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_OnBluetoothDeviceDiscovered 35
CP_OnWiFiP2PDeviceDiscovered 36
CP_OnPortOpenedEvent 37
CP_OnPortOpenFailedEvent 38
CP_OnPortClosedEvent 39
CP_OnPortWrittenEvent 40
CP_OnPortReceivedEvent 41
CP_OnPrinterStatusEvent 42
CP_OnPrinterReceivedEvent 43
CP_OnPrinterPrintedEvent 44
Add Callback and Remove Callback 45
CP_Port_AddOnPortOpenedEvent 45
CP_Port_AddOnPortOpenFailedEvent 46
CP_Port_AddOnPortClosedEvent 47
CP_Port_AddOnPortWrittenEvent 48
CP_Port_AddOnPortReceivedEvent 49

CP_Port_RemoveOnPortOpenedEvent 50
CP_Port_RemoveOnPortOpenFailedEvent 51
CP_Port_RemoveOnPortClosedEvent 52
CP_Port_RemoveOnPortWrittenEvent 53
CP_Port_RemoveOnPortReceivedEvent 54
CP_Printer_AddOnPrinterStatusEvent 55
CP_Printer_AddOnPrinterReceivedEvent 56
CP_Printer_AddOnPrinterPrintedEvent 57
CP_Printer_RemoveOnPrinterStatusEvent 58
CP_Printer_RemoveOnPrinterReceivedEvent 59
CP_Printer_RemoveOnPrinterPrintedEvent 60
Port Function 61
CP_Port_EnumCom 61
CP_Port_EnumLpt 62
CP_Port_EnumUsb 63
CP_Port_EnumNetPrinter 64
CP_Port_EnumBtDevice 65
CP_Port_EnumBleDevice 66
CP_Port_EnumWiFiP2PDevice 67
CP_Port_OpenCom 68
CP_Port_OpenLpt 70
CP_Port_OpenUsb 71
CP_Port_OpenTcp 72
CP_Port_OpenBtSpp 73
CP_Port_OpenBtBle 74
CP_Port_OpenBtBleProtoV2 75
CP_Port_OpenMemoryBuffer 76
CP_Port_GetMemoryBufferDataPointer 77
CP_Port_GetMemoryBufferDataLength 78
CP_Port_ClearMemoryBufferData 79
CP_Port_WiFiP2P_Connect 80
CP_Port_WiFiP2P_Disconnect 81
CP_Port_WiFiP2P_IsConnected 82
CP_Port_Write 83
CP_Port_Read 84
CP_Port_ReadUntilByte 85
CP_Port_Available 86
CP_Port_SkipAvailable 87
CP_Port_IsConnectionValid 88
CP_Port_IsOpened 89
CP_Port_Close 90
Get Printer Info Function 91
CP_Printer_GetPrinterResolutionInfo 91
CP_Printer_GetPrinterFirmwareVersion 92
CP_Printer_GetPrinterStatusInfo 93
CP_Printer_GetPrinterReceivedInfo 94
CP_Printer_GetPrinterPrintedInfo 95
CP_Printer_GetPrinterLabelPositionAdjustmentInfo 96
CP_Printer_SetPrinterLabelPositionAdjustmentInfo 97

CP_Printer_ClearPrinterBuffer 98
CP_Printer_ClearPrinterError 99
Pos Function 100
CP_Pos_QueryRTStatus 100
CP_Pos_QueryPrintResult 101
CP_Pos_KickOutDrawer 102
CP_Pos_Beep 103
CP_Pos_FeedAndHalfCutPaper 104
CP_Pos_FullCutPaper 105
CP_Pos_HalfCutPaper 106
CP_Pos_FeedLine 107
CP_Pos_FeedDot 108
CP_Pos_PrintSelfTestPage 109
CP_Pos_PrintText 110
CP_Pos_PrintTextInUTF8 111
CP_Pos_PrintTextInGBK 112
CP_Pos_PrintTextInBIG5 113
CP_Pos_PrintTextInShiftJIS 114
CP_Pos_PrintTextInEUCKR 115
CP_Pos_PrintTextInBytes 116
CP_Pos_PrintBarcode 117
CP_Pos_PrintBarcode_Code128Auto 118
CP_Pos_PrintQRCode 119
CP_Pos_PrintQRCodeUseEpsonCmd 120
CP_Pos_PrintDoubleQRCode 121
CP_Pos_PrintPDF417BarcodeUseEpsonCmd 122
CP_Pos_PrintRasterImageFromFile 123
CP_Pos_PrintRasterImageFromData 124
CP_Pos_PrintRasterImageFromPixels 125
CP_Pos_PrintHorizontalLine 127
CP_Pos_PrintHorizontalLineSpecifyThickness 128
CP_Pos_PrintMultipleHorizontalLinesAtOneRow 129
CP_Pos_ResetPrinter 130
CP_Pos_SetPrintSpeed 131
CP_Pos_SetPrintDensity 132
CP_Pos_SetSingleByteMode 133
CP_Pos_SetCharacterSet 134
CP_Pos_SetCharacterCodepage 135
CP_Pos_SetMultiByteMode 136
CP_Pos_SetMultiByteEncoding 137
CP_Pos_SetMovementUnit 138
CP_Pos_SetPrintAreaLeftMargin 139
CP_Pos_SetPrintAreaWidth 140
CP_Pos_SetHorizontalAbsolutePrintPosition 141
CP_Pos_SetHorizontalRelativePrintPosition 142
CP_Pos_SetVerticalAbsolutePrintPosition 143
CP_Pos_SetVerticalRelativePrintPosition 144
CP_Pos_SetAlignment 145
CP_Pos_SetTextScale 146

CP_Pos_SetAsciiTextFontType 147
CP_Pos_SetTextBold 148
CP_Pos_SetTextUnderline 149
CP_Pos_SetTextUpsideDown 150
CP_Pos_SetTextWhiteOnBlack 151
CP_Pos_SetTextRotate 152
CP_Pos_SetTextLineHeight 153
CP_Pos_SetAsciiTextCharRightSpacing 154
CP_Pos_SetKanjiTextCharSpacing 155
CP_Pos_SetBarcodeUnitWidth 156
CP_Pos_SetBarcodeHeight 157
CP_Pos_SetBarcodeReadableTextFontType 158
CP_Pos_SetBarcodeReadableTextPosition 159
Page Mode Function 160
CP_Page_SelectPageMode 160
CP_Page_SelectPageModeEx 161
CP_Page_ExitPageMode 162
CP_Page_PrintPage 163
CP_Page_ClearPage 164
CP_Page_SetPageArea 165
CP_Page_SetPageDrawDirection 166
CP_Page_DrawRect 167
CP_Page_DrawBox 168
CP_Page_DrawText 169
CP_Page_DrawTextInUTF8 170
CP_Page_DrawTextInGBK 171
CP_Page_DrawTextInBIG5 172
CP_Page_DrawTextInShiftJIS 173
CP_Page_DrawTextInEUCKR 174
CP_Page_DrawBarcode 175
CP_Page_DrawQRCode 176
CP_Page_DrawRasterImageFromFile 177
CP_Page_DrawRasterImageFromData 178
CP_Page_DrawRasterImageFromPixels 179
Black Marker Function 181
CP_BlackMark_EnableBlackMarkMode 181
CP_BlackMark_DisableBlackMarkMode 182
CP_BlackMark_SetBlackMarkMaxFindLength 183
CP_BlackMark_FindNextBlackMark 184
CP_BlackMark_SetBlackMarkPaperPrintPosition 185
CP_BlackMark_SetBlackMarkPaperCutPosition 186
CP_BlackMark_FullCutBlackMarkPaper 187
CP_BlackMark_HalfCutBlackMarkPaper 188
Label Function 189
CP_Label_EnableLabelMode 189
CP_Label_DisableLabelMode 190
CP_Label_CalibrateLabel 191
CP_Label_FeedLabel 192
CP_Label_PageBegin 193

CP_Label_PagePrint....194
CP_Label_DrawText....195
CP_Label_DrawTextInUTF8....196
CP_Label_DrawTextInGBK....197
CP_Label_DrawTextInBIG5....198
CP_Label_DrawTextInShiftJIS....199
CP_Label_DrawTextInEUCKR....200
CP_Label_DrawTextInBytes....201
CP_Label_DrawBarcode....203
CP_Label_DrawQRCode....205
CP_Label_DrawQRCodeInUTF8....207
CP_Label_DrawQRCodeInBytes....209
CP_Label_DrawPDF417Code....211
CP_Label_DrawImageFromFile....213
CP_Label_DrawImageFromData....214
CP_Label_DrawImageFromPixels....215
CP_Label_DrawLine....217
CP_Label_DrawRect....218
CP_Label_DrawBox....219
Other Function....220
CP_Library_Version....220
CP_Proto_QueryBatteryLevel....221

SDK Introduction

1 The development kit comes with a lot of detailed examples. Before development, please run the corresponding example tests. The test is completely ok, and then consider development

2 The development kit supports a variety of printers, including but not limited to ticket printing, label printing, page mode printing, black label printing

3 Development package supports automatic return function, which requires printer to support automatic return function

4 All the developer functions are prefixed with CP_ to avoid confusion with other developers

5 The development package mainly consists of macro definition, enumeration, callback, port function, ticket printing function, label printing function, page mode printing function, black mark related function

Port functions start with CP_Port_ and include functions such as open Port, close Port, read/write Port, etc
Note printing function begins with CP_Pos_ and mainly encapsulates various note instructions, which can print text, bar code, TWO-DIMENSIONAL code, pictures, etc
The Label printing function begins with CP_Label_ and mainly encapsulates Label instructions. It can print text, bar code, TWO-DIMENSIONAL code, pictures, etc
Page mode printing function begins with CP_Page_, mainly encapsulates the Page mode related instructions, can print text, bar code, TWO-DIMENSIONAL code, pictures, etc
The BlackMark correlation function begins with CP_BlackMark_ and mainly encapsulates the black calibration bit correlation instruction

6 A complete printing process is to open the port, print various functions, and close the port
Call-back interface, available or not, does not affect the normal printing process, call-back is only used for prompt message

7 Engineering code, including the header file, 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

typedef enum CP_ComDataBits { CP_ComDataBits_4 = 4, CP_ComDataBits_5 = 5, CP_ComDataBits_6 = 6, CP_ComDataBits_7 = 7, CP_ComDataBits_8 = 8 } CP_ComDataBits;

CP_ComParity

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

Syntax

typedef enum CP_ComParity { CP_ComParity_NoParity, CP_ComParity_OddParity, CP_ComParity_EvenParity, CP_ComParity_MarkParity, CP_ComParity_SpaceParity } CP_ComParity;

CP_ComStopBits

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

Syntax

typedef enum CP_ComStopBits { CP_ComStopBits_One, CP_ComStopBits_OnePointFive, CP_ComStopBits_Two } CP_ComStopBits;

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

typedef enum CP_ComFlowControl { CP_ComFlowControl_None, CP_ComFlowControl_XonXoff, CP_ComFlowControl_RtsCts, CP_ComFlowControl_DtrDsr } CP_ComFlowControl;

CP_CharacterSet

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

Syntax

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

CP_CharacterCodepage

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

Syntax

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

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

CP_MultiByteEncoding

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

Syntax

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

CP_ImageBinarizationMethod

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

Syntax

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

CP_ImageCompressionMethod

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

Syntax

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

CP_ImagePixelsFormat

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

Syntax

```
typedef enum CP_ImagePixelsFormat {
    CP_ImagePixelsFormat_MONO = 1,
    CP_ImagePixelsFormat_MONOLSB = 2,
    CP_ImagePixelsFormat_GRAY8 = 3,
    CP_ImagePixelsFormat_BYTEORDERED_RGB24 = 4,
    CP_ImagePixelsFormat_BYTEORDERED_BGR24 = 5,
    CP_ImagePixelsFormat_BYTEORDERED_ARGB32 = 6,
    CP_ImagePixelsFormat_BYTEORDERED_RGBA32 = 7,
    CP_ImagePixelsFormat_BYTEORDERED_ABGR32 = 8,
    CP_ImagePixelsFormat_BYTEORDERED_BGRA32 = 9
} CP_ImagePixelsFormat;
```

CP_ImagePixelsFormat_MONO = 1,
    Monochrome bitmap, high in front

CP_ImagePixelsFormat_MONOLSB = 2,
    Monochromatic bitmap, low in front

CP_ImagePixelsFormat_GRAY8 = 3,
    A grayscale image in which each color takes up one byte

CP_ImagePixelsFormat_BYTEORDERED_RGB24 = 4,
    R G B takes up one byte per color in byte order

CP_ImagePixelsFormat_BYTEORDERED_BGR24 = 5,
    B G R takes up one byte per color in byte order

CP_ImagePixelsFormat_BYTEORDERED_ARGB32 = 6,
    A R G B takes up one byte per color in byte order

CP_ImagePixelsFormat_BYTEORDERED_RGBA32 = 7,
    R G B A takes up one byte per color in byte order

CP_ImagePixelsFormat_BYTEORDERED_ABGR32 = 8,
    A B G R takes up one byte per color in byte order

CP_ImagePixelsFormat_BYTEORDERED_BGRA32 = 9
    B G R A takes up one byte per color in byte order

CP_QRCodeECC

Qr code error correction level

Syntax

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

CP_Pos_Alignment

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

Syntax

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

CP_Pos_BarcodeType

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

Syntax

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

CP_Pos_BarcodeTextPrintPosition

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

Syntax

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

CP_Page_DrawDirection

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

Syntax

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

CP_Page_DrawAlignment

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

Syntax

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

CP_Label_BarcodeType

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

Syntax

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

CP_Label_BarcodeTextPrintPosition

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

Syntax

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

CP_Label_Rotation

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

Syntax

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

CP_Label_Color

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

Syntax

typedef enum CP_Label_Color { CP_Label_Color_White, CP_Label_Color_Black } CP_Label_Color;

CP_PRINTERSTATUS

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

Syntax

```
#ifndef MarcoDefinitionCPPrinterStatus
#define MarcoDefinitionCPPrinterStatus
#define CP_PRINTERSTATUS_ERROR_CUTTER(error_status) (error_status & 0x01)
#define CP_PRINTERSTATUS_ERROR_FLASH(error_status) (error_status & 0x02)
#define CP_PRINTERSTATUS_ERROR_NOPAPER(error_status) (error_status & 0x04)
#define CP_PRINTERSTATUS_ERROR_VOLTAGE(error_status) (error_status & 0x08)
#define CP_PRINTERSTATUS_ERROR_MARKER(error_status) (error_status & 0x10)
#define CP_PRINTERSTATUS_ERROR_ENGINE(error_status) (error_status & 0x20)
#define CP_PRINTERSTATUS_ERROR_OVERHEAT(error_status) (error_status & 0x40)
#define CP_PRINTERSTATUS_ERROR_COVERUP(error_status) (error_status & 0x80)
#define CP_PRINTERSTATUS_ERROR_MOTOR(error_status) (error_status & 0x100)
#define CP_PRINTERSTATUS_INFO_LABELPAPER(info_status) (info_status & 0x02)
#define CP_PRINTERSTATUS_INFO_LABELMODE(info_status) (info_status & 0x04)
#define CP_PRINTERSTATUS_INFO_HAVEDATA(info_status) (info_status & 0x08)
#define CP_PRINTERSTATUS_INFO_NOPAPERCANCELED(info_status) (info_status & 0x10)
#define CP_PRINTERSTATUS_INFO_PAPERNOFETCH(info_status) (info_status & 0x20)
#define CP_PRINTERSTATUS_INFO_PRINTIDLE(info_status) (info_status & 0x40)
#define CP_PRINTERSTATUS_INFO_RECVIDLE(info_status) (info_status & 0x80)
#endif

// 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

```
#ifndef MarcoDefinitionCPRTStatus
#define MarcoDefinitionCPRTStatus
#define CP_RTSTATUS_DRAWER_OPENED(status) (((status >> 0) & 0x04) == 0x00)
#define CP_RTSTATUS_OFFLINE(status) (((status >> 0) & 0x08) == 0x08)
#define CP_RTSTATUS_COVERUP(status) (((status >> 8) & 0x04) == 0x04)
#define CP_RTSTATUS_FEED_PRESSED(status) (((status >> 8) & 0x08) == 0x08)
#define CP_RTSTATUS_NOPAPER(status) (((status >> 8) & 0x20) == 0x20)
#define CP_RTSTATUS_ERROR_OCCURED(status) (((status >> 8) & 0x40) == 0x40)
#define CP_RTSTATUS_CUTTER_ERROR(status) (((status >> 16) & 0x08) == 0x08)
#define CP_RTSTATUS_UNRECOVERABLE_ERROR(status) (((status >> 16) & 0x20) == 0x20)
#define CP_RTSTATUS_DEGREE_OR_VOLTAGE_OVERRANGE(status) (((status >> 16) & 0x40) == 0x40)
#define CP_RTSTATUS_PAPER_NEAREND(status) (((status >> 24) & 0x08) == 0x08)
#define CP_RTSTATUS_PAPER_TAKEOUT(status) (((status >> 24) & 0x04) == 0x04)
#endif

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

```
//          Unrecoverable Error
// DEGREE_OR_VOLTAGE_OVERRANGE
//          Degree or voltage error
// PAPER_NEAREND
//          Paper Near End
// PAPER_TAKEOUT
//          Paper takeout
```

CP_LABEL_TEXT_STYLE

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

Syntax

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

Callback

CP_OnNetPrinterDiscovered

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

Syntax

typedef void (*CP_OnNetPrinterDiscovered)(const char *local_ip, const char *discovered_mac, const char *discovered_ip, const char *discovered_name, const void *private_data);

CP_OnBluetoothDeviceDiscovered

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

Syntax

typedef void (*CP_OnBluetoothDeviceDiscovered)(const char *device_name, const char *device_address, const void *private_data);

CP_OnWiFiP2PDeviceDiscovered

When enumerating WiFiP2P devices, the function is called back when the incoming callback interface is searched for the WiFiP2P device

Syntax

typedef void (*CP_OnWiFiP2PDeviceDiscovered)(const char *device_name, const char *device_address, const char *device_type, const void *private_data);

CP_OnPortOpenedEvent

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

Syntax

typedef void (*CP_OnPortOpenedEvent)(void *handle, const char *name, void *private_data);

CP_OnPortOpenFailedEvent

When port opening fails, the function is called back

Syntax

typedef void (*CP_OnPortOpenFailedEvent)(void *handle, const char *name, void *private_data);

CP_OnPortClosedEvent

When the port is closed, the interface is called back

Syntax

typedef void (*CP_OnPortClosedEvent)(void *handle, void *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

typedef void (*CP_OnPortWrittenEvent)(void *handle, const unsigned char *buffer, unsigned int count, void *private_data);

CP_OnPortReceivedEvent

This function is called back when the port receives data

Syntax

typedef void (*CP_OnPortReceivedEvent)(void *handle, const unsigned char *buffer, unsigned int count, void *private_data);

CP_OnPrinterStatusEvent

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

Syntax

typedef void (*CP_OnPrinterStatusEvent)(void *handle, const long long printer_error_status, const long long printer_info_status, void *private_data);

CP_OnPrinterReceivedEvent

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

Syntax

typedef void (*CP_OnPrinterReceivedEvent)(void *handle, const unsigned int printer_received_byte_count, void *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

typedef void (*CP_OnPrinterPrintedEvent)(void *handle, const unsigned int printer_printed_page_id, void *private_data);

Add Callback and Remove Callback

CP_Port_AddOnPortOpenedEvent

Add Callback, Open Port Success

Syntax

AUTOREPLYPRINT_API int CP_Port_AddOnPortOpenedEvent(const CP_OnPortOpenedEvent event, void *private_data);

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

CP_Port_AddOnPortOpenFailedEvent

Add Callback, Open Port Failed

Syntax

AUTOREPLYPRINT_API int CP_Port_AddOnPortOpenFailedEvent(const CP_OnPortOpenFailedEvent event, void *private_data);

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

CP_Port_AddOnPortClosedEvent

Add Callback, Port Closed

Syntax

```
AUTOREPLYPRINT_API int CP_Port_AddOnPortClosedEvent(const CP_OnPortClosedEvent event, void *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

```
AUTOREPLYPRINT_API int CP_Port_AddOnPortWrittenEvent(const CP_OnPortWrittenEvent event, void *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

AUTOREPLYPRINT_API int CP_Port_AddOnPortReceivedEvent(const CP_OnPortReceivedEvent event, void *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_RemoveOnPortOpenedEvent

Remove Callback

Syntax

```
AUTOREPLYPRINT_API int CP_Port_RemoveOnPortOpenedEvent(const CP_OnPortOpenedEvent event);

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

CP_Port_RemoveOnPortOpenFailedEvent

Remove Callback

Syntax

```
AUTOREPLYPRINT_API int CP_Port_RemoveOnPortOpenFailedEvent(const CP_OnPortOpenFailedEvent event);

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

CP_Port_RemoveOnPortClosedEvent

Remove Callback

Syntax

```
AUTOREPLYPRINT_API int CP_Port_RemoveOnPortClosedEvent(const CP_OnPortClosedEvent event);

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

CP_Port_RemoveOnPortWrittenEvent

Remove Callback

Syntax

```
AUTOREPLYPRINT_API int CP_Port_RemoveOnPortWrittenEvent(const CP_OnPortWrittenEvent event);

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

CP_Port_RemoveOnPortReceivedEvent

Remove Callback

Syntax

```
AUTOREPLYPRINT_API int CP_Port_RemoveOnPortReceivedEvent(const CP_OnPortReceivedEvent event);

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

CP_Printer_AddOnPrinterStatusEvent

Add Callback, Printer Status Updated

Syntax

AUTOREPLYPRINT_API int CP_Printer_AddOnPrinterStatusEvent(const CP_OnPrinterStatusEvent event, void *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

AUTOREPLYPRINT_API int CP_Printer_AddOnPrinterReceivedEvent(const CP_OnPrinterReceivedEvent event, void *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

AUTOREPLYPRINT_API int CP_Printer_AddOnPrinterPrintedEvent(const CP_OnPrinterPrintedEvent event, void *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

```
AUTOREPLYPRINT_API int CP_Printer_RemoveOnPrinterStatusEvent(const CP_OnPrinterStatusEvent event);

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

CP_Printer_RemoveOnPrinterReceivedEvent

Remove Callback

Syntax

```
AUTOREPLYPRINT_API int CP_Printer_RemoveOnPrinterReceivedEvent(const CP_OnPrinterReceivedEvent event);

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

CP_Printer_RemoveOnPrinterPrintedEvent

Remove Callback

Syntax

```
AUTOREPLYPRINT_API int CP_Printer_RemoveOnPrinterPrintedEvent(const CP_OnPrinterPrintedEvent event);

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

Port Function

CP_Port_EnumCom

Enumerate serial port

Syntax

```
AUTOREPLYPRINT_API unsigned int CP_Port_EnumCom(char *pBuf, unsigned int cbBuf, unsigned int *pcbNeeded);

//      Enumerate serial port
//
//  pBuf
//      The buffer to save enumerated port list
//
//  cbBuf
//      The buffer size
//
//  pcbNeeded
//      The buffer bytes needed
//
//  return
//      Enumerated port count
```

CP_Port_EnumLpt

Enumerate parallel port

Syntax

```
AUTOREPLYPRINT_API unsigned int CP_Port_EnumLpt(char *pBuf, unsigned int cbBuf, unsigned int *pcbNeeded);

//      Enumerate parallel port
//
//  pBuf
//      The buffer to save enumerated port list
//
//  cbBuf
//      The buffer size
//
//  pcbNeeded
//      The buffer bytes needed
//
//  return
//      Enumerated port count
```

CP_Port_EnumUsb

Enumerate usb port

Syntax

AUTOREPLYPRINT_API unsigned int CP_Port_EnumUsb(char *pBuf, unsigned int cbBuf, unsigned int *pcbNeeded);

```
//      Enumerate usb port
//
//  pBuf
//      The buffer to save enumerated port list
//
//  cbBuf
//      The buffer size
//
//  pcbNeeded
//      The buffer bytes needed
//
//  return
//      Enumerated port count
```

CP_Port_EnumNetPrinter

Enumerate net printer

Syntax

AUTOREPLYPRINT_API void CP_Port_EnumNetPrinter(unsigned int timeout, int *cancel, CP_OnNetPrinterDiscovered on_discovered, const void *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_EnumBtDevice

Enumerate bt printer

Syntax

AUTOREPLYPRINT_API void CP_Port_EnumBtDevice(unsigned int timeout, int *cancel, CP_OnBluetoothDeviceDiscovered on_discovered, const void *private_data);

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

CP_Port_EnumBleDevice

Enumerate BLE printer

Syntax

AUTOREPLYPRINT_API void CP_Port_EnumBleDevice(unsigned int timeout, int *cancel, CP_OnBluetoothDeviceDiscovered on_discovered, const void *private_data);

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

CP_Port_EnumWiFiP2PDevice

Enumerate WiFi P2P printer

Syntax

AUTOREPLYPRINT_API void CP_Port_EnumWiFiP2PDevice(unsigned int timeout, int *cancel, CP_OnWiFiP2PDeviceDiscovered on_discovered, const void *private_data);

```
//      Enumerate WiFi P2P 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

AUTOREPLYPRINT_API void *CP_Port_OpenCom(const char *name, unsigned int baudrate, CP_ComDataBits databits, CP_ComParity parity, CP_ComStopBits stopbits, CP_ComFlowControl 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

```
AUTOREPLYPRINT_API void *CP_Port_OpenLpt(const char *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

AUTOREPLYPRINT_API void *CP_Port_OpenUsb(const char *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

AUTOREPLYPRINT_API void *CP_Port_OpenTcp(const char *local_ip, const char *dest_ip, unsigned short dest_port, unsigned int 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 Addres 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_OpenBtSpp

Connect Bluetooth Printer Via SPP

Syntax

```
AUTOREPLYPRINT_API void *CP_Port_OpenBtSpp(const char *address, int autoreplymode);

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

CP_Port_OpenBtBle

Connect Bluetooth Printer Via BLE

Syntax

```
AUTOREPLYPRINT_API void *CP_Port_OpenBtBle(const char *address, int autoreplymode);

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

CP_Port_OpenBtBleProtoV2

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

Syntax

```
AUTOREPLYPRINT_API void *CP_Port_OpenBtBleProtoV2(const char *address, int autoreplymode);

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

CP_Port_OpenMemoryBuffer

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

Syntax

```
AUTOREPLYPRINT_API void *CP_Port_OpenMemoryBuffer(unsigned int buffer_size);

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

CP_Port_GetMemoryBufferDataPointer

get memory buffer data pointer

Syntax

```
AUTOREPLYPRINT_API unsigned char *CP_Port_GetMemoryBufferDataPointer(void *handle);

//      get memory buffer data pointer
//
// handle
//      Port handle, returned by OpenXXX
//
// return
//      return memory buffer data pointer. or return zero means failed
```

CP_Port_GetMemoryBufferDataLength

get memory buffer data length

Syntax

```
AUTOREPLYPRINT_API unsigned int CP_Port_GetMemoryBufferDataLength(void *handle);

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

CP_Port_ClearMemoryBufferData

Clear memory buffer data

Syntax

```
AUTOREPLYPRINT_API int CP_Port_ClearMemoryBufferData(void *handle);

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

CP_Port_WiFiP2P_Connect

Connect WiFi P2P Printer

Syntax

```
AUTOREPLYPRINT_API int CP_Port_WiFiP2P_Connect(const char *device_address, unsigned int timeout);

//		Connect WiFi P2P Printer
//
// address
//		printer address
//		for example: "01:02:03:04:05:06"
//
// timeout
//		connect timeout ms, you can set to 10000
//
// return
//		return printer IP address (network byte ordered)
//		  If open success, return non-zero value, else return zero.
//
// remarks
//		only for android
```

CP_Port_WiFiP2P_Disconnect

Disconnect WiFi P2P Connection

Syntax

AUTOREPLYPRINT_API void CP_Port_WiFiP2P_Disconnect();

CP_Port_WiFiP2P_IsConnected

Check WiFi P2P Is Connected

Syntax

```
AUTOREPLYPRINT_API int CP_Port_WiFiP2P_IsConnected();

//      Check WiFi P2P Is Connected
//
// return
//      if connected, return true.
//      if not connected, return false.
```

CP_Port_Write

Write data to port

Syntax

AUTOREPLYPRINT_API int CP_Port_Write(void *handle, const unsigned char *buffer, unsigned int count, unsigned 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

AUTOREPLYPRINT_API int CP_Port_Read(void *handle, unsigned char *buffer, unsigned int count, unsigned 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

AUTOREPLYPRINT_API int CP_Port_ReadUntilByte(void *handle, unsigned char *buffer, unsigned int count, unsigned int timeout, unsigned char 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

```
AUTOREPLYPRINT_API int CP_Port_Available(void *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

```
AUTOREPLYPRINT_API int CP_Port_SkipAvailable(void *handle);

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

CP_Port_IsConnectionValid

Check Connection Valid

Syntax

```
AUTOREPLYPRINT_API int CP_Port_IsConnectionValid(void *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

```
AUTOREPLYPRINT_API int CP_Port_IsOpened(void *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

```
AUTOREPLYPRINT_API int CP_Port_Close(void *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

AUTOREPLYPRINT_API int CP_Printer_GetPrinterResolutionInfo(void *handle, unsigned int *width_mm, unsigned int *height_mm, unsigned int *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

AUTOREPLYPRINT_API int CP_Printer_GetPrinterFirmwareVersion(void *handle, char *pBuf, unsigned int cbBuf, unsigned int *pcbNeeded);

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

CP_Printer_GetPrinterStatusInfo

Get Printer Status

Syntax

AUTOREPLYPRINT_API int CP_Printer_GetPrinterStatusInfo(void *handle, long long *printer_error_status, long long *printer_info_status, long 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

AUTOREPLYPRINT_API int CP_Printer_GetPrinterReceivedInfo(void *handle, unsigned int *printer_received_byte_count, long 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

AUTOREPLYPRINT_API int CP_Printer_GetPrinterPrintedInfo(void *handle, unsigned int *printer_printed_page_id, long 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

AUTOREPLYPRINT_API int CP_Printer_GetPrinterLabelPositionAdjustmentInfo(void *handle, double *label_print_position_adjustment, double *label_tear_position_adjustment, long 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

AUTOREPLYPRINT_API int CP_Printer_SetPrinterLabelPositionAdjustmentInfo(void *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

```
AUTOREPLYPRINT_API int CP_Printer_ClearPrinterBuffer(void *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

```
AUTOREPLYPRINT_API int CP_Printer_ClearPrinterError(void *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

AUTOREPLYPRINT_API int CP_Pos_QueryRTStatus(void *handle, unsigned int 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

```
AUTOREPLYPRINT_API int CP_Pos_QueryPrintResult(void *handle, unsigned int 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

AUTOREPLYPRINT_API int CP_Pos_KickOutDrawer(void *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

```
AUTOREPLYPRINT_API int CP_Pos_Beep(void *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

```
AUTOREPLYPRINT_API int CP_Pos_FeedAndHalfCutPaper(void *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

```
AUTOREPLYPRINT_API int CP_Pos_FullCutPaper(void *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

AUTOREPLYPRINT_API int CP_Pos_HalfCutPaper(void *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

AUTOREPLYPRINT_API int CP_Pos_FeedLine(void *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

```
AUTOREPLYPRINT_API int CP_Pos_FeedDot(void *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

```
AUTOREPLYPRINT_API int CP_Pos_PrintSelfTestPage(void *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

```
AUTOREPLYPRINT_API int CP_Pos_PrintText(void *handle, const char *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

AUTOREPLYPRINT_API int CP_Pos_PrintTextInUTF8(void *handle, const wchar_t *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

```
AUTOREPLYPRINT_API int CP_Pos_PrintTextInGBK(void *handle, const wchar_t *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

AUTOREPLYPRINT_API int CP_Pos_PrintTextInBIG5(void *handle, const wchar_t *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

AUTOREPLYPRINT_API int CP_Pos_PrintTextInShiftJIS(void *handle, const wchar_t *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

AUTOREPLYPRINT_API int CP_Pos_PrintTextInEUCKR(void *handle, const wchar_t *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_PrintTextInBytes

print text

Syntax

AUTOREPLYPRINT_API int CP_Pos_PrintTextInBytes(void *handle, const char *str, int len);

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

CP_Pos_PrintBarcode

print 1D barcode

Syntax

AUTOREPLYPRINT_API int CP_Pos_PrintBarcode(void *handle, CP_Pos_BarcodeType nBarcodeType, const char *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

AUTOREPLYPRINT_API int CP_Pos_PrintBarcode_Code128Auto(void *handle, const char *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

AUTOREPLYPRINT_API int CP_Pos_PrintQRCode(void *handle, int nVersion, CP_QRCodeECC nECCLevel, const char *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

AUTOREPLYPRINT_API int CP_Pos_PrintQRCodeUseEpsonCmd(void *handle, int nQRCodeUnitWidth, CP_QRCodeECC nECCLevel, const char *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

AUTOREPLYPRINT_API int CP_Pos_PrintDoubleQRCode(void *handle, int nQRCodeUnitWidth, int nQR1Position, int nQR1Version, CP_QRCodeECC nQR1ECCLevel, const char *strQR1, int nQR2Position, int nQR2Version, CP_QRCodeECC nQR2ECCLevel, const char *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

AUTOREPLYPRINT_API int CP_Pos_PrintPDF417BarcodeUseEpsonCmd(void *handle, int columnCount, int rowCount, int unitWidth, int rowHeight, int nECCLevel, int dataProcessingMode, const char *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

AUTOREPLYPRINT_API int CP_Pos_PrintRasterImageFromFile(void *handle, int dstw, int dsth, const char *pszFile, CP_ImageBinarizationMethod binaryzation_method, CP_ImageCompressionMethod 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

AUTOREPLYPRINT_API int CP_Pos_PrintRasterImageFromData(void *handle, int dstw, int dsth, const unsigned char *data, unsigned int data_size, CP_ImageBinarizationMethod binaryzation_method, CP_ImageCompressionMethod 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

AUTOREPLYPRINT_API int CP_Pos_PrintRasterImageFromPixels(void *handle, const unsigned char *img_data, unsigned int img_datalen, int img_width, int img_height, int img_stride, CP_ImagePixelsFormat img_format, CP_ImageBinarizationMethod binaryzation_method, CP_ImageCompressionMethod 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

```
AUTOREPLYPRINT_API int CP_Pos_PrintHorizontalLine(void *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

AUTOREPLYPRINT_API int CP_Pos_PrintHorizontalLineSpecifyThickness(void *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

AUTOREPLYPRINT_API int CP_Pos_PrintMultipleHorizontalLinesAtOneRow(void *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

```
AUTOREPLYPRINT_API int CP_Pos_ResetPrinter(void *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 suppert)

Syntax

```
AUTOREPLYPRINT_API int CP_Pos_SetPrintSpeed(void *handle, int nSpeed);

//          set print speed (some printer suppert)
//
// 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 suppert)

Syntax

AUTOREPLYPRINT_API int CP_Pos_SetPrintDensity(void *handle, int nDensity);

```
//          set print density (some printer suppert)
//
// 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

```
AUTOREPLYPRINT_API int CP_Pos_SetSingleByteMode(void *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

```
AUTOREPLYPRINT_API int CP_Pos_SetCharacterSet(void *handle, CP_CharacterSet 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

AUTOREPLYPRINT_API int CP_Pos_SetCharacterCodepage(void *handle, CP_CharacterCodepage 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

```
AUTOREPLYPRINT_API int CP_Pos_SetMultiByteMode(void *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

AUTOREPLYPRINT_API int CP_Pos_SetMultiByteEncoding(void *handle, CP_MultiByteEncoding 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

AUTOREPLYPRINT_API int CP_Pos_SetMovementUnit(void *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

AUTOREPLYPRINT_API int CP_Pos_SetPrintAreaLeftMargin(void *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

```
AUTOREPLYPRINT_API int CP_Pos_SetPrintAreaWidth(void *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

```
AUTOREPLYPRINT_API int CP_Pos_SetHorizontalAbsolutePrintPosition(void *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

```
AUTOREPLYPRINT_API int CP_Pos_SetHorizontalRelativePrintPosition(void *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

```
AUTOREPLYPRINT_API int CP_Pos_SetVerticalAbsolutePrintPosition(void *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

```
AUTOREPLYPRINT_API int CP_Pos_SetVerticalRelativePrintPosition(void *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

AUTOREPLYPRINT_API int CP_Pos_SetAlignment(void *handle, CP_Pos_Alignment 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

AUTOREPLYPRINT_API int CP_Pos_SetTextScale(void *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

AUTOREPLYPRINT_API int CP_Pos_SetAsciiTextFontType(void *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

AUTOREPLYPRINT_API int CP_Pos_SetTextBold(void *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

AUTOREPLYPRINT_API int CP_Pos_SetTextUnderline(void *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

```
AUTOREPLYPRINT_API int CP_Pos_SetTextUpsideDown(void *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

AUTOREPLYPRINT_API int CP_Pos_SetTextWhiteOnBlack(void *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

AUTOREPLYPRINT_API int CP_Pos_SetTextRotate(void *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

AUTOREPLYPRINT_API int CP_Pos_SetTextLineHeight(void *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

AUTOREPLYPRINT_API int CP_Pos_SetAsciiTextCharRightSpacing(void *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

AUTOREPLYPRINT_API int CP_Pos_SetKanjiTextCharSpacing(void *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

```
AUTOREPLYPRINT_API int CP_Pos_SetBarcodeUnitWidth(void *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

```
AUTOREPLYPRINT_API int CP_Pos_SetBarcodeHeight(void *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

AUTOREPLYPRINT_API int CP_Pos_SetBarcodeReadableTextFontType(void *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

AUTOREPLYPRINT_API int CP_Pos_SetBarcodeReadableTextPosition(void *handle, CP_Pos_BarcodeTextPrintPosition 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

```
AUTOREPLYPRINT_API int CP_Page_SelectPageMode(void *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

AUTOREPLYPRINT_API int CP_Page_SelectPageModeEx(void *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

```
AUTOREPLYPRINT_API int CP_Page_ExitPageMode(void *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

```
AUTOREPLYPRINT_API int CP_Page_PrintPage(void *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

```
AUTOREPLYPRINT_API int CP_Page_ClearPage(void *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

AUTOREPLYPRINT_API int CP_Page_SetPageArea(void *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

```
AUTOREPLYPRINT_API int CP_Page_SetPageDrawDirection(void *handle, CP_Page_DrawDirection 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

AUTOREPLYPRINT_API int CP_Page_DrawRect(void *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

AUTOREPLYPRINT_API int CP_Page_DrawBox(void *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

```
AUTOREPLYPRINT_API int CP_Page_DrawText(void *handle, int x, int y, const char *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

```
AUTOREPLYPRINT_API int CP_Page_DrawTextInUTF8(void *handle, int x, int y, const wchar_t *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

AUTOREPLYPRINT_API int CP_Page_DrawTextInGBK(void *handle, int x, int y, const wchar_t *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

AUTOREPLYPRINT_API int CP_Page_DrawTextInBIG5(void *handle, int x, int y, const wchar_t *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

```
AUTOREPLYPRINT_API int CP_Page_DrawTextInShiftJIS(void *handle, int x, int y, const wchar_t *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

```
AUTOREPLYPRINT_API int CP_Page_DrawTextInEUCKR(void *handle, int x, int y, const wchar_t *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

AUTOREPLYPRINT_API int CP_Page_DrawBarcode(void *handle, int x, int y, CP_Pos_BarcodeType nBarcodeType, const char *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

AUTOREPLYPRINT_API int CP_Page_DrawQRCode(void *handle, int x, int y, int nVersion, CP_QRCodeECC nECCLevel, const char *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

AUTOREPLYPRINT_API int CP_Page_DrawRasterImageFromFile(void *handle, int x, int y, int dstw, int dsth, const char *pszFile, CP_ImageBinarizationMethod 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

AUTOREPLYPRINT_API int CP_Page_DrawRasterImageFromData(void *handle, int x, int y, int dstw, int dsth, const unsigned char *data, unsigned int data_size, CP_ImageBinarizationMethod 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

AUTOREPLYPRINT_API int CP_Page_DrawRasterImageFromPixels(void *handle, int x, int y, const unsigned char *img_data, unsigned int img_datalen, int img_width, int img_height, int img_stride, CP_ImagePixelsFormat img_format, CP_ImageBinarizationMethod 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

```
AUTOREPLYPRINT_API int CP_BlackMark_EnableBlackMarkMode(void *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

```
AUTOREPLYPRINT_API int CP_BlackMark_DisableBlackMarkMode(void *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

```
AUTOREPLYPRINT_API int CP_BlackMark_SetBlackMarkMaxFindLength(void *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

```
AUTOREPLYPRINT_API int CP_BlackMark_FindNextBlackMark(void *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

```
AUTOREPLYPRINT_API int CP_BlackMark_SetBlackMarkPaperPrintPosition(void *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

```
AUTOREPLYPRINT_API int CP_BlackMark_SetBlackMarkPaperCutPosition(void *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

AUTOREPLYPRINT_API int CP_BlackMark_FullCutBlackMarkPaper(void *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

AUTOREPLYPRINT_API int CP_BlackMark_HalfCutBlackMarkPaper(void *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

```
AUTOREPLYPRINT_API int CP_Label_EnableLabelMode(void *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

```
AUTOREPLYPRINT_API int CP_Label_DisableLabelMode(void *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

```
AUTOREPLYPRINT_API int CP_Label_CalibrateLabel(void *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

```
AUTOREPLYPRINT_API int CP_Label_FeedLabel(void *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_PageBegin

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

Syntax

AUTOREPLYPRINT_API int CP_Label_PageBegin(void *handle, int x, int y, int width, int height, CP_Label_Rotation 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

```
AUTOREPLYPRINT_API int CP_Label_PagePrint(void *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

AUTOREPLYPRINT_API int CP_Label_DrawText(void *handle, int x, int y, int font, int style, const char *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

AUTOREPLYPRINT_API int CP_Label_DrawTextInUTF8(void *handle, int x, int y, int font, int style, const wchar_t *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

AUTOREPLYPRINT_API int CP_Label_DrawTextInGBK(void *handle, int x, int y, int font, int style, const wchar_t *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_DrawTextInBIG5

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

Syntax

AUTOREPLYPRINT_API int CP_Label_DrawTextInBIG5(void *handle, int x, int y, int font, int style, const wchar_t *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 BIG5 encoding.
```

CP_Label_DrawTextInShiftJIS

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

Syntax

AUTOREPLYPRINT_API int CP_Label_DrawTextInShiftJIS(void *handle, int x, int y, int font, int style, const wchar_t *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 ShiftJIS encoding.
```

CP_Label_DrawTextInEUCKR

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

Syntax

AUTOREPLYPRINT_API int CP_Label_DrawTextInEUCKR(void *handle, int x, int y, int font, int style, const wchar_t *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 EUCKR encoding.
```

CP_Label_DrawTextInBytes

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

Syntax

AUTOREPLYPRINT_API int CP_Label_DrawTextInBytes(void *handle, int x, int y, int font, int style, const char *str, int len);

```
//      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
//
// len
//      the length to print
//
// return
//      If command is written successfully, it returns true else it returns false.
```

CP_Label_DrawBarcode

Draw 1D code in the assigned position of label page

Syntax

AUTOREPLYPRINT_API int CP_Label_DrawBarcode(void *handle, int x, int y, CP_Label_BarcodeType nBarcodeType, CP_Label_BarcodeTextPrintPosition nBarcodeTextPrintPosition, int height, int unitwidth, CP_Label_Rotation rotation, const char *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

AUTOREPLYPRINT_API int CP_Label_DrawQRCode(void *handle, int x, int y, int nVersion, CP_QRCodeECC nECCLevel, int unitwidth, CP_Label_Rotation rotation, const char *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_DrawQRCodeInUTF8

print qrcode in the assigned position of label page

Syntax

AUTOREPLYPRINT_API int CP_Label_DrawQRCodeInUTF8(void *handle, int x, int y, int nVersion, CP_QRCodeECC nECCLevel, int unitwidth, CP_Label_Rotation rotation, const wchar_t *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_DrawQRCodeInBytes

print qrcode in the assigned position of label page

Syntax

AUTOREPLYPRINT_API int CP_Label_DrawQRCodeInBytes(void *handle, int x, int y, int nVersion, CP_QRCodeECC nECCLevel, int unitwidth, CP_Label_Rotation rotation, const char *str, int len);

```
//      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
//
// len
//          the length 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

AUTOREPLYPRINT_API int CP_Label_DrawPDF417Code(void *handle, int x, int y, int column, int nAspectRatio, int nECCLevel, int unitwidth, CP_Label_Rotation rotation, const char *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

AUTOREPLYPRINT_API int CP_Label_DrawImageFromFile(void *handle, int x, int y, int dstw, int dsth, const char *pszFile, CP_ImageBinarizationMethod binaryzation_method, CP_ImageCompressionMethod 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

AUTOREPLYPRINT_API int CP_Label_DrawImageFromData(void *handle, int x, int y, int dstw, int dsth, const unsigned char *data, unsigned int data_size, CP_ImageBinarizationMethod binaryzation_method, CP_ImageCompressionMethod 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

AUTOREPLYPRINT_API int CP_Label_DrawImageFromPixels(void *handle, int x, int y, const unsigned char *img_data, unsigned int img_datalen, int img_width, int img_height, int img_stride, CP_ImagePixelsFormat img_format, CP_ImageBinarizationMethod binaryzation_method, CP_ImageCompressionMethod 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 stirde. means bytes per line.
//
// img_format
//      image pixel data format, values are defined as follow
//      value define
//      1       mono
//      2       monolsb
//      3       gray
//      4       r.g.b in byte-ordered
//      5       b.g.r in byte-ordered
//      6       a.r.g.b in byte-ordered
//      7       r.g.b.a in byte-ordered
```

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

CP_Label_DrawLine

draw line in the assigned position of label page

Syntax

AUTOREPLYPRINT_API int CP_Label_DrawLine(void *handle, int startx, int starty, int endx, int endy, int linewidth, CP_Label_Color 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

AUTOREPLYPRINT_API int CP_Label_DrawRect(void *handle, int x, int y, int width, int height, CP_Label_Color 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

AUTOREPLYPRINT_API int CP_Label_DrawBox(void *handle, int x, int y, int width, int height, int borderwidth, CP_Label_Color 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

```
AUTOREPLYPRINT_API const char *CP_Library_Version(void);

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

CP_Proto_QueryBatteryLevel

Query battery level

Syntax

```
AUTOREPLYPRINT_API int CP_Proto_QueryBatteryLevel(void *handle, unsigned int 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.
```