

Windows SDK

说明手册 (V1.25)



GTSP.LDLL 函式庫支援開發平台/語言

開發平台/語言	範例代碼說明
C Sharp	第 25~35 頁
Java	第 36~49 頁
Javascript	第 50~52 頁
ASP.NET	第 52~54 頁
VB.NET	第 54~55 頁
Access-VBA	第 55~56 頁
Excel-VBA	第 57 頁
Visual C++	第 58~68 頁
FoxPro	第 69~70 頁
Python	第 71~75 頁
VB6	第 76~81 頁
PHP	第 82~88 頁
Qt C++	第 89~141 頁
JSP	第 142~148 頁

GTSP.L.DLL 函式庫使用說明(V1.25)

1. openport(name)、openport_USB()、openports_USB(PrinterModel)、openport_Ethernet(IP, port)

- 函式說明：指定並開啟電腦端的輸出埠
- 參數說明：
 - ➔ name：字串型別
 - (1) 單機列印時，請指定印表機驅動程式名稱
 - (2) 若連接網路印表機，請指定印表機路徑及共用印表機名稱
 - 如：\\192.168.1.104\Printer Model
 - ➔ PrinterModel：字串型別，印表機機型名稱
 - ➔ IP：字串型別，印表機 IP 位址
 - ➔ port：int 型別，印表機連接埠號碼

2. closeport()、closeport_USB()、closeport_Ethernet()

- 函式說明：關閉輸出埠
- 參數說明：無

3. detectUSB_USB()

- 函式說明：偵測印表機 USB 接口插拔狀況
- 參數說明：無
- 回傳說明：字串陣列型別，以 USB 連接所有開啟的印表機

4. detectUSBStr_USB()

- 函式說明：偵測印表機 USB 接口插拔狀況
- 參數說明：無
- 回傳說明：字串型別，以 USB 連接所有開啟的印表機

5. sendcommand(command)、sendcommand_USB(command)、sendcommand_Ethernet(command)

- 函式說明：將內建命令傳送至印表機(結尾會自動加入\r\n 命令結束用語)
- 參數說明：
 - ➔ command：字串型別，設定指令內容，詳細指令請參考 TSPL 使用手冊

6. `sendcommand_Array(command)`、`sendcommand_Array_USB(command)`、

`sendcommand_Array_Ethernet(command)`

- 函式說明：將內建命令傳送至印表機(結尾會自動加入\r\n 命令結束用語)
- 參數說明：
 - ➔ `command`：位元組陣列型別，設定指令內容，詳細指令請參考 TSPL 使用手冊

7. `sendcommand_noCIRf(command)`、`sendcommand_noCIRf_USB(command)`、

`sendcommand_noCIRf_Ethernet(command)`

- 函式說明：將內建命令傳送至印表機(結尾不會自動加入\r\n 命令結束用語)
- 參數說明：
 - ➔ `command`：字串型別，設定指令內容，詳細指令請參考 TSPL 使用手冊

8. `clearbuffer()`、`clearbuffer_USB()`、`clearbuffer_Ethernet()`

- 函式說明：清除圖像緩衝
- 參數說明：無

9. `setup(width, height, speed, density, sensor, sensorDistance, sensorOffset)`、

`setup_USB(width, height, speed, density, sensor, sensorDistance, sensorOffset)`、

`setup_Ethernet(width, height, speed, density, sensor, sensorDistance, sensorOffset)`

- 函式說明：設定標籤的寬度、高度、列印速度、列印熱度、感應器類別、間隙/黑標垂直間距、間隙/黑標偏移距離
- 參數說明：

參數	型別	說明
width	字串	設定標籤寬度，單位 mm
height	字串	設定標籤高度，單位 mm
speed	字串	設定列印速度，1~15，代表每秒 1~15 吋列印速度(隨機型不同會有不同列印最高上限，最高為每秒 15 吋列印速度)
density	字串	設定列印濃度，0~15，數字越大列印結果越黑
sensor	字串	設定使用感應器之類別；

		0：表示使用間隙感測器(gap sensor) 1：表示使用黑標感測器(black mark sensor)
sensorDistance	字串	設定間隙/黑標垂直間距高度，單位 mm
sensorOffset	字串	設定間隙/黑標垂直間距高度，單位 mm，此參數若使用一般標籤時均設為 0

10. barcode(x, y, type, height, readable, rotation, narrow, wide, content) 、

barcode_USB(x, y, type, height, readable, rotation, narrow, wide, content) 、

barcode_Ethernet(x, y, type, height, readable, rotation, narrow, wide, content)

■ 函式說明：使用印表機內建條碼列印

■ 參數說明：

參數	型別	說明
x	字串	條碼 X 方向起始點，以點(dot)表示
y	字串	條碼 Y 方向起始點，以點(dot)表示
type	字串	設定條碼類型(Code Type) ， 請參考附件
height	字串	設定條碼高度，高度以點來表示
readable	字串	設定是否列印條碼碼文 0:不列印 1:列印條碼碼文置左 2:列印條碼碼文置中 3:列印條碼碼文置右
rotation	字串	設定條碼旋轉角度 0：旋轉0度 90：旋轉90度 180：旋轉180度 270：旋轉270度
narrow	字串	設定條碼窄 bar 比例因子， 請參考附件
wide	字串	設定條碼寬 bar 比例因子， 請參考附件
content	字串	設定欲列印之條碼內容

11. qrcode(x, y, ECCLevel, cellWidth, mode, rotation, content) 、

qrcode_USB(x, y, ECCLevel, cellWidth, mode, rotation, content) 、

qrcode_Ethernet(x, y, ECCLevel, cellWidth, mode, rotation, content)

■ 函式說明：使用印表機列印 QRcode

■ 參數說明：

參數	型別	說明
x	字串	QRCode X 方向起始點，以點(dot)表示
y	字串	QRCode Y 方向起始點，以點(dot)表示
ECCLevel	字串	容錯率 L : 7% M : 15% Q : 25% H : 30%
cellWidth	字串	設定 QRCode 大小，1~10
mode	字串	設定自動或手動編碼 A : 自動 M : 手動
rotation	字串	設定QRCode旋轉角度 0 : 旋轉0度 90 : 旋轉90度 180 : 旋轉180度 270 : 旋轉270度
content	字串	設定資料內容 資料內容限制： 1) 數字資料: (數字 0~9) 2) 字母資料 數字 0-9 大寫字母 A-Z 9 種其它字元: 空格, \$ % * + - . / :) *如果“A”是資料內容的第一個字元，那麼資料內容將會被設置為字母數據。 *如果“N”是資料內容的第一個字元，那麼資料內容將會被設置為數字數據。 * “！”用來轉換資料的格式，“N”、“A”等資料類型可通過“！”來轉換。

12. `printerfont(x, y, size, rotation, x_scale, y_scale, content)` 、

`printerfont_USB(x, y, size, rotation, x_scale, y_scale, content)` 、

`printerfont_Ethernet(x, y, size, rotation, x_scale, y_scale, content)`

■ 函式說明：使用印表機內建文字列印

■ 參數說明：

參數	型別	說明
x	字串	文字 X 方向起始點，以點(dot)表示
y	字串	文字 Y 方向起始點，以點(dot)表示
size	字串	內建字型名稱 1: 8*/12 dots 2: 12*20 dots 3: 16*24 dots 4: 24*32 dots 5: 32*48 dots TST24.BF2: 繁體中文24*24 TST16.BF2: 繁體中文16*16 TSS24.BF2: 簡體中文24*24 TSS16.BF2: 簡體中文16*16
rotation	字串	設定文字旋轉角度 0：旋轉0度 90：旋轉90度 180：旋轉180度 270：旋轉 270 度
x_scale	字串	設定文字 X 方向放大倍率，1~10
y_scale	字串	設定文字Y方向放大倍率，1~10
content	字串	設定欲列印之文字內容

13. `formfeed()` 、 `formfeed_USB()` 、 `formfeed_Ethernet()`

■ 函式說明：跳頁，該函式需在 `setup` 後使用

■ 參數說明：無

14. `nobackfeed()` 、 `nobackfeed_USB()` 、 `nobackfeed_Ethernet()`

■ 函式說明：設定紙張不回吐

■ 參數說明：無

15. printlabel (set, copy) 、 printlabel _USB(set, copy) 、 printlabel _Ethernet(set, copy)

- 函式說明：列印標籤內容
- 參數說明：
 - ➔ set：字串型別，設定列印標籤式數(set)
 - ➔ copy：字串型別，設定列印標籤份數(copy)

16. downloadpcx (filename,memoryname) 、 downloadpcx _USB (filename,memoryname) 、 downloadpcx _Ethernet(filename,memoryname)

- 函式說明：下載單色 PCX 格式圖檔至印表機
- 參數說明：
 - ➔ filename：字串型別，檔案名稱(可包含路徑)
 - ➔ memoryname：下載至印表機記憶體內之檔名(請使用大寫檔名)

17. downloadbmp (filename, memoryname) 、 downloadbmp _USB (filename, memoryname) 、 downloadbmp _Ethernet (filename, memoryname)

- 函式說明：下載單色 BMP 格式圖檔至印表機
- 參數說明：
 - ➔ filename：字串型別，檔案名稱(可包含路徑)
 - ➔ memoryname：下載至印表機記憶體內之檔名(請使用大寫檔名)

18. windowfont (x,y,height,rotation,fontstyle,underline,fontname,content)

windowfont_USB (x,y,height,rotation,fontstyle,underline,fontname,content)

windowfont_Ethernet(x,y,height,rotation,fontstyle,underline,fontname,content)

■ 函式說明：使用 Windows TTF 字型列印文字

■ 參數說明：

參數	型別	說明
x	int	文字 X 方向起始點，以點(dot)表示
y	int	文字 Y 方向起始點，以點(dot)表示
height	int	字體高度，以點(dot)表示
rotation	int	設定旋轉角度，逆時鐘方向旋轉 0：旋轉0度 90：旋轉90度 180：旋轉180度 270：旋轉 270 度
fontstyle	int	設定字體外型 0：標準(Normal) 1：斜體(Italic) 2：粗體(Bold) 3：粗斜體(Bold and Italic)
underline	int	設定底線 0：無底線 1：加底線
fontname	字串	Windows字體名稱，如: Arial, Times new Roman, 細明體, 標楷體
content	字串	設定欲列印之文字內容

19. getDLLVersion (returnWay)、getDLLVersion_USB(returnWay)、getDLLVersion_Ethernet(returnWay)

■ 函式說明：回傳此 SDK 版本號

■ 參數說明：

➔ returnWay：int 型別，輸入 0 除返回 SDK 版本號外，會跳出 SDK 版本訊息

20. printerstatus_USB ()、printerstatus_Ethernet()

- 函式說明：回傳印表機狀態，需用字串變數接收回傳訊息
- 參數說明：無
- 回傳字串說明：

回傳字串	印表機狀態
00	就緒
01	上蓋開啟
02	卡紙
03	卡紙且上蓋開啟
04	標籤用盡
05	標籤用盡且上蓋開啟
08	碳帶用盡
09	碳帶用盡且上蓋開啟
0A	碳帶用盡且卡紙
0B	碳帶用盡、卡紙且上蓋開啟
0C	碳帶用盡且標籤用盡
0D	碳帶用盡、標籤用盡且上蓋開啟
10	暫停
20	列印中
80	其他錯誤

21. writeUHF (dataFormat,startBlockNo,byteSize,Gen2MemoryBank,datastring)

writeUHF_USB (dataFormat,startBlockNo,byteSize,Gen2MemoryBank,datastring)

writeUHF_Ethernet (dataFormat,startBlockNo,byteSize,Gen2MemoryBank,datastring)

- 函式說明：將資料寫入 UHF Gen2 標籤記憶體中
- 參數說明：

參數	型別	說明
dataFormat	string	設定字串資料編碼格式，預設為 H A：ASCII H：Hexadecimal
startBlockNo	int	設定資料區塊起始位置，Gen2 預設為 2
byteSize	int	設定寫入資料byte長度，預設為1
Gen2Memory Bank	string	設定 Gen2 資料區段，預設為 E R：保留 E：EPC

		T : TID(Tag ID) U : User
datastring	string	欲寫入之字串資料

22. EPCPWD_Action(action, password)、EPCPWD_Action_USB(action, password)、

EPCPWD_Action_Ethernet(action, password)

- 函式說明：將 UHF GNE2 的 EPC 資料區塊上鎖或解鎖
- 參數說明：

參數	型別	說明
action	string	設定執行動作 U：解鎖資料區塊 L：上鎖資料區塊 O：永久解鎖資料區塊 P：永久上鎖資料區塊
password	string	密碼，應為 8 hex 字元(0~9,A,B,C,D,E,F)

23. TIDPWD_Action(action, password)、TIDPWD_Action_USB(action, password)、

TIDPWD_Action_Ethernet(action, password)

- 函式說明：將 UHF GNE2 的 TID 資料區塊上鎖或解鎖
- 參數說明：

參數	型別	說明
action	string	設定執行動作 U：解鎖資料區塊 L：上鎖資料區塊 O：永久解鎖資料區塊 P：永久上鎖資料區塊
password	string	密碼，應為 8 hex 字元(0~9,A,B,C,D,E,F)

24. USERPWD_Action(action, password)、USERPWD_Action_USB(action, password)、

USERPWD_Action_Ethernet(action, password)

- 函式說明：將 UHF GNE2 的 USER 資料區塊上鎖或解鎖
- 參數說明：

參數	型別	說明
action	string	設定執行動作 U：解鎖資料區塊 L：上鎖資料區塊 O：永久解鎖資料區塊 P：永久上鎖資料區塊
password	string	密碼，應為 8 hex 字元(0~9,A,B,C,D,E,F)

25. AccessPWD_Action (action, password)、AccessPWD_Action_USB (action, password)、

AccessPWD_Action_Ethernet (action, password)

- 函式說明：將 UHF GNE2 的存取密碼進行設定、上鎖或解鎖
- 參數說明：

參數	型別	說明
action	string	設定執行動作 U：解鎖存取密碼 L：上鎖存取密碼 O：永久解鎖存取密碼 P：永久上鎖存取密碼 S：設定存取密碼
password	string	密碼，應為 8 hex 字元(0~9,A,B,C,D,E,F)

26. KillPWD_Action (action, password)、KillPWD_Action_USB (action, password)、

KillPWD_Action_Ethernet (action, password)

- 函式說明：將 UHF GNE2 的刪除密碼進行設定、上鎖或解鎖
- 參數說明：

參數	型別	說明
action	string	設定執行動作 U：解鎖刪除密碼 L：上鎖刪除密碼 O：永久刪除存取密碼 P：永久刪除存取密碼 S：設定刪除密碼
password	string	密碼，應為 8 hex 字元(0~9,A,B,C,D,E,F)

27. Set_RFIDPorcedure (tagType, rw_position, void_printout, tryEncodie_times,error_handle, speed, retry_times)

Set_RFIDPorcedure_USB (tagType, rw_position, void_printout, tryEncodie_times,error_handle, speed, retry_times)

Set_RFIDPorcedure_Ethernet (tagType, rw_position, void_printout, tryEncodie_times,error_handle, speed, retry_times)

■ 函式說明：RFID 設定

■ 參數說明：

參數	型別	說明
tagType	int	設定標籤類型，1~10，預設值為 8 1：EPC Class 1 Generation 2-Q，8：EPC Class 1 Generation 2-R，10：UHF-J
rw_position	int	設標籤讀寫位置(標籤頂部起算)，範圍為 0~9999(dot)，預設為 0
void_printout	int	設定無效列印長度(dot)，範圍為0~標籤長度，預設為標籤長度
tryEncodie_times	int	設定最大無效標籤數，範圍為 0~10，預設為 3
error_handle	string	設定無效時採取的動作，預設為 N N：No action(繼續) P：Pause mode(暫停) E：Error mode(停止)
speed	int	設定無效打印速度，範圍 2~10(IPS)，預設值 2(IPS)
retry_times	int	設定標籤重試次數，範圍 0~10，預設值 6

28. Set_RFIDPorcedure_mm (tagType, rw_position, void_printout, tryEncodie_times,error_handle, speed, retry_times,dpi)

Set_RFIDPorcedure_mm_USB (tagType, rw_position, void_printout, tryEncodie_times,error_handle, speed, retry_times,dpi)

Set_RFIDPorcedure_mm_Ethernet (tagType, rw_position, void_printout, tryEncodie_times,error_handle, speed, retry_times,dpi)

■ 函式說明：RFID 設定

■ 參數說明：

參數	型別	說明
tagType	int	設定標籤類型，1~10，預設值為 8 1：EPC Class 1 Generation 2-Q，8：EPC Class 1 Generation 2-R，10：UHF-J

rw_position	double	設標籤讀寫位置(標籤頂部起算)， 範圍為 203dpi:0 ~ 1251 (mm)、 300dpi:0 ~ 846 (mm)、 600dpi:0 ~ 423 (mm)，預設為 0
void_printout	double	設定無效列印長度(mm)，範圍為0~標籤長度，預設為標籤長度
tryEncodie_times	int	設定最大無效標籤數，範圍為 0~10，預設為 3
error_handle	字串	設定無效時採取的動作，預設為 N N：No action(繼續) P：Pause mode(暫停) E：Error mode(停止)
speed	int	設定無效打印速度，範圍 2~10(IPS)，預設值 2(IPS)
retry_times	int	設定標籤重試次數，範圍 0~10，預設值 6
dpi	字串	設定打印機的 DPI 203: 203 dpi 300: 300 dpi 600: 600 dpi

29. writeHF (dataFormat,startBlockNo,byteSize,datastring)

writeHF_USB (dataFormat,startBlockNo,byteSize,datastring)

writeHF_Ethernet (dataFormat,startBlockNo,byteSize,datastring)

■ 函式說明：將資料寫入 HF 標籤記憶體中

■ 參數說明：

參數	型別	說明
dataFormat	string	設定字串資料編碼格式，預設為 H A：ASCII H：Hexadecimal
startBlockNo	int	設定資料區塊起始位置，預設為 2
byteSize	int	設定寫入資料byte長度，預設為1
datastring	string	欲寫入之字串資料

30. **printerfontblock** (**x**, **y**, **width**, **height**, **fontname**, **rotation**, **x_scale**, **y_scale**, **space**, **align**, **content**) 、
printerfontblock_USB(**x**, **y**, **width**, **height**, **fontname**, **rotation**, **x_scale**, **y_scale**, **space**, **align**, **content**) 、
printerfontblock_Ethernet(**x**, **y**, **width**, **height**, **fontname**, **rotation**, **x_scale**, **y_scale**, **space**, **align**,
content)

■ 函式說明：列印段落文字內容

■ 參數說明：

參數	型別	說明
x	字串	文字 X 方向起始點，以點(dot)表示
y	字串	文字 Y 方向起始點，以點(dot)表示
width	字串	設定段落區塊寬度，以點(dot)表示
height	字串	設定段落區塊高度，以點(dot)表示
fontname	字串	內建字型名稱 1: 8*/12 dots 2: 12*20 dots 3: 16*24 dots 4: 24*32 dots 5: 32*48 dots TST24.BF2: 繁體中文24*24 TST16.BF2: 繁體中文16*16 TSS24.BF2: 簡體中文24*24 TSS16.BF2: 簡體中文16*16
rotation	字串	設定文字旋轉角度 0：旋轉0度 90：旋轉90度 180：旋轉180度 270：旋轉 270 度
x_scale	字串	設定文字 X 方向放大倍率，1~10
y_scale	字串	設定文字Y方向放大倍率，1~10
space	字串	行距，以點(dot)表示
align	字串	對齊位置 0：預設(置左) 1：置左 2：置中 3：置右
content	字串	設定欲列印之文字內容

31. readUHF_USB (dataFormat,startBlockNo,byteSize,Gen2MemoryBank)

readUHF_Ethernet (dataFormat,startBlockNo,byteSize,Gen2MemoryBank)

■ 函式說明：讀取 UHF Gen2 標籤記憶體資料，需用字串變數接收回傳訊息

■ 參數說明：

參數	型別	說明
dataFormat	string	設定字串資料編碼格式，預設為 H A：ASCII H：Hexadecimal
startBlockNo	int	設定資料區塊起始位置，預設為 0
byteSize	int	設定讀取資料byte長度，預設為1
Gen2MemoryBank	string	設定 Gen2 資料區段，預設為 E R：保留 E：EPC T：TID(Tag ID) U：User

■ 回傳字串說明(標籤資料)：

dataFormat	回傳字串(範例)
A	標籤資料以 ASCII 顯示 (ex: 24051324000103456400)
H	標籤資料以 Hexadecimal 顯示 (ex: 3234303531333234303030313033343536343030)

■ 回傳字串說明(錯誤代碼)：

回傳字串	錯誤代碼說明
64000000000000000000000000000000	其他錯誤
65000000000000000000000000000000	超過記憶體空間
66000000000000000000000000000000	記憶體被鎖住
67000000000000000000000000000000	讀取功率不足
68000000000000000000000000000000	非特定的錯誤
69000000000000000000000000000000	CRC錯誤
6A000000000000000000000000000000	寫入中若發生錯誤時，回覆已寫入多少 words 數
6B000000000000000000000000000000	寫入中若 Tag 標籤回覆錯誤時，錯誤碼加上已寫入多少 words 數
6C000000000000000000000000000000	沒有標籤存在
6D000000000000000000000000000000	指令格式錯誤
6E000000000000000000000000000000	設定電源強度失敗
6F000000000000000000000000000000	設定法規失敗

32. setPWD_Action(passwordArea, action, NewPassword, WritePassword)

setPWD_Action_USB(passwordArea, action, NewPassword, WritePassword)

setPWD_Action_Ethernet(passwordArea, action, NewPassword, WritePassword)

■ 函式說明：設定 UHF GJB 各密碼區域新密碼

■ 參數說明：

參數	型別	說明
passwordArea	string	設定密碼區域，預設為 W K：Kill W：Write R：Read S：Status
action	string	設定動作 S：Set Password
NewPassword	string	設定密碼區域的新密碼，應為8 hex字元(0~9,A,B,C,D,E,F)
WritePassword	string	寫入密碼，應為 8 hex 字元(0~9,A,B,C,D,E,F)

33. writeGJB_UHF(dataFormat, startBlockNo, byteSize, Gen2MemoryBank, datastring, WritePassword)

writeGJB_UHF_USB(dataFormat, startBlockNo, byteSize, Gen2MemoryBank, datastring, WritePassword)

writeGJB_UHF_Ethernet(dataFormat, startBlockNo, byteSize, Gen2MemoryBank, datastring, WritePassword)

■ 函式說明：將資料寫入 UHF GJB 標籤記憶體中

■ 參數說明：

參數	型別	說明
dataFormat	string	設定字串資料編碼格式，預設為 H A：ASCII H：Hexadecimal
startBlockNo	int	設定資料區塊起始位置，GJB 預設為 1
byteSize	int	設定寫入資料byte長度，預設為1
Gen2MemoryBank	string	設定 GJB 資料區段，預設為 E R：安全區 E：EPC T：TID(Tag ID) U：User
datastring	string	欲寫入之字串資料
WritePassword	string	寫入密碼，應為 8 hex 字元(0~9,A,B,C,D,E,F)

34. readGJB_UHF_USB(dataFormat, startBlockNo, byteSize, Gen2MemoryBank, ReadPassword) readGJB_UHF_Ethernet(dataFormat, startBlockNo, byteSize, Gen2MemoryBank, ReadPassword)

■ 函式說明：讀取 UHF GJB 標籤記憶體資料，需用字串變數接收回傳訊息

■ 參數說明：

參數	型別	說明
dataFormat	string	設定字串資料編碼格式，預設為 H A：ASCII H：Hexadecimal
startBlockNo	int	設定資料區塊起始位置，預設為 0
byteSize	int	設定讀取資料byte長度，預設為1
Gen2MemoryBank	string	設定 GJB 資料區段，預設為 E R：安全區 E：EPC T：TID(Tag ID) U：User
ReadPassword	string	讀取密碼，應為 8 hex 字元(0~9,A,B,C,D,E,F)

■ 回傳字串說明：

dataFormat	回傳字串(範例)
A	標籤資料以 ASCII 顯示 (ex: 24051324000103456400)
H	標籤資料以 Hexadecimal 顯示 (ex: 3234303531333234303030313033343536343030)

35. statusGJB_UHF(Gen2MemoryBank, action, StatusPassword) statusGJB_UHF_USB(Gen2MemoryBank, action, StatusPassword) statusGJB_UHF_Ethernet(Gen2MemoryBank, action, StatusPassword)

■ 函式說明：設定 UHF GJB 各資料區塊讀寫狀態

■ 參數說明：

參數	型別	說明
Gen2MemoryBank	string	設定 GJB 資料區段，預設為 E F：安全區 E：EPC T：TID(Tag ID) U：User

action	string	設定狀態，預設為 A A=Lock0(可讀可寫) B=Lock1(可讀不可寫) C=Lock2(不可讀可寫) D=Lock3(不可讀不可寫)
StatusPassword	string	狀態密碼，應為8 hex字元(0~9,A,B,C,D,E,F)

36. killGJB_UHF(KillPassword)

killGJB_UHF_USB(KillPassword)

killGJB_UHF_Ethernet(KillPassword)

■ 函式說明：刪除 UHF GJB 標籤

■ 參數說明：

參數	型別	說明
KillPassword	string	刪除密碼，應為 8 hex 字元(0~9,A,B,C,D,E,F)

37. query_UHF_USB (dataFormat, PCReturnStatus, CRCReturnStatus)

query_UHF_Ethernet (dataFormat, PCReturnStatus, CRCReturnStatus)

■ 函式說明：以 Q 指令讀取 UHF Gen2 標籤記憶體 EPC 資料區段資料，需用字串變數接收回傳訊息

■ 參數說明：

參數	型別	說明
dataFormat	string	設定字串資料編碼格式，預設為 H A：ASCII H：Hexadecimal
PCReturnStatus	int	PC 返回狀態，預設為 0 0：不回傳 PC 值 1：回傳 PC 值
CRCReturnStatus	int	CRC-16 返回狀態，預設為 0 0：不回傳 CRC-16 1：回傳CRC-16

■ 回傳字串說明(標籤資料)：

以 query_UHF_USB("A", 1, 1)和 query_UHF_USB("H", 1, 1)為例：(PC 值和 CRC-16 皆回傳)

dataFormat	回傳字串(範例)
A	標籤資料以 ASCII 顯示 (ex: 24051324000103456400)

H	標籤資料以 Hexadecimal 顯示 (ex: 3234303531333234303030313033343536343030)
---	--

以 query_UHF_USB("A", 0, 0)和 query_UHF_USB("H", 0, 0)為例：(PC 值和 CRC-16 皆不回傳)

dataFormat	回傳字串(範例)
A	標籤資料以 ASCII 顯示 (ex: 0513240001034564)
H	標籤資料以 Hexadecimal 顯示 (ex: 30353133323430303031303334353634)

■ 回傳字串說明(錯誤代碼)：

回傳字串	錯誤代碼說明
64000000000000000000000000000000	其他錯誤
65000000000000000000000000000000	超過記憶體空間
66000000000000000000000000000000	記憶體被鎖住
67000000000000000000000000000000	讀取功率不足
68000000000000000000000000000000	非特定的錯誤
69000000000000000000000000000000	CRC錯誤
6A000000000000000000000000000000	寫入中若發生錯誤時，回覆已寫入多少 words 數
6B000000000000000000000000000000	寫入中若 Tag 標籤回覆錯誤時，錯誤碼加上已寫入多少 words 數
6C000000000000000000000000000000	沒有標籤存在
6D000000000000000000000000000000	指令格式錯誤
6E000000000000000000000000000000	設定電源強度失敗
6F000000000000000000000000000000	設定法規失敗

38. RFIDAutoCalibration()、RFIDAutoCalibration_USB()、RFIDAutoCalibration_Ethernet()

- 函式說明：進行 RFID 標籤自動校準至最佳位置
- 參數說明：無

39. setDirectionAndMirror(direction,mirror)、setDirectionAndMirror_USB(direction,mirror)、setDirectionAndMirror_Ethernet(direction,mirror)

- 函式說明：設定標籤列印時的出紙方向與是否使用鏡像列印
- 參數說明：

參數	型別	說明
direction	int	設定出紙方向，預設為 0 0：頂端出紙 1：底端出紙

mirror	int	設定是否鏡像列印 0：否 1：是
---------------	-----	------------------------

40. setShift (shiftY) 、 setShift_USB (shiftY) 、 setShift_Ethernet (shiftY)

- 函式說明：設定圖像垂直位移距離，數值為正時，圖像會往列印方向移動，數值為負時，圖像會背離列印方向
- 參數說明：
➔ shiftY：int 型別，垂直位移距離，單位為 dot

41. printReverse(x_start, y_start, x_width, y_height) 、 printReverse_USB(x_start, y_start, x_width, y_height) 、 printReverse_Ethernet (x_start, y_start, x_width, y_height)

- 函式說明：將指定的區域於列印時反白
- 參數說明：

參數	型別	說明
x_start	int	指定 X 起始座標位置，以點(dot)表示
y_start	int	指定 Y 起始座標位置，以點(dot)表示
x_width	int	指定 X 座標寬度，以點(dot)表示
y_height	int	指定 Y 座標高度，以點(dot)表示

42. setOffset(offset) 、 setOffset_USB(offset) 、 setOffset_Ethernet(offset)

- 函式說明：設定每次出紙後額外偏移的距離(通常與剝紙模式和裁切模式組合使用)
- 參數說明：
➔ offset：double 型別，額外的出紙偏移，單位為 mm

43. setCutMode(mode, piece) 、 setCutMode_USB (mode, piece) 、 setCutMode_Ethernet (mode, piece)

- 函式說明：設定裁切模式與張數
- 參數說明：

參數	型別	說明
mode	int	設定裁切方式，預設為 1 0：反切 1：正切
piece	int	設定裁切張數

44. setAfterPrintAction(mode)、setAfterPrintAction_USB(mode)、setAfterPrintAction_Ethernet(mode)

- 函式說明：設定列印後動作
- 參數說明：

參數	型別	說明
mode	int	設定列印後動作，預設為 1 0：停在原地 1：撕紙 2：剝紙 3：裁切

45. genericDefault ()、genericDefault_USB ()、genericDefault_Ethernet()

- 函式說明：將印表機之一般設定值初始化
- 參數說明：無

46. sensorDefault ()、sensorDefault_USB ()、sensorDefault_Ethernet ()

- 函式說明：將印表機之感應器設定值初始化
- 參數說明：無

47. rfidSetupDefault ()、rfidSetupDefault_USB ()、rfidSetupDefault_Ethernet ()

- 函式說明：將 RFID 設定值初始化
- 參數說明：無

48. WifiFrequency(Frequency)、WifiFrequency_USB(Frequency)、WifiFrequency_Ethernet (Frequency)

- 函式說明：使用兼容 5G 頻段 WIFI 模塊時，可用於切換使用頻段
- 參數說明：

參數	型別	說明
Frequency	string	設定模塊頻段 2.4G：使用 2.4G 頻段 5G：使用 5G 頻段 BOTH：使用雙頻頻段

49. Bitmap(x,y, width,height,mode,filename)

Bitmap_USB(x,y, width,height,mode,filename)

Bitmap_Ethernet(x,y, width,height,mode,filename)

- 函式說明：將圖片轉為單色點陣圖，使用印表機直接列印
- 參數說明：

參數	型別	說明
x	string	文字 X 方向起始點，以點(dot)表示
y	string	文字 Y 方向起始點，以點(dot)表示
width	int	圖片寬度，以位元組(byte)表示
height	int	圖片高度，以點(dot)表示
mode	int	圖片格式 0: OVERWRITE 1: OR 2: XOR
filename	string	檔案名稱(可包含路徑) 圖檔僅支援以下格式： 1. BMP (Bitmap)：位圖格式 2. JPG (JPEG)：壓縮的圖像格式 3. PNG (Portable Network Graphics)：無損壓縮的圖像格式 4. GIF (Graphics Interchange Format)：支援多張圖片的格式，通常用於動畫 5. TIFF (Tagged Image File Format)：高品質的無損壓縮圖像格式 6. ICO (Icon)：圖示格式，用於顯示檔案、程式或資料夾的圖示 7. WMF (Windows Metafile)：Windows 繪圖文件格式 8. EMF (Enhanced Metafile)：擴展的 Windows 繪圖文件格式

50. compressBitmap(x,y, width,height,filename)

compressBitmap_USB(x,y, width,height,filename)

compressBitmap_Ethernet(x,y, width,height,filename)

■ 函式說明：將圖片轉為單色點陣圖，壓縮後再使用印表機列印

■ 參數說明：

參數	型別	說明
x	string	文字 X 方向起始點，以點(dot)表示
y	string	文字 Y 方向起始點，以點(dot)表示
width	int	圖片寬度，以位元組(byte)表示
height	int	圖片高度，以點(dot)表示
filename	string	檔案名稱(可包含路徑) 圖檔僅支援以下格式： 1. BMP (Bitmap)：位圖格式 2. JPG (JPEG)：壓縮的圖像格式 3. PNG (Portable Network Graphics)：無損壓縮的圖像格式 4. GIF (Graphics Interchange Format)：支援多張圖片的格式，通常用於動畫 5. TIFF (Tagged Image File Format)：高品質的無損壓縮圖像格式 6. ICO (Icon)：圖示格式，用於顯示檔案、程式或資料夾的圖示 7. WMF (Windows Metafile)：Windows 繪圖文件格式 8. EMF (Enhanced Metafile)：擴展的 Windows 繪圖文件格式

51. setResponse_USB(jobName, Mode)

■ 函式說明：自動響應指令

■ 參數說明：

參數	型別	說明
jobName	string	列印任務名稱。設置任務 ID，默認值為 Null
Mode	string	自動響應模式： ON:打開自動響應功能，任務開始每列印一張的開始與結束或途中有異常狀態皆返回列印任務狀態 OFF:關閉自動響應功能 ※列印內容一定要使用 printlabel 函式列印，後續才能成功接收

52. setResponseReadFromPrinter_USB()

- 函式說明：當 setResponse 為 ON，接收回應狀態
- 參數說明：無
- 回傳字串說明：

字串型別，回傳格式：{ 印表機狀態,任務編號,任務名稱,任務狀態}

Ex: {20,000001,Test,START}，START:開始列印時返回

{01,000001,,ERR}，ERR:進入錯誤時

{20,000001,,ERR}，ERR:退出錯誤時

{20,000001,,FS}，FS:開蓋後開始校準

{20,000001,Test,START}，START:錯誤後再次重新開始列印時返回

{20,000001,Test,STOP}，STOP:結束列印時返回

任務編號：共 6 碼，000001~999999(連續列印時，會從 000001 向上計數)

回傳字串	印表機狀態
00	就緒
01	上蓋開啟
02	卡紙
03	卡紙且上蓋開啟
04	標籤用盡
05	標籤用盡且上蓋開啟
08	碳帶用盡
09	碳帶用盡且上蓋開啟
0A	碳帶用盡且卡紙
0B	碳帶用盡、卡紙且上蓋開啟
0C	碳帶用盡且標籤用盡
0D	碳帶用盡、標籤用盡且上蓋開啟
10	暫停
20	列印中
80	其他錯誤

53. setResponseReceiveSTOP_USB()

- 函式說明：判斷 setResponse_USB 任務是否結束，停止接收
- 參數說明：無
- 回傳說明：int 型別，若為 0，表示任務還未結束；若為 1，表示任務結束

● C#(.Netframework)

1.需先將 GTSPPL_SDK.dll 加入參考

2.匯入 GTSPPL_SDK：

```
using GTSPPL_SDK;
```

3.範例程式：

```
//單機列印
```

```
Driver driver = new Driver();
```

```
driver.openport("Printer Model");
```

```
driver.setup("54", "30", "2", "3", "0", "3", "0");
```

```
driver.sendcommand("DIRECTION 1");
```

```
driver.setDirectionAndMirror(1, 1);
```

```
driver.setShift(50);
```

```
driver.setAfterPrintAction(2);
```

```
driver.setOffset(20);
```

```
driver.setCutMode(0,2);
```

```
driver.clearbuffer();
```

```
driver.sendcommand("TEXT 100,100,\"3\",0,1,1,\"REVERSE\");
```

```
driver.printReverse(90, 90, 128, 40);
```

```
driver.barcode("30", "30", "128", "100", "1", "0", "2", "2", "barcode1234567");
```

```
driver.qrcode("220", "260", "M", "3", "A", "0", "AABCB03abcN123");
```

```
driver.printerfont("50", "10", "3", "0", "1", "1", "Print Font 123456");
```

```
driver.printerfontblock("35", "15", "790", "90", "1", "0", "8", "8", "0", "1", "We stand behind our products  
with one of the most comprehensive support programs in the Auto-ID industry.");
```

```
string file = Environment.CurrentDirectory;
```

```

driver.downloadbmp(file + "\\CIRCLE.BMP", "CIRCLE.BMP");

driver.sendcommand ("PUTBMP 150,30,\"CIRCLE.BMP\"");

driver.windowfont(100, 20, 48, 0, 0, 0, "arial", "C# Driver test");

//BitmapResult 為 Bitmap 處理結果，成功為 1；失敗為 0

int BitmapResult = driver.Bitmap("-500", "70", 400, 350, 1, "Thunder.png ");

BitmapResult = driver.compressBitmap("-600", "70", 666, 357, "Cat1.jpg");

driver.printlabel("1", "1");

driver.genericDefault();

driver.sensorDefault();

driver.WifiFrequency("5G");


//RFID

driver.RFIDAutoCalibration();

driver.rfidSetupDefault();

//UHF GEN2

driver.writeUHF("H", 2, 12, "E", "414142424343444445454646"); // startBlockNo: Gen2 預設為 2

driver.printlabel("1", "1");

driver.EPCPWD_Action("U", "12345678");

driver.TIDPWD_Action("L", "12345678");

driver.USERPWD_Action("L", "12345678");

driver.AccessPWD_Action("S", "12345678");

driver.KillPWD_Action("L", "12345678");

driver.Set_RFIDPorcedure(8, 8, 32, 3, "N", 2, 2);

driver.Set_RFIDPorcedure_mm(5, 40, 32, 5, "N", 5, 5, "203");

driver.writeHF("H", 0, 12, "414142424343444445454646");

driver.printlabel("1", "1");

```

//UHF GJB 寫入資料

```
driver.writeGJB_UHF("H", 1, 12, "E", "414142424343444445454646", "12345678");
```

*帶入寫入密碼，寫入資料(startBlockNo: GJB 預設為 1)

//UHF GJB 設定各密碼區域新密碼

```
driver.setPWD_Action("S", "S", "11112222", "12345678"); //帶入寫入密碼，設定新狀態密碼
```

```
driver.setPWD_Action("R", "S", "33334444", "12345678"); //帶入寫入密碼，設定新讀取密碼
```

```
driver.setPWD_Action("K", "S", "55556666", "12345678"); //帶入寫入密碼，設定新刪除密碼
```

```
driver.setPWD_Action("W", "S", "87654321", "12345678"); //帶入舊寫入密碼，設定新寫入密碼
```

//UHF GJB 設定資料區塊狀態

```
driver.statusGJB_UHF("E", "B", "11112222"); //帶入狀態密碼，設定狀態
```

```
driver.printlabel("1", "1");
```

//UHF GJB 刪除標籤

```
driver.killGJB_UHF ("55556666"); //帶入刪除密碼，刪除標籤
```

```
driver.printlabel("1", "1");
```

```
driver.
```

```
();
```

//指定 USB 傳輸介面

```
USB usb = new USB();
```

```
usb.openport_USB();
```

```
string[] PrinterName = usb.detectUSB_USB(); //假設偵測印表機有一台以上
```

```
string[] PrinterName = usb.detectUSBStr_USB() == null ? null : usb.detectUSBStr_USB().Split('\n');
```

*先判斷 usb.detectUSBStr_USB()是否為 null，若不是，再以'\n'區分各個機型

```
usb.openports_USB(PrinterName[0]); //利用偵測到的第一台印表機連線
```

```
usb.setup_USB("54", "30", "2", "3", "0", "3", "0");
```

```
usb.sendcommand_USB("DIRECTION 1");
```

```

usb.setDirectionAndMirror_USB(1,1);

usb.setShift_USB(50);

usb.setAfterPrintAction_USB(2);

usb.setOffset_USB(20);

usb.setCutMode_USB(0,2);

usb.clearbuffer_USB();

usb.sendcommand_USB("TEXT 100,100,\"3\",0,1,1,\"REVERSE\");

usb.printReverse_USB(90,90,128,40);

usb.barcode_USB("30", "30", "128", "100", "1", "0", "2", "2", "barcode1234567");

usb.qrcode_USB("220", "260", "M", "3", "A", "0", "AABCB03abcN123");

usb.printerfont_USB("50", "10", "3", "0", "1", "1", "Print Font 123456");

string file = Environment.CurrentDirectory;

usb.downloadpcx_USB(file + "\\UL.PCX", "UL.PCX");

usb.windowfont_USB(10, 100, 48, 0, 0, 0, "arial", "C# WIN TEST");

//BitmapResult 為 Bitmap 處理結果，成功為 1；失敗為 0

int BitmapResult = usb.Bitmap_USB("-500", "70", 400, 350, 1, "Thunder.png");

BitmapResult=usb.compressBitmap_USB("-600", "70", 666, 357, "Cat1.jpg");

usb.printlabel_USB("1", "1");

var status = usb.printerstatus_USB();

usb.genericDefault_USB();

usb.sensorDefault_USB();

usb.WifiFrequency_USB("5G");

//簡中打印

string stString="默认简体中文测试";

usb.clearbuffer_USB();

usb.printerfont_USB ("100", "10", "TSS24.BF2", "0", "1", "1", stString);

```

```
usb.printlabel_USB (1, 1, this);
```

```
//繁中打印
```

```
string ttString="默認繁體中文測試";
```

```
usb.clearbuffer_USB();
```

```
usb.printerfont_USB("100", "10", " TST24.BF2", "0", "1", "1", ttString);
```

```
usb.printlabel_USB(1, 1, this);
```

```
//BLOCK 打印
```

```
string ttString="We stand behind our products with one of the most comprehensive support programs  
in the Auto-ID industry.";
```

```
usb.clearbuffer_USB();
```

```
usb.printerfontblock_USB("35", "15", "790", "90", "1", "0", "8", "8", "0", "1", ttString);
```

```
usb.printlabel_USB(1, 1, this);
```

```
//RFID
```

```
usb.RFIDAutoCalibration_USB();
```

```
usb.rfidSetupDefault_USB();
```

```
//UHF GEN2
```

```
usb.writeUHF_USB("H", 2, 12, "E", "414142424343444445454646"); // startBlockNo: Gen2 預設為 2
```

```
usb.printlabel_USB("1", "1");
```

```
string data = usb.readUHF_USB("H", 0, 12, "E");
```

```
string data_Q = usb.query_UHF_USB( "A", 0, 0); //以 Q 指令讀取 EPC 資料
```

```
usb.EPCPWD_Action_USB("L", "12345678");
```

```
usb.TIDPWD_Action_USB("L", "12345678");
```

```
usb.USERPWD_Action_USB("L", "12345678");
```

```
usb.AccessPWD_Action_USB("U", "12345678");
```

```
usb.KillPWD_Action_USB("U", "12345678");
```

```

usb.Set_RFIDPorcedure_USB(8,8,32,3,"N",2,2);

usb.Set_RFIDPorcedure_mm_USB(5, 40, 32, 5, "N", 5, 5, "203");

usb.writeHF_USB("H", 0, 12, "414142424343444445454646");

usb.printlabel_USB("1", "1");

//UHF GJB 寫入資料

usb.writeGJB_UHF_USB("H", 1, 12, "E", "414142424343444445454646", "12345678");

*帶入寫入密碼，寫入資料(startBlockNo: GJB 預設為 1)

//UHF GJB 設定各密碼區域新密碼

usb.setPWD_Action_USB ("S", "S", "11112222", "12345678"); //帶入寫入密碼，設定新狀態密碼
usb.setPWD_Action_USB ("R", "S", "33334444", "12345678"); //帶入寫入密碼，設定新讀取密碼
usb.setPWD_Action_USB ("K", "S", "55556666", "12345678"); //帶入寫入密碼，設定新刪除密碼
usb.setPWD_Action_USB ("W", "S", "87654321", "12345678"); //帶入舊寫入密碼，設定新寫入密碼

//UHF GJB 設定資料區塊狀態

usb.statusGJB_UHF_USB ("E", "B", "11112222"); //帶入狀態密碼，設定狀態

usb.printlabel_USB ("1", "1");

//UHF GJB 讀取資料

string data =usb. readGJB_UHF_USB("H", 0, 12, "E", "33334444"); //帶入讀取密碼，讀取標籤資料

//UHF GJB 刪除標籤

usb.killGJB_UHF_USB ("55556666"); //帶入刪除密碼，刪除標籤

usb.printlabel_USB ("1", "1");


//setResponse 自動響應功能

Timer receiveResponseTimer; //接收響應指令計時器

private void SetResponseBtn_Click(object sender, RoutedEventArgs e)

{

    usb.openport_USB();

```

```

usb.setResponse_USB("Test", "ON");

usb.sendcommand_USB("SIZE 4,2");

usb.sendcommand_USB("GAP 0,0");

usb.printlabel_USB("1", "5");    //列印標籤必須使用 printlabel 才可正常接收

Task.Factory.StartNew(() =>

{
    startReceiveResponseTimer();    //開啟自動響應 Timer

});

}

//開啟自動響應 Timer

public void startReceiveResponseTimer()

{

    receiveResponseTimer = new Timer();

    receiveResponseTimer.Interval = 800;

    receiveResponseTimer.Elapsed += timer_Elapsed;

    receiveResponseTimer.Start();

}

//Timer 的處理

public void timer_Elapsed(object sender, ElapsedEventArgs e)

{

    Dispatcher.BeginInvoke(new Action(() =>

    {

        string ReceiveStr=usb.getResponseReadFromPrinter_USB();

        if (usb.getResponseReceiveSTOP_USB() == 1)

        {

            usb.setResponse_USB("Test", "OFF");


```



```

        stopReceiveResponseTimer();
    }
    });
}

//關閉自動響應 Timer

public void stopReceiveResponseTimer()
{
    if (receiveResponseTimer != null)
    {
        receiveResponseTimer.Stop();
        receiveResponseTimer.Dispose();
        receiveResponseTimer = null;
    }
}

usb.closeport_USB();

usb.getDLLVersion_USB(1);

//指定網口傳輸介面

SocketConnect socketconnect = new SocketConnect();
socketconnect.openport_Ethernet("192.168.66.177", 9100);
socketconnect.setup_Ethernet ("54", "30", "2", "3", "0", "3", "0");
socketconnect.sendcommand_Ethernet ("DIRECTION 1");
socketconnect.setDirectionAndMirror_Ethernet(1,1);
socketconnect.setShift_Ethernet(50);
socketconnect.setAfterPrintAction_Ethernet(2);

```

```

socketconnect.setOffset_Ethernet(20);

socketconnect.setCutMode_Ethernet(0,2);

socketconnect.clearbuffer_Ethernet ();

socketconnect.sendcommand_Ethernet("TEXT 100,100,\"3\",0,1,1,\"REVERSE\");

socketconnect.printReverse_Ethernet(90, 90, 128, 40);

socketconnect.barcode_Ethernet ("30", "30", "128", "100", "1", "0", "2", "2", "barcode1234567");

socketconnect.qrcode_Ethernet ("220", "260", "M", "3", "A", "0", "AABCB03abcN123");

socketconnect.printerfont_Ethernet ("50", "10", "3", "0", "1", "1", "Print Font 123456");

string file = Environment.CurrentDirectory;

socketconnect.downloadpcx_Ethernet (file + "\\UL.PCX", "UL.PCX");

socketconnect.windowfont_Ethernet (10, 100, 48, 0, 0, 0, "arial", "C# WIN TEST");

//BitmapResult 為 Bitmap 處理結果，成功為 1；失敗為 0

int BitmapResult = socketconnect.Bitmap_Ethernet("-500", "70", 400, 350, 1, "Thunder.png");

BitmapResult = socketconnect.compressBitmap_Ethernet("-600", "70", 666, 357, "Cat1.jpg");

socketconnect.printlabel_Ethernet ("1", "1");

var status = socketconnect.printerstatus_Ethernet ();

socketconnect.genericDefault_Ethernet();

socketconnect.sensorDefault_Ethernet();

socketconnect.WifiFrequency_Ethernet("5G");

//簡中打印

string stString="默认简体中文测试";

socketconnect.clearbuffer_Ethernet ();

socketconnect.printerfont_Ethernet ("100", "10", "TSS24.BF2", "0", "1", "1", stString);

socketconnect.printlabel_Ethernet (1, 1, this);

//繁中打印

string ttString="默認繁體中文測試";

```

```

socketconnect.clearbuffer_Ethernet ();

socketconnect.printerfont_Ethernet ("100", "10", " TST24.BF2", "0", "1", "1", ttString);

socketconnect.printlabel_Ethernet (1, 1, this);

//BLOCK 打印

string ttString="We stand behind our products with one of the most comprehensive support programs
in the Auto-ID industry.";

socketconnect.clearbuffer_Ethernet ();

socketconnect.printerfontblock_Ethernet ("35","15","790","90","1","0","8","8","0","1", ttString);

socketconnect.printlabel_Ethernet (1, 1, this);


//RFID

socketconnect.RFIDAutoCalibration_Ethernet();

socketconnect.rfidSetupDefault_Ethernet();

//UHF GEN2

socketconnect.writeUHF_Ethernet("H", 2, 12,"E","414142424343444445454646"); //startBlockNo :
Gen2 預設為 2

socketconnect.printlabel_Ethernet ("1", "1");

string data = socketconnect.readUHF_Ethernet ("H", 0, 12, "E");

string data_Q = socketconnect.query_UHF_Ethernet ( "A", 0, 0); //以 Q 指令讀取 EPC 資料

socketconnect.EPCPWD_Action_Ethernet ("L","12345678");

socketconnect.TIDPWD_Action_Ethernet ("L", "12345678");

socketconnect.USERPWD_Action_Ethernet ("L", "12345678");

socketconnect.AccessPWD_Action_Ethernet ("U", "12345678");

socketconnect.KillPWD_Action_Ethernet ("U", "12345678");

socketconnect.Set_RFIDPorcedure_Ethernet (8,8,32,3,"N",2,2);

socketconnect.Set_RFIDPorcedure_mm_Ethernet(5, 40, 32, 5, "N", 5, 5, "203");

```

```

socketconnect.writeHF_Ethernet ("H", 0, 12, "414142424343444445454646");

socketconnect.printlabel_Ethernet ("1", "1");

//UHF GJB 寫入資料

socketconnect.writeGJB_UHF_Ethernet ("H", 1, 12, "E", "414142424343444445454646", "12345678");

*帶入寫入密碼，寫入資料(startBlockNo: GJB 預設為 1)

//UHF GJB 設定各密碼區域新密碼

socketconnect.setPWD_Action_Ethernet("S", "S", "11112222", "12345678"); //帶入寫入密碼，設定
新狀態密碼

socketconnect.setPWD_Action_Ethernet ("R", "S", "33334444", "12345678");//帶入寫入密碼，設定
新讀取密碼

socketconnect.setPWD_Action_Ethernet ("K", "S", "55556666", "12345678"); //帶入寫入密碼，設
定新刪除密碼

socketconnect.setPWD_Action_Ethernet("W", "S", "87654321", "12345678"); //帶入舊寫入密碼，設
定新寫入密碼

//UHF GJB 設定資料區塊狀態

socketconnect.statusGJB_UHF_Ethernet ("E", "B", "11112222");//帶入狀態密碼，設定狀態

socketconnect.printlabel_Ethernet ("1", "1");

//UHF GJB 讀取資料

string data = socketconnect. readGJB_UHF_Ethernet ("H", 0, 12, "E", "33334444"); //帶入讀取密碼，
讀取標籤資料

//UHF GJB 刪除標籤

socketconnect.killGJB_UHF_Ethernet ("55556666"); //帶入刪除密碼，刪除標籤

socketconnect.printlabel_Ethernet ("1", "1");

socketconnect.closeport_Ethernet ();

socketconnect.getDLLVersion_Ethernet (1);

```

1.需先匯入 jan-5.5.0.jar :

```
import com.sun.jna.Library;
```

```
import com.sun.jna.Native;
```

2.建立調用 dll 的 class 調用 dll

```
class GTSPL {

    interface GTSPL_SDK extends Library {

        GTSPL_SDK INSTANCE = (GTSPL_SDK) Native.load("GTSPL_SDK_C", GTSPL_SDK.class);

        int openport_USB();

        int openport(String PrinterName);

        ....

    }

}
```

3.範例程式：

```
System.load(System.getProperty("user.dir") + "\\GTSPL_SDK_C.dll");

//單機列印

GTSPL.GTSPL_SDK.INSTANCE.openport("Printer Model");

GTSPL.GTSPL_SDK.INSTANCE.setup("54", "30", "2", "3", "0", "3", "0");

GTSPL.GTSPL_SDK.INSTANCE.sendcommand("DIRECTION 1");

GTSPL.GTSPL_SDK.INSTANCE.setDirectionAndMirror(1, 1);

GTSPL.GTSPL_SDK.INSTANCE.setShift(50);

GTSPL.GTSPL_SDK.INSTANCE.setAfterPrintAction(2);

GTSPL.GTSPL_SDK.INSTANCE.setOffset(20);

GTSPL.GTSPL_SDK.INSTANCE.setCutMode(0,2);

GTSPL.GTSPL_SDK.INSTANCE.clearbuffer();

GTSPL.GTSPL_SDK.INSTANCE.sendcommand("TEXT 100,100,\"3\",0,1,1,\"REVERSE\");

GTSPL.GTSPL_SDK.INSTANCE.printReverse(90, 90, 128, 40);
```

```

GTSPL.GTSPL_SDK.INSTANCE.barcode("30", "30", "128", "100", "1", "0", "2", "2",
"barcode1234567");

GTSPL.GTSPL_SDK.INSTANCE.qrcode("50", "100", "H", "4", "A", "0", "QRcode1234567");

GTSPL.GTSPL_SDK.INSTANCE.printerfont("50", "10", "2", "0", "1", "1", "Print Font 123456");

GTSPL.GTSPL_SDK.INSTANCE.downloadbmp(System.getProperty("user.dir") + "\\CIRCLE.BMP",
"CIRCLE.BMP");

GTSPL.GTSPL_SDK.INSTANCE.windowfont(5, 150, 48, 0, 0, 0, "arial", "JAVA WIN DRIVER");

GTSPL.GTSPL_SDK.INSTANCE.sendcommand("PUTBMP 150,30,\"CIRCLE.BMP\"");

GTSPL.GTSPL_SDK.INSTANCE.printerfontblock("15", "15", "790", "90", "0", "0", "8", "8", "20",
"2","We stand behind our products with one of the most comprehensive support programs in the
Auto-ID industry.");

```

//初始化

```

GTSPL.GTSPL_SDK.INSTANCE.genericDefault();

GTSPL.GTSPL_SDK.INSTANCE.sensorDefault();

GTSPL.GTSPL_SDK.INSTANCE.rfidSetupDefault();

```

//修改 Wifi 频段

```

GTSPL.GTSPL_SDK.INSTANCE.WifiFrequency("5G");

```

//BLOCK 打印

```

WString str = new WString("-- 默认简体中文测试：海天米醋白米醋 1 瓶 450ml --");

GTSPL.GTSPL_SDK.INSTANCE.printerfontblockUnicode("15", "15", "790", "90", "TSS24.BF2", "0", "1",
"1", "0", "1", str);

GTSPL.GTSPL_SDK.INSTANCE.printlabel("1", "1");

GTSPL.GTSPL_SDK.INSTANCE.closeport();

```

```
GTSP.LGTSP.L_SDK.INSTANCE.getDLLVersion(0);
```

```
//UHF GEN2
```

```
GTSP.LGTSP.L_SDK.INSTANCE.writeUHF("H", 0, 12, "E", "11223344556677889900AABB");
```

```
GTSP.LGTSP.L_SDK.INSTANCE.printlabel("1", "1");
```

```
GTSP.LGTSP.L_SDK.INSTANCE.EPCPWD_Action("L", "12345678");
```

```
GTSP.LGTSP.L_SDK.INSTANCE.TIDPWD_Action("L", "12345678");
```

```
GTSP.LGTSP.L_SDK.INSTANCE.USERPWD_Action("L", "12345678");
```

```
GTSP.LGTSP.L_SDK.INSTANCE.AccessPWD_Action("U", "12345678");
```

```
GTSP.LGTSP.L_SDK.INSTANCE.KillPWD_Action("U", "12345678");
```

```
GTSP.LGTSP.L_SDK.INSTANCE.Set_RFIDPorcedure(8,8,32,"3","N",2,2);
```

```
GTSP.LGTSP.L_SDK.INSTANCE.Set_RFIDPorcedure_mm(5, 40, 32, 5, "N", 5, 5, "203");
```

```
GTSP.LGTSP.L_SDK.INSTANCE.writeHF("H", 0, 12, "414142424343444445454646")
```

```
GTSP.LGTSP.L_SDK.INSTANCE.printlabel("1", "1");
```

```
//UHF GJB 寫入資料
```

```
GTSP.LGTSP.L_SDK.INSTANCE.writeGJB_UHF("H", 1, 12, "E", "414142424343444445454646",  
"12345678");
```

*帶入寫入密碼，寫入資料(startBlockNo: GJB 預設為 1)

```
//UHF GJB 設定各密碼區域新密碼
```

```
GTSP.LGTSP.L_SDK.INSTANCE.setPWD_Action("S", "S", "11112222", "12345678");
```

//帶入寫入密碼，設定新狀態密碼

```
GTSP.LGTSP.L_SDK.INSTANCE.setPWD_Action("R", "S", "33334444", "12345678");
```

//帶入寫入密碼，設定新讀取密碼

```
GTSP.LGTSP.L_SDK.INSTANCE.setPWD_Action("K", "S", "55556666", "12345678");
```

//帶入寫入密碼，設定新刪除密碼

```
GTSP.LGTSP.L_SDK.INSTANCE.setPWD_Action("W", "S", "87654321", "12345678");
```

//帶入舊寫入密碼，設定新寫入密碼

//UHF GJB 設定資料區塊狀態

GTSP.LGTSP.L_SDK.INSTANCE.statusGJB_UHF("E", "B", "11112222");//帶入狀態密碼，設定狀態

GTSP.LGTSP.L_SDK.INSTANCE.printlabel("1", "1");

//UHF GJB 刪除標籤

GTSP.LGTSP.L_SDK.INSTANCE.killGJB_UHF ("55556666"); //帶入刪除密碼，刪除標籤

GTSP.LGTSP.L_SDK.INSTANCE.printlabel("1", "1");

GTSP.LGTSP.L_SDK.INSTANCE.closeport();

//簡中打印

WString str=new WString("默认简体中文测试");//需 jni 的 WString 才能正確傳送中文

GTSP.LGTSP.L_SDK.INSTANCE.clearbuffer();

GTSP.LGTSP.L_SDK.INSTANCE.printerfontUnicode("100", "10", "TSS24.BF2", "0", "1", "1", str);

GTSP.LGTSP.L_SDK.INSTANCE.printlabel(1, 1);

//繁中打印

WString str=new Wstring("默認繁體中文測試");//需 jni 的 WString 才能正確傳送中文

GTSP.LGTSP.L_SDK.INSTANCE.clearbuffer();

GTSP.LGTSP.L_SDK.INSTANCE. printerfontUnicode("100", "10", " TST24.BF2", "0", "1", "1", str);

GTSP.LGTSP.L_SDK.INSTANCE.printlabel(1, 1);

GTSP.LGTSP.L_SDK.INSTANCE.closeport();

GTSP.LGTSP.L_SDK.INSTANCE.getDLLVersion(0);

//RFID 自動校準

GTSP.LGTSP.L_SDK.INSTANCE.clearbuffer();

GTSP.LGTSP.L_SDK.INSTANCE.RFIDAutoCalibration();

GTSP.LGTSP.L_SDK.INSTANCE.closeport();

//指定 USB 傳輸介面

//偵測多台印表機


```

String PrinterName = GTSPL.GTSPL_SDK.INSTANCE.detectUSBStr_USB();

String[] PrinterNameStringArray = PrinterName.split("\\n+");

GTSPL.GTSPL_SDK.INSTANCE.openport_USB("Printer Model");

//單機列印

GTSPL.GTSPL_SDK.INSTANCE.openport_USB();

GTSPL.GTSPL_SDK.INSTANCE.setup_USB("54", "30", "2", "3", "0", "3", "0");

GTSPL.GTSPL_SDK.INSTANCE.sendcommand_USB("DIRECTION 1");

GTSPL.GTSPL_SDK.INSTANCE.setDirectionAndMirror_USB(1, 1);

GTSPL.GTSPL_SDK.INSTANCE.setShift_USB(50);

GTSPL.GTSPL_SDK.INSTANCE.setAfterPrintAction_USB(2);

GTSPL.GTSPL_SDK.INSTANCE.setOffset_USB(20);

GTSPL.GTSPL_SDK.INSTANCE.setCutMode_USB(0,2);

GTSPL.GTSPL_SDK.INSTANCE.clearbuffer_USB();

GTSPL.GTSPL_SDK.INSTANCE.sendcommand_USB("TEXT 100,100,\"3\",0,1,1,\"REVERSE\"");

GTSPL.GTSPL_SDK.INSTANCE.printReverse_USB(90, 90, 128, 40);

GTSPL.GTSPL_SDK.INSTANCE.barcode_USB("30", "30", "128", "100", "1", "0", "2", "2",
"barcode1234567");

GTSPL.GTSPL_SDK.INSTANCE.qrcode_USB("50", "100", "H", "4", "A", "0", "QRcode1234567");

GTSPL.GTSPL_SDK.INSTANCE.printerfont_USB("50", "10", "2", "0", "1", "1", "Print Font 123456");

GTSPL.GTSPL_SDK.INSTANCE.windowfont_USB(5, 150, 48, 0, 0, 0, "arial", "JAVA WIN DRIVER");

GTSPL.GTSPL_SDK.INSTANCE.downloadpcx_USB(System.getProperty("user.dir")+ "\\UL.PCX",
"UL.PCX");

GTSPL.GTSPL_SDK.INSTANCE.sendcommand_USB("PUTPCX 50,10,\"UL.PCX\"");

GTSPL.GTSPL_SDK.INSTANCE.printerfontblock_USB("15", "15", "790", "90", "0", "0", "8", "8", "20",
"2","We stand behind our products with one of the most comprehensive support programs in the
Auto-ID industry.");

```

//初始化

```
GTSP.LGTSP.L_SDK.INSTANCE.genericDefault_USB();
GTSP.LGTSP.L_SDK.INSTANCE.sensorDefault_USB();
GTSP.LGTSP.L_SDK.INSTANCE.rfidSetupDefault_USB();
```

//修改 Wifi 频段

```
GTSP.LGTSP.L_SDK.INSTANCE.WifiFrequency_USB ("5G");
```

//BLOCK 打印

```
WString str = new WString("-- 默认简体中文测试：海天米醋白米醋 1 瓶 450ml --");
GTSP.LGTSP.L_SDK.INSTANCE.printerfontblockUnicode_USB("15", "15", "790", "90", "TSS24.BF2",
"0", "1", "1", "0", "1", str);
GTSP.LGTSP.L_SDK.INSTANCE.printlabel_USB("1", "1");
String status = GTSP.LGTSP.L_SDK.INSTANCE.printerstatus_USB();
```

//UHF GEN2

```
GTSP.LGTSP.L_SDK.INSTANCE.writeUHF_USB("H", 0, 12, "E", "11223344556677889900AABB");
GTSP.LGTSP.L_SDK.INSTANCE.printlabel_USB("1", "1");
String status = GTSP.LGTSP.L_SDK.INSTANCE.readUHF_USB("A", 0, 12, "E");
GTSP.LGTSP.L_SDK.INSTANCE.EPCPWD_Action_USB("L", "12345678");
GTSP.LGTSP.L_SDK.INSTANCE.TIDPWD_Action_USB("L", "12345678");
GTSP.LGTSP.L_SDK.INSTANCE.USERPWD_Action_USB("L", "12345678");
GTSP.LGTSP.L_SDK.INSTANCE.AccessPWD_Action_USB("U", "12345678");
GTSP.LGTSP.L_SDK.INSTANCE.KillPWD_Action_USB("U", "12345678");
GTSP.LGTSP.L_SDK.INSTANCE.Set_RFIDPorcedure_USB(8,8,32,"3","N",2,2);
```

```

GTSP.LGTSP.L_SDK.INSTANCE.Set_RFIDPorcedure_mm_USB(5, 40, 32, 5, "N", 5, 5, "203");
GTSP.LGTSP.L_SDK.INSTANCE.writeHF_USB("H", 0, 12, "4141424243434444445454646")
GTSP.LGTSP.L_SDK.INSTANCE.printlabel_USB("1", "1");

```

//UHF GJB 寫入資料

```

GTSP.LGTSP.L_SDK.INSTANCE.writeGJB_UHF_USB("H", 1, 12, "E", "4141424243434444445454646",
"12345678");

```

*帶入寫入密碼，寫入資料(startBlockNo: GJB 預設為 1)

//UHF GJB 設定各密碼區域新密碼

```

GTSP.LGTSP.L_SDK.INSTANCE.setPWD_Action_USB ("S", "S", "11112222", "12345678");

```

//帶入寫入密碼，設定新狀態密碼

```

GTSP.LGTSP.L_SDK.INSTANCE.setPWD_Action_USB ("R", "S", "33334444", "12345678");

```

//帶入寫入密碼，設定新讀取密碼

```

GTSP.LGTSP.L_SDK.INSTANCE.setPWD_Action_USB ("K", "S", "55556666", "12345678");

```

//帶入寫入密碼，設定新刪除密碼

```

GTSP.LGTSP.L_SDK.INSTANCE.setPWD_Action_USB ("W", "S", "87654321", "12345678");

```

//帶入舊寫入密碼，設定新寫入密碼

//UHF GJB 設定資料區塊狀態

```

GTSP.LGTSP.L_SDK.INSTANCE.statusGJB_UHF_USB ("E", "B", "11112222");

```

//帶入狀態密碼，設定狀態

```

GTSP.LGTSP.L_SDK.INSTANCE.printlabel_USB( ("1", "1");

```

//UHF GJB 讀取資料

```

String data = GTSP.LGTSP.L_SDK.INSTANCE. readGJB_UHF_USB("H", 0, 12, "E", "33334444");

```

//帶入讀取密碼，讀取標籤資料

//UHF GJB 刪除標籤

```

GTSP.LGTSP.L_SDK.INSTANCE.killGJB_UHF_USB ("5555666"); //帶入刪除密碼，刪除標籤

GTSP.LGTSP.L_SDK.INSTANCE.printlabel_USB( "1", "1");

GTSP.LGTSP.L_SDK.INSTANCE.closeport_USB();

GTSP.LGTSP.L_SDK.INSTANCE.getDLLVersion_USB(1);

//簡中打印
WString str=new WString("默认简体中文测试");//需 jni 的 WString 才能正確傳送中文

GTSP.LGTSP.L_SDK.INSTANCE.clearbuffer_USB();

GTSP.LGTSP.L_SDK.INSTANCE.printerfontUnicode_USB ("100", "10", "TSS24.BF2", "0", "1", "1", str);

GTSP.LGTSP.L_SDK.INSTANCE.printlabel_USB (1, 1);

//繁中打印
WString str=new Wstring("默認繁體中文測試");//需 jni 的 WString 才能正確傳送中文

GTSP.LGTSP.L_SDK.INSTANCE.clearbuffer_USB();

GTSP.LGTSP.L_SDK.INSTANCE. printerfontUnicode_USB ("100", "10", " TST24.BF2", "0", "1", "1", str);

GTSP.LGTSP.L_SDK.INSTANCE.printlabel_USB(1, 1);

GTSP.LGTSP.L_SDK.INSTANCE.closeport_USB();

GTSP.LGTSP.L_SDK.INSTANCE.getDLLVersion_USB (0);

//RFID 自動校準

GTSP.LGTSP.L_SDK.INSTANCE.clearbufferr_USB ();

GTSP.LGTSP.L_SDK.INSTANCE.RFIDAutoCalibrationr_USB ();

GTSP.LGTSP.L_SDK.INSTANCE.closeportr_USB ();


//setResponse 自動響應功能
Timer receiveResponseTimer;    //接收響應指令計時器

private void set_response_usb_btnMouseClicked(java.awt.event.MouseEvent evt)
{
    GTSP.LGTSP.L_SDK.INSTANCE.setResponse_USB("Test","ON");

```

```

GTSP.LGTSP.L_SDK.INSTANCE.sendcommand_USB("SIZE 4,2");
GTSP.LGTSP.L_SDK.INSTANCE.sendcommand_USB("GAP 0,0");
GTSP.LGTSP.L_SDK.INSTANCE.printlabel_USB("1","5");
//列印標籤必須使用 printlabel 才可正常接收

ExecutorService executor = Executors.newSingleThreadExecutor();
executor.submit(() -> {
    startReceiveResponseTimer();    //開啟自動響應 Timer
});
executor.shutdown();
}

```

//開啟自動響應 Timer

```

public void startReceiveResponseTimer() {
    receiveResponseTimer = new Timer();
    receiveResponseTimer.scheduleAtFixedRate(new TimerTask() {
        @Override
        public void run() {
            timerElapsed();
        }
    }, 800, 800);    // 延遲 800 毫秒後開始，每 800 毫秒重複執行
}

```

//Timer 的處理

```

public void timerElapsed() {
    String response = GTSP.LGTSP.L_SDK.INSTANCE.getResponseReadFromPrinter_USB();
    SwingUtilities.invokeLater(() -> {

```

```

// 在 UI 執行續執行
set_response_usb_txt.setText(response + "\r\n");
});

if (GTSPL.GTSPL_SDK.INSTANCE.setResponseReceiveSTOP_USB() == 1) {
    stopReceiveResponseTimer();
}
}

//關閉自動響應 Timer

public void stopReceiveResponseTimer() {
    if (receiveResponseTimer != null) {
        receiveResponseTimer.cancel(); // 停止計時器任務
        receiveResponseTimer.purge(); // 移除所有已取消的任務
        receiveResponseTimer = null; // 釋放資源
    }
}

//指定 Ethernet 傳輸介面

GTSPL.GTSPL_SDK.INSTANCE.openport_Ethernet(IP,Port);
GTSPL.GTSPL_SDK.INSTANCE.setup_Ethernet("54", "30", "2", "3", "0", "3", "0");
GTSPL.GTSPL_SDK.INSTANCE.sendcommand_Ethernet("DIRECTION 1");
GTSPL.GTSPL_SDK.INSTANCE.setOffset_Ethernet(20);
GTSPL.GTSPL_SDK.INSTANCE.setCutMode_Ethernet(0,2);
GTSPL.GTSPL_SDK.INSTANCE.setDirectionAndMirror_Ethernet(1, 1);
GTSPL.GTSPL_SDK.INSTANCE.setShift_Ethernet(50);
GTSPL.GTSPL_SDK.INSTANCE.setAfterPrintAction_Ethernet(2);

```

```

GTSP.LGTSP.L_SDK.INSTANCE.clearbuffer_Ethernet();

GTSP.LGTSP.L_SDK.INSTANCE.sendcommand_Ethernet("TEXT 100,100,\"3\",0,1,1,\"REVERSE\"");

GTSP.LGTSP.L_SDK.INSTANCE.printReverse_Ethernet(90, 90, 128, 40);

GTSP.LGTSP.L_SDK.INSTANCE.barcode_Ethernet("30", "30", "128", "100", "1", "0", "2", "2",
"barcode1234567");

GTSP.LGTSP.L_SDK.INSTANCE.qrcode_Ethernet("50", "100", "H", "4", "A", "0", "QRcode1234567");

GTSP.LGTSP.L_SDK.INSTANCE.printerfont_Ethernet("50", "10", "2", "0", "1", "1", "Print Font
123456");

GTSP.LGTSP.L_SDK.INSTANCE.windowfont_Ethernet(5, 150, 48, 0, 0, 0, "arial", "JAVA WIN DRIVER");

GTSP.LGTSP.L_SDK.INSTANCE.downloadpcx_Ethernet(System.getProperty("user.dir")+ "\\UL.PCX",
"UL.PCX");

GTSP.LGTSP.L_SDK.INSTANCE.sendcommand_Ethernet("PUTPCX 50,10,\"UL.PCX\"");

GTSP.LGTSP.L_SDK.INSTANCE.printerfontblock_Ethernet("15", "15", "790", "90", "0", "0", "8", "8",
"20", "2", "We stand behind our products with one of the most comprehensive support programs in
the Auto-ID industry.");

//初始化

GTSP.LGTSP.L_SDK.INSTANCE.genericDefault_Ethernet();

GTSP.LGTSP.L_SDK.INSTANCE.sensorDefault_Ethernet();

GTSP.LGTSP.L_SDK.INSTANCE.rfidSetupDefault_Ethernet();

//修改 Wifi 频段

GTSP.LGTSP.L_SDK.INSTANCE.WifiFrequency_Ethernet ("5G");

//BLOCK 打印

WString str = new WString("-- 默认简体中文测试：海天米醋白米醋 1 瓶 450ml --");

```

```
GTSP.LGTSP.L_SDK.INSTANCE.printerfontblockUnicode_Ethernet("15", "15", "790",
"90", "TSS24.BF2", "0", "1", "1", "0", "1", str);
```

```
GTSP.LGTSP.L_SDK.INSTANCE.printlabel_Ethernet("1", "1");
```

```
String status = GTSP.LGTSP.L_SDK.INSTANCE.printerstatus_Ethernet();
```

```
//UHF GEN2
```

```
GTSP.LGTSP.L_SDK.INSTANCE.writeUHF_Ethernet("H", 0, 12, "E", "11223344556677889900AABB");
```

```
GTSP.LGTSP.L_SDK.INSTANCE.printlabel_Ethernet("1", "1");
```

```
String status = GTSP.LGTSP.L_SDK.INSTANCE.readUHF_Ethernet("A", 0, 12, "E");
```

```
GTSP.LGTSP.L_SDK.INSTANCE.EPCPWD_Action_Ethernet("L", "12345678");
```

```
GTSP.LGTSP.L_SDK.INSTANCE.TIDPWD_Action_Ethernet("L", "12345678");
```

```
GTSP.LGTSP.L_SDK.INSTANCE.USERPWD_Action_Ethernet("L", "12345678");
```

```
GTSP.LGTSP.L_SDK.INSTANCE.AccessPWD_Action_Ethernet("U", "12345678");
```

```
GTSP.LGTSP.L_SDK.INSTANCE.KillPWD_Action_Ethernet("U", "12345678");
```

```
GTSP.LGTSP.L_SDK.INSTANCE.Set_RFIDPorcedure_Ethernet(8, 8, 32, "3", "N", 2, 2);
```

```
GTSP.LGTSP.L_SDK.INSTANCE.Set_RFIDPorcedure_mm_Ethernet(5, 40, 32, 5, "N", 5, 5, "203");
```

```
GTSP.LGTSP.L_SDK.INSTANCE.writeHF_Ethernet("H", 0, 12, "414142424343444445454646")
```

```
GTSP.LGTSP.L_SDK.INSTANCE.printlabel_Ethernet("1", "1");
```

```
//UHF GJB 寫入資料
```

```
GTSP.LGTSP.L_SDK.INSTANCE.writeGJB_UHF_Ethernet("H", 1, 12, "E",
```

```
"414142424343444445454646", "12345678");
```

*帶入寫入密碼，寫入資料(startBlockNo: GJB 預設為 1)

```
//UHF GJB 設定各密碼區域新密碼
```

```
GTSP.LGTSP.L_SDK.INSTANCE.setPWD_Action_Ethernet("S", "S", "11112222", "12345678");
```

```
//帶入寫入密碼，設定新狀態密碼
```



```
GTSP.LGTSP.L_SDK.INSTANCE.setPWD_Action_Ethernet("R", "S", "33334444", "12345678");
```

```
//帶入寫入密碼，設定新讀取密碼
```

```
GTSP.LGTSP.L_SDK.INSTANCE.setPWD_Action_Ethernet("K", "S", "55556666", "12345678");
```

```
//帶入寫入密碼，設定新刪除密碼
```

```
GTSP.LGTSP.L_SDK.INSTANCE.setPWD_Action_Ethernet("W", "S", "87654321", "12345678");
```

```
//帶入舊寫入密碼，設定新寫入密碼
```

```
//UHF GJB 設定資料區塊狀態
```

```
GTSP.LGTSP.L_SDK.INSTANCE.statusGJB_UHF_Ethernet("E", "B", "11112222");
```

```
//帶入狀態密碼，設定狀態
```

```
GTSP.LGTSP.L_SDK.INSTANCE.printlabel_Ethernet( ("1", "1");
```

```
//UHF GJB 讀取資料
```

```
String data = GTSP.LGTSP.L_SDK.INSTANCE.readGJB_UHF_Ethernet("H", 0, 12, "E", "33334444");
```

```
//帶入讀取密碼，讀取標籤資料
```

```
//UHF GJB 刪除標籤
```

```
GTSP.LGTSP.L_SDK.INSTANCE.killGJB_UHF_Ethernet("55556666"); //帶入刪除密碼，刪除標籤
```

```
GTSP.LGTSP.L_SDK.INSTANCE.printlabel_Ethernet( ("1", "1");
```

```
GTSP.LGTSP.L_SDK.INSTANCE.closeport_Ethernet();
```

```
GTSP.LGTSP.L_SDK.INSTANCE.getDLLVersion_Ethernet(1);
```

```
//簡中打印
```

```
WString str=new WString("默认简体中文测试"); //需 jni 的 WString 才能正確傳送中文
```

```
GTSP.LGTSP.L_SDK.INSTANCE.clearbuffer_Ethernet();
```

```
GTSP.LGTSP.L_SDK.INSTANCE.printerfontUnicode_Ethernet("100", "10", "TSS24.BF2", "0", "1", "1",
```

```
str);
```

```
GTSP.LGTSP.L_SDK.INSTANCE.printlabel_Ethernet(1, 1);
```

```
//繁中打印
```

WString str=new Wstring("默認繁體中文測試");//需 jni 的 WString 才能正確傳送中文

```

GTSP.LGTSP.L_SDK.INSTANCE.clearbuffer_Ethernet();

GTSP.LGTSP.L_SDK.INSTANCE. printerfontUnicode_Ethernet("100", "10", " TST24.BF2", "0", "1", "1",
str);

GTSP.LGTSP.L_SDK.INSTANCE.printlabel_Ethernet(1, 1);

GTSP.LGTSP.L_SDK.INSTANCE.closeport_Ethernet();

GTSP.LGTSP.L_SDK.INSTANCE.getDLLVersion_Ethernet(0);

//RFID 自動校準

GTSP.LGTSP.L_SDK.INSTANCE.clearbufferr_Ethernet();

GTSP.LGTSP.L_SDK.INSTANCE.RFIDAutoCalibrationr_Ethernet();

GTSP.LGTSP.L_SDK.INSTANCE.closeport_Ethernet();

```

4.DLL 放置位置：

〔利用 IDE 開發時〕

GTSP.L_SDK.dll 放在 jdk 的 bin 資料夾下

如：C:\Program Files\Java\jdk1.8.0_221\bin

GTSP.L_SDK_C.dll 放在 java 專案資料夾下

〔未安裝 jdk 直接執行.jar 檔時〕

GTSP.L_SDK.dll 放在 jre 的 bin 資料夾下

如：C:\Program Files\Java\jre1.8.0_251\bin

GTSP.L_SDK_C.dll 放在.jar 檔相同路徑下

.jar 檔相同路徑下，還需要有 lib 資料夾，且裡面要有 jna-5.5.0.jar

5.Driver 模式下，需要安裝印表機的驅動才能執行

● Javascript

1.使用管理員身分開啟命令提示字元

2.註冊 dll

將 x86 的 dll 放到 Windows/Syswow64 , x64 的 dll 放在 Windows/System32

進入 microsoft.net framework 的安裝路徑下

如：cd C:\Windows\Microsoft.NET\Framework64\v4.0.30319

*x64 的 dll 註冊在 C:\Windows\Microsoft.NET\Framework64\v4.0.30319

*x86 的 dll 註冊在 C:\Windows\Microsoft.NET\Framework\v4.0.30319

輸入指令 regasm.exe /register /tlb <dll path>

輸入指令 regasm.exe <dll path> /codebase

如：regasm.exe /register /tlb C:\Users\User\Desktop\Javascript_Sample_Code\GTSPL_SDK.dll

regasm.exe C:\Users\User\source\repos\product\GTSPL_SDK \GTSPL_SDK.dll /codebase

3.範例程式：

```
<script language="javascript" type="text/javascript">
```

```
//單機列印
```

```
var driverObject = new ActiveXObject("GTSPL_SDK.Driver");
```

```
driverObject.openport("Printer Model");
```

```
driverObject.setup("54", "30", "2", "3", "0", "3", "0");
```

```
driverObject.sendcommand("DIRECTION 1");
```

```
driverObject.clearbuffer();
```

```
driverObject.barcode("30", "30", "128", "100", "1", "0", "2", "2", "barcode1234567");
```

```
driverObject.printerfont("50", "10", "3", "0", "1", "1", "Print Font 123456");
```

```
var file = getFilepath();
```

```
driverObject.downloadpcx(file + "\\UL.PCX", "UL.PCX");
```

```
driverObject.sendcommand("PUTPCX 50,10,\"UL.PCX\"");
```

```
driverObject.windowfont(50, 20, 48, 0, 0, 0, "arial", "Windows Font Test123467");
```

```
driverObject.getDLLVersion(1)

driverObject.printlabel("1", "1");

driverObject.closeport();
```

//指定 USB 傳輸介面

```
var usbObject = new ActiveXObject("GTSPL_SDK.USB");

usbObject.openport_USB();

usbObject.setup_USB("54", "30", "2", "3", "0", "3", "0");

usbObject.sendcommand_USB("DIRECTION 1");

usbObject.clearbuffer_USB();

usbObject.barcode_USB("30", "30", "128", "100", "1", "0", "2", "2", "barcode1234567");

usbObject.printerfont_USB("50", "10", "3", "0", "1", "1", "Print Font 123456");

var file = getFilePath();

usbObject.downloadbmp_USB(file + "\\CIRCLE.BMP", "CIRCLE.BMP");

usbObject.sendcommand_USB("PUTBMP 150,30,\"CIRCLE.BMP\"");

usbObject.windowfont_USB(10, 100, 48, 0, 0, 0, "arial", "Windows Font Test");

var status = usbObject.printerstatus_USB();

usbObject.printlabel_USB("1", "1");
```

//簡中打印

```
string stString="默认简体中文测试";

usbObject.clearbuffer_USB();

usbObject.printerfont_USB("100", "10", "TSS24.BF2", "0", "1", "1", stString);

usbObject.printlabel_USB(1, 1,);
```

//繁中打印

```

string ttString="默認繁體中文測試";

usbObject.clearbuffer_USB();

usbObject.printerfont_USB ("100", "10", " TST24.BF2", "0", "1", "1", ttString);

usbObject.printlabel_USB (1, 1);

usbObject.getDLLVersion_USB(1);

usbObject.closeport_USB();

usbObject.getDLLVersion_USB(0)

</script>

```

● Asp.Net

1.加入 x86 的 dll 進專案

2.範例程式：

//單機列印

```

GTSPL_SDK.Driver driver = new GTSPL_SDK.Driver();

driver.openport("Printer Model");

driver.setup("54", "30", "2", "3", "0", "3", "0");

driver.sendcommand("DIRECTION 1");

driver.clearbuffer();

driver.barcode("30", "30", "128", "100", "1", "0", "2", "2", "barcode1234567");

string path = HttpContext.Current.Server.MapPath("~/");

driver.downloadpcx(path + "\\CIRCLE.BMP", "CIRCLE.BMP");

driver.sendcommand("PUTBMP 150,30,\"CIRCLE.BMP\"");

driver.windowfont(50, 20, 48, 0, 0, 0, "impact", "Windows Font ASP.NET");

driver.getDLLVersion(1);

driver.printlabel("1", "1");

driver.closeport();

```

//指定 USB 傳輸介面

```

GTSPL_SDK.USB usb = new GTSPL_SDK.USB();

usb.openport_USB();

usb.setup_USB("54", "30", "2", "3", "0", "3", "0");

usb.sendcommand_USB("DIRECTION 1");

usb.clearbuffer_USB();

usb.barcode_USB("30", "30", "128", "100", "1", "0", "2", "2", "barcode1234567usb");

usb.printerfont_USB("50", "10", "3", "0", "1", "1", "Print Font usb Asp");

usb.downloadpcx_USB(path + "\\UL.PCX", "UL.PCX");

usb.sendcommand_USB("PUTPCX 50,10,\"UL.PCX\"");

usb.windowfont_USB(10, 100, 48, 0, 0, 0, "impact", "Windows Font ASP USB");

usb.printlabel_USB("1", "1");

var status = usb.printerstatus_USB();

usb.getDLLVersion_USB(1);

```

//簡中打印

```

string stString="默认简体中文测试";

usb.clearbuffer_USB();

usb.printerfont_USB("100", "10", "TSS24.BF2", "0", "1", "1", stString);

usb.printlabel_USB(1, 1);

```

//繁中打印

```

string ttString="默認繁體中文測試";

usb.clearbuffer_USB();

usb.printerfont_USB ("100", "10", " TST24.BF2", "0", "1", "1", ttString);

usb.printlabel_USB (1, 1);

```

```
usb.closeport_USB();
```

● VB.Net

1.加入 dll 進專案

2.範例程式：

```
Dim path = Environment.CurrentDirectory

//單機列印

Dim driver As New GTSPL_SDK.Driver

driver.openport("Printer Model")

driver.setup("54", "30", "2", "3", "0", "3", "0")

driver.sendcommand("DIRECTION 1")

driver.clearbuffer()

driver.barcode("30", "30", "128", "100", "1", "0", "2", "2", "barcode1234567")

driver.printerfont("50", "10", "3", "0", "1", "1", "PrintFont in vb.net")

driver.downloadbmp(path + "\\CIRCLE.BMP", "CIRCLE.BMP")

driver.sendcommand("PUTBMP 50,10, ""CIRCLE.BMP""")

driver.windowfont(50, 20, 48, 0, 0, 0, "impact", "VB Imapct Driver")

driver.printlabel("1", "1")

driver.closeport()


//指定 USB 傳輸介面

Dim usb As New GTSPL_SDK.USB

usb.openport_USB()

usb.setup_USB("54", "30", "2", "3", "0", "3", "0")

usb.sendcommand_USB("DIRECTION 1")
```

```

usb.clearbuffer_USB()

usb.barcode_USB("30", "30", "128", "100", "1", "0", "2", "2", "barcode1234567")

usb.downloadbmp_USB(path + "\\CIRCLE.BMP", "CIRCLE.BMP")

usb.sendcommand_USB("PUTBMP 50,10,""CIRCLE.BMP"")

usb.windowfont_USB(50, 20, 48, 0, 0, 0, "impact", "VB Imapct Driver")

Dim status = usb.printerstatus_USB()

Dim vcode = usb.getDLLVersion_USB("1")

usb.printlabel_USB("1", "1")

usb.closeport_USB()

//簡中打印

usb.clearbuffer_USB();

usb.printerfont_USB("100", "10", "TSS24.BF2", "0", "1", "1", "默认简体中文测试");

usb.printlabel_USB(1, 1);

//繁中打印

usb.clearbuffer_USB();

usb.printerfont_USB ("100", "10", " TST24.BF2", "0", "1", "1", "默認繁體中文測試");

usb.printlabel_USB (1, 1);

usb.closeport_USB();

```

● Access VBA

1.使用管理員身分開啟命令提示字元

2.註冊 dll

將 x86 的 dll 放到 Windows/Syswow64

進入 microsoft.net framework 的安裝路徑下

如：cd C:\Windows\[Microsoft.NET](#)\Framework\v4.0.30319

輸入指令 regasm.exe /register /tlb <dll path>

輸入指令 regasm.exe <dll path> /codebase

如：regasm.exe /register /tlb C:\Users\User\Desktop\Javascript_Sample_Code\GTSPL_SDK.dll

regasm.exe C:\Users\User\source\repos\product\GTSPL_SDK \GTSPL_SDK.dll /codebase

3.範例程式：

```
Dim usb As New GTSPL_SDK.usb
```

```
Dim path As String
```

```
usb.openport_USB()
```

```
Call usb.clearbuffer_USB
```

```
Call usb.setup_USB("54", "39", "4", "7", "0", "3", "0")
```

```
Call usb.clearbuffer_USB
```

```
Call usb.printerfont_USB("10", "20", "2", "0", "1", "1", "Product Management")
```

```
Call usb.printerfont_USB("10", "40", "2", "0", "1", "1", "Product No. : ")
```

```
Call usb.barcode_USB("10", "60", "128", "50", "1", "0", "1", "1", ProductNo)
```

```
Call usb.printerfont_USB("10", "140", "2", "0", "1", "1", "Product Name : " + ProductName)
```

```
Call usb.printerfont_USB("10", "160", "2", "0", "1", "1", "Unit Price : " + UnitPrice)
```

```
Call usb.printerfont_USB("10", "180", "2", "0", "1", "1", "Amount : " + Amount)
```

```
Call usb.printerfont_USB("10", "200", "2", "0", "1", "1", "Total Price : " + TotalPrice)
```

```
Call usb.sendcommand_USB("PUTBMP 10,230,\"\"LOGO.BMP\"\"")
```

```
Call usb.printlabel_USB("1", "1")
```

```
Call usb.closeport_USB
```

1.使用管理員身分開啟命令提示字元

2.註冊 dll

將 x86 的 dll 放到 Windows/Syswow64

進入 microsoft.net framework 的安裝路徑下

如：cd C:\Windows\[Microsoft.NET](#)\Framework\v4.0.30319

輸入指令 regasm.exe /register /tlb <dll path>

輸入指令 regasm.exe <dll path> /codebase

如：regasm.exe /register /tlb C:\Users\User\Desktop\Javascript_Sample_Code\GTSPL_SDK.dll

regasm.exe C:\Users\User\source\repos\product\GTSPL_SDK \GTSPL_SDK.dll /codebase

3.範例程式：

```
Dim driver As New GTSPL_SDK.driver
```

```
Dim path As String
```

```
openport = driver.openport("Printer Model")
```

```
Call driver.setup("54", "20", "2", "3", "0", "3", "0")
```

```
Call driver.sendcommand("DIRECTION 1")
```

```
Call driver.clearbuffer
```

```
Call driver.barcode("30", "30", "128", "100", "1", "0", "2", "2", "barcode1234567")
```

```
Call driver.printerfont("50", "10", "3", "0", "1", "1", "PrintFont in vba_xls")
```

```
Call driver.downloadpcx(path + "\\UL.PCX", "UL.PCX")
```

```
Call driver.sendcommand("PUTPCX 50,10,\"\"UL.PCX\"\"")
```

```
Call driver.windowfont(50, 20, 48, 0, 0, 0, "impact", "VBAEX Driver")
```

```
Call driver.printlabel("1", "1")
```

```
Call driver.closeport
```

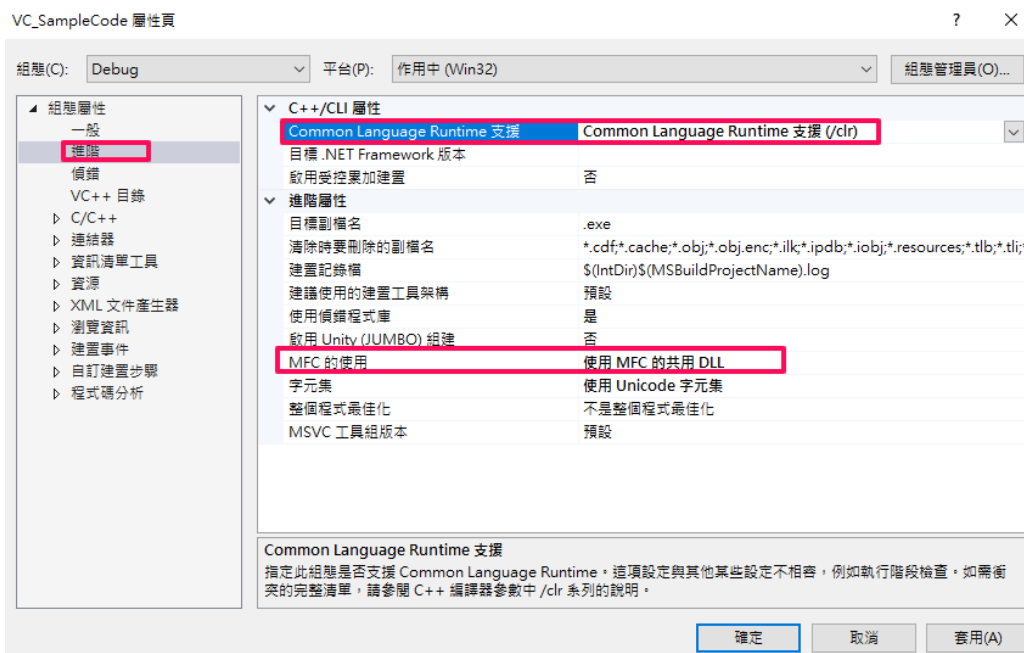
1.設定專案屬性

要支援 dll 必須將 Common Language Runtime 支援啟動

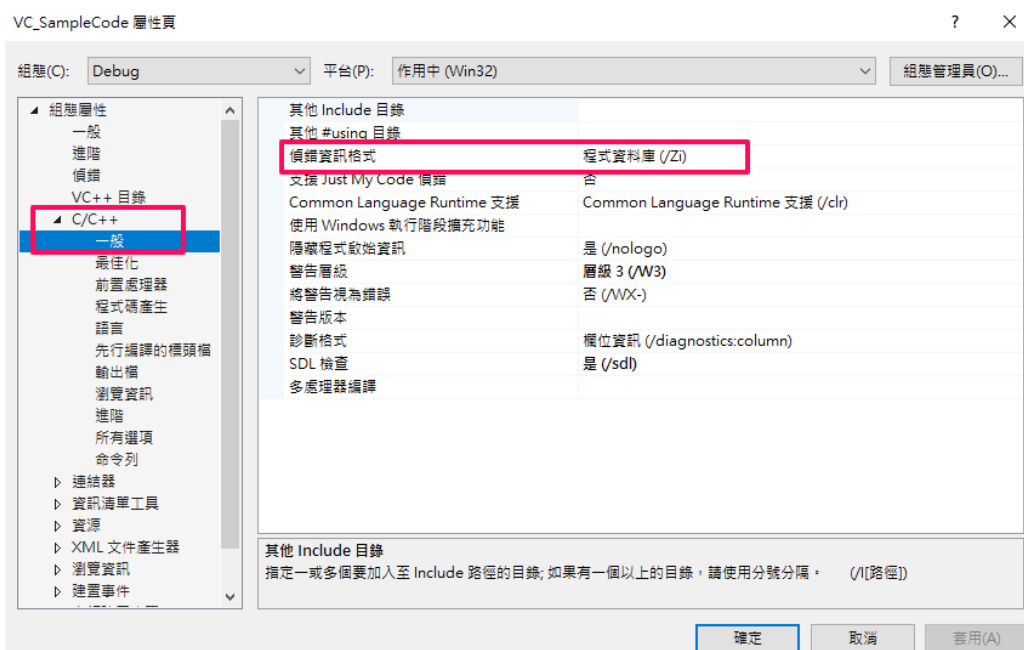
=>開啟專案屬性

=>組態屬性=>進階=>Common Language Runtime 支援設定為(/clr)

=>組態屬性=>進階=>MFC 的使用設定為使用 MFC 的共用 DLL

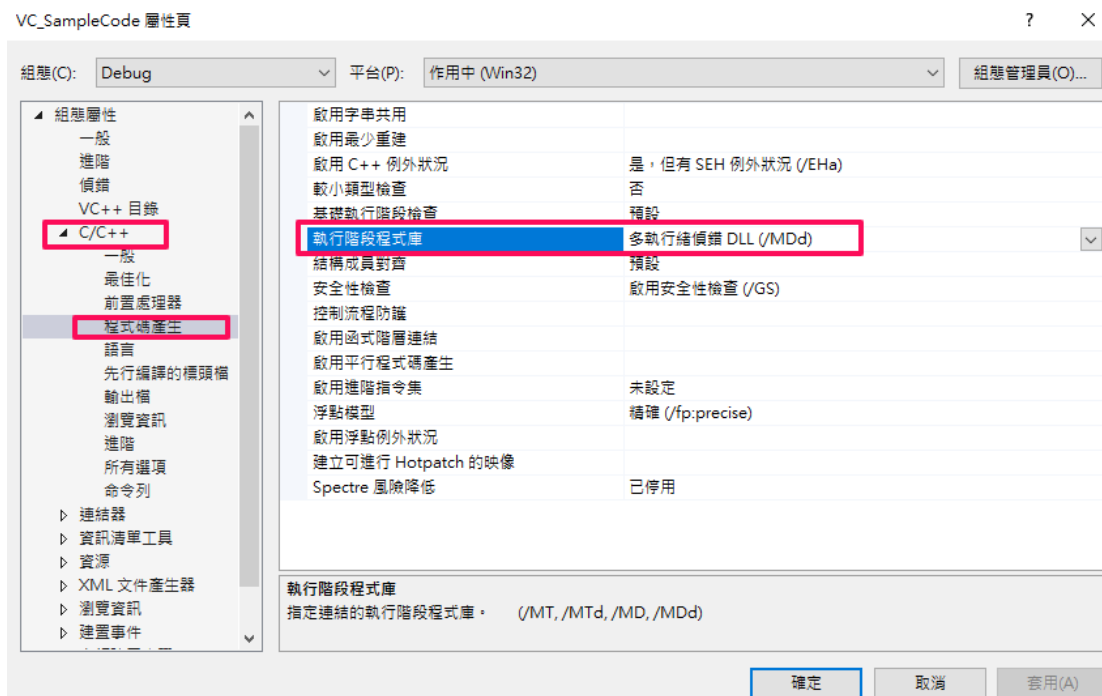


屬性=>C/C++=>一般=>偵錯資訊格式設定為"程式資料庫/Zi"(不可以設定為/ZI)



屬性=>C/C++=>程式碼產生

=>執行階段程式庫設定為"多執行緒偵錯 DLL(/MDd)"或"多執行緒(/MD)"



3.範例程式：

```
#using "GTSPL_SDK.dll"

using namespace System;

using namespace GTSPL_SDK;

//單機列印

Driver^ driver = gcnew Driver();

String^ printerName = gcnew String(str);

driver->openport(printerName);

driver->setup("30", "30", "2", "3", "0", "3", "0");

driver->clearbuffer();

driver->barcode("30", "30", "128", "100", "1", "0", "2", "2", "barcode VC5678 ");

driver->printerfont("10", "150", "2", "0", "1", "1", "Print Font VC DRIVER");

driver->downloadpcx(_T("../VC_SampleCode /UL.PCX"), "UL.PCX");
```

```

driver->sendcommand("PUTPCX 50,10,\"UL.PCX\"");

driver ->downloadbmp_USB(_T("../VC_SampleCode /CIRCLE.BMP"), "CIRCLE.BMP");

driver ->sendcommand_USB("PUTBMP 150,30,\"CIRCLE.BMP\"");

driver->windowsfont(10, 120, 48, 0, 0, 0, " arial ", "VC Driver test");

driver ->qrcode ("220", "260", "M", "3", "A", "0", "AABCB03abcN123");

driver ->printerfontblock("35","15","790","90","1","0","8","8","0","1", ttString);

driver ->setDirectionAndMirror (1,1);

driver ->setShift (50);

driver ->setAfterPrintAction(2);

driver ->setOffset (20);

driver ->setCutMode (0,2);

driver ->printReverse (90,90,128,40);

driver->printlabel("1", "1");

driver->sendcommand("DIRECTION 1");

driver->genericDefault ();

driver->sensorDefault ();

int BitmapResult = driver->Bitmap ("0", "0", 640, 484, 1, "octopus.png");

BitmapResult = driver->compressBitmap ("0", "0", 640, 484, 1, "octopus.jpg");

driver->printlabel ("1", "1");

// RFID

driver-> rfidSetupDefault();

driver->RFIDAutoCalibration ();

//UHF GEN2

// startBlockNo: Gen2 預設為 2

driver->writeUHF("H", 2, 12, "E", "414142424343444445454646");

driver->printlabel("1", "1");

```

```

driver->EPCPWD_Action("U", "12345678");

driver->TIDPWD_Action("L", "12345678");

driver->USERPWD_Action("L", "12345678");

driver->AccessPWD_Action("S", "12345678");

driver->KillPWD_Action("L", "12345678");

driver->Set_RFIDPorcedure(8, 8, 32, 3, "N", 2, 2);

driver->Set_RFIDPorcedure_mm (5, 40, 32, 5, "N", 5, 5, "203");

driver->writeHF("H", 0, 12, "414142424343444445454646");

driver->printlabel("1", "1");

//UHF GJB 寫入資料

driver->writeGJB_UHF("H", 1, 12, "E", "414142424343444445454646", "12345678");

*帶入寫入密碼，寫入資料(startBlockNo: GJB 預設為 1)

//UHF GJB 設定各密碼區域新密碼

driver->setPWD_Action("S", "S", "11112222", "12345678"); //帶入寫入密碼，設定新狀態密碼

driver->setPWD_Action("R", "S", "33334444", "12345678"); //帶入寫入密碼，設定新讀取密碼

driver->setPWD_Action("K", "S", "55556666", "12345678"); //帶入寫入密碼，設定新刪除密碼

//帶入舊寫入密碼，設定新寫入密碼*

driver->setPWD_Action("W", "S", "87654321", "12345678");

//UHF GJB 設定資料區塊狀態

driver->statusGJB_UHF("E", "B", "11112222"); //帶入狀態密碼，設定狀態

driver->printlabel("1", "1");

//UHF GJB 刪除標籤

driver->killGJB_UHF ("55556666"); //帶入刪除密碼，刪除標籤

driver->printlabel("1", "1");

driver->closeport();

```

//指定 USB 傳輸介面

```

USB^ usb = gcnew USB();

usb->openport_USB();

usb->setup_USB("54", "30", "2", "3", "0", "3", "0");

usb->sendcommand_USB("DIRECTION 1");

usb->clearbuffer_USB();

usb->barcode_USB("10", "80", "128", "100", "1", "0", "2", "2", "barcode vc");

usb->printerfont_USB("50", "100", "2", "0", "1", "1", "Print Font VC123");

usb->downloadpcx_USB(_T("../VC_SampleCode/UL.PCX "), "UL.PCX");

usb->sendcommand_USB("PUTPCX 50,10,\"UL.PCX\"");

usb->downloadbmp_USB(_T("../VC_SampleCode/ CIRCLE.BMP "), "CIRCLE.BMP");

usb->sendcommand_USB("PUTBMP 150,30,\"CIRCLE.BMP\"");

usb->windowsfont_USB(10, 20, 48, 0, 0, 0, "arial", "VC WIN TEST");

usb ->qrcode_USB("220", "260", "M", "3", "A", "0", "AABCB03abcN123");

usb ->printerfontblock_USB("35","15","790","90","1","0","8","8","0","1", ttString);

usb->getDLLVersion_USB(1);

CString status;

status=usb->printerstatus_USB();

usb->printlabel_USB("1", "1");

```

//簡中打印

```

usb.clearbuffer_USB();

usb->printerfont_USB("10", "110", "TSS24.BF2", "0", "1", "1", L"默认简体中文测试：海天米醋  
白米醋 1 瓶 450ml");

usb->printlabel_USB("1", "1");;

```

//繁中打印

```

usb.clearbuffer_USB();

usb->printerfont_USB("10", "110", "TST24.BF2", "0", "1", "1", L"默認繁體中文測試：海天米醋
白米醋 1 瓶 450ml");

usb ->setDirectionAndMirror_USB (1,1);

usb ->setShift_USB (50);

usb ->setAfterPrintAction_USB (2);

usb ->setOffset_USB (20);

usb ->setCutMode_USB (0,2);

usb ->printReverse_USB (90,90,128,40);

usb ->printlabel_USB ("1", "1");

usb ->sendcommand_USB ("DIRECTION 1");

usb ->genericDefault_USB ();

usb ->sensorDefault_USB ();

int BitmapResult = usb ->Bitmap_USB ("0", "0", 640, 484, 1, "octopus.png");

BitmapResult = usb ->compressBitmap_USB ("0", "0", 640, 484, 1, "octopus.jpg");

usb ->printlabel_USB ("1", "1");

// RFID

usb -> rfidSetupDefault_USB ();

usb ->RFIDAutoCalibration_USB ();

//UHF GEN2

usb ->writeUHF_USB("H", 2, 12, "E", "414142424343444445454646");    // startBlockNo: Gen2
預設為 2

usb->printlabel_USB("1", "1");

string data = usb->readUHF_USB("H", 2, 12, "E");

string data_Q = usb.query_UHF_USB("A", 0, 0);    //以 Q 指令讀取 EPC 資料

```



```

usb->EPCPWD_Action_USB("L","12345678");

usb->TIDPWD_Action_USB("L", "12345678");

usb->USERPWD_Action_USB("L", "12345678");

usb->AccessPWD_Action_USB("U", "12345678");

usb->KillPWD_Action_USB("U", "12345678");

usb->Set_RFIDPorcedure_USB(8,8,32,3,"N",2,2);

usb->Set_RFIDPorcedure_mm_USB(5, 40, 32, 5, "N", 5, 5, "203");

usb->writeHF_USB("H", 2, 12, "414142424343444445454646");

usb->printlabel_USB("1", "1");

//UHF GJB 寫入資料

usb ->writeGJB_UHF_USB("H", 2, 12, "E", "414142424343444445454646", "12345678");

*帶入寫入密碼，寫入資料(startBlockNo: GJB 預設為 1)

//UHF GJB 設定各密碼區域新密碼

//帶入寫入密碼，設定新狀態密碼

usb ->setPWD_Action_USB ("S", "S", "11112222", "12345678");

//帶入寫入密碼，設定新讀取密碼

usb ->setPWD_Action_USB ("R", "S", "33334444", "12345678");

//帶入寫入密碼，設定新刪除密碼

usb ->setPWD_Action_USB ("K", "S", "55556666", "12345678");

//帶入舊寫入密碼，設定新寫入密碼*

usb ->setPWD_Action_USB ("W", "S", "87654321", "12345678");

//UHF GJB 設定資料區塊狀態

usb ->statusGJB_UHF_USB ("E", "B", "11112222"); //帶入狀態密碼，設定狀態

usb ->printlabel_USB ("1", "1");

//UHF GJB 讀取資料

//帶入讀取密碼，讀取標籤資料

```

```
string data =usb->readGJB_UHF_USB("H", 0, 12, "E", "33334444");
```

```
//UHF GJB 刪除標籤
```

```
usb ->killGJB_UHF_USB ("55556666"); //帶入刪除密碼，刪除標籤
```

```
usb ->printlabel_USB ("1", "1");
```

```
usb ->closeport_USB ();
```

```
usb->getDLLVersion_USB(1);
```

```
//指定網口傳輸介面
```

```
SocketConnect socketconnect = gcnew SocketConnect();
```

```
socketconnect ->openport_Ethernet("xxx.xxx.xx.xxx", 9100);
```

```
socketconnect->setup_Ethernet ("54", "30", "2", "3", "0", "3", "0");
```

```
socketconnect->sendcommand_Ethernet ("DIRECTION 1");
```

```
socketconnect->clearbuffer_Ethernet ();
```

```
socketconnect->barcode_Ethernet ("30", "30", "128", "100", "1", "0", "2", "2",
```

```
"barcode1234567");
```

```
socketconnect->qrcode_Ethernet ("220", "260", "M", "3", "A", "0", "AABCB03abcN123");
```

```
socketconnect ->downloadpcx_Ethernet (_T("../VC_SampleCode/UL.PCX "), "UL.PCX");
```

```
socketconnect ->sendcommand_Ethernet ("PUTPCX 50,10,\"UL.PCX\");
```

```
socketconnect ->downloadbmp_Ethernet (_T("../VC_SampleCode/ CIRCLE.BMP "),
```

```
"CIRCLE.BMP");
```

```
socketconnect ->sendcommand_Ethernet ("PUTBMP 150,30,\"CIRCLE.BMP\");
```

```
socketconnect->windowsfont_Ethernet (10, 100, 48, 0, 0, 0, "arial", "VC WIN TEST");
```

```
socketconnect ->qrcode_Ethernet("220", "260", "M", "3", "A", "0", "AABCB03abcN123");
```

```
socketconnect ->printerfontblock_Ethernet ("35", "15", "790", "90", "1", "0", "8", "8", "0", "1",
```

```
ttString);
```

```
socketconnect->printlabel_Ethernet ("1", "1");
```

```
string status = socketconnect->printerstatus_Ethernet ();
```

```
//簡中打印
```

```
string stString="默认简体中文测试";
```

```
socketconnect->clearbuffer_Ethernet ();
```

```
socketconnect->printerfont_Ethernet ("100", "10", "TSS24.BF2", "0", "1", "1", stString);
```

```
socketconnect->printlabel_Ethernet (1, 1, this);
```

```
//繁中打印
```

```
string ttString="默認繁體中文測試";
```

```
socketconnect->clearbuffer_Ethernet ();
```

```
socketconnect->printerfont_Ethernet ("100", "10", " TST24.BF2", "0", "1", "1", ttString);
```

```
socketconnect->printlabel_Ethernet (1, 1, this);
```

```
//BLOCK 打印
```

```
string ttString="We stand behind our products with one of the most comprehensive support  
programs in the Auto-ID industry.";
```

```
socketconnect->clearbuffer_Ethernet ();
```

```
socketconnect->printerfontblock_Ethernet ("35","15","790","90","1","0","8","8","0","1",  
ttString);
```

```
socketconnect ->setDirectionAndMirror_Ethernet (1,1);
```

```
socketconnect ->setShift_Ethernet (50);
```

```
socketconnect ->setAfterPrintAction_Ethernet (2);
```

```
socketconnect ->setOffset_Ethernet (20);
```

```
socketconnect ->setCutMode_Ethernet (0,2);
```

```
socketconnect ->printReverse_Ethernet (90,90,128,40);
```

```
socketconnect ->printlabel_Ethernet ("1", "1");
```

```
socketconnect ->sendcommand_Ethernet ("DIRECTION 1");
```

```
socketconnect ->genericDefault_Ethernet ();
```

```

socketconnect ->sensorDefault_Ethernet ();

int BitmapResult = socketconnect ->Bitmap_Ethernet ("0", "0", 640, 484, 1, "octopus.png");

BitmapResult = socketconnect ->compressBitmap_Ethernet ("0", "0", 640, 484, 1,
"octopus.jpg");

socketconnect ->printlabel_Ethernet ("1", "1");

// RFID

socketconnect -> rfidSetupDefault_Ethernet ();

socketconnect ->RFIDAutoCalibration_Ethernet ();

//UHF GEN2

socketconnect->writeUHF_Ethernet("H", 2, 12,"E","414142424343444445454646");

//startBlockNo : Gen2 預設為 2

socketconnect->printlabel_Ethernet ("1", "1");

string data = socketconnect->readUHF_Ethernet ("H", 0, 12, "E");

string data_Q = socketconnect->query_UHF_Ethernet ( "A", 0, 0);           //以 Q 指令讀取 EPC
資料

socketconnect->EPCPWD_Action_Ethernet ("L","12345678");

socketconnect->TIDPWD_Action_Ethernet ("L", "12345678");

socketconnect->USERPWD_Action_Ethernet ("L", "12345678");

socketconnect->AccessPWD_Action_Ethernet ("U", "12345678");

socketconnect->KillPWD_Action_Ethernet ("U", "12345678");

socketconnect->Set_RFIDPorcedure_Ethernet (8,8,32,3,"N",2,2);

socketconnect->Set_RFIDPorcedure_mm_Ethernet(5, 40, 32, 5, "N", 5, 5, "203");

socketconnect->writeHF_Ethernet ("H", 0, 12, "414142424343444445454646");

socketconnect->printlabel_Ethernet ("1", "1");

//UHF GJB 寫入資料

socketconnect->writeGJB_UHF_Ethernet ("H", 1, 12, "E", "414142424343444445454646",

```

"12345678");

*帶入寫入密碼，寫入資料(startBlockNo: GJB 預設為 1)

//UHF GJB 設定各密碼區域新密碼

socketconnect->setPWD_Action_Ethernet("S", "S", "11112222", "12345678"); //帶入寫入密碼，
設定新狀態密碼

socketconnect->setPWD_Action_Ethernet ("R", "S", "33334444", "12345678");//帶入寫入密碼，
設定新讀取密碼

socketconnect->setPWD_Action_Ethernet ("K", "S", "55556666", "12345678"); //帶入寫入密碼，
設定新刪除密碼

socketconnect->setPWD_Action_Ethernet("W", "S", "87654321", "12345678"); //帶入舊寫入密碼，
設定新寫入密碼

//UHF GJB 設定資料區塊狀態

socketconnect->statusGJB_UHF_Ethernet ("E", "B", "11112222"); //帶入狀態密碼，設定狀態

socketconnect->printlabel_Ethernet ("1", "1");

//UHF GJB 讀取資料

string data = socketconnect->readGJB_UHF_Ethernet ("H", 0, 12, "E", "33334444"); //帶入讀取
密碼，讀取標籤資料

//UHF GJB 刪除標籤

socketconnect->killGJB_UHF_Ethernet ("55556666"); //帶入刪除密碼，刪除標籤

socketconnect->printlabel_Ethernet ("1", "1");

socketconnect->closeport_Ethernet ();

socketconnect->getDLLVersion_Ethernet (1);

1.使用管理員身分開啟命令提示字元

2.註冊 dll

將 x86 的 dll 放到 Windows/Syswow64 , x64 的 dll 放在 Windows/System32

進入 microsoft.net framework 的安裝路徑下

如：cd C:\Windows\Microsoft.NET\Framework64\v4.0.30319

*x64 的 dll 註冊在 C:\Windows\Microsoft.NET\Framework64\v4.0.30319

*x86 的 dll 註冊在 C:\Windows\Microsoft.NET\Framework\v4.0.30319

輸入指令 regasm.exe /register /tlb <dll path>

輸入指令 regasm.exe <dll path> /codebase

如：regasm.exe /register /tlb C:\Users\User\Desktop\Javascript_Sample_Code\GTSPL_SDK.dll

regasm.exe C:\Users\User\source\repos\product\GTSPL_SDK \GTSPL_SDK.dll /codebase

3.範例程式：

LOCAL driver as GTSPL_SDK.Driver

LOCAL filePath

LOCAL printerStatus

filePath= JUSTPATH(SYS(16,0))

driver =CREATEOBJECT("GTSPL_SDK.Driver")

driver.openport("Printer Model")

driver.setup("54", "40", "2", "3", "0", "3", "0")

driver.sendcommand("DIRECTION 1")

driver.clearbuffer()

driver.barcode("30", "2", "128", "50", "1", "0", "2", "2", "barcode Foxpro Driver")

driver.printerfont("30", "75", "3", "0", "1", "1", "Print Font Foxpro Driver")

driver.downloadbmp(filePath+"\CIRCLE.BMP", "CIRCLE.BMP")

driver.sendcommand("PUTBMP 200,150,"+CHR(34)+"CIRCLE.BMP"+CHR(34))

```
driver.downloadbmp(filePath+"\\impact.ttf", "impact.ttf")  
driver.windowfont(10, 100, 36, 0, 0, 0, "impact", "Foxpro Driver")  
driver.printlabel("1", "1")  
driver.getDLLVersion(0)  
driver.closeport()
```

● Python

1. 為調用 dll，請先安裝 pythonnet

安裝命令 `pip install pythonnet` 或 `python -m pip install pythonnet`

2. 在程式中導入 clr

```
import clr
```

3. 加入 GTSPL_SDK：

```
clr.AddReference("GTSPL_SDK") # C# DLL
```

```
from GTSPL_SDK import USB # 從 DLL 中導入 USB Class
```

```
from GTSPL_SDK import Driver # 從 DLL 中導入 Driver Class
```

```
from GTSPL_SDK import SocketConnect # 從 DLL 中導入 Socket Class
```

4. 範例程式：

```
//單機列印
```

```
driver=Driver()
```

```
dresult=driver.openport("Printrer Model")
```

```
print(dresult)
```

```
driver.clearbuffer()
```

```
driver.setup("54", "30", "2", "3", "0", "3", "0")
```

```
driver.setDirectionAndMirror(1,1)
```

```
driver.setShift(50)
```

```
driver.setAfterPrintAction(2)
```

```
driver.setOffset(20)
```

```
driver.setCutMode(0,2)
```

```
driver.printReverse(90,90,128,40)
```

```
driver.barcode("30","30", "128", "100", "1", "0", "2","2", "barcode1234567")
```

```
driver.qrcode("220", "260", "M", "3", "A", "0", "AABCB03abcN123")
```

```
driver.printerfont("50", "10", "3", "0", "1", "1", "Print Font 123456")
```


ttString="We stand behind our products with one of the most comprehensive support programs in the
Auto-ID industry."

```
driver.printerfontblock("35","15","500","90","1","0","2","2","0","1", ttString)
```

```
driver.downloadpcx("UL.PCX", "UL.PCX")
```

```
driver.sendcommand("PUTPCX 150,30,\"UL.PCX\\")
```

```
driver.downloadbmp("CIRCLE.BMP", "CIRCLE.BMP")
```

```
ex:D:\\VS_Project\\python_sample_code\\UL.PCX
```

```
driver.sendcommand("PUTBMP 150,30,\"CIRCLE.BMP\\")
```

```
driver.windowfont(10, 100, 48, 0, 0, 0, "arial", "C# WIN TEST")
```

```
BitmapResult = driver.Bitmap("500", "70", 300, 250, 1, "test.png")
```

```
BitmapResult=driver.compressBitmap("300", "70", 400, 350, "test.jpg")
```

```
stString="默认简体中文测试"
```

```
driver.printerfont("100", "10", "TSS24.BF2", "0", "1", "1", stString)
```

```
driver.formfeed()
```

```
driver.printlabel("1", "1")
```

```
driver.closeport()
```

//USB 列印

```
usb = USB()
```

```
result=usb.openport_USB()
```

```
def check_connection(result):
```

```
    if result!=1:
```

```
        return "fail to connect"
```

```
    return "connect successfully"
```

```
connection =check_connection(result)
```

```
print(connection)
```

```

usb.clearbuffer_USB()

usb.setup_USB("54", "30", "2", "3", "0", "3", "0")

usb.setDirectionAndMirror_USB(0,0)

usb.setShift_USB(50)

usb.setAfterPrintAction_USB(2)

usb.setOffset_USB(20)

usb.setCutMode_USB(0,2)

usb.printReverse_USB(90,90,128,40)

usb.barcode_USB("30","30", "128", "100", "1", "0", "2","2", "barcode1234567")

usb.qrcode_USB("220", "260", "M", "3", "A", "0", "AABCB03abcN123")

usb.printerfont_USB("50", "10", "3", "0", "1", "1", "Print Font 123456")

ttString="We stand behind our products with one of the most comprehensive support programs in
the Auto-ID industry."

usb.printerfontblock_USB("35","15","500","90","1","0","2","2","0","1", ttString)

usb.downloadpcx_USB("UL.PCX", "UL.PCX")

usb.sendcommand_USB("PUTPCX 150,30,\"UL.PCX\"")

usb.downloadbmp_USB("CIRCLE.BMP", "CIRCLE.BMP")

ex:D:\\VS_Project\\python_sample_code\\UL.PCX

usb.sendcommand_USB("PUTBMP 150,30,\"CIRCLE.BMP\"")

usb.windowfont_USB(10, 100, 48, 0, 0, 0, "arial", "C# WIN TEST")

BitmapResult = usb.Bitmap_USB("500", "70", 300, 250, 1, "test.png")

BitmapResult=usb.compressBitmap_USB("300", "70", 400, 350, "test.jpg")

stString="默认简体中文测试"

usb.printerfont_USB ("100", "10", "TSS24.BF2", "0", "1", "1", stString)

usb.printlabel_USB("1","1")

usb.formfeed_USB()

```

```

status=usb.printerstatus_USB()

print(status)

usb.closeport_USB( )

```

//乙太網路列印

```

socket = SocketConnect()

erresult=socket.openport_Ethernet("192.168.66.137", 9100)

socket.clearbuffer_Ethernet()

socket.setup_Ethernet("54", "30", "2", "3", "0", "3", "0")

socket.setDirectionAndMirror_Ethernet(1,1)

socket.setShift_Ethernet(50)

socket.setAfterPrintAction_Ethernet(2)

socket.setOffset_Ethernet(20)

socket.setCutMode_Ethernet(0,2)

socket.printReverse_Ethernet(90,90,128,40)

socket.barcode_Ethernet("30","30", "128", "100", "1", "0", "2","2", "barcode1234567")

socket.qrcode_Ethernet("220", "260", "M", "3", "A", "0", "AABCB03abcN123")

socket.printerfont_Ethernet("50", "10", "3", "0", "1", "1", "Print Font 123456")

ttString="We stand behind our products with one of the most comprehensive support programs in the
Auto-ID industry."

socket.printerfontblock_Ethernet("35","15","500","90","1","0","2","2","0","1", ttString)

socket.downloadpcx_Ethernet("UL.PCX", "UL.PCX")

socket.sendcommand_Ethernet("PUTPCX 150,30,\"UL.PCX\"")

socket.downloadbmp_Ethernet("CIRCLE.BMP", "CIRCLE.BMP")

ex:D:\\VS_Project\\python_sample_code\\UL.PCX

socket.sendcommand_Ethernet("PUTBMP 150,30,\"CIRCLE.BMP\"")

```

```
socket.windowfont_Ethernet(10, 100, 48, 0, 0, 0, "arial", "C# WIN TEST")  
  
BitmapResult = socket.Bitmap_Ethernet("500", "70", 300, 250, 1, "test.png")  
  
BitmapResult=socket.compressBitmap_Ethernet("300", "70", 400, 350, "test.jpg")  
  
stString="默认简体中文测试"  
  
socket.printerfont_Ethernet("100", "10", "TSS24.BF2", "0", "1", "1", stString)  
  
socket.formfeed_Ethernet()  
  
status=socket.printerstatus_Ethernet()  
  
print(status )  
  
socket.printlabel_Ethernet("1", "1")  
  
socket.closeport_Ethernet()
```

● VB6

1. 使用系統管理員開啟命令提示字元
2. 進入 Windows 的 Microsoft.NET Framework 目錄 (系統預設為下列位置，若有變更路徑請自行調整)

`cd C:\Windows\Microsoft.NET\Framework\v4.0.30319`

3. 註冊 dll

`RegAsm.exe` dll 的絕對路徑 `/tlb:dll 檔名.tlb /codebase`

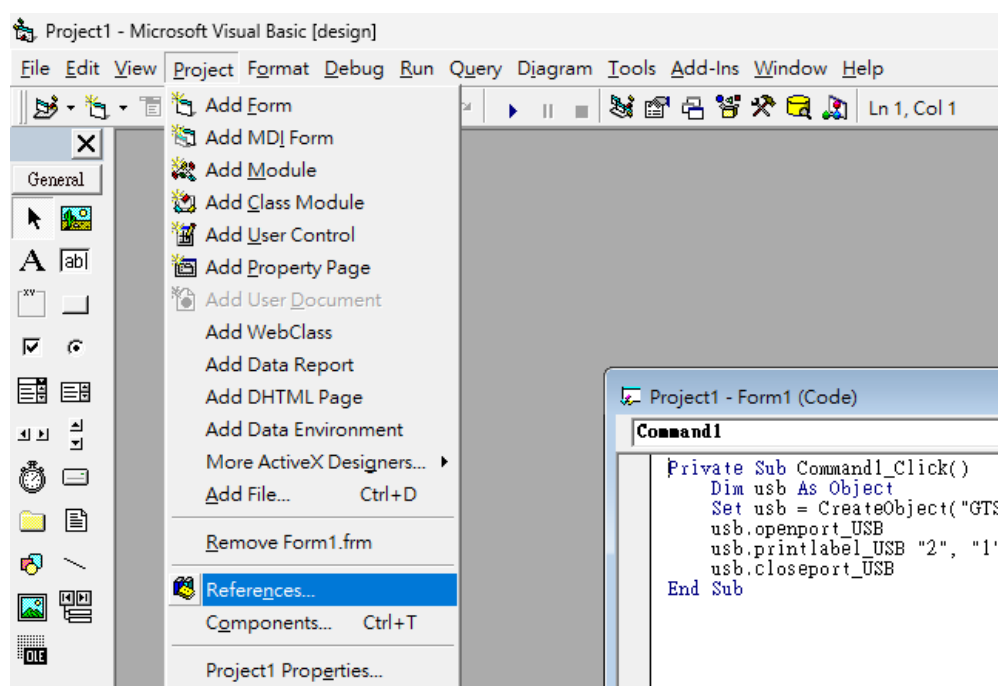
命令範例：

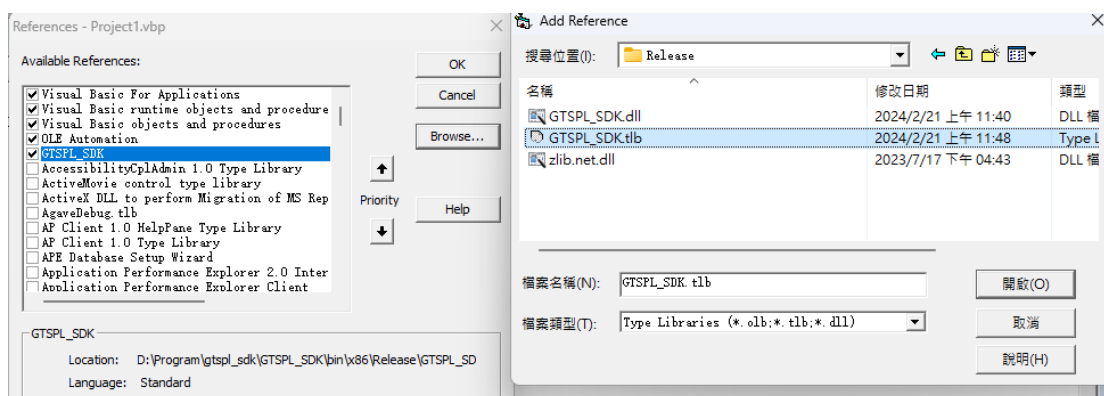
`RegAsm.exe D:\Program\gtspl_sdk\GTSP_SDK\bin\x86\Release\GTSP_SDK.dll /tlb: GTSP_SDK.tlb /codebase`

```
C:\Windows\System32>cd C:\Windows\Microsoft.NET\Framework\v4.0.30319
C:\Windows\Microsoft.NET\Framework\v4.0.30319>RegAsm.exe D:\Program\gtspl_sdk\GTSP_SDK\bin\x86\Release\GTSP_SDK.dll /tlb:GTSP_SDK.tlb /codebase
Microsoft .NET Framework Assembly Registration Utility version 4.8.9032.0
for Microsoft .NET Framework version 4.8.9032.0
Copyright (C) Microsoft Corporation. All rights reserved.

RegAsm : warning RA0000 : Registering an unsigned assembly with /codebase can cause your assembly to interfere with other applications that may be installed on the same computer. The /codebase switch is intended to be used only with signed assemblies. Please give your assembly a strong name and re-register it.
Types registered successfully
Type library exporter warning processing 'GTSP_SDK.Driver+DOCINFOA, GTSP_SDK'. Warning: The reference type had sequential or explicit layout, and so was exported as a struct.
Assembly exported to 'D:\Program\gtspl_sdk\GTSP_SDK\bin\x86\Release\GTSP_SDK.tlb', and the type library was registered successfully
```

4. 加入參考：
- VB 開啟後，Project>References...裡面加入 SDK(選擇 tlb)





5. 範例程式：

//單機列印

Dim driver. As Object

Dim dresult As Integer

Dim ttString As String

Dim BitmapResult As Integer

Dim stString As String

Dim VerString As String

Set driver = CreateObject("GTSPL_SDK.Driver")

dresult =driver.openport("Printrer Model")

Print dresult

driver.clearbuffer

driver.setup "54", "30", "2", "3", "0", "3", "0"

driver.setDirectionAndMirror 1,1

driver.setShift 50

driver.setAfterPrintAction 2

driver.setOffset 20

driver.setCutMode 0,2

driver.printReverse 90,90,128,40

driver.barcode "30","30", "128", "100", "1", "0", "2","2", "barcode1234567"

```

driver.qrcode "220", "260", "M", "3", "A", "0", "AABCB03abcN123"

driver.printerfont "50", "10", "3", "0", "1", "1", "Print Font 123456"

ttString="We stand behind our products with one of the most comprehensive support programs in the
Auto-ID industry."

driver.printerfontblock "35", "15", "500", "90", "1", "0", "2", "2", "0", "1", ttString

driver.downloadpcx "[UL.PCX 的絕對路徑]", "UL.PCX"

driver.sendcommand "PUTPCX 150,30,\"UL.PCX\""

driver.downloadbmp "[CIRCLE.BMP 的绝对路径]", "CIRCLE.BMP"

ex: D:\Microsoft Visual Studio\VB98\Project\UL.PCX

driver.sendcommand "PUTBMP 150,30,\"CIRCLE.BMP\""

driver.windowfont 10, 100, 48, 0, 0, 0, "arial", "C# WIN TEST"

BitmapResult = driver.Bitmap("500", "70", 300, 250, 1, "[test.png 的絕對路徑]")

BitmapResult=driver.compressBitmap("300", "70", 400, 350, "[test.jpg 的絕對路徑]")

stString="默认简体中文测试"

driver.printerfont "100", "10", "TSS24.BF2", "0", "1", "1", stString

driver.formfeed

driver.printlabel "1", "1"

VerString =driver.getDLLVersion(0)

driver.closeport

```

//USB 列印

```

Dim usb As Object

Dim ttString As String

Dim BitmapResult As Integer

Dim stString As String

Dim Status As String

```

Dim VerString As String

Set usb = CreateObject("GTSPL_SDK.USB")

usb.openport_USB

usb.clearbuffer_USB

usb.setup_USB "54", "30", "2", "3", "0", "3", "0"

usb.setDirectionAndMirror_USB 0,0

usb.setShift_USB 50

usb.setAfterPrintAction_USB 2

usb.setOffset_USB 20

usb.setCutMode_USB 0,2

usb.printReverse_USB 90,90,128,40

usb.barcode_USB "30","30", "128", "100", "1", "0", "2","2", "barcode1234567"

usb.qrcode_USB "220", "260", "M", "3", "A", "0", "AABCB03abcN123"

usb.printerfont_USB "50", "10", "3", "0", "1", "1", "Print Font 123456"

ttString="We stand behind our products with one of the most comprehensive support programs in the Auto-ID industry."

usb.printerfontblock_USB "35","15", "500", "90", "1", "0", "2", "2", "0", "1", ttString

usb.downloadpcx_USB "[UL.PCX 的絕對路徑]", "UL.PCX"

usb.sendcommand_USB "PUTPCX 150,30,\"UL.PCX\""

usb.downloadbmp_USB "[CIRCLE.BMP 的絕對路徑]", "CIRCLE.BMP"

ex: D:\Microsoft Visual Studio\VB98\Project\UL.PCX

usb.sendcommand_USB "PUTBMP 150,30,\"CIRCLE.BMP\""

usb.windowfont_USB 10, 100, 48, 0, 0, 0, "arial", "C# WIN TEST"

BitmapResult = usb.Bitmap_USB("500", "70", 300, 250, 1, "[test.png 的絕對路徑]")

BitmapResult=usb.compressBitmap_USB("300", "70", 400, 350, "[test.jpg 的絕對路徑]")

stString="默认简体中文测试"


```

usb.printerfont_USB "100", "10", "TSS24.BF2", "0", "1", "1", stString
usb.printlabel_USB "1", "1"
usb.formfeed_USB
status=usb.printerstatus_USB()
Print status
VerString =usb.getDLLVersion_USB(0)
usb.closeport_USB

```

//乙太網路列印

```

Dim socket As Object
Dim ttString As String
Dim BitmapResult As Integer
Dim stString As String
Dim Status As String
Dim VerString As String
Set socket = CreateObject("GTSPL_SDK.SocketConnect")
socket.openport_Ethernet("192.168.66.137", 9100)
socket.clearbuffer_Ethernet
socket.setup_Ethernet "54", "30", "2", "3", "0", "3", "0"
socket.setDirectionAndMirror_Ethernet 1,1
socket.setShift_Ethernet 50
socket.setAfterPrintAction_Ethernet 2
socket.setOffset_Ethernet 20
socket.setCutMode_Ethernet 0,2
socket.printReverse_Ethernet 90,90,128,40
socket.barcode_Ethernet "30","30","128","100","1","0","2","2","barcode1234567"

```

socket.qrcode_Ethernet "220", "260", "M", "3", "A", "0", "AABCB03abcN123"

socket.printerfont_Ethernet "50", "10", "3", "0", "1", "1", "Print Font 123456"

ttString="We stand behind our products with one of the most comprehensive support programs in the Auto-ID industry."

socket.printerfontblock_Ethernet "35", "15", "500", "90", "1", "0", "2", "2", "0", "1", ttString

socket.downloadpcx_Ethernet "[UL.PCX 的絕對路徑]", "UL.PCX"

socket.sendcommand_Ethernet "PUTPCX 150,30,\"UL.PCX\""

socket.downloadbmp_Ethernet "[CIRCLE.BMP 的絕對路徑]", "CIRCLE.BMP"

ex: D:\Microsoft Visual Studio\VB98\Project\UL.PCX

socket.sendcommand_Ethernet "PUTBMP 150,30,\"CIRCLE.BMP\""

socket.windowfont_Ethernet 10, 100, 48, 0, 0, 0, "arial", "C# WIN TEST"

BitmapResult = socket.Bitmap_Ethernet "500", "70", 300, 250, 1, "[test.png 的絕對路徑]"

BitmapResult=socket.compressBitmap_Ethernet "300", "70", 400, 350, "[test.jpg 的絕對路徑]"

stString="默认简体中文测试"

socket.printerfont_Ethernet "100", "10", "TSS24.BF2", "0", "1", "1", stString

socket.formfeed_Ethernet

status=socket.printerstatus_Ethernet()

Print status

socket.printlabel_Ethernet "1", "1"

VerString=socket.getDLLVersion_Ethernet(0)

socket.closeport_Ethernet

● PHP

1. 使用系統管理員開啟命令提示字元
2. 進入 Windows 的 Microsoft.NET Framework 目錄 (系統預設為下列位置，若有變更路徑請自行調整)

64 位元：

`cd C:\Windows\Microsoft.NET\Framework64\v4.0.30319`

32 位元：

`cd C:\Windows\Microsoft.NET\Framework64\v4.0.30319`

3. 註冊 dll

`regasm /codebase dll` 的絕對路徑

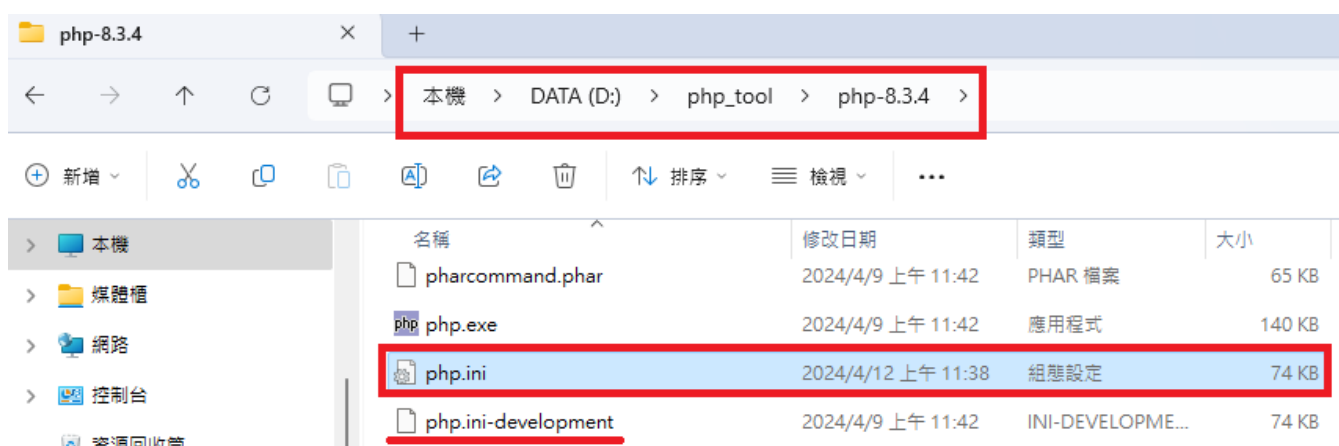
命令範例：

`regasm /codebase D:\Program\gtspl_sdk\GTSPL_SDK\bin\x64\Release\GTSPL_SDK.dll`

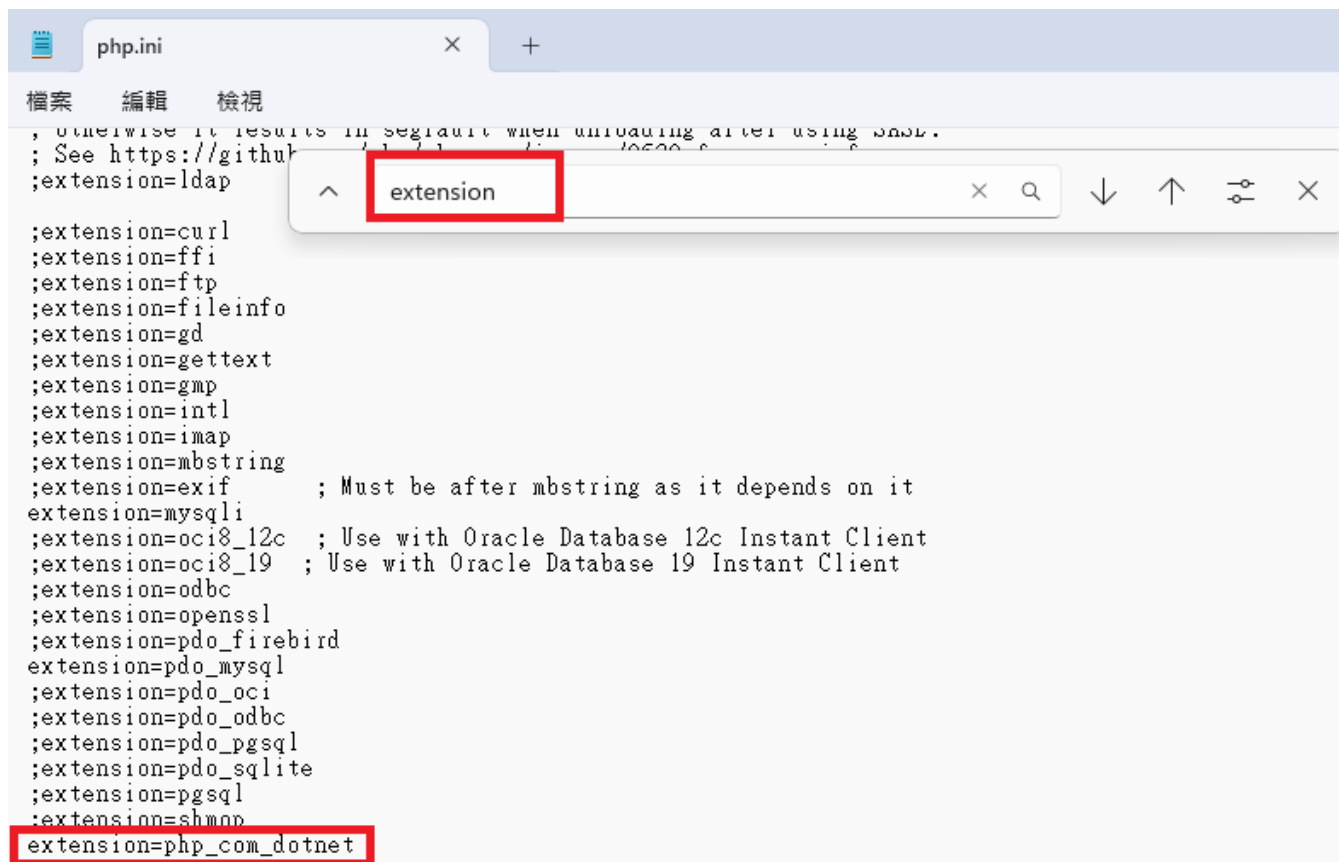
```
C:\Windows\System32>cd C:\Windows\Microsoft.NET\Framework64\v4.0.30319
C:\Windows\Microsoft.NET\Framework64\v4.0.30319>regasm /codebase D:\Program\gtspl_sdk\GTSPL_SDK\bin\x64\Release\GTSPL_SDK.dll
Microsoft .NET Framework Assembly Registration Utility 版本 4.8.9032.0
for Microsoft .NET Framework 版本 4.8.9032.0
Copyright (C) Microsoft Corporation. 著作權所有，並保留一切權利。
RegAsm : warning RA0000 : 使用 /codebase 選項註冊一個 unsigned 組件，您的組件可能會和安裝在同一部電腦上的其他應用程式相衝突。/codebase 選項通常只在已簽署的組件。請以強式名稱命名您的組件並重新註冊。
已成功註冊類型
C:\Windows\Microsoft.NET\Framework64\v4.0.30319>
```

4. 啟用 com 類別

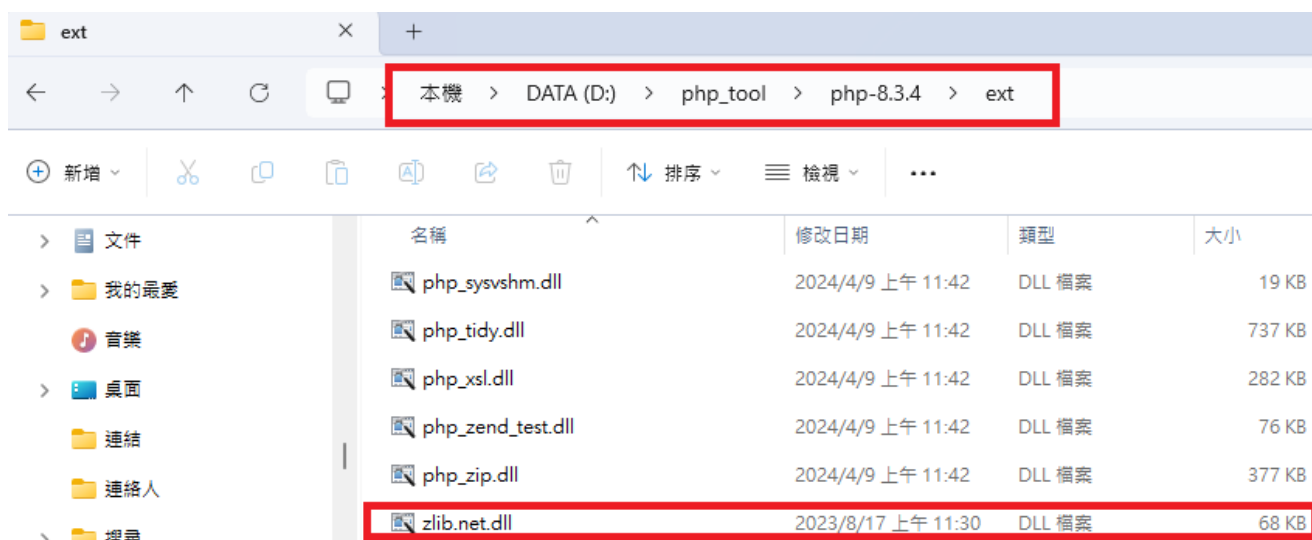
到 PHP 的資料夾尋找 php.ini(若沒有，將 php.ini-development 備份，再將其改為 php.ini 即可)
範例如下：



開啟 php.ini 後，搜尋"extension="，在一堆"extension="的最後一行新增
"extension=php_com_dotnet"，若文件本身已經有，則不用再新增，若前面有分號則刪掉分號儲存即可。



※若需使用 **compressBitmap**，則需要到以下路徑，放進已提供的 **zlib.net.dll**，才可正常使用。



5. 範例程式：

//單機列印

```
$dll = new Com("GTSPL_SDK.Driver");
```

```

$dll->openport("Printrer Model");

$dll->clearbuffer();

$dll ->setup("54", "30", "2", "3", "0", "3", "0");

$dll->setDirectionAndMirror(1,1);

$dll->setShift(50);

$dll->setAfterPrintAction(2);

$dll->setOffset(20);

$dll->setCutMode(0,2);

$dll->printReverse(90,90,128,40);

$dll->barcode("30","30", "128", "100", "1", "0", "2","2", "barcode1234567");

$dll->qrcode("220", "260", "M", "3", "A", "0", "AABCB03abcN123");

$dll->printerfont("50", "10", "3", "0", "1", "1", "Print Font 123456");

$ttString="We stand behind our products with one of the most comprehensive support programs in
the Auto-ID industry.";

$dll->printerfontblock("35","15","500","90","1","0","2","2","0","1", $ttString);

$path="C:\Users\sw_user1\Desktop\TEMP\TEST_PIC\UL.PCX ";

$dll->downloadpcx($path, "UL.PCX");

$dll->sendcommand("PUTPCX 150,30,\"UL.PCX\");

$path1="C:\Users\sw_user1\Desktop\TEMP\TEST_PIC\ CIRCLE.BMP ";

$dll->downloadbmp($path1, "CIRCLE.BMP") ;

$dll->$dll->sendcommand("PUTBMP 150,30,\"CIRCLE.BMP\");

$dll->windowsfont(10, 100, 48, 0, 0, 0, "arial", "C# WIN TEST");

$path2="C:\Users\sw_user1\Desktop\TEMP\TEST_PIC\ test.png ";

$BitmapResult = $dll->Bitmap("500", "70", 300, 250, 1, $path2);

$path3="C:\Users\sw_user1\Desktop\TEMP\TEST_PIC\ test.jpg ";

$BitmapResult1=$dll->compressBitmap("300", "70", 400, 350, $path3) ;

```

```

$stString="默认简体中文测试";

$dll->printerfont("100", "10", "TSS24.BF2", "0", "1", "1", $stString);

$dll->formfeed();

$dll->printlabel("1", "1");

$verString=$dll-> getDLLVersion(1);

$dll->closeport();

```

//USB 列印

```

$dll = new Com("GTSPL_SDK.USB");

$dll->usb.openport_USB();

$dll->clearbuffer_USB();

$dll->setup_USB("54", "30", "2", "3", "0", "3", "0");

$dll->setDirectionAndMirror_USB(0,0);

$dll->setShift_USB(50);

$dll->setAfterPrintAction_USB(2);

$dll->setOffset_USB(20);

$dll->setCutMode_USB(0,2);

$dll->printReverse_USB(90,90,128,40);

$dll->barcode_USB("30","30", "128", "100", "1", "0", "2","2", "barcode1234567");

$dll->qrcode_USB("220", "260", "M", "3", "A", "0", "AABCB03abcN123");

$dll->printerfont_USB("50", "10", "3", "0", "1", "1", "Print Font 123456");

$ttString="We stand behind our products with one of the most comprehensive support programs in
the Auto-ID industry.";

$dll->printerfontblock_USB("35","15","500","90","1","0","2","2","0","1", $ttString);

$path="C:\Users\sw_user1\Desktop\TEMP\TEST_PIC\UL.PCX ";

$dll->downloadpcx_USB($path, "UL.PCX");

```

```

$dll->sendcommand_USB("PUTPCX 150,30,\"UL.PCX\"");

$path1="C:\\Users\\sw_user1\\Desktop\\TEMP\\TEST_PIC\\ CIRCLE.BMP ";

$dll->downloadbmp_USB($path1, "CIRCLE.BMP") ;

$dll->sendcommand_USB("PUTBMP 150,30,\"CIRCLE.BMP\"");

$dll->windowsfont_USB(10, 100, 48, 0, 0, 0, "arial", "C# WIN TEST");

$path2="C:\\Users\\sw_user1\\Desktop\\TEMP\\TEST_PIC\\ test.png ";

$BitmapResult = $dll->Bitmap_USB("500", "70", 300, 250, 1, $path2);

$path3="C:\\Users\\sw_user1\\Desktop\\TEMP\\TEST_PIC\\ test.jpg ";

$BitmapResult1=$dll->compressBitmap_USB("300", "70", 400, 350, $path3);

$stString="默认简体中文测试";

$dll->printerfont_USB ("100", "10", "TSS24.BF2", "0", "1", "1",$stString);

$dll->printlabel_USB("1","1");

$dll->formfeed_USB();

$status=$dll->usb.printerstatus_USB();

echo $status;

$VerString=$dll-> getDLLVersion_USB(1);

$dll->closeport_USB( );

```

//乙太網路列印

```

socket = SocketConnect();

$dll->openport_Ethernet("192.168.66.137", 9100);

$dll->clearbuffer_Ethernet();

$dll->setup_Ethernet("54", "30", "2", "3", "0", "3", "0");

$dll->setDirectionAndMirror_Ethernet(1,1);

$dll->setShift_Ethernet(50);

$dll->setAfterPrintAction_Ethernet(2);

```

```

$dll->setOffset_Ethernet(20);

$dll->setCutMode_Ethernet(0,2);

$dll->$dll->printReverse_Ethernet(90,90,128,40);

$dll->barcode_Ethernet("30","30","128","100","1","0","2","2","barcode1234567");

$dll->qrcode_Ethernet("220","260","M","3","A","0","AABCB03abcN123");

$dll->printerfont_Ethernet("50","10","3","0","1","1","Print Font 123456");

$ttString="We stand behind our products with one of the most comprehensive support programs in
the Auto-ID industry.";

$dll->printerfontblock_Ethernet("35","15","500","90","1","0","2","2","0","1",$ttString);

$path="C:\Users\sw_user1\Desktop\TEMP\TEST_PIC\UL.PCX ";

$dll->downloadpcx_Ethernet($path, "UL.PCX");

$dll->sendcommand_Ethernet("PUTPCX 150,30,\"UL.PCX\"");

$path1="C:\Users\sw_user1\Desktop\TEMP\TEST_PIC\ CIRCLE.BMP ";

$dll->downloadbmp_Ethernet($path1, "CIRCLE.BMP") ;

$dll->sendcommand_Ethernet("PUTBMP 150,30,\"CIRCLE.BMP\"");

$dll->windowsfont_Ethernet(10, 100, 48, 0, 0, 0, "arial", "C# WIN TEST");

$path2="C:\Users\sw_user1\Desktop\TEMP\TEST_PIC\ test.png ";

$BitmapResult = $dll->Bitmap_Ethernet("500", "70", 300, 250, 1, $path2);

$path3="C:\Users\sw_user1\Desktop\TEMP\TEST_PIC\ test.jpg ";

$BitmapResult1=$dll->compressBitmap_Ethernet("300", "70", 400, 350, $path3);

$stString="默认简体中文测试";

$dll->printerfont_Ethernet("100", "10", "TSS24.BF2", "0", "1", "1", $stString);

$dll->formfeed_Ethernet();

$status=$dll->printerstatus_Ethernet();

echo $status;

$dll->printlabel_Ethernet("1", "1");

```

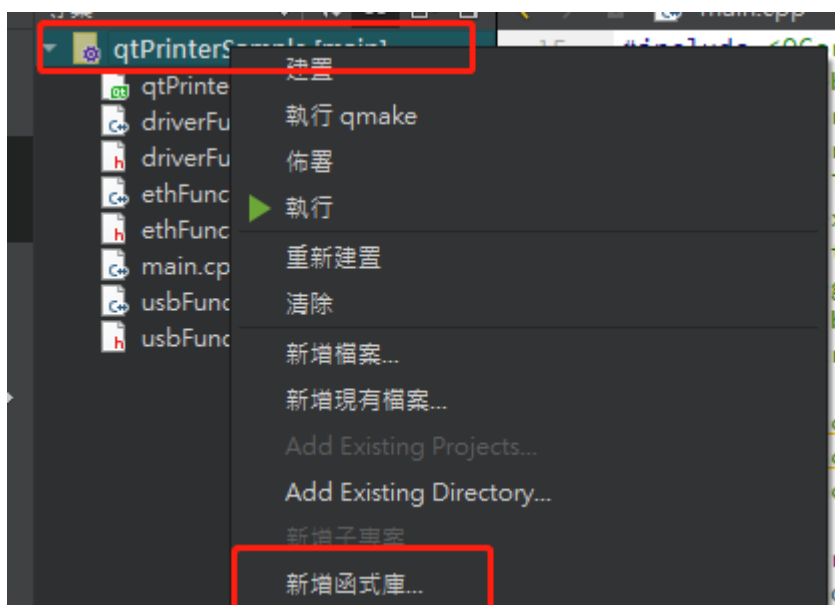


```
$VerString =$dll-> getDLLVersion_Ethernet (1);
```

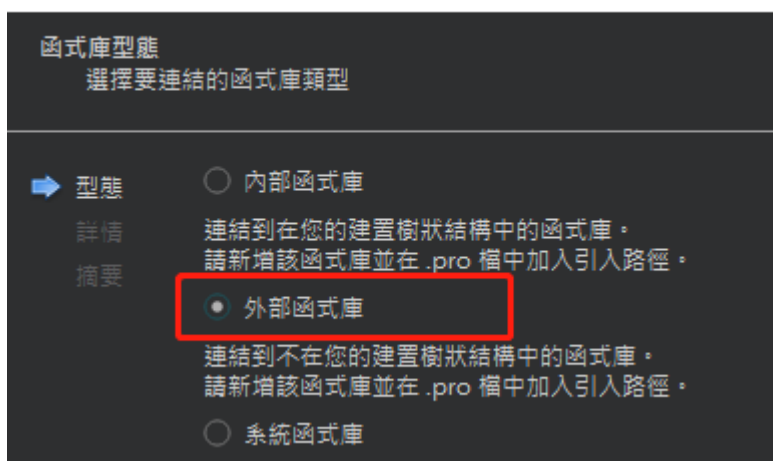
```
$dll->closeport_Ethernet();
```

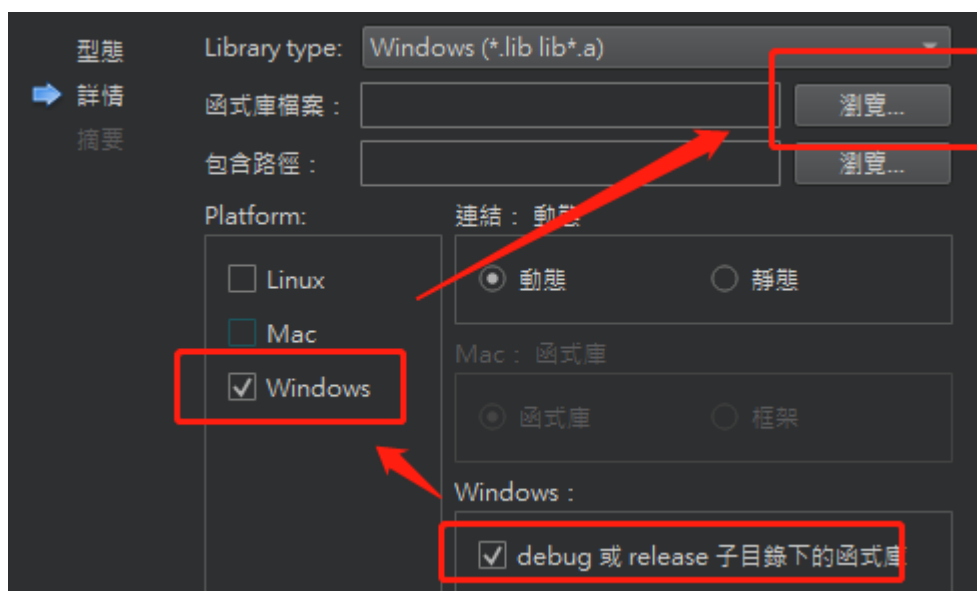
● Qt C++

1. 請確保專案使用 Desktop Qt MSVC2019 64bit 運行
2. 按照步驟導入 lib (資料夾下需放置.lib 與.dll 檔)



QC 新增函式庫 - Qt Creator





qt5-Desktop_Qt_5_15_2_MSVC2019_64bit-Debug

名稱	信息
GTSP_L_SDK_C.lib	2

2. 先透過 `typedef` 定義函式指標，再透過 `HMODULE (LoadLibrary)` 與 `GetProcAddress` 來取得函式指針並調用。

//引入 dll

```
HMODULE hModule = LoadLibrary(L"GTSP_L_SDK_C.dll");
if (!hModule)
{
    qDebug() << "Failed to load CLR DLL";
    return 1;
}
```

單機列印範例

////===== 定義函式指標=====

```
typedef void (*sendcommand)(char* command);
typedef int (*openport)(char* PrinterName);
typedef void (*closeport)();
```

```
typedef void (*setup)(char*, char*, char*, char*, char*, char*, char*);
```

```

typedef void (*barcode)(char*, char*, char*, char* , char* , char* , char* , char* , char* );
typedef void (*printLabel)(char*,char*);
typedef void (*printerfont)(char*, char*, char* , char* , char* , char* , char* );
typedef void (*printerfontUnicode)(char* , char* , char* , char* , char* , char* , wchar_t* );
typedef void (*clearbuffer)();
typedef void (*downloadpcx)(char*,char*);
typedef void (*downloadbmp)(char* , char* );
typedef void (*windowsfont)(int , int , int , int , int , int , char* , char* );
typedef void (*printerfontblock)(char* , char* , char* , char* , char* , char* , char* , char* , char* , char* , char* , char* );
typedef void (*qrcode)(char* , char* , char* , char* , char* , char* , char* );
typedef void (*setDirectionAndMirror)(int , int );
typedef void (*setShift)(int);
typedef void (*printReverse)(int , int , int , int );
typedef void (*setAfterPrintAction)(int);
typedef void (*setOffset)(double);
typedef void (*setCutMode)(int , int );
typedef void (*genericDefault)();
typedef void (*sensorDefault)();
typedef void (*WifiFrequency)(char*);
typedef int (*Bitmap)(char* , char* , int , int , int , char* );
typedef int (*compressBitmap)(char* , char* , int , int , char* );

```

////===== 動態函式調用撰寫 =====

// 直接發送命令

```

void sendcommandDriver(const std::string& command)
{
    sendcommand sendCommand = (sendcommand)GetProcAddress(g_hModu_Driver, "sendcommand");
    if (!sendCommand) {
        qDebug() << "Failed to resolve openport_Ethernet function from CLR DLL";
        return;
    }

    sendCommand(const_cast<char*>(command.c_str()));
}

```

// 開啟 Driver 通道

```

int openPortDriver(const std::string& printerName)
{
    openport openPort = (openport)GetProcAddress(g_hModu_Driver, "openport");
    if (!openPort) {
        qDebug() << "Failed to resolve openport_USB function from CLR DLL";
        return 0;
    }

    int result = openPort(const_cast<char*>(printerName.c_str()));

    return result;
}

// 關閉 Driver 通道
void closeportDriver(void)
{
    closeport closePort = (closeport)GetProcAddress(g_hModu_Driver, "closeport");
    if (!closePort) {
        qDebug() << "Failed to resolve closeport_USB function from CLR DLL";
        return ;
    }
    closePort();
}

// 發送設定值
void setupDriver(const std::string& width, const std::string& height, const std::string& speed,
                const std::string& density, const std::string& sensor, const std::string& vertical,
                const std::string& offset)
{
    setup setPrinter = (setup)GetProcAddress(g_hModu_Driver, "setup");
    if (!setPrinter) {
        qDebug() << "Failed to resolve openport function from CLR DLL";
        return;
    }

    setPrinter(const_cast<char*>(width.c_str()), const_cast<char*>(height.c_str()),
              const_cast<char*>(speed.c_str()), const_cast<char*>(density.c_str()),
              const_cast<char*>(sensor.c_str()), const_cast<char*>(vertical.c_str()),
              const_cast<char*>(offset.c_str()));
}

```

```
}
```

```
// 列印條碼指令
```

```
void barcode_Driver(const std::string& x, const std::string& y, const std::string& type,
    const std::string& height, const std::string& readable, const std::string& rotation,
    const std::string& narrow, const std::string& wide, const std::string& code)
```

```
{
```

```
    barcode barcodeDriver = (barcode)GetProcAddress(g_hModu_Driver, "barcode");
```

```
    if (!barcodeDriver) {
```

```
        qDebug() << "Failed to resolve barcode function from CLR DLL";
```

```
        return;
```

```
    }
```

```
    barcodeDriver(const_cast<char*>(x.c_str()), const_cast<char*>(y.c_str()),
```

```
    const_cast<char*>(type.c_str()),
```

```
        const_cast<char*>(height.c_str()), const_cast<char*>(readable.c_str()),
```

```
    const_cast<char*>(rotation.c_str()),
```

```
        const_cast<char*>(narrow.c_str()), const_cast<char*>(wide.c_str()),
```

```
    const_cast<char*>(code.c_str()));
```

```
}
```

```
// 發送列印指令
```

```
void printLabelDriver(const std::string& setting, const std::string& copy) {
```

```
    printLabel printLabelDriver = (printLabel)GetProcAddress(g_hModu_Driver, "printlabel");
```

```
    if (!printLabelDriver) {
```

```
        qDebug() << "Failed to resolve printLabelDriver function from CLR DLL";
```

```
        return;
```

```
    }
```

```
    printLabelDriver(const_cast<char*>(setting.c_str()), const_cast<char*>(copy.c_str()));
```

```
}
```

```
// 列印文字
```

```
void printFontDriver(const std::string& x, const std::string& y, const std::string& fontSize,
```

```
    const std::string& rotation, const std::string& xMul, const std::string& yMul,
```

```
    const std::string& text)
```

```

{

printerfont printerFontDriver = (printerfont)GetProcAddress(g_hModu_Driver, "printerfont");
if (!printerFontDriver) {
    qDebug() << "Failed to resolve printerfont function from CLR DLL";
    return;
}

const char* xChar = x.c_str();
const char* yChar = y.c_str();
const char* fontSizeChar = fontSize.c_str();
const char* rotationChar = rotation.c_str();
const char* xMulChar = xMul.c_str();
const char* yMulChar = yMul.c_str();

const char* textChar = text.c_str();

printerFontDriver(const_cast<char*>(xChar), const_cast<char*>(yChar),
const_cast<char*>(fontSizeChar),
                const_cast<char*>(rotationChar), const_cast<char*>(xMulChar),
const_cast<char*>(yMulChar),
                const_cast<char*>(textChar));
}

// 列印 Unicode 格式文字
void printFontUnicodeDriver(const std::string& x, const std::string& y, const std::string& fontSize,
                        const std::string& rotation, const std::string& xMul, const std::string& yMul,
                        const QString& text)
{

    printerfontUnicode printerFontUnicode = (printerfontUnicode)GetProcAddress(g_hModu_Driver,
"printerfontUnicode");
    if (!printerFontUnicode) {
        qDebug() << "Failed to resolve printerfont_USB function from CLR DLL";
        return;
    }
}

```

```

const char* xChar = x.c_str();
const char* yChar = y.c_str();
const char* fontSizeChar = fontSize.c_str();
const char* rotationChar = rotation.c_str();
const char* xMulChar = xMul.c_str();
const char* yMulChar = yMul.c_str();

```

```

QVector<wchar_t> wcharArray(text.length() + 1); // +1 用於終止空字符
text.toWCharArray(wcharArray.data());

```

```

printerFontUnicode(const_cast<char*>(xChar), const_cast<char*>(yChar),
const_cast<char*>(fontSizeChar),
                const_cast<char*>(rotationChar), const_cast<char*>(xMulChar),
const_cast<char*>(yMulChar),
                wcharArray.data());
}

```

// 清除 label 版面

```

void clearBufferDriver(void)
{

```

```

    clearbuffer clearBufferDriver = (clearbuffer)GetProcAddress(g_hModu_Driver, "clearbuffer");
    if (!clearBufferDriver) {
        qDebug() << "Failed to resolve clearbuffer function from CLR DLL";
        return;
    }

```

```

    clearBufferDriver();
}

```

// 下載列印 PCX 圖片

```

void downloadPcxDriver(const std::string& filename, const std::string& imagename) {

```

```

    downloadpcx downloadPcxFunc = (downloadpcx)GetProcAddress(g_hModu_Driver, "downloadpcx");
    if (!downloadPcxFunc) {
        qDebug() << "Failed to resolve downloadpcx function from CLR DLL";
        return;
    }

```



```
downloadPcxFunc(const_cast<char*>(filename.c_str()), const_cast<char*>(imagename.c_str())); // 將
std::string 轉換為 const char*
}
```

// 下載列印 BMP 圖片

```
void downloadBmpDriver(const std::string& filename, const std::string& imagename) {
```

```
downloadbmp downloadBmpFunc = (downloadbmp)GetProcAddress(g_hModu_Driver,
"downloadbmp");
```

```
if (!downloadBmpFunc) {
    qDebug() << "Failed to resolve downloadpcx function from CLR DLL";
    return;
}
```

```
downloadBmpFunc(const_cast<char*>(filename.c_str()), const_cast<char*>(imagename.c_str())); // 將
std::string 轉換為 const char*
}
```

// 使用 font 列印文字

```
void windowsFontDriver(int x, int y, int fontheight, int rotation, int fontstyle, int fontunderline, const
std::string& szFaceName, const std::string& content)
{
```

```
windowsfont windowsfontDriver = (windowsfont)GetProcAddress(g_hModu_Driver, "windowsfont");
if (!windowsfontDriver) {
    qDebug() << "Failed to resolve downloadpcx function from CLR DLL";
    return;
}
```

```
const char* szFaceNameChar = szFaceName.c_str();
const char* contentChar = content.c_str();
windowsfontDriver(x, y, fontheight, rotation, fontstyle, fontunderline,
const_cast<char*>(szFaceNameChar), const_cast<char*>(contentChar));
}
```

// 列印文字塊 (可換行)

```
void printFontblockDriver(const std::string& x, const std::string& y, const std::string& width, const
std::string& height,
```

```
    const std::string& fontname, const std::string& rotation, const std::string& xmul,
    const std::string& ymul, const std::string& space, const std::string& align,
    const std::string& text)
```

```
{
```

```
    printerfontblock printerFontBlockDriver = (printerfontblock)GetProcAddress(g_hModu_Driver,
"printerfontblock");
```

```
    if (!printerFontBlockDriver) {
```

```
        qDebug() << "Failed to resolve downloadpcx function from CLR DLL";
```

```
        return;
```

```
    }
```

```
    const char* xChar = x.c_str();
```

```
    const char* yChar = y.c_str();
```

```
    const char* widthChar = width.c_str();
```

```
    const char* heightChar = height.c_str();
```

```
    const char* fontnameChar = fontname.c_str();
```

```
    const char* rotationChar = rotation.c_str();
```

```
    const char* xmulChar = xmul.c_str();
```

```
    const char* ymulChar = ymul.c_str();
```

```
    const char* spaceChar = space.c_str();
```

```
    const char* alignChar = align.c_str();
```

```
    const char* textChar = text.c_str();
```

```
    printerFontBlockDriver(const_cast<char*>(xChar), const_cast<char*>(yChar),
```

```
const_cast<char*>(widthChar),
```

```
        const_cast<char*>(heightChar), const_cast<char*>(fontnameChar),
```

```
const_cast<char*>(rotationChar),
```

```
        const_cast<char*>(xmulChar), const_cast<char*>(ymulChar),
```

```
const_cast<char*>(spaceChar),
```

```
        const_cast<char*>(alignChar), const_cast<char*>(textChar));
```

```
}
```

```
// 列印 qrcode
```

```
void qrcodeDriver(const std::string& x, const std::string& y, const std::string& ECClevel, const std::string&
cellwidth,
```

```
    const std::string& mode, const std::string& rotation, const std::string& content)
```

```

{

qrcline qrclineDriver = (qrcline)GetProcAddress(g_hModu_Driver, "qrcline");
if (!qrclineDriver) {
    qrclineDebug() << "Failed to resolve qrclineDriver function from CLR DLL";
    return;
}

const char* xChar = x.c_str();
const char* yChar = y.c_str();
const char* ECClevelChar = ECClevel.c_str();
const char* cellwidthChar = cellwidth.c_str();
const char* modeChar = mode.c_str();
const char* rotationChar = rotation.c_str();
const char* contentChar = content.c_str();
qrclineDriver(const_cast<char*>(xChar), const_cast<char*>(yChar), const_cast<char*>(ECClevelChar),
              const_cast<char*>(cellwidthChar), const_cast<char*>(modeChar),
const_cast<char*>(rotationChar),
              const_cast<char*>(contentChar));
}

// 設定打印方向與鏡像
void setDirectionAndMirroDriver(int direction,int mirror)
{
    setDirectionAndMirror setDAM = (setDirectionAndMirror)GetProcAddress(g_hModu_Driver,
"setDirectionAndMirror");

    if (!setDAM) {
        qrclineDebug() << "Failed to resolve qrclineDriver function from CLR DLL";
        return;
    }

    setDAM(direction,mirror);
}

// 設定 shift
void setShiftDriver(int shift)
{

```

```

setShift setShiftDriver = (setShift)GetProcAddress(g_hModu_Driver, "setShift");

if (!setShiftDriver) {
    qDebug() << "Failed to resolve qrCodeDriver function from CLR DLL";
    return;
}

setShiftDriver(shift);
}

// 列印反色區塊
void printReverseDriver(int x_start, int y_start, int x_width, int y_height)
{
    printReverse setReverseDriver = (printReverse)GetProcAddress(g_hModu_Driver, "printReverse");

    if (!setReverseDriver) {
        qDebug() << "Failed to resolve qrCodeDriver function from CLR DLL";
        return;
    }

    setReverseDriver(x_start, y_start, x_width, y_height);
}

// 設定列印後動作
void setAfterPrintModeDriver(int mode)
{
    setAfterPrintAction setAPM
    =(setAfterPrintAction)GetProcAddress(g_hModu_Driver, "setAfterPrintAction");

    if (!setAPM) {
        qDebug() << "Failed to resolve qrCodeDriver function from CLR DLL";
        return;
    }

    setAPM(mode);
}

// 設定 offset
void setOffsetDriver(double offset)

```

```

{
    setOffset setOff =(setOffset)GetProcAddress(g_hModu_Driver,"setOffset");

    if (!setOff) {
        qDebug() << "Failed to resolve qrcodeDriver function from CLR DLL";
        return;
    }

    setOff(offset);
}

// 設定裁切模式
void setCutModeDriver(int mode, int piece)
{
    setCutMode setCut = (setCutMode)GetProcAddress(g_hModu_Driver,"setCutMode");

    if (!setCut) {
        qDebug() << "Failed to resolve qrcodeDriver function from CLR DLL";
        return;
    }

    setCut(mode,piece);
}

// 設定 generic 設定初始化
void genericDefaultDriver(void)
{
    genericDefault setGenericDef = (genericDefault)GetProcAddress(g_hModu_Driver,"genericDefault");

    if (!setGenericDef) {
        qDebug() << "Failed to resolve qrcodeDriver function from CLR DLL";
        return;
    }

    setGenericDef();
}

// 設定 sensor 設定初始化
void sensorDefaultDriver(void)

```

```

{
    sensorDefault setSensorDefDriver = (sensorDefault)GetProcAddress(g_hModu_Driver,"sensorDefault");

    if (!setSensorDefDriver) {
        qDebug() << "Failed to resolve sensorDefault function from CLR DLL";
        return;
    }

    setSensorDefDriver();
}

// 設定 wifi 頻段
void setWifiFrequencyDriver(const std::string& frequency)
{

    WifiFrequency wifiFrequencyDriver = (WifiFrequency)GetProcAddress(g_hModu_Driver,
"WifiFrequency");
    if (!wifiFrequencyDriver) {
        qDebug() << "Failed to resolve WifiFrequency function from CLR DLL";
        return;
    }

    const char* frequencyChar = frequency.c_str();

    wifiFrequencyDriver(const_cast<char*>(frequencyChar));
}

// 直接打印 bitmap
int BitmapDriver(const std::string& x, const std::string& y, int width, int height, int mode, const std::string&
filename)
{

    Bitmap bitmapDriver = (Bitmap)GetProcAddress(g_hModu_Driver, "Bitmap");
    if (!bitmapDriver) {
        qDebug() << "Failed to resolve Bitmap function from CLR DLL";
        return 0;
    }

    const char* xChar = x.c_str();

```

```

const char* yChar = y.c_str();
const char* filenameChar = filename.c_str();

    int result = bitmapDriver(const_cast<char*>(xChar), const_cast<char*>(yChar), width, height, mode,
const_cast<char*>(filenameChar));
    return result;
}

// 直接打印壓縮的 bitmap
int CompressBitmapDriver(const std::string& x, const std::string& y, int width, int height, const std::string&
filename)
{

    compressBitmap bitmapDriver = (compressBitmap)GetProcAddress(g_hModu_Driver,
"compressBitmap");
    if (!bitmapDriver) {
        qDebug() << "Failed to resolve Compress Bitmap function from CLR DLL";
        return 0;
    }

    const char* xChar = x.c_str();
    const char* yChar = y.c_str();
    const char* filenameChar = filename.c_str();

    int result = bitmapDriver(const_cast<char*>(xChar), const_cast<char*>(yChar), width,
height,const_cast<char*>(filenameChar));
    return result;
}

////===== 實際調用範本 =====
#define PRINTER_NAME "Gainscha GS-2406T"

void driverSettingTest(HMODULE hModu)
{
    g_hModu_Driver = hModu;
    int result = 0;

```

```
result = openPortDriver(PRINTER_NAME);
if(result)
{
    // 設定方向與鏡像

    setDirectionAndMirroDriver(0,0);

    setDirectionAndMirroDriver(1,0);

    setDirectionAndMirroDriver(0,1);

    setDirectionAndMirroDriver(1,1);
    closeportDriver();
}
result = openPortDriver(PRINTER_NAME);
if(result)
{
    // 設定 Shift

    setShiftDriver(80);

    setShiftDriver(50);
    closeportDriver();
}
result = openPortDriver(PRINTER_NAME);
if(result)
{
    // 設定 AfterPrinteAction

    setAfterPrintModeDriver(0);

    setAfterPrintModeDriver(1);

    setAfterPrintModeDriver(2);

    setAfterPrintModeDriver(3);
    closeportDriver();
}
result = openPortDriver(PRINTER_NAME);
```



```
if(result)
{
    // 設定 Offset

    setOffsetDriver(50);

    setOffsetDriver(5);
    closeportDriver();
}
result = openPortDriver(PRINTER_NAME);
if(result)
{
    // 設定切刀

    setCutModeDriver(0, 2);

    setCutModeDriver(1, 5);
    closeportDriver();
}
result = openPortDriver(PRINTER_NAME);
if(result)
{
    // 設定 genetic 初始化

    genericDefaultDriver();
    closeportDriver();
}
result = openPortDriver(PRINTER_NAME);
if(result)
{
    // 設定 WIFI 頻段

    setWifiFrequencyDriver("5G");

    setWifiFrequencyDriver("2.4G");
    closeportDriver();
}
}
```

```

void driverSample(HMODULE hModu)
{
    g_hModu_Driver = hModu;
    int result = 0;

    result = openPortDriver(PRINTER_NAME);
    if(result)
    {
        // 設定印表機尺寸
        clearBufferDriver();
        setupDriver("104", "80", "2", "7", "0", "3", "0");
        sendcommandDriver("DIRECTION 1");
        closeportDriver();
    }
    result = openPortDriver(PRINTER_NAME);
    if(result)
    {
        // 印一張 barcode
        barcode_Driver("30", "30", "128", "100", "1", "0", "2", "2", "barcodeDriver1234567");
        printLabelDriver("1", "1");
        closeportDriver();
    }
    result = openPortDriver(PRINTER_NAME);
    if(result)
    {
        // 印一張文字
        clearBufferDriver();
        printFontDriver("50", "10", "TSS24.BF2", "0", "1", "1", "testFontDriver12345678");
        printLabelDriver("1", "1");
        closeportDriver();
    }
    result = openPortDriver(PRINTER_NAME);
    if(result)
    {
        // 印一張中文字
        clearBufferDriver();
        printFontUnicodeDriver("50", "10", "TSS24.BF2", "0", "1", "1", "Driver 默认简体中文测试：海天米醋
白米醋 1 瓶 450ml");
        printLabelDriver("1", "1");
    }
}

```

```

    closeportDriver();
}
result = openPortDriver(PRINTER_NAME);
if(result)
{
    // 印一張 PCX 圖片
    QString currentDir = QDir::currentPath();
    std::string filenameStr = (currentDir + "\\UL.PCX").toStdString();
    clearBufferDriver();
    downloadPcxDriver(filenameStr, "UL.PCX");
    sendcommandDriver("PUTPCX 50,10,\"UL.PCX\"\\r\\n");
    printLabelDriver("1", "1");
    closeportDriver();
}
result = openPortDriver(PRINTER_NAME);
if(result)
{
    // 印一張 BMP
    QString currentDir = QDir::currentPath();
    std::string filenameStr = (currentDir + "\\CIRCLE.BMP").toStdString();
    clearBufferDriver();
    downloadBmpDriver(filenameStr, "CIRCLE.BMP");
    sendcommandDriver("PUTBMP 10,10,\"CIRCLE.BMP\"\\r\\n");
    printLabelDriver("1", "1");
    closeportDriver();
}
result = openPortDriver(PRINTER_NAME);
if(result)
{
    // 印一張 windows font
    QString currentDir = QDir::currentPath();
    clearBufferDriver();
    std::string filenameStr = (currentDir + "\\impact.ttf").toStdString();
    downloadBmpDriver(filenameStr, "impact.ttf");
    windowsFontDriver(10, 100, 48, 0, 0, 0, "impact", "C# WIN TEST");
    printLabelDriver("1", "1");
    closeportDriver();
}
result = openPortDriver(PRINTER_NAME);

```

```

if(result)
{
    // 印一張 block text
    clearBufferDriver();
    printFontblockDriver("35", "15", "300", "90", "TSS24.BF2", "0", "1", "1", "0", "1",
"DriverPrintFontBlockTestPrintFontBlockTestPrintFontBlockTestPrintFontBlockTest");
    printLabelDriver("1", "1");
    closeportDriver();
}
result = openPortDriver(PRINTER_NAME);
if(result)
{
    // 印一張 qrcode
    qrcodeDriver("220", "260", "M", "3", "A", "0", "AABCB03abcN123");
    printLabelDriver("1", "1");
    closeportDriver();
}
result = openPortDriver(PRINTER_NAME);
if(result)
{
    // 設定 Reverse
    sendcommandDriver("TEXT 100,100,\"3\",0,1,1,\"REVERSE\"");
    printReverseDriver(90, 90, 128, 40);
    printLabelDriver("1", "1");
    closeportDriver();
}
result = openPortDriver(PRINTER_NAME);
if(result)
{
    // 直接打印 BMP
    if(BitmapDriver("0", "0", 400, 350, 1, "Thunder.png")!=0)
        printLabelDriver("1", "1");
    closeportDriver();
}
result = openPortDriver(PRINTER_NAME);
if(result)
{
    // 直接打印壓縮 BMP
    if(CompressBitmapDriver("0", "0", 666, 357, "Cat1.jpg")!=0)

```

```

        printLabelDriver("1", "1");
    closeportDriver();
}
}

```

Ethernet 打印範例

////===== 定義函式指標=====

```
typedef void (*sendcommand_Ethernet)(char*);
```

```
typedef int (*openport_Ethernet)(char*, int);
```

```
typedef void (*closeport_Ethernet)();
```

```
typedef char* (*printerstatus_Ethernet)();
```

```
typedef void (*setup_Ethernet)(char*, char*, char*, char*, char*, char*, char*);
```

```
typedef void (*barcode_Ethernet)(char*, char*, char*, char*, char*, char*, char*, char*, char*, char* );
```

```
typedef void (*printlabel_Ethernet)(char*,char*);
```

```
typedef void (*printerfont_Ethernet)(char*, char*, char*, char*, char*, char*, char* );
```

```
typedef void (*printerfontUnicode_Ethernet)(char* , char* , char* , char* , char* , char* , wchar_t* );
```

```
typedef void (*clearbuffer_Ethernet)();
```

```
typedef void (*downloadpcx_Ethernet)(char*,char*);
```

```
typedef void (*downloadbmp_Ethernet)(char* , char* );
```

```
typedef void (*windowsfont_Ethernet)(int , int , int , int , int , int , char* , char* );
```

```
typedef void (*printerfontblock_Ethernet)(char* , char* , char* , char* , char* , char* , char* , char* ,
char* , char* , char* );
```

```
typedef void (*qrcode_Ethernet)(char* , char* , char* , char* , char* , char* , char* );
```

```
typedef void (*setDirectionAndMirror_Ethernet)(int , int );
```

```
typedef void (*setShift_Ethernet)(int);
```

```
typedef void (*printReverse_Ethernet)(int , int , int , int );
```

```
typedef void (*setAfterPrintAction_Ethernet)(int);
```

```
typedef void (*setOffset_Ethernet)(double);
```

```
typedef void (*setCutMode_Ethernet)(int , int );
```

```
typedef void (*genericDefault_Ethernet)();
```

```
typedef void (*sensorDefault_Ethernet)();
```

```
typedef void (*WifiFrequency_Ethernet)(char*);
```

```
typedef int (*Bitmap_Ethernet)(char* , char* , int , int , int , char* );
```

```
typedef int (*compressBitmap_Ethernet)(char* , char* , int , int , char* );
```

////===== 動態函式調用撰寫 =====

```

void sendcommand_Eth(const std::string& command)
{

    sendcommand_Ethernet sendCommand = (sendcommand_Ethernet)GetProcAddress(g_hModu,
"sendcommand_Ethernet");
    if (!sendCommand) {
        qDebug() << "Failed to resolve openport_Ethernet function from CLR DLL";
        return;
    }

    sendCommand(const_cast<char*>(command.c_str()));
}

// 開啟乙太網路通道
int openPortEth(const std::string& ipAddress, int portNumber)
{

    openport_Ethernet openPortEthFunc = (openport_Ethernet)GetProcAddress(g_hModu,
"openport_Ethernet");
    if (!openPortEthFunc) {
        qDebug() << "Failed to resolve openport_Ethernet function from CLR DLL";
        return 0;
    }

    int result = openPortEthFunc(const_cast<char*>(ipAddress.c_str()), portNumber);

    return result;
}

// 關閉乙太網路通道
void closePortEth(void)
{

    closeport_Ethernet closePortEthFunc = (closeport_Ethernet)GetProcAddress(g_hModu,
"closeport_Ethernet");
    if (!closePortEthFunc) {
        qDebug() << "Failed to resolve openport_Ethernet function from CLR DLL";
        return;
    }
}

```

```

    closePortEthFunc();
}
// 詢問印表機狀態
char* printerStatusEth()
{
    printerstatus_Ethernet getStatus =(printerstatus_Ethernet)GetProcAddress(g_hModu,
"printerstatus_Ethernet");
    if (!getStatus) {
        qDebug() << "Failed to resolve openport_Ethernet function from CLR DLL";
        return NULL;
    }

    char* status = getStatus();
    return status;
}
// 發送設定值
void setupEth(const std::string& width, const std::string& height, const std::string& speed,
              const std::string& density, const std::string& sensor, const std::string& vertical,
              const std::string& offset)
{

    setup_Ethernet setPrinter = (setup_Ethernet)GetProcAddress(g_hModu, "setup_Ethernet");
    if (!setPrinter) {
        qDebug() << "Failed to resolve openport_Ethernet function from CLR DLL";
        return;
    }

    setPrinter(const_cast<char*>(width.c_str()), const_cast<char*>(height.c_str()),
              const_cast<char*>(speed.c_str()), const_cast<char*>(density.c_str()),
              const_cast<char*>(sensor.c_str()), const_cast<char*>(vertical.c_str()),
              const_cast<char*>(offset.c_str()));
}

// 列印條碼指令
void barcode_Eth(const std::string& x, const std::string& y, const std::string& type,
                 const std::string& height, const std::string& readable, const std::string& rotation,
                 const std::string& narrow, const std::string& wide, const std::string& code)
{

```

```

barcode_Ethernet barcodeEth = (barcode_Ethernet)GetProcAddress(g_hModu, "barcode_Ethernet");
if (!barcodeEth) {
    qDebug() << "Failed to resolve barcode_Ethernet function from CLR DLL";
    return;
}

barcodeEth(const_cast<char*>(x.c_str()), const_cast<char*>(y.c_str()), const_cast<char*>(type.c_str()),
    const_cast<char*>(height.c_str()), const_cast<char*>(readable.c_str()),
const_cast<char*>(rotation.c_str()),
    const_cast<char*>(narrow.c_str()), const_cast<char*>(wide.c_str()),
const_cast<char*>(code.c_str()));
}

// 發送列印指令
void printLabel_Eth(const std::string& setting, const std::string& copy) {

    printlabel_Ethernet printLabelEth = (printlabel_Ethernet)GetProcAddress(g_hModu,
"printlabel_Ethernet");
    if (!printLabelEth) {
        qDebug() << "Failed to resolve printLabel_Eth function from CLR DLL";
        return;
    }

    printLabelEth(const_cast<char*>(setting.c_str()), const_cast<char*>(copy.c_str()));
}

// 列印文字
void printFontEth(const std::string& x, const std::string& y, const std::string& fontSize,
    const std::string& rotation, const std::string& xMul, const std::string& yMul,
    const std::string& text)
{

    printerfont_Ethernet printerFontEth = (printerfont_Ethernet)GetProcAddress(g_hModu,
"printerfont_Ethernet");
    if (!printerFontEth) {
        qDebug() << "Failed to resolve printerfont_Ethernet function from CLR DLL";
        return;
    }
}

```



```

const char* xChar = x.c_str();
const char* yChar = y.c_str();
const char* fontSizeChar = fontSize.c_str();
const char* rotationChar = rotation.c_str();
const char* xMulChar = xMul.c_str();
const char* yMulChar = yMul.c_str();

const char* textChar = text.c_str();

printerFontEth(const_cast<char*>(xChar), const_cast<char*>(yChar), const_cast<char*>(fontSizeChar),
               const_cast<char*>(rotationChar), const_cast<char*>(xMulChar), const_cast<char*>(yMulChar),
               const_cast<char*>(textChar));
}

// 列印 Unicode 格式文字
void printFontUnicodeEth(const std::string& x, const std::string& y, const std::string& fontSize,
                        const std::string& rotation, const std::string& xMul, const std::string& yMul,
                        const QString& text)
{
    printerfontUnicode_Ethernet printerFontEthUnicode =
    (printerfontUnicode_Ethernet)GetProcAddress(g_hModu, "printerfontUnicode_Ethernet");
    if (!printerFontEthUnicode) {
        qDebug() << "Failed to resolve printerfontUnicode_Ethernet function from CLR DLL";
        return;
    }

    const char* xChar = x.c_str();
    const char* yChar = y.c_str();
    const char* fontSizeChar = fontSize.c_str();
    const char* rotationChar = rotation.c_str();
    const char* xMulChar = xMul.c_str();
    const char* yMulChar = yMul.c_str();

    QVector<wchar_t> wcharArray(text.length() + 1);
    text.toWCharArray(wcharArray.data());

```

```

printerFontEthUnicode(const_cast<char*>(xChar), const_cast<char*>(yChar),
const_cast<char*>(fontSizeChar),
                    const_cast<char*>(rotationChar), const_cast<char*>(xMulChar),
const_cast<char*>(yMulChar),
                    wcharArray.data());
}

```

// 清除 label 版面

```

void clearBufferEth(void)
{

```

```

    clearbuffer_Ethernet clearBufferEth = (clearbuffer_Ethernet)GetProcAddress(g_hModu,
"clearbuffer_Ethernet");
    if (!clearBufferEth) {
        qDebug() << "Failed to resolve clearbuffer_Ethernet function from CLR DLL";
        return;
    }

```

```

    clearBufferEth();
}

```

// 下載列印 PCX 圖片

```

void downloadPcxEth(const std::string& filename, const std::string& imagename) {

```

```

    downloadpcx_Ethernet downloadPcxFunc = (downloadpcx_Ethernet)GetProcAddress(g_hModu,
"downloadpcx_Ethernet");
    if (!downloadPcxFunc) {
        qDebug() << "Failed to resolve downloadpcx_USB function from CLR DLL";
        return;
    }

```

```

    downloadPcxFunc(const_cast<char*>(filename.c_str()), const_cast<char*>(imagename.c_str())); // 將
std::string 轉換為 const char*
}

```

// 下載列印 BMP 圖片

```
void downloadBmpEth(const std::string& filename, const std::string& imagename) {
```

```
    downloadbmp_Ethernet downloadBmpFunc = (downloadbmp_Ethernet)GetProcAddress(g_hModu,
"downloadbmp_Ethernet");
    if (!downloadBmpFunc) {
        qDebug() << "Failed to resolve downloadpcx_USB function from CLR DLL";
        return;
    }
```

```
    downloadBmpFunc(const_cast<char*>(filename.c_str()), const_cast<char*>(imagename.c_str())); // 將
std::string 轉換為 const char*
}
```

// 使用 font 列印文字

```
void windowsFontEth(int x, int y, int fontheight, int rotation, int fontstyle, int fontunderline, const
std::string& szFaceName, const std::string& content)
{
```

```
    windowsfont_Ethernet windowsfontEth = (windowsfont_Ethernet)GetProcAddress(g_hModu,
"windowsfont_Ethernet");
    if (!windowsfontEth) {
        qDebug() << "Failed to resolve downloadpcx_USB function from CLR DLL";
        return;
    }
```

```
    const char* szFaceNameChar = szFaceName.c_str();
    const char* contentChar = content.c_str();
    windowsfontEth(x, y, fontheight, rotation, fontstyle, fontunderline, const_cast<char*>(szFaceNameChar),
const_cast<char*>(contentChar));
}
```

// 列印文字塊 (可換行)

```
void printFontblockEth(const std::string& x, const std::string& y, const std::string& width, const std::string&
height,
    const std::string& fontname, const std::string& rotation, const std::string& xmul,
    const std::string& ymul, const std::string& space, const std::string& align,
    const std::string& text)
```

```

{

    printerfontblock_Ethernet printerFontBlockEth = (printerfontblock_Ethernet)GetProcAddress(g_hModu,
"printerfontblock_Ethernet");
    if (!printerFontBlockEth) {
        qDebug() << "Failed to resolve downloadpcx_USB function from CLR DLL";
        return;
    }

    const char* xChar = x.c_str();
    const char* yChar = y.c_str();
    const char* widthChar = width.c_str();
    const char* heightChar = height.c_str();
    const char* fontnameChar = fontname.c_str();
    const char* rotationChar = rotation.c_str();
    const char* xmulChar = xmul.c_str();
    const char* ymulChar = ymul.c_str();
    const char* spaceChar = space.c_str();
    const char* alignChar = align.c_str();
    const char* textChar = text.c_str();
    printerFontBlockEth(const_cast<char*>(xChar), const_cast<char*>(yChar),
const_cast<char*>(widthChar),
        const_cast<char*>(heightChar), const_cast<char*>(fontnameChar),
const_cast<char*>(rotationChar),
        const_cast<char*>(xmulChar), const_cast<char*>(ymulChar), const_cast<char*>(spaceChar),
        const_cast<char*>(alignChar), const_cast<char*>(textChar));
    }
// 列印 qrcode
void qrcodeEth(const std::string& x, const std::string& y, const std::string& ECClevel, const std::string&
cellwidth,
    const std::string& mode, const std::string& rotation, const std::string& content)
{

    qrcode_Ethernet qrcodeEth = (qrcode_Ethernet)GetProcAddress(g_hModu, "qrcode_Ethernet");
    if (!qrcodeEth) {
        qDebug() << "Failed to resolve qrcodeEth function from CLR DLL";
        return;
    }
}

```

```

const char* xChar = x.c_str();
const char* yChar = y.c_str();
const char* ECClevelChar = ECClevel.c_str();
const char* cellwidthChar = cellwidth.c_str();
const char* modeChar = mode.c_str();
const char* rotationChar = rotation.c_str();
const char* contentChar = content.c_str();
qrcodeEth(const_cast<char*>(xChar), const_cast<char*>(yChar), const_cast<char*>(ECClevelChar),
          const_cast<char*>(cellwidthChar), const_cast<char*>(modeChar),
const_cast<char*>(rotationChar),
          const_cast<char*>(contentChar));
}

// 設定打印方向與鏡像
void setDirectionAndMirroEth(int direction,int mirror)
{
    setDirectionAndMirror_Ethernet setDAM = (setDirectionAndMirror_Ethernet)GetProcAddress(g_hModu,
"setDirectionAndMirror_Ethernet");

    if (!setDAM) {
        qDebug() << "Failed to resolve qrcodeEth function from CLR DLL";
        return;
    }

    setDAM(direction,mirror);
}

// 設定 shift
void setShiftEth(int shift)
{
    setShift_Ethernet setShiftEth = (setShift_Ethernet)GetProcAddress(g_hModu, "setShift_Ethernet");

    if (!setShiftEth) {
        qDebug() << "Failed to resolve qrcodeEth function from CLR DLL";
        return;
    }
}

```

```

    setShiftEth(shift);
}

// 列印反色區塊
void printReverseEth(int x_start, int y_start, int x_width, int y_height)
{
    printReverse_Ethernet setReverseEth =
    (printReverse_Ethernet)GetProcAddress(g_hModu,"printReverse_Ethernet");

    if (!setReverseEth) {
        qDebug() << "Failed to resolve qrcodeEth function from CLR DLL";
        return;
    }

    setReverseEth(x_start,y_start,x_width,y_height);
}

// 設定列印後動作
void setAfterPrintModeEth(int mode)
{
    setAfterPrintAction_Ethernet setAPM
    =(setAfterPrintAction_Ethernet)GetProcAddress(g_hModu,"setAfterPrintAction_Ethernet");

    if (!setAPM) {
        qDebug() << "Failed to resolve qrcodeEth function from CLR DLL";
        return;
    }

    setAPM(mode);
}

// 設定 offset
void setOffsetEth(double offset)
{
    setOffset_Ethernet setOffset =(setOffset_Ethernet)GetProcAddress(g_hModu,"setOffset_Ethernet");

    if (!setOffset) {
        qDebug() << "Failed to resolve qrcodeEth function from CLR DLL";
        return;
    }
}

```

```

    }

    setOffset(offset);
}

// 設定裁切模式
void setCutModeEth(int mode, int piece)
{
    setCutMode_Ethernet setCutMode =
(setCutMode_Ethernet)GetProcAddress(g_hModu,"setCutMode_Ethernet");

    if (!setCutMode) {
        qDebug() << "Failed to resolve qrcodeEth function from CLR DLL";
        return;
    }

    setCutMode(mode,piece);
}

// 設定 generic 設定初始化
void genericDefaultEth(void)
{
    genericDefault_Ethernet setGenericDef =
(genericDefault_Ethernet)GetProcAddress(g_hModu,"genericDefault_Ethernet");

    if (!setGenericDef) {
        qDebug() << "Failed to resolve qrcodeEth function from CLR DLL";
        return;
    }

    setGenericDef();
}

// 設定 sensor 設定初始化
void sensorDefaultEth(void)
{
    sensorDefault_Ethernet setSensorDefEth =
(sensorDefault_Ethernet)GetProcAddress(g_hModu,"sensorDefault_Ethernet");

```

```

if (!setSensorDefEth) {
    qDebug() << "Failed to resolve qrcodeEth function from CLR DLL";
    return;
}

setSensorDefEth();
}

// 設定 wifi 頻段
void setWifiFrequencyEth(const std::string& frequency)
{

    WifiFrequency_Ethernet wifiFrequencyEth = (WifiFrequency_Ethernet)GetProcAddress(g_hModu,
"WifiFrequency_Ethernet");
    if (!wifiFrequencyEth) {
        qDebug() << "Failed to resolve WifiFrequency_Ethernet function from CLR DLL";
        return;
    }

    const char* frequencyChar = frequency.c_str();

    wifiFrequencyEth(const_cast<char*>(frequencyChar));
}

// 直接打印 bitmap
int BitmapEth(const std::string& x, const std::string& y, int width, int height, int mode, const std::string&
filename)
{

    Bitmap_Ethernet bitmapEth = (Bitmap_Ethernet)GetProcAddress(g_hModu, "Bitmap_Ethernet");
    if (!bitmapEth) {
        qDebug() << "Failed to resolve Bitmap_Ethernet function from CLR DLL";
        return 0;
    }

    const char* xChar = x.c_str();
    const char* yChar = y.c_str();
    const char* filenameChar = filename.c_str();

```



```

    int result = bitmapEth(const_cast<char*>(xChar), const_cast<char*>(yChar), width, height, mode,
const_cast<char*>(filenameChar));
    return result;
}

```

// 直接打印壓縮的 bitmap

```

int CompressBitmapEth(const std::string& x, const std::string& y, int width, int height, const std::string&
filename)
{

```

```

    compressBitmap_Ethernet bitmapEth = (compressBitmap_Ethernet)GetProcAddress(g_hModu,
"compressBitmap_Ethernet");
    if (!bitmapEth) {
        qDebug() << "Failed to resolve Bitmap_Ethernet function from CLR DLL";
        return 0;
    }

```

```

    const char* xChar = x.c_str();
    const char* yChar = y.c_str();
    const char* filenameChar = filename.c_str();

```

```

    int result = bitmapEth(const_cast<char*>(xChar), const_cast<char*>(yChar), width,
height,const_cast<char*>(filenameChar));
    return result;
}

```

```

void sendAnnotationEth(void)
{
    sendcommand_Eth(TSPL_ANNOTATION_COMMAND);
}

```

//===== MainSampleFunction =====

```

#define SAMPLE_IP_ADDRESS "192.168.66.135"
#define SAMPLE_PORT_NUMBER 9100

```

```

void ethernetSettingSample(HMODULE hModu)
{

```

```
g_hModu = hModu;
int result = 0;
result = openPortEth(SAMPLE_IP_ADDRESS,SAMPLE_PORT_NUMBER);
if(result)
{
    // 設定方向與鏡像
    sendAnnotationEth();
    setDirectionAndMirroEth(0,0);
    sendAnnotationEth();
    setDirectionAndMirroEth(1,0);
    sendAnnotationEth();
    setDirectionAndMirroEth(0,1);
    sendAnnotationEth();
    setDirectionAndMirroEth(1,1);

    // 設定 Shift
    sendAnnotationEth();
    setShiftEth(180);
    sendAnnotationEth();
    setShiftEth(50);

    // 設定 AfterPrinteAction
    sendAnnotationEth();
    setAfterPrintModeEth(0);
    sendAnnotationEth();
    setAfterPrintModeEth(1);
    sendAnnotationEth();
    setAfterPrintModeEth(2);
    sendAnnotationEth();
    setAfterPrintModeEth(3);

    // 設定 Offset
    sendAnnotationEth();
    setOffsetEth(50);
    sendAnnotationEth();
    setOffsetEth(5);

    // 設定切刀
    sendAnnotationEth();
```

```

setCutModeEth(0, 2);
sendAnnotationEth();
setCutModeEth(1, 5);

// 設定 genetic 初始化
sendAnnotationEth();
genericDefaultEth();

// 設定 WIFI 頻段
sendAnnotationEth();
setWifiFrequencyEth("5G");
sendAnnotationEth();
setWifiFrequencyEth("2.4G");
}
}
////===== 實際調用範本 =====
#define SAMPLE_IP_ADDRESS "192.168.66.135"
#define SAMPLE_PORT_NUMBER 9100

void ethernetSettingSample(HMODULE hModu)
{
    g_hModu = hModu;
    int result = 0;
    result = openPortEth(SAMPLE_IP_ADDRESS,SAMPLE_PORT_NUMBER);
    if(result)
    {
        // 設定方向與鏡像
        sendAnnotationEth();
        setDirectionAndMirroEth(0,0);
        sendAnnotationEth();
        setDirectionAndMirroEth(1,0);
        sendAnnotationEth();
        setDirectionAndMirroEth(0,1);
        sendAnnotationEth();
        setDirectionAndMirroEth(1,1);

        // 設定 Shift
        sendAnnotationEth();
        setShiftEth(180);
    }
}

```

```
sendAnnotationEth();
setShiftEth(50);
```

```
// 設定 AfterPrinteAction
sendAnnotationEth();
setAfterPrintModeEth(0);
sendAnnotationEth();
setAfterPrintModeEth(1);
sendAnnotationEth();
setAfterPrintModeEth(2);
sendAnnotationEth();
setAfterPrintModeEth(3);
```

```
// 設定 Offset
sendAnnotationEth();
setOffsetEth(50);
sendAnnotationEth();
setOffsetEth(5);
```

```
// 設定切刀
sendAnnotationEth();
setCutModeEth(0, 2);
sendAnnotationEth();
setCutModeEth(1, 5);
```

```
// 設定 genetic 初始化
sendAnnotationEth();
genericDefaultEth();
```

```
// 設定 WIFI 頻段
sendAnnotationEth();
setWifiFrequencyEth("5G");
sendAnnotationEth();
setWifiFrequencyEth("2.4G");
```

```
}
}
```

```
void ethernetSample(HMODULE hModu)
{
```

```

g_hModu = hModu;
int result = 0;
result = openPortEth(SAMPLE_IP_ADDRESS,SAMPLE_PORT_NUMBER);
if(result)
{
// 取得印表機資訊
char* printerStatus = printerStatusEth();

// 設定印表機尺寸
clearBufferEth();
setupEth("104", "76", "2", "7", "0", "3", "0");
sendcommand_Eth("DIRECTION 1");

// 印一張 barcode
barcode_Eth("30", "30", "128", "100", "1", "0", "2", "2", "barcode1234567");
printLabel_Eth("1", "1");
// 印一張中文字
printFontUnicodeEth("50", "10", "TSS24.BF2", "0", "1", "1", "以太网路默认简体中文测试：海天米醋白
米醋 1 瓶 450ml");
printLabel_Eth("1", "1");
// 印一張文字
clearBufferEth();
printFontEth("50", "10", "TSS24.BF2", "0", "1", "1", "testFont12345678");
printLabel_Eth("1", "1");

// 印一張 PCX 圖片
QString currentDir = QDir::currentPath();
std::string filenameStr = (currentDir + "\\UL.PCX").toStdString();
clearBufferEth();
downloadPcxEth(filenameStr, "UL.PCX");
sendcommand_Eth("PUTPCX 50,10,\"UL.PCX\"\\r\\n");
printLabel_Eth("1", "1");

// 印一張 BMP
currentDir = QDir::currentPath();
filenameStr = (currentDir + "\\CIRCLE.BMP").toStdString();
clearBufferEth();
downloadBmpEth(filenameStr, "CIRCLE.BMP");
sendcommand_Eth("PUTBMP 10,10,\"CIRCLE.BMP\"\\r\\n");

```

```

printLabel_Eth("1", "1");

//印一張 windows font
currentDir = QDir::currentPath();
clearBufferEth();
filenameStr = (currentDir + "\\impact.ttf").toStdString();
downloadBmpEth(filenameStr, "impact.ttf");
windowsFontEth(10, 100, 48, 0, 0, 0, "impact", "C# WIN TEST");
printLabel_Eth("1", "1");

// 印一張 block text
clearBufferEth();
printFontblockEth("35", "15", "300", "90", "TSS24.BF2", "0", "1", "1", "0", "1",
"PrintFontBlockTestPrintFontBlockTestPrintFontBlockTestPrintFontBlockTest");
printLabel_Eth("1", "1");

// 印一張 qrcode
qrcodeEth("220", "260", "M", "3", "A", "0", "AABCB03abcN123");
printLabel_Eth("1", "1");

// 設定 Reverse
sendcommand_Eth("TEXT 100,100,\"3\",0,1,1,\"REVERSE\"");
printReverseEth(90, 90, 128, 40);
printLabel_Eth("1", "1");

// 直接打印 BMP
if(BitmapEth("0", "0", 400, 350, 1, "Thunder.png")!=0)
    printLabel_Eth("1", "1");

// 直接打印壓縮 BMP
if(CompressBitmapEth("0", "0", 666, 357, "Cat1.jpg")!=0)
    printLabel_Eth("1", "1");

closePortEth();
}
}

```

USB 打印範例

////===== 定義函式指標=====

```

typedef void (*sendcommand_USB)(char* command);

typedef int (*openport_USB)();
typedef void (*closeport_USB)();

typedef char* (*printerstatus_USB)();
typedef void (*setup_USB)(char*, char*, char*, char*, char*, char*, char*);
typedef void (*barcode_USB)(char*, char*, char*, char*, char*, char*, char*, char*, char*, char* );
typedef void (*printLabel_Usb)(char*,char*);
typedef void (*printerfont_USB)(char*, char*, char*, char*, char*, char*, char* );
typedef void (*printerfontUnicode_USB)(char*, char*, char*, char*, char*, char*, char*, wchar_t* );
typedef void (*clearbuffer_USB)();
typedef void (*downloadpcx_USB)(char*,char*);
typedef void (*downloadbmp_USB)(char*, char* );
typedef void (*windowsfont_USB)(int , int , int , int , int , int , char* , char* );
typedef void (*printerfontblock_USB)(char* , char* , char* , char* , char* , char* , char* , char* , char* , char* , char* , char* );
typedef void (*qrcode_USB)(char* , char* , char* , char* , char* , char* , char* );
typedef void (*setDirectionAndMirror_USB)(int , int );
typedef void (*setShift_USB)(int);
typedef void (*printReverse_USB)(int , int , int , int );
typedef void (*setAfterPrintAction_USB)(int);
typedef void (*setOffset_USB)(double);
typedef void (*setCutMode_USB)(int , int );
typedef void (*genericDefault_USB)();
typedef void (*sensorDefault_USB)();
typedef void (*WifiFrequency_USB)(char*);
typedef int (*Bitmap_USB)(char* , char* , int , int , int , char* );
typedef int (*compressBitmap_USB)(char* , char* , int , int , char* );
////===== 動態函式調用撰寫 =====
void sendcommandUsb(const std::string& command)
{
    sendcommand_USB sendCommand = (sendcommand_USB)GetProcAddress(g_hModu_usb,
"sendcommand_USB");
    if (!sendCommand) {
        qDebug() << "Failed to resolve openport_Ethernet function from CLR DLL";
        return;
    }
}

```

```

    sendCommand(const_cast<char*>(command.c_str()));
}
// 開啟 USB 通道
int openPortUsb(void)
{
    openport_USB openPortUsb = (openport_USB)GetProcAddress(g_hModu_usb, "openport_USB");
    if (!openPortUsb) {
        qDebug() << "Failed to resolve openport_USB function from CLR DLL";
        return 0;
    }
    int result = openPortUsb();

    return result;
}

// 關閉 USB 通道
void closeportUSB(void)
{
    closeport_USB closePortUsb = (closeport_USB)GetProcAddress(g_hModu_usb, "closeport_USB");
    if (!closePortUsb) {
        qDebug() << "Failed to resolve closeport_USB function from CLR DLL";
        return ;
    }
    closePortUsb();
}

// 詢問印表機狀態
char* printerStatusUsb()
{
    printerstatus_USB getStatus = (printerstatus_USB)GetProcAddress(g_hModu_usb, "printerstatus_USB");
    if (!getStatus) {
        qDebug() << "Failed to resolve openport_USB function from CLR DLL";
        return NULL;
    }
    char* status = getStatus();
    return status;
}

// 發送設定值
void setupUsb(const std::string& width, const std::string& height, const std::string& speed,
              const std::string& density, const std::string& sensor, const std::string& vertical,

```



```

        const std::string& offset)
{
    setup_USB setPrinter = (setup_USB)GetProcAddress(g_hModu_usb, "setup_USB");
    if (!setPrinter) {
        qDebug() << "Failed to resolve openport_USB function from CLR DLL";
        return;
    }
    setPrinter(const_cast<char*>(width.c_str()), const_cast<char*>(height.c_str()),
        const_cast<char*>(speed.c_str()), const_cast<char*>(density.c_str()),
        const_cast<char*>(sensor.c_str()), const_cast<char*>(vertical.c_str()),
        const_cast<char*>(offset.c_str()));
}

// 列印條碼指令
void barcode_Usb(const std::string& x, const std::string& y, const std::string& type,
    const std::string& height, const std::string& readable, const std::string& rotation,
    const std::string& narrow, const std::string& wide, const std::string& code)
{

    barcode_USB barcodeUsb = (barcode_USB)GetProcAddress(g_hModu_usb, "barcode_USB");
    if (!barcodeUsb) {
        qDebug() << "Failed to resolve barcode_USB function from CLR DLL";
        return;
    }

    barcodeUsb(const_cast<char*>(x.c_str()), const_cast<char*>(y.c_str()), const_cast<char*>(type.c_str()),
        const_cast<char*>(height.c_str()), const_cast<char*>(readable.c_str()),
const_cast<char*>(rotation.c_str()),
        const_cast<char*>(narrow.c_str()), const_cast<char*>(wide.c_str()),
const_cast<char*>(code.c_str()));
}

// 發送列印指令
void printLabelUsb(const std::string& setting, const std::string& copy) {

    printLabel_Usb printLabelUsb = (printLabel_Usb)GetProcAddress(g_hModu_usb, "printlabel_USB");
    if (!printLabelUsb) {
        qDebug() << "Failed to resolve printLabelUsb function from CLR DLL";

```

```

    return;
}

// 調用 printLabelUsb 函數發送列印命令
printLabelUsb(const_cast<char*>(setting.c_str()), const_cast<char*>(copy.c_str()));
}

// 列印文字
void printFontUsb(const std::string& x, const std::string& y, const std::string& fontSize,
                  const std::string& rotation, const std::string& xMul, const std::string& yMul,
                  const std::string& text)
{
    printerfont_USB printerFontUsb = (printerfont_USB)GetProcAddress(g_hModu_usb, "printerfont_USB");
    if (!printerFontUsb) {
        qDebug() << "Failed to resolve printerfont_USB function from CLR DLL";
        return;
    }

    const char* xChar = x.c_str();
    const char* yChar = y.c_str();
    const char* fontSizeChar = fontSize.c_str();
    const char* rotationChar = rotation.c_str();
    const char* xMulChar = xMul.c_str();
    const char* yMulChar = yMul.c_str();

    const char* textChar = text.c_str();

    printerFontUsb(const_cast<char*>(xChar), const_cast<char*>(yChar), const_cast<char*>(fontSizeChar),
                   const_cast<char*>(rotationChar), const_cast<char*>(xMulChar), const_cast<char*>(yMulChar),
                   const_cast<char*>(textChar));
}

// 列印 Unicode 格式文字
void printFontUnicodeUsb(const std::string& x, const std::string& y, const std::string& fontSize,
                        const std::string& rotation, const std::string& xMul, const std::string& yMul,
                        const QString& text)
{

```

```

printerfontUnicode_USB printerFontUsbUnicode =
(printerfontUnicode_USB)GetProcAddress(g_hModu_usb, "printerfontUnicode_USB");
if (!printerFontUsbUnicode) {
    qDebug() << "Failed to resolve printerfont_USB function from CLR DLL";
    return;
}

```

```

const char* xChar = x.c_str();
const char* yChar = y.c_str();
const char* fontSizeChar = fontSize.c_str();
const char* rotationChar = rotation.c_str();
const char* xMulChar = xMul.c_str();
const char* yMulChar = yMul.c_str();

```

```

QVector<wchar_t> wcharArray(text.length() + 1);
text.toWCharArray(wcharArray.data());

```

```

printerFontUsbUnicode(const_cast<char*>(xChar), const_cast<char*>(yChar),
const_cast<char*>(fontSizeChar),
                    const_cast<char*>(rotationChar), const_cast<char*>(xMulChar),
const_cast<char*>(yMulChar),
                    wcharArray.data());
}

```

// 清除 label 版面

```

void clearBufferUsb(void)
{

```

```

clearbuffer_USB clearBufferUsb = (clearbuffer_USB)GetProcAddress(g_hModu_usb, "clearbuffer_USB");
if (!clearBufferUsb) {
    qDebug() << "Failed to resolve clearbuffer_USB function from CLR DLL";
    return;
}

```

```

clearBufferUsb();
}

```

// 下載列印 PCX 圖片

```
void downloadPcxUsb(const std::string& filename, const std::string& imagename) {
```

```
    downloadpcx_USB downloadPcxFunc = (downloadpcx_USB)GetProcAddress(g_hModu_usb,
"downloadpcx_USB");
    if (!downloadPcxFunc) {
        qDebug() << "Failed to resolve downloadpcx_USB function from CLR DLL";
        return;
    }
```

```
    downloadPcxFunc(const_cast<char*>(filename.c_str()), const_cast<char*>(imagename.c_str())); // 將
std::string 轉換為 const char*
}
```

```
// 下載列印 BMP 圖片
```

```
void downloadBmpUsb(const std::string& filename, const std::string& imagename) {
```

```
    downloadbmp_USB downloadBmpFunc = (downloadbmp_USB)GetProcAddress(g_hModu_usb,
"downloadbmp_USB");
    if (!downloadBmpFunc) {
        qDebug() << "Failed to resolve downloadpcx_USB function from CLR DLL";
        return;
    }
```

```
    downloadBmpFunc(const_cast<char*>(filename.c_str()), const_cast<char*>(imagename.c_str())); // 將
std::string 轉換為 const char*
}
```

```
// 使用 font 列印文字
```

```
void windowsFontUsb(int x, int y, int fontheight, int rotation, int fontstyle, int fontunderline, const
std::string& szFaceName, const std::string& content)
{
```

```
    windowsfont_USB windowsfontUsb = (windowsfont_USB)GetProcAddress(g_hModu_usb,
"windowsfont_USB");
    if (!windowsfontUsb) {
        qDebug() << "Failed to resolve downloadpcx_USB function from CLR DLL";
        return;
    }
```

```

}

const char* szFaceNameChar = szFaceName.c_str();
const char* contentChar = content.c_str();
windowsfontUsb(x, y, fontheight, rotation, fontstyle, fontunderline,
const_cast<char*>(szFaceNameChar), const_cast<char*>(contentChar));
}

```

// 列印文字塊 (可換行)

```

void printFontblockUsb(const std::string& x, const std::string& y, const std::string& width, const std::string&
height,
    const std::string& fontname, const std::string& rotation, const std::string& xmul,
    const std::string& ymul, const std::string& space, const std::string& align,
    const std::string& text)
{

```

```

    printerfontblock_USB printerFontBlockUsb = (printerfontblock_USB)GetProcAddress(g_hModu_usb,
"printerfontblock_USB");
    if (!printerFontBlockUsb) {
        qDebug() << "Failed to resolve downloadpcx_USB function from CLR DLL";
        return;
    }

```

```

const char* xChar = x.c_str();
const char* yChar = y.c_str();
const char* widthChar = width.c_str();
const char* heightChar = height.c_str();
const char* fontnameChar = fontname.c_str();
const char* rotationChar = rotation.c_str();
const char* xmulChar = xmul.c_str();
const char* ymulChar = ymul.c_str();
const char* spaceChar = space.c_str();
const char* alignChar = align.c_str();
const char* textChar = text.c_str();
printerFontBlockUsb(const_cast<char*>(xChar), const_cast<char*>(yChar),
const_cast<char*>(widthChar),
    const_cast<char*>(heightChar), const_cast<char*>(fontnameChar),
const_cast<char*>(rotationChar),

```

```

        const_cast<char*>(xmulChar), const_cast<char*>(ymulChar), const_cast<char*>(spaceChar),
        const_cast<char*>(alignChar), const_cast<char*>(textChar));
    }
// 列印 qrcode
void qrcodeUsb(const std::string& x, const std::string& y, const std::string& ECClevel, const std::string&
cellwidth,
               const std::string& mode, const std::string& rotation, const std::string& content)
{
    qrcode_USB qrcodeUsb = (qrcode_USB)GetProcAddress(g_hModu_usb, "qrcode_USB");
    if (!qrcodeUsb) {
        qDebug() << "Failed to resolve qrcodeUsb function from CLR DLL";
        return;
    }

    const char* xChar = x.c_str();
    const char* yChar = y.c_str();
    const char* ECClevelChar = ECClevel.c_str();
    const char* cellwidthChar = cellwidth.c_str();
    const char* modeChar = mode.c_str();
    const char* rotationChar = rotation.c_str();
    const char* contentChar = content.c_str();
    qrcodeUsb(const_cast<char*>(xChar), const_cast<char*>(yChar), const_cast<char*>(ECClevelChar),
              const_cast<char*>(cellwidthChar), const_cast<char*>(modeChar),
const_cast<char*>(rotationChar),
              const_cast<char*>(contentChar));
    }

// 設定打印方向與鏡像
void setDirectionAndMirroUsb(int direction,int mirror)
{
    setDirectionAndMirror_USB setDAM = (setDirectionAndMirror_USB)GetProcAddress(g_hModu_usb,
"setDirectionAndMirror_USB");

    if (!setDAM) {
        qDebug() << "Failed to resolve qrcodeUsb function from CLR DLL";
        return;
    }
}

```

```

    setDAM(direction,mirror);
}

// 設定 shift
void setShiftUsb(int shift)
{
    setShift_USB setShiftUsb = (setShift_USB)GetProcAddress(g_hModu_usb, "setShift_USB");

    if (!setShiftUsb) {
        qDebug() << "Failed to resolve qrCodeUsb function from CLR DLL";
        return;
    }

    setShiftUsb(shift);
}

// 列印反色區塊
void printReverseUsb(int x_start, int y_start, int x_width, int y_height)
{
    printReverse_USB setReverseUsb =
(printReverse_USB)GetProcAddress(g_hModu_usb,"printReverse_USB");

    if (!setReverseUsb) {
        qDebug() << "Failed to resolve qrCodeUsb function from CLR DLL";
        return;
    }

    setReverseUsb(x_start,y_start,x_width,y_height);
}

// 設定列印後動作
void setAfterPrintModeUsb(int mode)
{
    setAfterPrintAction_USB setAPM
=(setAfterPrintAction_USB)GetProcAddress(g_hModu_usb,"setAfterPrintAction_USB");

    if (!setAPM) {
        qDebug() << "Failed to resolve qrCodeUsb function from CLR DLL";

```

```

    return;
}

setAPM(mode);
}

// 設定 offset
void setOffsetUsb(double offset)
{
    setOffset_USB setOffset =(setOffset_USB)GetProcAddress(g_hModu_usb,"setOffset_USB");

    if (!setOffset) {
        qDebug() << "Failed to resolve qrcodeUsb function from CLR DLL";
        return;
    }

    setOffset(offset);
}

// 設定裁切模式
void setCutModeUsb(int mode, int piece)
{
    setCutMode_USB setCutMode = (setCutMode_USB)GetProcAddress(g_hModu_usb,"setCutMode_USB");

    if (!setCutMode) {
        qDebug() << "Failed to resolve qrcodeUsb function from CLR DLL";
        return;
    }

    setCutMode(mode,piece);
}

// 設定 generic 設定初始化
void genericDefaultUsb(void)
{
    genericDefault_USB setGenericDef =
(genericDefault_USB)GetProcAddress(g_hModu_usb,"genericDefault_USB");

    if (!setGenericDef) {

```



```

    qDebug() << "Failed to resolve qrCodeUsb function from CLR DLL";
    return;
}

setGenericDef();
}

// 設定 sensor 設定初始化
void sensorDefaultUsb(void)
{
    sensorDefault_USB setSensorDefUsb =
(sensorDefault_USB)GetProcAddress(g_hModu_usb,"sensorDefault_USB");

    if (!setSensorDefUsb) {
        qDebug() << "Failed to resolve qrCodeUsb function from CLR DLL";
        return;
    }

    setSensorDefUsb();
}

// 設定 wifi 頻段
void setWifiFrequencyUsb(const std::string& frequency)
{
    WifiFrequency_USB wifiFrequencyUsb = (WifiFrequency_USB)GetProcAddress(g_hModu_usb,
"WifiFrequency_USB");
    if (!wifiFrequencyUsb) {
        qDebug() << "Failed to resolve WifiFrequency_USB function from CLR DLL";
        return;
    }

    const char* frequencyChar = frequency.c_str();

    wifiFrequencyUsb(const_cast<char*>(frequencyChar));
}

```

// 直接打印 bitmap

```
int BitmapUsb(const std::string& x, const std::string& y, int width, int height, int mode, const std::string&
filename)
```

```
{

    Bitmap_USB bitmapUsb = (Bitmap_USB)GetProcAddress(g_hModu_usb, "Bitmap_USB");
    if (!bitmapUsb) {
        qDebug() << "Failed to resolve Bitmap_USB function from CLR DLL";
        return 0;
    }
```

```
    const char* xChar = x.c_str();
    const char* yChar = y.c_str();
    const char* filenameChar = filename.c_str();
```

```
    int result = bitmapUsb(const_cast<char*>(xChar), const_cast<char*>(yChar), width, height, mode,
const_cast<char*>(filenameChar));
    return result;
}
```

// 直接打印壓縮的 bitmap

```
int CompressBitmapUsb(const std::string& x, const std::string& y, int width, int height, const std::string&
filename)
```

```
{

    compressBitmap_USB bitmapUsb = (compressBitmap_USB)GetProcAddress(g_hModu_usb,
"compressBitmap_USB");
    if (!bitmapUsb) {
        qDebug() << "Failed to resolve Bitmap_USB function from CLR DLL";
        return 0;
    }
```

```
    const char* xChar = x.c_str();
    const char* yChar = y.c_str();
    const char* filenameChar = filename.c_str();
```

```

int result = bitmapUsb(const_cast<char*>(xChar), const_cast<char*>(yChar), width,
height,const_cast<char*>(filenameChar));
return result;
}

```

```

void sendAnnotationUsb(void)
{
    sendcommandUsb(TSPL_ANNOTATION_COMMAND);
}

```

////===== 實際調用範本 =====

```

void usbSettingTest(HMODULE hModu)

```

```

{
    g_hModu_usb = hModu;
    int result = 0;

```

```

    result=openPortUsb();
    if(result)
    {

```

```

        // 設定方向與鏡像
        sendAnnotationUsb();
        setDirectionAndMirroUsb(0,0);
        sendAnnotationUsb();
        setDirectionAndMirroUsb(1,0);
        sendAnnotationUsb();
        setDirectionAndMirroUsb(0,1);
        sendAnnotationUsb();
        setDirectionAndMirroUsb(1,1);

```

```

        // 設定 Shift
        sendAnnotationUsb();
        setShiftUsb(180);
        sendAnnotationUsb();
        setShiftUsb(50);

```

```

        // 設定 AfterPrinteAction
        sendAnnotationUsb();
        setAfterPrintModeUsb(0);
        sendAnnotationUsb();

```

```

setAfterPrintModeUsb(1);
sendAnnotationUsb();
setAfterPrintModeUsb(2);
sendAnnotationUsb();
setAfterPrintModeUsb(3);

```

```

// 設定 Offset
sendAnnotationUsb();
setOffsetUsb(50);
sendAnnotationUsb();
setOffsetUsb(5);

```

```

// 設定切刀
sendAnnotationUsb();
setCutModeUsb(0, 2);
sendAnnotationUsb();
setCutModeUsb(1, 5);

```

```

// 設定 genetic 初始化
sendAnnotationUsb();
genericDefaultUsb();

```

```

// 設定 WIFI 頻段
sendAnnotationUsb();
setWifiFrequencyUsb("5G");
sendAnnotationUsb();
setWifiFrequencyUsb("2.4G");

```

```

// 設定 Shift
sendAnnotationUsb();
setShiftUsb(80);
sendAnnotationUsb();
setShiftUsb(50);

```

```

}
}

```

```

void usbSample(HMODULE hModu)
{
    g_hModu_usb = hModu;

```

```

int result = 0;

result=openPortUsb();
if(result)
{
    // 取得印表機資訊
    char* printerStatus = printerStatusUsb();

    // 設定印表機尺寸
    clearBufferUsb();
    setupUsb("104", "76", "2", "7", "0", "3", "0");
    sendcommandUsb("DIRECTION 1");

    // 印一張中文字
    printFontUnicodeUsb("50", "10", "TSS24.BF2", "0", "1", "1", "USB 接口默认简体中文测试：海天米醋
白米醋 1 瓶 450ml");
    printLabelUsb("1", "1");

    // 印一張 barcode
    barcode_Usb("30", "30", "128", "100", "1", "0", "2", "2", "barcode1234567");
    printLabelUsb("1", "1");

    // 印一張文字
    clearBufferUsb();
    printFontUsb("50", "10", "TSS24.BF2", "0", "1", "1", "testFont12345678");
    printLabelUsb("1", "1");

    // 印一張 PCX 圖片
    QString currentDir = QDir::currentPath();
    std::string filenameStr = (currentDir + "\\UL.PCX").toStdString();
    clearBufferUsb();
    downloadPcxUsb(filenameStr, "UL.PCX");
    sendcommandUsb("PUTPCX 50,10,\"UL.PCX\"\\r\\n");
    printLabelUsb("1", "1");

    // 印一張 BMP
    currentDir = QDir::currentPath();
    filenameStr = (currentDir + "\\CIRCLE.BMP").toStdString();
    clearBufferUsb();

```

```

downloadBmpUsb(filenameStr, "CIRCLE.BMP");
sendcommandUsb("PUTBMP 10,10,\"CIRCLE.BMP\"\\r\\n");
printLabelUsb("1", "1");

//印一張 windows font
currentDir = QDir::currentPath();
clearBufferUsb();
filenameStr = (currentDir + "\\impact.ttf").toString();
downloadBmpUsb(filenameStr, "impact.ttf");
windowsFontUsb(10, 100, 48, 0, 0, 0, "impact", "C# WIN TEST");
printLabelUsb("1", "1");

// 印一張 block text
clearBufferUsb();
printFontblockUsb("35", "15", "300", "90", "TSS24.BF2", "0", "1", "1", "0", "1",
"PrintFontBlockTestPrintFontBlockTestPrintFontBlockTestPrintFontBlockTest");
printLabelUsb("1", "1");

// 印一張 qrcode
qrcodeUsb("220", "260", "M", "3", "A", "0", "AABCB03abcN123");
printLabelUsb("1", "1");

// 設定 Reverse
sendcommandUsb("TEXT 100,100,\"3\",0,1,1,\"REVERSE\"");
printReverseUsb(90, 90, 128, 40);
printLabelUsb("1", "1");

// 直接打印 BMP
if(BitmapUsb("0", "0", 400, 350, 1, "Thunder.png")!=0)
    printLabelUsb("1", "1");

// 直接打印壓縮 BMP
if(CompressBitmapUsb("0", "0", 666, 357, "Cat1.jpg")!=0)
    printLabelUsb("1", "1");

closeportUSB();
}
}

```

1.需先匯入 jan-5.5.0.jar :

```
import com.sun.jna.Library;
```

```
import com.sun.jna.Native;
```

2.建立調用 dll 的 class 調用 dll

```
class GTSPL {  
  
    interface GTSPL_SDK extends Library {  
  
        GTSPL_SDK INSTANCE = (GTSPL_SDK) Native.load("GTSPL_SDK_C", GTSPL_SDK.class);  
  
        int openport_USB();  
  
        int openport(String PrinterName);  
  
        ....  
  
    }  
}
```

3.範例程式：

```
//單機列印
```

```
GTSPL.GTSPL_SDK.INSTANCE.openport("Printer Model");  
  
GTSPL.GTSPL_SDK.INSTANCE.setup("54", "30", "2", "3", "0", "3", "0");  
  
GTSPL.GTSPL_SDK.INSTANCE.sendcommand("DIRECTION 1");  
  
GTSPL.GTSPL_SDK.INSTANCE.setDirectionAndMirror(1, 1);  
  
GTSPL.GTSPL_SDK.INSTANCE.setShift(50);  
  
GTSPL.GTSPL_SDK.INSTANCE.setAfterPrintAction(2);  
  
GTSPL.GTSPL_SDK.INSTANCE.setOffset(20);  
  
GTSPL.GTSPL_SDK.INSTANCE.setCutMode(0,2);  
  
GTSPL.GTSPL_SDK.INSTANCE.clearbuffer();  
  
GTSPL.GTSPL_SDK.INSTANCE.sendcommand("TEXT 100,100,\"3\",0,1,1,\"REVERSE\");  
  
GTSPL.GTSPL_SDK.INSTANCE.printReverse(90, 90, 128, 40);
```

```

GTSP.LGTSP.L_SDK.INSTANCE.barcode("30", "30", "128", "100", "1", "0", "2", "2",
"barcode1234567");

GTSP.LGTSP.L_SDK.INSTANCE.qrcode("50", "100", "H", "4", "A", "0", "QRcode1234567");

GTSP.LGTSP.L_SDK.INSTANCE.printerfont("50", "10", "2", "0", "1", "1", "Print Font 123456");

GTSP.LGTSP.L_SDK.INSTANCE.downloadbmp(System.getProperty("user.dir") + "\\CIRCLE.BMP",
"CIRCLE.BMP");

GTSP.LGTSP.L_SDK.INSTANCE.windowfont(5, 150, 48, 0, 0, 0, "arial", "JAVA WIN DRIVER");

GTSP.LGTSP.L_SDK.INSTANCE.sendcommand("PUTBMP 150,30,\"CIRCLE.BMP\"");

GTSP.LGTSP.L_SDK.INSTANCE.printerfontblock("15", "15", "790", "90", "0", "0", "8", "8", "20",
"2","We stand behind our products with one of the most comprehensive support programs in the
Auto-ID industry.");

```

//初始化

```

GTSP.LGTSP.L_SDK.INSTANCE.genericDefault();

GTSP.LGTSP.L_SDK.INSTANCE.sensorDefault();

GTSP.LGTSP.L_SDK.INSTANCE.rfidSetupDefault();

```

//簡中打印

WString str=new WString("默认简体中文测试"); //需 jni 的 WString 才能正確傳送中文

```

GTSP.LGTSP.L_SDK.INSTANCE.clearbuffer();

GTSP.LGTSP.L_SDK.INSTANCE.printerfontUnicode("100", "10", "TSS24.BF2", "0", "1", "1", str);

GTSP.LGTSP.L_SDK.INSTANCE.printlabel(1, 1);

```

//繁中打印

WString str=new Wstring("默認繁體中文測試"); //需 jni 的 WString 才能正確傳送中文

```

GTSP.LGTSP.L_SDK.INSTANCE.clearbuffer();

GTSP.LGTSP.L_SDK.INSTANCE. printerfontUnicode("100", "10", " TST24.BF2", "0", "1", "1", str);

```



```

GTSP.LGTSP.L_SDK.INSTANCE.printlabel(1, 1);

GTSP.LGTSP.L_SDK.INSTANCE.closeport();

GTSP.LGTSP.L_SDK.INSTANCE.getDLLVersion(0);


//指定 USB 傳輸介面
//偵測多台印表機

String PrinterName = GTSP.LGTSP.L_SDK.INSTANCE.detectUSBStr_USB();

String[] PrinterNameStringArray = PrinterName.split("\\n+");

GTSP.LGTSP.L_SDK.INSTANCE.openport_USB("Printer Model");

//單機列印

GTSP.LGTSP.L_SDK.INSTANCE.openport_USB();

GTSP.LGTSP.L_SDK.INSTANCE.setup_USB("54", "30", "2", "3", "0", "3", "0");

GTSP.LGTSP.L_SDK.INSTANCE.sendcommand_USB("DIRECTION 1");

GTSP.LGTSP.L_SDK.INSTANCE.setDirectionAndMirror_USB(1, 1);

GTSP.LGTSP.L_SDK.INSTANCE.setShift_USB(50);

GTSP.LGTSP.L_SDK.INSTANCE.setAfterPrintAction_USB(2);

GTSP.LGTSP.L_SDK.INSTANCE.setOffset_USB(20);

GTSP.LGTSP.L_SDK.INSTANCE.setCutMode_USB(0,2);

GTSP.LGTSP.L_SDK.INSTANCE.clearbuffer_USB();

GTSP.LGTSP.L_SDK.INSTANCE.sendcommand_USB("TEXT 100,100,\"3\",0,1,1,\"REVERSE\"");

GTSP.LGTSP.L_SDK.INSTANCE.printReverse_USB(90, 90, 128, 40);

GTSP.LGTSP.L_SDK.INSTANCE.barcode_USB("30", "30", "128", "100", "1", "0", "2", "2",
"barcode1234567");

GTSP.LGTSP.L_SDK.INSTANCE.qrcode_USB("50", "100", "H", "4", "A", "0", "QRcode1234567");

GTSP.LGTSP.L_SDK.INSTANCE.printerfont_USB("50", "10", "2", "0", "1", "1", "Print Font 123456");

GTSP.LGTSP.L_SDK.INSTANCE.windowfont_USB(5, 150, 48, 0, 0, 0, "arial", "JAVA WIN DRIVER");

```

```

GTSP.LGTSP.L_SDK.INSTANCE.downloadpcx_USB(System.getProperty("user.dir")+ "\\UL.PCX",
"UL.PCX");

GTSP.LGTSP.L_SDK.INSTANCE.sendcommand_USB("PUTPCX 50,10,\"UL.PCX\\");

GTSP.LGTSP.L_SDK.INSTANCE.printerfontblock_USB("15", "15", "790", "90", "0", "0", "8", "8", "20",
"2", "We stand behind our products with one of the most comprehensive support programs in the
Auto-ID industry.");

```

//初始化

```

GTSP.LGTSP.L_SDK.INSTANCE.genericDefault_USB();

GTSP.LGTSP.L_SDK.INSTANCE.sensorDefault_USB();

GTSP.LGTSP.L_SDK.INSTANCE.rfidSetupDefault_USB();

String status = GTSP.LGTSP.L_SDK.INSTANCE.printerstatus_USB();

```

//簡中打印

WString str=new WString("默认简体中文测试"); //需 jni 的 WString 才能正确传送中文

```

GTSP.LGTSP.L_SDK.INSTANCE.clearbuffer_USB();

GTSP.LGTSP.L_SDK.INSTANCE.printerfontUnicode_USB ("100", "10", "TSS24.BF2", "0", "1", "1", str);

GTSP.LGTSP.L_SDK.INSTANCE.printlabel_USB (1, 1);

```

//繁中打印

WString str=new Wstring("默认繁体中文测试"); //需 jni 的 WString 才能正确传送中文

```

GTSP.LGTSP.L_SDK.INSTANCE.clearbuffer_USB();

GTSP.LGTSP.L_SDK.INSTANCE. printerfontUnicode_USB ("100", "10", " TST24.BF2", "0", "1", "1", str);

GTSP.LGTSP.L_SDK.INSTANCE.printlabel_USB(1, 1);

GTSP.LGTSP.L_SDK.INSTANCE.closeport_USB();

GTSP.LGTSP.L_SDK.INSTANCE.getDLLVersion_USB (0);

```

//指定 Ethernet 傳輸介面

```

GTSP.LGTSP.L_SDK.INSTANCE.openport_Ethernet(IP,Port);

GTSP.LGTSP.L_SDK.INSTANCE.setup_Ethernet("54", "30", "2", "3", "0", "3", "0");

GTSP.LGTSP.L_SDK.INSTANCE.sendcommand_Ethernet("DIRECTION 1");

GTSP.LGTSP.L_SDK.INSTANCE.setOffset_Ethernet(20);

GTSP.LGTSP.L_SDK.INSTANCE.setCutMode_Ethernet(0,2);

GTSP.LGTSP.L_SDK.INSTANCE.setDirectionAndMirror_Ethernet(1, 1);

GTSP.LGTSP.L_SDK.INSTANCE.setShift_Ethernet(50);

GTSP.LGTSP.L_SDK.INSTANCE.setAfterPrintAction_Ethernet(2);

GTSP.LGTSP.L_SDK.INSTANCE.clearbuffer_Ethernet();

GTSP.LGTSP.L_SDK.INSTANCE.sendcommand_Ethernet("TEXT 100,100,\"3\",0,1,1,\"REVERSE\"");

GTSP.LGTSP.L_SDK.INSTANCE.printReverse_Ethernet(90, 90, 128, 40);

GTSP.LGTSP.L_SDK.INSTANCE.barcode_Ethernet("30", "30", "128", "100", "1", "0", "2", "2",
"barcode1234567");

GTSP.LGTSP.L_SDK.INSTANCE.qrcode_Ethernet("50", "100", "H", "4", "A", "0", "QRcode1234567");

GTSP.LGTSP.L_SDK.INSTANCE.printerfont_Ethernet("50", "10", "2", "0", "1", "1", "Print Font
123456");

GTSP.LGTSP.L_SDK.INSTANCE.windowfont_Ethernet(5, 150, 48, 0, 0, 0, "arial", "JAVA WIN DRIVER");

GTSP.LGTSP.L_SDK.INSTANCE.downloadpcx_Ethernet(System.getProperty("user.dir")+ "\\UL.PCX",
"UL.PCX");

GTSP.LGTSP.L_SDK.INSTANCE.sendcommand_Ethernet("PUTPCX 50,10,\"UL.PCX\"");

GTSP.LGTSP.L_SDK.INSTANCE.printerfontblock_Ethernet("15", "15", "790", "90", "0", "0", "8", "8",
"20", "2", "We stand behind our products with one of the most comprehensive support programs in
the Auto-ID industry.");

```

//初始化

```
GTSP.LGTSP.L_SDK.INSTANCE.genericDefault_Ethernet();
GTSP.LGTSP.L_SDK.INSTANCE.sensorDefault_Ethernet();
GTSP.LGTSP.L_SDK.INSTANCE.rfidSetupDefault_Ethernet();
```

```
String status = GTSP.LGTSP.L_SDK.INSTANCE.printerstatus_Ethernet();
```

//簡中打印

WString str=new WString("默认简体中文测试"); //需 jni 的 WString 才能正確傳送中文

```
GTSP.LGTSP.L_SDK.INSTANCE.clearbuffer_Ethernet();
GTSP.LGTSP.L_SDK.INSTANCE.printerfontUnicode_Ethernet("100", "10", "TSS24.BF2", "0", "1", "1",
str);
GTSP.LGTSP.L_SDK.INSTANCE.printlabel_Ethernet(1, 1);
```

//繁中打印

WString str=new Wstring("默認繁體中文測試");//需 jni 的 WString 才能正確傳送中文

```
GTSP.LGTSP.L_SDK.INSTANCE.clearbuffer_Ethernet();
GTSP.LGTSP.L_SDK.INSTANCE. printerfontUnicode_Ethernet("100", "10", " TST24.BF2", "0", "1", "1",
str);
GTSP.LGTSP.L_SDK.INSTANCE.printlabel_Ethernet(1, 1);
GTSP.LGTSP.L_SDK.INSTANCE.closeport_Ethernet();
GTSP.LGTSP.L_SDK.INSTANCE.getDLLVersion_Ethernet(0);
```

4.DLL 放置位置：

〔 利用 IDE 開發時 〕

GTSP.L_SDK.dll 放在 jdk 的 bin 資料夾下

如：C:\Program Files\Java\jdk1.8.0_221\bin

GTSPK_SDK_C.dll 放在 java 專案資料夾下

〔未安裝 jdk 直接執行.jar 檔時〕

GTSPK_SDK.dll 放在 jre 的 bin 資料夾下

如：C:\Program Files\Java\jre1.8.0_251\bin

GTSPK_SDK_C.dll 放在.jar 檔相同路徑下

.jar 檔相同路徑下，還需要有 lib 資料夾，且裡面要有 jna-5.5.0.jar

5.Driver 模式下，需要安裝印表機的驅動才能執行

附件

Code Type	Description	Narrow : Width					Max. data length
		1:1	1:2	1:3	2:5	3:7	
128	Code 128, switching code subset automatically.	V					
128M	Code 128, switching code subset manually.	V					
EAN128	EAN128, switching code subset automatically.	V					
EAN128M	EAN128M, switching code subset manually.	V					
25	Interleaved 2 of 5.		V	V	V		Length is even
25C	Interleaved 2 of 5 with check digit.		V	V	V		Length is odd
25S	Standard 2 of 5.		V	V	V		
25I	Industrial 2 of 5.		V	V	V		
39	Code 39, switching standard and full ASCII mode automatically.		V	V	V		
39C	Code 39 with check digit.		V	V	V		
93	Code 93.			V			
EAN13	EAN 13.	V					12
EAN13+2	EAN 13 with 2 digits add-on.	V					14
EAN13+5	EAN 13 with 5 digits add-on.	V					17
EANB	EAN 8.	V					7
EANB+2	EAN 8 with 2 digits add-on.	V					96
EANB+5	EAN 8 with 5 digits add-on.	V					12
CODA	Codabar.		V	V	V		
POST	Postnet.	V					5,9,11
UPCA	UPC-A.	V					11
UPCA+2	UPC-A with 2 digits add-on.	V					13
UPA+5	UPC-A with 5 digits add-on.	V					16
UPCE	UPC-E.	V					6
UPCE+2	UPC-E with 2 digits add-on.	V					8
UPE+5	UPC-E with 5 digits add-on.	V					11
MSI	MSI.		V	V	V		
MSIC	MSI with check digit.		V	V	V		
PLESSEY	PLESSEY.		V	V	V		
CPOST	China post.					V	
ITF14	ITF14.		V	V	V		13
EAN14	EAN14.	V					13
11	Code 11.		V	V	V		
TELEPEN	Telepen. *Since V6.89EZ.		V	V	V		
TELEPENN	Telepen number. *Since V6.89EZ.		V	V	V		
PLANET	Planet. *Since V6.89EZ.	V					
CODE49	Code 49. *Since V6.89EZ.	V					
DPI	Deutsche Post Identcode. *Since V6.91EZ.		V	V	V		11
DPL	Deutsche Post Leitcode. *Since V6.91EZ.		V	V	V		13
LOGMARS	A special use of Code 39. *Since V6.88EZ.		V	V	V		