報告題名：

以 8051 整合進行網際網路遠端家電控制之開發與應用

作者：張原誌

系級：通訊系四年甲班

學號：D9157354

開課老師：楊豐瑞

課程名稱：專題研究

開課系所：通訊工程學系

開課學年：　93　學年度 第 2 學期

目錄

圖目錄

表目錄

摘要

隨著現今網路技術的突飛猛進，光纖技術的應用及日益提高的網路頻寬，網路設備幾乎遍及世界各個角落。如此便利的網路，除了可以上網獲取豐富的資訊之外，也可以用來遠端遙控一些硬體設備，這樣的概念及可應用在家庭電器的遠端控制上。例如遠端控制門禁監視器，一發現可疑人物，透過對步進馬達的控制，調整監視器攝影機角度，持續跟拍監控。或是人在公司時，回家之前設定好家裡的冷氣溫度與運作時間。本專題在此即是透過網際網路傳輸技術，讓使用者能控制遠端的硬體週邊設備；藉由網際網路的便利，不管使用者身在何處只要能連上網際網路，不需要安裝任何遠端控制軟體，就可以輕鬆的透過網頁瀏覽器控制遠端設備。

以 ASP 語言建構的 Client /Server 架構，因為是一種網頁語言，並無法直接操控遠端硬體設備，過程中間必須透過 C 語言所寫的硬體控制程式，才能達到硬體控制的目的。整個系統是以一般人最熟悉的微軟 Windows XP 做為作業系統，使用 IIS 網頁伺服器提供客戶端遠端控制網頁讓使用者讀取。使用者只需利用網頁瀏覽器(Internet Explorer、Netscape)，即可讀取 Client 端遠端控制網頁。當 Client 端連上遠端控制網頁之後，便可在網頁上輸入原先設計好的控制指令，透過對 Server 端 c 程式的呼叫，再透過 RS-232 傳輸到 MCS-51；MCS-51 判別所接收到的指令選擇要控制的其他硬體，接下來再透過一些轉換電路或硬體設備即可達到遠端控制家電的目的。不只家電的應用，在工業生產線上的應用，例如步進馬達的控制..等等都是可以做到的。使用 MCS-51 控制硬體的好處，在於其功能強大，價格便宜，而且可以每個 I/O port 的 Pin 都可控制一個硬體。

第一章 緒論

1.1 製作動機與目的

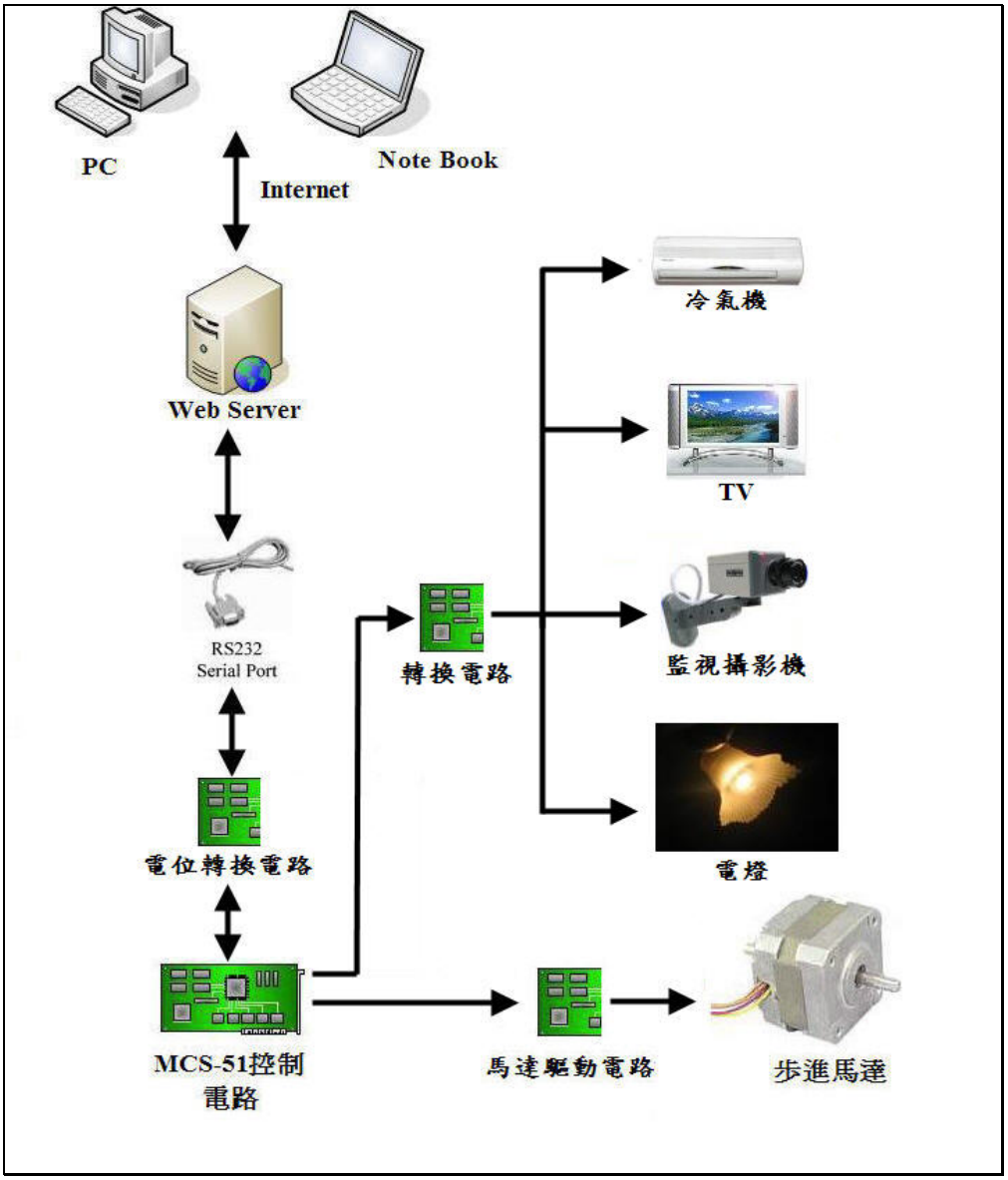
有鑑於目前大部分的家電用品都必須靠人親自操作與設定，雖然有些電器可以預先設定開機時間或是於一定時間自動運作，例如錄影機，冷氣機..等等；但是，當有突發狀況發生時，往往無法做出任何反應。例如本來預定六點回家吃飯，電鍋設定於六點煮好飯，但臨時要加班，不就白煮了一鍋飯了。因此利用目前國內家庭網路的普及，以及光纖技術帶來的廣大頻寬，來達到遠端控制家電的目的。

至於使用者控制介面，為了讓使用者不需要隨身攜帶遠端控制軟體的考量，所以選擇 ASP 動態網頁為控制介面。不管使用者身在何處，只要可以上網連線到遠端控制網頁伺服器，即可輕易控制遠端硬體。如此一來，就算電器本身或是人為因素出現突發狀況，遠端的使用者也能立即監控，做出較正確的反應。在系統上，未來更可搭配網頁伺服器的資料庫系統，對於遠端使用者的管理與權限分配，定期對控制的硬體狀態與保養維修日期進行紀錄，以降低發生意外的機率。而硬體方面選擇了 8051 來做控制硬體的中樞系統，因其價格低廉而且功能強大，加上 I/O 腳位多，可以應付種類眾多的家電設備，甚至可以做到單一指令，同時控制多組硬體。而遠端應用不只在家電設備，也可應用在工業生產線上，例如跟步進馬達相關的硬體設備，或是錄影監視器也都能透過 8051 加上驅動電路去做精準的定位控制，對於陌生的可疑人物，錄影鏡頭轉動追蹤，在保安上亦可應用。

1.2 專題架構

本專題遠端控制系統架構可參考圖 1.1，使用者可使用 PC、Notebook、PDA 等設備透過 Internet 連線登入 Web Server，經帳號與密碼驗證無誤即可進入網頁控制介面。而此使用者介面使是使用 ASP 網頁語言製作，而 ASP 本身並無法直接控制硬體，而是利用呼叫 Web Server 端硬體控制應用程式的方式。此硬體控制之程式則是利用 Windows API 來達到控制 serial port 的目的，使網頁的控制指令能透過 serial port 傳送到 MCS51-控制電路；而就在 Web Server 端透過 serial port 傳送資料到 MCS-51 控制電路之間，由於兩者的電壓準位定義不同，因此需要透過以 MAX-232 IC 為核心的電位轉換電路的幫助，才不至於發生資料讀寫錯誤的問題。而 MSC-51 控制電路所扮演的的角色則是將 SBUF 接收暫存器所接收的控制指令進行判別，並且執行該指令之 I/O 動作與相關設定。而硬體控制電路之所以選擇 MCS-51 單晶片為核心，主要是因為 MCS-51

符合經濟效益，而且 I/O 腳位充足，可銜接多種周邊硬體，做為硬體控制電路核心非常合適。



▲ 圖 1.1 專題架構

至此，已經完成遠端控制硬體所需的開關插座，中間只需特定的轉換電路或是驅動電路即可連結許多欲控制的硬體設備。例如，欲控制步進馬達只需透過馬達趨動電路與 MCS-51 控制電路連結，修改 MCS-51 控制程式對馬達輸入連續脈衝信號造成激磁效果，即可輕易控制步進馬達正反轉、轉速與定位的控制。相同的，欲控制其他家電設備例如，冷氣機開關的時間設定與溫度控制、電視開關、監視攝影機鏡頭拍攝角度調整與電燈的開關設定等等，皆只需要透過相對應之轉換電路或是驅動電路設備，修改 MCS-51 控制程式及 I/O 動作，即可完成遠端控制。

第二章 軟硬體工具簡介

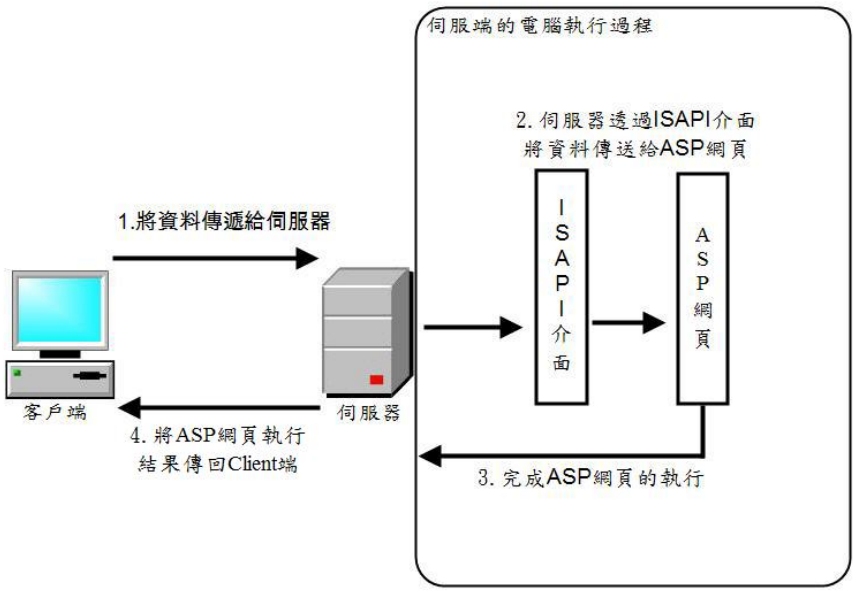
2.1 ASP (Active Server Pages)

2.1.1 ASP 簡介

Active Server Pages 簡稱 ASP (動態伺服器網頁)，並不是一種語言，而是指由 WEB 伺服器提供的網頁語言執行環境。 為 Microsoft IIS(for Windows NT ，Windows 2000 之後版本)與 PWS(for Windows 98)伺服器所提供的一項用來替代傳統 CGI 程式的技術。可以看成類似 HTML(Hypertext Markup Language) ，Script ，CGI (Common Gateway Interface 通用閘道介面)的結合體，其執行效率比 CGI 更好，程式寫作也比 HTML 更便利且更有彈性，程式安全及保密性也遠比 Script 佳。除此之外，還提供伺服器端程式物件(檔案與資料庫存取，郵件傳送。。等等)的使用語法，有效地增強了 ASP 的功能。

2.1.2 ASP 程式執行架構

ASP 程式是以*.asp 為副檔名的 HTML 格式檔案，除了可以包含一般的 HTML 標記與各類型用戶端 Script 之外，也可以包含在伺服器上執行的 ASP 程式碼，所以運作流程為副檔名為*.asp 的程式經由 ASP 直譯器的解釋及執行後，再將執行果與其他 HTML 碼組合成最後的網頁資料， 傳回到用戶端的瀏覽器上，ASP 執行流程如下圖所示：



▲ 圖 2.1 ASP 程式執行架構

ISAPI(Internet Server Application Programming Interface)是微軟所研發出來的，用在 IIS 伺服器上的 CGI 程式解決方案，ISAPI 使用 Win32 程式設計中常用的 DLL 程式庫來發展。不論同時有多少使用者執行 ISAPI 程式，伺服器記憶只會有一份程式，可解決 CGI 耗用主機資源問題，而且 ISAPI 程式是經過編譯後的程式， 因此執行效能上遠遠超過 CGI 程式，不過 ISAPI 程式只能適用在微軟的伺服器中。

2.1.3 ASP 的特性

ASP 程式除了有一般傳統直譯是 CGI 程式的優點外，更延伸出許多在程式編寫與執行效率上的重大改進，其特性如下：

1. 由於 ASP 的敘述可以直接撰寫在包含於*.asp 為副檔名的 HTML 格式檔案中，也可以和其他 Script 語言(VB Script 與 Java Script)相互內嵌，因此易於修改與測試。
2. 可以使用 asp 來存取伺服器上的 ActiveX 元件，因此可以透過在伺服器上製作新的 ActiveX 元件，來增加 ASP 的功能。
3. 伺服器上的 ASP 直譯器會在伺服器端執行 ASP 程式，並將結果以 HTML 格式傳送到用戶端瀏覽器上，因此各種瀏覽器都瀏覽 ASP 網頁。
4. ASP 除了預設用 VBScript 語言外，也可以使用 Jscript ，PerlScript 等等語言來編寫。只要將使用該語言的程式內插<script language="使用的語言名稱” runat=”seerver”>~</script>裡面，便可以使用上述語言來混合編寫。
5. 可以利用 ASP 內建的 Session 與 Application 物件來記錄不同使用者自己本身與共用的資料，由於這些資料都直接存放在伺服器的主記憶體上，可以有效提升執行效能。
6. ASP 以物件導向為基礎，因此可以使用 ActiveX 元件無限擴充功能。
7. 可以使用 ASP 內建的 ADO 元件，輕鬆的存取各種資料庫，大量縮減開發時間。
8. 由於伺服器是將 ASP 程式執行的結果以 HTNL 格式傳回用戶端瀏覽器，所以使用者是無法看到 ASP 所編寫的原始程式碼，因此程式碼可以完全保密。

2.1.4 ASP 3.0 基本物件

由 Microsoft 提供的 ASP 物件目前版本為 ASP 3.0，總共有七種物件，其物件用途說明如下：

1. Application 物件：

   以 ASP 觀點來說，某虛擬目錄下所有 ASP 網頁構成一個 Web 應用程式，而 Application 物件便代表著這整個應用程式。

2. ASPError 物件：

   ASP 網頁執行錯誤時，所產生的物件。

3. ObjectContext 物件：

   此物件可提供程式設計師利用 Microsoft Transaction Server 處理交易。

4. Request 物件：

   用於取得從客戶端瀏覽器中，透過 HTTP 協定送至伺服器上 Web 伺服器的資訊。

5. Response 物件：

   用於處理從伺服端 Web Server 輸出到客戶端資料的物件。

6. Server 物件：

   Server 物件用於代表 Web Server，透過此物件的應用，可以去得 Web Server 的資料與執行狀態。

7. Session 物件：

   當有使用者連線至網頁時，Session 物件便用於代表某個使用者的連線。

2.2 RS-232 串列介面

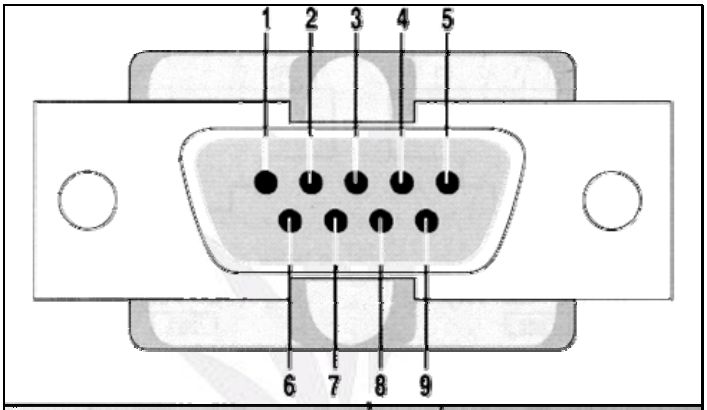
2.2.1 串列傳輸

串列傳輸為 CPU 與周邊裝置或 CPU 與 CPU 間的資料傳輸方法之一，最簡單的串列傳輸只需兩條傳輸線，使用時的方式每次傳輸一個位元的資料，所以具有傳輸線少的優點，並且容易防止雜訊干擾，適合較遠距離的資料傳輸。然而，由於資料傳輸一次僅送一個位元，因此傳輸資料的速度慢是其缺點。串列通訊的方式可以分為同步模式 (Synchronous) 及非同步模式 (Asynchronous) 兩種。同步模式在通訊的兩端使用同步訊號做為通訊的依據;而非同步模式則使用起始位元 (Start Bit) 及停止位元 (Stop Bit) 作為通訊的判斷，現在則是以非同步傳輸較多。非同步傳輸只要 9 支腳位;如果要採用同步傳輸模式則需使用到 25 支腳位。

2.2.2 RS-232 腳位意義

D-Type-25 Pin No.	D-Type-9 Pin No.	Abbreviation	Full Name
Pin 2	Pin 3	TD	Transmit Data
Pin 3	Pin 2	RD	Receive Data
Pin 4	Pin 7	RTS	Request To Send
Pin 5	Pin 8	CTS	Clear To Send
Pin 6	Pin 6	DSR	Data Set Ready
Pin 7	Pin 5	SG	Signal Ground
Pin 8	Pin 1	CD	Carrier Detect
Pin 20	Pin 4	DTR	Data Terminal Ready
Pin 22	Pin 9	RI	Ring Indicator

▲ 表 2.1 RS-232 腳位說明



Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		

▲ 圖 2.2 COM Port 腳位說明

RS-232 的每一支腳位都有其功能與訊號流動方向;由於 RS-232 設計之初是用來接數據機作傳輸之用，因此它的腳位意義通常也和數據機有密切的關聯。可參考圖 2.2 與表 2.1。以下是 9 支腳位的說明:

1. **CD：** 由數據機所控制，當電話接通後，傳送的訊號載在載波上面，數據機利用此腳位通知電腦有載波被偵測到，也就是表示目前為On-Line狀態；若電腦未收到此訊號，則會回應訊息，並且將數據機掛線(Hang Up)。
2. **RXD:** 此腳位會接收從遠端傳送過來的資料。
3. **TXD:** 此腳位會將PC所欲傳送出去的資料傳送出去。
4. **DTR:** 此腳位由電腦控制，高電位時，表示電腦已準備好接收資料，通知數據

機可以傳輸。

5. **GND:** 此腳位為地線，做為電腦與遠端設備之間的準位參考。傳輸雙方的地線準位必須一樣，以避免準位不同而造成資料的傳送錯誤。
6. **DSR:** 此腳位由數據機控制，腳位高電位代表數據機通知PC可以傳送資料過來。
7. **RTS:** 此腳位由電腦控制，用來通知數據機馬上傳送資料到電腦。當數據機收到此信號後，便會將它收到的資料傳送給電腦。
8. **CTS:** 此腳位由數據機控制，用以通知PC將欲傳送的資料送至數據機;而當電腦收到此訊號後，便會將準備送出的資料送至數據機。
9. **RI:** 數據機通知電腦有電話進來，是否接聽電話則由電腦決定。

2.2.3 串列傳輸的資料格式

因為採用非同步式串列介面 (Universal Asynchronous Receiver Transmitter，簡稱 UART)，其資料格式如下：

標記	起始位元	資料位元…	同位位元	停止位元

▲ 表 2.2 串列傳輸資料格式

1. **標記：**

當串列傳輸線上不傳送資料時，所處的狀態稱為標記狀態，用以告知對方目前是處於 idle 狀態下，此信號會一直保持高電位。

2. **起始位元：**

當發送端準備開始傳送資料前，會在要送出的字元前面加上高電位的起始位元(邏輯 0)，接收端會因為起始位元的觸發而開始接收資料。

3. **資料位元：**

發送端真正要傳送的資料，在起始位元送出後，依照位元 0，位元 1..的順序，一個 bit 接一個 bit 地傳送出去，資料長度範圍 5~8 。

4. **同位位元：**

為了預防傳輸過程中錯誤的發生，所以使用同位位元做為檢查的機制;同位位元是一種用來檢查傳送資料的正確性的一種核對碼，可分成奇同位(Odd Parity)跟偶同位(Even Parity)。奇同位檢查，就是將所有資料位元加上同位位元後，1

的個數為奇數，才算正確;而偶同位檢查，反之，偶同位檢查就是將所有資料位元加上同位位元後，1 的個數為偶數，才算正確。

5. 停止位元：

而在同位位元之後的最後一個位元，稱為停止位元，用於表示一個位元組的資料已經傳送完成。停止位元可以是 1 個，1.5 個，2 個;可以使用者需要做選擇。

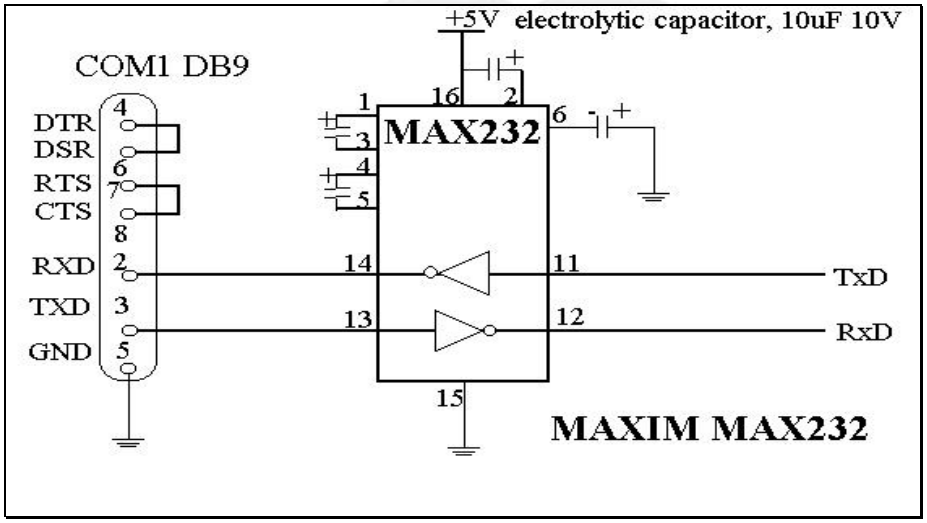
2.2.4 RS-232 傳輸速率 (Baud Rate)

串列傳輸是以分時的方式，將一個 bit 資料狀態(0 或 1)呈現在傳輸線上，如果這個 bit 在傳輸線上呈現的時間越短，則資料傳輸的速度就越快。通常是以每秒傳送多少 bit 來衡量傳輸速率，其單位為 bit/sec，稱之為位元率(Bit Rate)或鮑率(Baud Rate)。目前常用的鮑率有 19200 ，9600， 4800， 2400， 1200(bps)。要注意的是，傳送端與接收端鮑率務必設定一致，否則會無法正確接收資料，在本專題中使用 1200bps，因為並沒有複雜的控制資料傳輸，也沒有需要 real time，因此選擇較低的鮑率，另一方面速率較低也可降低傳輸錯誤的機率。

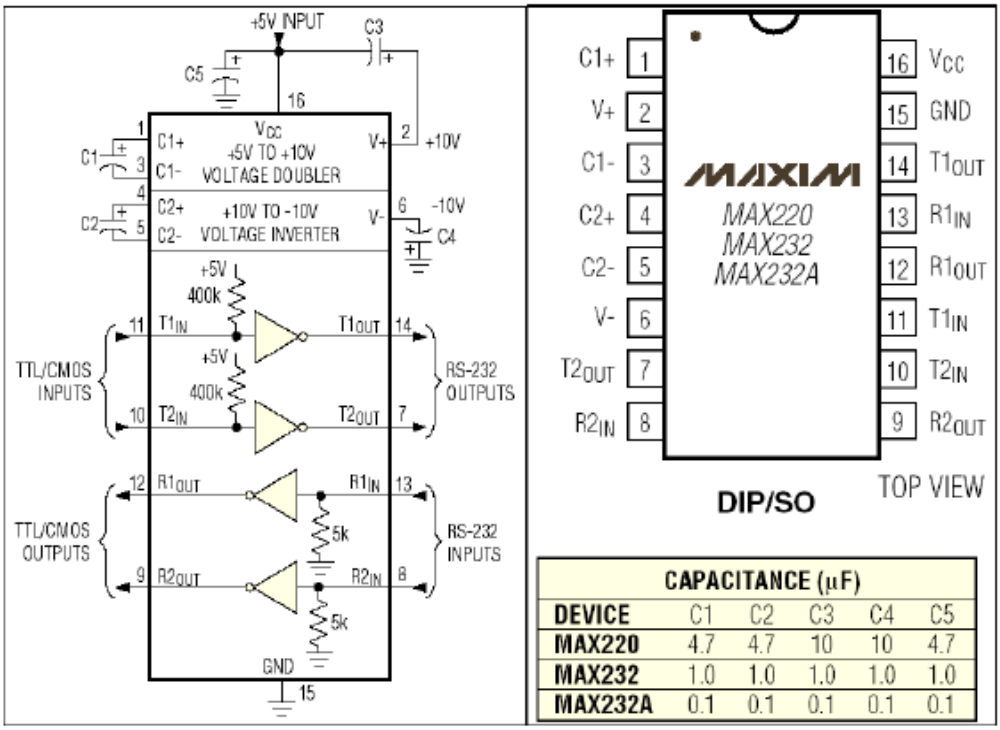
2.2.5 訊號位準轉換 IC

PC 端串列埠與 8051 的串列介面溝通需要一個訊號位準轉換 IC，一般來講是使用 MAX232 這顆 IC.。MAX232 之用途在於轉換數位訊號準位。8051 所用的 Logic Level 為 High=5V，Low=0V，而 RS232 則 High=-3~-12V， Low=3~12V。藉由 MAX232 將電位轉換，才使得電腦與 8051 可以正確溝通。亦可使用 ICL232 替代。

MAX 232IC 與 Serial port 連接圖示如下：



▲ 圖 2.3 MAX-232 與 COM Port 電位轉換電路示意圖



CAPACITANCE (μF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	4.7	4.7	10	10	4.7
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

▲ 圖 2.4 MAX-232 內部電路

2.3 MCS-8051 單晶片介紹

2.3.1 8051 版本與特性

MSC-8051 系列單晶片是美國 INTEL 公司推出 MCS-8048 系列晶片之後所開發出來功能完備的單晶。MCS-8051 系列單晶片依照電路結構的不同，大致上可分為三大類 (1)晶片內不含程式記憶體(ROM 型態) (2)晶片內含程式記憶體(ROM 型態) (3)晶片內含程式記憶體(EPROM 型態)。下表為常用的 MCS-51 晶片編號及特性。

ROM 版本	EPROM 版本	ROMLESS 版本	ROM 容量	RAM 容量	計時器	電路型態
8051	8751	8031	4K bytes	128bytes	2	HMOS
8052	8752	8032	8K bytes	256bytes	3	HMOS
80C51	87C51	80C31	4K bytes	128bytes	2	CHMOS
80C52	--	80C32	8K bytes	256bytes	3	CHMOS

▲ 表 2.3 8051 版本分類介紹表

8051 單晶片重要特性歸納如下:

(1) 8-bit CPU.

(2) extensive Boolean processing(single bit logic) capabilities.

(3) 64-KB program memory address space.

(4) 64-KB data memory address space.

(5) 4 KB of on-chip program memory.

(6) 128 bytes of on-chip data random access memory.

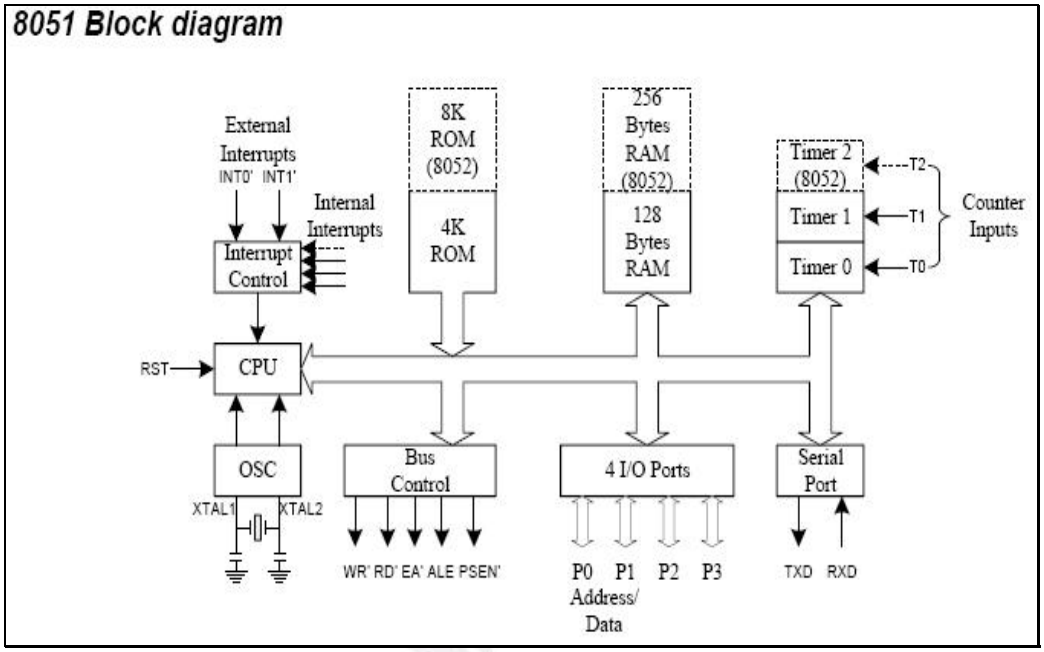
(7) 32 bidirectional and individually addressable I/O lines.

(8) two 16-bit timers/counters.

(9) full duplex universal asynchronous receiver transmitter(UART).

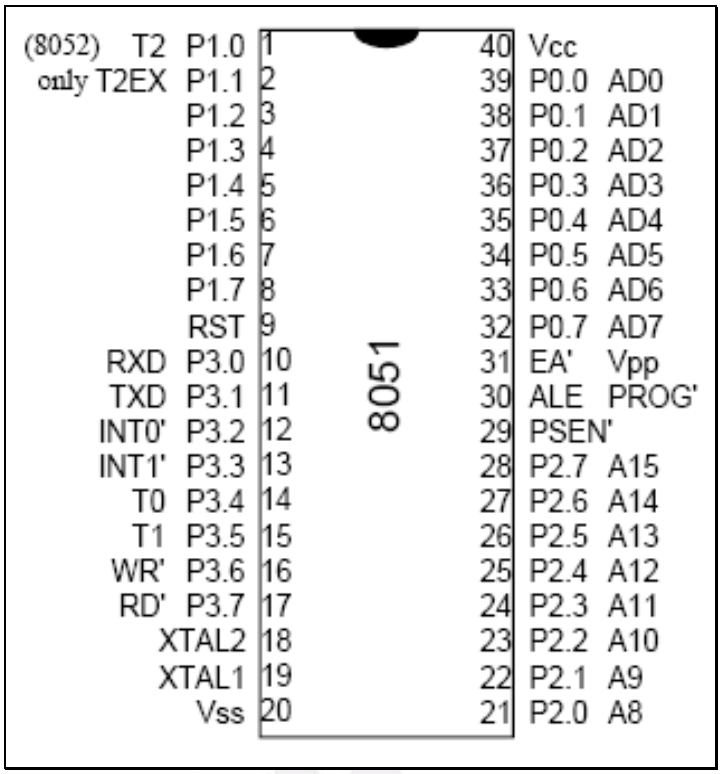
(10) six source/five-vector interrupt structure with two priority levels.

(11) on-chip clock oscillator.



▲ 圖 2.5 8051 核心運作圖

2.3.2 MCS-51 晶片腳位說明



▲ 圖 2.6 8051 IC 腳位圖

MCS-51 腳位功能說明:

- 腳位編號: Pin 1~8

接腳名稱: P1.0~P1.7

功能說明: P1 這 8 支腳是 8051 的 I/O 埠，內部已經具有提升電阻的 8 位元雙向 I/O 埠腳。 第 1 腳(P1.0)是(P1.0~P1.7) LSB，第 8 支腳(P1.7)是 MSB。 P1 的每支腳可推動 4 個 LS TTL。

- 腳位編號: Pin 9

接腳名稱: RESET

功能說明: 當這支腳由外部輸入 High(RESET) (+5V) 的信號時，8051 就被重置，8051 被重置後就從位址 0000H 開始執行程式。而特殊功能暫存器 (SFR) 裡的所有暫存器都會被設成初設狀態。

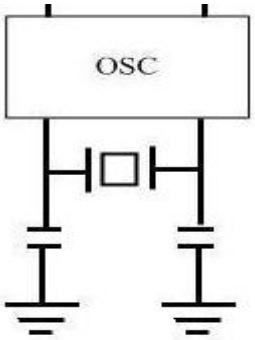
- 腳位編號: Pin 10~17

接腳名稱: P3.0~P3.7

功能說明: 第 10 腳(P3.0)為(P3.0~P3.7) LSB，第 17 腳(P3.7)為 MSB 。P3 裡的每支 I/O 腳除了可以當作單純的輸入/輸出使用外，也當作 8051 內部的某些週邊與外界溝通的 I/O 腳。例如 P3.0 和 P3.1 接腳的另外一個名稱為 RXD 和 TXD，8051 內部的 UART 被軟體啟動後，UART 會將串列資料從 TXD 腳輸出，而 UART 也接收由外部送進來的串列信號。INT0 和 INT1 是 8051 的兩個外部中斷輸入部。T0 是 Timer0 的外部脈波輸入腳。T1 是 Timer1 的外部脈波輸入腳。WR、RD，當您在 8051 的外部擴充資料記憶體(RAM)時，這兩條線是控制寫與讀的信號。P3 上的每一支 I/O 腳都可以作兩種用途。這 2 種用途，8051 會自動判斷這些腳位是何種用途，假設使用者要使用 UART 時，只要將 Pin 10 看成 RXD 將 Pin 11 看成 TXD 來使用即可。但是有一點要特別注意的是，這些腳位若是不當成 I/O 使用時，使用的那腳位內部栓鎖器的內容必須設為 1，其他的功能才會有作用。

- 腳位編號: Pin 18，19

接腳名稱: XTAL2(18) ， XTAL1(19)



腳位功能: 這兩支腳是 8051 內部時脈振盪器的輸入端，在這兩支腳上跨一個多少頻率的石英晶體(Crystal)，內部的振盪器就會產生與之相同的工作頻率供 8051 內部使用。8051 會根據這個速度工作。若未特別註明，這個振盪器的工作頻率是在 1MHz~12MHz 之間的任何一個。

- 腳位編號: Pin 20

接腳名稱: VSS

腳位說明: 8051 的接地腳，使用時此腳必須與系統的地線接在一起。

- 腳位編號: Pin 21~28

接腳名稱: P2.0~P2.7

腳位功能: P2.0 為 LSB，P2.7 為 MSB。若是想為 8051 擴充程式記憶體或資料記憶體時，P2 就變成 8051 的位址匯流排的高位元組(即 A8~A15)，與 P0 的低位元組(即 A0~A7)共同組成 16 位元的位址匯流排，對外存取外部記體，此時 P2 就不能當作 I/O 使用。P2 上的每支 I/O 腳可推動 4 個 LS TTL。

- 腳位編號: Pin 29

接腳名稱: PSEN=0(Program Store Enable)

腳位功能: 8051 用來讀取放在外部程式記憶體的指令時所用的讀取信號，通常這支腳是接到 EPROM (程式記憶體)的 OE 腳。8051 分別致能外部 EPROM 與 RAM(資料記憶體)，因此以 8051 的角度去看程式記憶體與資料記憶體是兩塊獨立的記憶體。

- 腳位編號: Pin 30

接腳名稱: ALE

腳位功能: 位址栓鎖致能接腳:(Address Latch Enable，簡稱 ALE) 。8051 可以發出信號，觸發外部的 8 位元栓鎖器，將 P0 上的位址匯流排信號(A0~A7) 鎖入栓鎖器中，以使 P0 具有資料/位址匯流排能力。

- 腳位編號: Pin 31

接腳名稱: EA=0(External Latch Enable)

腳位功能: 外部存取致能接腳：擴充或選擇內部記憶體。這是一支輸入腳，當 EA=0 時，8051 一律執行外部程式記憶裡的程式，因此 8051 內部的 4K 程式記憶體就沒有用了，如果要使用內部的程式記憶體時，一定要將 EA 接+5V。

- 腳位編號: Pin 32~39

接腳名稱: P0.0~P0.7

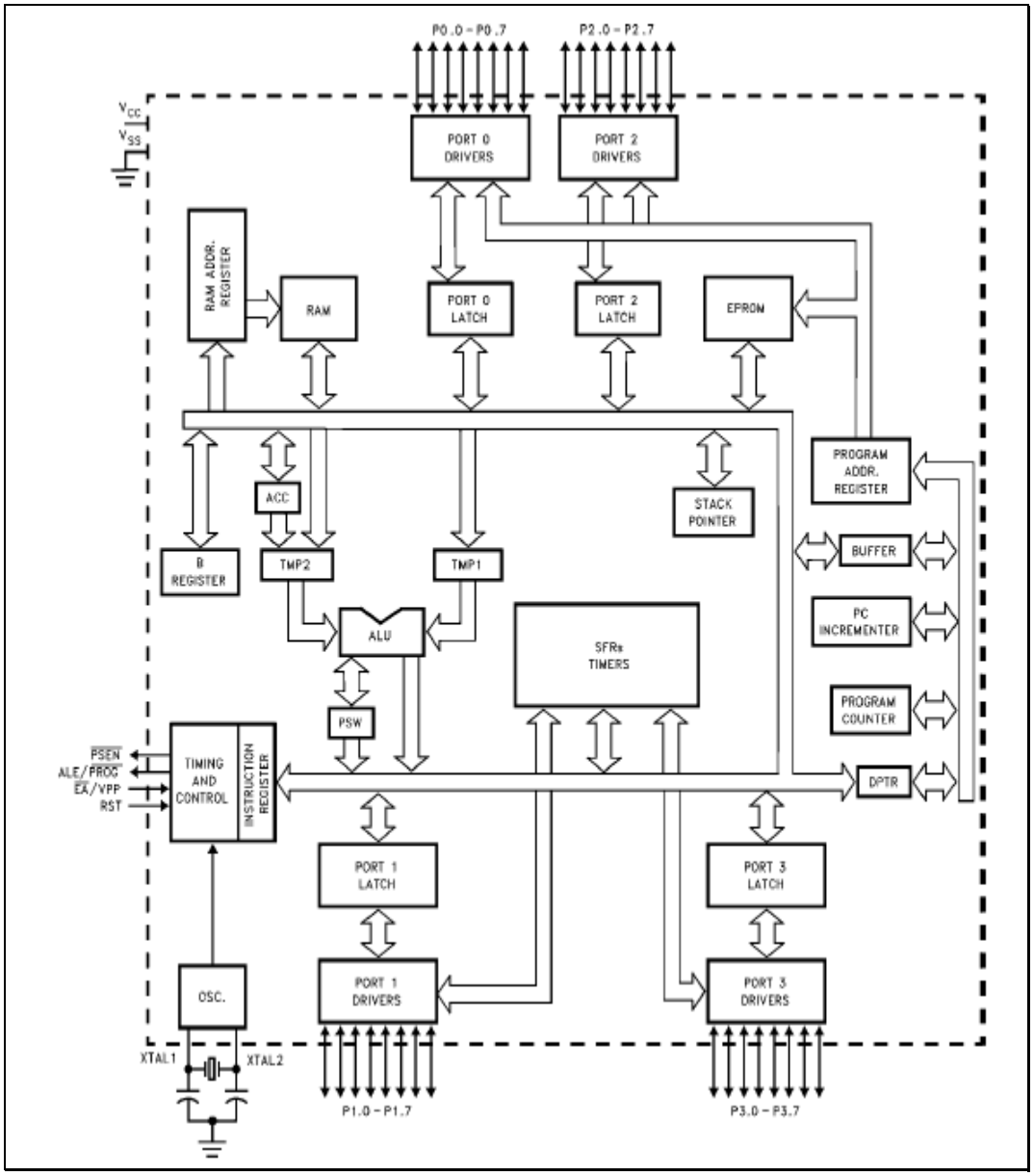
腳位功能: 是 8051 的 I/O 埠，稱為 P0 其中 P0.0 為 LSB，P0.7 為 MSB。P0 當作 I/O 使用時必須特別注意 P0 的輸出型態是 Open Drain，其他三個 I/O 埠(P1、P2、P3)內部有 pull-up 電阻電路，P0 則沒有，要自己外接。P0 除了當作 I/O 使用外，如果在 8051 的外面擴充程式記憶體或資料記憶體時，P0 就當作位址匯流排(A0~A7)和資料匯流排(D0~D7)多工使用。必須在外部加一個 8 位元栓鎖器將位址匯流排從 PC 上分離出來，這個 A0~A7 與 P2 所提供的 A8~A15 合成一個 16 位元的位址匯流排，因此 8051 可以在外部定址到 64K 的記憶體。

- 腳位編號: Pin 40

接腳名稱: VCC

腳位功能: 8051 的電源輸入端，電源規格是+5V± 10%。

2.3.3 MCS-51 單晶片系統架構

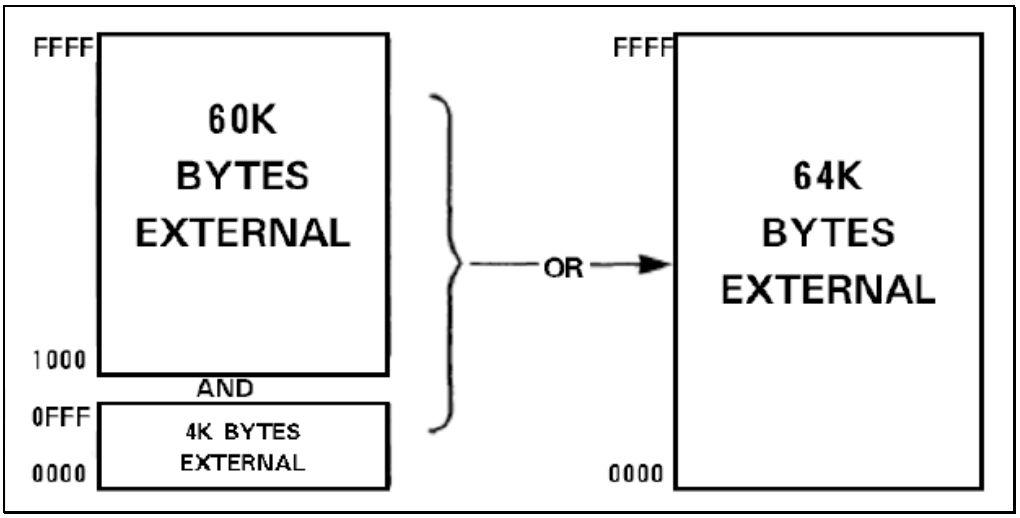


▲ 圖 2.7 8XC5X Block Diagram

2.3.4 MCS-51 記憶體結構

- **程式記憶體(Program memory)**

8051 具有 4K Bytes 的內部程式記憶體，除此之外也可在外部再擴 60K Bytes EPROM ，可參考圖 2.8。程式記憶體是存放 8051 所執行的程式碼地方，CPU 會主動到這裡來讀取要執行的指令碼，8051 要讀取程式記體時需激發信號 PSEN，因此這塊記憶體的資料只能被 CPU 讀取，而無法寫入資料。

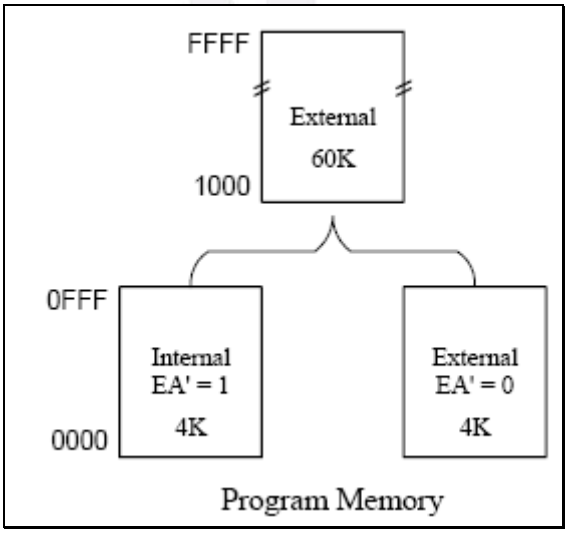


▲ 圖 2.8 MCS51 程式記憶體結構

MCS-51辨別從內部還是外部程式記憶體讀取指令的方法：

(1) 當 EA 接為高電位，則自內部 0000H 開始讀取程式碼；當超過 0FFFH(4KByte) 時，就跳至外部程式記憶體，讀取程式碼，參考圖 2.9。

(2) 當EA爲Low電位，就一定自外部記憶體0000H，開始讀取程式記憶體。



▲ 圖 2.9 EA 電位辨別讀取內外部記憶體

8051 到外部讀取一個指令碼時，P0 和 P2 這兩個 I/O 埠就變成外部 EPROM 時所需的匯流排，其中 P0 當作位址匯流排和資料匯流排多工使用，當 ALE 接腳輸出為 High 時，此刻 P0 上所輸出的是位址信號(A0~A7)。因此外部的位址栓鎖電路必須在此刻將 P0 上的位址信號捕捉起來，當 ALE 降為 LOW，且 PSEN 為 LOW 時，P0 就變成資料匯流(D0~D7)，8051 會在 PSEN 的輸出狀態由 LOW 轉態成 High 時讀入 P0 上的資料且將它解釋成指令碼；P2 在 8051 讀取外部程記憶體時會固定輸出位址匯流排的高位元組(A8~A15)。

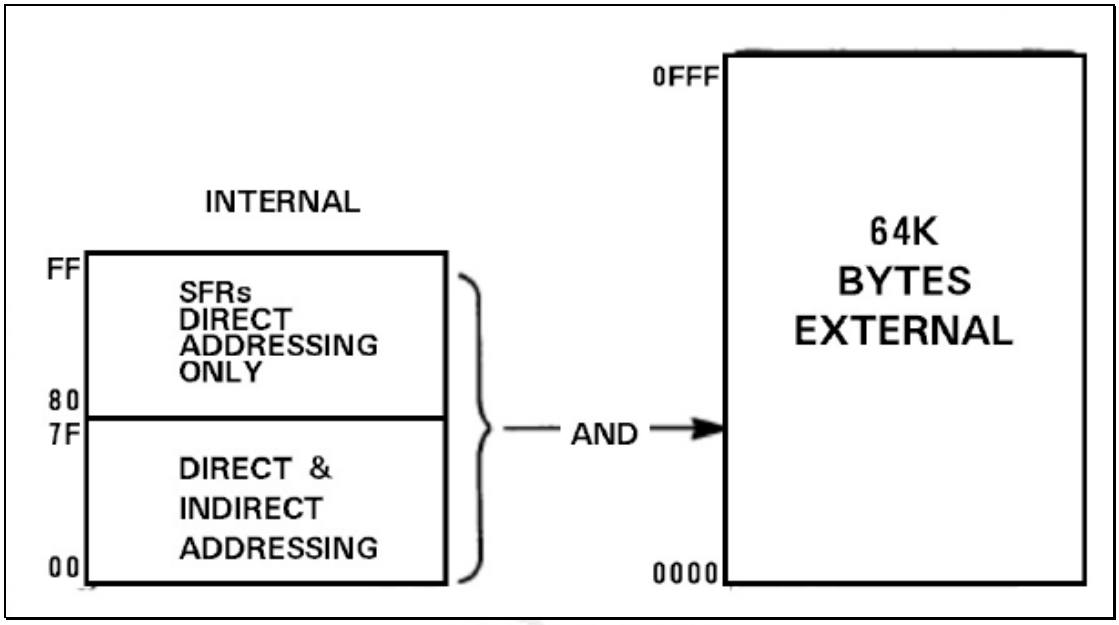
- **資料記憶體(Data memory)**

A．外部資料記憶體:

8051 允許你在外部擴充 64K byte 資料記憶體。而定址 64K 資料記憶體空間需要 16 條位址線和 8 條資料線，而這 16 條位址線跟 8 條資料線與程式記憶體使用相同的匯流排，然後 8051 以控制匯流排來區分這兩塊不同的記憶體。如讀取外部程式記憶體時使用 PSEN，而讀寫外部資料記憶體使用 RD 和 WR 信號。

B. 內部資料記憶體:

8051 內部有一塊 256 個 Byte 的位址空間，這塊空間是存放資料記憶(RAM)和特殊功能暫存器(SFR)的地方，可參考圖 2.10。



▲ 圖 2.10 MCS-51 資料記憶體結構

8051 系列單晶片具有 128 Bytes 的內部資料記憶體，其中位址編號為 00H~7FH，可用直接定址法來存取資料。依單晶片的特性又可將這些內部資料記憶體分成三個不同的部分：完整內部資料記憶體結構可參考圖 2.11。

The first 256 bytes of internal data memory

F8									Special function registers (SFR) (128 bytes)
F0	B								
E8									
E0	ACC								
D8									
D0	PSW								
C8	T2CON		RCAP2L	RCAP2H	TL2	TH2			
C0									
B8	IP								
B0	P3								
A8	IE								
A0	P2								
98	SCON	SBUF							
90	P1								
88	TCON	TMOD	TL0	TL1	TH0	TH1			
80	P0	SP	DPL	DPH				PCON	
78									Scratch pad area (80 bytes)
70									
68									
60									
58									
50									
48									
40									
38									
30									
28	Can be addressed as 16 bytes or 128 individual bits. Byte addresses are 20H to 2F. Bit addresses are 00H to 7F.								
20									
18	Reg 0	Reg 1	Reg 2	Reg 3	Reg 4	Reg 5	Reg 6	Reg 7	Bank 3
10	Reg 0	Reg 1	Reg 2	Reg 3	Reg 4	Reg 5	Reg 6	Reg 7	Bank 2
8	Reg 0	Reg 1	Reg 2	Reg 3	Reg 4	Reg 5	Reg 6	Reg 7	Bank 1
0	Reg 0	Reg 1	Reg 2	Reg 3	Reg 4	Reg 5	Reg 6	Reg 7	Bank 0

▲ 圖 2.11 內部資料記憶體方塊圖

1.暫存器庫(Register Banks)：

位址 00H~1FH(共 32Bytes)，可分為 4 個暫存器庫(0~3)，每一個暫存器庫各有 R0~R7 共 8 個暫存器。當作暫存器用時，使用者只能使用其中一組暫存器庫來使用，稱之為工作暫存器庫(Working Register Bank)，這是由單晶片內部 PSW 暫存器之 RS0 及 RS1 位元來指定要使用哪一個暫存器庫。

2.可位元定址(Bit-addressable)區：

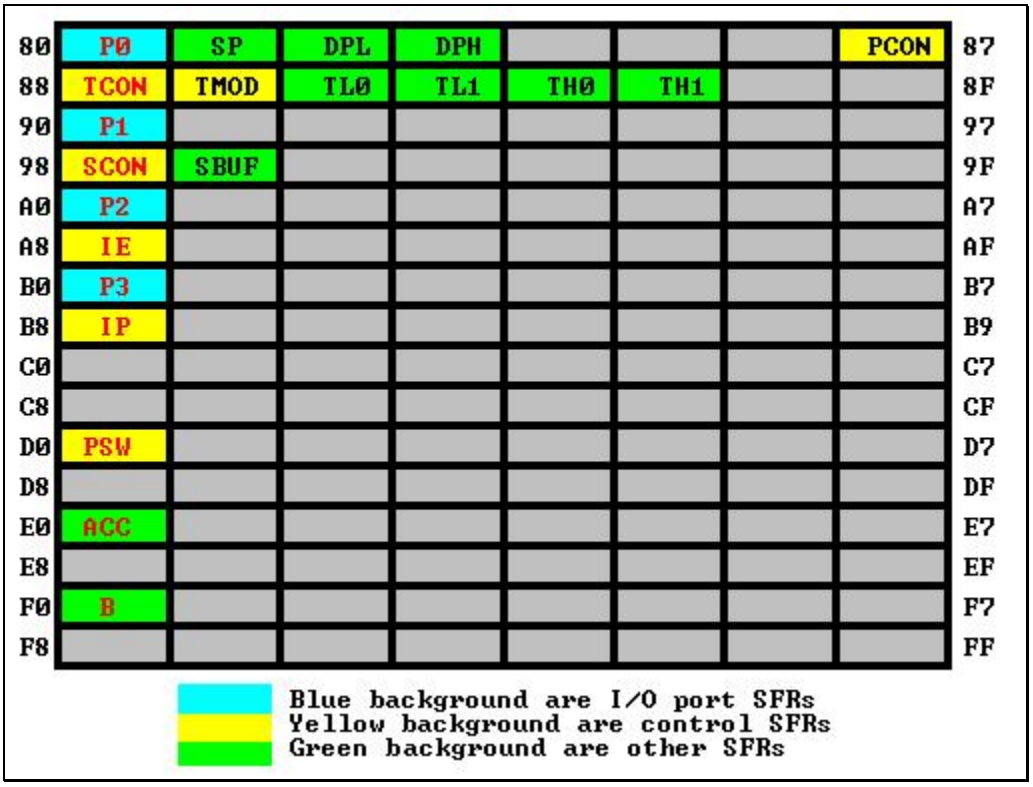
位址 20H~2FH(共 16 Bytes)，這 16 Bytes 記憶體中的每一個位元皆可單獨設定為 0 或 1，因此共有 128 個位元可單獨使用，其位元位址的編號為 00H~7FH(0~127)。

3.一般用途區：

在 8051 中，此位址空間(30H~7FH)並未加以定義，由使用者自由使用，可以存放程式變數用，然而程式執行時，可設定為堆疊區(執行 CALL、PUSH、POP 指令會用到)，堆疊區的大小由使用者自行設定。

2.3.5 特殊功能暫存器(SFR)

位址為 80H~FFH，是一塊 128 Bytes 可直接定址的記憶體區。它是用來存放週邊元件控制、狀態及資料的暫存器，稱之為特殊功能暫存器(SFR)。除了累積器(ACC)、暫存器 B、程式狀態字組(PSW)、堆疊(SP)以及資料指標器(DPTR)外，其餘特殊功能暫存器均與 8051 所擁有的周邊介面有關。以下說明 SFR 中各暫存器的功能與用途，可參考圖 2.12。



▲ 圖 2.12 特殊功能暫存器結構

- **累加器(Accumulator，ACC) :**

累加器(Accumulator)簡稱 ACC 或 A，在 MCS-51 中做算數/邏輯運算時，都必須透過此暫存器才可運算。

- **B暫存器：**

B 暫存器在乘法運算時，用來存放乘數數值，及相乘後乘積的較高位元組;在除法運算時，用來存放除數，及相除後的餘數，如果不用在乘法運算，可以當成一個通用暫存器。

- **PSW暫存器：**

PSW 全名為(Program Status Word) 即程式狀態字組，用來記錄 CPU 執行的狀態，內含 8 位元，用途說明參見下表。

位元	7	6	5	4	3	2	1	0
PSW	CY	AC	F0	RS1	RS0	OV	---	P

▲ 表 2.4 PSW 暫存器位元名稱

♦ P(PSW.0)；同位元旗標

P=1 則 ACC 暫存器 1 的個數為奇數；

P=0 則 ACC 暫存器 1 的個數為偶數。

♦ ---(PSW.1)；暫時保留。

♦ OV(PSW.2)；溢位旗標，當 OV=1，表示運算時有溢位產生。

♦ RS0，RS1(PSW.3，PSW.4)；Register Bank 選擇位元。

RS1	RS0	使用的暫存器庫	記憶體位址
0	0	Register Bank 0	00H~07H
0	1	Register Bank 1	08H~0FH
1	0	Register Bank 2	10H~17H
1	1	Register Bank 3	18H~1FH

▲ 表 2.5 RS0，RS1 選擇位元功能說明

♦ F0(PSW.5) ；

一般用途旗標，由使用者自行設定。

♦ AC(PSW.6)；輔助進位旗標

當運算時，較低的 4 位元有進位或借位時，AC 會被設定為 1，常在 BCD 碼運算時用到。

♦ CY(PSW.7)；進位旗標

CY=1 時，表示運算時有進位或是借位產生。

- **堆疊指標器(Stack Pointer) :**

用來指明堆疊空間的起始點，CPU 重置時 SP 內含值為 07H，表示堆疊空間自 08H 開始，資料擺入堆疊時，SP 會先加 1 再放入資料，取出時會先取出資料 SP 再減 1。

- **DPTR暫存器(Data Pointer) :**

DPTR 是一個 16 位元的暫存器，它是由兩個 8 位元的暫存器 DPH(高位元組)及 DPL(低位元組)所組成。 DPTR 的最主要用途是用來指向程式或資料記憶體的每一個位址，以便存取程式碼或資料。當 DPTR 指向程式記憶體時，我們可以用 MOVC 指令來讀取程式記憶體中的資料，當 DPTR 指向資料記憶體時，我們可用 MOVX 指令來存放或讀取資料記憶體中的資料。

- **P0，P1，P2，P3埠暫存器 :**

這四個埠暫存器可存放 8051 單晶片的 4 個 I/O 埠的輸出閂鎖(Latch)，主要是存放並保持 I/O 的輸出資料。

- **IE暫存器(Interrupt Enable) :**

中斷致能暫存器，用來設定各個中斷是為致能或禁能(Enable/Disable) 。

位元	7	6	5	4	3	2	1	0
IE	EA	--	ET2	ES	ET1	EX1	ET0	EX0

▲ 表 2.6 IE 暫存器位元名稱

位置	符號	功能
IE.0	EX0	第 0 個外部中斷的致能/禁能位元
IE.1	ET0	第 0 個計時中斷的致能/禁能位元
IE.2	EX1	第 1 個外部中斷的致能/禁能位元
IE.3	ET1	第 1 個計時中斷的致能/禁能位元
IE.4	ES	串列輸入/輸出的致能/禁能位元
IE.5	ET2	8052 才有的，對第 2 個計時中斷
IE.6	--	暫時保留
IE.7	EA	全體的致能/禁能位元 EA=0，所有中斷被禁能。EA=1，中斷才可配合其他各位元致能。

▲ 表 2.7 IE 位元功能說明

- **中斷優先權暫存器(Interrupt Priority，IP) :**

每一個 IP 暫存器位元可用來控制各中斷的優先權階層，當設定為 1 時，表示享有較高的中斷優先權，而設定為 0 時其優先權較低。

位元	7	6	5	4	3	2	1	0
IP	---	---	PT2	PS	PT1	PX1	PT0	PX0

▲ 表 2.8 IP 暫存器位元名稱

位置	符號	功能
IP.0	PX0	設定第 0 個外部中斷源優先權
IP.1	PT0	設定第 0 個計時中斷優先權
IP.2	PX1	設定第 1 個外部中斷源優先權
IP.3	PT1	設定第 1 個計時中斷優先權
IP.4	PS	設定串列埠的中斷優先權
IP.5	PT2	設定 8052 的第 2 個計時中斷優先權
IP.6	---	保留
IP.7	---	保留

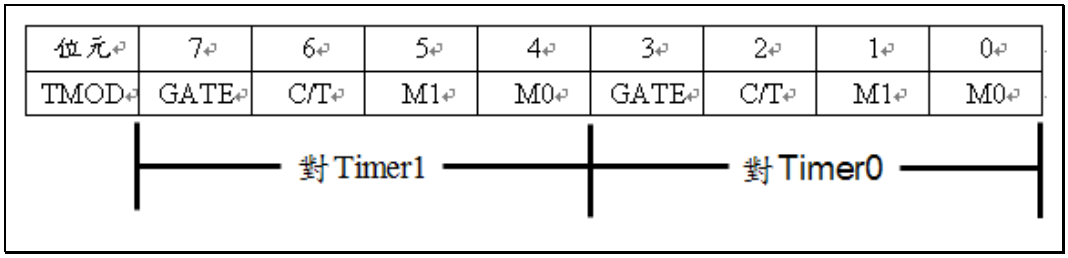
▲ 表 2.9 IP 位元功能說明

- **TH0~TH2，TL0~TL2計時/計數暫存器:**

這 3 組 16 位元的暫存器是分別用來儲存計時器/計數器的計時/計數值。TH0、TH1、TH2 為高位元組，TL0、TL1、TL2 為低位元組。TH0 及 TL0 對應於計時器/計數器 0，TH1 及 TL1 對應於計時器/計數器 1，TH2 及 TL2 對應於計時器/計數器 2(8052 系列)，參考圖 2.13。

- **計時器模式控制暫存器(TMOD):**

全名為 Timer/Counter Mode Control Register，可利用此暫存器，設定第 0 個和第 1 個計時/計數器工作型態和模式。可參考表 2.10。



▲ 圖 2.13 TMOD 暫存器位元名稱

位置	符號			
第 3 第 7	GATE	計時器動作閘控位元:當GATE=1須INT0和TR0同時爲1，計時器才動作；當GATE=0只需TR0爲1，就可動作。		
第 2 第 6	C/T	C/T=0 時，就當計時器使用 C/T=1 時，就當計數器使用		
第 1 第 5	M1	模式選擇位元		
第 0 第 4	M0	M1	M0	
		0	0	當作 13 位元計時/計數器
		0	1	當做 16 位元計時/計數器
		1	0	當作 8 位元計時/計數器
		1	1	當作 2 個 8 位元計時/計數器

▲ 表 2.10 TMOD 位元功能說明

- **計時器控制暫存器(Timer Control ，TCON):**

內部位元用來設定計時/計數器的啓動，及記錄計時溢位和外部中斷動作型態。

位元	7	6	5	4	3	2	1	0
TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

▲ 表 2.11 TCON 暫存器位元名稱

位置	符號	功能
TCON.7	TF1	計時器1 溢位旗號，當計時溢位時，由硬體設定爲1，在執行過相對的中斷服務常式後則自動清除爲0。
TCON.6	TR1	計時器1 啓動控制位元，可以由軟體來設定或清除。
TCON.5	TF0	計時器0 溢位旗號，當計時溢位時，由硬體設定爲1，在執行過相對的中斷服務常式後則自動清除爲0。
TCON.4	TR0	計時器0 啓動控制位元，可以由軟體來設定或清除。

TCON.3	IE1	外部中斷1 動作旗號，當外部中斷被偵測出來時，硬體自動設定此位元，在執行過中斷服務常式後，則消除爲0。
TCON.2	IT1	外部中斷1 動作型態選擇，當IT1=1 時，中斷型態爲負緣觸發，當IT1=0 時，中斷型態則爲低準位觸發。
TCON.1	IE0	外部中斷0 動作旗號，當外部中斷被偵測出來時，硬體自動設定此位元，在執行過中斷服務常式後，則消除爲0。
TCON.0	IT0	外部中斷0 動作型態選擇，當IT1=1 時，中斷型態爲負緣觸發，當IT1=0 時，中斷型態則爲低準位觸發。

▲ 表 2.12 TCON 位元說明

- **串列埠控制暫存器(Serial Control Register，SCON):**

內部含串列埠模式選擇位元和串列埠工作時所須要旗號。

位元	7	6	5	4	3	2	1	0
SCON	SM0	SM1	SM2	REN	TB8	RB8	TI	RI

▲ 表 2.13 SCON 暫存器位元名稱

位置	符號	功能
7 6	SM0 SM1	串列工作模式選擇位元。 SM1 SM0 模式 功能 鮑率 0 0 0 移位暫存器 工作頻率/12 0 1 1 8 位元 UART 可以改變 1 0 2 9 位元 UART 工作頻率/32 或/64 1 1 3 9 位元 UART 可以改變
5	SM2	在串列傳輸動作模式2 或模式3 時，作多處理機控制功能用。
4	REN	串列介面接收位元，當REN=1 時表示接收致能，開始接收。
3	TB8	在模式2 或3 時，所送出的第9 個資料位元，可以由軟體指令來做控制設定或清除。
2	RB8	在模式2 或3 時，所接收到的第9 個資料位元，存放在此位元中。
1	TI	串列資料傳送中斷旗號，在工作模式0 時，送出8 個資料位元後，TI 設爲1，而在其他模式時，在送出停止位元時，TI 也會被設爲1；此位元必須由軟體來清除。
0	RI	串列資料接收中斷旗號，在工作模式0 時，收到第8 個串列輸

		入資料位元後，RI 會設爲1，在其他模式時，收到停止位元的一半時，硬體會自動將此位元設爲1。此位元必須由軟體來清除。

▲ 表 2.14 SCON 位元說明

- **串列資料緩衝暫存器(Serial Data Buffer，SBUF):**

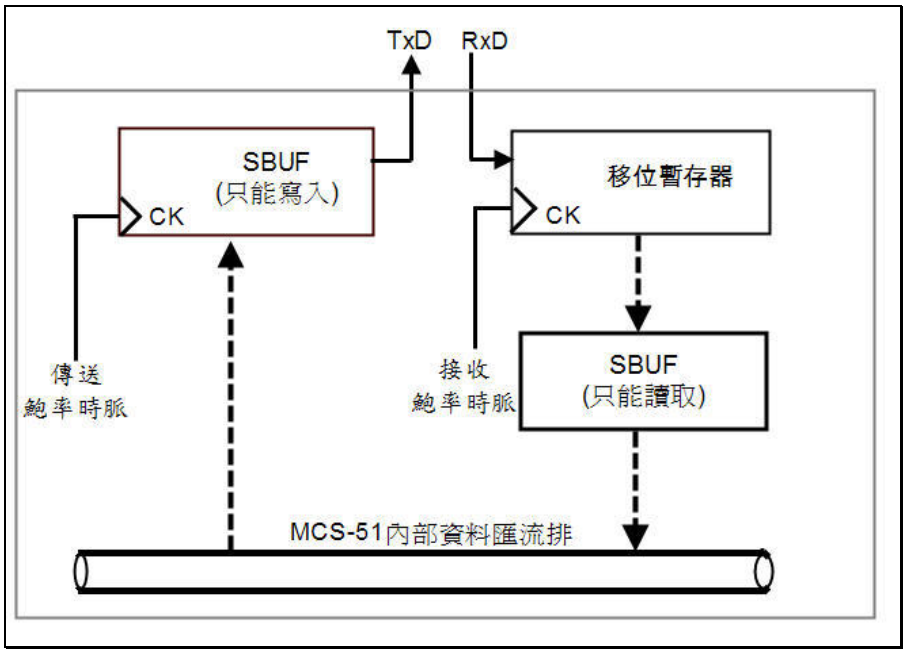
8051 單晶片的串列埠是全雙工的，故實際上 SBUF 暫存器分開為兩個不同的暫存器，一個是當作 UART 傳送資料的緩衝區，另一個是當作 UART 接收資料的緩衝區。若將資料寫到 SBUF 時，就會將資料放入傳送緩衝區，UART 就會將這個資料轉成串列資料透過 TXD 傳出去。若去讀 SBUF，就會讀到接收緩衝區的資料。

2.4 MCS-51 串列傳輸介面

MCS-51 的串列通信埠為一個全多工的串列通信通道，即它可以同時傳送與接收資料。同時，它的接收部分具有緩衝器可以接收 2 Bytes 的資料，外部電路可以透過 TxD (P3.1) 與 RxD (P3.0)兩個 I/O 埠位元線，存取串列通信埠的資料。

2.4.1 MCS-51 串列通信埠基本結構

MCS-51 的串列通信埠的基本結構如下圖，輸出端主要由一個只能寫入的並行輸入串列輸出的移位暫存器稱為 SBUF(serial port buffer)組成；輸入端主要由一個串列輸入並型輸出的移位暫存器與一個只能讀取的門閂電路組成，門閂電路也稱為 SBUF 組成。SBUF 的位址為 99H。欲傳送資料時，寫入資料於 SBUF 中；讀取 SBUF 時，相當於讀取接收端的資料。



▲ 圖 2.14 MCS-51 串列通信埠邏輯方塊

2.4.2 MCS-51 串列傳輸工作模式

MCS-51 串列傳輸埠的操作，首先必須透過軟體指令設定 SCON 的值，已設定串列傳輸埠的工作模式，之後再將所要傳送的資料存入 SBUF 中或啟動接收動作，再從 SBUF 中讀入接收到的資料，至於此通訊介面與 CPU 之間仍然透過中斷的方式來完成。8051 內部含有一組全雙工的串列介面，其中提供了 4 種操作模式，由設計者來自由使用，以下為四種操作模式的說明。

1.工作模式 0：

此模式基本上是做串列傳輸 I/O 控制，而非真正的串列通訊應用，工作於此模式時，由 TXD 接腳送出移位同步脈波，由 RXD 接腳送出或接收串列資料。而串列資料的形式，不具有起始及結束位元，純粹為 8 位元資料，至於同步脈波的寬度是固定的，為系統工作振盪週期的 1/12，等於是 8051 一個機械週期的時間。

2.工作模式 1：

此為經常使用的串列傳輸工作模式，串列資料位元由 TXD 接腳傳送出去，由 RXD 接腳將對方送來的串列資料接收進來。而資料格式共有 10 位元，包括前方的起始位元，8 位元串列資料位元及最後的停止位元。至於傳輸率(鮑率)快慢則由計時器 1 來規劃，只要將不同的計時初值載入計時器中，可以做不同的鮑率值設定。

3.工作模式 2：

此傳輸模式與模式 1 類似，不過資料一共送出了 11 個位元，包括 1 個起始位元，8 個資料位元及 1 個可程式設定的第 9 個資料位元和停止位元。此第 9 個可程式設定的資料位元是位於特殊目的暫存器 SCON 中的位元 3 中(TB8)，8051 可以利用此一特殊位元來做多處理機的系統連線控制用。至於傳送速度只有 2 種，分別為系統工作時脈頻率的 1/32 或 1/64。

4.工作模式 3：

模式 3 的傳輸方式與模式 2 很類似，同樣是傳送 11 個位元串列資料，差別在於其傳輸速度是可變的，如同模式 1 一樣是由 8051 內部計時器 1 所控制，可由程式設計者規劃。

2.4.3 MCS-51 串列傳輸鮑率設定

1.工作模式 0 鮑率設定：

串列模式 0 的鮑率固定為振盪頻率的 1/12，是不可改變的。鮑率 = fosc/12。

2.工作模式 2 鮑率設定：

在模式 2 操作下鮑率= [(2^{SMOD}) / 64] * (工作振盪頻率)］，其中 SMOD 為 SFR 中的 PCON 位元 7。

鮑率=(工作頻率)/32 (當 SMOD=1 時)

鮑率=(工作頻率)/64 (當 SMOD=0 時)

3.工作模式 1 與 3 鮑率設定：

模式 1 與模式 3 的鮑率，可以任意改變。其鮑率設定由 TIMER 1 來控制，TIMER 1 的工作模式有 4 種，一般會設定 TIMER 1 的模式 2 有自動載入的計時器模式。在模式 2 的計時下，使用的計時器暫存器為 TL1，而 TH1 則是在做自動載入計時值的設定，而鮑率的計算公式為：

鮑率= [2^{SMOD} / 32] * {(工作振盪頻率) / 12 * [256 - TH1]}

設計時我們是先定出鮑率再求 TH1 之值，將上式加以整理可得：

TH1 = 256 - [2^{SMOD} * (工作振盪頻率)] / (384 * 鮑率)

表 2.15 表所示為常用的鮑率與其他因素的關係

Baud Rate	Fosc	SMOD	TIMER 1		
			C/T	Mode	Reload Value
Mode 0 Max: 1 MHZ	12 MHZ	X	X	X	X
Mode 2 Max: 375K	12 MHZ	1	X	X	X
Modes1，3: 62.5K	12 MHZ	1	0	2	FFH
19.2K	11.059 MHZ	1	0	2	FDH
9.6K	11.059 MHZ	0	0	2	FDH
4.8K	11.059 MHZ	0	0	2	FAH
2.4K	11.059 MHZ	0	0	2	F4H
1.2K	11.059 MHZ	0	0	2	E8H
137.5	11.986 MHZ	0	0	2	1DH
110	6 MHZ	0	0	2	72H
110	12 MHZ	0	0	1	FEEBH

▲ 表 2.15 鮑率與震盪頻率關係

2.5 步進馬達（Stepper Motor)簡介

步進馬達(Stepper Motor)可藉開環路控制法低成本進行變速控制、定位控制， 而且不需要複雜的閉迴路回授控制。具備高速化、靜音化、省能量化，因此被廣泛應用。例如: 步進馬達優秀的定位精度，適合 FDD 的磁頭驅動用。而步進馬達在開環路的定速運轉適用於列表機的送紙，滑箱的驅動。

2.5.1 步進馬達的特色

1. 步進馬達必須加驅動電路才能轉動，驅動電路的信號輸出端，必須輸入脈波信號，轉子是以一定的角度（稱為步角）轉動。故若加入連續脈波時，則轉子旋轉的角度與脈波頻率成正比。

2. 步進馬達若不加脈衝信號時，步進馬達有很高的保持轉矩(Holding Torque)，可保

持在停止的位置，不需使用煞車迴路就不會自由轉動。因此，步進馬達具有瞬間啟動與急速停止之優越特性。

3. 步進角極小，如每步0.9°或1.8°等，而且沒有累積誤差，可做精密的角度運轉控制。

4. 改變線圈激磁的順序，可以較輕易的改變馬達的轉動方向。

5. 可靠性高，整個系統的價格低。

2.5.2 步進馬達的分類

步進馬達依定子線圈的相數不同可分成二相、三相、四相及五相式，小型步進馬達以四相式較為普遍。當送入一個脈衝電流至步進馬達，可在相對應處停止轉動，這種走一步即停住而得到的角度稱為基本步進角。步進角會因激磁方式不同而有所不同。

基本步進角計算公式如下:

STEP ANGLE

The step angle, the number of degrees a rotor will turn per step, is calculated as follows:

$$\text{Step Angle}\,(\Theta_S) = \frac{360°}{S}$$

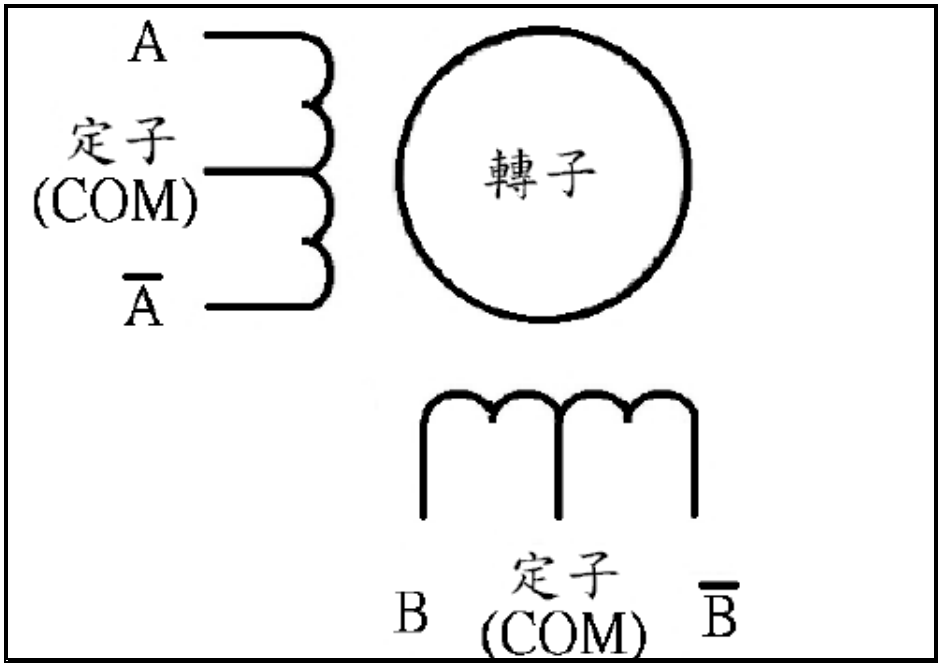
$$S = mN_r$$

$$m = number\ of\ phases$$

$$N_r = number\ of\ rotor\ teeth$$

例如本專題所使用的步進馬達

相數=4 ， 轉子齒數=50 ，所以基本步進角為 **360°/(4× 50)=1.8°**



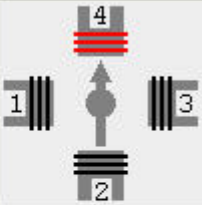
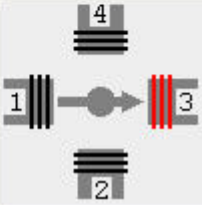
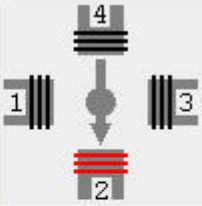
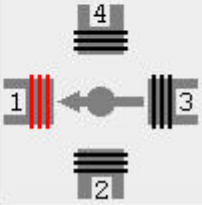
▲ 圖 2.15 四相步進馬達內部簡化結構

2.5.3 步進馬達的激磁方式

所謂激磁即是令步進馬達的線圈通過電流，以四相步進馬達而言，其定子線圈共有四個相，分別為 A、A + B 、B+ ，參考圖 2.15。而步進馬達的激磁方式有下列三種方式：

- **一相激磁(Single-Coil Excitation)：**

當一個脈波信號輸入後，四組線圈相位中只有一組相位激磁，也就是電流只通過其中一組線圈，所以每次可移動一個基本步進角，這種方式轉動時力矩較小、振動較大等缺點。以下圖示說明皆為正轉，若要反轉，可改變激磁順序。一相激磁可參考圖 2.16。

Step	Coil 4	Coil 3	Coil 2	Coil 1	
a.1	on	off	off	off	
a.2	off	on	off	off	
a.3	off	off	on	off	
a.4	off	off	off	on	

▲ 圖 2.16 一相激磁說明圖

- **二相激磁(Two-Coil Excitation)：**

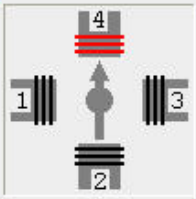
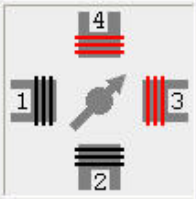
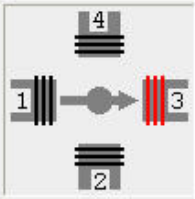
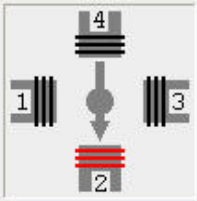
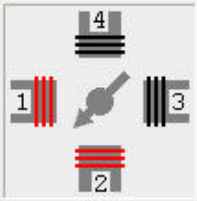
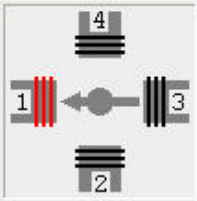
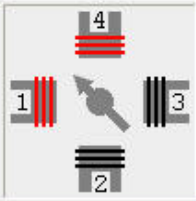
當一個脈波信號輸入後，有兩組相位被激磁，也就是電流只通過其中二組線圈，使步進馬達每次一動一個步進角。因為同時有二相被激磁，固產生的力矩較大，振動較小，可參考圖 2.16。

Step	Coil 4	Coil 3	Coil 2	Coil 1	
b.1	on	on	off	off	
b.2	off	on	on	off	
b.3	off	off	on	on	
b.4	on	off	off	on	

▲ 圖 2.17 二相激磁說明圖

- **一、二相激磁：**

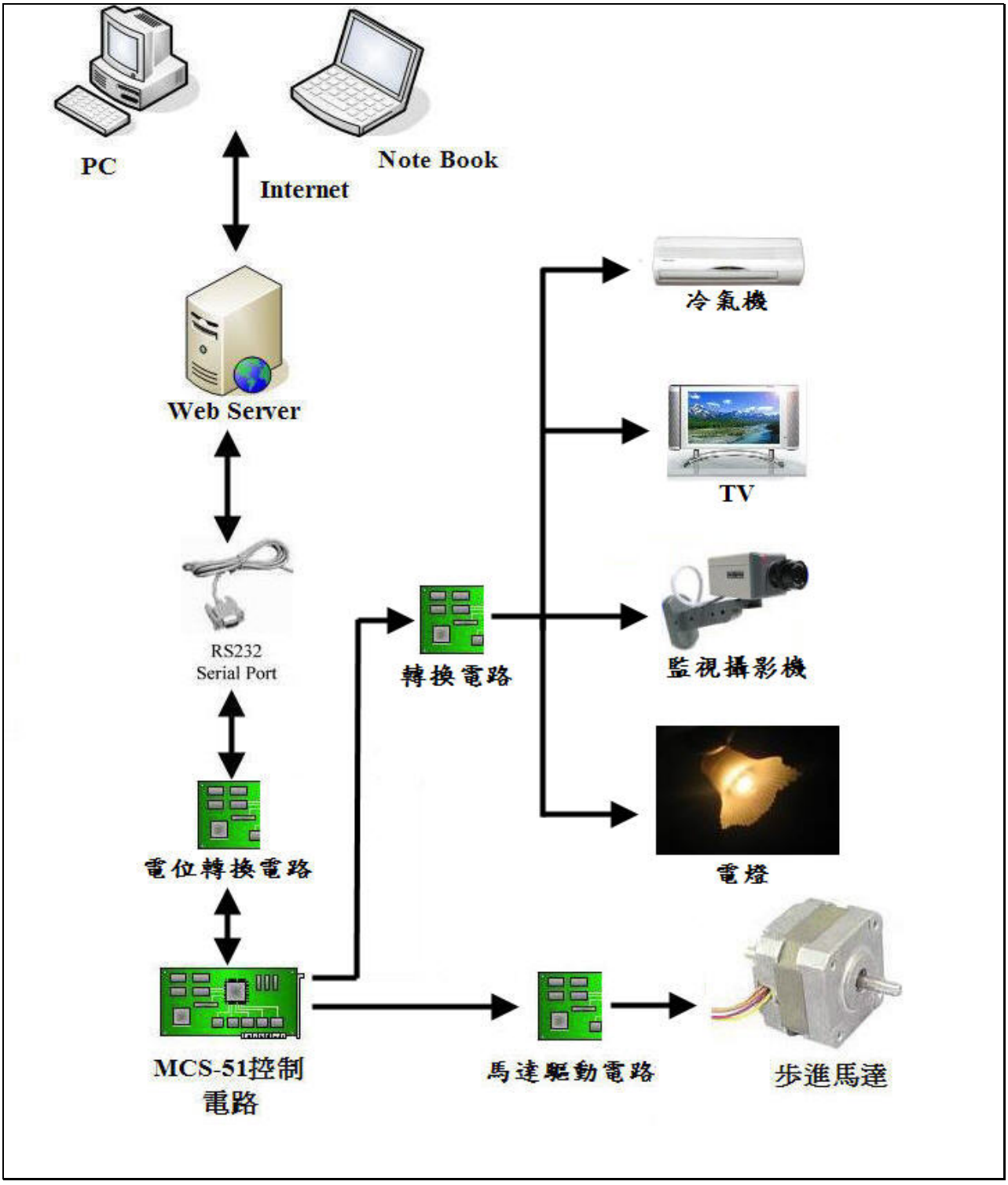
將上述兩種激磁方式合併，其中 A 相和 B 相採交互激磁的順序，故每次步進馬達只移動半個基本步進角，因此解析度提高一倍，這種方式旋轉時較平滑，且振動的程度最低，詳細設定可參考圖 2.18。

Step	Coil 4	Coil 3	Coil 2	Coil 1	
a.1	on	off	off	off	
b.1	on	on	off	off	
a.2	off	on	off	off	
b.2	off	on	on	off	
a.3	off	off	on	off	
b.3	off	off	on	on	
a.4	off	off	off	on	
b.4	on	off	off	on	

▲ 圖 2.18 一、二相激磁說明圖

第三章　程式與硬體實做測試流程敘述

3.1 遠端控制系統架構



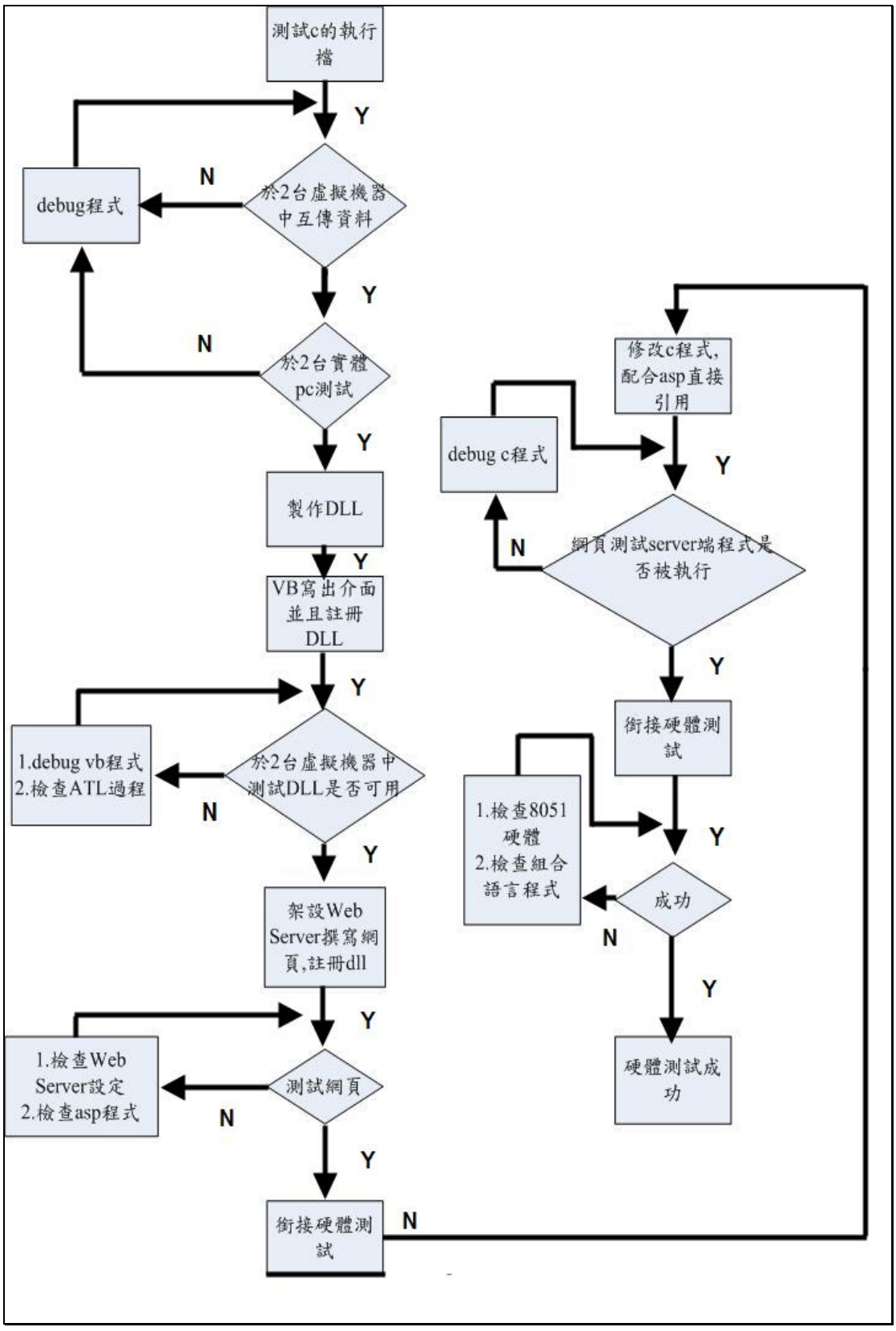
▲　圖 3.1　遠端控制系統架構

- 遠端控制系統架構說明

本專題遠端控制系統架構可參考圖 3.1，使用者可使用 PC、Notebook、PDA 等設備透過 Internet 連線登入 Web Server，經帳號與密碼驗證無誤即可進入網頁控制介面。而此使用者介面使是使用 ASP 網頁語言製作，而 ASP 本身並無法直接控制硬體，而是利用呼叫 Web Server 端硬體控制應用程式的方式。此硬體控制之程式則是利用 Windows API 來達到控制 serial port 的目的，使網頁的控制指令能透過 serial port 傳送到 MCS-51 控制電路；而就在 Web Server 端透過 serial port 傳送資料到 MCS-51 控制電路之間，由於兩者的電壓準位定義不同，因此需要透過以 MAX-232 IC 為核心的電位轉換電路的幫助，才不至於發生資料讀寫錯誤的問題。而 MSC-51 控制電路所扮演的的角色則是將 SBUF 接收暫存器所接收的控制指令進行判別，並且執行該指令之 I/O 動作與相關設定。而硬體控制電路之所以選擇 MCS-51 單晶片為核心，主要是因為 MCS-51 符合經濟效益，而且 I/O 腳位充足，可銜接多種周邊硬體，做為硬體控制電路核心非常合適。

至此，已經完成遠端控制硬體所需的開關插座，中間只需特定的轉換電路或是驅動電路即可連結許多欲控制的硬體設備。例如，欲控制步進馬達只需透過馬達驅動電路與 MCS-51 控制電路連結，修改 MCS-51 控制程式對馬達輸入連續脈衝信號造成激磁效果，即可輕易控制步進馬達正反轉、轉速與定位的控制。相同的，欲控制其他家電設備例如，冷氣機開關的時間設定與溫度控制、電視開關、監視攝影機鏡頭拍攝角度調整與電燈的開關設定等等，皆只需要透過相對應之轉換電路或是驅動電路設備，修改 MCS-51 控制程式及 I/O 動作，即可完成遠端控制。

3.2 程式流程簡介與 COM Port 設定



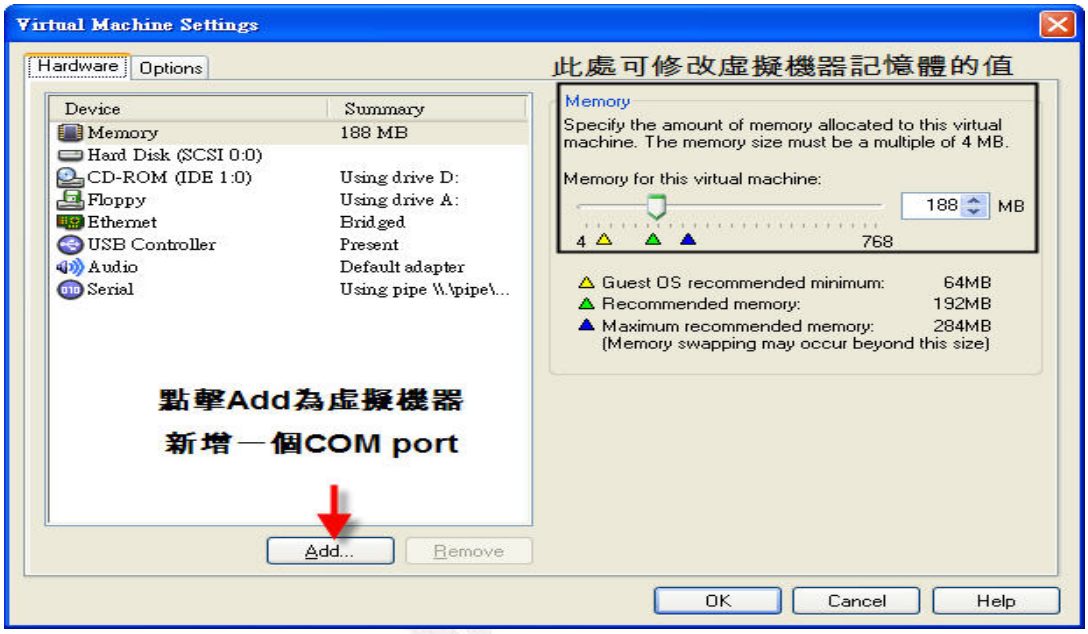
▲ 圖 3.2 整體程式流程

Windows 2000 虛擬機器 COM Port 的連接設定

新增 COM Port :

1. Command底下的Edit virtual machine settings 出現底下視窗。

2. 點擊Add 新增serial port 在此選擇COM1->use physical serial port on the host (2 台虛擬機器皆需新增COM port)。

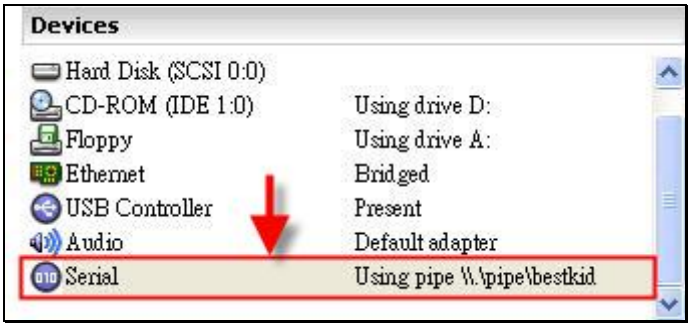


▲ 圖 3.3 新增 Serial port 設定(一)

3. 最後確認Physical serial port為COM1 按下[完成]，即表示新增成功。

COM port 的連線相關參數設定：

1. 新增COM port成功之後，可以在[Devices]看到[Serial]的項目，雙擊此項目進行Serial port相關參數設定。

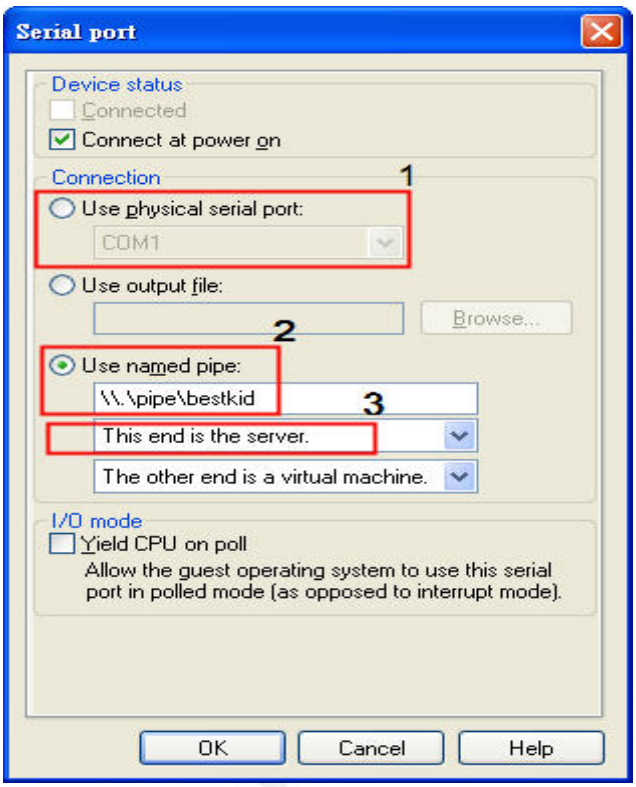


▲ 圖 3.4 新增 Serial port 設定(二)

2. 步驟 1. 檢查使用physical serial port 是否為COM1。

步驟 2. 檢查named pipe第一個欄位是否為\\.\pipe\xxxx之形式，此處設為\\.\pipe\bestkid (特別注意此處；2 台虛擬機器需設定相同)。

步驟 3. named pipe 第 2 個欄位的設定方法，藉由 COM port 連接的 2 台虛擬機器，1 台需設為 This end is the server 反之，另 1 台則需設為 This end is client 。



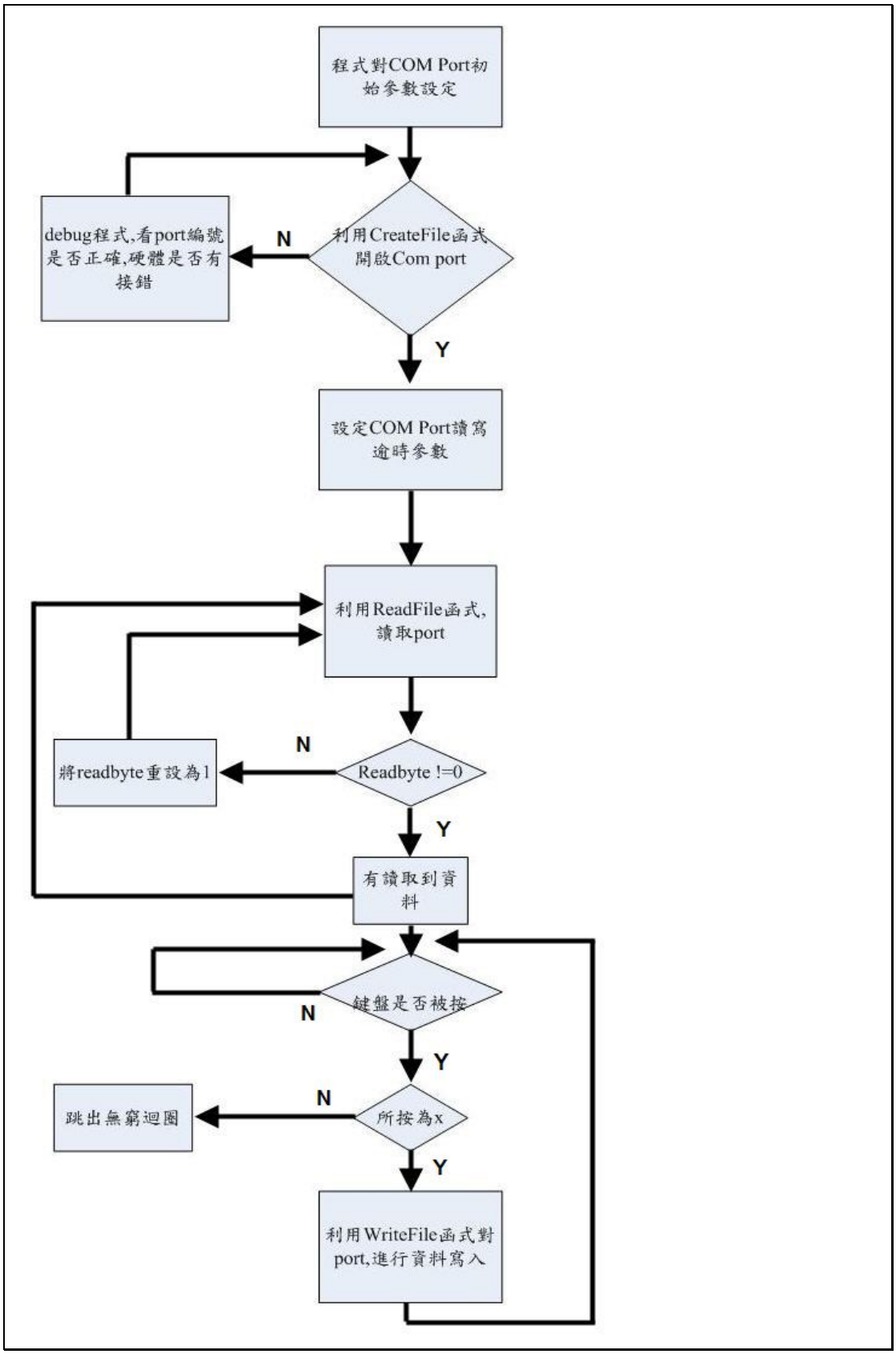
▲ 圖 3.5 新增 Serial port 設定(三)

3.3 進行虛擬機器第一階段 C 程式測試

1. 按下power on 開起 2 台COM port已連結之虛擬機器。

2. 將利用WINDOWS API 完成的correctrs232.c編譯成功，而後將執行檔 correctrs232.exe 由實體本機WINDOWS XP環境下分別拖曳至 2 台 Windows 2000 環境的虛擬機器桌面。

3. C程式概略架構如下：



▲ 圖 3.6 第一階段 C 程式測試流程

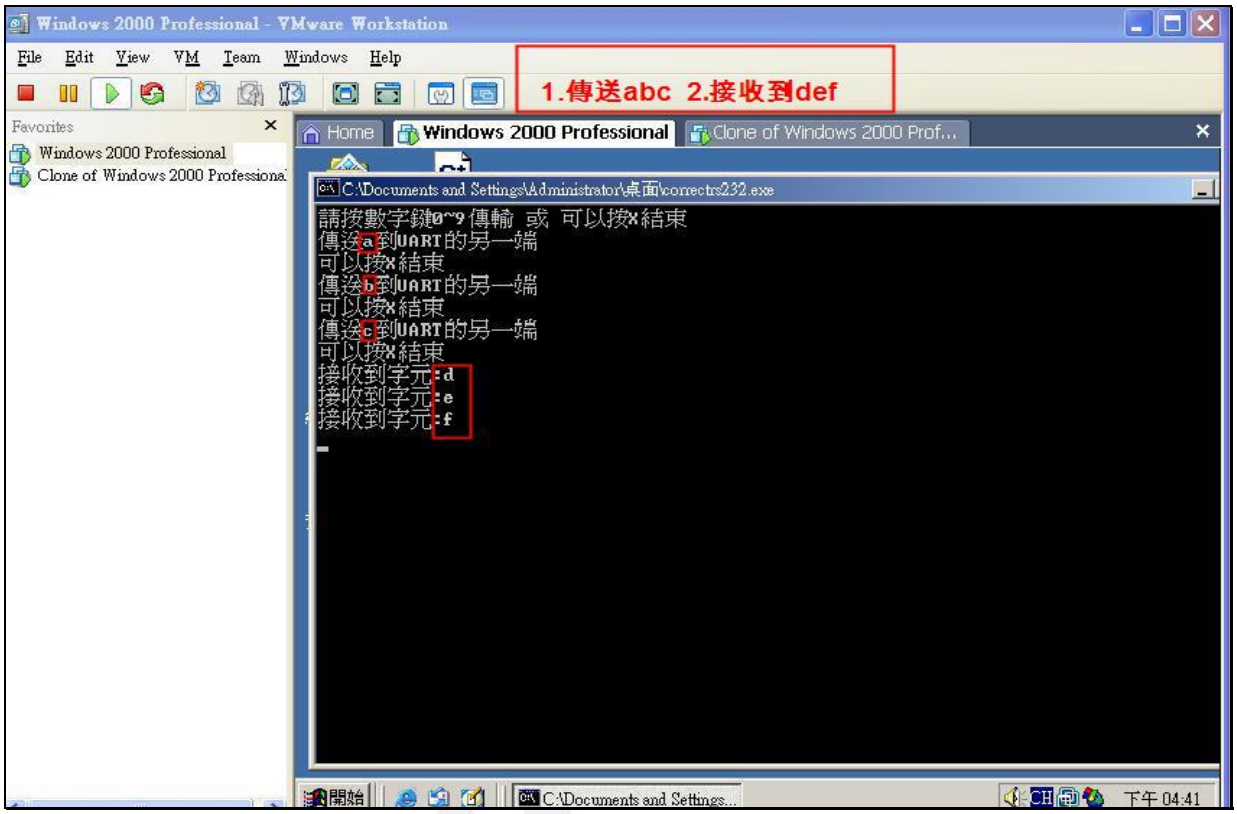
4. correctrs232.c程式碼內容如下:

```
#include<windows.h>
#include<stdio.h>
#include<conio.h>

int main(int argc， char* argv[])
{
HANDLE com1_handle ; //RS-232 的 Com1 的 handle
DCB dcb ; //設定傳輸參數所需之結構
char buffer[10]; //讀取資料所需的緩衝區
DWORD read_bytes = 1 ; //每次讀取的 byte 數
COMMTIMEOUTS time_out ;  //設定讀寫逾時參數的結構
com1_handle=CreateFile("COM1"，
GENERIC_READ|GENERIC_WRITE，0 ，NULL，OPEN_EXISTING，
FILE_ATTRIBUTE_NORMAL，NULL);
//開啓 com1，設爲一般讀取寫入，一般屬性，不使用非同步 IO
BuildCommDCB( "baud=4800 parity=N data=8 stop=2"，&dcb); /*設定傳輸
參數結構的內含值:鮑率:4800，無同位檢查，資料位元 8，停止位元 2*/
SetCommState( com1_handle ， &dcb ); //設定傳輸參數

time_out.ReadIntervalTimeout = MAXDWORD ;
/*設爲 MAXDWORD；如果沒有資料供讀取，ReadFile 函式將立即返回*/
time_out.ReadTotalTimeoutMultiplier = 0 ;
time_out.ReadTotalTimeoutConstant = 0 ;
//不使用讀取總和時間來判斷是否讀取逾時
time_out.WriteTotalTimeoutMultiplier = 5 ;
```

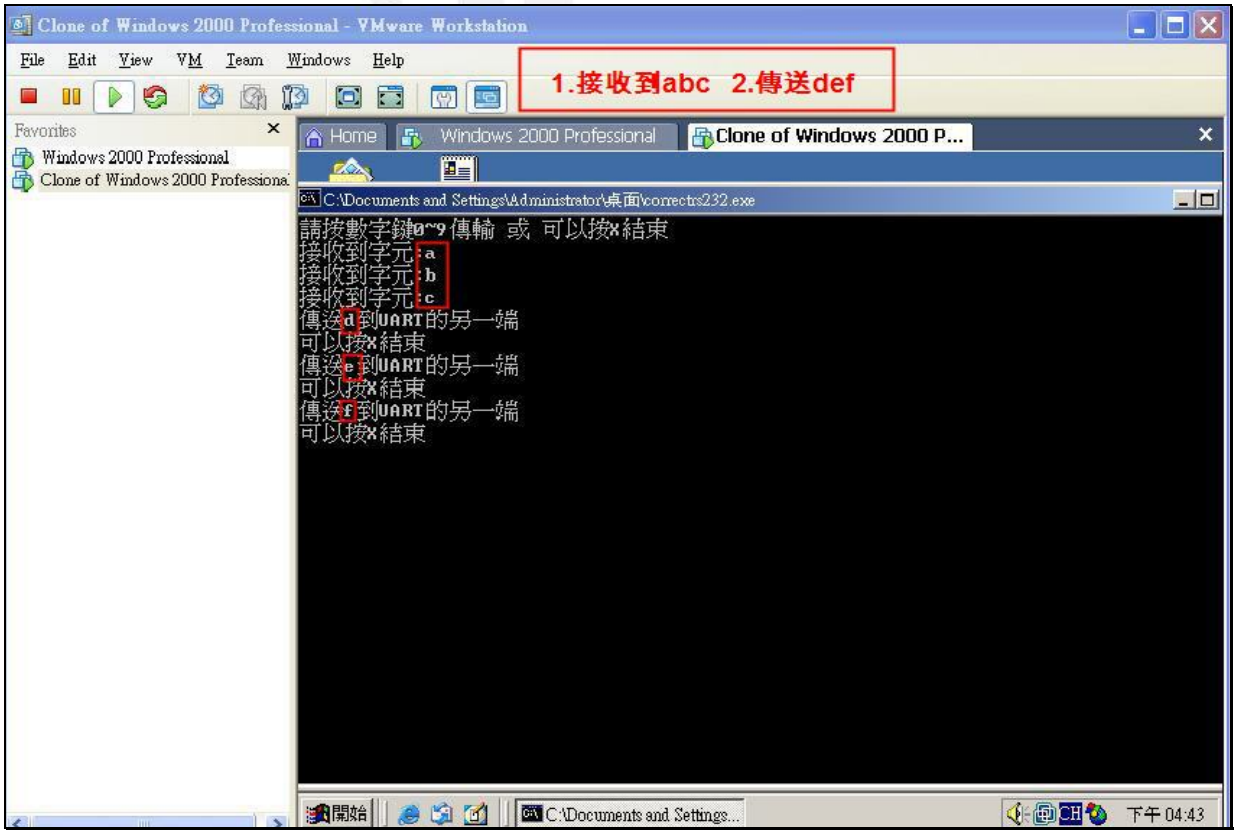
```
time_out.WriteTotalTimeoutConstant = 50 ;
/*使用寫入總和時間來判斷是否寫入逾時，逾時 WriteFile 函式將立即返回*/
SetCommTimeouts( com1_handle ， &time_out ) ;
/*設定 com1 讀寫逾時返回*/
printf(" 請按數字鍵 0~9 傳輸 或 可以按 X 結束\n");
while( 1 )
{
        ReadFile( com1_handle ，buffer ，read_bytes ，&read_bytes ，
        NULL);
          //由 com1 讀取 1byte
        /*因爲 ReadFile 函式逾時的時候，不僅沒讀到資料(所以 buffer[0]沒變
        動)就返回，還會把 read_bytes 內含值改爲 0，因此 read_bytes!= 0 時
        表示有讀到資料*/
        if(read_bytes!= 0)
        {
              printf("接收到字元:%c\n"，buffer[0]);
        }
        else
        {
              read_bytes = 1 ;
        /*因爲 ReadFile 函式逾時的時候，不僅沒讀到資料(所以 buffer[0]沒變
        動)          就返回，還會把 read_bytes 內含值改爲 0，所以我們要設
        回初值，不然下一輪   會讀不到資料(因爲內函值爲 0，表示要讀取
        0byte，所以讀不到資料)*/
        }
        if( kbhit() != 0 )
```

```
        {
            //如果鍵盤有被按到，kbhit 函式傳回值不等於 0
            int key ;        //存放被按的鍵的鍵值
            key = getch() ;     //把被鍵值讀出來
            if( key == 'x' )
            {
                break ; //按下小寫 x 結束本程式
            }
            else
            {
                WriteFile( com1_handle，&key，read_bytes，&read_bytes，
NULL);
                printf("傳送%c 到 UART 的另一端\n"，key);
                printf("可以按 X 結束\n");
            }
        }
    }
    CloseHandle( com1_handle );   //關閉 com1
    return 0;
}
```

5. 分別執行 2 台虛擬機器桌面之correctrs232.exe，開始透過COM1 port互傳資料，測試圖如下。



▲ 圖 3.6 第一階段 C 程式執行檔測試(一)



▲ 圖 3.7 第一階段 C 程式執行檔測試(二)

6. 測試成功之後，改以 2 台實體PC以RS-232 連接COM Port測試也成功。

3.4 利用 ATL 技術製作 DLL

1. 開啟Visual C++ ， File->New。

2. Project name: ATLtest，Server Type中選擇Dynamic Link Library(DLL)->Finish->OK。

3. Insert中選擇New ATL Object->Simple Object->輸入test。

▲ 圖 3.8 ATL Object Wizard 填寫內容

說明: 在 Short Name 欄位打入 test ，則 ATL Object Wizard 會自動填滿其他欄位。

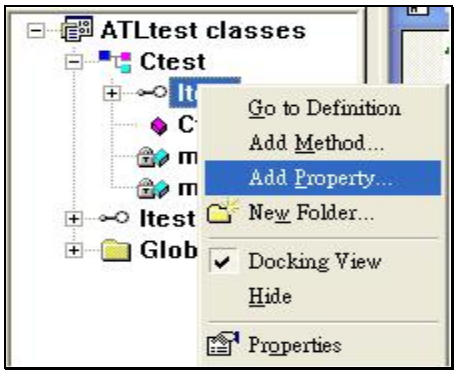
(1) Class欄位所顯示的名稱就是控制項的類別名稱，這個類別會繼承test控制項要支援的所有介面。

(2) ATL Object Wizard將會Ctest類別的原始程式寫入畫面所指定的test.h和test.cpp 檔案。

(3) CoClass欄位中放有控制項的元件類別，也就是控制項的型別函式庫。

(4) Interface欄位中的名稱是控制項公開屬性和方法給外界使用的介面。

(5) Type和ProgID欄位中的名稱則是test物件的型態及程式開發者要使用的代碼。



▲ 圖 3.9 Add Property

4. 加入物件參數，參考圖 3.9 於Itest上按右鍵 ->Add Property -> short/wvalue

重複此步驟，增加另一參數 short/rvalue。

5. 開啟Ctest[ ]進行設定，在Ctest[ ]中加入

Private:

short m_wvalue ;

short m_rvalue ;

/*在此 2 變數就是做為 c 程式與其他欲引用此 DLL 檔的其他程式(例如 VB) 之間的媒介，互相溝通之用*/

6. 在Itest中，找到get和put方法，將其編輯成如下圖所示內容：

```
STDMETHODIMP Ctest::get_wvalue(short *pVal)
{
    // TODO: Add your implementation code here
    *pVal=m_wvalue;
    return S_OK;
}

STDMETHODIMP Ctest::put_wvalue(short newVal)
{
    // TODO: Add your implementation code here
    m_wvalue=newVal;
    return S_OK;
}
```

▲ 圖 3.10 m_wvalue 之 get/put 方法設定

```
STDMETHODIMP Ctest::get_rvalue(short *pVal)
{
    // TODO: Add your implementation code here
    *pVal=m_rvalue;
    return S_OK;
}

STDMETHODIMP Ctest::put_rvalue(short newVal)
{
    // TODO: Add your implementation code here
    m_rvalue=newVal;
    return S_OK;
}
```

▲ 圖 3.11 m_rvalue 之 get/put 方法設定

函式說明:

(1) get和put方法之目的在於存取test控制項的Property 資料。

(2) *pVal ： 指向目前Property 資料的指標。

(3) newVal ：放置新的Property的值.。

7. 標頭檔與變數宣告部份：

(1) 將correctrs232.c中所需要include的標頭檔複製到test.cpp。

(2) 將correctrs232.c中變數宣告部份，一樣複製到test.cpp，參考圖 3.12。

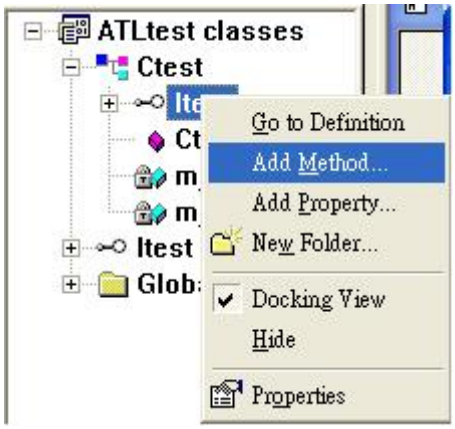
```
// test.cpp : Implementation of Ctest
#include "stdafx.h"
#include "ATLtest.h"
#include "test.h"
#include<windows.h>
#include<stdio.h>
#include<conio.h>

////////////////////////////////////////////////////////////////////////////
// Ctest

HANDLE com1_handle ; //RS-232的Com1的handle
DCB dcb ; //設定傳輸參數所需之結構
char buffer[10]; //讀取資料所需的緩衝區
DWORD read_bytes = 1 ; //每次讀取的byte數
COMMTIMEOUTS time_out ;  //設定讀寫逾時參數的結構
```

▲ 圖 3.12 製作 DLL 變數宣告與標頭檔

8. 新增 4 個函式：

(1) 在Itest上，按右鍵，Add Method ，參考圖 3.13。

(2) 依照此方式，依序新增 openport( )，readport( )，writeport( )，closeport( )。



▲ 圖 3.13 Add Method

Openport( )函式程式碼：

```
STDMETHODIMP Ctest::openport()
{
// TODO: Add your implementation code here
com1_handle=CreateFile("COM1"，
GENERIC_READ|GENERIC_WRITE，0 ，NULL，OPEN_EXISTING，
FILE_ATTRIBUTE_NORMAL，NULL);
//開啓 com1，設爲一般讀取寫入，一般屬性，不使用非同步 IO
BuildCommDCB( "baud=4800 parity=N data=8 stop=2"，&dcb); /*設定傳輸
參數結構的內含值:鮑率:4800，無同位檢查，資料位元 8，停止位元 2*/
SetCommState( com1_handle ， &dcb ); //設定傳輸參數


time_out.ReadIntervalTimeout = MAXDWORD ;
/*設爲 MAXDWORD:如果沒有資料供讀取，ReadFile 函式將立即返回*/
time_out.ReadTotalTimeoutMultiplier = 0 ;
time_out.ReadTotalTimeoutConstant = 0 ;
//不使用讀取總和時間來判斷是否讀取逾時
time_out.WriteTotalTimeoutMultiplier = 5 ;
time_out.WriteTotalTimeoutConstant = 50 ; /*使用寫入總和時間來判斷是否
```

```
寫入逾時，逾時 WriteFile 函式將立即返回*/
SetCommTimeouts( com1_handle ， &time_out ) ;
//設定 com1 讀寫逾時返回
return S_OK;
}
Readport( )函式程式碼:
STDMETHODIMP Ctest::readport()
{
// TODO: Add your implementation code here
ReadFile( com1_handle ， buffer ， read_bytes ，&read_bytes ，NULL);
//由 com1 讀取 1byte
/*因爲 ReadFile 函式逾時的時候，不僅沒讀到資料(所以 buffer[0]沒變動)就返回，還會把 read_bytes 內含值改爲 0，因此 read_bytes!= 0 時表示有讀到資料*/
        if(read_bytes!= 0)
        {
            m_rvalue=buffer[0]; //假如有接收到字元
        }
        else
        {
          m_rvalue='x'; //假設沒接收到字元..將 m_rvalue 設爲字元 x
          read_bytes = 1;
        /*因爲 ReadFile 函式逾時的時候，不僅沒讀到資料(所以 buffer[0]沒變動)就返回，還會把 read_bytes 內含值改爲 0，所以我們要設回初值，不然下一輪會讀不到資料(因爲內函值爲 0，表示要讀取 0byte，所以讀不到資料)*/
```

```
        }
  return S_OK;
}
```

Writeport( )函式程式碼:

```
STDMETHODIMP Ctest::writeport()
{
// TODO: Add your implementation code here
int key;
key=m_wvalue;
WriteFile( com1_handle，&key，read_bytes，&read_bytes，NULL);
return S_OK;
}
```

Closeport( )函式程式碼:

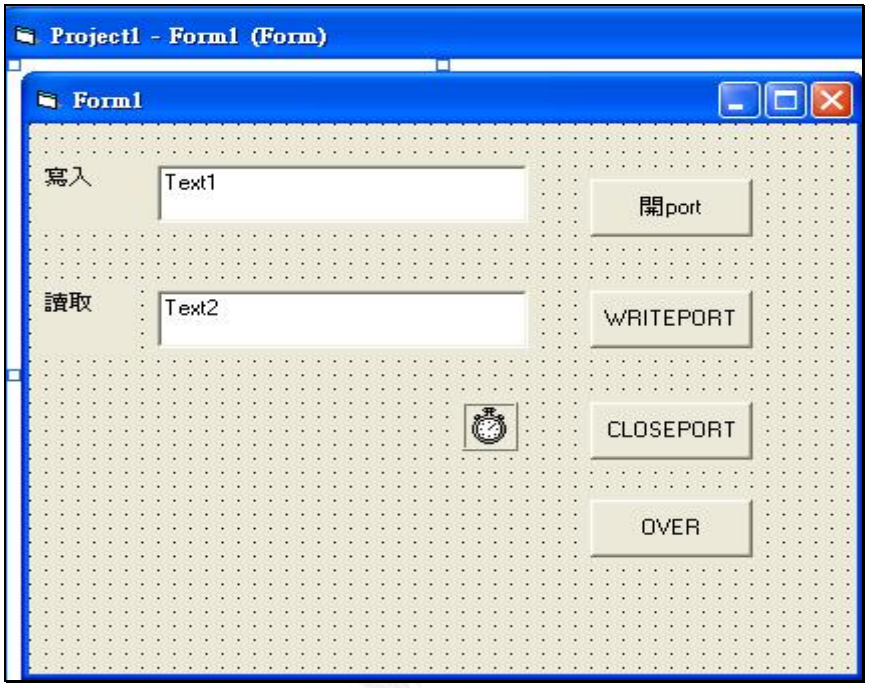
```
STDMETHODIMP Ctest::closeport()
{
// TODO: Add your implementation code here
CloseHandle( com1_handle );    //關閉 com1
return S_OK;
}
```

9. 進行編譯，Compile and Build，此時ATLtest.dll檔已建構完成，接下來，即是測試此DLL是否能有效被註冊與引用。

3.5 進行第二階段 VB 連結 DLL 檔測試:

1. 分別於 2 台虛擬機器內安裝Microsoft Visual Studio 6.0，依照 3.4 之方法建置好ATLtest.dll檔。

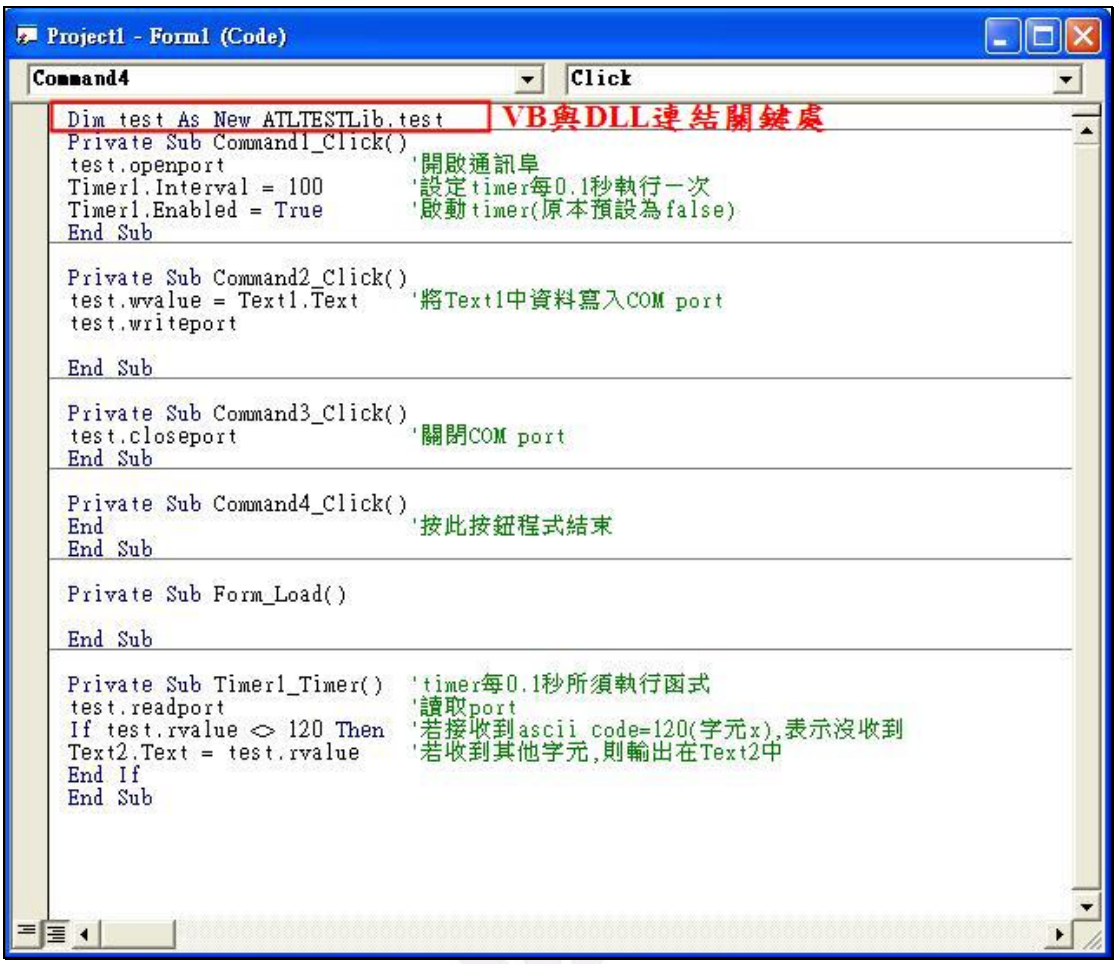
2. 於 2 台虛擬機器Windows 2000 中開啟Visual Basic，製作基本控制如下圖所示：



▲ 圖 3.14 VB 連結 DLL 測試介面

3. 於 2 台Project -> References ->勾選ATLtest 1.0 Type Library->OK ，完成註冊所需要的DLL檔。

4. VB程式碼如下圖所示：



▲ 圖 3.15 VB 程式碼視窗

5. project1 存檔完畢之後，分別開啟兩台虛擬機器Windows 2000 ，開啟 project1.exe 依照下列步驟測試。

(1) 點擊OPENPORT按鈕開啟COM port。

(2) 在Text1 中(上方文字欄位)輸入所要傳送的資料，按下WRITE 按鈕。

(3) 在Text2 中(下方文字欄位)觀察接收到的資料，是否與另一端傳送資料互相吻合。

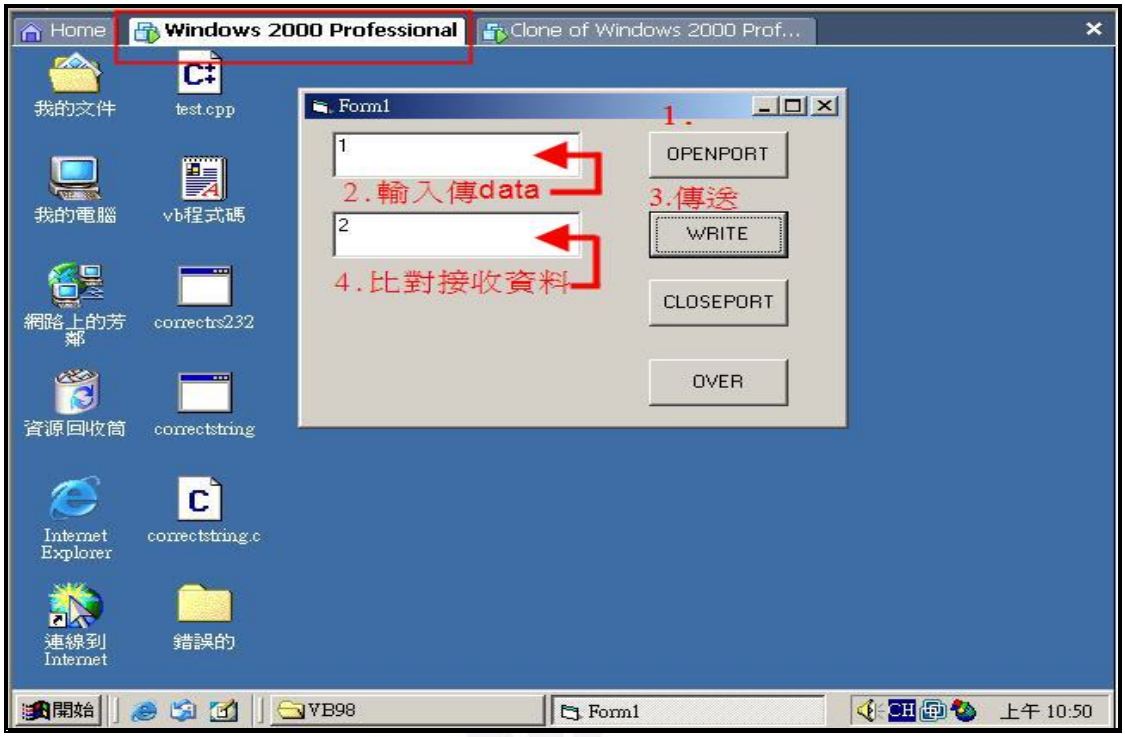
(4) 比對無誤，點擊CLOSEPORT按鈕 關閉COM port。

(5) 點擊OVER結束VB測試程式，測試結果參考圖 3.16 與圖 3.17。

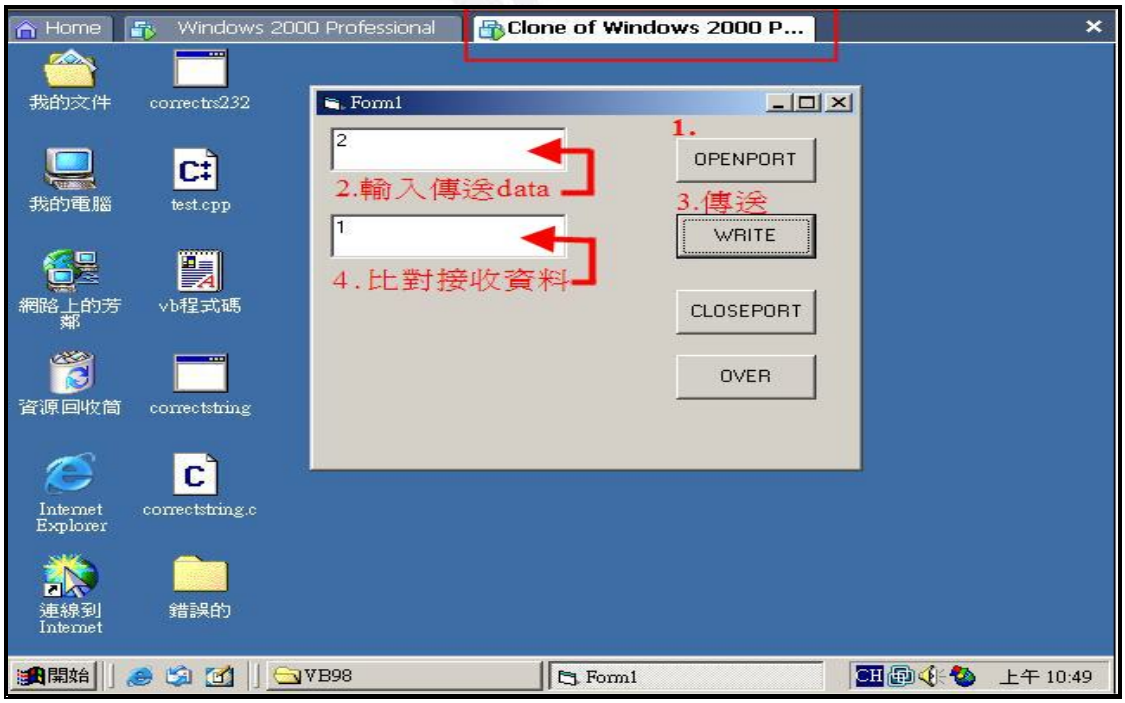
6. 測試目的：

測試目的在於了解利用 ATL 製作出的 ATLtest.dll 是否能被 VB 註冊與呼叫，而 VB 連結此 DLL 檔之後是否能順利引用 DLL 提供的函式與

Property 進行 COM port 之間資料的傳輸，所有測試到目前為止，確定了互連 COM port 傳輸的可行性，而後下一階段再進入到網頁遠端控制測試。



▲ 圖 3.16 VB 連結 DLL 測試結果(一)



▲ 圖 3.17 VB 連結 DLL 測試結果(二)

3.6 進行第三階段 ASP 網頁測試

3.6.1 測試 ASP 是否能註冊並呼叫 DLL 檔之函式

1. 先製作一個很簡單的c程式，其中inc函式，只是將value這個property加 1。
2. 使用ATL技巧，將簡單c程式製作成.DLL檔。
3. 架設好Web Server，製做一個簡單的網頁a.asp.有 1 個按鈕，按下後即會執行test.asp網頁。
4. 撰寫test.asp內容如下：

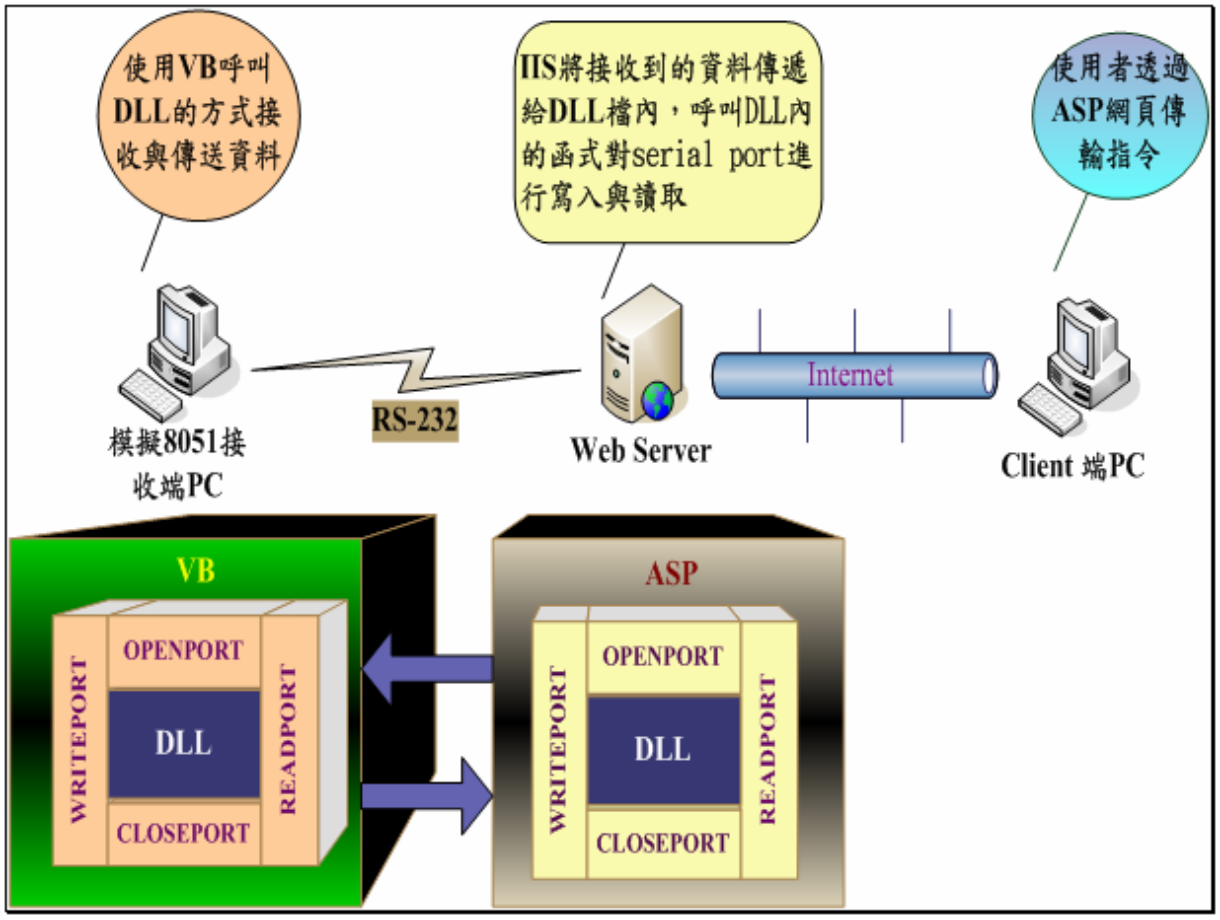
```
<%
Dim test
Set test=server.createobject(“DLL 的檔名”)
Test.inc   ‘呼叫 dll 檔中 inc 函式
%>
```

5. 將test.asp存檔在wwwroot資料夾中。
6. 將要註冊的DLL檔也傳到web server的wwwroot資料夾中。
7. 開始註冊，開始->執行 指令為Regsvr32.exe DLL檔路徑。(ex:c:\ProgramObject\ATLtest\Debug\ATLtest.dll)，註冊成功之後 。
8. 在瀏覽器網址列上打入 http://web server IP/a.asp 。
9. 測試asp註冊並引用DLL檔成功。

3.6.2 測試ASP網頁呼叫控制RS232 之DLL

一. 使用 VB 呼叫 DLL 方式接收測試

測試流程架構如下圖所示:



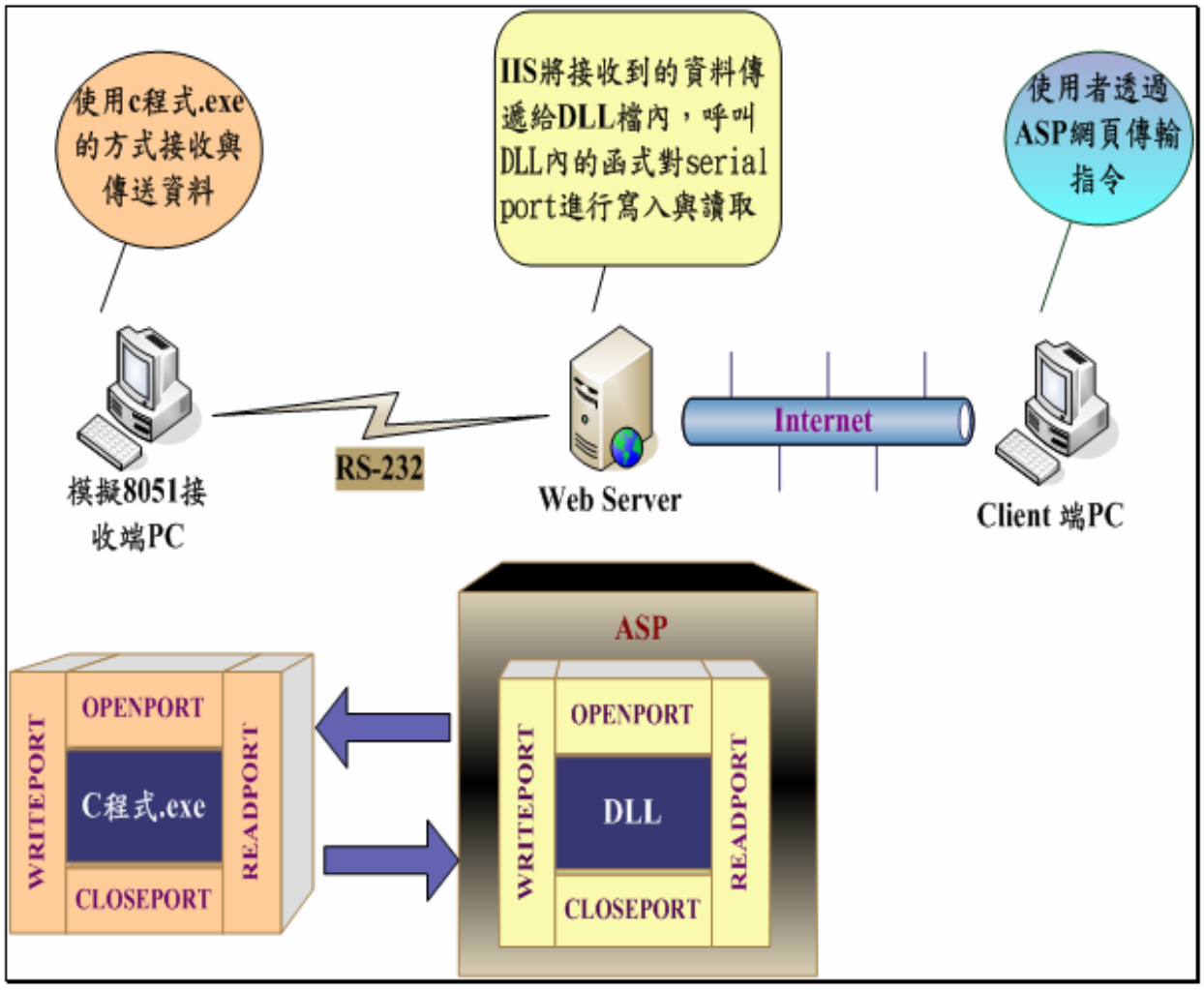
▲ 圖 3.18 ASP 與(VB 連結 DLL)測試流程架構

測試結果：

使用 ASP 呼叫 Server 端 DLL 檔，透過 openport()、readport()、writeport()、closeport() 函式對 serial port 進行讀取與寫入資料，而模擬 8051 接收端 PC 則是使用 VB 連結 DLL 檔的方式來接收與傳送資料。以上測試結果成功，Client 端 PC 與模擬 8051 接收端 PC 可以互相傳遞字元資料無誤。

二.使用 C 程式執行檔方式接收測試

測試流程架構如下圖:



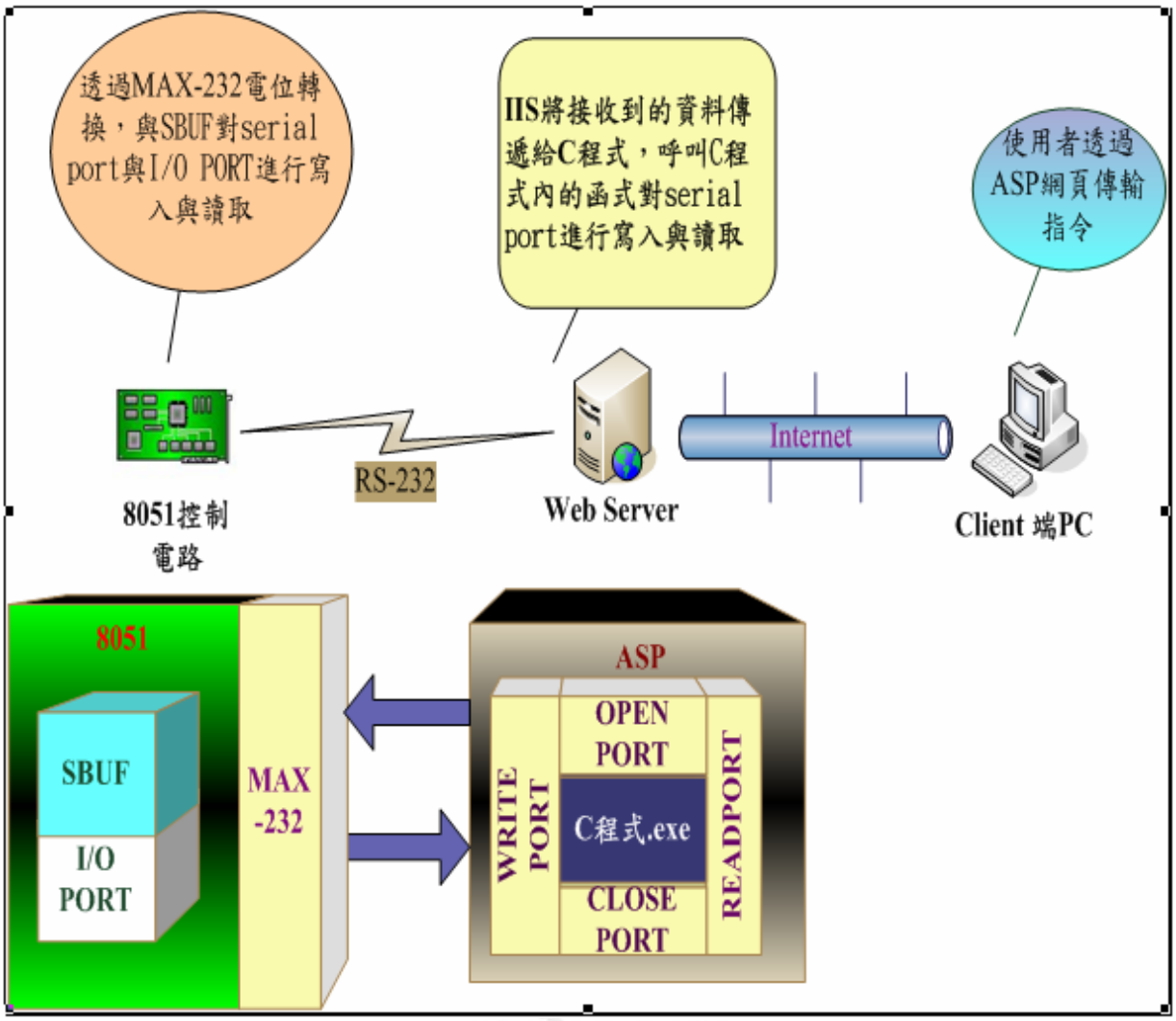
▲ 圖 3.19 (ASP 連結 DLL)與 C 程式測試流程架構

測試結果：

以上測試結果卻是失敗，模擬 8051 接收端 PC 收到的值，不是 126 就是 127 這兩個整數；之後也真正連接 8051 硬體作測試，結果仍一樣錯誤；經過不斷測試與除錯，仍無解。之後詢問袁世一老師，才了解，原來是軟體跟軟體之間透過 DLL 檔方式時，資料透過 property 作為軟體與 DLL 的媒介時，發生了對於暫存區資料的位址定義不同，導致這種情形發生。這也是光靠理論所無法發現的，為有實做才可能會遇到；為解決此問題，只好另尋他法，更改了 Web Server 端的程式架構；最後找到能以 ASP 直接呼叫 Web Server 端的應用程式，解決了問題。

3.6.3 修改後之 ASP 與 8051 架構

最後測試流程架構如下

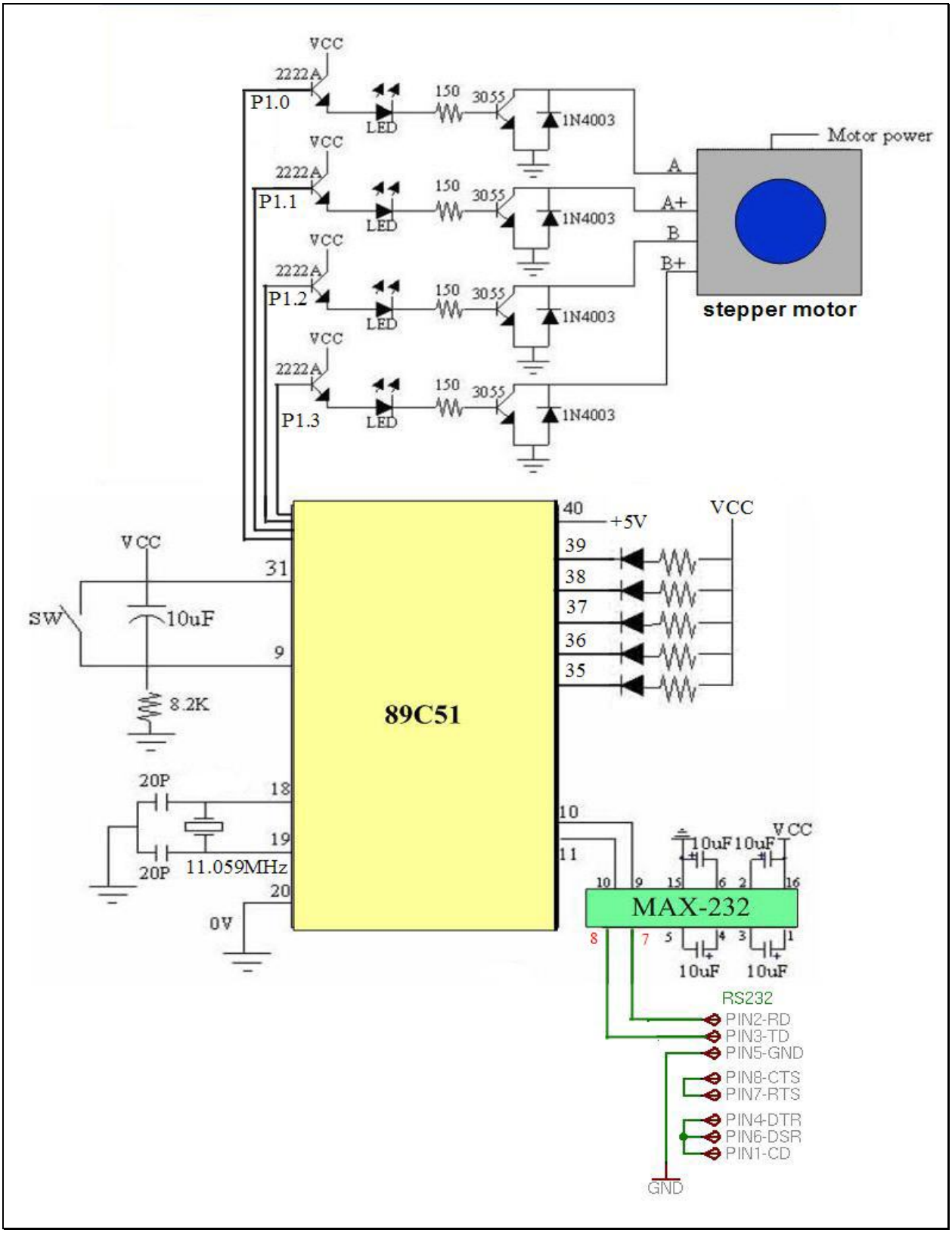


▲ 圖 3.20 (ASP 呼叫 C 程式)與 8051 測試流程架構

測試結果：

經修改架構之後，完全避開不同軟體，對於資料暫存區的位址定義問題。此架構軟體與硬體首次整合測試是成功的。之後便確定使用此架構，接下來進行對遠端控制網頁的美化工作。

3.7 遠端控制硬體電路介紹



▲ 3.21 遠端控制硬體電路圖

電路解說：

1. MAX-232電路部份：

即是在做 PC 與 8051 電位訊號轉換的工作。8051 所用的 Logic Level 為 High=5V，Low=0V，而 RS232 則 High=-3~-12V， Low=3~12V。藉由 MAX232 將電位轉換，才使得電腦與 8051 可以正確溝通，亦可使用 ICL232 替代。

2. 步進馬達驅動電路方面：

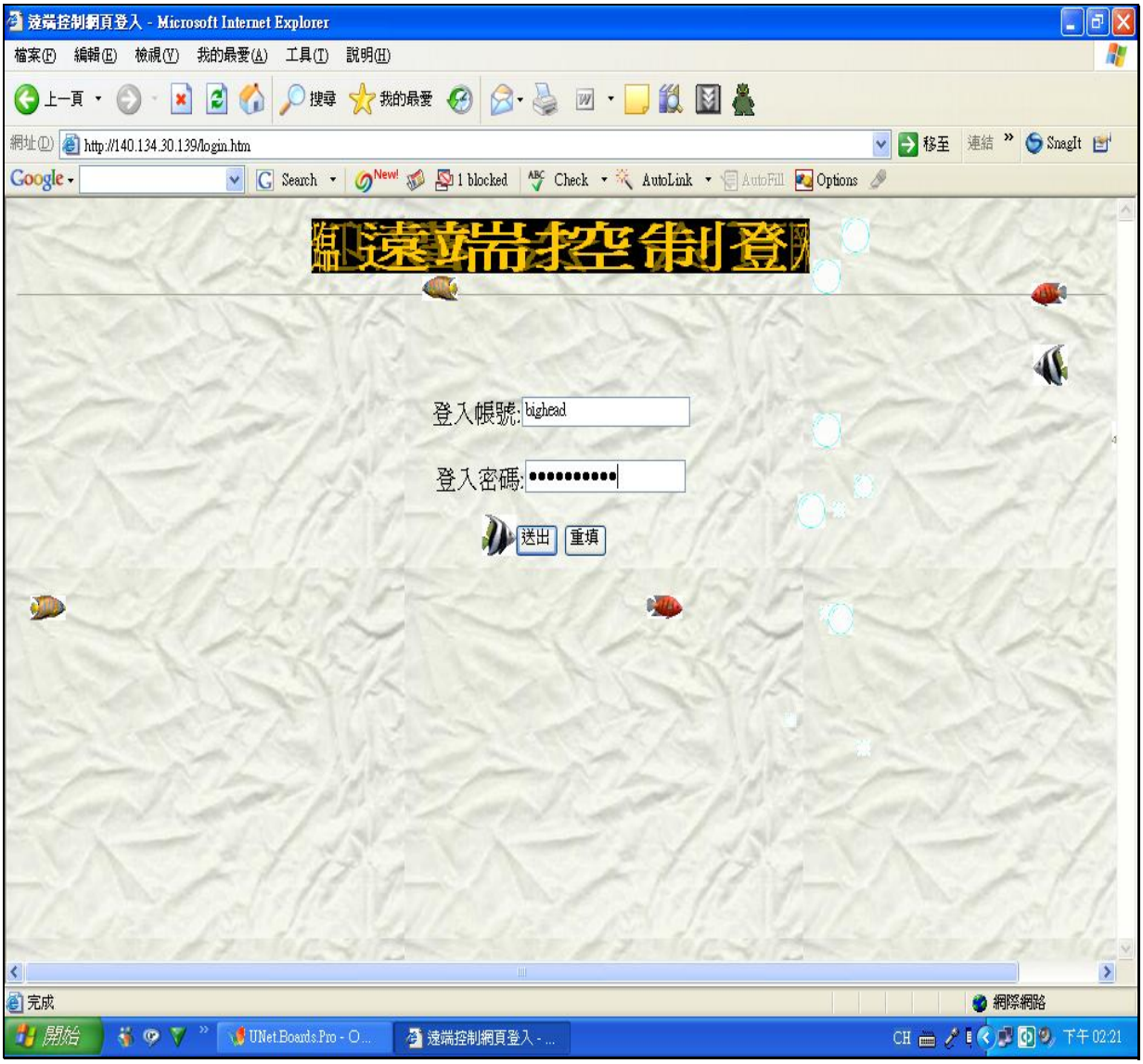
當圖 3.21 中 P1.0~P1.3(Pin1~4)，其中一腳位為 LOW(低電位)時，電晶體 222A 電路就會導通，此時電流會由 VCC 經 222A 電晶體流過 LED 與 150Ω的電阻；而電晶體 3055 是作開關動作，當 3055 電晶體基極有電流通過時，電晶體導通，此時步進馬達電源隨即供應電流流到 3055 電晶體導通的那組定子，以步進馬達一相激磁原理，也就是依序讓 4 個電晶體導通，便能讓步進馬達產生正反轉。在此處，要選用三種激磁方式的哪一種，可透過 8051 程式碼更改 P1 的值，即可選擇激磁方式，本專題在此是採用一相激磁的方式。

規格	數量
NPN 電晶體 222A	4
LED	9
NPN 電晶體 3055	4
二極體 1N4003	4
電阻 150Ω	4
電阻 220Ω	5
五線式步進馬達	1

▲ 表 3.1 硬體規格零件

第四章 ASP 網頁遠端控制敘述

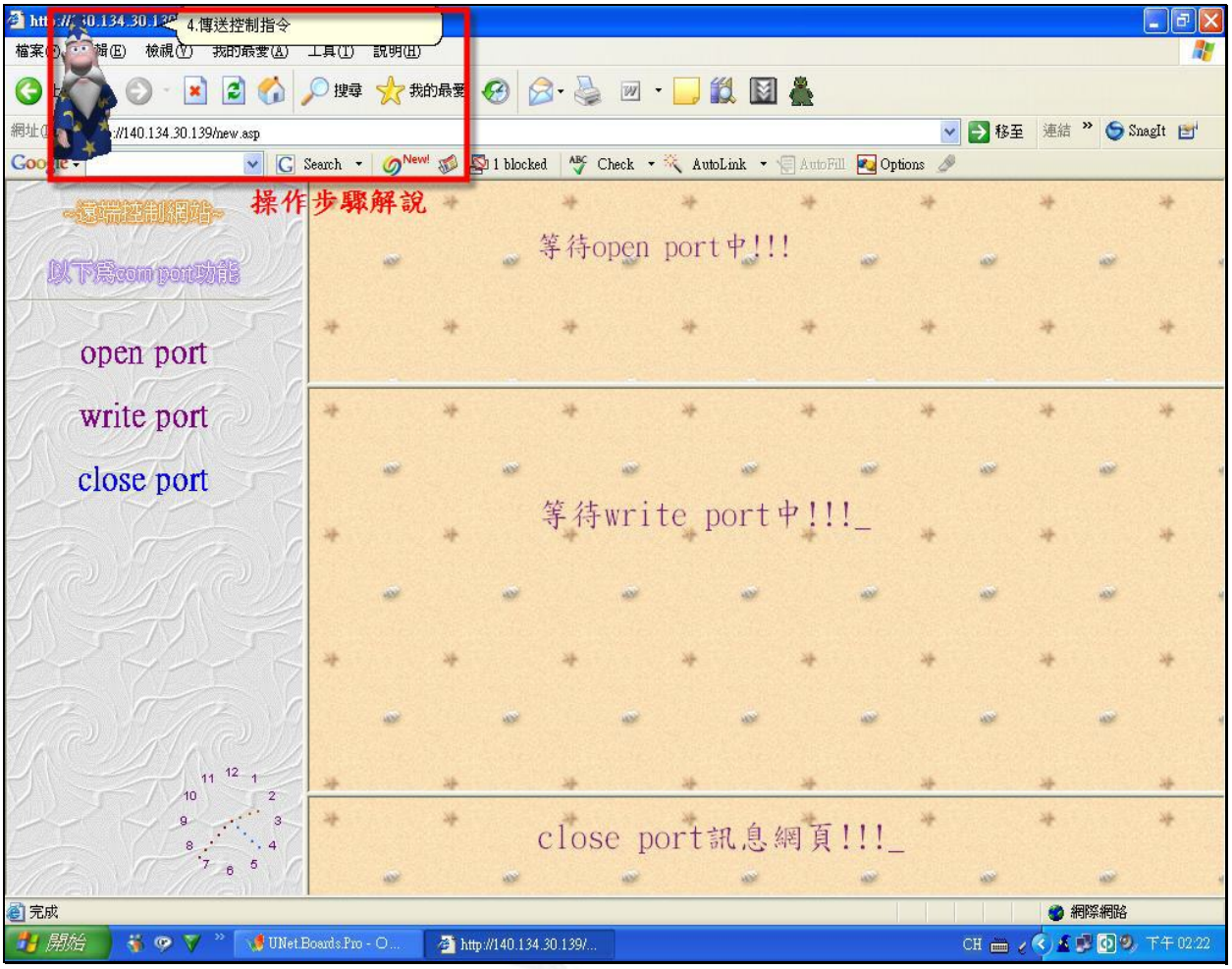
依照 3.6.3 的軟硬體系統架構實做，並將網頁美化；而考慮到對遠端使用者的控管，也是對整個控制系統的安全考量，因此也做了簡易的帳號登入管制，以下就是整個遠端控制操作流程，首先，先連上遠端控制帳號登入網頁 http://140.134.30.139/login.htm，進行使用者帳號與密碼確認。



▲ 圖 4.1 遠端控制帳號登入網頁

4.1 遠端控制步驟解說

經使用者帳號密碼確認無誤之後，登入遠端控制操作網頁。接下來觀看左上角會有特別設計的魔法師圖示，step by step 講解遠端控制步驟。



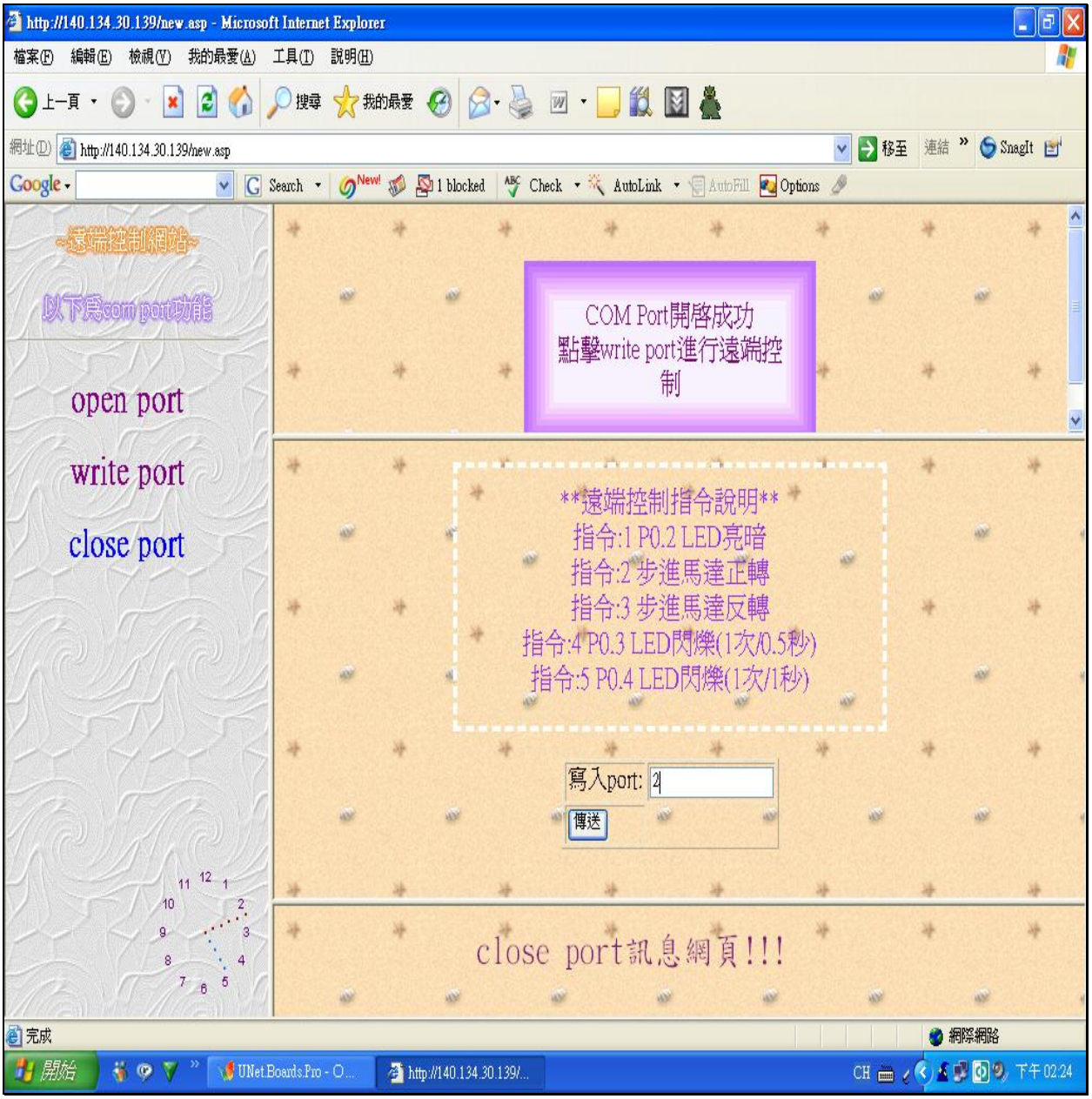
▲ 圖 4.2 遠端控制步驟解說網頁

講解內容如下：

1. 首先點擊openport開啟serial port 。
2. 接下來，點擊writeport，準備控制選擇指令。
3. 觀看控制指令說明，選擇控制指令。
4. 離開之前，請記得點擊closeport。

4.2 串列埠初始化設定與傳輸控制指令

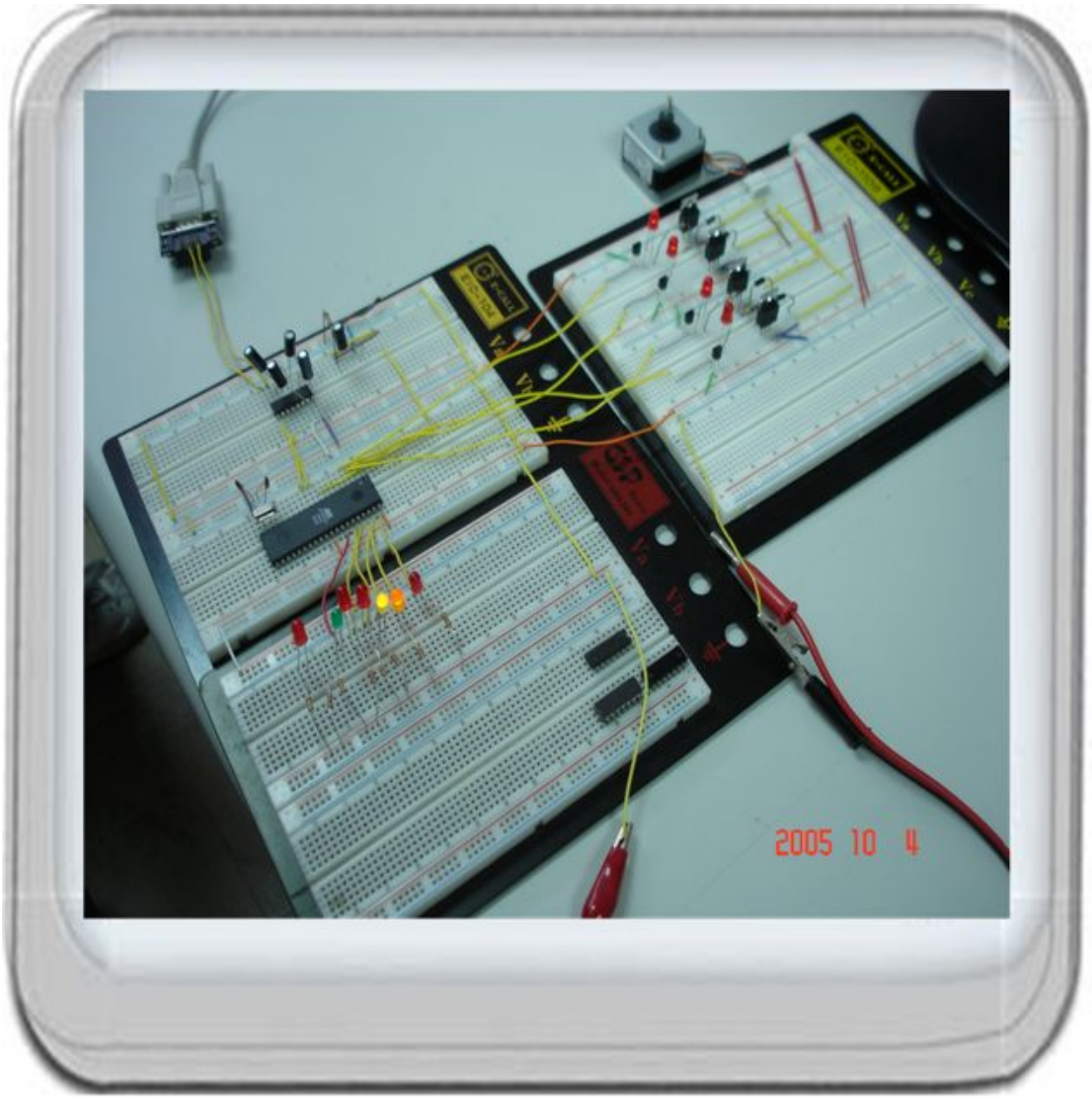
點擊左邊框架頁內的 openport 之後，表示完成對 Web Server 端 serial port 初始化設定，例如鮑率、資料傳輸格式、有無同位檢查、讀寫逾時設定..等等；並且開啟 serial port，準備傳輸資料。接下來點擊 writeport，會出現控制指令的說明板，觀看完指令說明，即可輸入控制指令，按下傳送按鈕即完成一次控制指令的任務。



▲ 圖 4.3 遠端控制網頁指令傳輸

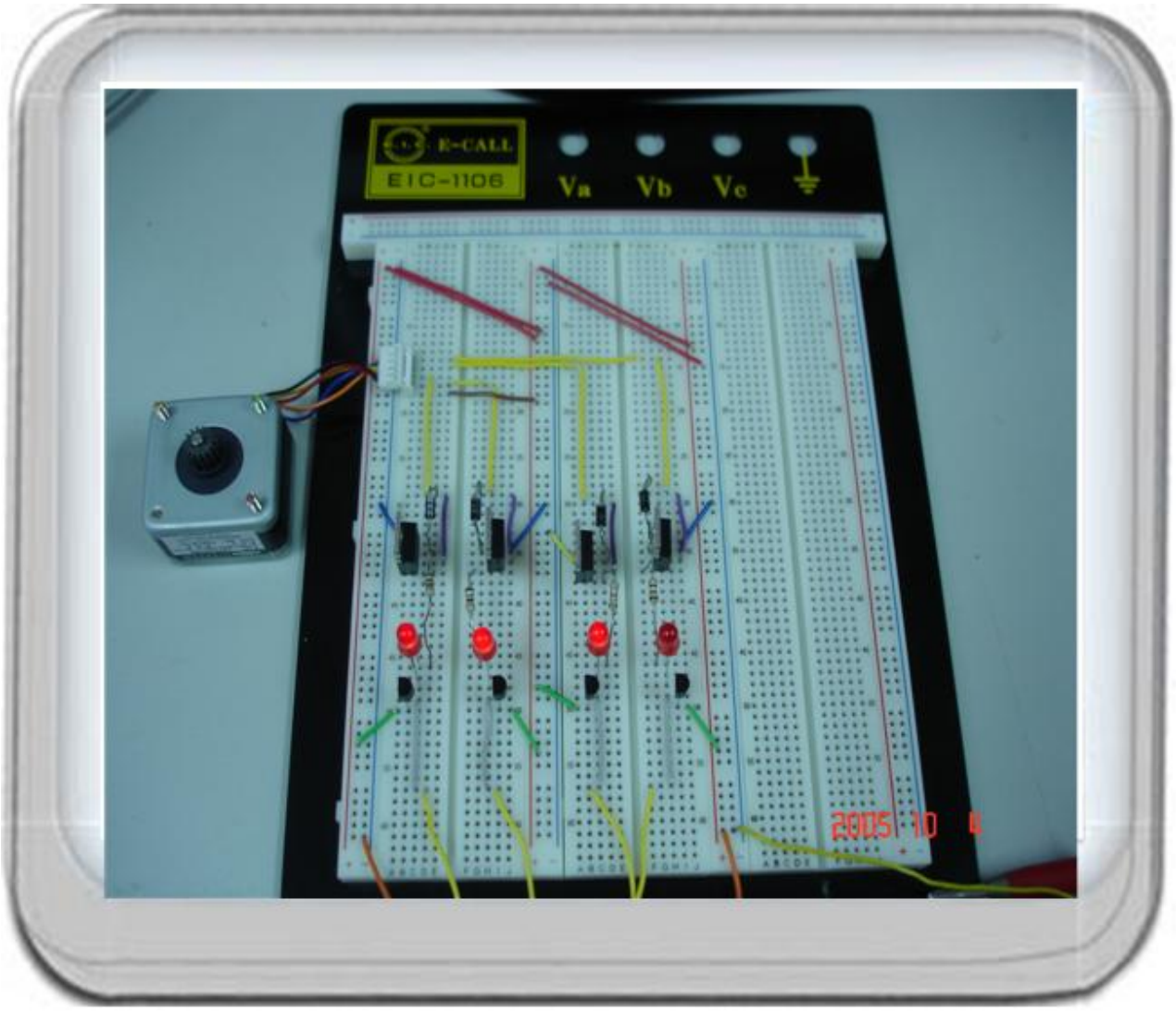
4.3 實際遠端控制硬體測試圖

執行指令 1: P0.2 LED 亮暗與指令 3: 步進馬達反轉之測試圖



▲ 圖 4.4 完整硬體測試圖

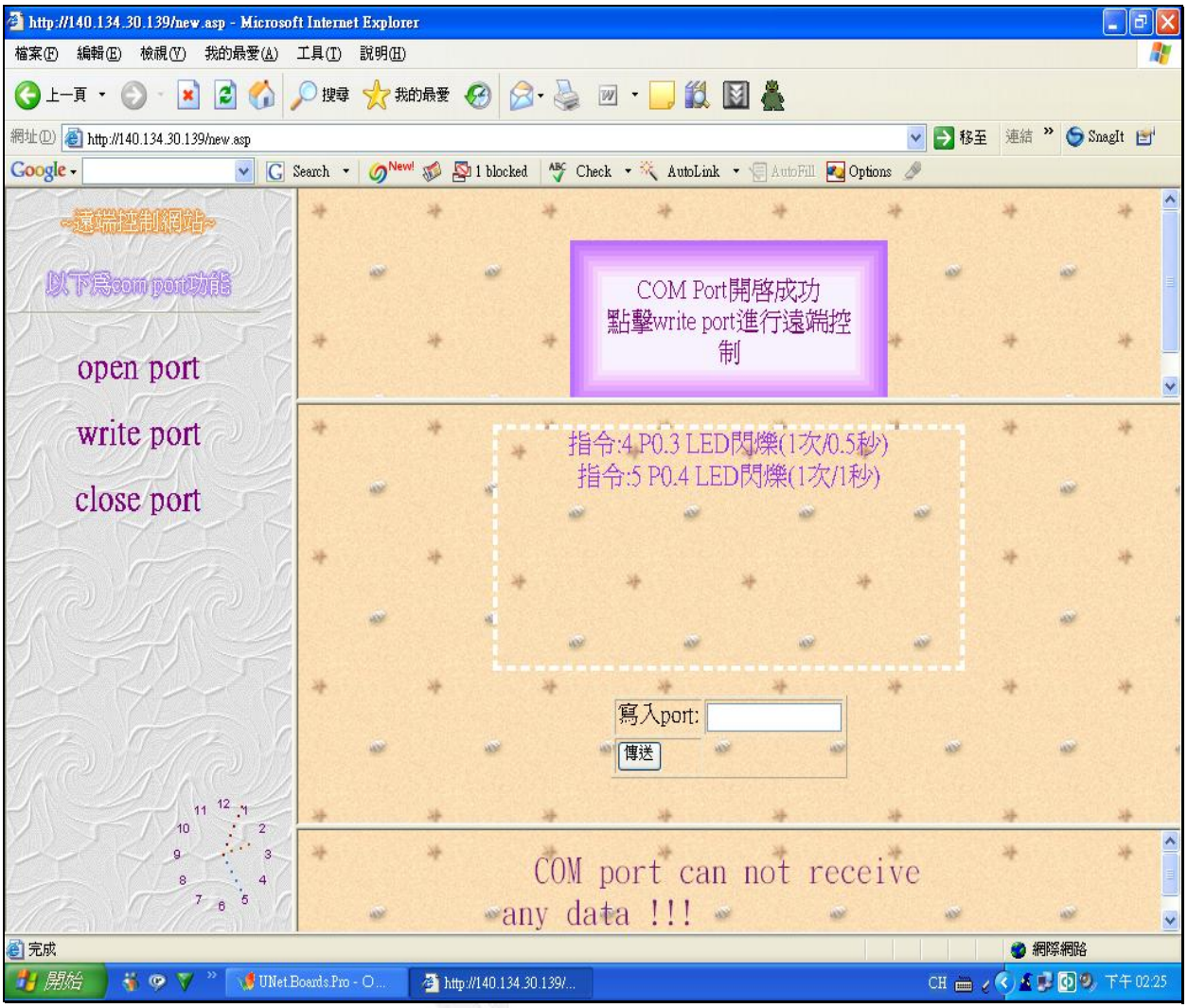
執行指令 3：步進馬達正轉指令測試圖



▲ 圖 4.5 Motor 驅動電路特寫

4.4 結束控制

控制完畢之後，記得點擊左邊框架頁內的 closeport，以免一直佔據 Web Server 的 serial port；此控制流程告一段落，參考圖 4.6。



▲ 圖 4.6 close port 測試

此專題已經完成了遠端控制的電器插頭，只需轉換電路或設備，即可接上任何想遠端控制的家電產品。

第五章 問題討論與心得感想

5.1 問題討論一

在測試完兩台 PC 之 C 程式透過 RS-232 傳輸字元之後，改用字串傳輸發現，竟然出現了讀取緩衝區 Buffer 出現了讀取 2 次的情形。而且讀取的值正確的字串並混雜著一些多餘的錯誤字串，經過反覆除錯測試，終於發現以下解決方法：

解決方法:

首先變數與函數使用的改變。

1. readbyte需換成字串長度， DWORD read_bytes = 15；
2. 寫入port的緩衝區也要設成字串， char key[15]；
3. 讀取字串函數改使用gets()函數， gets(key);

主要解決關鍵步驟：

1. 在讀取port之前先清理讀取緩衝區buffer。

   Buffer[0]='\0';

2. 更改原本timeout設定改成

   time_out.ReadIntervalTimeout = 0 ;

   time_out.ReadTotalTimeoutMultiplier = 0 ;

   time_out.ReadTotalTimeoutConstant = 600 ;

   time_out.WriteTotalTimeoutMultiplier = 0;

   time_out.WriteTotalTimeoutConstant = 600 ;

   //使用寫入總和時間來判斷是否寫入逾時，逾時 WriteFile 函式將立即返回

5.2 問題討論二

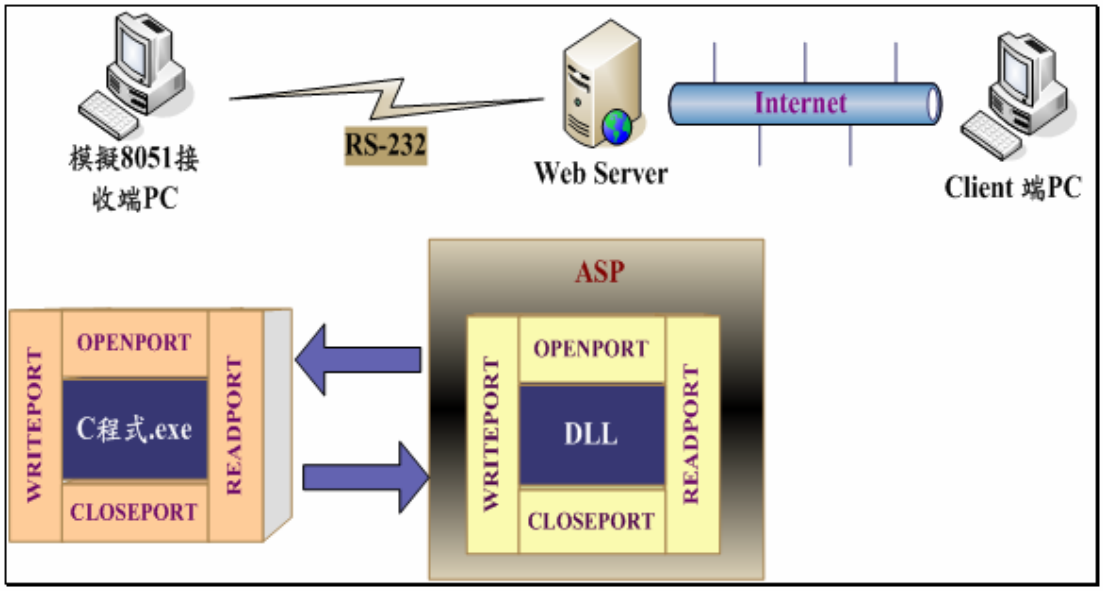
網頁接收資料上，是使用呼叫 C 程式的 readport( )函式，而原本 readport ()函式是使用無窮回圈一值持續偵測 serial port；但是這樣的作法，卻會造成使用者一旦點選 readport 連結，呼叫 server 端 readport 函式，會使的網頁非常嚴重的傳輸延遲。不僅如此，若是要一方面對 server 端進行寫入 port 一方面要讀取 port，則除非與網頁伺服器透過 rs-232 連結之 PC 傳送資料到網頁伺服器，否則使用者端將會一直持續等待 serial port 接收。

解決方法：

會造成此嚴重延遲的現象的原因在於，C 程式 readport( )函式內使用無窮迴圈不斷偵測 serial port。所以只要使用者在網頁開始讀取 port 接收資料，對方一直沒傳送資料過來，則無窮迴圈將一值持續下去，造成使用者介面端跟網頁伺服器之間嚴重延遲。解決方法使利用近似不間斷偵測 serial port 的方式接收資料，做法如下:

1. 在c程式readport( )函式部份，使用readfile函式時不使用無窮迴圈，改採用兩層有限迴圈，loop count 皆設為 32000 。
2. 而網頁 read.asp 部份利用 <meta http-equiv="refresh" content="1; url=read.asp"> 指令讓網頁自動每一秒重複執行網頁上的程式碼，也就是每一秒call一次c程式的readport ( )函式，這樣一來既可以做到接近不間斷偵測 serial port以防資料接收不完整的情形，而且也讓伺服器端讀取serial port可以約略每一秒，即可執行完一次連續性偵測；藉由網頁每秒自動呼叫readport ( )函式，再重新讀取serial port。如此一來便解決了網頁讀取serial port時嚴重延遲的問題，而且即使使用者一方面讀取serial port ，一方面寫資料到serial port也不會互相干擾到，傳輸也順暢起來。

5.3 問題討論三

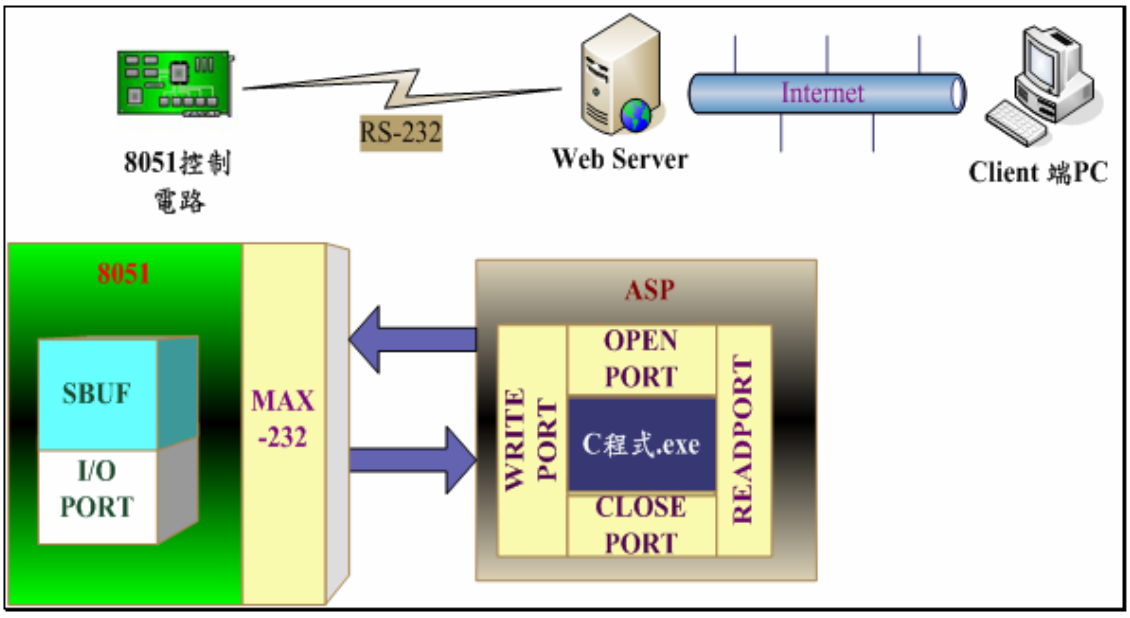


▲ 圖 5.1 問題討論三之錯誤架構圖

於 3.6.2 與 3.6.3 有提到架構修改的問題。原本使用圖 5.1 這種 ASP 呼叫 Web Server 端 DLL 檔的方式，透過 openport()、readport()、writeport()與 closeport()四種函式，將網頁指令資料透過 RS-232 傳到模擬 8051 接收的 PC 端，藉以了解網頁伺服器送出的資料是否正確。而模擬 8051 的 PC 端使用 C 程式的執行檔接收與傳送料，結果卻發現 C 程式的執行檔接收到的資料並非網頁伺服器所傳送的資料，而 C 程式所傳送給使端者端控制網頁的資料也都解讀錯誤。不僅如此，縱使將 Web Server 直接與 8051 連結，8051 中的 SBUF 收到的資料，竟然與 C 程式解讀的資料一樣，都是錯誤的。

解決方法：

以上問題，經由不斷地 try and error 發現，若將模擬 8051 接收端 PC 改使用 VB 呼叫 DLL 檔的方式來接收與傳送資料，則雙方傳送與接收資料正確無誤，並無出現資料解讀錯誤的情形。在查了許多 DLL 相關資料與詢問老師之後終於找出錯誤原因，那就是使用 ASP 呼叫 DLL 時與 C 程式溝通的過程中，此兩種軟體彼此之間對於資料暫存區的定義不同，導致雞同鴨講的情形發生。雖然只要雙方都是透過呼叫 DLL 來溝通即可解決，但是 8051 組合語言無法呼叫 DLL。於是經過了多日的尋找，終於找到解決方法，那就是將 Web Server 端架構改為利用 ASP 呼叫 C 程式執行檔，如此便解決了這個難纏的問題。修正後的架構參考下圖，圖 5.2。



▲ 圖 5.2 問題討論三之修正 Web Server 架構

5.4 心得感想

其實原本專題大方向確定之後，開始尋找論文資料，之後決定用 JAVA 的 Applet 去編寫使用者介面；但是因為以前沒學過 JAVA 導致進度緩慢，而那時也還沒找到整合 java 控制 serial port 的方法，在與瑋隆學長討論過後，決定改變換成以 ASP 網頁為使用者介面利用熟悉的 C 語言與 Windows API 來做。不過當時已浪費不少的時間了，接下來的專題利用暑假加緊趕進度，在尋找合適的程式範例之後，便開始查閱 MSDN 網站上 serial communication in win32 文章的函式與結構的用法，幾乎沒有 C/C++的中文書對 serial port 傳輸有這麼詳細的介紹，讓我感覺到這是很珍貴的資源，所以將文章截取精簡重要之處做成附錄 A。而這整個專題讓我獲益匪淺，就軟體層面來說，感謝瑋隆學長大方向的引導，讓我學到了規劃一個大架構的專題之後，如何穩紮穩打；程式每寫好一個功能便馬上測試，而且先利用虛擬機器做測試，可暫時排除硬體問題，更加速了程式開發的速度，也易於除錯。在學習軟體的深度方面，更學到了如何製作 DLL 檔，也學到了軟體控制硬體比較底層的知識。但是過程中最難忘的是，軟體與硬體結合整個大架構開始 RUN 之後，很多問題就慢慢浮現出來；原本使用 ASP 呼叫 DLL 檔方式在 PC 與 PC 連接的測試時以 VB 呼叫 DLL 檔方式接收沒問題，但是接收方 PC 一用 C 程式接收，卻出現了資料只有 126 或是 127 兩種形式;而且將 PC 接收方換成 8051 硬體，一樣無法辨識接收到的資料。為了這個問題與同學針對很多可能去除錯，找了許多書也上網找了許多資料將近一周仍無法解決，也曾經有很重的失落感。最後詢問袁世一老師，才知道是不同軟體之間呼叫 DLL 時對於軟體與軟體之間資料暫存區的定義不同，所導致這種情況，這對專題來講算是一個重大的變數，因為這個問題導致必須修改專題程式的架構，這也是專題一步步以虛擬機器模擬成功所無法發現的問題。對我來說這算是一種震撼吧，畢竟一步步模擬都成功，而且實際上 PC 使用 VB 呼叫 DLL 檔程式接收也沒問題。不過也讓我學到了寶貴的經驗，實做與理論還是有差距，往往實做就是能找出人考慮不周之處。之後，只好另尋他法，所幸在 ASP 論壇上找到了能以 ASP 呼叫伺服器端應用程式的方法，如此才解決了這個麻煩的問題；而原本不會寫動態網頁的我，也因為此專題學到不少動態網頁的知識。

最後感謝楊豐瑞老師指導與相關設備的支持，包括 8051 燒錄器，電源供應器等硬體設備；特別感謝徐瑋隆學長暑假期間中正與逢甲每週往返指導程式與硬體實做大方向跟除錯觀念，讓我們避免掉一些錯誤，加速專題開發進度。而 DLL 檔連結相關問題與 RS-232 傳輸問題，非常感謝袁世一老師的解惑。最後感謝煥凱同學友情贊助數位相機，協助拍攝專題測試硬體畫面。

第六章 應用與未來展望

一. 應用

遠端醫療照護系統

本專題可應用在醫療設備上，將需要的醫療儀器做整合，集中由一台 Server 做控管。例如醫生或護士需定時替病患作一些例行性的檢查，舉凡量體溫、血壓等等，相當耗費人力資源。因此可以利用遠端控制系統所連結的醫療儀器，定時為病患做檢查；不僅節省人力，而且可將檢查結果傳送到 Server 端資料庫儲存病患資料。系統也可以定時更新病患最新資料，讓醫護人員可以隨時掌握病患最新的狀況，提高整個醫療效率；另一方面，例如 SARS 這類的病患經過空氣或是飛沫接觸及即可傳播病菌，透過此系統的幫助，也可以盡量減少醫護人員與高危險群病患的直接接觸。

資訊網路家電

本專題的實驗架構已經可以控制遠端 8051 I/O 訊號，進而控制 8051 連結之周邊硬體，未來可以與家庭電器做整合。目前透過 8051 算是已經做好多個遠端控制家電插座，透過一些轉換電路或設備，即可輕易控制家電。或者是撰寫一些可供 ASP 呼叫的驅動程式，那麼只要將多個家電驅動程式，集中於 Web Server 管理，即可輕易透過瀏覽器遠端控制家電。例如智慧屋應用(Smart House Applications)，驅動家庭中空調、門窗、燈光、警鈴等開啟或關閉，營造安全、舒適、便利的生活。

二. 未來展望

- 目前本專題的遠端控制權限是由先登入者取得，未來希望搭配存取控制與資料庫，以管制使用者的權限。當以控制者身分登入時，可以觀看並且控制遠端硬體；而非此身分登入者，僅能觀看無法控制硬體。
- 至於系統安全上，未來希望使用者與 Web Server 之間的資料傳輸能使用加密的方式，或是能透過 https 協定進行資料傳輸與 SSL 安全保護機制。而在 Web Server 端則可以設置防火牆，希望能避免系統被駭客所侵入或竊取資訊。
- 未來系統希望可以增加視訊影像的傳輸，藉由攝影機拍攝受控的硬體畫面，

並且傳回使用者端，於瀏覽器中顯示其受控硬體的操作現況；並提供控制拍攝角度的功能。

參考文獻

[1] 范逸之，廖錦棋，Visual Basic .Net 自動化系統監控 RS-232 串列通訊篇，文魁資訊出版社，2004 年 6 月 30 日。

[2] Beck Zaratian原著; 陳威志譯，Microsoft Visual C++6.0 程式開發手冊Microsoft Visual C++6.0 Programer Guide，松崗電腦圖書公司 1998 年 8 月 5 日。

[3] 林金霖，ASP(Active Server Page)實務經典，第三波資訊股份有限公司，1999 年。

[4] 松橋工作室編著，HTML&ASP完美的演繹，知城數位科技出版社，2004 年 4 月 29 日初版。

[5] 蘇俊昇，"Java與Real-Time Linux整合應用於遠端工廠監控之探討"，國立中山大學海下技術研究所，2003。

[6] RS232 Data Interface (a Tutorial on Data Interface and cables). (http://www.arcelect.com/rs232.htm)

[7] Allen Denver，Serial Communications in Win32(Microsoft Windows Developer Support)，December 11, 1995. (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnfiles/html/msdn_serial.asp)

[8] Stepping Sequences for a Four-Phase Unipolar Permanent Magnet Stepper Motor. (http://www.doc.ic.ac.uk/~ih/doc/stepper/control2/sequence.html#single-coil)

[9] Han-Way Huang ,"Using the MCS-51 Microcontroller" ,OXFORD, Dec 1999, In Stock.

[10] 林銘波，微算機基本原理與應用-MCS-51 嵌入式微算機系統軟體與硬體，全華科技圖書股份有限公司，2003 年 10 月初版。

[11] 黃啟芳，馬達電子技術應用: 基礎理論. 應用實務，復漢出版社，2001 年 11 月，一版。

附錄 A. RS-232 相關之 Windows API 函式結構

A.1 CreateFile

範例碼: 對照函式說明

```
com2_handle =
CreateFile("COM2",GENERIC_READ|GENERIC_WRITE,0 ,NULL,OPEN_EXI
STING,FILE_ATTRIBUTE_NORMAL,NULL);
//開啟 com2,設為一般讀取寫入,一般屬性,不使用非同步 IO
```

CreateFile 函式參數意義簡略說明

```
HANDLE CreateFile(
LPCTSTR lpFileName
DWORD dwDesiredAccess
DWORD dwShareMode
LPSECURITY_ATTRIBUTES lpSecurityAttributes
DWORD dwCreationDisposition
DWORD dwFlagsAndAttributes
HANDLE hTemplateFile
);
```

Parameters

lpFileName 對照上面範例的值: "COM2" (要開啓的COM PORT名稱)

[in] A pointer to a null-terminated string that specifies the name of an object to create or open.

dwDesiredAccess 對照上面範例的值:
GENERIC_READ|GENERIC_WRITE (存取模式)

爲何要設這樣->原因請看後面 readfile 與 writefile 介紹部分->因爲使用了 readfile 與 writefile 這 2 個函式,所以 ReadFile 函式條件: GENERIC_READ WriteFile 函式條件: GENERIC_WRITE

[in] The access to the object, which can be read, write, or both.

dwShareMode, 對照上面範例的值: 0 (此時無法COM PORT無法被其他程式共用)

If this parameter is 0 (zero) and CreateFile succeeds, the object cannot be shared and cannot be opened again until the handle is closed. (若開啓 PORT 成功,此時 COM PORT 無法被其他程式共用,也無法再開啓同 1 個 COM PORT, 除非將 handle 關閉)

lpSecurityAttributes　對照上面範例的值:　NULL　(使用預設安全屬性)

[in]　A pointer to a SECURITY_ATTRIBUTES structure that determines whether or not the returned handle can be inherited by child processes.

If lpSecurityAttributes is NULL, the handle cannot be inherited.

If lpSecurityAttributes is NULL, the object gets a default security descriptor.

dwCreationDisposition　對照上面範例的值:　OPEN_EXISTING

[in]　An action to take on files that exist and do not exist. in order to open an existing file (serial port) all we need to know the name of the device (COM1) and pass the creation flags as OPEN_EXISTING.

dwFlagsAndAttributes　:　對照上面範例的值:
FILE_ATTRIBUTE_NORMAL

[in]　The file attributes and flags.

This parameter can include any combination of the file attributes. All other file attributes override FILE_ATTRIBUTE_NORMAL.

When CreateFile opens a file, it combines the file flags with existing file attributes, and ignores any supplied file attributes.

FILE_ATTRIBUTE_NORMAL	A file does not have other attributes set. This attribute is valid only if used

	alone.

hTemplateFile 對照上面範例的值: NULL (使用COM PORT必然固定的設法)

A handle to a template file with the GENERIC_READ access right. The template file supplies file attributes and extended attributes for the file that is being created.

When opening an existing file， CreateFile ignores the template file.

Windows Me/98/95: The hTemplateFile parameter must be NULL.

Return values

If the function succeeds, the return value is an open handle to a specified file.

If the function fails, the return value is INVALID_HANDLE_VALUE.

Communication Resources

The CreateFile function can create a handle to a communications resource, such as the serial port COM1. For communications resources, the dwCreationDisposition parameter must be OPEN_EXISTING, and the template parameter must be NULL. Read, write, or read/write access can be specified, and the handle can be opened for overlapped I/O. For more information about communications, see Communications.

有關 communication 可參考以下網址

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/devio/base/communications_resources.asp

開啓 com port 所必須說明的參數屬性

When the process calls CreateFile to open a communications resource, it specifies the following attributes:

- What type of read/write access exists for the specified resource.
- Whether the handle can be inherited by child processes.
- Whether the handle can be used in overlapped (asynchronous) I/O (非同步)

  operations . (For a description of overlapped operations, see Synchronization.)

開啓 com port 必定遵守的參數配置:

When the process uses CreateFile to open a communications resource(就是指 com port 之類的), it must specify certain values for the following parameters:

- The fdwShareMode parameter must be zero, opening the resource for exclusive access.
- The fdwCreate parameter must specify the OPEN_EXISTING flag.
- The hTemplateFile parameter must be NULL.

有關 dcb 結構: (用來控制"com port 參數"的結構) 詳細結構的定義可參考

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/devio/base/dcb_str.asp

To determine the initial configuration of a serial communications resource, a process calls the GetCommState function, which fills in a serial port DCB structure with the current configuration settings. To modify this configuration, a process specifies a DCB structure in a call to the SetCommState function.

A.2 BuildCommDCB

GetCommState->獲得初始com port設定　SetCommState->修改com port設定

有關BuildCommDCB　(修改dcb參數的方法) 必須include <windows.h>。

例如:BuildCommDCB("baud=4800 parity=N data=8 stop=2",&dcb); //設定傳輸參數結構的內含值:鮑率:4800,無同位檢查,資料位元 8,停止位元 2,其他沒列出的設定皆使用 DCB 結構設定預設值。

The BuildCommDCB function provides another way to modify a DCB structure. BuildCommDCB uses a string with the same form as the command-line arguments of the mode command to specify the baud rate， parity scheme，number of stop bits，and number of data bits. The remaining members of DCB are not changed by this function, except that the appropriate members are set to disable XON/XOFF and hardware flow control. BuildCommDCB only modifies a DCB structure; it does not reconfigure the device.

BuildCommDCB( ) Return Values

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero.

The **BuildCommDCB** function only fills in the members of the **DCB** structure. To apply these settings to a serial port, use the **SetCommState** function.

BuildCommDCB 只是修改 dcb結構的參數值 ，要使這些參數值作用到 com port 需要使用SetCommState 函式 。

函式用法參數詳細說明在:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/devio/base/buildcommdcb.asp

A.3 SetCommState

例如:SetCommState(com2_handle , &dcb); //設定傳輸參數

函式詳細說明網址:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/devio/base/setcommstate.asp

The SetCommState function configures a communications device according to the specifications in a device-control block (a DCB structure). The function reinitializes all hardware and control settings, but it does not empty output or input queues.

```
BOOL SetCommState
(
HANDLE hFile,
LPDCB lpDCB
);
```

Parameters

hFile

[in] Handle to the communications device. The CreateFile function returns this handle.

lpDCB

[in] Pointer to a DCB structure that contains the configuration information for the specified communications device.

Retrun Values

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero.

A.4 Configuring Timeouts

專題部分範例碼互相對照:

```
COMMTIMEOUTS time_out ;   //設定讀寫逾時參數的結構
time_out.ReadIntervalTimeout = MAXDWORD ;
//設爲 MAXDWORD:如果沒有資料供讀取,ReadFile 函式將立即返回
time_out.ReadTotalTimeoutMultiplier = 0 ;
time_out.ReadTotalTimeoutConstant = 0 ;
//不使用讀取總和時間來判斷是否讀取逾時
time_out.WriteTotalTimeoutMultiplier = 5 ;
time_out.WriteTotalTimeoutConstant = 50 ;
//使用寫入總和時間來判斷是否寫入逾時,逾時 WriteFile 函式將立即返回
SetCommTimeouts( com2_handle , &time_out ) ; //設定 com2 讀寫逾時返回
```

COMMTIMEOUTS 結構說明:

The COMMTIMEOUTS structure is used in the SetCommTimeouts and GetCommTimeouts functions to set and query the time-out parameters for a communications device. The parameters determine the behavior of ReadFile, WriteFile, ReadFileEx, and WriteFileEx operations on the device.

```
typedef struct _COMMTIMEOUTS
{
DWORD ReadIntervalTimeout;         (MAXDWORD)
DWORD ReadTotalTimeoutMultiplier;     (0)
DWORD ReadTotalTimeoutConstant;     (0)
DWORD WriteTotalTimeoutMultiplier;     (5)
DWORD WriteTotalTimeoutConstant;     (50)
} COMMTIMEOUTS,
*LPCOMMTIMEOUTS;
```

Parameters

ReadIntervalTimeout	time_out.ReadIntervalTimeout = MAXDWORD ;

//設爲 MAXDWORD:如果沒有資料供讀取,ReadFile 函式將立即返回

Maximum time allowed to elapse between the arrival of two bytes on the communications line, in milliseconds. During a ReadFile operation, the time period begins when the first byte is received. If the interval between the arrival of any two bytes exceeds this amount, the ReadFile operation is completed and any buffered data is returned. A value of zero indicates that interval time-outs are not used.

A value of MAXDWORD, combined with zero values for both the ReadTotalTimeoutConstant and ReadTotalTimeoutMultiplier members, specifies that the read operation is to return immediately with the bytes that have already been received, even if no bytes have been received .

ReadTotalTimeoutMultiplier	time_out.ReadTotalTimeoutMultiplier = 0 ;

Multiplier used to calculate the total time-out period for read operations, in milliseconds. For each read operation, this value is multiplied by the requested number of bytes to be read.

ReadTotalTimeoutConstant	time_out.ReadTotalTimeoutConstant = 0 ;

Constant used to calculate the total time-out period for read operations, in milliseconds. For each read operation, this value is added to the product of the ReadTotalTimeoutMultiplier member and the requested number of bytes.
A value of zero for both the ReadTotalTimeoutMultiplier and ReadTotalTimeoutConstant members indicates that total time-outs are not used for read operations. //不使用讀取總和時間來判斷是否讀取逾時

WriteTotalTimeoutMultiplier　time_out.WriteTotalTimeoutMultiplier = 5 ;

Multiplier used to calculate the total time-out period for write operations, in milliseconds. For each write operation, this value is multiplied by the number of bytes to be written.

WriteTotalTimeoutConstant　time_out.WriteTotalTimeoutConstant = 50 ;

//使用寫入總和時間來判斷是否寫入逾時,逾時 WriteFile 函式將立即返回

Constant used to calculate the total time-out period for write operations, in milliseconds. For each write operation, this value is added to the product of the WriteTotalTimeoutMultiplier member and the number of bytes to be written.

A value of zero for both the WriteTotalTimeoutMultiplier and WriteTotalTimeoutConstant members indicates that total time-outs are not used for write operations .

更詳細的參數說明參考網頁

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/devio/base/commtimeouts_str.asp

A.5 SetCommTimeouts

SetCommTimeouts(com2_handle , &time_out) ;
//設定 com2 讀寫逾時返回

The SetCommTimeouts function sets the time-out parameters for all read and write operations on a specified communications device.

BOOL SetCommTimeouts
(
HANDLE hFile,

LPCOMMTIMEOUTS lpCommTimeouts
);

Parameters

hFile

[in] Handle to the communications device. The CreateFile function returns this handle.

lpCommTimeouts

[in] Pointer to a COMMTIMEOUTS structure that contains the new time-out values.

Return Values

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero.

A.6 ReadFile

ReadFile(com2_handle , buffer , read_bytes ,&read_bytes ,NULL);

//由 com2 讀取 1byte .

//因爲 ReadFile 函式逾時的時候,不僅沒讀到資料(所以 buffer[0]沒變動)就返回,還會把 read_bytes 內含值改爲 0,因此 read_bytes!= 0 時表示有讀到資料

BOOL ReadFile
(
HANDLE hFile,
LPVOID lpBuffer,
DWORD nNumberOfBytesToRead,
LPDWORD lpNumberOfBytesRead,
LPOVERLAPPED lpOverlapped,
);

函式參考網頁:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/fileio/fs/readfile.asp

Parameters

hFile	對照範例	com2_handle

[in] A handle to the file to be read.

The file handle must be created with the GENERIC_READ access right.

lpBuffe r	對照範例	buffer

[out] A pointer to the buffer that receives the data read from a file(PORT).

nNumberOfBytesToRead	對照範例	read_bytes

[in] The number of bytes to be read from a file(PORT).

lpNumberOfBytesRead	對照範例	&read_bytes

[out] A pointer to the variable that receives the number of bytes read.

//因爲 ReadFile 函式逾時的時候,不僅沒讀到資料(所以 buffer[0]沒變動)就返回,

//還會把 read_bytes 內含值改爲 0,因此 read_bytes!= 0 時表示有讀到資料

lpOverlapped	對照範例	NULL

[in] A pointer to an OVERLAPPED structure.

This structure is required if hFile is created with FILE_FLAG_OVERLAPPED.

(在 CreateFile 函式中設定爲 FILE_ATTRIBUTE_NORMAL)->

所以 lpOverlapped　　對照範例　NULL

Retrrn Values

The ReadFile function returns when one of the following conditions occur:

- A write operation completes on the write end of the pipe.
- The number of bytes requested is read.
- An error occurs.

If the function succeeds, the return value is nonzero.

If the function fails, the return value is 0 (zero).

A.7 WriteFile

WriteFile(com2_handle,&key,read_bytes,&read_bytes,NULL);

BOOL WriteFile
(
HANDLE hFile
LPCVOID lpBuffer
DWORD nNumberOfBytesToWrite
lpNumberOfBytesWritten
LPOVERLAPPED lpOverlapped
);

Parameters

hFile　對照範例com2_handle

[in] Handle to the file. The file handle must have been created with the

GENERIC_WRITE access right.

lpBuffer　對照範例　&key　(key 可看成是寫入 port 的緩衝區變數)

[in] Pointer to the buffer containing the data to be written to the file.

nNumberOfBytesToWrite 對照範例 read_bytes

[in] Number of bytes to be written to the file. (1 次寫入 port 的 byte 數)

lpNumberOfBytesWritten, 對照範例 &read_bytes

[out] Pointer to the variable that receives the number of bytes written

lpOverlapped對照範例 NULL

[in] Pointer to an OVERLAPPED structure.

This structure is required if hFile is created with FILE_FLAG_OVERLAPPED.

(在 CreateFile 函式中設定爲 FILE_ATTRIBUTE_NORMAL)

->所以 lpOverlapped 對照範例 NULL

補充: stdio.h 中，或是 conio.h 中的輸入函式 scanf() gets() getchar() getch() 等等函式都有一個共同的表現： 呼叫過該函式後，必須等到操作的人鍵入一些資料以後，函式才會回返到呼叫點繼續執行下去。

使用 conio.h 中的 kbhit() 及 getch() 兩個函式，呼叫 kbhit() 函式之時，不管操作者有沒有按鍵，該函式都會立刻執行完畢，傳回一個數值，傳回 0 代表沒有任何按鍵被按下，傳回 1 代表有按鍵被按下，通常使用如下的程式來讀取按鍵值。

```
if (kbhit()) c = getch();
```

那爲何要用 kbhit()函式? 是爲了解決底下的問題：

如果在程式執行到了讀取按鍵的時候，操作的人沒有按下任何按鍵的話，程式就停在那兒等待。

Kbhit()函式參考說明：

http://ciips.ee.uwa.edu.au/~morris/Year1/CLP110/Labs/kbhit.html

附錄 B. 8051 遠端控制指令判斷程式碼

```
        ORG     00H             ;初始設定
START:  JMP     MAIN            ;
        ORG     30H             ;
MAIN:   MOV     SP,#5FH         ;
;-----------------------------------------;
        CALL    INIT_RS232  ;呼叫 INIT_RS232 副程式，設定串列埠的通訊
協定
        CPL     P0.0                ;P0.0 亮，表示正在設定串列埠
        MOV     R5,#25          ;設定 R5=25，延遲 25*20ms = 500ms = 0.5 秒
        CALL    DELAYms         ;呼叫 DELAYms 副程式
        CPL     P0.0                ;P0.0 暗，表示設定完成
;-----------------------------------------;
WAIT:   JNB     RI,WAIT         ;等待程式，檢查 RI 是否爲 1?RI=1 表示收到資
料
        CPL     P0.1                ;P0.1 亮，表示正在接收，開始比對
        CLR     RI              ;清除 RI 以便繼續接收
        MOV     A,SBUF          ;從串列埠讀入資料
;-----------------------------------------;
;------------SELECT CASE------------;以下比對 A(SBUF)的資料
;--------------CASE "A"-----------------;
MA:
        CJNE    A,#41H,MB   ;CASE-A，收到"字元 1"往下執行，否則跳至
MB
        CPL     P0.2                ;P0.2 亮
```

```
        JMP     MEND        ;執行結束，跳至 MEND
;-------------CASE "B"-----------------;
MB:
        CJNE    A,#42H,MC   ;CASE-B，收到"字元 2"往下執行，否則跳至
MC
        MOV     R1,#1       ;設定 R1=1，執行步進馬達正轉副程式一次
MBLOOP:
        CALL    STEPRIGHT ;呼叫步進馬達正轉副程式
            DJNZ    R1,MBLOOP ;R1-1，判斷 R1 是否爲 0?跳至 MBLOOP，
                              ;執行迴圈一次
        JMP     MEND        ;執行結束，跳至 MEND
;---------------CASE "C"----------------;
MC:
        CJNE    A,#43H,MD   ;CASE-C，收到"字元 3"往下執行，否則跳至
MD
        MOV     R1,#1       ;設定 R1=1，執行步進馬達反轉副程式一次
MCLOOP:
        CALL    STEPLEFT    ;呼叫步進馬達反轉副程式
        DJNZ    R1,MCLOOP ;R1-1，判斷 R1 是否爲 0?跳至 MCLOOP，
                            ;執行迴圈一次
        JMP     MEND        ;執行結束，跳至 MEND
;--------------CASE "D"----------------;
MD:
        CJNE    A,#44H,ME   ;CASE-D，收到"字元 4"往下執行，否則跳至
ME
        MOV     R1,#10      ;設定 R1=10，執行閃爍程式 10 次
```

```
MDLOOP:
        CPL     P0.3            ;P0.3 每 0.5 秒閃爍，共 10 次
        MOV     R5,#25          ;設定 R5=25，延遲 25*20ms= 500ms = 0.5 秒
        CALL    DELAYms         ;呼叫 DELAYms 副程式
        CPL     P0.3            ;
        MOV     R5,#25          ;設定 R5=25，延遲 25*20ms= 500ms = 0.5 秒
        CALL    DELAYms         ;
        DJNZ    R1,MDLOOP       ;R1-1，判斷 R1 是否爲 0?跳至 MDLOOP，
                                ;執行迴圈 10 次
        JMP     MEND            ;執行結束，跳至 MEND
;----------------CASE "E"----------------;
ME:
        CJNE    A,#45H,MEND;CASE-E，收到"字元 5"往下執行，否則跳至
MEND
        MOV     R1,#10          ;設定 R1=10，執行閃爍程式 10 次
MELOOP:
        CPL     P0.4            ;P0.4 亮，每 1 秒閃爍，共 10 次
        MOV     R5,#50          ;設定 R5=50 延遲 50*20ms= 1000ms = 1 秒
        CALL    DELAYms         ;呼叫 DELAYms 副程式
        CPL     P0.4            ;
        MOV     R5,#50          ;設定 R5=50 延遲 50*20ms= 1000ms = 1 秒
        CALL    DELAYms         ;
        DJNZ    R1,MELOOP       ;R1-1，判斷 R1 是否爲 0?跳至 MELOOP，
                                ;執行迴圈 10 次
        JMP     MEND            ;執行結束，跳至 MEND
;--------------CASE ELSE--------------;如果不是上面之數據，則跳到 MEND 選項，
```

```
                                ;將 P0.1 滅，表示接收完畢
MEND:                           ;比對結束
        CPL     P0.2                ;
        MOV     R5,#50          ;設定 R5=50，延遲 50*20ms = 1000ms = 1 秒
        CALL    DELAYms         ;呼叫 DELAYms 副程式
        CPL     P0.1                ;P0.1 暗，表示程式比對結束
        JMP     WAIT            ;跳回 WAIT 程式繼續等待
;-------------CASE END-------------;
;----------------------------------------;
INIT_RS232:                     ;RS232 轉換副程式
        MOV     TMOD,#20H ;設定計時器 1 模式 1，自動載入功能計時器
        MOV     TH1,#0E8H   ;設定鮑率爲 1200bps
        SETB    TR1             ;啓動計時器 1
        MOV     SCON,#01010000B;設定傳輸協定爲模式 1，
                                ;8 個資料位元，鮑率由 Timer1 決定
        RET                     ;
;----------------------------------------;
STEPRIGHT:                      ;步進馬達正轉副程式，一個步進角
        MOV     R5,#1           ;設定延遲時間爲 5ms
        MOV     P1,#11111110B;使(P1.0=0)，P1.1=1，P1.2=1，P1.3=1
                                ;第一支腳位 P1.0 輸出爲 0，激磁
        CALL    DELAYus             ;
        MOV     P1,#11111101B;使 P1.0=1，(P1.1=0)，P1.2=1，P1.3=1
                                ;第二支腳位 P1.1 輸出爲 0，激磁
        CALL    DELAYus             ;
        MOV     P1,#11111011B;使 P1.0=1，P1.1=1，(P1.2=0)，P1.3=1
```

```
                        ;第三支腳位 P1.2 輸出爲 0，激磁
        CALL    DELAYus         ;
        MOV     P1,#11110111B;使 P1.0=1，P1.1=1，P1.2=1，(P1.3=0)
                        ;第四支腳位 P1.3 輸出爲 0，激磁
        CALL    DELAYus         ;
        RET                     ;
;-----------------------------------------;
STEPLEFT:               ;步進馬達反轉副程式，一個步進角
        MOV     R5,#1           ;設定延遲時間爲 5ms
        MOV     P1,#11110111B;使 P1.0=1，P1.1=1，P1.2=1，(P1.3=0)
                        ;第四支腳位 P1.3 輸出爲 0，激磁
        CALL    DELAYus         ;
        MOV     P1,#11111011B;使 P1.0=1，P1.1=1，(P1.2=0)，P1.3=1
                        ;第三支腳位 P1.2 輸出爲 0，激磁
        CALL    DELAYus         ;
        MOV     P1,#11111101B;使 P1.0=1，(P1.1=0)，P1.2=1，P1.3=1
                        ;第二支腳位 P1.1 輸出爲 0，激磁
        CALL    DELAYus         ;
        MOV     P1,#11111110B;使(P1.0=0)，P1.1=1，P1.2=1，P1.3=1
                        ;第一支腳位 P1.0 輸出爲 0，激磁
        CALL    DELAYus         ;
        RET                     ;
;-----------------------------------------;
DELAYms:                        ;延遲副程式
; t={1+[100*(1+100*2)]+2+2+2}=20107us=20.107ms
        MOV     R6,#100     ;藉由計算 R6，R7 的迴圈，產生延遲時間 20ms
```

```
DELAYm :MOV    R7,#100        ;
        DJNZ    R7,$                ;
        DJNZ    R6,DELAYm ;
        DJNZ    R5,DELAYms      ;總延遲時間由 R5 決定，時間爲
(R5*20ms)，
                                ;IF R5=5，則 5*20ms=0.1 秒
        RET                     ;
;-----------------------------------------;
DELAYus:                        ;延遲副程式
;t={1+[50*(1+50*2)]+2+2+2}=5057us=5.057ms
        MOV     R6,#50          ;藉由計算 R6，R7 的迴圈，產生延遲時間 5ms
DELAYu: MOV     R7,#50              ;
        DJNZ    R7,$                ;
        DJNZ    R6,DELAYu   ;
        DJNZ    R5,DELAYus ;總延遲時間由 R5 決定，時間爲(R5*5ms)，
                                ;IF R5=1，則 1*5ms=0.005 秒
        RET                     ;
;-----------------------------------------;
        END                             ;程式結束
;-----------------------------------------;
```