逢　甲　大　學

資訊工程學系專題報告

◎8051應用◎

◎ 電子密碼鎖◎

學　生：　顏 吉 村

指導教授：徐 弘 洋

中華民國九十三年四月

逢甲大學

資訊工程學系

專題報告

8051應用電子密碼鎖

顏吉村

93

圖 表 目 錄

第一章　　導論

在修習數位電路時，有許多作業，當時並不太了解其原理，當時發現許多 IC 經由電路連接後，可以產生許多功能，覺得相當有意思。之後有一門課為「微處理機介面」時，發現一顆小小的 8051IC 可以做許多的事情，於是想說如果可以將其應用在的常生活中，也是一件不錯的事情，譬如現在社會上經濟不景氣，許多人便挺而走險做些不正當的行為造成他人的損失，如偷竊，若能運用 8051 做一個密碼鎖，也是一個不錯的方法，當然這個專題只是一個小小的密碼鎖功能不一定很強大，但若持續研究那也可能成為一個不錯密碼鎖。

第二章　　8051 概述

2.1 MCS-51 簡介

2.2 MCS-51 的接腳

2.3 8051 的記憶體結構

2.3.1　程式記憶體

2.3.2　外部資料記憶體

2.3.3　內部資料記憶體

2.4 MCS-51 的中斷結構

2.4.1 中斷致能與優先權結構

2.4.2 中斷如何動作

2.1 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。

圖 2-1 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)

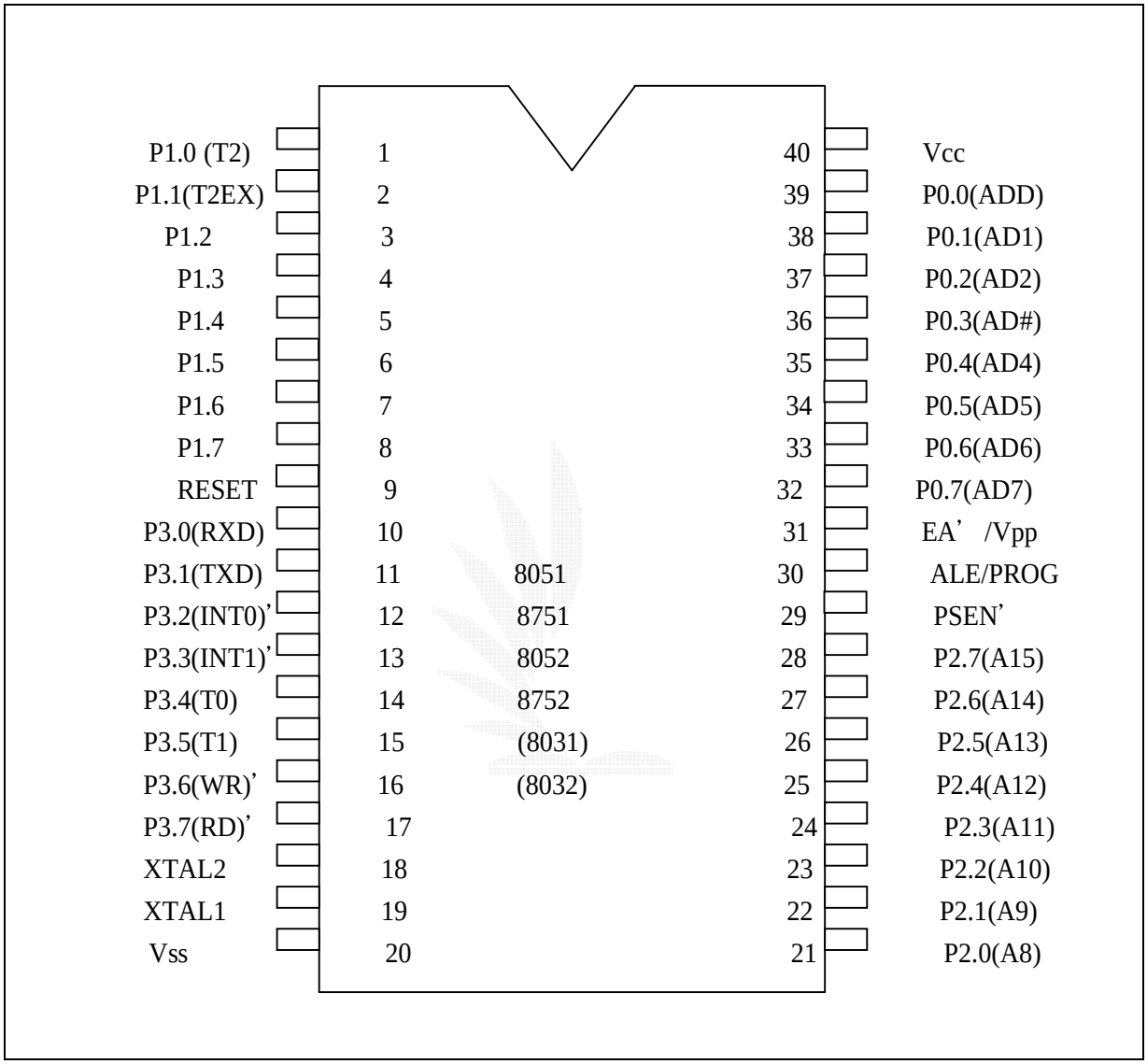
9. 內部有 4K 的程式記憶體

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)

2.2 MCS-51 的接腳

圖 2-2　MCS-51 的 40Pin DIP 包裝的接腳圖



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。

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。

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部。T0 是 Timer0 的外部脈波輸入腳。T1 是 Timer1 的外部脈波輸入腳。WR，RD，當您再 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。

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。

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% 。

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。

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。

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。

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。

31 腳(EA)：

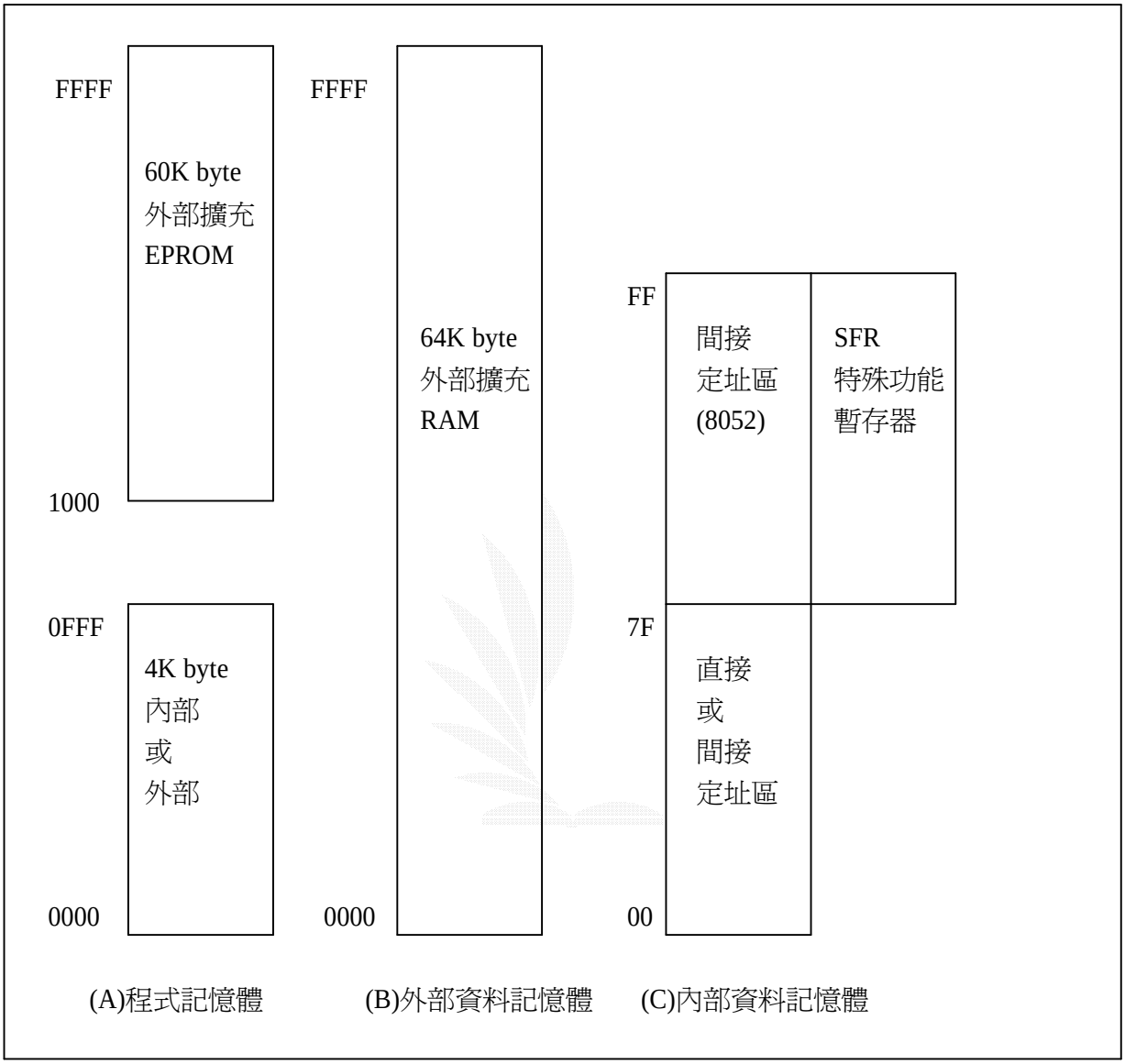
這是一支輸入腳，當 EA=0 時，8051 一律執行外部程式記憶體裡的程式，因此 8051 內部的 4K 程式記憶體就沒有用了。因此如果您要使用內部的程式記憶體時，一定要將 EA 接+5V。因為 8031(或 8032)內部沒有程式記憶體，它的 EA 必須接地。

2.3 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

圖 2-3　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。

圖 2-4　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。

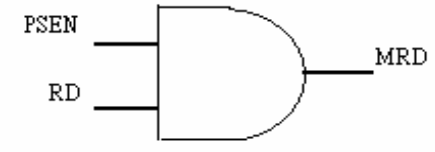


圖 2-5　將 PSEN 與 RD 合併成 MRD

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：

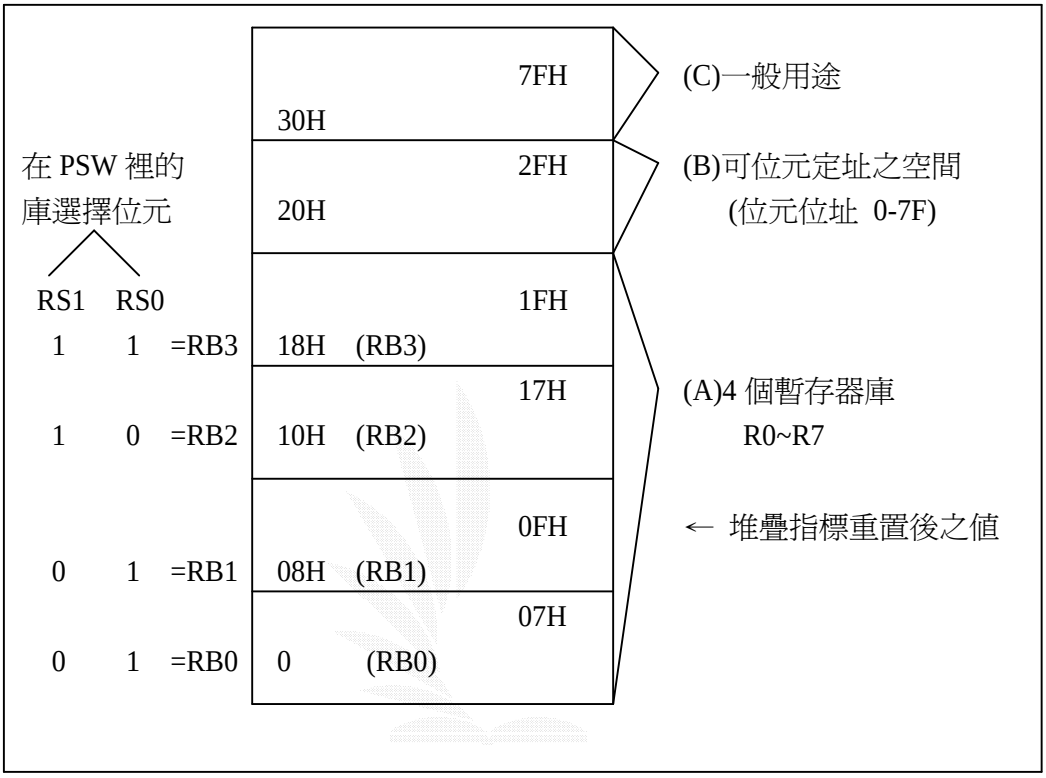
1. 位址 00H~7FH 的 RAM
2. 位址 80H~FFH 的 RAM
3. 位址 80H~FFH 的 SFR

位址 00H~7FH 的RAM

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 2-6　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。

位元位址區的另外 128 個 bit(80H~FFH)，是在 SFR 暫存器裡，如下圖所示。

(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。

圖 2-7　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)							
	RB0 (8 bytes)							
00H								

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

圖 2-9 SFR 各個暫存器重置後的初始值

*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	

1.累加器　　(ACC)

ACC 就是累加器暫存器，累加器指令所使用的助憶符號是 A。

2.B 暫存器 (B)

B 暫存器是使用在乘法(MUL AB)和除法(DIV AB)指令時。在其它的指令，它可以被當成一般暫存器處理。

3.PSW(程式狀態字語)暫存器

PSW 實際上就是一般 CPU 理所稱的旗號(Flags)暫存器，內部包含有 CPU 的系統狀態資料。

4.SP 暫存器(堆疊指標)

SP 暫存器只有 8 位元寬，因此 MCS-51 的堆疊區最多只有 256byte，並且一定在內部 RAM 裡。當執行 PUSH 或 CALL 指令存入資料到堆疊區之前，SP 的內容會先被加 1。因為 SP 的內容可由指令任意改變，因此堆疊區可以由使用者設定在內部 RAM 裡的任一個位址。但是，在 RESET 之後，SP 的內容會被設成 07H，因此 RESET 後堆疊區是從位址 08H 開始。

5.DPTR(資料指標)

DPTR 是一個 16 位元暫存器，它是由兩個 8 位元暫存器所組成，高位元組為 DPH，低位元組為 DPL。因 DPTR 可看成一個 16

位元暫存器或看成兩個 8 位元暫存器加以處理。DPRT 的用途是用來定址外部資料記憶體

(MOVX A,@DPTR)，或程式記憶體使用(MOVC A,@A+DPTR)，因此 MCS-51 可以定址程式記憶體或資料記憶體各 64K。

6.P0，P1，P2，P3 暫存器

P0，P1，P2，P3，是 MCD-51 四個 I/O port 的輸出栓鎖器(Latch)。

7.SBUF(串列阜緩衝區)

SBUF 暫存器，實際上是由兩個暫存器構成，一個是當作 UAR 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 這條線傳出去。若去讀 SBUF，就會讀到接收緩衝區的資料。

8.計時器暫存器

(TH0，TL0)，(TH1，TL1)和(TH2，TL2)這三對暫存器，是三個 16 位元計時器的名稱分別為 Timer0，Timer1 和 Timer2。

9.捕捉式暫存器(Capture Register)

(RCAP2H，RCAP2L)這一對暫存器稱為捕捉式暫存器，當 8052 的 Timer2 工作在 ”捕捉模式” (“Capture mode”)時，當

T2EX(P1.1)這支接腳上的輸入信號 1←0 轉態時，TH2，TL2 的內容會被移入 RCAP2H，RCAP2L 就是保存 TH2，TL2 的重新載入值的地方。

10.控制暫存器(Control Registers)

IP，IE 暫存器控制 MCS-51 的中斷系統；TMOD，TCON 暫存器是控制 Timer0 和 Timer1 的工作模式；T2CON 控制 Timer2 的工作模式；SCON 控制 UART 的工作模式等。

以下就將這些暫存器裡的各個位元的名稱及功能以圖形表示：

PSW：程式狀態字語(PROGRAM STATUS WORD,可位元定址)
位址：(D0H)

CY	AC	F0	RS1	RS0	0V	-	P

CY PSW.7 進位旗標

AC PSW.6 輔助進位旗號

F0 PSW.5 一般用途

RS1 PSW.4 暫存器庫選擇位元 1

RS0 PSW.3 暫存器庫選擇位元 0

0V PSW.2 溢位旗號

- PSW.1 保留將來使用

P PSW. 同位元旗號，在每個指令週期，由硬體設定或清除，以指出累加器內 ”1” 的個數爲奇數或偶數

RS1	RS0	暫存器庫	位　址
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

圖 2-10　PWS 暫存器

PCON：電源控制暫存器(POWER CONTROL REGISTER)
位址：(87H)

SMOD	-	-	-	GF1	GF0	PD	IDL

SMOD　雙倍率位元，當串列工作於模式 1，2 或 3 時，如使用 Timer1 做鮑率產生器，且 SMOD=1 則鮑率爲雙倍

\-　保留將來使用

\-　保留將來使用

\-　保留將來使用

GF1　一般用途

GF0　一般用途

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

IDL　IDLE 模式位元，80C51BH 時，設定此位元就進入 IDLE 模式

若同時寫 1 至 PD 和 IDL 時，PD 優先

圖 2-11　PCON 電源控制暫存器

IE：中斷致能暫存器(INTERRUPT ENABLE REGISTER，可位址定址)
位址：(A8H)

EA	-	ET2	ES	ET1	EX1	ET0	EX0

EA	IE.7	如果 EA=0 時，禁止所有中斷，如果 EA=1，則各中斷是否被接受，由各自之中斷致能位元加以設定。
-	IE.6	位使用；保留給將來使用
ET2	IE.5	致能 Timer2 溢位或補入之中斷(8052)
ES	IE.4	致能串列阜之中斷
ET1	IE.3	致能 Timer1 之中斷
EX	IE.2	致能外部中斷 1 之中斷
ET0	IE.1	致能 Timer0 之中斷
EX0	IE.0	致能外部中斷 0 之中斷

圖 2-12　IE 中斷致能暫存器

IP：中斷優先權暫存器(INTERRUPT PRIORITY REGISTER，可位元定址)
位址：(B8H)

-	-	PT2	PS	PT1	RX1	PT0	PX0

-	IP.7	未使用；保留給將來使用
-	IP.6	未使用；保留給將來使用
PT2	IP.5	定義 Timer2 之優先權層次(8052)
PS	IP.4	定義串列阜之優先權層次
PT1	IP.3	定義 Timer1 之優先權層次
PX1	IP.2	定義外部中斷 1 之優先權層次
PT0	IP.1	定義 Timer0 之優先權層次
PX0	IP.0	定義外部中斷 0 之優先權層次

圖 2-13　IP 中斷優先權暫存器

TCON：計時器/計數器控制暫存器
(TIMER/COUNTER CONTROL REGISTER)
位址：(88H)

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1： TCON.7 計時器 1 之溢位旗號，當計時器/計數器 1 溢位時，會被硬體設定爲 1。當處理器執行中斷服務時，硬體會自動清除此位元。

TR1： TCON.6 計時器 1 之啓動位元。由軟體設定/清除以啓動/停止計數。

TF0： TCON.5 計時器 0 之溢位旗號，當計時器 0 溢位時，會被硬體設定爲 1。當處理器執行中斷服務時，硬體會自動清除此位元。

TR0： TCON.4 計時器 1 之啓動位元。軟體設爲 1 時啓動，0 時停止。

IE1： TCON.3 外部中斷 1 之中斷旗號。當外部中斷被檢知時，硬體會設定此位元。當中斷被處理時，硬體會自動清除此位元。

IT1： TCON.2 外部中斷型態控制。當設定爲 1 時中斷型態爲負緣觸發。當此位元若爲 0 時，則爲低準位觸發。

IE0： TCON.1 外部中斷 0 之邊緣旗號。當檢知外部中斷時，此位元會被硬體設爲 1，執行中斷服務程式時，硬體會將之清除。

圖 2-14 TCON 計時器/計數器控制暫存器

TMOD：計時器/計數器模式控制暫存器
(TIMER/COUNTER MODE CONTROL REGISTER)
位址：(89H)

GATE	C/T	M1	M0	GATE	C/T	M1	M0
| - - - - - - - - Timer 1 - - - - - - - - |				| - - - - - - - - Timer 2 - - - - - - - - |			

GATE　當 TRx(在 TCON) =1 且 GATE=1，則計時器只有在 INTx 接腳爲高電位時才會計時。當 GATE=0，則計時器只在 TRx=1 時會計時。

C/T　計時器或計數器之選擇位元。C/T=0 時，爲計時器；C/T=1 時，位計數器。

M1　模式選擇位元。

M0　模式選擇位元。

M1	M0		工作模式
0	0	0	13-bit　計時器
0	1	1	16-bit　計時器/計數器
1	0	2	8-bit　自動載入
1	1	3	(Timer 0)TL0 爲 8 位元計時器/計數器，由標準之計時器 0 之控制位元控制，TH0 爲 8 位元計時器，且由計時器 1 控制位元控制。
1	1	3	(Timer 1)計時器/計數器 1 停止。

圖 2-15　TMOD 計時器/計數器模式控制暫存器

T2CON：TIMER2 控制暫存器

(TIMER/COUNTER 2 CONTROL REGISTER，可位元定址)

只有 8052 有

位址：(C8H)

TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2

TF2	T2CON.7	計時器 2 之溢位旗號，由硬體設定，軟體清除，當 RCLK=TCLK=1 時 TF2 不會被設定
EXF2	T2CON.6	計時器 2 的外部旗號，在 T2EX 受負緣觸發或 XEN2=1 而發生補入或重新載入時此位元會被設定，當計時器 2 之中斷致能時，EXF2=1 會使 CPU 執行計時器 2 之中斷服務程式，若 RCLK=0 時串列阜能使用計時器 1。
RCLK	T2CON.5	接收脈波旗號，此位元被軟體設爲 1 時，串列阜就使用計時器 26 溢位脈波當作其接收脈波(串列阜在模式 1，3 時)。若 RCLK=0 時串列阜能使用計時器 1。
TCLK	T2CON.4	傳送脈波旗號
EXEN2	T2CON.3	計時器 2 之外部致能旗號。此位元爲 1 時，允許 T2EX 接腳上有逆緣觸發信號(1→0)時，有插入或重新載入功能，但必須計時器 2 沒有作串列埠的鮑率產生器時。當 EXEN2=0 時，會使計時器 2 忽略 T2EX 之信號。
TR2	T2CON.2	計時器 2 之啓動/停止開關，TR2=1 時啓動計時器。
C/T2	T2CON.1	計時器或計數器之選擇開關，0=內部計時器，1=外部事件計數器(負緣觸發)。
CP/RL2	T2CON.0	補入/重新載入旗號。若此位址爲 1 且 EXEN2=1 時，當 T2EX 有負緣信號時會有補入動作。若此位元爲 0，當計時器 2 溢位或 EXEN2=1 且 T2EX 上有負緣觸發信號時會有重新載入之動作。當 RCLK=1 或 TCLK=1 時，此位元會被忽略，且計時器會被強迫成自動重新載入。

圖 2-16　T2CON：TIMER2 控制暫存器

SCON：串列埠控制暫存器

(SERIAL PORT CONTROL REGISTER，可位元定址)

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 在串列埠爲模式 2 和 3 時，致能多處理器通信之功能。在模式 2 或 3；如果 SM2=1 則當接收到的第 9 個資料位元爲 0 時 RI 不動作。在模式 1 時，若 SM2=1，當接到的停止位元不正確時 RI 也不動作。在模式 0 時 SM2 必須爲 0。

REN SCON.4 由軟體去設定或清除，決定是否接收。(REN=1 接收)

TB8 SCON.3 在模式 2 或 3 時，傳送的第 9 個資料位元。由軟體控制。

RB8 SCON.2 在模式 2 或 3 時，接收到的第 9 個資料位元放在這個位元裡。在模式 1 時；如果 SM2=0，則 RB8 爲接收到的停止位元；模式 0 時，RB8 沒有用。

TI SCON.1 傳送中的旗號，在模式 0 時，在第 8 位元結束時由硬體設爲 1，或再其它模式時，是在停止位元的開始時設定爲 1。此位元必須由軟體清除。

RI SCON.0 接收中的旗標，在模式 0 時，在第 8 位元結束時，硬體會將其設爲 1，在其它模式時，在停止位元的一半的時候由硬體設定(看 SM2)。此位元必須由軟體清除。

SM0	SM1	模式	說明	鮑率
0	0	0	移位暫存器	Fosc./12
0	1	1	8-bit UATR	可變
1	0	2	9-bit UATR	Fosc./64 或 Fosc./32
1	1	3	9-bit UATR	可變

圖 2-17 SCON 串列埠控制暫存器

2.4 MCS-51 的中斷結構

8051 提供了 5 個中斷源,8052 則提供了 6 個中斷源,外部中斷 INT0 和 INT1 可以規劃成半位元觸發或邊緣觸發的方式動作，它是由 TCON 暫存器裡的 IT0 和 IT1 位元所選擇；而 TE0 和 TE1 是真正產生中斷的旗號(當 IT0 和 IT1 等於 1 時)；如果中斷是設成邊緣觸發的話，當中斷產生時,在執行中斷服務程式的同時,硬體會將這個旗號(IE0 或 IE1)清除；而準位觸發方式則不會清除。

Timer0 和 Timer1 的中斷是由 TF0 和 TF1 旗號所產生，這些旗號是在它們各自的計時/計數暫存器溢位時所設定(Timer0 在模式 3 時例外)。當計時器產生中斷時，硬體會在執行到其服務程式時自動將這些旗號(TF0 或 TF1)清除成 0。

串列埠的中斷是由 RI 和 TI 的邏輯 OR 之後所產生，在執行 RI 和 TI 的中斷服務程式時，硬體不會自動將 RI 或 TI 清除，因為串列埠的中斷服務程式必須判斷中斷是由 RI 或 TI 產生，因此產生中斷的旗號必須由軟體加以清除。

在 8052 時 Timer2 的中斷是由 TF2 和 EXF2 的邏輯 OR 的結果所產生，當執行到其中斷服務程式時，這些旗號不會被硬體清除，因為，中斷服務程式必須判斷中斷是由 TF2 或 EXF2 所產生，然後以軟體將

其清除。

以上所討論的每個產生中斷的旗號(IE0，IE1，TF0，TF1，TI，RI，TF2，EXF2)，都可以透過軟體加以設定或清除，其結果與硬體所產生設定或清除的效果完全一樣。如此一來就可以透過軟體產生中斷，或將正在等待的中斷清除。

2.4.1 中斷致能與優先權結購

透過設定或清除 SFR 裡的 TE 暫存器的位元，這些中斷裡的每個中斷源可以個別的加以致能(Enable)或禁能(Disable)產生中斷。IE 暫存器裡也包含了一個禁止所有中斷產生的位元 EA，它可以立即禁止所有中斷產生。

MCS-51 的優先權層次共有兩層，即高優先權和低優先權，每一個中斷源都可以透過軟體規劃 SFR 裡的 IP 暫存器裡的位元，以設定各中斷優先權順序，當設為 ”1” 時列在高優先權層次，”0” 時列在較低優先權層次。較高優先層的中斷源可以中斷較低優先層的中斷服務程式，同優先權層次的中斷副程式不能相互中斷。

如果兩個不相同優先權層次的中斷同時發生，則優先權層較高的中斷會先被接受；如果相同優先權層次的中斷同時到達，就由內部硬體的輪詢(polling)順序決定要先接受哪一個。因此在每個優先權層

次裡又有一個由輪詢順序所決定的第二個優先權結構，這個輪詢順序如下表 4 所示：

圖 2-18　優先權輪詢順序

中　斷　源	優　先　權　層　次
1.　IE0	最高
2.　TF0	
3.　IE1	↓
4.　TF1	
5.　RI+TI	最低
6.　TF2+EXF2	

2.4.2　中斷如何動作

在 IE 暫存器裡的中斷旗號會在每個機械週期的 S5P2 時間備取樣，取樣進來的旗號會在下一個機械週期加以檢查(polling)。如果這些旗號中有一個為”1”，會在這個週期裡發現，然後中斷系統會產生 LCALL 到其相對應的中斷服務程式，但是如果有下面所列的狀況時就不產生中斷動作：

甲、　有相同或更高優先權的中斷正在處理時。

乙、　目前的週期(檢查，polling)不是正在執行的這個指令的

丙、　最後一個週期。

丁、　正在執行的指令是 RETI，或任何寫入 IE 或 IP 的指令。

以上所別的三個狀況都會禁止產生 LCALL 至中斷服務程式。條件 2 是保證目前正在執行的指令會在跳至服務程式前被完整的執行完。條件 3 是保證在執行完 RETI 或寫至 IE，IP 的指令後會再執行一個指令後才會被中斷。

檢查中斷旗號的週期，會在每個機械週期重複動作，且這個被檢查的旗號是在前一個機械週期的 S5P2 時取樣進來的狀態。因此，如果有一個中斷要求動作了，但是卻因上面所說三個狀況的關係而未被 8051 反應，當以上所列的狀況解除了，如果中斷的動作旗號沒有繼續維持，則上面所要求的中斷不會被接受。換句話說，事實上中斷旗號一旦動作，但沒有被接受，它也不會被記住，每一個檢查週期的值都是新的。

CPU 回應中斷要求的方法是，由硬體產生 LCALL 指令至中斷要求的服務程式；在某些狀況下，硬體也會清除引起中斷的旗號，有些則不會；UART 和 Timer2 的中斷旗號不會被清除，而必須要由使用者的軟體加以清除。外部中斷旗號(IE0 和 IE1)只有在其為邊緣觸發模式時會自動被硬體清除。在硬體產生 LCALL 時，會將程式計數器

(Program Counter)的值堆入堆疊(但沒有存 PSW)，然後載入中斷源之中斷向量位址,最後跳入中斷副程式,各中斷源之向量位址如下表 5:

圖 2-19　中斷源之向量位址

中　斷　源	中　斷　向　量
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
RI ＆ TI	0023H
TF2 ＆EXF2	002BH

中斷副程式的執行是由以上所列之位址開始，直到執行到 RETI 為止。RETI 指令是告知 CPU 這個中斷已經結束，然後從堆疊區的頂端取回兩個位元組的返回位址到程式計數器，以返回原來被中斷的程式。

請特別注意，RET 指令也會使中斷副程式返回被中斷的程式，但如此會使得中斷系統認為中斷仍在動作中，而無法接受其它的中斷。

外部中斷源可規劃成準位動作(Level-activated)或邊緣動作(Transition-activated)，這個動作可透過規劃 TCON 的 IT1 或 IT0 位元加以設定。如果 Itx=0，則外部中斷 x 就會在檢知 INTx 接腳的信號為低電位(Low)時觸發。如果 Itx=1，則外部中斷 x 是使用邊緣觸發。在這個模式下，如果在 INTx 接腳在上一個週期裡被檢知為高電位，下一個週期檢知為低電位時，則中斷要求旗號 Iex 就被設定成 ”1” ，此 Iex 就對 CPU 提出中斷要求。

因為外部中斷輸入腳是在每個機械週期被取樣一次，因此輸入的高電位或低電位信號必須至少維持 12 個振盪週期，以保證中斷被檢知。

如果外部中斷型式是邊緣觸發式時，外部中斷源必須維持高電位一個週期，然後維持低電位也必須至少一個週期，以保證位準的轉移被檢知到，外部中斷要求旗號 IEx 才會被設定為”1”。當中斷服務程式被呼叫時，CPU 會自動將 IEx 清除。

如果外部中斷是用準位觸發型態時，外部中斷源必須維持中斷要求信號確實被 CPU 接受而產生中斷為止。並且在中斷服務程式結束前，中斷要求信號必須拿開，否則就會一直產生下去。

第三章　LCD 的介紹

3.1 LCD 的簡介

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。

3.2 LCD 硬體說明

下表為文字型 LCD 接腳說明

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。

(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 (Data Register ， DR) 。功能為：

a. 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。

b. 當寫資料到 LCD 時，前一個位址指令已決定了是要寫入 DD RAM 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 。

c. 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 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；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，而本組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	顯示位置
00H	01H	02H	03H	04H	·····	26H	27H	第一列位址
40H	41H	42H	43H	44H	·····	66H	67H	第二列位址

01H	02H	03H	04H	05H	·····	27H	00H	設顯示左移
41H	42H	43H	44H	45H	·····	67H	40H	(Left shift)

27H	00H	01H	02H	03H	·····	25H	26H	設顯示右移
67H	40H	41H	42H	43H	·····	65H	66H	(Right shift)

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) ， CG ROM 中存放著 160 個 5*7 點矩陣字型 (dot-martrix character pattern) 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。 CG RAM 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 CG ROM 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。

3.3 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為游標所在位置的字元閃爍	40us
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 DD RAM 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

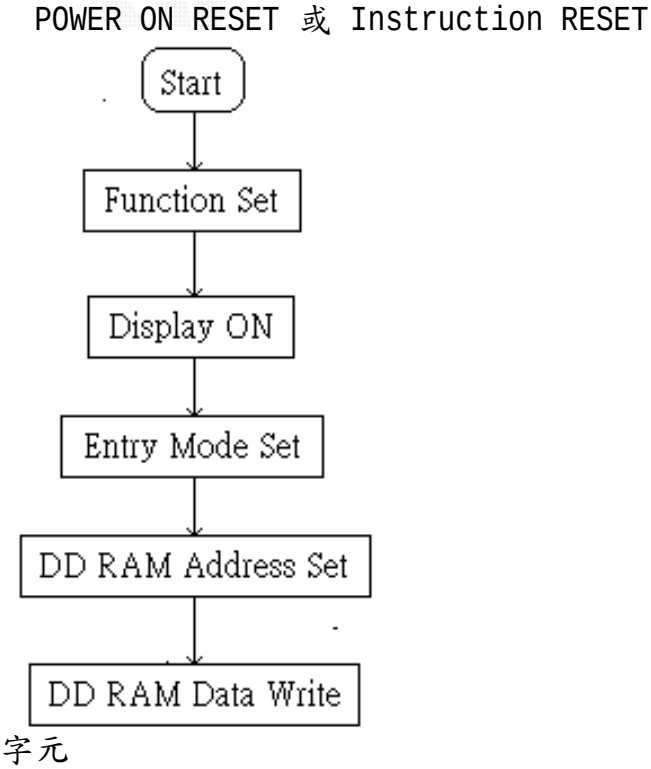
(11)字元產生器 RAM /顯示資料 RAM 的資料讀取 (CG RAM / DD RAM Data Read) ，資料是由 CG RAM 還是 DD RAM 中的哪一個位址讀出，皆由先前所下的位址設定指令決定，與寫入類似。讀或寫資料都會使位址自動加一或減一。指令碼為：

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

D：Data bit

3.4 軟體規畫

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：



3.5 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 Power

ON Reset ，並做 LCD 初始化工作如下：(初始化時 BF=1， VDD 達 3.5V 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　　實作

□ 單晶片可以使用 89C51 系列晶片及 LO51 晶片(ISP 型 8051 晶片)

□使用 LO51 晶片可以擴充簡易 ICE 模擬器功能

□使用 LO51 晶片免燒錄，免 ICE 可由串列介面下載程式碼來執行

□一組 4X4 鍵盤輸入

□ 壓電喇叭或一般喇叭輸出

□ 含文字型 LCD 介面(16X2)

□ 含斷電資料保存介面 93C46

□ 1 組 5V 繼電器界面

連接座:

J0　2P MOLAX 腳座 +9V/GND　電源輸入

J1 2P MOLAX 腳座 +5V/GND 電源輸入

J2 7X2 LCD 母座

J3 8P 母座 連接 4X4 鍵盤

J4 20X2 多功能 I/O 擴充接腳，有 3 種功能:

功能 : 20X2 排針座，經由 40 PIN 排線可以擴充簡易 ICE 模擬器功能。當做簡易 ICE 模擬器功能時，原先在 IO51 板上的硬體零件會造成負載效應，查看電路圖，取下必要的 IC， 才能經由 40 PIN 排線送出標準的 8051 控制信號。

J0 9V DC 接頭 中間為 +9V

零件表

積體電路 IC: IC 座

編號 規格

LCD 16X2 文字型 LCD

U1 ISP LO64 40P IC 座

U2 7660 8P IC 座

U3 DA08 16P IC 座

U4 7805 加散熱片

電容 : 電解電容有極性(長腳為正)，陶瓷電容無極性

C1 10 uF 陶瓷電容 C2 10 uF 陶瓷電容

C3 100 uF 電解電容 C4 100 uF 電解電容

C5 100 uF 電解電容

電晶體 : Q1、Q2 2SC945 X4 有極性 (底視圖 由左至右 ECB)

二極體 : D1、D2 LED X2 有極性(長腳為正)

其他 :

X1 石英振盪晶體 11.0592 MHz

RY1 5V 繼電器

SW1 4 P 按鍵開關

BZ 壓電喇叭

4X4 鍵盤

基本知識如下:

LCD 是插在 J2 7X2 LCD 母座上。

J3 8P 母座用來連接 4X4 鍵盤

8051 控制板 DIY 不動作時，常見的問題有以下幾種:

□ 8051 不動作

□ 零件接腳極性方向插錯

□焊接不良

8051 要動作，基本電路(以下 6 接腳)要接對

<1>8051 接腳 40 電源是否接 5V

<2>8051 接腳 20 接地是否接地

<3>8051 接腳 18 19 是否接石英震盪晶體

<4>8051 接腳 31 EA 是否接 5V

<5>8051 接腳 9 重置是否接對

□準備必要電源及串列介面傳輸線

接著要自備 +5v 電源或是 +9V 電源 或是 9V 直流電源(350 mA)接頭中間為正。

□ 如何觀察原始程式 ：

以文書處理器查看 PA.ASM。

□ 如何執行程式 ：

1.準備一 +5V 電源。

2.經由 2 PIN 電線接至 J1 接點(紅色+5V，黑色接地)

3 .打開電源，工作 LED 閃爍，表示開機正常。

4.原定密碼為 "1234"。

5.按 B 鍵 ： 檢查密碼輸入，輸入 4 位密碼，若正確則繼電器開啟約 2 秒後關閉，表示打開門鎖，若密碼比對錯惡則警報響起。

7.按 A 鍵 ： 輸入新的 4 位密碼，關機後資料可以永久保存。

□ 組譯有錯時怎辦 ：

組譯有錯時會自動跳出批次檔，而結束程式執行，可以文書處理器查看檔案，看看到底錯在那邊。

第五章 總結

5.1 實驗心得

在這個專題裡遇到許多困難，首先在於如何推動這 8051 晶片，書上提到需要 5V 的電源可是在各個電子專賣店裡總是找不到 5V 的電源供應器，更別說是 1.5V 的電池了怎麼樣也串不到 5V，在參考了許多書與詢問同學後，發現有一種藉由 9V 的電源供應器、7805 跟 5V 穩壓 IC，才完成了自制 5V 電源，對於燒 IC 為了不必要的損失，決定使用 89C51 可重覆燒錄，也對 8051 這個 IC 更了解了，也深深感到這個 IC 其強大的功能。雖然這只是一個小小的專題，不過我相信若是在這個小小的基礎上，繼續研究的話，以後或許會有更深的發現。

5.2 問題討論

Ｑ１：程式如何燒錄進ＩＣ？

Ａｎｓ：這是我準備將程式燒進８０５１時所遇到的問題，向器材室詢問得知，器材室有一台８０５１之ＩＣ燒錄器可供借用，而由於８９Ｃ５１較易於重覆燒錄，因此我決定用８９Ｃ５１來燒錄程式，軟體的取得是從網站上下載。一開始以為燒錄失敗，後來才知道是燒錄時間極短，且線路接觸不良（麵包板），與程式有小 bug 導致失敗，經過重覆測試，終於燒錄成功，並且完成測試。

Q2 為何使用８９Ｃ５１要用石英振盪器？

Ａｎｓ：石英振盪器主要的目的是要產生脈波，產生所需工作頻率讓８９Ｃ５１產生動作

Q3: ＣＤ為何無法做動態顯示？

Ａｎｓ：測試數個動態顯示的程式仍舊無法做動態顯示，因此推斷因為非繪圖型ＬＣＤ，因此無法做動態顯示。

Q4: 為何無法連續兩次接收資料?

Ａｎｓ：電路不穩，是因為電壓不夠，因此只要換過新的電池，問題即解決。

Q5：為何ＬＣＤ顯示跳動的很厲害？

Ａｎｓ：程式出錯，經 debug 過後再做顯示，問題就消失了。

5.3 參考資料：

鄧錦城，8051 單晶片實作寶典，P.1~P.206、P.397~P.427，松崗電腦圖書資料股份有限公司，八十一年九月出版。

偉林資訊，單晶片微電腦－8051/52 原理與應用，P.10~P.95，松崗電腦圖書資料股份有限公司，八十年十月出版。

王信福，MCS-51 單晶片微電腦專題製作，P.1-3~P.1-17、P.C-2~P.C-11，松崗電腦圖書資料股份有限公司，八十六年一月出版。

陳粵初、沈德金，單晶片微處理機界面電路與應用程式實例，P.4~P.16、P.191~P.234，格致圖書公司，八十一年四月出版。

劉銘中、林琮烈、陶德福，MCS-51 單晶片原理與 I/O 應用，P.1-3~P.3-19、P.4-2~P.6-33、P.10-2~P.10-17，八十六年九月出版。

李齊雄、游國幹，8051 單晶片微電腦原理與實作，P539~P555，格致圖書公司，八十一年七月出版。

附錄:

```
;-------------------------------------------------
; PA.ASM  8051 LCD PASSWORD 4  WITH 93C46
; 8051 ASM USE: 2500 A.D.
;-------------------------------------------
; 8051 I/O DEFINE
;-------------------------------------------
; P0: P0.0 LCD RS     P0.4--P0.7  LCD D4--D7
;     C46: CLK-P0.5  DI-P0.6  DO-P0.7
; P1: P1.0 WORKING LED
;
; P2: P2.0~P2.3 : SCAN CODE
;     P2.4~P2.7 : K1 K2 K3 K4
; P3.0 : RXD   RS232 RX
; P3.1 : TXD   RS232 TX
; P3.2 : INT0\ ALARM
; P3.3 : INT1\
; P3.4 : T0    C46 CS HIGH
; P3.5 : T1    RELAY TO OPEN DOOR
; P3.6 :       LCD EN
; P3.7 :
;-------------------------------------------------------
ALSEC     EQU 1      ; ALARM SEC 1=1 SEC   10=10 SEC
KA   EQU 0AH  ; KEY NO DEFINE
KB   EQU 0BH
KC   EQU 0CH
KD   EQU 0DH
KE   EQU 0EH
KF   EQU 0FH

; VAR DEFINE..................
PASS EQU 30H  ; 30 31 32 33 34 35  I/P TO CHECK
PASSNEW EQU 36H  ; 36 37 38 39 3A 3B

BUF_KEY EQU 4AH  ; KEY DATA BUFFER
KEY_NO    EQU 4BH  ; KEY NO
CO   EQU 4CH  ; COUNTER
LCDE EQU 4DH  ; LCD EN DELAY
```

```
ADD  EQU 50H
CHI  EQU 51H
CLO  EQU 52H
X    EQU 53H  ; LCD X POS.

; USER RAM .....5FH60H-->SP
;-------------------------------------------
FKEY REG  20H.0  ; KEY PRESSED
ALF  REG  20H.1  ; ALARM FLAG

; I/O DEFINE ......................
; 93C46
CLK  REG P0.5    ; C46 CLK
DI   REG P0.6    ; C46 DI
DO   REG P0.7    ; C46 DO
CS   REG P3.4    ; ACTIVE HI 93C46
; LCD
EN   REG P3.6    ; ACTIVE HI LCD
RS   REG P0.0    ; LCD REG. SEL.

REL  REG P3.5
AL   REG P3.2
WLED REG P1.0
;--------------------
     ORG 0H
     JMP BEGIN
;---------------------
BEGIN:   MOV SP,#60H
     CALL INIT_PORT
     CALL INIT_C46

     CALL LED_BL
     CALL SET_LCD
     CALL READ_PASS ; READ PASS FROM C46
     CALL LOOK_PASS

     CALL MLOOP
     JMP $
```

```
;-----------------------------------------
MESS:  DB "PA.ASM 8051 PASSWORD WITH 93C46"
PASS0: DB 1,2,3,4
;-----------------------------------------
; DELAY Xms   R5*10 ms
DELAY:
     MOV R6,#50
$1:  MOV R7,#100
     DJNZ R7,$
     DJNZ R6,$1
     DJNZ R5,DELAY
     RET
;---------------------
LED_BL:
     MOV R4,#6
$0:  CPL WLED
     MOV R5,#3
     CALL DELAY
     DJNZ R4, $0
     RET
;-----------------------------------------------
; FIND DIG CODE 0--F
; KEY PAD:STANDARD
; 1 2 3 C
; 4 5 6 D
; 7 8 9 E
; A 0 B F
;TABLE_DIG:
;     DB  01H, 04H, 07H, 0AH
;     DB  02H, 05H, 08H, 00H
;     DB  03H, 06H, 09H, 0BH
;     DB  0CH, 0DH, 0EH, 0FH

; KEY PAD: TYPE A
; F E D C
; B 3 6 9
; A 2 5 8
; 0 1 4 7
TABLE_DIG:
```

```
      DB  0FH, 0BH, 0AH, 00H
      DB  0EH, 03H, 02H, 01H
      DB  0DH, 06H, 05H, 04H
      DB  0CH, 09H, 08H, 07H
;-------------------------------------------------------
; USE P2
; R0 : BUF PT NO USE
; R1 : COUNT 1
; R2 : COUNT 2
; R3 : SCAN SIGNAL
; R4 : COUNT KEY_NO
;-----------------
SCAN:
     MOV R3, #FEH   ; LOAD INITIAL SCAN SIGNAL
     MOV R4, #0     ; KEY NO COUNT
     MOV R1, #4     ; COUNT TIMES
     CLR  PSW.5     ; NO KEY ON
;.....................................
L1:
; SEND OUT SCAN SIGNAL VIA P2.0~P2.3
     MOV A, R3
     MOV P2,A
;.................................
; DELAY A WHILE .......
     MOV R5, #1
     CALL DELAY

; READ I/P DATA FROM P2.4~P2.7
     MOV A,P2
     ANL A,#F0H     ; GET HIGH 4 BIT  EX:1110XXXX

; CHECK KEY PRESSED ?
     MOV R2,#4 ; COUNT 2
;.................................
L2:
     JB ACC.4, N1   ; NO KEYED
     MOV KEY_NO, R4
     SETB PSW.5     ; KEY PRESS......
N1:
```

```
     INC R4         ; KEY NO ++
     RR A      ; CHECK KEY NO
     DJNZ R2, L2

; CHANGE SCAN CODE  EX:XXXX1101
     MOV A, R3
     RL A
     MOV R3, A
     DJNZ R1, L1

;.................................
; SET FKEY
BACK:
     JNB PSW.5, NO_KEY
     SETB FKEY
     RET
NO_KEY:
     CLR FKEY  ; NO KEY PRESSED.........
     RET
;-------------------------------------------------------
; SCAN KEY ON AND OFF THEN PASS KEY_NO
SCAN_KEY:
     CLR FKEY
$1:
     CALL SCAN
; IF(KEY_ON) WAIT  KEY_OFF
     JB FKEY, WAIT_OFF
; NO KEY ON RETURN....
     RET
WAIT_OFF:
     CALL SCAN
     JB FKEY, WAIT_OFF
     RET
;--------------------------------------
; WAIT KEY IN.............
; RETURN A: DIGIT NO
GET_KEY:
     CLR FKEY
     CALL SCAN  ;
```

```
; IF(KEY_ON) WAIT  KEY_OFF
     JB FKEY, $1
; NO KEY ON WAIT KEY IN.............
     JMP GET_KEY
$1:
     CALL SCAN
     JB FKEY, $1
; KEY INDEX TO DIGIT NO.
     MOV A,KEY_NO
     MOV DPTR, #TABLE_DIG
     MOVC A,@A+DPTR
     RET
;-----------------------------------------------
LOOK:
     MOV CO,#4
     MOV R0,#PASSNEW
$1:
     MOV A,@R0
     INC R0
     DJNZ CO,$1
     RET
;----------------------
CHECK_PASS:
     CALL IP_PASS
     CALL LOOK

;COMP PASS[] TO PASS0[]...........
     MOV CO,#4
     MOV R0,#PASS
     MOV DPTR, #PASS0
$1:
     MOV A, #0
     MOVC A,@A+DPTR
     MOV B, @R0
     CJNE A,B, $2
     INC DPTR
     INC R0
;.............................
     DJNZ CO,$1
```

```
     CALL OK
     MOV R4,#1
     CALL LED_BL
     CALL LED_BL
     RET
$2:
;.........................
     JMP COMP
ERROR:
     CALL ERR
     CALL LED_BL
     RET
;-------------------------
COMP:
;COMP PASS[] TO PASSNEW[]..........
     MOV CO,#4
     MOV R0,#PASS
     MOV R1,#PASSNEW
$11:
     MOV A, @R0
     MOV B, @R1
     CJNE A,B, ERROR
     INC R0
     INC R1
;.........................
     DJNZ CO,$11
     CALL OK
     RET
;-----------------------------------------
; KEY IN 4 PASS SHOW ON LCD LINE 2
IP_PASS:
     MOV DPTR,#M_IP
     MOV A,#2  ; LINE2
     CALL LCD_PRINT

     MOV B,#9   ; X POS.
     MOV CO,#4  ; GET 4 KEYS...........
     MOV R0, #PASS
$1:
```

```
     CALL GET_KEY
     MOV @R0,A  ; LOAD TO PASS[]

     ADD A,#30H ; CONVERT 0--9 TO ASC.
     CALL LCDP2
     INC B
     INC R0
     DJNZ CO, $1
     RET
;---------------------------
OK:  MOV DPTR,#M_OK
     MOV A,#2  ; LINE2
     CALL LCD_PRINT
     CALL DOOR_OPEN
     RET
;----------------------------
ERR: MOV DPTR,#M_ERR
     MOV A,#2  ; LINE2
     CALL LCD_PRINT
     CALL ALARM
     RET
;----------------------------
SLINE2:
     MOV DPTR,#LMESS2
     MOV A,#2  ; LINE2
     CALL LCD_PRINT
     RET
;---------------------------
; SET NEW PASS..........
SET_PASS:
; GET 4 KEYS...........
     MOV DPTR,#M_IP_NEW
     MOV A,#2  ; LINE2
     CALL LCD_PRINT

     MOV B,#12   ; X POS.
     MOV CO,#4  ; GET 4 KEYS...........
     MOV R0, #PASSNEW
$1:
```

```
     CALL GET_KEY
     MOV @R0,A  ; LOAD TO PASS[]

     ADD A,#30H ; CONVERT 0--9 TO ASC.
     CALL LCDP2
     INC B
     INC R0
     DJNZ CO, $1
     RET
;---------------------------------------------------------------------
M_IP:       DB "I/P PASS:....   ",0
M_OK:       DB "PASSWORD OK !!! ",0
M_ERR:      DB "PASSWORD ERROR !",0
M_IP_NEW: DB "I/P NEWPASS:....",0
;-----------------------------------------------------------
; SHOW ON LCD
LOOK_PASS:
     MOV B,#0
     MOV CO,#4
     MOV R0,#PASSNEW
$1:
     MOV A,@R0
     ADD A,#30H
     CALL LCDP2
     INC R0
     INC B
     DJNZ CO,$1

     MOV R5,#100
     CALL DELAY
     CALL SLINE2
     RET
;---------------------------------------
AL_ON:    SETB AL
     RET
;-------------------
AL_OFF: CLR AL
     RET
;-------------------
```

```
REL_ON:  SETB REL
      RET
;-------------------
REL_OFF: CLR REL
     RET
;-------------------
INIT_PORT:
     CALL AL_OFF
     CALL REL_OFF
     RET
;---------------------
DOOR_OPEN:
     CALL REL_ON
     MOV R5, #200; 2000 mS
     CALL DELAY
     CALL REL_OFF
     RET
;--------------------------------------
ALARM:
     CALL LED_BL
     CALL AL_ON
;.........................
     MOV R4,#ALSEC ;
$1:  MOV R5,#100 ; 100x10 mS=1000 mS
     CALL DELAY

     DJNZ R4,$1
     CALL AL_OFF
     RET
;------------------------------------------------------------
MLOOP:
     CALL GET_KEY
; JUDGE WHICH KEY.............
; 'A'--> SET NEW PASS.
; 'B'--> CHECK PASS.

     CJNE A,#KA, $3
; 'A'--> SET NEW PASS. ...........
     CALL SET_PASS
```

```
     CALL SLINE2
; WR. PASS TO C46
     CALL WR_PASS
     JMP MLOOP

$3:  CJNE A,#KB, $4
; 'B'--> CHECK PASS.  ..............
     CALL CHECK_PASS
     CALL SLINE2
     JMP MLOOP
$4:
     JMP MLOOP
     RET
;-----------------------------------
;  LCD I/O
;-----------------------------------
SET_LCD:
     CLR  EN ;  en=0
     CALL INIT_LCD
     MOV R5,#10
     CALL DELAY

     MOV DPTR,#LMESS1
     MOV A,#1  ; LINE1
     CALL LCD_PRINT

;    MOV DPTR,#LMESS2
;    MOV A,#2  ; LINE2
;    CALL LCD_PRINT
     RET
;--------------------------------
LMESS1: DB "PASSWORD & 93C46",0
LMESS2: DB "A OR B KEY .....",0

INIT_LCD1:
     MOV A,#28H
     CALL WCOM
     MOV A,#0CH
     CALL WCOM
```

```
     MOV A,#0EH
     CALL WCOM
     MOV A,#01H
     CALL WCOM
     RET
;----------------------------------------
; A=LINE 1 OR 2
; DPTR = MESSAGE POINTER
LCD_PRINT:
     CJNE A,#1,LINE2
LINE1:    MOV A,#80H
     CALL WCOM
     CALL CLR_LINE
     MOV A,#80H
     CALL WCOM
     JMP FILL

LINE2:    MOV A,#C0H
     CALL WCOM
     CALL CLR_LINE
     MOV A,#C0H
     CALL WCOM
FILL:
     CLR A
     MOVC A,@A+DPTR
     CJNE A,#0,$1
     RET
$1:
     CALL WDATA
     INC DPTR
     JMP FILL
     RET
;---------------------------
CLR_LINE:
     MOV R0,#24
$1:  MOV A,#' '
     CALL WDATA
     DJNZ R0,$1
     RET
```

```
;------------------------------
DE:  MOV  LCDE, #100
     DJNZ LCDE,$
     RET
;-------------------
EN1:
     SETB EN
     CALL DE
     CLR  EN
     CALL DE
     RET
;-------------------
INIT_LCD:
; software reset 3 TIMES
     MOV P0, #30H
     CALL EN1
     MOV P0, #30H
     CALL EN1
     MOV P0, #30H
     CALL EN1

     MOV P0, #20H
     CALL EN1
     CALL INIT_LCD1
     RET
;----------------------
WCOM:
; WRITE HI 4  BIT
     MOV P0, A
     CLR RS ; SET COMMAND
     CALL EN1
     RLC A
     RLC A
     RLC A
     RLC A
; WRITE LO 4  BIT
     MOV P0, A
     CLR RS ; SET COMMAND
     CALL EN1
```

```
     RET
;-------------------------------------
WDATA:
; WRITE HI 4  BIT
     MOV P0, A
     SETB RS ; SET DATA
     CALL EN1
     RLC A
     RLC A
     RLC A
     RLC A
; WRITE LO 4 BIT
     MOV P0, A
     SETB RS ; SET DATA
     CALL EN1
     RET
;-------------------------------------
; PRINT A CHAR ON LCD LINE 1
; A=ASC DATA
; B=LINE X POS.
LCDP1:
; LINE 1
     PUSH A
     MOV A,B
     ADD A,#80H
     CALL WCOM
     POP A
     CALL WDATA
     RET
;------------------------------
; PRINT A CHAR ON LCD LINE 2
; A=ASC DATA
; B=LINE X POS.
LCDP2:
; LINE 1
     PUSH A
     MOV A,B
     ADD A,#C0H
     CALL WCOM
```

```
     POP A
     CALL WDATA
     RET
;-------------------------------------
; 93C46 I/O
;-------------------------------------
DEL1:
     MOV R6,#1
$1:  MOV R7,#100
$2:  DJNZ R7,$2
     DJNZ R6,$1
     RET
;--------------------
INIT_C46:
     CLR EN     ;LCD OFF
     CLR CS     ;C46 OFF
     SETB DO
     CLR  CLK
     RET
;-------------------------
; PULSE IN CLK
PU:
     SETB CLK
     CALL DEL1
     CLR  CLK
     CALL DEL1
     RET
;------------------------
; WRITE A BYTE IN A
; BIT 7 O/P
WDV:
     MOV B,#8
$1:
;RLC A    ROL C<--B7 ....B0    C:O/P BIT
     RLC A
     JNC $2
     SETB DI
     JMP $3
$2:  CLR DI
```

```
$3:
; CLK PULSE
     CALL PU
     DJNZ B,$1
     RET
;---------------------
; READ DATA RETURN IN A REG.
RDV:
     MOV B,#8
$1:
     CALL PU
     JNB DO,$2
     SETB C    ; DO=1
     JMP $3
$2:  CLR C     ; DO=0
$3:
;RLC A    B0<-ROL C<--B7 ....B0 <--C (I/P BIT IN C)
     RLC A
     DJNZ B,$1
     RET
;----------------------
CON0:
     CALL INIT_C46
     SETB CS
     SETB DI
     CALL PU
     RET
;----------------------------------------
;USE : READ ADD. 16 BIT DATA  (CH CL)
;ADD=0~127
CON_READ:
     CALL CON0
; WRITE ADD
     MOV A, ADD
     ADD A,#80H
     CALL WDV

; READ 16 BIT
     CALL RDV
```

```
     MOV CHI, A
     CALL RDV
     MOV CLO, A
     CLR CS
     RET
;-------------------------------------------
;USE : WRITE ADD. 16 BIT DATA  (CH CL)
CON_WRITE:
     CALL CON0
; WRITE ADD
     MOV A, ADD
     ADD A,#40H
     CALL WDV
; WRITE 16 BIT
     MOV A,CHI
     CALL WDV
     MOV A,CLO
     CALL WDV
     CLR CS
     RET
;------------------------
CON_EWEN:
     CALL CON0
     MOV A,#30H
     CALL WDV
     CLR CS
     RET
;----------------------
CON_EWDS:
     CALL CON0
     MOV A,#0
     CALL WDV
     CLR CS
     RET
;---------------------
; 93c46 64x16 bit   add=0  1   ..... 63
; USE : ADD, CHI, CLO   CALL WR
WR:
     CALL CON_EWEN
```

```
     CALL CON_WRITE
     CALL CON_EWDS
     RET
;--------------------------
; WRITE PASS 4 TO C46 ADD 5,6
WR_PASS:
     MOV ADD,#5
     MOV CHI, PASSNEW
     MOV CLO, PASSNEW+1
     CALL WR
     MOV ADD,#6
     MOV CHI, PASSNEW+2
     MOV CLO, PASSNEW+3
     CALL WR
     RET
;------------------------
; READ PASS 4 FROM C46
READ_PASS:
     MOV ADD, #5
     CALL CON_READ
     MOV A,CHI
     MOV PASSNEW,A

     MOV A,CLO
     MOV PASSNEW+1,A
;
     MOV ADD, #6
     CALL CON_READ
     MOV A,CHI
     MOV PASSNEW+2,A

     MOV A,CLO
     MOV PASSNEW+3,A
     RET
;---------------------
```