

Boca Systems SDK Library Routines

Establish Interface connection

Use the following routines to open and close a communication session via WiFi or Bluetooth. In both cases, the session must be closed prior to closing the app.

Open a WIFI session using the printer's IP address (the printer's IP address can be found on the printer's self-test ticket).

```
BOOL status = OpenSessionWiFi(NSString *IP_Address)
void CloseSessionWiFi()
```

Open a BlueTooth session (The printer must be "paired" with the iPad prior to opening the session).

```
BOOL status = OpenSessionBT()
BOOL status = OpenSessionBTPicker(NSString *name)
void CloseSessionBT()
```

Read Printer Status

This routines reads the status of the printer and returns a string describing the current state.

```
NSString *ReadPrinter()
```

Examples of status strings returned:

```
"Ticket ACK"
"Invalid Checksum"
"Valid Checksum"
"Out of Tickets"
"X-On"
"Power On"
"X-Off"
"Ticket NAK"
"Ticket Jam"
"Cutter Jam"
```

Define Printer Configuration

This routine allows the user to select several common printer settings using a single call. The following settings are usually issued just once at the start of each ticket; the printer's paper path, scaling multiplier and print orientation.

```
void ChangeConfiguration(NSString *path, int multiplier, NSString *orientation)
```

- path - 1 (default) or 2 (dual path printer only)
- multiplier - 1 (default), 2, 3, 4 (Multiplies both height and width of images)
- orientation - <LM> (landscape, default) or <PM> (portrait)

Enable Image Dithering

Dithering is a feature that allows a black and white device, such as the printer, to simulate shades of gray. This feature is most useful on photographic image or images with color graduations where the gradual transition from white to black or black to white can improve image quality. Dithering should **not** be applied to images containing text or barcodes, this can soften the edges of these objects and make barcodes unreadable.

```
void EnableDithering(bool bEnableDithering)
```

- bEnableDithering = true (enable) or false (disabled)

Set Image Start Position

This function is used to set the starting position of a graphic image using FGL coordinates. The function is intended for positioning graphic images only. For positioning text or barcodes, see the "SendString" function and the <RC#,#> command in the FGL programming guide.

```
void SetStartPosition(int row, int column);
```

- row – FGL row position.
- column – FGL column position.

NOTE: FGL row/column positions start in the upper left corner of the ticket and are spaced in accordance with the printer's resolution. Printers with 200 dpi resolutions are spaced approximately at .005" intervals; 300 dpi printers at .0033"; 600 dpi at .00166".

Send Text and Barcodes

The optimal way to print text and barcodes is to use FGL commands and send them to the printer via the SendString function. The FGL language is text based and can be easily combined with user text strings to accomplish the desired results. Please refer to the FGL Programming Guide for an explanation of FGL command language and a list of valid FGL commands.

```
void SendString(NSString *string)
```

- string – string containing FGL commands and data in ASCII format.

Example:

```
SendString("<NR><RC100,100><F12>This is a text string<p>")
```

This example sets the text orientation, positions the initial print location, selects the font and inserts the text string. A FGL print command has been added to the end of the text string so a separate call to the PrintCut function (see below) is unnecessary.

FGL Print Commands

The following functions provide a convenient way to issue FGL print commands. The FGL printers are page printers and must receive a print command at the end of each page to cause the ticket or receipt to be printed. Printers may be equipped with cutters that automatically cut the ticket stock after printing is complete. The PrintNoCut function can be used to print a ticket without performing the normal cut. Both commands perform a print only function on printers not equipped with cutters.

```
void PrintCut()
```

```
void PrintNoCut()
```

Send Graphic Image

Images can be sent to the printer either from locations on the web or stored locally on the iPad. The SendWebImage routine accesses a web image by referencing the image's URL or a local image using the image's path and filename. The images are automatically converted from color into black and white and sent to the printer in a compressed graphic format.

```
bool SendWebImage(NSString* urlString, int row, int column);
```

- urlString – Pointer to URL address or local filename.
- row – FGL starting row position.
- column – FGL starting column position.

This function supports the following images formats:

```
.bmp  
.png  
.jpg  
.jpeg
```

Examples:

Web Image URL:

<https://bocasystems.com/images/ComicPicJPG.png>

Local Filename using fully resolved path:

```
/var/mobile/Containers/Data/Application/8483652C-4E59-4130-B726-2502433B756F/Documents/
```

test.bmp

Note: When saving and retrieving local files it is recommended that you use a subdirectory within the application's home directory as the default location. The following iOS routine can be used to retrieve the path name to the application's home directory.

```
NSHomeDirectory()
```

```
/var/mobile/Containers/Data/Application/8483652C-4E59-4130-B726-2502433B756F
```

Send UIImage

The SendFileImage function converts and sends a UIImage image along with the desired starting row and column position to the printer. Unlike a web image, the UIImage image object maintains a local copy of the image and can be sent multiple times using the built-in “print/send” function. Just like the web images, the UIImage is converted to black and white and sent to the printer in a compressed graphic format.

```
bool SendFileImage(UIImage *image, int row, int column)
```

- image – Pointer to UIImage image object.
- row – FGL starting row position.
- column – FGL starting column position.

Send Image Files (using file picker)

This function provides an alternative method for accessing a locally stored image file using a NSURL object. This routine retrieves an image from outside of the application’s sandbox using iOS to retrieve the file using the file picker.

```
bool SendFile(NSURL *filename, int row, int column)
```

SendFile is called from the test app GUI. The GUI specifies which directory to select using the following:

```
UIImagePickerControllerSourceTypeSavedPhotosAlbum  
UIImagePickerControllerSourceTypeSavedPhotosLibrary
```

The pathname supplied by the FilePicker is a reference to an image asset. For example:

```
assets-library://asset/asset.PNG?id=318AA50D-21B7-491E-9BC6-C0852EE7B070&ext=PNG
```

Note: The asset-library reference string cannot be used in the SendWebImage text box function.

Download FGL logo

An FGL logo is an image that has been downloaded to the printer and saved in flash memory. The saved image can then be referenced and inserted into a ticket using the print logo function as explained below. For images that are going to be used repeatedly this eliminates the communications overhead associated with sending the same image for each ticket.

```
bool DownloadLogo(NSURL *filename, int idnum)
```

- filename – Pointer to image filename.
- Idnum – User assigned integer value to uniquely identify the logo.

NOTE: The user must keep track of downloaded logos, to avoid overwriting previously stored logos.

Print Logo

The PrintLogo function requires the logo's integer ID value and its starting row/column location. After inserting the logo on the ticket use the PrintCut or PrintNoCut function to print the ticket. (NOTE: The multiplier function is ignored when printing images from logos.)

```
bool PrintLogo(NSString *idnum, int row, int column)
```

Example:

```
PrintLogo("1", 100, 100);  
void PrintCut()
```

The above example prints user logo 1 at position 100, 100 on the ticket.

Clear User Download Memory

This function clears the printer's flash memory of all downloaded user files including logos.

```
void ClearMemory()
```

Make Subdirectory in Home Directory

This function returns the full pathname of a subdirectory in the Home Directory. The returned pathname is the Home Directory (returned by NSHomeDirectory()) appended with the specified subDir. It can be used to Create a Directory (see below).

```
NSString *MakeSubDirInHomeDirectory(NSString *subDir)
```

Create a Directory

This function creates a directory using the specified pathname.

```
bool CreateDirectory(NSString *pathname)
```

Read File

This function reads and returns the data from a specified pathname.

```
NSData *ReadFile(NSString *pathname)
```

Write Data to File

This function writes data to a specified file (pathname).

```
bool WriteDataToFile(NSString *pathname, NSData *data)
```

General Overview

While the printer will support any combination of text, barcode and images, the highest throughput is achieved by sending as little data as possible with each ticket. The following techniques should help to minimize data transmissions:

- Send as much data as possible as text and barcode via FGL commands.
- Download the fixed portion of your ticket as a logo
 - Use the PrintLogo command to access the logo
 - Use SendString for text and barcode, if possible
 - Use SendWebImage or SendFileImage for non-fixed images
- Minimize the size of images sent via SendWebImage and/or SendFileImage
- Send scaled down images and use multiplier (a 2x multiplier will reduce transmission time by a factor of 4 ; a 4x multiplier will reduce by a factor of 16)

NOTE: Scaled down images will lose resolution, particularly with dithered output. Dithering should only be used when printing pictures or attempting to represent shading and/or colors.