NPrinterLib.framework(Ver 1.0.0.0)

- **Environmental Construction Procedure**
- **SDK Guide(iOS)**

Overview

About SDK Overview

API Reference

About API Syntax

Precaution

- Please contact us if there is anything to be corrected and missed.
- This contents may change without prior notice.
  Please contact us for the latest information.

- Please do not reprint, copy, duplicate or fabricate all this contents or partially without prior consent.

- Please be aware that we will not be responsible for the result arising from the usage.

- Please be aware well that we are not responsible for damage arising from incorrect usage and handling by user on contrary to this description, and for repair or modification by the third party.

About Signage

Following signage are used in this guide. Please understand the meaning of each signage before using the product.

Note	Vital observance in use is described. Incorrect handling against this indication may cause breakdown or operation failure of the product.
reference	Supplemental remarks and related matters are described.

Restrictions in use

Please use the product with attention to safety design of whole system by fail safety design Or redundant design to sustain reliability and safety of whole system in case of being used Applications which require for high reliability and safety in function and accuracy such as Equipment, disaster prevention and security device directly related to operation of airplane, train, vessel, auto-mobile etc.
This product is not designed for use of aerospace hardware, main communication equipment, atomic control device, medical instrument etc. which require extremely higher Reliability and safety. Please make a decision upon user's enough acknowledgement To adequateness of this product in such applications.

Table of Contents

Overview

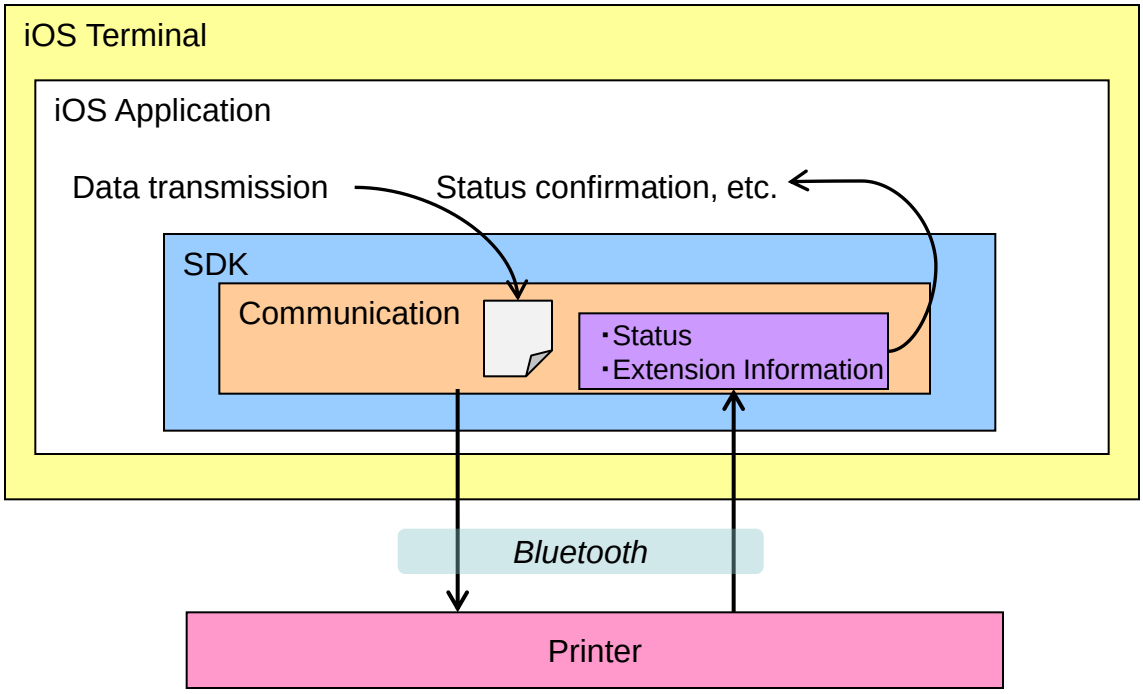
SDK enables application to have functions such as printing and monitoring printers.

“framework” is the file distributed as SDK.
Unzip the file since it is zipped when distribution of it.

Configuration NPrinterLib.framework

- Headers (Functions provided from SDK are arranged as header file.
- Resources
- Versions (Version Information）

System configuration when using SDK



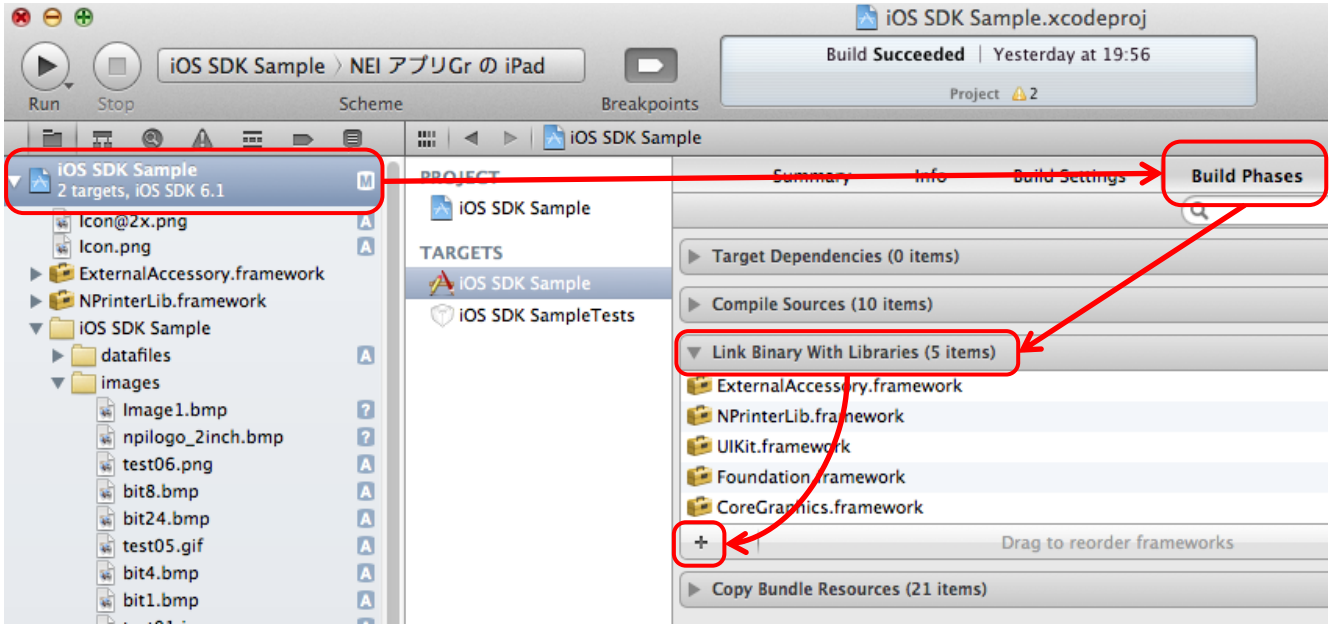
・Development Language
Objective-C

・iOS Compliant Version
iOS6, iOS7

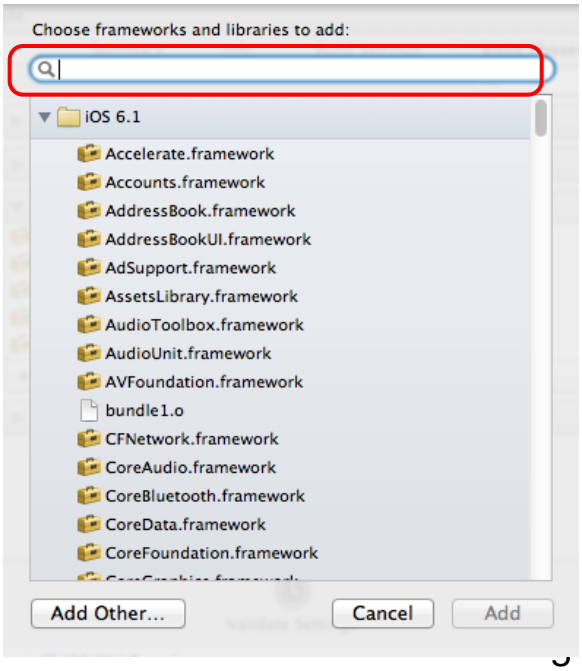
・Interface
Bluetooth(2.1+EDR)

Adding “ExternalAccessory.frameword” to the project

(1) Click your iOS application project
(2) Click “TARGETS” and then “Build Phases”
(3) Lay out “Link Binary With Libraries”
(4) After Laying out the contents, current usage framework is displayed.
Click “+” button at the bottom of this content.

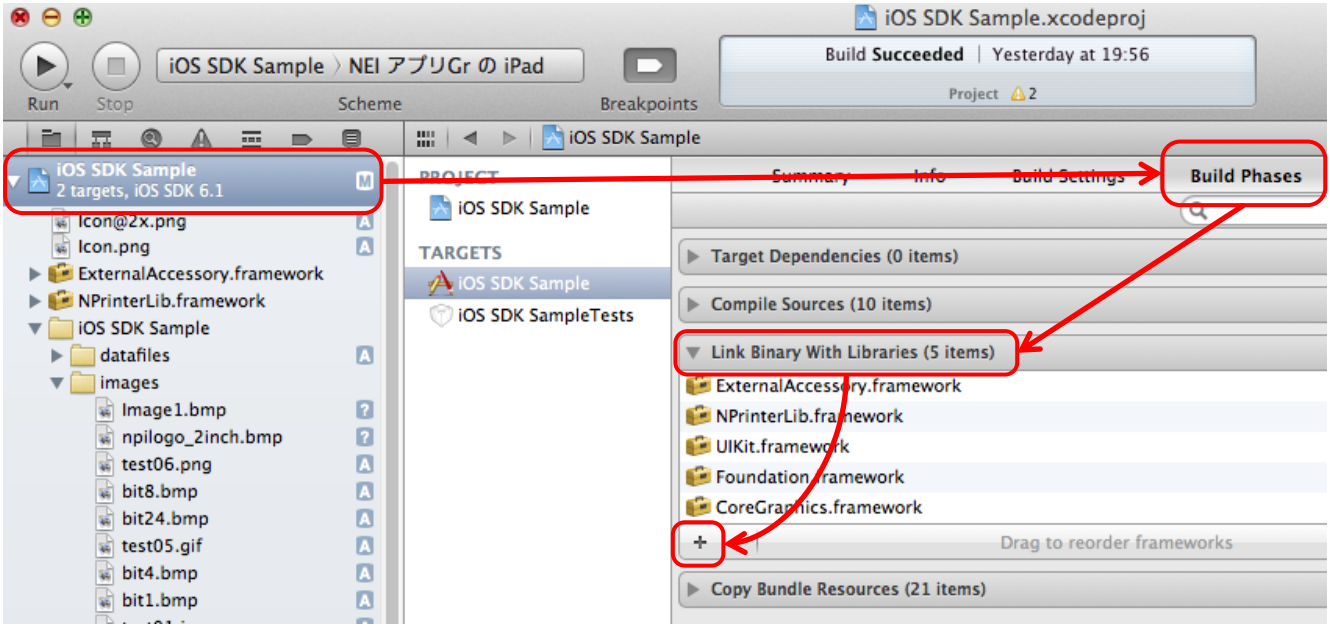


(5) When entering “ExternalAccessory” within the red box on the right picture, “ExternalAccessory.framework” is displayed. Select what you need and click “Add” button to add it to your project.



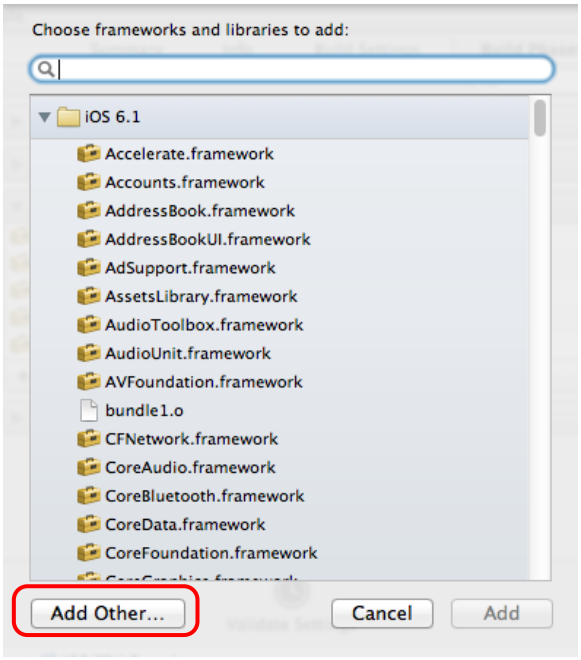
Adding “NPrinterLib.framework” to the project

(1) Click your iOS application project
(2) Click “TARGETS” and then “Build Phases”
(3) Lay out “Link Binary With Libraries”
(4) After Laying out the contents, current usage framework is displayed.
Click “+” button at the bottom of this content.

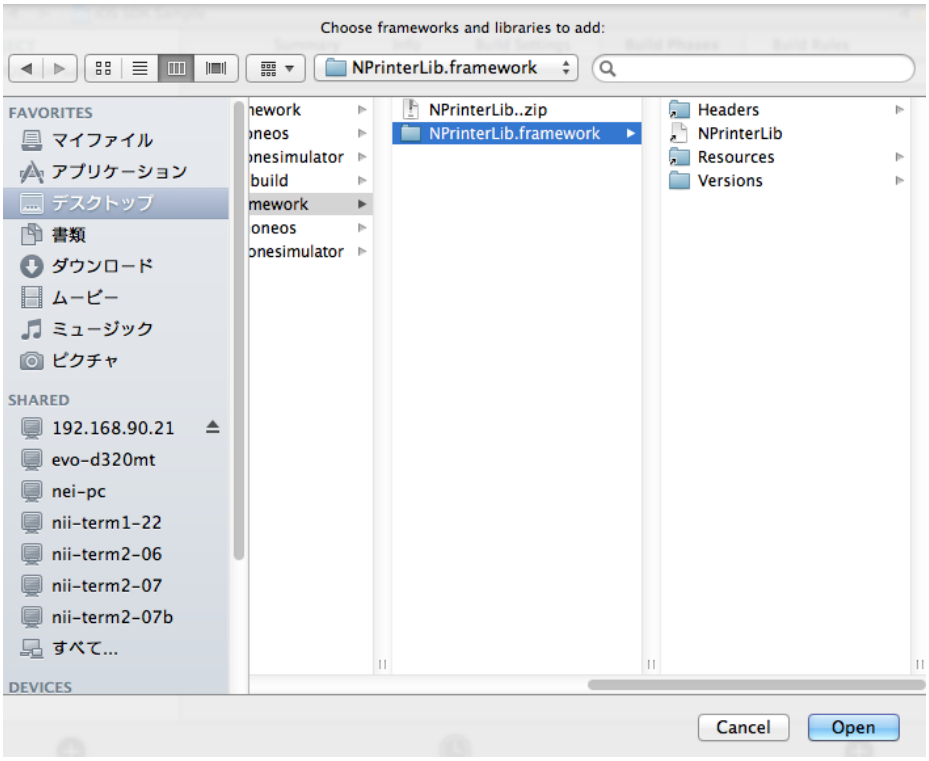


Continuation of Adding “NPrinterLib.framework” to the project

(5) Click “Add Other...” button.



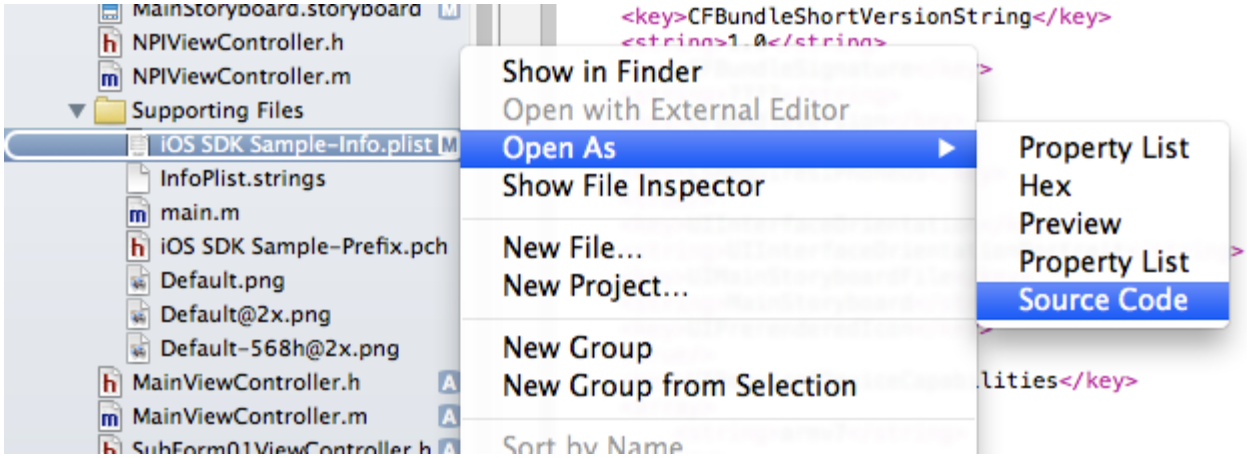
(6) Refer to “NPrinterLib.framework” where NprinterLib SDK is arranged, click “Open.”



Setting Info.plist

Adding "**UISupportedExternalAccessoryProtocols**" to Info.plist file of your iOS application is necessary for connection to the printer. For this addition, assign protocol which is available to communication with external accessory hardware.

(1) Open Info.plist file as "Source Code" as follows.



(2) Add code in a red box to the Info.plist.
Add "**UISupportedExternalAccessoryProtocols**" to <key> and "**com.fujitsu.fcl** " to <array>.

```
    <true/>
    <key>UIInterfaceOrientation</key>
    <string>UIInterfaceOrientationPortrait</string>
    <key>UIMainStoryboardFile</key>
    <string>MainStoryboard</string>
    <key>UIPrerenderedIcon</key>
    <true/>
    <key>UIRequiredDeviceCapabilities</key>
    <array>
        <string>armv7</string>
    </array>
    <key>UISupportedExternalAccessoryProtocols</key>
    <array>
        <string>com.fujitsu.fcl</string>
    </array>
    <key>UISupportedInterfaceOrientations</key>
    <array>
        <string>UIInterfaceOrientationLandscapeLeft</string>
        <string>UIInterfaceOrientationLandscapeRight</string>
        <string>UIInterfaceOrientationPortrait</string>
    </array>
</dict>
</plist>
```

About Bluetooth Connection

For this SDK, pairing the printer with iOS device is necessary before Bluetooth connection. Refer to manual of each device and conduct pairing.

About Sample Application

Connection of sample application and the printer is killed when this application runs in background with pressing home button. Invoke “NOpenPrinter” function for recovering connection with the printer.

About Home Directory

For this SDK, setting file and barcode generating image file are stored in following directory. The directory is gotten from "NPrinterLib SDK home directory: ***NSDocumentDirectory***" For this guide, directory where "npi" folder is made under the path above is desplayed as "**Home Directory.**" This folder is deleted at the same time of uninstalling application.

<How to get> "***strNpiDir***" is home directory (~/Documents/npi)

```
NSArray* arrPaths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory.,
                                                        NSUserDomainMask,
                                                        YES);

NSString* strHomeDir = [arrPaths objectAtIndex:0];
NSString* strNpiDir = [NSString stringWithFormat : @"%@/npi", strHomeDir];
```

About Setting File

These file are not changed manually but are edited/written by SDK function.

Printer information control file : /*[Home Directory]*/NPrinterInf
Log setting file : /*[Home Directory]*/NLogInf
Barcode information control file: /*[Home Directory]*/NBarcodeInf

About Barcode Image File

When Nbarcode function is invoked, barcode image file is made (refer to Nbarcode below) In case of printing, assign file name including path under home directory with "NImagePrintgF" function.

Example of image file path
~/Documents/npi/image/barcode1.png

About Log File

Usage of this SDK leads to that the following log is output by log setting.

Log file : /*[Home Directory]*/log/SDKLog and SDKLog_bk

About Barcode Setting

1. Array barcode information control file. (/*[Home Directory]* /NBarcodeInf)
   In case of using “Nbarcode” function in this SDK, making setting file is necessary.
   It is necessary that this file is set by customers with “Nbarcode1DSetting” function.

Note

Although barcode information control file can be edited directly in customers’ program, the performance is not guaranteed.

About barcode setting

2. How to describe barcode setting file(1D barcodes).
   Please set Barcode1～Barcode10 with ini file format under the following specifications. Also, please be sure to input correctly because barcode font name and each item distinguish large and small letter. All but “FIMENAME” are set by number.

```
[Barcode1]
TYPE=0
WIDTH=2
HEIGHT=100
HRI=1
HEXMODE=0
ROTATE=0
STARTBIT=0
STOPBIT=0
FILENAME=barcode01.bmp

[Barcode2]
TYPE=1
WIDTH=3
HEIGHT=150
HRI=1
HEXMODE=0
ROTATE=0
STARTBIT=0
STOPBIT=0
FILENAME=barcode02.bmp

        ...
        ...

[Barcode10]
```

← Barcode1～Barcode10 is assignable.(Barcode font name)

Use the following default value when there is not NBarcodeInf file.

TYPE	: 0	STARTBIT	: 0
WIDTH	: 2	STOPBIT	: 0
HEIGHT	: 100	FILENAME	: barcodeXX.bmp
HRI	: 0		(XX is between 01～10)
HEXMODE	: 0		
ROTATE	: 0		

TYPE
0:UPC-A
1:UPC-E
2:EAN13
3:EAN8
4:CODE39
5:ITF
6:CODABAR
7:CODE128
8:CODE93

WIDTH
Assign magnification of data width between 2～4.

HEIGHT
Assign data height by dots.

HRI
0:None
1:Top(Font A)
2:Bottom(Font A)
3:Top&Bottom(Font A)
4: Top(Font B)
5: Bottom(Font B)
6: Top&Bottom(Font B)

HEXMODE
0:Input data normally (0,1,2,･･･)
1:Input data hexadecimally **(0x30,0x31,･･･)**

ROTATE
0:No rotation
1:90° rotation
2:180° rotation
3:270° rotation

STARTBIT,STOPBIT (Only CODABAR setting is valid)
0:A
1:B
2:C
3:D

FILENAME
Assign image file name of barcode generated on [Internal storage]/npi/image.
(When blank is selected, file will not be generated)

About Barcode Setting

3. Explanation of barcode file setting (about HEXMODE)
   In case of entering barcode data (byte array of NBarcode 8th parameter)

- Normal entry (assigning HEXMODE = 0)
  In case of assigning 0x31, 0x32, 0x33, 0x34 (”1234”), the data is transmitted as-is.

- Hexadecimal entry (assigning HEXMODE = 0)
  In case of assigning 0x31, 0x32, 0x33, 0x34 (”1234”), the data is transmitted as ··· 0x12, 0x34.

4. Explanation of barcode file setting (about FILENAME)
   File name is assigned to “FILENAME” and image file is made under “**[Home Directory]/image**” when NBarcode makes success.
   Development pass cannot be changed. File name is assigned without any pass (assigned only file name).

About Log Output (Output Setting)

This SDK output log file called SDKLog under **[Home Directory]/log**. Log output is set by “NLogSetProperty” function. (Refer to function explanation page below)

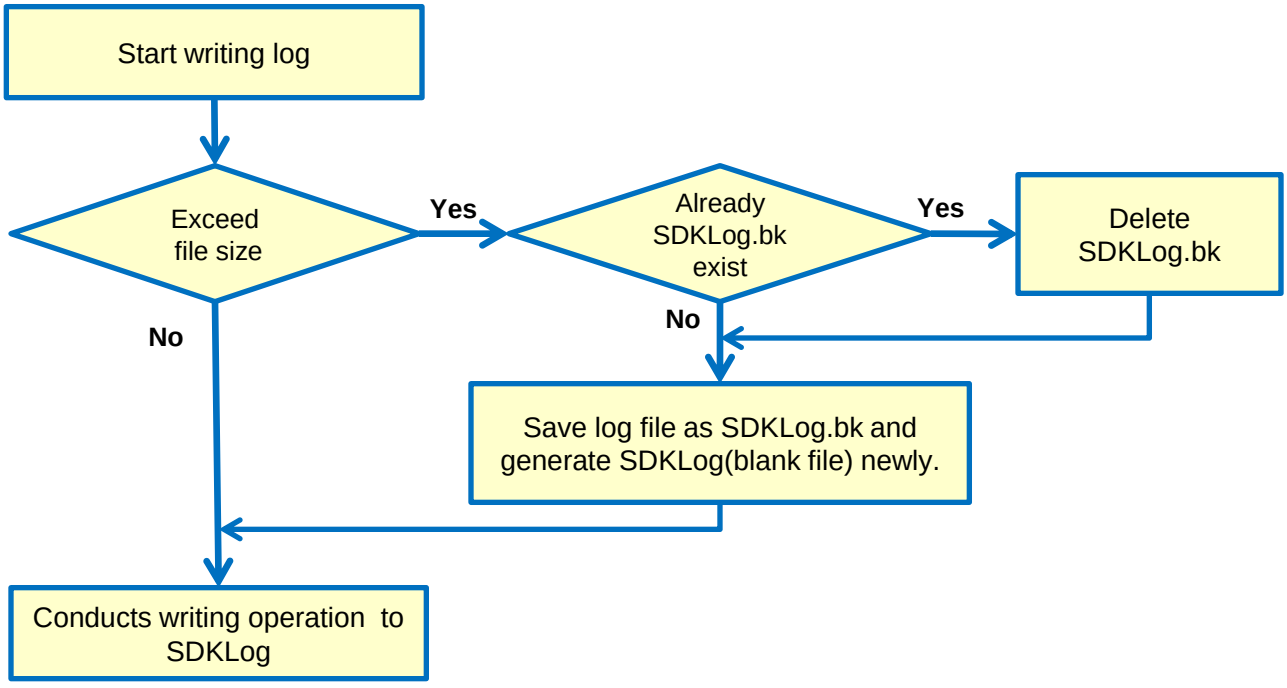
About log type（OUTPUT）
1 ERROR : Error
2 WARN : Warning
3 FUNC : Function call
4 IN : Port transmission data
5 OUT : Port receive data

* For Log type 4 (IN) and 5 (OUT), in case amount of data to transmit is large, speed of SDK performance will decrease. (log output spends time)
* In case of assigning large file size, log file might not be opened. Change file size depending on your device.
* When application install, setting is log type: 1,2 and log size: 1Mbytes. (outputs only ERROR and WARN)

About log output (Output file)

Also, file will be saved as past log(SDKLog.bk) and new blank log will be generated when exceeding the byte number assigned by setting file.

If there is already SDKLog.bk, it will be deleted.(Only one file for past log could be saved.)



【 Log output example】
2013/02/12 14:00:01.000 [FNC] xxxx.
2013/02/12 14:00:01.100 [WRN] xxxx.
2013/02/12 14:00:01.200 [ERR] xxxx.
2013/02/12 14:00:01.350 [FNC] xxxx.
2013/02/12 14:00:01.614 [I N] xxxx.
2013/02/12 14:00:02.100 [OUT] xxxx.
2013/02/12 14:00:03.040 [I N] xxxx.
2013/02/12 14:00:03.500 [FNC] xxxx.
2013/02/12 14:00:04.010 [FNC] xxxx.

Out put date by Year/Month/Day hour：min：sec.msec .
Display log type between brackets[].

1 ERR : Error
2 WRN : Warning
3 FNC : Function call
4 I N : Port receive data
5 OUT : Port transmission data

How to use function

- First of all, import header file
  #import <NPrinterLib/NPrinterLib.h>

Generate NPrinterLib class and operate by calling each function.

In case of using NPrinterLib in several monitor, constract program with sharing the class generated at the beginning with each monitor. (DO NOT generate the class in each monitor)

* Generate NPrinterLib class ONCE for ONE application.

Ex) Conduction full cut command (1B69)

```
int numJobID = 0;
NPrinterLib* objLib = [[NPrinterLib alloc] init];

~ Generating printer information control file (invoking “NEnumPrinters” *2) ~
~ Printer open (invoking “NOpenPrinter”) ~

[objLib NPrint : @”NP-xxxx” : @”1B69” : 4 : & numJobID];

~ Printer close (invoking “NClosePrinter”) ~
```

*Refer to sample program for detail.

How to get version information

- The following information is defined in “NPrinterLib.framework” (#define)
  Version information: SDKVERSION
  Date of release : RELEASEDATE

Note

* 1 About sequential printing and batch printing

If either “Nprint, NDPrint,” “NImagePrint” and “NImagePrintF” is conducted as solo, data will be transmitted to printer in each time. Batch transmission of data to printer is available by NStartDoc and NEndDoc.

Use sequential printing and batch printing for each purpose.

* 2 In order generating printer information control file, invoke “NEnumPrinters” function with Bluetooth pairing. For printer name generated once, invoking “NEnumPrinters” function again is not necessary. (The first time only)

Types of API

The following functions are prepared

Application	A P I	Description
Getting Printer Information	NEnumPrinters	Getting the list of printer name
Getting Printer Name from ID	NGetPrinterFromID	Getting existent printer name from Bluetooth serial ID
Deleting Printer Name	NDeletePrinter	Deleting Printer Name (Selection or Batch)
Changing Printer Name	NRenamePrinter	Changing printer name controlled by API
Getting Printer Information	NGetPrinterInf	Getting information from printer name
Auto Printer Setting	NAutoOpen	* Non-iOS-compliant
Opening Printer	NOpenPrinter	Specifying printer name and open
Closing Printer	NClosePrinter	Closing printer already opened
Closing All Printers	NClosePrinters	Closing all printers already opened
Transmission of command and data	NPrint	Converting assigned hex character string data and transmitting to printer
Transmission of command and data	NDPrint	Transmitting assigned data to printer
Image Output	NImagePrint	Transmitting assigned device context to printer in raster bit image
Image Output	NImagePrintF	Transmit assigned file (bmp/jpg/png) to printer in raster bit image

Types of API

The following functions are prepared.

Application	A P I	Description
Getting Status	NGetStatus	Returning status acquired by assigned printer
Getting Extended information	NGetInformation	Getting information saved corresponding type ID
Resetting Printer	NResetPrinter	* Non-iOS-compliant
Managing Document	NStartDoc	Starting document
Managing Document	NEndDoc	Finishing document
Managing Document	NCancelDoc	Canceling document
Generating Barcode	NBarcode	Generating barcode image
Updating F/W	NFirmwareDL	Updating F/W corresponding by fwf file
Getting Log (array)	NLogGetLines	Getting log already written from array (NS Array)
Getting Log (character string)	NLogGetString	Getting log already written from character string (NSString)
Deleting Log	NLogDelete	Deleting log already written
Setting Log	NLogSetProperty	Setting log output types and log size
Confirmation of Barcode Setting	NBarcode1DGetData	Getting data already set from barcode setting file
Setting Barcode	NBarcode1DSetting	Editing barcode setting file (1D)

Class Name	NPrinterLib		
Contractor Argument Name	IN/OUT	Type	Explanation
Return Value			
Process Contents			

After generating this class, make a process with invoking each function.

At the time of generating the class, log setting file is read in.

If log setting file does not exist, log output is conducted with default setting (only ERROR/WARN log, Max. 1Mbytes).

Function Name		NEnumPrinters	
Argument name	IN/OUT	Type	Explanation
o_printers o_size	O O	NSMutableString* Int*	Printer name (enumerated with csv comma-delimited format) printer name byte size of o_ printer (NULL is assignable)
Return value	Int		
・Error(-Value), Normal finish(0) * Refer to error code table			

Process contents

• Generate printer information management file(NPrinterInf) under Home directory and save list of connectable printer name in argument “o_printers.”

Ex.) . PRT001, PRT002, AAA

Allocate name of “PRTxxx“（xxx is the value between 001～999）and return by enumerate with csv(comma- delimited) after detecting port of USB・Bluetooth.

Make sure to invoke this function at the first time use because printer open (NOpenPrinter) is unavailable at the stage that printer information management file is not generated. **Also, invoke function is required when printer addition**.

In case of using a printer name generated once, it is unnecessary to invoke this function again.

Also, printer list acquired by this function is also available by referring to printer information management file (NPrinterInf).

* Pairing with printer is required before calling this function when using Bluetooth.

* Once printer name is generated, it will not be deleted by release Bluetooth pairing. In order to delete, use “NDeletePrinter” file.

* Maximum 999 is available for xxx allocating printer name and exceeding this number become generation error. Deletion of unnecessary data by NDeletePrinter or rewriting printer name by NRenamePrinter will be required.

Function Name	NGetPrinterFromID		
Argument Name	IN/OUT	Type	Explanation
i_ID o_printer	I O	NSString* NSMutableString*	Bluetooth serial ID Printer Name
Return Value	int		
・Error(-Value), Normal finish(0) * Refer to error code table			
Process Contents			

- Printer name is stored in argument “o_printer” by assigning Bluetooth serial ID to argument “i_ID.”
  It is able to confirm Bluetooth serial ID by self-diagnostic print.

Note

It is necessary that printer name is generated in printer information control file (NPrinterInf) by “NEnumPrinter” function in advance.

Function Name	NDeletePrinter		
Argument Name	IN/OUT	Type	Explanation
i_prt	I	NSString*	Printer name to delete
Return Value	int		
・Error(-Value), Normal finish(0) * Refer to error code table			
Process Contents			

- This function is used for deleting printer name in printer information control file.
  Assign printer name to delete to argument “i_prt.”
  Also, printer information control file itself is deleted by assigning “Space.” (Deleting all printer name)

Function Name	NRenamePrinter		
Argument Name	IN/OUT	Type	Explanation
i_beforeprt i_afterprt	I I	NSString* NSString*	Printer name to change Changed printer name
Return Value	int		
・Error(-Value), Normal finish(0) * Refer to error code table			
Process Contents			

- Use when rewriting printer name in printer information management file.

  Although the format starting with PRT~ like “PRT001” is formed as default, printer name is changeable by this function.

  Also, printer name should be in the range of 50 half size alphanumeric characters and the following characters are prohibited to use.

Prohibited characters to use (Including half-size space) []¥/:?*”<>|’, .

In addition, error is returned in case of opening printer to change its name.

Use this function with closing connection to this printer.

Function Name	NGetPrinterInf		
Argument Name	IN/OUT	Type	Explanation
i_prt o_ports o_size	I O O	NSString* NSMutableString* int*	Printer Name Printer name (enumerated with csv comma-delimited format) Printer name byte size of o_ printer (NULL is assignable)
Return Value	int		
・Error(-Value), Normal finish(0) * Refer to error code table			

Process Contents

The following information will be searched in printer information management file by printer name assigned by argument and stored.

- Connection type
  USB :1
  Bluetooth : 2

- Connection information (USB : non-iOS-compliant
  Bluetooth: the name/serial ID of paired device)

Ex) 1. 10511000
2. AAA-XX123/0003F3598C1A

The information acquiring by “o_ports” is also available by referring to printer information management file(NPrinterInf).

Note

In case of using the same pairing name among a number of printers with Bluetooth interface, please discern by serial ID. 12 digits character string following “/” of pairing name is a serial ID. (red color character string above)

iOS API Reference

Function Name	NAutoOpen		
Argument Name	IN/OUT	Type	Explanation
i_flg	I	int	Automatically printer open flug
Return Value	bool		

・Constantly false (non-compliant)

Process Contents

・This function is non-compliant for iOS

Function Name	NOpenPrinter		
Argument Name	IN/OUT	Type	Explanation
i_prt i_statusFlg	I I	NSString* bool	Printer name to open Non-iOS-compliant (unused)
Return Value	int		
・Error(-Value), Normal finish(0) * Refer to error code table			

Process Contents

- Conducts printer open process
  Argument “i_prt” is assigned printer name gotten from “NEnumPrints.”
  Argument “i_statusFlg” exist for compatibility of Windows/Android and this argument is not used for iOS.

Note

In case of running application in the background, Bluetooth connection is killed.
When opening application again, conduct open process by this function again.

* The sample application conduct close process demonstratively with invoking “NClosePrinters” function at the time of running the application in the background .

Function Name	NClosePrinter		
Argument Name	IN/OUT	Type	Explanation
i_prt	I	NSString*	Printer name
Return Value	int		

・Error(-Value), Normal finish(0) * Refer to error code table

Process Contents

- Close process is conducted by this function.

Function Name	NClosePrinters		
Argument Name	IN/OUT	Type	Explanation
Return Value	int		

・Normal finish(0)

Process Contents

- Close all printers already opened.

Note

This function returns only “normal finish” as return value.

Even if printers has already closed, “0” (normal finish) is returned. If returning error value is required, use “NClosePrinter” (closing only one printer) function.

Function Name	NPrint		
Argument Name	IN/OUT	Type	Explanation
i_prt i_dat i_size io_jobid	I I I IO	NSString* NSString* int int*	Printer name Transmitting data (hexadecimal string) Amount of output bytes Print job ID (NULL is assignable)
Return Value	int		
・Error(-Value), Normal finish(0) * Refer to error code table			

Process Contents

- Transmit assigned hex character string data to printer.

Also, the following three patterns will be proceeded as special data if they are detected during data analysis.

1. Character string in “(Double-quotation)
   ⇒ Convert as character string (”ABC“ ⇒ 0x41,0x42, 0x43）
2. File name in <> （angle brackets）*including path
   ⇒Out put file name (Binary data).
3. Character string which has‘(Single-quotation) at the head
   ⇒ It will be proceeded as comment(not to be output).

Function Name	NDPrint		
Argument Name	IN/OUT	Type	Explanation
i_prt i_dat i_size io_jobid	I I I IO	NSString* NSData* int int*	Printer name Transmitting data (Hexadecimal string) Amount of output bytes Print job ID (NULL is assignable)
Return value	int		
・Error(-Value), Normal finish(0) * Refer to error code table			
Process Contents			

- Assigned data is transmitted to printer.(No conversion)

Function Name	NImagePrint		
Argument Name	IN/OUT	Type	Explanation
i_prt	I	NSString*	Printer name
i_bmp	I	UIImage*	Image area
i_width	I	int	Width
i_height	I	int	Height
i_putType	I	int	Output method 0x00: Raster format line unit 0x01: Raster format block unit 0x02: Raster format gray scale display of block unit 0x10: Bit image format
io_jobid	IO	int*	Print job ID (null is assignable)
Return Value	int		
・Error(-Value), Normal finish(0) * Refer to error code table			

Process Contents

- Transmit assigned “UIIImage” class to printer.
  It is impossible to assign width・height beyond “UIImage” class size

Note

- It takes time to out put Raster format gray scale display of block unit in comparison with other format. Moreover, character corruption may occur in case of using models which are not applied this function..

Function Name	NImagePrintF		
Argument Name	IN/OUT	Type	Explanation
i_prt	I	NSString*	Printer name
i_bmp	I	NSString*	Image file name (including path)
i_putType	I	int	Output method 0x00: Raster format line unit 0x01: Raster format block unit 0x02: Raster format gray scale display of block unit 0x10: Bit image format
io_jobid	IO	int*	Print job ID (null is assignable)
Return Value	int		
・Error(-Value), Normal finish(0) * Refer to error code table			
Process Contents			

- Transmit assigned “jpg/png/bmp (bit depth is only 24)” file to printer.

Note

- It takes time to out put Raster format gray scale display of block unit in comparison with other format. Moreover, character corruption may occur in case of using models which are not applied this function..

- DO NOT print other “bmp” files other than bit depth 24. Otherwise, print operation is incorrect.

Function Name	NGetStatus		
Argument Name	IN/OUT	Type	Explanation
i_prt o_status	I O	NSString* int*	Printer name Status
Return Value	int		
・Error(-Value), Normal finish(0) * Refer to error code table			
Process Contents			

- Return acquired status on assigned printer.

* Refer to specification of corresponding printer for returned value

Function Name	NGetInformation		
Argument Name	IN/OUT	Type	Explanation
i_prt	I	NSString*	Printer name
i_id	I	int	ID type
o_dat	O	NSMutableData*	Storage area of extended information
o_time	O	UInt64*	Update flag (acquire the time of system with msec) (NULL is assignable)
Return Value	int		
・Error(-Value), Normal finish(0) * Refer to error code table			
Process Contents			

- Acquire the information saved in ID for corresponding type of extended information.
  * It is required to transmit request for necessary information from higher application to printer in advance.
    (There are some cases which do not require the requests such as extended status, transfer completion and request for information of print completion, etc.)
  * Refer to specification of corresponding printers regarding return value.

iOS API Reference

Function Name	NResetPrinter		
Argument Name	IN/OUT	Type	Explanation
i_prt	I	NSString*	Printer name
Return Value	int		
・Error(-Value) * Refer to error code table			
Return Value			

- This function is not compliant for iOS

Function Name	NStartDoc		
Argument Name	IN/OUT	Type	Explanation
i_prt o_jobid	I O	NSString* int*	Printer name Print job ID
Return Value	int		
・Error(-Value), Normal finish(0) * Refer to error code table			
Process Contents			

- Start document.
  Buffer data of “NPrint,” “NDPrint,” “NImagePrint” and “NImagePrintF” in memory after issuing “NStartDoc.”

  Buffered data can be output to printer by calling “NEndDoc.”
  Also, clear buffered data by calling “NCancelDoc.”

  One document is available by one printer.

Function Name	NEndDoc		
Argument Name	IN/OUT	Type	Explanation
i_prt	I	NSString*	Printer name
Return Value	int		
・Error(-Value), Normal finish(0) * Refer to error code table			
Process Contents			

- Finish document.
  Output buffered data to printer after calling “NStartDoc.”
  Not to proceed when buffered data do not exist.

Function Name	NCancelDoc		
Argument Name	IN/OUT	Type	Explanation
i_prt	I	NSString*	Printer name
Return Value	int		
・Error(-Value), Normal finish(0) * Refer to error code table			
Process Contents			

- Document will be cancelled.
  Not to proceed when buffered data do not exist.

Function Name	NBarcode		
Argument Name	IN/OUT	Type	Explanation
i_prt	I	NSString*	Printer name
i_fontName	I	NSString*	Barcode font name
i_bmp	IO	UIImage*	Barcode drawing area
			Left
i_x	I	int	Top
i_y	I	int	Width
i_width	I	int	Height
i_height	I	int	Barcode data
i_dat	I	char*	Data size
i_size	I	int	
Return Value	int		
·Error(-Value), Normal finish(0) * Refer to error code table			

Process Contents

- Barcode specified in printer's barcode font is drawn on "i_bmp."
  Although 1D barcode is available to iOS SDK, this SDK does not support 2D barcode.

  Assign font name which is used for barcode setting file (NBarcodeInf) as font name of second argument. (Barcode 1 ~ Barcode 10)

  Refer to "**About Barcode Setting**" in this guide for "NBarcodeInf."

Note

- Barcode data deleting excessive part is generated without error even if generated barcode data exceed barcode drawing area. In this case, adjust data size to set all data in the size since it is impossible to read generated barcode.

Function Name	NBarcode
Process Contents	Specified image figure of argument

UIImage class area (Third argument the part in gray box)

* Set barcode width・height・with or without HRI character, etc. **in red box** on barcode setting file (NBarcodeInf).

* Barcode with red frame will rotate by rotate setting of barcode. Please set area with width and height after rotation since the white part in the above figure will not rotate

Function Name	NFirmwareDL		
Argument Name	IN/OUT	Type	Explanation
i_prt	I	NSString*	Printer name
i_file	I	NSString*	fwf file name (NULL is assignable)
i_errflg	I	int	Error check 0x00: Invalid(Forcible transmission) 0x01: Valid
io_chksum	IO	short*	Check sum of fwf file (null is assignable, imperative when firm ware check)
io_jobid	IO	int*	Print job ID(null is assignable)
Return Value	int		
・Error(-Value), Normal finish(0) * Refer to error code table			
Process Contents			

- This function is not compliant for iOS.
  Conduct process by Windows/Android SDK.

Function Name	NLogGetLines		
Argument Name	IN/OUT	Type	Explanation
i_view	I	NSString*	Log display flag 0: Present log (SDKLog) 1: Past log (SDKLog_bk) 2: Both present and past log
o_lines	O	NSMutableArray*	Output log array
Return Value	int		
・Error(-Value), Normal finish(+, file size) * Refer to error code table (another page)			

Process Contents

- Contents of log file output by SDK are stored in argument “o_lines” (NSMutableArray clacc) as character array by lines.
  When log file is empty, nothing is set in “o_lines.”
  Assign log which is acquired from first argument.

  Although “-” value is returned to return value in case of error, file size (byte) is returned in case of normal finish.

  When “log display flag: 2” is set in first argument, byte value which is total of present log and past log is returned.

Function Name	NLogGetString		
Argument Name	IN/OUT	Type	Explanation
i_view	I	NSString*	Log display flag 0: Present log (SDKLog) 1: Past log (SDKLog_bk) 2: Both present and past log
o_str	O	NSMutableString*	Output log
Return Value	int		
・Error(-Value), Normal finish(+, file size) * Refer to error code table (another page)			

Process Contents

- Contents of log file output by SDK is stored in argument “o_str” (NSMutableString class).
  When log file is empty, nothing is set in “o_str.”
  Assign log which is acquired from first argument “i_view.”

  Although “-” value is returned to return value in case of error, file size (byte) is returned in case of normal finish.

  When “log display flag: 2” is set in first argument, byte value which is total of present log and past log is returned.

Function Name	NLogDelete		
Argument Name	IN/OUT	Type	Explanation
i_view	I	NSString*	Log display flag 0: Present log (SDKLog) 1: Past log (SDKLog_bk) 2: Both present and past log
Return Value	int		
・Error(-Value), Normal finish(0) * Refer to error code table			
Process Contents			

- Log file output by SDK is deleted
  Assign log to delete in first argument “i_view.”

Function Name	NLogSetProperty		
Argument Name	IN/OUT	Type	Explanation
i_LogType i_LogSize	I I	NSString* int	Output log type Output log maximum size
Return Value	int		

・Error(-Value), Normal finish(0) * Refer to error code table

Process Contents

- Conducts log file output by SDK setting.
  Assign log to output in first argument “i_LogType.”
  Refer to “～” for log type.

  Ex) Default 1, 2: Only ERROR and WARN are output
  1, 2 ← Set log type to output by comma separated value.

  Maximum size of log file is set in second argument “i_LogSize.”
  When log file exceed this size, file which is be written is renamed to backup log and write it to log file which is generate anew

Function Name	NBarcode1DGetData		
Argument Nama	IN/OUT	Type	Explanation
i_fontName	I	int	Font name (Barcode1 ~ Barcode10)
o_type	O	int*	Barcode Type
o_width	O	int*	Data width ratio
o_height	O	int*	Barcode height
o_HRI	O	int*	HRI character existence or non-existence
o_hexMode	O	int*	Assigning HEXMODE
o_rotate	O	int*	Barcode rotation type
o_startBit	O	int*	Assigning start bit
o_stopBit	O	int*	Assigning stop bit
o_fileName	O	NSMutableString*	Generated file name
Return Value	int		
·Error(-Value), Normal finish(0) * Refer to error code table			
Process Contents			

- This function reads in barcode information control file **(/[Home Directory]/NBarcodeInf**) and gets data of each barcode with font name as a key.

  Barcode information control file is edited by “NBarcode1DSetting.”

Function Name	NBarcode1DSetting		
Argument Name	IN/OUT	Type	Explanation
i_fontName	I	int	Font name (Barcode1 ~ Barcode10)
i_type	I	int*	Barcode Type
i_width	I	int*	Data width ratio
i_height	I	int*	Barcode height
i_HRI	I	int*	HRI character existence or non-existence
i_hexMode	I	int*	Assigning HEXMODE
i_rotate	I	int*	Barcode rotation type
i_startBit	I	int*	Assigning start bit
i_stopBit	I	int*	Assigning stop bit
i_fileName	I	NSMutableString *	Generated file name
Return Value	int		
·Error(-Value), Normal finish(0) * Refer to error code table			
Process Contents			

- This is a function which edits barcode information control file **(/[Home Directory]/NBarcodeInf**).

  Refer to "**2. How to describe barcode setting file**" in "**About Barcode Setting**" for explanation of set value.

  The range of set value is as follows.

i_type : 0 ~ 8
i_width : 2 ~ 4
i_height : 1 or more
i_HRI : 0 ~ 6
i_hexMode : 0, 1
i_rotate : 0 ~ 3
i_startBit : 0 ~ 3
i_stopBit : 0 ~ 3
i_fileName : ~ .png (png file only)

Error Code List（1/2）

N_SUCCESS	0	Normal finish
NERR_PRTOPEN	-2	Printer open error
NERR_OFFLINE	-5	Off line
NERR_PRTCLOSE	-6	Printer close error
NERR_FILEOPEN	-10	File open error
NERR_PRTOUTPUT	-13	Printer output error
NERR_PRTINPUT	-14	Printer data receiving error
NERR_PRTALREADYOPEN	-21	Printer already opened
NERR_NOHANDLE	-22	Printer not opened
NERR_NOTSUPPORTED	-40	Non-compliant
NERR_LOADFROMFILE	-50	Failed to read picture file
NERR_IMAGESIZE	-51	Incorrect image size
NERR_LOADBITMAP	-52	Failed to Read in picture data from UIImage class
NERR_DOCNOTSTARTED	-71	Not document start status
NERR_ALREADYSTARTDOC	-72	Already document start status
NERR_FWFFILE	-80	fwf file error
NERR_FWF_CHECKSUM	-81	Check sum which is entered by argument does not correspond to check sum which is gotten from printer.
NERR_FWDL_TIMEOUT	-82	Firmware download time-outs (Checking command of print start time-outs)
NERR_FWCHK_TIMEOUT	-83	Confirmation of firmware check sum time-outs
NERR_FWCHK_FOUNDERROR	-84	Error is detected from status confirmation when downloading firmware
NERR_ARGUMENT	-90	Argument incorrect error
NERR_ARGUMENT_01	-91	First argument is incorrect
NERR_ARGUMENT_02	-92	Second argument is incorrect
NERR_ARGUMENT_03	-93	Third argument is incorrect
NERR_ARGUMENT_04	-94	Forth argument is incorrect
NERR_ARGUMENT_05	-95	Fifth argument is incorrect
NERR_ARGUMENT_06	-96	Sixth argument is incorrect
NERR_ARGUMENT_07	-97	Seventh argument is incorrect
NERR_ARGUMENT_08	-98	Eighth argument is incorrect
NERR_ARGUMENT_09	-99	Ninth argument is incorrect

Error Code List(2/2)

NERR_PRTINFO_CREATE	-130	Failed to generate printer information file
NERR_PRTINFO_READ	-131	Failed to read in printer information file
NERR_PRTINFO_WRITE	-132	Failed to write printer information file
NERR_PRTNAME_ALLOC	-133	Failed to allot printer name
NERR_PRTRENAME_BEFORE	-134	Printer name before changed does not existent
NERR_PRTRENAME_AFTER	-135	printer name to change has already used
NERR_PRTINFO_GET	-136	Failed to get printer status
NERR_PRTINFO_ILLEGAL	-137	Printer information file is incorrect
NERR_PRTINFO_DELETE	-138	Failed to delete printer name
NERR_PRTINFO_NOTFOUND	-139	Printer name does not existent
NERR_DEVICE_NOTSUPPORT	-150	Connection type is not supported
NERR_BCDSETTINGFILE	-160	Barcode setting file is incorrect
NERR_CREATEBCDFILE	-161	Failed to generate barcode file
NERR_CREATEBCDDATA	-162	Failed to generate barcode data
NERR_GETBCDDATA	-163	Failed to get data from barcode setting file
NERR_LOGCREATE	-170	Failed to generate log file
NERR_LOGDELETE	-171	Failed to delete log file
NERR_LOGRENAME	-172	Failed to rename log file
NERR_LOGOPEN	-173	Failed to open log file
NERR_LOGWRITE	-174	Failed to write log file
NERR_LOGREAD	-175	Failed to read in log file

Extended information

Type 1 : 4 byte (Fixed) : Update flag(4 byte) <Extended status>*1Byte：7～0,2Byte:15～8,*
3Byte：23～16,4Byte：31～24
Type 2 : 32 byte (delimiter) : Update flag(4 byte) <Model name>
Type 3 : 8 byte(Fixed) : Update flag (4 byte) <F/W version>
Type 4 : 8 byte(Fixed) : Update flag (4 byte) <Boot version>
Type 5 : 4 byte(Fixed) : Update flag (4 byte) <Reserve>
Type 6 : 4 byte(Fixed) : Update flag (4 byte) <Dot line number of current head>
Type 7 : 4 byte(Fixed) : Update flag (4 byte) <Dot line to drive>
Type 8 : 4 byte(Fixed) : Update flag (4 byte) <Cut times>
Type 9 : 16 byte(Fixed) : Update flag (4 byte) <User maintenance counter:
Current dot line number,
Dot line number to drive,
Cut times,
backup>
Type 10 : 16 byte (Fixed) : Update flag (4 byte)<Reserve>
Type 11 : 64 byte (delimiter) : Update flag (4 byte)
Type 12 : 32 byte (delimiter) : Update flag (4 byte)
Type 13 : 32 byte (Fixed) : Update flag (4 byte) <NV registration status>
Type 14 : 32 byte (Fixed) : Update flag (4 byte) <Reserve>
Type 15 : 16 byte (Fixed) : Update flag (4 byte)
Type 16 : 16 byte (Fixed) : Update flag (4 byte)
Type 17 : 16 byte (Fixed) : Update flag (4 byte)
Type 18 : 16 byte (Fixed) : Update flag (4 byte)
Type 19 : 8 byte (Fixed) : Update flag (4 byte)<Notice of print completion: Arbitrary ID and finish status will be described at proceeding finish command by assigning print start/finish command.>
Type 20 : 8 byte (Fixed) : Update flag (4 byte) <Reserve>
Type 21 : 8 byte (Fixed) : Update flag (4 byte)
Type 22 : 8 byte (Fixed) : Update flag (4 byte)
Type 23 : 8 byte (Fixed) : update flag (4 byte)
Type 24 : 4 byte (Fixed) : Update flag (4 byte)<Reserve>
Type 25 : 4 byte (Fixed) : Update flag (4 byte)<Notice of transfer completion: Transferred job ID>
Type 26 : 4 byte (Fixed) : Update flag (4 byte)<Reserve>
Type 27 : 4 byte (Fixed) : Update flag (4 byte)<Reserve>
Type 28 : 2 byte (Fixed) : Update flag (4 byte)<F/W check sum>
Type 29 : 2 byte (Fixed) : Update flag (4 byte)
Type 30 : 2 byte (Fixed) : Update flag (4 byte)
Type 31 : 2 byte (Fixed) : Update flag (4 byte) <Information of communication status:
USB: 0x0000 fixed COM：1st byte CTS Second byte DSR
*Timestamp to acquire final signal status to update flag>

*There are no validity about acquired information which do not have the function with printer except 25, 31.

* Described contents do not mean that they are available with all of printers.