高雄縣高英高級工商職業學校
Kao Ying Industrial Commercial Vocational High School

專題製作報告

紅綠燈模擬控制器

學生姓名：蔡　耀　緯
胡　睿　辰
鄭　智　文
指導老師：葉　忠　賢　老師

中　華　民　國　102　年 05 月

誌　謝

進入高中職業學校開始學習技藝課程，面對不同的技能專長領域，除了艱辛，更覺漫長。如今，我們小組幸運地已來到了三年級，回頭俯看這一路上的學習歷程，細數點滴。我想，是我們該將每個階段的感動，留下紀錄的時候了。

感謝導師三年來的教導，再我們專題遇到困難時，總是在旁協助我們，讓我們的專題能更好一點，在老師的陪伴下，我們的專題順利完成。

感謝同學幫我們小組買材料，以及解決一些問題，還有分享他們遇到的問題及經驗分享給我們，讓我們的困難解決。

蔡耀緯、鄭智文、胡睿辰　謹上　2013/05

摘要

本篇研究報告旨在透過單晶片 89C51 的學習，了解單晶片的功能及使用方法，且經由實際製作 8*8 矩陣燈的過程中去對單晶片運作有更深入的了解。會想製作紅綠燈模擬控制器，是為了讓幼小兒童更了解紅綠燈，不用外出，就能了解紅綠燈的進行方式；故想要藉由設計一單晶片電路，配合組合語言程式去達到將小綠人行走及七段顯示器倒數，且也因為透過設計及製作模擬控制器的過程中，可以知道及了解，如何透過程式組合語言去設計、控制紅燈跳至黃燈的功能。小組成員同學預期此設計能先達成做到小綠人能行走的功能及目的，若此階段沒問題了，再會進階地針對小綠人走路速度進行改進，因為一般的紅綠燈倒數十秒時，小綠人行走會變快，可是若能充分利用 89C51 程式碼，就能使小綠人行走速度變快，故現行之小組專題製作的目標即是想透過單晶片的學習。

關鍵詞：單晶片 、組合語言、8*8 矩陣燈

目　錄

表目錄

圖目錄

壹、前言

在人們生活中，無論是大馬路小、巷子都跟紅綠燈有關，所以紅綠燈自然是生活中不可缺乏的需求項，且隨著科技產品的更新，有著更進步、更科技、更便利及更人性的紅綠燈系統應用在平常生活中。

經過這學期的微處理機課程後，我們學習了如何使用微處理晶片來控制和數學運算的機制，在這次的計畫裡，我們準備運用 89S51 這個功能強大的晶片來執行紅綠燈的流程規劃功能。整個計畫中用到顯示原理和微處理器原理，讓我們獲得一次將課堂理論用來實做的機會。

一、製作動機

在科技人性化的資訊潮流當中，「科技化」的自控平台常常因科技的創新結合依據人們生活需求有著更新穎的技術整合，以達到更人性化的科技生活。

而這些監控系統即正是所謂的「人機介面多元整合系統」因這些成熟的設計是要有一定程度的專業領域能力及科技設備為背景，但若在高職基礎專業能力上要去實作是較困難的。

但因資訊科技的各種紅綠燈早已在人們生活中，所以為滿足學生對紅綠燈好奇及問題需求，也為了要讓學生對紅綠燈有更進一步的了解，進而產生想製作此專題的想法念頭，並著手去執行。

二、製作目的

以高職三年專業背景能力，其中利用所學軟體設計能力及硬體設計能力來設計一套既簡單又實用的紅綠燈模擬控制器，而此套簡單的控制系統實作對高三生而言不難，並也可實質應用在生活上達到「做中學、學中做」的道理。

並讓學生了解一套系統並非是只要有一門專業技術能力就夠了，而是要多種專業能力的融合才能設計出成果。

同時也讓學生了解此專題紅綠燈的運轉方式。專題成品完成後又可作為讓學生控制系統學習軟體設計教材

三、製作架構

(一) 專題製作流程

我們的作品架構，先尋找專題相關資料，再擬定專題題目，小組分配任務後開始購買材料，將程式燒入 89C51 控制其 IC:UN2803.7447*2.74HC578 再由 IC 去控制矩陣燈及七段顯示器，再由程式碼去做調整。

(二)專題流程圖

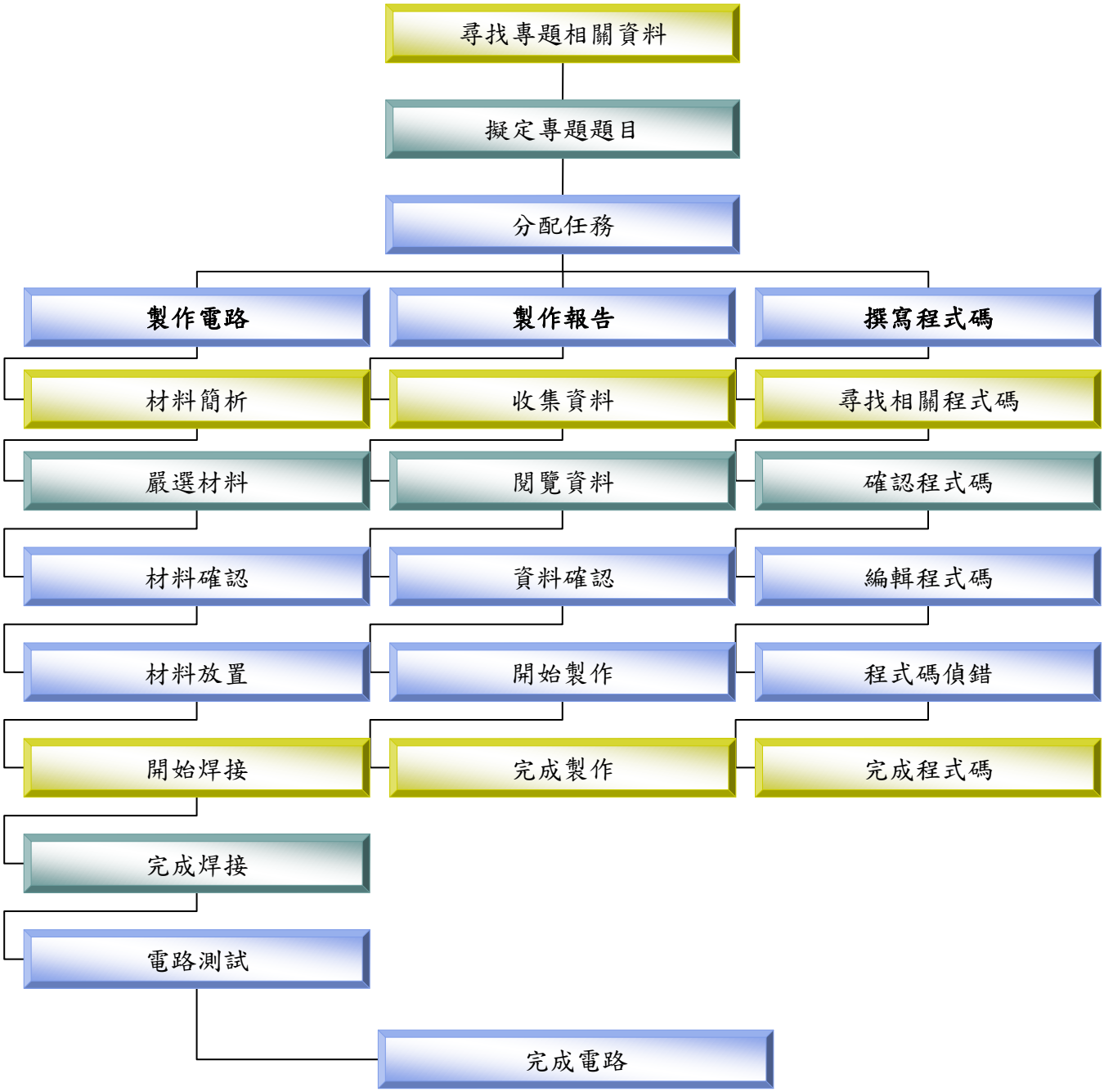


圖 1-3-1 專題流程圖

四、製作預期成效

本專題預計實做完成軟硬體整合的「單工監控系統」。此專題配合專業課程的理論內容及實習技能，同時利用此專題培養學生的科技研究精神。且本紅綠燈這套系統之主要以社會交通太煩雜。本系統製作將達到以下之目標：

(一）使 8051 程式碼可以使小紅人行走 。

(二）矩陣燈具有小紅人將可告知行人。

(三）七段顯示器倒數九秒跑到黃燈時會閃爍。

(四）九秒時小紅人會走的更加快速。

(五）藉由控製時間的設定以達到準確的安全需求。

貳、理論探討

一、電子相關零組件

(一) LED 點矩陣

點矩陣 LED 顯示器是把一些 LED 組合在同一個包裝中，常見的規格有 5*7 及 8*8 兩種可供選購，通常要顯示阿拉伯數字、英文字母、日文字母、特殊符號等均採用 5*7 的點矩陣顯示器即夠用，若要顯示中文字，則需用 4 片 8*8 的點矩陣顯示器組合成 16*16，的點矩陣顯示器才能夠顯示一個中文字，而我們在此專題將使用 8X8 點矩陣來顯示簡單的行人號誌。

8*8 點矩陣顯示方式是採用掃瞄方式點亮，在某一時刻只會有一行 8 顆 LED 被驅動，因為由於人眼有視覺暫留現象所以在該行影像消失前，再將要消失的影像點亮，如此人眼就不會查覺該影像曾經消失，所以人眼看到的是整個 8*8 字形，若是人眼查覺該影像曾經消失就會呈現閃爍現象，所以每行掃瞄頻率必須大於視覺暫留頻率 24Hz。

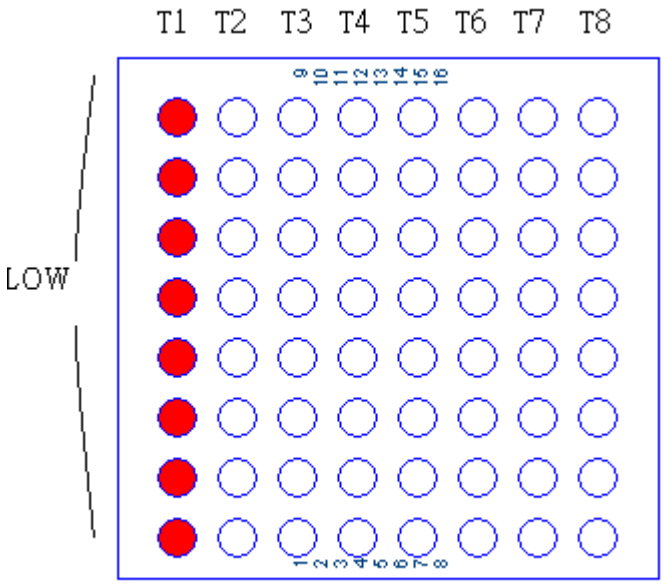


圖 2-1-1 點矩陣設計原理

上圖 T1 到 T8 是表示不同時刻，T1 時會令第一行點亮，而點亮點由 row 方向輸出的資料樣式決定。

人類視覺暫留的最短時間為 1/24 秒，因此掃描完整的畫面的時間不得大於 1/24 秒，才不會有遺失的感覺，而使畫面產生閃爍的現象。當然如果掃描頻率愈高，愈不會有閃爍現象，但因掃描頻率太高時，每行所分配到的顯示時間變短，如此將會造成 LED 亮度不夠的問題。一般而言，選擇 1/48 秒，即可有不錯的顯示結果。因此如果要掃描 n 行，則每行所分得的掃描時

間為掃描頻率的選擇與工作週期有關，所謂工作週期是指每行掃描時間佔整個掃描時間的百分比。

點矩陣電路實作如下圖：我們使用 74LS573 直接提供點矩陣 LED 的點亮圖型，再由 ULN2803 進行掃描點亮每行，ULN2803 為一顆 NPN 的達靈頓電晶體 ULN2803，將 LED 之負極分別接到 ULN2803 的 1C~8C 的腳位上，假設當 1B 為 HIHG 電位時，LED 就經由 1C 增加其 SINK 電流，使 LED 發亮，因此我們可以利用 8051 的 P1 埠，掃描分別讓 1B~8B 輪流為 HIGH，例如：

10000000->01000000->00100000->00010000->00000100->00000010->00000001，則可以讓 LED 看起來全亮。

(二) 七段顯示器

BCD 至七段解碼器 74LS47

a.專題中要顯示秒數為兩位數字，若要直接以 8051 來驅動的話，會導致接腳不足，所以需要解碼 IC 來節省 8051 的接腳。BCD 碼為 4 位元(例如 1001)，可直接對應到七段顯示器並點亮，這種 IC 稱為 BCD 至七段解碼器。在此專題中我們用 74LS47。

b.74LS47 的用法：由於 74LS47 的七隻輸出腳都是開集極(OPEN COLLECTOR)，因此要配合共陽極的七段顯示器。

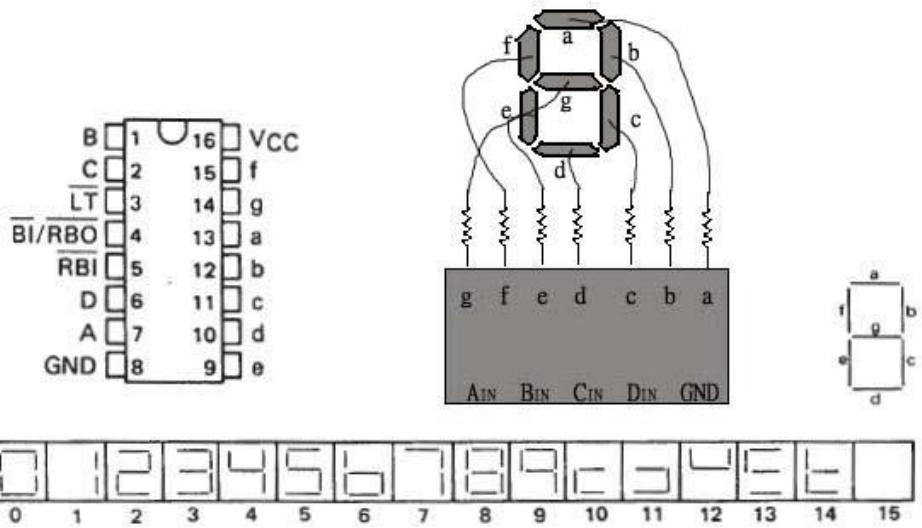


圖 2-1-2 74LS47 接腳及接線

表 2-1-1 74LS47 對應真值表

功能	輸入						$\overline{BI}/\overline{RBO}$†	各段之熄亮						
	$\overline{LT}$	$\overline{RBI}$	D	C	B	A		a	b	c	d	e	f	g
0	H	H	L	L	L	L	H	ON	ON	ON	ON	ON	ON	OFF
1	H	X	L	L	L	H	H	OFF	ON	ON	OFF	OFF	OFF	OFF
2	H	X	L	L	H	L	H	ON	ON	OFF	ON	ON	OFF	ON
3	H	X	L	L	H	H	H	ON	ON	ON	ON	OFF	OFF	ON
4	H	X	L	H	L	L	H	OFF	ON	ON	OFF	OFF	ON	ON
5	H	X	L	H	L	H	H	ON	OFF	ON	ON	OFF	ON	ON
6	H	X	L	H	H	L	H	OFF	OFF	ON	ON	ON	ON	ON
7	H	X	L	H	H	H	H	ON	ON	ON	OFF	OFF	OFF	OFF
8	H	X	H	L	L	L	H	ON	ON	ON	ON	ON	ON	ON
9	H	X	H	L	L	H	H	ON	ON	ON	OFF	OFF	ON	ON
10	H	X	H	L	H	L	H	OFF	OFF	OFF	ON	ON	OFF	ON
11	H	X	H	L	H	H	H	OFF	OFF	ON	ON	OFF	OFF	ON
12	H	X	H	H	L	L	H	OFF	ON	OFF	OFF	OFF	ON	ON
13	H	X	H	H	L	H	H	ON	OFF	OFF	ON	OFF	ON	ON
14	H	X	H	H	H	L	H	OFF	OFF	OFF	ON	ON	ON	ON
15	H	X	H	H	H	H	H	OFF	OFF	OFF	OFF	OFF	OFF	OFF
BI	X	X	X	X	X	X	L	OFF	OFF	OFF	OFF	OFF	OFF	OFF
RBI	H	L	L	L	L	L	L	OFF	OFF	OFF	OFF	OFF	OFF	OFF
LT	L	X	X	X	X	X	H	ON	ON	ON	ON	ON	ON	ON

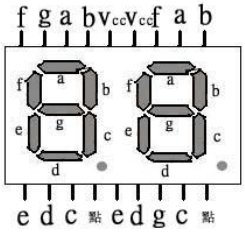


圖 2-1-4 七段顯示器腳位

二、微電腦單晶片

(一) MCS-51 簡介

MCS-51是Intel公司所設計的8051系列單晶片的總稱，較具知名度的編號有8051、8751和8031，這些不同的單晶片都使用相同的核心CPU與指令集，只是能在靠製造IC時給予不同的周邊設計，分別賦予這些IC一個特別編號。

表2-1-2 MCS-51的成員

名稱	ROMLESS	EPROM	ROM(位元組)	RAM(位元組)	16位元計時器	電路型式
8051	8031	(8751)	2K	128	2	HMOS
8051AH	8031AH	8751H	2K	128	2	HMOS
8052AH	8032AH	8752H	2K	256	3	HMOS
80C51BH	80C31BH	87C51	2K	128	2	CHMOS

以下將MCS-51系列單晶片的主要功能列舉如下：

1. 專為控制應用所設計的8位元CPU
2. 有完整的單位元邏輯運算指令
3. 有32條(4個Port)雙向且每條都可以被單獨定址的I/O
4. 內部有128byte可供讀/寫的RAM
5. 內部有兩個16位元Timer/Counter
6. 有一個通信用的全雙工UART(串列I/O)
7. 可接受5個中斷源，且有2層優先權的中斷結構
8. 內部有時脈振盪器(最高頻率可到12MHz)
9. 內部有4K的程式記憶體
10. 可在外部擴充到64K程式記憶體(EPROM)
11. 可在外部擴充64K資料記憶體(RAM)

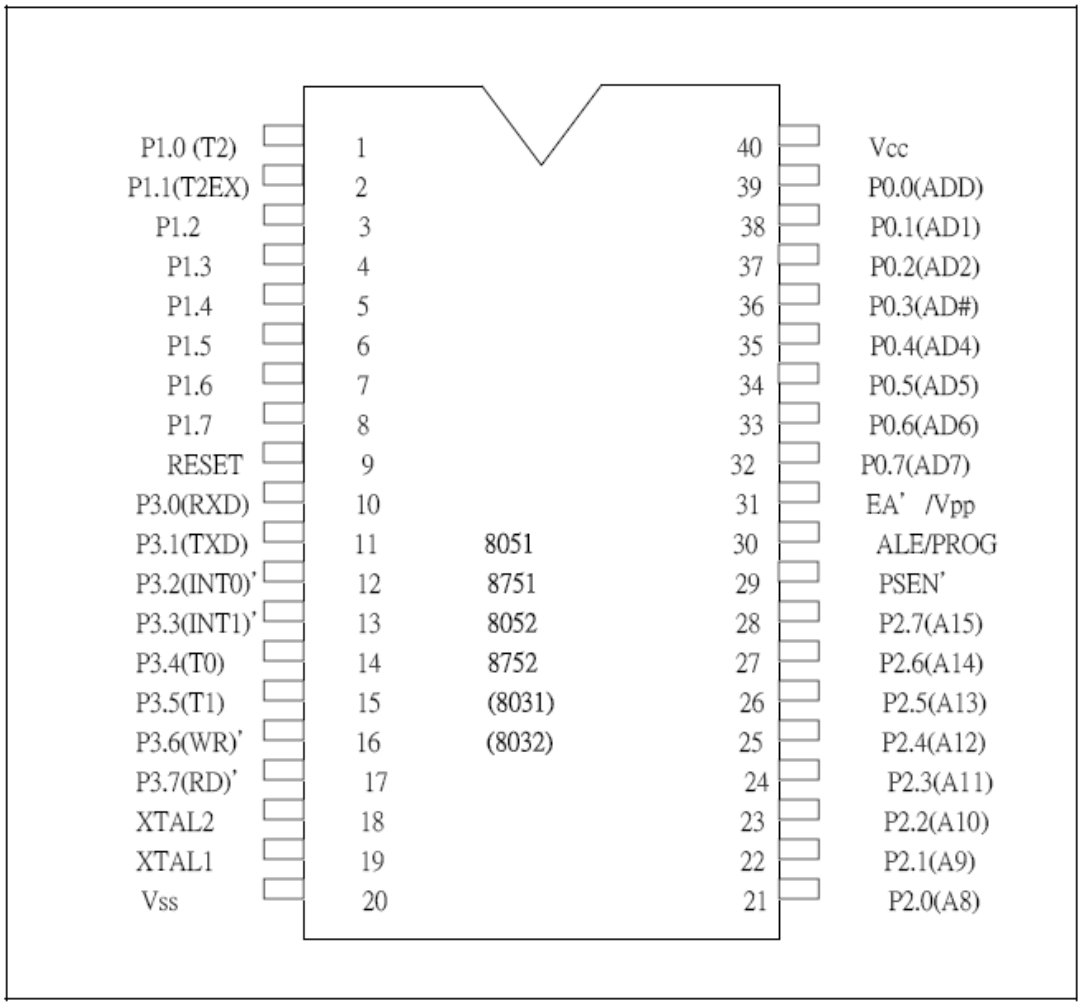


圖2-2-1 MCS-51的40Pin DIP包裝的接腳圖

1. 1~8腳 (P1.0~P1.7)：

這8支角是8051的I/O port，稱為PI。第一腳(P1.0)是LSB，第8支腳(P1.7)是MSB。如果是8052(8032或8752)時，P1.0又可當作Timer2的外部脈波輸入腳，P1.1又當作T2EX，可當作另外一個外部中斷觸發輸入腳。P1上的每支腳都可推動4個LS TTL。

2. 9腳(RESET)：

8051的重置(RESET)輸入腳，當這支腳由外部輸入High(+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。且特殊功能暫存器(SFR)裡的所有暫存器都會被設成已知狀態。

3. 10~17腳 (P3.0~P3.7)：

這8支腳是8051的I/O port，稱為P3。第10支腳(P3.0)為LSB，第17支腳(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個I/O腳。例如P3.0和P3.1接腳的另外一個名稱為RxD和TxD，當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。T0是Timer0的外部脈波輸入腳。T1是Timer1的外部脈波輸入腳。WR，RD，當您再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上的每一隻I/O腳都可以做兩種用途。那8051怎麼知道P3上的某支腳是當I/O或當另一種用途，例如您要使用UART時您將第10腳看成RxD，第11腳看成TxD加以使用就可以了。但是有一點必須特別注意，那就是當作其他功能(不當I/O使用)使用的那支腳的內部栓鎖器的內容必須設為1，其他的功能(如TxD，RxD，RD，ER，…等)才會有作用。P3上的每支I/O腳都可推動4個LS TTL。

4. 18~19腳(XTAL2，XTAL1)：

這兩支腳是8051內部時脈振盪器的輸入端，您可以在這兩支腳上跨街一個12MHz的工作頻率，供內部使用。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。如果線路板上已有振盪器，那這個振盪器所產生的脈波(Clock)也可以直接輸入給8051使用。這個外部送給8051使用的脈波是從第18腳(XTAL2)輸入，而19腳(XTAL1)必須接地，以上的接法是CMOS的8051(如8051AH)。

如果您是使用CMOS的8051(80C51，80C31等)，外部的脈波必須從19腳(XTAL1)輸入而18腳空接，這個差別必須特別注意。

5. 40，20腳(Vcc，Vss)：

這是8051的電源輸入端，40腳接電源的正端的20腳接地。

電源規格是5V +/- 10% 。

6. 21~28腳(P2.0~P2.7)：

這8支腳是8052的I/O port，稱為P2，P2.0為LSB，P2.7為MSB。P2除了當作I/O使用之外。如果您在8051的外面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)，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。

7. 39~32腳(P0.0~P0.7)：

這8支腳也是8051的I/O port，稱為P0其中P0.0為LSB，P0.7為MSB。如果將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 port(P1，P2，P3)內部有pull high電路。P0除了當作I/O使用外，如果您在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。您必須再外部加一個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。

8. 29腳(PSEN)：

這支腳是8051用來讀取放在外部程式記憶體的指令時所用的讀去信號，通常這支腳是接到EPROM的OE腳。8051分別致能放在外部的EPROM(程式記憶體)與RAM資料記憶體是兩塊獨立的記憶體，且這兩塊記憶體都可以接到64K，因此我們說8051的定址能力可達128K。

9. 30腳(ALE)：

這支腳的名稱為 ”位址拴住致能” (Address Latch Enable，簡稱ALE)，8051可以使用這支腳觸發外部的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。

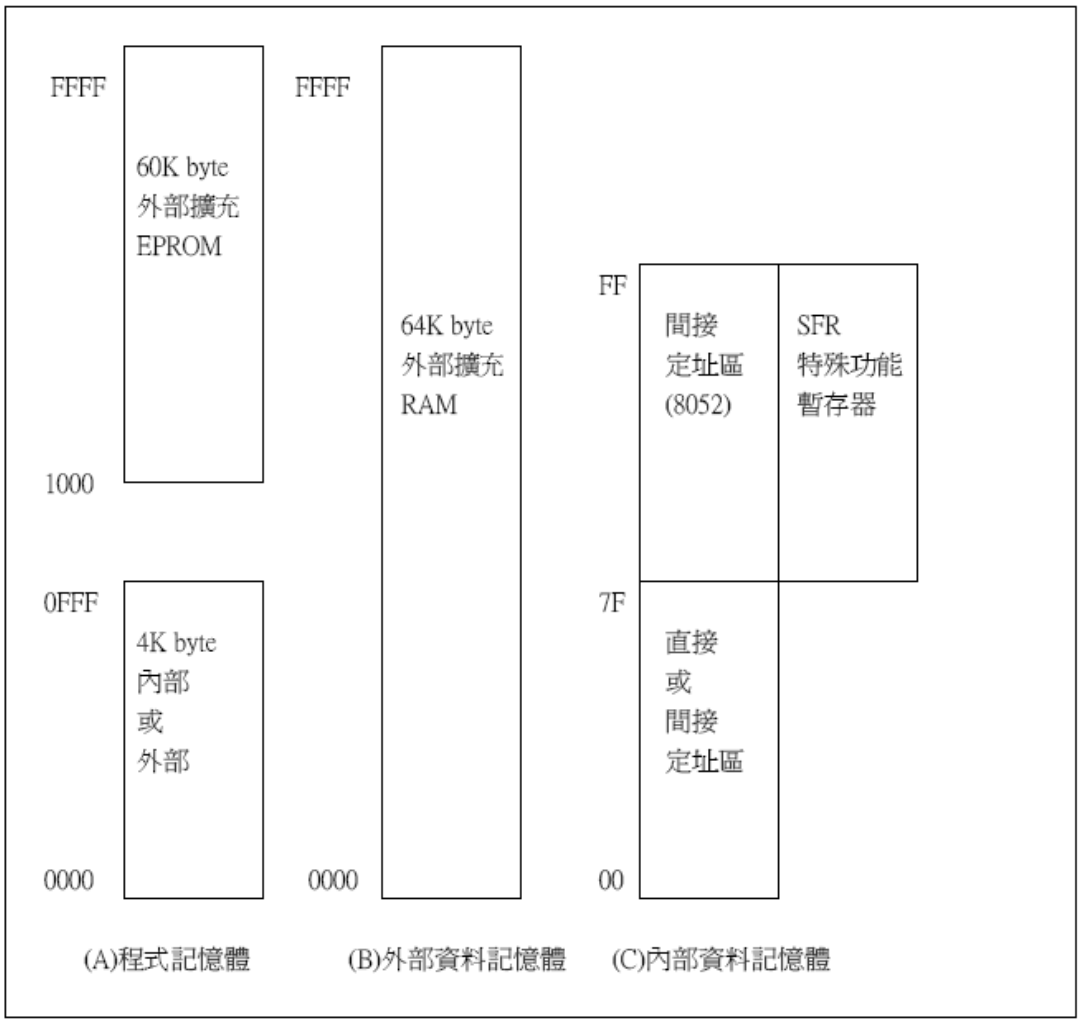
10. 31腳(EA)：

這是一支輸入腳，當EA=0時，8051一律執行外部程式記憶體裡的程式，因此8051內部的4K程式記憶體就沒有用了。因此如果您要使用內部的程式記憶體時，一定要將EA接+5V。因為8031(或8032)內部沒有程式記憶體，它的EA必須接地。

2-2 8051的記憶體結構

8051的記憶體可以分成3塊獨立的記憶體，如圖所示：

1. 內部加上外部的程式記憶體(ROM)共64K byte
2. 可在外部擴充64K byte資料記憶體(RAM)
3. 內部資料記憶體空間256 byte



2-2-2 8051記憶體映像圖

2-3-1 程式記憶體

程式記憶體是存放8051所執行的程式碼的地方，CPU會主動到這塊記憶體讀取要執行的指令碼，因此這塊記憶體的資料只能被CPU讀取，而無法寫入資料。程式記憶體的空間最多可達64K byte，在8051，8751裡已有最低的4K byte(0000H ~ 0FFFH)，因此在外部可再擴充60K byte EPROM；而8031H，8032H內部沒有ROM，因此外部可擴充64K byte EPROM；8052AH，8752AH內部已有8K byte的程式記憶體，因此可以在外部擴充56K byte EPROM。8051讀取程式記憶體的激發信號是PSEN。8051是如何決定程式記憶體的前面4K byte(8052是8K)要到內部或到外接程式記憶體去讀取指令呢？這就是8051的EA腳(第31腳)的功能，如果我們將EA腳接地(邏輯0)，則8051會將前面4K移到外部，也就是說原來在8051內部的4K byte的程式記憶體無效，就算將程式燒到內部的4K byte程式記憶體裡，8051也看不到。如果將EA接到+5V(邏輯1)，則8051就會到內部去讀去前面4K的程式記憶體，超過4K的部分(1000H~FFFFH)，8051會自動切換到外部來讀取。因此EA接腳是決定內部程式記憶體是否有效的控制腳，當EA=0，內部程式記憶體無效；當EA=1內部程式記憶體有效。例如8031AH，8032AH內部沒有ROM，因此使用8031AH或8032AH時，必須將它的EA腳接地。在寫8051的程式時，必須知道幾個程式記憶體的特殊位址，這些位址是各種中斷服務程式的進入點，表2列出了各種中斷的進入點位址，其中位址0000H是重置(RESET)的進入點，這意思是說，8051被重置時，從位址0000H開始執行程式。

表2-2-1 中斷服務程式的進入點

中　斷　源	向 量 位 址
RESET	0000h
TNT0	0003h
Timer0	000Bh
INT1	000Bh
Timer1	001Bh
UART	0023h
Timer2	002Bh

2-3-2 外部資料記憶體

8051允許您在外部擴充64K byte資料記憶體(RAM)。這64K位址空間裡，除了可以放RAM以外，也可以採用Memory Map I/O的方式將一些標準I/O(例如8255，8253等)的位址解在這一塊記憶體裡。定址64K資料記憶體空間需要16條位址線和8條資料線，這16條位址匯流排和8條資料匯流排與程式記憶體使用相同的匯流排，然後8051是以控制匯流排來區分這兩塊不同的記憶體。8051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。如此一來程式記憶體和資料記憶體就是兩個完全獨立的64K空間。8051是執行到MOVX A,@DPTR和MOVX A,@Ri 指令時，就會到外部資料記憶體讀入一個byte資料，當執行MOVX @DPTR,A 或 MOVX @Ri,A 就會將資料寫到外部資料記憶體。有時候在外部擴充程式記憶體和資料記憶體，其總和不超過64K時，我們可以採用兩塊記憶體合併成一個64K的設計方式，合併的好處是可以讓程式設計更具彈性。合併的方法很容易，因為8051將程式記憶體和資料記憶體分開的方法是將這兩塊記憶體的讀取激發信號分別使用不同的信號，即PSEN讀取程式記憶體，RD讀取資料記憶體，因此要將這兩塊記憶體合併，只要將PSEN及RD信號合併成一個信號即可，方法是將PSEN與RD使用AND閘做邏輯AND即可，如圖所示，可將AND閘的輸出看成一個記憶體讀取激發信號(MRD)激發(MRD=0)，然後我們就將MRD接到程式記憶體(EPROM)的輸出致能，或資料記憶體(RAM)的輸出致能就可以讀到這兩塊記憶體的內容。

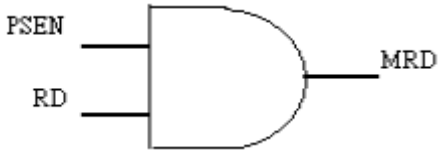


圖2-2-3 將PSEN與RD合併成MRD

程式記憶體與資料記憶體合併之後，8051的整個記憶體空間就縮減成64K，也可以使用合併外接32K EPROM (27256)和32K SRAM (62256)的方法。在這種合併的記憶體結構裡就沒有所謂程式記憶體或資料記憶體之分別，不管是MOVC或MOVX指令都可以定址到這64K的內容。換句話說，也可以將程式放入RAM(62256)裡執行。

2-3-3 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體(RAM)和特殊功能暫存器(SFR)的地方。這塊記憶體空間雖然只有256byte，但是8051將其中位指教高的128byte(80H~FFH)採用不同的定址方式而容納了兩組128byte的記憶空間，因此總共的空間為128+128+128=384 byte。以下三個部分開加以解說：

1. 位址 00H~7FH 的RAM
2. 位址 80H~FFH 的RAM
3. 位址 80H~FFH 的 SFR

不論8051或8052都有這塊記憶體，並且可以使用直接定址或間接定址法讀/寫其內部資料。8051將這塊記憶體分成數種不同的用途。下圖是8051對這128byte定義的用途說明。

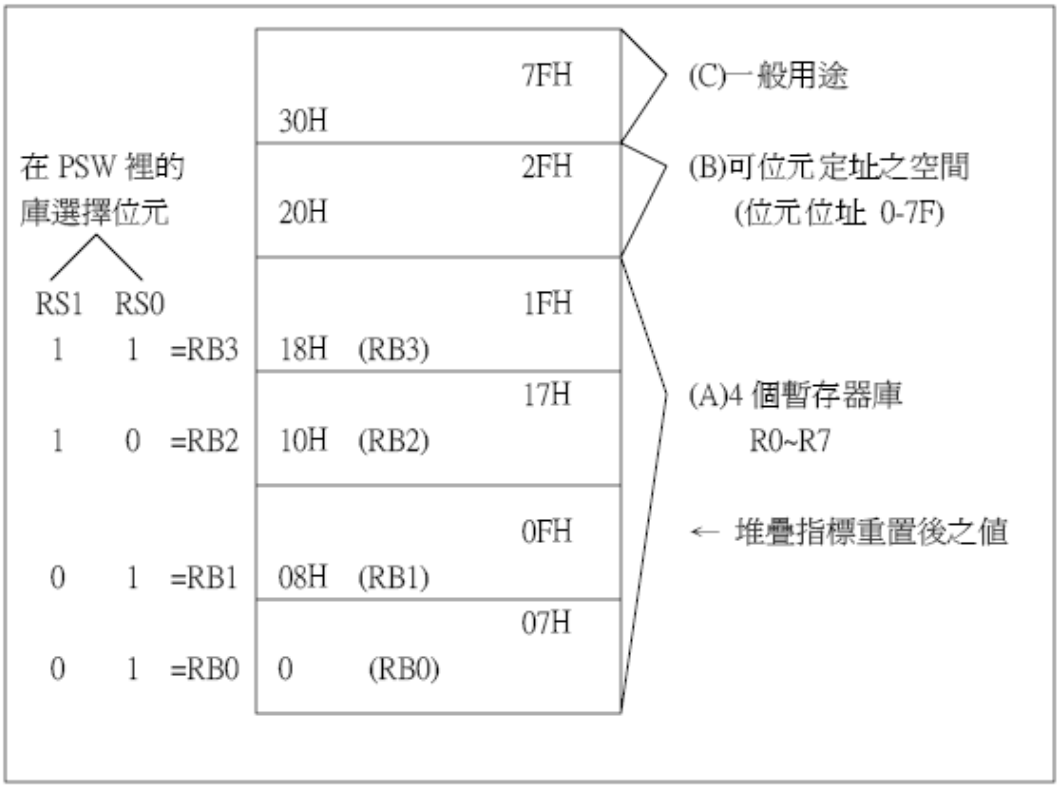


圖2-2-4 內部RAM的較低128位元組

(A) 暫存器庫

由上圖我們看到，位址00~1FH這32個byte被分成4組工作暫存器(Register Bank)，分別稱為RB0，RB1，RB2和RB3，每一組暫存器庫有8個byte。程式指令將每組裡的8個byte稱為R0~R7。但是8051共有四組R0~R7，到底目前所指的R0~R7是屬於哪一組的R0~R7呢？它是由PSW暫存器裡的RS1和RS0 這兩個bit加以選擇，如上圖所示，當RS1=0和RS0=0時，就指到RB0，當RS1=0和RS0=1，就指到RB1。

(B) 可位元定址區

位址20H~2FH這16個byte是8051內部256個位元位址中的128個位元的所在位址。每個位元組佔了8個位元位址，下圖是這16個位元組裡每一個bit的位元位址，例如20H這個byte的第0位元，其位元位址為00H，然後依序編到2FH這個位元組的第7位元為7FH，8051有一組單位元運算指令可以直接對這些位元作運算。

(C) 一般用途

內部RAM的30H~7FH這些位元組，8051並未定義這些位元作任何用途，8051稱這塊區域為使用者的RAM(User RAM)。因此可以規劃這塊區域當作其他用途，例如計時器的緩衝區或印表機資料的緩衝區等。但是有一點必須注意的是，8051的堆疊區也是使用內部RAM，因此必須保留一塊足夠大的RAM給堆疊區使用，堆疊區的大小是依所寫的程式所需而定。

表2-2-2 8051的前面128個位元位址

7FH	一般資料存放區或堆疊區							
2FH	7F	7E	7D	7C	7B	7A	79	78
2EH	77	76	75	74	73	72	71	70
2DH	6F	6E	6D	6C	6B	6A	69	68
2CH	67	66	65	64	63	62	61	60
2BH	5F	5E	5D	5C	5B	5A	59	58
2AH	57	56	55	54	53	52	51	50
29H	4F	4E	4D	4C	4B	4A	49	48
28H	47	46	45	44	43	42	41	40
27H	3F	3E	3D	3C	3B	3A	39	38
26H	37	36	35	34	33	32	31	30
25H	2F	2E	2D	2C	2B	2A	29	28
24H	27	26	25	24	23	22	21	20
23H	1F	1E	1D	1C	1B	1A	19	18
22H	17	16	15	14	13	12	11	10
21H	0F	0E	0D	0C	0B	0A	09	08
20H	07	06	05	04	03	02	01	00
1FH	RB3 (8 bytes)							
	RB2 (8 bytes)							
	RB1 (8 bytes)							
00H	RB0 (8 bytes)							

(a)　　表2-3-2 RAM裡的位元位址

FFH									
F0H	F7	F6	F5	F4	F3	F2	F1	F0	B
E0H	E7	E6	E5	E4	E3	E2	E1	E0	ACC
	CY	AC	F0	RS1	RS0	OV		P	
D0H	D7	D6	D5	D4	D3	D2	D1	D0	PSW
				PS	PT1	PX1	PT0	PX0	
B8H	-	-	-	BC	BB	BA	B9	B8	IP
	P3. 7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	
B0H	B7	B6	B5	B4	B3	B2	B1	B0	P3
	EA		ET2	ES	ET1	EX1	ET0	EX0	
A8H	AF	-	AD	AC	AB	AA	A9	A8	IE
	P2. 7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	
A0H	A7	A6	A5	A4	A3	A2	A1	A0	P2
	SM0	SM1	SM2	REN	TB8	RB8	T1	R1	
98H	9F	9E	9D	9C	9B	9A	99	98	SCON
	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	
90H	97	96	95	94	93	92	91	90	P1
	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	
88H	8F	8E	8D	8C	8B	8A	89	88	TCON
	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	
80H	87	86	85	84	83	82	81	80	P0

(b) SFR RAM裡的位元位址

1.位址80H~FFH的RAM

只有8052，8752和8032的內部RAM有這128byte，8031，8051和8751則沒有。這塊RAM的內容必須使用間接定址法。

2.位址80H~FFH的SFR

特殊功能暫存器是一塊 128byte 的記憶空間，它是存放 8051 內部的週邊所使用的暫存器的地方，例如 I/O port 的輸出栓鎖器(P0，P1，P2，P3)，計時器的 counter，致能中斷系統的 IE 暫存器等。因為 8051 的週邊設備並不多，因此 SFR 裡 128 個位址空間並未用完，這些目前沒有用到的位址，裡面是空的。

SFR 所使用個位址是 80H~FFH，這塊區域與 8051 的較高 128 位元組的 RAM 使用了同一塊記憶空間，8051 採用了不同的指令的定址法來區分這兩塊記憶體，如前面所述，RAM 是使用間接定址法，SFR 是使用直接定址法。

在 SFR 裡的各種位元組都有其個別的名稱，在寫程式時，要用到這些位元組，可直接呼叫其名稱，而不需要使用位址。

在 8051 被重置後(RESET=1)，在 SFR 裡面的各個暫存器都會被設定一個固定值，這些僅在每次 RESET 後都是一樣，表 3，列出了 SFR 重置後的初始值。SFR 裡面的各個暫存器：

1. PSW(程式狀態字語)暫存器
2. SP 暫存器(堆疊指標)
3. DPTR(資料指標)
4. P0，P1，P2，P3 暫存器
5. SBUF(串列阜緩衝區)
6. 計時器暫存器
7. 捕捉式暫存器(Capture Register)
8. 控制暫存器(Control Register)

暫存器名稱	以二進制表示之值

*ACC	00000000
*B	00000000
*PSW	00000000
SP	00000111
DPTR：	
DPH	00000000
DPL	00000000
*P0	11111111
*P1	11111111
*P2	11111111
*P3	11111111
*IP	8051 XXX00000
	8052 XXX00000
*IE	8051 0XX00000
	8052 0X000000
TMOD	00000000
*TCON	00000000
*+T2CON	00000000
TH0	00000000
TL0	00000000
TH1	00000000
TL1	00000000
+TH2	00000000
+TL2	00000000
+RCAP2H	00000000
+RCAP2L	00000000
*SCON	00000000
SBUF	未定
PCON	HMOS 0XXXXXXX
	CHMOS 0XXX0000
X = 未定	
* = 可位元定址	
+ = 只 8052 有	

圖 2-2-5 SFR 各個暫存器重置後的初始值

參、專題製作

此章分為三大重點設備及器材、製作方法與步驟及專題製作等。

一、設備及器材

表 3-1-1 專題製作使用儀器(軟體)設備一覽表

儀器(軟體) 設備名稱	應　用　說　明
個人電腦	專題報告、電路圖製作及進行專題成品電路測試
數位相機	拍攝小組合作過程、專題功能使用及紀錄整個專題製作流程
雷射印表機	列印專題資料、圖片及專題報告成果
三用電錶	測量零件有無損壞及專題電路板各信號之量測
IC 萬用燒錄器	利用燒錄器將程式燒錄至 89C51 單晶片中
電源供應器	提供專題成品所需之電源
Office Word	專題報告、製作過程的撰寫
Power Point	進行口頭報告、製作及專題成品報告呈現
Keil-C	單晶片組合語言程式之編輯、燒錄軟體
Protel 99SE	繪畫專題電路之線路圖

二、製作方法與步驟

本專題研究採用的是行動研究法，主要是由循環的研究歷程所構成，包括準備、實驗教學、電路資料分析及報告撰寫等階段。本研究之製作方法與步驟，如圖 3-2-1 所示。

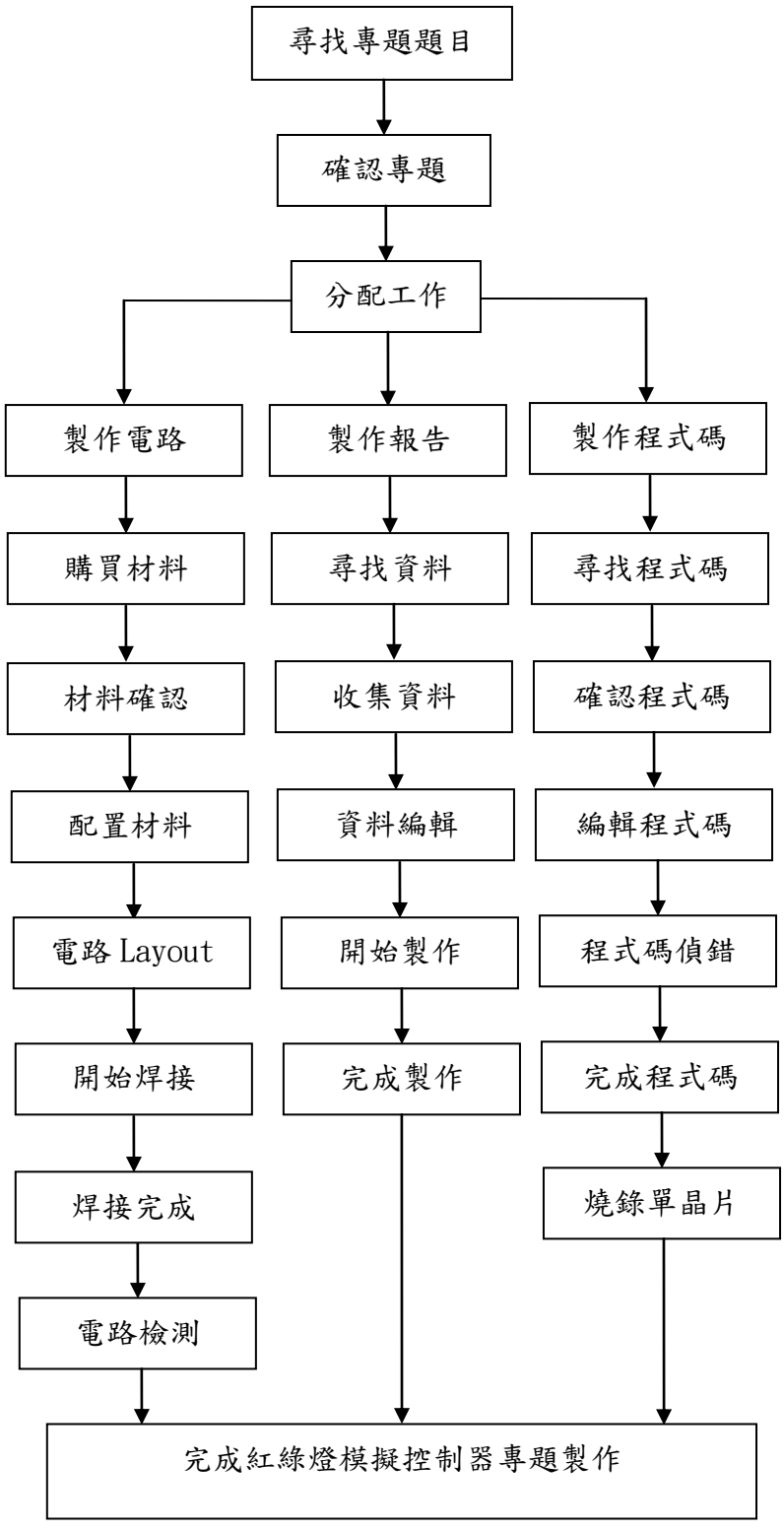
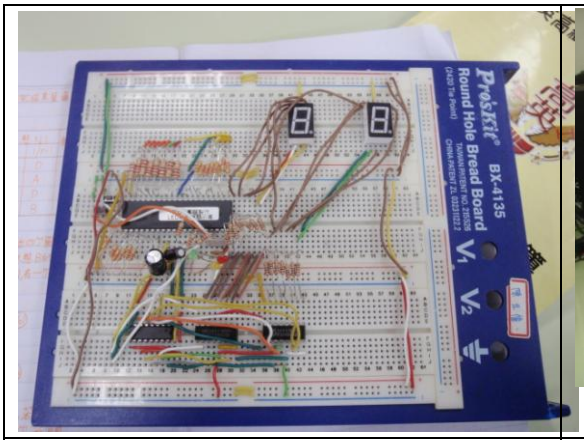
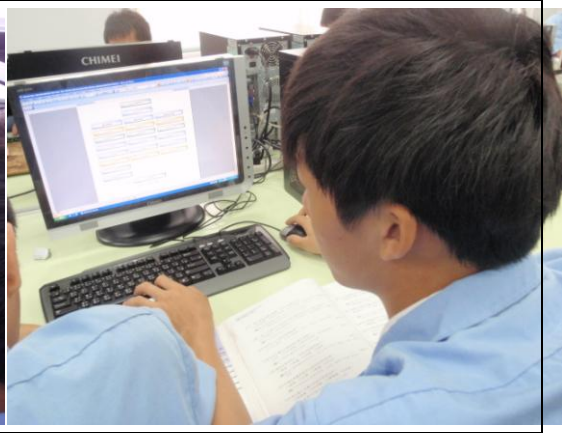
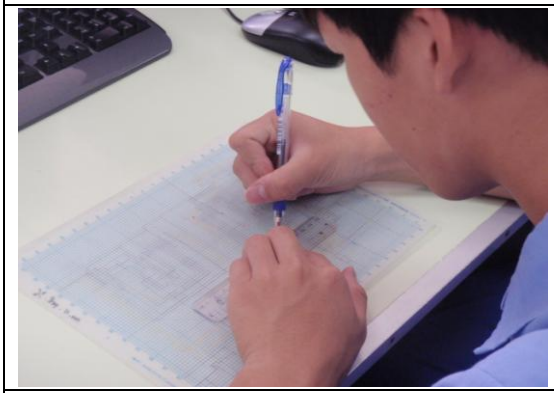


圖 3-2-1 製作方法與步驟

三、專題製作

表 3-3 專題製作計劃書

<table>
<tr><td colspan="2">專題型別</td><td colspan="2">☐ 個人型專題　　☑ 團隊型專題</td></tr>
<tr><td colspan="2">專題性質</td><td colspan="2">單晶片微電腦控制系統</td></tr>
<tr><td colspan="2">科別／年級</td><td colspan="2">資訊科　　三年級</td></tr>
<tr><td rowspan="2">專題名稱</td><td>中文名稱</td><td colspan="2">紅綠燈模擬控制器</td></tr>
<tr><td>英文名稱</td><td colspan="2">Traffic lights analog controller</td></tr>
<tr><td colspan="2" rowspan="5">專題內容簡述</td><td colspan="2">瞭解 8051 單晶片功能，了解 8051 單</td></tr>
<tr><td colspan="2">晶片程式碼使矩陣燈跑出小紅人，在</td></tr>
<tr><td colspan="2">模擬出斑馬線小綠人的行走方式，當</td></tr>
<tr><td colspan="2">七段顯示器 9 秒，小紅人行走方式變</td></tr>
<tr><td colspan="2">快，當 0 秒時，小紅人會停止行走。</td></tr>
<tr><td colspan="2">指導老師姓名</td><td colspan="2">葉忠賢　老師</td></tr>
<tr><td colspan="2" rowspan="2">參與同學姓名</td><td>蔡耀緯</td><td>胡睿辰</td></tr>
<tr><td>鄭智文</td><td></td></tr>
<tr><td colspan="2">專題執行日期</td><td colspan="2">101 年 9 月 1 日至 102 年 5 月 31 日</td></tr>
</table>

<table>
<tr><td> 
</td><td>  </td></tr>
<tr><td>圖 3-3-1 麵包板模擬</td><td>圖 3-3-2 檢測面包板</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>圖 3-3-3 簡報製作</td><td>圖 3-3-4 文書檔製作</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>圖 3-3-5Layout 圖設計</td><td>圖 3-3-6 檢測電路板</td></tr>
</table>

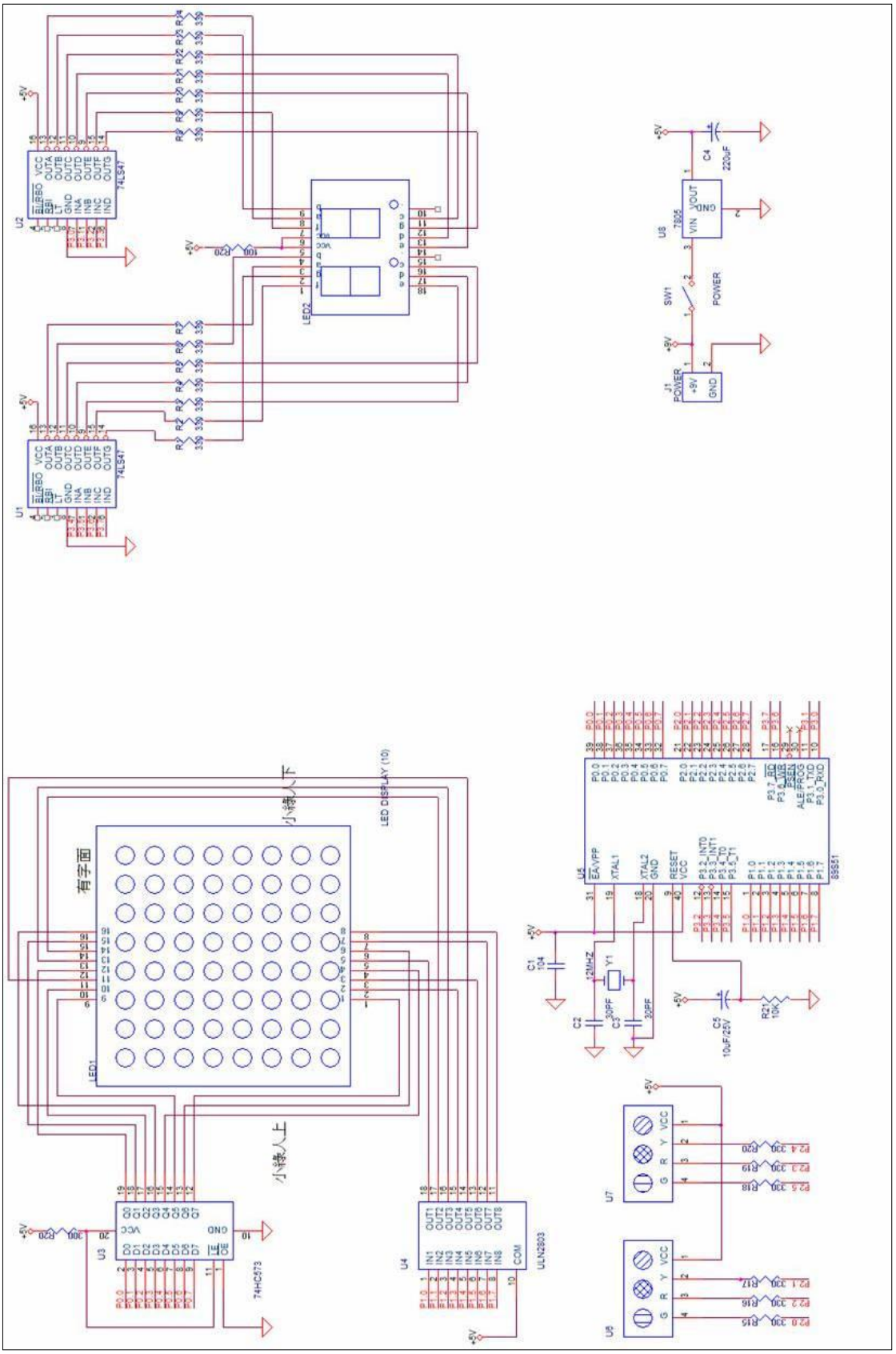


圖 3-7 紅綠燈模擬控制器電路

表 3-2 專題製作使用材料名稱覽表

材料名稱	規格	單位	數量
電阻	3K	個	1
電阻	50K	個	1
電阻	1K	個	2
電阻	10K	個	1
充電電池	6V	個	1
電解電容	220uF	個	1
陶瓷電容	104	個	1
陶瓷電容	30PF	個	2
電解電容	10uF/25V	個	1
LED		個	1
電阻	330Ω	個	20
電阻	50K	個	1
電阻	1K	個	2
電阻	10K	個	1
電源開關		個	1
七段顯示 IC	74LS47	個	2
BUFFER	74HC573	個	1
電晶體 IC	ULN2803	個	1
5V 穩壓 IC	7805	個	1
單晶片	89S51	個	1
2.5 穩壓 IC	LM385-2.5	個	1

四、小組分工配置

耀緯負責小組的資料，及整合簡報內容，過程中亦有購買相關書籍當成參考資料，選擇要如何去製作專題，讓智文及睿辰知道作何專題，然後再經小組討論、商量，有問題時，會再去徵詢老師意見。

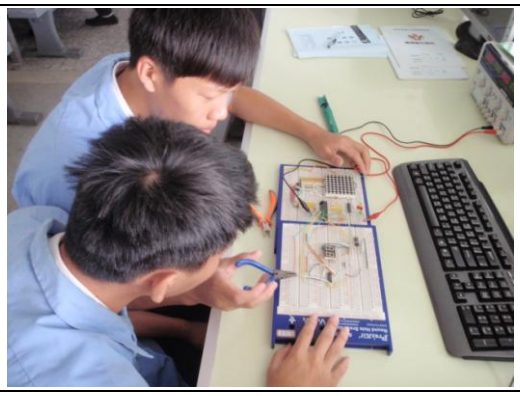
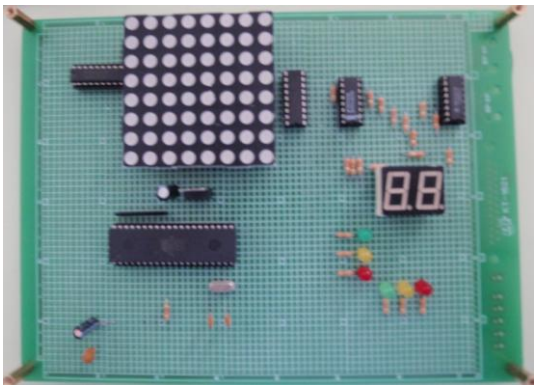
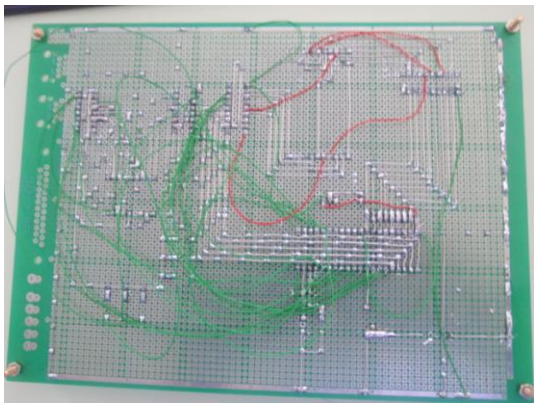
睿辰則是懂得如何做出紅綠燈模擬控制器的電路，負責焊接及畫出電路圖與焊接背面的電路圖；在製作過程中，如發現錯誤時，會在和小組想辦法如何補救，順便了解程式是否有作用。

耀緯和智文負責購買電路中所需零件，我們也討論過要多買一組當備用零件，假如一次就成功就可以把那份當作練習在完成一次，若不成功，則就用備份零件；且我們也可少買一些重複零件，如：IC、單晶片等，藉此便可節約這方面材料費，大家在去分配所需金額。

肆、製作成果

我們小組由決定題目，製作模擬電路、繪製設計電路圖，近而完成焊接製作整個電路；這整個流程，我們小組都用數位相機及相關電腦設備將之紀錄下來，經將這些資料整理過後，開始進行焊接工作，焊接前先設計Layout圖經由電腦呈現畫出後，就是開始進行焊接，由於電路複雜跨的線也不少，完成成品後，接著測試沒有任何反應動作，可能是ok線部分假焊了但是檢查過後，發現好像不是，於是接著從8051、7447、74HC573、ULN2803無論跨接、裸銅線找了再找也不是，發現電源部分還沒查緊接不放棄繼續尋找但始終無結果，最後七段及矩陣部份去詢問其他同學才發現矩陣接腳剪錯支了只好把大電路板縮小成普通PC板利用剩餘不多時間焊出完成品，相較之下OK線明顯的整塊都會是斷掉機率也大幅增加必須小心保護這是較不好的缺點。

以下是我們實作成果，如下圖所示：

<table>
<tr><td>  </td><td>  </td></tr>
<tr><td>圖4-1-1電路板成品量測</td><td>圖4-1-2麵包板檢測</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>圖4-1-3電路板成品圖(一)</td><td>圖4-1-4電路板成品圖(二)</td></tr>
</table>

伍、結論及建議

一、結論

(一) 透過此次專題製作學習的方式能幫助我們提升對課程的學習興趣。

(二) 透過此次專題製作學習的方式更能幫助我們獲得知識的建構及整合，且亦可以幫助我們提升其創造思考的能力。

(三) 小組同學認為專題製作學習為一主動探究的學習，學習中強調學習者必須負起主動探究學習的責任。

(四) 專題製作學習鼓勵小組成員分工和合作學習的精神。

(五) 小組同學認同資訊科技的知識在專題製作學習過程中，扮演著重要的角色，因其對電路製作、資料呈現及成果報告製作是很有幫助的。

(六) 專題製作學習可以培養我們學習者具備問題解決、研究、反省、團體合作及應用資訊科技等多項能力。

(七) 小組同學認為專題製作學習的階段中，會遇到不同的困難及問題，但看到自己的成品時，會很有成就感。

(八) 整體而言，我們小組同學認為專題製作學習是一有價值的學習方式，因其確實可以增進自己資訊科技的能力及其技能。

二、建議

我們在進行專題製作學習的過程後，提出以下幾點建議：

(一)焊接技術方面；再排零件時要注意零件大小，我們小組第一次沒有注意到就發生了8*8矩陣燈無法插下角座的局面，還有焊接時要注意OK線不要焊的長，會造成檢測電路板的困難度，而且很容易被扯斷，所以越短越好。

(二)學習前清楚的說明：請老師在進行專題製作學習前，能對學生清楚的說明整個專題進行的方式，包括專題報告的格式、課程進度的安排、需要的準備工具以及評量方式等，如都能在事前做好詳細的說明、規範，如此則能避免學生因疑惑而做錯方向。

(三)在學習過程中給予回饋：同學建議，在專題製作學習研究過程中，老師能否可以在學習的進行過程，給予立即性的回饋，讓學生可以及早發現其缺失，盡早進行改善。

(四)增長專題製作學習的時間：進行專題活動的學習，每個階段皆需完成一個學習報告，而單元學習的時間太少，連帶影響了期末完整報告的製作，所以希望老師能增長同學學習時間，讓成果報告的製作能更加完整，避免同學因時間緊迫而草率完成其作品。

參考文獻

1.蔡朝洋，2005 年，電子電路實作技術。台北：全華科技圖書

2.黃宏彥、余文俊、楊國輝編著，感測器原理與應用電路實習，高立圖書有限公司，1999。

3.蔡朝陽，電工實習（四），全華科技圖書股份有限公司，1994

4. 鄧錦城編著，8051單晶片實作寶典，益眾資訊有限公司，2000。

5.蔡朝洋，2005 年，單晶片微電腦8051/8951 原理與應用。台北：全華科技書。

6. 張義和，2003 年，主流電腦輔助電路設計（拼經濟版）。台北：全華科技圖書。

7.李春雄，2003 年，Visual Basic 6.0 學習實務。台北：新文京開發出版。

8. 傅榮鈞、林偉政、WonDerSun，2008 年，專題製作－單晶片篇（組合語言版）。台北：台科大圖書。

9.維基百科。8051。2013年5月6日，取自網址http://zh.wikipedia.org/zh-tw/8051。

10.維基百科。七段顯示器。2013年5月6日，取自網址 http://zh.wikipedia.org/zh-tw/%E4%B8%83%E6%AE%B5%E9%A1%AF%E7%A4%BA%E5%99%A8

附錄一　紅綠燈模擬控制器之程式碼

```
        ORG     0000H
        SETB    P3.0
        MOV     P2,#11111111B
MOVE:   JB  P2.4,M1
        AJMP    RFMOVE
M1: JB  P2.5,M2
        AJMP    RFRIGHT
M2: JB  P2.6,M3
        AJMP    RFLEFT
M3: JB  P2.7,M4
        AJMP    RFBACK
M4: CLRP2.0
        SETB    P2.1
        CLRP2.2
        SETB    P2.3
        ACALL   DELAYB
        SETB    P2.0
        SETB    P2.1
        SETB    P2.2
        SETB    P2.3
        ACALL   DELAYG
        JB  P3.0,M5
        AJMP    STOPR
M5: JB  P3.1,M6
        AJMP    STOPL
M6: AJMP    MOVE
BACK:
        SETB    P2.0
        CLRP2.1
        SETB    P2.2
        CLRP2.3
        ACALL   DELAYB
        SETB    P2.0
        SETB    P2.1
        SETB    P2.2
        SETB    P2.3
```

```
        ACALL   DELAYG
        RET
RFMOVE:     ACALL   WAIT2
RFMOVE_2:
        JNB P2.4,RFM1
        ACALL   WAIT2
        AJMP    MOVE
RFM1:   CLR P2.0
        SETB    P2.1
        CLR P2.2
        SETB    P2.3
        ACALL   DELAY
        AJMP    RFMOVE_2
RFRIGHT:
        ACALL   WAIT2
RFRIGHT_2:
        JNB P2.5,RFR1
        ACALL   WAIT2
        AJMP    MOVE
RFR1:   CLR P2.0
        SETB    P2.1
        SETB    P2.2
        CLR P2.3
        ACALL   DELAY
        AJMP    RFRIGHT_2
RFLEFT:
        ACALL   WAIT2
RFLEFT_2:
        JNB P2.6,RFL1
        ACALL   WAIT2
        AJMP    MOVE
RFL1:   SETB    P2.0
        CLR P2.1
        CLR P2.2
        SETB    P2.3
        ACALL   DELAY
        AJMP    RFLEFT_2
RFBACK:
```

```
        ACALL   WAIT2
RFBACK_2:
        JNB P2.7,RFB1
        ACALL   WAIT2
        AJMP    MOVE
RFB1:   SETB    P2.0
        CLRP2.1
        SETB    P2.2
        CLRP2.3
        ACALL   DELAY
        AJMP    RFBACK_2
STOPR:
        SETB    P2.0
        SETB    P2.1
        SETB    P2.2
        SETB    P2.3
        ACALL   DELAY
        MOV     R3,#70
STOPR_B:
        ACALL   BACK
        JB  P2.4,R1
        AJMP    RFMOVE
R1:  JB  P2.5,R2
        AJMP    RFRIGHT
R2:  JB  P2.6,R3
        AJMP    RFLEFT
R3:  JB  P2.7,R4
        AJMP    RFBACK
R4:  DJNZ    R3,STOPR_B
        SETB    P2.0
        SETB    P2.1
        SETB    P2.2
        SETB    P2.3
        ACALL   DELAY
RR:  MOV     R2,#80
RIGHT:
        CLRP2.0
        SETB    P2.1
```

```
        SETB    P2.2
        CLRP2.3
        ACALL   DELAYB
        SETB    P2.0
        SETB    P2.1
        SETB    P2.2
        SETB    P2.3
        ACALL   DELAYG
        JB  P2.4,RR1
        AJMP    RFMOVE
RR1:    JB  P2.5,RR2
        AJMP    RFRIGHT
RR2:    JB  P2.6,RR3
        AJMP    RFLEFT
RR3:    JB  P2.7,RR4
        AJMP    RFBACK
RR4:    DJNZ    R2,RIGHT
        JB  P3.0,WAIT
        AJMP    RR

STOPL:
        SETB    P2.0
        SETB    P2.1
        SETB    P2.2
        SETB    P2.3
        ACALL   DELAY
        MOV     R3,#70
STOPL_B:
        ACALL   BACK
        JB  P2.4,L1
        AJMP    RFMOVE
L1:     JB  P2.5,L2
        AJMP    RFRIGHT
L2:     JB  P2.6,L3
        AJMP    RFLEFT
L3:     JB  P2.7,L4
        AJMP    RFBACK
L4:     DJNZ    R3,STOPL_B
```

```
        SETB    P2.0
        SETB    P2.1
        SETB    P2.2
        SETB    P2.3
        ACALL   DELAY
LL:     MOV     R2,#80
LEFT:
        SETB    P2.0
        CLRP2.1
        CLRP2.2
        SETB    P2.3
        ACALL   DELAYB
        SETB    P2.0
        SETB    P2.1
        SETB    P2.2
        SETB    P2.3
        ACALL   DELAYG
        JB  P2.4,LL1
        AJMP    RFMOVE
LL1:JB  P2.5,LL2
        AJMP    RFRIGHT
LL2:JB  P2.6,LL3
        AJMP    RFLEFT
LL3:JB  P2.7,LL4
        AJMP    RFBACK
LL4:DJNZ    R2,LEFT
        JB  P3.1,WAIT
        AJMP    LL
WAIT:   SETB    P2.0
        SETB    P2.1
        SETB    P2.2
        SETB    P2.3
        ACALL   DELAY
        AJMP    MOVE
WAIT2:  SETB    P2.0
        SETB    P2.1
        SETB    P2.2
        SETB    P2.3
```

```
        ACALL   DELAY2
        RET
DELAYG:
DLL0:   MOV R6,#1
DLL1:   MOV R7,#11
DLL2:
        DJNZ R7,DLL2
        DJNZ R6,DLL1
        RET
DELAYB:
DLB0:   MOV R6,#50
DLB1:   MOV R7,#100
DLB2:
        DJNZ R7,DLB2
        DJNZ R6,DLB1
        RET
DELAY2:  MOV  R5,#1
DL20:   MOV R6,#250
DL21:   MOV R7,#200
DL22:   DJNZ R7,DL22
        DJNZ R6,DL21
        DJNZ R5,DL20
        RET
DELAY:  MOV   R5,#5
DL0:    MOV R6,#250
DL1:    MOV R7,#200
DL2:    DJNZ R7,DL2
        DJNZ R6,DL1
        DJNZ R5,DL0
        RET
END
```