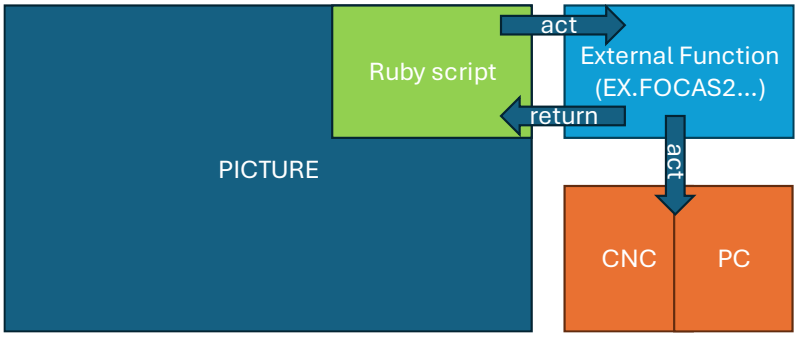
PICTURE 外部函式呼叫功能（PANEL iH / iH PRO）

目錄

1. 概要

此功能可透過在 FANUC PICTURE 專案的 FPScript 資料夾中建立外部函式定義檔（funcdef.txt）後，從 Ruby 腳本中執行 Windows API 函式或外部的動態連結程式庫（DLL）內的函式。

此說明文件內將會以呼叫 WindowsAPI 與 FOCAS2 Library 來進行範例說明，關於 WindowsAPI 與 FOCAS2 Library 詳細內容請參考各自說明文件。

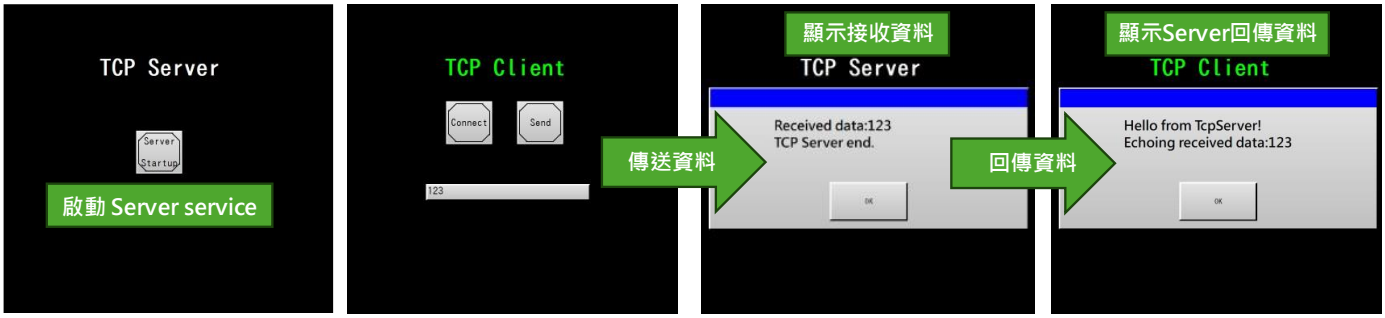


註.

由於 PANEL iH Pro 和 PANEL iH 搭載的作業系統不同，Windows API 的規格也有所差異，因此請依據所使用的環境使用各自的 Windows API。

應用範例. 藉由 FOCAS2 函式庫寫入刀具補正

應用範例. 藉由 WinCE Winsock 函式庫進行 TCP/IP 傳輸



1.1 使用環境及開發環境

使用環境

· PANEL iH、PANEL iH Pro、FANUC iPC (CNC GUIDE)

開發環境

· PANEL iH Pro 或是 FANUC iPC : Visual Studio 2008 以上

PICTURE 11.1 版以上

· PANEL iH : Visual Studio 2008 Professional(需安裝 iHMI FANUC SDK)

PICTURE 11.3 版以上

PANEL iH 開發工具的安裝與環境設定

①. 安裝 Visual Studio 2008。

②. 安裝 Microsoft Visual Studio 2008 Service Pack 1。

③. 安裝 Visual Studio 2008 用於 Windows Embedded Compact 7 的更新程式。

④. 安裝 Windows Embedded Compact 7 ATL Update for Visual Studio 2008 SP1。

⑤. 執行 iHMI FANUC SDK 中的 \FANUC_SDK\FANUC700B.msi，安裝適用於 Windows Embedded Compact 7 的 FANUC SDK。

(iHMI Application SDK for PANEL iH：A08B-9110-J713)

2. 範例專案

本章將介紹兩個實作範例，說明如何在實際應用中使用函式庫進行開發。第一個範例將說明使用 FOCAS2 函式庫進行刀具補正資料的寫入操作；第二個範例則是透過 WinCE 的 WINSOCK 函式庫實現 TCP/IP 通訊。

PICTURE & C++範例專案資料夾於 Project_Sample.zip

- Project_Sample
  - FOCAS2(cnc_wrtofs)
    - Dll_ WR_Tooloffset ... WR_Tooloffset C++動態連結檔專案
    - PICTURE_ WR_Tooloffset ... WR_Tooloffset PICTURE 專案
  - Winsock(TCP_IP)
    - Dll_TCP_Client ... TCP_Client C++動態連結檔專案
    - Dll_TCP_Server ... TCP_Server C++動態連結檔專案
    - PICTURE_TCP_Client ... TCP_Client PICTURE 專案
    - PICTURE_TCP_Server ... TCP_Server PICTURE 專案

2.1 使用 FOCAS2 函式庫寫入刀具補正

2.1.1 製作 FOCAS2 動態連結檔

- 宣告引用函式庫[C++]

Panel iH

```
#include "./PANELi/Fwlib32.h"
```

CNC GUIDE, Panel iH PRO

```
#include "./NewPlatform/Fwlib32.h"
```

- 宣告 FOCAS2 寫入刀補函式 (關於 cnc_wrtofs，請參考 FOCAS2 函式庫說明文件)[C++]

```
extern "C" {
__declspec(dllexport) short Call_Focas2_wrtol(int number, int type, int data)
{
    short ret = cnc_wrtofs( s_handle_ex1, number, type, 8, data ) ;
    return ret;   // 回傳
}
}// End of extern "C"
```

註.

編譯後產生的 dll 檔案需與 funcdef.txt 內 LibName 連結，LibName 指定檔案路徑，或由 LibName 指定 dll 檔案路徑，且 dll 檔案名稱命名規則為 { 專案名稱_fpusrfunc.dll }例. TCP_Tooloffset_fpusrfunc.dll。

2.1.2 PICTURE 宣告呼叫製作完成的 FOCAS2 外部函式

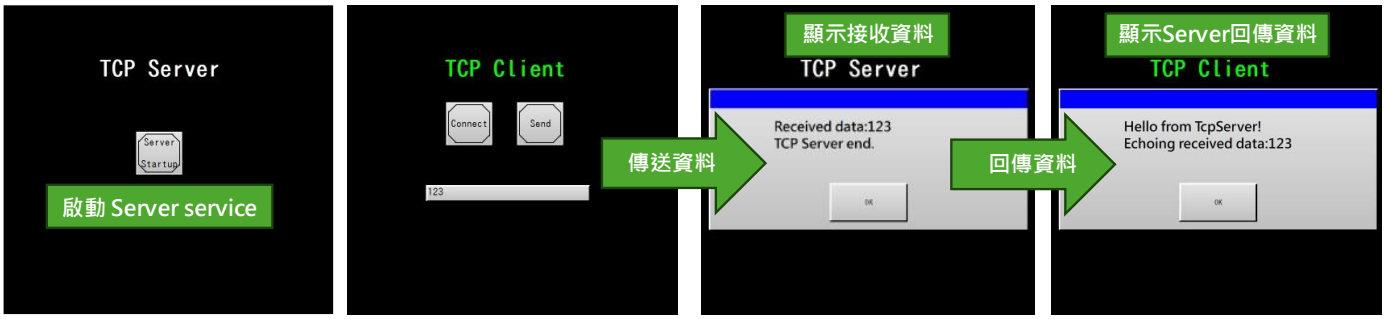
- 宣告 funcdef.txt (詳細請參考附錄 A.)

```
//////////////Call_Focas2_wrtofs//////////////////////
function short Call_Focas2_wrtol library "D:\\FANUC PICTURE\\WR_Tooloffset \\Driver\\NCGuide\\WR_Tooloffset_fpusrfunc.dll" (
in int number,
        in int type,
        in int data
);
```

- Script[ruby]

```
Begin
        # 根據 ARG[0] 的值選擇要執行的動作
        case ARG[0]
        when 1
        number = 200 # 寫入的刀補號碼
        type = 2 # 寫入的刀補類型
        data = 123456 # 寫入的刀補資料
        ret = Call_Focas2_wrtol(number, type, data) # 呼叫 Call_Focas2_wrtol 函式寫入刀補
        end
rescue=>ex
        indent="\n"
        str="No.1"+ex.message+indent+ex.backtrace[0]
        MsgBoxShow(str,0)
end
```

2.2 使用 WinCE WINSOCK 函式庫進行 TCP/IP 傳輸



2.2.1 製作 TCP_Server 用 WinCE WINSOCK 動態連結檔

- 宣告引用函式庫 [C++]

Panel iH

```
//Panel iH
#include <winsock.h>
#pragma comment(lib, "winsock.lib")   // 或改為 ws2.lib 視 SDK 而定
```

CNC GUIDE, Panel iH PRO

```
//CNC GUIDE, iH PRO
#include <winsock2.h>   // 修改為 winsock2.h
#include <ws2tcpip.h>
#pragma comment(lib, "ws2_32.lib")   // 使用 ws2_32.lib
```

- 宣告 TCP/IP Server 函式 (以下為範例，詳細請自行參考 TCP/IP 函式庫)[C++]

```
extern "C" {
__declspec(dllexport) int TcpStartup()
{
	WSADATA wsaData;
	return WSAStartup(MAKEWORD(1, 1), &wsaData);   // WinCE 通常使用 Winsock 1.1
}
__declspec(dllexport) int TcpCleanup()
{
	return WSACleanup();
}
__declspec(dllexport) int TcpSend(int sock, const char* data, int len)
{
	return send((SOCKET)sock, data, len, 0);
}
__declspec(dllexport) int TcpRecv(int sock, char* buffer, int len)
{
	return recv((SOCKET)sock, buffer, len, 0);
}
__declspec(dllexport) int TcpClose(int sock)
{
	return closesocket((SOCKET)sock);
}
__declspec(dllexport) int TcpListen()
{
	int port = 12345;
	SOCKET listenSock = socket(AF_INET, SOCK_STREAM, 0);
	if (listenSock == INVALID_SOCKET)
		return -1;
	struct sockaddr_in addr;
	addr.sin_family = AF_INET;
	addr.sin_port = htons(port);
	addr.sin_addr.s_addr = htonl(INADDR_ANY);   // 監聽所有本機 IP

	if (bind(listenSock, (struct sockaddr*)&addr, sizeof(addr)) == SOCKET_ERROR) {
		closesocket(listenSock);
		return -2;
	}
	if (listen(listenSock, 1) == SOCKET_ERROR) {
		closesocket(listenSock);
		return -3;
	}
	return (int)listenSock;   // 回傳 listen socket
}
__declspec(dllexport) int TcpAccept(int listenSock)
{
	SOCKET clientSock = accept((SOCKET)listenSock, NULL, NULL);
	if (clientSock == INVALID_SOCKET)
		return -1;
	return (int)clientSock;   // 回傳 client socket
}
}// End of extern "C"
```

註.

編譯後產生的 dll 檔案需與 funcdef.txt 內 LibName 連結，LibName 指定檔案路徑，或由 LibName 指定 dll 檔案路徑，且 dll 檔案名稱命名規則為 { 專案名稱_fpusrfunc.dll }，例. TCP_Server_fpusrfunc.dll。

2.2.2 PICTURE 宣告呼叫 TCP_Server 用 WinCE WINSOCK 外部函式

- 宣告 funcdef.txt (詳細請參考附錄 A.)

```
//////////////TCP_Server//////////////////////
function int TcpStartup library "D:\\FANUC PICTURE\\TCP_Server\\Driver\\NCGuide\\TCP_Server_fpusrfunc.dll" ();
function int TcpCleanup library "D:\\FANUC PICTURE\\TCP_Server\\Driver\\NCGuide\\TCP_Server_fpusrfunc.dll" ();
function int TcpClose library "D:\\FANUC PICTURE\\TCP_Server\\Driver\\NCGuide\\TCP_Server_fpusrfunc.dll" (in int socket);
function int TcpRecv library "D:\\FANUC PICTURE\\TCP_Server\\Driver\\NCGuide\\TCP_Server_fpusrfunc.dll" (
        in int socket,
        inout string buffer,   // 指定為 inout
        in int bufferSize
);
function int TcpSend library "D:\\FANUC PICTURE\\TCP_Server\\Driver\\NCGuide\\TCP_Server_fpusrfunc.dll" (
        in int socket,
        in string data,
        in int length
);
function int TcpListen library "D:\\FANUC PICTURE\\TCP_Server\\Driver\\NCGuide\\TCP_Server_fpusrfunc.dll" ();
function int TcpAccept library "D:\\FANUC PICTURE\\TCP_Server\\Driver\\NCGuide\\TCP_Server_fpusrfunc.dll" (in int listenSocket);
```

- Script[ruby]

```
begin
        case ARG[0]   # 根據啟動參數 ARG[0] 判斷要執行哪個功能
        when 1 # TCP_Server
                Startup = 1   # 啟用 server 的控制變數
                buffer_size = 1024   # 設定接收資料的緩衝區大小
                Serverbuffer = "\0" * buffer_size   # 初始化接收緩衝區為全空字元
                TcpStartup()   # 初始化 TCP 系統
                listenSock = TcpListen()   # 啟動 TCP 監聽 ( 建立 server socket 並綁定/監聽 )
                if listenSock < 0
                        MsgBoxShow(listenSock.to_s, 0)   # 若失敗則顯示錯誤訊息
                end
                clientSock = TcpAccept(listenSock)   # 等待並接受 client 的連線
                if clientSock < 0
                        MsgBoxShow(clientSock.to_s, 0)   # 若接受失敗顯示錯誤
                end
                while Startup == 1   # 進入主接收迴圈 ( 目前設計只跑一次 )
                        recv_len = TcpRecv(clientSock, Serverbuffer, buffer_size - 1)
                        # 從 client 接收資料存入 Serverbuffer，最多接收 buffer_size - 1 個字元
                        if recv_len > 0   # 若成功收到資料
                                msg = "Hello from TcpServer!" + "\n" + "Echoing received data:" + Serverbuffer.to_s
                                # 建立要回傳的訊息：包含歡迎詞與接收到的資料
                                TcpSend(clientSock, msg, msg.length)   # 回傳該訊息給 client ( echo 回傳 )
                                Startup = 0   # 結束接收迴圈 ( 目前只跑一次 )
                        end
                end
                TcpClose(clientSock)   # 關閉 client socket
                TcpClose(listenSock)   # 關閉 server 監聽 socket
                TcpCleanup()   # 清理 TCP 系統
                MsgBoxShow("Received data:" + Serverbuffer.to_s + "\n" + "TCP Server end.", 0)
                # 顯示結束訊息與接收到的資料
        end
rescue => ex   # 異常處理：顯示錯誤訊息與錯誤位置
        indent = "\n"
        str = "No.1" + ex.message + indent + ex.backtrace[0]
        MsgBoxShow(str, 0)
end
```

2.2.3 製作 TCP_Client 用 WinCE WINSOCK 動態連結檔

- 宣告引用函式庫 [C++]

Panel iH

```
//Panel iH
#include <winsock.h>
#pragma comment(lib, "winsock.lib")   // 或改為 ws2.lib 視 SDK 而定
```

CNC GUIDE, Panel iH PRO

```
//CNC GUIDE, iH PRO
#include <winsock2.h>   // 修改為 winsock2.h
#include <ws2tcpip.h>
#pragma comment(lib, "ws2_32.lib")   // 使用 ws2_32.lib
```

- 宣告 TCP/IP Client 函式 (以下為範例，詳細請自行參考 TCP/IP 函式庫)

```
extern "C" {
__declspec(dllexport) int TcpStartup()
{
	WSADATA wsaData;
	return WSAStartup(MAKEWORD(1, 1), &wsaData);   // WinCE 通常使用 Winsock 1.1
}

__declspec(dllexport) int TcpConnect()
{
	const char* ip = "127.0.0.1";
	int port = 12345;
	SOCKET sock = socket(AF_INET, SOCK_STREAM, 0);
	if (sock == INVALID_SOCKET)
		return -1;

	struct sockaddr_in addr;
	addr.sin_family = AF_INET;
	addr.sin_port = htons(port);
	addr.sin_addr.s_addr = inet_addr(ip);

	if (connect(sock, (struct sockaddr*)&addr, sizeof(addr)) == SOCKET_ERROR) {
		closesocket(sock);
		return -2;
	}
	return (int)sock;   // 成功回傳 socket 整數值
}

__declspec(dllexport) int TcpSend(int sock, const char* data, int len)
{
	return send((SOCKET)sock, data, len, 0);
}

__declspec(dllexport) int TcpRecv(int sock, char* buffer, int len)
{
	return recv((SOCKET)sock, buffer, len, 0);
}
}// End of extern "C"
```

註.

編譯後產生的 dll 檔案需與 funcdef.txt 內 LibName 連結，LibName 指定檔案路徑，或由 LibName 指定 dll 檔案路徑，且 dll 檔案名稱命名規則為 { 專案名稱_fpusrfunc.dll }，例. TCP_ Client_fpusrfunc.dll。

2.2.4 PICTURE 宣告呼叫 TCP_ Client 用 WinCE WINSOCK 外部函式

- 宣告 funcdef.txt (詳細請參考附錄 A.)

```
//////////////TCP_Client//////////////////////
function int TcpStartup library "D:\\FANUC PICTURE\\TCP_Client\\Driver\\NCGuide\\TCP_Client_fpusrfunc.dll" ();
function int TcpConnect library "D:\\FANUC PICTURE\\TCP_Client\\Driver\\NCGuide\\TCP_Client_fpusrfunc.dll" ();
function int TcpSend library "D:\\FANUC PICTURE\\TCP_Client\\Driver\\NCGuide\\TCP_Client_fpusrfunc.dll" (
        in int socket,
        in string data,
        in int length
);
function int TcpRecv library "D:\\FANUC PICTURE\\TCP_Client\\Driver\\NCGuide\\TCP_Client_fpusrfunc.dll" (
        in int socket,
        inout string buffer,   // 指定為 inout
        in int bufferSize
);
```

- Script

```
begin
        # 根據 ARG[0] 的值選擇要執行的動作
        case ARG[0]
        when 1 # TCP_TcpConnect
                TcpStartup()   # 初始化 TCP/IP 函式庫
                $P_sock = TcpConnect()   # 建立 TCP 連線並取得 socket 存到專案變數$P_sock
                if $P_sock < 0 # 取得 socket 錯誤時
                        MsgBoxShow($P_sock.to_s, 0)   # 顯示錯誤代碼
                end
                MsgBoxShow("Connected.", 0)   # 顯示連線成功訊息
        when 2 # TCP_Send
                buffer_size = 1024   # 定義接收資料的緩衝區大小
                Tcpbuffer = "\0" * buffer_size   # 建立空的接收緩衝區
                keyin_data = rdstr(400, 0, 0, 10, 99, 0)   # 從系統變數讀取輸入資料 ( 例：畫面鍵入 )
                # 傳送資料
                TcpSend($P_sock, keyin_data.to_s, keyin_data.to_s.length)
                # 接收回傳資料
                recv_len = TcpRecv($P_sock, Tcpbuffer, buffer_size - 1)
                if recv_len > 0 # 接收到回傳資料時
                        TcpbufferSave = Tcpbuffer   # 若有資料則儲存接收到的內容
                end
                MsgBoxShow(TcpbufferSave.to_s, 0)   # 顯示接收到的資料
        end
rescue => ex
        # 異常處理：顯示錯誤訊息與錯誤位置
        indent = "\n"
        str = "No.1" + ex.message + indent + ex.backtrace[0]
        MsgBoxShow(str, 0)
end
```

3. 參考資料

1. FANUC PICTURE Graphic drawing library Specification A-42148-00830EN/02

2. FANUC PICTURE Specification (Edition 06.2 to less than 08.0) A-40712EN

3. FANUC Drivers and Libraries (FOCAS1/2) 說明文件（FOCAS2 安裝光碟內 Document 資料夾）

附錄 A：外部函式定義規則

請於 FANUC PICTURE 專案內 FPScript 資料夾生成外部函式定義用文字檔案命名為 funcdef.txt，並依照下列規則定義函式。

function Sign Type CallSeq FuncName entry EntryName library "LibName" (IODir Param, IODir Param, ...);

Sign（optional）

符号

Type 是整数型的場合、指定 signed 或是 unsigned

Type

返回值的型別

可指定的資料型別有：整數、浮點數、字串（string 或 wstring 型）、void、以及 C 語言的指標（void*）。

字串僅限用於返回值為 NULL 結尾字串指標的函式。

無法指定陣列、結構體，以及 void* 以外的指標。

CallSeq（optional）

呼叫規則

指定 cdecl 或是 stdcall

省略時預設為 stdcall

FuncName

函式名（最大 64 字元）

EntryName（optional、使用 entry 指定詞時可以省略）

DLL 中實際的函式名稱。

如果省略，將使用 FuncName 作為 DLL 中的函式名稱。

LibName

DLL 檔案名稱

副檔名 ".dll" 可省略。

DLL 的搜尋規則遵循 Windows API 的 LoadLibrary 規範。

建議使用絕對路徑。

IODir（optional）

引數的輸入輸出方向。

作為函式輸入時指定為 in，作為輸出時指定為 out，若同時為輸入與輸出則指定為 inout。

若省略，預設為 in。

Param

引數

若為陣列、結構體，或輸入輸出方向為 out 或 inout 時，必須以參照方式（指標傳遞）指定。