LNC 控制器

PC 通訊手冊

2017/04 版本：V05.00

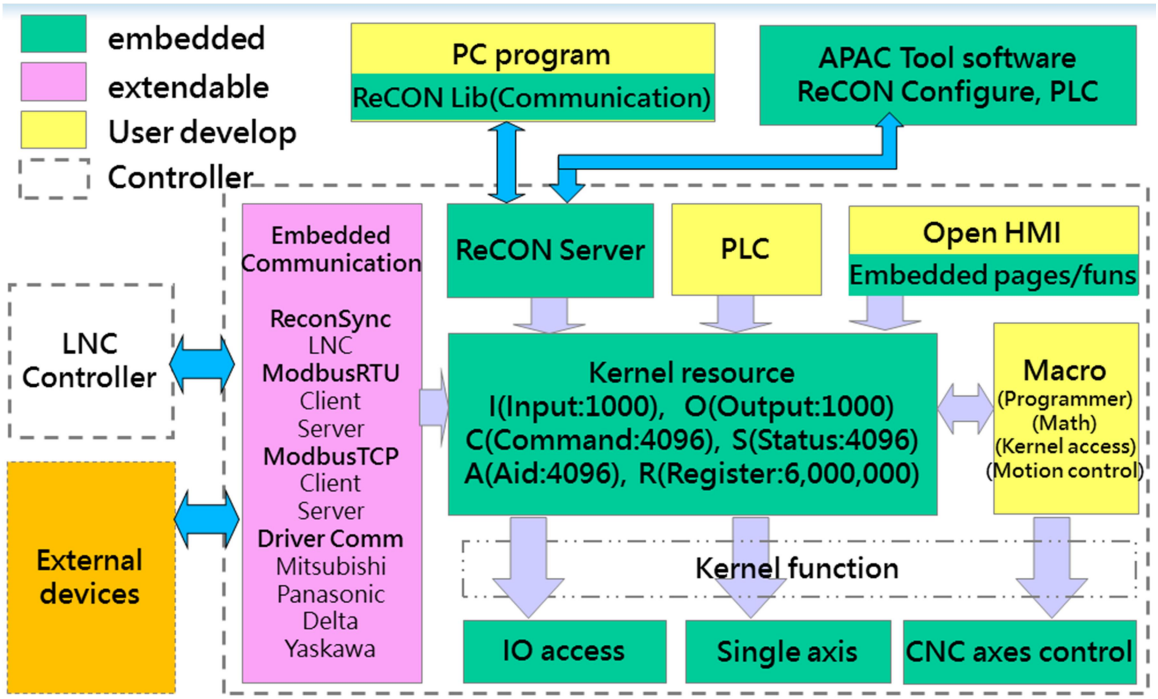
Leading Numerical Controller

ADVANTECH LNC

1. 目錄

2. 控制器資源說明

寶元控制器為一個以核心資源為中心的多功控制系統，核心功能透過檢查核心資源的內容，決定實體 IO 與軸控的作動。



資源的範圍很大，部份區段已經被定義，做為各項功能的參數或是存放運行時的狀態資訊之用，或是供開發者自行規劃之用。

R(Register)資源的範圍最大，某些地址區段是會存檔的，當寫入到該區段時，系統就會定時將其寫入到磁碟(SD, CF)中。

以下是一部份的規劃範例，詳細的核心資源規劃定義，請參閱各產品的資源規劃書。

■ JOG 功能

軸發生警報時，此功能失效。

R 值	Description	R/W	適用模式
R11096	JOG 啓動/關閉訊號。0：OFF，1：ON (bit0: 第 1 軸, bit1:第 2 軸..., bit31：第 32 軸)	W	P Mode
R11097	JOG 啓動/關閉完成訊號 (bit0: 第 1 軸, bit1:第 2 軸..., bit31：第 32 軸) (JOG 啓動/關閉訊號確爲 ON 時，JOG 啓動/關閉完成訊號爲 1；JOG 啓動/關閉訊號確爲 OFF 時，JOG 啓動/關閉完成訊號爲 0)	R	P Mode
R11098	JOG 方向。0：+，1：- (bit0: 第 1 軸, bit1:第 2 軸..., bit31：第 32 軸)	W	P Mode
R11000~ R11031	第 1~32 軸 JOG+速度(KLU/min) (K：軸位置模式速度倍率常數，由參數	W	P Mode

諸多程序(Macro, OpenHMI, PLC, ReCONServer, Embedded Communication)均可同時存取核

心資源以對系統進行控制，其中 ReCONServer 是寶元自行定義通訊格式的伺服程序，同時寶元也為 PC 端建立與其通訊的函式庫(ReCONLib)，開發者可直接透過此函式庫存取到核心資源的內容，不需自行研究撰寫通訊協議，簡化開發並縮短時程。

此份文件就是介紹 ReCONLib 的所提供的函式與介紹使用方法。

3. 函式庫架構

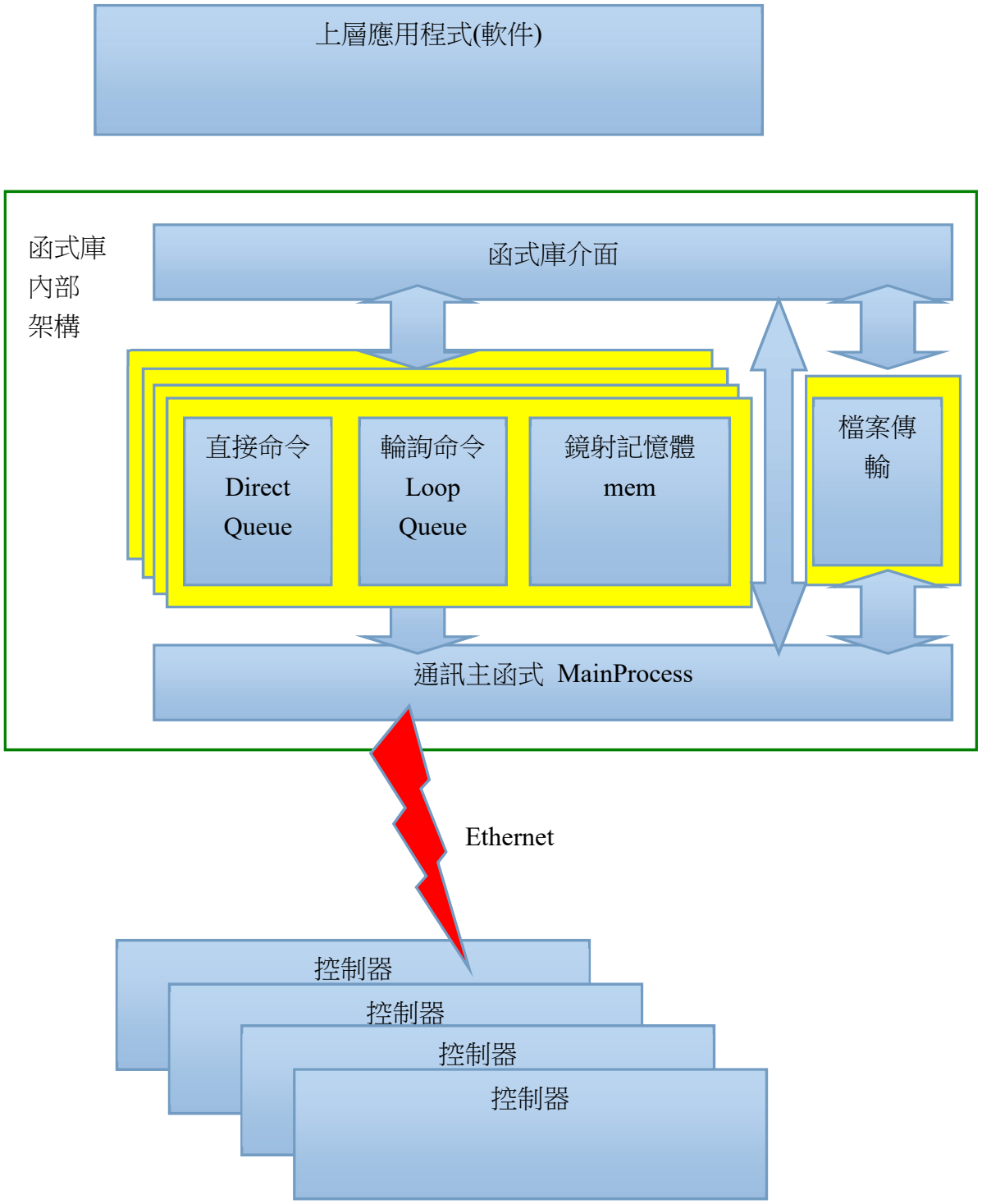
檔案傳輸：
所有連線共用同一個檔案傳輸功能，需於傳輸前用 FtpSetConnection 函式設定所對應的連線。
列舉控制器檔案時，最大檔案清單數目：300

最大連線數：255

每個連線包含項目
循環(Loop)命令佇列個數：64
直接(Direct)命令佇列個數：64
鏡射記憶體：根據 LibraryInitial 時設定的數值決定宣告的大小。

直接(Direct)命令 與 循環(Loll)命令 觀念說明：
直接命令(Direct)：此類型的命令，會被存放到「直接命令佇列 Direct Queue」中，當函式庫成功執行該佇列中的命令後，就會將該命令自佇列中移除，也就是該命令只會被執行一次。
循環命列(Loop)：此類型的命令，會被存放到「循環命令佇列 Loop Queue」中，函式庫會重覆循的執行該佇列中的命令。直到主程式下「清除佇列命令」。

命令的優先順序
「直接(Direct)命令」的優先權是較「循環(Loop)命令」高的，因此只有在「直接命令」已完全被執行完畢後，「循環命令」才有機會被執行。


上層應用程式(軟件)
函式庫
內部
架構
函式庫介面
直接命令
Direct
Queue
輪詢命令
Loop
Queue
鏡射記憶體
mem
檔案傳
輸
通訊主函式 MainProcess
Ethernet
控制器
控制器
控制器
控制器

4. 函式庫版本

此函式庫提供多個版本，但使用同一個標頭檔，藉由定義不同的編譯參數來達到共用的目的。

標頭檔：Scif2.h

__QT : QT Creator 用的。

__CLASS : 使用類別方式，方便上層開發軟件快速列舉函式。

SC2_LIBRARY : 編譯函式庫用的，上層應用軟體請勿宣告

__BCB : Borland C++ 用的。

Borland C++, 無定義__CLASS 時 產生的版本 scif2_bcb.dll 為遵循標準的函式庫的命名法，可供其他開發軟體使用。

VB.net 使用時，已經提供包裝完成的轉換定義檔案 scif2.vb

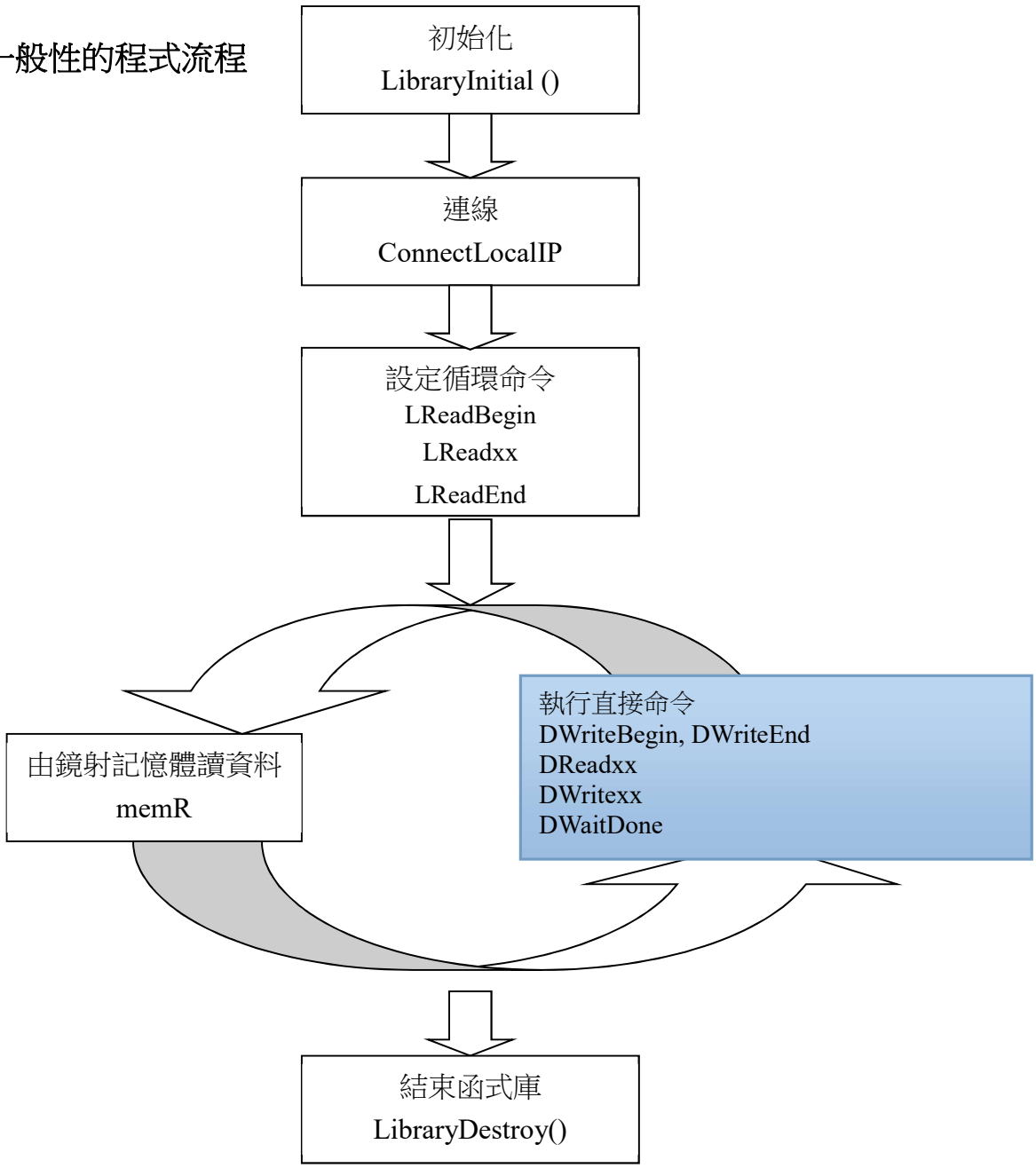
C#.net 使用時，已經提供包裝完成的轉換定義檔案 scif2.cs

```
#ifndef SC2_H
#define SC2_H

#include "scif2_define.h"

#ifdef __QT
    #include <QtCore/qglobal.h>
    #define __CLASS
    #if defined(SC2_LIBRARY)
    #  define SC2SHARED_EXPORT Q_DECL_EXPORT
    #else
    #  define SC2SHARED_EXPORT Q_DECL_IMPORT
    #endif
    #define IMEXPORT
    #define CALBY
#else
    #if defined(__CLASS)
        #if defined(SC2_LIBRARY)
        #  define SC2SHARED_EXPORT __declspec(dllexport)
        #  define IMEXPORT
        #  define CALBY
        #else
        #  define SC2SHARED_EXPORT __declspec(dllimport)
        #  define IMEXPORT
        #  define CALBY
        #endif
    #else
        #ifdef __BCB
            #if defined(SC2_LIBRARY)
            #  define IMEXPORT __declspec(dllexport)
            #  define CALBY      __stdcall
            #else
            #  define IMEXPORT __declspec(dllimport)
            #  define CALBY    __stdcall
            #endif
        #else
            #if defined(SC2_LIBRARY)
            #  define IMEXPORT __declspec(dllexport)
            #  define CALBY    __cdecl
            #else
            #  define IMEXPORT __declspec(dllimport)
            #  define CALBY    __cdecl
            #endif
        #endif
    #endif
#endif

#if defined(__CLASS)
class SC2SHARED_EXPORT SC2
{
public:
    IMEXPORT CALBY SC2();
```

	編譯器	適用上層開發環境	編譯參數, 轉換定義檔
Sc2_bcb.dll Sc2_bcb.lib	Borland C++ 6.0	Borland C++ 6.0	__CLASS;
Scif2_bcb.dll Scif2_bcb.lib	Borland C++ 6.0	Borland C++ 6.0	__BCB;
		VB.net	Scif2.vb
		C#.net	Scif2.cs
Sc2_vc.dll Sc2_vc.lib	Vitual C++ 2008	Vitual C++	__CLASS;
Scif2_vc.dll Scif2_vc.lib	Vitual C++ 2008	Vitual C++	
Sc2_qt.dll Libsc2_qt.a	QT Creator 5.8	QT Creator 5.8	__QT

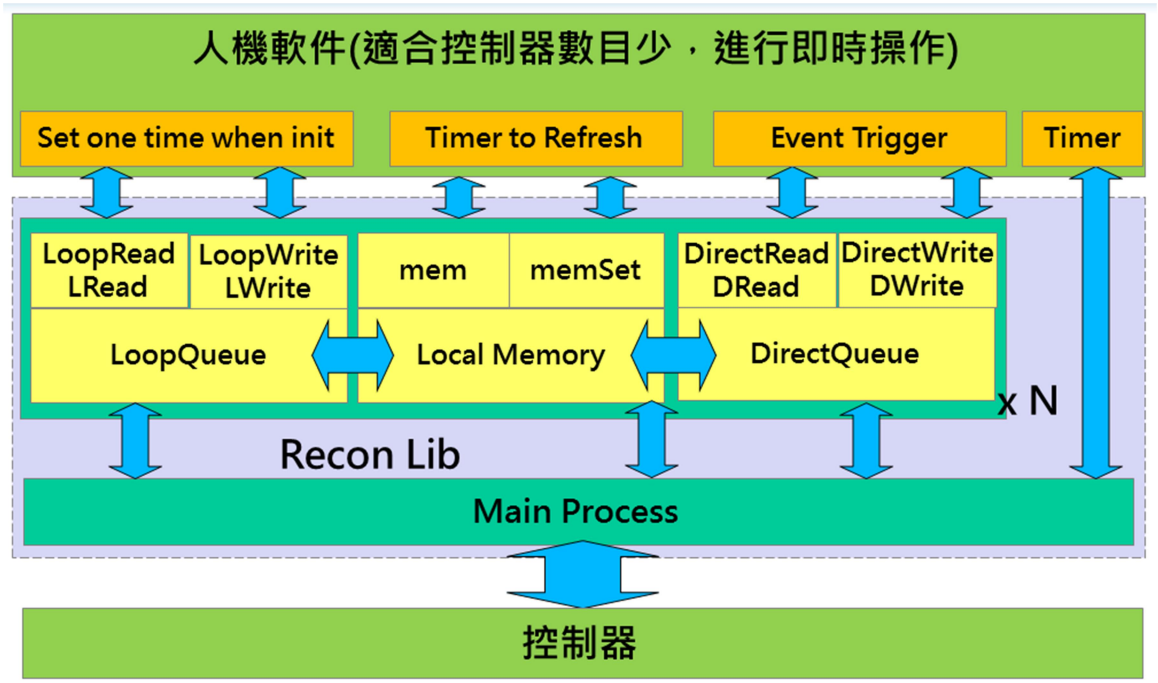
5. 一般性的程式流程

一般性的程式流程



6. 函式庫應用場合

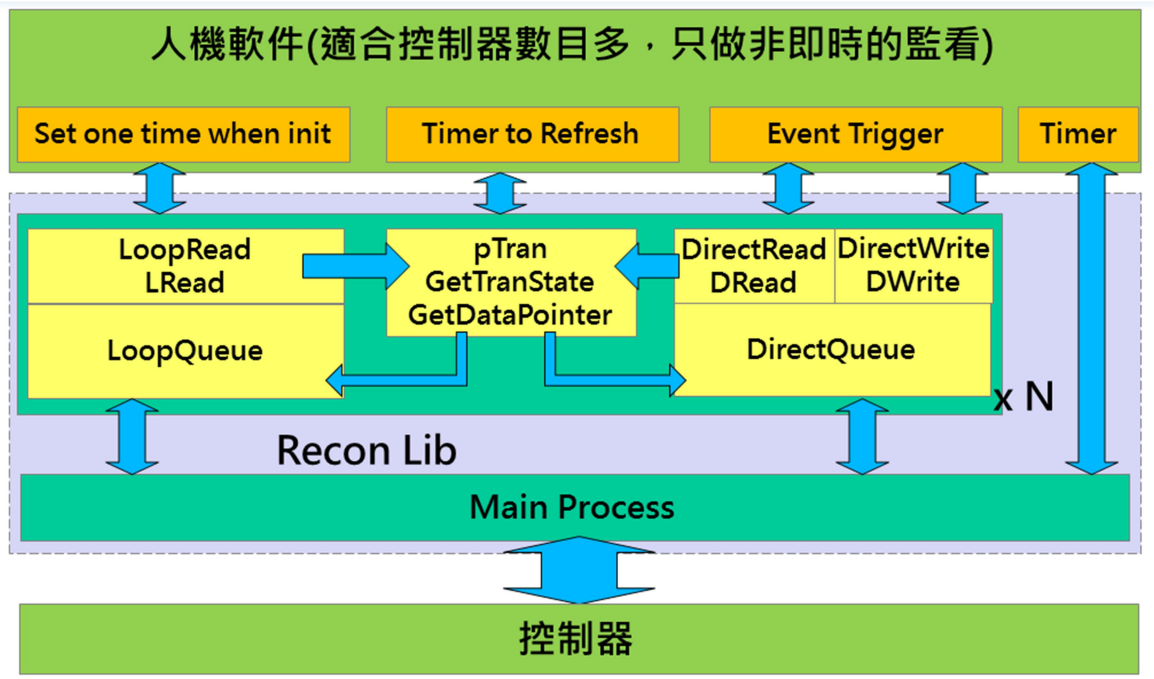
對單一控制器即時操控



因為只對一台控制器，為了方便使用，函式庫初始化時，可以將一台控制器的所有資源均鏡射到 PC 上，大約會用到 25MB 的記憶體空間。

因為要有比較好的操控性，呼叫 MainProcess 的 Timer 的頻率可以設定在 20ms 左右。

對多台控制器做非即時的監看



因為要對應多台控制器，若將控制器資源均鏡射到 PC，如果控制器數目多，佔用的 PC 記憶體會較多，因此可使用直接讀取封包的內容的方式來取得通訊結果。

依照需要的即時性、連接控制器數目、所處的網路頻寬，決定呼叫 MainProcess 的 Timer 的頻率，20ms~1000ms 都是可以的。

8. 函式庫初始化

函式庫需先執行初始化動作，初始化的內容包含

1. 連線數目：連線的控制器數量，當用於監控使用，或以兩台控制器同時組成系統時，會需要同時連線到多台控制器，此數值即代表要連線的控制器數量。
2. 各項資源使用數量：為了函式庫的使用方便，會在 PC 端為每個連線建立一個鏡射記憶體，用以存放自控制器讀回的資料，此宣告要開啟的記憶體大小，當連線目量多時，更該特別注意此設定值，以免將 PC 的記憶體全部耗用完畢。
3. 製造商識別密碼字串：用來確認開發的軟體種類為系統提供已定義名稱的軟體，或是使用者自行開發的軟體。若密碼字串錯誤，初始化將會失敗，函式庫將無法正確運作。初始化成功後，呼叫 MainProcess 時才會有作用。

函式庫初始化 LibraryInitial

此函式為建立在本機端所需的鏡射記憶體與執行緒，並根據提供的參數作為函式庫中的使用設定。

語法

int LibraryInitial (DLL_USE_SETTING *pUseSetting, int MakerID, char *pEncString);

參數

pUseSetting：

函式庫的使用設定，可在 scif2_define.h 檔中找到其 struct 的內容。其中的 SoftwareType 代表軟體種類，ConnectNum 代表連線數目。

MakerID：

製造商編號，此參數會由原廠寶元數控提供。

pEncString：

加密字串，包含資訊有 MakerID、區域網路偵測功能、是否使用鏡射記憶體、可否資料寫入、是否有檔案傳輸功能，此參數亦會由原廠寶元數控提供。

回傳值

0：初始化失敗。

10：初始化成功、但解密功能字串失敗。

100：成功。

範例

```
DLL_USE_SETTING DllSetting;
DllSetting.SoftwareType = 3;            //軟體種類(即第幾個連線，新版控制器此參數無作用。)
DllSetting.ConnectNum= 5;               //連線數目
//以下參數的設定代表在PC端所宣告鏡射記憶體所能讀取各參數值位址的
//範圍，需要特別注意的為MemSizeR的大小，如果將其設定成與控制器預設//R值(6000000)大小相同的話，則會在PC端使用約25MB的大小，再乘以連
//線數目，在本範例中則會使用約125MB，故我們在本範例中特地將
//MemSizeR設為10000，而非預設值，但設定為10000後，則R位址值超過
//10000以上的數值，不會儲存在鏡射記憶體中，請使用者根據自己所需要的
//各參數位址範圍來設定以下的MemSize值，數字代表PC上對每個控制器連線
//開啟鏡射記憶體時，每項資源的使用個數。
DllSetting.MemSizeI    =   4096;          //共使用  4096 * 1byte
DllSetting.MemSizeO    =   4096;          //共使用  4096 * 1byte
DllSetting.MemSizeC    =   4096;          //共使用  4096 * 1byte
DllSetting.MemSizeS    =   4096;          //共使用  4096 * 1byte
DllSetting.MemSizeA    =   4096;          //共使用  4096 * 1byte
DllSetting.MemSizeR    =   10000;     //共使用  10000 * 4byte
DllSetting.MemSizeF    =   100000;        //共使用  100000* 8byte

int makerid, Status;
char pencstring[64];
Status = LibraryInitial(&DllSetting, makerid, pencstring);
```

結束函式庫 LibraryDestroy

此函式為終結建立的記憶體與執行緒。

語法

```
void LibraryDestroy();
```

參數

無。

回傳值

無。

範例

```
LibraryDestroy();
```

9. 定時呼叫程序

呼叫函式庫主程序 MainProcess

此函式為由外部呼叫函式庫主程序運行使用，一般使用 Timer 來循環呼叫執行。

語法

int MainProcess ();

參數

無。

回傳值

0：設定指令失敗，表示函式庫尚未初始化 。

其餘數值：表示上一次呼叫與本次呼叫間隔時間，其單位為 ms。

範例

```
int Status;
Status = MainProcess ();
```

10. 偵測控制器

此章節會說明如何將我們的函式庫初始化，並根據提供的參數作為函式庫中的使用設定，在這章節介紹的函式為如何偵測區域網路內的控制器，包括其數量與控制器資訊，也會說明與控制器的連線方式。

偵測區網內的控制器 LocalDetectControllers

此函式會自動偵測區域網路內有多少控制器，並讀取其控制器資訊，並在函式庫內依序建立每個控制器的資料索引，若無呼叫此函式，則呼叫 LocalReadControllerCount 與 LocalReadController 兩函式時，不會有正確的回傳值。

語法

int LocalDetectControllers();

參數

無。

回傳值

區域網路內偵測到的控制器數量。也可在偵測試，再透過函式 LocalReadControllerCount 讀取。

範例

```
int Count;
Count = LocalDetectControllers();
```

取得偵測到的控制器數量 LocalReadControllerCount

此函式為讀取在函式庫內記錄的控制器資料筆數，在呼叫此函式前，必須先呼叫 scif_LocalDetectControllers 函式，才會有正確的回傳值。

語法

int LocalReadControllerCount ();

參數

無。

回傳值

函式庫中記錄偵測到的控制器數量。

範例

```
int Count;
Count = LocalReadControllerCount ();
```

取得偵測到的控制器資訊 LocalReadController

此函式會根據傳入的控制器資料索引，將每個控制器的資料存放入函式庫的控制器資料結構中，但在呼叫此函式前，必須先呼叫 scif_LocalDetectControllers 函式，才會有正確的回傳值。

語法

int LocalReadController (int Index, LOCAL_CONTROLLER_INFO *Info);

參數

Index：

函式庫內記錄的區域控制器資料索引。

Info：

函式庫的使用設定，可在 scif2_define.h 檔中找到其 struct 的內容。內容中會有控制器 IP、名稱等資訊。

```
typedef struct tag_LOCAL_CONTROLLER_INFO1
{
    unsigned int        IPLong;
    char                IP[16];
    char                Name[16];
}LOCAL_CONTROLLER_INFO;
```

回傳值

0：失敗。

1：成功。

範例

```
//在LOCAL_CONTROLLER_INFO的struct中，會存放控制器的IP和Name，
//其中Name為使用者在設定控制器時，對控制器命名的名稱，使用者可經由
//控制器的名稱搭配控制器IP判斷此連線的控制器是否為其所想連線的對
//象。

LOCAL_CONTROLLER_INFO Info;
int Status;
int controllerindex;
Status = LocalReadController (controllerindex, &Info);
```

11. 控制器連線

輸入連線密碼 ConnectSetPwd

當控制器沒有設定密碼時，不需使用此函式。

語法

int ConnectSetPwd (int TgrConn, char *Pwd);

參數

TgrConn：連線索引

*Pwd：連線密碼

回傳值

輸入是否被接受。只有當連線索引值無效時，才會回傳 0，其他狀況回傳 1。

範例

```
int Success;
Success = ConnectSetPwd (0, “1234”);
```

連線到偵測到的控制器 ConnectLocalList

此函式為與函式庫內記錄的控制器資料索引進行連線設定，在呼叫此函式前，必須呼叫過 LocalReadController 函式，才會有正確的控制器資料索引。執行此函式成功只代表連線設定成功，有無真正建立起連線，必須呼叫 GetConnectionMsg 函式來檢查連線狀態。

語法

int ConnectLocalList (int TgrConn, int Index);

參數

TgrConn：

使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。

Index：

函式庫內記錄的區域控制器資料索引。

回傳值

0：設定指令失敗 。

1：設定指令被接受。

範例

```
int TgrConn;
int controllerindex;
int Status;
Status = scif_ ConnectLocalList (TgrConn, controllerindex);
```

直接連線到指定 IP 的控制器 ConnectLocalIP

此函式為直接輸入控制器 IP 進行連線設定。執行此函式成功只代表連線設定成功，有無真正建立起連線，必須呼叫 GetConnectionMsg 函式來檢查連線狀態。

語法

int ConnectLocalIP (int TgrConn, char *IP);

參數

TgrConn：

使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。

IP：

欲連線的控制器 IP。

回傳值

0：設定指令失敗 。

1：設定指令被接受。

範例

```
int TgrConn;
char ip[32];
int Status;
Status = scif_ LocalConnectIP(TgrConn, ip);
```

中斷連線 Disconnect

呼叫此函式中斷與控制器的連線。

語法

int Disconnect (int TgrConn);

參數

TgrConn：

使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。

回傳值

0：設定指令失敗 。

1：設定指令被接受。

範例

```
int TgrConn;
int Status;
Status = scif_ Disconnect(TgrConn);
```

取得連線資訊 GetConnectionMsg

呼叫此函式可取得連線通訊的資訊。

語法

int GetConnectionMsg(int TgrConn, char id);

參數

TgrConn：

使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。

id：

scif2_define.h 檔中有可填入的 id 資訊，如填入 SCIF_CONNECT_STATE 為取得連線的狀態。

回傳值

資訊內容；scif2_define.h 檔中有回傳資訊內容代表的意義。

範例

```
int TgrConn;
int Status;
Status = GetConnectionMsg(TgrConn, SCIF_CONNECT_STATE);
```

連線狀態補充說明

- 可使用同一個 IP 對控制器建立多個連線，但前提是必須使用不同 SoftwareType(軟體種類)。

- 連線時可能的狀態有

```
#define SC_CONN_STATE_DISCONNECT      0    //連線關閉
#define SC_CONN_STATE_CONNECTING      1    //連線中
#define SC_CONN_STATE_FAIL            2    //連線失敗
#define SC_CONN_STATE_OK              3    //連線正常
#define SC_CONN_STATE_NORESPONSE      4    //連線無回應
```

  - 剛啟動函式庫或要求中斷連線後，狀態為 SC_CONN_STATE_DISCONNECT。
  - 要求連線後，狀態變為 SC_CONN_STATE_CONNECTING。

- 若連線失敗，狀態變為 SC_CONN_STATE_FAIL，之後變成 SC_CONN_STATE_NORESPONSE。
- 若連線成功，狀態變為 SC_CONN_STATE_OK。
- 若斷線或控制器關機，狀態變為 SC_CONN_STATE_NORESPONSE。
- 在非 SC_CONN_STATE_DISCONNECT 狀態下，會自動嘗試重新連線。

- 與控制器的連線設定完成後，必須等待到連線狀態回傳為連線正常，才可以確定與控制器的連線真的成功。

12. 循環命令

循環命令是指在設定後的命令會一直重覆被執行的命令。
若所要讀取值的位址太過分散、可使用 xxBegin, xxEnd 函式進行組合，以減少通訊封包數目，提高通訊即時性。
循環寫出的命令，會由鏡射記憶體中讀取資料，寫到控制器中，使用時需搭配 memSet 命令。

清除循環(Loop)命令設定 LClearQueue

此函式是用來清除先前已設定的循環(Loop)命令資料讀取及寫入，以便重新設定要讀取及寫入的內容。在切換頁面時，需將原本設定讀/寫更新的內容清除，再重新設定。

語法

void LClearQueue (int TgrConn);

參數

TgrConn：連線索引。

回傳值

無。

範例

LClearQueue (0);

設定循環(Loop)命令讀取 Bit 型態資料 LRead(Bit)

包含的含式有：
LReadNI、LReadNO、LReadNC、LReadNS、LReadNA

這些函式皆是用來作為連續資料讀取的設定，根據資料類型的不同，分為不同的函式去讀取，但所需輸入參數的意義皆相同。以函式 LReadI 為例來說明。

語法

int LReadNI(int TgrConn, int addr, int num);

參數

TgrConn：
使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。-1 代表所有連線都要套用此設定。

addr：
起始位址。

num：
讀取數量，num 的最大值為 MAX_BIT_NUM(scif2_define.h 檔中定義)。

回傳值

通訊封包的指標，若回傳值為 0，代表命令設定失敗。(若函式被包在 LReadBegin 與 LReadEnd 函式之間，回傳值為 1，之後無法用此指標判斷命令是否已被執行。)

範例

```
int pTran;
pTran = LReadI(0, 2000, 5);
```

設定循環(Loop)命令讀取 Register 資料

語法

int LReadNR(int TgrConn, int addr, int num);

參數

TgrConn：

使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。-1 代表所有連線都要套用此設定。

addr：

起始位址。

num：

讀取數量，num 的最大值為 MAX_INT_NUM(scif2_define.h 檔中定義)。

回傳值

通訊封包的指標，若回傳值為 0，代表命令設定失敗。(若函式被包在 LReadBegin 與 LReadEnd 函式之間，回傳值為 1，之後無法用此指標判斷命令是否已被執行。)

範例

```
int pTran;
pTran = LReadNR(0, 4000, 10);
```

設定循環(Loop)命令讀取 Float 資料

語法

int LReadNF(int TgrConn, int addr, int num);

參數

TgrConn：

使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。-1 代表所有連線都要套用此設定。

addr：

起始位址。

num：

讀取數量，num 的最大值為 MAX_FIX_NUM(scif2_define.h 檔中定義)。

回傳值

通訊封包的指標，若回傳值為 0，代表命令設定失敗。(若函式被包在 LReadBegin 與 LReadEnd 函式之間，回傳值為 1，之後無法用此指標判斷命令是否已被執行。)

範例

```
int pTran;
pTran = LReadNF(0, 0, 100);
```

設定循環(Loop)命令讀取指令開始組合封包 LReadBegin

此函式是用來作為設定自動組合旗標，呼叫此函式後，需再呼叫 LReadEnd 函式，才能完成封包組合的設定。

語法

void LReadBegin (int TgrConn);

參數

TgrConn：

使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。

回傳值

無。

範例

LReadBegin (0);

設定循環(Loop)命令讀取指令完成組合封包 LReadEnd

此函式為完成自動組合設定並開始產生組合封包，呼叫此函式前必須先呼叫 LReadBegin。下一章節有對封包組合作進一步的補充說明。

語法

int LReadEnd (int TgrConn);

參數

TgrConn：

使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。

回傳值

最後一個組合後封包的指標，若回傳值為 0，代表命令設定失敗。

範例

```
int pTran;
LReadBegin (0);
LReadNR(0, 4000, 10);
LReadNR(0, 5000, 10);
LReadNR(0, 6000, 10);
pTran = LReadEnd (0);
```

設定循環(Loop)命令寫入 Bit 型態資料 LWrite(Bit)

包含的函式有：
LWriteNO、LWriteNC、LWriteNA

這些函式皆是用來作為連續資料的寫入，根據資料類型的不同，分為不同的函式去寫值，但所需輸入參數的意義皆相同。以函式 LWriteNO 為例來說明。

語法

int LWriteNO(int TgrConn, int addr, int num);

參數

TgrConn：
使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。-1 代表所有連線都要套用此設定。

addr：
起始位址。

num：
讀取數量，num 的最大值為 MAX_INT_NUM(scif2_define.h 檔中定義)。

回傳值

通訊封包的指標，若回傳值為 0，代表命令設定失敗。

範例

```
int Status;
Status = LWriteNO (0, 2000, 10);
```

設定循環(Loop)命令寫入 Register 資料

語法

int LWriteNR(int TgrConn, int addr, int num);

參數

TgrConn：

使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。-1 代表所有連線都要套用此設定。

addr：

起始位址。

num：

讀取數量，num 的最大值為 MAX_INT_NUM(scif2_define.h 檔中定義)。

回傳值

通訊封包的指標，若回傳值為 0，代表命令設定失敗。

範例

int Status;

Status = LWriteNR(0, 1000, 5);

設定循環(Loop)命令寫入 Float 資料

語法

int LWriteNF(int TgrConn, int addr, int num);

參數

TgrConn：

使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。-1 代表所有連線都要套用此設定。

addr：

起始位址。

num：

讀取數量，num 的最大值為 MAX_INT_NUM(scif2_define.h 檔中定義)。

回傳值

通訊封包的指標，若回傳值為 0，代表命令設定失敗。

範例

int Status;

Status = LWriteNF (0, 0, 12);

設定循環(Loop)命令寫入指令開始組合封包 LWriteBegin

此函式是用來作為設定自動組合旗標，呼叫此函式後，需再呼叫 LWriteEnd 函式，才能完成封包組合的設定。

語法

void LWriteBegin (int TgrConn);

參數

TgrConn：

使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。

回傳值

無。

範例

LWriteBegin (0);

設定循環(Loop)命令寫入指令完成組合封包 LWriteEnd

此函式為完成自動組合設定並開始產生組合封包，呼叫此函式前必須先呼叫 LWriteBegin。

語法

int LWriteEnd (int TgrConn);

參數

TgrConn：
使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。

回傳值

通訊封包的指標，若回傳值為 0，代表命令設定失敗。

範例

```
int Status;
LWriteBegin (0);
LWriteNR(0, 1000, 5);
LWriteNR(0, 2000, 5);
LWriteNR(0, 3000, 5);
Status = LWriteEnd (0);
```

13. 直接命令

清除循環(Direct)命令設定 DClearQueue

此函式是用來清除先前已設定的直接(Direct)命令資料讀取及寫入，以便重新設定要讀取及寫入的內容。在切換頁面時，需將原本設定讀/寫更新的內容清除，再重新設定。

語法

void DClearQueue (int TgrConn);

參數

TgrConn：連線索引。

回傳值

無。

範例

DClearQueue (0);

等待直接命令執行完成 DWaitDone

此函式用來讓等待先前所設定的直接命令完成後，再繼續執行下去。

語法

int DWaitDone (int TgrConn, int MaxWaitTime);

參數

TgrConn：連線索引

MaxWaitTime：最大等待時間(ms)

回傳值

1：直接命令已在最大等待時間內被執行完畢。
0：直接命令在最大等待時間結束前仍未被執行完畢。

範例

```
int rt;
DReadNR(0, 1000, 10);
rt = DWaitDone (0, 2000);
if (rt==1)
      …
else
      ShowMessage(“Timeout”);
Int R1000 = memR(0, 1000);     //等待執行完成後再讀取資料
```

取得某一封包的通訊狀態 GetTranState

此函式用於當不可使用 DWaitDone 命令做等待檢查封包是否已執行完畢的狀況下使用，可記錄在設定命令時回傳的 pTran，然後再 Timer 事件中檢查該封包是否已經執行完畢。

語法

int GetTranState(int pTran);

參數

pTran：
連續或連續通訊資料命令設定時函式回傳的通訊封包指標。

回傳值

命令狀態。

```
#define SC_TRANSACTION_PENDING      0    //等待處理中
#define SC_TRANSACTION_PORCESSING   1    //處理中
#define SC_TRANSACTION_FINISH       2    //完成
#define SC_TRANSACTION_INVALID      3    //無效的索引
```

範例

```
int command_status;
command_status = GetTranState(pTran);
```

以直接(Direct)命令讀取 Bit 型態資料 DRead(Bit)

包含的含式有：
DReadNI、DReadNO、DReadNC、DReadNS、DReadNA

這些函式皆是用來作為連續資料讀取的設定，根據資料類型的不同，分為不同的函式去讀取，但所需輸入參數的意義皆相同。以函式 DReadNI 為例來說明。

語法

int DReadNI(int TgrConn, int addr, int num);

參數

TgrConn：
使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。-1 代表所有連線都要套用此設定。

addr：
起始位址。

num：
讀取數量，num 的最大值為 MAX_CB_BIT_NUM(scif2_define.h 檔中定義)。

回傳值

通訊封包的指標，若回傳值為 0，代表命令設定失敗。(若該設定會被重新組合，回傳值為 1，之後無法用此指標判斷命令是否已被執行。)

範例

```
int pTran;
pTran = DReadNI(0, 1000, 5);
```

以直接(Direct)命令讀取 Register 資料

語法

int DReadNR(int TgrConn, int addr, int num);

參數

TgrConn：

使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。-1 代表所有連線都要套用此設定。

addr：

起始位址。

num：

讀取數量，num 的最大值為 MAX_CB_INT_NUM(scif2_define.h 檔中定義)。

回傳值

通訊封包的指標，若回傳值為 0，代表命令設定失敗。(若該設定會被重新組合，回傳值為 1，之後無法用此指標判斷命令是否已被執行。)

範例

```
int pTran;
pTran = DReadNR(0, 1000, 10);
```

以直接(Direct)命令讀取 Float 資料

語法

int DReadNF(int TgrConn, int addr, int num);

參數

TgrConn：

使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。-1 代表所有連線都要套用此設定。

addr：

起始位址。

num：

讀取數量，num 的最大值為 MAX_CB_FIX_NUM(scif2_define.h 檔中定義)。

回傳值

通訊封包的指標，若回傳值為 0，代表命令設定失敗。(若該設定會被重新組合，回傳值為 1，之後無法用此指標判斷命令是否已被執行。)

範例

int pTran;

pTran = DReadF(0, 1000, 5);

設定直接(Direct)命令讀取指令開始組合封包 DReadBegin

此函式是用來作為設定自動組合旗標，呼叫此函式後，需再呼叫 DReadEnd 函式，才能完成封包組合的設定。

語法

void DReadBegin (int TgrConn);

參數

TgrConn：

使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。

回傳值

無。

範例

DReadBegin (0);

設定直接(Direct)命令讀取指令完成組合封包 DReadEnd

此函式為完成自動組合設定並開始產生組合封包，呼叫此函式前必須先呼叫 DReadBegin。下一章節有對封包組合作進一步的補充說明。

語法

int DReadEnd (int TgrConn);

參數

TgrConn：

使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。

回傳值

通訊封包的指標，若回傳值為 0，代表命令設定失敗。

範例

```
int pTran;
DReadBegin (0);
DReadNR(0, 1000, 10);
DReadNR(0, 2000, 10);
DReadNR(0, 3000, 10);
pTran = DReadEnd (0);
```

以直接(Direct)命令寫入一個 Bit 型態資料 DWrite1(Bit)

包含的函式有：
DWrite1O、DWrite1C、DWrite1A

這些函式皆是用來作為單筆資料的寫入。以函式 DWrite1O 為例來說明。

語法
int DWrite1O (int TgrConn, int addr, int val);

參數
TgrConn：
使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。-1 代表所有連線都要套用此設定。

addr：
要寫入的位址。

val：
要寫入的值。

回傳值
通訊封包的指標，若回傳值為 0，代表命令設定失敗。

範例
```
int pTran;
pTran = DWrite1O (0, 1000, 1);
```

以直接(Direct)命令寫入一個 Register 資料

語法

int DWrite1R(int TgrConn, int addr, int val);

參數

TgrConn：

使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。-1 代表所有連線都要套用此設定。

addr：

要寫入的位址。

val：

要寫入的值。

回傳值

通訊封包的指標，若回傳值為 0，代表命令設定失敗。

範例

```
int pTran;
pTran = DWrite1R (0, 2000, 32);
```

以直接(Direct)命令寫入一個 Float 資料

語法

int DWrite1F(int TgrConn, int addr, double val);

參數

TgrConn：

使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。-1 代表所有連線都要套用此設定。

addr：

要寫入的位址。

val：

要寫入的值。

回傳值

通訊封包的指標，若回傳值為 0，代表命令設定失敗。

範例

int pTran;

pTran = DWrite1F (0, 0, 123.4);

以直接(Direct)命令寫入一個 RBit　DWrite1RBit

包含的函式有：
DWrite1Rbit

此函式是用來作為 R 值單個 bit 位址資料的寫入。

語法
int DWrite1RBit (int TgrConn, int addr, int BitIdx, int BitValue);

參數
TgrConn：
使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。-1 代表所有連線都要套用此設定。

addr：
要寫入 R 值的位址。

BitIdx：
要寫入 R 值的位元位址。

BitValue：
設定值，0 或 1。

回傳值
通訊封包的指標，若回傳值為 0，代表命令設定失敗。

範例
```
int pTran;
pTran = DWrite1RBit (0, 1000, 2, 1);
```

以直接(Direct)命令寫入多個 Bit 型態資料 DwriteN(Bit)

包含的函式有：
DWriteNO、DWriteNC、DWriteNA

這些函式皆是用來作為連續資料的寫入。以函式 DWriteNO 為例來說明。

語法
int DWriteNO (int TgrConn, int addr, int num, int *data);

參數
TgrConn：
使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。-1 代表所有連線都要套用此設定。

addr：
起始位址。

num：
要寫入的數量，最大值為 MAX_BIT_NUM(scif2_define.h 檔中定義)。

data：
要寫入值的陣列指標。

回傳值
通訊封包的指標，若回傳值為 0，代表命令設定失敗。

範例
```
int pTran;
int data[]={1, 1, 0, 0};
pTran = DWriteNO(0, 1200, 4, data);
```

以直接(Direct)命令寫入多個 Register 資料

語法

int DWriteNR(int TgrConn, int addr, int num, int *data);

參數

TgrConn：

使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。-1 代表所有連線都要套用此設定。

addr：

起始位址。

num：

要寫入的數量，最大值為 MAX_INT_NUM (scif2_define.h 檔中定義)。

data：

要寫入值的陣列指標。

回傳值

通訊封包的指標，若回傳值為 0，代表命令設定失敗。

範例

```
int pTran;
int data[]={1, 1, 5, 234};
pTran = DWriteNR(0, 1200, 4, data);
```

以直接(Direct)命令寫入多個 Float 資料

語法

int DWriteNF(int TgrConn, int addr, int num, double *data);

參數

TgrConn：

使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。-1 代表所有連線都要套用此設定。

addr：

起始位址。

num：

要寫入的數量，最大值為 MAX_INT_NUM (scif2_define.h 檔中定義)。

data：

要寫入值的陣列指標。

回傳值

通訊封包的指標，若回傳值為 0，代表命令設定失敗。

範例

```
int pTran;
double data[]={1.1, 2.2, 3.3, 4.4};
pTran = DWriteNF(0, 200, 4, data);
```

以直接(Direct)命令寫入 R 值字串 DWriteRString

包含的函式有：
DwriteRString

此函式是寫入字串到控制器中的 R 值。

語法
int DWriteRString(int TgrConn, int addr, int RSize, char *Buf);

參數
TgrConn：
使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。

addr：
要寫入的資料位址。

BufSize：
要寫入的數量 (Bytes)。

Buf：
要寫入的字串內容。

回傳值
通訊封包的指標，若回傳值為 0，代表命令設定失敗。

範例
int pTran;
char buf[]=”Filename.txt”;
pTran = DWriteRString(0, 17022, 8, buf);

設定直接(Direct)命令寫入指令開始組合封包 DWriteBegin

此函式是用來作為設定自動組合旗標，呼叫此函式後，需再呼叫 DWriteEnd 函式，才能完成封包組合的設定。

語法

void DWriteBegin (int TgrConn);

參數

TgrConn：
使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。

回傳值

無。

範例

DWriteBegin (0);

設定直接(Direct)命令寫入指令完成組合封包 DWriteEnd

此函式為完成自動組合設定並開始產生組合封包，呼叫此函式前必須先呼叫 DWriteBegin。下一章節有對封包組合作進一步的補充說明。

語法

int DWriteEnd (int TgrConn);

參數

TgrConn：

使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。

回傳值

通訊封包的指標，若回傳值為 0，代表命令設定失敗。

範例

```
int pTran;
DWriteBegin (0);
DWrite1R(0, 1000, 3);
DWrite1R(0, 1100, 12);
DWrite1R(0, 1200, 4);
pTran = DWriteEnd (0);
```

14. 由通訊封包中直接讀取通訊資料

取得資料指標 GetDataPointer

若欲以指標結構方式來讀取資料，則必須呼叫此函式由交易封包指標取得通訊命令資料指標，但在呼叫此函式前，必須呼叫 GetTranState 函式確認通訊命令被正確執行。

語法

SC_DATA* GetDataPointer (int pTran);

參數

pTran：
Loop 命令在設定時的順序，由 0 開始。

回傳值

資料結構指標，在 scif2_define.h 檔中定義。

```
union SC_DATA
{
    char    Bytes[MAX_BIT_NUM];      //bit 或 byte 資料
    short   Words[MAX_WORD_NUM];     //word 資料
    int     Ints[MAX_INT_NUM];       //整數
    double  Fixs[MAX_FIX_NUM];       //double
};


//在Timer中
```

補充說明

每筆通訊命令的資料是一個聯集的資料內容，依據命令的資料型態不同，存放在不同的成員中(但其實是同一份記憶體)。

I,O,C,S,A → Bytes。

R → Ints。

F → Fixs。

範例

```
//----全域宣告
SC_DATA *pData;


//----設定通訊內容時
int pTran;
LReadBegin (0);
LReadNR(0, 4000, 3);
LReadNR(0, 5000, 3);
LReadNR(0, 6000, 3);
pTran = LReadEnd (0);
pData = GetDataPointer (pTran);


//----在 Timer 中
Int R4000 = pData.Ints[0];
Int R4001 = pData.Ints[1];
Int R4002 = pData.Ints[2];
Int R5000 = pData.Ints[3];
Int R5001 = pData.Ints[4];
Int R5002 = pData.Ints[5];
Int R6000 = pData.Ints[6];
Int R6001 = pData.Ints[7];
Int R6002 = pData.Ints[8];
```

15. 由鏡射記憶體讀寫資料

由鏡射記憶體讀取 Bit 型式資料 mem(Bit)

包含的函式有：
memI、memO、memC、memS、memA

這些函式用來直接讀取某連線鏡射記憶體資料的值，根據資料類型的不同，分為不同的函式去讀取，但所需輸入參數的意義皆相同。以函式 memI 為例來說明。

語法

int memI(int TgrConn, int addr);

參數

TgrConn：
使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。-1 代表所有連線都要套用此設定。

addr：
資料位址。

回傳值

資料內容。

範例

```
int data;
data = memI(0, 1000);
```

由鏡射記憶體讀取 Register 資料

語法

int memR(int TgrConn, int addr);

參數

TgrConn：

使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。-1 代表所有連線都要套用此設定。

addr：

資料位址。

回傳值

資料內容。

範例

```
int data;
data = memR (0, 1000);
```

由鏡射記憶體讀取 RBit 資料

語法

int memRBit(int TgrConn, int addr, int BitIdx);

參數

TgrConn：

使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。-1 代表所有連線都要套用此設定。

addr：

資料位址。

BitIdx :

位元次，0~31

回傳值

資料內容。

範例

int data;

data = memRBit (0, 1000, 0);

由鏡射記憶體讀取 Float 資料

語法

double memF(int TgrConn, int addr);

參數

TgrConn：

使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。-1 代表所有連線都要套用此設定。

addr：

資料位址。

回傳值

資料內容。

範例

```
double data;
data = memF (0, 1000);
```

由鏡射記憶體讀取 R 值組成的字串 memRString

這些函式用來直接讀取某連線鏡射記憶體中讀取字串。

語法

int memRString(int TgrConn, int addr, int RSize, char *Buf);

參數

TgrConn：
使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。-1 代表所有連線都要套用此設定。

addr：
資料位址。

BufSize：
要讀取的數量 (Bytes)。

Buf：
要回傳的字串內容。

回傳值

要回傳的字串內容的數量(Bytes)。

範例

```
int num, addr, bufsize;
char buf[32];
num = memRString(0 , 17022, 8, buf);
```

寫入鏡射記憶體 Bit 型式資料 memSet(Bit)

包含的函式有：
memSetO、memSetC、memSetS、memSetA

這些函式用來設定鏡射記憶體資料的值，以搭配 LoopWrite 指令循環將資料寫到控制器裡。根據資料類型的不同，分為不同的函式去讀取，但所需輸入參數的意義皆相同。以函式 memSetO 為例來說明。

語法
int memSetO(int TgrConn, int addr, char val);

參數
TgrConn：
使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。-1 代表所有連線都要套用此設定。

addr：
資料位址。

Val ：
資料內容

回傳值
寫入結果。

範例
memSetO(0, 1000, 1);

寫入鏡射記憶體 Register 資料

語法

int memSetR(int TgrConn, int addr, int val);

參數

TgrConn：

使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。-1 代表所有連線都要套用此設定。

addr：

資料位址。

Val ：

資料內容

回傳值

寫入結果。

範例

memSetR(0, 1000, 1234);

寫入鏡射記憶體 RBit 資料

語法

int memSetRBit(int TgrConn, int addr, int BitIdx, int val);

參數

TgrConn：

使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。-1 代表所有連線都要套用此設定。

addr：

資料位址。

BitIdx：

位元次, 0~31

Val ：

資料內容，0/1

回傳值

寫入結果。

範例

memSetR(0, 1000, 0, 1);

寫入鏡射記憶體 Float 資料

語法

int memSetF(int TgrConn, int addr, double val);

參數

TgrConn：

使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。-1 代表所有連線都要套用此設定。

addr：

資料位址。

回傳值

寫入結果。

範例

memSetF(0, 1000, 1.234);

16. 檔案傳輸

此章節說明如何透過我們的函式與控制器間作檔案的傳輸，包括上傳檔案、下載檔案，在控制器上建立資料夾等動作。

設定檔案傳輸的連線 FtpSetConnection

此函式用來設定後續檔案傳輸指令所對應的連線。

語法

int FtpSetConnection(int TgrConn);

參數

TgrConn：

使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。

回傳值

0：設定失敗 。

1：設定成功。

範例

```
int Status;
Status = FtpSetConnection(0);
```

取得控制器端檔案清單 FtpGetFileList

取得控制器資料夾內的檔案清單，並建立檔案索引，呼叫此函式後必須呼叫 FtpCheckDone 函式來確認執行狀態，並且等到回傳值為 1，告知執行動作已完成後，才能再進行下一個 ftp 檔案傳輸的動作。

語法

int FtpGetFileList(int Folder, char *SubFolder, char *HeadFilter, char *TailFilter);

參數

Folder：
控制器所的目標資料夾，可在 scif2_define.h 檔中找到定義，
例：#define FTP_FOLDER_NCFILES 10

SubFolder：
控制器所要上傳檔案的目標下的資料夾名稱，一般皆填""。

HeadFilter：
檔案名稱前導字元過濾字串。

TailFilter：
檔案名稱結束字元過濾字串。

回傳值

0：命令設定失敗。
1：命令設定成功。

範例

```
char folder = FTP_FOLDER_NCFILES;
char headfilter []="";
char tailfilter[]=".nc";
int Status;
Status = FtpGetFileList(folder, "", headfilter, tailfilter);
```

讀取控制器端檔案個數 FtpReadFileCount

讀取 FTP 檔案清單中的檔案個數，呼叫此函式前，必須過執行 FtpGetFileList 函式。

語法

int FtpReadFileCount();

參數

無。

回傳值

檔案清單中的檔案個數。

範例

```
int Status;
Status = FtpReadFileCount();
```

讀取控制器端檔案資訊 FtpReadFile

讀取 FTP 檔案名稱，呼叫此函式前，必須執行過 FtpGetFileList 函式。

語法

int FtpReadFile(int Index, FTP_FILE *File);

參數

Index：

要讀取的檔案索引。

File：

用來接收檔案屬性資料的結構，在 scif2_define.h 檔中定義。

```
struct FTP_FILE
{
    char            filename[FILENAME_LENGTH];//檔案名稱的最大字元數
    unsigned int    filesize;
    unsigned short  year;
    unsigned char   month;
    unsigned char   day;
    unsigned char   hour;
    unsigned char   minute;
    unsigned char   second;
    unsigned char   Reserve;
};
```

回傳值

0：命令設定失敗。

1：命令設定成功。

範例

```
FTP_FILE file
int Status;
Status = FtpReadFile(0, &file);
```

建立資料夾 FtpMakeDir

在控制器上建立資料夾，呼叫此函式後必須呼叫 FtpCheckDone 函式來確認執行狀態，並且等到回傳值為 1，告知執行動作已完成後，才能再進行下一個 ftp 檔案傳輸的動作。

語法

int FtpMakeDir (int Folder, char *DirName);

參數

Folder：
控制器所的目標資料夾，可在 scif2_define.h 檔中找到定義，
例：#define FTP_FOLDER_NCFILES 10

DirName：
資料夾名稱指標。

回傳值

0：命令設定失敗。
1：命令設定成功。

範例

```
int folder = FTP_FOLDER_NCFILES;
char dirname[]=”Bak”;
int Status;
Status = FtpMakeDir (folder, dirname);
```

上傳一個檔案 FtpUpload1File

此函式為上傳單一檔案到控制器的資料夾。呼叫此函式後必須呼叫 FtpCheckDone 函式來確認執行狀態，並且等到回傳值為 1，告知執行動作已完成後，才能再進行下一個 ftp 檔案傳輸的動作。

語法

int FtpUpload1File(int Folder, char *SubFolder, char *Filename, char *LocalFilename);

參數

Folder：
控制器所要上傳檔案的目標資料夾，可在 scif2_define.h 檔中找到定義，
例：#define FTP_FOLDER_NCFILES 10

SubFolder：
控制器所要上傳檔案的目標下的資料夾名稱，一般皆填""。

Filename：
檔案名稱。

LocalFilename：
PC 端的檔案完整路徑。

回傳值

0：設定失敗 。
1：設定成功。

範例

```
char folder = FTP_FOLDER_NCFILES;
char filename[]=”1234.txt”;
char localfilename[]=”D:\\ncfiles\\1234.txt”;
int Status;
Status = FtpUpload1File(folder, ””, filename, localFilename);
```

下載一個檔案 FtpDownload1File

此函式為從控制器下載單一檔案到本地的資料夾。呼叫此函式後必須呼叫 scif_FtpCheckDone 函式來確認執行狀態，並且等到回傳值為 1，告知執行動作已完成後，才能再進行下一個 ftp 檔案傳輸的動作。

語法

int scif_FtpDownload1File(int Folder, char *SubFolder, char *Filename, char *LocalFilename);

參數

Folder：
控制器所要下載檔案的目標資料夾，可在 scif2_define.h 檔中找到定義，
例：#define FTP_FOLDER_NCFILES 10

SubFolder：
控制器所要上傳檔案的目標下的資料夾名稱，一般皆填""。

Filename：
檔案名稱。

LocalFilename：
PC 端的檔案完整路徑。

回傳值

0：設定失敗 。
1：設定成功。

範例

```
char folder = FTP_FOLDER_NCFILES;
char filename[]=”1234.txt”;
char localfilename[]=”D:\\ncfiles\\1234.txt”;
int Status;
Status = FtpDownload1File(folder, ””, filename, localFilename);
```

刪除控制器端一個檔案 FtpDelete1File

此函式為刪除控制器上的檔案。呼叫此函式後必須呼叫 FtpCheckDone 函式來確認執行狀態，並且等到回傳值為 1，告知執行動作已完成後，才能再進行下一個 ftp 檔案傳輸的動作。

語法

int FtpDelete1File(int Folder, char *SubFolder, char *Filename);

參數

Folder：

控制器所要刪除檔案的目標資料夾，可在 scif2_define.h 檔中找到定義，

例：#define FTP_FOLDER_NCFILES 10

SubFolder：

控制器所要上傳檔案的目標下的資料夾名稱，一般皆填""。

Filename：

檔案名稱。

回傳值

0：設定失敗 。

1：設定成功。

範例

```
char folder = FTP_FOLDER_NCFILES;
char filename[]=”1234.txt”;
int Status;
Status = FtpDelete1File (folder, ””, filename);
```

重置檔案傳輸列表 FtpTransferFileReset

重置「檔案傳輸列表」，之後再以 FtpTransferFileAdd 函式來新增檔案至列表中。

語法

int FtpTransferFileReset();

參數

無。

回傳值

0：設定失敗 。

1：設定成功。

範例

```
int Status;
Status = FtpTransferFileReset();
```

新增「檔案傳輸列表」的項目空間 FtpTransferFileAdd

上傳/下載/刪除多筆檔案時，需使用此函式將欲上傳/下載/刪除的檔案，先行新增至「檔案傳輸列表」內。呼叫此函式前可呼叫 FtpTransferFileReset 函式來重置檔案記憶體空間，確保已無其它檔案記錄。

語法

int FtpTransferFileAdd (int Folder, char *SubFolder, char *Filename, char *LocalFilename);

參數

Folder：
控制器所要上傳檔案的目標資料夾，可在 scif2_define.h 檔中找到定義，
例：#define FTP_FOLDER_NCFILES 10

SubFolder：
控制器所要上傳檔案的目標下的資料夾名稱，一般皆填""。

Filename：
檔案名稱。

LocalFilename：
PC 端的檔案完整路徑。

回傳值

0：設定失敗 。
1：設定成功。

範例

```
char folder = FTP_FOLDER_NCFILES;
char filename[]=”1234.txt”;
char localfilename[]=”D:\\ncfiles\\1234.txt”;
int Status;
Status = FtpTransferFileAdd (folder, ””, filename, localFilename);
```

上傳多個檔案 FtpUploadFiles

根據「檔案傳輸列表」中的設定，進行批次檔案上傳。呼叫此函式後必須呼叫 FtpCheckDone 函式來確認執行狀態，並且等到回傳值為 1，告知執行動作已完成後，才能再進行下一個 ftp 檔案傳輸的動作。

語法

int FtpUploadFiles();

參數

無。

回傳值

0：設定失敗 。

1：設定成功。

範例

```
int Status;
Status = FtpUploadFiles ();
```

下載多個檔案 FtpDownloadFiles

根據「檔案傳輸列表」中的設定，進行批次檔案下載。呼叫此函式後必須呼叫 FtpCheckDone 函式來確認執行狀態，並且等到回傳值為 1，告知執行動作已完成後，才能再進行下一個 ftp 檔案傳輸的動作。

語法

int FtpDownloadFiles ();

參數

無。

回傳值

0：設定失敗 。

1：設定成功。

範例

```
int Status;
Status = FtpDownloadFiles ();
```

刪除多個檔案 FtpDeleteFiles

根據「檔案傳輸列表」中的設定，進行批次檔案刪除。呼叫此函式後必須呼叫 FtpCheckDone 函式來確認執行狀態，並且等到回傳值為 1，告知執行動作已完成後，才能再進行下一個 ftp 檔案傳輸的動作。

語法

int FtpDeleteFiles ();

參數

無。

回傳值

0：設定失敗 。

1：設定成功。

範例

```
int Status;
Status = FtpDeleteFiles ();
```

檢查檔案傳輸指令完成 FtpCheckDone

此函式為取得 ftp 檔案傳輸執行結果，每當呼叫完關於上傳、下載、刪除檔案、建立目錄等函式後，皆需要呼叫此函式，當此函式回傳值為 1 後，代表動作完成，FTP 的狀態將回復成閒置狀態，如此才能再進行下一個有關於 FTP 傳輸的動作。

語法

int FtpCheckDone();

參數

無。

回傳值

1：要求執行的動作已完成。

0：未完成。

範例

```
int Status;
Status = scif_FtpCheckDone();
```

取得本地端檔案清單 LocalGetFileList

取得本地端檔案清單，並建立起檔案索引。

語法

int LocalGetFileList (char *Path, char *HeadFilter, char *TailFilter);

參數

Path：

本地端資料夾路徑。

HeadFilter：

檔案名稱前導字元過濾字串。

TailFilter：

檔案名稱結束字元過濾字串。

回傳值

0：命令設定失敗。

1：命令設定成功。

範例

```
char path[]=”D:\\ncfiles”
char headfilter []=””;
char tailfilter[]=”.nc”;
int Status;
Status = LocalGetFileList(path, headfilter, tailfilter);
```

讀取本地端檔案個數 LocalReadFileCount

讀取本地端檔案清單中的檔案個數，呼叫此函式前，必須過執行 LocalGetFileList 函式。

語法

int LocalReadFileCount();

參數

無。

回傳值

檔案清單中的檔案個數。

範例

```
int Status;
Status = LocalReadFileCount();
```

讀取本地端檔案資訊 LocalReadFile

讀取本地端檔案名稱，呼叫此函式前，必須執行過 LocalGetFileList 函式。再以回圈讀取檔案資訊。

語法

int LocalReadFile(int Index, FTP_FILE *File);

參數

Index：

要讀取的檔案索引。

File：

用來接收檔案屬性資料的結構，在 scif2_define.h 檔中定義。

```
struct FTP_FILE
{
    char            filename[FILENAME_LENGTH];//檔案名稱的最大字元數
    unsigned int    filesize;
    unsigned short  year;
    unsigned char   month;
    unsigned char   day;
    unsigned char   hour;
    unsigned char   minute;
    unsigned char   second;
    unsigned char   Reserve;
};
```

回傳值

0：命令設定失敗。

1：命令設定成功。

範例

```
FTP_FILE file
int Status;
Status = LocalReadFile(0, &file);
```

刪除本地端一個檔案 LocalDeleteFile

此函式為刪除本地端檔案。呼叫此函式前，必須執行過 LocalGetFileList 函式。

語法

int LocalDeleteFile(int Index);

參數

Index：

要刪除的檔案在本地檔案清單中的索引。

回傳值

0：設定失敗 。

1：設定成功。

範例

```
int Status;
Status = LocalDeleteFile(0);
```

17. 函式庫內部資訊

此章節說明如何取得函式庫內部資訊，資訊內容分為一般資料、連線資訊及錯誤資訊取得。

取得函式庫資訊 GetLibraryMsg

此函式為從函式庫內部資訊取得一般的資訊。

語法

int GetLibraryMsg (int MsgID);

參數

MsgID：可以為以下值，在 scif2_define.h 檔中定義。

```
#define SCIF_PROC_COUNTER          1    //porcess counter
#define SCIF_MEDIA_STEP            5    //與媒合主機通訊的處理步驟
#define SCIF_FTP_STATE             11   //FTP 狀態
#define SCIF_FTP_RESULT            12   //FTP 處理結果
#define SCIF_FTP_STEP              13   //FTP 處理步驟
#define SCIF_FTP_TOTAL_PACKAGE     21   //FTP 傳送總封包數
#define SCIF_FTP_CURRENT_PACKAGE 22   //FTP 已處理的封包數
#define SCIF_FTP_TOTAL_FILE        31   //FTP 傳輸檔案
#define SCIF_FTP_CURRENT_FILE      32   //FTP 已處理的檔案數
```

回傳值

內部的資料內容，與 MsgID 有關。

範例

```
int Status;
Status = GetLibraryMsg (SCIF_FTP_RESULT);
```

取得連線資訊 GetConnectionMsg

此函式為從函式庫內部資訊取得連線資訊。

語法

int GetConnectionMsg(int TgrConn, int MsgID);

參數

TgrConn：
使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。

MsgID：可以為以下值，在 scif2_define.h 檔中定義。

```
#define SCIF_CONNECT_STATE              2   //連線狀態
#define SCIF_REMOTE_IPLONG              3   //目前的連線對象
#define SCIF_CONNECT_STEP               4   //連線步驟
#define SCIF_CONNECT_RESPONSE           5   //連線回應狀態
#define SCIF_TALK_STATE                 6   //資料通訊狀態
#define SCIF_RESPONSE_TIME              11  //目前封包的反應時間
#define SCIF_OK_COUNT                   12  //正確封包次數
#define SCIF_CRC_ERR_CNT                13  //CRC 錯誤次數
#define SCIF_LOOP_QUEUE_PKG_COUNT       21  //LOOP QUEUE 中的封
                                            包筆數
#define SCIF_DIRECT_QUEUE_PKG_COUNT 22  //Direct Queue 中的封包
                                            筆數
#define SCIF_LOOP_COUNT                 23  //LOOP QUEUE 的查詢
                                            迴圈次數
```

回傳值

內部的資料內容，與 MsgID 有關。

範例

```
int Status;
Status = GetConnectionMsg(0, SCIF_CONNECT_STATE);
```

取得連線錯誤資訊 GetConnectionError

此函式為從函式庫內部資訊取得錯誤訊息，而錯誤資料被讀取後即會立即被清除。

語法

void GetConnectionError (int TgrConn, ERROR_MSG *Msg);

參數

TgrConn：

使用的連線索引，使用者可自訂，但此值必須小於 LibraryInitial 函式初始化時，struct DLL_USE_SETTING 中 ConnectNum 所設定的連線數目。

Msg：

錯誤資料的結構指標，用來回傳錯誤內容，在 scif2_define.h 檔中定義。

```
struct ERROR_MSG
{
    unsigned char Type;
    unsigned char Cmd;
    unsigned int    addr;
    unsigned int    num;
    unsigned char Error;
};
```

回傳值

無。

範例

```
ERROR_MSG    Msg;
GetConnectionError (0, & Msg);
```