研華寶元 Linux 系列控制器連線函式庫使用觀念與說明

最大連線數：256

每個連線規格

最大檔案清單數目：300
循環(Polling)命令佇列個數：64
直接(Direct)命令佇列個數：64

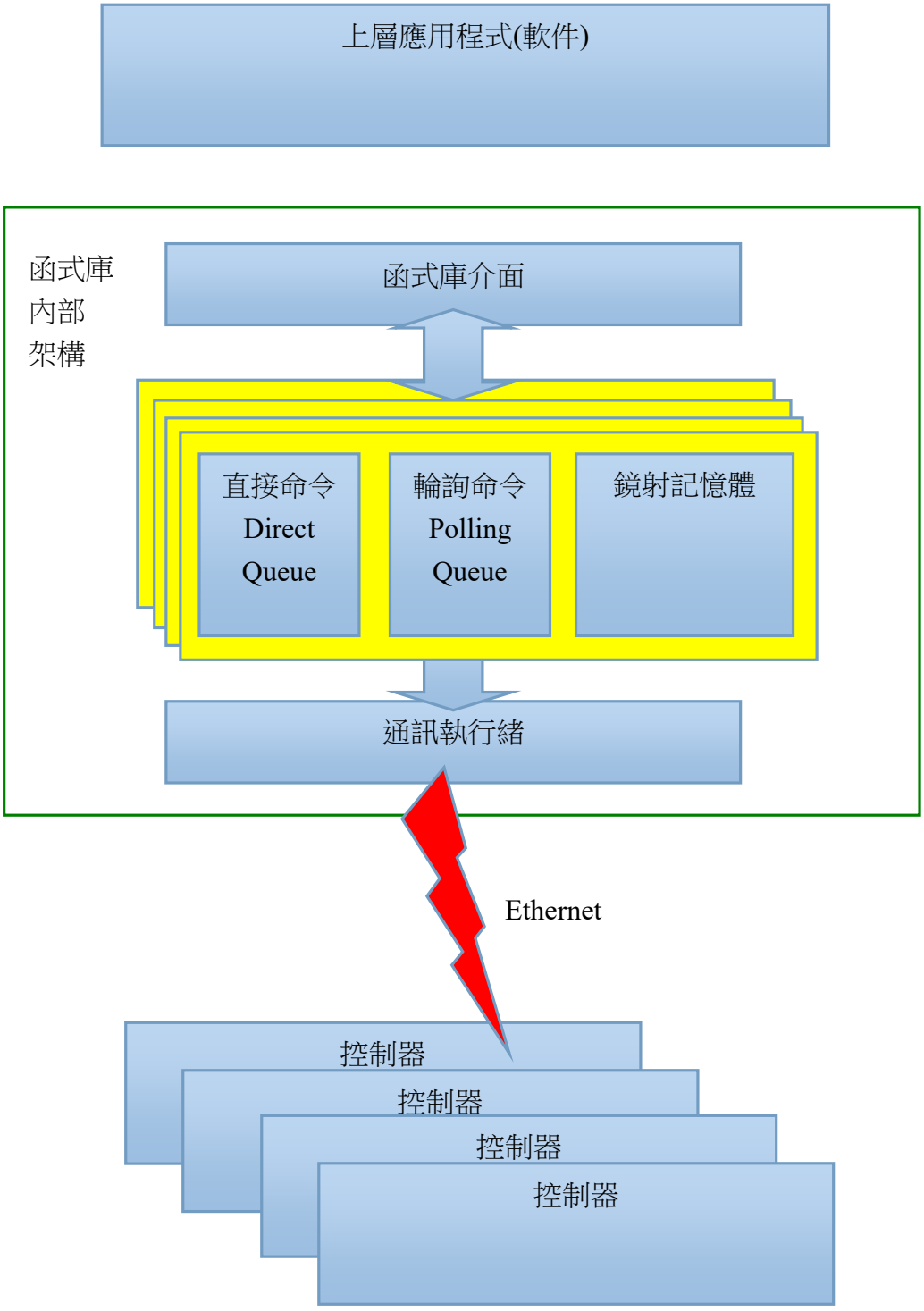
直接(Direct)命令 與 循環(Polling)命令 觀念說明：

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(Polling)：此類型的命令，會被存放到「循環命令佇列 Polling 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 scif_cmd_ClearAll」。循環命令又可分為兩區部份，DEFAULT 及 POLLING。

命令的優先順序

「直接(Direct)命令」的優先權是較「循環(Polling)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。

函式庫架構


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
Polling
Queue
鏡射記憶體
通訊執行緒
Ethernet
控制器
控制器
控制器
控制器

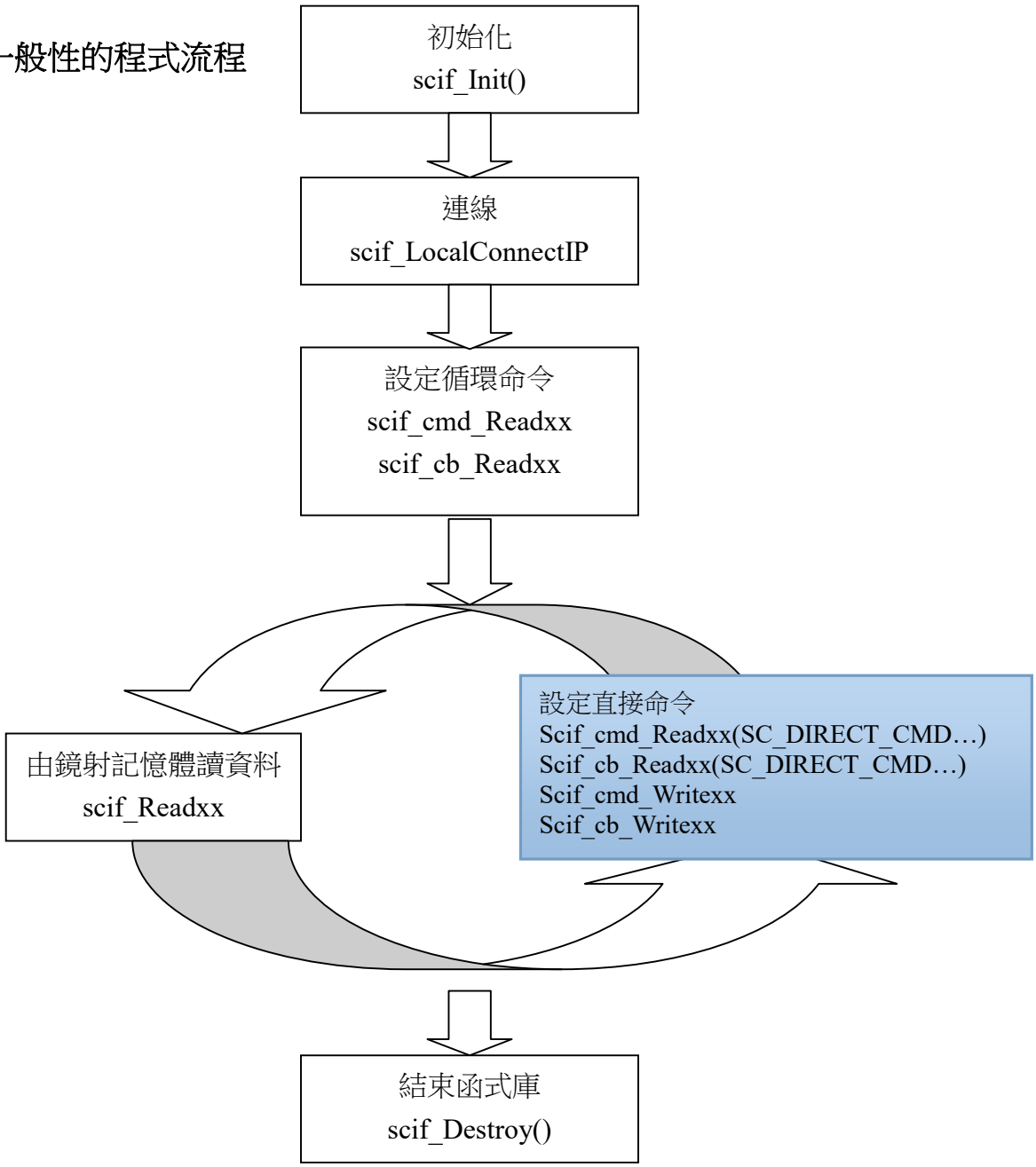
函式庫初始化說明：

為了管控製造商只能連線到屬於他的控制器，及一些其他的設定，函式庫需先執行初始化動作，初始化的內容包含

1. 所使用的控制器連線 1~5：每台控制器最多可同時支援五個連線，軟體連線時需設定其要連入的連線編號。
2. 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。
3. 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。
4. 製造商識別密碼字串：用來確認製造商身份，以確認能夠連入屬於其的控制器，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，函式庫才會建立通訊的執行緒。

一般性的程式流程

一般性的程式流程



控制器連線方式

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。

函式庫初始化 scif_Init

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

參數

pUseSetting：

函式庫的使用設定，可在 scif_define.h 檔中找到其 struct 的內容。其中的 SoftwareType 代表軟體種類，TalkInfoNum 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)
DllSetting.TalkInfoNum = 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.MemSizeTT   =   256;          //共使用  256 * 1byte
DllSetting.MemSizeCT   =   256;          //共使用  256 * 1byte
DllSetting.MemSizeR    =   10000;      //共使用  10000 * 4byte
DllSetting.MemSizeTS   =   256;        //共使用  256 * 4byte
DllSetting.MemSizeTV   =   256;        //共使用  256 * 4byte
DllSetting.MemSizeCS   =   256;        //共使用  256 * 4byte
DllSetting.MemSizeCV   =   256;        //共使用  256 * 4byte
DllSetting.MemSizeF    =   100000;          //共使用  100000* 8byte

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

偵測區網內的控制器 scif_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 scif_LocalReadControllerCount 與 scif_LocalReadController 兩函式時，不會有正確的回傳值。

語法

int scif_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 scif_LocalReadControllerCount 讀取。

範例

```
int Count;
Count = scif_LocalDetectControllers();
```

輸入連線密碼 scif_SetConnectPwd

當控制器沒有設定密碼時，不需使用此函式。

語法

int scif_SetConnectPwd (char SessionIdx, char *Pwd);

參數

SessionIdx：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 = scif_SetConnectPwd(0, “1234”);

取得偵測到的控制器數量

scif_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 scif_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 = scif_LocalReadControllerCount();
```

取得偵測到的控制器資訊

scif_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 scif_LocalReadController(unsigned short 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 scif_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;
unsigned short controllerindex;
Status = scif_LocalReadController(controllerindex, &Info);
```

連線到偵測到的控制器 scif_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 scif_LocalDetectControllers 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 scif_GetTalkMsg 函式來檢查連線狀態。

語法

int scif_ConnectLocalList(char ServerIdx, unsigned short Index);

參數

ServerIdx：

使用的連線索引，使用者可自訂，但此值必須小於 scif_Init 函式初始化時，struct DLL_USE_SETTING 中 TalkInfoNum 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

```
char serverindex;
unsigned short controllerindex;
int Status;
Status = scif_ ConnectLocalList (serverindex, controllerindex);
```

直接連線到指定 IP 的控制器 scif_LocalConnectIP

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 scif_GetTalkMsg 函式來檢查連線狀態。

語法

int scif_LocalConnectIP(char ServerIdx, char *IP);

參數

ServerIdx：

使用的連線索引，使用者可自訂，但此值必須小於 scif_Init 函式初始化時，struct DLL_USE_SETTING 中 TalkInfoNum 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

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

中斷連線 scif_Disconnect

呼叫此函式中斷與控制器的連線。

語法

int scif_Disconnect(char ServerIdx);

參數

ServerIdx：

使用的連線索引，使用者可自訂，但此值必須小於 scif_Init 函式初始化時，struct DLL_USE_SETTING 中 TalkInfoNum 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

```
char serverindex;
int Status;
Status = scif_ Disconnect(serverindex);
```

取得連線資訊 scif_GetTalkMsg

呼叫此函式可取得連線通訊的資訊。

語法

unsigned int scif_GetTalkMsg(char ServerIdx, char id);

參數

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

id：
scif_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；scif_define.h 檔中有回傳資訊內容代表的意義。

範例

```
char serverindex;
int Status;
Status = scif_GetTalkMsg(serverindex, 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。

結束函式庫 scif_Destroy

此函式為終結建立的記憶體與執行緒。

語法

void scif_Destroy();

參數

無。

回傳值

無。

範例

scif_Destroy();

通訊資料設定與讀取方式

此章節會說明如何建立與控制器間的通訊命令，並可根據使用者的需求分為連續和離散兩種通訊方式，連續的通訊設定意指讀取連續區間位址的設定(如位址區間為 0~10)，而離散的通訊方式則可一次輸入不相連位址的設定(如將位址設定為 1、4、13)。並提供鏡射記憶體和指標結構兩種方式來讀取所需的資料。在處理通訊資料的方面，若所要讀取值的位址太過分散、設定讀取函式時所讀取的數量太少，造成呼叫了多次讀取函式向控制器讀值，這種情況下會造成大量的通訊封包，且在輪詢資料時會較無效益，故我們提供了封包組合函式，有效的減少命令封包數量。在讀取資料方面，也提供了函式來組合讀取回來的資料，提供更好的資料處理效益。

通訊命令型態說明

- SC_DEFAULT_CMD：
  用於同時要監看多個連線控制器的資料時，此種命令會與 SC_POLLING_CMD 一同被輪流執行。

- SC_POLLING_CMD：
  用以持續同步更新控制器與 PC 端的資料。

- SC_DIRECT_CMD：
  此種命令執行過一次後即被刪除，且會被優先處理。

- 通訊命令為通訊資料設定中的參數，當通訊命令被設為 SC_DEFAULT_CMD 和 SC_POLLING_CMD 時，所設定要被讀取的位址資料，將會持續被更新，若通訊命令設為 SC_DIRECT_CMD，則要讀寫的位址資料只會被執行一次，在函式庫中，用來寫入控制器中資料的函式，預設的通訊命令皆為 SC_DIRECT_CMD。

設定讀取連續的 Bit 資料 Scif_cmd_Read(Bit)

包含的含式有：
scif_cmd_ReadI、scif_cmd_ReadO、scif_cmd_ReadC、scif_cmd_ReadS、scif_cmd_ReadA、scif_cmd_ReadTMR、scif_cmd_ReadCNT

這些函式皆是用來作為連續資料讀取的設定，根據資料存放類型的不同，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 scif_cmd_ReadI 為例來說明。

語法

int scif_cmd_ReadI(char type, char ServerIdx, unsigned int addr, unsigned int num);

參數

type：
命令型態 SC_DEFAULT_CMD，SC_POLLING_CMD，SC_DEFAULT_CMD。

ServerIdx：
使用的連線索引，使用者可自訂，但此值必須小於 scif_Init 函式初始化時，struct DLL_USE_SETTING 中 TalkInfoNum 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(scif_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

```
char serverindex;
unsigned int addr, num;
int pTran;
pTran = scif_cmd_ReadI(SC_POLLING_CMD, serverindex, addr, num);
```

設定讀取連續的 4Byte 資料 Scif_cmd_Read(4Byte)

包含的含式有：

scif_cmd_ReadR、scif_cmd_ReadTMRV、scif_cmd_ReadTMRS、scif_cmd_ReadCNTV、scif_cmd_ReadCNTS

這些函式皆是用來作為連續資料讀取的設定，根據資料存放類型的不同，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
scif_cmd_ReadR 為例來說明。

語法

int scif_cmd_ReadR(char type, char ServerIdx, unsigned int addr, unsigned int num);

參數

type：
命令型態 SC_DEFAULT_CMD，SC_POLLING_CMD，SC_DEFAULT_CMD。

ServerIdx：
使用的連線索引，使用者可自訂，但此值必須小於 scif_Init 函式初始化時，struct DLL_USE_SETTING 中 TalkInfoNum 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(scif_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

```
char serverindex;
unsigned int addr, num;
int pTran;
pTran = scif_cmd_ReadR(SC_POLLING_CMD, serverindex, addr, num);
```

設定讀取連續的 8Byte 資料 Scif_cmd_Read(8Byte)

包含的含式有：

scif_cmd_ReadF、scif_cmd_ReadP

這些函式皆是用來作為連續資料讀取的設定，根據資料存放類型的不同，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
scif_cmd_ReadF 為例來說明。

語法

int scif_cmd_ReadF(char type, char ServerIdx, unsigned int addr, unsigned int num);

參數

type：
命令型態 SC_DEFAULT_CMD，SC_POLLING_CMD，SC_DEFAULT_CMD。

ServerIdx：
使用的連線索引，使用者可自訂，但此值必須小於 scif_Init 函式初始化時，struct DLL_USE_SETTING 中 TalkInfoNum 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(scif_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

```
char serverindex;
unsigned int addr, num;
int pTran;
pTran = scif_cmd_ReadF(SC_POLLING_CMD, serverindex, addr, num);
```

設定讀取離散的 Bit 資料 scif_cb_Read(Bit)

包含的含式有：
scif_cb_ReadI、scif_cb_ReadO、scif_cb_ReadC、scif_cb_ReadS、scif_cb_ReadA、scif_cb_ReadTMR、scif_cb_ReadCNT

這些函式皆是用來作為離散資料讀取的設定，根據資料存放類型的不同，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
scif_cb_ReadI 為例來說明。

語法

int scif_cb_ReadI(char type, char ServerIdx, unsigned int num, unsigned int *addr);

參數

type：
命令型態 SC_DEFAULT_CMD，SC_POLLING_CMD，SC_DEFAULT_CMD。

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

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

addr：
要讀取的位址陣列指標，在陣列中填入所要讀取資料的位址。

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

```
char serverindex;
unsigned int num;
unsigned int addr[32];
int pTran;
pTran = scif_cb_ReadI(SC_POLLING_CMD, serverindex, num, addr);
```

設定讀取離散的 4Byte 資料 scif_cb_Read(4Byte)

包含的含式有：
scif_cb_ReadR、scif_cb_ReadTMRV、scif_cb_ReadTMRS、scif_cb_ReadCNTV、scif_cb_ReadCNTS

這些函式皆是用來作為離散資料讀取的設定，根據資料存放類型的不同，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
scif_cb_ReadR 為例來說明。

語法

int scif_cb_ReadR(char type, char ServerIdx, unsigned int num, unsigned int *addr);

參數

type：

命令型態 SC_DEFAULT_CMD，SC_POLLING_CMD，SC_DEFAULT_CMD。

ServerIdx：

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

num：

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

addr：

要讀取的位址陣列指標，在陣列中填入所要讀取資料的位址。

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

```
char serverindex;
unsigned int num;
unsigned int addr[32];
int pTran;
pTran = scif_cb_ReadR(SC_POLLING_CMD, serverindex, num, addr);
```

設定讀取離散的 8Byte 資料 scif_cb_Read(8Byte)

包含的含式有：

scif_cb_ReadF、scif_cb_ReadP

這些函式皆是用來作為離散資料讀取的設定，根據資料存放類型的不同，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 scif_cb_ReadF 為例來說明。

語法

int scif_cb_ReadF(char type, char ServerIdx, unsigned int num, unsigned int *addr);

參數

type：
命令型態 SC_DEFAULT_CMD，SC_POLLING_CMD，SC_DEFAULT_CMD。

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

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

addr：
要讀取的位址陣列指標，在陣列中填入所要讀取資料的位址。

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

```
char serverindex;
unsigned int num;
unsigned int addr[32];
int pTran;
pTran = scif_cb_ReadF(SC_POLLING_CMD, serverindex, num, addr);
```

取得某一封包的通訊狀態 scif_GetTranState

在呼叫完連續或離散通訊資料讀取設定的函式後，如果通訊封包沒有設定被重新組合，且欲以指標結構方式來讀取資料，則必須呼叫此函式取得通訊命令狀態，才能得知通訊命令有無被正確執行。

語法

unsigned char scif_GetTranState(int pTran);

參數

pTran：
連續或離散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

```
unsigned char command_status;
int pTran;
command_status = scif_GetTranState(pTran);
```

取得資料指標 scif_GetDefaultQueueDataPointer

若欲以指標結構方式來讀取資料，則必須呼叫此函式來取得 Default Queue 通訊命令資料指標，但在呼叫此函式前，必須呼叫 scif_GetTranState 函式確認通訊命令被正確執行。

語法

SC_DATA* scif_GetDefaultQueueDataPointer(char ServerIdx, unsigned char TranIdx);

參數

ServerIdx：

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

TranIdx：

Default 命令在設定時的順序，由 0 開始。

回傳值

資料結構指標，在 scif_define.h 檔中定義。

```
typedef union tag_SC_DATA
{
    unsigned char     Bytes[MAX_BIT_NUM];     //bit 或 byte 資料
    unsigned short    Words[MAX_WORD_NUM];    //word 資料
    unsigned int      Ints[MAX_INT_NUM];      //整數
    double            Fixs[MAX_FIX_NUM];      //double
    SC_POINTER        Ptrs[MAX_PTR_NUM];      //指標的值
}SC_DATA;
```

範例

```
char serverindex;
unsigned char tranindex;
SC_DATA *sc_data;
sc_data = scif_GetDefaultQueueDataPointer(serverindex, tranindex);
```

取得資料指標 scif_GetDataPointerByTranPointer

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 scif_GetTranState 函式確認通訊命令被正確執行，並執行完 scif_GetDefaultQueueDataPointer 函式。

語法

SC_DATA* scif_GetDataPointerByTranPointer(int TranPointer);

參數

TranPointer：

連續或離散通訊資料命令設定時函式回傳的通訊封包指標。

回傳值

資料結構指標，在 scif_define.h 檔中定義。

```
typedef union tag_SC_DATA
{
    unsigned char     Bytes[MAX_BIT_NUM];     //bit 或 byte 資料
    unsigned short    Words[MAX_WORD_NUM];    //word 資料
    unsigned int      Ints[MAX_INT_NUM];      //整數
    double            Fixs[MAX_FIX_NUM];      //double
    SC_POINTER        Ptrs[MAX_PTR_NUM];      //指標的值
}SC_DATA;
```

範例

```
SC_DATA *sc_data;
int pTran;
sc_data = scif_GetDataPointerByTranPointer(pTran);
```

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,TMR,CNT → Bytes。

R,TMRV,TMRS,CNTV,CNTS → Ints。

F → Fixs。

P → Ptrs。

組合通訊封包

清除讀取資料設定 scif_cmd_ClearAll

此函式是用來清除先前已設定的資料讀取，以便重新設定要讀取的內容。在人機包含很多頁面時，讀取得資料也會很多，為了加快頁面資料的更新速度，只需要更新該頁面所需的資訊，因此在切換頁面時，需將原本設定讀更新的內容清除，再重新設定。

語法

```
void scif_cmd_ClearAll(unsigned char type, char SessionIdx);
```

參數

Type : SC_DEFAULT_CMD 或 SC_POLLING_CMD
SessionIdx：連線索引。

回傳值

無。

範例

```
char serverindex;
scif_cmd_ClearAll (SC_POLLING_CMD);
```

設定開始組合封包 scif_StartCombineSet

此函式是用來作為同步資料的設定之用，呼叫此函式來設定自動組合旗標，呼叫此函式後，需再呼叫 scif_FinishCombineSet 函式，才能完成封包組合的設定。

語法

void scif_StartCombineSet(char ServerIdx);

參數

ServerIdx：

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

回傳值

無。

範例

```
char serverindex;
scif_StartCombineSet(serverindex);
```

設定完成組合封包 scif_FinishCombineSet

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 scif_StartCombineSet。下一章節有對封包組合作進一步的補充說明。

語法

void scif_FinishCombineSet(char ServerIdx);

參數

ServerIdx：

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

回傳值

無。

範例

```
char serverindex;
scif_FinishCombineSet(serverindex);
```

封包組合設定補充說明

- 呼叫 scif_StartComboin 之後，未呼叫 scif_FinishComboin 之前，若 scif_cmd_Readxxx 時傳入的 num 太小，該命令的內容將會被放到一暫存的 Buf，等到呼叫 scif_FinishComboin 之時，才會將 Buf 的內容自動重新整理以減少封包數量。每個組合封包的大小為 1440 個 byte，若為設定連續資料的函式，如 scif_cmd_ReadI、scif_cmd_ReadR、scif_cmd_ReadF 等，這些函式所讀取的資料量可經由其所設定的讀取位址數量和資料型態算出，如 ReadI 的資料型態回傳為 char 則只佔了 1 個 byte，ReadR 的資料回傳型態為 unsigned int 則佔了 4 個 byte，ReadF 的資料回傳型態為 double 則佔了 8 個 bytem。若為設定離散資料函式，如 scif_cb_ReadI、scif_cb_ReadR、scif_cb_ReadF 等，因為其離散資料必須在組合封包中加上其位址對應回傳值，而一個位址佔了 4 個 byte，故 ReadI 的離散資料型態回傳共佔了(4+1)個 byte，ReadR 的離散資料型態回傳佔了(4+4)個 byte，ReadF 的資料型態回傳佔了(4+8)個 byte。
  在範例中，所回傳的資料量一共是
  100*1 + 50*4 + 10*8 + 10*(4+1) + 20*(4+4) + 30*(4+8) = 950 個 byte，故可
  將這些回傳資料都放在一個組合封包內，以減少封包數量。
  範例

```
char serverindex;
unsigned int addr[32];
scif_StartCombineSet(serverindex);
scif_cmd_ReadI(SC_POLLING_CMD, serverindex, 0, 100);
scif_cmd_ReadR(SC_POLLING_CMD, serverindex, 10, 50);
scif_cmd_ReadF(SC_POLLING_CMD, serverindex, 20, 10);
scif_cb_ReadI(SC_POLLING_CMD, serverindex, 10, addr);
scif_cb_ReadR(SC_POLLING_CMD, serverindex, 20, addr);
scif_cb_ReadF(SC_POLLING_CMD, serverindex, 30, addr);
scif_FinishCombineSet(serverindex);
```

- 被重新組合的標準：
  命令型態為 SC_POLLING_CMD。
  I,O,C,S,A,TMR,CNT→num 小於等於 MAX_CB_BIT_NUM/2。
  R,TMRV,TMRS,CNTV,CNTS→num 小於等於 MAX_CB_INT_NUM/2。
  F,P→num 小於等於 MAX_CB_FIX_NUM/2。
  MAX_CB_BIT_NUM、MAX_CB_INT_NUM、MAX_CB_FIX_NUM 在
  scif_define.h 檔中定義。

由鏡射記憶體讀取資料

指定所使用的鏡射記憶體 SetMirror

在呼叫完連續或離散通訊資料讀取設定的函式後，若欲以鏡射記憶體方式來讀取資料，則必須先呼叫此函式來做鏡射記憶體的設定。在考慮會連線多台控制器的情況下，透過設定的方式，讓每個控制器對應到各自的鏡射記憶體，請注意在呼叫讀取鏡射記憶體的函式，如 scif_ReadXX 之前，務必要呼叫此函式，經由此函式的 ServerIdx 參數，來對應所要讀取該連線控制器的鏡射記憶體。

語法

int scif_SetMirror(char ServerIdx);

參數

ServerIdx：

使用的連線索引，使用者可自訂，但此值必須小於 scif_Init 函式初始化時，struct DLL_USE_SETTING 中 TalkInfoNum 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;
char serverindex;
Status = scif_SetMirror(serverindex);
```

讀取 Bit 型式資料

包含的函式有
scif_ReadI、scif_ReadO、scif_ReadC、scif_ReadS、scif_ReadA、scif_ReadTMR、scif_ReadCNT

這些函式皆是用來讀取鏡射記憶體資料的值，根據資料存放類型的不同，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 scif_ReadI 為例來說明。因考慮會有連線到多個控制器的情況下，在呼叫這些函式讀取資料前，必須先呼叫 scif_SetMirror 函式，來選擇所要讀取控制器的鏡射記憶體。

語法

char scif_ReadI(unsigned int addr);

參數

addr：
資料位址。

回傳值

資料內容。

範例

```
char data;
unsigned int addr;
data = char scif_ReadI(addr );
```

讀取 4Byte 型式資料

包含的函式有：

scif_ReadR、scif_ReadTMRV、scif_ReadTMRS、scif_ReadCNTV、scif_ReadCNTS

這些函式皆是用來讀取鏡射記憶體資料的值，根據資料存放類型的不同，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 scif_ReadR 為例來說明。因考慮會有連線到多個控制器的情況下，在呼叫這些函式讀取資料前，必須先呼叫 scif_SetMirror 函式，來選擇所要讀取控制器的鏡射記憶體。

語法

unsigned int scif_ReadR(unsigned int addr);

參數

addr：

資料位址。

回傳值

資料內容。

範例

unsigned int data, addr;

data = char scif_ ReadR (addr);

讀取 8Byte 型式資料

包含的函式有：
scif_ReadF

此函式是用來讀取鏡射記憶體資料的值，因考慮會有連線到多個控制器的情況下，在呼叫此函式讀取資料前，必須先呼叫 scif_SetMirror 函式，來選擇所要讀取控制器的鏡射記憶體。

語法
double scif_ReadF(unsigned int addr);

參數
addr：
資料位址。

回傳值
資料內容。

範例
unsigned int addr;
double data;
data = char scif_ ReadR (addr);

讀取 R 值組成的字串 scif_ReadRString

此函式是由鏡射記憶體中讀取字串，因考慮會有連線到多個控制器的情況下，在呼叫此函式讀取資料前，必須先呼叫 scif_SetMirror 函式，來選擇所要讀取控制器的鏡射記憶體。

語法

unsigned int scif_ReadRString(unsigned int addr, unsigned int BufSize, char *Buf);

參數

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

```
unsigned int num, addr, bufsize;
char buf[32];
num = scif_ReadRString( addr, bufsize, buf);
```

直接由連線鏡射讀取 Bit 型式資料

包含的函式有
scif_mem_ReadI、scif_ mem_ReadO、scif_ mem_ReadC、scif_ mem_ReadS、scif_ mem_ReadA、scif_ mem_ReadTMR、scif_ mem_ReadCNT

這些函式用來直接讀取某連線鏡射記憶體資料的值，不需先呼叫 scif_SetMirror 函式。

語法
char scif_ mem_ReadI(char SessionIdx, unsigned int addr);

參數
SessionIdx：連線索引。
addr：資料位址。

回傳值
資料內容。

範例
char data;
unsigned int addr;
data = char scif_ mem_ReadI(0, addr);

直接由連線鏡射讀取 4Byte 型式資料

包含的函式有：
scif_ mem_ReadR、scif_ mem_ReadTMRV、scif_ mem_ReadTMRS、scif_ mem_ReadCNTV、scif_ mem_ReadCNTS

這些函式用來直接讀取某連線鏡射記憶體資料的值，不需先呼叫 scif_SetMirror 函式。

語法
unsigned int scif_ mem_ReadR(char SessionIdx, unsigned int addr);

參數
SessionIdx：連線索引。
addr：資料位址。

回傳值
資料內容。

範例
unsigned int data, addr;
data = char scif_ mem_ReadR (0, addr);

直接由連線鏡射讀取 8Byte 型式資料

包含的函式有：
scif_ mem_ReadF

這些函式用來直接讀取某連線鏡射記憶體資料的值，不需先呼叫 scif_SetMirror 函式。

語法
double scif_ mem_ReadF(char SessionIdx, unsigned int addr);

參數
SessionIdx：連線索引。
addr：資料位址。

回傳值
資料內容。

範例
unsigned int addr;
double data;
data = char scif_ mem_ReadR (0, addr);

直接由連線鏡射讀取 R 值組成的字串

scif_ReadRString

這些函式用來直接讀取某連線鏡射記憶體資料的值，不需先呼叫 scif_SetMirror 函式。

語法

unsigned int scif_ mem_ReadRString(char SessionIdx, unsigned int addr, unsigned int BufSize, char *Buf);

參數

SessionIdx：連線索引。

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

```
unsigned int num, addr, bufsize;
char buf[32];
num = scif_ mem_ReadRString(0, addr, bufsize, buf);
```

執行直接命令

寫入 R 值字串 scif_cmd_WriteRString

此函式是寫入字串到控制器中的 R 值。

語法

int scif_cmd_WriteRString(char ServerIdx, unsigned int addr, unsigned int BufSize, char *Buf);

參數

ServerIdx：
使用的連線索引，使用者可自訂，但此值必須小於 scif_Init 函式初始化時，struct DLL_USE_SETTING 中 TalkInfoNum 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 Status;
char serverindex;
unsigned int addr, bufsize;
char buf[32];
Status = scif_cmd_WriteRString(serverindex, addr, bufsize, buf);
```

寫入一個 Bit 型式資料

包含的函式有：

scif_cmd_WriteO、scif_cmd_WriteC、scif_cmd_WriteA

這些函式皆是用來作為單筆資料的寫入，根據資料存放類型的不同，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 scif_cmd_WriteI 為例來說明。

語法

int scif_cmd_WriteI(char ServerIdx, unsigned int addr, char val);

參數

ServerIdx：

使用的連線索引，使用者可自訂，但此值必須小於 scif_Init 函式初始化時，struct DLL_USE_SETTING 中 TalkInfoNum 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 Status;
char serverindex;
unsigned int addr;
char val;
Status = scif_cmd_WriteI(serverindex, addr, val);
```

寫入一個 4Byte 型式資料

包含的函式有：
scif_cmd_WriteR

這些函式皆是用來作為單筆資料的寫入，根據資料存放類型的不同，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 scif_cmd_WriteR 為例來說明。

語法
int scif_cmd_WriteR(char ServerIdx, unsigned int addr, unsigned int val);

參數
ServerIdx：
使用的連線索引，使用者可自訂，但此值必須小於 scif_Init 函式初始化時，struct DLL_USE_SETTING 中 TalkInfoNum 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 Status;
char serverindex;
unsigned int addr;
unsigned int val;
Status = scif_cmd_WriteI(serverindex, addr, val);

寫入一個 8Byte 型式資料

包含的函式有：
scif_cmd_WriteF

此函式是用來作為單筆資料的寫入。

語法

int scif_cmd_WriteF(char ServerIdx, unsigned int addr, double val);

參數

ServerIdx：

使用的連線索引，使用者可自訂，但此值必須小於 scif_Init 函式初始化時，struct DLL_USE_SETTING 中 TalkInfoNum 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 Status;
char serverindex;
unsigned int addr;
double val;
Status = scif_cmd_WriteI(serverindex, addr, val);
```

寫入連續多個 Bit 型式資料

包含的函式有：
scif_cmd_WriteMultiO、scif_cmd_WriteMultiC、scif_cmd_WriteMultiA

這些函式皆是用來作為連續資料的寫入，根據資料存放類型的不同，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 scif_cmd_WriteMultiI 為例來說明。

語法

int scif_cmd_WriteMultiI(char ServerIdx, unsigned int addr, unsigned int num, char *data);

參數

ServerIdx：
使用的連線索引，使用者可自訂，但此值必須小於 scif_Init 函式初始化時，struct DLL_USE_SETTING 中 TalkInfoNum 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。

num：
要寫入的數量，最大值為 MAX_BIT_NUM(scif_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 Status;
char serverindex, data[32];
unsigned int addr, num;
Status = scif_cmd_WriteMultiI(serverindex, addr, num, data);
```

寫入連續多個 4Byte 型式資料

包含的函式有：
scif_cmd_WriteMultiR

這些函式皆是用來作為連續資料的寫入，根據資料存放類型的不同，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 scif_cmd_WriteMultiR 為例來說明。

語法

int scif_cmd_WriteMultiR(char ServerIdx, unsigned int addr, unsigned int num, unsigned int *data);

參數

ServerIdx：
使用的連線索引，使用者可自訂，但此值必須小於 scif_Init 函式初始化時，struct DLL_USE_SETTING 中 TalkInfoNum 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。

num：
要寫入的數量，最大值為 MAX_INT_NUM (scif_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 Status;
char serverindex;
unsigned int addr, num, data[32];
Status = scif_cmd_WriteMultiI(serverindex, addr, num, data);
```

寫入連續多個 8Byte 型式資料

包含的函式有：

scif_cmd_WriteMultiF

此函式是用來作為連續資料的寫入。

語法

int scif_cmd_WriteMultiF(char ServerIdx, unsigned int addr, unsigned int num, double *data);

參數

ServerIdx：

使用的連線索引，使用者可自訂，但此值必須小於 scif_Init 函式初始化時，struct DLL_USE_SETTING 中 TalkInfoNum 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。

num：

要寫入的數量，最大值為 MAX_INT_NUM (scif_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 Status;
char serverindex;
unsigned int addr, num;
double data[32];
Status = scif_cmd_WriteMultiF(serverindex, addr, num, data);
```

寫入離散多個 Bit 型式資料

包含的函式有：

scif_cb_WriteO、scif_cb_WriteC、scif_cb_WriteA

這些函式皆是用來作為離散位址資料的寫入，根據資料存放類型的不同，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 scif_cb_WriteO 為例來說明。

語法

int scif_cb_WriteO(char ServerIdx, unsigned int num, unsigned int *addr, char *data);

參數

ServerIdx：

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

num：

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

addr：

要寫入的位址陣列指標。

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 Status;
char serverindex, data[32];
unsigned int addr[32], num;
Status = scif_cb_WriteI(serverindex, num, addr, data);
```

寫入離散多個 4Byte 型式資料

包含的函式有：

scif_cb_WriteR

這些函式皆是用來作為離散位址資料的寫入，根據資料存放類型的不同，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 scif_cb_WriteR 為例來說明。

語法

int scif_cb_WriteR(char ServerIdx, unsigned int num, unsigned int *addr, unsigned int *data);

參數

ServerIdx：

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

num：

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

addr：

要寫入的位址陣列指標。

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 Status;
char serverindex;
unsigned int addr[32], num, data[32];
Status = scif_cb_WriteR(serverindex, num, addr, data);
```

寫入離散多個 8Byte 型式資料

包含的函式有：

scif_cb_WriteF

此函式是用來作為離散位址資料的寫入。

語法

int scif_cb_WriteF(char ServerIdx, unsigned int num, unsigned int *addr, double *data);

參數

ServerIdx：

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

num：

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

addr：

要寫入的位址陣列指標。

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 Status;
char serverindex;
unsigned int addr[32], num;
double data[32];
Status = scif_cb_WriteF(serverindex, num, addr, data);
```

寫入一個 RBit　scif_cmd_WriteRBit

此函式是用來作為 R 值單個 bit 位址資料的寫入。

語法

int scif_cmd_WriteRBit(char ServerIdx, unsigned int addr, unsigned char BitIdx, unsigned char BitValue);

參數

ServerIdx：

使用的連線索引，使用者可自訂，但此值必須小於 scif_Init 函式初始化時，struct DLL_USE_SETTING 中 TalkInfoNum 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 Status;
char serverindex;
unsigned int addr;
unsigned char bitindex, bitvalue;
Status = scif_cmd_WriteRBit(serverindex, addr, bitindex, bitvalue);
```

寫入離散多個 RBit scif_cb_WriteRBit

此函式是用來作為多個 R 值 bit 位址資料的寫入。

語法

int scif_cb_WriteRBit(char ServerIdx, unsigned int num, unsigned int *addr, unsigned char *BitIdx, unsigned char *BitValue);

參數

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

num：
要寫入 R 值的數量。

addr：
存放 R 的位址的陣列指標。

BitIdx：
存放位元位址的陣列指標。

BitValue：
存放設定值的陣列指標，設定值為 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 Status;
char serverindex;
unsigned int num, addr[32];
unsigned char bitindex[32], bitvalue[32];
Status = scif_cb_WriteRBit(serverindex, num, addr, bitindex, bitvalue);
```

等待直接命令執行完成 scif_WaitDirectCmdDone

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 scif_WaitDirectCmdDone (char SessionIdx, unsigned int MaxWaitTime);;

參數

ServerIdx：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;
rt = scif_ WaitDirectCmdDone(0, 2000);
if (rt==1)
        …
else
        ShowMessage(“Timeout”);
```

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 scif_FtpSetTalk

此函式用來設定後續檔案傳輸指令所對應的連線。

語法

int scif_FtpSetTalk(char ServerIdx);

參數

ServerIdx：

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

回傳值

0：設定失敗 。

1：設定成功。

範例

```
char serverindex;
int Status;
Status = scif_FtpSetTalk(serverindex);
```

上傳一個檔案 scif_FtpUploadFile

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 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_FtpUploadFile(char Folder, char *Filename, char *LocalFilename);

參數

Folder：

控制器所要上傳檔案的目標資料夾，可在 scif_define.h 檔中找到定義，

例：#define FTP_FOLDER_NCFILES 10

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, filename[32], localfilename[32];
int Status;
Status = scif_FtpUploadFile(folder, filename, localFilename);
```

下載一個檔案 scif_FtpDownloadFile

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_FtpDownloadFile(char Folder, char *Filename, char *LocalFilename);

參數

Folder：

控制器所要下載檔案的目標資料夾，可在 scif_define.h 檔中找到定義，

例：#define FTP_FOLDER_NCFILES 10

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, filename[32], localfilename[32];
int Status;
Status = scif_FtpDownloadFile(folder, filename, localFilename);
```

刪除一個檔案 scif_FtpDeleteFile

此函式為刪除控制器上的檔案。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_FtpDeleteFile(char Folder, char *Filename);

參數

Folder：

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

例：#define FTP_FOLDER_NCFILES 10

Filename：

檔案名稱。

回傳值

0：設定失敗 。

1：設定成功。

範例

```
char folder, filename[32];
int Status;
Status = scif_FtpDeleteFile(folder, filename);
```

上傳多個檔案 scif_FtpUploadFiles

此函式為上傳多個檔案到控制器的資料夾。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_FtpUploadFiles(unsigned char Count, FTP_TRANFER_FILE *TransferFiles);

參數

Count：

所要上傳的檔案數量。

TransferFiles：

檔案傳輸的設定資料結構指標，在 scif_define.h 檔中定義。

```
typedef struct tag_FTP_TRANFER_FILE
{
    char Folder;
    char Filename[FILENAME_LENGTH];      //檔案名稱的最大字元數
    char LocalFilename[256];
}FTP_TRANFER_FILE;
```

回傳值

0：設定失敗 。

1：設定成功。

範例

```
unsigned char count;
FTP_TRANFER_FILE TransferFiles[8];
int Status;
Status = scif_FtpUploadFiles(count, TransferFiles);
```

下載多個檔案 scif_FtpDownloadFiles

此函式為從控制器的資料夾下載多個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_FtpDownloadFiles(unsigned char Count, FTP_TRANFER_FILE *TransferFiles);

參數

Count：

所要下載的檔案數量。

TransferFiles：

檔案傳輸的設定資料結構指標，在 scif_define.h 檔中定義。

```
typedef struct tag_FTP_TRANFER_FILE
{
    char Folder;
    char Filename[FILENAME_LENGTH];      //檔案名稱的最大字元數
    char LocalFilename[256];
}FTP_TRANFER_FILE;
```

回傳值

0：設定失敗 。

1：設定成功。

範例

```
unsigned char count;
FTP_TRANFER_FILE TransferFiles[8];
int Status;
Status = scif_FtpDownloadFiles(count, TransferFiles);
```

刪除多個檔案 scif_FtpDeleteFiles

此函式為刪除控制器資料夾內的多個檔案。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_FtpDeleteFiles(unsigned char Count, FTP_TRANFER_FILE *TransferFiles);

參數

Count：

所要刪除的檔案數量。

TransferFiles：

檔案傳輸的設定資料結構指標，在 scif_define.h 檔中定義。

```
typedef struct tag_FTP_TRANFER_FILE
{
    char Folder;
    char Filename[FILENAME_LENGTH];      //檔案名稱的最大字元數
    char LocalFilename[256];
}FTP_TRANFER_FILE;
```

回傳值

0：設定失敗 。

1：設定成功。

範例

```
unsigned char count;
FTP_TRANFER_FILE TransferFiles[8];
int Status;
Status = scif_FtpDeleteFiles(count, TransferFiles);
```

檢查檔案傳輸指令完成 scif_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 scif_FtpCheckDone(unsigned char *State, unsigned char *Result);

參數

State：

用以回傳檔案傳輸最後狀態。在 scif_define.h 檔中定義。

Result：

用以回傳檔案傳輸結果，在 scif_define.h 檔中定義。

回傳值

1：要求執行的動作已完成。

0：未完成。

範例

```
unsigned char state, result;
int Status;
Status = scif_FtpCheckDone(&state, &result);
```

建立資料夾 scif_FtpMakeDir

在控制器上建立資料夾，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_FtpMakeDir(char Folder, char *DirName);

參數

Folder：

控制器所的目標資料夾，可在 scif_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

```
char folder, dirname[32];
int Status;
Status = scif_FtpMakeDir(folder, dirname);
```

取得檔案清單 scif_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 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_FtpGetFileList(char Folder, char *HeadFilter, char *TailFilter);

參數

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

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, headfilter [8], tailfilter[8];
int Status;
Status = scif_FtpGetFileList(folder, headfilter, tailfilter);
```

讀取檔案個數 scif_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 scif_FtpGetFileList 函式。

語法

int scif_FtpReadFileCount();

參數

無。

回傳值

檔案清單中的檔案個數。

範例

```
int Status;
Status = scif_FtpReadFileCount();
```

讀取檔案資訊 scif_FtpReadFile

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

語法

int scif_FtpReadFile(unsigned short 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 scif_define.h 檔中定義。

```
typedef struct tag_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;
}FTP_FILE;
```

回傳值

0：命令設定失敗。

1：命令設定成功。

範例

```
unsigned short index;
FTP_FILE file
int Status;
Status = scif_FtpReadFile(index, &file);
```

取得本地檔案清單 scif_FileGetFileList

取得本地端檔案清單，並建立起檔案索引。

語法

int scif_FileGetFileList(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[32], headfilter [8], tailfilter[8];
int Status;
Status = scif_FileGetFileList(path, headfilter, tailfilter);
```

讀取本地檔案個數 scif_FileReadFileCount

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 scif_FileGetFileList 函式。

語法

int scif_FileReadFileCount();

參數

無。

回傳值

檔案清單中的檔案個數。

範例

```
int Status;
Status = scif_FileReadFileCount();
```

讀取本地檔案資訊 scif_FileReadFile

讀取本地端檔案名稱，呼叫此函式前，必須執行過 scif_FileGetFileList 函式。

語法

int scif_FileReadFile(unsigned short 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 scif_define.h 檔中定義。

```
typedef struct tag_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;
}FTP_FILE;
```

回傳值

0：命令設定失敗。

1：命令設定成功。

範例

```
unsigned short index;
FTP_FILE file
int Status;
Status = scif_FileReadFile(index, &file);
```

刪除本地檔案 scif_FileDeleteFile

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

語法

int scif_FileDeleteFile(unsigned short 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

unsigned short index;

int Status;

Status = scif_FileDeleteFile(index);

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 scif_GetCommonMsg

此函式為從函式庫內部資訊取得一般的資訊。

語法

unsigned int scif_GetCommonMsg(char id);

參數

id：可以為以下值，在 scif_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_MEDIA_STATE           6   //與媒合主機通訊的結果
#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 id 有關。

範例

```
unsigned int Status;
char id;
Status = scif_GetCommonMsg(id);
```

取得連線資訊 scif_GetTalkMsg

此函式為從函式庫內部資訊取得連線資訊。

語法

unsigned int scif_GetTalkMsg(char ServerIdx, char id);

參數

ServerIdx：

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

id：可以為以下值，在 scif_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 id 有關。

範例

```
unsigned int Status;
char serverindex, id;
Status = scif_GetTalkMsg(serverindex, id);
```

取得連線錯誤資訊 scif_GetTalkError

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 scif_GetTalkError(char ServerIdx, ERROR_MSG *Msg);

參數

ServerIdx：

使用的連線索引，使用者可自訂，但此值必須小於 scif_Init 函式初始化時，struct DLL_USE_SETTING 中 TalkInfoNum 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 scif_define.h 檔中定義。

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

回傳值

無。

範例

scif_GetTalkError()

附錄 A scif_define.h 檔內容

```
//功能限定的定義
#define RECON_CONFIGURE            1
#define RECON_DEBUGGER             2
#define RECON_FTP                  3
#define RECON_HMI                  4
#define RECON_SHOPFLOOR            5
//--------------------------------
#define MAX_SYNC_COUNT                          10        //Mapper 中所用的最大 Sync 數
#define MAX_CONTROLLER_NUM_PER_MAKER   300              //向 Media 讀取控制器清單時，最大的允許數量
//--------------------------------
#define BIT_CB_SIZE                4096      //Combin 封包中包含的最大位址數 for Bit(I,O,C,S,A)
#define INT_CB_SIZE                4096 //Combin 封包中包含的最大位址數 for Int(R)
#define FIX_CB_SIZE                4096      //Combin 封包中包含的最大位址數 for Fix(double)
#define MAX_DEFAULT_SIZE           32          //Loop Queue 可容納的通訊筆數
#define MAX_POLLING_SIZE          64           //Loop Queue 可容納的通訊筆數
#define MAX_DIRECT_SIZE            64          //Direct Queue 可容納的通訊筆數
#define DIRECT_ADDR_MASK          0x3F      //Direct 位址的 mask，要與 MAX_DIRECT_SIZE 搭配
//--------------------------------
#define MAX_DATA_BYTES             1440
#define MAX_BIT_NUM                1440 // MAX_DATA_BYTES / 1
#define MAX_WORD_NUM               720       // MAX_DATA_BYTES / 2
#define MAX_INT_NUM           360       // MAX_DATA_BYTES / 4
#define MAX_FIX_NUM           180       // MAX_DATA_BYTES / 8
#define MAX_PTR_NUM           180       // MAX_DATA_BYTES / 8
#define MAX_CB_NUM            288       //MAX_DATA_BYTES /(4+1)
                                          //位址 4bytes, data 為 byte 時(1byte)
#define MAX_CB_BIT_NUM        288       // MAX_DATA_BYTES /(4+1)
#define MAX_CB_WORD_NUM            240       // MAX_DATA_BYTES /(4+2)
#define MAX_CB_INT_NUM             180   // MAX_DATA_BYTES /(4+4)
#define MAX_CB_FIX_NUM        120       // MAX_DATA_BYTES /(4+8)
#define MAX_CB_PTR_NUM             120       // MAX_DATA_BYTES /(4+8)

#define I_OFFSET                        0
#define O_OFFSET                   (5120*1)
#define C_OFFSET                   (5120*2)
#define S_OFFSET                   (5120*3)
```

```
#define A_OFFSET                (5120*4)
#define TT_OFFSET               (5120*5)
#define CT_OFFSET               (5120*6)
#define RBIT_OFFSET             100000

#define R_OFFSET                0
#define TV_OFFSET               10000000
#define TS_OFFSET               10500000
#define CV_OFFSET               11000000
#define CS_OFFSET               11500000
#define F_OFFSET                10000000

#define I_NUM                   4096
#define O_NUM                   4096
#define C_NUM                   4096
#define S_NUM                   4096
#define A_NUM                   4096
#define TT_NUM                  256
#define CT_NUM                  256
#define R_NUM                   6000000
#define TV_NUM                  256
#define TS_NUM                  256
#define CV_NUM                  256
#define CS_NUM                  256
#define F_NUM                   100000

//---------scif_GetCommonMsg 的引數
#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_MEDIA_STATE        6   //與媒合主機通訊的結果
#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

//-----scif_GetTalkMsg 的引數
#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

//連線狀態 由   scif_GetTalkMsg(SCIF_CONNECT_STATE)   取得
#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

//------------連線回應狀態
#define CONNECT_RESULT_NORESPONSE                0
#define CONNECT_RESULT_OLD_SOFTWARE_CLEAR    1 //原本佔用的軟體已經清除
#define CONNECT_RESULT_INVALID_SOFTWARE      11//無效的軟體代號
#define CONNECT_RESULT_DISABLE_SOFTWARE      12//軟體功能停用
#define CONNECT_RESULT_DISABLE_INTERNET      13 //停用自外網來的連線
#define CONNECT_RESULT_CLOSED_SOFTWARE       14   //軟體功能關閉
#define CONNECT_RESULT_CLOSED_INTERNET       15  //關閉自外網來的連線
#define CONNECT_RESULT_INVALID_MAKERID       16   //不相符的 MakerID
#define CONNECT_RESULT_WAIT_CONFIRM          21  //等待人機確認中
#define CONNECT_RESULT_SOFTWARE_CONNECTED    31  //軟體已連線
#define CONNECT_RESULT_SOFTWARE_REJECTED     32  //停用自外網來的連線
#define CONNECT_RESULT_SOFTWARE_INREQ        41  //其他使用者佔用此軟體
#define CONNECT_RESULT_PENDING               50

//-----FTP 目標資料夾
#define FTP_FOLDER_NCFILES                   10
#define FTP_FOLDER_MACRO_MAKER               20
#define FTP_FOLDER_MACRO                     21
#define FTP_FOLDER_MACHINE                   30
#define FTP_FOLDER_SETUP                     40
#define FTP_FOLDER_SETUP_MACHINE             41
#define FTP_FOLDER_BAK                       50
#define FTP_FOLDER_DATA                      51
```

```
#define FTP_FOLDER_LANGUAGE                 52
#define FTP_FOLDER_LANGUAGE_DEF             53
#define FTP_FOLDER_LOG                      54
#define FTP_FOLDER_RECORD                   55

//------FTP 狀態
#define FTP_STATE_IDLE                      0   //閒置
#define FTP_STATE_UPLOAD                    1   //上傳
#define FTP_STATE_DOWNLOAD                  2   //下載
#define FTP_STATE_DELETE                    3   //刪除
#define FTP_STATE_LIST                      11   //取得目錄
#define FTP_STATE_UPLOAD_MANY               21   //上傳多個
#define FTP_STATE_DOWNLOAD_MANY             22   //下載多個
#define FTP_STATE_DELETE_MANY               23   //刪除多個
#define FTP_STATE_MAKE_DIR                  30   //建立目錄
#define FTP_STATE_PENDING                   99   //命令設定中

//------FTP 處理結果
#define FTP_RESULT_IDLE                     0    //無
#define FTP_RESULT_PROCESSING               1    //處理中
#define FTP_RESULT_SUCCESS                  2    //成功
#define FTP_RESULT_FAIL_TO_READ_LOCAL_FILE  11   //讀取本地檔案失敗
#define FTP_RESULT_FAIL_TO_WRITE_LOCAL_FILE 12   //寫入本地檔案失敗
#define FTP_RESULT_FAIL_TO_READ_REMOTE_FILE 13    //讀取遠端檔案失敗
#define FTP_RESULT_FAIL_TO_WRITE_REMOTE_FILE 14  //寫入遠端檔案失敗
#define FTP_RESULT_FAIL_TO_SET_COMMAND      15    //命令傳送失敗
#define FTP_RESULT_FAIL_TO_COMMUNICATION    16    //通訊錯誤
#define FTP_RESULT_FILE_MISMATCH            17   //檔案比對不正確

//------MEDIA 處理結果
#define MEDIA_RESULT_IDLE                   0
#define MEDIA_RESULT_PENDING                1
#define MEDIA_RESULT_PROCESSING             2
#define MEDIA_RESULT_SUCCESS                3
#define MEDIA_RESULT_FAIL                   4

//------------資料通訊狀態
#define TALK_STATE_NORMAL                   0
#define TALK_STATE_ERROR                    1
#define TALK_STATE_OVER_RETRY               2
```

```
//------單筆通訊資料的狀態
#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
//------通訊封包錯誤編號
//錯誤碼為 0                                              //沒有發生錯誤
#define SCIF_ERROR_INVALID_PACKET_SET   255         //Local 檢查到此封包設定無效
//其他編號的錯誤碼由主機傳回的錯誤---直接記錄代碼即可
//---一些定義
#define FILENAME_LENGTH                 32   //檔案名稱的最大字元數
#define MAX_FILE_LIST_NUM               240   //最大的檔案清單大小
#define MAX_TRANSFER_FILE_COUNT         128   //一次傳輸的最大檔案量
#define MAX_SOFTWARE_COUNT              5     //最大的軟體種類數

//錯誤訊息來源
#define ERROR_TYPE_NONE                 0
#define ERROR_TYPE_POLLING              1
#define ERROR_TYPE_DIRECT               2

//命令種類
#define SC_DEFAULT_CMD          0    // Default Command--當要在畫面上同時顯示多個控制器的資訊時，應使用此種
                                封包
#define SC_POLLING_CMD          1    // Polling Command
#define SC_DIRECT_CMD           2    // Direct read & setting

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

//指標資料的結構
typedef struct tag_SC_POINTER
{
       unsigned int PtrA;
        unsigned int PtrV;
}SC_POINTER;
```

```
//單筆通訊的資料結構
typedef union tag_SC_DATA
{
    unsigned char     Bytes[MAX_BIT_NUM];     //bit 或 byte 資料
    unsigned short    Words[MAX_WORD_NUM];    //word 資料
    unsigned int      Ints[MAX_INT_NUM];      //整數
    double            Fixs[MAX_FIX_NUM];      //double
    SC_POINTER        Ptrs[MAX_PTR_NUM];      //指標的值
}SC_DATA;

//自動偵測主機的回應封包
typedef struct tag_LOCAL_CONTROLLER_INFO1
{
    unsigned int IPLong;
    char              IP[16];
    char              Name[16];
}LOCAL_CONTROLLER_INFO;

//由媒介主機取回的控制器資訊
typedef struct tag_MEDIA_CONTROLLER_INFO1
{
     int              Idx;
    unsigned int CtrID;
    unsigned int      Port;
    unsigned int IPLong;
    char              IP[16];
    char              Name[16];
}MEDIA_CONTROLLER_INFO;

//FTP 或本地列舉檔案清單傳回的檔案資訊
typedef struct tag_FTP_FILE
{
     char             filename[FILENAME_LENGTH];
    unsigned int      filesize;
    unsigned short    year;
    unsigned char     month;
    unsigned char     day;
    unsigned char     hour;
    unsigned char     minute;
    unsigned char     second;
    unsigned char     Reserve;
```

```
}FTP_FILE;

//FTP 檔案傳輸的設定資料
typedef struct tag_FTP_TRANFER_FILE
{
     char Folder;
    char Filename[FILENAME_LENGTH];
    char LocalFilename[256];
}FTP_TRANFER_FILE;

typedef struct tag_DLL_USE_SETTING
{
    unsigned int    TalkInfoNum;            //連線數目
    unsigned int    SoftwareType;           //軟體種類
    unsigned int    MemSizeI;
    unsigned int        MemSizeO;
    unsigned int        MemSizeC;
    unsigned int        MemSizeS;
    unsigned int        MemSizeA;
    unsigned int        MemSizeTT;
    unsigned int        MemSizeCT;
    unsigned int        MemSizeR;
    unsigned int        MemSizeTV;
    unsigned int        MemSizeTS;
    unsigned int        MemSizeCV;
    unsigned int        MemSizeCS;
    unsigned int        MemSizeF;
}DLL_USE_SETTING;
```