BIXOLON

API Reference Guide

Linux POS/Mobile SDK

Ver. 2.18

http://www.bixolon.com

Table of Contents

1. Manual guide 5
1-1 Supported Kernel & Platform, O/S 5
1-2 Supported Interfaces 6
1-3 Supported Printers 7

2. Property 9
2-1 Character Set (int R/W) 9
2-2 International Character Set (int R/W) 11
2-3 State (int R) 13

3. Method 14
3-1 Connect To Printer 14
3-2 Disconnect Printer 16
3-3 Initialize Printer 17
3-4 Line Feed 18
3-5 Set Left Margin 19
3-6 Set Upside Down 20
3-7 Partial Cut 21
3-8 Open Cash Drawer 22
3-9 Print Text 23
3-10 Print Text W 26
3-11 Print Barcode 29
3-12 Direct IO 35
3-13 Print Image 37
3-14 Download NV Image 39
3-15 Print NV Image 40
3-16 Remove All NV Image 41
3-17 Remove NV Image 42
3-18 Get NV Image Key Code 43
3-19 Set Auto Status Check 44
3-20 Get Status 45
3-21 Select Mode 46
3-22 Next Print Pos 47
3-23 Auto Calibration 48
3-24 Select Page Mode 49
3-25 Print Data In PM 50
3-26 Set Print Area In PM 51
3-27 Set Print Direction In PM 53
3-28 Set Vertical Position In PM 54
3-29 Set Horizontal Position In PM 56
3-30 Read Start MSR 58
3-31 Read Cancel MSR 59
3-32 Read MSR Data 60
3-33 Scr Power Up 62
3-34 Scr Power Down 64

3-35 Scr Operation Mode 65
3-36 Scr Exchange APDU 67
3-37 Scr Check Status 69
3-38 Scr Select Card 70
3-39 Get Battery Status 72
3-40 Set Connection Timeout 73
3-41 Paper Eject 74
3-42 Get Presenter Status 75

Copyright

© BIXOLON Co., Ltd. All rights reserved.

This user manual and all property of the product are protected under copyright law.
It is strictly prohibited to copy, store, and transmit the whole or any part of the manual and any property of the product without the prior written approval of BIXOLON Co., Ltd.
The information contained herein is designed only for use with this BIXOLON product. BIXOLON is not responsible for any direct or indirect damages, arising from or related to use of this information.

• The BIXOLON logo is the registered trademark of BIXOLON Co., Ltd.
• All other brand or product names are trademarks of their respective companies or organizations.

BIXOLON Co., Ltd. maintains ongoing efforts to enhance and upgrade the functions and quality of all our products.
In the following, product specifications and/or user manual content may be changed without prior notice.

Caution

Some semiconductor devices are easily damaged by static electricity. You should turn the printer “OFF”, before you connect or remove the cables on the rear side, in order to guard the printer against the static electricity. If the printer is damaged by the static electricity, you should turn the printer “OFF”.

1. Manual guide

This SDK manual describes the contents of the library required to develop Linux OS application programs. Root privilege is required to use this library.

1-1 Supported Kernel & Platform, O/S

• Kernel
  - Kernel 2.6.32 or higher

• Platform
  - Linux 32bit / 64bit
  - Raspberry PI

• O/S
  - openSUSE 11.3 32bit / 64bit
  - Red Hat Enterprise Linux 7.3 64bit
  - CentOS 6.6 32bit / 64bit
  - Ubuntu 10.04 LTS 32bit / 64bit

1-2 Supported Interfaces

• USB, Serial, Parallel, Bluetooth, Ethernet, WLAN

1-3 Supported Printers

Model	DPI	Max Printable Width
SRP-E300	180 dpi	512 dots
SRP-E302	203 dpi	576 dots
SRP-Q300	180 dpi	512 dots
SRP-Q302	203 dpi	576 dots
SRP-QE300	180 dpi	512 dots
SRP-QE302	203 dpi	576 dots
SRP-Q200	203 dpi	432 dots
SRP-S300	203 dpi	576 dots
SRP-380	180 dpi	512 dots
SRP-382	203 dpi	576 dots
SRP-383	300 dpi	864 dots
SRP-380II	180 dpi	512 dots
SRP-382II	203 dpi	576 dots
SRP-330II	180 dpi	512 dots
SRP-332II	203 dpi	576 dots
SRP-330III	180 dpi	512 dots
SRP-332III	203 dpi	576 dots
SRP-F310II	180 dpi	512 dots
SRP-F312II	203 dpi	576 dots
SRP-F313II	203 dpi	640 dots
SRP-350III	180 dpi	512 dots
SRP-352III	203 dpi	576 dots
SRP-350plusIII	180 dpi	512 dots
SRP-352plusIII	203 dpi	576 dots
SRP-350V	180 dpi	512 dots
SRP-352V	203 dpi	576 dots
SRP-350plusV	180 dpi	512 dots
SRP-352plusV	203 dpi	576 dots
SRP-340II	180 dpi	512 dots
SRP-342II	203 dpi	576 dots
BK3-2	203 dpi	432 dots
BK3-3	203 dpi	576 dots
BK5-3	203 dpi	576 dots
STP-103III	203 dpi	384 dots
SPP-R200II	203 dpi	384 dots
SPP-R200III	203 dpi	384 dots
SPP-R210	203 dpi	384 dots
SPP-C200	203 dpi	384 dots
SPP-C300	203 dpi	576 dots

SPP-R300	203 dpi	576 dots
SPP-R310	203 dpi	576 dots
SPP-R400	203 dpi	832 dots
SPP-R410	203 dpi	832 dots
SPP-R220	203 dpi	384 dots
SPP-100II	203 dpi	384 dots
SMB690	203 dpi	384 dots
SRP-B300	203 dpi	576 dots (42CPL: 546 dots)
SRP-S320	203 dpi	576 dots
SRP-S3000	203 dpi	576 dots
G30	203 dpi	576 dots
SRP-Q300II	203 dpi	576 dots
SRP-S300II	203dpi	576dots
SRP-B300II	180dpi	512dots
SRP-380plus	180 dpi	512 dots
SRP-382plus	203 dpi	576 dots
BK3R-3	203 dpi	576 dots
BK5R-3	203 dpi	576 dots
BK5-4	203 dpi	576 dots
SRP-G300	203 dpi	576 dots

2. Property

The constants used by the library are declared in bxlConst.c.
The development environment is based on C.

2-1 Character Set (int R/W)

• This is the property for defining the printer's code page and set to CS_PC437 by default. The values can be set and read using Set Char Set() and Get Char Set().

The following code pages can be used:

Constant	Value	Description
CS_PC437	0	Code page PC437
CS_KATAKANA	1	Katakana
CS_PC850	2	Code page PC850
CS_PC860	3	Code page PC860
CS_PC863	4	Code page PC863
CS_PC865	5	Code page PC865
CS_WPC1252	16	Code page WPC1252
CS_PC866	17	Code page PC866
CS_PC852	18	Code page PC852
CS_PC858	19	Code page PC858
CS_PC864	22	Code page PC864
CS_THAI42	23	Code page THAI42
CS_WPC1253	24	Code page WPC1253
CS_WPC1254	25	Code page WPC1254
CS_WPC1257	26	Code page WPC1257
CS_FARSI	27	Code page FARSI
CS_WPC1251	28	Code page WPC1251
CS_PC737	29	Code page PC737
CS_PC775	30	Code page PC775
CS_THAI14	31	Code page THAI14
CS_PC862	33	Code page PC862
CS_PC855	36	Code page PC855
CS_PC857	37	Code page PC857
CS_PC928	38	Code page PC928
CS_THAI16	39	Code page THAI16
CS_WPC1256	40	Code page WPC1256
CS_PC1258	41	Code page PC1258
CS_KHMER	42	Code page KHMER
CS_PC1250	47	Code page PC1250
CS_USER	255	User set page

* Example

```
ConnectToPrinter(port);

……

SetCharSet(CS_PC850);

……

int32 nCharSet;

nCharSet = GetCharSet();

……
```

2-2 International Character Set (int R/W)

• This is the property for defining International character Set and set to ICS_USA by default.
The values can be set or read using Set International Char() and Get International Char().

Character Set settings may need to be verified in the following cases
1. When character strings other than the one you tried to print are printed
2. When a broken string is printed in the same form as hieroglyphic characters
3. When characters are printed in the form of '?' (question mark)

The following International character Set can be used:

Constant	Value	Description
ICS_USA	0	USA code
ICS_FRANCE	1	FRANCE code
ICS_GERMANY	2	GERMANY code
ICS_UK	3	UK code
ICS_DENMARK1	4	DENMARK1 code
ICS_SWEDEN	5	SWEDEN code
ICS_ITALY	6	ITALY code
ICS_SPAIN	7	SPAIN code
ICS_JAPAN	8	JAPAN code
ICS_NORWAY	9	NORWAY code
ICS_DENMARK2	10	DENMARK 2 code
ICS_SPAIN2	11	SPAIN 2 code
ICS_LATIN	12	LATIN AMERICA code
ICS_KOREA	13	KOREA code

* Example

```
ConnectToPrinter(port)

……

SetInternationalChar(ICS_SPAIN);

……

int32 nCharSet;

nCharSet = GetInternationalChar();

……
```

2-3 State (int R)

• This is the property that sets the printer status. It is read only and calls Get Status() to read the printer status. The status value can be set in duplicate and each value can be checked using bitwise operation.

These are the printer status values:

Constant	Value	Description
STS_NORMAL	0	The printer status is normal.
STS_PAPEREMPTY	1	There is no paper.
STS_COVEROPEN	2	The paper cover is open.
STS_NEAREND	4	The paper is low (Near end).

* Example

```
ConnectToprinter(port)
……

SetAutoStatusCheck(true);

int status = GetStatus();


if ((status & STS_PAPEREMPTY) == STS_PAPEREMPTY)
     ……
if ((status & STS_COVEROPEN) == STS_COVEROPEN)
     ……

…….
```

3. Method

The functions provided by Linux SDK are declared in Bxl Pos API.h.
The development environment is based on C.

3-1 Connect To Printer

• Set the connection for communication with the printer.
int ConnectToPrinter(const char *port)

[Parameters]

* const char *port
[in] Interface to be connected to the printer

Interface	Input Data	Example
USB	USB:	Connect To Printer(“USB:”)
Serial	serial:(baudrate) /dev/ttySX:(baudrate)	Connect To Printer(“serial:115200”) Connect To Printer(“/dev/ttyS0:115200”)
Parallel	parallel /dev/lpX	Connect To Printer(“parallel”) Connect To Printer(“/dev/lp0”)
Bluetooth	Device MAC address /dev/ttySX:(baudrate)	Connect To Printer(“7d:f0:7d:e4:e0:78”) Connect To Printer(“/dev/ttyS0:115200”)
Ethernet, Wifi	IP address, port no.	Connect To Printer(“192.168.0.10:9100”)

[Return Values]

Constant	Value	Description
SUCCESS	0	The operation is successful.
PORT_OPEN_ERROR	-99	The communication port cannot be opened.
NO_CONNECTED_PRINTER	-100	The printer is not connected.
NO_BIXOLON_PRINTER	-101	It is not a BIXOLON printer.

* Example

```
int ret;

// USB
ret = ConnectToPrinter("USB:");

// Serial
ret = ConnectToPrinter("serial:115200");

// Parallel
ret = ConnectToPrinter("parallel");

// bluetooth
ret = ConnectToPrinter("7d:f0:7d:e4:e0:78");

// Ethernet or WiFi
ret = ConnectToPrinter("192.168.0.10:9100");
.....
```

3-2 Disconnect Printer

• Disconnect the printer.
  void Disconnect Printer();

[Parameters]
  None

[Return Values]
  None

* Example

```
ConnectToPrinter(.....);

......

DisconnectPrinter();
```

3-3 Initialize Printer

• Cancel the existing settings and initialize.
int Initialize Printer();

[Parameters]
None

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
WRITE_ERROR	-105	Data transmission failed.

* Example

```
.....

InitializePrinter();

.....
```

3-4 Line Feed

• Line feed to the amount of the integer value set as a factor.
int Line Feed(const unsigned int line Num);

[Parameters]

* const unsigned int line Num
[in] Send the number of line feeding lines in an integer value as a factor.

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
WRITE_ERROR	-105	Data transmission failed.
MEM_ALLOC_ERROR	-120	The allocation of internal memory failed.

* Example

```
ConnectToPrinter(.....);

.....

LineFeed(5);

.....
```

3-5 Set Left Margin

• Set the left margin.
int Set Left Margin(long margin);

[Parameters]

* long margin
[in] Margin size: 0 ~ Max printable width (dots)

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
WRITE_ERROR	-105	Data transmission failed.
BAD_ARGUMENT	-117	Incorrect values have been entered.

* Example

```
ConnectToPrinter(.....);

.....

SetLeftMargin(10);

.....
```

3-6 Set Upside Down

• Set the upside-down function.
int Set Upsize Down(bool upside Down);

[Parameters]

* bool upside Down
[in] Enable/disable the upside-down function.

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
WRITE_ERROR	-105	Data transmission failed.

* Example

```
ConnectToPrinter(.....);

......

SetUpsideDown(true);

......
```

3-7 Partial Cut

• Enable the partial cut function.
int Partial Cut();

[Parameters]
None

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
WRITE_ERROR	-105	Data transmission failed.
NOT_SUPPORT	-124	The function is not supported.

* Example

```
ConnectToPrinter(.....);

.....

PartialCut();

.....
```

3-8 Open Cash Drawer

• Open the cash drawer.
int Open Cash Drawer(unsigned int milli Sec);

[Parameters]

* unsigned int milli Sec
[in] Set the open signal length (0 ~ 255).

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
WRITE_ERROR	-105	Data transmission failed.
NOT_SUPPORT	-124	The function is not supported.

* Example

```
ConnectToPrinter(.....);

.....

OpenCashDrawer(100);

.....
```

3-9 Print Text

• Print texts.

int Print Text(const char* text, const int alignment, const unsigned int attribute,
const unsigned int text Size);

[Parameters]

* const char* text

[in] String with null as a terminator. Send the text data to print.

* const int alignment

[in] Set the text alignment.

Constant	Value	Description
ALIGNMENT_LEFT	0	Align to the left
ALIGNMENT_CENTER	1	Align to the center
ALIGNMENT_RIGHT	2	Align to the right

* const unsigned int attribute

[in] Set the text attributes. The following values can be applied in duplicate.

Constant	Value	Description
ATTR_FONTTYPE_A	0	Set to Font A. (default)
ATTR_FONTTYPE_B	1	Set to Font B
ATTR_FONTTYPE_C	2	Set to Font C
ATTR_BOLD	4	Add Bold
ATTR_UNDERLINE_1	8	Add 1-dot underline
ATTR_UNDERLINE_2	16	Add 2-dot underline
ATTR_REVERSE	32	Add the reverse text attribute.

* const unsigned int text Size
  [in] Set the text size. The width and height scale can be used in duplicate.

Constant	Value	Description
TS_WIDTH_0	0x00	Set the width scale to x1
TS_WIDTH_1	0x10	Set the width scale to x2
TS_WIDTH_2	0x20	Set the width scale to x3
TS_WIDTH_3	0x30	Set the width scale to x4
TS_WIDTH_4	0x40	Set the width scale to x5
TS_WIDTH_5	0x50	Set the width scale to x6
TS_WIDTH_6	0x60	Set the width scale to x7
TS_WIDTH_7	0x70	Set the width scale to x8

Constant	Value	Description
TS_HEIGHT_0	0x00	Set the height scale to x1
TS_HEIGHT_1	0x01	Set the height scale to x2
TS_HEIGHT_2	0x02	Set the height scale to x3
TS_HEIGHT_3	0x03	Set the height scale to x4
TS_HEIGHT_4	0x04	Set the height scale to x5
TS_HEIGHT_5	0x05	Set the height scale to x6
TS_HEIGHT_6	0x06	Set the height scale to x7
TS_HEIGHT_7	0x07	Set the height scale to x8

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
STATUS_ERROR	-103	Not ready to print.
WRITE_ERROR	-105	Data transmission failed.

* Example

```
ConnectToPrinter(.....);

......

PrintText("Bixolon Linux SDK Text.\n", ALIGNMENT_LEFT,
            ATTR_FONTTYPE_A, TS_HEIGHT_0 | TS_WIDTH_0);
......
```

3-10 Print Text W

• Print 2Bytes texts. (Use UTF-8)

int Print Text W(const char* text, const int alignment, const unsigned int attribute, const unsigned int text Size, const unsigned int code Page);

[Parameters]

* const char* text

[in] String with null as a terminator. Send the text data to print.

* const int alignment

[in] Set the text alignment.

Constant	Value	Description
ALIGNMENT_LEFT	0	Align to the left
ALIGNMENT_CENTER	1	Align to the center
ALIGNMENT_RIGHT	2	Align to the right

* const unsigned int attribute

[in] Set the text attributes. The following values can be applied in duplicate.

Constant	Value	Description
ATTR_FONTTYPE_A	0	Set to Font A. Print with the default device font.
ATTR_FONTTYPE_B	1	Set to Font B
ATTR_FONTTYPE_C	2	Set to Font C
ATTR_BOLD	4	Add Bold
ATTR_UNDERLINE_1	8	Add 1-dot underline
ATTR_UNDERLINE_2	16	Add 2-dot underline
ATTR_REVERSE	32	Add the reverse text attribute.

* const unsigned int text Size
  [in] Set the text size. The width and height scale can be used in duplicate.

Constant	Value	Description
TS_WIDTH_0	0x00	Set the width scale to x1
TS_WIDTH_1	0x10	Set the width scale to x2
TS_WIDTH_2	0x20	Set the width scale to x3
TS_WIDTH_3	0x30	Set the width scale to x4
TS_WIDTH_4	0x40	Set the width scale to x5
TS_WIDTH_5	0x50	Set the width scale to x6
TS_WIDTH_6	0x60	Set the width scale to x7
TS_WIDTH_7	0x70	Set the width scale to x8

Constant	Value	Description
TS_HEIGHT_0	0x00	Set the height scale to x1
TS_HEIGHT_1	0x01	Set the height scale to x2
TS_HEIGHT_2	0x02	Set the height scale to x3
TS_HEIGHT_3	0x03	Set the height scale to x4
TS_HEIGHT_4	0x04	Set the height scale to x5
TS_HEIGHT_5	0x05	Set the height scale to x6
TS_HEIGHT_6	0x06	Set the height scale to x7
TS_HEIGHT_7	0x07	Set the height scale to x8

* const unsigned int code Page
[in] Set the encoding type for character strings.

Constant	Value	Description
ENCODING _ASCII	0	ASCII
ENCODING _EUCKR	1	Korean (EUC-KR)
ENCODING _CP949	2	Korean (CP949)
ENCODING _EUCCN	3	Chinese (EUC-CN)
ENCODING _GB18030	4	Chinese (GB18030)
ENCODING _BIG5	5	Chinese (BIG5)
ENCODING _CP950	6	Chinese (CP950)
ENCODING _EUCJP	7	Japanese (EUC-JP)
ENCODING _CP932	8	Japanese (CP932)
ENCODING _CP874	9	Thai (CP874)

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
STATUS_ERROR	-103	Not ready to print.
WRITE_ERROR	-105	Data transmission failed.

* Example

```
ConnectToPrinter(.....);

......

PrintTextW("Korean Printing Test.\n", ALIGNMENT_LEFT,
ATTR_FONTTYPE_A, TS_HEIGHT_0 | TS_WIDTH_0, ENCODING _CP949);
......
```

3-11 Print Barcode

• Print 1- and 2-dimensional barcodes.

int Print Barcode(const int barcode Type, const char* barcode Data,
const barcode Info_s* barcode Info);

[Parameters]

* const int barcode Type

[in] Set the barcode type. Defined in bxl Const.c.

Constant	Value	Description
BARCODE_UPCA	0	Print UPC-A barcode.
BARCODE_UPCE	1	Print UPC-E barcode.
BARCODE_EAN13 BARCODE_JAN13	3 5	Print JAN-13(EAN-13) barcode.
BARCODE_EAN8 BARCODE_JAN8	2 4	Print JAN-8(EAN-8) barcode.
BARCODE_ITF	6	Print ITF barcode.
BARCODE_CODABAR	7	Print CODABAR barcode.
BARCODE_CODE39	8	Print CODE39 barcode.
BARCODE_CODE93	9	Print CODE93 barcode.
BARCODE_CODE128	10	Print CODE128 barcode.
BARCODE_PDF417	11	Print PDF417 barcode.
BARCODE_QRCODE	12	Print QR CODE barcode.
BARCODE_DATAMATRIX	13	Print DATAMATRIX barcode.
BARCODE_MAXICODE	14	Print MAXICODE barcode.
BARCODE_AZTEC	15	Print AZTEC barcode.
BARCODE_GS1	16	Print GS1 barcode.

* const char* barcode Data

[in] Send the barcode data to print.

* const barcode Info_s* barcode Info
  [in] Structure for storing barcode attributes

```
Struct _barcode Info
{
    unsigned int mode;
    unsigned int height;
    unsigned int width;
    unsigned int alignment;
    unsigned int text Position;
    unsigned int attribute;
};
```

unsigned int mode
[in] Send the model value when printing QR Code.

Constant	Value	Description
BARCODE_QR_MODEL1	1	Model 1
BARCODE_QR_MODEL2	2	Model 2

Send the mode value when printing Maxi Code.

Constant	Value	Description
BARCODE_MAXI_MODE2	1	Mode 2
BARCODE_MAXI_MODE3	2	Mode 3
BARCODE_MAXI_MODE4	3	Mode 4

Send the mode value when printing AZTEC.

Constant	Value	Description
BARCODE_AZTEC_DATAMODE	1	Data mode
BARCODE_AZTEC_GS1MODE	2	gs1 mode
BARCODE_AZTEC_UNICODE	3	Unicode mode

the mode value when printing GS1.

Constant	Value	Description
BARCODE_GS1_RSS14	1	GS1 Data Bar Omnidirectional
BARCODE_GS1_RSS14TRUNCATED	2	GS1 Data Bar Truncated
BARCODE_GS1_RSS14STACKED	3	GS1 Data Bar Stacked
BARCODE_GS1_RSS14STACKEDOMNI	4	GS1 Data Bar Stacked Omnidirectional
BARCODE_GS1_UPCA	5	UPC-A
BARCODE_GS1_UPCE	6	UPC-E
BARCODE_GS1_EAN13	7	EAN-13
BARCODE_GS1_EAN8	8	EAN-8
BARCODE_GS1_EAN128AB	9	UCC/EAN-128&CC-A/B
BARCODE_GS1_EAN128C	10	UCC/EAN-128&CC-C

unsigned int height
[in] Set the barcode height (1~255). If the barcode is larger than the paper size, the barcode may not be printed. 2-dimensional barcodes are not subject to this value.

unsigned int width
[in] Set the barcode width (2~7). If the barcode is larger than the paper size, the barcode may not be printed. 2-dimensional barcodes are not subject to this value.

unsigned int alignment
[in] Set the barcode alignment.

Constant	Value	Description
ALIGNMENT_LEFT	0	Align to the left
ALIGNMENT_CENTER	1	Align to the center
ALIGNMENT_RIGHT	2	Align to the right

unsigned int text Position
[in] Set the barcode data printing position.
2-dimensional barcode has only BXL_BC_TEXT_NONE.

Constant	Value	Description
BARCODE _TEXT_NONE	0	No barcode data is printed.
BARCODE _TEXT_ABOVE	1	The barcode data is printed above the barcode.
BARCODE _TEXT_BELOW	2	The barcode data is printed below the barcode.

unsigned int attribute
[in] Set the separator height of 2D and 1D barcodes when printing GS1 (1 or 2).

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
WRITE_ERROR	-105	Data transmission failed.
WRONG_BARCODE_TYPE	-115	The barcode type is not supported.
WRONG_BC_DATA_ERROR	-116	The barcode data is incorrect.

* Example

```
ConnectToPrinter(.....);

char* barcodeData = “123456789012”;
barcodeInfo_s barcodeInfo;

.....

barcodeInfo.height = 50;
barcodeInfo.width = 2;
barcodeInfo.textPosition = BARCODE_TEXT_BELOW;
PrintBarcode(BARCDE_UPCA, barcodeData, &barcodeInfo);

barcodeInfo.height = 50;
barcodeInfo.width = 2;
barcodeInfo.textPosition = BARCODE_TEXT_BELOW;
PrintBarcode(BARCDE_UPCE, barcodeData, &barcodeInfo);

barcodeInfo.height = 50;
barcodeInfo.width = 2;
barcodeInfo.textPosition = BARCODE_TEXT_BELOW;
PrintBarcode(BARCDE_EAN13, barcodeData, &barcodeInfo);

barcodeInfo.height = 50;
barcodeInfo.width = 2;
barcodeInfo.textPosition = BARCODE_TEXT_BELOW;
PrintBarcode(BARCDE_JAN13, barcodeData, &barcodeInfo);

barcodeInfo.height = 50;
barcodeInfo.width = 2;
barcodeInfo.textPosition = BARCODE_TEXT_BELOW;
PrintBarcode(BARCDE_EAN8, “12345678”, &barcodeInfo);

barcodeInfo.height = 50;
barcodeInfo.width = 2;
barcodeInfo.textPosition = BARCODE_TEXT_BELOW;
PrintBarcode(BARCDE_JAN8, “12345678”, &barcodeInfo);

barcodeInfo.height = 50;
barcodeInfo.width = 2;
barcodeInfo.textPosition = BARCODE_TEXT_BELOW;
PrintBarcode(BARCDE_CODE39, barcodeData, &barcodeInfo);

barcodeInfo.height = 50;
barcodeInfo.width = 2;
barcodeInfo.textPosition = BARCODE_TEXT_BELOW;
PrintBarcode(BARCDE_CODE93, barcodeData, &barcodeInfo);
```

```
barcodeInfo.height = 50;
barcodeInfo.width = 2;
barcodeInfo.textPosition = BARCODE_TEXT_BELOW;
PrintBarcode(BARCDE_CODE128, barcodeData, &barcodeInfo);

barcodeInfo.height = 50;
barcodeInfo.width = 2;
barcodeInfo.textPosition = BARCODE_TEXT_BELOW;
PrintBarcode(BARCDE_ITF, barcodeData, &barcodeInfo);

barcodeInfo.height = 50;
barcodeInfo.width = 2;
barcodeInfo.textPosition = BARCODE_TEXT_BELOW;
PrintBarcode(BARCDE_CODABAR, barcodeData, &barcodeInfo);

// *************** 2D barcode

barcodeInfo.height = 0;
barcodeInfo.width = 2;
barcodeInfo.attribute = 0;
PrintBarcode(BARCDE_PDF417, barcodeData, &barcodeInfo);

barcodeInfo.mode = BARCODE_QR_MODEL1;
barcodeInfo.height = 0;
barcodeInfo.width = 2;
barcodeInfo.attribute = 0;
PrintBarcode(BARCDE_QRCODE, barcodeData, &barcodeInfo);
barcodeInfo.mode = BARCODE_QR_MODEL2;
barcodeInfo.height = 0;
barcodeInfo.width = 2;
barcodeInfo.attribute = 0;
PrintBarcode(BARCDE_QRCODE, barcodeData, &barcodeInfo);

barcodeInfo.height = 0;
barcodeInfo.width = 2;
barcodeInfo.attribute = 0;
PrintBarcode(BARCDE_DATAMATRIX, barcodeData, &barcodeInfo);

barcodeInfo.mode = BARCODE_MAXI_MODE4;
barcodeInfo.height = 0;
barcodeInfo.width = 2;
barcodeInfo.attribute = 0;
PrintBarcode(BARCDE_MAXICODE, barcodeData, &barcodeInfo);
```

3-12 Direct IO

• Send and read the user-defined data.

int Direct IO(char* write Data, int write Len, char* read Data, int* read Len,
unsigned int m Timeout);

[Parameters]

* char* write Data,
[in] Data to be sent to the printer

* int write Len
[in] Size of data to be sent to the printer
write does not function if 0 is entered to write Data for NULL, write Len.

* char* read Data,
[in] Data to be read from the printer

* int* read Len
[in] Read the size of the data to be read by the caller.
read does not function if 0 is entered to read Data for NULL, read Len.

* unsigned int m Timeout
[in] Waiting time before reading the data. Even if no data has been read, it returns after the set amount of time. If set to 0, it waits until there is incoming data.

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
WRITE_ERROR	-105	Data transmission failed.
READ_TIMEOUT	-127	It timed out before reading the data.

* Example

```
char cmd[3] = { 0x10, 0x04, 0x01};
char readBuf[20] = {0x00,};
int readLen;

ConnectToPrinter(.....);

DirectIO(cmd, sizeof(cmd), readBuf, &readLen, 0);

.....
```

3-13 Print Image

• Print image files.

int Print Image(const char *image Path, const int width, const bool compress,
const unsigned int alignment);

[Parameters]

* const char *image Path
[in] String for the image file path. JPG, BMP and GIF are supported.

* const int width
[in] Set the width of the print image.

* const bool compress
[in] Set whether to compress RLE image.

* const unsigned int alignment
[in] Set the image alignment.

Constant	Value	Description
ALIGNMENT_LEFT	0	Align to the left
ALIGNMENT_CENTER	1	Align to the center
ALIGNMENT_RIGHT	2	Align to the right

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
IMAGE_OPEN_ERROR	-118	The image file cannot be opened.
MEM_ALLOC_ERROR	-120	The allocation of internal memory failed.

* Example

```
ConnectToPrinter(.....);

.....

PrintImage(filePath, 150, true, ALIGNMENT_CENTER);

.....
```

3-14 Download NV Image

• Save images to the non-volatile memory area of the printer.
int Download NV Image(const char *image Path, const unsigned int key Code);

[Parameters]

* const char *image Path
[in] String for the image file path. JPG, BMP and GIF are supported.

* const unsigned int key Code
[in] Address of the memory area where image is stored (0 ~ 255).

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
IMAGE_OPEN_ERROR	-118	The image file cannot be opened.
MEM_ALLOC_ERROR	-120	The allocation of internal memory failed.

* Example

```
ConnectToPrinter(.....);

.....

DownloadNVImage(filePath, 0x01);

.....
```

3-15 Print NV Image

• Print the images stored in the non-volatile memory area of the printer.
int Print NV Image(const unsigned int key Code);

[Parameters]

* const unsigned int key Code
[in] Address code of the image to be printed (0 ~ 255)

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
WRITE_ERROR	-105	Data transmission failed.

* Example

```
ConnectToPrinter(.....);

......

DownloadNVImage(filePath, 0x01);

......

PrintNVImage(0x01);
```

3-16 Remove All NV Image

• Remove all the images stored in the non-volatile memory area of the printer.
int Remove All NV Image();

[Parameters]

None

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
WRITE_ERROR	-105	Data transmission failed.
NOT_SUPPORT	-124	The function is not supported.

* Example

```
ConnectToPrinter(.....);
.....

RemoveAllNVImage();
```

3-17 Remove NV Image

• Remove all the images with the specified address stored in the non-volatile memory area of the printer.

int Remove NV Image(const unsigned int key Code);

[Parameters]

* const unsigned int key Code

[in] Address code of the image to be printed (0 ~ 255)

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
WRITE_ERROR	-105	Data transmission failed.
NOT_SUPPORT	-124	The function is not supported.

* Example

```
ConnectToPrinter(.....);

DownloadNVImage(filePath, 0x01);
.....

RemoveNVImage(0x01);
```

3-18 Get NV Image Key Code

• Read the address list of the images stored in the non-volatile memory area of the printer.
int Get NV Image Key Code(char *key Code List, unsigned int *list Len);

[Parameters]

* char *key Code List
[in, out] Buffer to save the list of image address

* unsinged int *list Len
[in, out] Length of key Code List

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
WRITE_ERROR	-105	Data transmission failed.
NV_NO_KEY	-121	No NV key is defined.
WRONG_RESPONSE	-122	Incorrect NV data response
NOT_SUPPORT	-124	The function is not supported.

* Example

```
char keyList[128] = {0x00, };
unsigned int listLen = 0;
int ret;

ConnectToPrinter(.....);

DownloadNVImage(filePath, 0x01);

.....

ret = GetNVImageKeyCode(keyList, &listLen);
```

3-19 Set Auto Status Check

• Enable/disable ASB mode to check the printer status (cover open, no paper).
int Set Auto Status Check(bool enable);

[Parameters]

* bool enable
[in] Enable/disable ASB mode.

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
WRITE_ERROR	-105	Data transmission failed.

* Example

```
int status = 0x00;

ConnectToPrinter(.....);

SetAutoStatusCheck(true);

......

status = GetStatus();
......

if ((status & BXL_STS_PAPEREMPTY) == BXL_STS_PAPEREMPTY)
    ......
```

3-20 Get Status

• Read the printer status (cover open, no paper).
int Get Status()

[Parameters]
None

[Return Values]

Constant	Value	Description
STS_NORMAL	0	The printer status is normal.
STS_PAPEREMPTY	1	There is no paper.
STS_COVEROPEN	2	The paper cover is open.
STS_NEAREND	4	The paper is low (Near end).
STS_AUTO_CUTTER_ERROR	8	Cutter error
STS_TPH_OVERHEATING	16	TPH is overheated
STS_UNRECOVERABLE_ERROR	32	Unrecoverable error is occurred
STS_PAPER_TO_BE_TAKEN	64	Printed paper to be taken

* Example

```
int status = 0x00;

ConnectToPrinter(.....);

SetAutoStatusCheck(true);

......

status = GetStatus();
......

if ((status & STS_PAPEREMPTY) == STS_PAPEREMPTY)
    ......
```

3-21 Select Mode

• Select Label/Receipt Mode. Only mobile printers are supported.
int Select Mode(bool label Mode);

[Parameters]

* bool label Mode
[in] Send whether to use Label Mode.
If the value is TRUE, the label mode is selected.

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
WRITE_ERROR	-105	Data transmission failed.
NOT_SUPPORT	-124	The function is not supported.

* Example

```
ConnectToPrinter(.....);

......

// Select Label Mode
if (SelectMode(true) != SUCCESS)
    return;

NextPrintPos();

// Select Receipt Mode
if (SelectMode(false) != SUCCESS)
    return;

......
```

3-22 Next Print Pos

• Feed the paper to the starting point of the next label paper.
The function is only enabled if he mobile printer is set to label mode.
int Next Print Pos();

[Parameters]

None

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
WRITE_ERROR	-105	Data transmission failed.
NOT_SUPPORT	-124	The function is not supported.

* Example

```
ConnectToPrinter(.....);

......

if (SelectMode(true) != SUCCESS)
    return;

NextPrintPos();

......
```

3-23 Auto Calibration

• Perform Auto Calibration if set to Label Mode. The function is only enabled if the mobile printer is set to label mode.
int Auto Calibration();

[Parameters]

None

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
WRITE_ERROR	-105	Data transmission failed.
NOT_SUPPORT	-124	The function is not supported.

* Example

```
ConnectToPrinter(.....);

......

if (SelectMode(true) != SUCCESS)
    return;

AutoCalibration();

......
```

3-24 Select Page Mode

• Enable/disable Page Mode.
int Select Page Mode(bool page Mode);

[Parameters]

* bool page Mode
[in] Set whether to use Page Mode.
Page Mode is selected if set to TRUE.

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
WRITE_ERROR	-105	Data transmission failed.
NOT_SUPPORT	-124	The function is not supported.

* Example

```
ConnectToPrinter(.....);

......

// Select Page Mode
if (SelectPageMode(true) != SUCCESS)
    return;

.......

// Select Standard Mode
if (SelectPageMode(false) != SUCCESS)
    return;

......
```

3-25 Print Data In PM

• Prints all the data in the printer buffer if set to Page Mode and the printer is switched to Standard Mode after printing.
int Print Data In PM();

[Parameters]

None

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
WRITE_ERROR	-105	Data transmission failed.
NOT_SUPPORT	-124	The function is not supported.

* Example

```
ConnectToPrinter(.....);

......

// Select Page Mode
if (SelectPageMode(true) != SUCCESS)
    return;

......

PrintDataInPM();
```

3-26 Set Print Area In PM

• Sets the size and position of the printing area when set to Page Mode.
int Set Print Area In PM(long x, Iong y, Iong width, Iong height);

[Parameters]

* Iong x
[in] x-coordinates of the printing area

* Iong y
[in] y-coordinates of the printing area

* Iong width
[in] width of the printing area

* Iong height
[in] height of the printing area

Width of 58mm: x = 0, y = 0, width = 384, height = 840

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
WRITE_ERROR	-105	Data transmission failed.
BAD_ARGUMENT	-117	The specified argument is not correct.
NOT_SUPPORT	-124	The function is not supported.

* Example

```
ConnectToPrinter(.....);

.....

// Select Page Mode
if (SelectPageMode(true) != SUCCESS)
    return;

SetPrintAreaInPM(0, 0, 416, 416);

.....

PrintDataInPM();
```

3-27 Set Print Direction In PM

• Set the printing direction in the Page Mode.
int Set Print Direction In PM(int print Direction);

[Parameters]

* int print Direction

Constant	Value	Direction	Starting Position	Rotation
PAGEMODE_ROTATE_0	48	Left -> Right	Top left	0 degree
PAGEMODE_ROTATE_90	51	Top -> Bottom	Top right	90 degrees
PAGEMODE_ROTATE_180	50	Right -> Left	Bottom right	180 degrees
PAGEMODE_ROTATE_270	49	Bottom -> Top	Bottom left	270 degrees

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
WRITE_ERROR	-105	Data transmission failed.
BAD_ARGUMENT	-117	The specified argument is not correct.
NOT_SUPPORT	-124	The function is not supported.

* Example

```
ConnectToPrinter(.....);

......

// Select Page Mode
if (SelectPageMode(true) != SUCCESS)
    return;

SetPrintAreaInPM(0, 0, 416, 416);
SetPrintDirectionInPM(PAGEMODE_ROTATE_90);

......

PrintDataInPM();
```

3-28 Set Vertical Position In PM

• Set the vertical position for printing in the Page Mode.
int Set Vertical Position In PM(long position, bool relative);

[Parameters]

* long position
[in] Starting position to be set

* bool relative
[in] Set whether it is relative or absolute position from the current position.
If TRUE, it is set to relative position.

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
WRITE_ERROR	-105	Data transmission failed.
BAD_ARGUMENT	-117	The specified argument is not correct.
NOT_SUPPORT	-124	The function is not supported.

* Example

```
ConnectToPrinter(.....);

......

// Select Page Mode
if (SelectPageMode(true) != SUCCESS)
    return;

SetPrintAreaInPM(0, 0, 416, 416);
SetPrintDirectionInPM(PAGEMODE_ROTATE_90);

SetVerticalPositionInPM(160, false);
SetHorizontalPositionInPM(40);
PrintText("Bixolon Printer.", ALIGNMENT_CENTER, ATTR_FONTTYPE_A,
            TS_WIDTH_0 | TS_HEIGHT_0);

......

PrintDataInPM();
```

3-29 Set Horizontal Position In PM

• Set the horizontal position for printing.
int Set Horizontal Position In PM(Iong position, bool relative);

[Parameters]

* long position
[in] Starting position to be set

* bool relative
[in] Set whether it is relative or absolute position from the current position.
If TRUE, it is set to relative position.

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
WRITE_ERROR	-105	Data transmission failed.
BAD_ARGUMENT	-117	The specified argument is not correct.
NOT_SUPPORT	-124	The function is not supported.

* Example

```
ConnectToPrinter(.....);

......

// Select Page Mode
if (SelectPageMode(true) != SUCCESS)
    return;

SetPrintAreaInPM(0, 0, 416, 416);
SetPrintDirectionInPM(PAGEMODE_ROTATE_90);

SetVerticalPositionInPM(160, false);
SetHorizontalPositionInPM(40, false);
PrintText("Bixolon Printer.", ALIGNMENT_CENTER, ATTR_FONTTYPE_A,
            TS_WIDTH_0 | TS_HEIGHT_0);

......

PrintDataInPM();
```

3-30 Read Start MSR

• Switch the printer status to MSR Ready. If it returns SUCCESS, it is identified as normal. Only mobile printers are supported.

long Read Start MSR();

[Parameters]

None

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
NOT_SUPPORT	-124	MSR function is not supported.
WRITE_ERROR	-105	Data transmission failed.

* Example

```
int ret;

ConnectToPrinter(.....);

ret = ReadStartMSR();

if (SUCCESS != ret)
    return;

.....
```

3-31 Read Cancel MSR

• Disable the MSR Ready status. Only mobile printers are supported.
long Read Cancel MSR();

[Parameters]
None

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
WRITE_ERROR	-105	Data transmission failed.
NOT_SUPPORT	-124	MSR function is not supported.

* Example

```
ConnectToPrinter(.....);

ret = ReadStartMSR();

if (SUCCESS != ret)
    return;

.....

ReadCancelMSR();

.....
```

3-32 Read MSR Data

• Read the MSR data. When the data is read, it returns the data value otherwise the read mode can be canceled using Read Cancel MSR. Only mobile printers are supported.

long Read MSR Data(char *pMSRData1, char *pMSRData2, char *pMSRData3,
const unsigned int buf Len);

[Parameters]

* char *pMSRData1
[out] Read MSR Data Track 1 to the buffer defined by the caller.

* char *pMSRData2
[out] Read MSR Data Track 2 to the buffer defined by the caller.

* char *pMSRData3
[out] Read MSR Data Track 3 to the buffer defined by the caller.

* const unsigned int buf Len
[in] Buffer size of pMSRData1, pMSRData2, and pMSRData3 buf

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
NOT_SUPPORT	-124	MSR function is not supported.

* Example

```
ConnectToPrinter(.....);
......

ret = ReadStartMSR();

if (SUCCESS != ret)
   return;

char track1[120] = {0x00, };
char track2[120] = {0x00, };
char track3[120] = {0x00, };

ret = ReadMSRData(track1, track2, track3, sizeof(track1));

if (SUCCESS == ret)
     ......
else
     ......
```

3-33 Scr Power Up

• Turn on the SCR. This function is only available on SPP-R210 SCR.
long Scr Power Up(char *pATR, unsigned int *ATR Len, char *pResponse);

[Parameters]

* char *pATR
[in, out] ATR(Answer To Reset) data

* unsigned int *ATR Len
[in, out] ATR data length

* char *pResponse
[in, out] Response to power up

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
WRITE_ERROR	-105	Data transmission failed.
BAD_ARGUMENT	-117	The specified argument is not correct.
NOT_SUPPORT	-124	The function is not supported.
SCR_RESPONSE_ERROR	-126	The response data is incorrect.

* Example

```
int ret;
char response = 0xff;
char pATR[512] = {0x00,};
unsigned int atrLne = 512;

ConnectToPrinter(.....);

.....

ret = ScrPowerUp(pATR, &atrLen, &response);

if (0x00 != scrData.ResponseS)
    return;

.....
```

3-34 Scr Power Down

• Turn off the SCR. This function is only available on SPP-R210 SCR.
long Scr Power Down(char *pResponse);

[Parameters]

* char *pResponse
[in, out] Response to power up

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
WRITE_ERROR	-105	Data transmission failed.
BAD_ARGUMENT	-117	The specified argument is not correct.
NOT_SUPPORT	-124	The function is not supported.
SCR_RESPONSE_ERROR	-126	The response data is incorrect.

* Example

```
int ret;
char response = 0xff;

ConnectToPrinter(.....);

......

ret = ScrPowerDown(&response);

if (0x00 != response)
    return;

......
```

3-35 Scr Operation Mode

• Set the operation mode. This function is only available on SPP-R210 SCR.
long ScrOperationMode(uint mode, char *pResponse);

[Parameters]

* UINT mode
[in] Operating mode.

Constant	Value	Description
SCR_MODE_APDU	0	APDU mode
SCR_MODE_TPDU	1	TPDU mode

* char *pResponse
[in, out] Response to power up

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
WRITE_ERROR	-105	Data transmission failed.
NOT_SUPPORT	-124	The function is not supported.
SCR_RESPONSE_ERROR	-126	The response data is incorrect.

* Example

```
int ret;
char response = 0xff;

ConnectToPrinter(.....);

......

ret = ScrOperationMode(SCR_MODE_APDU, &response);

if (0x00 != response)
    return;

......
```

3-36 Scr Exchange APDU

• Enable APDU/TPDU data communication.
This function is only available on SPP-R210 SCR.
long Scr Exchange APDU (const char *APDU Cmd, unsigned int APDU Len,
char *APDU Rsp, unsigned int *APDU Rsp Len, char *pResponse);

[Parameters]

* const char *APDU Cmd
[in] APDU data command

* unsigned int APDU Len
[in] APDU command length

* char *APDU Rsp
[in, out] APDU data response

* unsigned int APDU Rsp Len
[in, out] APDU response length

* char *pResponse
[in, out] Response to power up

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
WRITE_ERROR	-105	Data transmission failed.
NOT_SUPPORT	-124	The function is not supported.
SCR_RESPONSE_ERROR	-126	The response data is incorrect.

* Example

```
int ret;
char response = 0xff;
char cmdAPDU[512] = {0x00, };
char rspAPDU[512] = {0x00, };
unsigned int cmdLen, rspLen;

ConnectToPrinter(.....);

......

scrData = ScrExchangeAPDU(cmdAPDU, cmdLen, rspAPDU, rspLen, &response);

if (0x00 != response)
    return;

......
```

3-37 Scr Check Status

• Check the smart card status. This function is only available on SPP-R210 SCR.
long Scr Check Status (char *status, char *pResponse);

[Parameters]

* char *status
[in, out] Smart card status data

* char *pResponse
[in, out] Response to the command

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
WRITE_ERROR	-105	Data transmission failed.
NOT_SUPPORT	-124	The function is not supported.
SCR_RESPONSE_ERROR	-126	The response data is incorrect.

* Example

```
int ret;
char response = 0xff;
char status = 0x00;

ConnectToPrinter(.....);

......

ret = ScrCheckStatus(&status, &response);

if (0x00 != response)
    return;

......
```

3-38 Scr Select Card

• Select Smart card, SAM1 and SAM2 for communication.
This function is only available on SPP-R210 SCR.
long Scr Select Card (uint card, char *pResponse);

[Parameters]

* UINT card
[in] Smart card for communication

Constant	Value	Description
SCR_SMARTCARD	48	Set Smart card for communication.
SCR_SAM1	49	Set SAM1 for communication.
SCR_SAM2	50	Set SAM2 for communication.

* char *pResponse
[in, out] Response to the command

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
WRITE_ERROR	-105	Data transmission failed.
NOT_SUPPORT	-124	The function is not supported.
SCR_RESPONSE_ERROR	-126	The response data is incorrect.

* Example

```
int ret;
char response = 0xff;

ConnectToPrinter(.....);

......

ret = ScrSelectCard(SCR_SMARTCARD, &response);

if (0x00 != response)
    return;

......
```

3-39 Get Battery Status

• Read the SRP-Q300/SRP-Q302 Printer Battery status (FULL, HIGH, MIDDLE, LOW).
int get Battery Status();

[Parameters]

None

[Return Values]

Constant	Value	Description
SUCCESS	0 to 3	The function is successful. 0: Battery Full 1: Battery High 2: Battery Middle 3: Battery Low
READ_TIMEOUT	-1	It timed out before reading the data

* Example

```
int status = 0x00;

ConnectToPrinter(.....);

.....

status = getBatteryStatus();
.....
```

3-40 Set Connection Timeout

• This function sets the maximum waiting time when connecting to the printer with the Connect To Printer function.
void Set Connection Timeout(unsigned int milli Sec)

[Parameters]

* unsigned int milli Sec
[in] Waiting time for printer connection. If the connect to the printer is not succeeded during the waiting time, Connect To Printer returns Fail.
If waiting time with this function is not set via this funciton, it waits up to 3 seconds (3000ms). If the waiting time is set to 0, the printer waits indefinitely for connection.

[Return Values]

None

* Example

```
unsigned int timeout = 2000;

//It waits for up to 2 seconds for connection.
SetConnectionTimeout(timeout);

ConnectToPrinter(.....);

.....
```

3-41 Paper Eject

• This function supports presenter built printer to eject paper
This function is only available on BK3-3 Presenter.
int Paper Eject(int option)

[Parameters]

* int option
[in] It selects paper eject option.

Constant	Value	Description
EJT_NONE	0	The printer eject all paper. No paper is detected by Presenter out sensor.
EJT_HOLD (Recommend)	1	The printer eject paper in state that user can pull out. Paper is detected by Presenter out sensor

[Return Values]

Constant	Value	Description
SUCCESS	0	The function is successful.
WRITE_ERROR	-105	Data transmission failed.
BAD_ARGUMENT	-117	The specified argument is not correct.
NOT_SUPPORT	-124	The function is not supported.

* Example

```
int status = 0x00;

ConnectToPrinter(.....);

.....

status = PaperEject(EJT_HOLD);
.....
```

3-42 Get Presenter Status

• This function supports checking Presenter's status. The status value can be set in duplicate and each value can be checked using bitwise operation.
This function is only available on BK3-3 Presenter.
int Get Presenter Status()

[Parameter]

None

[Return Values]

Constant	Value	Description
PRT_STS_NORMAL	0	Normal
PRT_STS_PAPER_NEAR_END	1	Paper near end
PRT_STS_PAPER_EMPTY	4	Paper empty
PRT_STS_PAPER_IN	8	Paper is detected
PRT_STS_PAPER_JAM	128	Paper jam
WRITE_ERROR	-105	Data transmission failed.
NOT_SUPPORT	-124	The function is not supported.
READ_TIMEOUT	-127	It timed out before reading the data.

* Example

```
int status = 0x00;

ConnectToPrinter(.....);

......

status = GetPresenterStatus();

if(0 > status)
   return; //Error

if (0x00 == status)
   printf("Presenter Normal\n");

else
{
   if (status & PRT_STS_PAPER_NEAR_END)
      printf("Presenter : Near end\n");
   if (status & PRT_STS_PAPEREMPTY)
      printf("Presenter : Paper empty\n");
   if (status & PRT_STS_PAPER_IN)
      printf("Presenter : Paper in\n");
   if (status & PRT_STS_PAPER_JAM)
      printf("Presenter : Paper jam\n");
}
```

Revision history

Rev.	Date	Description
2.00	2018-05-23	New
2.01	2018-07-30	Add model SRP-383, return value modify Code page and ICS modify
2.02	2018-10-24	Add model BK3-3, Code page modify, Method modify, Example modify
2.03	2019-03-05	Add model SRP-Q200
2.04	2019-12-03	Add model BK3-2, Add Set Connection Timeout function, Add Paper Eject function, Add Get Presenter Status function, Typo correction
2.05	2020-04-29	Add model SPP-100II, SMB690, SPP-R220 Direct IO Function read Data parameter description modifications
2.06	2020-09-15	Add model SRP-B300, SRP-S320, SRP-S3000 Add const of Get Status API -STS_AUTO_CUTTER_ERROR -STS_TPH_OVERHEATING -STS_UNRECOVERABLE_ERROR -STS_PAPER_TO_BE_TAKEN
2.07	2021-02-03	Add model : SRP-S200 Add character set - CS_PC864, CS_FARSI, CS_WPC1256, CS_KHMER
2.08	2021-11-02	Add model : SPP-C200
2.09	2022-03-30	Add model : SPP-C300
2.10	2022-05-19	Add model : SRP-330III / 332III
2.11	2022-06-07	Add model : SRP-350V / 352V / 350plusV / 352plusV Modify the const name of the Print Barcode
2.12	2022-07-22	Add model : SRP-380II / 382II
2.13	2023-02-16	Add model : BK5-3
2.14	2023-03-13	Add model : G30
2.15	2023-10-10	Add model : SRP-Q300II
2.16	2024-08-09	Add model: SRP-S300II
2.17	2025-04-25	Add model SRP-B300II
2.18	2026-03-12	Add model: SRP-350plus, SRP-352plus, BK3R-3, BK5R-3, BK5-4, SRP-G300