如何使用 Win32 API 存取 RS232

作者 : 鄭旭崇 http://www.finetech.idv.tw/techdoc/bcbrs232/bcbrs232.htm

§前言

以往我們在 DOS 作業系統的真實模式下,想要存取 Serial port 可說是件輕而易舉的事,我們可以透過 BIOS 提供的中斷服務常式(ISR)對 Serial port 進行存取,或者直接經由 I/O Port 規化 UART 晶片,達到串列通訊的目的.而今在 Windows 作業系統的保護模式下,欲直接存取 I/O Port 卻不是那麼簡單,而且如果要從低階 I/O 來控制 Serial port 的話,又會涉及到硬體中斷(interrupt)的問題,我們都知道在 Windows 下處理硬體中斷也是相當麻煩的.幸好 Windows API 提供了一套通訊函式(Communication Functions) 可以專門用來解決串列通訊的問題.關於 Communication Functions 總共有二十幾個, 但實際上如果只想做單純的傳送及接收的話,只須用到五個 ommunication Functions.
也就是說,讀者只要學會使用這幾個 Communication Functions 那麼關於串列通訊的問題便可迎刃而解了.

§Win32 API Communication Functions 說明

以下茲對相關的 Win32 API Communication Functions 作簡短的介紹.

1. BuildCommDCB

函數原型: BOOL BuildCommDCB(LPCTSTR lpDef, LPDCB lpDCB). BuildCommDCB 函數是用來填寫 DCB 的資料,何謂 DCB 呢? DCB 的全名為 Device Control Block 是一資料結構,裡面定義著所有有關串列通訊(Serial Communication)的設定值,DCB 資料結構定義如下:

```
// winbase.h
// https://docs.microsoft.com/en-us/windows/win32/api/winbase/ns-winbase-dcb
#define NOPARITY       0
#define ODDPARITY      1
#define EVENPARITY     2
#define MARKPARITY     3
#define SPACEPARITY    4

#define ONESTOPBIT         0
#define ONE5STOPBITS       1
#define TWOSTOPBITS        2

#define CBR_110           110
```

```
#define CBR_300            300
#define CBR_600            600
#define CBR_1200           1200
#define CBR_2400           2400
#define CBR_4800           4800
#define CBR_9600           9600
#define CBR_14400          14400
#define CBR_19200          19200
#define CBR_38400          38400
#define CBR_56000          56000
#define CBR_57600          57600
#define CBR_115200         115200
#define CBR_128000         128000
#define CBR_256000         256000

#define DTR_CONTROL_DISABLE    0
#define DTR_CONTROL_ENABLE     1
#define DTR_CONTROL_HANDSHAKE  2

#define RTS_CONTROL_DISABLE    0
#define RTS_CONTROL_ENABLE     1
#define RTS_CONTROL_HANDSHAKE  2
#define RTS_CONTROL_TOGGLE     3

typedef struct _DCB {
    DWORD DCBlength;           // DCB 的大小 (in byte)
    DWORD BaudRate;            // 傳輸率 (baud rate) bit/秒
    DWORD fBinary: 1;          // 二進制模式,沒有 EOF 檢查.
    DWORD fParity: 1;          // 同位元檢查 (parity checking)
    DWORD fOutxCtsFlow:1;      // CTS output flow control
    DWORD fOutxDsrFlow:1;      // DSR output flow control
    DWORD fDtrControl:2;       // DTR flow control type
    DWORD fDsrSensitivity:1;   // DSR sensitivity
    DWORD fTXContinueOnXoff:1; // XOFF continues Tx
    DWORD fOutX: 1;            // XON/XOFF out flow control
    DWORD fInX: 1;             // XON/XOFF in flow control
    DWORD fErrorChar: 1;       // enable error replacement
    DWORD fNull: 1;            // enable null stripping
    DWORD fRtsControl:2;       // RTS flow control
```

```
    DWORD fAbortOnError:1;     // abort reads/writes on error
    DWORD fDummy2:17;          // reserved
    WORD wReserved;            // not currently used
    WORD XonLim;               // transmit XON threshold
    WORD XoffLim;              // transmit XOFF threshold
    BYTE ByteSize;             // 每一筆資料的 bit 數, 4-8
    BYTE Parity;               // 同位元 (0-4) = 無,奇同位,偶同位,mark,space
    BYTE StopBits;             // 停止位元 (0,1,2) = 1, 1.5, 2
    char XonChar;              // Tx and Rx XON character
    char XoffChar;             // Tx and Rx XOFF character
    char ErrorChar;            // error replacement character
    char EofChar;              // end of input character
    char EvtChar;              // received event character
    WORD wReserved1;           // reserved; do not use
} DCB;
```

這個 DCB Structure 幾乎函蓋了所有有關串列通訊的參數，提供以後使用 SetCommState API 函數對硬體進行初始化設定．透過 **BuildCommDCB** 函數，我們可以最簡單的方法來向 DCB Structure 填值，其中 LPCTSTR lpDef 指向一個叫做 device-control string 的位址,而 LPDCB lpDCB 正指向 DCB Structure．我們較感興趣的是 device-control string，它使用設定串列埠的語法就跟以往 DOS 時代的 Mode 指令一樣，唯一不同的是它的傳輸率 (baud rate)已經不再限制於 19200 bit/秒以下，並且最高可達 256000 bit/秒．例如:欲設定 COM2 的 Baud = 57600，沒有同位元，資料 = 8 bit，一個停止位元，可以寫成 BuildCommDCB("19200, n, 8, 1", &dcb)．若涵數執行成功，將傳回一個非零的值．若執行失敗則傳回零．

2. CreatFile

函數原型:

```
HANDLE CreateFile(
    LPCTSTR lpFileName,                        // pointer to name of the file
    DWORD dwDesiredAccess,                     // access (read-write) mode
    DWORD dwShareMode,                         // share mode
    LPSECURITY_ATTRIBUTES lpSecurityAttributes,    // pointer to security attributes
    DWORD dwCreationDistribution,              // how to create
    DWORD dwFlagsAndAttributes,                // file attributes
    HANDLE hTemplateFile                       // handle to file with attributes to copy
);
```

這個 Windows API 函數在此是用來開啟系統的 Serial Port 並取得一個 HANDLE 代碼. 以後想要存取 Serial Port 時, 只須面對這個 HANDLE 代碼即可. 當然 CreateFile 函數也可以開啟其他類型的檔案, 例如 files, pipes, mailslots 等等, 有興趣的朋友可以參考 Borland C++ 或 C++Builder 附加的 Help file 內皆有詳盡的說明.

此函數共有七個變數欄位, 第一個變數欄位 lpFileName 指向檔案名稱或裝置名稱, 要當做串列通訊時, **lpFileName** 為 "COM1" 或 "COM2". 變數 **dwDesiredAccess** 設定存取型式. 常數 GENERIC_READ 代表可讀不可寫, 常數 GENERIC_WRITE 代表可寫不可讀, **GENERIC_READ + GENERIC_WRITE** 代表可寫亦可讀. 變數 **dwShareMode** 設定共享模式.常數 FILE_SHARE_READ 代表可以同時被多個程式讀取, 而常數 FILE_SHARE_READ 代表可以同時被多個程式寫入, NULL 代表不開放共享. 當然在 Serial Port 下 dwShareMode 是不能開放共享的.
變數 **lpSecurityAttributes** 設定保密程度.
變數 **dwCreationDistribution** 決定檔案開啟模式.
常數 **OPEN_EXISTING** 代表開啟一個已存在的舊檔.
常數 **CREATE_ALWAYS** 代表開啟新檔.
變數 **dwFlagsAndAttributes** 決定檔案屬性. 在使用 COM Port 時, 檔案屬性必須設為 FILE_ATTRIBUTE_NORMAL.
變數 **hTemplateFile** 指定檔案屬性原型.

3. SetCommState

函數原型:

```
BOOL SetCommState(
    HANDLE hFile,// handle of communications device
    LPDCB lpDCB     // address of device-control block structure
);
```

SetCommState 函數會根據 DCB 資料結構的內容, 來設定串列通訊裝置, 並對硬體重新初始化(initialize).
變數 **hFile** 存放著執行 CreatFile 函數後所傳回的 HANDLE 代碼.
變數 **lpDCB** 指向 DCB 資料結構的位址.
例如 : SetCommState(handle,&dcb)
若涵數執行成功,將傳回一個非零的值.若執行失敗則傳回零.

4. SetupComm

函數原型:

```
BOOL SetupComm(
```

```
    HANDLE hFile,    // 通訊設備的控制碼
    DWORD dwInQueue, // 輸入緩衝區的大小（位元組數）
    DWORD dwOutQueue // 輸出緩衝區的大小（位元組數）
);
```

SetupComm 函數主要是用來設定輸入資料儲列(Queue)與輸出資料儲列大小.
變數 **hFile** 一樣也是存放著執行 CreatFile 函數後所傳回的 HANDLE 代碼.
變數 **dwInQueue** 設定輸入資料儲列的大小單位為 Byte.
變數 **dwOutQueue** 設定輸出資料儲列的大小單位為 Byte.
若涵數執行成功,將傳回一個非零的值.若執行失敗則傳回零.

5. SetCommTimeouts

函數原型:

```
BOOL SetCommTimeouts(
    HANDLE hFile,        // handle of communications device
    LPCOMMTIMEOUTS lpCommTimeouts   // address of communications time-out structure
);
```

本函數執行的結果, 會影響 ReadFile 的讀取時間與 WriteFile 的寫入時間.
變數 hFile 存放著執行 CreatFile 函數後所傳回的 HANDLE 代碼.
變數 lpCommTimeouts 指向一個 communications time-out 的資料結構,

```
typedef struct _COMMTIMEOUTS {
    DWORD ReadIntervalTimeout;          // msec
    DWORD ReadTotalTimeoutMultiplier;    // msec
    DWORD ReadTotalTimeoutConstant;      // msec
    DWORD WriteTotalTimeoutMultiplier;   // msec
    DWORD WriteTotalTimeoutConstant;     // msec
} COMMTIMEOUTS, *LPCOMMTIMEOUTS;         // msec
```

變數 **ReadIntervalTimeout** 設定讀取第一個字元與第二個字元之間的 time out 時間, 單位為毫秒(msec). 當使用 ReadFile 函數從串列埠讀取一個字元時, 若在 ReadIntervalTimeout 時間內讀取第二個字元, 則 ReadFile 函數會繼續讀取下一個字元. 若未在 ReadIntervalTimeout 時間內讀取第二個字元, 則 ReadFile 函數將完成工作也就是跳出 ReadFile 執行下一行敘述. ReadIntervalTimeout 設為 0 表示關閉此功能.

變數 **ReadTotalTimeoutMultiplier** 視讀取的字元數來決定總 time out 時間.

總 time out 時間 = ReadTotalTimeoutMultiplier * 欲讀取字元數 + ReadTotalTimeoutConstant.

將 ReadTotalTimeoutMultiplier 設為 0 表示關閉此功能.

變數 **ReadTotalTimeoutConstant** 為 time out 時間常數. 加在 ReadTotalTimeoutMultiplier 之後.

變數 **WriteTotalTimeoutMultiplier** 與 **WriteTotalTimeoutConstant** 設定寫入串列埠的總 time out 時間.

寫入串列埠的總 time out 時間 =
WriteTotalTimeoutMultiplier * 欲寫入之字元數 + WriteTotalTimeoutConstant.

將 WriteTotalTimeoutMultiplier 和 WriteTotalTimeoutConstant 設為 0 表示關閉此功能.

讀取串列埠之 time out 設定的恰當與否，將會影響程式的執行效率，如果設定的總 time out 時間過長，而遠端裝置又沒有回應(No Response)時通常會導致讓 User 誤判成電腦當機，所以不可以亂設.
一般通常都設成:

```
TimeOut.ReadIntervalTimeout         = 0;
TimeOut.ReadTotalTimeoutMultiplier  = 0;
TimeOut.ReadTotalTimeoutConstant    = 500;  //(總讀取 time out 時間 = 0.5 秒)
TimeOut.WriteTotalTimeoutMultiplier = 0;
TimeOut.WriteTotalTimeoutConstant   = 500; //(總寫入 time out 時間 = 0.5 秒)
```

再執行 SetCommTimeouts(handle, &TimeOut), 即完成了 communications time-out 的設定. 若涵數執行成功，將傳回一個非零的值. 若執行失敗則傳回零.

6. PurgeComm

在讀寫序列埠之前，還要用 PurgeComm() 函數清空緩衝區，該函數原型:

```
BOOL PurgeComm(
    HANDLE hFile, //序列埠控制碼
    DWORD dwFlags ); // 需要完成的操作
```

參數 dwFlags 指定要完成的操作，可以是下列值的組合:
PURGE_TXABORT 中斷所有寫操作並立即返回，即使寫操作還沒有完成.
PURGE_RXABORT 中斷所有讀操作並立即返回，即使讀操作還沒有完成.
PURGE_TXCLEAR 清除輸出緩衝區
PURGE_RXCLEAR 清除輸入緩衝區

7. summary

最後再將上述的五個 Communication Function 簡單地複習一遍.

初始化 COM PORT 的步驟
1. BuildCommDCB : 建立 DCB (Device Control Block).
2. CreatFile : 開啟 COM Port 並取得 Handle 代碼.
3. SetCommState : 根據 DCB 資料結構的內容, 來設定串列通訊裝置, 並對硬體重新初始化.
4. SetupComm : 用來設定輸入資料儲列(Queue)與輸入資料儲列.
5. SetCommTimeouts :設定 ReadFile 的最大讀取時, 間與 WriteFile 的最大寫入時間.
此時的 Com Port 已可隨意存取, 欲存取 Com Port 請用 ReadFile() 與 WriteFile() API 函數.
從 Com Port 讀取一個字元: `ReadFile(handle, &lpBuf, 1, &dwRead, NULL);`
寫一個字元至 Com Port : `WriteFile(handle, &WriteBuffer, 1, &NoWrite,0);`

§範例

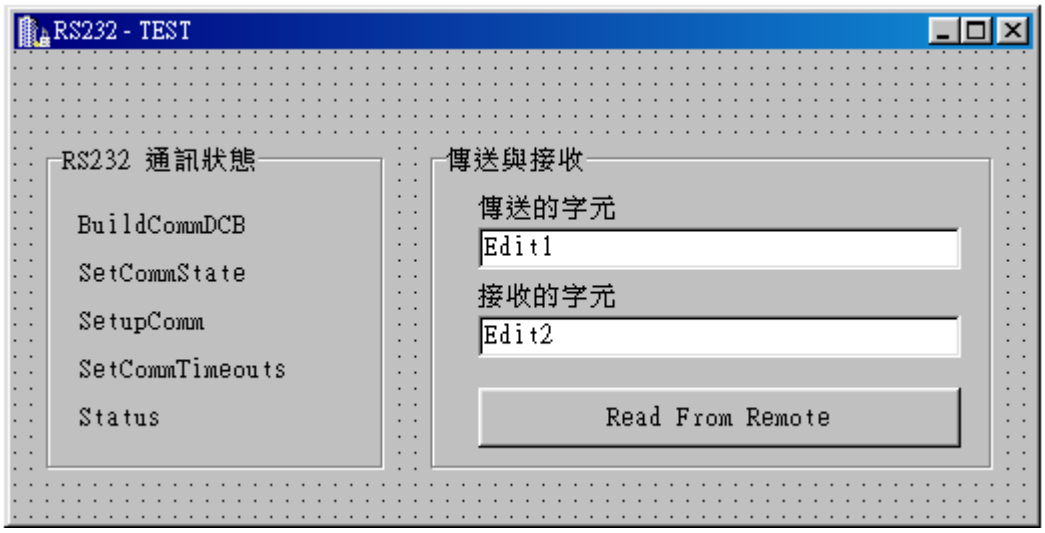
本次以一個 8051 單板當作遠端裝置(Remote), 透過一條 RS232 傳輸線與電腦相互溝通做為範例. 在 PC 端使用 Borland C++ Builder, 而在 Remote 8051 端則使用組合語言. 由於本文章是以如何使用 Win32 API 存取 Serial port 為主, 所以在 8051 的程式上並沒有太複雜的流程與技巧, 故不作詳細的說明.

壹、功能:

讓游標停留在 "傳送的字元" 下的文字方塊內, 當 User 按下鍵盤上任一字元時, PC 端送出一個字元給 8051 Remote, 8051 Remote 收到此字元後, 立即回覆相同的字元給 PC 端, 並將此字元顯示在 "接收的字元" 下的文字方塊中. 若當 User 按下 Read From Remote 時, 8051 單板收到 "!" 字元, 立即傳回一個二進制檔(binary file)給 PC, 此二進制檔的內容是一個 16 色的 Bitmap File (BMP 檔). PC 端在接收完整個 Bitmap File 後, 自動以 C16.BMP 存檔. 8051 單板的線路圖請參照圖二.

貳、程式&說明

一、PC (Host) 方面
1.放置如圖一所示之控制項.



圖一

2.設定物件屬性，見表一.

物件	Name	Caption
Label	lbBuildCommDCB	BuildCommDCB
Label	lbSetCommState	SetCommState
Label	lbSetupComm	SetupComm
Label	lbSetCommTimeouts	SetCommTimeouts
Label	lbStatus	Status
Label	Label1	傳送的字元
Label	Label2	接收的字元
GroupBox	GroupBox1	RS232 通訊狀態
GroupBox	GroupBox2	傳送與接收
Edit	Edit1	Edit1
Edit	Edit2	Edit2
Button	btnRead	Read From Remote

表一

3.程式碼.

```
HANDLE handle;
FILE *infile;
FILE *outfile;
char TxdBuffer;
char RxdBuffer;
DWORD dwNoByte;
```

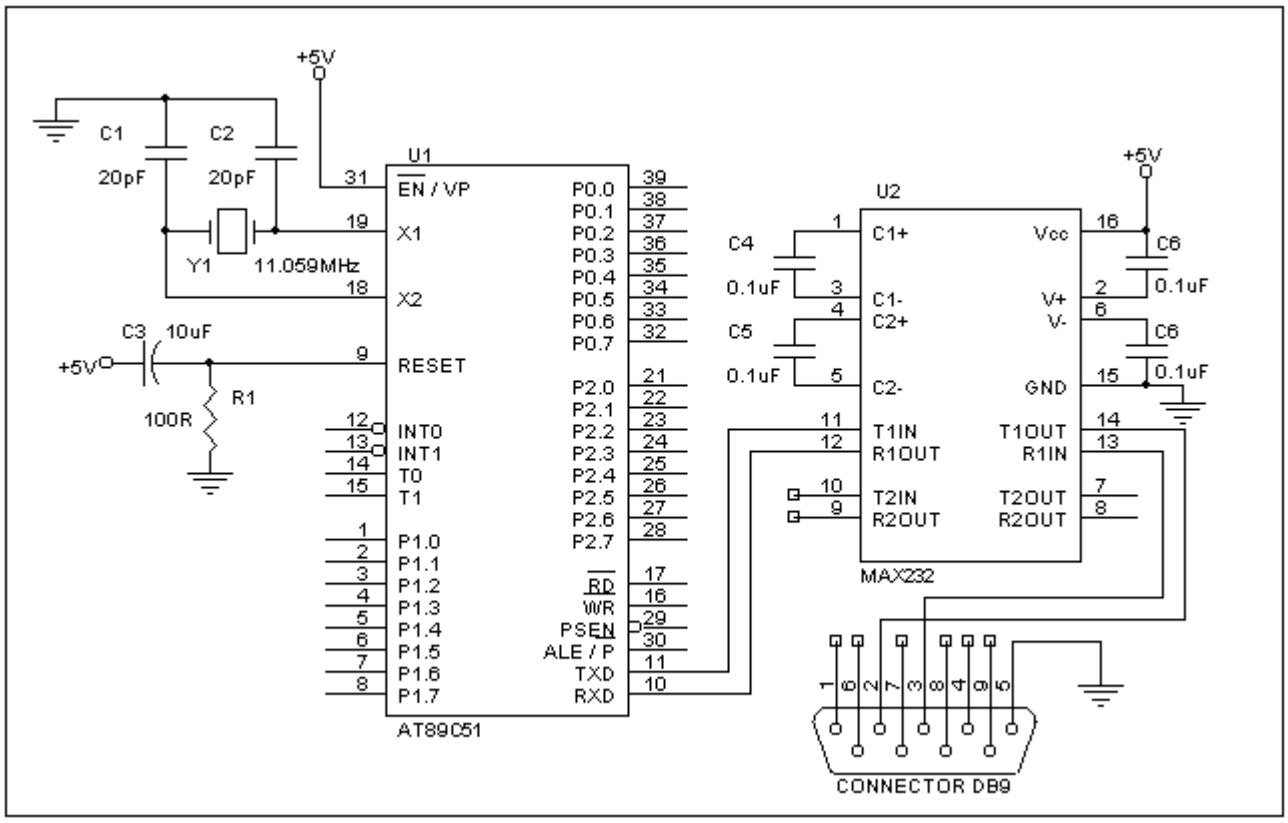
```
DCB dcb;

void __fastcall TForm1::FormCreate(TObject *Sender)
{
    Edit1 - Text = "";
    Edit2 - Text = "";

    COMMTIMEOUTS TimeOut;
    TimeOut.ReadIntervalTimeout         = 0;
    TimeOut.ReadTotalTimeoutMultiplier  = 0;
    TimeOut.ReadTotalTimeoutConstant    = 500;  //(總讀取 time out 時間 = 0.5 秒)
    TimeOut.WriteTotalTimeoutMultiplier = 0;
    TimeOut.WriteTotalTimeoutConstant   = 500; //(總寫入 time out 時間 = 0.5 秒)

    if (BuildCommDCB("9600,n,8,1", &dcb))
    {
        lbBuildCommDCB - Caption = "BuildCommDCB 成功";
    }
    else
    {
        lbBuildCommDCB - Caption = "BuildCommDCB 失敗";
    }

    handle = CreateFile("Com1",
                        GENERIC_READ | GENERIC_WRITE,
                        0,
                        0,
                        OPEN_EXISTING,
                        FILE_ATTRIBUTE_NORMAL,
                        0);

    if (SetCommState(handle, &dcb))
        lbSetCommState - Caption = "SetCommState 成功";
    else
        lbSetCommState - Caption = "SetCommState 失敗";

    if (SetupComm(handle, 1024, 1024))
        lbSetupComm - Caption = "SetupComm 成功";
    else
```

```
        lbSetupComm - Caption = "SetupComm 失敗";

    if (SetCommTimeouts(handle, &TimeOut))
        lbSetCommTimeouts - Caption = "SetCommTimeouts 成功";
    else
        lbSetCommTimeouts - Caption = "SetCommTimeouts 失敗";

}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnReadClick(TObject *Sender)
{
    TxdBuffer = '!';
    char Name[] = {"C16.bmp"};
    int a = 0;

    outfile = fopen(Name, "wb");
    WriteFile(handle, &TxdBuffer, 1, &dwNoByte, 0);

    ReadFile(handle, &RxdBuffer, 1, &dwNoByte, NULL);
    if (dwNoByte == 0 )
    {
        lbStatus - Caption = "沒有回應 !";
        return;
    }

    a++;
    fwrite(&RxdBuffer, 1, 1, outfile);
    while(dwNoByte != 0)
    {
        ReadFile(handle, &RxdBuffer, 1, &dwNoByte, NULL);
        fwrite(&RxdBuffer, 1, 1, outfile);
        a++;
    }

    fclose(outfile);

    lbStatus - Caption = "共接收了 " + IntToStr(a) + " Bytes";

}
```

```
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit1KeyPress(TObject *Sender, char &Key)
{
    char lpBuf;
    char WriteBuffer;
    char *ptr;
    DWORD dwWrite;
    DWORD dwRead;

    ptr = &Key;
    WriteBuffer = *ptr;
    WriteFile(handle, &WriteBuffer, 1, &dwWrite, 0);
    ReadFile(handle, &lpBuf, 1, &dwRead, NULL);
    if (dwRead == 0)
    {
        lbStatus - Caption = "沒有回應 !";
    }
    else
    {
        Edit2 - Text = Edit2 - Text + lpBuf;
    }
}
//---------------------------------------------------------------------------
```

二、8051 (Remote) 方面

1.硬體如圖二所示



<圖二 >

2.程式碼.

```
;*********************************************************
;* 使用 RS232 傳送 Binrary File For 8051.  作者:鄭旭崇  *
;*********************************************************

    LLCHAR ?                ;宣告 Local Label Character = '?'

    ORG 0H
    JMP BEGIN
;-------------------------------------------------------------------------
BEGIN:
    MOV     SP,#60H         ;
    MOV     SCON,#50H       ;Serial Port 傳輸格式: 9600,N,8,1
    MOV     TMOD,#20H       ;
    MOV     TH1,#0FDH       ;
    SETB    TR1
    SETB    TI
```

```
    MOV     DPTR,#Hello     ;8051 單板一開機,先送出歡迎詞給 PC 端.
    CALL    SendStr         ;
Again:
    JNB     RI,$            ;等待 PC 送字元過來.
    CLR     RI              ;
    MOV     A,SBUF          ;

    CJNE    A,'!',?10       ;不是 '!' 就將原字元傳回 PC 端.
    CALL    SendBinaryFile      ;是 '!' 就傳回二進制檔(Binary File).
    JMP Again

?10 CALL    SendByte
    JMP     Again
;===============================================================================
SendStr:        ;送出一字串.
?10 CLR     A
    MOVC    A,@A+DPTR       ;從 ROM 裡面取一個 Byte.
    CJNE    A,0,?20         ;
    RET

?20 CALL    SendByte
    INC     DPTR
    JMP     ?10
    RET
;===============================================================================
SendBinaryFile:     ;送出二進制檔(Binary File).

    MOV DPTR,#BinData
    MOV R4,#4           ;準備傳送出 200 * 4 = 800 個 Bytes.

?10 MOV R5,#200
?20 CLR     A
    MOVC    A,@A+DPTR       ;從 ROM 裡面取一個 Byte.
    CALL    SendByte
    INC     DPTR
    DJNZ    R5,?20
    DJNZ    R4,?10

    RET
```

```
;===============================================================================
SendByte:       ;送出一字元.
    JNB     TI,$
    CLR     TI
    MOV     SBUF,A
    RET
;===============================================================================

Hello:  DB 'Hello Welcome !',0

BinData:
    DB   42H,4DH,1EH,03H,00H,00H,00H,00H,00H,00H,76H,00H,00H,00H,28H,00H
    DB   00H,00H,22H,00H,00H,00H,22H,00H,00H,00H,01H,00H,04H,00H,00H,00H
    DB   00H,00H,A8H,02H,00H,00H,C4H,0EH,00H,00H,C4H,0EH,00H,00H,00H,00H
    DB   00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,80H,00H,00H,80H
    DB   00H,00H,00H,80H,80H,00H,80H,00H,00H,00H,80H,00H,80H,00H,80H,80H
    DB   00H,00H,C0H,C0H,C0H,00H,80H,80H,80H,00H,00H,00H,FFH,00H,00H,FFH
    DB   00H,00H,00H,FFH,FFH,00H,FFH,00H,00H,00H,FFH,00H,FFH,00H,FFH,FFH
    DB   00H,00H,FFH,FFH,FFH,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H
    DB   00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H
    DB   00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,AAH,AAH
    DB   AAH,AAH,AAH,AAH,99H,99H,99H,99H,FFH,FFH,FFH,99H,99H,00H,00H,00H
    DB   00H,00H,AAH,AAH,AAH,AAH,AAH,AAH,99H,99H,99H,99H,FFH,FFH,FFH,99H
    DB   99H,00H,00H,00H,00H,00H,00H,99H,AAH,AAH,AAH,AAH,AAH,99H,99H,99H
    DB   99H,FFH,FFH,FFH,99H,99H,00H,00H,00H,00H,00H,99H,AAH,AAH,AAH,AAH
    DB   AAH,99H,99H,99H,99H,FFH,FFH,FFH,99H,99H,00H,00H,00H,00H,00H,00H
    DB   99H,AAH,99H,99H,AAH,AAH,99H,99H,FFH,FFH,FFH,FFH,99H,00H,00H,00H
    DB   00H,00H,00H,00H,99H,AAH,99H,99H,AAH,AAH,99H,99H,FFH,FFH,FFH,FFH
    DB   99H,00H,00H,00H,00H,00H,00H,00H,00H,AAH,99H,99H,99H,AAH,AAH,FFH
    DB   FFH,FFH,FFH,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,AAH,99H,99H
    DB   99H,AAH,AAH,FFH,FFH,FFH,FFH,00H,00H,00H,00H,00H,00H,00H,00H,00H
    DB   00H,AAH,99H,99H,99H,AAH,AAH,AAH,AAH,AAH,AAH,AAH,00H,00H,00H,00H
    DB   00H,00H,00H,00H,00H,AAH,99H,99H,99H,AAH,AAH,AAH,AAH,AAH,AAH,AAH
    DB   00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,AAH,99H,AAH,AAH,00H,00H
    DB   00H,FFH,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,AAH,99H
    DB   AAH,AAH,00H,00H,00H,FFH,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H
    DB   00H,99H,AAH,AAH,AAH,AAH,AAH,FFH,FFH,AAH,FFH,AAH,AAH,00H,00H,00H
    DB   00H,00H,00H,00H,00H,99H,AAH,AAH,AAH,AAH,AAH,FFH,FFH,AAH,FFH,AAH
    DB   AAH,00H,00H,00H,00H,00H,00H,00H,99H,99H,AAH,AAH,AAH,AAH,FFH,FFH
```

```
    DB   00H,AAH,00H,FFH,AAH,00H,00H,00H,00H,00H,00H,00H,99H,99H,AAH,AAH
    DB   AAH,AAH,FFH,FFH,00H,AAH,00H,FFH,AAH,00H,00H,00H,00H,00H,00H,00H
    DB   00H,00H,AAH,AAH,AAH,AAH,FFH,FFH,00H,AAH,00H,FFH,AAH,00H,00H,00H
    DB   00H,00H,00H,00H,00H,00H,AAH,AAH,AAH,AAH,FFH,FFH,00H,AAH,00H,FFH
    DB   AAH,00H,00H,00H,00H,00H,00H,00H,00H,99H,AAH,AAH,AAH,AAH,FFH,FFH
    DB   00H,AAH,00H,FFH,00H,00H,00H,00H,00H,00H,00H,00H,00H,99H,AAH,AAH
    DB   AAH,AAH,FFH,FFH,00H,AAH,00H,FFH,00H,00H,00H,00H,00H,00H,00H,00H
    DB   99H,99H,99H,AAH,AAH,AAH,FFH,FFH,FFH,AAH,FFH,FFH,00H,00H,00H,00H
    DB   00H,00H,00H,00H,99H,99H,99H,AAH,AAH,AAH,FFH,FFH,FFH,AAH,FFH,FFH
    DB   00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,AAH,AAH,AAH,AAH,FFH
    DB   FFH,AAH,FFH,AAH,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,AAH
    DB   AAH,AAH,AAH,FFH,FFH,AAH,FFH,AAH,00H,00H,00H,00H,00H,00H,00H,00H
    DB   00H,00H,99H,99H,AAH,AAH,AAH,AAH,AAH,AAH,AAH,00H,00H,00H,00H,00H
    DB   00H,00H,00H,00H,00H,00H,99H,99H,AAH,AAH,AAH,AAH,AAH,AAH,AAH,00H
    DB   00H,00H,00H,00H,00H,00H,00H,00H,00H,99H,99H,99H,99H,AAH,AAH,AAH
    DB   AAH,AAH,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,99H,99H,99H
    DB   99H,AAH,AAH,AAH,AAH,AAH,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H
    DB   00H,00H,00H,00H,00H,99H,99H,99H,00H,00H,00H,00H,00H,00H,00H,00H
    DB   00H,00H,00H,00H,00H,00H,00H,00H,00H,99H,99H,99H,00H,00H,00H,00H
    DB   00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,99H,00H
    DB   00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H
    DB   00H,00H,99H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,8BH,D1H

        END
;-------------------------------------------------------------------------
```

3.組譯程式，在 DOS 提示字元下輸入:

```
C:\X8051 REMOTE51

          8051 Macro Assembler   -   Version 4.03a
       Copyright (C) 1985 by 2500 A.D. Software, Inc.


               ***** Active Commands *****

                 Ctrl S = Stop Output
                 Ctrl Q = Start Output
                 Esc  C = Stop Assembly
                 Esc  T = Terminal Output
```

```
                Esc  P = Printer Output
                Esc  D = Disk Output
                Esc  M = Multiple Output
                Esc  N = No Output


      2500 A.D. 8051 Macro Assembler  -  Version 4.03a
      ------------------------------------------------

             Input  Filename : REMOTE51.asm
             Output Filename : REMOTE51.obj

   Lines Assembled :  116           Assembly Errors :  0
```

4.連結程式，在 DOS 提示字元下輸入:

```
C:\LINK51 -c REMOTE51 -O -X

2500 A.D. Linker  Copyright (C) 1985  -  Version 4.03a

*******************************************************************************
*                          L O A D      M A P                                 *
*******************************************************************************
*    Section Name             Starting Address    Ending Address    Size    *
*******************************************************************************
* remote51.obj                                                                *
*    CODE                          0000              0383         0384    *
*******************************************************************************

                Linker Output Filename : remote51.tsk
                Disk Listing  Filename :
                Symbol Table  Filename :

        Link Errors :  0              Output Format :  Executable
```

參、執行例

請看圖三至圖五.

<圖三>鍵入 "Lovely",Remote 8051 回覆 "Lovely"

<圖四> 按下 Read From Remote 鈕, Remote 8051 傳回一個二進制檔(binary file)給 PC,此二進制檔的內容是一個 16 色的 Bitmap File (BMP 檔).

<圖五> C16.BMP 的內容

§結語

看完了本篇文章後, 是否對 Win32 API 如何存取 Serial Port 更加瞭解呢? 雖然我用 Borland C++ Builder 來講解程式, 但小弟覺得用何種 Tool 或程式語言的本身並不重要, 重要的是能夠攝取文章中的靈魂, 進而加以吸收, 變成自己的東西, 在各種程式語言或情況下都能運用自如.

讀者朋友對於本文章有任何問題或建議，歡迎您 E-mail 給我，小弟的 E-mail 是: pttgood@cm1.hinet.net 當然，讀者朋友們想要程式的 Source Code 的話，也可以來信索取.

幾個月前我的好友 Tony 的老婆 LCM 生了一個白白胖胖的兒子，他們叫他 "阿肥". 看到胖嘟嘟的阿肥和他們倆如此幸福的臉龐，頓時讓我覺得人活著就應該感恩惜福，自然就會幸福快樂。祝福每一位讀者健康快樂。