逢甲大學

資訊工程學系專題報告

非觸碰式感應卡門禁系統

專題學生： 陳坤誠
指導教授： 徐弘洋老師

中華民國九十年十月

目錄：

圖表目錄:

第 0 章：專題實驗的目地、動機。

0.1 實驗動機

現今都市大樓林立，所以大樓的安全管理越來越被人們所重視，治安好壞與大樓管理的制度成為人們居住工作的必要考慮之一，但是一般利用人力來管理大樓或社區的人員出入，不免會顯的浪費人力與效果不彰顯，所以如果能利用一套電子系統來管理大樓人員的進出就顯得有效率多了，並且可以省下一筆可觀的人事費用，而且可以避免掉一些人為疏失的情形發生，並且可以做人員進出的時間紀錄，當有事故案件發生時，可以調出出勤紀錄，對當時段進出大樓的人員做調查，比起人力管理大樓訪客紀錄簿，來的確實且有效率許多。

現今社會上對於非接觸式的感應卡運用非常的廣泛，而非接觸式的感應卡又可以分為很多種；感應卡、條碼卡、磁條卡、IC 卡等等皆是，而其用途功能也多不大相同，例如條碼卡多運用在超商的商品紀錄，利用紅外線感測器就可以辨識商品本身價格，而磁條卡則被運用在大眾運輸系統，例如捷運月卡；IC 卡則被運用在信用卡或是電話卡上，因其有可重複讀寫的功能，且不會被變造複製；但是相對於感應卡，IC 卡片的編碼保密就顯的重要多了，所以這次專題就是運用到感應卡本身不易被複製，而且卡片也不容易出現到重複的特性，來進行身分確認的工作，以對大樓進出人員做管制，簡單有效的管理大樓人員出入，加上可以連接到個人電腦，傳輸資料至電腦，再以 JAVA 撰寫的簡易資料庫存取的應用程式便可對出入人員的時間做紀錄，使得安全性更高，有效的掌控人員進出的時刻，運用在學校或公司，也可當作學生或是員工的出勤紀錄器，可以清楚了解是否有學生人員曠課或是缺席。

0.2 實驗目的

這個實驗的目的是讓我們了解到下列幾點的運用：

＊LCD 顯示模組的運用

＊串列傳輸介面的運用

＊RS232 串列介面輸出入電位的轉換

＊非接觸式感應卡資掉的讀取及其應用

第一章：非觸碰式感應卡門禁系統介紹

1.1 什麼是非觸碰式感應卡

目前市面上應用叫普遍的感應卡有兩種形式：

第一種是 MTP 型的，允許資料重複寫入多次，一般拿來當作系統上的儲值卡運用，例如大眾捷運系統的儲值月卡。

第二種為 OPT 型，只能寫入一次資料；一般是拿來當作身分識別的卡片，如門禁管制，出勤管理等等。(表 1.1 卡片的比較)

比較項目	磁卡	條碼卡	感應卡	ＩＣ卡	非接觸式ＩＣ卡
讀取方式	能單一方向等速刷卡，卡片與讀取頭需接觸易磨損。	僅能單一方向等速刷卡，卡片與讀取頭不需接觸。	不怕物體阻隔，讀卡距離至少 8cm 以上。	需作插卡式動作，晶卡與讀取頭接觸。	不怕物體阻隔，讀卡距離至少 8cm 以上。
讀卡時間	約 1 秒	約 1 秒	約 1 秒	約 0.5 秒	約 0.5 秒
卡片壽命	壽命較短	磁帶易消磁、刮損	壽命較短易刮損	永久使用	永久使用
卡片安全	可使用錄碼機複製或變造	可使用印表機、影印機複製	不易偽造	無法偽造	無法偽造
卡片價格	低	低	高	高	高
維護費用	磁頭易磨損需更換	易受灰塵影響需常保養	毋需耗材、保養	需常保養	毋需耗材、保養
卡片讀/寫	讀	讀	讀	讀/寫	讀/寫

A.感應卡（薄卡、厚卡）

內部構造為感應線圈，不易損壞，缺點是成本高且須製作不易需工廠訂製。

B. 磁條卡

卡號無重複性，成本低廉；缺點是卡片容易消磁及被仿製，刷卡機磁頭為耗損品。

C. 條碼卡

易於自行製作，成本低廉；缺點是條碼不清時感測不易及被仿製，紅外線感測器是耗材。

D. IC 卡

IC 可存取資料，卡號無重複性，安全性高不易被仿製；缺點是製作不易成本高需由工廠代製。

（圖 1.1 常見卡片的樣式）

這個專題便是以 OPT 型的感應卡製作，這個專題使用的是第一種感應卡來製作，該卡分為薄卡跟厚卡，其內部含有感應線圈；其感應卡資料長度有 64 個位元，故其內碼多達 2 的 64 次方種變化，所以理論上會重複的機會是微乎其微，所以拿來對大樓人員出入的控管較為適當，當人員拿代表其身份的感應卡，經由感應頭感應確認其身份後，才允許該名人員進出，並經由電腦的資料庫記錄該名人員的出入時間，對人員出勤及大樓安全做有效的管理。

1.2 非觸碰式感應卡門禁系統特色

使用非接觸式感應卡來管理人員出入，其最大的特色就是有別於傳統大樓的管理方式，傳統來說，大樓管理員可能要認識所有的住戶才可能管理人員的進出，如果人數一多，可能就要採取登記身份的方式來過濾進出的人員，這樣對於進出人員頻繁的大樓或是社區而言，都是非常的不方便而且耗費人力，所以這個系統就是針對這個缺點進行改善，利用感應卡重複性不高的特色，來對為數眾多的人員進行身分確認，每個人都有每個人專屬的身分識別卡，符合身分者才得以進出，經由這套系統的管理，這樣一來可以使得大樓管理變得更簡單且有效率了。另一特色便是感應不用接觸到讀卡機，可以提高使用期限，將其放置於包包內只要阻隔距離不大，依然可以被感測到，這點提供了很大的便利性。

1.3 非觸碰式感應卡門禁系統原理

1.3.1 RFID 概述

「無線射頻身份辨別系統」，英文名稱為 Radio Frequency Identification System，簡稱 RFID，是針對接觸式系統的缺點，利用射頻訊號以無線方式傳送數碼資料，因此識別卡不需與讀卡機接觸即可做資料的交換。此種無線方式的資料傳送並無方向性的要求，且卡片可以置於口袋、皮包內，不必取出就能直接辨識，免除現代人經常要從數張卡片中要尋求特定卡片的煩惱，所以增加更多使用上的便利性。利用射頻電路訊號以無線方式傳送數位資料
在無線射頻身份辨別系統中，射頻電路主要由兩大部份功能組成。一是利用射頻訊號充電，另一為利用射頻訊號以負載調變方式進行數據收發(ASK 方式)。

1.3.2 利用 RFID 特性製作感應門禁系統

這個系統是以 RFID 中 ASK 方式來讀取感應卡上的資料，讀卡機讀取到卡片資料時經由 8951 單晶片顯示在 LCD 上，同時經由 RS232 的傳輸機制把資料傳送給電腦並在電腦上進行存取，這些檔案透過電腦的資料庫程式管理，這筆資料可當作日後出勤紀錄的比對，而由卡片上讀取的資料也可來進行比對，當以上鎖的大門開關是否應該打開，來達成門禁管理的效果。

第二章：MCS-51 單晶片介紹

2.1 單晶片簡介

MSC-8051 系列單晶片是美國INTEL公司推出MCS-8048系列晶片之後所推出的後續晶片，8051系列單晶片主要是改進8048系列單晶片的硬體架構及軟體能力。MCS-8051系列單晶片依其電路結構又可分為三種版本：(1)晶片內部不含ROM的版本、(2)晶片內含ROM的版本、以及(3)晶片內含EPROM的版本。下表列出各版本晶片的編號及一些特性：

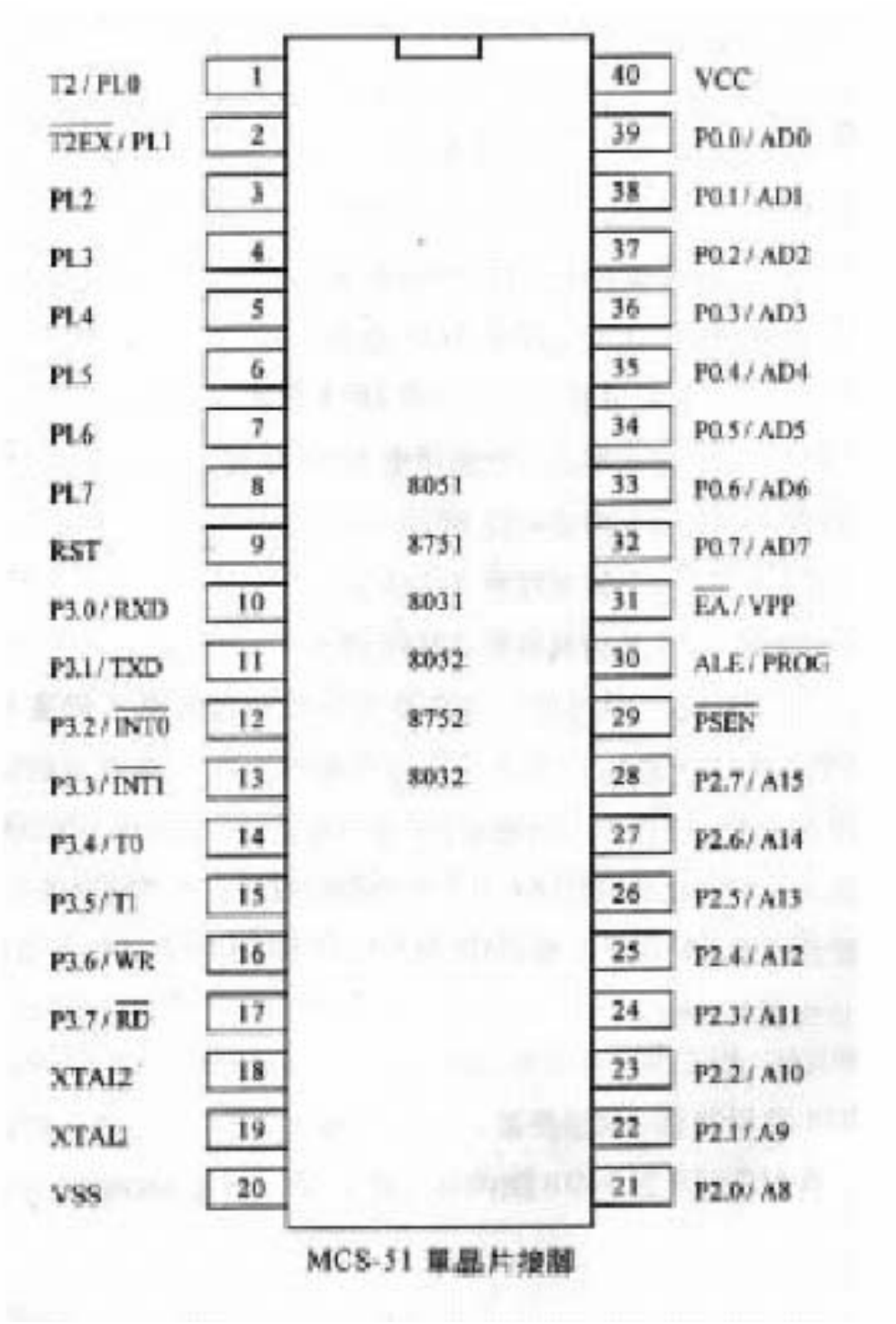
ROM 版本	EPROM 版本	ROMless 版本	ROM(Btyes)	RAM(Bytes)	16 位元計時器	電路型態
8051	8751	8031	4K	128	2	HMOS
8052	8752	8032	8K	256	3	HMOS
80C51	87C51	80C31	4K	128	2	CHMOS

(表2.1 MCS-51系列版本比較表)

以下將8051/8751/8031單晶片的重要特性說明如下：

＊單晶片8位元微電腦晶片。
＊4K Bytes 的內部程式記體(8031 除外)。
＊128 Bytes 可供讀/寫的內部RAM。
＊可在外部擴充到64K Bytes 程式記憶體及64K Bytes 資料記憶體。
＊12 組16位元的計時器/計數器(Timer/Counter)。
＊74組8位元的I/O並列埠，共32條可單獨規劃為輸入或輸出的I/O點。
＊1組全雙工的串列埠，可連接RS-232等標準的串列通信介面。
＊可擴充為128K Bytes 的外部記憶體，其中64K Bytes 為程式記憶體，另外64K Bytes 為資料記憶體。
＊可處理5 個中斷來源，並可規劃為2 層中斷優先權。
＊內部具有時脈振盪器，最高工作時脈可達12 MHz。

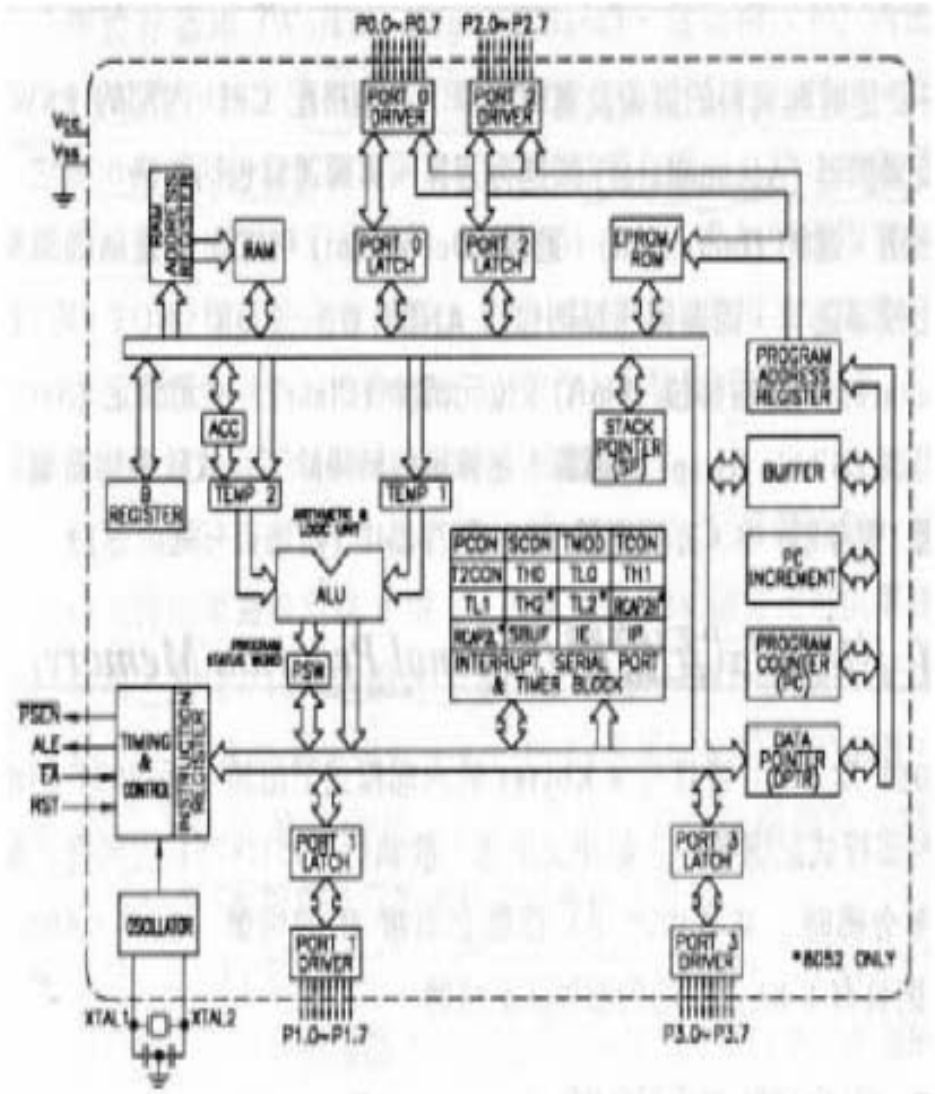
圖為MSC51的接腳示意圖:



（圖2.1 MCS-51接腳示意圖）

2.2 MCS-8051 單晶片系統架構

下圖為8051系列單晶片之內部功能方塊圖:



(圖 2.2 8051 系列單晶片之內部功能方塊圖)

以下為幾個重要的功能說明：

2.2.1 指令解碼器(Instruction Decoder)

指令解碼器的功能是負責將擷取自程式記憶體中的指令加以解碼，並送出內部信號以控制單晶片各單元的動作。這些內部控制信號可控制資料的來源及目的地，以及控制算術/邏輯處理單元的功能。

2.2.2 程式計數器(Program Counter,PC)

程式計數器是用來控制儲存在程式記憶體中指令碼的執行順序的一個16位元暫存器，也就是說，程式計數器的存放內容是下一個執行指令的程式記憶體的位址(A0~A15), 其中低階位址A0~A7 存放在8位元的PCL暫在器中，而高階位址

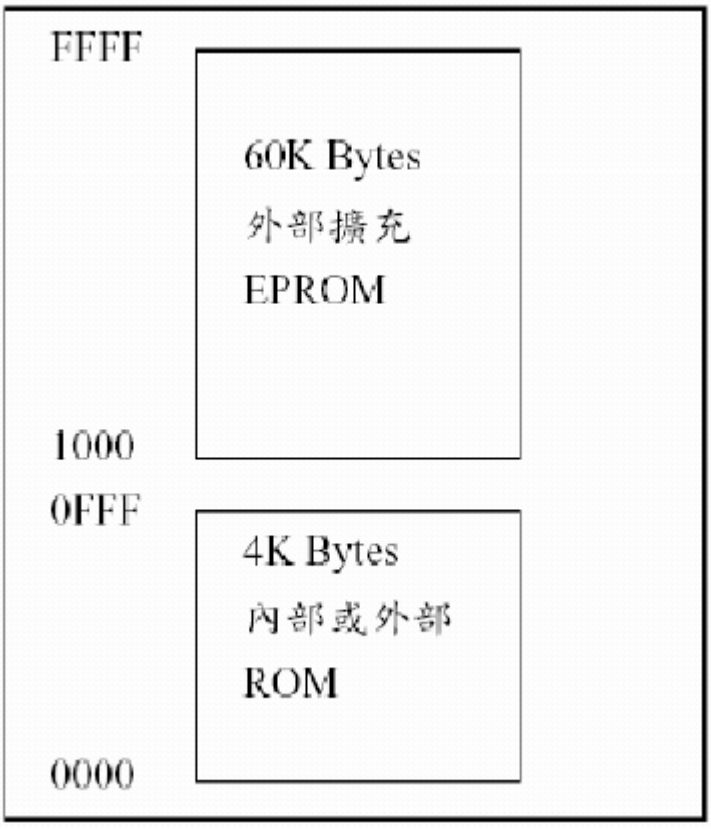
A8~A15則存放在8位元的PCH暫存器中。

2.2.3 算術術/邏輯單元(Arithmetic & Logic Unit,ALU)

這個單元主要是處理資料的算術及邏輯運算，必須搭配單晶片內部的PSW暫存器及累加器來進行運算。算術運算包括加法、減法、乘法、除法、遞加、遞減、大小比較等運算，而邏輯運算則包括AND、OR、XOR、NOT、左/右旋轉(Rotate)、在/右移位(Shift)、位元清除(Clear)、位元設定(Set)等運算。運算後的結果除了存放於累加器或其它記憶體(暫存器)中，亦反應於PSW暫存器中(如是否有進位等)。

2.2.4 程式記憶體(Program Memory)

8051 及8071 皆具有4K Bytes 的內部程式記憶體，並可在外部再擴充60K Bytes EPROM，如圖3.3所示，而8031則沒有這些內部程式記憶體。在程式記憶體中所存放的是8051所要執行的程式碼,單晶片會主動到這塊記憶體要執行的指令碼，而8051要讀取程式記體時需激發信號PSEN。



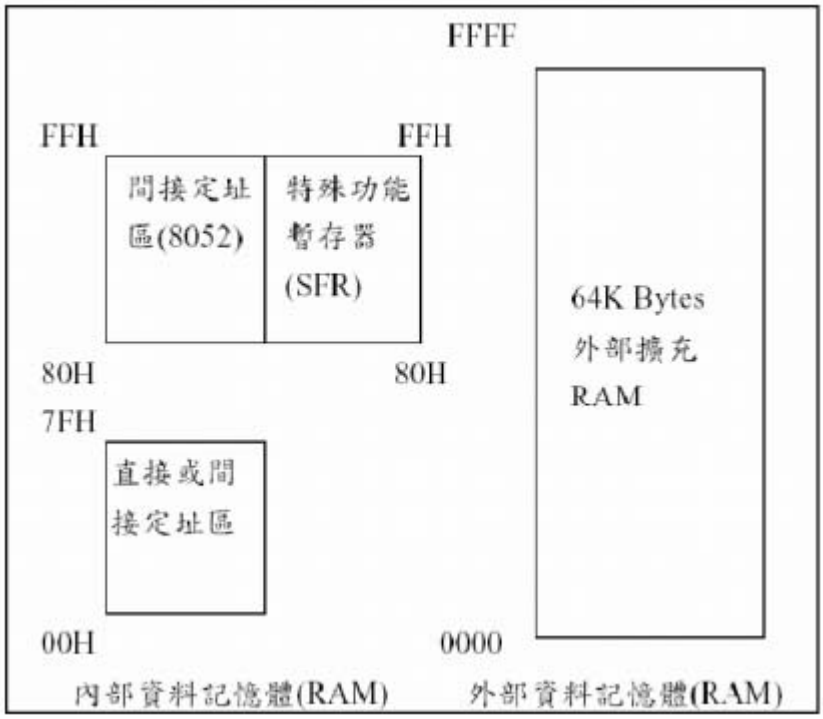
(圖3.3 8051程式記憶體結構圖)

8051是如何決定程式記憶體的前面4K Bytes要內部或外部程式記憶體去讀取指令呢?這就要靠8051的EA接腳來決定內部程式記憶體是否有效，當EA=0，代表內部程式記憶體無效，8051會將前面4K移到外部；當EA=1，則內部程式記憶體有效。

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 讀取外部程式記憶體時會固定輸出位址匯流排的高位元組(A8~A15)。

2.2.5 資料記憶體(Data Memory)

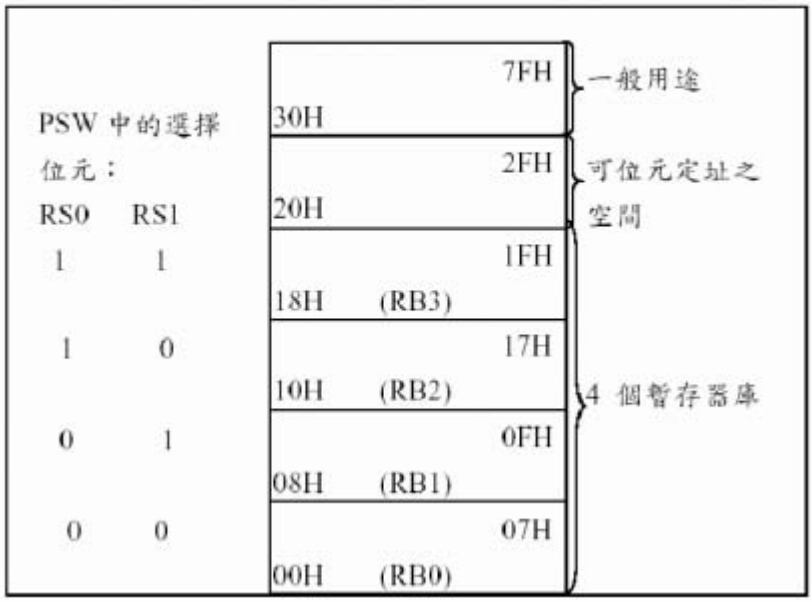
8051 內部有一塊256 個Byte的位址空間，這塊空間是存放資料記憶體(RAM)和特殊功能暫存器(SFR)的地方，並可在外部擴充64KBytes的資料記憶體。其資料記憶體的結構圖如下：



(圖2.4 8051資料記憶體結構圖)

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。這些內部資料記憶體可供使用者的程式自由存取資料，不過，00H~7FH記憶體的資料可用直接定址法來存取資料，而8052系列的80H~FFH記憶體的資料則必須間接定址法才可以存取。依單晶片的特性又可將這些內部資料記憶體(00H~7FH)分成三個不同的部分：

＊暫存器庫(Register Banks)
＊可位元定址(Bit-addressable)區
＊一般用途區



（圖 2.5 內部資料記憶體的較低 128 位元組）

2.2.6 特殊功能暫存器(Special Function Register，SFR)

8051系列單晶片單晶片內部有一塊128 Bytes 可直接定址的記憶體區，其直接定址位址為80H~FFH，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)。下表說明特殊功能暫存器的名稱與各記憶體之間的關係。

F8H									FFH
F0H	B								F7H
E8H									EFH
E0H	ACC								E7H
D8H									DFH
D0H	PSW								D7H
C8H	T2CON		PCAP2L	PCAP2H	TL2	TH2			CFH
C0H									C7H
B8H	IP								BFH
B0H	P3								B7H
A8H	IE								AFH
A0H	P2								A7H
98H	SCON	SBUF							9FH
90H	P1								97H
88H	TCON	TMOD	TL0	TL1	TH0	TH1			8FH
80H	P0	SP	DPL	DPH				PCON	87H

（表2.2 特殊功能暫存器說明）

以下將說明SFR中各暫存器的功能及用途：

(1) 累加器(Accumulator，ACC)：
累加器又可稱之為ACC或A暫存器，這是一個使用頻率頗高的一個
通用暫存器，而有許多指令是以其為操作對象。
(2) B暫存器：
在做乘法指令(MUL)及除法指令(DIV)運算時，必須以B暫存器
為操作對象之一。也就是說，在做乘法/除法運算之前必須將運算資放入A及B暫存器中，而運算之後的結果會放入A，B暫存器中。

(3) 程式狀態字語(Program Status Word，PSW)暫存器：
PSW暫存器主要是記錄及控制單晶片之運算。如下表所示：

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

(表 2.3 PSW 暫存器)
＊CY(PSW.7)：進位旗標。
＊AC(PSW.6)：輔助進位旗標。
＊F0(PSW.5)：一般用途旗標。
＊RS1(PSW.4)：暫存器庫選擇位元1。
＊PS0(PSW.3)：暫存器庫選擇位元0。
＊OV(PSW.2)：溢位旗標。
＊----(PSW.1)：保留。
＊P(PSW.0)：同位旗標。
(4) 堆疊指標器(Stack Pointer，SP)：

SP是管理堆疊的一個暫存器，用來指出最近一次資料推入(Push)堆疊時的內部資料記憶體位址。每次執行PUSH指令時，SP值自動加一，然後再將資料推入堆疊中；反之，執行POP指令時，資料先彈出(Pop)堆疊後，SP再自動減一。另外執行副程式呼叫指令(CALL)或中斷時，程式計數器(PC)的值亦會推入堆疊中，而執行副程式返回指
(RET/RETI)時會將堆疊內的資料回存到程式計數器中，以正確地返回到原程式的呼叫點。
(5) 資料指標器(Data Pointer，DPTR)：
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)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。
(6) 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。
(7) 中斷優先權(Interrupt Priority，IP)暫存器：
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。IP暫存器的格式
如下表所示：

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

(表2.4 IP暫存器)
＊---- (IP.7)：保留。
＊---- (IP.6)：保留。
＊PT2(IP.5)：設定Timer2之中斷優先順序(8052 用)。
＊PS(IP.4) ：設定串列埠之中斷優先順序。
＊PT1(IP.3)：設定Timer1 之優先順序。
＊PX1(IP.2)：設定外部中斷INT1 之優先順序。
＊PT0(IP.1)：設定Timer0 之優先順序。
＊PX0(IP.0)：設定外部中斷INT0 之優先順序。
(8) 中斷致能(Interrupt Enable，IE):
由於所有的中斷皆為可遮罩的(Maskable)，這些中斷就是由IE暫存器來加以致
能/除能(Enable/Disable)的，其格式如下表：

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

(表2.5 IE暫存器)
＊EA (IE.7)：EA=0時，所有中斷除能。
＊EA=1 時，各中斷之產生由個別的致能位元決定。
＊---- (IE.6)：保留。
＊ET2 (IE.5)：致能Timer2之中斷(8052 用)。
＊ES (IE.4)：致能/除能串列埠之中斷。
＊ET1(IE.3)：致能Timer1之中斷。
＊EX1(IE.2)：致能外部中斷INT1 之中斷。
＊ET0(IE.1)：致能Timer0之中斷。
＊EX0(IE.0)：致能外部中斷INT0 之中斷。
(9) TH0~TH2、TL0~TL2計時器/計數時暫存器：
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)。

(10) 計時器模式控制(Timer/Counter Mode Control，TMOD)暫存器：

位元	7	6	5	4	3	2	1	0
TMOD	GATE	C/T	M1	M0	GATE	C/T	M1	M0
計時器1					計時器0			

(表2.6 TMOD暫存器)

＊GATE：計時器動作閘控位元，當GATE=1 時，INT0 或INT1 接腳為高電位，同時TCON中的TR0 或TR1控制位元為1時，計時/計數器0 或1 才會動作。若GATE=0，則只要將TR0或TR1控制位元設為1，計時/計數器0 或1 即可動作。

＊C / T ：做計時器或計數器功能之選擇位元。C/T=1為計數器，由外部接腳T0或T1輸入計數脈波。C/T=0為計時器，由內部系統時脈提供計時工作脈波。

＊M1：模式選擇位元1。

＊M0：模式選擇位元0。

M1	M0	動作模式
0	0	13位元計時/計數器
0	1	16位元計時/計數器
1	0	8位元自動載入計時/計數器
1	1	計時器1停止工作，計時器0分為兩個獨立的8位元計時器TH0及TL0

(表2.7 計時器的選擇位元表)

(11) 計時器控制(Timer Control，TCON)暫存器：

位元	7	6	5	4	3	2	1	0
TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IR1

(表2.8 TCON暫存器)

＊TF1(TCON.7)：計時器1溢位旗號，當計時溢位時，由硬體設定為1，在執行過相對的中斷服務常式後則自動清除為0。

＊TR1(TCON.6)：計時器1 啟動控制位元，可以由軟體來設定定或清除。

＊TF0(TCON.5)：計時器0溢位旗號，當計時溢位時，由硬體設定為1，在執行過相對的中斷服務常式後則自動清除為0。

＊TR0(TCON.4)：計時器0 啟動控制位元，可以由軟體來設定定或清除。

＊IE1(TCON.3)：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，則消除為0。
＊IT1(TCON.2)：外部中斷1動作型態選擇，當IT1=1時，中斷型態為負緣觸發，當IT1=0 時，中斷型態則為低準位觸發。
＊IE0(TCON.1)：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，則消除為0。
＊IT0(TCON.0)：外部中斷0動作型態選擇，當IT1=1時，中斷型態為負緣觸發，當IT1=0 時，中斷型態則為低準位觸發。
(12) 串列埠控制(Serial Port Control)暫存器：

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

(表2.9 串列埠控制暫存器)
＊SM0(SCON.7)：串列傳輸模式選擇，共有4 種模式。
＊SM1(SCON.6)：串列傳輸模式選擇，共有4 種模式。
＊SM2(SCON.5)：在串列傳輸動作模式2 或模式3時，作多處處機控制功能用。
＊REN(SCON.4)：串列介面接收位元，當REN=1時表示接收致能。
＊TB8(SCON.3)：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。
＊RB8(SCON.2)：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。
＊TI(SCON.1)：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設為1，而在其他模式時，在送出停止位元時，TI也會被設為1；此位元必須由軟體來清除。
＊RI(SCON.0)：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會設為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為1。此位元必須由軟體來清除。

SM0	SM1	模式	動作
0	0	0	移位暫存器控制I/O,鮑率固定為(工作頻率/12)
0	1	1	8位元串列資料傳送,鮑率由計時器1來控制
1	0	2	9位元串列資料傳送,鮑率為(工作頻率/32)或(工作頻率/64)
1	1	3	9位元串列資料傳送,鮑率由計時器1來控制

(表2.10 串列傳輸模式選擇表)
(13) 串列資料緩衝(Serial Data Buffer，SBUF)暫存器：
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。

2.3 NE555晶片簡介

在數位電路的實習，使用時脈電路場合很多，時脈就是一個需要整形和具有寬度及電壓水準的連續波形。產生時脈的方式很多，大致可分為兩種，一為自行產生，如信號產生器一樣，另一種為外激方式產生，其脈波須由外加電壓信號控制而產生。此兩種電路均可由一般之電晶體、積體電路或基本邏輯閘來組成。

NE555 是屬於 555 系列的計時 IC 的其中的一種型號，555 系列 IC 的接腳功能及運用都是相容的，只是型號不同的因其價格不同其穩定度、省電、可產生的震盪頻率也不大相同；而 555 是一個用途很廣且相當普遍的計時 IC，只需少數的電阻和電容，便可產生數位電路所需的各種不同頻率之脈波訊號。

555 的特點有：

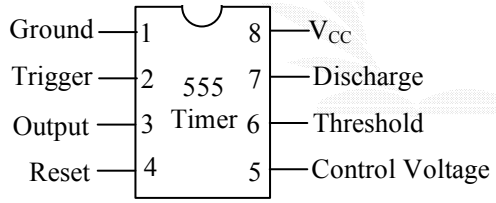
(1) 只需簡單的電阻器、電容器，即可完成特定的振盪延時作用。其延時範圍極廣，可由幾微秒至幾小時之久。

(2) 它的操作電源範圍極大，可與 TTL，CMOS 等邏輯閘配合，也就是它的輸出準位及輸入觸發準位，均能與這些邏輯系列的高、低態組合。

(3) 其輸出端的供給電流大，可直接推動多種自動控制的負載。

(4) 它的計時精確度高、溫度穩定度佳，且價格便宜。

555 晶片系列接腳圖：



(圖 2.6 晶片 555 接腳圖)

555 系列晶片有 8 根接腳，其接腳功能如下：

(1) Pin1 接地(Ground)：接電源負極。

(2) Pin2 觸發(Trigger)：當 Pin2 之電壓低於 1/3Vcc 時，會令 Output 輸出高電位及 Pin7 對地開路。

(3) Pin3 輸出(Output)：輸出腳，由 Pin2、4、6 控制其為高電位或低電位。

(4) Pin4 重置(Reset)：Pin4 之電壓小於 0.4V 時，Output 之輸出為低電位、Pin7 對地短路。所以不使用 Pin4 時，應接於 1V 以上之電壓。

(5) Pin5 控制電壓(Control Voltage)：Pin5 直接與比較器的參考電壓相通，充許由外界電路改變 Pin2、Pin6 之動作電壓。平時多接一個 0.01μF 以上之電容到 Ground，以避免雜訊干擾。

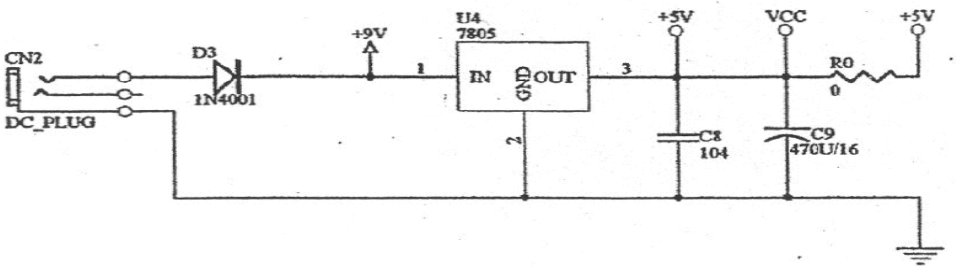
(6) Pin6 臨界(Threshold):Pin6 之電壓高於 2/3Vcc 時，會使 Output 低電位、Pin7 對地短路。

(7)　Pin7 放電(Discharge)：與 Output 同步動作。當 Output 輸出高電位時，Pin7 對地開路；Output 輸出低電位時，Pin7 對地短路。

(8)　Pin8(+Vcc)：電壓輸入，最大可至 15V。

第三章：電源模組、感應卡讀取模組、串列介面處理模組、資料顯示模組

3.1 電源模組



（圖 3.1 電源模組接線圖）

7085 穩壓 IC：市電經整流器轉換成 7.5V~300mA 經過此 5V 穩壓 IC 轉換成 5V。

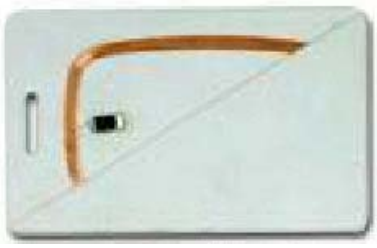
1N4001 整流二極體：其作用是防止電壓逆向。

104pF 傍路電容：對高頻傍路的濾波電容。

470uF/16V：作為穩壓濾波電容。

3.2 感應卡讀取模組

3.2.1 感應卡



（圖 3.2 厚卡封裝透視圖）

這次使用的感應卡是屬於感應卡的厚卡，他的內部構造為線圈跟一個特殊晶片組成，他可以記錄 64BIT 的資料，但是只能做讀取無法寫入，該卡片的相關資料如下表：

EM 感應卡規格明細	
通訊形式	Contact Less Passive
材　　質	PVC, PET, PCT

IC Chip	EM or Temic….Chip Inside
卡片壽命	正常情況下 2 至 5 年『視使用者維護程度』
頻率	125 KHZ
讀取範圍	約 10~15cm(一般情況)最大至 90cm(採用長距離讀卡機)
電源	Contact Less Power Supply.
Momory	64bit(僅可讀取)
表面粗細	Gloss / Mat
表面印刷	(薄卡)可視客戶需求全彩印刷
工作溫度	-20℃ to +50℃
工作溫度	-20℃ to +50℃
儲存溫度	-20℃ to +50℃
濕度	0- 80%.
尺寸	85.6 mm +-0.12*53.98 mm +-0.05*0.76 mm +-0.08
重量	厚卡 8.4g 薄卡 5.6 g
封裝方式	1.冷貼法 2.熱壓法

感應卡特色

1. 符合 ISO10536 規範
2. 非接觸式感應卡片，卡片壽命大幅提昇
3. 防潮、防塵、防污，並能適應惡劣使用環境
4. 不易仿造複製，安全性極佳
5. 可選用可讀寫感應卡，無需電池
6. 與信用卡尺寸相同
7. 存取距離約 10cm 至 90 公分(視讀卡機支援程度)
8. 非磁性條碼，不會消磁。不怕刮傷、污垢穩定性佳
9. 使用場所無環境限制。
10. 薄卡可支援多卡合一

(表 3.1 感應厚卡規格明細表)

3.2.2 讀取裝置

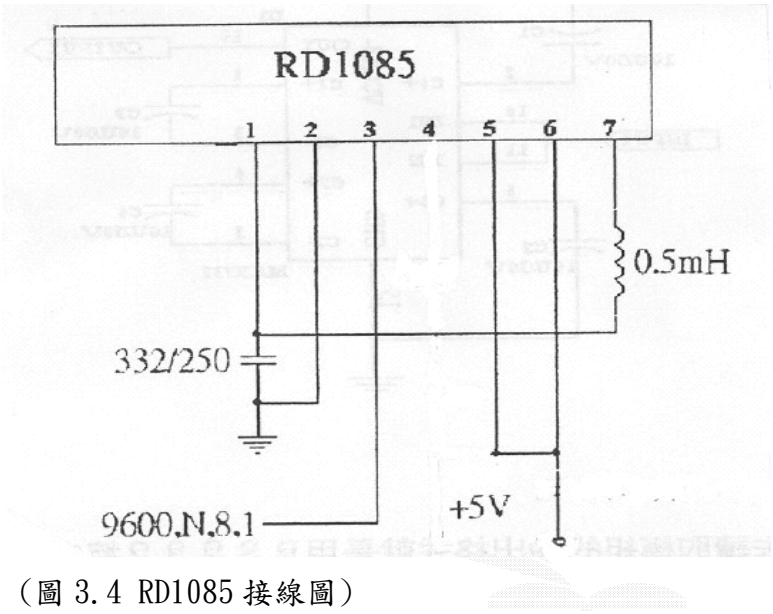
感應卡的讀取部分就是我們是利用市面上販售的 RD1085 非接觸式感應卡讀取模組，該模組在讀取卡片資料後，是用 RS232 之 9600.N.8.1 的協定將資料以 ASCII 格式送出，其整個資料串中共有 10 個 Bytes 的卡片資料及一個 byte

的帶頭碼 02 和一個 byte 的結尾碼 03 所以總長度是 12bytes，其資料如下圖：

0 2	1 2 3 4 5 6 7 8 9 A	0 3
帶頭碼	10 碼卡片資料(ASCII)	結尾碼

(圖 3.3 9600.N.8.1 資料格式圖)

RD1085 的標準應用電路除了所需的電源外，還需要外加 2 個元件，一個是 332PF 的電容，一個是 0.5Mh 左右的空心天線線圈；卡片資料的讀取距離最大約 20cm 左右，但這必須要看天線線圈截面積的大小以及是否有金屬物體干擾而定；這次專題所使用的線圈大約可以感應 5 至 10cm 左右，是擺放位置而定，下圖為 rd1085 的標準接線圖：



(圖 3.4 RD1085 接線圖)

3.3 串列介面處理模組

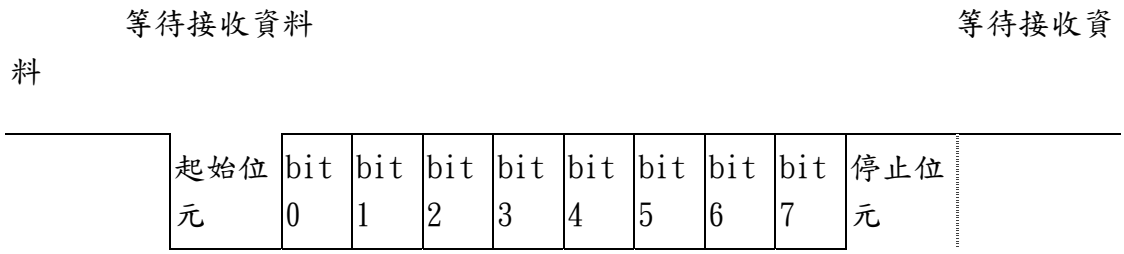
3.3.1 MCS51 介面簡介

＊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。

串列傳輸的結構雖然簡單，但也由於太簡略所以產生許多問題，必須藉由傳輸協定來解決。然而，一個完整的傳輸協定包括從硬體到軟體，相當複雜。其中最基本的一種非同步式串列介面（Universal Asynchronous Receiver Transmitter，簡稱 UART）常被用於一般的串列傳輸應用中。

串列傳輸在傳送一個位元組時，必須要傳送 8 次，而 UART 的串列傳輸方式是在傳送 8 個位元資料之前加上一個起始位元，並在傳送 8 個位元資料之後加上一個停止位元，於是原先傳送一個位元組要傳送 8 次就增為 10 次。以下是 UART 串列傳輸的示意圖，傳輸時間順序由左至右：



(圖 3.5 UART 串列傳輸的示意圖)

在 UART 的傳輸結構中，起始位元固定為 0，停止位元固定為 1，所以接收端的動作是一直不斷的檢查傳輸線的狀態。當傳輸線上的信號一直為 1 就表示沒有資料傳送；當傳輸線上的信號由 1 變為 0，即表示有資料將傳送，接收端就會開始準備接收 8 個位元資料，直到傳送完 8 個位元資料，傳送端最後會送出停止位元，並使傳輸線的信號保持為 1，以等待下一次的資料傳輸。經由增加起始位元與停止位元方式，雖然會使串列傳輸效率更降低，但可解決位元資料傳輸的起始與停止之問題。另一串列傳輸協定為傳輸速度，通常以鮑率(Buat Rate)，即每秒傳輸的位元數來衡量，一般 UART 常使用的鮑率有 1200、2400、4800、9600 及 19200 等。兩種裝置在進行串列傳輸時，必須決定以何種鮑率來進行資料傳輸，當兩種裝置使用同一鮑率才能確保資料傳輸正確無誤。

＊ UART 的結構

8051 單晶片的串列埠是一組全雙工的 UART，即 8051 的 UART 可以在同一時間進行串列資料的傳送與接收。8051 單晶片使用 P3.0 接腳做為串列傳輸的接收端(RXD)，P3.1 接腳做為串列傳輸的輸出端(TXD)，並利用特殊功能暫存器(Special Function Register，簡稱 SFR)中的串列埠緩衝器(Serial Port Buffer，簡稱 SBUF)執行串列傳輸的工作。當串列傳輸工作設定完成之後，傳送端會存入一筆資料到 SBUF 中，並藉以引發資料傳送的動作；當串列傳輸工作設定完成之後，接收端會將接收資料放入 SBUF 中。但在 8051 單晶片的 UART 結構中，接收資料端與傳送資料端實際使用的暫存器並不是同一個，只不過它們均對應到相同的定址位址，因此在傳送或接收資料時，8051 單晶片會自動選擇使用不同的暫存器，所以 8051 的串列埠可以同時進行資料的傳送與接收。

8051 單晶片進行串列資料傳輸時，串列埠具有輸入緩衝的功能，即當串列埠接收到一筆資料後，會把資料存放至 SBUF 中，然後繼續接收資料，並在接收或等待接收下一筆資料的過程中處理 SBUF 中的資料。因此，串列埠可以持續不

斷的接收資料,而不必在接收一筆資料後等待該資料完全處理完畢才進行下一筆資料的接收。但在第二筆資料被 UART 接收完畢前，第一筆資料須被處理完畢由程式讀入，否則會產生資料流失的問題。

＊UART 相關暫存器

在 SFR 記憶體中與 UART 相關的暫存器有兩個,分別為串列埠控制暫存器(Serial Port Control register，簡稱為 SCON)及電源控制暫存器(Power Control register，簡稱為 PCON)。以下為此二暫存器的結構圖：

SCON：串列埠控制暫存器 (SERIAL PORT CONTROL REGISTER)

位址：98H

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0 SCON.7 串列埠模式選擇。

SM1 SCON.6 串列埠模式選擇。

SM2 SCON.5 在串列埠為模式 2 和 3 時，致能多處理器通信的功能。在模式 2 或 3，如果 SM2=1，則當接收到的第 9 資料位元為 0 時，RI 不動作。在模式 1 時，若 SM2=1」，當接收到的停止位元不正確時，RI 也不動作，在模式 0 時，SM2 必須為 0。

REN SCON.4 由軟體去設定或清除，以決定是否接收串列輸入資料，(REN=1 接收)。

TB8 SCON.3 在模式 2 或 3 時，為傳送資料時的可規劃資料位元，由軟體控制。

RB8 SCON.2 在模式 2 或 3 時，接收的可規劃資料位元放在這個位元裡。在模式 1 時，如果 SM2=0，RB8 為接收到的停止位元；在模式 0 時，RB8 沒有用。

TI SCON.1 傳送中斷旗號，在模式 0 時，在第 8 位元結束時，硬體會將它設為 1，其他模式時，是在停止位元的開始時設定為 1。此位元必須由軟體清除。

RI SCON.0 接收中斷旗號，在模式 0 時，在第 8 位元結束時，硬體會將它設為 1，其他模式時,在停止位元的一半的時候由硬體設定(參考 SM2)此位元必須由軟體清除。

(表 3.2 SCON：串列埠控制暫存器 (SERIAL PORT CONTROL REGISTER))

PCON：電源控制暫存器　　(POWER CONTROLREGISTER)

位址：(87H)

SMOD	－	－	－	GF1	GF0	PD	IDL

SMOD　雙倍鮑率位元，當串列埠工作於模式 1、2 或 3 時，如使用 Timer1 作鮑率產生器，且 SMOD=1，則鮑率為雙倍。

－　保留將來使用

－　保留將來使用

－　保留將來使用

CF1　一般用途

CF0　一般用途

PD　電源下降位元，80C51BH 時，設定此位元為”1”就進入電源下降模式(僅 CHMOS 可以)。

IDL　IDLE 模式位元，80C51BH 時，設定此位元為”1”就進入 IDLE 模式(僅 CHMOS 可以)。

若同時寫 1 至 PD 和 IDL 時，PD 優先

(表 3.3 PCON：電源控制暫存器 (POWER CONTROLREGISTER))

＊UART 串列埠的四種工作模式

在 SCON 結構圖中可知 SCON 位元是由模式選擇位元,可規劃資料位元及旗標位元所組成。而 PCON 結構圖中可知只有 SMOD 位元與串列埠傳輸速度有關，其他位元則是用於省電模式的設定。

利用 SCON 的 SM0 及 SM1 可以來選擇四種工作模式：

1. 模式 0：SM1＝SM0＝1

串列埠設定為模式 0 時，串列資料的傳送與接收都是利用 RXD 接腳進行，而 TXD 接腳則做為輸出移位脈波，此脈波的鮑率固定為 8051 單晶片的振盪頻率之 1/12。當要從串列埠傳送資料時，只要執行一個資料寫入 SBUF 指令，則會引發資料傳送的動作；資料傳送完畢後，8051CPU 會將 SCON 中的 TI 位元設定為 1，通知串列中斷產生。當要從串列埠接收資料時，須先以軟體設定 SCON 中的 REN 位元，然後執行清除 RI 位元，串列埠就會依時序進行接收的工作，資料接收完畢後，8051CPU 會將 SCON 中的位元設定為 1，通知串列中斷產生。

模式 0 通常是用於 I/O 的擴充，而非用於串列通訊。只要將 RXD 及 TXD 接腳連接到一個並入串出(PISO)的 IC，就可以擴充一個 8 位元的輸入埠；將 RXD 及 TXD 接腳連接到一個串入並出(SIPO)的 IC，就可以擴充一個 8 位元的輸出埠。

2. 模式 1：SM1＝1、SM0＝0

串列埠設定為模式 1 時，8051CPU 每次傳送與接收的資料為 10 位元，這 10 位元分成下列 3 部分，分別為：

(1)起始位元：固定為 0，佔用一個位元。

(2)資料位元：佔 8 個位元，依低位元至高位元傳輸順序。

(3)停止位元：固定為 1，佔用一個位元。

模式 1 資料傳輸的鮑率是由 Timer 1 設定，其設定如下表：

常用的鮑率值

			Timer 1		
鮑率	振盪器頻率	SMOD	C/T	模式	載入值
模式 0（最大 1M）	12M Hz	×	×	×	×
模式 2（最大 375K）	12M Hz	1	×	×	×
模式 1、3（最大 62.5K）	12M Hz	1	0	2	FFH
19200	11.0592M Hz	1	0	2	FDH
9600	11.0592M Hz	0	0	2	FDH
4800	11.0592M Hz	0	0	2	FAH
2400	11.0592M Hz	0	0	2	F4H
1200	11.0592M Hz	0	0	2	E8H
137.5	11.0592M Hz	0	0	2	1DH
110	6M Hz	0	0	2	72H
110	12M Hz	0	0	1	FEEBH

(表 3.4 常用鮑率值表)

串列埠設定完畢後，8051CPU 執行寫入資料到 SBUF 指令時，就會進行資料傳送的動作。當資料傳送完畢後，CPU 會將 SCON 中的 TI 位元設定，通知串列中斷產生。而在資料接收時，當 RXD 接腳由 1 變為 0 時開始接收資料，CPU 依序接收 10bit 資料；接收資料完畢後，CPU 會測試 RI、SM2 及停止位元是否符合下列條件：

(1)RI 位元清除為 0

(2)SM2 位元清除為 0 或所接收到的停止位元設定為 1

當上列條件都符合時，8051CPU 則將所接收到的 8 位元資料存入 SBUF 中，並將所接收到的停止位元存入 SCON 的 RB8 位元中，再將 RI 位元設定為 1，通知串列中斷產生。若上列條件不符合時，則該次所接收的資料將會流失。

3. 模式 2：SM1＝0、SM0＝1

串列模式設定為 2 時，8051CPU 每次傳送與接收的資料為 11 位元，這 11 位元是由下列 4 部分所組成，分別為：

(1)起始位元：固定為 0，佔用一個位元

(2)資料位元：佔 8 個位元，依低位元至高位元傳輸順序

(3)可規劃資料位元：佔用一個位元(TB8 或 RB8)

(4)停止位元：固定為 1，佔用一個位元

模式 2 資料傳輸的鮑率是由 SMOD 決定，當 SMOD＝0 時，鮑率為 375K Hz；當 SMOD＝1 時，鮑率為 187.5K Hz。當傳送資料時，必須先由軟體設定 SCON 中 TB8 的位元值，然後再執行資料寫入 SBUF 指令，以驅動資料開始傳送的動作，然後串列埠會依序傳送起始位元、資料位元、可規劃資料位元 TB8 及停止位元；傳送完畢後，8051CPU 會設定 SCON 中 TI 位元值，以通知串列中斷產生。

當接收資料時，若 RXD 接腳信號由 1 變為 0 時開始接收資料，8051CPU 會依序接收 11 位元資料；接收資料完畢後，CPU 會測試 RI、SM2 及停止位元是否符合下列條件：

(1)RI 位元清除為 0

(2)SM2 位元清除為 0 或所接收之可規劃資料位元為 1

當上列條件都符合時，8051CPU 則將所接收到的 8 位元資料存入 SBUF 中，且將所接收到的可規劃資料位元存入 SCON 的 RB8 位元，再將 RI 位元設定為 1，以通知串列中斷發生。若上列條件不能同時符合時，則該次所接收的資料將會流失。

4. 模式 3：SM1＝1、SM0＝1

串列埠設定為模式 3 時，其動作與模式 2 相似，其唯一的差別在於模式 3 的傳輸速度之鮑率值設定與模式 1 相同，是由 Timer1 設定。

以上四種串列埠模式，在傳輸資料時，鮑率的準確與否對資料之接收非常的重要，因此因此在使用模式 1 與模式 3 時，要先啟動 Timer 工作。以下列串列埠使用的步驟以檢查串列埠設定是否正確：

(1) 設定 Timer1 工作模式並根據傳輸鮑率設定 TH1 及 TL1(UART 模式 0 與模式 2 不用此項)

(2)決定 SMOD 位元值為 0 或 1

(3)設定串列埠工作模式，並清除 RI、TI 位元為 0，及設定 REN 位元為 1

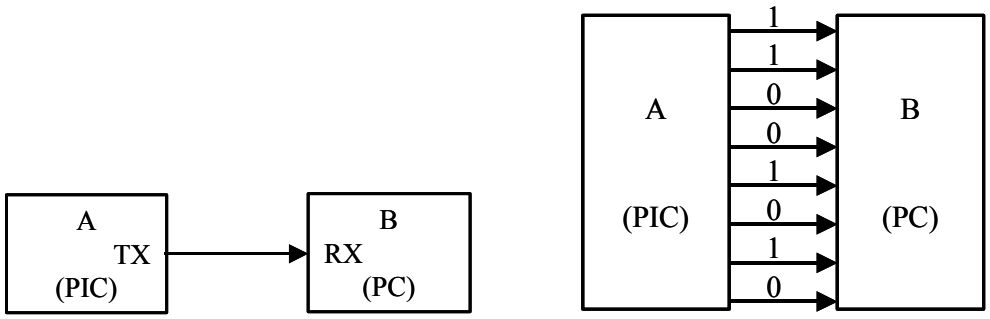
(4)致能串列埠中斷

(5)啟動 Timer1 開始計時(UART 模式 0 與模式 2 不用)

(6)執行〝MOV SBUF，XX〞指令，來啟動 UART 傳送資料

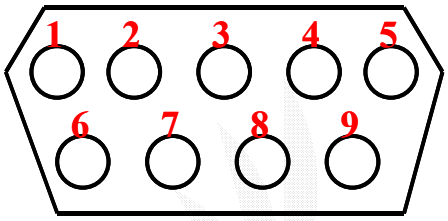
3.3.2 RS232 介面簡介

在數位傳輸中，有串列(serial)傳輸與並列(parallel)傳輸兩類。如圖 1 所示，以由 A 端傳送一個 Byte 資料到 B 端為例，串列傳輸僅有一條傳送資料的資料線，如此傳完 8 個 bits，花費了 8 個 clock 的時間。而並列傳輸的做法簡單來說，則是利用 8 條線，每一條線負責傳送 1 個 bit，如此 1 個 clock 時間即可傳完 8 個 bits 了。串列傳輸的優點在於傳輸線少，配線簡單，傳送距離可以較遠，而並列傳輸的優點則為傳輸速度較快。



（圖 3.6 串列傳輸（左）與並列傳輸（右））

電腦上 RS232 為一串列傳輸介面，如圖 3.6 及表 3.5 所示，RS-232 的 9 支腳位各有其功用和訊號流動方向，RS-232 設計之初是用來連接數據機做傳輸之用，也因此它的腳位意義通常也和數據機傳輸有關。



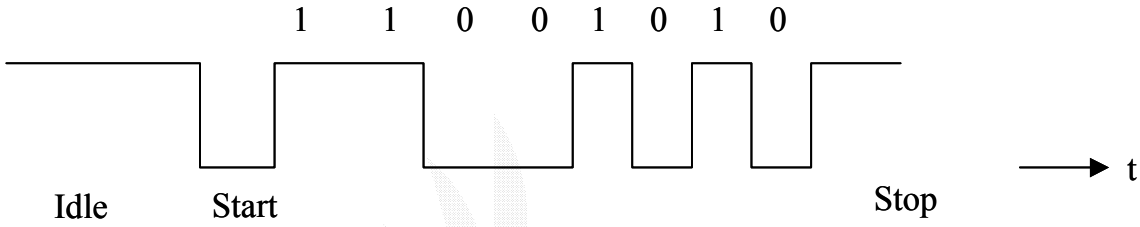
（圖 3.7 RS-232 腳位方向）

（表 3.5 腳位編號及其意義）

腳位	簡寫	意義
Pin1	CD	載波偵測(Carrier Detect)
Pin2	RXD	接收字元(Receive)
Pin3	TXD	傳送字元(Transmit)
Pin4	DTR	資料端備妥(Data Terminal Ready)
Pin5	GND	地線(Ground)
Pin6	DSR	資料備妥(Data Set Ready)
Pin7	RTS	要求傳送(Request To Send)
Pin8	CTS	清除以傳送(Clear To Send)
Pin9	RI	響鈴偵測(Ring Indicator)

由於 RS-232 在傳送資料時，並不需要另外使用一條傳輸線來傳送同步訊號，就能正確的將資料順利傳送到對方，因此叫做「非同步傳輸」，簡稱

UART(Universal Asynchronous Receiver Transmitter)，比起「同步傳輸」的方式，如 SPI（串列週邊介面）或 I^2C（積體電路間通訊）等，可以節省一條傳輸線。不過節省一條傳輸線的代價，是必須在每一筆資料的前後都加上同步訊號，把同步訊號與資料混和之後，使用同一條傳輸線來傳輸。例如圖 3 中的資料 11001010 被傳輸時，資料的前後就加入 Start(Low)以及 Stop(High)等兩個位元，因此，傳送 11001010 這樣一個 Byte 資料，實際上是需要 10 個 clock 的時間才能完成。值得注意的是，Start 訊號固定為一個位元，但 Stop 位元則可以是 1、1.5 或者是 2 位元，由使用 RS-232 的傳送與接收兩方面自行選擇，但需注意傳送與接受兩者的選擇必須一致。此外由於 RS-232 在每一筆資料的前後都加了同步訊號，因此每筆資料的真正長度就不一定都是非 8 個位元不可，也因此 RS-232 的資料長度是可以由使用者在 5~8 之間自行選擇。



（圖 3.8 RS-232 之非同步傳輸）

在非同步、串列傳輸中，傳送與接收兩端除事先設定好如圖 3.7 相同的傳輸格式之外，還必須事先雙方設定好位元傳送的速度，如每秒鐘傳送幾個 bits，這樣接收端才知道每隔多久時間就該去取下一個 bit 的邏輯位準。而這個“每秒鐘傳送幾個 bits”的規格，就是所謂的“Baud Rate（鮑率）”，例如 Baud Rate=9600 的意思就是每秒傳送 9600 個位元。

一般序列通訊阜所傳送的資料是字元型態，若用來傳輸檔案，則會使用二進位制的資料型態。目前常用的資料長度，除了上述的 8 個位元之外，由於國際通用的標準字元碼(ASCII)只使用了 7 個位元，因此也有許多人選用 7 位元的資料長度來做 RS-232 傳輸，而將不用的第 8 個位元拿來作為除錯之用，稱為「同位元(Parity)」。除錯的方式是計算整筆資料中，1 的個數為奇數或偶數。如果為奇數，則稱為「奇同位(Odd Parity)」；反之，如果是偶數，就叫做「偶同位(Even Parity)」。如果使用者選擇資料長度為 8 位元，則因為沒有多餘的位元可被用來作為同位元，因此就叫做「無位元(Non Parity)」。

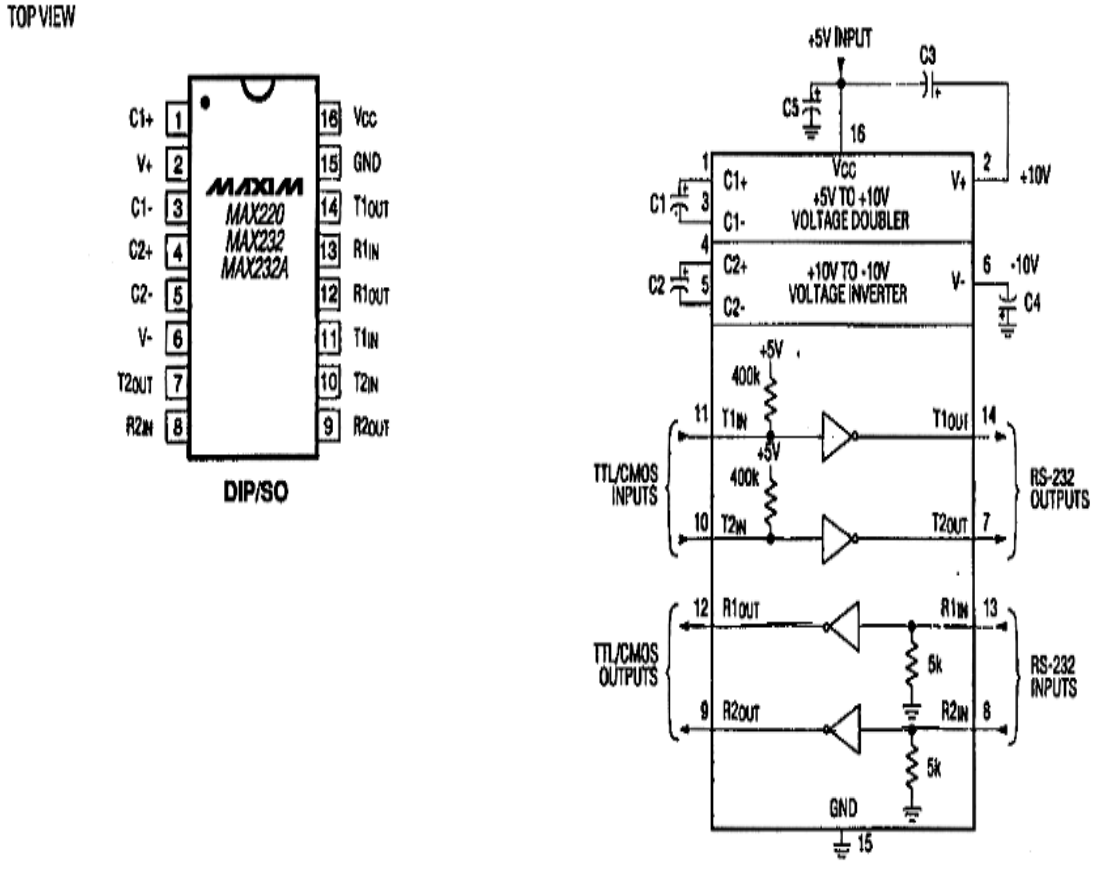
綜合上述說明，RS-232 的通訊格式可表示如“9600, N, 8, 1”，表示該系統的 Baud Rate 是 9600，而且該系統沒有使用同位元，資料長度為 8 個位元，最後的“1”代表使用者把停止位元長度定為 1 個位元。

3.3.3 傳輸介面轉換

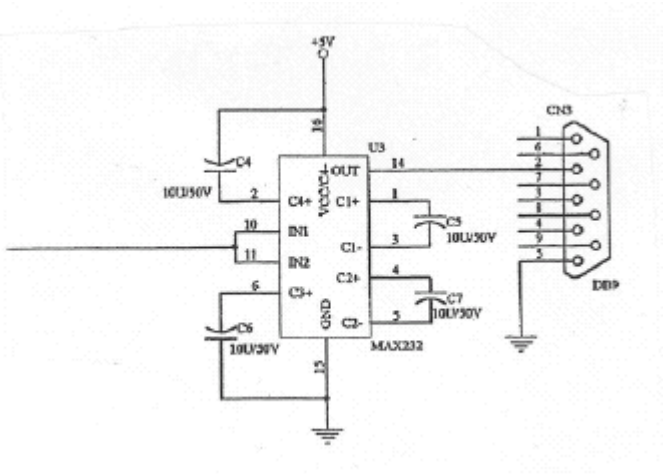
89C51 單晶片內含的傳輸串列介面是微 TTL 電位，故在讀取 RD1085 的資料時並沒有問題，但若要與外部的 RS232 介面通訊時就必須專換成標準的 RS232 電位，在此我們利用市面上販售的 RS232 介面專用 IC 如：MAX232 再加上幾科電容，就可以達成這個要求，如此便可跟一般 PC 連結，加上一個小程式便可看見成果。
＊MAX232

MAX232 這只 IC，這是專為 TTL 與 RS-232 之間的位準相互轉換用的，以 555 為中心的 DC-DC 轉換線路，只要外加幾只電容即可。

以這只 MAX232 IC 為中心的轉換器介面，線路非常簡捷，只要外加 +5V 電源、幾只電容，如此而已。



（圖 3.9 MAX232 接腳圖及內部電路圖）



（圖 3.10 MAX232 接線圖）

3.4 資料顯示模組(LCD 顯示)

資料顯示部分是採用目前市面上標準的 16*2 文字 LCD 顯示模組，其背面含有控制電路，專門用來完成 LCD 的控制動作，所以我們只送入適當的指合所想顯示的資料，LCD 便會將其字元顯示出來，所以在程式撰寫上就顯的比較簡單。

LCD 為Liquid Crystal Display 縮寫，也分為繪圖模式LCD 及文字模式 LCD，市面上很容易買到不同廠牌之顯示文字的LCD，但似乎LCD 的控制器皆為同樣一顆，編號為HD44780A。所以不論16 字*1列，20 字*2 列之文字模式:LCD，其控制方法皆同。HD44780A 之特性為：

＊顯示資料RAM（ Display Data RAM ）共80 個Bytes。

＊字元產生器ROM（ Character Generator ROM，簡稱CG ROM)有160 個5*7 點矩陣字型(pattern) 。

＊字元產生器RAM（ Character Generator RAM，簡稱CG RAM ）

＊可寫入8 個字型，使用者可自行設計8 個5*7 點矩陣字型。

＊多種控制指令如清除顯示器，游標歸位(Cursor Home) ，顯示器關閉/開啟，游標關閉/開啟、字元閃爍，游標移位、顯
示移位等等。

硬體說明

欄位	接腳符號	方向	名稱及功能
1	VSS		電源地（ Ground ）
2	VDD		電源正極：接+5 伏特
3	V0		亮度調整電壓輸入（ Contrast adjustment voltoge ）。通常輸入零伏特時字元最清晰，或以半可變電阻來調整。

4	RS	I	暫存器選擇(Register Select) 線：一般接CPU 之位址線A0 ，此腳輸入”0”，並做寫入動作，即可寫入指令暫存器(Instruction Register)；若輸入”0”，做讀取動作則可讀取忙碌旗標(Busy flag) 及位址計數器。此腳輸入”1”，為讀寫資料暫存器(Data Register) 。
5	R/W	I	讀寫線：”0”表示寫入LCD 控制器，”1”表示讀取LCD 控制器。
6	E	I	致能線(Enable) ：此腳為”1”時，讀寫才有效。注意，其下緣時，有效資料需在匯流排上。
7-14	DB0-DB7	I/O	資料匯流排：以8 位元資料讀寫方式則DB0~DB7 皆有效。若以4 位元做資料讀寫則僅DB4~DB7 有用到，DB0~DB3 不必連接。

(表 3.6 LCD pin 腳說明表)

(1) 暫存器(Registers) ，: LCD 共有兩個八位元暫存器，即指令暫存器(Instruction Register ， IR) 和資料暫存器(DataRegister ， DR) 功能為：

＊指令暫存器可存放指令碼(Instruction code) 或DD RAM位址或CG RAM 位址，此暫存器僅可寫入。

＊當寫資料到LCD 時，前一個位址指令已決定了是要寫入DDRAM 或CG RAM ，資料首先存放在資料暫存器，再自動地移入DD RAM或CG RAM

＊當讀取LCD 資料時DD RAM 或CG RAM 的資料會暫時存放在資料暫存器，再由CPU讀去。其實，在前一個位址值寫入指令暫存器時，資料已先從DD RAM 或CG RAM 移入資料暫存器，每讀取一次，下一個RAM 位址的資料自動地又移入資料暫存器

(2) 忙碌旗標(Busy Flag ， BF) ，當BF=”1”，表示LCD 正在執行內部運作，並且不接受新的指令。BF 可由RS=”0”時讀取位元7 得到旗標狀態；直到BF=”0”，才可輸入下一個指令。

(3) 位址計數器(Address Counter ， AC) ，位址計數器可產生DDRAM 及CG RAM 之位址。位址值及選擇哪一個RAM 可一次寫入指令暫存器決定之；讀寫RAM 資料後，位址計數器將自動遞增(Increment) 或遞減(Decrement) 。位址計數器亦可由RS=”0”時讀取低的7 個位元得之。

(4) 顯示資料RAM (Display Data RAM ， DD RAM) ，共80 bytes ，存放顯示資料的字元碼(character code) ，一個DD RAM 位址(address) 對應一個顯示位置(position) ，寫入不同的字元碼就顯示不同的字型。沒有對應到顯示

的RAM 可由使用者儲存資料或自行運用。RAM 位址與顯示位置對應關係有三種形式，而本專題所採用的形式是雙列顯示器(Dual-line display)，茲介紹如下：

1	2	3	4	5		39	40
00H	01H	02H	03H	04H	·····	26H	27H
40H	41H	42H	43H	44H	·····	66H	67H

顯示位置
第一列位址
第二列位址

01H	02H	03H	04H	05H	·····	27H	00H
41H	42H	43H	44H	45H	·····	67H	40H

設顯示左移
(Left shift)

27H	00H	01H	02H	03H	·····	25H	26H
67H	40H	41H	42H	43H	·····	65H	66H

設顯示右移
(Right shift)

(圖3.11 LCD雙列顯示位址圖)

雙列顯示器位置與位址對應須注意，第二列之位址與第一列位址不連續，但總數仍為80 個位址，第一列由00H~27H，第二列由40H~67H，下位址指令時要注意。若顯示器一列僅有16 或20 字，則由第1 位置到地16 或20 個位置對應之。

(5) 字元產生器ROM (Character Generator ROM，CG ROM)，CGROM 中存放著160 個5*7 點矩陣字型(dot-martrix characterpattern)及32 個5*10 點矩陣字型。每個字型對應著不同的8 位元字元碼(character code) ，部份與ASCII 相同，其餘為日文字型。字元碼由00~07H 可由使用者自行設計5*7 字型共8 個字型，20H~7FH 與ASCII 相同，A0H~DFH 為日文字型，E0H~FFH 為32 個5*10 字型。所以，可看出字元碼並不連續，有些碼沒用到。

(6) 字元產生器RAM (Character Generator RAM ， CG RAM) ，使用者可自行設計任意的5*7 點矩陣字型，字型輸入資料先寫入CG RAM ，再將對應的字元碼寫入DD RAM ，就可以顯示字型了CGRAM 共有64 個bytes ，每個字型佔用8 個bytes ，對應00H~07H八個字元碼，即CG RAM 的0~7 bytes 對應字元碼00H，8~15 bytes對應字元碼01H ，依此類推。若未設計字型，則CG RAM 可由使用者自由運用。

(7) 時序產生器(Timing Generator) ，供應DD RAM ， CG RAM ，
及CG ROM 內部運作之時序信號。

(8) 游標/閃爍控制器(Cursor / Blink Controller) ，用來產生一個游標和一個閃爍的字元，顯示在DD RAM 目前位址所指的顯示位置，游標為字元下的一條橫線。

(9) 並列對串列轉換器(Parallel-to-Serial Converter) ，將CGROM 或CG RAM 中讀出的並列資料轉成串列資料送到顯示驅動器(display driver) 。

(10) 偏壓產生器(Blias Voltage Generator) ，用來產生液晶顯示器(LCD) 顯示時所需的偏壓準位。

(11) LCD 驅動器(LCD Driver) ，此電路接收顯示資料，時序信號和偏壓來產生共用背景顯示及各段顯示信號。
(12) LCD 面板(LCD Panel) ，點矩陣液晶顯示面板，有一列16字，兩列16 字，兩列20 字及兩列40 字等種類。字元與字元間有一個點距離的間隙。

(表 3.7 LCD 指令控制碼一纜表)

暫存器及指令碼說明

下表為 LCD 指令控制碼一覽表

項目	R/W	RS	操作名稱	位元值								功能說明	執行時間
				D7	D6	D5	D4	D3	D2	D1	D0		
1	寫入	0	清除顯示器	0	0	0	0	0	0	0	1	清除顯示器，並將游標移回左上角位置	1.64ms
2	寫入	0	游標歸位	0	0	0	0	0	0	1	X	游標移回左上角位置，但顯示 RAM 中資料未變	1.64ms
3	寫入	0	設定進入模式	0	0	0	0	0	1	I/D	S	I/D=1：位置遞增，I/D=0：位置遞減；S=1：移位功能致能	40us
4	寫入	0	顯示器開/閉	0	0	0	0	1	D	C	B	1 表 ON，0 表 OFF，D 為顯示器，C 為游標，B 為游標所在位置的	40us

												字元閃爍	
5	寫入	0	顯示器/游標移位	0	0	0	1	S/C	R/L	X	X	S/C=1：顯示移位， S/C=0：移動游標； R/L=1：向右移，R/L=0：向左移	40us
6	寫入	0	功能設定	0	0	1	DL	N	F	X	X	DL=1：資料長8位元， DL=0：資料長度4位元； N=1：雙列字，N=0：單列字；F=0：5*7點	40us
7	寫入	0	設CG RAM位址	0	1	CG RAM 位址						將CG RAM位址寫入位址計數器，接著讀寫CG RAM資料	40us
8	寫入	0	設DD RAM位址	1	DD RAM 位址							將DD RAM位址寫入位址計數器，接著讀寫DD RAM資料	40us
9	讀出	0	讀忙碌/位址	BF	位址計數器							讀BF及位址計數器內容	40us
10	寫入	1	寫入資料暫存器	寫入之資料								將資料寫入CG RAM或DD RAM	40us
11	讀出	1	讀取資料暫存器	讀取之資料								從CG RAM或DD RAM讀出資料	40us

其中1~8 項為RS=0，寫入指令暫存器，寫入不同的控制碼即產生不同的控制功能。第9 項為令RS=0 時，讀取LCD ，此時讀入資料的位元7 為忙碌旗標，其他位元為位址計數器內容。第10、11 項為令RS=1，即可讀寫CG RAM 或DD RAM 中的資料，至於是前者或後者，要由使用者所設定位址是CG RAM 位址或DD RAM 位址決定(第7 或第8 項) 。

右邊一行為每個指令的執行時間(Execution time) ，此為使用者每一次讀寫LCD 暫存器後要等待的時間，因為LCD 控制器本身接收一個指令後，在內部做處理及運算，需要花費時間，此時間過後CPU 才可再下另一LCD 控制指令，否則會被忽略。各指令說明如下：

(1)清除顯示器(Display clear) ，將DD RAM 內資料皆填入空白碼(space code) 20H 。位址計數器清為零。若顯示移位過，也會恢復原始位置。執行此指令，使所有顯示消失，游標及字元閃爍位置移到左上角。指令碼為：

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	0	1

(2)顯示/游標歸位(Display/Cursor Home) ，位址計數器清為零，顯示恢復原始位置，游標移到左上角。DD RAM 中資料無影響。指令碼為：

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	1	X

(3)進入模式設定(Entry Mode Set) ，指令碼為：

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	1	I/D	S

I/D：位址計數器遞增(I/D=1) 或遞減(I/D=0) ，每讀寫DDRAM 中字元碼一次則位址計數器加一或減一。游標顯示之位置亦同時向右(I/D=1) 或向左(I/D=0) 移一個位置。讀寫CG RAM 時亦相同效果。

S：當S=1 ，寫入一個字元碼到DD RAM 時，整個顯示幕向左(I/D=1) 或向右(I/D=0) 移一格位置，而游標仍停留在相對的顯示位置。當S=0 ，顯示幕不移動。寫入CG RAM 時，顯示幕不移動。

(4)顯示啟/閉(Display ON/OFF) ，指令碼為：

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	1	D	C	B

D（ Display ）：當D=1，顯示幕開啟(turn on ）。當D=0，顯示幕關閉(turn off ），顯示資料仍保存在DD RAM 中。C（ Cursor ）：當C=1，游標被顯示在位址計數器所指的位置上。當C=0，游標不會出現。游標是在5*8 矩陣的第8 列五個點構成，餘5*7 點為字型。對5*10 之字型則在第11 列的五個點構成。B（ Blink ）：當B=1，在游標位置的字元會閃爍，閃爍方式為所有5*8 的點(含游標）變黑409.6ms ，然後字元出現409.6ms ，如此交替顯示所有點及字元而成閃爍現象。

(5)顯示/游標歸位(Display/Cursor Shift ），顯示幕和/或游標被向右或向左移動。對於雙列顯示器游標會從第一列的第40 個位置移到第二列的第一個位置，但移到第二列第40 個位置後不會歸回原點，而是移到第二列第一個位置。指令碼為：

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	1	S/C	R/L	X	X

S/C	R/L	
0	0	游標向左移（ AC ← AC-1 ）
0	1	游標向右移（ AC ← AC+1 ）
1	0	整個顯示幕和游標向左移
1	1	整個顯示幕和游標向右移

(6)功能設定(Function Set ），此指令一定要在所有其他指令碼之前。指令碼為：

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	DL	N	F	X	X

DL（ Data Length ）：選擇介面資料長度， DL=1 為2-bit 資料轉移（ Transfer ）， DL=0 為4-bit 資料轉移。使用4-bit 資料轉移時，一個完整字元資料要讀或寫兩次。

F（ Font ）： F=1，5*10 點矩陣。F=0，5*7 點矩陣。N（ Number of display line ）：選擇顯示之列數為雙列或單列，N=0 表單列(Single line ）。N=1 表示雙列(dual line ），若單列顯示器以雙列定址方式仍要選N=1。

(7)字元產生器RAM 位址設定(CG RAM Address Set) ，可將CG RAM 位址載入位址計數器中，位址由6 個位元所組成。指令碼為：

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	1	A	A	A	A	A	A

A：Address bit

(8)顯示資料RAM 位址設定(DD RAM Address Set) ，可將DD RAM位址載入位址計數器中，位址由7 個位元所組成。指令碼為：

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	1	A	A	A	A	A	A	A

A：Address bit

(9) 忙碌旗標/位址計數器讀取(Busy Flag/Address counter read) ，指令碼為：

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	1	BF	A	A	A	A	A	A	A

A：Address bit

BF=1，表示LCD 正在執行內部運作，並且不接受新的指令。BF 可由RS=0 時讀取位元7 得到旗標狀態，BF=0 才可輸入下一指令。
AC (Address Counter) 可產生DD RAM 及CG RAM 之位址。位址值及選擇哪一個RAM 可一次寫入指令暫存器決定之。
(10)字元產生器RAM /顯示資料RAM 的資料寫入(CG RAM / DD RAM Data Write) ，資料是寫入CG RAM 還是DD RAM 中的哪一個位址，皆由先前所下的位址設定指令決定。指令碼為：

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	0	D	D	D	D	D	D	D	D

D：Data bit

(11)字元產生器RAM /顯示資料RAM 的資料讀取(CG RAM / DD RAM Data Read) ，資料是由CG RAM 還是DD RAM 中的哪一個位址讀出，皆由先前所下的位址設定指令決定，與寫入類似。讀或寫資料都會使位址自動加一或減一。指令碼為：

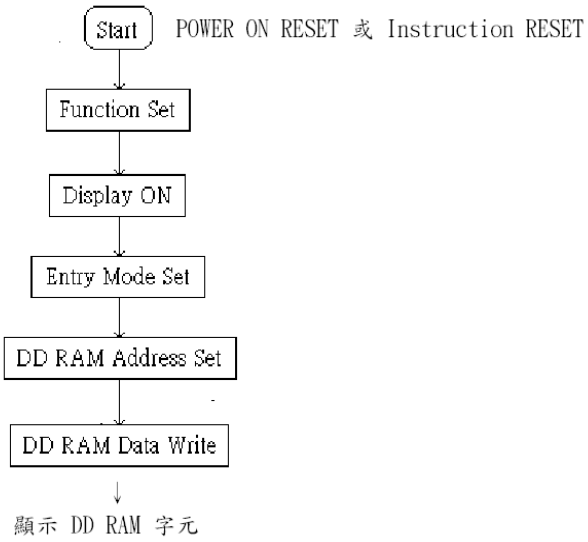
RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	1	D	D	D	D	D	D	D	D

D：Data bit

軟體規劃

要令LCD 顯示文字，首先要寫入指令暫存器(IR) 規劃LCD各項功能、模式，接著將字元碼(ASCII)寫入資料暫存器(DR) 移入DD RAM 中，即可顯示字元。適當地在每個指令後加些延遲，大於LCD 指令控制表內所列之執行時間，這樣可省卻測試忙碌旗標(BF)之動作。下圖為軟體流程：

↓



(圖 3.12 LCD 顯示文字軟體流程圖)

LCD 重置及初始化

LCD 有一個內部重置電路(Internal Reset Circuit) ，只要電源電壓(VDD 腳) 在10ms 內上升到4.5V 以上，即可產生PowerON Reset ，並做LCD 初始化工作如下：(初始化時BF=1， VDD 達4.5V 後約延續10msec ， BF 才降為0) 。

＊清除顯示(Clear Display)

＊功能設定， DL=1， N=0， F=0，8 位元介面，單列顯示，5*7 點矩陣字形。

＊顯示關閉， D=0， C=0， B=0。顯示幕、游標，閃爍功能關閉。

＊進入模式， I/D=1， S=0。遞增模式，顯示幕不移動。

第四章：系統架構及流程

4.1 系統主要架構

整個專題的架構可分為五個部分

＊中央控制部分

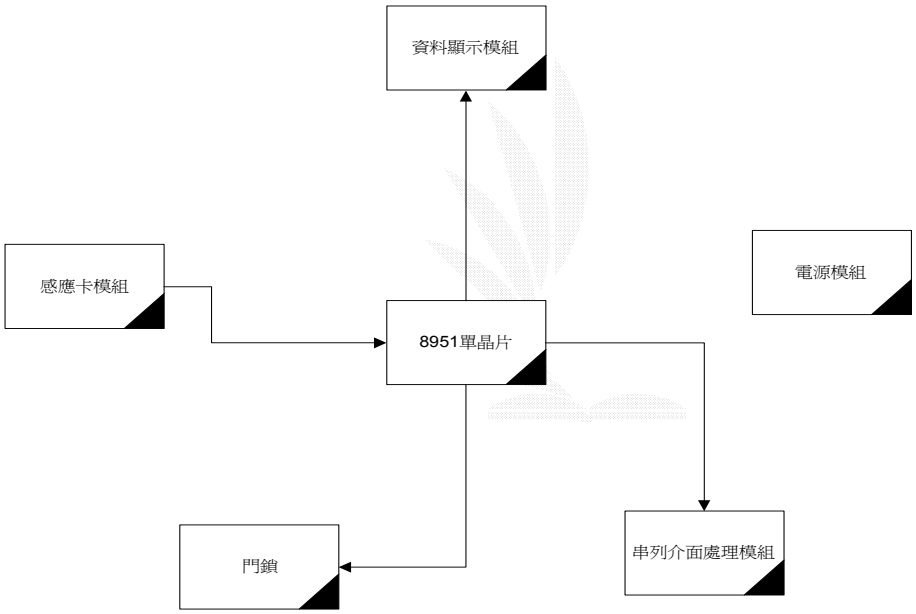
＊感應卡資料讀取部分

＊資料顯示部分

＊串列電位轉換部分

＊電源供應部分

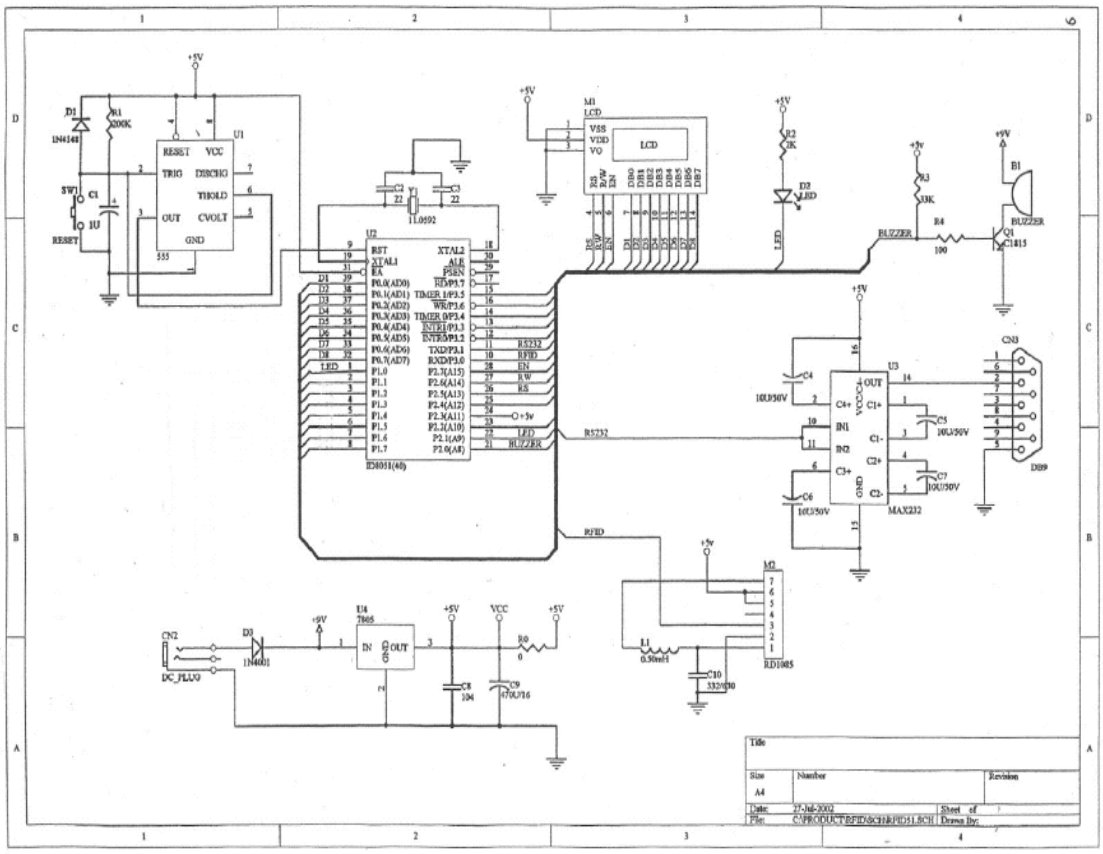
＊開關門鎖部分(用 LED 取代門鎖)



(圖 4.1 系統主要架構圖)

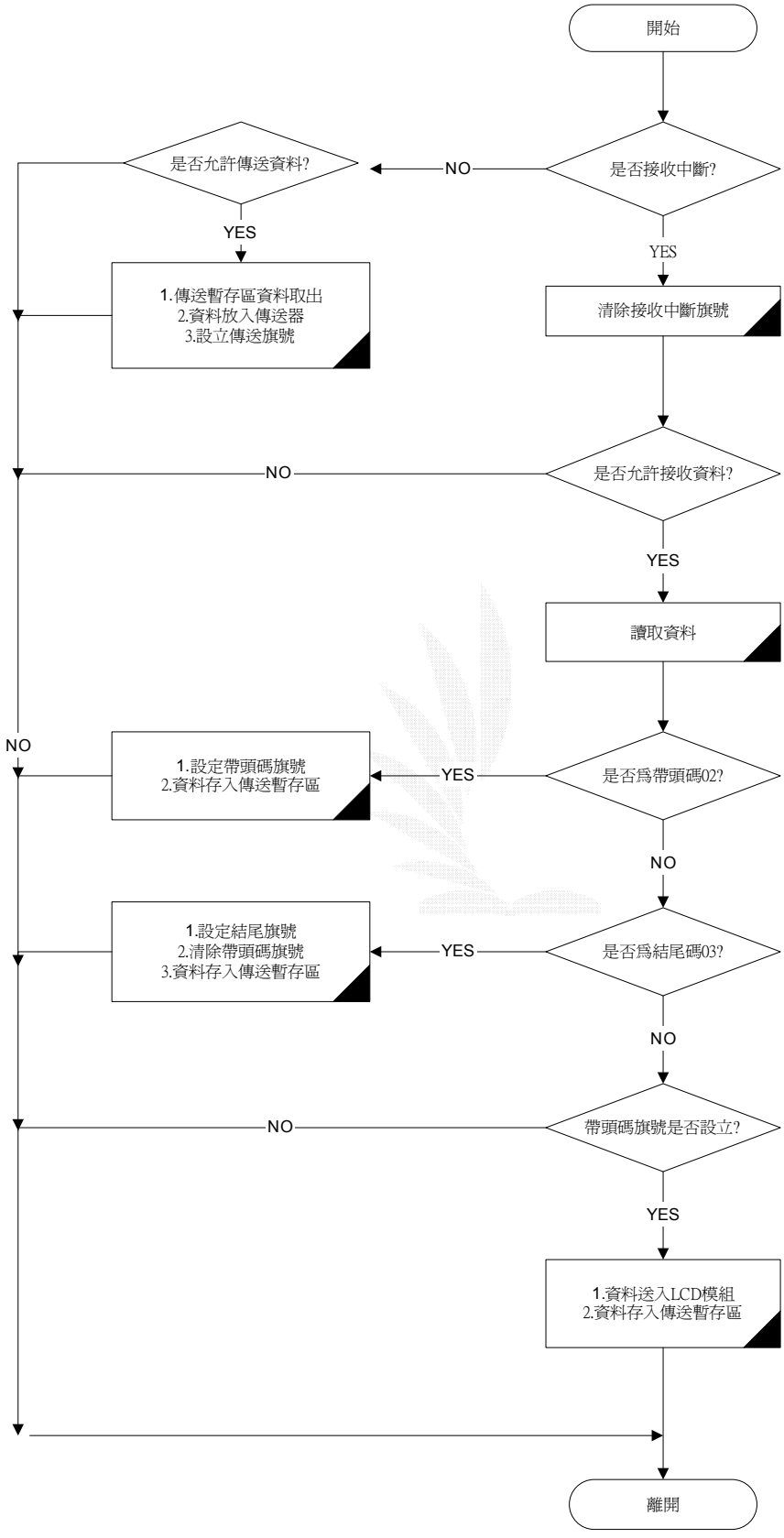
這個架構圖說明了單晶片和其他模組的工作關希,主要是由感應模組接收到卡片資料，傳道 8951 作處理再由 8951 傳至 LCD 顯示卡號，也經由 8951 經過串列介面處理傳輸到電腦做資料庫管理,其中門鎖部分因為門鎖電路不是這次專題討論的重點，所以已 LED 代表接收開門訊號，以其明滅狀態顯示門鎖木狀態是開或關，電源部分則採用 7.5 伏特，300mA 的電源供應。

4.2 系統電路圖



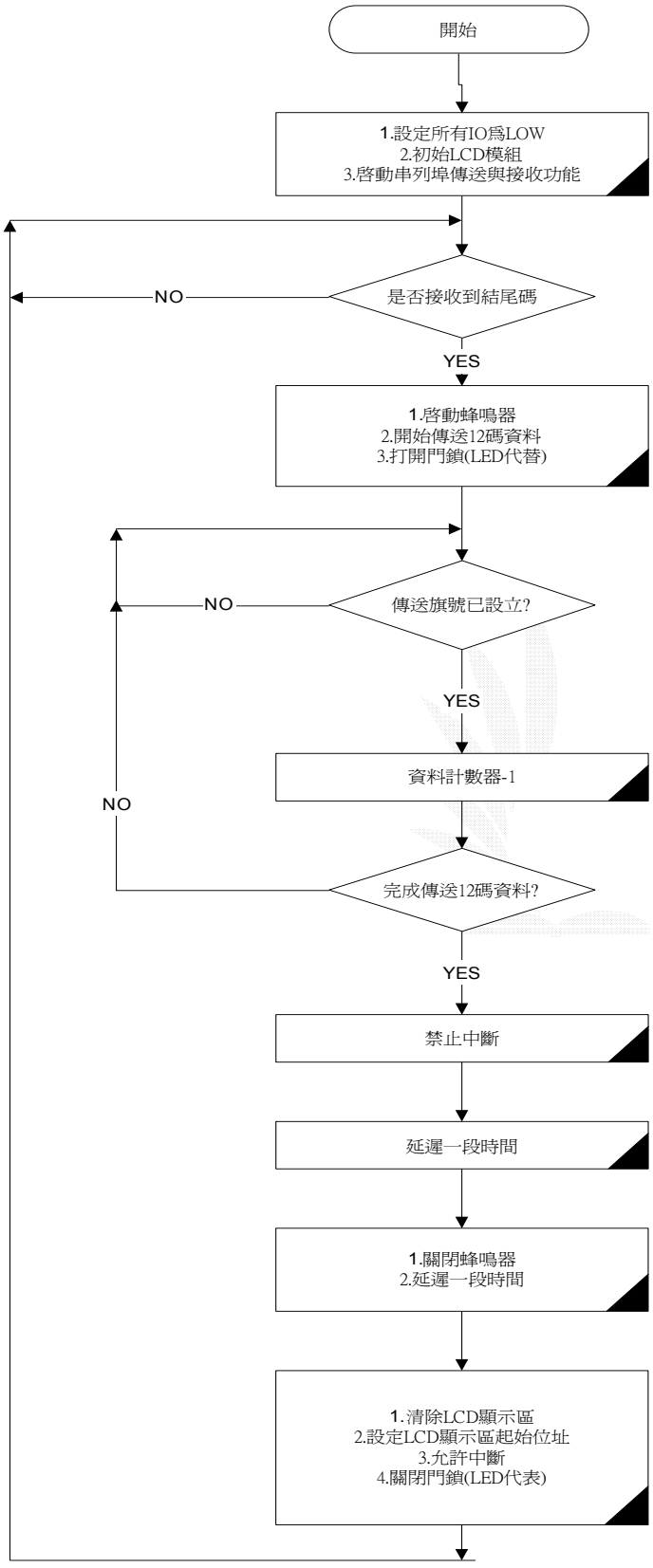
（圖 4.2 系統電路圖）

4.3 8951 之 UART 中斷流程圖



（圖 4.3 8951UART 中斷流程圖）

4.4 8951 之主程式中斷流程圖



（圖 4.4 8951 之主程式中斷流程圖）

4.5 8051 程式碼及其說明

```
;===================================
;   LCD FINCTION SET VALUE
;===================================
FUNSET      EQU 00111000B   ;設定資料長度為８位元,２列顯示行,５＊１０
矩陣字型
SHOWSET     EQU 00001100B   ;顯示開啟,游標不顯示不閃爍
MODESET     EQU 00000110B   ;游標右移,位置加一
CLEARSET    EQU 00000001B   ;清除ＬＣＤ
READYADR    EQU 11000101B   ;秀出ＲＥＡＤＹ位置
DATAADDR    EQU 10000011B   ;秀出卡片資料的位置
HOMEADR     EQU 10000000B   ;第一行第一個位置
IR_ADDR     EQU 00H     ;ＬＣＤ命令暫存器位置
DR_ADDR     EQU 01H     ;ＬＣＤ資料暫存器位置
TEMPADR     EQU 30H     ;傳送資料暫存位置
;
SHOWFG      EQU     00H     ;已將一個ＢＹＴＥ的資料顯示在ＬＣＤ的旗號
LEADFG      EQU     01H     ;已讀到帶頭碼０２的旗號
TXFG        EQU 02H     ;已將一個ＢＹＴＥ的資料送出的旗號
UNRXFG      EQU 03H     ;不允許讀取ＲＳ２３２埠的資料
ENDFG       EQU 04H     ;已讀到結尾碼０３的旗號
;
BUZZER      EQU     A0H     ;Ｐ２．０驅動蜂鳴器
LED     EQU     A1H     ;Ｐ２．１驅動ＬＥＤ
LCDRS       EQU     A5H     ;Ｐ２．５ＬＣＤ的ＲＳ腳
LCDRW       EQU     A6H     ;Ｐ２．６ＬＣＤ的Ｒ／Ｗ腳

LCDEN       EQU     A7H     ;Ｐ２．７ＬＣＤ的ＥＮＢ腳
RFID        EQU     B0H     ;ＲＳ２３２的接收
RS232       EQU     B1H     ;ＲＳ２３２的發送
;
BEEP        EQU 90H     ;門的開關 0 開 1 關
;
    ORG 00H     ;
    SJMP    START       ;
    ORG 23H     ;
    AJMP    UARTINT     ;
```

```
    ORG 30H      ;
START:
;-----------------------------------------------
;   主程式初始設定
;-----------------------------------------------
    MOV SP,#5FH      ;設定堆疊起始位置
    MOV P0,#00H      ;設定ＬＯＷ
    MOV P1,#00H      ;設定ＬＯＷ
    MOV P2,#00H      ;設定ＬＯＷ
    SETB    RFID         ;設定ＨＩ
    SETB    RS232        ;設定ＨＩ
        SETB    BEEP         ;關門
;
    MOV 20H,#0       ;所有旗號清為０
;
    CALL    Init_LCD     ;初始ＬＣＤ
;
    CALL    INITUART     ;初始ＵＡＲＴ功能
    CALL    SHOWREADY
    MOV A,#DATAADDR ;設定顯示資料的位置
    CALL    PutIR        ;寫入ＬＣＤ命令暫存器
    MOV IE,#10010000B    ;致能中斷
;-----------------------------------------------
;   等待中斷程式收到１２個位元組
;-----------------------------------------------

GETRFID:
    JNB ENDFG,GETRFID    ;等待接收到結尾碼
    CLR ENDFG
;
    SETB    BUZZER       ;啟動蜂鳴器
;-----------------------------------------------
;   開始傳送１２個位元組
;-----------------------------------------------
    SETB    UNRXFG       ;設定不再讀取ＲＳ２３２埠的資料
    CALL    CHECKCODE
    MOV R4,#0CH      ;設定傳送１２個ＢＹＴＥＳ的長度計數器
    MOV R0,#TEMPADR ;設定待傳送資料位置
```

```
    MOV A,@R0       ;取得第一個欲傳送的資料
    MOV SBUF,A      ;放入傳送暫存器
    SETB    TXFG        ;設定已經傳送旗號
;-----------------------------------------------
;   等待中斷程式傳送其他位元組
;-----------------------------------------------
TX1:
    JNB TXFG,TX1    ;已經傳送旗號是否被設定？
    CLR TXFG        ;是則清除已經傳送旗號
    DJNZ    R4,TX1      ;長度計數器減一,不為零則繼續等待
;-----------------------------------------------
;   所顯示的資料延遲一段時間後清除
;-----------------------------------------------
    MOV IE,#00      ;禁止任何中斷
    MOV R5,#5       ;延遲一段時間
    CALL    DELAY       ;
    CLR BUZZER      ;關閉蜂鳴器
    MOV R5,#90      ;延遲一段時間
    CALL    DELAY       ;
;
    CALL    SHOWEMPTY   ;清除ＬＣＤ所顯示的資料
;
    MOV A,#DATAADDR ;重新設定顯示資料的位置
    CALL    PutIR       ;寫入ＬＣＤ命令暫存器
;
    MOV 20H,#0      ;清除所有旗號
    MOV IE,#10010000B   ;允許ＵＡＲＴ中斷
;
    MOV R5,#5
    CALL    DELAY
    CLR BEEP
    MOV R5,#100
    CALL    DELAY
    MOV R5,#100
    CALL    DELAY
    MOV R5,#100
    CALL    DELAY
    MOV R5,#100
```

```
    CALL    DELAY
    SETB    BEEP
    JMP GETRFID
;
;***********************************************
;   UARTINT     中斷程式
;***********************************************
UARTINT:
    PUSH    A       ;
    JBC TI,TXINT    ;是否發送中斷？是則跳至發送服務程式
RXINT:
    CLR RI      ;是接收中斷,清除接收中斷旗號
    JB  UNRXFG,UARTEND  ;是否不允許接收街資料？是則離開中斷程式
    MOV A,SBUF      ;讀取接收資料
    CJNE    A,#02H,R_1  ;比對是否為０２帶頭碼
    SETB    LEADFG      ;是則設定帶頭碼旗號
    CLR ENDFG
    MOV R0,#TEMPADR ;取得欲傳送資料暫存位置
    MOV @R0,A       ;將資料放入
    SJMP    UARTEND     ;離開中斷程式
R_1:
    CJNE    A,#03H,R_2  ;比對是否為０３結尾碼
    CLR LEADFG      ;是則清除帶頭碼旗號
    SETB    ENDFG
    INC R0      ;
    MOV @R0,A       ;將資料放入欲傳送資料暫存位置
    SJMP    UARTEND     ;
R_2:
    JB  LEADFG,R_3  ;帶頭碼旗號是否設定？
    SJMP    UARTEND     ;否,則離開中斷程式
R_3:
    INC R0      ;是,則將資料放入欲傳送資料暫存位置
    MOV @R0,A       ;
    CALL    PutDR       ;將資料ＬＣＤ資料暫存器位置
    SETB    SHOWFG      ;
    SJMP    UARTEND     ;
;
TXINT:              ;發送服務程式
```

```
    JNB UNRXFG,UARTEND
    INC R0      ;
    MOV A,@R0       ;欲傳送資料取出
    MOV SBUF,A      ;放入傳送暫存器
    SETB    TXFG        ;
    CALL    CHECKCODE
UARTEND:
    POP A       ;
    RETI
;************************************************
;   INITIAL  LCD
;************************************************
Init_LCD:
    MOV A,#FUNSET   ;設定資料長度為８位元,２列顯示行,５＊１０矩陣字型
    CALL    PutIR       ;寫入ＬＣＤ命令暫存器
;
    MOV A,#SHOWSET  ;顯示開啟,游標不顯示不閃爍
    CALL    PutIR       ;寫入ＬＣＤ命令暫存器
;
    MOV A,#MODESET  ;游標右移,位置加一
    CALL    PutIR       ;寫入ＬＣＤ命令暫存器
;
    RET
;************************************************
;   PutDR
;************************************************
PutDR:
    CALL    CheckBusy   ;等待ＬＣＤ空閒
    SETB    LCDRS       ;設定ＬＣＤ的ＲＳ為ＨＩ
    CLR LCDRW       ;設定ＬＣＤ的Ｒ／Ｗ為ＬＯＷ
    MOV P0,A        ;將ＡＣＣ的值寫入Ｐ０
    SETB    LCDEN       ;啟動ＬＣＤ讀取資料
    NOP         ;延遲一段時間
    NOP
    NOP
    NOP
    NOP
    NOP
```

```
    NOP
    NOP
    NOP
    NOP
    CLR LCDEN       ;關閉ＬＣＤ讀取資料動作
;
    RET
;***********************************************
;   PutIR
;***********************************************
PutIR:
    CALL    CheckBusy   ;等待ＬＣＤ空閒
    CLR LCDRS       ;設定ＬＣＤ的ＲＳ為ＬＯＷ
    CLR LCDRW       ;設定ＬＣＤ的Ｒ／Ｗ為ＬＯＷ
    MOV P0,A        ;將ＡＣＣ的值寫入Ｐ０
    SETB    LCDEN       ;啟動ＬＣＤ讀取資料
    NOP         ;延遲一段時間
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    CLR LCDEN       ;關閉ＬＣＤ讀取資料動作
;
    RET
;
;***********************************************
;   CheckBusy
;***********************************************
CheckBusy:
    PUSH    A       ;
    CLR LCDRS       ;設定ＬＣＤ的ＲＳ為ＬＯＷ
    SETB    LCDRW       ;設定ＬＣＤ的Ｒ／Ｗ為ＨＩ
    SETB    LCDEN       ;啟動ＬＣＤ送出資料
```

```
;
WAIT:
    MOV A,P0          ;讀取ＬＣＤ資料
    JB  A.7,WAIT      ;若ＢＩＴ７不為ＨＩ則等待
;
    CLR LCDEN         ;關閉ＬＣＤ送出資料動作
    POP A         ;
;
    RET

;***********************************************
;   PRINTSTRING
;***********************************************
PRINTSTRING:
    PUSH    A
PRT_LOOP:
    CLR A         ;清除ＡＣＣ
    MOVC    A,@A+DPTR    ;將字串讀入
    CJNE    A,#'$',NEXT ;比較是否為” $”
    SJMP    ENDPRT       ;是,則離開
NEXT:
    CALL    PutDR        ;否,則將資料ＬＣＤ資料暫存器位置
    INC DPTR          ;指標加一
    SJMP    PRT_LOOP     ;繼續
ENDPRT:
    POP A
    RET
;***********************************************
;   INITUART
;***********************************************
INITUART:
    MOV SCON,#01010000B ;串列埠工作於模式１且致能接收
    MOV TMOD,#00100000B ;計時器工作於模式２
    MOV TH1,#253      ;設定９６００鮑率
    SETB    TR1       ;啟動計時器一
    MOV IE,#00        ;禁能中斷
    RET
;***********************************************
```

```
;    DELAY
;***********************************************
DELAY:
    MOV R6,#50       ;
$1:
    MOV R7,#100      ;
$2:
    DJNZ    R7,$2        ;
    DJNZ    R6,$1        ;
    DJNZ    R5,DELAY     ;
    RET
;***********************************************
;    SHOWREADY
;***********************************************
SHOWREADY:
    MOV A,#READYADR ;設定顯示ＲＥＡＤＹ的位置
    CALL    PutIR        ;寫入ＬＣＤ命令暫存器
    MOV A,#READYADR ;設定顯示ＲＥＡＤＹ的位置
    CALL    PutIR        ;寫入ＬＣＤ命令暫存器
;
    MOV DPTR,#READYSTR  ;取得 READYSTR 位置
    CALL    PRINTSTRING ;顯示ＲＥＡＤＹ
    RET
;***********************************************
;    SHOWEMPTY
;***********************************************
SHOWEMPTY:
    MOV A,#HOMEADR  ;第一行第一個位置
    CALL    PutIR        ;寫入ＬＣＤ命令暫存器
    MOV DPTR,#EMPTYSTR  ;取得空白字串位置
    CALL    PRINTSTRING ;清除所顯示的資料
    RET
;***********************************************
;       CLEAR  LCD
;***********************************************
CLRLCD:
        MOV     A,#CLEARSET
        CALL    PutIR
```

```
        RET
;************************************************
;       CHECK CODE
;************************************************
CHECKCODE:
    CLR BEEP
CC_END:
    RET
;************************************************
;       DATA
;************************************************
READYSTR DB  'READY','$'
EMPTYSTR DB  '                ','$'
       END
```

第五章：系統操作說明及錯誤檢查

5.1 感應主機部分

首先要檢查所有零件組是否有接好有無反插，如有焊接部分是否有假焊的情形發生，接線是否正確，確實接上 7.5V 左右的電源電流大於 300mA。

當電源供應後：

＊LED 是否有亮

＊蜂鳴器是否叫一聲

＊LCD 是否顯示 READY

以上代表 8951 工作正常，接著拿著卡片接近線圈，感應卡片。

＊ 蜂鳴器是否有叫一聲

＊ LCD 有無顯示卡片號碼

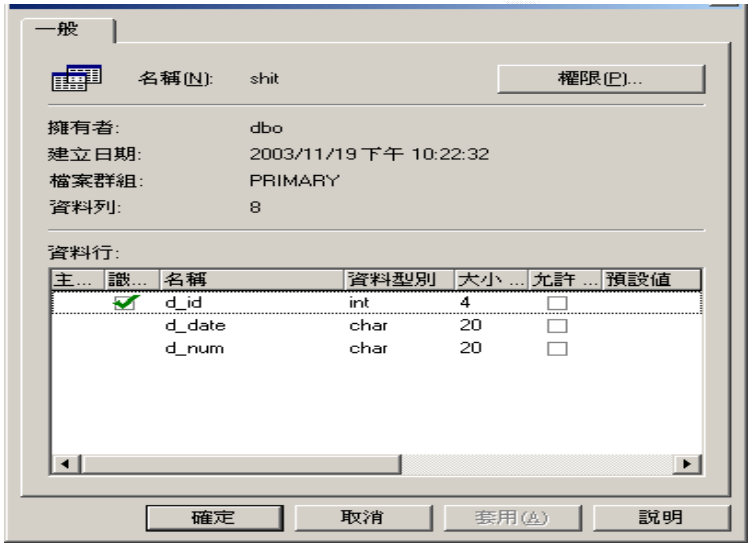
＊ 代表大門開關的 LED 是否熄滅

接著測試另一張卡片完成測試。

5.2 JAVA 應用程式部分

連上電腦執行連線測試，是否可以接收到卡片資料，代表 RS232 電位轉換是否成功，以下是軟體操作步驟：

首先資料庫會被存在一個表格內其表格欄位設定如下:



d_id 是代表資料為第幾筆資料。

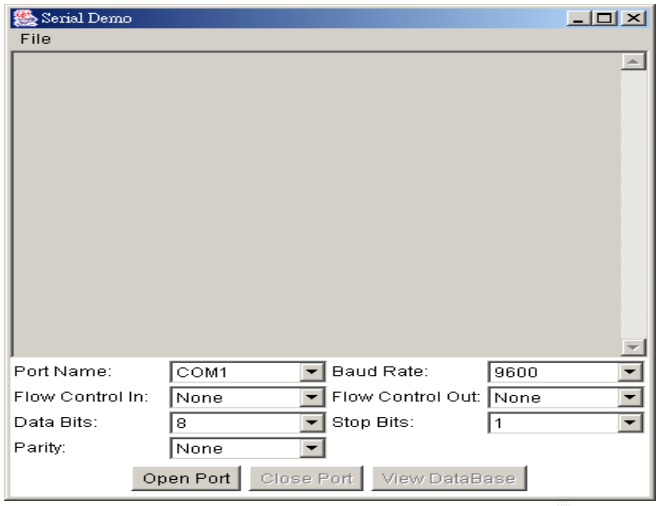
d_data 是代表時間還有日期。

d_num 是代表卡號也進出人員的身分代表。

在執行 Dos 下執行 java SerialDemo 如下圖所示:

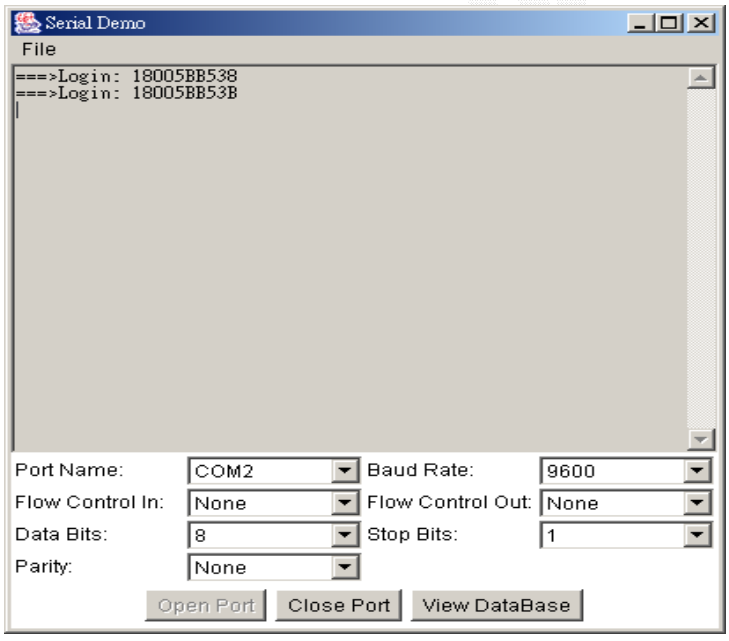
```
C:\SerialDemo>
C:\SerialDemo>
C:\SerialDemo>
C:\SerialDemo>java SerialDemo
```

執行後會出現下面視窗：

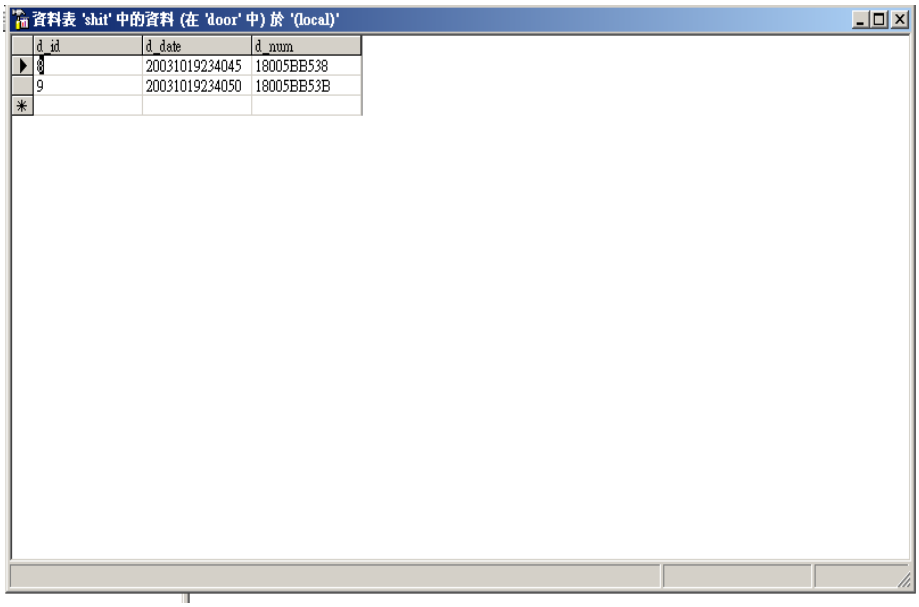


然後按下 Open Port 按鍵。

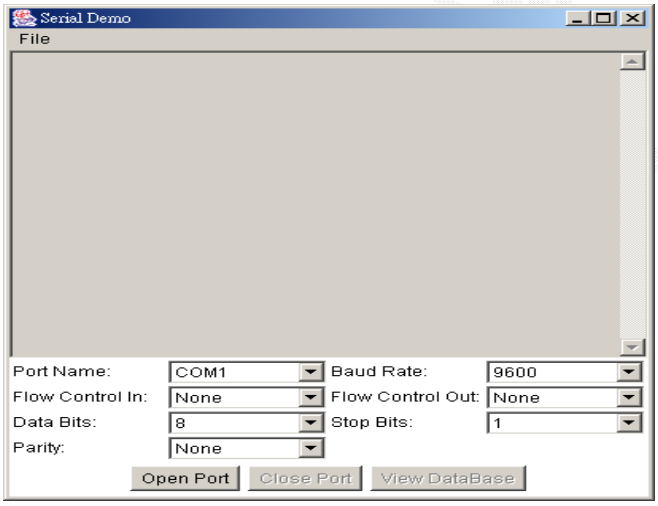
開始感應測試…先測試兩次，下圖顯示成功讀取 2 筆：



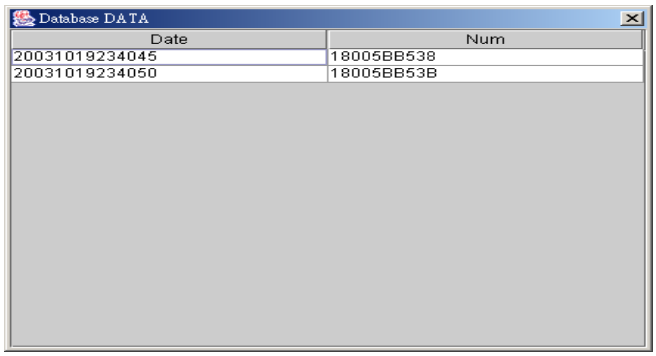
資料庫表格成功存入 2 筆資料如下圖所示：



其中時間資料 20031019234045=2003 年 10 月 19 號 23 點 40 分 45 秒。
其順序為年、月、日、時、分、秒。
其後的資料 18005BB538 是測試的卡片資料。

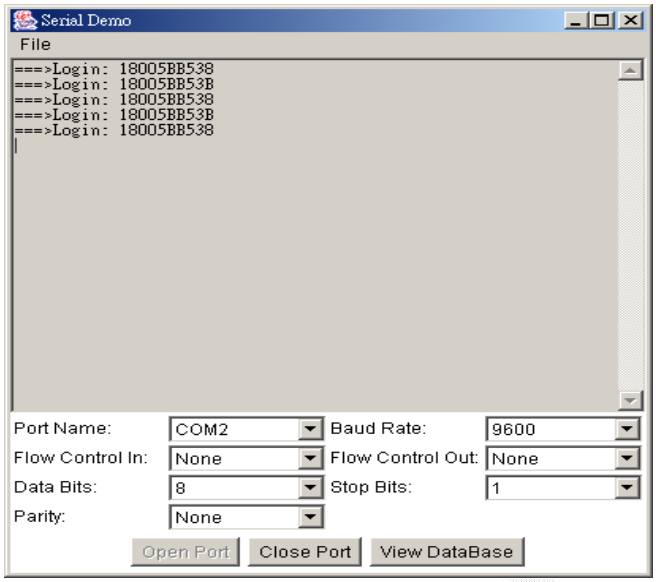


按下上圖的 View DataBase 按鍵可以觀看資料庫明細表如下圖(圖於下頁)：

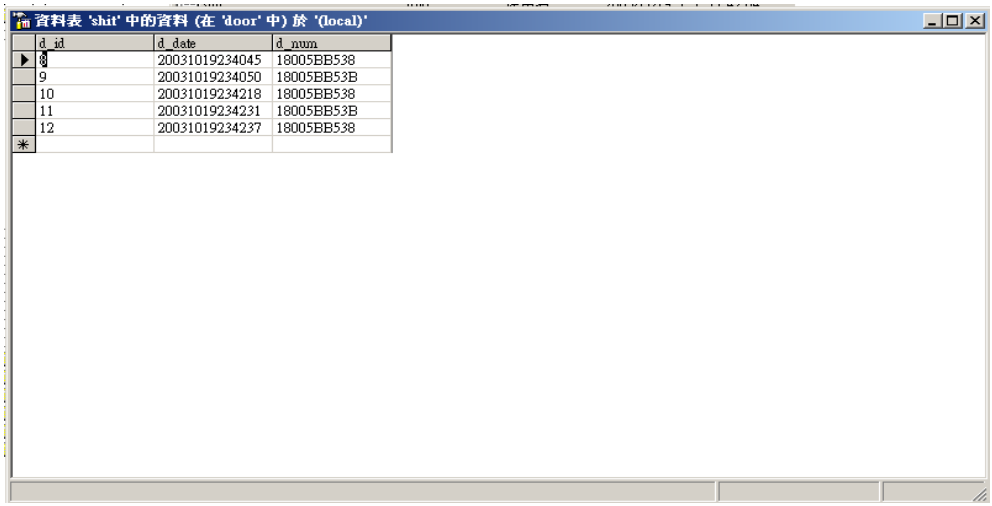


接著在讀取 3 筆資料看對照是否資料傳輸正確：

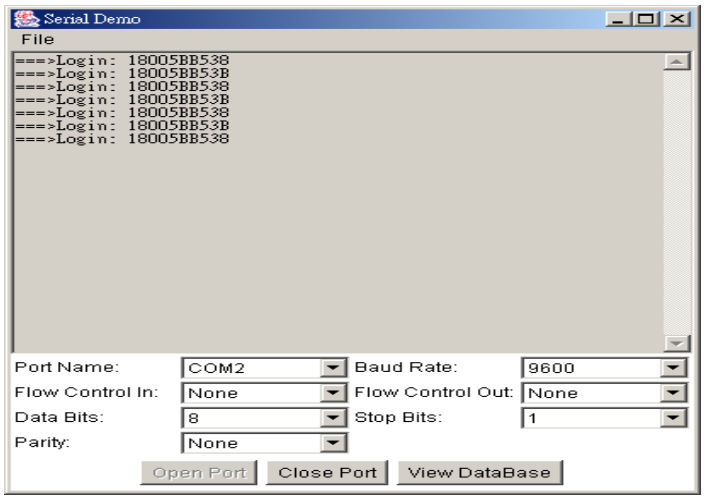
檢查讀取卡號是否無誤：

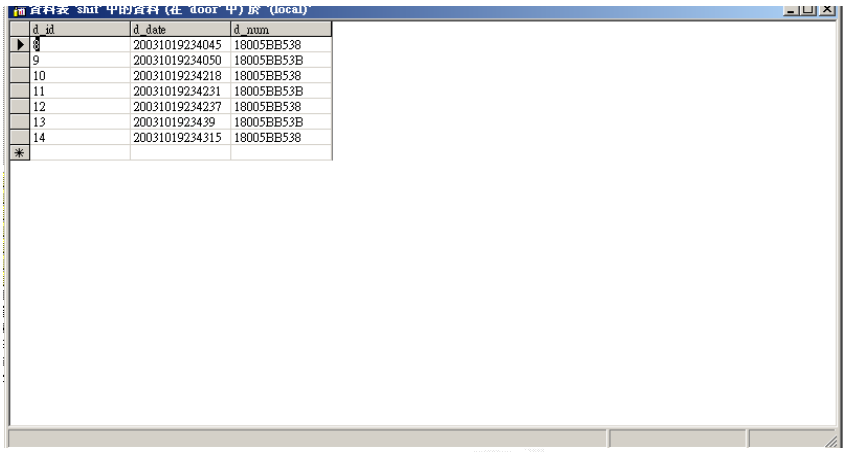


檢查資料庫傳輸紀錄的資料是否正確：

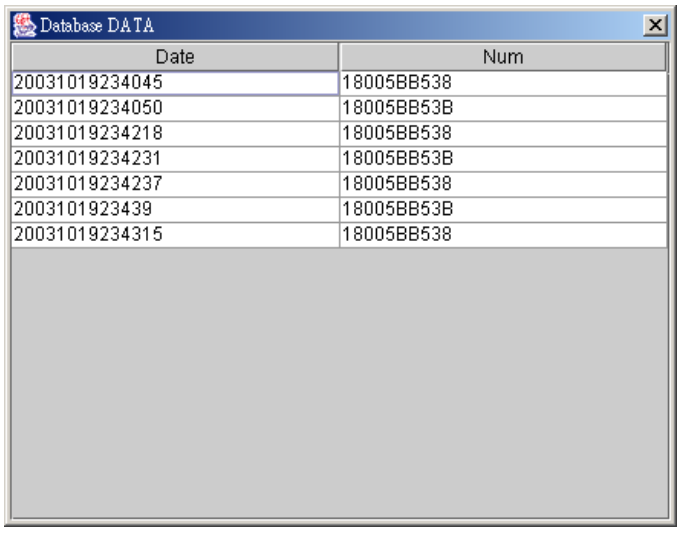


最後在測試 2 筆進行最後的資料比對：





d_id	d_date	d_num
8	20031019234045	18005BB538
9	20031019234050	18005BB53B
10	20031019234218	18005BB538
11	20031019234231	18005BB53B
12	20031019234237	18005BB538
13	2003101923439	18005BB53B
14	20031019234315	18005BB538



Database DATA

Date	Num
20031019234045	18005BB538
20031019234050	18005BB53B
20031019234218	18005BB538
20031019234231	18005BB53B
20031019234237	18005BB538
2003101923439	18005BB53B
20031019234315	18005BB538

比對無誤後完成測試！

5.3 錯誤檢查

當 8951 工作正常卻無法接收卡片資料時，應檢查：

＊ RD1085 是否有反插的情況

＊ 諧振電容 332pf 是否正確

＊ 天線是否因絕緣漆刮不乾淨造成假焊

＊ 再 RD1085 的第一個接腳即和諧振電容兩天線的串接點應可量到 125KHZ 的近似正弦波

當 8951 無法正常工作時

＊確認 5V 電壓是否存在

＊RESET 功能是否正常可用示波器測量是否電源 ON 時 555 的第三根接腳隔幾個 mS 才起來

＊石英體是否正常工作可用示波器測量是否有振盪訊號

＊程式是否燒錄正確

電腦收不到資料時

＊用示波器測量電路板上的 9 針 D 型做的第 2 腳是否有訊號

＊連接線是否有錯誤不可跳線

＊程式連接埠是否接正確

＊程式是否開啟連接埠

心得

為了完成這一個專題可以說是上網找了許多的資料,再找資料的同時也看到許多公司所生產的感應卡系列產品,功能非常的多樣化,深深感覺到相同的電子設備,可經由一些巧思以及更好的技術來使產品顯的用途廣泛,功能更強大,也讓我知道這次專題其實還有很多的功能可以被運用到。不過也借由這次專題的製作以可以讓我了解學習到 8951 運用在電子產品上的廣泛用途這次專體也讓我了解到 8951 的控制、RS232 的電位轉換的串列控制、LCD 的顯示的控制、感應卡的原理以及感應卡的種類、什麼是 RFID、也學到了如何利用好用的 JAVA 來製作一個簡單接收 RS232 資料的應用程式,雖然說對每一項的了解不算是很深入,不過對我的幫助真的很大,像這次請教同學 JAVA 的問題,才赫然實際的體驗到 JAVA 的功能如此的強大且齊全,這次專題當然也遇到很多的小難題,尤其是沒什麼焊接經驗的我,就好幾次把電子零件給焊接壞掉,這次我也學到了 8951 的燒錄方式,還有剛剛提到的 JAVA 程式一直無法收到卡片資料,這也是我跟同學研究了好一陣子才得以解決,不過也因為問題越多代表學的也越多,更認識了很多以往沒見過的 IC 以及電子材料,往常用的感應卡雖然常見,但是對他的構造和原理也都不是很懂,其實很多身邊常見的東西都是 8951 的運用有著密切的關希,學會 8951 的一些控制的確對自己的幫助很大,對以後找工作的相關背景知識也多一層了解。

附錄 A JAVA 應用程式

```
import javax.comm.*;
import java.awt.*;
import java.awt.event.*;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.FileNotFoundException;
import java.util.Properties;
import java.util.Enumeration;
/**
```

這個程式的主要目的，是從 com port 讀取由門卡管制管置所傳遞過來的門卡卡號，

當程式被執行時，首先需要設定下列項目:

Port: (com1、 com2、....)

Flow Control In: (None、Xon/Xoff In、RTS/CTS In)

Data Bits: (5、6、7、8)

Parity: (None、Even、Odd)

Baud Rate: (300、2400、9600、14400、28800....)

Flow Control Out: (None、Xon/Xoff In、RTS/CTS In)

Stop Bits: (1、1.5、2)

理論上，程式已有設定預設值，若有特別情況時，則再另行選擇。

程式一被執行，及馬上利用 java JDBC 來連結資料庫，資料庫可以爲遠端，也可以是本地端，不過預設是位於本地端的 MS SQL server 2000。

資料庫的表格(table)欄位是長成這樣子:

d_id: 第幾筆資料的索引

d_date: 此筆資料填進資料庫的「日期」

d_num: 此筆資料的門卡資料。

當程式按下「Open Port」後，程式就占住所選擇之 Com port 的使用權，且進入 Listen 的狀態，當有門卡刷過裝置時，則 8051 會將此門卡的卡號透過 RS232 傳到電腦中的此程式，則當程式接收到這個訊號後，就將此卡號顯示在視窗中，且馬上填入到資料庫中

當按下「Close Port」後，程式就釋放所霸占住的 Com port 使用權，如此別的程式才可使用此 Port。

當按下「View DataBase」時，程式會彈跳出一個額外的視窗，此視窗中的資料是讀取由之前程序所存進的資料，借以了解什麼時候有使用觸動此裝置。

SerialDemo.java:

```
public class SerialDemo extends Frame implements ActionListener {
    final int HEIGHT = 450;
```

```
final int WIDTH    = 410;
private MenuBar mb;
private Menu fileMenu;
private MenuItem openItem;
private MenuItem saveItem;
private MenuItem exitItem;
private Button openButton;
private Button closeButton;
private Button breakButton;
private Panel buttonPanel;
private Panel       messagePanel;
private TextArea messageAreaOut;
private TextArea messageAreaIn;
private ConfigurationPanel configurationPanel;
private SerialParameters parameters;
private SerialConnection connection;
private Properties props = null;
/**
          主程式(程式的進入點，由此建構出整個程式的視窗。
*/
public static void main(String[] args) {
   if ((args.length > 0)
         && (args[0].equals("-h")
         || args[0].equals("-help"))) {
         System.out.println("usage: java SerialDemo [configuration File]");
         System.exit(1);
   }
   SerialDemo serialDemo = new SerialDemo(args);
   serialDemo.setVisible(true);
   serialDemo.repaint();
}
/**
          程式的建構式，設定程式初值的地方。
   */
public SerialDemo(String[] args){
   super("Serial Demo");
   parameters = new SerialParameters();
   addWindowListener(new CloseHandler(this));
```

```
mb = new MenuBar();
fileMenu = new Menu("File");
openItem = new MenuItem("Load");
openItem.addActionListener(this);
fileMenu.add(openItem);
saveItem = new MenuItem("Save");
saveItem.addActionListener(this);
fileMenu.add(saveItem);
exitItem = new MenuItem("Exit");
exitItem.addActionListener(this);
fileMenu.add(exitItem);
mb.add(fileMenu);
setMenuBar(mb);
messagePanel = new Panel();
//messagePanel.setLayout(new GridLayout(2, 1));
messagePanel.setLayout(new GridLayout(1,1));
/*
messageAreaOut = new TextArea();
messagePanel.add(messageAreaOut);
*/
messageAreaIn = new TextArea();
messageAreaIn.setEditable(false);
messagePanel.add(messageAreaIn);
add(messagePanel, "Center");
configurationPanel = new ConfigurationPanel(this);
buttonPanel = new Panel();
openButton = new Button("Open Port");
openButton.addActionListener(this);
buttonPanel.add(openButton);
closeButton = new Button("Close Port");
closeButton.addActionListener(this);
closeButton.setEnabled(false);
buttonPanel.add(closeButton);
//breakButton = new Button("Send Break");
breakButton = new Button("View DataBase");
breakButton.addActionListener(this);
breakButton.setEnabled(false);
buttonPanel.add(breakButton);
Panel southPanel = new Panel();
```

```
    GridBagLayout gridBag = new GridBagLayout();
    GridBagConstraints cons = new GridBagConstraints();
    southPanel.setLayout(gridBag);
    cons.gridwidth = GridBagConstraints.REMAINDER;
    gridBag.setConstraints(configurationPanel, cons);
    cons.weightx = 1.0;
    southPanel.add(configurationPanel);
    gridBag.setConstraints(buttonPanel, cons);
    southPanel.add(buttonPanel);
    add(southPanel, "South");
    parseArgs(args);
    //connection = new SerialConnection(this, parameters,
    //                                  messageAreaOut, messageAreaIn);
    connection = new SerialConnection(this, parameters,
                                      messageAreaIn);
    setConfigurationPanel();
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    setLocation(screenSize.width/2 - WIDTH/2,
                screenSize.height/2 - HEIGHT/2);
    setSize(WIDTH, HEIGHT);
}
/**
Sets the GUI elements on the configurationPanel.
*/
public void setConfigurationPanel() {
    configurationPanel.setConfigurationPanel();
}
/**
Responds to the menu items and buttons.
*/
public void actionPerformed(ActionEvent e) {
    String cmd = e.getActionCommand();
    // Loads a configuration file.
    if (cmd.equals("Load")) {
        if (connection.isOpen()) {
            AlertDialog ad = new AlertDialog(this, "Port Open!",
                                             "Configuration may not",
                                             "be loaded",
                                             "while a port is open.");
```

```
        } else {
            FileDialog fd = new FileDialog(this,
                                           "Load Port Configuration",
                                           FileDialog.LOAD);
            fd.setVisible(true);
            String file = fd.getFile();
            if (file != null) {
                String dir = fd.getDirectory();
                File f = new File(dir + file);
                try {
                    FileInputStream fis = new FileInputStream(f);
                    props = new Properties();
                    props.load(fis);
                    fis.close();
                } catch (FileNotFoundException e1) {
                    System.err.println(e1);
                } catch (IOException e2) {
                    System.err.println(e2);
                }
                loadParams();
            }
        }

    // Saves a configuration file.
    if (cmd.equals("Save")) {
        configurationPanel.setParameters();
        FileDialog fd = new FileDialog(this, "Save Port Configuration",
                                       FileDialog.SAVE);
        fd.setFile("serialdemo.properties");
        fd.setVisible(true);
        String fileName = fd.getFile();
        String directory = fd.getDirectory();
        if ((fileName != null) && (directory != null)) {
            writeFile(directory + fileName);
        }
    }
    // Calls shutdown, which exits the program.
    if (cmd.equals("Exit")) {
        shutdown();
```

```
    }
    // Opens a port.
    if (cmd.equals("Open Port")) {
        openButton.setEnabled(false);
        //Cursor previousCursor = getCursor();
        //setNewCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));
        configurationPanel.setParameters();
        try {
           connection.openConnection();
        } catch (SerialConnectionException e2) {
                AlertDialog ad = new AlertDialog(this,
                                "Error Opening Port!",
                                "Error opening port,",
                                e2.getMessage() + ".",
                                "Select new settings, try again.");
             openButton.setEnabled(true);
                //setNewCursor(previousCursor);
                return;
        }
        portOpened();
        //setNewCursor(previousCursor);
    }
    // Closes a port.
    if (cmd.equals("Close Port")) {
        portClosed();
    }
    // Sends a break signal to the port.
    if (cmd.equals("View DataBase")) {
       // connection.sendBreak();
AdvancedTableExample ddview    = new AdvancedTableExample();
ddview.setVisible( true );        }
}
/**
當按下「Open Port」按鈕時，所呼叫的函式
*/
public void portOpened() {
    openButton.setEnabled(false);
    closeButton.setEnabled(true);
    breakButton.setEnabled(true);
```

```
}
/**
        當按下「Close Port」按鈕時，所呼叫的函式
*/
public void portClosed() {
   connection.closeConnection();
   openButton.setEnabled(true);
   closeButton.setEnabled(false);
   breakButton.setEnabled(false);
}
/**
Sets the <code>Cursor</code> for the application.
@param c New <code>Cursor</code>
*/
private void setNewCursor(Cursor c) {
   setCursor(c);
   messageAreaIn.setCursor(c);
   messageAreaOut.setCursor(c);
}
private void writeFile(String path) {
      Properties newProps;
      FileOutputStream fileOut = null;
      newProps = new Properties();
      newProps.put("portName", parameters.getPortName());
      newProps.put("baudRate", parameters.getBaudRateString());
      newProps.put("flowControlIn", parameters.getFlowControlInString());
      newProps.put("flowControlOut", parameters.getFlowControlOutString());
      newProps.put("parity", parameters.getParityString());
      newProps.put("databits", parameters.getDatabitsString());
      newProps.put("stopbits", parameters.getStopbitsString());
      try {
            fileOut = new FileOutputStream(path);
      } catch (IOException e) {
            System.out.println("Could not open file for writiing");
      }
      newProps.save(fileOut, "Serial Demo poperties");
      try {
            fileOut.close();
      } catch (IOException e) {
```

```
                System.out.println("Could not close file for writiing");
            }
        }
        private void shutdown() {
            connection.closeConnection();
            System.exit(1);
        }
        private void parseArgs(String[] args) {
            if (args.length < 1) {
                return;
            }
            File f = new File(args[0]);
            if (!f.exists()) {
                f = new File(System.getProperty("user.dir")
                        + System.getProperty("path.separator")
                        + args[0]);
            }
            if (f.exists()) {
                try {
                    FileInputStream fis = new FileInputStream(f);
                    props = new Properties();
                    props.load(fis);
                    fis.close();
                    loadParams();
                } catch (IOException e) {
                }
            }
        }
        /**
        Set the parameters object to the settings in the properties object.
        */
        private void loadParams() {
            parameters.setPortName(props.getProperty("portName"));
            parameters.setBaudRate(props.getProperty("baudRate"));
            parameters.setFlowControlIn(props.getProperty("flowControlIn"));
            parameters.setFlowControlOut(props.getProperty("flowControlOut"));
            parameters.setParity(props.getProperty("parity"));
            parameters.setDatabits(props.getProperty("databits"));
            parameters.setStopbits(props.getProperty("stopbits"));
```

```
        setConfigurationPanel();
    }
    class ConfigurationPanel extends Panel implements ItemListener {
        private Frame parent;
        private Label portNameLabel;
        private Choice portChoice;
        private Label baudLabel;
        private Choice baudChoice;
        private Label flowControlInLabel;
        private Choice flowChoiceIn;
        private Label flowControlOutLabel;
        private Choice flowChoiceOut;
        private Label databitsLabel;
            private Choice databitsChoice;
        private Label stopbitsLabel;
            private Choice stopbitsChoice;
        private Label parityLabel;
            private Choice parityChoice;
        public ConfigurationPanel(Frame parent) {
            this.parent = parent;
            setLayout(new GridLayout(4, 4));
            portNameLabel = new Label("Port Name:", Label.LEFT);
            add(portNameLabel);
            portChoice = new Choice();
            portChoice.addItemListener(this);
            add(portChoice);
            listPortChoices();
            portChoice.select(parameters.getPortName());
            baudLabel = new Label("Baud Rate:", Label.LEFT);
            add(baudLabel);
            baudChoice = new Choice();
            baudChoice.addItem("300");
            baudChoice.addItem("2400");
            baudChoice.addItem("9600");
            baudChoice.addItem("14400");
            baudChoice.addItem("28800");
            baudChoice.addItem("38400");
            baudChoice.addItem("57600");
            baudChoice.addItem("152000");
```

```
baudChoice.select(Integer.toString(parameters.getBaudRate()));
baudChoice.addItemListener(this);
add(baudChoice);
flowControlInLabel = new Label("Flow Control In:", Label.LEFT);
add(flowControlInLabel);
flowChoiceIn = new Choice();
flowChoiceIn.addItem("None");
flowChoiceIn.addItem("Xon/Xoff In");
flowChoiceIn.addItem("RTS/CTS In");
flowChoiceIn.select(parameters.getFlowControlInString());
flowChoiceIn.addItemListener(this);
add(flowChoiceIn);
flowControlOutLabel = new Label("Flow Control Out:", Label.LEFT);
add(flowControlOutLabel);
flowChoiceOut = new Choice();
flowChoiceOut.addItem("None");
flowChoiceOut.addItem("Xon/Xoff Out");
flowChoiceOut.addItem("RTS/CTS Out");
flowChoiceOut.select(parameters.getFlowControlOutString());
flowChoiceOut.addItemListener(this);
add(flowChoiceOut);
databitsLabel = new Label("Data Bits:", Label.LEFT);
add(databitsLabel);
databitsChoice = new Choice();
databitsChoice.addItem("5");
databitsChoice.addItem("6");
databitsChoice.addItem("7");
databitsChoice.addItem("8");
databitsChoice.select(parameters.getDatabitsString());
databitsChoice.addItemListener(this);
add(databitsChoice);
stopbitsLabel = new Label("Stop Bits:", Label.LEFT);
add(stopbitsLabel);
stopbitsChoice = new Choice();
stopbitsChoice.addItem("1");
stopbitsChoice.addItem("1.5");
stopbitsChoice.addItem("2");
stopbitsChoice.select(parameters.getStopbitsString());
stopbitsChoice.addItemListener(this);
```

```
        add(stopbitsChoice);
        parityLabel = new Label("Parity:", Label.LEFT);
        add(parityLabel);
        parityChoice = new Choice();
        parityChoice.addItem("None");
        parityChoice.addItem("Even");
        parityChoice.addItem("Odd");
        parityChoice.select("None");
        parityChoice.select(parameters.getParityString());
        parityChoice.addItemListener(this);
        add(parityChoice);
    }
    public void setConfigurationPanel() {
        portChoice.select(parameters.getPortName());
        baudChoice.select(parameters.getBaudRateString());
        flowChoiceIn.select(parameters.getFlowControlInString());
        flowChoiceOut.select(parameters.getFlowControlOutString());
        databitsChoice.select(parameters.getDatabitsString());
        stopbitsChoice.select(parameters.getStopbitsString());
        parityChoice.select(parameters.getParityString());
    }
      public void setParameters() {
            parameters.setPortName(portChoice.getSelectedItem());
          parameters.setBaudRate(baudChoice.getSelectedItem());
            parameters.setFlowControlIn(flowChoiceIn.getSelectedItem());
            parameters.setFlowControlOut(flowChoiceOut.getSelectedItem());
            parameters.setDatabits(databitsChoice.getSelectedItem());
            parameters.setStopbits(stopbitsChoice.getSelectedItem());
            parameters.setParity(parityChoice.getSelectedItem());
      }
      void listPortChoices() {
            CommPortIdentifier portId;
            Enumeration en = CommPortIdentifier.getPortIdentifiers();
            // iterate through the ports.
            while (en.hasMoreElements()) {
                  portId = (CommPortIdentifier) en.nextElement();
                  if (portId.getPortType() == CommPortIdentifier.PORT_SERIAL) {
                        portChoice.addItem(portId.getName());
                  }
```

```
            }
            portChoice.select(parameters.getPortName());
        }
    public void itemStateChanged(ItemEvent e) {
        // Check if port is open.
        if (connection.isOpen()) {
            // If port is open do not allow port to change.
            if (e.getItemSelectable() == portChoice) {
                // Alert user.
                AlertDialog ad = new AlertDialog(parent, "Port Open!",
                                                 "Port can not",
                                                 "be changed",
                                                 "while a port is open.");
                // Return configurationPanel to pre-choice settings.
                setConfigurationPanel();
                return;
            }
            // Set the parameters from the choice panel.
            setParameters();
            try {
                // Attempt to change the settings on an open port.
                connection.setConnectionParameters();
            } catch (SerialConnectionException ex) {
                // If setting can not be changed, alert user, return to
                // pre-choice settings.
                AlertDialog ad = new AlertDialog(parent,
                   "Unsupported Configuration!",
                   "Configuration Parameter unsupported,",
                   "select new value.",
                   "Returning to previous configuration.");
                setConfigurationPanel();
            }
        } else {
            // Since port is not open just set the parameter object.
            setParameters();
        }
    }
}
class CloseHandler extends WindowAdapter {
```

```
        SerialDemo sd;
        public CloseHandler(SerialDemo sd) {
            this.sd = sd;
        }
        public void windowClosing(WindowEvent e) {
            sd.shutdown();
        }
    }
}
```

SerialConnection.java:

```
import javax.comm.*;
import java.io.*;
import java.awt.TextArea;
import java.awt.event.*;
import java.util.TooManyListenersException;
public class SerialConnection implements SerialPortEventListener,
                                         CommPortOwnershipListener {
    private SerialDemo parent;
    private TextArea messageAreaOut;
    private TextArea messageAreaIn;
    private SerialParameters parameters;
    private OutputStream os;
    private InputStream is;
    private KeyHandler keyHandler;
    private CommPortIdentifier portId;
    private SerialPort sPort;
    private boolean open;
    JDBCQuery database = null;
    public SerialConnection(SerialDemo parent,
                            SerialParameters parameters,
                            TextArea messageAreaOut,
                            TextArea messageAreaIn) {
        this.parent = parent;
        this.parameters = parameters;
        this.messageAreaOut = messageAreaOut;
        this.messageAreaIn = messageAreaIn;
        open = false;
    }
    public SerialConnection(SerialDemo parent,
```

```
                        SerialParameters parameters,
                        //TextArea messageAreaOut,
                        TextArea messageAreaIn) {
    this.parent = parent;
    this.parameters = parameters;
    //this.messageAreaOut = messageAreaOut;
    this.messageAreaIn = messageAreaIn;
    open = false;
    database = new JDBCQuery();
    if ( database.jisconnect()) {
        System.out.println("Opned JDBC");
    } else {
        System.out.println("Open JDBC error");
    }
}
public void openConnection() throws SerialConnectionException {
    // Obtain a CommPortIdentifier object for the port you want to open.
    try {
        portId =
          CommPortIdentifier.getPortIdentifier(parameters.getPortName());
    } catch (NoSuchPortException e) {
        throw new SerialConnectionException(e.getMessage());
    }
    try {
        sPort = (SerialPort)portId.open("SerialDemo", 30000);
    } catch (PortInUseException e) {
        throw new SerialConnectionException(e.getMessage());
    }
    try {
        setConnectionParameters();
    } catch (SerialConnectionException e) {
        sPort.close();
        throw e;
    }
    try {
        os = sPort.getOutputStream();
        is = sPort.getInputStream();
    } catch (IOException e) {
        sPort.close();
```

```
            throw new SerialConnectionException("Error opening i/o streams");
        }
        try {
            sPort.addEventListener(this);
        } catch (TooManyListenersException e) {
            sPort.close();
            throw new SerialConnectionException("too many listeners added");
        }
        sPort.notifyOnDataAvailable(true);
        sPort.notifyOnBreakInterrupt(true);
        try {
            sPort.enableReceiveTimeout(30);
        } catch (UnsupportedCommOperationException e) {
        }
        portId.addPortOwnershipListener(this);
        open = true;
    }
    public void setConnectionParameters() throws SerialConnectionException {
        int oldBaudRate = sPort.getBaudRate();
        int oldDatabits = sPort.getDataBits();
        int oldStopbits = sPort.getStopBits();
        int oldParity     = sPort.getParity();
        int oldFlowControl = sPort.getFlowControlMode();
        try {
            sPort.setSerialPortParams(parameters.getBaudRate(),
                                      parameters.getDatabits(),
                                      parameters.getStopbits(),
                                      parameters.getParity());
        } catch (UnsupportedCommOperationException e) {
            parameters.setBaudRate(oldBaudRate);
            parameters.setDatabits(oldDatabits);
            parameters.setStopbits(oldStopbits);
            parameters.setParity(oldParity);
            throw new SerialConnectionException("Unsupported parameter");
        }
        try {
            sPort.setFlowControlMode(parameters.getFlowControlIn()
                                     | parameters.getFlowControlOut());
        } catch (UnsupportedCommOperationException e) {
```

```
            throw new SerialConnectionException("Unsupported flow control");
        }
    }
    public void closeConnection() {
        // If port is alread closed just return.
        if (!open) {
            return;
        }
        if ( database.jisconnect()) {
                database.jclose();
                System.out.println("Closed JDBC");
        } else {
                System.out.println("Error close JDBC");
        }
        if (sPort != null) {
            try {
                // close the i/o streams.
                os.close();
                is.close();
            } catch (IOException e) {
                System.err.println(e);
            }
            // Close the port.
            sPort.close();
            // Remove the ownership listener.
            portId.removePortOwnershipListener(this);
        }
        open = false;
    }
    public void sendBreak() {
        sPort.sendBreak(1000);
    }
    public boolean isOpen() {
        return open;
    }
    public void serialEvent(SerialPortEvent e) {
        // Create a StringBuffer and int to receive input data.
                StringBuffer inputBuffer = new StringBuffer();
                int newData = 0;
```

```
                // Determine type of event.
                switch (e.getEventType()) {
                // Read data until -1 is returned. If \r is received substitute
                // \n for correct newline handling.
                case SerialPortEvent.DATA_AVAILABLE:
                        while (newData != -1) {
                                try {
                                newData = is.read();
                                        if (newData == -1) {
                                                break;
                                        }
                                        if ('\r' == (char)newData) {
                                                inputBuffer.append('\n');
                                        } else {
                                                inputBuffer.append((char)newData);
                                        }
                                } catch (IOException ex) {
                                System.err.println(ex);
                                        return;
                                        }
                        }
                                messageAreaIn.append("===>Login: ");
                                String num = new String(inputBuffer);
                                num = num.substring(1, 11);
                                messageAreaIn.append(num);
                                messageAreaIn.append("\n");
                                if ( database.jisconnect()) {
                                        database.jinsert(num);
                                } else {
                                        System.out.println("JDBC: no operate");
                                }
                                break
                // If break event append BREAK RECEIVED message.
                case SerialPortEvent.BI:
                   messageAreaIn.append("\n--- BREAK RECEIVED ---\n");
           }
        }
        public void ownershipChange(int type) {
           if (type == CommPortOwnershipListener.PORT_OWNERSHIP_REQUESTED) {
```

```
                PortRequestedDialog prd = new PortRequestedDialog(parent);
            }
        }
        class KeyHandler extends KeyAdapter {
            OutputStream os;
            public KeyHandler(OutputStream os) {
                super();
                this.os = os;
            }
              public void keyTyped(KeyEvent evt) {
                  char newCharacter = evt.getKeyChar();
                try {
                  os.write((int)newCharacter);
                } catch (IOException e) {
                  System.err.println("OutputStream write error: " + e);
                }
              }
        }
}
```

JDBCQuery.java:

```
import java.awt.*;
//import java.util.Date;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.sql.Date;
import java.sql.*;
import java.lang.Integer;
/**
        這個類別主要是處理所有關於資料庫的動作，如 SELECT、INSERT...
        首先要用建構出這個物件。
*/
public class JDBCQuery {
        Connection conn = null;
        Statement stmt = null;
        public JDBCQuery() {
                jconnect();
        }
        public void jconnect() {
                try {
```

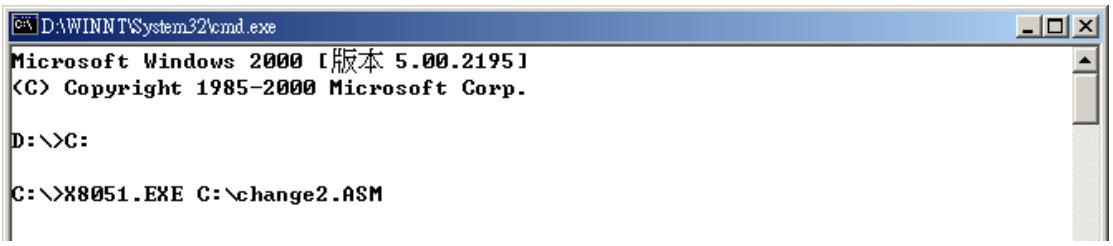
```
                Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
                conn = DriverManager.getConnection (
                        "jdbc:odbc:mydoor",
                        "sa", "123456");        // user, passwd
                stmt = conn.createStatement();
        } catch (ClassNotFoundException e) {
                System.out.println("Can't load driver " + e);
        } catch (SQLException e) {
                System.out.println("Database access failed " + e);
        }
}
public ResultSet jselect() {
        //Statement stmt = conn.createStatement();
        try {
          return stmt.executeQuery("SELECT * FROM shit");
        } catch (SQLException e) {
                System.out.println("Database access failed " + e);
        }
        return null;
}
public boolean jisconnect() {
        if ( conn == null) {
                return false;
        } else {
                if ( stmt == null) {
                        return false;
                } else {
                        return true;
                }
        }
}
public void jinsert(String data) {
Calendar cal = new GregorianCalendar();
// Get the components of the time
int year = cal.get(Calendar.YEAR);
int month = cal.get(Calendar.MONTH);
int day = cal.get(Calendar.DAY_OF_MONTH);
        int hour24 = cal.get(Calendar.HOUR_OF_DAY);         // 0..23
int min = cal.get(Calendar.MINUTE);                     // 0..59
```

```
        int sec = cal.get(Calendar.SECOND);
        String date =
Integer.toString(year)+Integer.toString(month)+Integer.toString(day)+Integer.toString(hour24)+Integer.toString(min)+Integer.toS
tring(sec);
                System.out.println(date);
                try {
                        stmt.executeUpdate("INSERT    INTO shit (d_date, d_num) VALUES ( '"+date+"', '"+data+"')");
                } catch (SQLException e) {
                        System.out.println("Database access failed " +e );
                }
        }
        public void jclose() {
                try {
                        stmt.close();    // All done with that statement
                        conn.close();   // All done with that DB connection
                } catch (SQLException e) {
                        System.out.println("Database access failed " +e );
                }
        }
        public static void main(String[] av) {
//              try {
                        System.out.println("Loading Driver (with Class.forName)");
                        System.out.println("Getting Connection");
                        JDBCQuery a = new JDBCQuery();
                        System.out.println("Creating Statement");
                        System.out.println("Executing Query");
                        ResultSet rs = a.jselect();
                        System.out.println("Retrieving Results");
                        int i = 0;
                        try {
                        while (rs.next()) {
                                System.out.println("Retrieving Company ID");
                                String x = rs.getString("d_date");
                                System.out.println("Retrieving Name");
                                String s = rs.getString("d_num");
                                System.out.println("ROW " + ++i + ": " +
                                        x + "; " + s + "; " + ".");
                        }
                        a.jinsert("A00000045");
```

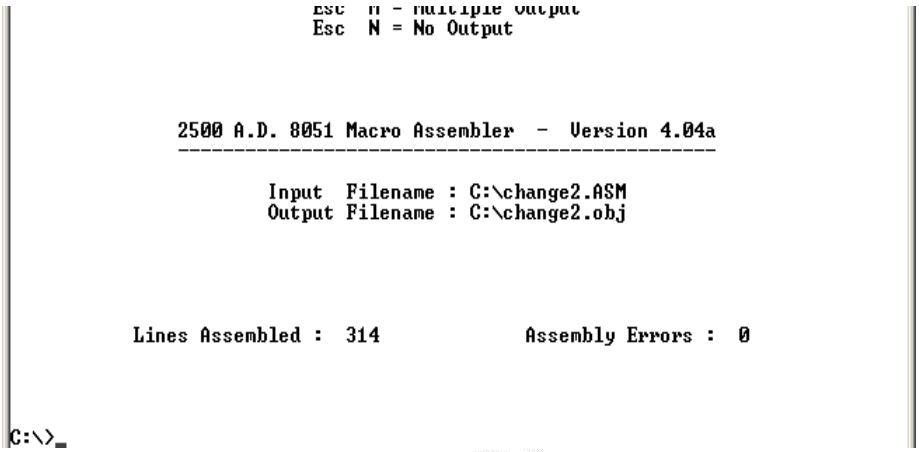
```
            } catch (SQLException e) {
                    System.out.println("Database access failed " + e);
            }
/*
                    rs.close();                 // All done with that resultset
                    stmt.close();   // All done with that statement
                    conn.close();   // All done with that DB connection
*/
              try {
                    rs.close();
              } catch (SQLException e) {
                    System.out.println("Database access failed " +e );
              }
                    a.jclose();
        }
        private static void checkForWarning(SQLWarning warn) throws SQLException    {
            if (warn != null) {
                System.out.println("*** Warning ***\n");
                while (warn != null) {
                    System.out.println("SQLState: " +
                        warn.getSQLState());
                    System.out.println("Message:    " +
                        warn.getMessage());
                    System.out.println("Vendor:     " +
                        warn.getErrorCode());
                    System.out.println("");
                    warn = warn.getNextWarning();
                }
            }
      }
}
```

附錄 B MCS-51 燒錄說明

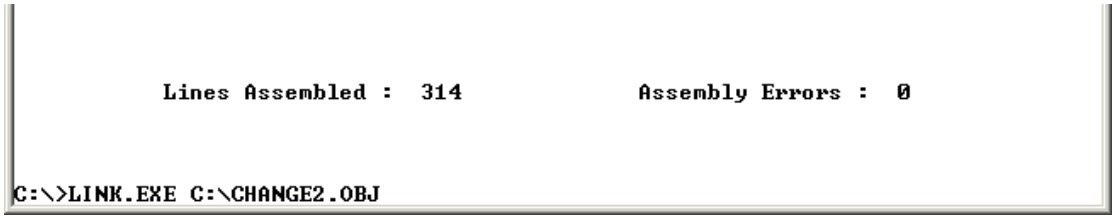
首先要先把程式原始碼組譯:



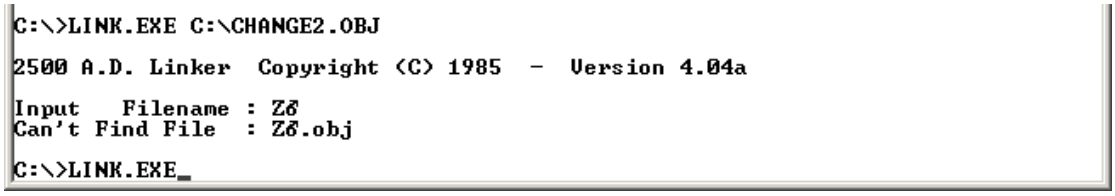
打上 X8051.EXE C: 檔名.ASM 按 ENTER



然後產生一個 檔名.OBJ 的檔案



打上 LINK.EXE C:\CHANGE2.OBJ 按下輸入



輸入 LINK.EXE 按輸入

```
C:\>LINK.EXE C:\CHANGE2.OBJ

2500 A.D. Linker  Copyright (C) 1985  -  Version 4.04a

Input   Filename : Z8
Can't Find File  : Z8.obj

C:\>LINK.EXE

2500 A.D. Linker  Copyright (C) 1985  -  Version 4.04a

Input   Filename : C:\CHANGE2.OBJ_
```

在 Input Filename 輸入 C: 檔名.OBJ

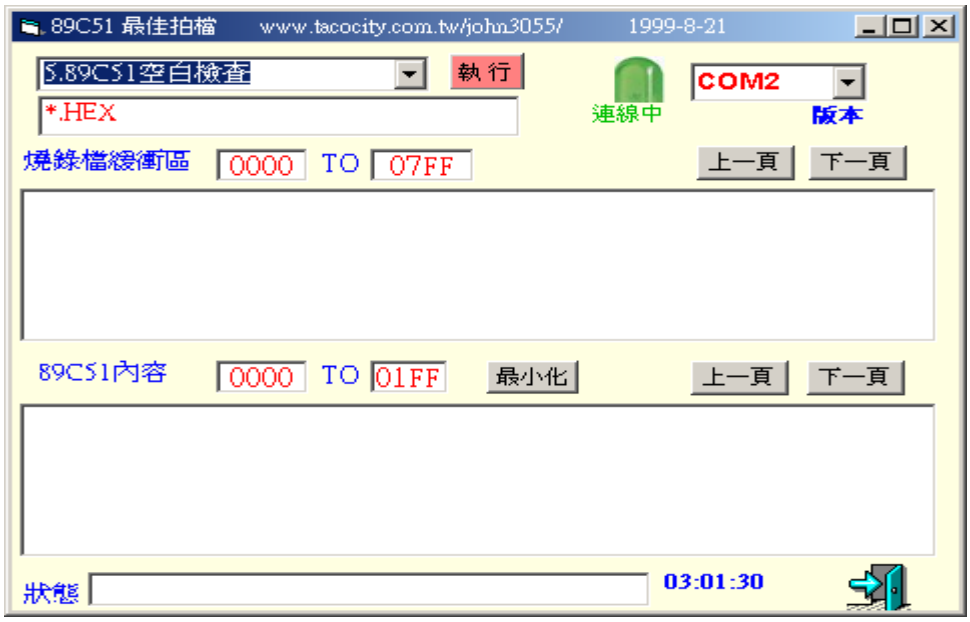
```
2500 A.D. Linker  Copyright (C) 1985  -  Version 4.04a

Input   Filename : C:\CHANGE2.OBJ
        Enter Offset For 'CODE'                    :
Input   Filename :

Output  Filename : CHANGE2.HEX_
```

在 Output Filename 輸入 C:CHANGE2.HEX

產生一個 檔名.HEX 的檔案。
接下來是燒錄工作，執行燒錄應用程式：

確定為連線狀態後先清除 8951 為空白；

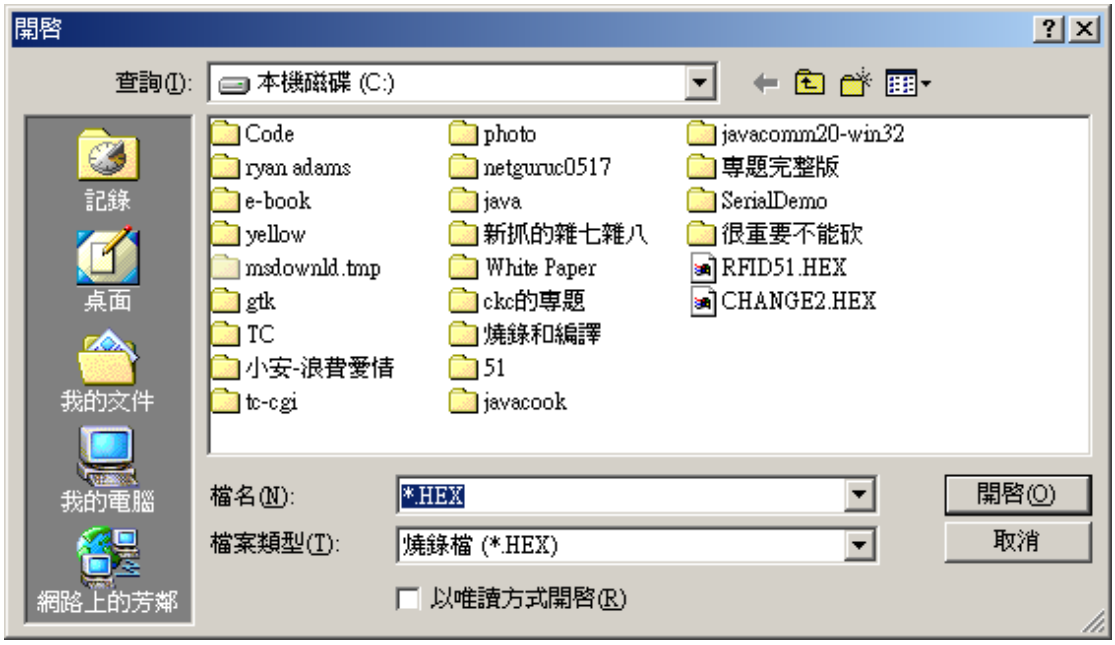
出現成功訊息!



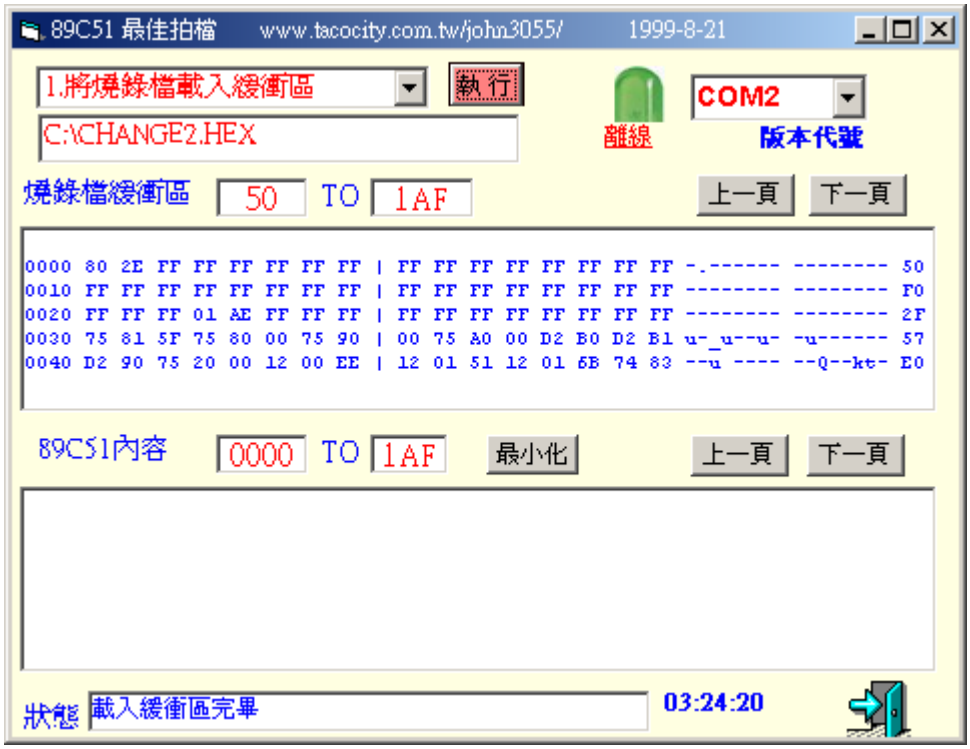
然後確認 8951 是否為空白。

確認為空白

選擇選項至將燒錄檔載入緩衝區；



選擇想要的 檔名.HEX 燒路檔案下開啟



按下執行待成功訊息出現，按下確定，等 8051 燒錄器 LED 停止閃爍便可取下 IC 進行硬體測試。

附錄 C 使用材料表

項目	名稱	規格	功能說明
1	電阻器 R0	0Ω 1/4W	跳線用
2	電阻器 R1	200KΩ 1/4W	RC 充放電電阻
3	電阻器 R2	2KΩ 1/4W	限流電阻
4	電阻器 R3	33KΩ 1/4W	提升電阻
5	電阻器 R4	100Ω 1/4W	限流電阻
6	電容器 C1	1μF/16V 電解電容	RC 充放電電容
7	電容器 C2	22Pf 陶瓷電容	振盪電容
8	電容器 C3	22Pf 陶瓷電容	振盪電容
9	電容器 C4	10μF/50V 電解電容	電壓提升電容
10	電容器 C5	10μF/50V 電解電容	電壓提升電容
11	電容器 C6	10μF/50V 電解電容	電壓提升電容
12	電容器 C7	10μF/50V 電解電容	電壓提升電容
13	電容器 C8	104pF 麥拉電容	高頻傍路電容
14	電容器 C9	470μF/16V 電解電容	穩壓濾波電容
15	電容器 C10	332pF/400V 麥拉電容	LC 諧振電容
16	電感器 L1	0.5Mh 空心線圈	LC 諧振電感 ,當天線使用
17	二極體 D1	5ØLED1N4148 二極體	快速放電
18	發光二極體 D2	5ØLED	8951 工作指示燈
19	二極體 D3	1N4001 整流二極體	防止逆向電壓
20	石英晶體 Y1	11.0592MHZ	振盪晶體
21	計時 IC U1	NE555	RESET IC
22	單晶片 IC U2	8951	單晶片 IC
23	電位轉換 IC U3	MAX232	TTL 與 RS232 電位轉換
24	穩壓 IC U4	7805	5V 穩壓 IC
25	液晶顯示模組 M1		顯示卡片資料
26	感應模組 M2	RD1085	讀取卡片資料
27	電晶體 Q1	C1815	蜂鳴器開關
28	蜂鳴器 B1	9V 蜂鳴器	聲響指示
29	按鈕開關 SW1	2P 按鈕開關	RESET 按鈕開關
30	連接座 CN1	10 PIN 連接座	未指定功能
31	連接座 CN2	2.1Ø DC_PLUG	電源輸入
32	連接座 CN3	9 針 D 型公座	RS232 輸出
33	連接線	9 針 D 型雙母頭	RS232 連接線 ,不跳線
34	電路板		
35	散熱片及螺絲		7805 散熱用
36	塑膠柱及塑膠螺帽	1mm 高度 4 組	LCD 模組固定用
37	排針	14PIN	LCD 模組連接電路板用
38	束線帶	CV075 兩條	天線固定用