



史卡拉機器手臂控制器 連線函式庫

使用觀念與說明





多軸機器人

Multi Axis Robot

取放作業/組裝/整列與包裝/半導體/光電業/汽車工業/食品業

- 關節式機器手臂
- 並聯式機器手臂
- 史卡拉機器手臂
- 晶圓機器人
- 電動夾爪
- 整合型電爪
- 旋轉接頭



單軸機器人

Single Axis Robot

高精密產業/半導體/醫療自動化/FPD面板搬運

- KK, SK
- KS, KA
- KU, KE, KC



直驅馬達迴轉工作台

Direct Drive Rotary Table

航太/醫療/汽車工業/工具機/產業機械

- RAB系列
- RAS系列
- RCV系列
- RCH系列



滾珠螺桿

Ballscrew

精密研磨/精密轉造

- Super S 系列 (高 Dm-N 值/高速化)
- Super T 系列 (低噪音/低振動)
- 微小型研磨級
- E2 環保潤滑模組
- R1 螺帽旋轉式
- C1 節能溫控螺桿
- RD 高DN節能重負荷
- 滾珠花鍵

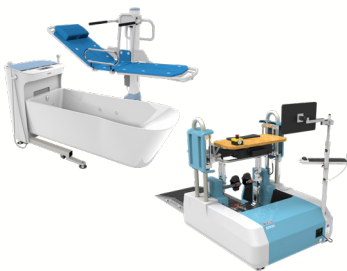


線性滑軌

Linear Guideway

精密機械/電子半導體/生技醫療

- 滾珠式—
- HG重負荷型, EG低組裝, WE寬幅型, MG微小型, CG扭矩型
- 靜音式—
- QH重負荷型, QE低組裝型, QW寬幅型, QR滾柱型
- 其他—
- RG滾柱型, E2自潤型, PG定位型, SE金屬端蓋型, RC強化型



醫療設備

Medical Equipment

醫療院所/復健中心/療養中心

- 下肢肌力訓練機
- 沐浴水療系統
- 內視鏡扶持機器手臂



特殊軸承

Bearing

工具機產業/機械手臂

- 交叉滾柱軸承
- 滾珠螺桿軸承
- 精密線性軸承
- 軸承座



AC伺服馬達&驅動器

AC Servo Motor & Drive

半導體設備/包裝機/SMT機台/食品業機台/LCD設備

- 驅動器—D1, D1-N, D2T
- 伺服馬達—50W-2000W

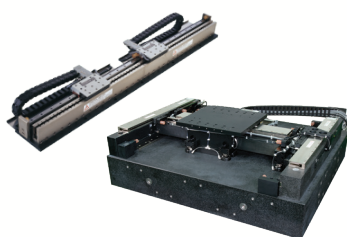


動力刀座

Driven Tool Holders

各式刀塔

- VDI系統
- 軸向動力刀座, 軸向偏心動力刀座, 徑向動力刀座, 徑向縮頭動力刀座, MT
- BMT系統
- DS, NM, GW, F0, MT, OM, MS



線性馬達

Linear Motor

自動化搬運/AOI光學檢測/精密加工/電子半導體

- 鐵心式線性馬達
- 無鐵心式線性馬達
- 棒狀線性馬達
- 平面馬達
- 空氣軸承定位平台
- X-Y平台
- 龍門系統



轉矩馬達

Torque Motor

(Direct Drive Motor)

檢測設備/工具機/機器人

- 旋轉平台系列—TMS, TMY, TMN
- 水冷式系列—TMRW
- 高轉速水冷系列—TMRI

版權聲明：

本手冊內容僅供授權合法持有本手冊的客戶與廠商學習、參考與使用。非經上銀科技股份有限公司正式授權，不得以任何形式複製、抄襲、轉載或透過網路散佈其內容。

版權所有，翻印必究

目錄

	頁數
1. 史卡拉機械手控制器連線函式庫使用觀念與說明.....	1
1.1. 函式庫架構.....	2
1.2. 函式庫初始化.....	3
1.3. 一般性的程式流程.....	4
2. 控制器連線方式.....	5
2.1. scif_Init.....	6
2.2. scif_Destroy.....	8
2.3. scif_LocalDetectControllers.....	9
2.4. scif_LocalReadControllerCount.....	10
2.5. scif_LocalReadController.....	11
2.6. scif_ConnectLocalList.....	12
2.7. scif_LocalConnectIP.....	13
2.8. scif_Disconnect.....	14
2.9. scif_GetTalkMsg.....	15
2.10. 連線狀態補充.....	16
3. 通訊資料設定與讀取方式.....	17
3.1. 通訊命令型態說明.....	18
3.2. scif_cmd_ReadI, scif_cmd_ReadO, scif_cmd_ReadC, scif_cmd_ReadS, scif_cmd_ReadA,scif_cmd_ReadTMR, scif_cmd_ReadCNT.....	19
3.3. scif_cmd_ReadR, scif_cmd_ReadTMRV, scif_cmd_ReadTMRS, scif_cmd_ReadCNTV, scif_cmd_ReadCNTS.....	20
3.4. scif_cmd_ReadF, scif_cmd_ReadP.....	21
3.5. scif_cb_ReadI, scif_cb_ReadO, scif_cb_ReadC, scif_cb_ReadS, scif_cb_ReadA, scif_cb_ReadTMR, scif_cb_ReadCNT.....	22
3.6. scif_cb_ReadR, scif_cb_ReadTMRV, scif_cb_ReadTMRS, scif_cb_ReadCNTV, scif_cb_ReadCNTS.....	23
3.7. scif_cb_ReadF, scif_cb_ReadP.....	24
3.8. scif_GetTranState.....	25
3.9. scif_GetDefaultQueueDataPointer.....	26
3.10. scif_GetDataPointerByTranPointer.....	27
3.11. scif_SetMirror.....	28
3.12. scif_ReadI, scif_ReadO, scif_ReadC, scif_ReadS, scif_ReadA, scif_ReadTMR, scif_ReadCNT.....	29
3.13. scif_ReadR, scif_ReadTMRV, scif_ReadTMRS, scif_ReadCNTV, scif_ReadCNTS.....	30
3.14. scif_ReadF.....	31

3.15.	scif_ReadRString.....	32
3.16.	scif_cmd_WriteRString.....	33
3.17.	scif_cmd_WriteI, scif_cmd_WriteO, scif_cmd_WriteC, scif_cmd_WriteS, scif_cmd_WriteA, scif_cmd_WriteTMR, scif_cmd_WriteCNT.....	34
3.18.	scif_cmd_WriteR, scif_cmd_WriteTMRV, scif_cmd_WriteTMRS, scif_cmd_WriteCNTV, scif_cmd_WriteCNTS.....	35
3.19.	scif_cmd_WriteF.....	36
3.20.	scif_cmd_WriteMultiI, scif_cmd_WriteMultiO, scif_cmd_WriteMultiC, scif_cmd_WriteMultiS, scif_cmd_WriteMultiA, scif_cmd_WriteMultiTMR, scif_cmd_WriteMultiCNT.....	37
3.21.	scif_cmd_WriteMultiR, scif_cmd_WriteMultiTMRV, scif_cmd_WriteM ultiCNTV, scif_cmd_WriteMultiCNTS, scif_cmd_WriteMultiTMRS.....	38
3.22.	scif_cmd_WriteMultiF.....	39
3.23.	scif_cb_WriteI, scif_cb_WriteO, scif_cb_WriteC, scif_cb_WriteS, scif_cb_WriteA, scif_cb_WriteTMR, scif_cb_WriteCNT.....	40
3.24.	scif_cb_WriteR, scif_cb_WriteTMRV, scif_cb_WriteTMRS, scif_cb_WriteCNTV, scif_cb_WriteCNTS.....	41
3.25.	scif_cb_WriteF.....	42
3.26.	scif_cmd_WriteRBit.....	43
3.27.	scif_cb_WriteRBit.....	44
3.28.	scif_StartCombineSet.....	45
3.29.	scif_FinishCombineSet.....	46
3.30.	封包組合設定補充說明.....	47
3.31.	scif_SetMaxGroupPkg.....	48
4.	檔案傳輸.....	49
4.1.	scif_FtpSetTalk.....	50
4.2.	scif_FtpUploadFile.....	51
4.3.	scif_FtpDownloadFile.....	52
4.4.	scif_FtpDeleteFile.....	53
4.5.	scif_FtpUploadFiles.....	54
4.6.	scif_FtpDownloadFiles.....	55
4.7.	scif_FtpDeleteFiles.....	56
4.8.	scif_FtpCheckDone.....	57
4.9.	scif_FtpMakeDir.....	58
4.10.	scif_FtpGetFileList.....	59
4.11.	scif_FtpReadFileCount.....	60
4.12.	scif_FtpReadFile.....	61
4.13.	scif_FileGetFileList.....	62

4.14. scif_FileReadFileCount.....	63
4.15. scif_FileReadFile.....	64
4.16. scif_FileDeleteFile.....	65
5. 函式庫內部資訊.....	66
5.1. scif_GetCommonMsg.....	67
5.2. scif_GetTalkMsg.....	68
5.3. scif_GetTalkError.....	69
附錄 A scif_define.h 內容.....	70

1. 史卡拉機械手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：

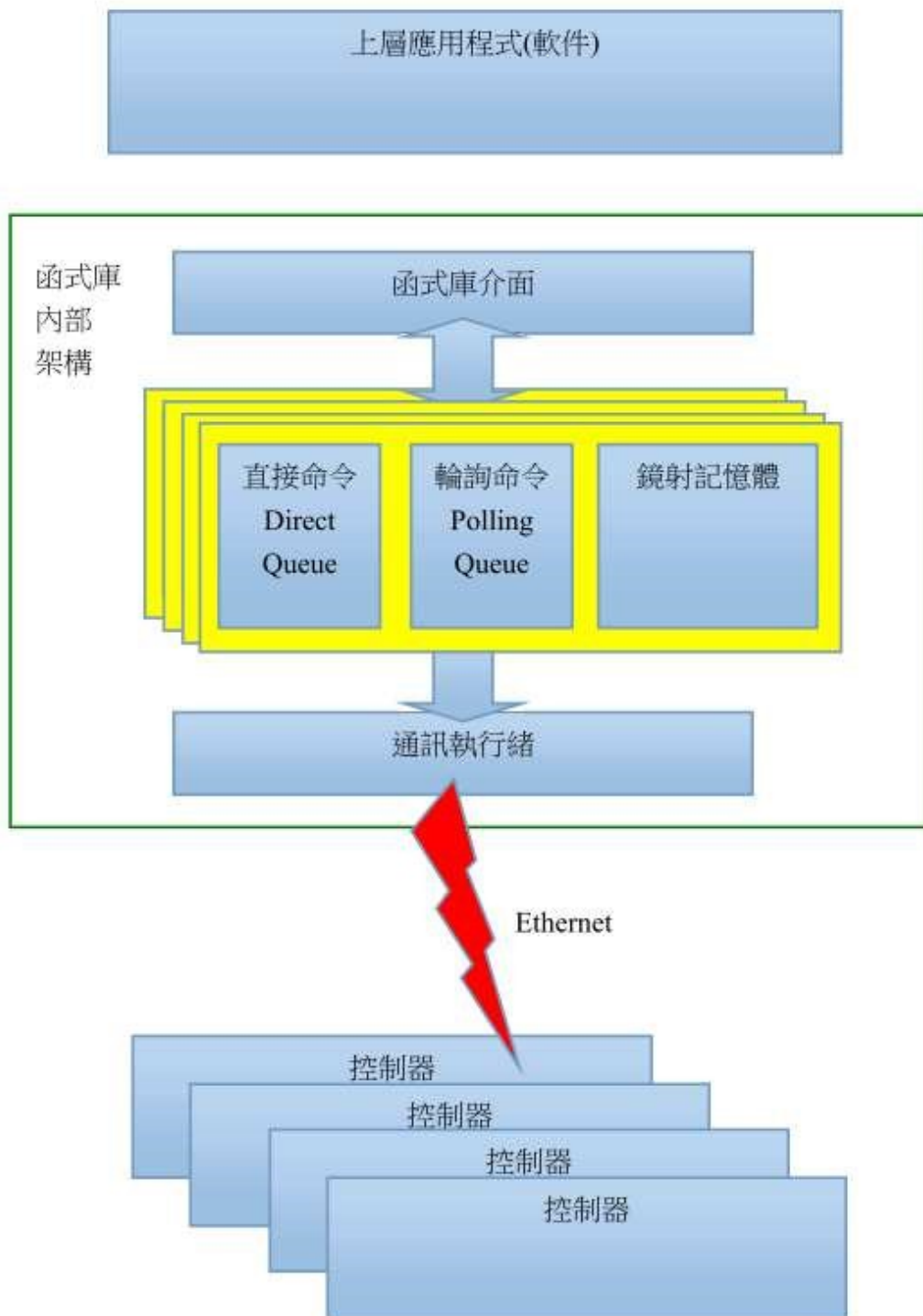
直接命令(Direct)：此類型的命令，會被存放到「直接命令佇列 Direct Queue」中，當函式庫成功執行該佇列中的命令後，就會將該命令自佇列中移除，也就是該命令只會被執行一次。

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。

1.1. 函式庫架構

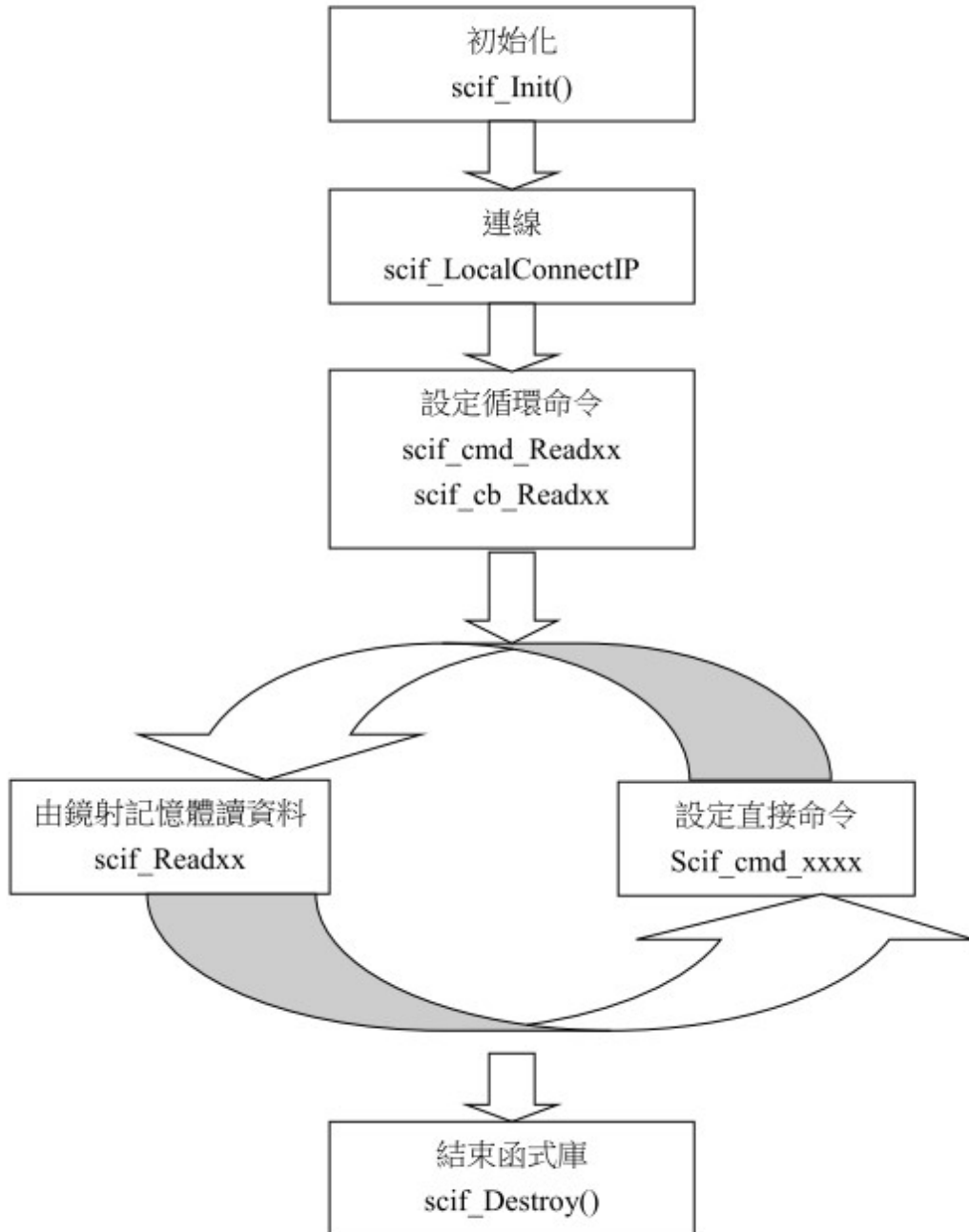


1.2. 函式庫初始化說明

為了管控制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。初始化成後，函式庫才會建立通訊的執行緒。

1.3. 一般性的程式流程



2. 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。

2.1. 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 DIISetting;  
DIISetting.SoftwareType = 3;    //軟體種類  
DIISetting.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。

```

DIISetting.MemSizeI = 4096;
DIISetting.MemSizeO = 4096;
DIISetting.MemSizeC = 4096;
DIISetting.MemSizeS = 4096;
DIISetting.MemSizeA = 4096;
DIISetting.MemSizeTT = 256;
DIISetting.MemSizeCT = 256;
DIISetting.MemSizeR = 10000; //控制器端預設大小為 6000000
DIISetting.MemSizeTS = 256;
DIISetting.MemSizeTV = 256;
DIISetting.MemSizeCS = 256;
DIISetting.MemSizeCV = 256;
DIISetting.MemSizeF = 100000;
  
```

```

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

2.2. 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();
```

2.3. scif_LocalDetectControllers

此函式會自動偵測區域網路內有多 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。

範例

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


2.4. 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();
```

2.5. 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);
```

2.6. 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);
```

2.7. 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);
```

2.8. 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);
```

2.9. 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);
```

2.10. 連線狀態補充

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。

3. 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多，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。

3.1. 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
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。

3.2. 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_Readl(SC_POLLING_CMD, serverindex, addr, num);
```

3.3. 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);
```

3.4. 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);
```


3.5. 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);
```

3.6. scif_cb_ReadR, scif_cb_ReadTMRV, scif_cb_ReadTMR, 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);
```

3.7. 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);
```

3.8. 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);
```

3.9. 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);
```

3.10. 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。

3.11. scif_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);
```

3.12. 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 );
```

3.13. 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 );
```

3.14. 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 );
```

3.15. 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);
```


3.16. 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);
```

3.17.scif_cmd_WriteI,scif_cmd_WriteO,scif_cmd_WriteC, scif_cmd_WriteS, scif_cmd_WriteA, scif_cmd_WriteTMR, scif_cmd_WriteCNT

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_Writel(serverindex, addr, val);
```

3.18. scif_cmd_WriteR, scif_cmd_WriteTMRV, scif_cmd_WriteTMRs, scif_cmd_WriteCNTV, scif_cmd_WriteCNTS

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_Writel(serverindex, addr, val);
```

3.19. 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_WriteF(serverindex, addr, val);
```

3.20. scif_cmd_WriteMultil, scif_cmd_WriteMultiO, scif_cmd_WriteMultiC, scif_cmd_WriteMultiS, scif_cmd_WriteMultiA, scif_cmd_WriteMultiTMR, scif_cmd_WriteMultiCNT

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同。以函式 `scif_cmd_WriteMultil` 為例來說明。

語法

```
int scif_cmd_WriteMultil(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_WriteMultil(serverindex, addr, num, data);
```


3.21. scif_cmd_WriteMultiR, scif_cmd_WriteMultiTMRV, scif_cmd_WriteMultiCNTV, scif_cmd_WriteMultiCNTS, scif_cmd_WriteMultiTMRS

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同。以函 `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_WriteMultil(serverindex,  
addr, num, data);
```

3.22. 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);
```

3.23. scif_cb_Writel, scif_cb_WriteO, scif_cb_WriteC, scif_cb_WriteS, scif_cb_WriteA, scif_cb_WriteTMR, scif_cb_WriteCNT

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同。以函式 `scif_cb_Writel` 為例來說明。

語法

```
int scif_cb_Writel(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_Write1(serverindex, num, addr,  
data);
```

3.24.scif_cb_WriteR,scif_cb_WriteTMRV,scif_cb_WriteT MRS, scif_cb_WriteCNTV, scif_cb_WriteCNTS

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);
```


3.25. 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);
```

3.26. 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);
```

3.27. 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);
```

3.28. 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。3.30 節有對封包組合作進一步的補充說明。

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);
```

3.29. 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。2.30 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);
```


3.30. 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 byte。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

scif_define.h 檔中定義。

3.31. scif_SetMaxGroupPkg

此函式為設定合併封包中包含命令封包的最大數量。當所要同步的命令封包數量很多時，將所有命令封包全部輪詢一遍的時間也會相對的較長，PC 端的資料更新也會變慢。此項設定能讓資料更新速度變快，但是若網路品質不佳，可能較容易發生資料遺失而需重送的情況。除非資料量真的快大，反應速度無法接受，或是一對一連線，才建議進行設定，以免佔據過多網路頻寬。

語法

```
int scif_SetMaxGroupPkg(char ServerIdx, unsigned char Count);
```

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。

Count:

最大組合封包數量。0、1 代表不執行合併動作，2 以上代表合併封包中包含命令封包的最數量，目前若設入大於 10 的數字，將會自動被記錄為 10。

回傳值

0: 設定失敗。

1: 設定成功。

範例

```
char serverindex, count;  
int Status;  
Status = scif_SetMaxGroupPkg(serverindex, count);
```

4. 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。

4.1. 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);
```

4.2. 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);
```

4.3. 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);
```

4.4. 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);
```


4.5. 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_TRANSFER_FILE TransferFiles[8];
```

```
int Status;
```

```
Status = scif_FtpUploadFiles(count, TransferFiles);
```

4.6. 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_TRANSFER_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_TRANSFER_FILE  
{  
    char Folder;  
    char Filename[FILENAME_LENGTH]; //檔案名稱的最大字元數  
    char LocalFilename[256];  
}FTP_TRANSFER_FILE;
```

回傳值

0: 設定失敗。

1: 設定成功。

範例

```
unsigned char count;
```

```
FTP_TRANSFER_FILE TransferFiles[8];
```

```
int Status;
```

```
Status = scif_FtpDownloadFiles(count, TransferFiles);
```

4.7. 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  
{  
    charFolder;  
    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_TRANSFER_FILE TransferFiles[8];  
int Status;  
Status = scif_FtpDeleteFiles(count, TransferFiles);
```

4.8. 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);
```

4.9. 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);
```


4.10. 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);
```

4.11. 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();
```

4.12. 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);
```

4.13. 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);
```

4.14. 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();
```

4.15. 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);

4.16. 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);
```

5. 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。

5.1. 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	//process 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);
```

5.2. 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);
```

5.3. 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 //process 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
//案失敗
#define FTP_RESULT_FAIL_TO_READ_REMOTE_FILE 13 //讀取遠端檔案
//案失敗
#define FTP_RESULT_FAIL_TO_WRITE_REMOTE_FILE 14 //寫入遠端檔案
//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  CtrlID;
    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_TRANSFER_FILE
{
    char Folder;
    char Filename[FILENAME_LENGTH];
    char LocalFilename[256];
}FTP_TRANSFER_FILE;
```

//功能設定的結構

```
typedef struct tag_FUNCTION_SETTING
{
    unsigned int    MakerID;
    unsigned int    MakerPwd;
    unsigned int    MediaPLong1;
    unsigned int    MediaPLong2;
    unsigned char   HaveMediaFunction;    //有使用媒合主機功能
}FUNCTION_SETTING;
```

```
typedef struct tag_DLL_USE_SETTING
{
    unsigned short  TalkInfoNum;    //連線數目
    unsigned char   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;
```


//-----控制器的功能設定

typedef struct tag_FUNC_SETTING

{

unsigned int MediaServerIP1; //媒介主機 IP 1

unsigned int MediaServerIP2; //媒介主機 IP 2

unsigned char LocalDetectEnable; //是否回應區網的偵測

unsigned char MediaPrivateEnable; //向媒介主機取得 Private ID 功能

unsigned char MediaReportEnable; //向媒介主機通報上線

unsigned char Reserve1;

unsigned char SoftwareEnable[MAX_SOFTWARE_COUNT];

//軟體功能//bit0(啟用/停用)

//bit1(是否允許外網連入)

//bit2(是否無條件接受連線)

unsigned char Reserve2[3];

}FUNC_SETTING;

史卡拉機器手臂控制器連線函式庫使用觀念與說明

出版日期：2018年07月第一版印行

-
1. HIWIN為上銀科技的註冊商標，請勿購買來路不明之仿冒品以維護您的權益。
 2. 本型錄所載規格、照片有時會與實際產品有所差異，包括因為改良而導致外觀或規格等發生變化的情況。
 3. 凡受”貿易法”等法規限制之相關技術與產品，HIWIN將不會違規擅自出售。若要出口HIWIN受法律規範限制出口的產品，應根據相關法律向主管機關申請出口許可，並不得供作生產或發展核子、生化、飛彈等軍事武器之用。
 4. HIWIN產品專利清單查詢網址：http://www.hiwin.tw/Products/Products_patents.aspx



海外廠 / 研發中心

德國 歐芬堡

HIWIN GmbH
OFFENBURG, GERMANY
www.hiwin.de
www.hiwin.eu

日本 神戶 · 東京 · 名古屋 · 長野 · 東北 · 靜岡 · 北陸 · 廣島 · 福岡 · 熊本

HIWIN JAPAN
KOBE · TOKYO · NAGOYA · NAGANO ·
TOHOKU · SHIZUOKA · HOKURIKU ·
HIROSHIMA · FUKUOKA · KUMAMOTO, JAPAN
www.hiwin.co.jp

美國 芝加哥 · 矽谷

HIWIN USA
CHICAGO · SILICON VALLEY, U.S.A.
www.hiwin.com

義大利 米蘭

HIWIN Srl
BRUGHERIO, ITALY
www.hiwin.it

瑞士 優納

HIWIN Schweiz GmbH
JONA, SWITZERLAND
www.hiwin.ch

捷克 布爾諾

HIWIN s.r.o.
BRNO, CZECH REPUBLIC
www.hiwin.cz

新加坡

HIWIN SINGAPORE
SINGAPORE
www.hiwin.sg

韓國 水原 · 馬山

HIWIN KOREA
SUWON · MASAN, KOREA
www.hiwin.kr

中國 蘇州

HIWIN CHINA
SUZHOU, CHINA
www.hiwin.cn

以色列 海法

Mega-Fabs Motion Systems, Ltd.
HAIFA, ISRAEL
www.mega-fabs.com

上銀科技股份有限公司

HIWIN TECHNOLOGIES CORP.

40852 台中市精密機械園區精科路7號

Tel: 04-23594510

Fax: 04-23594420

www.hiwin.tw

business@hiwin.tw