SAM4S iOS SDK

Programmers Manual
Version 6.0

11/12/2015

Table of Contents

1. Introduction

The SDK Sample for iOS is an SDK aimed at development engineers who are developing iPhone/iPad applications for printing on SAM4S printer. This document contains the purpose of the SDK, the methods which are used in API to refer commands.
3 phases involved in this document:

- Bluetooth
- Wireless
- Ethernet

2. Purpose

The purpose of this document is to provide the necessary technical details for using the SAM4S iOS SDK to develop iOS applications for SAM4S Printers.

3. SAM4S iOS SDK

Sam4S iOS SDK is used to develop the iOS application for SAM4S printer. It contains the api code for all the features of sam4s printer.
Using SAM4S SDK programmers can search and connect the printer on the specified network (Ethernet, Bluetooth and Wireless) and implement the following printer functionalities.

- Text Printing
- Image Printing
- Bar Code Printing
- 2D Code Printing
- Page Mode
- Status of the Printer
- Beep, Printer Name, Printer Information and Cut.

4. Using the SAM4S SDK

This SDK helps developers to search SAM4S printers via Bluetooth, Ethernet and Wireless and establish connection and printing from iPhone and iPad application.

1. Features
    - Allows search SAM4S printers from iPhone or iPad application.
    - Allows establish connection from iPhone or iPad application.
    - Allows printing from iPhone or iPad application.
    - Allows acquisition printer status from iPhone or iPad application.

SAM4S iOS SDK Classes

Sam4SPrintIO class

This class contains the commands to communicate with the Printer. To implement the various print features like text image and barcode etc. the Sam4SPrintIO class is used.

Sl.NO	Features	Method Name	Description
1	**Printer Information**	PrinterSetup	Initialize the SAM4S Printer
2		getPrinterFirmware	Get the printer firmware version
3		getPrinterManufacturer	Get the printer Manufacturer name
4		getPrinterFont	Get the printer font
5		getPrinterName	Get the printer Name
6	**Cut**	addCut	Cuts the receipt, with paper feed or without paper feed as specified by the argument.
7		addBlackMark	Used to add a black mark in the specified cutting position.
8	**Beep**	addTone	Generates an audible beep tone.
9	**Paper Feed**	addFeedLine	Used to add and specify no of feed in lines after print
10		addFeedUnit	Used to add the paper feed in dots to the command buffer
11	**Text Print**	addTextAlignment	Used to specify the alignment of characters, graphics, logos, and bar codes on the receipt station
12		addLineSpace	Used to specify the line space for characters on the receipt station
13		addTextData	Adds text to printer buffer
14		addTextLanguage	Used to specify the language of characters on the receipt station
15		addTextBold	Used to specify the bold for characters on the receipt station.
16		addTextUnderline	Used to specify the line space for characters on the receipt station.
17		addReverse	Used to specify the reverse printing is on or off for characters on the receipt station.
18		addTextDirection	Used to specify the text printing direction for characters on the receipt station.
17		addTextDouble	Used to specify the text width or height or both as double-sized for characters

			on the receipt station.
18		addTextSize	used to specify the text height & width for characters on the receipt station
19		addTextPosition	Used to specify the text x position for characters on the receipt station.
20		addTextStyle	Used to specify the text style for characters on the receipt station.
21		addTextFont	Used to specify the font of characters, graphics, logos, and barcodes on the receipt station.
22	**ImagePrint**	addLogo	This function is used to print downloaded logo on receipt. mode - should be one of (Normal,Double-Wide,Double-High,Quadruple)
23		addImageWithData	Used to add the image & size of the image on the receipt station with sharpen & brighten value of the image
24		addImageData	Used to add the image & size of the image on the receipt station.
25	**BarCode**	addBarcode	used to generate the barcode based on the Parameter
26	**2D Code**	addPDF417Symbol	Generate the two dimensional barcode - PDF 417
27		addQRCodeSymbol	Generate the two dimensional barcode - QRCode
28	**PageMode**	addPageBegin	adds page begin to printer buffer for page mode
29		addPageEnd	adds page end to printer buffer for page mode
30		addPageDirection	Used to specify the direction of the page on the receipt station.
31		addPagePosition	Used to specify the page position of x and y for characters on the receipt station.
32		addPageArea	Used to specify the page area of x, y, height & width for page on the receipt station.
33		addPageFormFeed	In page mode, print all buffered data in the printing area collectively.
34	**Status**	turnOFFASB	Used to turn off the ASB
35		getPrinterStatus	To retrieve the printer status
36	**OpenCashDrawer**	addOpenCashDrawer	Used to open a cash drawer with

			generate pulse and time
37	**Send/Receive Command**	getDataTosend	Add all the commands into an array and send to Printer
38		didReceivedDataFromPrinter	If there is any event occurs in the printer, this method will receive the response (Automatic Status Back) as data.

WiredConnection Class

WiredConnection is derived from PrinterConnection class and it is used to establish the connection with IP Address and send/receive data from/to printer for Ethernet and Wireless interface.

Sl.NO	Function Name	Description
1	initConnectionWithPrinter	Initialize the printer connection with the Wireless or Wired instance
2	openConnection	Open the communication with the Printer in the specific interface
3	closeConnection	Close the communication with the Printer in the specific interface
4	sendDataToPrinter	Send all the commands to the printer
5	readDataFromPrinter	Receives data from the printer

BTConnection Class

BTConnection class derived from PrinterConnection class and it is used to establish the connection with IP Address and send/receive data from/to printer for Bluetooth interface.

Sl.NO	Function Name	Description
1	initWithExternalAccessory	Initialize the printer connection with the Wireless or Wired instance
2	openConnection	Open the communication with the Printer in the specific interface
3	closeConnection	Close the communication with the Printer in the specific interface
4	sendDataToPrinter	Send all the commands to the printer
5	readDataFromPrinter	Receives data from the printer

WiredConnectionManager Class

WiredConnectionManager class is derived from ConnectionManager class and it is used to discover SAM4S printers with IP Address for Ethernet interface.

Sl.NO	Function Name	Description
1	connectedPrintersIPAddress	Used for discovery of Ethernet printer available in the network in which the iOS device is connected
2	isValidIpAddress	used to check the validity of IP address
3	getDefaultGatewayIp	used to get the Gateway IP address

BTConnectionManager Class

BTConnectionManager class is derived from ConnectionManager class and it is used to discover the printer with the protocol name for Bluetooth interface.

Sl.NO	Function Name	Description
1	connectedPrintersNameForBluetooth	Used for the discovery of Bluetooth printer available in the specific range in which the iOS device is connected.

5. Operating Environment

- **Supported iOS versions**
  - OS Version 7.0 or above
- **Supported iOS Devices**
  - iPhone 5, 5S, 5C or above
  - iPad 2, iPad 3rd gen, iPad 4th gen, iPad Air or above
  - iPad Mini 2nd gen, iPad Mini 3rd gen or above
- **Supported SAM4S Printers**
  - ELLIX 20II, 30, 35, 40, 45
  - GIANT - 100
- **Development Environment**
  - Xcode Version 5.0 or above

6. Programming Guide

This section explains how to write programs in the application development using SAM4S SDK Sample.

1. How to include SDK Sample in the iOS application

Following are the steps to include SDK Sample in iOS application.

i. Create a new project in XCode.
   Drag the provided Objective-C header (Printer.h, , Printer+Private.h, Connection Manager.h, PrinterConnection.h etc) and drop it anywhere you like in the target project's hierarchy in Xcode's [Project Navigator].

ii. Use the following procedure to integrate ExternalAccessory.framework:
   - In Project Navigator, select the Project file (the root file).
   - Select Targets → Build Phases.
   - Expand "Link Binary With Libraries", and tap the [+] button
   - Select ExternalAccessory.framework, and tap the Add button.
   - Select SystemConfiguration.framework, and tap the Add button

iii. Write the Objective-C header import declaration in the *.m source file(s) of the application you would like to use this SDK in as follows:

```
#import "Printer.h"
#import "BTConnection.h"
#import "WiredConnection.h"
```

2. Search Printers

a. Ethernet Interface

I. Developer need to call connectedPrintersIPAddress method of ConnectionManager class to get list of available printer device. Following is code snippet for this:

```
// Ethernet  interface discovery
wiredManager = [ConnectionManager sharedConnectionWithType:ConnectionTypeWiFi
delegate:self];
// Ethernet interface - Auto Discovery
[printerListArray addObjectsFromArray:[wiredManager connectedPrintersIPAddress]];
```

II. If the Ethernet Printer IP address is not discovered, we can call the custom protocol WiredConnectionUDPDelegate and implement the didReceiveIPAddress method in our app.

Steps to send/Receive UDP packets

I. GCDAsyncUdpSocket class is used to discover the Ethernet printer.

II. Add the custom protocol WiredConnectionUDPDelegate in interface file to receive the IP from the printer. didReceiveIPAddress method comes under the custom protocol.

III. Calculate the broadcast address of the Router in which the device is connected.

IV. Send the UDP packet to the broadcast address which will return the ethernet Printer. The following is the code defined inside the sdk to initialize the udp socket and send data to the UDP socket.

```
// Send the broadcast request, ie "Any upnp devices out there?"
char request[35];
request[0] = (Byte) 0xAA;
request[1] = (Byte) 0xAA;
request[sizeof(request)-2] = (Byte)0x55;
request[sizeof(request)-1] = (Byte)0x55;

NSData *data = [NSData dataWithBytes:request length:sizeof(request)];
UInt16 port=2918;
if (![udpSocket bindToPort:port error:&error])
{ return;}
if (![udpSocket beginReceiving:&error])
{    return;}
if (![udpSocket enableBroadcast:YES error:&error]) {
  return;}
[udpSocket enableBroadcast:YES error: &error];
   [udpSocket setIPv4Enabled:YES];
   [udpSocket setIPv6Enabled:NO];
[udpSocket sendData :data toHost:[self getBroadcastAdress] port:port withTimeout:timeout
tag:0];
```

V. Define didReceiveIPAddress method in the applicationwhich is used to receive the discovered IP address.

```
// Ethernet Discovery
- (void)didReceiveIPAddress : (NSArray *)discoveredIP
{
// discoveredIP is the array of IP address discovered from the UDP broadcasts.
 }
```

b. Bluetooth Interface

Developer need to call connectedPrintersNameForBluetooth method of ConnectionManager class to get list of paired printer device. Following is the code snippet:

```
    bluetoothManager = [ConnectionManager
sharedConnectionWithType:ConnectionTypeBlueTooth delegate:self];

    [printerListArray addObjectsFromArray:[bluetoothManager
connectedPrintersNameForBluetooth]];
```

c. Wireless Interface

There is no discovery mechanism for wireless interface. The IP address for the wireless printer should be added manually.

```
// Wireless interface and Manual Entry
printerListArray =  printerListArray = [NSMutableArray arrayWithArray:[manualEntryIP
arrayByAddingObjectsFromArray:[wiredManager connectedPrintersIPAddress]]];
```

3. Establish Connection

To create connection and disconnection between iOS devices and SAM4S printers, developer needs to use following piece of code:

a. Wireless/Ethernet Interface

```
wiredManager = [ConnectionManager sharedConnectionWithType:ConnectionTypeWiFi
delegate:self];

//To Connect a device use following code:-
connectedPrinter = [wiredManager makeConnectWithSelectedPrinter:selectedIPAddr];

//To Disconnect a device use following code:-
[wiredManager disconnectPrinter: connectedPrinter];
      connectedPrinter = nil;
```

b. Bluetooth Interface

```
    bluetoothManager = [ConnectionManager
sharedConnectionWithType:ConnectionTypeBlueTooth delegate:self];

//To Connect a device use following code:-
            connectedPrinter = [bluetoothManager
makeConnectWithSelectedPrinter:selectedAccessory];

//To Disconnect a device use following code:-
[bluetoothManager disconnectPrinter: connectedPrinter];
       connectedPrinter = nil;
```

7. SAM4S iOS Printer SDK Commands

Printer Info Commands

PrinterSetup

Description: Used to initialize the printer.
Syntax: (void)PrinterSetup;
Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[printer.printIO.connection sendDataToPrinter:[posPrinter
getDataTosend:posPrinter]];
```

getPrinterFirmware

Description: Used to get the printer firmware version.
Syntax: (void)getPrinterFirmware;
Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter getPrinterFirmware];
[printer.printIO.connection sendDataToPrinter:[posPrinter
getDataTosend:posPrinter]];
```

getPrinterManufacturer

Descriptions: Used to get the printer manufacturer name.
Syntax: (void)getPrinterManufacturer;
Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter getPrinterManufacturer];
[printer.printIO.connection sendDataToPrinter:[posPrinter
getDataTosend:posPrinter]];
```

getPrinterFont

Descriptions: Used to get the printer font.
Syntax: (void)getPrinterFont;
Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter getPrinterFont];
[printer.printIO.connection sendDataToPrinter:[posPrinter
getDataTosend:posPrinter]];
```

getPrinterName

Descriptions: Used to get the printer name.
Syntax: (void)getPrinterName;
Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter getPrinterName];
[printer.printIO.connection sendDataToPrinter:[posPrinter
getDataTosend:posPrinter]];
```

Paper Cut Command

addCut

Descriptions: Used to cut the paper with paper feed or without the paper feed.
Syntax: (void)addCut:(enum CUTTYPE)cutType;
Parameter: cutType:should be CUT_FEED or CUT_NO_FEED
Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter addCut:CUT_FEED];
[printer.printIO.connection sendDataToPrinter:[posPrinter
getDataTosend:posPrinter]];
```

Beep Command

addTone

Descriptions: Used to generate the audible tone.
Syntax: (void)addTone;
Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter addTone];
[printer.printIO.connection sendDataToPrinter:[posPrinter
getDataTosend:posPrinter]];
```

Paper Feed Commands

addFeedUnit

Description: Used to add the paper feed in dots to the command buffer.
Syntax: (void)addFeedUnit : (int)Unit;
Parameter: Unit: The value of Unit should be 1 to 255
Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter addFeedUnit:30];
[printer.printIO.connection sendDataToPrinter:[posPrinter
getDataTosend:posPrinter]];
```

addFeedLine

Description: Used to add the number of lines of paper feed after text or other print.
Syntax: (void)addFeedLine : (int)line;
Parameter: Unit: The value of Unit should be 1 to 255
Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter addFeedLine:30];
[printer.printIO.connection sendDataToPrinter:[posPrinter
getDataTosend:posPrinter]];
```

Text Print Commands

addTextAlignment

Descriptions: Used to align the text to print.
Syntax: (void)addTextAlignment:(enum ALIGNMENT)alignment;
Parameter: alignment: The value of alignment should be LEFT or RIGHT or CENTER
Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter addTextAlignment:LEFT];
[printer.printIO.connection sendDataToPrinter:[posPrinter
getDataTosend:posPrinter]];
```

addLineSpace

Descriptions: Used to feed lines of paper.
Syntax: (void)addLineSpace:(int)space;
Parameter: space: The value of space should be integer
Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter addLineSpace:30];
[printer.printIO.connection sendDataToPrinter:[posPrinter getDataTosend:posPrinter]];
```

addTextData

Description: Used to add text to print.
Syntax: (void)addTextData:(NSString*)text;
Parameter: text: The value of text should be string
Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter addTextData:@"Sample text to print"];
[printer.printIO.connection sendDataToPrinter:[posPrinter
getDataTosend:posPrinter]];
```

addTextLanguage

Description: Used to add language for text to print.
Syntax: (void) addTextLanguage: (enum LANGUAGE) language;
Parameter: language: The value of language should be any language code. Select International character set where numeric language is:

0 - USA	7 - Spain I
1 - France	8 - Japan
2 - Germany	9 - Norway
3 - United Kingdom	10 - Denmark II
4 - Denmark I	11 - Spain II
5 - Sweden	12 - Latin America
6 - Italy	13 - Korea

Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter addTextLanguage:ENG];
[printer.printIO.connection sendDataToPrinter:[posPrinter
getDataTosend:posPrinter]];
```

addCharacterCode

Description: Used to add the character code setting to a command buffer.
Syntax: (void) addCharacterCode : (enum CHARACTERCODE)characterCode;
The value of the characterCode is as follows

i. CHARACTER_CODE_PC437 = 0;
ii. CHARACTER_CODE_KATAKANA = 1;
iii. CHARACTER_CODE_PC850 = 2;
iv. CHARACTER_CODE_PC860 = 3;
v. CHARACTER_CODE_PC863 = 4;
vi. CHARACTER_CODE_PC865 = 5;
vii. CHARACTER_CODE_PC737 = 14;
viii. CHARACTER_CODE_WPC1252 = 16;

ix. CHARACTER_CODE_PC866 = 17;
x. CHARACTER_CODE_PC852 = 18;
xi. CHARACTER_CODE_PC858 = 19;
xii. CHARACTER_CODE_THAI11 = 21;
xiii. CHARACTER_CODE_THAI18 = 26;
xiv. CHARACTER_CODE_PC775 = 33;
xv. CHARACTER_CODE_PC855 = 34;
xvi. CHARACTER_CODE_PC862 = 36;
xvii. CHARACTER_CODE_PC864 = 37;
xviii. CHARACTER_CODE_FARSI = 27;
xix. CHARACTER_CODE_WPC1250 = 45;
xx. CHARACTER_CODE_WPC1251 = 46;
xxi. CHARACTER_CODE_WPC1253 = 47;
xxii. CHARACTER_CODE_WPC1255 = 49;
xxiii. CHARACTER_CODE_WPC1256 = 50;
xxiv. CHARACTER_CODE_WPC1257 = 51;
xxv. CHARACTER_CODE_THAI_STANDARD_620 = 95;
xxvi. CHARACTER_CODE_THAI42 = 96;
xxvii. CHARACTER_CODE_THAI14 = 97;
xxviii. CHARACTER_CODE_THAI16 = 98;
xxix. CHARACTER_CODE_SYSTEM_IRAN_CODE = 99;
xxx. CHARACTER_CODE_SPACE_PAGE = 255

Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter addCharacterCode:CHARACTER_CODE_PC850];
[posPrinter getDataTosend:posPrinter]];
```

addTextBold

Description: Used to add bold with text to print.
Syntax: (void)addTextBold : (BOOL)set;
Parameter: set: The value of set should be YES or NO
Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter addTextBold:YES];
[printer.printIO.connection sendDataToPrinter:[posPrinter
getDataTosend:posPrinter]];
```

addTextUnderline

Description: Used to add underline with text to print.
Syntax: (void)addTextUnderline : (BOOL)set;
Parameter: set: The value of set should be YES or NO
Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter addTextUnderline:YES];
[printer.printIO.connection sendDataToPrinter:[posPrinter
getDataTosend:posPrinter]];
```

addReverse

Description: Used to add reverse text to print.
Syntax: (void)addReverse : (BOOL)set;
Parameter: set: The value of set should be YES or NO
Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter addReverse:YES];
[printer.printIO.connection sendDataToPrinter:[posPrinter
getDataTosend:posPrinter]];
```

addTextDirection

Description: Used to add or specify the text printing direction.
Syntax: (vid)addTextDirection : (enum TEXTDIRECTION)direction;
Parameter: Direction: The value of direction should be DIRECTION_LEFT_TO_RIGHT,

1) DIRECTION_BOTTOM_TO_TOP,
2) DIRECTION_RIGHT_TO_LEFT,
3) DIRECTION_TOP_TO_BOTTOM

Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter addTextDirection:DIRECTION_LEFT_TO_RIGHT];
[printer.printIO.connection sendDataToPrinter:[posPrinter
getDataTosend:posPrinter]];
```

addTextDouble

Description: Used to print text width as double-sized or height as double-sized or both.
Syntax: (void)addTextDouble : (BOOL)doubleWidth : (BOOL)doubleHeight;
Parameter:

doubleWidth: The value of direction should be YES or NO
doubleHeight : The value of direction should be YES or NO

Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter addTextDouble:YES : NO];
[printer.printIO.connection sendDataToPrinter:[posPrinter
getDataTosend:posPrinter]];
```

addTextSize

Description: Used to print text width and height as specified parameter.
Syntax: (void)addTextSize : (int)width :(int)height;
Parameter:

width: The value of width should be 1 to 8
height: The value of height should be 1 to 8

Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter addTextSize:2 : 2];
[printer.printIO.connection sendDataToPrinter:[posPrinter
getDataTosend:posPrinter]];
```

addTextPosition

Description: Used to add the print position of text.
Syntax: (void)addTextPosition: (int)xposition;
Parameter: xposition: The value of xposition should be 1 to 255
Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter addTextPosition:2];
[printer.printIO.connection sendDataToPrinter:[posPrinter
getDataTosend:posPrinter]];
```

addTextStyle

Description: Used to add style for text to print.
Syntax: (void)addTextStyle:(BOOL)reverse :(BOOL)underline :(BOOL)bold :(int)color;
Parameter:

color: The value of set should be YES or NO
reverse: The value of set should be YES or NO
underline: The value of set should be YES or NO
bold: The value of set should be YES or NO

Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter addTextStyle:YES :NO : YES :NO];
[printer.printIO.connection sendDataToPrinter:[posPrinter
getDataTosend:posPrinter]];
```

addTextFont

Description: Used to add font to print text.
Syntax: (void)addTextFont:(enum FONT)font;
Parameter: font: The value of font should FONT_A, FONT_B
Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter addTextFont:FONT_A];
[printer.printIO.connection sendDataToPrinter:[posPrinter
getDataTosend:posPrinter]];
```

Image Print Commands

addLogo

Description: This function is used to print downloaded logo on receipt
Syntax: - (void) addLogo:(int)imageNo : (enum PRINTMODE)logomode;
Parameter:

imageNo : Specifies an integer value of image stored .
logomode : specifies one of following mode

- Normal
- Double-Wide
- Double-High

- Quadruple

Example:

```
SAM4SPrintIO *sam4sIO = (SAM4SPrintIO *)printer.printIO;
printer.printIO.connection.delegate = self;
[sam4sIO PrinterSetup];
[sam4sIO addLogo:3 : NORMAL];
[printer.printIO.connection sendDataToPrinter:[sam4sIO getDataTosend:sam4sIO]]
```

addImageData

Description: Used to add the image with image size to print. Sharpen and brightness options added to the existing functions with the default value..

Syntax: -(void) addImageData:(UIImage *) chosenImage imageWidth:(int) width imageHeight:(int)height imageMode:(enum PRINTMODE)imageMode;

Parameters:

data : Specifies an instance of the image.
width : Specifies width of print area. Specifies an integer from 1 - 65535.
height : Specifies height of print area. Specifies an integer from 1- 65535.
imageMode : Specifies the image print mode

Example:

```
SAM4SPrintIO *sam4sIO = (SAM4SPrintIO *)printer.printIO;
printer.printIO.connection.delegate = self;
[sam4sIO PrinterSetup];
//chosenImage is the image selected from gallery

//Without sharpen & brightness property
[sam4sIO addImageData:chosenImage imageWidth: chosenImage.size.width imageHeight:
chosenImage.size.height imageMode:NORMAL];
[printer.printIO.connection sendDataToPrinter:[sam4sIO getDataTosend:sam4sIO]];
```

addImageWithData

Description: Used to add the image with image size to print. Sharpen and brightness options added to the existing functions.

Syntax: (void) addImageWithData:(UIImage *) chosenImage imageWidth:(int) width imageHeight:(int)height imageMode:(enum PRINTMODE)imageMode sharpenMode : (int)shapern_mode imageBrightness :(int)brightness;

Parameters:

data : Specifies an instance of the image.
width : Specifies width of print area. Specifies an integer from 1 - 65535.
height : Specifies height of print area. Specifies an integer from 1- 65535.
imageMode : Specifies the image print mode
sharpen_mode : Specifies the correct integer value for sharpen. Here are the values

1. SHARPEN_MODE_OFF = 0;
2. SHARPEN_MODE_ON = 1;
3. SHARPEN_MODE_ON_MORE = 2;

Brightness : Specifies the correct value for brighten the image. Default value is 128

Example:

```
SAM4SPrintIO *sam4sIO = (SAM4SPrintIO *)printer.printIO;
printer.printIO.connection.delegate = self;
[sam4sIO PrinterSetup];
//chosenImage is the image selected from gallery - With sharpen & brightness property
[sam4sIO addImageWithData: chosenImage imageWidth: chosenImage.size.width
imageHeight: chosenImage.size.height imageMode:NORMAL] sharpenMode :
SHARPEN_MODE_OFF imageBrightness: BRIGHTNESS_DEFAULT];
[printer.printIO.connection sendDataToPrinter:[sam4sIO getDataTosend:sam4sIO]];
```

Barcode Commands

addBarcode

Description: Used to print the barcode based on the type

Syntax: (void)addBarcode :(NSData *)text : (enum BARCODETYPE)type : (enum HRI)hri : (enum FONT)font : (int)width : (int)height;

Parameter:

text: The value of text is numeric or alphanumeric
type : The value of type is bar code type
hri : the values are BELOW / ABOVE/ BOTH/ NONE
font: FONT_A or FONT_B
width: ranges from 2 to 6. Default value is 3
height: ranges from 1 to 255. Default value: 162

type	Bar Code System	Number of Characters in text
0,65	UPC-A	11<=k<=12
1,66	UPC-E	11<=k<=12

2,67	EAN13/JAN13	12<=k<=13
3,68	EAN8/JAN8	7<k<8
4,69	CODE 39	1<k<255
5,70	ITF	1<k<255(even no)
6,71	CODABAR	1<k<255
72	CODE93	1<k<255
73	CODE128	2<k<255

Example:

```
SAM4SPrintIO *posPrinter = (SAM4SPrintIO *)printer.printIO;
[sam4sIO PrinterSetup];
commandText.text = @"01234567890";
[posPrinter addBarcode:[commandText.text
dataUsingEncoding:NSUTF8StringEncoding] :UPC_A :BELOW :FONT_A :3 :162];
[printer.printIO.connection sendDataToPrinter:[posPrinter getDataTosend:posPrinter]];
```

addPDF417Symbol

Description: Used for generating pdf417 two dimensional bar codes.
Syntax: (void)addPDF417Symbol : (NSData *)text : (enum TWODCODETYPE)type : (enum
ERRORCORRECTIONSTANDARD) error : (int)width : (int)height;
Parameter:

text: The value of text to generate pdf 417 barcode.
type: The value of type is PDF417 - 48
error: the values are error correction level
Width: ranges from 2 to 8. Default value is 3
Height: ranges from 2 to 8. Default value: 3

Value of error level	Function	Error Correction Code Word
48	Select error correction level 0	2
49	Select error correction level 1	4

50	Select error correction level 2	8
51	Select error correction level 3	16
52	Select error correction level 4	32
53	Select error correction level 5	64
54	Select error correction level 6	128
55	Select error correction level 7	256
56	Select error correction level 8	512

Example:

```
SAM4SPrintIO *sam4sIO = (SAM4SPrintIO *)printer.printIO;
[sam4sIO PrinterSetup];
if(Type == STANDARD)
{
   commandText.Text = @"ABCDE"
   [sam4sIO addPDF417Symbol :[commandText.text dataUsingEncoding:NSUTF8StringEncoding]
   :STANDARDPDF417 :ErrorLevel 0 :3 :3];
}
[printer.printIO.connection sendDataToPrinter:[sam4sIO getDataTosend:sam4sIO]];
```

addQRCodeSymbol

Description: Used for generating pdf417 two dimensional bar codes.
Syntax: (void)addQRCodeSymbol : (NSData *)text : (enum TWODCODETYPE)type : (enum ERRORCORRECTIONQRMODEL)error : (int)maxSize;
Parameter:

text: The value of text to generate the qrcode.
type : The value of type is QRCODE - 49
error : the values are error correction level
width: ranges from 1 to 16. Default value is 3

Value of Type	Function
48	Select error correction level L
49	Select error correction level M

50	Select error correction level Q
51	Select error correction level H

Example:

```
SAM4SPrintIO *sam4sIO = (SAM4SPrintIO *)printer.printIO;
[sam4sIO PrinterSetup];
if(Type == QR_CODE_MODEL1)
{
    commandText.Text = @"ABCDE"
    [sam4sIO addQRCodeSymbol :[commandText.text dataUsingEncoding:NSUTF8StringEncoding]
    :QR_CODE_MODEL1 :ErrorLevel P :3 ];
}
[printer.printIO.connection sendDataToPrinter:[sam4sIO getDataTosend:sam4sIO]];
```

Page Mode Commands

addPageBegin

Description: Used to begin the page in the page mode.
Syntax: (void)addPageBegin;
Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter addPageBegin];
[printer.printIO.connection sendDataToPrinter:[posPrinter getDataTosend:posPrinter]];
```

addPageEnd

Description: Used to end the page in the page mode.

Syntax: (void)addPageEnd;

Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter addPageEnd];
[printer.printIO.connection sendDataToPrinter:[posPrinter
getDataTosend:posPrinter]];
```

addPageDirection

Description: Used to specify the page direction in the page mode.
Syntax:-(void)addPageDirection : (enum PAGEDIRECTION)direction;
Parameter: direction values are the followings.

I. PAGE_DIRECTION_LEFT_TO_RIGHT = 0
II. PAGE_DIRECTION_BOTTOM_TO_TOP = 1
III. PAGE_DIRECTION_RIGHT_TO_LEFT = 2
IV. PAGE_DIRECTION_TOP_TO_BOTTOM = 3

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter addPageDirection: PAGE_DIRECTION_LEFT_TO_RIGHT];
[printer.printIO.connection sendDataToPrinter:[posPrinter getDataTosend:posPrinter]];
```

addPagePosition

Description: Used to specify the page position in the page mode.
Syntax: (void)addPagePosition: (int)x : (int)y;
Parameter: x and y values are integer to specify the position
Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter addPagePosition: 10 : 100];
[printer.printIO.connection sendDataToPrinter:[posPrinter getDataTosend:posPrinter]];
```

addPageArea

Description: Used to specify the printing area in the page mode.
Syntax: (void)addPageArea:(int)x0 :(int)y0 : (int)dx : (int)dy;
Parameter:

When a paper width of 80mm: x0 = y0 = 0, dx = 512, dy = 1662
When a paper width of 58mm: x0 = y0 = 0, dx = 360, dy = 1662

Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter addPageArea: 100 : 100 :300 :300];
[printer.printIO.connection sendDataToPrinter:[posPrinter getDataTosend:posPrinter]];
```

addPageFormFeed

Description: Used to specify all buffered data in the printing area collectively in page mode.

Syntax: (void)addPageFormFeed;

Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter addPageFormFeed];
[printer.printIO.connection sendDataToPrinter:[posPrinter getDataTosend:posPrinter]];
```

ASB and turn off ASB (Automatic Status Back) Commands

turnOFFASB

Description: Used to turn off the ASB (Automatic status back).

Syntax: (void)turnOFFASB;

Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter turnOFFASB];
[printer.printIO.connection sendDataToPrinter:[posPrinter getDataTosend:posPrinter]];
```

getPrinterStatus

Description: Used to get the Printer status by enabling ASB(Automatic status back).

Syntax: (void)getPrinterStatus;

Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter getPrinterStatus];
[printer.printIO.connection sendDataToPrinter:[posPrinter getDataTosend:posPrinter]];
```

OpenCashDrawer Command

addOpenCashDrawer

Descriptions: Used to open a cash drawer and generate pulse.
Syntax: - (void)addOpenCashDrawer : (enum CASHDRAWER)drawerNo : (int)time;
Parameter:

drawerNo - This is the Pin number to generate pulse.
time - This is the Start time to generate pulse.

Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter addOpenCashDrawer: DRAWER_1 :10];
[printer.printIO.connection sendDataToPrinter:[posPrinter getDataTosend:posPrinter]];
```

Send and Receive data methods

getDataTosend

Description: Used to get all the data which are sent to the printer.
Syntax: (void)getDataToSend;
Example:

```
SAM4SPrintIO *posPrinter = [SAM4SPrintIO alloc]init];
[posPrinter PrinterSetup];
[posPrinter getPrinterStatus];
[posPrinter getDataTosend:posPrinter]];
```

didReceivedDataFromPrinter

Description: Used to get the data or information from the Printer if the printer sends.

ConnectionPrinterDelegate is a delegate contains the method to receive data from printer.

Steps to receive data from printer:

I. Import the header file (#import "WiredConnection.h") and add the delegate <ConnectionPrinterDelegate> in .h file.
II. Declare the delegate as self and declare didReceivedDataFromPrinter mehod in the implementation file
III. Send the command to the printer to receive status from printer. ASB should not be turned OFF.

```
SAM4SPrintIO *sam4sIO = (SAM4SPrintIO *)printer.printIO;
printer.printIO.connection.delegate = self;
[sam4sIO PrinterSetup];
[sam4sIO getPrinterStatus];
[printer.printIO.connection sendDataToPrinter:[sam4sIO getDataTosend:sam4sIO]];
```

IV. Automatically receive the response data in didReceivedDataFromPrinter Method and parsed in the proper way.

Syntax: (void) didReceivedDataFromPrinter:(NSData*) data

Example:

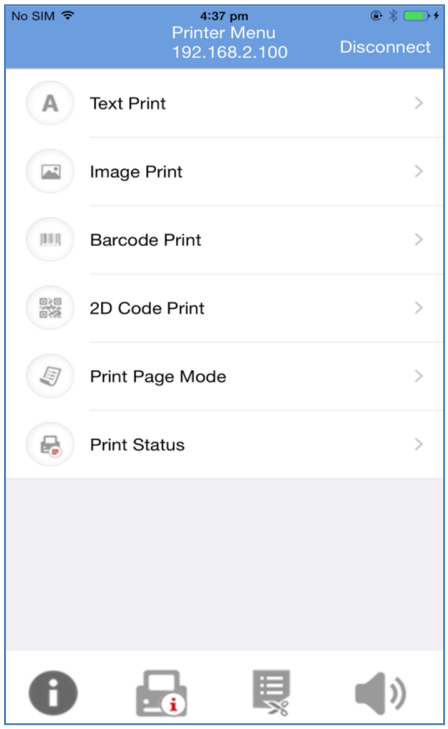
```
(void) didReceivedDataFromPrinter:(NSData *)data
{
//data is sent by the printer.
   unsigned char * bytePtr = (unsigned char  * )[data bytes];
//Parse the data as required
}
```

8. Sample iOS Application

The sample program has the following functionalities:

1. Auto discovery of printers (Ethernet Interface and Bluetooth Interface)
2. Manual Entry of IP Address (Wireless Interface)
3. Getting printer information and Printer name
4. Beep the buzzer
5. Cut (Cut_Feed, Cut_No_Feed)
6. Text Print
7. Image Print
8. BarCode Print
9. 2D Code Print
10. Printing in a Page mode
11. Getting the Status from the printer.

Main Application Screen



Using the Sample Application

1. Start the Sample Program
2. Search for Printers. Discover the printers and list in the table when you select the device type.
    a. Ethernet - Discovery by IP Address
    b. Bluetooth - Discovery by Protocol Name
3. Select the IP address/ Name of the printer which you want to use and do printing.
4. Sample application opens the printer and with the specified port no (e.g.: 6001).
5. This establishes the connection with the Printer and navigates to Sample Print. This is the main screen of the sample application.
6. When the process is completed, tap disconnect on the sample print screen.

Sl.No.	Process	Description
1	Printer Info	Tap first icon in the bottom toolbar of the main screen
2	Printer Name	Tap second icon in the bottom toolbar of the main screen
3	Paper cutting	Tap third icon in the bottom toolbar of the main screen
4	Beep the buzzer	Tap last icon in the bottom toolbar of the main screen
5	Text Print	Tap Text Print in the main screen
6	Image Print	Tap Image Print in the main screen
7	Barcode Print	Tap Barcode Print in the main screen
8	2D Code Print	Tap 2D Code Print in the main screen
9	Print Page Mode	Tap Print Page Mode in the main screen
10	Print Status	Tap Print Status in the main screen

1. **Printer Info**
   I. Tap the first icon in the bottom toolbar of the main screen.
   II. Printer info includes the following
      - Firmware
      - Manufacturer
      - Font

2. **Printer Name**
   I. Tap the second icon in the bottom toolbar of the main screen.
   II. Printer name is displayed in the pop-up.

3. **Paper cutting**
   I. Tap the third icon in the bottom toolbar of the main screen.

II. Pop-up will be displayed with the Cutting options. Select one among the following.
- CUT_FEED - Used to cut after feeding the paper
- CUT_NO_FEED - Used to cut without feed the paper.

4. Beep the buzzer

I. Select the last icon in the bottom toolbar of the main screen.
II. We will receive the audible tone from the printer.

5. Text Printing

I. Enter a string to print.
II. Select the character properties for the string to print. Followings are the properties used for text print.
III. Tap [Print] to print the image.

Property	Description
Font	Set the character font.
Alignment	Set the alignment.
Language	Set the language
Size	Set the character scales for width & height(horizontal/vertical)
Style	Set the character style (Bold/ Underline)
Linespace	Set the line feed space
X Position	Set the horizontal starting position
Feed	Set the paper feed unit

6. Image Print

I. Tab [Select Image] button to select the image from gallery.
II. Tap [Print] to print the image.

7. Barcode Print

I. Enter the data to generate the barcode.
II. Set the following properties for generating the barcode.
III. Tap [Print] to print the image.

Property	Description
Data	Enter the barcode data
Type	Set the barcode type.
HRI	Set the HRI position
Font	Set the HRI font
Module Size	Set the barcode module size(width/height)

8. 2D Code Print

I. Enter the data to generate the Two - Dimensional barcode.
II. Set the following properties for generating the 2D code.
III. Tap [Print] to print the image.

Property	Description
Data	Enter the 2D code data
Type	Set the 2D code type(Standard/QR Code)
Error Correction	Set the error correction level for the selected type
Module Size	Set the 2D code module size(width/height)
Max Size	Set the maximum 2D code size.

9. Print Page Mode

I. Enter the string to print characters in page mode.
II. Set the following properties for generating the 2D code.
III. Tap [Print] to print the image.

Property	Description
X	Set the origin of X axis for the printing area
Y	Set the origin of Y axis for the printing area

Width	Set the width for the printing area
Height	Set the height for the printing area

10. Print Status

I. Tap the Printer status on the main screen.
II. If any even occurred, this status of the event will be displayed in the screen.

Printer Document Creation

Create a print document using the SAM4SPrintIO class. Create a SAM4SPrintIO class instance to create a print document using the APIs of the SAM4SPrintIO class.

1. **To create a text Print Document**
   i. To create a text print document, using the APIs store the command settings in the command buffer.
   ii. For the String "Hello World", to create a print document based on the following settings:
      - Font : FONTA
      - Scale : X3(horizontal) and X3(Vertical)
      - Style : Bold
      - Alignment : CENTRE
      - Feed Unit :

Example:

```
SAM4SPrintIO *sam4sIO = (SAM4SPrintIO *)printer.printIO;
 [sam4sIO PrinterSetup];
  [sam4sIO addTextFont:FONT_A];
  [sam4sIO addTextAlignment:CENTER];
 [sam4sIO addTextSize:3 :3];
[sam4sIO addTextStyle:NO :NO :YES :COLOR_1];
[sam4sIO addTextData:@"Hello World"];
[sam4sIO addFeedUnit:30];
```

2. **To create a Graphic Print Document**
   1. To create a graphic print document, store the UIImage class object in the command buffer.

Example:

```
SAM4SPrintIO *sam4sIO = (SAM4SPrintIO *)printer.printIO;
```

```
[sam4sIO PrinterSetup];
UIImage *testImage = [UIImage imageWithName: “test.png”];
[sam4sIO addImageData:testImage imageWidth:testImage.size.width imageHeight:
testImage.size.height imageMode:NORMAL];
```

3. To create a Page Mode Print Document

1. To create a page mode print document, store the UIImage class object in the command buffer.

Example:

```
SAM4SPrintIO *sam4sIO = (SAM4SPrintIO *)printer.printIO;
[sam4sIO PrinterSetup];
[sam4sIO addPageBegin];
[sam4sIO addPageArea:0:0 :300:300];
[sam4sIO addTextFont:FONT_A];
[sam4sIO addTextAlignment:CENTER];
[sam4sIO addTextSize:3 :3];
[sam4sIO addTextStyle:NO :NO :YES :COLOR_1];
[sam4sIO addTextData:@”Hello World”];
[sam4sIO addPageEnd];
[sam4sIO  addCut:CUT_FEED];
```