USER MANUAL

IOS SDK FOR P3L

CUSTOM®

CUSTOM S.p.A.
Via Berettine 2/B
43010 Fontevivo (PARMA) - Italy
Tel. : +39 0521-680111
Fax : +39 0521-610701
http: www.custom.biz

Customer Service Department:
www.custom4u.it

© 2023 CUSTOM S.p.A. – Italy.
All rights reserved. Total or partial reproduction of this manual in whatever form, whether by printed or electronic means, is forbidden. While guaranteeing that the information contained in it has been carefully checked, CUSTOM S.p.A. and other entities utilized in the realization of this manual bear no responsibility for how the manual is used. Information regarding any errors found in it or suggestions on how it could be improved are appreciated. Since products are subject to continuous check and improvement, CUSTOM S.p.A. reserves the right to make changes in information contained in this manual without prior notification.

The pre-installed multimedia contents are protected from Copyright CUSTOM S.p.A. Other company and product names mentioned herein may be trademarks of their respective companies. Mention of third-party products is for informational purposes only and constitutes neither an endorsement nor a recommendation. CUSTOM S.p.A. assumes no responsibility with regard to the performance or use of these products.

TABLE OF CONTENTS

IMPORT SDK TO PROJECT

CONNECT THE SDK TO THE PRINTER

Import SDK 8

Search for printers 9

Search for printer callback and connect the printer in the callback 10

Adding notification of successful or failed printer connection 11

Process printer connection success and failure notifications 12

SDK sends data 13

SDK disconnects the printer 14

MANUALLY INPUT THE PRINTER IP AND CONNECT IT

Import SDK 15

Adding notification of successful or failed printer connection 16

Process printer connection success and failure notifications 17

DRAW PRINT CONTENT

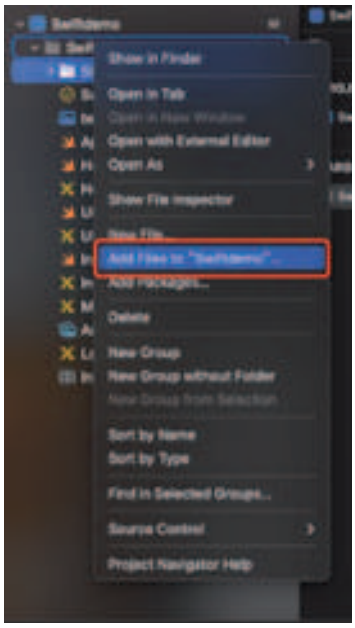
Drawing Text 18

Draw barcode/qrcode 19

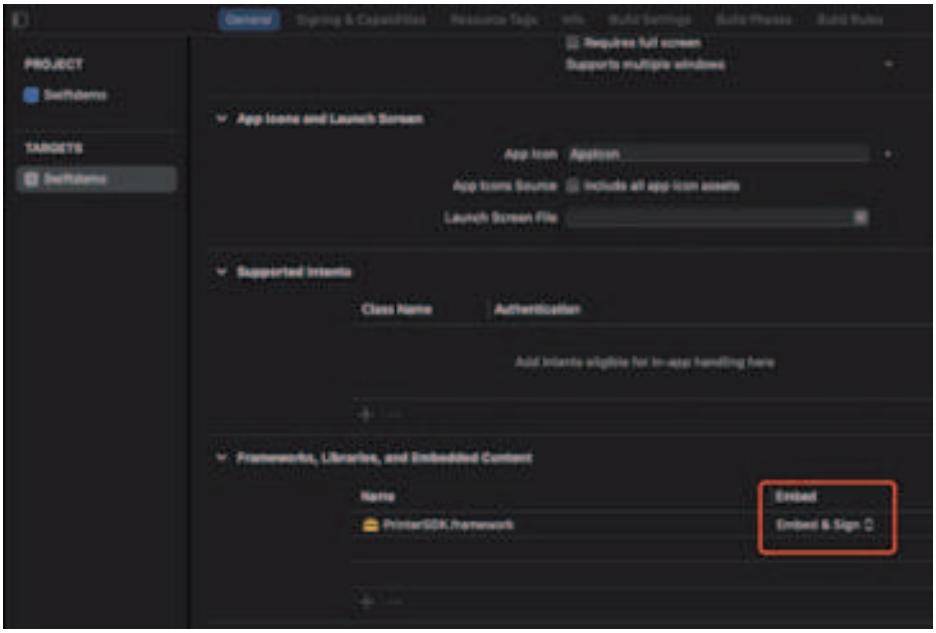
Draw Picture 20

IMPORT SDK TO PROJECT

- Right click on the folder on the left side of the project and select “Add Files to xxx” to add the SDK.



- Embed & Sign SDK.



CONNECT THE SDK TO THE PRINTER

(using Swift as an example)
SDK is a bridge for communication between apps and printers on the same network. You can have the SDK automatically search for printers and connect them, or you can manually enter the IP address on the printer self-test page to connect.

- Automatically search and connect, refer to the “UDPSearchDeviceVC. swift” file.

Import SDK

import PrinterSDK

Search for printers

```
NetworkPrinter.sharedInstance().delegate = self
let err = NetworkPrinter.sharedInstance().startSearch()
if (err == nil)
        {
        self.infoLB.text = “start search...\nEnsure that your phone’s Wi-Fi and printer are on the same network”
        }
        else{
        self.infoLB.text = “start search fail:\(err!.localizedDescription)”
        }
```

Search for printer callback and connect the printer in the callback

```
extension UDPSearchDeviceVC:NetworkPrinterProtocol
	{
	func didStartSearch()
		{
		print(“start”)
		}
	func didEndSearch()
		{
		print(“stop”)
		}
	func didFoundWiFiDevice(_ netPrinter: NetPrinterInfo)
		{
		print(“netPrinter=\(netPrinter),\(netPrinter.ip),\(Int(netPrinter.port)!)”)

		if (self.isConncecting == false)
			{
			self.isConncecting = true
			NetworkPrinter.sharedInstance().stopSearch()

			self.printerInter.address = netPrinter.ip
			self.printerInter.port = Int(netPrinter.port)!
			self.printerInter.printerCmdtype = PrinterCmdType.init(rawValue: 0) // 0 = PrinterCmdESC
		//	self.printerInter1.isNeedCallBack
			self.printer.printerPi = self.printerInter
			self.printer.open()
			}
		}
	}
```

Adding notification of successful or failed printer connection

NotificationCenter.default.addObserver(self, selector: #selector(handle(noti:)), name: NSNotification.Name.PrinterConnected, object: nil)
NotificationCenter.default.addObserver(self, selector: #selector(handle(noti:)), name: NSNotification.Name.PrinterDisconnected, object: nil)

Process printer connection success and failure notifications

```
//MARK: Notice
@objc func handle(noti:Notification)
        {
        if noti.name == NSNotification.Name.PrinterConnected
                {
                if noti.object is ObserverObj
                        {
                        et obj = noti.object as! ObserverObj
                        if obj.msgobj is PrinterInterface
                                {
                                print(“=========”)
                                print(“noti.object.Msgobj=”,object_getClassName(obj.msgobj))
                                print(“noti.object.Msgvalue=”,obj.msgvalue)
                                print(“printer connect success”)
                                print(“=========”)

                                self.infoLB.text = “connect success\nip:”+self.printerInter.address+”\nport:\(self.printerInter.
                                port)”

                                self.printImgBtn.isEnabled = true
                                self.printTextBtn.isEnabled = true
                                self.printBarcodeBtn.isEnabled = true
                                self.printAllBtn.isEnabled = true

                                self.searchAndConnectBtn.setTitle(“Disconnect”, for: .normal)
                                }
                        }

                }else if noti.name == NSNotification.Name.PrinterDisconnected
                {
                if noti.object is ObserverObj
                        {
                        let obj = noti.object as! ObserverObj
                                if obj.msgobj is PrinterInterface
                                {
                                print(“=========”)
                                print(“noti.object.Msgobj=”,object_getClassName(obj.msgobj))
                                print(“noti.object.Msgvalue=”,obj.msgvalue)
                                print(“printer Disconnected”)
                                print(“=========”)

                                self.infoLB.text = “Disconnected success”
                                self.printImgBtn.isEnabled = false
                                self.printTextBtn.isEnabled = false
                                self.printBarcodeBtn.isEnabled = false
                                self.printAllBtn.isEnabled = false

                                self.searchAndConnectBtn.setTitle(“Start Search And Connect”, for: .normal)
                                }
                        }
                }
        }
```

SDK sends data

(using Swift as an example)

```
if printer.isOpen
        {
        let sendData = cmd.getCmd()!
        let state = printer.write(sendData as Data)
        print("send img = \(sendData as NSData )")
        if (state == false)
                {
                print("Have a Job!!!!")
                }else{
                print("Printing...")
                }
        }
```

SDK disconnects the printer

(using Swift as an example)
self.printer.close()

MANUALLY INPUT THE PRINTER IP AND CONNECT IT

refer to the “InputIPVC. swift” file

Import SDK

import PrinterSDK

- Enter the printer IP and connect.

```
self.printerInter.address = self.ipTF.text
self.printerInter.port = Int(self.portTF.text ?? “9100”)!
self.printerInter.printerCmdtype = PrinterCmdType.init(rawValue: 0) // 0 = PrinterCmdESC
self.printerInter.isNeedCallBack = true;
self.printerInter.callbackwhenSendProgressUpdate = { (ipackNo, ipackcnt, message) in
print(“Progress-->>>\(ipackNo) / \(ipackcnt), message = \(String(describing: message))”)
}
self.printerInter.callbackwhenSendSuccess = { (ipackNo, ipackcnt, message) in
print(“Success-->>>\(ipackNo) / \(ipackcnt), message = \(String(describing: message))”)
}
self.printerInter.callbackwhenSendFailure = { (ipackNo, ipackcnt, message) in
print(“Failure-->>>\(ipackNo) / \(ipackcnt), message = \(String(describing: message))”)
}

self.printer.printerPi = self.printerInter
self.printer.open()
```

Adding notification of successful or failed printer connection

```
NotificationCenter.default.addObserver(self, selector: #selector(handle(noti:)),
name:NSNotification.Name.PrinterConnected, object: nil)
NotificationCenter.default.addObserver(self, selector: #selector(handle(noti:)),
name: NSNotification.Name.PrinterDisconnected, object: nil)
```

Process printer connection success and failure notifications

```
//MARK: Notice
@objc func handle(noti:Notification)
        {
        if noti.name == NSNotification.Name.PrinterConnected
                {
                if noti.object is ObserverObj
                        {
                        let obj = noti.object as! ObserverObj
                        if obj.msgobj is PrinterInterface
                                {
                                print("=========")
                                print("noti.object.Msgobj=",object_getClassName(obj.msgobj))
                                print("noti.object.Msgvalue=",obj.msgvalue)
                                print("printer connect success")
                                print("=========")


                                self.printImgBtn.isEnabled = true
                                self.printTextBtn.isEnabled = true
                                self.printBarcodeBtn.isEnabled = true
                                self.printAllBtn.isEnabled = true

                                self.connectBtn.setTitle("Disconnect", for: .normal)
                                }
                        }

                        }else if noti.name == NSNotification.Name.PrinterDisconnected
                                {

                                if noti.object is ObserverObj {
                                let obj = noti.object as! ObserverObj
                                if obj.msgobj is PrinterInterface {
                                print("=========")
                                print("noti.object.Msgobj=",object_getClassName(obj.msgobj))
                                print("noti.object.Msgvalue=",obj.msgvalue)
                                print("printer Disconnected")
                                print("=========")

                                self.printImgBtn.isEnabled = false
                                self.printTextBtn.isEnabled = false
                                self.printBarcodeBtn.isEnabled = false
                                self.printAllBtn.isEnabled = false

                                self.connectBtn.setTitle("connect", for: .normal)
                                }
                        }
                }
        }
```

DRAW PRINT CONTENT

(using Swift as an example).

Drawing Text

```
let printString = “Print OK!”

let textSetting = printer.textSets!
//[textst setEscFonttype:ESCFontType_FontA];
textSetting.isBold = Set_Enabled
textSetting.isItalic = Set_Enabled
textSetting.isTimes_Wide = Set_Enabled
textSetting.isTimes_Heigh = Set_DisEnable
textSetting.isTimes4_Wide = Set_DisEnable
textSetting.alignmode = Align_Left
textSetting.isUnderline = Set_DisEnable
textSetting.rotate = Rotate0


let cmd = ESCFactory.create()!
cmd.clear()
cmd.encodingType = EncodingType.init(1) //Encoding_UTF8

cmd.append(cmd.getHeaderCmd())
let escCmd:ESCCmd = cmd as! ESCCmd
cmd.append(escCmd.getSetAreaWidthCmd(80*8))

let textData = cmd.getTextCmd(self.printer.textSets, text:printString)
cmd.append(textData)
//cmd.append(cmd.getLFCRCmd())
//cmd.append(cmd.getLFCRCmd())
cmd.append(escCmd.getSetAreaWidthCmd(80*8))
cmd.append(cmd.getLFCRCmd())
cmd.append(cmd.getCutPaperCmd(CutterMode_Full))
cmd.append(cmd.getPrintEnd())

//cmd.append(cmd.getBeepCmd(1, interval: 3))
```

Draw barcode/qrcode

```
var CodeError:PrinterCodeError = PrinterCodeOK;
let printQRString = “Print OK!”

        let barset = BarcodeSetting()
        barset.rotate = Rotate0
        barset.coord = CoordMake(10, 0, 3, 90)
        barset.alignmode = Align_Left;
        barset.hriPos = BarcodeHRIpos_Below;
        barset.hriFonttype = ESCFontType_FontA;

        let cmd = ESCFactory.create()!
        cmd.clear()

        let lblcommSetting = LabelCommonSetting()
        lblcommSetting.labelWidth = 72
        lblcommSetting.labelHeight = 30
        lblcommSetting.labelDriection = Direction_Forward;
        lblcommSetting.density = 6;
        self.printer.commonSetts = lblcommSetting;
        cmd.append(cmd.getHeaderCmd(lblcommSetting))

        let borcodeData = cmd.getBarCodeCmd(barset, codeType: BarcodeTypeQrcode, scode: printQRString, codeError:
        &CodeError)
        if (CodeError != PrinterCodeOK)
                {
                print(“error codeType or error code”)
                return;
                }
        cmd.append(borcodeData)

        //cmd.append(cmd.getLFCRCmd())
        //cmd.append(cmd.getLFCRCmd())
        cmd.append(cmd.getCutPaperCmd(CutterMode_Full))
        cmd.append(cmd.getPrintEnd())
```

Draw Picture

```
let cmd = ESCFactory.create()!
cmd.clear()
cmd.encodingType = EncodingType.init(1) //Encoding_UTF8

        let headerCmd = cmd.getHeaderCmd()
        cmd.append(headerCmd)
        let ilimitwidth = 72;   //72mm paper
        let bitmapSetting = printer.bitmapSetts!
        bitmapSetting.alignmode = Alignment.init(0) // Align_Left
        bitmapSetting.limitWidth = ilimitwidth * 8;   // 1mm = 8pix

        let image = UIImage.init(contentsOfFile: Bundle.main.path(forResource: “test_4”, ofType: “jpg”)!)
        let imageData = cmd.getBitMapCmd(bitmapSetting, image: image)
        cmd.append(imageData)
        cmd.append(cmd.getLFCmd())
        cmd.append(cmd.getCutPaperCmd(CutterMode_Full))
        cmd.append(cmd.getPrintEnd())
        //cmd.append(cmd.getOpenDrawerCmd(0, startTime: 50, endTime: 50))
```

Code : 0577200M000088 Rev. 1.00

CUSTOM S.p.A.
World Headquarters
Via Berettine, 2/B - 43010 Fontevivo, Parma ITALY
Tel. +39 0521 680111 - Fax +39 0521 610701
info@custom.biz - www.custom.biz

All rights reserved

www.custom.biz