目录

开发包简介 6
常见问题说明 7
接口说明 8
枚举和宏定义 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
回调接口 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
添加移除回调 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
端口函数 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
获取打印机信息函数 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
票据函数 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
页模式函数 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
黑标函数 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
标签函数 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
其他函数....220
CP_Library_Version....220
CP_Proto_QueryBatteryLevel....221

开发包简介

1 开发包带有很多详细的例子，开发之前，请先运行相应的例子测试，测试完全没问题，再考虑开发。

2 开发包支持各种打印机，包括但不限于票据打印，标签打印，页模式打印，黑标打印。

3 开发包支持自动回传功能，开启该功能需要打印机支持自动回传。

4 开发包所有的函数都以 CP_作为前缀，避免与别家开发包混淆。

5 开发包主要由宏定义，枚举，回调，端口函数，票据打印函数，标签打印函数，页模式打印函数，
黑标相关函数组成。

端口函数以 CP_Port_开头，包括打开端口，关闭端口，读写端口等函数。
票据打印函数以 CP_Pos_开头，主要封装了各种票据指令，可以打印文本，条码，二维码，图片等
标签打印函数以 CP_Label_开头，主要封装了标签指令，可以打印文本，条码，二维码，图片等
页模式打印函数以 CP_Page_开头，主要封装了页模式相关指令，可以打印文本，条码，二维码，图片等
黑标相关函数以 CP_BlackMark_开头，主要封装了黑标定位相关指令

6 一个完整的打印流程是，打开端口，各种打印函数，关闭端口。
回调接口，可用可不用，不影响正常打印流程，回调只是用于提示信息。

7 工程代码里面，包含头文件，即可调用开发包的所有函数。

常见问题说明

1 怎么分辨我的打印机是什么机型，该用什么函数？
要注意看机型，是什么机型，用什么函数，用错了就打印不了，或者会打印乱码
放票据纸打印的，就是票据机型，使用 CP_Pos_系列函数进行打印
放标签纸的，就是标签机型，使用 CP_Label_系列函数进行打印
有些机器既可以放票据纸打印，也可以放标签纸打印的。可以调用函数，开启或关闭标签模式。
最好是咨询卖家，看该用什么例子测试，参考例子来写最省事。

2 为什么打印一半关机了？
看是不是电源不够，额定电压下，一般是要 2A 的电源就够了。
打印机开机时，指示灯的闪烁一般是不一样的，能很明显的看出来。
出现问题，要仔细观察指示灯或声音，这样便于定位问题。

3 标签打印完，为什么没有定位到缝隙？
看是不是开启了标签模式，标签是否识别到，测试方法是按一下进纸按键，是不是完整的进一张纸

接口说明

枚举和宏定义

CP_ComDataBits

串口数据位。打开串口时，需要指定数据位，一般为 8 位数据位。

定义

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

串口校验位。打开串口时，需要指定校验位，一般是无校验。

定义

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

CP_ComStopBits

串口停止位。打开串口时，需要指定停止位，一般是一位停止位。

定义

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

CP_ComFlowControl

串口流控制。打开串口时，需要指定流控制，一般选无流控或者软件流控。硬件流控需要线连对才能用。

定义

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

CP_CharacterSet

单字节模式下的国际字符集。当打印机处于单字节模式下时，设置打印机国际字符集，会改变 0x20-0x7F 这个区间的部分文字的打印。比如货币符号人命币或美元。具体细节请看打印机指令集部分。当打印机处于多字节模式下时，设置该属性无影响。

定义

```
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

单字节模式下的字符代码页。当打印机处于单字节模式下时，设置打印机字符代码页，会改变 0x80-0xFF 这个区间的部分文字的打印。具体细节请看打印机指令集部分。当打印机处于多字节模式下时，设置该属性无影响。

定义

```
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

多字节模式下的字符编码。打印机处于多字节模式下时，收到的打印数据，将按照指定的编码进行打印。比如说，设置打印机为多字节模式，再指定多字节模式下字符编码为 UTF8 编码，应用程序需按照 UTF8 编码发送字符串给打印机，打印机就会将字符串打印出来。

定义

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

图像二值化算法。由于打印机只能打印黑白单色位图，打印图像的过程中，如果原图是彩图或灰度图，则需要使用二值化算法，将原图转为单色图。不同的算法有不同的效果。
阈值算法适用于图片内容都是文字的。
误差扩散法适用于所有的图片，但细看会有毛刺。
抖动算法效果不如误差扩散法，不建议使用，仅做兼容性保留。

定义

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

CP_ImageCompressionMethod

图片压缩算法。部分打印机支持使用压缩指令打印图片，提高数据传输效率。具体是否支持需要看实际测试结果才知道。

定义

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

CP_ImagePixelsFormat

图片像素格式。打印图片时，如果是直接传的像素数据打印的，那么数据和格式要对应。

定义

```
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,
    单色位图，高位在前

CP_ImagePixelsFormat_MONOLSB = 2,
    单色位图，低位在前

CP_ImagePixelsFormat_GRAY8 = 3,
    灰度图，每个颜色占一个字节

CP_ImagePixelsFormat_BYTEORDERED_RGB24 = 4,
    按照字节顺序，R G B 每个颜色占一个字节

CP_ImagePixelsFormat_BYTEORDERED_BGR24 = 5,
    按照字节顺序，B G R 每个颜色占一个字节

CP_ImagePixelsFormat_BYTEORDERED_ARGB32 = 6,
    按照字节顺序，A R G B 每个颜色占一个字节

CP_ImagePixelsFormat_BYTEORDERED_RGBA32 = 7,
    按照字节顺序，R G B A 每个颜色占一个字节

CP_ImagePixelsFormat_BYTEORDERED_ABGR32 = 8,
    按照字节顺序，A B G R 每个颜色占一个字节

CP_ImagePixelsFormat_BYTEORDERED_BGRA32 = 9
    按照字节顺序，B G R A 每个颜色占一个字节

CP_QRCodeECC

二维码纠错等级。

定义

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

票据模式下打印对齐方式。有左对齐，中对齐，右对齐。

定义

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

CP_Pos_BarcodeType

票据指令打印条码时，指定条码类型。

定义

```
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

票据指令打印条码时，指定条码文字打印位置。

定义

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

页模式下打印时，指定页面绘制方向。

定义

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

页模式下的相关绘制函数，坐标如果是大于等于零，就是实际坐标。也可以指定为此处的特定值，指定在区域内对齐打印。

定义

```
#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

标签指令打印条码时，指定条码类型。

定义

```
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

标签指令打印条码时，指定条码文字打印位置。

定义

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

标签指令绘制控件时，指定旋转角度。

定义

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

标签指令绘制控件时，指定绘制颜色。可以是白色或者黑色。

定义

typedef enum CP_Label_Color { CP_Label_Color_White, CP_Label_Color_Black } CP_Label_Color;

CP_PRINTERSTATUS

打印机自动回传的状态定义。一般只需要关注是否有错误，信息部分主要是起到提示功能。

定义

```
#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
//        切刀错误
// ERROR_FLASH
//        串行 FLASH 错误
// ERROR_NOPAPER
//        缺纸
// ERROR_VOLTAGE
//        电压错误
// ERROR_MARKER
//        黑标或缝标侦测错误
// ERROR_ENGINE
//        机芯未识别
// ERROR_OVERHEAT
//        过热
// ERROR_COVERUP
//        开盖或轴未压下
// ERROR_MOTOR
//        马达失步(一般为卡纸)
// INFO_LABELPAPER
//        当前纸张识别为标签纸(为 0 是连续纸)
```

// INFO_LABELMODE
//　　当前为标签模式
// INFO_HAVEDATA
//　　有数据开始处理
// INFO_NOPAPERCANCELED
//　　上次单据是缺纸后取消了
// INFO_PAPERNOFETCH
//　　单据未取走
// INFO_PRINTIDLE
//　　当前打印空闲
// INFO_RECVIDLE
//　　当前接收缓冲区为空

CP_RTSTATUS

实时状态查询返回的状态定义。此处说明仅供参考，适用于大部分机型，部分机型如不一致，以实际机型指令集为准。

定义

```
#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

// 这里的实时状态，共占四字节。
// 从低字节到高字节依次对应指令集中这四个指令：
//     10 04 01
//     10 04 02
//     10 04 03
//     10 04 04
// 部分机型由于定制或其他原因，状态值定义可能与此处不一致，以实测为准。
//
// DRAWER_OPENED
//         钱箱打开
// OFFLINE
//         脱机
// COVERUP
//         盖子打开
// FEED_PRESSED
//         走纸键按下
// NOPAPER
//         缺纸
// ERROR_OCCURED
//         出错
// CUTTER_ERROR
//         切刀错误
// UNRECOVERABLE_ERROR
//         不可恢复错误
```

// DEGREE_OR_VOLTAGE_OVERRANGE
//　　温度或电压错误
// PAPER_NEAREND
//　　纸将近
// PAPER_TAKEOUT
//　　纸取走

CP_LABEL_TEXT_STYLE

标签指令打印文本时，指定文字打印风格。分别为加粗，下划线，反色，删除线，旋转，宽高加倍。

定义

```
#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
```

回调接口

CP_OnNetPrinterDiscovered

枚举网络打印机时，传入的回调函数。查到到网络打印机时，会回调该函数。

定义

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

枚举蓝牙设备时，传入的回调函数。搜索到蓝牙设备时，会回调该函数。

定义

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

CP_OnWiFiP2PDeviceDiscovered

枚举 WiFiP2P 设备时，传入的回调接口。搜索到 WiFiP2P 设备时，会回调该函数。

定义

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

CP_OnPortOpenedEvent

端口打开成功时，回调该函数。

定义

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

CP_OnPortOpenFailedEvent

端口打开失败时，回调该函数。

定义

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

CP_OnPortClosedEvent

端口关闭时，会回调该接口。

定义
typedef void (*CP_OnPortClosedEvent)(void *handle, void *private_data);

备注
端口异常时，比如 USB 数据线拔出，会自动关闭端口，并触发回调。

CP_OnPortWrittenEvent

端口写入数据成功时，会回调该函数。

定义

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

CP_OnPortReceivedEvent

端口收到数据时，会回调该函数。

定义

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

CP_OnPrinterStatusEvent

收到打印机自动回传的状态时，会回调该函数。

定义

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

CP_OnPrinterReceivedEvent

收到打印机自动回传的已接收字节数信息时，会回调该函数。

定义

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

CP_OnPrinterPrintedEvent

收到打印机自动回传的单据打完信息时，会回调该函数。该函数将弃用，不建议使用。

定义

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

添加移除回调

CP_Port_AddOnPortOpenedEvent

添加回调接口，端口打开成功

定义

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

//      添加回调接口，端口打开成功
//
//  event
//      回调接口
//
//  private_data
//      传给回调接口的参数
//
// return
//      true on success.
//      false on failed.
```

CP_Port_AddOnPortOpenFailedEvent

添加回调接口，端口打开失败

定义

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

```
//        添加回调接口，端口打开失败
//
//   event
//        回调接口
//
//   private_data
//        传给回调接口的参数
//
// return
//        true on success.
//        false on failed.
```

CP_Port_AddOnPortClosedEvent

添加回调接口，端口关闭

定义

```
AUTOREPLYPRINT_API int CP_Port_AddOnPortClosedEvent(const CP_OnPortClosedEvent event, void *private_data);

//		添加回调接口，端口关闭
//
//	event
//		回调接口
//
//	private_data
//		传给回调接口的参数
//
// return
//		true on success.
//		false on failed.
```

CP_Port_AddOnPortWrittenEvent

添加回调接口，端口写入数据

定义

```
AUTOREPLYPRINT_API int CP_Port_AddOnPortWrittenEvent(const CP_OnPortWrittenEvent event, void *private_data);

//		添加回调接口，端口写入数据
//
//	event
//		回调接口
//
//	private_data
//		传给回调接口的参数
//
// return
//		true on success.
//		false on failed.
```

CP_Port_AddOnPortReceivedEvent

添加回调接口，端口收到数据

定义

AUTOREPLYPRINT_API int CP_Port_AddOnPortReceivedEvent(const CP_OnPortReceivedEvent event, void *private_data);

```
//		添加回调接口，端口收到数据
//
//	event
//		回调接口
//
//	private_data
//		传给回调接口的参数
//
// return
//		true on success.
//		false on failed.
```

CP_Port_RemoveOnPortOpenedEvent

移除回调接口

定义

AUTOREPLYPRINT_API int CP_Port_RemoveOnPortOpenedEvent(const CP_OnPortOpenedEvent event);

```
//        移除回调接口
//
//   event
//        回调接口
//
// return
//        true on success.
//        false on failed.
```

CP_Port_RemoveOnPortOpenFailedEvent

移除回调接口

定义

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

//		移除回调接口
//
//	event
//		回调接口
//
// return
//		true on success.
//		false on failed.
```

CP_Port_RemoveOnPortClosedEvent

移除回调接口

定义

AUTOREPLYPRINT_API int CP_Port_RemoveOnPortClosedEvent(const CP_OnPortClosedEvent event);

```
//        移除回调接口
//
//   event
//        回调接口
//
// return
//        true on success.
//        false on failed.
```

CP_Port_RemoveOnPortWrittenEvent

移除回调接口

定义

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

//		移除回调接口
//
//	event
//		回调接口
//
// return
//		true on success.
//		false on failed.
```

CP_Port_RemoveOnPortReceivedEvent

移除回调接口

定义

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

//		移除回调接口
//
//	event
//		回调接口
//
// return
//		true on success.
//		false on failed.
```

CP_Printer_AddOnPrinterStatusEvent

添加回调接口，打印机状态更新

定义

AUTOREPLYPRINT_API int CP_Printer_AddOnPrinterStatusEvent(const CP_OnPrinterStatusEvent event, void *private_data);

```
//		添加回调接口，打印机状态更新
//
//	event
//		回调接口
//
//	private_data
//		传给回调接口的参数
//
// return
//		true on success.
//		false on failed.
```

CP_Printer_AddOnPrinterReceivedEvent

添加回调接口，打印机已接收字节数更新

定义

AUTOREPLYPRINT_API int CP_Printer_AddOnPrinterReceivedEvent(const CP_OnPrinterReceivedEvent event, void *private_data);

```
//        添加回调接口，打印机已接收字节数更新
//
//   event
//        回调接口
//
//   private_data
//        传给回调接口的参数
//
// return
//        true on success.
//        false on failed.
```

CP_Printer_AddOnPrinterPrintedEvent

添加回调接口，打印机已打印页面 ID 更新

定义

AUTOREPLYPRINT_API int CP_Printer_AddOnPrinterPrintedEvent(const CP_OnPrinterPrintedEvent event, void *private_data);

```
//        添加回调接口，打印机已打印页面 ID 更新
//
//   event
//        回调接口
//
//   private_data
//        传给回调接口的参数
//
// return
//        true on success.
//        false on failed.
```

CP_Printer_RemoveOnPrinterStatusEvent

移除回调接口

定义

AUTOREPLYPRINT_API int CP_Printer_RemoveOnPrinterStatusEvent(const CP_OnPrinterStatusEvent event);

```
//		移除回调接口
//
//	event
//		回调接口
//
// return
//		true on success.
//		false on failed.
```

CP_Printer_RemoveOnPrinterReceivedEvent

移除回调接口

定义

AUTOREPLYPRINT_API int CP_Printer_RemoveOnPrinterReceivedEvent(const CP_OnPrinterReceivedEvent event);

```
//		移除回调接口
//
//	event
//		回调接口
//
// return
//		true on success.
//		false on failed.
```

CP_Printer_RemoveOnPrinterPrintedEvent

移除回调接口

定义

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

//		移除回调接口
//
//	event
//		回调接口
//
// return
//		true on success.
//		false on failed.
```

端口函数

CP_Port_EnumCom

枚举本地串口

定义

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

```
//      枚举本地串口
//
//  pBuf
//      用来保存端口列表的缓冲区
//
//  cbBuf
//      缓冲区字节数
//
//  pcbNeeded
//      需要的缓冲区字节数
//
//  return
//      枚举到的端口数量
```

CP_Port_EnumLpt

枚举本地并口

定义

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

//      枚举本地并口
//
//  pBuf
//      用来保存端口列表的缓冲区
//
//  cbBuf
//      缓冲区字节数
//
//  pcbNeeded
//      需要的缓冲区字节数
//
//  return
//      枚举到的端口数量
```

CP_Port_EnumUsb

枚举本地 USB 打印口

定义

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

//      枚举本地 USB 打印口
//
//  pBuf
//      用来保存端口列表的缓冲区
//
//  cbBuf
//      缓冲区字节数
//
//  pcbNeeded
//      需要的缓冲区字节数
//
//  return
//      枚举到的端口数量
```

CP_Port_EnumNetPrinter

枚举网络打印机

定义

AUTOREPLYPRINT_API void CP_Port_EnumNetPrinter(unsigned int timeout, int *cancel, CP_OnNetPrinterDiscovered on_discovered, const void *private_data);

```
//      枚举网络打印机
//
//  timeout
//      超时毫秒时间
//
//  cancel
//      取消标记位，如果设为非零，则枚举提前退出
//
//  on_discovered
//      枚举回调接口
//
//  private_data
//      传给回调接口的参数
//
//  return
//      无
```

CP_Port_EnumBtDevice

枚举蓝牙打印机

定义

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

```
//      枚举蓝牙打印机
//
//  timeout
//      超时毫秒时间
//
//  cancel
//      取消标记位，如果设为非零，则枚举提前退出
//
//  on_discovered
//      枚举回调接口
//
//  private_data
//      传给回调接口的参数
//
//  return
//      无
```

CP_Port_EnumBleDevice

枚举 BLE 蓝牙打印机

定义

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

//      枚举 BLE 蓝牙打印机
//
//  timeout
//      超时毫秒时间
//
//  cancel
//      取消标记位，如果设为非零，则枚举提前退出
//
//  on_discovered
//      枚举回调接口
//
//  private_data
//      传给回调接口的参数
//
//  return
//      无
```

CP_Port_EnumWiFiP2PDevice

枚举 WiFi P2P 打印机

定义

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

```
//      枚举 WiFi P2P 打印机
//
//  timeout
//      超时毫秒时间
//
//  cancel
//      取消标记位，如果设为非零，则枚举提前退出
//
//  on_discovered
//      枚举回调接口
//
//  private_data
//      传给回调接口的参数
//
//  return
//      无
```

CP_Port_OpenCom

打开串口

定义

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);

```
//      打开串口
//
// name
//      端口名称
//      例如：COM1，COM2，COM3，...COM11...
//
// baudrate
//      波特率
//      一般取 9600,19200,38400,57600,115200.
//      需要和打印机波特率保持一致，建议使用高波特率以获得较好的打印速度
//
// databits
//      数据位，范围[4,8]
//
// parity
//      校验位，各值定义如下：
//      值      定义
//      0       无校验
//      1       奇校验
//      2       偶校验
//      3       标记校验
//      4       空白校验
//
// stopbits
//      停止位，各值定义如下：
//      值      定义
//      0       1 位停止位
//      1       1.5 位停止位
//      2       2 位停止位
//
// flowcontrol
//      流控制
//
// autoreplymode
//      0 不开启自动回传模式
//      1 开启自动回传模式
//      注意：
```

```
//          仅部分机型支持自动回传模式，是否支持请询问卖家
//          启动自动回传模式之后，打印机会自动回传状态
//          不启动则无法自动获取打印机状态
//
// return
//          返回打开的端口句柄。非零表示打开成功，零表示打开失败。
//
// remarks
//          如果串口被占用，打开串口会失败。
//          如果波特率和打印机波特率不匹配，则无法打印。
```

CP_Port_OpenLpt

打开并口

定义

```
AUTOREPLYPRINT_API void *CP_Port_OpenLpt(const char *name);

//		打开并口
//
// name
//		端口名称
//		例如：LPT1,LPT2,LPT3...
//
// return
//		返回打开的端口句柄。非零表示打开成功，零表示打开失败。
//
// remarks
//		并口只有单向通讯，只可写不可读。
//		一切查询状态的函数，对并口来说均是无效的。
```

CP_Port_OpenUsb

打开 USB

定义

```
AUTOREPLYPRINT_API void *CP_Port_OpenUsb(const char *name, int autoreplymode);

//      打开 USB
//
// name
//      端口名称
//      可由 EnumUsb 获得
//      也可以不指定，这时候，如果找到 USB 打印机，会直接打开
//
// autoreplymode
//      0 不开启自动回传模式
//      1 开启自动回传模式
//      注意：
//      仅部分机型支持自动回传模式，是否支持请询问卖家
//      启动自动回传模式之后，打印机会自动回传状态
//      不启动则无法自动获取打印机状态
//
// return
//      返回打开的端口句柄。非零表示打开成功，零表示打开失败。
//
// remarks
//      USB 打印机接到电脑上，如果设备管理器中出现了 USB Printing Support，则可以使用该函数打开。
```

CP_Port_OpenTcp

打开网口

定义

AUTOREPLYPRINT_API void *CP_Port_OpenTcp(const char *local_ip, const char *dest_ip, unsigned short dest_port, unsigned int timeout, int autoreplymode);

```
//      打开网口
//
// local_ip
//      绑定到本地 IP
//      用于多网卡或多个本地 IP 时，选择指定的 IP
//      传入 0 表示不指定
//
// dest_ip
//      地址或名称
//      例如：192.168.1.87
//
// dest_port
//      端口号
//      固定值：9100
//
// timeout
//      连接超时
//
// autoreplymode
//      0 不开启自动回传模式
//      1 开启自动回传模式
//      注意：
//      仅部分机型支持自动回传模式，是否支持请询问卖家
//      启动自动回传模式之后，打印机会自动回传状态
//      不启动则无法自动获取打印机状态
//
// return
//      返回打开的端口句柄。非零表示打开成功，零表示打开失败。
//
// remarks
//      PC 和打印机需要同网段的才可以连接
```

CP_Port_OpenBtSpp

通过 SPP 连接蓝牙打印机

定义

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

//          通过 SPP 连接蓝牙打印机
//
// address
//          打印机地址
//          例如："01:02:03:04:05:06"
//
// autoreplymode
//          0 不开启自动回传模式
//          1 开启自动回传模式
//          注意：
//          仅部分机型支持自动回传模式，是否支持请询问卖家
//          启动自动回传模式之后，打印机会自动回传状态
//          不启动则无法自动获取打印机状态
//
// return
//          返回打开的端口句柄。非零表示打开成功，零表示打开失败。
//
// remarks
//          only for android
```

CP_Port_OpenBtBle

通过 BLE 连接蓝牙打印机

定义

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

//          通过 BLE 连接蓝牙打印机
//
// address
//          打印机地址
//          例如："01:02:03:04:05:06"
//
// autoreplymode
//          0 不开启自动回传模式
//          1 开启自动回传模式
//          注意：
//          仅部分机型支持自动回传模式，是否支持请询问卖家
//          启动自动回传模式之后，打印机会自动回传状态
//          不启动则无法自动获取打印机状态
//
// return
//          返回打开的端口句柄。非零表示打开成功，零表示打开失败。
//
// remarks
//          only for android,ios,macos
```

CP_Port_OpenBtBleProtoV2

通过 BLE 连接蓝牙打印机（使用特定蓝牙协议传输数据）

定义

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

//		通过 BLE 连接蓝牙打印机（使用特定蓝牙协议传输数据）
//
// address
//		打印机地址
//		例如："01:02:03:04:05:06"
//
// autoreplymode
//		0 不开启自动回传模式
//		1 开启自动回传模式
//		注意：
//		仅部分机型支持自动回传模式，是否支持请询问卖家
//		启动自动回传模式之后，打印机会自动回传状态
//		不启动则无法自动获取打印机状态
//
// return
//		返回打开的端口句柄。非零表示打开成功，零表示打开失败。
//
// remarks
//		only for ios,macos
```

CP_Port_OpenMemoryBuffer

开辟一段内存缓冲区，后续所有的打印指令，均写入这个内存缓冲区

定义

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

//      开辟一段内存缓冲区，后续所有的打印指令，均写入这个内存缓冲区
//
// buffer_size
//      缓冲区大小
//
// return
//      返回打开的端口句柄。非零表示打开成功，零表示打开失败。
//
// remarks
//      这个可用于以下两个场景
//      场景一：
//          一张单据需要发送多个指令，希望将所有的指令拼在一起，一次性发送给打印机，以便提高传输速度
//      场景二：
//          有时候需要知道某个函数具体发送了什么指令，需要知道具体发送了什么数据
```

CP_Port_GetMemoryBufferDataPointer

获取内存缓冲区数据指针

定义

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

//		获取内存缓冲区数据指针
//
// handle
//		端口句柄，由 OpenXXX 返回
//
// return
//		返回内存缓冲区数据指针，零表示失败
```

CP_Port_GetMemoryBufferDataLength

获取内存缓冲区数据长度

定义

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

//		获取内存缓冲区数据长度
//
// handle
//		端口句柄，由 OpenXXX 返回
//
// return
//		返回内存缓冲区数据长度
```

CP_Port_ClearMemoryBufferData

清除内存缓冲区的数据

定义

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

//		清除内存缓冲区的数据
//
// handle
//		端口句柄，由 OpenXXX 返回
//
// return
//		true on success.
//		false on failed.
```

CP_Port_WiFiP2P_Connect

通过 WiFi P2P 连接打印机

定义

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

//		通过 WiFi P2P 连接打印机
//
// address
//		打印机地址
//		例如："01:02:03:04:05:06"
//
// timeout
//		连接超时毫秒时间，建议填 10000
//
// return
//		返回打印机的 IP 地址（网络字节序）。
//		非零表示连接成功，零表示连接失败。
//
// remarks
//		only for android
```

CP_Port_WiFiP2P_Disconnect

断开 WiFi P2P 连接

定义
AUTOREPLYPRINT_API void CP_Port_WiFiP2P_Disconnect();

CP_Port_WiFiP2P_IsConnected

WiFi P2P 是否已连接

定义

```
AUTOREPLYPRINT_API int CP_Port_WiFiP2P_IsConnected();

//          WiFi P2P 是否已连接
//
// return
//          如果已连接，则返回 true
//          如果未连接，则返回 false
```

CP_Port_Write

向端口写入数据

定义

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

```
//		向端口写入数据
//
// handle
//		端口句柄，由 OpenXXX 返回
//
// buffer
//		要写入的数据
//
// count
//		要写入的长度
//
// timeout
//		写入超时毫秒
//
// return
//		返回写入的字节数，-1 表示写入失败
```

CP_Port_Read

从端口接收数据

定义

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

```
//      从端口接收数据
//
// handle
//      端口句柄，由 OpenXXX 返回
//
// buffer
//      接收数据的缓冲区
//
// count
//      要接收的数据长度
//
// timeout
//      读取超时毫秒
//
// return
//      返回读取的字节数，-1 表示失败
```

CP_Port_ReadUntilByte

从端口接收数据

定义

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

```
//		从端口接收数据
//
// handle
//		端口句柄，由 OpenXXX 返回
//
// buffer
//		接收数据的缓冲区
//
// count
//		要接收的数据长度
//
// timeout
//		读取超时毫秒
//
// breakByte
//		结束读取字符
//
// return
//		返回读取的字节数，-1 表示失败
```

CP_Port_Available

返回可读取的字节数

定义

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

//          返回可读取的字节数
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// return
//          返回可读取的字节数，-1 表示失败
```

CP_Port_SkipAvailable

忽略接收缓冲区的数据

定义

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

//		忽略接收缓冲区的数据
//
// handle
//		端口句柄，由 OpenXXX 返回
//
// return
//		true on success.
//		false on failed.
```

CP_Port_IsConnectionValid

连接是否有效

定义

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

//      连接是否有效
//
// handle
//      端口句柄，由 OpenXXX 返回
//
// return
//      如果端口已打开，且状态持续更新，则返回 true
//      如果端口未打开，已关闭，或状态超过 6 秒未更新，则返回 false
```

CP_Port_IsOpened

检查端口是否打开

定义

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

//		检查端口是否打开
//
// handle
//		端口句柄，由 OpenXXX 返回
//
// return
//		如果端口已打开，且连接未断开未关闭，则返回 true
//		如果端口未打开，或连接已断开已关闭，则返回 false
```

CP_Port_Close

关闭端口

定义

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

//		关闭端口
//
// handle
//		端口句柄，由 OpenXXX 返回
//
// return
//		true on success.
//		false on failed.
```

获取打印机信息函数

CP_Printer_GetPrinterResolutionInfo

获取打印机分辨率信息

定义

AUTOREPLYPRINT_API int CP_Printer_GetPrinterResolutionInfo(void *handle, unsigned int *width_mm, unsigned int *height_mm, unsigned int *dots_per_mm);

```
//        获取打印机分辨率信息
//
// handle
//        端口句柄，由 OpenXXX 返回
//
//   width_mm
//        标签最大宽度
//
//   height_mm
//        标签最大高度
//
//   dots_per_mm
//        每毫米打印点数
//
// return
//        true on success.
//        false on failed.
```

CP_Printer_GetPrinterFirmwareVersion

获取打印机固件版本

定义

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

```
//        获取打印机固件版本
//
// handle
//        端口句柄，由 OpenXXX 返回
//
//   pBuf
//        缓冲区
//
//   cbBuf
//        缓冲区字节数
//
//   pcbNeeded
//        需要的缓冲区字节数
//
// return
//        true on success.
//        false on failed.
```

CP_Printer_GetPrinterStatusInfo

获取打印机自动回传的状态

定义

AUTOREPLYPRINT_API int CP_Printer_GetPrinterStatusInfo(void *handle, long long *printer_error_status, long long *printer_info_status, long long *timestamp_ms);

```
//		获取打印机自动回传的状态
//
// handle
//		端口句柄，由 OpenXXX 返回
//
//	printer_error_status
//		打印机错误状态
//
//	printer_info_status
//		打印机信息状态
//
//	timestamp_ms
//		时间戳
//
// return
//		true on success.
//		false on failed.
```

CP_Printer_GetPrinterReceivedInfo

获取打印机已接收字节数

定义

AUTOREPLYPRINT_API int CP_Printer_GetPrinterReceivedInfo(void *handle, unsigned int *printer_received_byte_count, long long *timestamp_ms);

```
//        获取打印机已接收字节数
//
// handle
//        端口句柄，由 OpenXXX 返回
//
//   printer_received_byte_count
//        打印机已接收字节数
//
//   timestamp_ms
//        时间戳
//
// return
//        true on success.
//        false on failed.
```

CP_Printer_GetPrinterPrintedInfo

获取打印机已打印的单据号

定义

AUTOREPLYPRINT_API int CP_Printer_GetPrinterPrintedInfo(void *handle, unsigned int *printer_printed_page_id, long long *timestamp_ms);

```
//		获取打印机已打印的单据号
//
// handle
//		端口句柄，由 OpenXXX 返回
//
//	printer_printed_page_id
//		打印机已打印的单据号
//
//	timestamp_ms
//		时间戳
//
// return
//		true on success.
//		false on failed.
```

CP_Printer_GetPrinterLabelPositionAdjustmentInfo

获取打印机标签位置微调信息

定义

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

```
//		获取打印机标签位置微调信息
//
// handle
//		端口句柄，由 OpenXXX 返回
//
//	label_print_position_adjustment
//		打印机标签打印位置微调
//
//	label_tear_position_adjustment
//		打印机标签撕纸位置微调
//
//	timestamp_ms
//		时间戳
//
// return
//		true on success.
//		false on failed.
```

CP_Printer_SetPrinterLabelPositionAdjustmentInfo

设置标签打印位置和撕纸位置微调

定义

AUTOREPLYPRINT_API int CP_Printer_SetPrinterLabelPositionAdjustmentInfo(void *handle, double label_print_position_adjustment, double label_tear_position_adjustment);

```
//      设置标签打印位置和撕纸位置微调
//
// handle
//      端口句柄，由 OpenXXX 返回
//
// label_print_position_adjustment
//      标签打印位置微调 mm（微调不超过±4mm）
//
// label_tear_position_adjustment
//      标签撕纸位置微调 mm（微调不超过±4mm）
//
// return
//      返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Printer_ClearPrinterBuffer

实时清除打印机缓存

定义

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

//		实时清除打印机缓存
//
// handle
//		端口句柄，由 OpenXXX 返回
//
// return
//		true on success.
//		false on failed.
```

CP_Printer_ClearPrinterError

实时清除打印机错误

定义

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

//		实时清除打印机错误
//
// handle
//		端口句柄，由 OpenXXX 返回
//
// return
//		true on success.
//		false on failed.
```

票据函数

CP_Pos_QueryRTStatus

查询打印机实时状态

定义

```
AUTOREPLYPRINT_API int CP_Pos_QueryRTStatus(void *handle, unsigned int timeout);

//        查询打印机实时状态
//        如果是支持自动回传的机型，状态会自动回传，不需要使用本指令查询
//        由于实时状态指令，无校验，无法保证结果一定正确
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// timeout
//        超时毫秒时间。
//        查询等待时间不超过此时间。
//
// return
//        返回值仅指示指令是否成功。成功返回实时状态，失败返回 0。
//        详细状态请查看 CP_RTSTATUS_XXX，如果状态定义与实际机型不符，以实测为准。
```

CP_Pos_QueryPrintResult

查询前面内容的打印结果

定义

```
AUTOREPLYPRINT_API int CP_Pos_QueryPrintResult(void *handle, unsigned int timeout);

//        查询前面内容的打印结果
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// timeout
//        超时毫秒时间。
//        查询打印结果等待时间不超过此时间。
//
// return
//        返回值仅指示指令是否打印成功。返回 true 表示打印成功，返回 false 表示打印失败或查询失败。
```

CP_Pos_KickOutDrawer

开钱箱（产生钱箱脉冲）

定义

AUTOREPLYPRINT_API int CP_Pos_KickOutDrawer(void *handle, int nDrawerIndex, int nHighLevelTime, int nLowLevelTime);

```
//        开钱箱（产生钱箱脉冲）
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// nDrawerIndex
//        钱箱编号，各值说明如下
//        编号          说明
//           0          钱箱引脚 2
//           1          钱箱引脚 5
//
// nHighLevelTime
//        高电平毫秒时间
//
// nLowLevelTime
//        低电平毫秒时间
//
// return
//        返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_Beep

蜂鸣器鸣叫

定义

```
AUTOREPLYPRINT_API int CP_Pos_Beep(void *handle, int nBeepCount, int nBeepMs);

//          蜂鸣器鸣叫
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// nBeepCount
//          鸣叫次数
//
// nBeepMs
//          蜂鸣毫秒时间，取值范围[100,900]。取整到百毫秒。
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_FeedAndHalfCutPaper

走纸到切刀位置并半切纸

定义

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

//          走纸到切刀位置并半切纸
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_FullCutPaper

切刀全切

定义

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

//          切刀全切
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_HalfCutPaper

切刀半切

定义

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

//          切刀半切
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_FeedLine

打印机进纸指定行数

定义

```
AUTOREPLYPRINT_API int CP_Pos_FeedLine(void *handle, int numLines);

//          打印机进纸指定行数
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// numLines
//          要进的行数
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_FeedDot

打印机进纸指定点数

定义

```
AUTOREPLYPRINT_API int CP_Pos_FeedDot(void *handle, int numDots);

//          打印机进纸指定点数
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// numDots
//          要进的点数
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_PrintSelfTestPage

打印机打印自检页

定义

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

//          打印机打印自检页
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_PrintText

打印文本

定义

```
AUTOREPLYPRINT_API int CP_Pos_PrintText(void *handle, const char *str);

//          打印文本
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// str
//          要打印的字符串
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_PrintTextInUTF8

打印文本

定义

```
AUTOREPLYPRINT_API int CP_Pos_PrintTextInUTF8(void *handle, const wchar_t *str);

//      打印文本
//
// handle
//      端口句柄，由 OpenXXX 返回
//
// str
//      要打印的字符串
//
// return
//      返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
//
// remarks
//      该函数会将数据转为 UTF8 编码发送。
```

CP_Pos_PrintTextInGBK

打印文本

定义

```
AUTOREPLYPRINT_API int CP_Pos_PrintTextInGBK(void *handle, const wchar_t *str);

//      打印文本
//
// handle
//      端口句柄，由 OpenXXX 返回
//
// str
//      要打印的字符串
//
// return
//      返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
//
// remarks
//      该函数会将数据转为 GBK 编码发送。
```

CP_Pos_PrintTextInBIG5

打印文本

定义

```
AUTOREPLYPRINT_API int CP_Pos_PrintTextInBIG5(void *handle, const wchar_t *str);

//      打印文本
//
// handle
//      端口句柄，由 OpenXXX 返回
//
// str
//      要打印的字符串
//
// return
//      返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
//
// remarks
//      该函数会将数据转为 BIG5 编码发送。
```

CP_Pos_PrintTextInShiftJIS

打印文本

定义

```
AUTOREPLYPRINT_API int CP_Pos_PrintTextInShiftJIS(void *handle, const wchar_t *str);

//          打印文本
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// str
//          要打印的字符串
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
//
// remarks
//          该函数会将数据转为 ShiftJIS 编码发送。
```

CP_Pos_PrintTextInEUCKR

打印文本

定义

```
AUTOREPLYPRINT_API int CP_Pos_PrintTextInEUCKR(void *handle, const wchar_t *str);

//          打印文本
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// str
//          要打印的字符串
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
//
// remarks
//          该函数会将数据转为 EUCKR 编码发送。
```

CP_Pos_PrintTextInBytes

打印文本

定义

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

//          打印文本
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// str
//          要打印的字符串
//
// len
//          要打印的长度
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_PrintBarcode

打印一维条码

定义

```
AUTOREPLYPRINT_API int CP_Pos_PrintBarcode(void *handle, CP_Pos_BarcodeType nBarcodeType, const char *str);

//      打印一维条码
//
// handle
//      端口句柄，由 OpenXXX 返回
//
// nBarcodeType
//      标识条码类型
//      各值定义如下：
//      值      类型
//      0x41        UPC-A
//      0x42        UPC-E
//      0x43        EAN13
//      0x44        EAN8
//      0x45        CODE39
//      0x46        ITF
//      0x47        CODABAR
//      0x48        CODE93
//      0x49        CODE128
//
// str
//      要打印的条码
//
// return
//      返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_PrintBarcode_Code128Auto

打印 CODE128 条码，该函数自动切换编码，以便节省空间

定义

```
AUTOREPLYPRINT_API int CP_Pos_PrintBarcode_Code128Auto(void *handle, const char *str);

//      打印 CODE128 条码，该函数自动切换编码，以便节省空间
//      正常情况下，请不要使用这个函数进行打印 CODE128 码
//      这个函数主要用于兼容部分老款机型
//      新款机型默认已经是支持自动切换编码的
//      新款机型使用这个函数是无法打印条码的
//
// handle
//      端口句柄，由 OpenXXX 返回
//
// str
//      要打印的条码
//
// return
//      返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_PrintQRCode

打印 QR 码

定义

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

```
//        打印 QR 码
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// nVersion
//        指定字符版本。取值范围：[0,16]。
//        当 version 为 0 时，打印机根据字符串长度自动计算版本号。
//
// nECCLevel
//        指定纠错等级。取值范围：[1, 4]。
//        各值定义如下：
//        ECC 纠错等级
//        1    L：7%，低纠错，数据多。
//        2    M：15%，中纠错
//        3    Q：优化纠错
//        4    H：30%，最高纠错，数据少。
//
// str
//        要打印的 QR 码
//
// return
//        返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_PrintQRCodeUseEpsonCmd

打印 QR 码

定义

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

```
//        打印 QR 码
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// nQRCodeUnitWidth
//        QRCode  码码块宽度，取值范围：[1, 16]。
//
// nECCLevel
//        指定纠错等级。取值范围：[1, 4]。
//        各值定义如下：
//        ECC  纠错等级
//        1     L：7%，低纠错，数据多。
//        2     M：15%，中纠错
//        3     Q：优化纠错
//        4     H：30%，最高纠错，数据少。
//
// str
//        要打印的 QR 码
//
// return
//        返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_PrintDoubleQRCode

打印两个 QR 码

定义

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);

```
//        打印两个 QR 码
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// nQRCodeUnitWidth
//        QRCode 码码块宽度，取值范围：[1, 8]。
//
// nQR1Position
// nQR2Position
//        QRCode position
//
// nQR1Version
// nQR2Version
//        指定字符版本。取值范围：[0,16]。
//        当 version 为 0 时，打印机根据字符串长度自动计算版本号。
//
// nQR1ECCLevel
// nQR2ECCLevel
//        指定纠错等级。取值范围：[1, 4]。
//        各值定义如下：
//        ECC 纠错等级
//        1    L：7%，低纠错，数据多。
//        2    M：15%，中纠错
//        3    Q：优化纠错
//        4    H：30%，最高纠错，数据少。
//
// strQR1
// strQR2
//        要打印的 QR 码
//
// return
//        返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_PrintPDF417BarcodeUseEpsonCmd

打印 PDF417 条码

定义

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

```
//        打印 PDF417 条码
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// columnCount
//        列数，取值范围[0,30]
//
// rowCount
//        行数，取值范围 0,[3,90]
//
// unitWidth
//        模块单元宽度，取值范围[2,8]
//
// rowHeight
//        行高，取值范围[2,8]
//
// nECCLevel
//        指定纠错等级。取值范围：[0,8]。
//
// dataProcessingMode
//        数据处理模式。0 选择标准 PDF417，1 选择截断 PDF417。
//
// str
//        要打印的 PDF417 码
//
// return
//        返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_PrintRasterImageFromFile

打印图片

定义

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

```
//      打印图片
//
// handle
//      端口句柄，由 OpenXXX 返回
//
// dstw
//      要打印的宽度
//
// dsth
//      要打印的高度
//
// pszFile
//      图片的路径
//
// binaryzation_method
//      图片二值化算法。0 表示抖动算法，1 表示阀值算法，2 表示误差扩散法。具体效果请测试查看。
//
// compression_method
//      最终打印数据的压缩方式，各值定义如下
//      值      定义
//      0       不压缩
//      1       一级压缩
//      2       二级压缩
//
// return
//      返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_PrintRasterImageFromData

打印图片（图片可由文件读取）

定义

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);

```
//      打印图片（图片可由文件读取）
//
// handle
//      端口句柄，由 OpenXXX 返回
//
// dstw
//      要打印的宽度
//
// dsth
//      要打印的高度
//
// data
//      图片数据。
//
// data_size
//      图片数据长度
//
// binaryzation_method
//      图片二值化算法。0 表示抖动算法，1 表示阀值算法，2 表示误差扩散法。具体效果请测试查看。
//
// compression_method
//      最终打印数据的压缩方式，各值定义如下
//      值      定义
//      0       不压缩
//      1       一级压缩
//      2       二级压缩
//
// return
//      返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_PrintRasterImageFromPixels

打印图片像素数据

定义

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);

```
//      打印图片像素数据
//
// handle
//      端口句柄，由 OpenXXX 返回
//
// img_data
//      图片的像素数据。
//
// img_datalen
//      图片的像素数据字节数。
//
// img_width
//      图片的像素宽度。
//
// img_height
//      图片的像素高度。
//
// img_stride
//      图片水平跨度。表示每行字节数。
//
// img_format
//      图片像素数据格式，各值定义如下
//      值      定义
//      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
//      图片二值化算法。0 表示抖动算法，1 表示阀值算法，2 表示误差扩散法。具体效果请测试查看。
//
```

// compression_method
//　　最终打印数据的压缩方式，各值定义如下
//　　值　　定义
//　　0　　不压缩
//　　1　　一级压缩
//　　2　　二级压缩
//
// return
//　　返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。

CP_Pos_PrintHorizontalLine

打印一条水平线

定义

```
AUTOREPLYPRINT_API int CP_Pos_PrintHorizontalLine(void *handle, int nLineStartPosition, int nLineEndPosition);

//		打印一条水平线
//
// handle
//		端口句柄，由 OpenXXX 返回
//
// nLineStartPosition
//		线段起点位置
//
// nLineEndPosition
//		线段终点位置
//
// return
//		返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_PrintHorizontalLineSpecifyThickness

打印一条水平线

定义

AUTOREPLYPRINT_API int CP_Pos_PrintHorizontalLineSpecifyThickness(void *handle, int nLineStartPosition, int nLineEndPosition, int nLineThickness);

```
//        打印一条水平线
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// nLineStartPosition
//        线段起点位置
//
// nLineEndPosition
//        线段终点位置
//
// nLineThickness
//        线段粗细
//
// return
//        返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_PrintMultipleHorizontalLinesAtOneRow

同一行上打印多条水平线，连续调用可打印曲线

定义

AUTOREPLYPRINT_API int CP_Pos_PrintMultipleHorizontalLinesAtOneRow(void *handle, int nLineCount, int *pLineStartPosition, int *pLineEndPosition);

```
//        同一行上打印多条水平线，连续调用可打印曲线
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// nLineCount
//        线段条数
//
// pLineStartPosition
//        线段起点位置
//
// pLineEndPosition
//        线段终点位置
//
// return
//        返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_ResetPrinter

复位打印机，清除设置

定义

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

//          复位打印机，清除设置
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_SetPrintSpeed

设置打印速度（部分机型支持）

定义

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

//		设置打印速度（部分机型支持）
//
// handle
//		端口句柄，由 OpenXXX 返回
//
// nSpeed
//		打印速度，单位毫米每秒
//
// return
//		返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_SetPrintDensity

设置打印浓度（部分机型支持）

定义

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

//          设置打印浓度（部分机型支持）
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// nDensity
//          设置打印浓度[0,15]
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_SetSingleByteMode

设置打印机为单字节编码

定义

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

//          设置打印机为单字节编码
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_SetCharacterSet

设置打印机字符集

定义

```
AUTOREPLYPRINT_API int CP_Pos_SetCharacterSet(void *handle, CP_CharacterSet nCharacterSet);

//          设置打印机字符集
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// nCharacterSet
//          打印机字符集，范围[0,15]
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_SetCharacterCodepage

设置字符代码页

定义

```
AUTOREPLYPRINT_API int CP_Pos_SetCharacterCodepage(void *handle, CP_CharacterCodepage nCharacterCodepage);

//      设置字符代码页
//
// handle
//      端口句柄，由 OpenXXX 返回
//
// nCharacterCodepage
//      字符代码页，范围[0,255]
//
// return
//      返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_SetMultiByteMode

设置打印机为多字节编码

定义

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

//          设置打印机为多字节编码
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_SetMultiByteEncoding

设置打印机多字节编码

定义

```
AUTOREPLYPRINT_API int CP_Pos_SetMultiByteEncoding(void *handle, CP_MultiByteEncoding nEncoding);

//        设置打印机多字节编码
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// nEncoding
//        多字节编码，各值定义如下：
//        值    定义
//        0     GBK
//        1     UTF8
//        3     BIG5
//        4     SHIFT-JIS
//        5     EUC-KR
//
// return
//        返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_SetMovementUnit

设置打印移动单位

定义

AUTOREPLYPRINT_API int CP_Pos_SetMovementUnit(void *handle, int nHorizontalMovementUnit, int nVerticalMovementUnit);

```
//        设置打印移动单位
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// nHorizontalMovementUnit
//        水平移动单位
//
// nVerticalMovementUnit
//        垂直移动单位
//
// return
//        返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
//
// remarks
//        移动单位设置为 200，则 1mm=8 点。
```

CP_Pos_SetPrintAreaLeftMargin

设置打印区域左边空白

定义

AUTOREPLYPRINT_API int CP_Pos_SetPrintAreaLeftMargin(void *handle, int nLeftMargin);

```
//        设置打印区域左边空白
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// nLeftMargin
//        左边空白
//
// return
//        返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_SetPrintAreaWidth

设置打印区域宽度

定义

```
AUTOREPLYPRINT_API int CP_Pos_SetPrintAreaWidth(void *handle, int nWidth);

//          设置打印区域宽度
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// nWidth
//          打印区域宽度
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_SetHorizontalAbsolutePrintPosition

设置横向绝对打印位置

定义

```
AUTOREPLYPRINT_API int CP_Pos_SetHorizontalAbsolutePrintPosition(void *handle, int nPosition);

//        设置横向绝对打印位置
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// nPosition
//        打印位置
//
// return
//        返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_SetHorizontalRelativePrintPosition

设置横向相对打印位置

定义

```
AUTOREPLYPRINT_API int CP_Pos_SetHorizontalRelativePrintPosition(void *handle, int nPosition);

//          设置横向相对打印位置
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// nPosition
//          打印位置
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_SetVerticalAbsolutePrintPosition

设置纵向绝对打印位置，仅在页模式下有效。

定义

```
AUTOREPLYPRINT_API int CP_Pos_SetVerticalAbsolutePrintPosition(void *handle, int nPosition);

//        设置纵向绝对打印位置，仅在页模式下有效。
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// nPosition
//        打印位置
//
// return
//        返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_SetVerticalRelativePrintPosition

设置纵向相对打印位置，仅在页模式下有效。

定义

```
AUTOREPLYPRINT_API int CP_Pos_SetVerticalRelativePrintPosition(void *handle, int nPosition);

//        设置纵向相对打印位置，仅在页模式下有效。
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// nPosition
//        打印位置
//
// return
//        返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_SetAlignment

设置打印对齐方式

定义

```
AUTOREPLYPRINT_API int CP_Pos_SetAlignment(void *handle, CP_Pos_Alignment nAlignment);

//          设置打印对齐方式
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// nAlignment
//          打印对齐方式，各值定义如下：
//          值    定义
//          0     左对齐
//          1     中对齐
//          2     右对齐
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_SetTextScale

设置文本放大倍数

定义

```
AUTOREPLYPRINT_API int CP_Pos_SetTextScale(void *handle, int nWidthScale, int nHeightScale);

//        设置文本放大倍数
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// nWidthScale
//        宽度放大倍数
//
// nHeightScale
//        高度放大倍数
//
// return
//        返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_SetAsciiTextFontType

设置英文字符字体类型

定义

```
AUTOREPLYPRINT_API int CP_Pos_SetAsciiTextFontType(void *handle, int nFontType);

//        设置英文字符字体类型
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// nFontType
//        英文字符字体类型，各值定义如下：
//        值   定义
//        0    字型 A（12x24）
//        1    字型 B（9x17）
//
// return
//        返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_SetTextBold

设置文本加粗打印

定义

```
AUTOREPLYPRINT_API int CP_Pos_SetTextBold(void *handle, int nBold);

//        设置文本加粗打印
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// nBold
//        是否加粗，各值定义如下：
//        值   定义
//        0    不加粗
//        1    加粗
//
// return
//        返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_SetTextUnderline

设置文本下划线

定义

```
AUTOREPLYPRINT_API int CP_Pos_SetTextUnderline(void *handle, int nUnderline);

//        设置文本下划线
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// nUnderline
//        文本下划线，各值定义如下：
//        值   定义
//        0    无下划线
//        1    1 点下划线
//        2    2 点下划线
//
// return
//        返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_SetTextUpsideDown

设置文本倒置打印

定义

```
AUTOREPLYPRINT_API int CP_Pos_SetTextUpsideDown(void *handle, int nUpsideDown);

//        设置文本倒置打印
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// nUpsideDown
//        倒置打印，各值定义如下：
//        值   定义
//        0    不倒置打印
//        1    倒置打印
//
// return
//        返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_SetTextWhiteOnBlack

设置黑白反显

定义

```
AUTOREPLYPRINT_API int CP_Pos_SetTextWhiteOnBlack(void *handle, int nWhiteOnBlack);

//		设置黑白反显
//
// handle
//		端口句柄，由 OpenXXX 返回
//
// nWhiteOnBlack
//		黑白反显，各值定义如下：
//		值	定义
//		0	不黑白反显
//		1	黑白反显
//
// return
//		返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_SetTextRotate

设置文本旋转 90 度打印

定义

```
AUTOREPLYPRINT_API int CP_Pos_SetTextRotate(void *handle, int nRotate);

//        设置文本旋转 90 度打印
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// nRotate
//        旋转打印，各值定义如下：
//        值   定义
//        0    不旋转打印
//        1    旋转 90 度打印
//
// return
//        返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_SetTextLineHeight

设置行高

定义

```
AUTOREPLYPRINT_API int CP_Pos_SetTextLineHeight(void *handle, int nLineHeight);

//		设置行高
//
// handle
//		端口句柄，由 OpenXXX 返回
//
// nLineHeight
//		行高，范围[1,255]
//
// return
//		返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_SetAsciiTextCharRightSpacing

设置 ASCII 字符右边空白

定义

```
AUTOREPLYPRINT_API int CP_Pos_SetAsciiTextCharRightSpacing(void *handle, int nSpacing);

//          设置 ASCII 字符右边空白
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// nSpacing
//          右边空白，范围[1,255]
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_SetKanjiTextCharSpacing

设置汉字文本字符左边空白和右边空白

定义

```
AUTOREPLYPRINT_API int CP_Pos_SetKanjiTextCharSpacing(void *handle, int nLeftSpacing, int nRightSpacing);

//		设置汉字文本字符左边空白和右边空白
//
// handle
//		端口句柄，由 OpenXXX 返回
//
// nLeftSpacing
//		右边空白，范围[1,255]
//
// nRightSpacing
//		右边空白，范围[1,255]
//
// return
//		返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_SetBarcodeUnitWidth

设置条码和二维码单元宽度

定义

```
AUTOREPLYPRINT_API int CP_Pos_SetBarcodeUnitWidth(void *handle, int nBarcodeUnitWidth);

//          设置条码和二维码单元宽度
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// nBarcodeUnitWidth
//          条码单元宽度，取值范围：[1,6]
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_SetBarcodeHeight

设置条码高度

定义

```
AUTOREPLYPRINT_API int CP_Pos_SetBarcodeHeight(void *handle, int nBarcodeHeight);

//          设置条码高度
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// nBarcodeHeight
//          定义条码高度。取值范围：[1,255]
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_SetBarcodeReadableTextFontType

设置条码可读字符字体类型

定义

```
AUTOREPLYPRINT_API int CP_Pos_SetBarcodeReadableTextFontType(void *handle, int nFontType);

//          设置条码可读字符字体类型
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// nFontType
//          指定可读字符的字体类型，各值定义如下：
//          值    类型
//          0     标准 ASCII
//          1     压缩 ASCII
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Pos_SetBarcodeReadableTextPosition

设置条码可读字符打印位置

定义

AUTOREPLYPRINT_API int CP_Pos_SetBarcodeReadableTextPosition(void *handle, CP_Pos_BarcodeTextPrintPosition nTextPosition);

```
//        设置条码可读字符打印位置
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// nTextPosition
//        条码可读字符位置，取值范围：[0, 3].
//        各值定义如下：
//        值  定义
//        0  不显示可读字符
//        1  在条码下方显示可读字符
//        2  在条码上方显示可读字符
//        3  在条码上方和条码下方显示可读字符
//
// return
//        返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

页模式函数

CP_Page_SelectPageMode

选择页模式

定义

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

//          选择页模式
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Page_SelectPageModeEx

选择页模式，并设置移动单位和页面大小，还会设置其他一系列参数为默认值

定义

AUTOREPLYPRINT_API int CP_Page_SelectPageModeEx(void *handle, int nHorizontalMovementUnit, int nVerticalMovementUnit, int x, int y, int width, int height);

```
//		选择页模式，并设置移动单位和页面大小，还会设置其他一系列参数为默认值
//
// handle
//		端口句柄，由 OpenXXX 返回
//
// nHorizontalMovementUnit
//		水平移动单位
//
// nVerticalMovementUnit
//		垂直移动单位
//
// x
//		横向起始位置
//
// y
//		纵向起始位置
//
// width
//		打印区域宽度
//
// height
//		打印区域高度
//
// return
//		返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Page_ExitPageMode

退出页模式并回到标准模式

定义

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

//		退出页模式并回到标准模式
//
// handle
//		端口句柄，由 OpenXXX 返回
//
// return
//		返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Page_PrintPage

页模式下打印内容

定义

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

//		页模式下打印内容
//
// handle
//		端口句柄，由 OpenXXX 返回
//
// return
//		返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Page_ClearPage

页模式下清除页面

定义

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

//		页模式下清除页面
//
// handle
//		端口句柄，由 OpenXXX 返回
//
// return
//		返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Page_SetPageArea

页模式下设置页区域，页面最高 2000 点（1mm8 点）

定义

```
AUTOREPLYPRINT_API int CP_Page_SetPageArea(void *handle, int x, int y, int width, int height);

//		页模式下设置页区域，页面最高 2000 点（1mm8 点）
//
// handle
//		端口句柄，由 OpenXXX 返回
//
// x
//		横向起始位置
//
// y
//		纵向起始位置
//
// width
//		打印区域宽度
//
// height
//		打印区域高度
//
// return
//		返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Page_SetPageDrawDirection

页模式下设置打印方向

定义

```
AUTOREPLYPRINT_API int CP_Page_SetPageDrawDirection(void *handle, CP_Page_DrawDirection nDirection);

//		页模式下设置打印方向
//
// handle
//		端口句柄，由 OpenXXX 返回
//
// nDirection
//		打印区域方向，各值定义如下：
//		0	从左到右
//		1	从下到上
//		2	从右到左
//		3	从上到下
//
// return
//		返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Page_DrawRect

页模式下画矩形

定义
AUTOREPLYPRINT_API int CP_Page_DrawRect(void *handle, int x, int y, int width, int height, int color);

```
//		页模式下画矩形
//
// handle
//		端口句柄，由 OpenXXX 返回
//
// x
//		横向坐标
//
// y
//		纵向坐标
//
// width
//		矩形宽度
//
// height
//		矩形高度
//
// color
//		矩形颜色，0 是白色，1 是黑色
//
// return
//		返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Page_DrawBox

页模式下画矩形框

定义

AUTOREPLYPRINT_API int CP_Page_DrawBox(void *handle, int x, int y, int width, int height, int borderwidth, int bordercolor);

```
//		页模式下画矩形框
//
// handle
//		端口句柄，由 OpenXXX 返回
//
// x
//		横向坐标
//
// y
//		纵向坐标
//
// width
//		矩形框宽度
//
// height
//		矩形框高度
//
// borderwidth
//		矩形框边框宽度
//
// bordercolor
//		矩形框边框颜色，0 是白色，1 是黑色
//
// return
//		返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Page_DrawText

页模式下画文本

定义
AUTOREPLYPRINT_API int CP_Page_DrawText(void *handle, int x, int y, const char *str);

```
//      页模式下画文本
//
// handle
//      端口句柄，由 OpenXXX 返回
//
// x
//      横向坐标
//
// y
//      纵向坐标
//
// str
//      要打印的字符串
//
// return
//      返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Page_DrawTextInUTF8

页模式下画文本

定义

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

//		页模式下画文本
//
// handle
//		端口句柄，由 OpenXXX 返回
//
// x
//		横向坐标
//
// y
//		纵向坐标
//
// str
//		要打印的字符串
//
// return
//		返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
//
// remarks
//		该函数会将数据转为 UTF8 编码发送。
```

CP_Page_DrawTextInGBK

页模式下画文本

定义

```
AUTOREPLYPRINT_API int CP_Page_DrawTextInGBK(void *handle, int x, int y, const wchar_t *str);

//      页模式下画文本
//
// handle
//      端口句柄，由 OpenXXX 返回
//
// x
//      横向坐标
//
// y
//      纵向坐标
//
// str
//      要打印的字符串
//
// return
//      返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
//
// remarks
//      该函数会将数据转为 GBK 编码发送。
```

CP_Page_DrawTextInBIG5

页模式下画文本

定义

```
AUTOREPLYPRINT_API int CP_Page_DrawTextInBIG5(void *handle, int x, int y, const wchar_t *str);

//      页模式下画文本
//
// handle
//      端口句柄，由 OpenXXX 返回
//
// x
//      横向坐标
//
// y
//      纵向坐标
//
// str
//      要打印的字符串
//
// return
//      返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
//
// remarks
//      该函数会将数据转为 BIG5 编码发送。
```

CP_Page_DrawTextInShiftJIS

页模式下画文本

定义

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

//      页模式下画文本
//
// handle
//      端口句柄，由 OpenXXX 返回
//
// x
//      横向坐标
//
// y
//      纵向坐标
//
// str
//      要打印的字符串
//
// return
//      返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
//
// remarks
//      该函数会将数据转为 ShiftJIS 编码发送。
```

CP_Page_DrawTextInEUCKR

页模式下画文本

定义

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

//      页模式下画文本
//
// handle
//      端口句柄，由 OpenXXX 返回
//
// x
//      横向坐标
//
// y
//      纵向坐标
//
// str
//      要打印的字符串
//
// return
//      返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
//
// remarks
//      该函数会将数据转为 EUCKR 编码发送。
```

CP_Page_DrawBarcode

页模式下打印一维条码

定义

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

```
//        页模式下打印一维条码
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// x
//        横向坐标
//
// y
//        纵向坐标
//
// nBarcodeType
//        标识条码类型
//        各值定义如下：
//        值      类型
//        0x41        UPC-A
//        0x42        UPC-E
//        0x43        EAN13
//        0x44        EAN8
//        0x45        CODE39
//        0x46        ITF
//        0x47        CODABAR
//        0x48        CODE93
//        0x49        CODE128
//
// str
//        要打印的条码
//
// return
//        返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Page_DrawQRCode

页模式下打印 QR 码

定义

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

```
//        页模式下打印 QR 码
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// x
//        横向坐标
//
// y
//        纵向坐标
//
// nVersion
//        指定字符版本。取值范围：[0,16]。
//        当 version 为 0 时，打印机根据字符串长度自动计算版本号。
//
// nECCLevel
//        指定纠错等级。取值范围：[1, 4]。
//        各值定义如下：
//        ECC 纠错等级
//        1    L：7%，低纠错，数据多。
//        2    M：15%，中纠错
//        3    Q：优化纠错
//        4    H：30%，最高纠错，数据少。
//
// str
//        要打印的 QR 码
//
// return
//        返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Page_DrawRasterImageFromFile

页模式下打印图片

定义

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

```
//		页模式下打印图片
//
// handle
//		端口句柄，由 OpenXXX 返回
//
// x
//		横向坐标
//
// y
//		纵向坐标
//
// dstw
//		要打印的宽度
//
// dsth
//		要打印的高度
//
// pszFile
//		图片的路径
//
// binaryzation_method
//		图片二值化算法。0 表示抖动算法，1 表示阀值算法，2 表示误差扩散法。具体效果请测试查看。
//
// return
//		返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Page_DrawRasterImageFromData

页模式下打印图片（图片可由文件读取）

定义

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);

```
//        页模式下打印图片（图片可由文件读取）
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// x
//        横向坐标
//
// y
//        纵向坐标
//
// dstw
//        要打印的宽度
//
// dsth
//        要打印的高度
//
// data
//        图片数据。
//
// data_size
//        图片数据长度
//
// binaryzation_method
//        图片二值化算法。0 表示抖动算法，1 表示阀值算法，2 表示误差扩散法，2 表示误差扩散法。具体效果请测试查看。
//
// return
//        返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Page_DrawRasterImageFromPixels

页模式下打印图片像素数据

定义

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);

```
//      页模式下打印图片像素数据
//
// handle
//      端口句柄，由 OpenXXX 返回
//
// x
//      横向坐标
//
// y
//      纵向坐标
//
// img_data
//      图片的像素数据。
//
// img_datalen
//      图片的像素数据字节数。
//
// img_width
//      图片的像素宽度。
//
// img_height
//      图片的像素高度。
//
// img_stride
//      图片水平跨度。表示每行字节数。
//
// img_format
//      图片像素数据格式，各值定义如下
//      值      定义
//      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
//          图片二值化算法。0 表示抖动算法，1 表示阀值算法，2 表示误差扩散法。具体效果请测试查看。
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

黑标函数

CP_BlackMark_EnableBlackMarkMode

启用黑标模式，重启打印机生效

定义

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

//          启用黑标模式，重启打印机生效
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_BlackMark_DisableBlackMarkMode

禁用黑标模式

定义

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

//		禁用黑标模式
//
// handle
//		端口句柄，由 OpenXXX 返回
//
// return
//		返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_BlackMark_SetBlackMarkMaxFindLength

设置黑标最大查找距离（重启仍有效）

定义

```
AUTOREPLYPRINT_API int CP_BlackMark_SetBlackMarkMaxFindLength(void *handle, int maxFindLength);

//        设置黑标最大查找距离（重启仍有效）
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// maxFindLength
//        最大查找距离（maxFindLength x 0.125  毫米）
//
// return
//        返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_BlackMark_FindNextBlackMark

查找下一个黑标

定义

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

//		查找下一个黑标
//
// handle
//		端口句柄，由 OpenXXX 返回
//
// return
//		返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_BlackMark_SetBlackMarkPaperPrintPosition

黑标模式下，设置起始打印位置的调整值

定义

```
AUTOREPLYPRINT_API int CP_BlackMark_SetBlackMarkPaperPrintPosition(void *handle, int position);

//        黑标模式下，设置起始打印位置的调整值
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// position
//        大于 0 则指定为进纸，小于 0 则指定为退纸。距离为  position x 0.125  毫米。
//
// return
//        返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_BlackMark_SetBlackMarkPaperCutPosition

黑标模式下，设置切纸位置

定义

```
AUTOREPLYPRINT_API int CP_BlackMark_SetBlackMarkPaperCutPosition(void *handle, int position);

//          黑标模式下，设置切纸位置
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// position
//          大于 0 则指定为进纸，小于 0 则指定为退纸。距离为  position x 0.125  毫米。
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_BlackMark_FullCutBlackMarkPaper

切刀全切

定义

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

//          切刀全切
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_BlackMark_HalfCutBlackMarkPaper

切刀半切

定义

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

//          切刀半切
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

标签函数

CP_Label_EnableLabelMode

启用标签模式

定义

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

//          启用标签模式
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Label_DisableLabelMode

关闭标签模式

定义

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

//          关闭标签模式
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Label_CalibrateLabel

校准标签纸（更换不同规格标签纸，需要校准）

定义

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

//        校准标签纸（更换不同规格标签纸，需要校准）
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// return
//        返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Label_FeedLabel

走纸到标签缝隙处

定义

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

//          走纸到标签缝隙处
//
// handle
//          端口句柄，由 OpenXXX 返回
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Label_PageBegin

指示一个标签页面的开始，并设置标签页的大小，参考点坐标和页面旋转角度

定义

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

```
//        指示一个标签页面的开始，并设置标签页的大小，参考点坐标和页面旋转角度
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// x
//        页面起始点 x 坐标
//
// y
//        页面起始点 y 坐标
//
// width
//        页面页宽
//
// height
//        页面页高
//
// rotation
//        页面旋转。 rotate 的取值范围为{0,1}。为 0，页面不旋转打印，为 1，页面旋转 90 度打印。
//
// return
//        返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Label_PagePrint

将标签页上的内容打印到标签纸上

定义

```
AUTOREPLYPRINT_API int CP_Label_PagePrint(void *handle, int copies);

//		将标签页上的内容打印到标签纸上
//
// handle
//		端口句柄，由 OpenXXX 返回
//
// copies
//		份数 [ 1 - 255 ]
//
// return
//		返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Label_DrawText

在标签页面上指定位置绘制文本。只能单行打印。

定义

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

//      在标签页面上指定位置绘制文本。只能单行打印。
//
// handle
//      端口句柄，由 OpenXXX 返回
//
// x
//      定义文本起始位置 x 坐标，取值范围：[0, Page_Width-1]
//
// y
//      定义文本起始位置 y 坐标，取值范围：[0, Page_Height-1]
//
// font
//      选择字体，可以使用 24。
//      带矢量字机型支持 16,[20,99]。
//
// style
//      字符风格。
//      数据位                  定义
//      0 加粗标志位：          置 1 字体加粗，清零则字体不加粗。
//      1 下划线标志位：        置 1 文本带下划线，清零则无下划线。
//      2 反白标志位：          置 1 文本反白(黑底白字)，清零不反白。
//      3 删除线标志位：        置 1 文本带删除线，清零则无删除线。
//      [5,4] 旋转标志位：    00 旋转 0° ；
//                              01 旋转 90°；
//                              10 旋转 180°；
//                              11 旋转 270°；
//      [11,8] 字体宽度放大倍数；
//      [15,12] 字体高度放大倍数；
//
// str
//      要打印的字符串
//
// return
//      返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Label_DrawTextInUTF8

在标签页面上指定位置绘制文本。只能单行打印。

定义

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

```
//      在标签页面上指定位置绘制文本。只能单行打印。
//
// handle
//      端口句柄，由 OpenXXX 返回
//
// x
//      定义文本起始位置 x 坐标，取值范围：[0, Page_Width-1]
//
// y
//      定义文本起始位置 y 坐标，取值范围：[0, Page_Height-1]
//
// font
//      选择字体，可以使用 24。
//      带矢量字机型支持 16,[20,99]。
//
// style
//      字符风格。
//      数据位                  定义
//      0 加粗标志位：          置 1 字体加粗，清零则字体不加粗。
//      1 下划线标志位：        置 1 文本带下划线，清零则无下划线。
//      2 反白标志位：          置 1 文本反白(黑底白字)，清零不反白。
//      3 删除线标志位：        置 1 文本带删除线，清零则无删除线。
//      [5,4] 旋转标志位：    00 旋转 0° ；
//                              01 旋转 90°；
//                              10 旋转 180°；
//                              11 旋转 270°；
//      [11,8] 字体宽度放大倍数；
//      [15,12] 字体高度放大倍数；
//
// str
//      要打印的字符串
//
// return
//      返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
//
// remarks
//      该函数会将数据转为 UTF8 编码发送。
```

CP_Label_DrawTextInGBK

在标签页面上指定位置绘制文本。只能单行打印。

定义

```
AUTOREPLYPRINT_API int CP_Label_DrawTextInGBK(void *handle, int x, int y, int font, int style, const wchar_t *str);

//      在标签页面上指定位置绘制文本。只能单行打印。
//
// handle
//      端口句柄，由 OpenXXX 返回
//
// x
//      定义文本起始位置 x 坐标，取值范围：[0, Page_Width-1]
//
// y
//      定义文本起始位置 y 坐标，取值范围：[0, Page_Height-1]
//
// font
//      选择字体，可以使用 24。
//      带矢量字机型支持 16,[20,99]。
//
// style
//      字符风格。
//      数据位                  定义
//      0 加粗标志位：          置 1 字体加粗，清零则字体不加粗。
//      1 下划线标志位：        置 1 文本带下划线，清零则无下划线。
//      2 反白标志位：          置 1 文本反白(黑底白字)，清零不反白。
//      3 删除线标志位：        置 1 文本带删除线，清零则无删除线。
//      [5,4] 旋转标志位：    00 旋转 0° ；
//                              01 旋转 90°；
//                              10 旋转 180°；
//                              11 旋转 270°；
//      [11,8] 字体宽度放大倍数；
//      [15,12] 字体高度放大倍数；
//
// str
//      要打印的字符串
//
// return
//      返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
//
// remarks
//      该函数会将数据转为 GBK 编码发送。
```

CP_Label_DrawTextInBIG5

在标签页面上指定位置绘制文本。只能单行打印。

定义

```
AUTOREPLYPRINT_API int CP_Label_DrawTextInBIG5(void *handle, int x, int y, int font, int style, const wchar_t *str);

//      在标签页面上指定位置绘制文本。只能单行打印。
//
// handle
//      端口句柄，由 OpenXXX 返回
//
// x
//      定义文本起始位置 x 坐标，取值范围：[0, Page_Width-1]
//
// y
//      定义文本起始位置 y 坐标，取值范围：[0, Page_Height-1]
//
// font
//      选择字体，可以使用 24。
//      带矢量字机型支持 16,[20,99]。
//
// style
//      字符风格。
//      数据位                  定义
//      0 加粗标志位：          置 1 字体加粗，清零则字体不加粗。
//      1 下划线标志位：        置 1 文本带下划线，清零则无下划线。
//      2 反白标志位：          置 1 文本反白(黑底白字)，清零不反白。
//      3 删除线标志位：        置 1 文本带删除线，清零则无删除线。
//      [5,4] 旋转标志位：    00 旋转 0° ；
//                              01 旋转 90°；
//                              10 旋转 180°；
//                              11 旋转 270°；
//      [11,8] 字体宽度放大倍数；
//      [15,12] 字体高度放大倍数；
//
// str
//      要打印的字符串
//
// return
//      返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
//
// remarks
//      该函数会将数据转为 BIG5 编码发送。
```

CP_Label_DrawTextInShiftJIS

在标签页面上指定位置绘制文本。只能单行打印。

定义

```
AUTOREPLYPRINT_API int CP_Label_DrawTextInShiftJIS(void *handle, int x, int y, int font, int style, const wchar_t *str);

//      在标签页面上指定位置绘制文本。只能单行打印。
//
// handle
//      端口句柄，由 OpenXXX 返回
//
// x
//      定义文本起始位置 x 坐标，取值范围：[0, Page_Width-1]
//
// y
//      定义文本起始位置 y 坐标，取值范围：[0, Page_Height-1]
//
// font
//      选择字体，可以使用 24。
//      带矢量字机型支持 16,[20,99]。
//
// style
//      字符风格。
//      数据位                  定义
//      0 加粗标志位：          置 1 字体加粗，清零则字体不加粗。
//      1 下划线标志位：        置 1 文本带下划线，清零则无下划线。
//      2 反白标志位：          置 1 文本反白(黑底白字)，清零不反白。
//      3 删除线标志位：        置 1 文本带删除线，清零则无删除线。
//      [5,4] 旋转标志位：    00 旋转 0° ；
//                              01 旋转 90°；
//                              10 旋转 180°；
//                              11 旋转 270°；
//      [11,8] 字体宽度放大倍数；
//      [15,12] 字体高度放大倍数；
//
// str
//      要打印的字符串
//
// return
//      返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
//
// remarks
//      该函数会将数据转为 ShiftJIS 编码发送。
```

CP_Label_DrawTextInEUCKR

在标签页面上指定位置绘制文本。只能单行打印。

定义

```
AUTOREPLYPRINT_API int CP_Label_DrawTextInEUCKR(void *handle, int x, int y, int font, int style, const wchar_t *str);

//      在标签页面上指定位置绘制文本。只能单行打印。
//
// handle
//      端口句柄，由 OpenXXX 返回
//
// x
//      定义文本起始位置 x 坐标，取值范围：[0, Page_Width-1]
//
// y
//      定义文本起始位置 y 坐标，取值范围：[0, Page_Height-1]
//
// font
//      选择字体，可以使用 24。
//      带矢量字机型支持 16,[20,99]。
//
// style
//      字符风格。
//      数据位                  定义
//      0 加粗标志位：          置 1 字体加粗，清零则字体不加粗。
//      1 下划线标志位：        置 1 文本带下划线，清零则无下划线。
//      2 反白标志位：          置 1 文本反白(黑底白字)，清零不反白。
//      3 删除线标志位：        置 1 文本带删除线，清零则无删除线。
//      [5,4] 旋转标志位：    00 旋转 0° ；
//                              01 旋转 90°；
//                              10 旋转 180°；
//                              11 旋转 270°；
//      [11,8] 字体宽度放大倍数；
//      [15,12] 字体高度放大倍数；
//
// str
//      要打印的字符串
//
// return
//      返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
//
// remarks
//      该函数会将数据转为 EUCKR 编码发送。
```

CP_Label_DrawTextInBytes

在标签页面上指定位置绘制文本。只能单行打印。

定义

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

```
//      在标签页面上指定位置绘制文本。只能单行打印。
//
// handle
//      端口句柄，由 OpenXXX 返回
//
// x
//      定义文本起始位置 x 坐标，取值范围：[0, Page_Width-1]
//
// y
//      定义文本起始位置 y 坐标，取值范围：[0, Page_Height-1]
//
// font
//      选择字体，可以使用 24。
//      带矢量字机型支持 16,[20,99]。
//
// style
//      字符风格。
//      数据位                  定义
//      0 加粗标志位：          置 1 字体加粗，清零则字体不加粗。
//      1 下划线标志位：        置 1 文本带下划线，清零则无下划线。
//      2 反白标志位：          置 1 文本反白(黑底白字)，清零不反白。
//      3 删除线标志位：        置 1 文本带删除线，清零则无删除线。
//      [5,4] 旋转标志位：    00 旋转 0° ；
//                              01 旋转 90°；
//                              10 旋转 180°；
//                              11 旋转 270°；
//      [11,8] 字体宽度放大倍数；
//      [15,12] 字体高度放大倍数；
//
// str
//      要打印的字符串
//
// len
//      要打印的长度
//
// return
//      返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Label_DrawBarcode

在标签页指定位置绘制一维条码。

定义

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);

```
//        在标签页指定位置绘制一维条码。
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// x
//        条码左上角 x 坐标值，取值范围：[0, Page_Width-1]。
//
// y
//        条码左上角 y 坐标值，取值范围：[0, Page_Height-1]。
//
// nBarcodeType
//        标识条码类型
//        各值定义看宏定义
//
// nBarcodeTextPrintPosition
//        条码可读字符位置，取值范围：[0, 3].
//        各值定义如下：
//        值 定义
//        0 不显示可读字符
//        1 在条码下方显示可读字符
//        2 在条码上方显示可读字符
//        3 在条码上方和条码下方显示可读字符
//
// height
//        定义条码高度。
//
// unitwidth
//        定义码块单元宽度。取值范围：[1, 4]。
//
// rotation
//        表示旋转角度。取值范围：[0, 3]。各值定义如下：
//        值 定义
//        0  不旋转绘制。
//        1  旋转 90°绘制。
//        2  旋转 180°绘制。
```

```
//          3   旋转 270°绘制。
//
// str
//          要打印的条码
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Label_DrawQRCode

在标签页指定位置绘制 QR 码。

定义

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);

```
//        在标签页指定位置绘制 QR 码。
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// x
//        左上角 x 坐标值，取值范围：[0，Page_Width-1]。
//
// y
//        左上角 y 坐标值，取值范围：[0, Page_Height-1]。
//
// nVersion
//        指定字符版本。取值范围：[0,16]。
//        当 version 为 0 时，打印机根据字符串长度自动计算版本号。
//
// nECCLevel
//        指定纠错等级。取值范围：[1, 4]。
//        各值定义如下：
//        ECC 纠错等级
//        1    L：7%，低纠错，数据多。
//        2    M：15%，中纠错
//        3    Q：优化纠错
//        4    H：30%，最高纠错，数据少。
//
// unitwidth
//        定义码块单元宽度。取值范围：[1, 4]。
//
// rotation
//        表示旋转角度。取值范围：[0, 3]。各值定义如下：
//        值 定义
//        0   不旋转绘制。
//        1   旋转 90°绘制。
//        2   旋转 180°绘制。
//        3   旋转 270°绘制。
//
// str
//        要打印的 QR 码
```

//
// return
//　　返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。

CP_Label_DrawQRCodeInUTF8

在标签页指定位置绘制 QR 码。

定义

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);

```
//      在标签页指定位置绘制 QR 码。
//
// handle
//      端口句柄，由 OpenXXX 返回
//
// x
//      左上角 x 坐标值，取值范围：[0，Page_Width-1]。
//
// y
//      左上角 y 坐标值，取值范围：[0, Page_Height-1]。
//
// nVersion
//      指定字符版本。取值范围：[0,16]。
//      当 version 为 0 时，打印机根据字符串长度自动计算版本号。
//
// nECCLevel
//      指定纠错等级。取值范围：[1, 4]。
//      各值定义如下：
//      ECC 纠错等级
//      1   L：7%，低纠错，数据多。
//      2   M：15%，中纠错
//      3   Q：优化纠错
//      4   H：30%，最高纠错，数据少。
//
// unitwidth
//      定义码块单元宽度。取值范围：[1, 4]。
//
// rotation
//      表示旋转角度。取值范围：[0, 3]。各值定义如下：
//      值 定义
//      0   不旋转绘制。
//      1   旋转 90°绘制。
//      2   旋转 180°绘制。
//      3   旋转 270°绘制。
//
// str
//      要打印的 QR 码
```

//
// return
//　　返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。

CP_Label_DrawQRCodeInBytes

在标签页指定位置绘制 QR 码。

定义

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);

```
//        在标签页指定位置绘制 QR 码。
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// x
//        左上角 x 坐标值，取值范围：[0，Page_Width-1]。
//
// y
//        左上角 y 坐标值，取值范围：[0, Page_Height-1]。
//
// nVersion
//        指定字符版本。取值范围：[0,16]。
//        当 version 为 0 时，打印机根据字符串长度自动计算版本号。
//
// nECCLevel
//        指定纠错等级。取值范围：[1, 4]。
//        各值定义如下：
//        ECC 纠错等级
//        1    L：7%，低纠错，数据多。
//        2    M：15%，中纠错
//        3    Q：优化纠错
//        4    H：30%，最高纠错，数据少。
//
// unitwidth
//        定义码块单元宽度。取值范围：[1, 4]。
//
// rotation
//        表示旋转角度。取值范围：[0, 3]。各值定义如下：
//        值 定义
//        0   不旋转绘制。
//        1   旋转 90°绘制。
//        2   旋转 180°绘制。
//        3   旋转 270°绘制。
//
// str
//        要打印的 QR 码
```

//
// len
// 要打印的长度
//
// return
// 返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。

CP_Label_DrawPDF417Code

在标签页指定位置绘制 PDF417 条码

定义

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);

```
//        在标签页指定位置绘制 PDF417 条码
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// x
//        左上角 x 坐标值，取值范围：[0, Page_Width-1]。
//
// y
//        左上角 y 坐标值，取值范围：[0, Page_Height-1]。
//
// column
//        ColNum 为列数，表述每行容纳多少码字。一个码字为 17*UnitWidth 个点。行数由打印机自动产生，行数范围限定为 3~90。ColNum 的取值范围：[1,30]。
//
// nECCLevel
//        指定纠错等级。取值范围：[0, 8]。
//        纠错等级取值 纠错码数 可存资料量（字节）
//        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
//        定义码块单元宽度。取值范围：[1, 3]。
//
// rotation
//        表示旋转角度。取值范围：[0, 3]。各值定义如下：
//        值 定义
//        0  不旋转绘制。
//        1  旋转 90°绘制。
//        2  旋转 180°绘制。
```

//　　3　旋转 270°绘制。
//
// str
//　　要打印的 PDF417 码
//
// return
//　　返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。

CP_Label_DrawImageFromFile

在标签页指定位置绘制位图

定义

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);

```
//      在标签页指定位置绘制位图
//
// handle
//      端口句柄，由 OpenXXX 返回
//
// x
//      左上角 x 坐标值，取值范围：[0, Page_Width-1]。
//
// y
//      左上角 y 坐标值，取值范围：[0, Page_Height-1]。
//
// dstw
//      要打印的宽度
//
// dsth
//      要打印的高度
//
// pszFile
//      图片的路径
//
// binaryzation_method
//      图片二值化算法。0 表示抖动算法，1 表示阀值算法，2 表示误差扩散法。具体效果请测试查看。
//
// compression_method
//      最终打印数据的压缩方式，各值定义如下
//      值      定义
//      0       不压缩
//      1       一级压缩
//      2       二级压缩
//
// return
//      返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Label_DrawImageFromData

在标签页指定位置绘制位图

定义

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);

```
//      在标签页指定位置绘制位图
//
// handle
//      端口句柄，由 OpenXXX 返回
//
// x
//      左上角 x 坐标值，取值范围：[0, Page_Width-1]。
//
// y
//      左上角 y 坐标值，取值范围：[0, Page_Height-1]。
//
// dstw
//      要打印的宽度
//
// dsth
//      要打印的高度
//
// data
//      图片数据。
//
// data_size
//      图片数据长度
//
// binaryzation_method
//      图片二值化算法。0 表示抖动算法，1 表示阀值算法，2 表示误差扩散法。具体效果请测试查看。
//
// compression_method
//      最终打印数据的压缩方式，各值定义如下
//      值      定义
//      0       不压缩
//      1       一级压缩
//      2       二级压缩
//
// return
//      返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Label_DrawImageFromPixels

在标签页指定位置绘制位图

定义

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);

```
//      在标签页指定位置绘制位图
//
// handle
//      端口句柄，由 OpenXXX 返回
//
// x
//      左上角 x 坐标值，取值范围：[0, Page_Width-1]。
//
// y
//      左上角 y 坐标值，取值范围：[0, Page_Height-1]。
//
// img_data
//      图片的像素数据。
//
// img_datalen
//      图片的像素数据字节数。
//
// img_width
//      图片的像素宽度。
//
// img_height
//      图片的像素高度。
//
// img_stride
//      图片水平跨度。表示每行字节数。
//
// img_format
//      图片像素数据格式，各值定义如下
//      值      定义
//      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
//          图片二值化算法。0 表示抖动算法，1 表示阀值算法，2 表示误差扩散法。具体效果请测试查看。
//
// compression_method
//          最终打印数据的压缩方式，各值定义如下
//          值      定义
//          0       不压缩
//          1       一级压缩
//          2       二级压缩
//
// return
//          返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Label_DrawLine

在标签页指定位置绘制线段

定义

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

```
//        在标签页指定位置绘制线段
//
// handle
//        端口句柄，由 OpenXXX 返回
//
// startx
//        直线段起始点 x 坐标值，取值范围：[0, Page_Width-1]。
//
// starty
//        直线段起始点 y 坐标值，取值范围：[0，Page_Height-1]。
//
// endx
//        直线段终止点 x 坐标值，取值范围：[0, Page_Width-1]。
//
// endy
//        直线段终止点 y 坐标值，取值范围：[0,Page_Height-1]。
//
// linewidth
//        直线段线宽，取值范围：[1，Page_Height-1]。
//
// linecolor
//        直线段颜色线条颜色，0 是白色，1 是黑色
//
// return
//        返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Label_DrawRect

在标签页指定位置绘制矩形

定义

```
AUTOREPLYPRINT_API int CP_Label_DrawRect(void *handle, int x, int y, int width, int height, CP_Label_Color color);

//		在标签页指定位置绘制矩形
//
// handle
//		端口句柄，由 OpenXXX 返回
//
// x
//		横向坐标
//
// y
//		纵向坐标
//
// width
//		矩形宽度
//
// height
//		矩形高度
//
// color
//		矩形颜色，0 是白色，1 是黑色
//
// return
//		返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

CP_Label_DrawBox

在标签页指定位置绘制矩形框

定义

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

```
//		在标签页指定位置绘制矩形框
//
// handle
//		端口句柄，由 OpenXXX 返回
//
// x
//		横向坐标
//
// y
//		纵向坐标
//
// width
//		矩形框宽度
//
// height
//		矩形框高度
//
// borderwidth
//		矩形框边框宽度
//
// bordercolor
//		矩形框边框颜色，0 是白色，1 是黑色
//
// return
//		返回值仅指示指令是否写入成功。返回 true 表示写入成功，返回 false 表示写入失败。
```

其他函数

CP_Library_Version

获取开发包版本字符串

定义

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

//          获取开发包版本字符串
//
//   return
//          返回开发包版本
```

CP_Proto_QueryBatteryLevel

查询电池电量

定义

```
AUTOREPLYPRINT_API int CP_Proto_QueryBatteryLevel(void *handle, unsigned int timeout);

//      查询电池电量
//      仅部分带电池的机型支持该指令
//
// handle
//      端口句柄，由 OpenXXX 返回
//
// timeout
//      超时毫秒时间。
//      查询等待时间不超过此时间。
//
// return
//      返回电池电量，范围在 0-100 之间。返回-1 表示查询失败。
```