

Contents

SDK Introduction	6
Frequently Asked Questions	7
Interface Specification	8
Enum & Macro	8
CP_ComDataBits.....	8
CP_ComParity.....	9
CP_ComStopBits.....	10
CP_ComFlowControl.....	11
CP_CharacterSet	12
CP_CharacterCodepage	13
CP_MultiByteEncoding	15
CP_ImageBinarizationMethod.....	16
CP_ImageCompressionMethod.....	17
CP_ImagePixelsFormat	18
CP_QRCodeECC.....	19
CP_Pos_Alignment	20
CP_Pos_BarcodeType.....	21
CP_Pos_BarcodeTextPrintPosition.....	22
CP_Page_DrawDirection.....	23
CP_Page_DrawAlignment	24
CP_Label_BarcodeType	25
CP_Label_BarcodeTextPrintPosition.....	26
CP_Label_Rotation.....	27
CP_Label_Color	28
CP_PRINTERSTATUS.....	29
CP_RTSTATUS.....	31
CP_LABEL_TEXT_STYLE	33
Callback	34
CP_OnNetPrinterDiscovered	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_AddOnPortClosedEvent	41
CP_Port_AddOnPortWrittenEvent	42
CP_Port_AddOnPortReceivedEvent	43
CP_Port_RemoveOnPortClosedEvent	44
CP_Port_RemoveOnPortWrittenEvent	45
CP_Port_RemoveOnPortReceivedEvent	46
CP_Printer_AddOnPrinterStatusEvent.....	47
CP_Printer_AddOnPrinterReceivedEvent	48
CP_Printer_AddOnPrinterPrintedEvent.....	49

CP_Printer_RemoveOnPrinterStatusEvent	50
CP_Printer_RemoveOnPrinterReceivedEvent	51
CP_Printer_RemoveOnPrinterPrintedEvent	52
Port Function	53
CP_Port_EnumCom	53
CP_Port_EnumLpt	54
CP_Port_EnumUsb	55
CP_Port_EnumNetPrinter	56
CP_Port_OpenCom	57
CP_Port_OpenLpt	59
CP_Port_OpenUsb	60
CP_Port_OpenTcp	61
CP_Port_Write	62
CP_Port_Read	63
CP_Port_ReadUntilByte	64
CP_Port_Available	65
CP_Port_SkipAvailable	66
CP_Port_IsConnectionValid	67
CP_Port_IsOpened	68
CP_Port_Close	69
Get Printer Info Function	70
CP_Printer_GetPrinterResolutionInfo	70
CP_Printer_GetPrinterFirmwareVersion	71
CP_Printer_GetPrinterStatusInfo	72
CP_Printer_GetPrinterReceivedInfo	73
CP_Printer_GetPrinterPrintedInfo	74
CP_Printer_GetPrinterLabelPositionAdjustmentInfo	75
CP_Printer_SetPrinterLabelPositionAdjustmentInfo	76
CP_Printer_ClearPrinterBuffer	77
CP_Printer_ClearPrinterError	78
Pos Function	79
CP_Pos_QueryRTStatus	79
CP_Pos_QueryPrintResult	80
CP_Pos_KickOutDrawer	81
CP_Pos_Beep	82
CP_Pos_FeedAndHalfCutPaper	83
CP_Pos_FullCutPaper	84
CP_Pos_HalfCutPaper	85
CP_Pos_FeedLine	86
CP_Pos_FeedDot	87
CP_Pos_PrintSelfTestPage	88
CP_Pos_PrintText	89
CP_Pos_PrintTextInUTF8	90
CP_Pos_PrintTextInGBK	91
CP_Pos_PrintTextInBIG5	92
CP_Pos_PrintTextInShiftJIS	93
CP_Pos_PrintTextInEUCKR	94
CP_Pos_PrintBarcode	95
CP_Pos_PrintBarcode_Code128Auto	96

CP_Pos_PrintQRCode.....	97
CP_Pos_PrintQRCodeUseEpsonCmd	98
CP_Pos_PrintDoubleQRCode	99
CP_Pos_PrintPDF417BarcodeUseEpsonCmd.....	100
CP_Pos_PrintRasterImageFromFile.....	101
CP_Pos_PrintRasterImageFromData	102
CP_Pos_PrintRasterImageFromPixels.....	103
CP_Pos_PrintHorizontalLine.....	105
CP_Pos_PrintHorizontalLineSpecifyThickness	106
CP_Pos_PrintMultipleHorizontalLinesAtOneRow.....	107
CP_Pos_ResetPrinter	108
CP_Pos_SetPrintSpeed.....	109
CP_Pos_SetPrintDensity.....	110
CP_Pos_SetSingleByteMode	111
CP_Pos_SetCharacterSet	112
CP_Pos_SetCharacterCodepage	113
CP_Pos_SetMultiByteMode.....	114
CP_Pos_SetMultiByteEncoding	115
CP_Pos_SetMovementUnit	116
CP_Pos_SetPrintAreaLeftMargin	117
CP_Pos_SetPrintAreaWidth.....	118
CP_Pos_SetHorizontalAbsolutePrintPosition.....	119
CP_Pos_SetHorizontalRelativePrintPosition.....	120
CP_Pos_SetVerticalAbsolutePrintPosition	121
CP_Pos_SetVerticalRelativePrintPosition	122
CP_Pos_SetAlignment.....	123
CP_Pos_SetTextScale.....	124
CP_Pos_SetAsciiTextFontType.....	125
CP_Pos_SetTextBold.....	126
CP_Pos_SetTextUnderline	127
CP_Pos_SetTextUpsideDown.....	128
CP_Pos_SetTextWhiteOnBlack	129
CP_Pos_SetTextRotate	130
CP_Pos_SetTextLineHeight	131
CP_Pos_SetAsciiTextCharRightSpacing	132
CP_Pos_SetKanjiTextCharSpacing	133
CP_Pos_SetBarcodeUnitWidth	134
CP_Pos_SetBarcodeHeight	135
CP_Pos_SetBarcodeReadableTextFontType	136
CP_Pos_SetBarcodeReadableTextPosition.....	137
Page Mode Function.....	138
CP_Page_SelectPageMode	138
CP_Page_SelectPageModeEx.....	139
CP_Page_ExitPageMode.....	140
CP_Page_PrintPage	141
CP_Page_ClearPage	142
CP_Page_SetPageArea	143
CP_Page_SetPageDrawDirection.....	144
CP_Page_DrawRect	145

CP_Page_DrawBox.....	146
CP_Page_DrawText.....	147
CP_Page_DrawTextInUTF8.....	148
CP_Page_DrawTextInGBK.....	149
CP_Page_DrawTextInBIG5.....	150
CP_Page_DrawTextInShiftJIS.....	151
CP_Page_DrawTextInEUCKR.....	152
CP_Page_DrawBarcode	153
CP_Page_DrawQRCode	154
CP_Page_DrawRasterImageFromFile	155
CP_Page_DrawRasterImageFromData.....	156
CP_Page_DrawRasterImageFromPixels	157
Black Marker Function	159
CP_BlackMark_EnableBlackMarkMode.....	159
CP_BlackMark_DisableBlackMarkMode.....	160
CP_BlackMark_SetBlackMarkMaxFindLength	161
CP_BlackMark_FindNextBlackMark	162
CP_BlackMark_SetBlackMarkPaperPrintPosition.....	163
CP_BlackMark_SetBlackMarkPaperCutPosition.....	164
CP_BlackMark_FullCutBlackMarkPaper.....	165
CP_BlackMark_HalfCutBlackMarkPaper.....	166
Label Function	167
CP_Label_EnableLabelMode.....	167
CP_Label_DisableLabelMode	168
CP_Label_CalibrateLabel	169
CP_Label_FeedLabel.....	170
CP_Label_BackPaperToPrintPosition.....	171
CP_Label_FeedPaperToTearPosition.....	172
CP_Label_PageBegin.....	173
CP_Label_PagePrint.....	174
CP_Label_DrawText.....	175
CP_Label_DrawTextInUTF8	176
CP_Label_DrawTextInGBK.....	177
CP_Label_DrawBarcode.....	178
CP_Label_DrawQRCode	180
CP_Label_DrawPDF417Code.....	182
CP_Label_DrawImageFromFile.....	184
CP_Label_DrawImageFromData	185
CP_Label_DrawImageFromPixels.....	186
CP_Label_DrawLine	188
CP_Label_DrawRect.....	189
CP_Label_DrawBox.....	190
Other Function	191
CP_Library_Version.....	191
CP_Proto_QueryBatteryLevel.....	192
CP_Proto_QuerySerialNumber	193
CP_Proto_SetSystemNameAndSerialNumber.....	194
CP_Proto_SetBluetoothNameAndPassword.....	195
CP_Proto_SetPTPBasicParameters.....	196

CP_Settings_Hardware_SetPrintSpeed	199
--	-----

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 All functions included in class AutoReplyPrint

Engineering code, using autoreplyprint.cs; , 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_ComDataBits

When opening the serial port, it is necessary to specify the data bit, which is generally 8 bits

Syntax

```
public const int CP_ComDataBits_4 = 4;  
public const int CP_ComDataBits_5 = 5;  
public const int CP_ComDataBits_6 = 6;  
public const int CP_ComDataBits_7 = 7;  
public const int CP_ComDataBits_8 = 8;
```


CP_ComParity

When opening the serial port, you need to specify the calibration bit, generally no calibration

Syntax

```
public const int CP_ComParity_NoParity = 0;  
public const int CP_ComParity_OddParity = 1;  
public const int CP_ComParity_EvenParity = 2;  
public const int CP_ComParity_MarkParity = 3;  
public const int CP_ComParity_SpaceParity = 4;
```

CP_ComStopBits

Serial port Stop Bit To open a serial port, you need to specify a stop bit, usually a stop bit

Syntax

```
public const int CP_ComStopBits_One = 0;  
public const int CP_ComStopBits_OnePointFive = 1;  
public const int CP_ComStopBits_Two = 2;
```

CP_ComFlowControl

When opening the serial port, it is necessary to specify the flow control. Generally, no flow control or software flow control or hardware flow control can only be used if the line is connected

Syntax

```
public const int CP_ComFlowControl_None = 0;  
public const int CP_ComFlowControl_XonXoff = 1;  
public const int CP_ComFlowControl_RtsCts = 2;  
public const int CP_ComFlowControl_DtrDsr = 3;
```

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

```
public const int CP_CharacterSet_USA = 0;
public const int CP_CharacterSet_FRANCE = 1;
public const int CP_CharacterSet_GERMANY = 2;
public const int CP_CharacterSet_UK = 3;
public const int CP_CharacterSet_DENMARK_I = 4;
public const int CP_CharacterSet_SWEDEN = 5;
public const int CP_CharacterSet_ITALY = 6;
public const int CP_CharacterSet_SPAIN_I = 7;
public const int CP_CharacterSet_JAPAN = 8;
public const int CP_CharacterSet_NORWAY = 9;
public const int CP_CharacterSet_DENMARK_II = 10;
public const int CP_CharacterSet_SPAIN_II = 11;
public const int CP_CharacterSet_LATIN = 12;
public const int CP_CharacterSet_KOREA = 13;
public const int CP_CharacterSet_SLOVENIA = 14;
public const int CP_CharacterSet_CHINA = 15;
```

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

```
public const int CP_CharacterCodepage_CP437 = 0;
public const int CP_CharacterCodepage_KATAKANA = 1;
public const int CP_CharacterCodepage_CP850 = 2;
public const int CP_CharacterCodepage_CP860 = 3;
public const int CP_CharacterCodepage_CP863 = 4;
public const int CP_CharacterCodepage_CP865 = 5;
public const int CP_CharacterCodepage_WCP1251 = 6;
public const int CP_CharacterCodepage_CP866 = 7;
public const int CP_CharacterCodepage_MIK = 8;
public const int CP_CharacterCodepage_CP755 = 9;
public const int CP_CharacterCodepage_IRAN = 10;
public const int CP_CharacterCodepage_CP862 = 15;
public const int CP_CharacterCodepage_WCP1252 = 16;
public const int CP_CharacterCodepage_WCP1253 = 17;
public const int CP_CharacterCodepage_CP852 = 18;
public const int CP_CharacterCodepage_CP858 = 19;
public const int CP_CharacterCodepage_IRAN_II = 20;
public const int CP_CharacterCodepage_LATVIAN = 21;
public const int CP_CharacterCodepage_CP864 = 22;
public const int CP_CharacterCodepage_ISO_8859_1 = 23;
public const int CP_CharacterCodepage_CP737 = 24;
public const int CP_CharacterCodepage_WCP1257 = 25;
public const int CP_CharacterCodepage_THAI = 26;
public const int CP_CharacterCodepage_CP720 = 27;
public const int CP_CharacterCodepage_CP855 = 28;
public const int CP_CharacterCodepage_CP857 = 29;
public const int CP_CharacterCodepage_WCP1250 = 30;
public const int CP_CharacterCodepage_CP775 = 31;
public const int CP_CharacterCodepage_WCP1254 = 32;
public const int CP_CharacterCodepage_WCP1255 = 33;
public const int CP_CharacterCodepage_WCP1256 = 34;
public const int CP_CharacterCodepage_WCP1258 = 35;
public const int CP_CharacterCodepage_ISO_8859_2 = 36;
public const int CP_CharacterCodepage_ISO_8859_3 = 37;
public const int CP_CharacterCodepage_ISO_8859_4 = 38;
public const int CP_CharacterCodepage_ISO_8859_5 = 39;
public const int CP_CharacterCodepage_ISO_8859_6 = 40;
public const int CP_CharacterCodepage_ISO_8859_7 = 41;
```

```
public const int CP_CharacterCodepage_ISO_8859_8 = 42;  
public const int CP_CharacterCodepage_ISO_8859_9 = 43;  
public const int CP_CharacterCodepage_ISO_8859_15 = 44;  
public const int CP_CharacterCodepage_THAI_2 = 45;  
public const int CP_CharacterCodepage_CP856 = 46;  
public const int CP_CharacterCodepage_CP874 = 47;  
public const int CP_CharacterCodepage_TCVN3 = 48;
```

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

```
public const int CP_MultiByteEncoding_GBK = 0;  
public const int CP_MultiByteEncoding_UTF8 = 1;  
public const int CP_MultiByteEncoding_BIG5 = 3;  
public const int CP_MultiByteEncoding_ShiftJIS = 4;  
public const int CP_MultiByteEncoding_EUCKR = 5;
```

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

```
public const int CP_ImageBinarizationMethod_Dithering = 0;  
public const int CP_ImageBinarizationMethod_Thresholding = 1;  
public const int CP_ImageBinarizationMethod_ErrorDiffusion = 2;
```


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

```
public const int CP_ImageCompressionMethod_None = 0;  
public const int CP_ImageCompressionMethod_Level1 = 1;  
public const int CP_ImageCompressionMethod_Level2 = 2;
```

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

```
public const int CP_ImagePixelFormat_MONO = 1;  
public const int CP_ImagePixelFormat_MONOLSB = 2;  
public const int CP_ImagePixelFormat_GRAY8 = 3;  
public const int CP_ImagePixelFormat_BYTEORDERED_RGB24 = 4;  
public const int CP_ImagePixelFormat_BYTEORDERED_BGR24 = 5;  
public const int CP_ImagePixelFormat_BYTEORDERED_ARGB32 = 6;  
public const int CP_ImagePixelFormat_BYTEORDERED_RGBA32 = 7;  
public const int CP_ImagePixelFormat_BYTEORDERED_ABGR32 = 8;  
public const int CP_ImagePixelFormat_BYTEORDERED_BGRA32 = 9;
```

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

```
public const int CP_QRCodeECC_L = 1;  
public const int CP_QRCodeECC_M = 2;  
public const int CP_QRCodeECC_Q = 3;  
public const int CP_QRCodeECC_H = 4;
```

CP_Pos_Alignment

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

Syntax

```
public const int CP_Pos_Alignment_Left = 0;  
public const int CP_Pos_Alignment_HCenter = 1;  
public const int CP_Pos_Alignment_Right = 2;
```

CP_Pos_BarcodeType

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

Syntax

```
public const int CP_Pos_BarcodeType_UPCA = 0x41;  
public const int CP_Pos_BarcodeType_UPCE = 0x42;  
public const int CP_Pos_BarcodeType_EAN13 = 0x43;  
public const int CP_Pos_BarcodeType_EAN8 = 0x44;  
public const int CP_Pos_BarcodeType_CODE39 = 0x45;  
public const int CP_Pos_BarcodeType_ITF = 0x46;  
public const int CP_Pos_BarcodeType_CODEBAR = 0x47;  
public const int CP_Pos_BarcodeType_CODE93 = 0x48;  
public const int CP_Pos_BarcodeType_CODE128 = 0x49;
```

CP_Pos_BarcodeTextPrintPosition

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

Syntax

```
public const int CP_Pos_BarcodeTextPrintPosition_None = 0;  
public const int CP_Pos_BarcodeTextPrintPosition_AboveBarcode = 1;  
public const int CP_Pos_BarcodeTextPrintPosition_BelowBarcode = 2;  
public const int CP_Pos_BarcodeTextPrintPosition_AboveAndBelowBarcode = 3;
```

CP_Page_DrawDirection

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

Syntax

```
public const int CP_Page_DrawDirection_LeftToRight = 0;  
public const int CP_Page_DrawDirection_BottomToTop = 1;  
public const int CP_Page_DrawDirection_RightToLeft = 2;  
public const int CP_Page_DrawDirection_TopToBottom = 3;
```

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

```
public const int CP_Page_DrawAlignment_Left = -1;  
public const int CP_Page_DrawAlignment_HCenter = -2;  
public const int CP_Page_DrawAlignment_Right = -3;  
public const int CP_Page_DrawAlignment_Top = -1;  
public const int CP_Page_DrawAlignment_VCenter = -2;  
public const int CP_Page_DrawAlignment_Bottom = -3;
```


CP_Label_BarcodeType

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

Syntax

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

CP_Label_BarcodeTextPrintPosition

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

Syntax

```
public const int CP_Label_BarcodeTextPrintPosition_None = 0;  
public const int CP_Label_BarcodeTextPrintPosition_AboveBarcode = 1;  
public const int CP_Label_BarcodeTextPrintPosition_BelowBarcode = 2;  
public const int CP_Label_BarcodeTextPrintPosition_AboveAndBelowBarcode = 3;
```

CP_Label_Rotation

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

Syntax

```
public const int CP_Label_Rotation_0 = 0;  
public const int CP_Label_Rotation_90 = 1;  
public const int CP_Label_Rotation_180 = 2;  
public const int CP_Label_Rotation_270 = 3;
```

CP_Label_Color

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

Syntax

```
public const int CP_Label_Color_White = 0;  
public const int CP_Label_Color_Black = 1;
```

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

```
public static bool CP_PRINTERSTATUS_ERROR_CUTTER(long error_status) { return (error_status & 0x01) != 0; }
public static bool CP_PRINTERSTATUS_ERROR_FLASH(long error_status) { return (error_status & 0x02) != 0; }
public static bool CP_PRINTERSTATUS_ERROR_NOPAPER(long error_status) { return (error_status & 0x04) != 0; }
public static bool CP_PRINTERSTATUS_ERROR_VOLTAGE(long error_status) { return (error_status & 0x08) != 0; }
public static bool CP_PRINTERSTATUS_ERROR_MARKER(long error_status) { return (error_status & 0x10) != 0; }
public static bool CP_PRINTERSTATUS_ERROR_ENGINE(long error_status) { return (error_status & 0x20) != 0; }
public static bool CP_PRINTERSTATUS_ERROR_OVERHEAT(long error_status) { return (error_status & 0x40) != 0; }
public static bool CP_PRINTERSTATUS_ERROR_COVERUP(long error_status) { return (error_status & 0x80) != 0; }
public static bool CP_PRINTERSTATUS_ERROR_MOTOR(long error_status) { return (error_status & 0x100) != 0; }
public static bool CP_PRINTERSTATUS_INFO_LABELPAPER(long info_status) { return (info_status & 0x02) != 0; }
public static bool CP_PRINTERSTATUS_INFO_LABELMODE(long info_status) { return (info_status & 0x04) != 0; }
public static bool CP_PRINTERSTATUS_INFO_HAVEDATA(long info_status) { return (info_status & 0x08) != 0; }
public static bool CP_PRINTERSTATUS_INFO_NOPAPERCANCELED(long info_status) { return (info_status & 0x10) != 0; }
public static bool CP_PRINTERSTATUS_INFO_PAPERNOFETCH(long info_status) { return (info_status & 0x20) != 0; }
public static bool CP_PRINTERSTATUS_INFO_PRINTIDLE(long info_status) { return (info_status & 0x40) != 0; }
public static bool CP_PRINTERSTATUS_INFO_RECVIDLE(long info_status) { 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

```
public static bool CP_RTSTATUS_DRAWER_OPENED(long status) { return (((status >> 0) & 0x04) == 0x00); }
public static bool CP_RTSTATUS_OFFLINE(long status) { return (((status >> 0) & 0x08) == 0x08); }
public static bool CP_RTSTATUS_COVERUP(long status) { return (((status >> 8) & 0x04) == 0x04); }
public static bool CP_RTSTATUS_FEED_PRESSED(long status) { return (((status >> 8) & 0x08) == 0x08); }
public static bool CP_RTSTATUS_NOPAPER(long status) { return (((status >> 8) & 0x20) == 0x20); }
public static bool CP_RTSTATUS_ERROR_OCCURED(long status) { return (((status >> 8) & 0x40) == 0x40); }
public static bool CP_RTSTATUS_CUTTER_ERROR(long status) { return (((status >> 16) & 0x08) == 0x08); }
public static bool CP_RTSTATUS_UNRECOVERABLE_ERROR(long status) { return (((status >> 16) & 0x20) == 0x20); }
public static bool CP_RTSTATUS_DEGREE_OR_VOLTAGE_OVERRANGE(long status) { return (((status >> 16) & 0x40) == 0x40); }
public static bool CP_RTSTATUS_PAPER_NEAREND(long status) { return (((status >> 24) & 0x08) == 0x08); }
public static bool CP_RTSTATUS_PAPER_TAKEOUT(long status) { 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

```
public const int CP_LABEL_TEXT_STYLE_BOLD = (1<<0);
public const int CP_LABEL_TEXT_STYLE_UNDERLINE = (1<<1);
public const int CP_LABEL_TEXT_STYLE_HIGHLIGHT = (1<<2);
public const int CP_LABEL_TEXT_STYLE_STRIKETHROUGH = (1<<3);
public const int CP_LABEL_TEXT_STYLE_ROTATION_0 = (0<<4);
public const int CP_LABEL_TEXT_STYLE_ROTATION_90 = (1<<4);
public const int CP_LABEL_TEXT_STYLE_ROTATION_180 = (2<<4);
public const int CP_LABEL_TEXT_STYLE_ROTATION_270 = (3<<4);
public static int CP_LABEL_TEXT_STYLE_WIDTH_ENLARGEMENT(int n) { return ((n)<<8); }
public static int CP_LABEL_TEXT_STYLE_HEIGHT_ENLARGEMENT(int n) { return ((n)<<12); }
```

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 delegate void CP_OnNetPrinterDiscovered(String local_ip, String discovered_mac, String discovered_ip, String discovered_name, UIntPtr private_data);
```

CP_OnPortClosedEvent

When the port is closed, the interface is called back

Syntax

```
public delegate void CP_OnPortClosedEvent(UIntPtr handle, UIntPtr private_data);
```

Remarks

When the port is abnormal, such as USB cable unplugging, 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 delegate void CP_OnPortWrittenEvent(UIntPtr handle, IntPtr buffer, int count, UIntPtr private_data);
```

CP_OnPortReceivedEvent

This function is called back when the port receives data

Syntax

```
public delegate void CP_OnPortReceivedEvent(UIntPtr handle, IntPtr buffer, int count, UIntPtr private_data);
```

CP_OnPrinterStatusEvent

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

Syntax

```
public delegate void CP_OnPrinterStatusEvent(UIntPtr handle, long printer_error_status, long printer_info_status, UIntPtr private_data);
```

CP_OnPrinterReceivedEvent

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

Syntax

```
public delegate void CP_OnPrinterReceivedEvent(UIntPtr handle, uint printer_received_byte_count, UIntPtr private_data);
```

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 delegate void CP_OnPrinterPrintedEvent(UIntPtr handle, uint printer_printed_page_id, UIntPtr private_data);
```


Add Callback and Remove Callback

CP_Port_AddOnPortClosedEvent

Add Callback, Port Closed

Syntax

```
public static extern int  
CP_Port_AddOnPortClosedEvent([MarshalAs(UnmanagedType.FunctionPtr)]CP_OnPortClosedEvent evt, UIntPtr  
private_data);
```

```
//      Add Callback, Port Closed  
//  
//  event  
//      callback function  
//  
//  private_data  
//      the parameter passed to callback function  
//  
// return  
//      true on success.  
//      false on failed.
```

CP_Port_AddOnPortWrittenEvent

Add Callback, Port Written Bytes

Syntax

```
public static extern int  
CP_Port_AddOnPortWrittenEvent([MarshalAs(UnmanagedType.FunctionPtr)]CP_OnPortWrittenEvent evt, UIntPtr  
private_data);
```

```
//      Add Callback, Port Written Bytes  
//  
//  event  
//      callback function  
//  
//  private_data  
//      the parameter passed to callback function  
//  
// return  
//      true on success.  
//      false on failed.
```

CP_Port_AddOnPortReceivedEvent

Add Callback, Port Received Bytes

Syntax

```
public static extern int  
CP_Port_AddOnPortReceivedEvent([MarshalAs(UnmanagedType.FunctionPtr)]CP_OnPortReceivedEvent evt, UIntPtr  
private_data);
```

```
//      Add Callback, Port Received Bytes  
//  
//  event  
//      callback function  
//  
//  private_data  
//      the parameter passed to callback function  
//  
// return  
//      true on success.  
//      false on failed.
```

CP_Port_RemoveOnPortClosedEvent

Remove Callback

Syntax

```
public static extern int  
CP_Port_RemoveOnPortClosedEvent([MarshalAs(UnmanagedType.FunctionPtr)]CP_OnPortClosedEvent evt);  
  
// Remove Callback  
//  
// event  
// callback function  
//  
// return  
// true on success.  
// false on failed.
```

CP_Port_RemoveOnPortWrittenEvent

Remove Callback

Syntax

```
public                                static                                extern                                int
CP_Port_RemoveOnPortWrittenEvent([MarshalAs(UnmanagedType.FunctionPtr)]CP_OnPortWrittenEvent evt);

//      Remove Callback
//
//  event
//      callback function
//
// return
//      true on success.
//      false on failed.
```

CP_Port_RemoveOnPortReceivedEvent

Remove Callback

Syntax

```
public static extern int  
CP_Port_RemoveOnPortReceivedEvent([MarshalAs(UnmanagedType.FunctionPtr)]CP_OnPortReceivedEvent evt);  
  
// Remove Callback  
//  
// event  
// callback function  
//  
// return  
// true on success.  
// false on failed.
```

CP_Printer_AddOnPrinterStatusEvent

Add Callback, Printer Status Updated

Syntax

```
public static extern int  
CP_Printer_AddOnPrinterStatusEvent([MarshalAs(UnmanagedType.FunctionPtr)]CP_OnPrinterStatusEvent evt, UIntPtr  
private_data);
```

```
//      Add Callback, Printer Status Updated  
//  
//  event  
//      callback function  
//  
//  private_data  
//      the parameter passed to callback function  
//  
// return  
//      true on success.  
//      false on failed.
```

CP_Printer_AddOnPrinterReceivedEvent

Add Callback, Printer Received Byte Count Updated

Syntax

```
public static extern int  
CP_Printer_AddOnPrinterReceivedEvent([MarshalAs(UnmanagedType.FunctionPtr)]CP_OnPrinterReceivedEvent evt,  
UIntPtr private_data);
```

```
//      Add Callback, Printer Received Byte Count Updated  
//  
//  event  
//      callback function  
//  
//  private_data  
//      the parameter passed to callback function  
//  
// return  
//      true on success.  
//      false on failed.
```


CP_Printer_AddOnPrinterPrintedEvent

Add Callback, Printer Printed Page ID Updated

Syntax

```
public                                static                                extern                                int
CP_Printer_AddOnPrinterPrintedEvent([MarshalAs(UnmanagedType.FunctionPtr)]CP_OnPrinterPrintedEvent evt,
UIntPtr private_data);
```

```
//      Add Callback, Printer Printed Page ID Updated
//
//  event
//      callback function
//
//  private_data
//      the parameter passed to callback function
//
// return
//      true on success.
//      false on failed.
```

CP_Printer_RemoveOnPrinterStatusEvent

Remove Callback

Syntax

```
public                                static                                extern                                int
CP_Printer_RemoveOnPrinterStatusEvent([MarshalAs(UnmanagedType.FunctionPtr)]CP_OnPrinterStatusEvent evt);
```

```
//      Remove Callback
//
//  event
//      callback function
//
// return
//      true on success.
//      false on failed.
```

CP_Printer_RemoveOnPrinterReceivedEvent

Remove Callback

Syntax

```
public                                static                                extern                                int
CP_Printer_RemoveOnPrinterReceivedEvent([MarshalAs(UnmanagedType.FunctionPtr)]CP_OnPrinterReceivedEvent
evt);
```

```
//      Remove Callback
//
//  event
//      callback function
//
// return
//      true on success.
//      false on failed.
```

CP_Printer_RemoveOnPrinterPrintedEvent

Remove Callback

Syntax

```
public static extern int  
CP_Printer_RemoveOnPrinterPrintedEvent([MarshalAs(UnmanagedType.FunctionPtr)]CP_OnPrinterPrintedEvent evt);
```

```
// Remove Callback  
//  
// event  
// callback function  
//  
// return  
// true on success.  
// false on failed.
```

Port Function

CP_Port_EnumCom

Enumerate serial port

Syntax

```
public static String[] CP_Port_EnumCom()
```

```
//    return
```

```
//    Enumerated port
```

CP_Port_EnumLpt

Enumerate parallel port

Syntax

```
public static String[] CP_Port_EnumLpt()
```

```
//    return
```

```
//    Enumerated port
```

CP_Port_EnumUsb

Enumerate usb port

Syntax

```
public static String[] CP_Port_EnumUsb()
```

```
//    return
```

```
//    Enumerated port
```

CP_Port_EnumNetPrinter

Enumerate net printer

Syntax

```
public static extern void CP_Port_EnumNetPrinter(uint timeout, ref int cancel,
[MarshalAs(UnmanagedType.FunctionPtr)]CP_OnNetPrinterDiscovered on_discovered, UIntPtr private_data);
```

```
// 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_OpenCom

Open com port

Syntax

```
public static extern UIntPtr CP_Port_OpenCom(String name, int baudrate, int databits, int parity, int stopbits, int flowcontrol, int autoreplymode);
```

```
//      Open com port
//
// name
//      port name
//      for example COM1,COM2,COM3,...COM11...
//
// baudrate
//      baudrate
//      Normally choose 9600,19200,38400,57600,115200.
//      Need to keep the same as printer baudrate, suggest using high baudrate to get better printing speed.
//
// databits
//      databits, value range is [4,8]
//
// parity
//      Parity bit, The values are defined as follows:
//      value      define
//      0          no parity
//      1          odd parity
//      2          even parity
//      3          mark parity
//      4          space parity
//
// stopbits
//      Parity bit, The values are defined as follows:
//      value      define
//      0          1bit stopbits
//      1          1.5bit stopbits
//      2          2bit stopbits
//
// flowcontrol
//      flow control
//
// 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
//      If serial port was occupied, open serial will fail
//      If baud rate don't match with printer baud rate, it won't print.
```

CP_Port_OpenLpt

Open Lpt

Syntax

```
public static extern UIntPtr CP_Port_OpenLpt(String name);
```

```
//      Open Lpt
//
// name
//      port name
//      for example: LPT1,LPT2,LPT3...
//
// return
//      Return handle, If open success, return non-zero value, else return zero.
//
// remarks
//      Parallel port just support one way communication, just can write, but can't read
//      All the query status function, are invalid for parallel port.
```

CP_Port_OpenUsb

Open USB

Syntax

```
public static extern UIntPtr CP_Port_OpenUsb(String name, int autoreplymode);
```

```
//      Open USB
//
// name
//      port name
//      Can get printer name through Port_EnumUSB
//      Can use any other strings, at this time, if find USB printer, will open directly.
//
// 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
//      USB printer connect to computer, if device manager appear USB Printing Support, then can open USE this
//      function.
```

CP_Port_OpenTcp

Open Tcp

Syntax

```
public static extern UIntPtr CP_Port_OpenTcp(String local_ip, String dest_ip, ushort dest_port, uint timeout, int autoreplymode);
```

```
//      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_Write

Write data to port

Syntax

```
public static extern int CP_Port_Write(UIntPtr handle, byte[] buffer, int count, int timeout);
```

```
//      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 static extern int CP_Port_Read(UIntPtr handle, byte[] buffer, int count, int timeout);
```

```
//      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 static extern int CP_Port_ReadUntilByte(UIntPtr handle, byte[] buffer, int count, int timeout, byte breakByte);
```

```
//      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 static extern int CP_Port_Available(UIntPtr handle);
```

```
//      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 static extern void CP_Port_SkipAvailable(UIntPtr handle);
```

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

CP_Port_IsConnectionValid

Check Connection Valid

Syntax

```
public static extern int CP_Port_IsConnectionValid(UIntPtr handle);
```

```
//      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 static extern int CP_Port_IsOpened(UIntPtr handle);
```

```
//      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 static extern void CP_Port_Close(UIntPtr handle);
```

```
//      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 static extern int CP_Printer_GetPrinterResolutionInfo(UIntPtr handle, ref uint width_mm, ref uint height_mm, ref uint dots_per_mm);
```

```
//      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 static String CP_Printer_GetPrinterFirmwareVersion(UIntPtr handle)
```

CP_Printer_GetPrinterStatusInfo

Get Printer Status

Syntax

```
public static extern int CP_Printer_GetPrinterStatusInfo(UIntPtr handle, ref long printer_error_status, ref long printer_info_status, ref long timestamp_ms);
```

```
//      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 static extern int CP_Printer_GetPrinterReceivedInfo(UIntPtr handle, ref uint printer_received_byte_count, ref long timestamp_ms);
```

```
//      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 static extern int CP_Printer_GetPrinterPrintedInfo(UIntPtr handle, ref uint printer_printed_page_id, ref long timestamp_ms);
```

```
//      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 static extern int CP_Printer_GetPrinterLabelPositionAdjustmentInfo(UIntPtr handle, ref double label_print_position_adjustment, ref double label_tear_position_adjustment, ref long timestamp_ms);
```

```
//      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 static extern int CP_Printer_SetPrinterLabelPositionAdjustmentInfo(UIntPtr handle, double label_print_position_adjustment, double label_tear_position_adjustment);
```

```
// 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 static extern int CP_Printer_ClearPrinterBuffer(UIntPtr handle);
```

```
//      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 static extern int CP_Printer_ClearPrinterError(UIntPtr handle);
```

```
//      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 static extern int CP_Pos_QueryRTStatus(UIntPtr handle, uint timeout);
```

```
//      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 static extern int CP_Pos_QueryPrintResult(UIntPtr handle, uint timeout);
```

```
//      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 static extern int CP_Pos_KickOutDrawer(UIntPtr handle, int nDrawerIndex, int nHighLevelTime, int nLowLevelTime);
```

```
//      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 static extern int CP_Pos_Beep(UIntPtr handle, int nBeepCount, int nBeepMs);
```

```
//      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 static extern int CP_Pos_FeedAndHalfCutPaper(UIntPtr handle);
```

```
//      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 static extern int CP_Pos_FullCutPaper(UIntPtr handle);
```

```
//      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 static extern int CP_Pos_HalfCutPaper(UIntPtr handle);
```

```
//      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 static extern int CP_Pos_FeedLine(UIntPtr handle, int numLines);
```

```
//      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 static extern int CP_Pos_FeedDot(UIntPtr handle, int numDots);
```

```
//      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 static extern int CP_Pos_PrintSelfTestPage(UIntPtr handle);
```

```
//      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 static extern int CP_Pos_PrintText(UIntPtr handle, String str);
```

```
//      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 static extern int CP_Pos_PrintTextInUTF8(UIntPtr handle, String str);
```

```
//      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 static extern int CP_Pos_PrintTextInGBK(UIntPtr handle, String str);
```

```
//      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 static extern int CP_Pos_PrintTextInBIG5(UIntPtr handle, String str);
```

```
//      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 static extern int CP_Pos_PrintTextInShiftJIS(UIntPtr handle, String str);

//      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 static extern int CP_Pos_PrintTextInEUCKR(UIntPtr handle, String str);
```

```
//      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 static extern int CP_Pos_PrintBarcode(UIntPtr handle, int nBarcodeType, String str);
```

```
//      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 static extern int CP_Pos_PrintBarcode_Code128Auto(UIntPtr handle, String str);
```

```
//      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 static extern int CP_Pos_PrintQRCode(UIntPtr handle, int nVersion, int nECCLevel, String str);
```

```
//      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 static extern int CP_Pos_PrintQRCodeUseEpsonCmd(UIntPtr handle, int nQRCodeUnitWidth, int nECCLevel, String str);
```

```
//      print qrcode
//
// handle
//      Port handle, returned by OpenXXX
//
// nQRCodeUnitWidth
//      QRCode code block width, the value range is [1, 16]
//
// 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_PrintDoubleQRCode

print 2 qrcode

Syntax

public static extern int CP_Pos_PrintDoubleQRCode(UIntPtr handle, int nQRCodeUnitWidth, int nQR1Position, int nQR1Version, int nQR1ECCLevel, String strQR1, int nQR2Position, int nQR2Version, int nQR2ECCLevel, String strQR2);

```
//      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.
//
// nQR1ECCLevel
// nQR2ECCLevel
//      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 static extern int CP_Pos_PrintPDF417BarcodeUseEpsonCmd(UIntPtr handle, int columnCount, int rowCount, int unitWidth, int rowHeight, int nECCLevel, int dataProcessingMode, String str);

```
//      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]
//
// nECCLevel
//      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 static extern int CP_Pos_PrintRasterImageFromFile(UIntPtr handle, int dstw, int dsth, String pszFile, int binaryzation_method, int compression_method);
```

```
//      print image
//
// handle
//      Port handle, returned by OpenXXX
//
// 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_Pos_PrintRasterImageFromData

print image (data can be readed from file)

Syntax

```
public static extern int CP_Pos_PrintRasterImageFromData(UIntPtr handle, int dstw, int dsth, byte[] data, int data_size,
int binaryzation_method, int compression_method);
```

```
//      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 static extern int CP_Pos_PrintRasterImageFromPixels(UIntPtr handle, byte[] img_data, int img_datalen, int
img_width, int img_height, int img_stride, int img_format, int binaryzation_method, int compression_method);
```

```
//      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 static extern int CP_Pos_PrintHorizontalLine(UIntPtr handle, int nLineStartPosition, int nLineEndPosition);
```

```
//      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 static extern int CP_Pos_PrintHorizontalLineSpecifyThickness(UIntPtr handle, int nLineStartPosition, int nLineEndPosition, int nLineThickness);
```

```
//      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 static extern int CP_Pos_PrintMultipleHorizontalLinesAtOneRow(UIntPtr handle, int nLineCount, int[] pLineStartPosition, int[] pLineEndPosition);
```

```
//      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 static extern int CP_Pos_ResetPrinter(UIntPtr handle);
```

```
//      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 static extern int CP_Pos_SetPrintSpeed(UIntPtr handle, int nSpeed);
```

```
//      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 static extern int CP_Pos_SetPrintDensity(UIntPtr handle, int nDensity);
```

```
//      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 static extern int CP_Pos_SetSingleByteMode(UIntPtr handle);
```

```
//      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 static extern int CP_Pos_SetCharacterSet(UIntPtr handle, int nCharacterSet);
```

```
//      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 static extern int CP_Pos_SetCharacterCodepage(UIntPtr handle, int nCharacterCodepage);
```

```
//      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 static extern int CP_Pos_SetMultiByteMode(UIntPtr handle);
```

```
//      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 static extern int CP_Pos_SetMultiByteEncoding(UIntPtr handle, int nEncoding);
```

```
//      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 static extern int CP_Pos_SetMovementUnit(UIntPtr handle, int nHorizontalMovementUnit, int nVerticalMovementUnit);
```

```
//      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 static extern int CP_Pos_SetPrintAreaLeftMargin(UIntPtr handle, int nLeftMargin);
```

```
//      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 static extern int CP_Pos_SetPrintAreaWidth(UIntPtr handle, int nWidth);
```

```
//      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 static extern int CP_Pos_SetHorizontalAbsolutePrintPosition(UIntPtr handle, int nPosition);
```

```
//      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 static extern int CP_Pos_SetHorizontalRelativePrintPosition(UIntPtr handle, int nPosition);
```

```
//      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 static extern int CP_Pos_SetVerticalAbsolutePrintPosition(UIntPtr handle, int nPosition);
```

```
//      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 static extern int CP_Pos_SetVerticalRelativePrintPosition(UIntPtr handle, int nPosition);
```

```
//      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 static extern int CP_Pos_SetAlignment(UIntPtr handle, int nAlignment);
```

```
//      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 static extern int CP_Pos_SetTextScale(UIntPtr handle, int nWidthScale, int nHeightScale);
```

```
//      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 static extern int CP_Pos_SetAsciiTextFontType(UIntPtr handle, int nFontType);
```

```
//      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 static extern int CP_Pos_SetTextBold(UIntPtr handle, int nBold);
```

```
//      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 static extern int CP_Pos_SetTextUnderline(UIntPtr handle, int nUnderline);
```

```
//      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 static extern int CP_Pos_SetTextUpsideDown(UIntPtr handle, int nUpsideDown);
```

```
//      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 static extern int CP_Pos_SetTextWhiteOnBlack(UIntPtr handle, int nWhiteOnBlack);
```

```
//      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 static extern int CP_Pos_SetTextRotate(UIntPtr handle, int nRotate);
```

```
//      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 static extern int CP_Pos_SetTextLineHeight(UIntPtr handle, int nLineHeight);
```

```
//      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 static extern int CP_Pos_SetAsciiTextCharRightSpacing(UIntPtr handle, int nSpacing);
```

```
//      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 static extern int CP_Pos_SetKanjiTextCharSpacing(UIntPtr handle, int nLeftSpacing, int nRightSpacing);
```

```
//      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 static extern int CP_Pos_SetBarcodeUnitWidth(UIntPtr handle, int nBarcodeUnitWidth);
```

```
//      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 static extern int CP_Pos_SetBarcodeHeight(UIntPtr handle, int nBarcodeHeight);
```

```
//      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 static extern int CP_Pos_SetBarcodeReadableTextFontType(UIntPtr handle, int nFontType);
```

```
//      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 static extern int CP_Pos_SetBarcodeReadableTextPosition(UIntPtr handle, int nTextPosition);
```

```
//      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 static extern int CP_Page_SelectPageMode(UIntPtr handle);
```

```
//      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 static extern int CP_Page_SelectPageModeEx(UIntPtr handle, int nHorizontalMovementUnit, int nVerticalMovementUnit, int x, int y, int width, int height);
```

```
//      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 static extern int CP_Page_ExitPageMode(UIntPtr handle);
```

```
//      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 static extern int CP_Page_PrintPage(UIntPtr handle);
```

```
//      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 static extern int CP_Page_ClearPage(UIntPtr handle);
```

```
//      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 static extern int CP_Page_SetPageArea(UIntPtr handle, int x, int y, int width, int height);
```

```
//      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 static extern int CP_Page_SetPageDrawDirection(UIntPtr handle, int nDirection);
```

```
//      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 static extern int CP_Page_DrawRect(UIntPtr handle, int x, int y, int width, int height, int color);
```

```
//      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 static extern int CP_Page_DrawBox(UIntPtr handle, int x, int y, int width, int height, int borderwidth, int bordercolor);

```
//      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 static extern int CP_Page_DrawText(UIntPtr handle, int x, int y, String str);
```

```
//      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 static extern int CP_Page_DrawTextInUTF8(UIntPtr handle, int x, int y, String str);
```

```
//      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 static extern int CP_Page_DrawTextInGBK(UIntPtr handle, int x, int y, String str);
```

```
//      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 static extern int CP_Page_DrawTextInBIG5(UIntPtr handle, int x, int y, String str);
```

```
//      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 static extern int CP_Page_DrawTextInShiftJIS(UIntPtr handle, int x, int y, String str);
```

```
//      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 static extern int CP_Page_DrawTextInEUCKR(UIntPtr handle, int x, int y, String str);
```

```
//      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 static extern int CP_Page_DrawBarcode(UIntPtr handle, int x, int y, int nBarcodeType, String str);
```

```
//      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 static extern int CP_Page_DrawQRCode(UIntPtr handle, int x, int y, int nVersion, int nECCLevel, String str);
```

```
//      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 static extern int CP_Page_DrawRasterImageFromFile(UIntPtr handle, int x, int y, int dstw, int dsth, String pszFile,
int binaryzation_method);
```

```
//      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
//
// 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.
```

CP_Page_DrawRasterImageFromData

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

Syntax

```
public static extern int CP_Page_DrawRasterImageFromData(UIntPtr handle, int x, int y, int dstw, int dsth, byte[] data,
int data_size, int binaryzation_method);
```

```
//      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
//
// 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.
```

CP_Page_DrawRasterImageFromPixels

print image pixels in pagemode

Syntax

```
public static extern int CP_Page_DrawRasterImageFromPixels(UIntPtr handle, int x, int y, byte[] img_data, int
img_datalen, int img_width, int img_height, int img_stride, int img_format, int binaryzation_method);
```

```
//      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 static extern int CP_BlackMark_EnableBlackMarkMode(UIntPtr handle);
```

```
//      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 static extern int CP_BlackMark_DisableBlackMarkMode(UIntPtr handle);
```

```
//      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 static extern int CP_BlackMark_SetBlackMarkMaxFindLength(UIntPtr handle, int maxFindLength);
```

```
//      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 static extern int CP_BlackMark_FindNextBlackMark(UIntPtr handle);
```

```
//      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 static extern int CP_BlackMark_SetBlackMarkPaperPrintPosition(UIntPtr handle, int position);
```

```
//      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 static extern int CP_BlackMark_SetBlackMarkPaperCutPosition(UIntPtr handle, int position);
```

```
//      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 static extern int CP_BlackMark_FullCutBlackMarkPaper(UIntPtr handle);
```

```
//      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 static extern int CP_BlackMark_HalfCutBlackMarkPaper(UIntPtr handle);
```

```
//      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 static extern int CP_Label_EnableLabelMode(UIntPtr handle);
```

```
//      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 static extern int CP_Label_DisableLabelMode(UIntPtr handle);
```

```
//      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 static extern int CP_Label_CalibrateLabel(UIntPtr handle);
```

```
//      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 static extern int CP_Label_FeedLabel(UIntPtr handle);
```

```
//      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 static extern int CP_Label_BackPaperToPrintPosition(UIntPtr handle);
```

```
//      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 static extern int CP_Label_FeedPaperToTearPosition(UIntPtr handle);
```

```
//      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 static extern int CP_Label_PageBegin(UIntPtr handle, int x, int y, int width, int height, int rotation);
```

```
//      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 static extern int CP_Label_PagePrint(UIntPtr handle, int copies);
```

```
//      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 static extern int CP_Label_DrawText(UIntPtr handle, int x, int y, int font, int style, String str);
```

```
//      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 static extern int CP_Label_DrawTextInUTF8(UIntPtr handle, int x, int y, int font, int style, String str);
```

```
//      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 static extern int CP_Label_DrawTextInGBK(UIntPtr handle, int x, int y, int font, int style, String str);

```
//      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_DrawBarcode

Draw 1D code in the assigned position of label page

Syntax

```
public static extern int CP_Label_DrawBarcode(UIntPtr handle, int x, int y, int nBarcodeType, int nBarcodeTextPrintPosition, int height, int unitwidth, int rotation, String str);
```

```
//      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 static extern int CP_Label_DrawQRCode(UIntPtr handle, int x, int y, int nVersion, int nECCLevel, int unitwidth, int rotation, String str);

```
//      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 static extern int CP_Label_DrawPDF417Code(UIntPtr handle, int x, int y, int column, int nAspectRatio, int nECCLevel, int unitwidth, int rotation, String str);

```
//      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 static extern int CP_Label_DrawImageFromFile(UIntPtr handle, int x, int y, int dstw, int dsth, String pszFile, int binaryzation_method, int compression_method);
```

```
//      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 static extern int CP_Label_DrawImageFromData(UIntPtr handle, int x, int y, int dstw, int dsth, byte[] data, int data_size, int binaryzation_method, int compression_method);
```

```
//      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
//
// 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_DrawImageFromPixels

Draw picture in the assigned position of label page

Syntax

```
public static extern int CP_Label_DrawImageFromPixels(UIntPtr handle, int x, int y, byte[] img_data, int img_dataLen,
int img_width, int img_height, int img_stride, int img_format, int binaryzation_method, int compression_method);
```

```
//      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 stride. 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 static extern int CP_Label_DrawLine(UIntPtr handle, int startx, int starty, int endx, int endy, int linewidth, int linecolor);
```

```
//      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 static extern int CP_Label_DrawRect(UIntPtr handle, int x, int y, int width, int height, int color);
```

```
//      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 static extern int CP_Label_DrawBox(UIntPtr handle, int x, int y, int width, int height, int borderwidth, int bordercolor);
```

```
//      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

```
public static String CP_Library_Version()
```

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

CP_Proto_QueryBatteryLevel

Query battery level

Syntax

```
public static extern int CP_Proto_QueryBatteryLevel(UIntPtr handle, uint timeout);
```

```
//      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 static String CP_Proto_QuerySerialNumber(UIntPtr handle, int timeout);
```

```
//      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 static extern int CP_Proto_SetSystemNameAndSerialNumber(UIntPtr handle, String systemName, String serialNumber);
```

```
//      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 static extern int CP_Proto_SetBluetoothNameAndPassword(UIntPtr handle, String bluetoothName, String bluetoothPassword);
```

```
//      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 static extern int CP_Proto_SetPTPBasicParameters(UIntPtr handle, int baudrate, int codepage, int density, int
asciiFontType, int lineFeed, int idleTime, int powerOffTime, int maxFeedLength, int pageLength);
```

```
//      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 Cyrillic #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 static extern int CP_Settings_Hardware_SetPrintSpeed(UIntPtr handle, int nSpeed);
```

```
//      set print speed
//
// 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.
```

