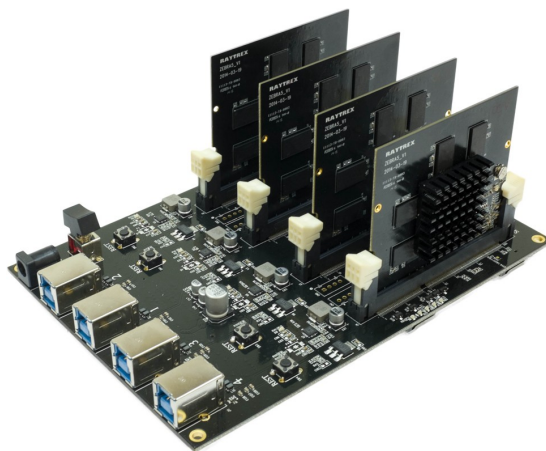


**Raytrex**  
**Zebra**  
**MIPI CSI-2 Capture card**  
**Software**  
**User's manual**  
**v1.0.1.4**  
**September, 25, 2014**



## Table of Contents

|   |    |
|---|----|
| Update History.....                           | 3  |
| 2014.09.24 (v1.0.1.4).....                    | 3  |
| 2014.07.14 (v1.0.1.0).....                    | 4  |
| 2014.06.03.....                               | 5  |
| 2014.05.16.....                               | 6  |
| Zebra API 的版本.....                            | 9  |
| Zebra win32 API 應含檔案.....                     | 10 |
| Zebra win32 API 說明.....                       | 10 |
| InitMipiUsbNotifyWin.....                     | 10 |
| StopMipiUsbNotifyWin.....                     | 10 |
| lookingViewableZebras .....                   | 10 |
| freeZebrasInfoListQueue .....                 | 11 |
| ZebraCardNumber.....                          | 11 |
| getZebraBoardInfo.....                        | 11 |
| QueryZebraAppState.....                       | 11 |
| bInitUsbDevice.....                           | 12 |
| CloseUsbDevice.....                           | 12 |
| getActivePortNname.....                       | 12 |
| queryZebraTestStepCount.....                  | 13 |
| KickoffRunStep.....                           | 13 |
| getRunStepstate.....                          | 14 |
| GrabImageRequest.....                         | 15 |
| resetZebraFromApp.....                        | 16 |
| resetZebraByUSBDev.....                       | 16 |
| getZebraStatus.....                           | 17 |
| getZebraSWVersion.....                        | 17 |
| LoadRegisterFiles.....                        | 18 |
| SearchingRegisterFiles.....                   | 18 |
| getCaptureCardTestProcedures.....             | 19 |
| createCaptureTestingTemplate.....             | 19 |
| getPicture2File.....                          | 20 |
| getPicture2Buffer.....                        | 20 |
| I2cRawDataWrite.....                          | 21 |
| I2cRawDataWrite2.....                         | 21 |
| I2cRawDataRead.....                           | 22 |
| I2cRawDataRead2.....                          | 22 |
| getMaxAliasRunSteps2AllowDownload.....        | 24 |
| getAliasRunStepsCntJustDownload.....          | 24 |
| get1stDownloadStepAliasName.....              | 25 |
| getNextDownloadStepAliasName.....             | 25 |
| getRunStepIdFromDownloadList.....             | 25 |
| RunStepByAliasName.....                       | 26 |
| getRegisterStepsCntJustDownload.....          | 26 |
| get1stDownloadRegisterStepName.....           | 26 |
| getNextDownloadRegisterStepName.....          | 27 |
| getRunStepIdFromDownloadRegisterStepList..... | 27 |
| RunTestStepFile.....                          | 27 |
| SetZebraSetting.....                          | 28 |
| RunRegisterFile.....                          | 28 |
| getDllVersion.....                            | 29 |
| getHardwareType.....                          | 29 |



## Update History

### 2014.09.24(v1.0.1.4)

主要的變更：

#### 1. 主要提升的功能 -

對多張 Zebra cards 支援方式:提供一 application 可以同時控制多張 Zebra cards 的能力;多個 applications 可以互斥地分別控制多張 Zebra cards (每張 Zebra card 僅能被一 application 所控制)。

#### 2. 新增加的 APIs -

int **lookingViewableZebras**(void) -

查詢 PC 上有多少 zebra cards, 建立此 Zebras cards 的 manage-queue list.

(InitMipiUsbNotifyWin () 可啟動 USB notification 在 background 維護此 manage-queue)

void **freeZebrasInfoListQueue**(void) -

Application program 結束時, free zebra cards 的 manage-queue.

int **ZebraCardNumber**(void) -

report 目前的 manage-queue list 中的 zebras cards 記錄筆數,即有多少張 Zebra cards 存在 (含已被其它 Application 使用中的卡).

\_\_PZEBRA\_BOARD\_INFO\_\_ **getZebraBoardInfo**(\_\_HZHANDLE\_\_ hZhandle, TCHAR  
\*\*pUsbPortName) -

report hZhandle 所指定 manage-queue 中記錄的 Zebra card 部份資訊給 application.

#### 3. 對支援 multi-Zebra cards API 的更動 -

參考 ZebraDllWin32.h (或 ZebraDllWinXP.h 或 ZebraDllWin64.h), 凡 API 函數功能會關係到 Zebra card 的 API, 會多出一個 handle 參數來指明是對那一 Zebra card 下操作命令的. 如下的 :

void CloseUsbDevice(\_\_HZHANDLE\_\_ **hZhandle**);

TCHAR \*getActivePortNname(\_\_HZHANDLE\_\_ **hZhandle**, int \*nPortNumber);

BOOL KickoffRunStep(\_\_HZHANDLE\_\_ **hZhandle**, int iRunId, int ExAct, int nOption);

....

#### 4. Library API 中的 TCHAR 對應到 unicode

5. 目前對 Zebra id 的支援以 USB-port 位置來模擬, 未來可能以 GPIO 在 Zebra 載板上的 dip-switch 設定.

#### 6. multi-Zebra program 啟始步驟/程序:

(1) program 開始之前, 先以 **lookingViewableZebras()** 找出系統上有那些 Zebra cards, 如:

nFoundZebras = **lookingViewableZebras**(); // 建立 Zebra manage-queue list;

(2) 再以 pZebraBoardInfo = **getZebraBoardInfo**((\_\_HZHANDLE\_\_)i, NULL);

查詢 Zebra manage-queue 中, 某一 Zebra card 是否為可以使用狀態

(pZebraBoardInfo->nZebraStatus == \_\_Zebra\_NotYet\_Init\_\_).

(3) 當某一 Zebra card 為可用時 (pZebraBoardInfo->nZebraStatus ==



\_\_Zebra\_NotYet\_Init\_\_), 再以  
nOpenHandle = **bInitUsbDevice**(i, &locTmpZHandle); // open 此 ready 的 Zebra...

如 **bInitUsbDevice** 成功 open 此 Zebra card, 則 nOpenHandle 等同此  
Zebra card 在 manage-queue 中的 index (>= 0);  
若 fail, 則為負值, 見 ZebraDllWinXP.h  
**bInitUsbDevice**() 成功後, 若再以 **getZebraBoardInfo** 查詢 狀態, 會為  
\_\_Zebra\_Open\_usable\_\_.

**未經 bInitUsbDevice() initialized 的 handle(index) 無法通過 USB 控制 Zebra card.**

## 7. 其它修訂

(1) typedef void(\***LOG\_HOOK\_FUNC**)(int hZhandle, char \*pszLogMsg, unsigned long logCbArg);

增加了 hZhandle 及 logCbArg 參數; 用於

```
int LoadRegisterFiles(__HZHANDLE__ hZhandle, char *pRegFilesDir,  
LOG_HOOK_FUNC log_cbFunc, unsigned long logCbArg);
```

```
int SearchingRegisterFiles(__HZHANDLE__ hZhandle, char *pScriptName,  
LOG_HOOK_FUNC log_cbFunc, unsigned long logCbArg);
```

在使用 log\_cb (callback) 可以使 callback function 知道此 callback 來自某一  
Zebra card. (見 ZebraDllWinXP.h.h)

## 2014.07.14 (v1.0.1.0)

主要的變更:

### 1. 主要功能定義的修正

在 PC Host side 支援多張 Zebra cards 可能的方式: (1) 一個 application 可同時控制多張 Zebra cards; (2) 每一 application 只控制一張 Zebra card; 但允許多個 applications 同時執行在 PC 上, 每個 application 可以是相同的, 或不相同.

~~目前 library 支援 (2) 每個 application 只可控制一張 Zebra card 的方式. 見 2014.09.24 修訂.~~  
函數 binitUsbDevice 會 lock/reserve 目前可使用的一張 Zebra card 來讀寫. 可以 Zebra4 或 Zebra5 併用.

函數 getActivePortNname 可以知道連線的 Zebra 在那一 USB controller 的 hub / port.

函數 getZebraStatus 可以返回 USB connection state. 可以 polling 來檢查 Zebra 是否拔除, 插入, ... 等狀況.

### 2. 在此版中參數有所更動修正的 APIs

函數 getZebraStatus 可以返回 USB connection state. 可以 polling 來檢查 Zebra 是否拔除, 插入, ... 等狀況.

其返回值如

```
#define __Zebra_Open_usable__ 1 // a Zebra open by me and usable now  
#define __Zebra_NotYet_Init__ 0 // doesn't init Zebra or a new valid  
// Zebra insert to USB hub-port  
#define __Zebra_Invalid__ -1 // No valid Zebra or they have locked by  
// other applications  
#define __Zebra_Removed__ -2 // my open Zebra has removed by user  
#define __Zebra_WritePipeFail__ -3 // Can't write to USB  
#define __Zebra_ReadPipeFail__ -4 // Can't read from USB
```



當 return value 為 `__Zebra_NotYet_Init__` 需呼叫 `binitUsbDevice` 來 initialize 一 Zebra 給 application 用. Application 可以以此 API 來 polling 查詢 Zebra 卡的狀態; 當 return `0(__Zebra_NotYet_Init__)` 時, 即知有一新 Zebra 接入 PC, 再以 `binitUsbDevice` 來啟用.

3. 在此版中新增加的 APIs

`char *getActivePortNname(int *nPortNumber) -`

When Zebra connected to PC, report which USB port connected it..

4. 在此版中更名的 APIs -

`int QueryDeviceAppState` 更名為 `QueryZebraAppState`.

## 2014.06.03

主要的變更:

1. 主要功能定義的修正:

Zebra 為定義測試 sensor 所定義一序列的設定, 如: i2c address, 圖像大小, 圖像格式, 及 i2c commands 及寫入資料等合稱為 test-step, 保留此原來名稱; 描述此 test-step 的為 test-step file; 在 Test-step file 中含對 Zebra 設定控制的部分可分離為一 setting file; 剩下的 [STREAM\_ON], [STREAM\_OFF] 和 [CONTROL] 也是對 i2c device 做讀寫的動作. 只是格式與下述的 register-file 有所不同.

OVT 原先使用的 i2c 對 i2c device 寫入的 data file 稱為 register-file.

所以一 test-step 可以用一個適合的 setting file 及 register-file 組合 ~ 如此 ovt 原先的 test-script 只要些微的修改即可再使用.

所以此版的 test-step file 及 register file 是有區別的.

2. 新函數功能的改變:

"RunRegisterFile" 由原先處理 test-step file; 改為配合新增的函數 "SetZebraSetting" 來執行所指定的 register file.

即一 "setting file" + "Register file" 也可 load 到 Zebra 上來執行.

"RunTestStepFile" 則用來 執行已 load 到 Zebra 上的 test-step file.

3. Test step configure 中對 register file 的描述方式:

<setting file name> | <register file name>

如: 在 RTXCaptureTestSteps.ZebraCfg 中有如下,

```
//
[Zebra test steps Configure]           // Test comment
-reserved test step                    // 0
Step000.setting.txt | Step000.Register.ovt    // 1
-reserved test step                    // 2
RTXCaptureStep002.txt                  // 3
-reserved test step                    // 4
Step001.setting.txt | Step001.Register.ovt    // 5
-reserved test step                    // 6
```

其中 Step000.setting.txt 及 Step001.setting.txt 為 setting file.

Step000.Register.ovt 及 Step001.Register.ovt 為 register-file.

4. 在此版中新增加的 APIs -

`void SetZebraSetting(char *pZebraSetting)`

- setup a Zebra setting file name and use this name to matching a register-file as test-step when download phase.



```
int RunRegisterFile(char *pRegisterName);  
- "getRunStepIdFromDownloadRegisterStepList" called internally to get test-step id  
and then calls to "KickoffRunStep".  
- this name should be register-file name; before this function valid, a useful  
setting file should be assigned by SetZebraSetting.
```

```
int RunTestStepFile(char *pTestStepName);  
- "getRunStepIdFromDownloadRegisterStepList" called internally to get test-step id  
and then calls to "KickoffRunStep".
```

#### 5. Zebra 對 register-file 中的支援格式 -

I2c register file format

Write format:

<i2c device address> <sub-address(offset) 1 byte or 2 bytes> <data bytes>

examples:

```
6c      0103  01
```

```
6c      : i2c device address (includes write flag),  
0103    : 2 bytes sub-address  
01      : 1 byte data to be write
```

Read format:

<i2c device address> <how many bytes to be read> <sub-address(offset) 1 byte  
or 2 bytes>

example 1:

```
6d      1      0120
```

```
6d      : i2c device address (includes read flag)  
1       : 1 byte data be read  
0120    : 2 byte sub-address
```

example 2:

```
6d      2
```

```
6d      : i2c device address (includes read flag)  
2       : 2 bytes data be read
```

## 2014.05.16

主要的變更:

#### 1. 新函數功能的改變:

"KickoffRunStep", "RunStepByAliasName" and "RunRegisterFile" 函數發送抓圖 command 到 Zebra, Zebra 執行抓圖的功能, Zebra 執行 command 後, 只返回(送出) run-step 狀態後, 不再直接繼續送出 image data. (之前的版本會再送 image data)

Application 可使用 "GrabImageRequest" API 要求 Zebra, 通知 Zebra 送出 Image data; 再在 PC host 以 "getPicture2File" or "getPicture2Buffer" 收取 image data.

在以 "KickoffRunStep", "RunStepByAliasName" or "RunRegisterFile" 通知 Zebra 執行一 test step 或 register file 後, application 可使用 "getRunStepstate" 查詢抓取 image 的狀態. 當所要抓取的 image 已 ready, 再以 "GrabImageRequest" 命令通知 Zebra 送出 image data, 並以 "getPicture2File" or "getPicture2File" 收取.

另一種取圖的方式為: 在呼叫 KickoffRunStep", "RunStepByAliasName" or "RunRegisterFile" APIs 後, 重複以 "GrabImageRequest" 及 "getPicture2File" (or "getPicture2File") 收取



image data; 並檢查 "getPicture2File" (or "getPicture2File") 是否取到 data; 若 return length 為 0, 則檢視 function report 的 error code. 見 "getPicture2File" (or "getPicture2File") 的定義.

2. Test step configuration 方式改變, 在 configuration file 中 test setp (register file) 設定方式修正如下:

改以直接將 test step (register file) 的 text file 名稱列寫於 configuration file 中. 之前 "maximum steps" 方式不再使用.

~~"Register file" 等同 "test step file" 見 2014.06.03 版的修正.~~

在新 library header 中增加的 prototype definition -

```
typedef void(*LOG_HOOK_FUNC)(char *);
```

用以 call back 輸出 download 到 Zebra 中的 step files name; 或在一 script 中找尋到的 register files; 並記錄某些 application 可能想獲得的資訊.

在此版 library 中將不再支援的 APIs

int DownloadTestProcedures(char \*TestSetName) 不再有效,  
請以 "LoadRegisterFiles" 函數來代替.

在此版中參數有所更動修正的 APIs -

```
int getPicture2File(char *pictureFileName); 及  
ULONG getPicture2Buffer(unsigned char *pPicBuffer, ULONG pPicBufSiz);
```

增加一 Argument 如下,

```
int getPicture2File(char *pictureFileName, UCHAR *errCode); 及  
ULONG getPicture2Buffer(unsigned char *pPicBuffer, ULONG pPicBufSiz, UCHAR  
*errCode);
```

都增加一 errCode 參數,

- 在 Zebra 收到 "GrabImageRequest" 後, Zebra 送出 image lengrh 及 image data 或在 image 未 ready 時, 送出 length 0 及 error code 以表示何種錯誤.
- "errCode" - \_\_Error\_None\_\_, \_\_Error\_No\_image\_\_, \_\_Error\_snapping\_\_,  
\_\_Error\_start\_index\_\_, \_\_Error\_end\_index\_\_, \_\_Error\_opCode\_\_.

在此版中新增加的 APIs -

```
int LoadRegisterFiles(char *pRegFilesDir, LOG_HOOK_FUNC log_cbFunc);  
- this function replaces old DownloadTestProcedures function. A callback entry to  
print out which register-files (test step files) are being download.
```

```
int GrabImageRequest(short imgStartIdx, short imgEndIdx, unsigned char  
imgGotOpCode, unsigned char bWaitSnapped);  
- Send a low level command to Zebra let it reports assigned image data. If more  
than one picture been requested, multiples calls of "getPicture2File" or  
"getPicture2Buffer" are need.
```

```
int SearchingRegisterFiles(char *pScriptName, LOG_HOOK_FUNC log_cbFunc);  
- Search a script file and cache the register-files name. When next  
"LoadRegisterFiles" calls with NULL arguments, these cached register-files would be  
download to Zebra.
```

```
int RunStepByAliasName (char *pAliasStepName);
```



- "getRunStepIdFromDownloadAliasList" called internally to get test-step id and then calls to "KickoffRunStep".

```
int getRegisterStepsCntJustDownload(void);
```

- reports how many test-step (register) files been downloaded.

```
char *get1stDownloadRegisterStepName(void);
```

- get file name that was 1st downloaded test-step (register) file.

```
char *getNextDownloadRegisterStepName(void);
```

- get next test step file name that was downloaded test-step (register) file.

```
int getRunStepIdFromDownloadRegisterStepList(char *pRegisterStepName);
```

- from a downloaded test step (register) file name figure out its test step id. This id is acceptable test step on Zebra.

```
int RunRegisterFile(char *pRegisterStepName);
```

- "getRunStepIdFromDownloadRegisterStepList" called internally to get test-step id and then calls to "KickoffRunStep".

```
int getHardwareType (void);
```

- to identify Zebra4 or Zebra5





## Zebra API 的版本

因應不同的 Window 版本及相容等因數, Zebra API 含有 32, 64 及 XP32 等三個版本. 其 API 在每個版本都相同, 所以以下只以 Win32 為例.

對應這三個 DLL 版本, 各有其相應的 Demo Application, 其在目前 Windows OS 本版下測試結果如以下列表.

|                                    | XP-32 SP2 | WIN7-32 | Win7-64 | Win8/8.1-32 | Win8/8.1-64 |
|------------------------------------|-----------|---------|---------|-------------|-------------|
| ZebraDllWin32 + Win32 APP Program  | X         | O       | O       | O           | O           |
| ZebraDllWin64 + Win64 APP Program  | X         | X       | O       | X           | O           |
| RTXCaptureDemoTool.exe (32 bits)   | X         | O       | O       | O           | O           |
| RTXCaptureDemoTool64.exe (64 bits) | X         | X       | O       | X           | O           |
| ZebraDllWinXP + WinXp APP Program  | O         | O       | O       | O           | O           |

O: OS is supported.

X: OS is not supported.

本份文件同時適用於和瑞科技所生產之 Zebra4 以及 Zebra5 MIPI CSI-2 高速 capture card 產品系列.



## Zebra win32 API 應含檔案

mipiCntrlDef.h, ZebraDllWin32.h, ZebraDllWin32.DLL, ZebraDllWin32.lib

## Zebra win32 API 說明

請對照參考 ZebraDllWin32.h, ZebraDllWin64.h or ZebraDllWinXP.h

### **InitMipiUsbNotifyWin**

啟始 background thread 監控 PC 上所有 Zebra cards 的插拔情形, 維護 manage-queue list 中每張 Zebras card 的插拔狀態(不含被其它 Application 開啟或關閉的狀態); 建立 window message 或 thread message call-back function; 當 Zebra USB 連接到 PC 時, 產生 WM\_MIPI\_DEVICEARRIVAL 到相關的 window 或 thread; 當 USB 失連時則送出 WM\_MIPI\_DEVICEREMOVECOMPLETE.

```
int InitMipiUsbNotifyWin(HWND hWndApp, DWORD idThrdHandle);
```

hWndApp: A window handle to receive WM\_MIPI\_DEVICEARRIVAL or WM\_MIPI\_DEVICEREMOVECOMPLETE call back message.

idThrdHandle: A thread id to receive WM\_MIPI\_DEVICEARRIVAL or WM\_MIPI\_DEVICEREMOVECOMPLETE call back message.

回傳值 1: succeed, 0: fail.

### **StopMipiUsbNotifyWin**

停止 InitMipiUsbNotifyWin 為 Zebra notification 所啟動的 background thread, 取消所建立的 message call-back.

```
void StopMipiUsbNotifyWin(void);
```

回傳值: 無.

### **lookingViewableZebras**

查詢 PC 上有多少 zebra cards, 建立此 Zebras cards 的 manage-queue list.

(InitMipiUsbNotifyWin () 可發動 USB notification 在 background 維護此 manage-queue)

```
int lookingViewableZebras(void);
```

回傳值: 在此 PC 上有多少 Zebra cards.



## **freeZebrasInfoListQueue**

program 結束時, free zebra cards 的 manage-queue.

```
void freeZebrasInfoListQueue(void);
```

回傳值: 無.

## **ZebraCardNumber**

report 目前的 manage-queue 中的 zebras cards 記錄筆數,即有多少張 Zebra cards 存在 (含已被其它 Application 使用中的卡).

```
int ZebraCardNumber(void);
```

回傳值: 在此 PC 上有多少 Zebra cards.

## **getZebraBoardInfo**

report 指定的 Zebra card 的部份資訊給 application.

```
__PZEBRA_BOARD_INFO__ getZebraBoardInfo(__HZHANDLE__ hZhandle, TCHAR  
**pUsbPortName);
```

hZhandle - 被查詢的 Zebra card 在 manage-queue list 中的 index (此 index 可不需經  
bInitUsbDevice() initialize).

\*\*pUsbPortName - 傳回此 Zebra card 所在 USB port 的名稱.

回傳值: 指向 Zebra card 的部份資訊,如 Zebra card Id, type, 及 status,  
如指定的 index 不存在,傳回 NULL.

## **QueryZebraAppState**

在 Zebra card 上有相關的 application 負責整合 mipi, capture function, I/o... USB 等相關功能。此 API 取得此 Zebra Application 是否已啟動的資訊。

```
int QueryZebraAppState(__HZHANDLE__ hZhandle);
```

hZhandle - 被查詢的 Zebra card 在 manage-queue list 中的 index (此 index 需經  
bInitUsbDevice() initialized).

回傳值 -1: 無法以 USB 取得 Zebra Application 的狀態.  
-2: USB can't return data to indicate this application state  
-3: USB driver is ready, but application isn't start  
-4: Zebra handle illegal  
1: Zebra application has attached on zebra USB.



## **bInitUsbDevice**

Open Zebra USB device and Zebra application to set/get Zebra setting or get picture from Zebra.

所有用到 hZhandle 的 API 應在此 function 成功後才能使用 (getZebraBoardInfo, getHardwareType 及 USB reset Zebra 為特例)。

```
int bInitUsbDevice(int nZebraIndexInQueue, __HZHANDLE__ *phZhandle);
```

nZebraIndexInQueue - Zebra card 在 manage-queue list 中的 index.

\*phZhandle - 傳回 open 的 index, 與 return handle 相同.

回傳值 -1: can't open this device  
-2: can't get USB descriptor.  
-3: can't retrieve mipi pipi (end points)  
-4: can't get USB i2c interface.  
-5: can't retrieve i2c pipi.  
-6: can't get USB uart interface.  
-7: can't retrieve uart pipi.  
-8: can't find zebra from assigned board id(index - nZebraIndexInQueue).  
-9: locked by other process  
\_\_USB\_DRV\_OK\_APP\_NOT\_READY\_\_:  
USB driver ready, but application isn't ready; 此狀態支援 USB reset Zebra 功能.  
0, 1, ... : hZhandle, 此時可由 Zebra USB 或 Zebra Application reset Zebra.

!!!若要對特定的 ID 做 open, 則先以 getZebraBoardInfo() 查知 id 後,再行呼叫 bInitUsbDevice().

## **CloseUsbDevice**

Close Zebra USB device that enabled by bInitUsbDevice function.

```
void CloseUsbDevice(__HZHANDLE__ hZhandle);
```

hZhandle - 要 close 的 Zebra card 在 manage-queue list 中的 index (此 index 需經 bInitUsbDevice() initialized).

回傳值: 無.

!!! Application program 結束時,應對 application 本身所 open 所有的 Zebra cards 做個別的 CloseUsbDevice 呼叫.

## **getActivePortNname**

when Zebra connected to PC, report which USB port connected it.

```
TCHAR *getActivePortNname(__HZHANDLE__ hZhandle, int *nPortNumber);
```

hZhandle - 要查詢的 Zebra card 在 manage-queue list 中的 index (此 index 需經 bInitUsbDevice() initialized).

\*nPortNumber - 傳回此 Zebra card 的 ID, 不需要時可設為 NULL.

回傳值: Port Name or NULL.



```
/*
Zebra test step setting is consisted by a test configuration file and some test
step files.
Test step setting could be downloaded to Zebra application then tester host could
run it of later.
The configuration file is a text file, that includes all test-step files (register
file) name with text string;
test step files (register-files) are text files also; that includes specified key-
word to assign I2C read/write register, data,
clock setting, reset pin condition...

for example:
The user selects a test step configuration that named
"RTXCaptureTestSteps.ZebraCfg"; API "LoadRegisterFiles" parses this file and gets
all test-step files name that would be downloaded to Zebra. This
"LoadRegisterFiles" also open these test-step files and then downloads their
setting content to Zebra.
*/
```

### **quireyZebraTestStepCount**

Get the test step's count that has downloaded to Zebra application.

```
int quireyZebraTestStepCount(__HZHANDLE__ hZhandle);
```

hZhandle - 要查詢的 Zebra card 在 manage-queue list 中的 index (此 index 需經  
bInitUsbDevice() initialized).

回傳值:

-1 : 指定的 hZhandle 不存在,  
其它 >= 0 的值, 即在 Zebra application 上設定的 step count. "test-step" 設定見  
RTXCaptureDemoTool.doc 或 MultiZebrasDemo.doc (.pdf) 範例.

```
#define __OPT_STREAMING_OFF__      0          // a picture captured by mipi when
// a KickoffRunStep called.
#define __OPT_STREAMING_ON__      1          // pictures captured by mipi after
// a KickoffRunStep called.
```

### **KickoffRunStep**

Run a test step that has downloaded to Zebra application.

```
BOOL KickoffRunStep(__HZHANDLE__ hZhandle, int iRunId, int ExAct, int nOption);
```

hZhandle - 要執行的 Zebra card 在 manage-queue list 中的 index (此 index 需經  
bInitUsbDevice() initialized).

iRunId - Test step id; that base 0. 由 quireyZebraTestStepCount 之回傳值 -1 為最大 id.

ExAct - reserve for future

nOption- \_\_OPT\_STREAMING\_OFF\_\_ or \_\_OPT\_STREAMING\_ON\_\_

A run state would be got after this API, but without image data been return; image  
is gotten by "GrabImageRequest" API.



回傳值：

\_\_USB\_HANDLE\_NOT\_FOUND\_\_: hZhandle not found in Zebra manage-queue list;  
TRUE 傳送成功, FALSE 失敗.

## getRunStepstate

詢問之前 run step CSI 在 Zebra card 上執行抓圖的狀態；在 KichoooffRunStep/RunStepByAliasName/ RunRegisterFile 之後執行。會很快返回 run step 目前的處理狀態。

```
int getRunStepstate(__HZHANDLE__ hZhandle, PSRUNSTEPSTATE pRunStepState);
```

hZhandle - 被詢問的 Zebra card 在 manage-queue list 中的 index (此 index 需經 bInitUsbDevice() initialized).

pRunStepState - a pointer of SRUNSTEPSTATE structure to report just run step result.

在 pRunStepState 內：

.runResultState - executed run step status

.resultType - one of \_\_noMoreData\_\_, \_\_SnapPicture\_\_, and \_\_AppendData\_\_ union:

.snapCnt - snap picture's count when .resultType is \_\_SnapPicture\_\_

.pSnaPicSiz - every picture size of snapped; maximum 128 items.

.appDataLen - a append user data length when .resultType is \_\_AppendData\_\_

.appendBuffer - a append user data buffer, maximum 255.

- if .runResultState reports "\_\_Sta\_step\_snapping\_\_", it implies pictures are snapping now.

- 處理的程式 virtual code 如下：

```
if (.runResultState == __Sta_step_succeed__)
{
    if (.resultType == __AppendData__) // only test data, no snapping images
    {
        // handle text data ....
    }
    else if (.resultType == __SnapPicture__)
    {
        // data image data has ready,
        //
        // call "GrabImageRequest" then "getPicture2File" or
        // "getPicture2Buffer" to received data
        ...
    }
    else
    {
    }
}
else if (.runResultState == __Sta_step_snapping__)
{
    if (.resultType == __AppendData__ && .appDataLen == 2)
```



```
{
    .appendBuffer[0] :: how many needs to snap
    .appendBuffer[1] :: how many pictures has snapped
    .....
}
else
{
    // other errors ...
}
}
else if (.runResultState == __Sta_None_Imgage__)
{
    // handle none image ....
}
else if (.runResultState == __Sta_Err_I2cWrite__)
{
    // read/write register i2c error ...
}
else if (.runResultState == __Sta_Invalid__)
{
    // handle ....
}
}
```

回傳值：

\_\_USB\_HANDLE\_NOT\_FOUND\_\_: hZhandle not found in Zebra manage-queue list;  
-1: USB access error,  
TRUE succeeds; else fail.

## **GrabImageRequest**

Issue a fetching image command to Zebra; the images could be a previous snapped; then next function 'getPicture2File' or 'getPicture2Buffer' could get the image data.

```
int GrabImageRequest(__HZHANDLE__ hZhandle, short imgStartIdx, short imgEndIdx,
unsigned char imgGotOpCode, unsigned char bWaitSnapped);
```

hZhandle - 要取圖的 Zebra card 在 manage-queue list 中的 index (此index 需經  
bInitUsbDevice() initialized).

imgStartIdx - Which index would be start getting (base 0).

imgEndIdx - ending index that is last getting.

imgGotOpCode - one of getting image operation; it could be

\_\_Grab\_opt\_noneOperation\_\_, \_\_Grab\_opt\_ImageAvg\_\_, ...

bWaitSnapped - when run step is snapping to wait it completed then report grabs  
image.

回傳值：

\_\_USB\_HANDLE\_NOT\_FOUND\_\_: hZhandle not found in Zebra manage-queue list;  
TRUE succeeds; else fails.



### **resetZebraFromApp**

Reset zebra from zebra application .

```
int resetZebraFromApp(__HZHANDLE__ hZhandle, int *resultState);
```

hZhandle - 要 reset 的 Zebra card 在 manage-queue list 中的 index (此 index 需經  
bInitUsbDevice() initialized).  
resultState - return this reset command status from device

回傳值:

\_\_USB\_HANDLE\_NOT\_FOUND\_\_: hZhandle not found in Zebra manage-queue list;  
-1, 0: USB fail, can't send reset command.  
1: command send succeed.

### **resetZebraByUSBDev**

Reset zebra from zebra USB device.

```
void resetZebraByUSBDev(__HZHANDLE__ hZhandle);
```

hZhandle - 要 reset 的 Zebra card 在 manage-queue list 中的 index (此 index 需經  
bInitUsbDevice() initialized).

回傳值:無.





```
#define __Zebra_Open_usable__      2      // a Zebra open by me and usable
                                     now
#define __Zebra_Open_DrvNoApp__    1      // a Zebra open but application
                                     (sensor) not ready
#define __Zebra_NotYet_Init__      0      // doesn't init Zebra or a new valid
                                     Zebra insert to USB hub-port
#define __Zebra_InValid__          -1     // No valid Zebra or they have locked by
                                     other applications
#define __Zebra_Removed__          -2     // that open Zebra has removed by user
#define __Zebra_WritePipeFail__    -3     // Can't write
#define __Zebra_ReadPipeFail__     -4     // Can't read
```

### **getZebraStatus**

return zebra application status.

```
int getZebraStatus(__HZHANDLE__ hZhandle, PSZEBRASSTATUS presultStatus);
```

hZhandle - 要查詢的 Zebra card 在 manage-queue list 中的 index (此index 需經  
bInitUsbDevice() initialized).  
presultStatus - a pointer of SZEBRASSTATUS to report got application status.  
.zebraState - 傳回 zebra application status.

回傳值:

\_\_USB\_HANDLE\_NOT\_FOUND\_\_: hZhandle not found in Zebra manage-queue list;  
-4, -3, ..., 0: USB fail condition, can't send reset command.  
1: command send succeed.

### **getZebraSWVersion**

Return zebra application software version.

```
int getZebraSWVersion(__HZHANDLE__ hZhandle, PSZEBRAVERSION psZebraVersion);
```

hZhandle - 要查詢的 Zebra card 在 manage-queue list 中的 index (此index 需經  
bInitUsbDevice() initialized).  
psZebraVersion - a pointer of SZEBRAVERSION to report software version  
.zebraMajorVer -  
.zebraMinorVer -  
.zebraBuildNumber -  
.zebraRevisionNumber -

回傳值:

\_\_USB\_HANDLE\_NOT\_FOUND\_\_: hZhandle not found in Zebra manage-queue list;  
-1, 0: USB fail, can't send reset command.  
1: command send succeed.



```
/*
Zebra test step setting is consisted by a test configuration file and some test
step files.
Test step setting could be downloaded to Zebra application then tester host could
run it of later.
The configuration file is a text file, that includes all test-step files (register
file) name with text string;
test step files (register-files) are text files also; that includes specified key-
word to assign I2C read/write register, data,
clock setting, reset pin condition...

for example:
The user selects a test step configuration that named
"RTXCaptureTestSteps.ZebraCfg"; API "LoadRegisterFiles" parses this file and gets
all test-step files name that would be downloaded to Zebra. This
"LoadRegisterFiles" also open these test-step files and then downloads their
setting content to Zebra.
*/
```

## **LoadRegisterFiles**

Download a test setting/test step/register files to Zebra.  
Function get these download files name from configuration file then download them  
to Zebra.

```
int LoadRegisterFiles(__HZHANDLE__ hZhandle, char *pRegFilesDir, LOG_HOOK_FUNC
log_cbFunc, unsigned long logCbArg);
```

hZhandle - 要下載的 Zebra card 在 manage-queue list 中的 index (此index 需經  
bInitUsbDevice() initialized).  
pRegFilesDir - a file name (configuration file) that registers all test steps  
files; when this arg is NULL, function downloads all files that found by  
SearchingRegisterFiles(). This condition is used in script operation.  
log\_cbFunc - a call back function used to report register(test step) file name  
that has been downloaded to Zebra. .  
LogCbArg - A argument from application passes to callback function, that could be  
any meaning value by application. 由 application 自行定義的任何值, 用於 call-back  
function log\_cbFunc 中.

回傳值:

\_\_USB\_HANDLE\_NOT\_FOUND\_\_: hZhandle not found in Zebra manage-queue list;  
或 0.

## **SearchingRegisterFiles**

Collect all being loaded test step file (register file) names that is used in a  
test script.

```
int SearchingRegisterFiles(__HZHANDLE__ hZhandle, char *pScriptName, LOG_HOOK_FUNC
log_cbFunc, unsigned long logCbArg);
```

hZhandle - 要搜尋的 Zebra card 在 manage-queue list 中的 index (此index 需經  
bInitUsbDevice() initialized).  
pScriptName - a file name that points to a test script.



log\_cbFunc - a call back function used to report register(test step) file name that has been found in script.  
LogCbArg - A argument from application passes to callback function, that could be any meaning value by application. 由 application 自行定義的任何值，用於 call-back function log\_cbFunc 中。

回傳值：

\_\_USB\_HANDLE\_NOT\_FOUND\_\_: hZhandle not found in Zebra manage-queue list;  
-1: Can't open script file  
    - if this function searching a open script file; please make sure script file sharable open to read.  
n: count of found test steps (register files)

### **getCaptureCardTestProcedures**

Get the test setting from Zebra application and save them as a configure file and test step files.

```
int getCaptureCardTestProcedures(__HZHANDLE__ hZhandle, char *szGotSettingName, char *pCardName, char *pSensorName, char *pSWVersion);
```

hZhandle - 要讀取的 Zebra card 在 manage-queue list 中的 index (此index 需經 bInitUsbDevice() initialized).  
szGotSettingName - user assigned this test set name, to save the test setting, API appends "SysCfg.got.ZebraCfg" as it configuration file. And appends "Step%03d.got.txt" as test step file names.  
pCardName - card name information in test set configure and step file.  
pSensorName - capture sensor information in test set configure and step file.  
pSWVersion - current software version in test set configure and step file.

回傳值：

\_\_USB\_HANDLE\_NOT\_FOUND\_\_: hZhandle not found in Zebra manage-queue list;  
0: 正常返回。

### **createCaptureTestingTemplate**

Create test set template(test configure and test step files) in local directory.

```
void createCaptureTestingTemplate(char *pCardName, char *pSensorName, char *pSWVersion, char *pTstSetName, int testSteps, ...);
```

pCardName - card name information in test set configure and step file.  
pSensorName - capture sensor information in test set configure and step file.  
pSWVersion - current software version in test set configure and step file.  
pTstSetName - test setting name for to be created, API appends "SysCfg.Tmp.ZebraCfg" as it configuration file.  
    and appends "Step%03d.Tmp.txt" as test step file names.  
testSteps - how many test steps in this test set.

回傳值：無



## **getPicture2File**

Get a picture data from Zebra application and save it to file; this function is call after GrabImageRequest().

```
int getPicture2File(__HZHANDLE__ hZhandle, char *pictureFileName, UCHAR *errCode);
```

hZhandle - 要取圖的 Zebra card 在 manage-queue list 中的 index (此index 需經 bInitUsbDevice() initialized).

pictureFileName - save name.

errCode - when image is snapping and report some error code; that is one of  
\_\_Error\_None\_\_, \_\_Error\_No\_image\_\_, \_\_Error\_snapping\_\_

回傳值

\_\_USB\_HANDLE\_NOT\_FOUND\_\_: hZhandle not found in Zebra manage-queue list;  
-1: can't save to file.  
-2: can't read picture size or data from USB.  
-3: read a 0 byte data multiple times.  
0: succeed.

## **getPicture2Buffer**

Get a picture data from Zebra application; this function is call after GrabImageRequest().

```
ULONG getPicture2Buffer(__HZHANDLE__ hZhandle, unsigned char *pPicBuffer, ULONG pPicBufSiz, UCHAR *errCode);
```

hZhandle - 要取圖的 Zebra card 在 manage-queue list 中的 index (此index 需經 bInitUsbDevice() initialized).

pPicBuffer - a buffer pointer points to a buffer to get picture.

pPicBufSiz - the buffer size of pPicBuffer, it could get from run

"getRunStepstate(...)" function.

errCode - when image is snapping and report some error code; that is one of

\_\_Error\_None\_\_, \_\_Error\_No\_image\_\_, \_\_Error\_snapping\_\_,  
\_\_Error\_getPicBufferTooShort(the data buffer size too short than actual image  
)

回傳值:

\_\_USB\_HANDLE\_NOT\_FOUND\_\_: hZhandle not found in Zebra manage-queue list;  
0: zebra mipi can't get image.  
-2: can't read picture from USB.  
-3: read a 0 byte data multiple times.  
> 0: got image length, if needs, please check errCode for  
\_\_Error\_getPicBufferTooShort

the size of picture that got; when return value is 0, implies an error happen, please check the errCode condition.



```
/*
I2C configure byte::
    b0: (0) - 7 bits address, (1) - 10 bits address
    b2~b1: sub-address when read
        00: No sub address
        01: 1 byte
        10: 2 bytes
        11: 3 bytes
    b3: Ack ignore
    b4~b6: reserve
    b7: read(1), write(0)
*/
```

```
#define _10BitI2C_Addr 0x1
#define _1byteRdSubAddr 0x2
#define _2bytesRdSubAddr 0x4
#define _3bytesRdSubAddr 0x6
#define _IgnoreAckChking 0x8
#define _ReadI2cDevice 0x80
```

## **I2cRawDataWrite**

Write data present by string to i2c device; this function would decode this text string to binary.

```
int I2cRawDataWrite(__HZHANDLE__ hZhandle, UCHAR cfgByte, USHORT i2cAddr, char
*pTxtWrData, int txtLen);
```

hZhandle - 要讀寫的 Zebra card 在 manage-queue list 中的 index (此index 需經  
bInitUsbDevice() initialized).  
cfgByte - I2C configure information  
i2cAddr - I2C device address  
pTxtWrData - text string of write data, sepearated by ',' - example: "0x1, 2, 0x3,  
0x4, 5"  
txtLen - the test string data length

回傳值:

```
__USB_HANDLE_NOT_FOUND__: hZhandle not found in Zebra manage-queue list;
-1: USB can't write,
-2: I2C write error.
-3: not all data write out.
0: write succeed.
```

## **I2cRawDataWrite2**

Write binary data to i2c device

```
int I2cRawDataWrite2(__HZHANDLE__ hZhandle, UCHAR cfgByte, USHORT i2cAddr, UCHAR
*pWrData, int WrDataLen);
```

hZhandle - 要讀寫的 Zebra card 在 manage-queue list 中的 index (此index 需經



bInitUsbDevice() initialized).  
cfgByte - I2C configure information  
i2cAddr - I2C device address  
pWrData - write binary data buffer.  
WrDataLen - how many bytes data to write.

回傳值:

\_\_USB\_HANDLE\_NOT\_FOUND\_\_: hZhandle not found in Zebra manage-queue list;  
-1: USB can't write,  
-2: I2C write error.  
-3: not all data write out.  
0: write succeed.

## **I2cRawDataRead**

read data from i2c device.

```
int I2cRawDataRead(__HZHANDLE__ hZhandle, UCHAR cfgByte, USHORT i2cAddr, char  
*pTxtSubAddr, int nReadLen, UCHAR *pReadBuffer);
```

hZhandle - 要讀寫的 Zebra card 在 manage-queue list 中的 index (此index 需經  
bInitUsbDevice() initialized).  
cfgByte - I2C configure information  
i2cAddr - I2C device address  
pTxtSubAddr - some i2c device has its sub-address, that here present by text  
string, it configure by cfgByte.  
nReadLen - how many data would be read.  
pReadBuffer - read buffer.

回傳值:

\_\_USB\_HANDLE\_NOT\_FOUND\_\_: hZhandle not found in Zebra manage-queue list;  
-1: USB can't write,  
-2: I2C write error.  
-3: not all data write out.  
-4: USB can't read.  
n: Read length.

## **I2cRawDataRead2**

read data from i2c device.

```
int I2cRawDataRead2(__HZHANDLE__ hZhandle, UCHAR cfgByte, USHORT i2cAddr, UCHAR  
*pSubAddr, int nReadLen, UCHAR *pReadBuffer);
```

hZhandle - 要讀寫的 Zebra card 在 manage-queue list 中的 index (此index 需經  
bInitUsbDevice() initialized).  
cfgByte - I2C configure information  
i2cAddr - I2C device address  
pSubAddr - some i2c device has its sub-address, it configure by cfgByte.  
nReadLen - how many data would be read.



pReadBuffer - read buffer.

回傳值:

\_\_USB\_HANDLE\_NOT\_FOUND\_\_: hZhandle not found in Zebra manage-queue list;  
-1: USB can't write,  
-2: I2C write error.  
-3: not all data write out.  
-4: USB can't read.  
n: Read length.



對每一個 test step (register file) 如果只能由它的 index 來執行, 而沒有一個較為有意義的名稱來記憶, 可能對使用者造成不小的負擔; 所以 Zebra Library 在 test step 檔案中提供一 \$ALIAS 功能, 允許使用者對每一 test step 可以定義自己方便的名稱. 當沒有指定此 \$ALIAS 時, 以 step file name 為 alias name.

另在 140516 以後的版本也提供以 test-step file name (register file name) 來辨別 test step.

/\*

Every test step file could include a special key "\$ALIAS" to assigned a user readable name for indicate current step purpose in testing.  
For example:

RtxCaptureStep001.txt includes as:

...  
\$ALIAS = Test UVL function  
...

The "Test UVL function" could be used as a name of this test step when testing; that is meaning than the RtxCaptureStep001.txt.

When \$ALIAS isn't assigned in test step file; the file name is default \$ALIAS name.

The maximum ALIAS name is less than 64 characters.  
This ALIAS name is built up after download test step operation; and could access by following functions.

Any readable text file name could be used as test step file;

Some functions support using file name as test-step (register file) is ready.

\*/

### ***getMaxAliasRunSteps2AllowDownload***

return the maximum test steps count in alias list; that would be same as the maximum test steps configured in system-configured file.

int getMaxAliasRunSteps2AllowDownload(\_\_HZHANDLE\_\_ hZhandle);

hZhandle - 取得下載 test-step count 的 Zebra card 在 manage-queue list 中的 index (此 index 需經 bInitUsbDevice() initialized).

回傳值:

\_\_USB\_HANDLE\_NOT\_FOUND\_\_: hZhandle not found in Zebra manage-queue list;  
或  
最大可能的 test steps count.

### ***getAliasRunStepsCntJustDownload***

return the actually download to Zebra test steps count; that would be less than the test steps configured in configuration file.





```
int getAliasRunStepsCntJustDownload(__HZHANDLE__ hZhandle);
```

hZhandle - 取得下載 test-step count 的 Zebra card 在 manage-queue list 中的 index  
(此 index 需經 bInitUsbDevice() initialized).

回傳值:

\_\_USB\_HANDLE\_NOT\_FOUND\_\_: hZhandle not found in Zebra manage-queue list;  
或

目前真正 download 到 Zebra 的 test steps count; 有些 test steps 可能沒設定.

### **get1stDownloadStepAliasName**

point to the 1st download alias name on alias list.

```
char *get1stDownloadStepAliasName(__HZHANDLE__ hZhandle);
```

hZhandle - 下載 test-steps 的 Zebra card 在 manage-queue list 中的 index  
(此 index 需經 bInitUsbDevice() initialized).

回傳值: 取得 alias list 的第一個 alias name; 當沒有任何 test step downloaded 到 Zebra 時,  
回傳 NULL.

### **getNextDownloadStepAliasName**

point to the next download alias name from just got.

```
char *getNextDownloadStepAliasName(__HZHANDLE__ hZhandle);
```

hZhandle - 下載 test-steps 的 Zebra card 在 manage-queue list 中的 index  
(此 index 需經 bInitUsbDevice() initialized).

回傳值: 由前一個 alias name 後, 取其後的 alias name; 當已經是最後一個 test step 時, 回傳 NULL.

### **getRunStepIdFromDownloadList**

Get a test step id from an assigned test step alias name; this id could be used by  
"KickoffRunStep(...)" function to kick off Zebra do testing.

```
int getRunStepIdFromDownloadList(__HZHANDLE__ hZhandle, char *pAliasStepName);
```

hZhandle - 讀取 test-step id 的 Zebra card 在 manage-queue list 中的 index  
(此 index 需經 bInitUsbDevice() initialized).

\*pAliasStepName: 指定要取得 ID 的 test-step alias name.

回傳值:

\_\_USB\_HANDLE\_NOT\_FOUND\_\_: hZhandle not found in Zebra manage-queue list;  
或

由一 alias name 取得一 test step id; 此 test step id 可以傳給 Zebra, 做為執行那一  
test step 的依據. 當所指定的 alias name 不存在 list 中時, 傳回 -1.



## **RunStepByAliasName**

kick off a test step by alias name; this function default to call  
getRunStepIdFromDownloadAliasList() and then KickoffRunStep() function to run a  
test step.

```
int RunStepByAliasName (__HZHANDLE__ hZhandle, char *pAliasStepName, int nOption);
```

hZhandle - 要執行 test-step 的 Zebra card 在 manage-queue list 中的 index  
(此 index 需經 bInitUsbDevice() initialized).

\*pAliasStepName: 要執行的 test-step 所代表的 alias name.

nOption- \_\_OPT\_STREAMING\_OFF\_\_ or \_\_OPT\_STREAMING\_ON\_\_

回傳值:

\_\_USB\_HANDLE\_NOT\_FOUND\_\_: hZhandle not found in Zebra manage-queue list;

-1: 當所指定的 alias name 不存在 alias name list 中時, 傳回-1.

1: succeed,

0: Fail to run alias name.

## **getRegisterStepsCntJustDownload**

Reports the actually download to Zebra test steps count; that could be less than  
the test steps configured in configuration file.

```
int getRegisterStepsCntJustDownload (__HZHANDLE__ hZhandle);
```

hZhandle - 要取得下載 test-step count 的 Zebra card 在 manage-queue list 中的 index  
(此 index 需經 bInitUsbDevice() initialized).

回傳值:

\_\_USB\_HANDLE\_NOT\_FOUND\_\_: hZhandle not found in Zebra manage-queue list;  
或

目前 download 到 Zebra 的 test steps count.

## **get1stDownloadRegisterStepName**

get file name that was 1st downloaded test-step (register) file.

```
char *get1stDownloadRegisterStepName (__HZHANDLE__ hZhandle);
```

hZhandle - 下載 test-steps 的 Zebra card 在 manage-queue list 中的 index  
(此 index 需經 bInitUsbDevice() initialized).

回傳值: 取得 downloaded test-step/register files list 的第一個 file name; 當沒有任何  
test step downloaded 到 Zebra 時, 回傳 NULL.



### **getNextDownloadRegisterStepName**

et next test step file name that was downloaded test-step (register) file.

```
char *getNextDownloadRegisterStepName(__HZHANDLE__ hZhandle);
```

hZhandle - 下載 test-steps 的 Zebra card 在 manage-queue list 中的 index  
(此 index 需經 bInitUsbDevice() initialized).

回傳值：由前一個 test step file (register file) name 後，取其後的 test step file name；  
當已經是最後一個 test step 時，回傳 NULL。

### **getRunStepIdFromDownloadRegisterStepList**

Get a test step id from an assigned test-step/register file name; this id could be used by "KickoffRunStep(...)" function to kick off Zebra do testing.

```
int getRunStepIdFromDownloadRegisterStepList(__HZHANDLE__ hZhandle, char  
*pRegisterStepName);
```

hZhandle - 下載 test-steps 的 Zebra card 在 manage-queue list 中的 index  
(此 index 需經 bInitUsbDevice() initialized).

\*pRegisterStepName: 指定要取得 ID 的 test-step register-file name.

回傳值：

\_\_USB\_HANDLE\_NOT\_FOUND\_\_: hZhandle not found in Zebra manage-queue list;  
或

由一 register-file name 取得一 test step id; 此 test step id 可以傳給 Zebra, 做為  
執行那一 test step 的依據。當所指定的 test-step/register name 不存在 list 中時，傳  
回 -1.

### **RunTestStepFile**

kick off a test step by test-step file name; this function default to call  
getRunStepIdFromDownloadAliasList() and then KickoffRunStep() function to run a  
test step.

```
int RunTestStepFile (__HZHANDLE__ hZhandle, char *pTestStepName, int nOption);
```

hZhandle - 要執行的 test-step 的 Zebra card 在 manage-queue list 中的 index  
(此 index 需經 bInitUsbDevice() initialized).

\*pTestStepName: 指定要執行的 test-step file name.

nOption- \_\_OPT\_STREAMING\_OFF\_\_ or \_\_OPT\_STREAMING\_ON\_\_

回傳值：

\_\_USB\_HANDLE\_NOT\_FOUND\_\_: hZhandle not found in Zebra manage-queue list;

-1: 當所指定的 test step name 不存在 test step name list 中時，傳回 -1.

1: succeed,

0: Fail to run test step name.



## **SetZebraSetting**

setup a Zebra setting file name and use this name to matching a register-file as test-step when download phase or RunRegisterFile function..

```
void SetZebraSetting(__HZHANDLE__ hZhandle, char *pZebraSetting);
```

hZhandle - 要設定的 test-step setting 的 Zebra card 在 manage-queue list 中的 index  
(此 index 需經 bInitUsbDevice() initialized).

\*pZebraSetting - 要給予將來 register-files 使用的 setting file name.

回傳值：無.

## **RunRegisterFile**

kick off a test step by a Zebra setting file (set by SetZebraSetting()) with register register file; this function default to call getRunStepIdFromDownloadRegisterStepList() and then KickoffRunStep() function to run a test step.

```
int RunRegisterFile (__HZHANDLE__ hZhandle, char *pRegisterName, int nOptions);
```

hZhandle - 要執行 register file 的 Zebra card 在 manage-queue list 中的 index  
(此 index 需經 bInitUsbDevice() initialized).

\*pRegisterName: 指定要執行的 register-file name; 此 name 與前 SetZebraSetting 所設的名字合成要執行的 test-step.

nOption- \_\_OPT\_STREAMING\_OFF\_\_ or \_\_OPT\_STREAMING\_ON\_\_

回傳值：

\_\_USB\_HANDLE\_NOT\_FOUND\_\_: hZhandle not found in Zebra manage-queue list;

-1: 當所指定的 register name 不存在 register name list 中時, 傳回 -1.

1: succeed,

0: Fail to run register name.



## **getDllVersion**

Report current using DLL library name and version.

```
void getDllVersion(char *pDllName, DWORD *dwMSVer, DWORD *dwLSVer);
```

pDllName - return DLL library name; if this DLL name can't be found, function return a zero length string.(1st byte is set to 0)  
dwMSVer - includes HIWORD(MajorVersion), LOWORD(MinorVersion); when function fail return 0xffffffff  
dwLSVer - includes HIWORD(BuildNumber), LOWORD(RevisionNumber);when function fail return 0xffffffff

回傳值：無。

```
#define __ZebraType_Z4__ 0
#define __ZebraType_Z5__ 1
#define __ZebraType_Z6__ 2 // to be later
```

## **getHardwareType**

report current card type..

```
int getHardwareType (__HZHANDLE__ hZhandle);
```

hZhandle - 要取得 hardware type 的 Zebra card 在 manage-queue list 中的 index  
(此 index 可不經 bInitUsbDevice() initialized).

回傳值：

\_\_USB\_HANDLE\_NOT\_FOUND\_\_: hZhandle not found in Zebra manage-queue list;  
或  
report Zebra type.