

## Contents

SDK Introduction.....	6
Frequently Asked Questions.....	7
Interface Specification.....	8
Enum & Macro.....	8
CP_CharacterSet.....	8
CP_CharacterCodepage.....	9
CP_MultiByteEncoding.....	11
CP_ImageBinarizationMethod.....	12
CP_ImageCompressionMethod.....	13
CP_ImagePixelsFormat.....	14
CP_QRCodeECC.....	15
CP_Pos_Alignment.....	16
CP_Pos_BarcodeType.....	17
CP_Pos_BarcodeTextPrintPosition.....	18
CP_Page_DrawDirection.....	19
CP_Page_DrawAlignment.....	20
CP_Label_BarcodeType.....	21
CP_Label_BarcodeTextPrintPosition.....	22
CP_Label_Rotation.....	23
CP_Label_Color.....	24
CP_PRINTERSTATUS.....	25
CP_RTSTATUS.....	28
CP_LABEL_TEXT_STYLE.....	30
Callback.....	31
CP_OnNetPrinterDiscovered.....	31
CP_OnBluetoothDeviceDiscovered.....	32
CP_OnPortOpenedEvent.....	33
CP_OnPortOpenFailedEvent.....	34
CP_OnPortClosedEvent.....	35
CP_OnPortWrittenEvent.....	36
CP_OnPortReceivedEvent.....	37
CP_OnPrinterStatusEvent.....	38
CP_OnPrinterReceivedEvent.....	39
CP_OnPrinterPrintedEvent.....	40
Add Callback and Remove Callback.....	41
CP_Port_AddOnPortOpenedEvent.....	41
CP_Port_AddOnPortOpenFailedEvent.....	42
CP_Port_AddOnPortClosedEvent.....	43
CP_Port_AddOnPortWrittenEvent.....	44
CP_Port_AddOnPortReceivedEvent.....	45
CP_Port_RemoveOnPortOpenedEvent.....	46
CP_Port_RemoveOnPortOpenFailedEvent.....	47
CP_Port_RemoveOnPortClosedEvent.....	48
CP_Port_RemoveOnPortWrittenEvent.....	49
CP_Port_RemoveOnPortReceivedEvent.....	50

CP_Printer_AddOnPrinterStatusEvent.....	51
CP_Printer_AddOnPrinterReceivedEvent.....	52
CP_Printer_AddOnPrinterPrintedEvent.....	53
CP_Printer_RemoveOnPrinterStatusEvent.....	54
CP_Printer_RemoveOnPrinterReceivedEvent.....	55
CP_Printer_RemoveOnPrinterPrintedEvent.....	56
Port Function.....	57
CP_Port_EnumNetPrinter.....	57
CP_Port_EnumBleDevice.....	58
CP_Port_OpenTcp.....	59
CP_Port_OpenBtBle.....	60
CP_Port_OpenBtBleProtoV2.....	61
CP_Port_OpenBtBleProtoV3.....	62
CP_Port_OpenMemoryBuffer.....	63
CP_Port_GetMemoryBufferData.....	64
CP_Port_ClearMemoryBufferData.....	65
CP_Port_Write.....	66
CP_Port_Read.....	67
CP_Port_ReadUntilByte.....	68
CP_Port_Available.....	69
CP_Port_SkipAvailable.....	70
CP_Port_IsConnectionValid.....	71
CP_Port_IsOpened.....	72
CP_Port_Close.....	73
Get Printer Info Function.....	74
CP_Printer_GetPrinterResolutionInfo.....	74
CP_Printer_GetPrinterFirmwareVersion.....	75
CP_Printer_GetPrinterStatusInfo.....	76
CP_Printer_GetPrinterReceivedInfo.....	77
CP_Printer_GetPrinterPrintedInfo.....	78
CP_Printer_GetPrinterLabelPositionAdjustmentInfo.....	79
CP_Printer_SetPrinterLabelPositionAdjustmentInfo.....	80
CP_Printer_ClearPrinterBuffer.....	81
CP_Printer_ClearPrinterError.....	82
Pos Function.....	83
CP_Pos_QueryRTStatus.....	83
CP_Pos_QueryPrintResult.....	84
CP_Pos_KickOutDrawer.....	85
CP_Pos_Beep.....	86
CP_Pos_FeedAndHalfCutPaper.....	87
CP_Pos_FullCutPaper.....	88
CP_Pos_HalfCutPaper.....	89
CP_Pos_FeedLine.....	90
CP_Pos_FeedDot.....	91
CP_Pos_PrintSelfTestPage.....	92
CP_Pos_PrintText.....	93
CP_Pos_PrintTextInUTF8.....	94
CP_Pos_PrintTextInGBK.....	95
CP_Pos_PrintTextInBIG5.....	96

CP_Pos_PrintTextInShiftJIS.....	97
CP_Pos_PrintTextInEUCKR.....	98
CP_Pos_PrintBarcode.....	99
CP_Pos_PrintBarcode_Code128Auto.....	100
CP_Pos_PrintQRCode.....	101
CP_Pos_PrintQRCodeUseEpsonCmd.....	102
CP_Pos_PrintDoubleQRCode.....	103
CP_Pos_PrintPDF417BarcodeUseEpsonCmd.....	104
CP_Pos_PrintRasterImageFromFile.....	105
CP_Pos_PrintRasterImageFromData.....	106
CP_Pos_PrintRasterImageFromPixels.....	107
CP_Pos_PrintHorizontalLine.....	109
CP_Pos_PrintHorizontalLineSpecifyThickness.....	110
CP_Pos_PrintMultipleHorizontalLinesAtOneRow.....	111
CP_Pos_ResetPrinter.....	112
CP_Pos_SetPrintSpeed.....	113
CP_Pos_SetPrintDensity.....	114
CP_Pos_SetSingleByteMode.....	115
CP_Pos_SetCharacterSet.....	116
CP_Pos_SetCharacterCodepage.....	117
CP_Pos_SetMultiByteMode.....	118
CP_Pos_SetMultiByteEncoding.....	119
CP_Pos_SetMovementUnit.....	120
CP_Pos_SetPrintAreaLeftMargin.....	121
CP_Pos_SetPrintAreaWidth.....	122
CP_Pos_SetHorizontalAbsolutePrintPosition.....	123
CP_Pos_SetHorizontalRelativePrintPosition.....	124
CP_Pos_SetVerticalAbsolutePrintPosition.....	125
CP_Pos_SetVerticalRelativePrintPosition.....	126
CP_Pos_SetAlignment.....	127
CP_Pos_SetTextScale.....	128
CP_Pos_SetAsciiTextFontType.....	129
CP_Pos_SetTextBold.....	130
CP_Pos_SetTextUnderline.....	131
CP_Pos_SetTextUpsideDown.....	132
CP_Pos_SetTextWhiteOnBlack.....	133
CP_Pos_SetTextRotate.....	134
CP_Pos_SetTextLineHeight.....	135
CP_Pos_SetAsciiTextCharRightSpacing.....	136
CP_Pos_SetKanjiTextCharSpacing.....	137
CP_Pos_SetBarcodeUnitWidth.....	138
CP_Pos_SetBarcodeHeight.....	139
CP_Pos_SetBarcodeReadableTextFontType.....	140
CP_Pos_SetBarcodeReadableTextPosition.....	141
Page Mode Function.....	142
CP_Page_SelectPageMode.....	142
CP_Page_SelectPageModeEx.....	143
CP_Page_ExitPageMode.....	144
CP_Page_PrintPage.....	145

CP_Page_ClearPage.....	146
CP_Page_SetPageArea.....	147
CP_Page_SetPageDrawDirection.....	148
CP_Page_DrawRect.....	149
CP_Page_DrawBox.....	150
CP_Page_DrawText.....	151
CP_Page_DrawTextInUTF8.....	152
CP_Page_DrawTextInGBK.....	153
CP_Page_DrawTextInBIG5.....	154
CP_Page_DrawTextInShiftJIS.....	155
CP_Page_DrawTextInEUCKR.....	156
CP_Page_DrawBarcode.....	157
CP_Page_DrawQRCode.....	158
CP_Page_DrawRasterImageFromFile.....	159
CP_Page_DrawRasterImageFromData.....	160
CP_Page_DrawRasterImageFromPixels.....	161
Black Marker Function.....	163
CP_BlackMark_EnableBlackMarkMode.....	163
CP_BlackMark_DisableBlackMarkMode.....	164
CP_BlackMark_SetBlackMarkMaxFindLength.....	165
CP_BlackMark_FindNextBlackMark.....	166
CP_BlackMark_SetBlackMarkPaperPrintPosition.....	167
CP_BlackMark_SetBlackMarkPaperCutPosition.....	168
CP_BlackMark_FullCutBlackMarkPaper.....	169
CP_BlackMark_HalfCutBlackMarkPaper.....	170
Label Function.....	171
CP_Label_EnableLabelMode.....	171
CP_Label_DisableLabelMode.....	172
CP_Label_CalibrateLabel.....	173
CP_Label_FeedLabel.....	174
CP_Label_BackPaperToPrintPosition.....	175
CP_Label_FeedPaperToTearPosition.....	176
CP_Label_PageBegin.....	177
CP_Label_PagePrint.....	178
CP_Label_DrawText.....	179
CP_Label_DrawTextInUTF8.....	180
CP_Label_DrawTextInGBK.....	181
CP_Label_DrawTextInBIG5.....	182
CP_Label_DrawTextInShiftJIS.....	183
CP_Label_DrawTextInEUCKR.....	184
CP_Label_DrawBarcode.....	185
CP_Label_DrawQRCode.....	187
CP_Label_DrawPDF417Code.....	189
CP_Label_DrawImageFromFile.....	191
CP_Label_DrawImageFromData.....	192
CP_Label_DrawImageFromPixels.....	193
CP_Label_DrawLine.....	195
CP_Label_DrawRect.....	196
CP_Label_DrawBox.....	197

Other Function.....	198
CP_Library_Version.....	198
CP_Proto_QueryBatteryLevel.....	199
CP_Proto_QuerySerialNumber.....	200
CP_Proto_SetSystemNameAndSerialNumber.....	201
CP_Proto_SetBluetoothNameAndPassword.....	202
CP_Proto_SetPTPBasicParameters.....	203
CP_Settings_Hardware_SetPrintSpeed.....	206

# SDK Introduction

- 1 The development kit comes with a lot of detailed examples. Before development, please run the corresponding example tests. The test is completely ok, and then consider development
- 2 The development kit supports a variety of printers, including but not limited to ticket printing, label printing, page mode printing, black label printing
- 3 Development package supports automatic return function, which requires printer to support automatic return function
- 4 All the developer functions are prefixed with CP\_ to avoid confusion with other developers
- 5 The development package mainly consists of macro definition, enumeration, callback, port function, ticket printing function, label printing function, page mode printing function, black mark related function
  - Port functions start with CP\_Port\_ and include functions such as open Port, close Port, read/write Port, etc
  - Note printing function begins with CP\_Pos\_ and mainly encapsulates various note instructions, which can print text, bar code, TWO-DIMENSIONAL code, pictures, etc
  - The Label printing function begins with CP\_Label\_ and mainly encapsulates Label instructions. It can print text, bar code, TWO-DIMENSIONAL code, pictures, etc
  - Page mode printing function begins with CP\_Page\_, mainly encapsulates the Page mode related instructions, can print text, bar code, TWO-DIMENSIONAL code, pictures, etc
  - The BlackMark correlation function begins with CP\_BlackMark\_ and mainly encapsulates the black calibration bit correlation instruction
- 6 A complete printing process is to open the port, print various functions, and close the port
  - Call-back interface, available or not, does not affect the normal printing process, call-back is only used for prompt message
- 7 Engineering code, import the library, you can call all the functions of the development package

# Frequently Asked Questions

## 1 How can I tell what type of printer I have and what functions To use?

Should pay attention to look at the model, is what model, what function, use the wrong can not print, or will print garbled code

Note paper printing, is the note type, using CP\_Pos\_ series of functions for printing

Put Label paper, is the Label model, using CP\_Label\_ series functions for printing

Some machines can print on both ticker paper and label paper and they can call functions to turn on and off label mode

Had better be to consult the seller, see should use what example to test, reference example will write the most save trouble

## 2 Why is printing half turned off?

Look whether the power supply is insufficient, below rated voltage, it is the power supply that wants 2A commonly enough

When the printer is started up, the flashing light of the indicator light is generally different, which can be clearly seen

If there is a problem, carefully observe the indicator light or sound so that you can easily locate the problem

## 3 After the label is printed, why is it not positioned in the gap

Check whether the label mode is turned on and whether the label has been recognized. The test method is to press the paper feed button to see whether a piece of paper is complete

# Interface Specification

## Enum & Macro

### CP\_CharacterSet

Single-byte mode of international character set when the printer is in single-byte mode, set up the printer international character set that will change the range 0x20–0x7f part of the text printing such as currency symbol of currency or dollars part details see instruction set the printer when the printer is in multi-byte mode, set this property has no effect

#### Syntax

```
typedef enum CP_CharacterSet {  
    CP_CharacterSet_USA = 0,  
    CP_CharacterSet_FRANCE = 1,  
    CP_CharacterSet_GERMANY = 2,  
    CP_CharacterSet_UK = 3,  
    CP_CharacterSet_DENMARK_I = 4,  
    CP_CharacterSet_SWEDEN = 5,  
    CP_CharacterSet_ITALY = 6,  
    CP_CharacterSet_SPAIN_I = 7,  
    CP_CharacterSet_JAPAN = 8,  
    CP_CharacterSet_NORWAY = 9,  
    CP_CharacterSet_DENMARK_II = 10,  
    CP_CharacterSet_SPAIN_II = 11,  
    CP_CharacterSet_LATIN = 12,  
    CP_CharacterSet_KOREA = 13,  
    CP_CharacterSet_SLOVENIA = 14,  
    CP_CharacterSet_CHINA = 15  
} CP_CharacterSet;
```



## CP\_CharacterCodepage

Character Code Page in Single-byte Mode When the printer is in single-byte mode, setting the printer character Code page changes the printing of parts of the interval 0x80–0xff for details see the printer instruction set section. Setting this property does not matter when the printer is in multi-byte mode

### Syntax

```
typedef enum CP_CharacterCodepage {  
    CP_CharacterCodepage_CP437 = 0,  
    CP_CharacterCodepage_KATAKANA = 1,  
    CP_CharacterCodepage_CP850 = 2,  
    CP_CharacterCodepage_CP860 = 3,  
    CP_CharacterCodepage_CP863 = 4,  
    CP_CharacterCodepage_CP865 = 5,  
    CP_CharacterCodepage_WCP1251 = 6,  
    CP_CharacterCodepage_CP866 = 7,  
    CP_CharacterCodepage_MIK = 8,  
    CP_CharacterCodepage_CP755 = 9,  
    CP_CharacterCodepage_IRAN = 10,  
    CP_CharacterCodepage_CP862 = 15,  
    CP_CharacterCodepage_WCP1252 = 16,  
    CP_CharacterCodepage_WCP1253 = 17,  
    CP_CharacterCodepage_CP852 = 18,  
    CP_CharacterCodepage_CP858 = 19,  
    CP_CharacterCodepage_IRAN_II = 20,  
    CP_CharacterCodepage_LATVIAN = 21,  
    CP_CharacterCodepage_CP864 = 22,  
    CP_CharacterCodepage_ISO_8859_1 = 23,  
    CP_CharacterCodepage_CP737 = 24,  
    CP_CharacterCodepage_WCP1257 = 25,  
    CP_CharacterCodepage_THAI = 26,  
    CP_CharacterCodepage_CP720 = 27,  
    CP_CharacterCodepage_CP855 = 28,  
    CP_CharacterCodepage_CP857 = 29,  
    CP_CharacterCodepage_WCP1250 = 30,  
    CP_CharacterCodepage_CP775 = 31,  
    CP_CharacterCodepage_WCP1254 = 32,  
    CP_CharacterCodepage_WCP1255 = 33,  
    CP_CharacterCodepage_WCP1256 = 34,  
    CP_CharacterCodepage_WCP1258 = 35,  
    CP_CharacterCodepage_ISO_8859_2 = 36,  
    CP_CharacterCodepage_ISO_8859_3 = 37,  
    CP_CharacterCodepage_ISO_8859_4 = 38,  
    CP_CharacterCodepage_ISO_8859_5 = 39,  
    CP_CharacterCodepage_ISO_8859_6 = 40,
```

```
CP_CharacterCodepage_ISO_8859_7 = 41 ,
CP_CharacterCodepage_ISO_8859_8 = 42 ,
CP_CharacterCodepage_ISO_8859_9 = 43 ,
CP_CharacterCodepage_ISO_8859_15 = 44 ,
CP_CharacterCodepage_THAI_2 = 45 ,
CP_CharacterCodepage_CP856 = 46 ,
CP_CharacterCodepage_CP874 = 47 ,
CP_CharacterCodepage_TCVN3 = 48
} CP_CharacterCodepage ;
```

## CP\_MultiByteEncoding

Multibyte mode in multi-byte character encoding printer mode, received the print data, will be carried out in accordance with the specified code printing, for example, set up the printer model, multiple bytes to specify multibyte character encoding UTF8 encoding, under the mode of application should be in accordance with the UTF8 encoding send a string to a printer, the printer will print the string

### Syntax

```
typedef enum CP_MultiByteEncoding { CP_MultiByteEncoding_GBK = 0, CP_MultiByteEncoding_UTF8 = 1,
CP_MultiByteEncoding_BIG5 = 3, CP_MultiByteEncoding_ShiftJIS = 4, CP_MultiByteEncoding_EUCKR = 5 }
CP_MultiByteEncoding;
```

## CP\_ImageBinarizationMethod

Image binarization algorithm because the printer can print black and white monochrome bitmap, the process of printing image, if the original image is colour or greyscale, and requires the use of binarization algorithm, the original image to monochrome figure the effect of different algorithms have different threshold algorithm for image content is text error diffusion method applied to all of the pictures, but scrutiny will have burrs error diffusion method, the effect is not as good as dithering algorithm is not recommended, only do compatibility

### Syntax

```
typedef enum CP_ImageBinarizationMethod { CP_ImageBinarizationMethod_Dithering, CP_ImageBinarizationMethod_Thresholding, CP_ImageBinarizationMethod_ErrorDiffusion } CP_ImageBinarizationMethod;
```

## CP\_ImageCompressionMethod

Image compression algorithm some printers support the use of compression instructions to print pictures , improve the efficiency of data transmission whether the specific support needs to see the actual test results

### Syntax

```
typedef enum CP_ImageCompressionMethod { CP_ImageCompressionMethod_None, CP_ImageCompressionMethod_Level1 ,  
CP_ImageCompressionMethod_Level2 } CP_ImageCompressionMethod ;
```

## CP\_ImagePixelFormat

When printing a picture in pixel format, if the image is printed with directly transmitted pixel data, the data and format should correspond

### Syntax

```
typedef enum CP_ImagePixelFormat {  
    CP_ImagePixelFormat_MONO = 1,  
    CP_ImagePixelFormat_MONOLSB = 2,  
    CP_ImagePixelFormat_GRAY8 = 3,  
    CP_ImagePixelFormat_BYTEORDERED_RGB24 = 4,  
    CP_ImagePixelFormat_BYTEORDERED_BGR24 = 5,  
    CP_ImagePixelFormat_BYTEORDERED_ARGB32 = 6,  
    CP_ImagePixelFormat_BYTEORDERED_RGBA32 = 7,  
    CP_ImagePixelFormat_BYTEORDERED_ABGR32 = 8,  
    CP_ImagePixelFormat_BYTEORDERED_BGRA32 = 9  
} CP_ImagePixelFormat;
```

CP\_ImagePixelFormat\_MONO = 1,  
Monochrome bitmap, high in front

CP\_ImagePixelFormat\_MONOLSB = 2,  
Monochromatic bitmap, low in front

CP\_ImagePixelFormat\_GRAY8 = 3,  
A grayscale image in which each color takes up one byte

CP\_ImagePixelFormat\_BYTEORDERED\_RGB24 = 4,  
R G B takes up one byte per color in byte order

CP\_ImagePixelFormat\_BYTEORDERED\_BGR24 = 5,  
B G R takes up one byte per color in byte order

CP\_ImagePixelFormat\_BYTEORDERED\_ARGB32 = 6,  
A R G B takes up one byte per color in byte order

CP\_ImagePixelFormat\_BYTEORDERED\_RGBA32 = 7,  
R G B A takes up one byte per color in byte order

CP\_ImagePixelFormat\_BYTEORDERED\_ABGR32 = 8,  
A B G R takes up one byte per color in byte order

CP\_ImagePixelFormat\_BYTEORDERED\_BGRA32 = 9  
B G R A takes up one byte per color in byte order

## CP\_QRCodeECC

Qr code error correction level

### Syntax

```
typedef enum CP_QRCodeECC { CP_QRCodeECC_L = 1, CP_QRCodeECC_M = 2, CP_QRCodeECC_Q = 3,  
CP_QRCodeECC_H = 4 } CP_QRCodeECC;
```

## CP\_Pos\_Alignment

In ticket mode, there are left, middle and right alignment for printing

### Syntax

```
typedef enum CP_Pos_Alignment { CP_Pos_Alignment_Left, CP_Pos_Alignment_HCenter, CP_Pos_Alignment_Right }  
CP_Pos_Alignment;
```



## CP\_Pos\_BarcodeType

When the bill instruction prints the barcode, specify the barcode type

### Syntax

```
typedef enum CP_Pos_BarcodeType {  
    CP_Pos_BarcodeType_UPCA = 0x41,  
    CP_Pos_BarcodeType_UPCE = 0x42,  
    CP_Pos_BarcodeType_EAN13 = 0x43,  
    CP_Pos_BarcodeType_EAN8 = 0x44,  
    CP_Pos_BarcodeType_CODE39 = 0x45,  
    CP_Pos_BarcodeType_ITF = 0x46,  
    CP_Pos_BarcodeType_CODEBAR = 0x47,  
    CP_Pos_BarcodeType_CODE93 = 0x48,  
    CP_Pos_BarcodeType_CODE128 = 0x49  
} CP_Pos_BarcodeType;
```

## CP\_Pos\_BarcodeTextPrintPosition

When the bill instruction prints the barcode, specify the printing position of the barcode character

### Syntax

```
typedef      enum      CP_Pos_BarcodeTextPrintPosition      {      CP_Pos_BarcodeTextPrintPosition_None ,  
CP_Pos_BarcodeTextPrintPosition_AboveBarcode ,      CP_Pos_BarcodeTextPrintPosition_BelowBarcode ,  
CP_Pos_BarcodeTextPrintPosition_AboveAndBelowBarcode } CP_Pos_BarcodeTextPrintPosition ;
```

## CP\_Page\_DrawDirection

When printing in page mode, specify the drawing direction of the page

### Syntax

```
typedef enum CP_Page_DrawDirection { CP_Page_DrawDirection_LeftToRight, CP_Page_DrawDirection_BottomToTop,  
CP_Page_DrawDirection_RightToLeft, CP_Page_DrawDirection_TopToBottom } CP_Page_DrawDirection;
```

## CP\_Page\_DrawAlignment

In page mode, the coordinate is greater than or equal to zero, the actual coordinate can also specify a specific value at this point, specify to align the print within the region

### Syntax

```
#ifndef MarcoDefinitionCPPPageDrawAlignment
#define MarcoDefinitionCPPPageDrawAlignment
#define CP_Page_DrawAlignment_Left -1
#define CP_Page_DrawAlignment_HCenter -2
#define CP_Page_DrawAlignment_Right -3
#define CP_Page_DrawAlignment_Top -1
#define CP_Page_DrawAlignment_VCenter -2
#define CP_Page_DrawAlignment_Bottom -3
#endif
```

## CP\_Label\_BarcodeType

When the label instruction prints the barcode, specify the barcode type

### Syntax

```
typedef enum CP_Label_BarcodeType {  
    CP_Label_BarcodeType_UPCA = 0,  
    CP_Label_BarcodeType_UPCE = 1,  
    CP_Label_BarcodeType_EAN13 = 2,  
    CP_Label_BarcodeType_EAN8 = 3,  
    CP_Label_BarcodeType_CODE39 = 4,  
    CP_Label_BarcodeType_ITF = 5,  
    CP_Label_BarcodeType_CODEBAR = 6,  
    CP_Label_BarcodeType_CODE93 = 7,  
    CP_Label_BarcodeType_CODE128 = 8,  
    CP_Label_BarcodeType_CODE11 = 9,  
    CP_Label_BarcodeType_MSI = 10,  
    CP_Label_BarcodeType_128M = 11,  
    CP_Label_BarcodeType_EAN128 = 12,  
    CP_Label_BarcodeType_25C = 13,  
    CP_Label_BarcodeType_39C = 14,  
    CP_Label_BarcodeType_39 = 15,  
    CP_Label_BarcodeType_EAN13PLUS2 = 16,  
    CP_Label_BarcodeType_EAN13PLUS5 = 17,  
    CP_Label_BarcodeType_EAN8PLUS2 = 18,  
    CP_Label_BarcodeType_EAN8PLUS5 = 19,  
    CP_Label_BarcodeType_POST = 20,  
    CP_Label_BarcodeType_UPCAPLUS2 = 21,  
    CP_Label_BarcodeType_UPCAPLUS5 = 22,  
    CP_Label_BarcodeType_UPCEPLUS2 = 23,  
    CP_Label_BarcodeType_UPCEPLUS5 = 24,  
    CP_Label_BarcodeType_CPOST = 25,  
    CP_Label_BarcodeType_MSIC = 26,  
    CP_Label_BarcodeType_PLESSEY = 27,  
    CP_Label_BarcodeType_ITF14 = 28,  
    CP_Label_BarcodeType_EAN14 = 29  
} CP_Label_BarcodeType;
```

## CP\_Label\_BarcodeTextPrintPosition

When the label instruction prints the barcode, specify the printing position of the barcode text

### Syntax

```
typedef      enum      CP_Label_BarcodeTextPrintPosition      {      CP_Label_BarcodeTextPrintPosition_None ,  
CP_Label_BarcodeTextPrintPosition_AboveBarcode ,      CP_Label_BarcodeTextPrintPosition_BelowBarcode ,  
CP_Label_BarcodeTextPrintPosition_AboveAndBelowBarcode } CP_Label_BarcodeTextPrintPosition ;
```

## CP\_Label\_Rotation

The label instruction specifies the rotation Angle when the control is drawn

### Syntax

```
typedef enum CP_Label_Rotation { CP_Label_Rotation_0, CP_Label_Rotation_90, CP_Label_Rotation_180,  
CP_Label_Rotation_270 } CP_Label_Rotation;
```

## CP\_Label\_Color

When the label instructs you to draw a control, specify that the paint color can be white or black

### Syntax

```
typedef enum CP_Label_Color { CP_Label_Color_White, CP_Label_Color_Black } CP_Label_Color;
```



## CP\_PRINTERSTATUS

The status definition of automatic printer return generally only needs to pay attention to whether there is an error, and the information part mainly plays the function of prompt

### Syntax

```
@discardableResult
public func CP_PRINTERSTATUS_ERROR_CUTTER(error_status: Int64) -> Bool {
    return (error_status & 0x01) != 0
}

@discardableResult
public func CP_PRINTERSTATUS_ERROR_FLASH(error_status: Int64) -> Bool {
    return (error_status & 0x02) != 0
}

@discardableResult
public func CP_PRINTERSTATUS_ERROR_NOPAPER(error_status: Int64) -> Bool {
    return (error_status & 0x04) != 0
}

@discardableResult
public func CP_PRINTERSTATUS_ERROR_VOLTAGE(error_status: Int64) -> Bool {
    return (error_status & 0x08) != 0
}

@discardableResult
public func CP_PRINTERSTATUS_ERROR_MARKER(error_status: Int64) -> Bool {
    return (error_status & 0x10) != 0
}

@discardableResult
public func CP_PRINTERSTATUS_ERROR_ENGINE(error_status: Int64) -> Bool {
    return (error_status & 0x20) != 0
}

@discardableResult
public func CP_PRINTERSTATUS_ERROR_OVERHEAT(error_status: Int64) -> Bool {
    return (error_status & 0x40) != 0
}

@discardableResult
public func CP_PRINTERSTATUS_ERROR_COVERUP(error_status: Int64) -> Bool {
    return (error_status & 0x80) != 0
}

@discardableResult
public func CP_PRINTERSTATUS_ERROR_MOTOR(error_status: Int64) -> Bool {
    return (error_status & 0x100) != 0
}

@discardableResult
public func CP_PRINTERSTATUS_INFO_LABELPAPER(info_status: Int64) -> Bool {
    return (info_status & 0x02) != 0
}
```

```

}
@discardableResult
public func CP_PRINTERSTATUS_INFO_LABELMODE(info_status: Int64) -> Bool {
    return (info_status & 0x04) != 0
}
@discardableResult
public func CP_PRINTERSTATUS_INFO_HAVEDATA(info_status: Int64) -> Bool {
    return (info_status & 0x08) != 0
}
@discardableResult
public func CP_PRINTERSTATUS_INFO_NOPAPERCANCELED(info_status: Int64) -> Bool {
    return (info_status & 0x10) != 0
}
@discardableResult
public func CP_PRINTERSTATUS_INFO_PAPERNOFETCH(info_status: Int64) -> Bool {
    return (info_status & 0x20) != 0
}
@discardableResult
public func CP_PRINTERSTATUS_INFO_PRINTIDLE(info_status: Int64) -> Bool {
    return (info_status & 0x40) != 0
}
@discardableResult
public func CP_PRINTERSTATUS_INFO_RECVIDLE(info_status: Int64) -> Bool {
    return (info_status & 0x80) != 0
}

// ERROR_CUTTER
//     Cutter error
// ERROR_FLASH
//     FLASH error
// ERROR_NOPAPER
//     No paper
// ERROR_VOLTAGE
//     Voltage error
// ERROR_MARKER
//     Black mark or seam mark error detected
// ERROR_ENGINE
//     Unrecognized printer engine
// ERROR_OVERHEAT
//     Overheat
// ERROR_COVERUP
//     Open cover or shaft not pressed down
// ERROR_MOTOR
//     Motor out of step (usually paper jam)
// INFO_LABELPAPER
//     Current paper identified as label paper (0 is continuous paper)
// INFO_LABELMODE
//     Currently in label mode
// INFO_HAVEDATA

```

```
//      We have data to start processing
// INFO_NOPAPERCANCELED
//      The last document was cancelled after it was short of paper
// INFO_PAPERNOFETCH
//      The documents were not taken
// INFO_PRINTIDLE
//      Current print idle
// INFO_RECVIDLE
//      The current receive buffer is empty
```

# CP\_RTSTATUS

The status definition returned by real-time status query here is for reference only. It is applicable to most models. If some models are inconsistent, the instruction set of actual models shall prevail

## Syntax

```
@discardableResult
public func CP_RTSTATUS_DRAWER_OPENED(status: Int) -> Bool {
    return (((status >> 0) & 0x04) == 0x00)
}

@discardableResult
public func CP_RTSTATUS_OFFLINE(status: Int) -> Bool {
    return (((status >> 0) & 0x08) == 0x08)
}

@discardableResult
public func CP_RTSTATUS_COVERUP(status: Int) -> Bool {
    return (((status >> 8) & 0x04) == 0x04)
}

@discardableResult
public func CP_RTSTATUS_FEED_PRESSED(status: Int) -> Bool {
    return (((status >> 8) & 0x08) == 0x08)
}

@discardableResult
public func CP_RTSTATUS_NOPAPER(status: Int) -> Bool {
    return (((status >> 8) & 0x20) == 0x20)
}

@discardableResult
public func CP_RTSTATUS_ERROR_OCCURED(status: Int) -> Bool {
    return (((status >> 8) & 0x40) == 0x40)
}

@discardableResult
public func CP_RTSTATUS_CUTTER_ERROR(status: Int) -> Bool {
    return (((status >> 16) & 0x08) == 0x08)
}

@discardableResult
public func CP_RTSTATUS_UNRECOVERABLE_ERROR(status: Int) -> Bool {
    return (((status >> 16) & 0x20) == 0x20)
}

@discardableResult
public func CP_RTSTATUS_DEGREE_OR_VOLTAGE_OVERRANGE(status: Int) -> Bool {
    return (((status >> 16) & 0x40) == 0x40)
}

@discardableResult
public func CP_RTSTATUS_PAPER_NEAREND(status: Int) -> Bool {
    return (((status >> 24) & 0x08) == 0x08)
}
```

```

}
@discardableResult
public func CP_RTSTATUS_PAPER_TAKEOUT(status: Int) -> Bool {
    return (((status >> 24) & 0x04) == 0x04)
}

// The real-time state here is four bytes
// From low byte to high byte corresponds to these four instructions in the instruction set:
// 10 04 01
// 10 04 02
// 10 04 03
// 10 04 04
// For some models, due to customization or other reasons, the definition of state value may be inconsistent with here,
// subject to the actual measurement
//
// DRAWER_OPENED
//     Drawer Opened
// OFFLINE
//     Offline
// COVERUP
//     Cover UP
// FEED_PRESSED
//     Feed Pressed
// NOPAPER
//     No Paper
// ERROR_OCCURED
//     Error Occured
// CUTTER_ERROR
//     Cutter Error
// UNRECOVERABLE_ERROR
//     Unrecoverable Error
// DEGREE_OR_VOLTAGE_OVERRANGE
//     Degree or voltage error
// PAPER_NEAREND
//     Paper Near End
// PAPER_TAKEOUT
//     Paper takeout

```

## CP\_LABEL\_TEXT\_STYLE

When the label instruction prints text, specify the text printing style as bold, underline, reverse color, delete line, rotate, double width and height

### Syntax

```
#ifndef MarcoDefinitionCPLabelTextStyle
#define MarcoDefinitionCPLabelTextStyle
#define CP_LABEL_TEXT_STYLE_BOLD (1<<0)
#define CP_LABEL_TEXT_STYLE_UNDERLINE (1<<1)
#define CP_LABEL_TEXT_STYLE_HIGHLIGHT (1<<2)
#define CP_LABEL_TEXT_STYLE_STRIKETHROUGH (1<<3)
#define CP_LABEL_TEXT_STYLE_ROTATION_0 (0<<4)
#define CP_LABEL_TEXT_STYLE_ROTATION_90 (1<<4)
#define CP_LABEL_TEXT_STYLE_ROTATION_180 (2<<4)
#define CP_LABEL_TEXT_STYLE_ROTATION_270 (3<<4)
#define CP_LABEL_TEXT_STYLE_WIDTH_ENLARGEMENT(n) ((n)<<8)
#define CP_LABEL_TEXT_STYLE_HEIGHT_ENLARGEMENT(n) ((n)<<12)
#endif
```

## Callback

### CP\_OnNetPrinterDiscovered

When enumerating network printers, the incoming callback function is called back when it is looked up to the network printer

#### Syntax

```
public typealias CP_OnNetPrinterDiscovered_SwiftCallback = (_ local_ip: String, _ discovered_mac: String, _ discovered_ip: String, _ discovered_name: String) -> Void
```

## CP\_OnBluetoothDeviceDiscovered

When enumerating bluetooth devices, the incoming callback function will call back when it is searched for the Bluetooth device

### Syntax

```
public typealias CP_OnBluetoothDeviceDiscovered_SwiftCallback = (_ device_name: String, _ device_address: String) -> Void
```



## CP\_OnPortOpenedEvent

When the port is opened successfully, the function is called back

### Syntax

```
public typealias CP_OnPortOpenedEvent_SwiftCallback = (_ handle: Int, _ name: String) -> Void
```

## CP\_OnPortOpenFailedEvent

When port opening fails, the function is called back

### Syntax

```
public typealias CP_OnPortOpenFailedEvent_SwiftCallback = (_ handle: Int, _ name: String) -> Void
```

## CP\_OnPortClosedEvent

When the port is closed, the interface is called back

### Syntax

```
public typealias CP_OnPortClosedEvent_SwiftCallback = (_ handle: Int) -> Void
```

### Remarks

When the port is abnormal, such as turned off bluetooth, it will automatically close the port and trigger a callback

## CP\_OnPortWrittenEvent

This function is called back when the port writes data successfully

### Syntax

```
public typealias CP_OnPortWrittenEvent_SwiftCallback = (_ handle: Int, _ buffer: [UInt8], _ count: Int) -> Void
```

## CP\_OnPortReceivedEvent

This function is called back when the port receives data

### Syntax

```
public typealias CP_OnPortReceivedEvent_SwiftCallback = (_ handle: Int, _ buffer: [UInt8], _ count: Int) -> Void
```

## CP\_OnPrinterStatusEvent

This function is called back when the status of automatic printer returns is received

### Syntax

```
public typealias CP_OnPrinterStatusEvent_SwiftCallback = (_ handle: Int, _ printer_error_status: Int64, _ printer_info_status: Int64) -> Void
```

## CP\_OnPrinterReceivedEvent

This function is called back when the number of bytes received is automatically returned by the printer

### Syntax

```
public typealias CP_OnPrinterReceivedEvent_SwiftCallback = (_ handle: Int, _ printer_received_byte_count: Int) -> Void
```

## CP\_OnPrinterPrintedEvent

This function will be called back when you receive the document automatically returned by the printer. This function will be deprecated and is not recommended

### Syntax

```
public typealias CP_OnPrinterPrintedEvent_SwiftCallback = (_ handle: Int, _ printer_printed_page_id: Int) -> Void
```



## Add Callback and Remove Callback

### CP\_Port\_AddOnPortOpenedEvent

Add Callback, Open Port Success

#### Syntax

```
public func CP_Port_AddOnPortOpenedEvent(event: @escaping CP_OnPortOpenedEvent_SwiftCallback, tag: Int) -> Bool
```

```
//      Add Callback, Open Port Success
//
//  event
//      callback function
//
//  tag
//      the callback id
//
//  return
//      true on success.
//      false on failed.
```

## CP\_Port\_AddOnPortOpenFailedEvent

Add Callback, Open Port Failed

### Syntax

```
public func CP_Port_AddOnPortOpenFailedEvent(event: @escaping CP_OnPortOpenFailedEvent_SwiftCallback, tag: Int) -> Bool
```

```
//      Add Callback, Open Port Failed
//
//  event
//      callback function
//
//  tag
//      the callback id
//
// return
//      true on success.
//      false on failed.
```

## CP\_Port\_AddOnPortClosedEvent

Add Callback, Port Closed

### Syntax

```
public func CP_Port_AddOnPortClosedEvent(event: @escaping CP_OnPortClosedEvent_SwiftCallback, tag: Int) -> Bool
```

```
//      Add Callback, Port Closed
//
//  event
//      callback function
//
//  tag
//      the callback id
//
// return
//      true on success.
//      false on failed.
```

## CP\_Port\_AddOnPortWrittenEvent

Add Callback, Port Written Bytes

### Syntax

public func CP\_Port\_AddOnPortWrittenEvent(event: @escaping CP\_OnPortWrittenEvent\_SwiftCallback, tag: Int) -> Bool

```
//      Add Callback, Port Written Bytes
//
//  event
//      callback function
//
//  tag
//      the callback id
//
// return
//      true on success.
//      false on failed.
```

## CP\_Port\_AddOnPortReceivedEvent

Add Callback, Port Received Bytes

### Syntax

public func CP\_Port\_AddOnPortReceivedEvent(event: @escaping CP\_OnPortReceivedEvent\_SwiftCallback, tag: Int) -> Bool

```
//      Add Callback, Port Received Bytes
//
//  event
//      callback function
//
//  tag
//      the callback id
//
// return
//      true on success.
//      false on failed.
```

## CP\_Port\_RemoveOnPortOpenedEvent

Remove Callback

### Syntax

```
public func CP_Port_RemoveOnPortOpenedEvent(tag: Int) -> Bool
```

```
//      Remove Callback
//
// tag
//      the callback id
//
// return
//      true on success.
//      false on failed.
```

## CP\_Port\_RemoveOnPortOpenFailedEvent

Remove Callback

### Syntax

```
public func CP_Port_RemoveOnPortOpenFailedEvent(tag: Int) -> Bool
```

```
//      Remove Callback
//
// tag
//      the callback id
//
// return
//      true on success.
//      false on failed.
```

## CP\_Port\_RemoveOnPortClosedEvent

Remove Callback

### Syntax

```
public func CP_Port_RemoveOnPortClosedEvent(tag: Int) -> Bool
```

```
//      Remove Callback
//
// tag
//      the callback id
//
// return
//      true on success.
//      false on failed.
```



## CP\_Port\_RemoveOnPortWrittenEvent

Remove Callback

### Syntax

```
public func CP_Port_RemoveOnPortWrittenEvent(tag: Int) -> Bool
```

```
//      Remove Callback
//
// tag
//      the callback id
//
// return
//      true on success.
//      false on failed.
```

## CP\_Port\_RemoveOnPortReceivedEvent

Remove Callback

### Syntax

```
public func CP_Port_RemoveOnPortReceivedEvent(tag: Int) -> Bool
```

```
//      Remove Callback
//
// tag
//      the callback id
//
// return
//      true on success.
//      false on failed.
```

## CP\_Printer\_AddOnPrinterStatusEvent

Add Callback, Printer Status Updated

### Syntax

```
public func CP_Printer_AddOnPrinterStatusEvent(event: @escaping CP_OnPrinterStatusEvent_SwiftCallback, tag: Int) -> Bool
```

```
//      Add Callback, Printer Status Updated
//
//  event
//      callback function
//
//  tag
//      the callback id
//
//  return
//      true on success.
//      false on failed.
```

## CP\_Printer\_AddOnPrinterReceivedEvent

Add Callback, Printer Received Byte Count Updated

### Syntax

```
public func CP_Printer_AddOnPrinterReceivedEvent(event: @escaping CP_OnPrinterReceivedEvent_SwiftCallback, tag: Int)
-> Bool
```

```
//      Add Callback, Printer Received Byte Count Updated
//
//  event
//      callback function
//
//  tag
//      the callback id
//
// return
//      true on success.
//      false on failed.
```

## CP\_Printer\_AddOnPrinterPrintedEvent

Add Callback, Printer Printed Page ID Updated

### Syntax

```
public func CP_Printer_AddOnPrinterPrintedEvent(event: @escaping CP_OnPrinterPrintedEvent_SwiftCallback, tag: Int) -> Bool
```

```
//      Add Callback, Printer Printed Page ID Updated
//
//  event
//      callback function
//
//  tag
//      the callback id
//
//  return
//      true on success.
//      false on failed.
```

## CP\_Printer\_RemoveOnPrinterStatusEvent

Remove Callback

### Syntax

```
public func CP_Printer_RemoveOnPrinterStatusEvent(tag: Int) -> Bool
```

```
//      Remove Callback
//
// tag
//      the callback id
//
// return
//      true on success.
//      false on failed.
```

## CP\_Printer\_RemoveOnPrinterReceivedEvent

Remove Callback

### Syntax

```
public func CP_Printer_RemoveOnPrinterReceivedEvent(tag: Int) -> Bool
```

```
//      Remove Callback
//
// tag
//      the callback id
//
// return
//      true on success.
//      false on failed.
```

## CP\_Printer\_RemoveOnPrinterPrintedEvent

Remove Callback

### Syntax

```
public func CP_Printer_RemoveOnPrinterPrintedEvent(tag: Int) -> Bool
```

```
//      Remove Callback
//
// tag
//      the callback id
//
// return
//      true on success.
//      false on failed.
```



## Port Function

### CP\_Port\_EnumNetPrinter

Enumerate net printer

#### Syntax

```
public func CP_Port_EnumNetPrinter(timeout: Int, cancel: CP_LoopController, on_discovered: @escaping  
CP_OnNetPrinterDiscovered_SwiftCallback)
```

```
// Enumerate net printer  
//  
// timeout  
// enumrate timeout ms  
//  
// cancel  
// cancel bit, if value is non-zero, enum process will exit.  
//  
// on_discovered  
// enumrated callback function  
//  
// private_data  
// the parameter passed to callback function  
//  
// return  
// none
```

## CP\_Port\_EnumBleDevice

Enumerate BLE printer

### Syntax

```
public func CP_Port_EnumBleDevice(timeout: Int, cancel: CP_LoopController, on_discovered: @escaping  
CP_OnBluetoothDeviceDiscovered_SwiftCallback)
```

```
// Enumerate BLE printer  
//  
// timeout  
// enumrate timeout ms  
//  
// cancel  
// cancel bit, if value is non-zero, enum process will exit.  
//  
// on_discovered  
// enumrated callback function  
//  
// private_data  
// the parameter passed to callback function  
//  
// return  
// none
```

# CP\_Port\_OpenTcp

Open Tcp

## Syntax

public func CP\_Port\_OpenTcp(local\_ip: String, dest\_ip: String, dest\_port: Int, timeout: Int, autoreplymode: Int) -> Int

```
//      Open Tcp
//
// local_ip
//      Bind to local IP
//      For multiple network CARDS or multiple local ips, select the specified IP
//      Passing in a 0 indicates that it is not specified
//
// dest_ip
//      IP Address or printer name
//      For example: 192.168.1.87
//
// dest_port
//      Port Number
//      Fixed value: 9100
//
// timeout
//      connect timeout
//
// autoreplymode
//      0 don't start autoreplymode
//      1 start autoreplymode
//      attention:
//      Only some models support automatic return mode, please ask the seller if you support it
//      After the automatic return mode is enabled, the printer will return the status automatically
//      The state of the printer cannot be automatically obtained if it is not started
//
// return
//      Return handle, If open success, return non-zero value, else return zero.
//
// remarks
//      PC and printer need in the same network segment, so they can connect
```

## CP\_Port\_OpenBtBle

Connect Bluetooth Printer Via BLE

### Syntax

```
public func CP_Port_OpenBtBle(address: String, autoreplymode: Int) -> Int
```

```
//      Connect Bluetooth Printer Via BLE
//
// address
//      bluetooth address
//
// autoreplymode
//      0 don't start autoreplymode
//      1 start autoreplymode
//      attention:
//      Only some models support automatic return mode, please ask the seller if you support it
//      After the automatic return mode is enabled, the printer will return the status automatically
//      The state of the printer cannot be automatically obtained if it is not started
//
// return
//      Return handle, If open success, return non-zero value, else return zero.
//
// remarks
//      only for android,ios,macos
```

## CP\_Port\_OpenBtBleProtoV2

Connect Bluetooth Printer Via BLE(Use Special Proto To Transfer Data)

### Syntax

public func CP\_Port\_OpenBtBleProtoV2(address: String, autoreplymode: Int) -> Int

```
//      Connect Bluetooth Printer Via BLE(Use Special Proto To Transfer Data)
//
// address
//      bluetooth address
//
// autoreplymode
//      0 don't start autoreplymode
//      1 start autoreplymode
//      attention:
//      Only some models support automatic return mode, please ask the seller if you support it
//      After the automatic return mode is enabled, the printer will return the status automatically
//      The state of the printer cannot be automatically obtained if it is not started
//
// return
//      Return handle, If open success, return non-zero value, else return zero.
//
// remarks
//      only for ios,macos
```

## CP\_Port\_OpenBtBleProtoV3

Connect Bluetooth Printer Via BLE(Use Special Proto To Transfer Data)

### Syntax

public func CP\_Port\_OpenBtBleProtoV3(address: String, autoreplymode: Int) -> Int

```
//      Connect Bluetooth Printer Via BLE(Use Special Proto To Transfer Data)
//
// address
//      bluetooth address
//
// autoreplymode
//      0 don't start autoreplymode
//      1 start autoreplymode
//      attention:
//      Only some models support automatic return mode, please ask the seller if you support it
//      After the automatic return mode is enabled, the printer will return the status automatically
//      The state of the printer cannot be automatically obtained if it is not started
//
// return
//      Return handle, If open success, return non-zero value, else return zero.
//
// remarks
//      only for ios,macos
```

## CP\_Port\_OpenMemoryBuffer

Creates a memory buffer that all subsequent print instructions are written to

### Syntax

```
public func CP_Port_OpenMemoryBuffer(buffer_size: Int) -> Int
```

```
//      Creates a memory buffer that all subsequent print instructions are written to
//
// buffer_size
//      buffer size
//
// return
//      Return handle, If open success, return non-zero value, else return zero.
//
// remarks
//      This can be used in the following two scenarios
//      scenario one:
//          A single document needs to send more than one instruction, and you want to put all the instructions together
and send them to the printer at once to improve the transmission speed
//      scenario two:
//          Sometimes you need to know exactly what a function is sending, you need to know exactly what data
is being sent, right
```

## CP\_Port\_GetMemoryBufferData

get memory buffer data

### Syntax

```
public func CP_Port_GetMemoryBufferData(handle: Int) -> [UInt8]
```

```
//      get memory buffer data pointer
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      return memory buffer data
```



## CP\_Port\_ClearMemoryBufferData

Clear memory buffer data

### Syntax

```
public func CP_Port_ClearMemoryBufferData(handle: Int) -> Bool
```

```
//      Clear memory buffer data
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      true on success.
//      false on failed.
```

## CP\_Port\_Write

Write data to port

### Syntax

```
public func CP_Port_Write(handle: Int, buffer: [UInt8], count: Int, timeout: Int) -> Int
```

```
//      Write data to port
//
// handle
//      Port handle, returned by OpenXXX
//
// buffer
//      buffer
//
// count
//      buffer length
//
// timeout
//      Timeout ms
//
// return
//      return bytes writted. or return -1 means failed
```

## CP\_Port\_Read

Receive data from port

### Syntax

```
public func CP_Port_Read(handle: Int, buffer: inout [UInt8], count: Int, timeout: Int) -> Int
```

```
//      Receive data from port
//
// handle
//      Port handle, returned by OpenXXX
//
// buffer
//      buffer
//
// count
//      buffer length
//
// timeout
//      Timeout ms
//
// return
//      return bytes readed. or return -1 means failed
```

## CP\_Port\_ReadUntilByte

Receive data from port

### Syntax

```
public func CP_Port_ReadUntilByte(handle: Int, buffer: inout [UInt8], count: Int, timeout: Int, breakByte: UInt8) -> Int
```

```
//      Receive data from port
//
// handle
//      Port handle, returned by OpenXXX
//
// buffer
//      buffer
//
// count
//      buffer length
//
// timeout
//      Timeout ms
//
// breakByte
//      break read byte
//
// return
//      return bytes readed. or return -1 means failed
```

## CP\_Port\_Available

get readable data from port

### Syntax

```
public func CP_Port_Available(handle: Int) -> Int
```

```
//      get readable data from port
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      return readable. or return -1 means failed
```

## CP\_Port\_SkipAvailable

Skip receive buffer

### Syntax

```
public func CP_Port_SkipAvailable(handle: Int) -> Bool
```

```
//      Skip receive buffer
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      true on success.
//      false on failed.
```

## CP\_Port\_IsConnectionValid

Check Connection Valid

### Syntax

```
public func CP_Port_IsConnectionValid(handle: Int) -> Bool
```

```
//      Check Connection Valid
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      Returns true if the port is open and the status is continuously updated
//      False is returned if the port is not open, closed, or the status is not updated for more than 6 seconds
```

## CP\_Port\_IsOpened

Check Port Is Opened

### Syntax

```
public func CP_Port_IsOpened(handle: Int) -> Bool
```

```
//      Check Port Is Opened
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      Returns true if the port is open and the connection is not broken or closed
//      Returns false if the port is not open, or if the connection is broken or closed
```



## CP\_Port\_Close

Close Port

### Syntax

```
public func CP_Port_Close(handle: Int) -> Bool
```

```
//      Close Port
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      true on success.
//      false on failed.
```

## Get Printer Info Function

### CP\_Printer\_GetPrinterResolutionInfo

Get Printer Resolution Info

#### Syntax

```
public func CP_Printer_GetPrinterResolutionInfo(handle: Int, width_mm: inout Int, height_mm: inout Int, dots_per_mm:
inout Int) -> Bool
```

```
//      Get Printer Resolution Info
//
// handle
//      Port handle, returned by OpenXXX
//
// width_mm
//      Max Page Width
//
// height_mm
//      Max Page Height
//
// dots_per_mm
//      Dots Per MM
//
// return
//      true on success.
//      false on failed.
```

## CP\_Printer\_GetPrinterFirmwareVersion

Get Printer Firmware Version

### Syntax

```
public func CP_Printer_GetPrinterFirmwareVersion(handle: Int) -> String
```

```
//      Get Printer Firmware Version
//
// handle
//      Port handle, returned by OpenXXX
//
// pBuf
//      The buffer
//
// cbBuf
//      The buffer size
//
// pcbNeeded
//      The buffer bytes needed
//
// return
//      true on success.
//      false on failed.
```

## CP\_Printer\_GetPrinterStatusInfo

Get Printer Status

### Syntax

public func CP\_Printer\_GetPrinterStatusInfo(handle: Int, printer\_error\_status: inout Int64, printer\_info\_status: inout Int64, timestamp\_ms: inout Int64) -> Bool

```
//      Get Printer Status
//
// handle
//      Port handle, returned by OpenXXX
//
// printer_error_status
//      Printer Error Status
//
// printer_info_status
//      Printer Info Status
//
// timestamp_ms
//      timestamp
//
// return
//      true on success.
//      false on failed.
```

## CP\_Printer\_GetPrinterReceivedInfo

Get Printer Received Byte Count

### Syntax

public func CP\_Printer\_GetPrinterReceivedInfo(handle: Int, printer\_received\_byte\_count: inout Int, timestamp\_ms: inout Int64)

→ Bool

```
//      Get Printer Received Byte Count
//
// handle
//      Port handle, returned by OpenXXX
//
// printer_received_byte_count
//      Printer Received Byte Count
//
// timestamp_ms
//      timestamp
//
// return
//      true on success.
//      false on failed.
```

## CP\_Printer\_GetPrinterPrintedInfo

Get Printer Printed Page ID

### Syntax

public func CP\_Printer\_GetPrinterPrintedInfo(handle: Int, printer\_printed\_page\_id: inout Int, timestamp\_ms: inout Int64) -> Bool

```
//      Get Printer Printed Page ID
//
// handle
//      Port handle, returned by OpenXXX
//
// printer_printed_page_id
//      Printer Printed Page ID
//
// timestamp_ms
//      timestamp
//
// return
//      true on success.
//      false on failed.
```

## CP\_Printer\_GetPrinterLabelPositionAdjustmentInfo

Get Printer Label Position Adjustment

### Syntax

```
public func CP_Printer_GetPrinterLabelPositionAdjustmentInfo(handle: Int, label_print_position_adjustment: inout Double,  
label_tear_position_adjustment: inout Double, timestamp_ms: inout Int64) -> Bool
```

```
//      Get Printer Label Position Adjustment  
//  
// handle  
//      Port handle, returned by OpenXXX  
//  
// label_print_position_adjustment  
//      Printer label print position adjustment  
//  
// label_tear_position_adjustment  
//      Printer label tear position adjustment  
//  
// timestamp_ms  
//      timestamp  
//  
// return  
//      true on success.  
//      false on failed.
```

## CP\_Printer\_SetPrinterLabelPositionAdjustmentInfo

adjust label print position and tear paper position

### Syntax

```
public func CP_Printer_SetPrinterLabelPositionAdjustmentInfo(handle: Int, label_print_position_adjustment: Double,  
label_tear_position_adjustment: Double) -> Bool
```

```
//      adjust label print position and tear paper position  
//  
// handle  
//      Port handle, returned by OpenXXX  
//  
// label_print_position_adjustment  
//      Label print position adjustment mm (adjustment can not exceed [-4,4])  
//  
// label_tear_position_adjustment  
//      Label tear position adjustment mm (adjustment can not exceed [-4,4])  
//  
// return  
//      If command is written successfully, it returns true else it returns false.
```



## CP\_Printer\_ClearPrinterBuffer

Clear Printer Buffer Runtime

### Syntax

```
public func CP_Printer_ClearPrinterBuffer(handle: Int) -> Bool
```

```
//      Clear Printer Buffer Runtime
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      true on success.
//      false on failed.
```

## CP\_Printer\_ClearPrinterError

Clear Printer Error Runtime

### Syntax

```
public func CP_Printer_ClearPrinterError(handle: Int) -> Bool
```

```
//      Clear Printer Error Runtime
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      true on success.
//      false on failed.
```

## Pos Function

### CP\_Pos\_QueryRTStatus

Query the real-time status of the printer

#### Syntax

```
public func CP_Pos_QueryRTStatus(handle: Int, timeout: Int) -> Int
```

```
//      Query the real-time status of the printer
//      If it is a machine that supports automatic return, the state will be returned automatically. No need to use this
command to query
//      Due to the real-time state instruction, there is no check, the result cannot be guaranteed to be correct
//
// handle
//      Port handle, returned by OpenXXX
//
// timeout
//      timeout ms
//      The wait time for query does not exceed this time
//
// return
//      If command is successfully, it returns rtstatus else it returns 0.
//      Please check CP_RTSTATUS_XXX for the detailed status. If the status definition is inconsistent with the actual
model, the actual measurement shall prevail.
```

## CP\_Pos\_QueryPrintResult

Query the print result of the previous content

### Syntax

```
public func CP_Pos_QueryPrintResult(handle: Int, timeout: Int) -> Bool
```

```
//      Query the print result of the previous content
//
// handle
//      Port handle, returned by OpenXXX
//
// timeout
//      timeout ms
//      The wait time for query print results does not exceed this time
//
// return
//      If print is successfully, it returns true else it returns false.
```

## CP\_Pos\_KickOutDrawer

Turn on cashbox

### Syntax

public func CP\_Pos\_KickOutDrawer(handle: Int, nDrawerIndex: Int, nHighLevelTime: Int, nLowLevelTime: Int) -> Bool

```
//      Turn on cashbox
//
// handle
//      Port handle, returned by OpenXXX
//
// nDrawerIndex
//      Cashbox no, value are defined as follow:
//      value      define
//      0          Cashbox pin 2
//      1          Cashbox pin 5
//
// nHighLevelTime
//      Cashbox pulse high potential ms time
//
// nLowLevelTime
//      Cashbox pulse low potential ms time
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_Beep

Buzzer call

### Syntax

```
public func CP_Pos_Beep(handle: Int, nBeepCount: Int, nBeepMs: Int) -> Bool
```

```
//      Buzzer call
//
// handle
//      Port handle, returned by OpenXXX
//
// nBeepCount
//      Calling times
//
// nBeepMs
//      Calling time ms, value range is [100,900], flour to 100 milliseconds.
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_FeedAndHalfCutPaper

feed to cutter position and half cut paper

### Syntax

```
public func CP_Pos_FeedAndHalfCutPaper(handle: Int) -> Bool
```

```
//      feed to cutter position and half cut paper
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_FullCutPaper

full cut paper

### Syntax

```
public func CP_Pos_FullCutPaper(handle: Int) -> Bool
```

```
//      full cut paper
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      If command is written successfully, it returns true else it returns false.
```



## CP\_Pos\_HalfCutPaper

half cut paper

### Syntax

```
public func CP_Pos_HalfCutPaper(handle: Int) -> Bool
```

```
//      half cut paper
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_FeedLine

printer feed numLines

### Syntax

```
public func CP_Pos_FeedLine(handle: Int, numLines: Int) -> Bool
```

```
//      printer feed numLines
//
// handle
//      Port handle, returned by OpenXXX
//
// numLines
//      number of lines to feed
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_FeedDot

printer feed numDots

### Syntax

```
public func CP_Pos_FeedDot(handle: Int, numDots: Int) -> Bool
```

```
//      printer feed numDots
//
// handle
//      Port handle, returned by OpenXXX
//
// numDots
//      number of dots to feed
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_PrintSelfTestPage

printer print self test page

### Syntax

```
public func CP_Pos_PrintSelfTestPage(handle: Int) -> Bool
```

```
//      printer print self test page
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_PrintText

print text

### Syntax

```
public func CP_Pos_PrintText(handle: Int, str: String) -> Bool
```

```
//      print text
//
// handle
//      Port handle, returned by OpenXXX
//
// str
//      the string to print
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_PrintTextInUTF8

print text

### Syntax

```
public func CP_Pos_PrintTextInUTF8(handle: Int, str: String) -> Bool
```

```
//      print text
//
// handle
//      Port handle, returned by OpenXXX
//
// str
//      the string to print
//
// return
//      If command is written successfully, it returns true else it returns false.
//
// remarks
//      The function converts the data to UTF8 encoding.
```

## CP\_Pos\_PrintTextInGBK

print text

### Syntax

```
public func CP_Pos_PrintTextInGBK(handle: Int, str: String) -> Bool
```

```
//      print text
//
// handle
//      Port handle, returned by OpenXXX
//
// str
//      the string to print
//
// return
//      If command is written successfully, it returns true else it returns false.
//
// remarks
//      The function converts the data to GBK encoding.
```

## CP\_Pos\_PrintTextInBIG5

print text

### Syntax

```
public func CP_Pos_PrintTextInBIG5(handle: Int, str: String) -> Bool
```

```
//      print text
//
// handle
//      Port handle, returned by OpenXXX
//
// str
//      the string to print
//
// return
//      If command is written successfully, it returns true else it returns false.
//
// remarks
//      The function converts the data to BIG5 encoding.
```



## CP\_Pos\_PrintTextInShiftJIS

print text

### Syntax

```
public func CP_Pos_PrintTextInShiftJIS(handle: Int, str: String) -> Bool
```

```
//      print text
//
// handle
//      Port handle, returned by OpenXXX
//
// str
//      the string to print
//
// return
//      If command is written successfully, it returns true else it returns false.
//
// remarks
//      The function converts the data to ShiftJIS encoding.
```

## CP\_Pos\_PrintTextInEUCKR

print text

### Syntax

public func CP\_Pos\_PrintTextInEUCKR(handle: Int, str: String) -> Bool

```
//      print text
//
// handle
//      Port handle, returned by OpenXXX
//
// str
//      the string to print
//
// return
//      If command is written successfully, it returns true else it returns false.
//
// remarks
//      The function converts the data to EUCKR encoding.
```

## CP\_Pos\_PrintBarcode

print 1D barcode

### Syntax

public func CP\_Pos\_PrintBarcode(handle: Int, nBarcodeType: CP\_Pos\_BarcodeType, str: String) -> Bool

```
//      print 1D barcode
//
// handle
//      Port handle, returned by OpenXXX
//
// nBarcodeType
//      barcode type
//      values are defined as follow:
//      value      type
//      0x41      UPC-A
//      0x42      UPC-E
//      0x43      EAN13
//      0x44      EAN8
//      0x45      CODE39
//      0x46      ITF
//      0x47      CODABAR
//      0x48      CODE93
//      0x49      CODE128
//
// str
//      the barcode data to print
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_PrintBarcode\_Code128Auto

print Code128 barcode, this function auto change type b and c to print more characters.

### Syntax

```
public func CP_Pos_PrintBarcode_Code128Auto(handle: Int, str: String) -> Bool
```

```
//      print Code128 barcode, this function auto change type b and c to print more characters.
//      Normally, do not use this function to print CODE128 code
//      This function is mainly used for compatibility with some older models
//      New models already support automatic switching codes by default
//      New models cannot print barcodes using this function
//
// handle
//      Port handle, returned by OpenXXX
//
// str
//      the barcode data to print
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_PrintQRCode

print qrcode

### Syntax

public func CP\_Pos\_PrintQRCode(handle: Int, nVersion: Int, nECCLevel: CP\_QRCodeECC, str: String) -> Bool

```
//      print qrcode
//
// handle
//      Port handle, returned by OpenXXX
//
// nVersion
//      Assign charater version. The value range is:[0,16]
//      When version is 0, printer caculates version number according to character set automatically.
//
// nECCLevel
//      Assign error correction level.
//      The value range is: [1, 4].
//      Definitios are as below:
//      ECC error correction level
//      1   L:7%, low error correction, much data.
//      2   M:15%, medium error correction
//      3   Q:optimize error correction
//      4   H:30%, the highest error correction, less data.
//
// str
//      the qrcode data to print
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_PrintQRCodeUseEpsonCmd

print qrcode

### Syntax

public func CP\_Pos\_PrintQRCodeUseEpsonCmd(handle: Int, nQRCodeUnitWidth: Int, nECCLLevel: CP\_QRCodeECC, str: String) -> Bool

```
//      print qrcode
//
// handle
//      Port handle, returned by OpenXXX
//
// nQRCodeUnitWidth
//      QRCode code block width, the value range is [1, 16]
//
// nECCLLevel
//      Assign error correction level.
//      The value range is: [1, 4].
//      Definitios are as below:
//      ECC error correction level
//      1   L:7%, low error correction, much data.
//      2   M:15%, medium error correction
//      3   Q:optimize error correction
//      4   H:30%, the highest error correction, less data.
//
// str
//      the qrcode data to print
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_PrintDoubleQRCode

print 2 qrcode

### Syntax

public func CP\_Pos\_PrintDoubleQRCode(handle: Int, nQRCodeUnitWidth: Int, nQR1Position: Int, nQR1Version: Int, nQR1ECCLLevel: CP\_QRCodeECC, strQR1: String, nQR2Position: Int, nQR2Version: Int, nQR2ECCLLevel: CP\_QRCodeECC, strQR2: String) -> Bool

```
//      print 2 qrcode
//
// handle
//      Port handle, returned by OpenXXX
//
// nQRCodeUnitWidth
//      QRCode code block width, the value range is [1, 8]
//
// nQR1Position
// nQR2Position
//      QRCode position
//
// nQR1Version
// nQR2Version
//      Assign charater version. The value range is:[0,16]
//      When version is 0, printer caculates version number according to character set automatically.
//
// nQR1ECCLLevel
// nQR2ECCLLevel
//      Assign error correction level.
//      The value range is: [1, 4].
//      Definitios are as below:
//      ECC error correction level
//      1   L:7%, low error correction, much data.
//      2   M:15%, medium error correction
//      3   Q:optimize error correction
//      4   H:30%, the highest error correction, less data.
//
// strQR1
// strQR2
//      the qrcode data to print
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_PrintPDF417BarcodeUseEpsonCmd

```
print pdf417 barcode
```

### Syntax

```
public func CP_Pos_PrintPDF417BarcodeUseEpsonCmd(handle: Int, columnCount: Int, rowCount: Int, unitWidth: Int,
rowCount: Int, nECCLLevel: Int, dataProcessingMode: Int, str: String) -> Bool
```

```
//      print pdf417 barcode
//
// handle
//      Port handle, returned by OpenXXX
//
// columnCount
//      column count, range is [0,30]
//
// rowCount
//      row count, range is 0,[3,90]
//
// unitWidth
//      module unit width, range is [2,8]
//
// rowHeight
//      row height, range is [2,8]
//
// nECCLLevel
//      ecc level, range is [0,8]
//
// dataProcessingMode
//      data processing mode, 0 select standard PDF417, 1 select cutoff PDF417
//
// str
//      the pdf417 data to print
//
// return
//      If command is written successfully, it returns true else it returns false.
```



## CP\_Pos\_PrintRasterImageFromFile

print image

### Syntax

public func CP\_Pos\_PrintRasterImageFromFile(handle: Int, dstw: Int, dsth: Int, pszFile: String, binarization\_method: CP\_ImageBinarizationMethod, compression\_method: CP\_ImageCompressionMethod) -> Bool

```
//      print image
//
// handle
//      Port handle, returned by OpenXXX
//
// dstw
//      the width to print
//
// dsth
//      the height to print
//
// pszFile
//      image file path
//
// binarization_method
//      image binarization method. 0 means use dithering, 1 means use thresholding, 2 means use error diffusion.
//
// compression_method
//      print data compress method, values are defined as follow
//      value define
//      0      no compress
//      1      compress level 1
//      2      compress level 2
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_PrintRasterImageFromData

print image (data can be readed from file)

### Syntax

public func CP\_Pos\_PrintRasterImageFromData(handle: Int, dstw: Int, dsth: Int, data: [UInt8], data\_size: Int, binarization\_method: CP\_ImageBinarizationMethod, compression\_method: CP\_ImageCompressionMethod) -> Bool

```
//      print image (data can be readed from file)
//
// handle
//      Port handle, returned by OpenXXX
//
// dstw
//      the width to print
//
// dsth
//      the height to print
//
// data
//      image data
//
// data_size
//      image data size
//
// binaryzation_method
//      image binaryzation method. 0 means use dithering, 1 means use thresholding, 2 means use error diffusion.
//
// compression_method
//      print data compress method, values are defined as follow
//      value define
//      0      no compress
//      1      compress level 1
//      2      compress level 2
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_PrintRasterImageFromPixels

print image pixels

### Syntax

public func CP\_Pos\_PrintRasterImageFromPixels(handle: Int, img\_data: [UInt8], img\_datalen: Int, img\_width: Int, img\_height: Int, img\_stride: Int, img\_format: CP\_ImagePixelFormat, binarization\_method: CP\_ImageBinarizationMethod, compression\_method: CP\_ImageCompressionMethod) -> Bool

```
//      print image pixels
//
// handle
//      Port handle, returned by OpenXXX
//
// img_data
//      image pixels data
//
// img_datalen
//      image pixels data length
//
// img_width
//      image pixel width
//
// img_height
//      image pixel height
//
// img_stride
//      image horizontal stirde. means bytes per line.
//
// img_format
//      image pixel data format, values are defined as follow
//      value define
//      1      mono
//      2      monolsb
//      3      gray
//      4      r.g.b in byte-ordered
//      5      b.g.r in byte-ordered
//      6      a.r.g.b in byte-ordered
//      7      r.g.b.a in byte-ordered
//      8      a.b.g.r in byte-ordered
//      9      b.g.r.a in byte-ordered
//
// binaryzation_method
//      image binaryzation method. 0 means use dithering, 1 means use thresholding, 2 means use error diffusion.
//
```

```
// compression_method
//      print data compress method, values are defined as follow
//      value define
//      0      no compress
//      1      compress level 1
//      2      compress level 2
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_PrintHorizontalLine

print one horizontal line

### Syntax

```
public func CP_Pos_PrintHorizontalLine(handle: Int, nLineStartPosition: Int, nLineEndPosition: Int) -> Bool
```

```
//      print one horizontal line
//
// handle
//      Port handle, returned by OpenXXX
//
// nLineStartPosition
//      line start position
//
// nLineEndPosition
//      line end position
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_PrintHorizontalLineSpecifyThickness

print one horizontal line

### Syntax

```
public func CP_Pos_PrintHorizontalLineSpecifyThickness(handle: Int, nLineStartPosition: Int, nLineEndPosition: Int, nLineThickness: Int) -> Bool
```

```
//      print one horizontal line
//
// handle
//      Port handle, returned by OpenXXX
//
// nLineStartPosition
//      line start position
//
// nLineEndPosition
//      line end position
//
// nLineThickness
//      line thickness
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_PrintMultipleHorizontalLinesAtOneRow

print multiple horizontal lines at one row, multi call can print curve

### Syntax

```
public func CP_Pos_PrintMultipleHorizontalLinesAtOneRow(handle: Int, nLineCount: Int, pLineStartPosition: [Int],  
pLineEndPosition: [Int]) -> Bool
```

```
//      print multiple horizontal lines at one row, multi call can print curve  
//  
// handle  
//      Port handle, returned by OpenXXX  
//  
// nLineCount  
//      Line count  
//  
// pLineStartPosition  
//      Line start position  
//  
// pLineEndPosition  
//      Line end position  
//  
// return  
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_ResetPrinter

reset printer, clear settings

### Syntax

```
public func CP_Pos_ResetPrinter(handle: Int) -> Bool
```

```
//      reset printer, clear settings
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      If command is written successfully, it returns true else it returns false.
```



## CP\_Pos\_SetPrintSpeed

set print speed (some printer support)

### Syntax

```
public func CP_Pos_SetPrintSpeed(handle: Int, nSpeed: Int) -> Bool
```

```
//      set print speed (some printer support)
//
// handle
//      Port handle, returned by OpenXXX
//
// nSpeed
//      print speed in mm/s
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_SetPrintDensity

set print density (some printer support)

### Syntax

```
public func CP_Pos_SetPrintDensity(handle: Int, nDensity: Int) -> Bool
```

```
//      set print density (some printer support)
//
// handle
//      Port handle, returned by OpenXXX
//
// nDensity
//      the print density[0,15]
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_SetSingleByteMode

set printer to single byte mode

### Syntax

```
public func CP_Pos_SetSingleByteMode(handle: Int) -> Bool
```

```
//      set printer to single byte mode
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_SetCharacterSet

set character set

### Syntax

```
public func CP_Pos_SetCharacterSet(handle: Int, nCharacterSet: CP_CharacterSet) -> Bool
```

```
//      set character set
//
// handle
//      Port handle, returned by OpenXXX
//
// nCharacterSet
//      character set, range is [0, 15]
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_SetCharacterCodepage

set character codepage

### Syntax

public func CP\_Pos\_SetCharacterCodepage(handle: Int, nCharacterCodepage: CP\_CharacterCodepage) -> Bool

```
//      set character codepage
//
// handle
//      Port handle, returned by OpenXXX
//
// nCharacterCodepage
//      character codepage, range is [0,255]
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_SetMultiByteMode

set printer to multi byte mode

### Syntax

```
public func CP_Pos_SetMultiByteMode(handle: Int) -> Bool
```

```
//      set printer to multi byte mode
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_SetMultiByteEncoding

set printer multi byte encoding

### Syntax

public func CP\_Pos\_SetMultiByteEncoding(handle: Int, nEncoding: CP\_MultiByteEncoding) -> Bool

```
//      set printer multi byte encoding
//
// handle
//      Port handle, returned by OpenXXX
//
// nEncoding
//      multi byte encoding, values defined as follow:
//      value define
//      0      GBK
//      1      UTF8
//      3      BIG5
//      4      SHIFT-JIS
//      5      EUC-KR
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_SetMovementUnit

set print movement unit

### Syntax

public func CP\_Pos\_SetMovementUnit(handle: Int, nHorizontalMovementUnit: Int, nVerticalMovementUnit: Int) -> Bool

```
//      set print movement unit
//
// handle
//      Port handle, returned by OpenXXX
//
// nHorizontalMovementUnit
//      horizontal movement unit
//
// nVerticalMovementUnit
//      vertical movement unit
//
// return
//      If command is written successfully, it returns true else it returns false.
//
// remarks
//      if set movement unit to 200, 1mm means 8point.
```



## CP\_Pos\_SetPrintAreaLeftMargin

set print area left margin

### Syntax

```
public func CP_Pos_SetPrintAreaLeftMargin(handle: Int, nLeftMargin: Int) -> Bool
```

```
//      set print area left margin
//
// handle
//      Port handle, returned by OpenXXX
//
// nLeftMargin
//      print area left margin
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_SetPrintAreaWidth

set print area width

### Syntax

```
public func CP_Pos_SetPrintAreaWidth(handle: Int, nWidth: Int) -> Bool
```

```
//      set print area width
//
// handle
//      Port handle, returned by OpenXXX
//
// nWidth
//      print area width
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_SetHorizontalAbsolutePrintPosition

set horizontal absolute print position

### Syntax

```
public func CP_Pos_SetHorizontalAbsolutePrintPosition(handle: Int, nPosition: Int) -> Bool
```

```
//      set horizontal absolute print position
//
// handle
//      Port handle, returned by OpenXXX
//
// nPosition
//      print position
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_SetHorizontalRelativePrintPosition

set horizontal relative print position

### Syntax

```
public func CP_Pos_SetHorizontalRelativePrintPosition(handle: Int, nPosition: Int) -> Bool
```

```
//      set horizontal relative print position
//
// handle
//      Port handle, returned by OpenXXX
//
// nPosition
//      print position
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_SetVerticalAbsolutePrintPosition

set vertical absolute print position, only valid in page mode.

### Syntax

```
public func CP_Pos_SetVerticalAbsolutePrintPosition(handle: Int, nPosition: Int) -> Bool
```

```
//      set vertical absolute print position, only valid in page mode.
//
// handle
//      Port handle, returned by OpenXXX
//
// nPosition
//      print position
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_SetVerticalRelativePrintPosition

set vertical relative print position, only valid in page mode.

### Syntax

```
public func CP_Pos_SetVerticalRelativePrintPosition(handle: Int, nPosition: Int) -> Bool
```

```
//      set vertical relative print position, only valid in page mode.  
//  
// handle  
//      Port handle, returned by OpenXXX  
//  
// nPosition  
//      print position  
//  
// return  
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_SetAlignment

set print alignment

### Syntax

public func CP\_Pos\_SetAlignment(handle: Int, nAlignment: CP\_Pos\_Alignment) -> Bool

```
//      set print alignment
//
// handle
//      Port handle, returned by OpenXXX
//
// nAlignment
//      print alignment, value are defined as follow:
//      value define
//      0      align left
//      1      align center
//      2      align right
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_SetTextScale

set text scale

### Syntax

```
public func CP_Pos_SetTextScale(handle: Int, nWidthScale: Int, nHeightScale: Int) -> Bool
```

```
//      set text scale
//
// handle
//      Port handle, returned by OpenXXX
//
// nWidthScale
//      width scale
//
// nHeightScale
//      height scale
//
// return
//      If command is written successfully, it returns true else it returns false.
```



## CP\_Pos\_SetAsciiTextFontType

set ascii text font type

### Syntax

```
public func CP_Pos_SetAsciiTextFontType(handle: Int, nFontType: Int) -> Bool
```

```
//      set ascii text font type
//
// handle
//      Port handle, returned by OpenXXX
//
// nFontType
//      ascii text font type, values defined as follow:
//      value define
//      0      FontA (12x24)
//      1      FontB (9x17)
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_SetTextBold

set text bold

### Syntax

```
public func CP_Pos_SetTextBold(handle: Int, nBold: Int) -> Bool
```

```
//      set text bold
//
// handle
//      Port handle, returned by OpenXXX
//
// nBold
//      text bold , values defined as follow:
//      value define
//      0      don't bold
//      1      bold
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_SetTextUnderline

set text underline

### Syntax

```
public func CP_Pos_SetTextUnderline(handle: Int, nUnderline: Int) -> Bool
```

```
//      set text underline
//
// handle
//      Port handle, returned by OpenXXX
//
// nUnderline
//      text underline, values defined as follow:
//      value define
//      0      no underline
//      1      1 point underline
//      2      2 point underline
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_SetTextUpsideDown

set text upside down

### Syntax

```
public func CP_Pos_SetTextUpsideDown(handle: Int, nUpsideDown: Int) -> Bool
```

```
//      set text upside down
//
// handle
//      Port handle, returned by OpenXXX
//
// nUpsideDown
//      upside down, values defined as follow:
//      value define
//      0      print text dont't upside down
//      1      print text upside down
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_SetTextWhiteOnBlack

set text black and white reverse

### Syntax

```
public func CP_Pos_SetTextWhiteOnBlack(handle: Int, nWhiteOnBlack: Int) -> Bool
```

```
//      set text black and white reverse
//
// handle
//      Port handle, returned by OpenXXX
//
// nWhiteOnBlack
//      black and white reverse, values defined as follow:
//      value define
//      0      print text normal
//      1      print text black and white reverse
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_SetTextRotate

set text rotate 90 print

### Syntax

public func CP\_Pos\_SetTextRotate(handle: Int, nRotate: Int) -> Bool

```
//      set text rotate 90 print
//
// handle
//      Port handle, returned by OpenXXX
//
// nRotate
//      set text rotate, value defined as follow:
//      value define
//      0      print normal
//      1      text print rotate 90 degree
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_SetTextLineHeight

set line height

### Syntax

```
public func CP_Pos_SetTextLineHeight(handle: Int, nLineHeight: Int) -> Bool
```

```
//      set line height
//
// handle
//      Port handle, returned by OpenXXX
//
// nLineHeight
//      line height, value range is [1,255]
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_SetAsciiTextCharRightSpacing

set ascii text char right spacing

### Syntax

```
public func CP_Pos_SetAsciiTextCharRightSpacing(handle: Int, nSpacing: Int) -> Bool
```

```
//      set ascii text char right spacing
//
// handle
//      Port handle, returned by OpenXXX
//
// nSpacing
//      right spacing, range is [1,255]
//
// return
//      If command is written successfully, it returns true else it returns false.
```



## CP\_Pos\_SetKanjiTextCharSpacing

set kanji text char left spacing and right spacing

### Syntax

```
public func CP_Pos_SetKanjiTextCharSpacing(handle: Int, nLeftSpacing: Int, nRightSpacing: Int) -> Bool
```

```
//      set kanji text char left spacing and right spacing
//
// handle
//      Port handle, returned by OpenXXX
//
// nLeftSpacing
//      left spacing, range is [1,255]
//
// nRightSpacing
//      right spacing, range is [1,255]
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_SetBarcodeUnitWidth

set barcode and qrcode unit width

### Syntax

```
public func CP_Pos_SetBarcodeUnitWidth(handle: Int, nBarcodeUnitWidth: Int) -> Bool
```

```
//      set barcode and qrcode unit width
//
// handle
//      Port handle, returned by OpenXXX
//
// nBarcodeUnitWidth
//      It assigns the code basic element width. range is [2,6]
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_SetBarcodeHeight

set barcode height

### Syntax

```
public func CP_Pos_SetBarcodeHeight(handle: Int, nBarcodeHeight: Int) -> Bool
```

```
//      set barcode height
//
// handle
//      Port handle, returned by OpenXXX
//
// nBarcodeHeight
//      Barcode height, range is [1,255]
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_SetBarcodeReadableTextFontType

set barcode readable text font type

### Syntax

```
public func CP_Pos_SetBarcodeReadableTextFontType(handle: Int, nFontType: Int) -> Bool
```

```
//      set barcode readable text font type
//
// handle
//      Port handle, returned by OpenXXX
//
// nFontType
//      It assigns HRI(Human Readable Interpretation) character font types.
//      value type
//      0      standard ASCII
//      1      small ASCII
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Pos\_SetBarcodeReadableTextPosition

set barcode readable text print position

### Syntax

public func CP\_Pos\_SetBarcodeReadableTextPosition(handle: Int, nTextPosition: CP\_Pos\_BarcodeTextPrintPosition) -> Bool

```
//      set barcode readable text print position
//
// handle
//      Port handle, returned by OpenXXX
//
// nTextPosition
//      barcode readable text position, value range is [0, 3].
//      value defined as follow:
//      value define
//      0      don't show readable text
//      1      show readable text below barcode
//      2      show readable text above barcode
//      3      show readable text above and below barcode
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## Page Mode Function

### CP\_Page\_SelectPageMode

select page mode

#### Syntax

```
public func CP_Page_SelectPageMode(handle: Int) -> Bool
```

```
//      select page mode
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Page\_SelectPageModeEx

select page mode and set movement unit , page area.and other params to default value.

### Syntax

```
public func CP_Page_SelectPageModeEx(handle: Int, nHorizontalMovementUnit: Int, nVerticalMovementUnit: Int, x: Int,
y: Int, width: Int, height: Int) -> Bool
```

```
//      select page mode and set movement unit , page area.and other params to default value.
//
// handle
//      Port handle, returned by OpenXXX
//
// nHorizontalMovementUnit
//      horizontal movement unit
//
// nVerticalMovementUnit
//      vertical movement unit
//
// x
//      horizontal start position
//
// y
//      vertical start position
//
// width
//      print area width
//
// height
//      print area height
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Page\_ExitPageMode

exit page mode and enter standard mode

### Syntax

```
public func CP_Page_ExitPageMode(handle: Int) -> Bool
```

```
//      exit page mode and enter standard mode
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      If command is written successfully, it returns true else it returns false.
//      none
```



## CP\_Page\_PrintPage

print page in pagemode

### Syntax

```
public func CP_Page_PrintPage(handle: Int) -> Bool
```

```
//      print page in pagemode
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Page\_ClearPage

clear page in pagemode

### Syntax

```
public func CP_Page_ClearPage(handle: Int) -> Bool
```

```
//      clear page in pagemode
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Page\_SetPageArea

set page area in pagemode, max height is 2000(8 dot per mm)

### Syntax

public func CP\_Page\_SetPageArea(handle: Int, x: Int, y: Int, width: Int, height: Int) -> Bool

```
//      set page area in pagemode, max height is 2000(8 dot per mm)
//
// handle
//      Port handle, returned by OpenXXX
//
// x
//      horizontal start position
//
// y
//      vertical start position
//
// width
//      print area width
//
// height
//      print area height
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Page\_SetPageDrawDirection

set print direction in pagemode

### Syntax

public func CP\_Page\_SetPageDrawDirection(handle: Int, nDirection: CP\_Page\_DrawDirection) -> Bool

```
//      set print direction in pagemode
//
// handle
//      Port handle, returned by OpenXXX
//
// nDirection
//      print area direction
//      0      left -> right
//      1      bottom -> top
//      2      right -> left
//      3      top -> bottom
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Page\_DrawRect

draw rect in pagemode

### Syntax

public func CP\_Page\_DrawRect(handle: Int, x: Int, y: Int, width: Int, height: Int, color: Int) -> Bool

```
//      draw rect in pagemode
//
// handle
//      Port handle, returned by OpenXXX
//
// x
//      horizontal position
//
// y
//      vertical position
//
// width
//      rect width
//
// height
//      rect height
//
// color
//      rect color, 0 means white, 1 means black.
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Page\_DrawBox

draw box in pagemode

### Syntax

public func CP\_Page\_DrawBox(handle: Int, x: Int, y: Int, width: Int, height: Int, borderwidth: Int, bordercolor: Int)  
→ Bool

```
//      draw box in pagemode
//
// handle
//      Port handle, returned by OpenXXX
//
// x
//      horizontal position
//
// y
//      vertical position
//
// width
//      box width
//
// height
//      box height
//
// borderwidth
//      box border width
//
// bordercolor
//      box border color, 0 means white, 1 means black.
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Page\_DrawText

draw text in pagemode

### Syntax

```
public func CP_Page_DrawText(handle: Int, x: Int, y: Int, str: String) -> Bool
```

```
//      draw text in pagemode
//
// handle
//      Port handle, returned by OpenXXX
//
// x
//      horizontal position
//
// y
//      vertical position
//
// str
//      the string to print
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Page\_DrawTextInUTF8

draw text in pagemode

### Syntax

```
public func CP_Page_DrawTextInUTF8(handle: Int, x: Int, y: Int, str: String) -> Bool
```

```
//      draw text in pagemode
//
// handle
//      Port handle, returned by OpenXXX
//
// x
//      horizontal position
//
// y
//      vertical position
//
// str
//      the string to print
//
// return
//      If command is written successfully, it returns true else it returns false.
//
// remarks
//      The function converts the data to UTF8 encoding.
```



## CP\_Page\_DrawTextInGBK

draw text in pagemode

### Syntax

```
public func CP_Page_DrawTextInGBK(handle: Int, x: Int, y: Int, str: String) -> Bool
```

```
//      draw text in pagemode
//
// handle
//      Port handle, returned by OpenXXX
//
// x
//      horizontal position
//
// y
//      vertical position
//
// str
//      the string to print
//
// return
//      If command is written successfully, it returns true else it returns false.
//
// remarks
//      The function converts the data to GBK encoding.
```

## CP\_Page\_DrawTextInBIG5

draw text in pagemode

### Syntax

```
public func CP_Page_DrawTextInBIG5(handle: Int, x: Int, y: Int, str: String) -> Bool
```

```
//      draw text in pagemode
//
// handle
//      Port handle, returned by OpenXXX
//
// x
//      horizontal position
//
// y
//      vertical position
//
// str
//      the string to print
//
// return
//      If command is written successfully, it returns true else it returns false.
//
// remarks
//      The function converts the data to BIG5 encoding.
```

## CP\_Page\_DrawTextInShiftJIS

draw text in pagemode

### Syntax

```
public func CP_Page_DrawTextInShiftJIS(handle: Int, x: Int, y: Int, str: String) -> Bool
```

```
//      draw text in pagemode
//
// handle
//      Port handle, returned by OpenXXX
//
// x
//      horizontal position
//
// y
//      vertical position
//
// str
//      the string to print
//
// return
//      If command is written successfully, it returns true else it returns false.
//
// remarks
//      The function converts the data to ShiftJIS encoding.
```

## CP\_Page\_DrawTextInEUCKR

draw text in pagemode

### Syntax

```
public func CP_Page_DrawTextInEUCKR(handle: Int, x: Int, y: Int, str: String) -> Bool
```

```
//      draw text in pagemode
//
// handle
//      Port handle, returned by OpenXXX
//
// x
//      horizontal position
//
// y
//      vertical position
//
// str
//      the string to print
//
// return
//      If command is written successfully, it returns true else it returns false.
//
// remarks
//      The function converts the data to EUCKR encoding.
```

## CP\_Page\_DrawBarcode

print 1D barcode in pagemode

### Syntax

public func CP\_Page\_DrawBarcode(handle: Int, x: Int, y: Int, nBarcodeType: CP\_Pos\_BarcodeType, str: String) -> Bool

```
//      print 1D barcode in pagemode
//
// handle
//      Port handle, returned by OpenXXX
//
// x
//      horizontal position
//
// y
//      vertical position
//
// nBarcodeType
//      barcode type
//      values are defined as follow:
//      value      type
//      0x41      UPC-A
//      0x42      UPC-E
//      0x43      EAN13
//      0x44      EAN8
//      0x45      CODE39
//      0x46      ITF
//      0x47      CODABAR
//      0x48      CODE93
//      0x49      CODE128
//
// str
//      the barcode data to print
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Page\_DrawQRCode

print qrcode in pagemode

### Syntax

public func CP\_Page\_DrawQRCode(handle: Int, x: Int, y: Int, nVersion: Int, nECCLevel: CP\_QRCodeECC, str: String)  
→ Bool

```
//      print qrcode in pagemode
//
// handle
//      Port handle, returned by OpenXXX
//
// x
//      horizontal position
//
// y
//      vertical position
//
// nVersion
//      Assign charater version. The value range is:[0,16]
//      When version is 0, printer caculates version number according to character set automatically.
//
// nECCLevel
//      Assign error correction level.
//      The value range is: [1, 4].
//      Definitios are as below:
//      ECC error correction level
//      1  L:7%, low error correction, much data.
//      2  M:15%, medium error correction
//      3  Q:optimize error correction
//      4  H:30%, the highest error correction, less data.
//
// str
//      the qrcode data to print
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Page\_DrawRasterImageFromFile

print image in pagemode

### Syntax

public func CP\_Page\_DrawRasterImageFromFile(handle: Int, x: Int, y: Int, dstw: Int, dsth: Int, pszFile: String, binarization\_method: CP\_ImageBinarizationMethod) -> Bool

```
//      print image in pagemode
//
// handle
//      Port handle, returned by OpenXXX
//
// x
//      horizontal position
//
// y
//      vertical position
//
// dstw
//      the width to print
//
// dsth
//      the height to print
//
// pszFile
//      image file path
//
// binarization_method
//      image binarization method. 0 means use dithering, 1 means use thresholding, 2 means use error diffusion.
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Page\_DrawRasterImageFromData

print image in pagemode(data can be readed from file)

### Syntax

public func CP\_Page\_DrawRasterImageFromData(handle: Int, x: Int, y: Int, dstw: Int, dsth: Int, data: [UInt8], data\_size: Int, binarization\_method: CP\_ImageBinarizationMethod) -> Bool

```
//      print image in pagemode(data can be readed from file)
//
// handle
//      Port handle, returned by OpenXXX
//
// x
//      horizontal position
//
// y
//      vertical position
//
// dstw
//      the width to print
//
// dsth
//      the height to print
//
// data
//      image data
//
// data_size
//      image data size
//
// binarization_method
//      image binarization method. 0 means use dithering, 1 means use thresholding, 2 means use error diffusion.
//
// return
//      If command is written successfully, it returns true else it returns false.
```



## CP\_Page\_DrawRasterImageFromPixels

print image pixels in pagemode

### Syntax

```
public func CP_Page_DrawRasterImageFromPixels(handle: Int, x: Int, y: Int, img_data: [UInt8], img_datalen: Int,
img_width: Int, img_height: Int, img_stride: Int, img_format: CP_ImagePixelsFormat, binarization_method:
CP_ImageBinarizationMethod) -> Bool
```

```
//      print image pixels in pagemode
//
// handle
//      Port handle, returned by OpenXXX
//
// x
//      horizontal position
//
// y
//      vertical position
//
// img_data
//      image pixels data
//
// img_datalen
//      image pixels data length
//
// img_width
//      image pixel width
//
// img_height
//      image pixel height
//
// img_stride
//      image horizontal stirde. means bytes per line.
//
// img_format
//      image pixel data format, values are defined as follow
//      value define
//      1      mono
//      2      monolsb
//      3      gray
//      4      r.g.b in byte-ordered
//      5      b.g.r in byte-ordered
//      6      a.r.g.b in byte-ordered
//      7      r.g.b.a in byte-ordered
```

```
//      8      a.b.g.r in byte-ordered
//      9      b.g.r.a in byte-ordered
//
// binaryzation_method
//      image binaryzation method. 0 means use dithering, 1 means use thresholding, 2 means use error diffusion.
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## Black Marker Function

### CP\_BlackMark\_EnableBlackMarkMode

enable black mark mode, need reboot printer.

#### Syntax

```
public func CP_BlackMark_EnableBlackMarkMode(handle: Int) -> Bool
```

```
//      enable black mark mode, need reboot printer.
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_BlackMark\_DisableBlackMarkMode

disable black mark mode

### Syntax

```
public func CP_BlackMark_DisableBlackMarkMode(handle: Int) -> Bool
```

```
//      disable black mark mode
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_BlackMark\_SetBlackMarkMaxFindLength

set black mark max search length(reboot will also valid)

### Syntax

public func CP\_BlackMark\_SetBlackMarkMaxFindLength(handle: Int, maxFindLength: Int) -> Bool

```
//      set black mark max search length(reboot will also valid)
//
// handle
//      Port handle, returned by OpenXXX
//
// maxFindLength
//      max find length (maxFindLength x 0.125 mm)
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_BlackMark\_FindNextBlackMark

find next black mark

### Syntax

```
public func CP_BlackMark_FindNextBlackMark(handle: Int) -> Bool
```

```
//      find next black mark
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_BlackMark\_SetBlackMarkPaperPrintPosition

in black mode, set start print position

### Syntax

```
public func CP_BlackMark_SetBlackMarkPaperPrintPosition(handle: Int, position: Int) -> Bool
```

```
//      in black mode, set start print position
//
// handle
//      Port handle, returned by OpenXXX
//
// position
//      position > 0 means feed, position < 0 means feedback. distance is position x 0.125 mm.
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_BlackMark\_SetBlackMarkPaperCutPosition

in black mark mode, set cut position

### Syntax

```
public func CP_BlackMark_SetBlackMarkPaperCutPosition(handle: Int, position: Int) -> Bool
```

```
//      in black mark mode, set cut position
//
// handle
//      Port handle, returned by OpenXXX
//
// position
//      position > 0 means feed, position < 0 means feedback. distance is position x 0.125 mm.
//
// return
//      If command is written successfully, it returns true else it returns false.
```



## CP\_BlackMark\_FullCutBlackMarkPaper

full cut paper

### Syntax

```
public func CP_BlackMark_FullCutBlackMarkPaper(handle: Int) -> Bool
```

```
//      full cut paper
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_BlackMark\_HalfCutBlackMarkPaper

half cut paper

### Syntax

```
public func CP_BlackMark_HalfCutBlackMarkPaper(handle: Int) -> Bool
```

```
//      half cut paper
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## Label Function

### CP\_Label\_EnableLabelMode

enable label mode

#### Syntax

```
public func CP_Label_EnableLabelMode(handle: Int) -> Bool
```

```
//      enable label mode
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Label\_DisableLabelMode

disable label mode

### Syntax

```
public func CP_Label_DisableLabelMode(handle: Int) -> Bool
```

```
//      disable label mode
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Label\_CalibrateLabel

calibrate label paper(change to different label paper, need calibration)

### Syntax

```
public func CP_Label_CalibrateLabel(handle: Int) -> Bool
```

```
//      calibrate label paper(change to different label paper, need calibration)
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Label\_FeedLabel

Feed paper to gap

### Syntax

```
public func CP_Label_FeedLabel(handle: Int) -> Bool
```

```
//      Feed paper to gap
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Label\_BackPaperToPrintPosition

printer feed paper back to print position (for label printing starts positioning)

### Syntax

```
public func CP_Label_BackPaperToPrintPosition(handle: Int) -> Bool
```

```
//      printer feed paper back to print position (for label printing starts positioning)
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Label\_FeedPaperToTearPosition

printer feed paper to tear position (for label printing ends positioning)

### Syntax

```
public func CP_Label_FeedPaperToTearPosition(handle: Int) -> Bool
```

```
//      printer feed paper to tear position (for label printing ends positioning)
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      If command is written successfully, it returns true else it returns false.
```



## CP\_Label\_PageBegin

assign the start of a label page, and set Page size, reference point coordinates and page rotation.

### Syntax

public func CP\_Label\_PageBegin(handle: Int, x: Int, y: Int, width: Int, height: Int, rotation: CP\_Label\_Rotation) —> Bool

```
//      assign the start of a label page, and set Page size, reference point coordinates and page rotation.
//
// handle
//      Port handle, returned by OpenXXX
//
// x
//      page start point x coordinates
//
// y
//      page start point y coordinates
//
// width
//      page width
//
// height
//      page height
//
// rotation
//      page rotating. The value range of rotate is {0,1}. Page doesn't rotate to print as 0, and rotate 90 degree to
print as 1.
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Label\_PagePrint

print the label page contents to label paper

### Syntax

```
public func CP_Label_PagePrint(handle: Int, copies: Int) -> Bool
```

```
//      print the label page contents to label paper
//
// handle
//      Port handle, returned by OpenXXX
//
// copies
//      Copies [ 1 - 255 ]
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Label\_DrawText

draw text in assigned position of label page.only for single line

### Syntax

public func CP\_Label\_DrawText(handle: Int, x: Int, y: Int, font: Int, style: Int, str: String) -> Bool

```
//      draw text in assigned position of label page.only for single line
//
// handle
//      Port handle, returned by OpenXXX
//
// x
//      define text start position x coordinates,the value range is: [0,Page_Width-1]
//
// y
//      define text start position y coordinates,the value range is: [0,Page_Height-1]
//
// font
//      Choose font, can use 24.
//      some printer can use 16, [20,99].
//
// style
//      chracter style.
//      Databits          define
//      0 Bold flag bit:   font bold for 1,don't bold if reset zero clearing.
//      1 underline flag bit: underline text for 1, don't underline if rest zero clearing
//      2 inverse flag bit: inverse for 1(white in black), don't inverse rest zero clearing
//      3 delete line flage bit:      for 1 text with delete line,don't delete line if reset zero clearing.
//      [5,4] rotate flag bit:      00 rotates 0 degree
//                                  01 rotates 90 degree
//                                  10 rotates 180 degree
//                                  11 rotates 270 degree
//      [11,8] font width magnification times;
//      [15,12] font height magnification times;
//
// str
//      the string to print
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Label\_DrawTextInUTF8

draw text in assigned position of label page.only for single line

### Syntax

public func CP\_Label\_DrawTextInUTF8(handle: Int, x: Int, y: Int, font: Int, style: Int, str: String) -> Bool

```
//      draw text in assigned position of label page.only for single line
//
// handle
//      Port handle, returned by OpenXXX
//
// x
//      define text start position x coordinates,the value range is: [0,Page_Width-1]
//
// y
//      define text start position y coordinates,the value range is: [0,Page_Height-1]
//
// font
//      Choose font, can use 24.
//      some printer can use 16, [20,99].
//
// style
//      chracter style.
//      Databits          define
//      0 Bold flag bit:      font bold for 1,don't bold if reset zero clearing.
//      1 underline flag bit:  underline text for 1, don't underline if rest zero clearing
//      2 inverse flag bit:    inverse for 1(white in black), don't inverse rest zero clearing
//      3 delete line flage bit: for 1 text with delete line,don't delete line if reset zero clearing.
//      [5,4] rotate flag bit: 00 rotates 0 degree
//                               01 rotates 90 degree
//                               10 rotates 180 degree
//                               11 rotates 270 degree
//      [11,8] font width magnification times;
//      [15,12] font height magnification times;
//
// str
//      the string to print
//
// return
//      If command is written successfully, it returns true else it returns false.
//
// remarks
//      The function converts the data to UTF8 encoding.
```

## CP\_Label\_DrawTextInGBK

draw text in assigned position of label page.only for single line

### Syntax

public func CP\_Label\_DrawTextInGBK(handle: Int, x: Int, y: Int, font: Int, style: Int, str: String) -> Bool

```
//      draw text in assigned position of label page.only for single line
//
// handle
//      Port handle, returned by OpenXXX
//
// x
//      define text start position x coordinates,the value range is: [0,Page_Width-1]
//
// y
//      define text start position y coordinates,the value range is: [0,Page_Height-1]
//
// font
//      Choose font, can use 24.
//      some printer can use 16, [20,99].
//
// style
//      chracter style.
//      Databits          define
//      0 Bold flag bit:      font bold for 1,don't bold if reset zero clearing.
//      1 underline flag bit:  underline text for 1, don't underline if rest zero clearing
//      2 inverse flag bit:    inverse for 1(white in black), don't inverse rest zero clearing
//      3 delete line flage bit: for 1 text with delete line,don't delete line if reset zero clearing.
//      [5,4] rotate flag bit: 00 rotates 0 degree
//                               01 rotates 90 degree
//                               10 rotates 180 degree
//                               11 rotates 270 degree
//      [11,8] font width magnification times;
//      [15,12] font height magnification times;
//
// str
//      the string to print
//
// return
//      If command is written successfully, it returns true else it returns false.
//
// remarks
//      The function converts the data to GBK encoding.
```

## CP\_Label\_DrawTextInBIG5

draw text in assigned position of label page.only for single line

### Syntax

public func CP\_Label\_DrawTextInBIG5(handle: Int, x: Int, y: Int, font: Int, style: Int, str: String) -> Bool

```
//      draw text in assigned position of label page.only for single line
//
// handle
//      Port handle, returned by OpenXXX
//
// x
//      define text start position x coordinates,the value range is: [0,Page_Width-1]
//
// y
//      define text start position y coordinates,the value range is: [0,Page_Height-1]
//
// font
//      Choose font, can use 24.
//      some printer can use 16, [20,99].
//
// style
//      chracter style.
//      Databits          define
//      0 Bold flag bit:      font bold for 1,don't bold if reset zero clearing.
//      1 underline flag bit:  underline text for 1, don't underline if rest zero clearing
//      2 inverse flag bit:    inverse for 1(white in black), don't inverse rest zero clearing
//      3 delete line flage bit: for 1 text with delete line,don't delete line if reset zero clearing.
//      [5,4] rotate flag bit: 00 rotates 0 degree
//                               01 rotates 90 degree
//                               10 rotates 180 degree
//                               11 rotates 270 degree
//      [11,8] font width magnification times;
//      [15,12] font height magnification times;
//
// str
//      the string to print
//
// return
//      If command is written successfully, it returns true else it returns false.
//
// remarks
//      The function converts the data to BIG5 encoding.
```

## CP\_Label\_DrawTextInShiftJIS

draw text in assigned position of label page.only for single line

### Syntax

public func CP\_Label\_DrawTextInShiftJIS(handle: Int, x: Int, y: Int, font: Int, style: Int, str: String) -> Bool

```
//      draw text in assigned position of label page.only for single line
//
// handle
//      Port handle, returned by OpenXXX
//
// x
//      define text start position x coordinates,the value range is: [0,Page_Width-1]
//
// y
//      define text start position y coordinates,the value range is: [0,Page_Height-1]
//
// font
//      Choose font, can use 24.
//      some printer can use 16, [20,99].
//
// style
//      chracter style.
//      Databits          define
//      0 Bold flag bit:      font bold for 1,don't bold if reset zero clearing.
//      1 underline flag bit:  underline text for 1, don't underline if rest zero clearing
//      2 inverse flag bit:    inverse for 1(white in black), don't inverse rest zero clearing
//      3 delete line flage bit: for 1 text with delete line,don't delete line if reset zero clearing.
//      [5,4] rotate flag bit: 00 rotates 0 degree
//                               01 rotates 90 degree
//                               10 rotates 180 degree
//                               11 rotates 270 degree
//      [11,8] font width magnification times;
//      [15,12] font height magnification times;
//
// str
//      the string to print
//
// return
//      If command is written successfully, it returns true else it returns false.
//
// remarks
//      The function converts the data to ShiftJIS encoding.
```

## CP\_Label\_DrawTextInEUCKR

draw text in assigned position of label page.only for single line

### Syntax

public func CP\_Label\_DrawTextInEUCKR(handle: Int, x: Int, y: Int, font: Int, style: Int, str: String) -> Bool

```
//      draw text in assigned position of label page.only for single line
//
// handle
//      Port handle, returned by OpenXXX
//
// x
//      define text start position x coordinates,the value range is: [0,Page_Width-1]
//
// y
//      define text start position y coordinates,the value range is: [0,Page_Height-1]
//
// font
//      Choose font, can use 24.
//      some printer can use 16, [20,99].
//
// style
//      chracter style.
//      Databits          define
//      0 Bold flag bit:      font bold for 1,don't bold if reset zero clearing.
//      1 underline flag bit:  underline text for 1, don't underline if rest zero clearing
//      2 inverse flag bit:    inverse for 1(white in black), don't inverse rest zero clearing
//      3 delete line flage bit: for 1 text with delete line,don't delete line if reset zero clearing.
//      [5,4] rotate flag bit: 00 rotates 0 degree
//                               01 rotates 90 degree
//                               10 rotates 180 degree
//                               11 rotates 270 degree
//      [11,8] font width magnification times;
//      [15,12] font height magnification times;
//
// str
//      the string to print
//
// return
//      If command is written successfully, it returns true else it returns false.
//
// remarks
//      The function converts the data to EUCKR encoding.
```



## CP\_Label\_DrawBarcode

Draw 1D code in the assigned position of label page

### Syntax

```
public func CP_Label_DrawBarcode(handle: Int, x: Int, y: Int, nBarcodeType: CP_Label_BarcodeType,
nBarcodeTextPrintPosition: CP_Label_BarcodeTextPrintPosition, height: Int, unitwidth: Int, rotation: CP_Label_Rotation,
str: String) -> Bool
```

```
//      Draw 1D code in the assigned position of label page
//
// handle
//      Port handle, returned by OpenXXX
//
// x
//      barcode top left corner x coordinates, the value range is: [0,Page_Width-1]
//
// y
//      barcode top left corner y coordinates, the value range is: [0,Page_Height-1]
//
// nBarcodeType
//      barcode type
//      values are defined as macros
//
// nBarcodeTextPrintPosition
//      barcode readable text position, value range is [0, 3].
//      value defined as follow:
//      value define
//      0      don't show readable text
//      1      show readable text below barcode
//      2      show readable text above barcode
//      3      show readable text above and below barcode
//
// height
//      define barcode height
//
// unitwidth
//      It assigns the basic element width. value range is [1, 4].
//
// rotation
//      Mean rotating angle,
//      the value range is: [0, 3].
//      Definitions are as below:
//      Rotate value define
//      0 doesn't rotate to draw
```

```
//      1 rotates 90 degree draw .
//      2 rotates 180 degree draw .
//      3 rotates 270 degree draw
//
// str
//      the barcode data to print
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Label\_DrawQRCode

print qrcode in the assigned position of label page

### Syntax

public func CP\_Label\_DrawQRCode(handle: Int, x: Int, y: Int, nVersion: Int, nECCLevel: CP\_QRCodeECC, unitwidth: Int, rotation: CP\_Label\_Rotation, str: String) -> Bool

```
//      print qrcode in the assigned position of label page
//
// handle
//      Port handle, returned by OpenXXX
//
// x
//      Top left corner x coordinates, the value range is: [0,Page_Width-1]
//
// y
//      Top left corner y coordinates, the value range is: [0,Page_Height-1]
//
// nVersion
//      Assign charater version. The value range is:[0,16]
//      When version is 0, printer caculates version number according to character set automatically.
//
// nECCLevel
//      Assign error correction level.
//      The value range is: [1, 4].
//      Definitios are as below:
//      ECC error correction level
//      1  L: 7%, low error correction, much data.
//      2  M: 15%, medium error correction
//      3  Q: optimize error correction
//      4  H: 30%, the highest error correction, less data.
//
// unitwidth
//      It assigns the basic element width. value range is [1, 4].
//
// rotation
//      Mean rotating angle,
//      the value range is: [0, 3].
//      Definitions are as below:
//      Rotate value define
//      0 doesn't rotate to draw
//      1 rotates 90 degree draw.
//      2 rotates 180 degree draw.
//      3 rotates 270 degree draw
```

```
//  
// str  
// the qrcode data to print  
//  
// return  
// If command is written successfully, it returns true else it returns false.
```

## CP\_Label\_DrawPDF417Code

print pdf417code in the assigned position of label page

### Syntax

public func CP\_Label\_DrawPDF417Code(handle: Int, x: Int, y: Int, column: Int, nAspectRatio: Int, nECCLevel: Int, unitwidth: Int, rotation: CP\_Label\_Rotation, str: String) -> Bool

```
//      print pdf417code in the assigned position of label page
//
// handle
//      Port handle, returned by OpenXXX
//
// x
//      Top left corner x coordinates, the value range is: [0,Page_Width-1]
//
// y
//      Top left corner y coordinates, the value range is: [0,Page_Height-1]
//
// column
//      ColNum is colnum, which means how many digits in per line. A digit is 17*UnitWidth dots. Line number
//      is produces automatically by printer,the limited range is 3~90. ColNum value range:[1,30].
//
// nECCLevel
//      Assign error correction level.
//      The value range is: [0, 8].
//      Ecc value, error correction number, stored files number(byte)
//      0 2 1108
//      1 4 1106
//      2 8 1101
//      3 16 1092
//      4 32 1072
//      5 64 1024
//      6 128 957
//      7 256 804
//      8 512 496
//
// unitwidth
//      It assigns the basic element width. value range is [1, 3].
//
// rotation
//      Mean rotating angle,
//      the value range is: [0, 3].
//      Definitions are as below:
//      Rotate value define
```

```
//      0 doesn't rotate to draw
//      1 rotates 90 degree draw.
//      2 rotates 180 degree draw.
//      3 rotates 270 degree draw
//
// str
//      the pdf417 data to print
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Label\_DrawImageFromFile

Draw picture in the assigned position of label page

### Syntax

```
public func CP_Label_DrawImageFromFile(handle: Int, x: Int, y: Int, dstw: Int, dsth: Int, pszFile: String,  
binarization_method: CP_ImageBinarizationMethod, compression_method: CP_ImageCompressionMethod) -> Bool
```

```
//      Draw picture in the assigned position of label page  
//  
// handle  
//      Port handle, returned by OpenXXX  
//  
// x  
//      Top left corner x coordinates, the value range is: [0,Page_Width-1]  
//  
// y  
//      Top left corner y coordinates, the value range is: [0,Page_Height-1]  
//  
// dstw  
//      the width to print  
//  
// dsth  
//      the height to print  
//  
// pszFile  
//      image file path  
//  
// binaryzation_method  
//      image binaryzation method. 0 means use dithering, 1 means use thresholding, 2 means use error diffusion.  
//  
// compression_method  
//      print data compress method, values are defined as follow  
//      value define  
//      0      no compress  
//      1      compress level 1  
//      2      compress level 2  
//  
// return  
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Label\_DrawImageFromData

Draw picture in the assigned position of label page

### Syntax

public func CP\_Label\_DrawImageFromData(handle: Int, x: Int, y: Int, dstw: Int, dsth: Int, data: [UInt8], data\_size: Int, binarization\_method: CP\_ImageBinarizationMethod, compression\_method: CP\_ImageCompressionMethod) -> Bool

```
//      Draw picture in the assigned position of label page
//
// handle
//      Port handle, returned by OpenXXX
//
// x
//      Top left corner x coordinates, the value range is: [0,Page_Width-1]
//
// y
//      Top left corner y coordinates, the value range is: [0,Page_Height-1]
//
// dstw
//      the width to print
//
// dsth
//      the height to print
//
// data
//      image data
//
// data_size
//      image data size
//
// binarization_method
//      image binarization method. 0 means use dithering, 1 means use thresholding, 2 means use error diffusion.
//
// compression_method
//      print data compress method, values are defined as follow
//      value define
//      0      no compress
//      1      compress level 1
//      2      compress level 2
//
// return
//      If command is written successfully, it returns true else it returns false.
```



## CP\_Label\_DrawImageFromPixels

Draw picture in the assigned position of label page

### Syntax

```
public func CP_Label_DrawImageFromPixels(handle: Int, x: Int, y: Int, img_data: [UInt8], img_datalen: Int, img_width: Int, img_height: Int, img_stride: Int, img_format: CP_ImagePixelFormat, binarization_method: CP_ImageBinarizationMethod, compression_method: CP_ImageCompressionMethod) -> Bool
```

```
//      Draw picture in the assigned position of label page
//
// handle
//      Port handle, returned by OpenXXX
//
// x
//      Top left corner x coordinates, the value range is: [0,Page_Width-1]
//
// y
//      Top left corner y coordinates, the value range is: [0,Page_Height-1]
//
// img_data
//      image pixels data
//
// img_datalen
//      image pixels data length
//
// img_width
//      image pixel width
//
// img_height
//      image pixel height
//
// img_stride
//      image horizontal stirde. means bytes per line.
//
// img_format
//      image pixel data format, values are defined as follow
//      value define
//      1      mono
//      2      monolsb
//      3      gray
//      4      r.g.b in byte-ordered
//      5      b.g.r in byte-ordered
//      6      a.r.g.b in byte-ordered
//      7      r.g.b.a in byte-ordered
```

```
//      8      a.b.g.r in byte-ordered
//      9      b.g.r.a in byte-ordered
//
// binaryzation_method
//      image binaryzation method. 0 means use dithering, 1 means use thresholding, 2 means use error diffusion.
//
// compression_method
//      print data compress method, values are defined as follow
//      value define
//      0      no compress
//      1      compress level 1
//      2      compress level 2
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Label\_DrawLine

draw line in the assigned position of label page

### Syntax

public func CP\_Label\_DrawLine(handle: Int, startx: Int, starty: Int, endx: Int, endy: Int, linewidth: Int, linecolor: CP\_Label\_Color) -> Bool

```
//      draw line in the assigned position of label page
//
// handle
//      Port handle, returned by OpenXXX
//
// startx
//      straightway start point x coordinates,the value range is: [0,Page_Width-1]
//
// starty
//      straightway start point y coordinates,the value range is: [0,Page_Height-1]
//
// endx
//      straightway end point x coordinates,the value range is: [0,Page_Width-1]
//
// endy
//      straightway end point y coordinates,the value range is:[0,Page_Height-1]
//
// linewidth
//      line width
//
// linecolor
//      line color, 0 means white, 1 means black.
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Label\_DrawRect

draw rect in the assigned position of label page

### Syntax

public func CP\_Label\_DrawRect(handle: Int, x: Int, y: Int, width: Int, height: Int, color: CP\_Label\_Color) -> Bool

```
//      draw rect in the assigned position of label page
//
// handle
//      Port handle, returned by OpenXXX
//
// x
//      horizontal position
//
// y
//      vertical position
//
// width
//      rect width
//
// height
//      rect height
//
// color
//      rect color, 0 means white, 1 means black.
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Label\_DrawBox

draw box in the assigned position of label page

### Syntax

```
public func CP_Label_DrawBox(handle: Int, x: Int, y: Int, width: Int, height: Int, borderwidth: Int, bordercolor:
CP_Label_Color) -> Bool
```

```
//      draw box in the assigned position of label page
//
// handle
//      Port handle, returned by OpenXXX
//
// x
//      horizontal position
//
// y
//      vertical position
//
// width
//      box width
//
// height
//      box height
//
// borderwidth
//      box border width
//
// bordercolor
//      box border color, 0 means white, 1 means black.
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## Other Function

### CP\_Library\_Version

get library version string

#### Syntax

```
func CP_Library_Version_SwiftFunction() -> String
```

```
//      get library version string
//
//  return
//      return library version string
```

## CP\_Proto\_QueryBatteryLevel

Query battery level

### Syntax

```
public func CP_Proto_QueryBatteryLevel(handle: Int, timeout: Int) -> Int
```

```
//      Query battery level
//      Only some models with batteries support this command
//
// handle
//      Port handle, returned by OpenXXX
//
// timeout
//      timeout ms
//      The wait time for query does not exceed this time
//
// return
//      Returns the battery power, and a range of 0–100. returns -1 to indicate that the query failed.
```

## CP\_Proto\_QuerySerialNumber

Query serial number

### Syntax

```
public func CP_Proto_QuerySerialNumber(handle: Int, timeout: Int) -> String?
```

```
//      Query serial number
//      Only some models support this command
//
// handle
//      Port handle, returned by OpenXXX
//
// timeout
//      timeout ms
//      The wait time for query does not exceed this time
//
// return
//      Returns the serial number
```



## CP\_Proto\_SetSystemNameAndSerialNumber

Set system name and serial number

### Syntax

```
public func CP_Proto_SetSystemNameAndSerialNumber(handle: Int, systemName: String, serialNumber: String) -> Bool
```

```
//      Set system name and serial number
//
// handle
//      Port handle, returned by OpenXXX
//
// systemName
//      system name
//
// serialNumber
//      serial number
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Proto\_SetBluetoothNameAndPassword

Set bluetooth name and bluetooth password

### Syntax

```
public func CP_Proto_SetBluetoothNameAndPassword(handle: Int, bluetoothName: String, bluetoothPassword: String) ->
Bool
```

```
//      Set bluetooth name and bluetooth password
//
// handle
//      Port handle, returned by OpenXXX
//
// bluetoothName
//      bluetooth name
//
// bluetoothPassword
//      bluetooth password
//
// return
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Proto\_SetPTPBasicParameters

Set basic parameters, include codepage,baudrate,density, like printersetting.exe ptp page.

### Syntax

```
public func CP_Proto_SetPTPBasicParameters(handle: Int, baudrate: Int, codepage: Int, density: Int, asciiFontType: Int,
lineFeed: Int, idleTime: Int, powerOffTime: Int, maxFeedLength: Int, pageLength: Int) -> Bool
```

### Parameters

```
//      Set basic parameters, include codepage,baudrate,density, like printersetting.exe ptp page.
//
// handle
//      Port handle, returned by OpenXXX
//
// baudrate
//      the baudrate to set
//
// codepage
//      the codepage to set
//      see following:
//      { ("Simplified Chinese"), 255 },
//      { ("Traditional Chinese"), 254 },
//      { ("UTF - 8"), 253 },
//      { ("SHIFT - JIS"), 252 },
//      { ("EUC - KR"), 251 },
//      { ("CP437[U.S.A., Standard Europe]"), 0 },
//      { ("Katakana"), 1 },
//      { ("CP850[Multilingual]"), 2 },
//      { ("CP860[Portuguese]"), 3 },
//      { ("CP863[Canadian - French]"), 4 },
//      { ("CP865[Nordic]"), 5 },
//      { ("WCP1251[Cyrillic]"), 6 },
//      { ("CP866 Cyrilliec #2"), 7 },
//      { ("MIK[Cyrillic / Bulgarian]"), 8 },
//      { ("CP755[East Europe, Latvian 2]"), 9 },
//      { ("Iran"), 10 },
//      { ("CP862[Hebrew]"), 15 },
//      { ("WCP1252 Latin I"), 16 },
//      { ("WCP1253[Greek]"), 17 },
//      { ("CP852[Latina 2]"), 18 },
//      { ("CP858 Multilingual Latin I + Euro"), 19 },
//      { ("Iran II"), 20 },
//      { ("Latvian"), 21 },
//      { ("CP864[Arabic]"), 22 },
//      { ("ISO - 8859 - 1[West Europe]"), 23 },
```

```

//      { ("CP737[Greek]"), 24 },
//      { ("WCP1257[Baltic]"), 25 },
//      { ("Thai"), 26 },
//      { ("CP720[Arabic]"), 27 },
//      { ("CP855"), 28 },
//      { ("CP857[Turkish]"), 29 },
//      { ("WCP1250[Central Eurpoe]"), 30 },
//      { ("CP775"), 31 },
//      { ("WCP1254[Turkish]"), 32 },
//      { ("WCP1255[Hebrew]"), 33 },
//      { ("WCP1256[Arabic]"), 34 },
//      { ("WCP1258[Vietnam]"), 35 },
//      { ("ISO - 8859 - 2[Latin 2]"), 36 },
//      { ("ISO - 8859 - 3[Latin 3]"), 37 },
//      { ("ISO - 8859 - 4[Baltic]"), 38 },
//      { ("ISO - 8859 - 5[Cyrillic]"), 39 },
//      { ("ISO - 8859 - 6[Arabic]"), 40 },
//      { ("ISO - 8859 - 7[Greek]"), 41 },
//      { ("ISO - 8859 - 8[Hebrew]"), 42 },
//      { ("ISO - 8859 - 9[Turkish]"), 43 },
//      { ("ISO - 8859 - 15[Latin 3]"), 44 },
//      { ("Thai2"), 45 },
//      { ("CP856"), 46 },
//      { ("Cp874"), 47 },
//      { ("Other(Vietnam)"), 48 },
//
// density
//      the density to set
//      0 - Light
//      1 - Normal
//      2 - Dark
//
// asciiFontType
//      the ascii text font type
//      0 - FontA(12x24)
//      1 - FontB(9x24)
//      2 - FontC(9x17)
//      3 - FontD(8x16)
//
// lineFeed
//      the line feed char
//      0 - LF(0x0A)
//      1 - CR(0x0D)
//
// idleTime
//      idle time (seconds)
//
// powerOffTime
//      power off time (seconds)

```

```
//  
// maxFeedLength  
//      max feed length (mm)  
//  
// pageLength  
//      page length (mm)  
//  
// return  
//      If command is written successfully, it returns true else it returns false.
```

## CP\_Settings\_Hardware\_SetPrintSpeed

set print speed

### Syntax

```
public func CP_Settings_Hardware_SetPrintSpeed(handle: Int, nSpeed: Int) -> Bool
```

### Parameters

```
//      set print speed
//
// nSpeed
//      print speed in mm/s
//
// return
//      If command is written successfully, it returns true else it returns false.
```

