves the line joining method from graphics state object.

Return Value
If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdSetMiterLimit

Name

fsgpdSetMiterLimit – Set the miter limit value.

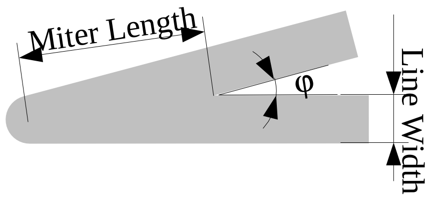
Synopsis

```
FSGPD_RESULT fsgpdSetMiterLimit(
       FSGPD_DC printerContext,
       FSGPD_FIX miterlimit);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

miterlimit – Maximum miter length.



Description

fsgpdSetMiterLimit() sets the maximum length of miter to graphics state object. The miter limit is effective only when line joining style is FSGPD_LINEJOIN_MITER. As to the definition of the miter limit, refer the figure right. The unit is in device coordinate system.

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdGetMiterLimit

Name

fsgpdGetMiterLimit – Get the miter limit value.

Synopsis

```
FSGPD_RESULT fsgpdGetMiterLimit(
       FSGPD_DC printerContext,
       FSGPD_FIX *pmiterlimit);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

pmiterlimit – Pointer to the maximum miter length.

Description

fsgpdGetMiterLimit() retrieves the maximum length of miter from graphics state object.

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdSetPaintMode

Name

fsgpdSetPaintMode – Set the background color painting mode.

Synopsis

```
FSGPD_RESULT fsgpdSetPaintMode(
       FSGPD_DC printerContext,
        FSGPD_PAINTMODE paintmode);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

paintmode – Painting mode enum. Either of FSGPD_PAINTMODE_OPAQUE and FSGPD_PAINTMODE_TRANSPARENT can be set.

Description

fsgpdSetPaintMode() sets paint mode for background to graphics state object.

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdGetPaintMode

Name

fsgpdGetPaintMode – Get the background color painting mode.

Synopsis

```
FSGPD_RESULT fsgpdGetPaintMode(
        FSGPD_DC printerContext,
        FSGPD_PAINTMODE ppaintmode);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

ppaintmode – Pointer to the painting mode enum.

Description

fsgpdGetPaintMode() retrieves paint mode for background from graphics state object.

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdSetStrokeColor

Name

fsgpdSetStrokeColor – Define stroke color and pattern for fsgpdStrokePath() and fsgpdStrokeFillPath() operations.

Synopsis

```
FSGPD_RESULT fsgpdSetStrokeColor(
        FSGPD_DC printerContext,
        FSGPD_BRUSH *brush);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

brush – Pointer to brush definition.

Description

fsgpdSetStrokeColor() registers the color or pattern for stroke drawing of fsgpdStrokePath() and fsgpdStrokeFillPath()

operations as a brush to graphics state object.

If pbrush in Brush structure is set to NULL, it specifies a solid brush. In this case, color of the solid brush is specified in color[4] in the color space by colorSpace.

If the pointer to the BrushData structure is set in pbrush, it specifies pattern brush. In the BrushData structure, width, hight and repetition pitch are specified in pixel value and pointer to the actual pattern data is specified by data. The color space for the pattern is specified by ColorSpace in Brush structure. In this case, color[4] specifies foreground color.

Structures

```
typedef struct _FSGPD_BRUSH {
      FSGPD_CSPACE colorSpace;
      FSGPD_INT color[4]; /* aRGB quadruplet */
      FSGPD_INT xorg, yorg;     /* brush origin */
                                /* ignored for fsgpdSetBgColor */
      FSGPD_BRUSHDATA *pbrush;  /* pointer to brush data */
                                /* solid brush used, if NULL */
} FSGPD_BRUSH;

typedef struct _FSGPD_BRUSHDATA {
      FSGPD_BDTYPE type;
      FSGPG_INT width, height, pitch;
      FSGPD_BYTE data[];        /* FSGPD_BYTE data[1] for GCC 2.x */
} FSGPD_BRUSHDATA;

typedef enum _FSGPD_BDTYPE {FSGPD_BDTYPE_NORMAL = 0} FSGPD_BDTYPE;
```

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdSetFillColor

Name

fsgpdSetFillColor – Define fill color and pattern for fsgpdFillPath() and fsgpdStrokeFillPath() operations.

Synopsis

```
FSGPD_RESULT fsgpdSetFillColor(
      FSGPD_DC printerContext,
      FSGPD_BRUSH *brush);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

brush – Pointer to brush definition.

Description

fsgpdSetFillColor() registers the color or pattern for fsgpdFillPath() and fsgpdStrokeFillPath() operations as a brush to graphics state object.

If successful, this function return FSGPD_OK. If an error is occurred

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

fsgpdSetBgColor

Name

fsgpdSetBgColor – Define background color and pattern for fsgpdStrokePath(), fsgpdFillPath() and fsgpdStrokeFillPath() operations.

Synopsis

```
FSGPD_RESULT fsgpdSetBgColor(
        FSGPD_DC printerContext,
        FSGPD_BRUSH *brush);
```

Arguments

printerContext – printer context value returned by fsgpdOpenPrinter()

brush – Pointer to brush definition.

Description

fsgpdSetBgColor() registers the background or pattern for fsgpdStrokePath(), fsgpdFillPath() and fsgpdStrokeFillPath() operations as brush.

The Members colorSpace and color[4] of the FSGPD_BRUSH structure are referred as brush data.

Xorg and yorg are ignored. The caller should set NULL to pbrush, otherwise it returned as an error (FSGPD_BADREQUEST).

Return Value

If successful, this function return FSGPD_OK. If an error is occurred, it returns -1 and stores error code in fsgpdErrorNo.

Path Operations

Path is used for the drawing command, such as “stroke”, “fill” and “stroke and fill” as well as for defining the clipping area. One Graphics State Object has only one path. Path is effected by CTM, and registered into the Graphics State along the CTM when calling each path operation function.

All path operation functions are OPTIONAL.

fsgpdNewPath

Name

fsgpdNewPath – start a new path

Synopsis

```
FSGPD_RESULT fsgpdNewPath(
        FSGPD_DC printerContext);
```

Arguments

printerContext – Printer Context obtained by the `fsgpdOpenPrinter()` function.

Description

Delete the current path and start a new path.

Return Value

FSGPD_OK when no error, or –1 when error and an error code is set into “fsgpdErrorNo”.

fsgpdEndPath

Name

fsgpdEndPath – end current path

Synopsis

```
FSGPD_RESULT fsgpdEndPath(
        FSGPD_DC printerContext);
```

Arguments

printerContext – Printer Context obtained by the `fsgpdOpenPrinter()` function.

Description

Declare the end of the current path, and the Graphics State Object retains the current path.

Return Value

FSGPD_OK when no error, or –1 when error and an error code is set into “fsgpdErrorNo”.

fsgpdStrokePath

Name

fsgpdStrokePath – stroke current path

Synopsis

```
FSGPD_RESULT fsgpdStrokePath(
        FSGPD_DC printerContext);
```

Arguments

printerContext – Printer Context obtained by the `fsgpdOpenPrinter()` function.

Description

Draw lines along the current path. The lines are drawn along the drawing attributes registered in the Graphics State Object when calling this function. The Graphics State Object retains the current path.

Return Value

FSGPD_OK when no error, or –1 when error and an error code is set into “fsgpdErrorNo”.

fsgpdFillPath

Name

fsgpdFillPath – fill current path

Synopsis

```
FSGPD_RESULT fsgpdFillPath(
        FSGPD_DC printerContext);
```

Arguments

printerContext – Printer Context obtained by the `fsgpdOpenPrinter()` function.

Description

Fill in the current path with the filling mode registered in the Graphics State Object. Filling is done along the “right and bottom exclusive” method to avoid a problem caused by filling the adjoining area with ROP. The way of how to draw each pixel depends on the driver implementation.

When the path is not closed, the starting point and end point of the current sub-path are connected by a line (but not stroked) and closed.

The current path MUST be retained in the Graphics State Object after calling this function.

Return Value

FSGPD_OK when no error, or –1 when error and an error code is set into “fsgpdErrorNo”.

fsgpdStrokeFillPath

Name

fsgpdStrokeFillPath – stroke and fill current path

Synopsis

```
FSGPD_RESULT fsgpdStrokeFillPath(
        FSGPD_DC printerContext);
```

Arguments

printerContext – Printer Context obtained by the `fsgpdOpenPrinter()` function.

Description

Draw lines along the current path, and fill in the current path.

When the path is not closed, the starting point and end point of the current sub-path are connected by a line (but not stroked) and closed.

The current path MUST be retained in the Graphics State Object after calling this function.

Return Value

FSGPD_OK when no error, or –1 when error and an error code is set into “fsgpdErrorNo”.

fsgpdSetClipPath

Name

fsgpdSetClipPath – Set current path as clipping region

Synopsis

```
FSGPD_RESULT fsgpdSetClipPath(
        FSGPD_DC printerContext,
        FSGPD_CLIPRRULE clipRule);
```

Arguments

printerContext – Printer Context obtained by the `fsgpdOpenPrinter()` function.

clipRule – Clipping rule enum. FSGPD_CLIPRULE_EVENODD or FSGPD_CLIPRULE_WINDING can be set.

Description

Set the current path as the clipping region.

When the path is not closed, the starting point and end point of the current sub-path are connected by a line (but not stroked) and closed.

The current path MUST be retained in the Graphics State Object after calling this function.

Clipping region is reset for covering the printable area of the page by calling the `fsgpdStartPage()` function. The clipping region set by this function is valid within the current page.

Return Value

FSGPD_OK when no error, or –1 when error and an error code is set into “fsgpdErrorNo”.

fsgpdResetClipPath

Name

fsgpdResetClipPath – Reset clipping region

Synopsis

```
FSGPD_RESULT fsgpdResetClipPath(
        FSGPD_DC printerContext);
```

Arguments

printerContext – Printer Context obtained by the `fsgpdOpenPrinter()` function.

Description

Reset the current clipping region for covering the printable area of the page.

Return Value

FSGPD_OK when no error, or –1 when error and an error code is set into “fsgpdErrorNo”.

fsgpdSetCurrentPoint

Name

fsgpdSetCurrentPoint – Set current point

Synopsis

```
FSGPD_RESULT fsgpdSetCurrentPoint(
      FSGPD_DC printerContext,
      FSGPD_FIX x,
      FSGPD_FIX y);
```

Arguments

printerContext – Printer Context obtained by the `fsgpdOpenPrinter()` function.

x – x coordinate value for setting the current point.

y – y coordinate value for setting the current point.

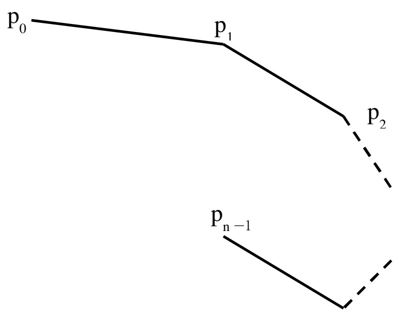
Description

Set the current point to (x, y).

Return Value

FSGPD_OK when no error, or –1 when error and an error code is set into “fsgpdErrorNo”.

fsgpdLinePath



Name

fsgpdLinePath – add multiple connected lines to the current path

Synopsis

```
FSGPD_RESULT fsgpdLinePath(
      FSGPD_DC printerContext,
      FSGPD_PATHMODE flag;
      FSGPD_INT npoints,
      FSGPD_POINT *points);
```

Arguments

printerContext – Printer Context obtained by the `fsgpdOpenPrinter()` function.

flag – PathClose: Close the sub path. In this case, the line connecting the first point and the last point in the array **points* is appended. PathOpen: Do not close the path.

npoints – Number of points in array.

points – Pointer to an array of points.

Description

Append lines specified by the array of points **points* from the current point.

In case of the *flag* is PathOpen, the current point is set to the last point in the array of points **points*. In case of the *flag* is PathClose, the current point is set to the first point in the array of points **points*.

Structures

```
typedef struct _FSGPD_POINT {
      FSGPD_FIX x, y;
} FSGPD_POINT;
```

Return Value

FSGPD_OK when no error, or –1 when error and an error code is set into “fsgpdErrorNo”.

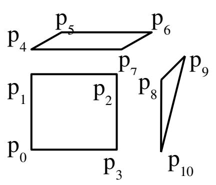
fsgpdPolygonPath

Name

fsgpdPolygonPath – add polygons to current path

Synopsis

```
FSGPD_RESULT fsgpdPolygonPath(
      FSGPD_DC printerContext,
      FSGPD_INT npolygons,
      FSGPD_INT *nvertexes,
      FSGPD_POINT *points);
```



In the above case, each argument is as following:
npolygons=3
*nvertexes={4, 4, 3}
*points={p0, p1, … p10}

Arguments

printerContext – Printer Context obtained by the `fsgpdOpenPrinter()` function.

npolygons – Number of polygon appended to the path.

nvertexes – Pointer to an array of the number of points of each polygon.

points – Pointer to an array of points. The number of points in an array must be *nvertexes*.

Description

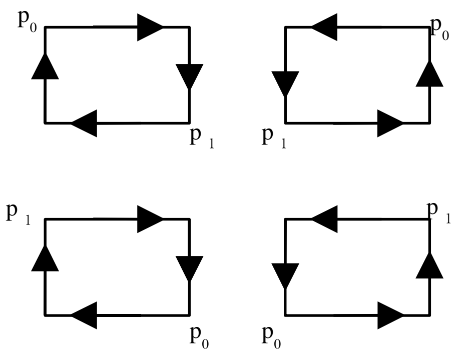
Append polygons specified by the array of points **points* into the current path from the current point.

The current point is set to the last point in the array of points **points*.

Return Value

FSGPD_OK when no error, or –1 when error and an error code is set into “fsgpdErrorNo”.

fsgpdRectanglePath



Name

fsgpdRectanglePath – Add rectangles to current path.

Synopsis

```
FSGPD_RESULT fsgpdRectanglePath(
      FSGPD_DC printerContext,
      FSGPD_INT nrectangles,
      FSGPD_RECTANGLE *rectangles);
```

Arguments

printerContext – Printer Context obtained by the `fsgpdOpenPrinter()` function.

nrectangles – Number of rectangles.

rectangles – Pointer to an array of rectangles.

Description

Append rectangles into the current path.

The current point is set to the starting point of the last rectangle appended.

Direction of the path of each rectangle depends on how to set the starting point and diagonal point as above figure.

Path is appended in order of (x0, y0)-(x1, y0)-(x1, y1)-(x0, y1)-(x0, y0) where the starting point “p0” is (x0,y0) and the diagonal point “p1” is (x1,y1).

Structures

```
typedef struct _FSGPD_RECTANGLE {
      FSGPD_POINT p0;     /* Starting point. */
      FSGPD_POINT p1;     /* Diagonal point. */
} FSGPD_RECTANGLE;
```

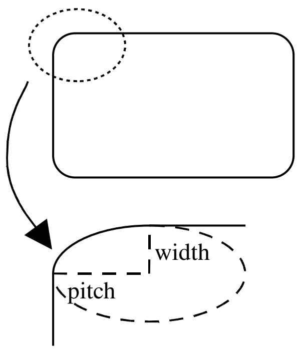
Return Value

FSGPD_OK when no error, or –1 when error and an error code is set into “fsgpdErrorNo”.

fsgpdRoundRectanglePath

Name

fsgpdRoundRectanglePath – Add round rectangles to current path.



Synopsis

```
FSGPD_RESULT fsgpdRoundRectanglePath(
      FSGPD_DC printerContext,
      FSGPD_INT nrectangles,
      FSGPD_ROUNDRECTANGLE *rectangles);
```

Arguments

printerContext – Printer Context obtained by the `fsgpdOpenPrinter()` function.

nrectangles – Number of round rectangles.

rectangles – Pointer to an array of RoundRectangle structure.

Description

Append round rectangles into the path.

Each corner of round rectangle is connected by elliptic arc defined by *xellippse* and *yellipse* in the RoundRectangle structure. Lines of a round rectangle are drawn as a rectangle path.

Structures

```
typedef struct _FSGPD_ROUNDRECTANGLE {
      FSGPD_POINT p0;     /* Starting point. */
      FSGPD_POINT p1;     /* Diagonal point. */
      FSGPD_FIX xellipse, yellipse;
} FSGPD_ROUNDRECTANGLE;
```

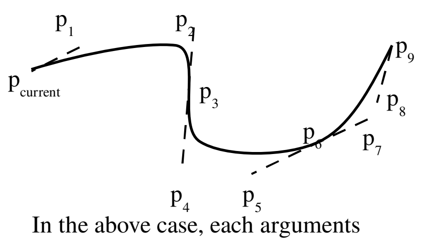
Return Value

FSGPD_OK when no error, or –1 when error and an error code is set into “fsgpdErrorNo”.

fsgpdBezierPath

Name

fsgpdBezierPath – Append a 3D-bezier path into the current path.



In the above case, each arguments are as following:
npoints = 9
*points = {p_1, p_2, … p_9}

Synopsis

```
FSGPD_RESULT fsgpdBezierPath(
        FSGPD_DC printerContext,
        FSGPD_INT npoints,
        FSGPD_POINT *points);
```

Arguments

printerContext – Printer Context obtained by the `fsgpdOpenPrinter()` function.

npoints –Number of points in an array **points*. It must be a multiple number of “3”.

points – Pointer to an array of end points and control points of bezier curves.

Description

Append multiple bezier path with the current point, end points and the control points specified by **points*. The first bezier path is specified by the current point and third point in an array of *points as the end points and the first point and second point as the control points. Other bezier paths are specified by the previous path’s end point as the first point of bezier path, and the following points as the control points and end points as the previous bezier path.

The current point is set to the last point of the bezier path.

If *npoints* is not a multiple number of “3”, this function returns error, and the error code is set as FSGPD_PARAMERROR.

Return Value

FSGPD_OK when no error, or –1 when error and an error code is set into “fsgpdErrorNo”.

fsgpdArcPath

Name

fsgpdArcPath – Add arcs, chords, pies, or ellipses to current path.

Synopsis

```
FSGPD_RESULT fsgpdArcPath(
        FSGPD_DC printerContext,
        FSGPD_ARCMODE kind,
        FSGPD_ARCDIR dir,
        FSGPD_FIX bbx0, FSGPD_FIX bby0, FSGPD_FIX bbx1, FSGPD_FIX bby1,
        FSGPD_FIX x0, FSGPD_FIX y0,
        FSGPD_FIX x1, FSGPD_FIX y1);
```

Arguments

printerContext – Printer Context obtained by the `fsgpdOpenPrinter()` function.

kind – Arc: an arc、Chord: a chord、Pie: a pie.

dir – The direction of the path。Clockwise: clockwise、Counterclockwise: counter-clockwise.

(bbx0, bby0)-(bbx1, bby1) – Circumscribe rectangle.

(x0, y0) – Starting point.

(x1, y1) – End point.

Description

Append an arc, chord or a pie to the current path. The center point of ellipse is the middle point of the circumscribe rectangle. The direction of the path is specified by *dir*. When “Arc” is set into *kind* and the same point is set into both *start* and *end*, an ellipse is appended into the path. If the circumscribe rectangle is a square, a circle is appended into the path.

In case of an arc, the current point is set as the end point of an arc. In case of a chord or pie, the current point is set as the left-top point of the circumscribe rectangle.

Return Value

FSGPD_OK when no error, or –1 when error and an error code is set into “fsgpdErrorNo”.

Text Operations

Text operations are used for obtain the information of device fonts,draw the device fonts,register the information of device fonts and erase the information of device fonts.

All text operation functions are OPTIONAL.

Detailed operation is not determined yet.

Bitmap Image Operations

Bitmap is the bit oriented image data which is drawn in rectangle region. When drawing the bitmap,the attribute of drawing set as Graphics State Object is applied.

The typical draw of bitmap is

- fsgpdStartDrawImage() – specified the image data
- fsgpdTransferDrawImage() – transfer the image data(1 time or more)
- fsgpdEndDrawImage() – end of transfer image data and draw it.

If called except fsgpdTransferDrawImage() function between fsgpdStartDrawImage() and fsgpdEndDrawImage() function ,it returns error and the error code is set as FSGPD_BADREQEST.

fsgpdDrawImage() is a function for performing these operations by functions call once.

All bitmap operation functions are OPTIONAL.

fsgpdDrawImage

Name

fsgpdDrawImage – Draw bitmap image

Synopsis

```
FSGPD_RESULT fsgpdDrawImage(
      FSGPD_DC printerContext,
      FSGPD_INT sourceWidth
      FSGPD_INT sourceHeight;
      FSGPD_INT sourcePitch;
      FSGPD_INT colorDepth;
      FSGPD_IMAGEFORMAT imageFormat,
      FSGPD_RECTANGLE destinationSize,
      void *imageData);
```

Arguments

printerContext – Printer Context obtained by the `fsgpdOpenPrinter()` function.

sourceWidth – Width of source image(pixels)

sourceHeight – Height of source image(pixels)

sourcePitch – Repetition pitch of source image (bytes)

colorDepth – The number of bits per pixel of source image.

imageFormat – Image format enum.

destinationSize – Size of destination image.

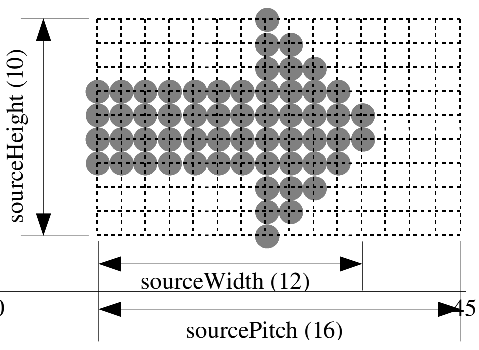
imageDate – The pointer of bitmap image data.

Description

Draw the rectangle bitmap and the current point is left-upper.

If current point is not integer,the reference point is revalued to the integer and the current point is not updated.

SourceWidth and sourceHeight are width and height of the bitmap image respectively. SourcePitch is the bytes of one raster data in *imageData (see the figure right for an example). This argument is effective only when the



imageFoirmat is Raw or RLE.

Image format can be used which the device obtained from device capability is supporting. （Raw, RLE, JPEG, PNG or the vendor supports uniquely can be specified.）

Return Value

FSGPD_OK when no error,or -1 when error and an error code is set into “fsgpdErrorNo”.

fsgpdStartDrawImage

Name

fsgpdStartDrawImage – start draw bitmap image

Synopsis

```
FSGPD_RESULT fsgpdStartDrawImage(
       FSGPD_DC printerContext,
       FSGPD_INT sourceWidth
       FSGPD_INT sourceHeight;
       FSGPD_INT sourcePitch;
       FSGPD_INT colorDepth;
       FSGPD_IMAGEFORMAT imageFormat,
       FSGPD_RECTANGLE destinationSize);
```

Arguments

printerContext – Printer Context obtained by the `fsgpdOpenPrinter()` function.

sourceWidth – Width of source image(pixels)

sourceHeight – Height of source image（pixels）

sourcePitch – Repetition pitch of source image (bytes)

colorDepth – The number of bits per pixel of source image.

imageFormat– Image format enum.

destinationSize – Size of destination image.

Description

Draw the rectangle bitmap and the current point is left-upper.

If current point is not integer,the reference point is revalued to the integer and the current point is not updated.

Image format can be used which the device obtained from device capability is supporting.（Raw, RLE, JPEG, PNG or the vendor supports uniquely can be specified.）

Return Value

FSGPD_OK when no error,or -1 when error and an error code is set into “fsgpdErrorNo”.

fsgpdTransferDrawImage

Name

fsgpdTransferDrawImage – transfer bitmap image data

Synopsis

```
FSGPD_RESULT fsgpdTransferDrawImage(
       FSGPD_DC printerContext,
       FSGPD_INT count,
       void *imageData);
```

Arguments

printerContext – Printer Context obtained by the `fsgpdOpenPrinter()` function.

count – The number of bytes of imageData.

imageDate – The pointer of bitmap image data.

Description

Transfer bitmap image data which is started by fsgpdStartDrawImage() function.

Return Value

FSGPD_OK when no error,or -1 when error and an error code is set into “fsgpdErrorNo”.

fsgpdEndDrawImage

Name

fsgpdEndDrawImage – end draw bitmap image

Synopsis

```
FSGPD_RESULT fsgpdEndDrawImage(
        FSGPD_DC printerContext);
```

Arguments

printerContext – Printer Context obtained by the `fsgpdOpenPrinter()` function.

Description

Declare the end of drawing image data which is started by fsgpdStartDrawImage() or fsgpdDrawImage() functions.

Return Value

FSGPD_OK when no error,or -1 when error and an error code is set into “fsgpdErrorNo”.

Scan Line Operations

Scan lines provide support for horizontal line drawing and are defined by a begin point and an end point. Scan lines are drawn according to the draw mode settings of the Graphics State Object.

All scan line operations are OPTIONAL. However support for scan line operations is recommended in the case where path operations are not supported and you want to switch the color scheme from bitmap oriented to path oriented.

fsgpdStartScanline

Name

`fsgpdStartScanline` – initiates scan line drawing

Synopsis

```
FSGPD_RESULT fsgpdStartScanline(
       FSGPD_DC printerContext,
       FSGPD_INT yposition);
```

Arguments

`printerContext` – a printer context returned by a call to `fsgpdOpenPrinter()`

`yposition` – the vertical position to start a scan line

Description

Initiates scan line drawing. Scan lines are drawn by fsgpdStartScanline() to declare the begging scan lines, fsgpdScanline()'s to set actual scan line data, then fsgpdEndScanline() to end the operations. Current point is not modified by fsgpdStartScanline().

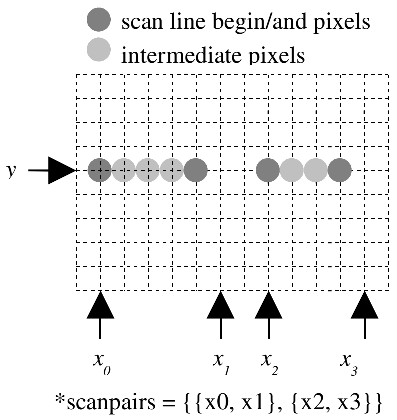
Return Value

Upon successful completion returns FSGPD_OK. Otherwise, -1 is returned and the variable `fsgpdErrorNo` is set to indicate the error.

fsgpdScanline

Name

`fsgpdScanline` – draws scan lines



● scan line begin/and pixels
● intermediate pixels
y
x_0
x_1
x_2
x_3
*scanpairs = {{x0, x1}, {x2, x3}}

Synopsis

```
FSGPD_RESULT fsgpdScanline(
       FSGPD_DC printerContext,
       FSGPD_INT nscanpairs,
       FSGPD_INT *scanpairs);
```

Arguments

`printerContext` – a printer context returned by a call to `fsgpdOpenPrinter()`

`nscanpairs` – number of scan line begin and end point pairs

`scanpairs` – pointer to an array of scan line begin and end point pairs

Description

Draws the scan lines passed. After the draw operation has completed, the current vertical coordinate is incremented by 1.

Current point is not modified.

Return Value

Upon successful completion returns FSGPD_OK. Otherwise, -1 is returned and the variable `fsgpdErrorNo` is set to indicate the error.

fsgpdEndScanline

Name

`fsgpdEndScanline` – terminates scan line drawing

Synopsis

```
FSGPD_RESULT fsgpdEndScanline(
       FSGPD_DC printerContext);
```

Arguments

`printerContext` – a printer context returned by a call to `fsgpdOpenPrinter()`

Description

Terminates scan line drawing.

Return Value

Upon successful completion returns FSGPD_OK. Otherwise, -1 is returned and the variable `fsgpdErrorNo` is set to indicate the error.

Raster Image Operations

The raster image operations provide support for raster image drawing. This functionality is required for devices that support neither bitmap nor path operations.

fsgpdStartRaster

Name

`fsgpdStartRaster` – initiates raster image drawing

Synopsis

```
FSGPD_RESULT fsgpdStartRaster(
        FSGPD_DC printerContext,
        FSGPD_INT rasterWidth);
```

Arguments

`printerContext` – a printer context returned by a call to `fsgpdOpenPrinter()`

`rasterWidth` – the width of the raster image in pixels (padding is not included)

Description

Initiates raster data drawing.

The raster image will be drawn from current point.

The color space as determined by the current Graphics State Object will be used. Possible values include: FSGPD_CSPACE_BW, FSGPD_CSPACE_DEVICEGRAY, FSGPD_CSPACE_DEVICECMY, FSGPD_CSPACE_DEVICECMYK, FSGPD_CSPACE_STANDARDRGB, FSGPD_CSPACE_STANDARDRGB64.

The bit and/or byte order is also determined by the value set in the current Graphics State Object.

Raster data compression is not supported. The driver is expected to perform suitable compression.

The raster data format depends on the color space as tabulated below.

Color Space	# of planes	Bits per Plane	Bits per Pixel	Bytes per Pixel	Note
FSGPD_CSPACE_BW	1	1	1	1/8	bit order required padding bits are added to the rightmost byte if necessary
FSGPD_CSPACE_DEVICEGRAY	1	8	8	1	
FSGPD_CSPACE_STANDARDRGB	3	8	24	3	
FSGPD_CSPACE_STANDARDRGB64	3	16	48	6	byte order required
FSGPD_CSPACE_DEVICECMY	3	8	24	3	
FSGPD_CSPACE_DEVICECMYK	4	8	32	4	

Return Value

Upon successful completion returns FSGPD_OK. Otherwise, -1 is returned and the variable `fsgpdErrorNo` is set to indicate the error.

fsgpdTransferRasterData

Name

`fsgpdTransferRasterData` – transfers raster image data

Synopsis

```
FSGPD_RESULT fsgpdTransferRasterData(
       FSGPD_DC printerContext,
       FSGPD_INT count,
       FSGPD_BYTE *data);
```

Arguments

`printerContext` – a printer context returned by a call to `fsgpdOpenPrinter()`

`count` – number of pixel data elements

`data` – pointer to an array holding the pixel data

Description

Transfers `count` pixel data elements of raster `data`. The `data` constitutes a single row of raster data. Data exceeding the width of the raster image is silently ignored. Should less data elements be transferred than required for a single row, then the part that was not transferred shall be treated as if non-existent. The y-coordinate of current point is incremented by 1.

Return Value

Upon successful completion returns FSGPD_OK. Otherwise, -1 is returned and the variable `fsgpdErrorNo` is set to indicate the error.

fsgpdSkipRaster

Name

`fsgpdSkipRaster` – skips lines during raster image drawing

Synopsis

```
FSGPD_RESULT fsgpdSkipRaster(
       FSGPD_DC printerContext,
       FSGPD_INT count);
```

Arguments

`printerContext` – a printer context returned by a call to `fsgpdOpenPrinter()`

`count` – number of lines to be skipped

Description

Skips `count` lines in the vertical direction during raster image drawing, producing a horizontal empty space. The y-coordinate of current point is incremented by `count`.

Return Value

Upon successful completion returns FSGPD_OK. Otherwise, -1 is returned and the variable `fsgpdErrorNo` is set to indicate the error.

fsgpdEndRaster

Name

`fsgpdEndRaster` – terminates raster image drawing

Synopsis

```
FSGPD_RESULT fsgpdEndRaster(
        FSGPD_DC printerContext);
```

Arguments

`printerContext` – a printer context returned by a call to `fsgpdOpenPrinter()`

Description

Terminates raster image drawing.

Return Value

Upon successful completion returns FSGPD_OK. Otherwise, -1 is returned and the variable `fsgpdErrorNo` is set to indicate the error.

Stream Data Operations

The “stream data operations” are used when an application directly creates a PDL or when it sends EPS data to a PostScript device. Be careful when using these together with other drawing instructions because the result of that printing is not guaranteed. All stream data operations are OPTIONAL.

fsgpdStartStream

Name

fsgpdStartStream – start of streaming data transfer

Synopsis

```
FSGPD_RESULT fsgpdStartStream(
       FSGPD_DC printerContext);
```

Arguments

printerContext – Specifies a printer context obtained using fsgpdOpenPrinter()

Description

fsgpdStartStream designates the start of streaming data transfer. Streaming data is the data that directly goes to the printer device without being edited or checked by the driver.

Return Value

The return value is FSGPD_OK in case of normal finish. In case of error, the return value is –1 and error code is stored in “fsgpdErrorNo”.

fsgpdTransferStreamData

Name

fsgpdTransferStreamData – send stream data

Synopsis

```
FSGPD_RESULT fsgpdTransferStreamData(
       FSGPD_DC printerContext,
       FSGPD_INT count,
       void *data);
```

Arguments

printerContext – Specifies a printer context obtained using fsgpdOpenPrinter()

count – number of data bytes transferred

data – pointer to stream data

Description

Send streaming data between fsgpdStartStream and fsgpdEndStream.

Return Value

The return value is FSGPD_OK in case of normal finish. In case of error, the return value is –1 and error code is stored in “fsgpdErrorNo”.

fsgpdEndStream

Name
fsgpdEndStream – end of streaming data transfer

Synopsis
```
FSGPD_RESULT fsgpdEndStream(
        FSGPD_DC printerContext);
```

Arguments
printerContext – Specifies a printer context obtained using fsgpdOpenPrinter()

Description
End of streaming data transfer.

Return Value
The return value is FSGPD_OK in case of normal finish. In case of error, the return value is –1 and error code is stored in "fsgpdErrorNo".

V.Graphic Operation Fallback

Normally, the printer driver declares the supported API through the fsgpdOpenPrinter() function depending upon the available functionality of the printing device. However, in some conditions, support to an API may not be possible. A graphic operation which can not be supported by the driver or the printer device, may be implemented by changing it to a “low level operation”. Implementing using “low level operation” is called “fallback”. Though it is possible to provide high level functionality which is not supported by a printing device by implementing it in the printer driver (bringing down the level in the driver to the available functionality of the printing device and sending appropriate instructions to the printer), but a driver developer would want to avoid implementing it except in case of providing some additional functionality, for example, improvement of print quality by doing high degree of rendering in the driver.

This chapter explains how to handle the functionality not made available by the printer and the driver.

Renderer Fallback

It is a fallback due to renderer. When the pointer to driver API is NULL, or when the return code of the API call indicates “no support”, the renderer, depending on the API, calls the driver after further breaking down the processing steps in accordance with the available functionality of the low level driver.

Driver Fallback

It is a fallback due to the printer driver. Consider a situation in which a driver behaves like it supports a particular functionality towards the renderer, however the printer device does not support the requested graphics function. In this situation, the driver breaks down the request into drawing instructions that can be implemented by the printing device. The breakdown methods are; the processing by the driver itself and using external drawing library. In the external drawing library, depending on the Scan Conversion Extension (explained in the next section), vector drawing instruction can be broken down into scan line instructions. Another example of external library that can be used is libart.

VI.Macros, Types, Enums and Structures

Following program is a header file which contains macros, types, enums and structures defined in this specification.

```
/*
 * OpenPrinting Vector Printer Driver API Definitions [fsgpd.h]
 *
 * Copyright (c) 2006 Free Standards Group
 * Copyright (c) 2006 Fuji Xerox Printing Systems Co., Ltd.
 * Copyright (c) 2006 Canon Inc.
 * Copyright (c) 2003-2006 AXE Inc.
 *
 * All Rights Reserved.
 *
 * Permission to use, copy, modify, distribute, and sell this software
 * and its documentation for any purpose is hereby granted without
 * fee, provided that the above copyright notice appear in all copies
 * and that both that copyright notice and this permission notice
 * appear in supporting documentation.
 *
 * The above copyright notice and this permission notice shall be
 * included in all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
 * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
 * NONINFRINGEMENT.  IN NO EVENT SHALL THE OPEN GROUP BE LIABLE FOR
 * ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF
 * CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
 * WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */

#ifndef _FSGPD_H_
#define _FSGPD_H_

/* Return Values and Error Codes */
#define FSGPD_OK           0      /* -1 for errors */
#define FSGPD_FATALERROR   -1     /* error: cannot be recovered */
#define FSGPD_BADREQUEST   -2     /* error: called where it should not be called */
#define FSGPD_BADCONTEXT   -3     /* error: invalid printer context */
#define FSGPD_NOTSUPPORTED        -4     /* error: combination of parameters are set */
                                  /*   which cannot be handled by driver or printer */
#define FSGPD_JOBCANCELED -5      /* error: job has been canceled by some cause */
#define FSGPD_PARAMERROR   -6     /* error: invalid parameter */

/* Basic Types */
typedef int FSGPD_DC;                    /* driver/device context */
typedef int FSGPD_RESULT;         /* return value */
typedef unsigned char FSGPD_BYTE;        /* BYTE */
typedef unsigned char FSGPD_CHAR;        /* character (string) */
typedef int FSGPD_INT;                   /* integer */
typedef int FSGPD_FIX;                   /* fixed integer */
typedef float FSGPD_FLOAT;               /* float */
typedef unsigned int FSGPD_FLAG; /* flags */
typedef unsigned int FSGPD_ROP;          /* raster operation */

/* graphic elements */
typedef struct _FSGPD_POINT {
      FSGPD_FIX x, y;
} FSGPD_POINT;

typedef struct _FSGPD_RECTANGLE {
        FSGPD_POINT p0;                     /* start point */
        FSGPD_POINT p1;                     /* diagonal point */
} FSGPD_RECTANGLE;

typedef struct _FSGPD_ROUNDRECTANGLE {
        FSGPD_POINT p0;                     /* start point */
        FSGPD_POINT p1;                     /* diagonal point */
        FSGPD_FIX xellipse, yellipse;
} FSGPD_ROUNDRECTANGLE;

/* Image Formats */
typedef enum _FSGPD_IMAGEFORMAT {
        FSGPD_IFORMAT_RAW          = 0,
        FSGPD_IFORMAT_RLE          = 1,
        FSGPD_IFORMAT_JPEG         = 2,
        FSGPD_IFORMAT_PNG          = 3
} FSGPD_IMAGEFORMAT;

/* Color Presentation */
typedef enum _FSGPD_COLORMAPPING {
        FSGPD_CMAP_DIRECT          = 0,
        FSGPD_CMAP_INDEXED         = 1
} FSGPD_COLORMAPPING;

typedef enum _FSGPD_CSPACE {
        FSGPD_CSPACE_BW                    = 0,
        FSGPD_CSPACE_DEVICEGRAY            = 1,
        FSGPD_CSPACE_DEVICECMY             = 2,
        FSGPD_CSPACE_DEVICECMYK            = 3,
        FSGPD_CSPACE_DEVICERGB             = 4,
        FSGPD_CSPACE_STANDARDRGB   = 5,
        FSGPD_CSPACE_STANDARDRGB64         = 6
} FSGPD_CSPACE;

/* Fill, Paint, Clip */
typedef enum _FSGPD_FILLMODE {
        FSGPD_FILLMODE_EVENODD             = 0,
        FSGPD_FILLMODE_WINDING             = 1
} FSGPD_FILLMODE;

typedef enum _FSGPD_PAINTMODE {
        FSGPD_PAINTMODE_OPAQUE             = 0,
        FSGPD_PAINTMODE_TRANSPARENT        = 1
} FSGPD_PAINTMODE;

typedef enum _FSGPD_CLIPRULE {
        FSGPD_CLIPRULE_EVENODD             = 0,
        FSGPD_CLIPRULE_WINDING             = 1
} FSGPD_CLIPRULE;

/* Line */
typedef enum _FSGPD_LINESTYLE {
        FSGPD_LINESTYLE_SOLID              = 0,
        FSGPD_LINESTYLE_DASH               = 1
} FSGPD_LINESTYLE;

typedef enum _FSGPD_LINECAP {
        FSGPD_LINECAP_BUTT         = 0,
        FSGPD_LINECAP_ROUND        = 1,
        FSGPD_LINECAP_SQUARE               = 2
} FSGPD_LINECAP;
```

```
typedef enum _FSGPD_LINEJOIN {
      FSGPD_LINEJOIN_MITER             = 0,
      FSGPD_LINEJOIN_ROUND             = 1,
      FSGPD_LINEJOIN_BEVEL             = 2
} FSGPD_LINEJOIN;

/* Brush */
typedef enum _FSGPD_BDTYPE {
      FSGPD_BDTYPE_NORMAL       = 0
} FSGPD_BDTYPE;

typedef struct _FSGPD_BRUSHDATA {
      FSGPD_BDTYPE type;
      FSGPD_INT width, height, pitch;
#if defined(__GNUC__) && __GNUC__ <= 2
      FSGPD_BYTE data[1];
#else
      FSGPD_BYTE data[];
#endif

} FSGPD_BRUSHDATA;

typedef struct _FSGPD_BRUSH {
      FSGPD_CSPACE colorSpace;
      FSGPD_INT color[4];       /* aRGB quadruplet */
      FSGPD_INT xorg, yorg;            /* brush origin */
                                /* ignored for fsgpdSetBgColor */
      FSGPD_BRUSHDATA *pbrush;  /* pointer to brush data */
                                /* solid brush used, if NULL */
} FSGPD_BRUSH;

/* Misc. Flags */
typedef enum _FSGPD_ARCMODE {
      FSGPD_ARC                 = 0,
      FSGPD_CHORD               = 1,
      FSGPD_PIE                 = 2
} FSGPD_ARCMODE;

typedef enum _FSGPD_ARCDIR {
      FSGPD_CLOCKWISE                  = 0,
      FSGPD_COUNTERCLOCKWISE           = 1
} FSGPD_ARCDIR;

typedef enum _FSGPD_PATHMODE {
      FSGPD_PATHCLOSE                  = 0,
      FSGPD_PATHOPEN                   = 1
} FSGPD_PATHMODE;

/* CTM */
typedef struct _FSGPD_CTM {
      FSGPD_FLOAT a, b, c, d, e, f;
} FSGPD_CTM;

/* Device Information and Capabilites */
typedef enum _FSGPD_QUERYINFOFLAGS {
  FSGPD_QF_DEVICERESOLUTION     = 0x00000001,
  FSGPD_QF_MEDIASIZE            = 0x00000002,
  FSGPD_QF_PAGEROTATION         = 0x00000004,
  FSGPD_QF_MEDIANUP       = 0x00000008,
  FSGPD_QF_MEDIADUPLEX          = 0x00000010,
  FSGPD_QF_MEDIASOURCE          = 0x00000020,
  FSGPD_QF_MEDIADESTINATION     = 0x00000040,
  FSGPD_QF_MEDIATYPE             = 0x00000080,
  FSGPD_QF_MEDIASIZE             = 0x00010000,       /* only for fsgpdQueryDeviceInfo
*/
  FSGPD_QF_PRINTREGION           = 0x00020000 /* only for fsgpdQueryDeviceInfo */
} FSGPD_QUERYINFOFLAGS;


/* Function prototype */
FSGPD_DC fsgpdOpenPrinter(
      FSGPD_INT outputFD,
      FSGPD_CHAR *printerModel,
      FSGPD_INT apiVersion[2],
      FSGPD_INT *nApiEntry,
      FSGPD_INT (**apiEntry[])());

/* API Procedure Entries */
typedef      struct _FSGPD_API_PROCS {
      FSGPD_DC
(*fsgpdOpenPrinter)(FSGPD_INT,FSGPD_CHAR*,FSGPD_INT*,FSGPD_INT*,struct
_FSGPD_API_PROCS**);
      FSGPD_RESULT (*fsgpdClosePrinter)(FSGPD_DC);
      FSGPD_RESULT (*fsgpdStartJob)(FSGPD_DC,FSGPD_CHAR*);
      FSGPD_RESULT (*fsgpdEndJob)(FSGPD_DC);
      FSGPD_RESULT (*fsgpdAbortJob)(FSGPD_DC);
      FSGPD_RESULT (*fsgpdStartDoc)(FSGPD_DC,FSGPD_CHAR*);
      FSGPD_RESULT (*fsgpdEndDoc)(FSGPD_DC);
      FSGPD_RESULT (*fsgpdStartPage)(FSGPD_DC,FSGPD_CHAR*);
      FSGPD_RESULT (*fsgpdEndPage)(FSGPD_DC);
      FSGPD_RESULT
(*fsgpdQueryDeviceCapability)(FSGPD_DC,FSGPD_FLAG,FSGPD_INT,FSGPD_BYTE*);
      FSGPD_RESULT
(*fsgpdQueryDeviceInfo)(FSGPD_DC,FSGPD_FLAG,FSGPD_INT,FSGPD_CHAR*);
      FSGPD_RESULT (*fsgpdResetCTM)(FSGPD_DC);
      FSGPD_RESULT (*fsgpdSetCTM)(FSGPD_DC,FSGPD_CTM*);
      FSGPD_RESULT (*fsgpdGetCTM)(FSGPD_DC,FSGPD_CTM*);
      FSGPD_RESULT (*fsgpdInitGS)(FSGPD_DC);
      FSGPD_RESULT (*fsgpdSaveGS)(FSGPD_DC);
      FSGPD_RESULT (*fsgpdRestoreGS)(FSGPD_DC);
      FSGPD_RESULT (*fsgpdQueryColorSpace)(FSGPD_DC,FSGPD_CSPACE*,FSGPD_INT*);
      FSGPD_RESULT (*fsgpdSetColorSpace)(FSGPD_DC,FSGPD_CSPACE);
      FSGPD_RESULT (*fsgpdGetColorSpace)(FSGPD_DC,FSGPD_CSPACE*);
      FSGPD_RESULT (*fsgpdQueryROP)(FSGPD_DC,FSGPD_INT*,FSGPD_ROP*);
      FSGPD_RESULT (*fsgpdSetROP)(FSGPD_DC,FSGPD_ROP);
      FSGPD_RESULT (*fsgpdGetROP)(FSGPD_DC,FSGPD_ROP*);
      FSGPD_RESULT (*fsgpdSetFillMode)(FSGPD_DC,FSGPD_FILLMODE);
      FSGPD_RESULT (*fsgpdGetFillMode)(FSGPD_DC,FSGPD_FILLMODE*);
      FSGPD_RESULT (*fsgpdSetAlphaConstant)(FSGPD_DC,FSGPD_FLOAT);
      FSGPD_RESULT (*fsgpdGetAlphaConstant)(FSGPD_DC,FSGPD_FLOAT*);
      FSGPD_RESULT (*fsgpdSetLineWidth)(FSGPD_DC,FSGPD_FIX);
      FSGPD_RESULT (*fsgpdGetLineWidth)(FSGPD_DC,FSGPD_FIX*);
      FSGPD_RESULT (*fsgpdSetLineDash)(FSGPD_DC,FSGPD_FIX*,FSGPD_INT);
      FSGPD_RESULT (*fsgpdGetLineDash)(FSGPD_DC,FSGPD_FIX*,FSGPD_INT*);
      FSGPD_RESULT (*fsgpdSetLineDashOffset)(FSGPD_DC,FSGPD_FIX);
      FSGPD_RESULT (*fsgpdGetLineDashOffset)(FSGPD_DC,FSGPD_FIX*);
      FSGPD_RESULT (*fsgpdSetLineStyle)(FSGPD_DC,FSGPD_LINESTYLE);
      FSGPD_RESULT (*fsgpdGetLineStyle)(FSGPD_DC,FSGPD_LINESTYLE*);
      FSGPD_RESULT (*fsgpdSetLineCap)(FSGPD_DC,FSGPD_LINECAP);
      FSGPD_RESULT (*fsgpdGetLineCap)(FSGPD_DC,FSGPD_LINECAP*);
      FSGPD_RESULT (*fsgpdSetLineJoin)(FSGPD_DC,FSGPD_LINEJOIN);
      FSGPD_RESULT (*fsgpdGetLineJoin)(FSGPD_DC,FSGPD_LINEJOIN*);
      FSGPD_RESULT (*fsgpdSetMiterLimit)(FSGPD_DC,FSGPD_FIX);
      FSGPD_RESULT (*fsgpdGetMiterLimit)(FSGPD_DC,FSGPD_FIX*);
```

```
       FSGPD_RESULT (*fsgpdSetPaintMode)(FSGPD_DC,FSGPD_PAINTMODE);
       FSGPD_RESULT (*fsgpdGetPaintMode)(FSGPD_DC,FSGPD_PAINTMODE*);
       FSGPD_RESULT (*fsgpdSetStrokeColor)(FSGPD_DC,FSGPD_BRUSH*);
       FSGPD_RESULT (*fsgpdSetFillColor)(FSGPD_DC,FSGPD_BRUSH*);
       FSGPD_RESULT (*fsgpdSetBgColor)(FSGPD_DC,FSGPD_BRUSH*);
       FSGPD_RESULT (*fsgpdNewPath)(FSGPD_DC);
       FSGPD_RESULT (*fsgpdEndPath)(FSGPD_DC);
       FSGPD_RESULT (*fsgpdStrokePath)(FSGPD_DC);
       FSGPD_RESULT (*fsgpdFillPath)(FSGPD_DC);
       FSGPD_RESULT (*fsgpdStrokeFillPath)(FSGPD_DC);
       FSGPD_RESULT (*fsgpdSetClipPath)(FSGPD_DC,FSGPD_CLIPRULE);
       FSGPD_RESULT (*fsgpdResetClipPath)(FSGPD_DC);
       FSGPD_RESULT (*fsgpdSetCurrentPoint)(FSGPD_DC,FSGPD_FIX,FSGPD_FIX);
       FSGPD_RESULT (*fsgpdLinePath)(FSGPD_DC,FSGPD_PATHMODE,FSGPD_INT,FSGPD_POINT*);
       FSGPD_RESULT (*fsgpdPolygonPath)(FSGPD_DC,FSGPD_INT,FSGPD_INT*,FSGPD_POINT*);
       FSGPD_RESULT (*fsgpdRectanglePath)(FSGPD_DC,FSGPD_INT,FSGPD_RECTANGLE*);
       FSGPD_RESULT
(*fsgpdRoundRectanglePath)(FSGPD_DC,FSGPD_INT,FSGPD_ROUNDRECTANGLE*);
       FSGPD_RESULT (*fsgpdBezierPath)(FSGPD_DC,FSGPD_INT,FSGPD_POINT*);
       FSGPD_RESULT
(*fsgpdArcPath)(FSGPD_DC,FSGPD_ARCMODE,FSGPD_ARCDIR,FSGPD_FIX,FSGPD_FIX,FSGPD_FIX,FSG
PD_FIX,FSGPD_FIX,FSGPD_FIX,FSGPD_FIX,FSGPD_FIX);
       FSGPD_RESULT
(*fsgpdDrawImage)(FSGPD_DC,FSGPD_INT,FSGPD_INT,FSGPD_INT,FSGPD_INT,FSGPD_IMAGEFORMAT,
FSGPD_RECTANGLE,void*);
       FSGPD_RESULT
(*fsgpdStartDrawImage)(FSGPD_DC,FSGPD_INT,FSGPD_INT,FSGPD_INT,FSGPD_INT,FSGPD_IMAGEFO
RMAT,FSGPD_RECTANGLE);
       FSGPD_RESULT (*fsgpdTransferDrawImage)(FSGPD_DC,FSGPD_INT,void*);
       FSGPD_RESULT (*fsgpdEndDrawImage)(FSGPD_DC);
       FSGPD_RESULT (*fsgpdStartScanline)(FSGPD_DC,FSGPD_INT);
       FSGPD_RESULT (*fsgpdScanline)(FSGPD_DC,FSGPD_INT,FSGPD_INT*);
       FSGPD_RESULT (*fsgpdEndScanline)(FSGPD_DC);
       FSGPD_RESULT (*fsgpdStartRaster)(FSGPD_DC,FSGPD_INT);
       FSGPD_RESULT (*fsgpdTransferRasterData)(FSGPD_DC,FSGPD_INT,FSGPD_BYTE*);
       FSGPD_RESULT (*fsgpdSkipRaster)(FSGPD_DC,FSGPD_INT);
       FSGPD_RESULT (*fsgpdEndRaster)(FSGPD_DC);
       FSGPD_RESULT (*fsgpdStartStream)(FSGPD_DC);
       FSGPD_RESULT (*fsgpdTransferStreamData)(FSGPD_DC,FSGPD_INT,void*);
       FSGPD_RESULT (*fsgpdEndStream)(FSGPD_DC);
} FSGPD_API_PROCS;

#endif /* _FSGPD_H_ */
```

VII.Authors and Contributers

Editors

Osamu Mihara – Fuji Xerox Printing Systems Co., Ltd.
Yasumasa Toratani – Canon Inc.

Authors

Osamu Mihara – Fuji Xerox Printing Systems Co., Ltd.
Yasumasa Toratani – Canon Inc.

Contributers

(alphabetical order) Hidekazu Hagiwara (MintWave), Masaki Iwata (AXE), Hidenori Kanjo (BBR), Shinpei Kitayama (Epson Kowa), Kenichi Maeda (E&D), Akio Maruyama (Ricoh), Hisao Nakamura (E&D), Koji Otani (AXE), Kenji Wakabayashi (MintWave), Toshihiro Yamagishi (Turbolinux), Akira Yoshiyama (NEC)

VIII.History

Version 0.1 (Japanese) Mihara/Toratani

Version 0.2 (Japanese) Mihara

Openprinting-0.1.1 by opfc have implemented based on this version

Version 0.2-en 2004-3-14 Mihara/Toratani/Kitayama/Yamagishi/Kanjo

Translation to English
Some feedback from opfc implementation

Version 1.0 RC1 2005-7-20 Mihara

Change copyright notice from FDL to MIT style copyright

Delete temporary font operations.

Version 1.0 RC2 2006-6-9/2006-6-17/2006-6-20/2006-6-26/2006-6-29

Add API version number to OpenPrinter() parameter

Add pitch and delete count to/from DrawImage()/StartDrawImage()

Add a figure for SetMiterLimit.

Change function names/constands/enum/structures names to add FSG prefixes.

Change copyright year and dates

Delete “Printer Driver Database”

Delete *cmap* from CSPASE chart.

Fix parameter colorSpace (wrong) to colorDepth (correct) for DrawImage()/StartDrawImage

Change description for fsgpdStartJob/fsgpdEndJob/fsgpdAbortJob

Add type FSGPD_CHAR for character type.