第7章

中斷系統

CPU與I/O溝通方式

- 程式I/O (Programmed I/O)
- 中斷I/O (Interrupt I/O)
- 直接記憶體存取 (Direct Memory Access, DMA)

程式I/O (Programmed I/O)

- 當CPU 與 I/O要連繫時， CPU詢問或測試週邊裝置是否備妥(ready)，若尚未則CPU等待(wait)一段時間後，再向週邊裝置測試是否備妥；若備妥，則CPU執行所要I/O動作，完畢後再繼續原工作。
- 優點：完全軟體方式進行，程式簡單易寫，不需額外硬體，成本低。
- 缺點：無效率，浪費CPU時間。

中斷定義

- 中斷是指電腦在執行某一程式的過程中，由於電腦系統內、外的某種原因，而必須終止原程式的執行，轉去執行相應的處理程式，待處理結束之後，再回來繼續執行被終止的原程式過程。

中斷I/O (Interrupt I/O)

- CPU 執行原工作，若週邊裝置有需求，則發出中斷信號通知CPU ，待CPU知道後，暫停目前工作(依中斷信號種類，CPU可以不理會，請看下節說明)，對週邊發出中斷認可(INTA)，並依中斷來源種類，跳至中斷服務常式(Interrupt Service Routine，ISR)執行I/O動作，完畢後，CPU再繼續原工作。
- 優點：1.有效率，CPU執行原工作，只有週邊有需求時，才對週邊服務。　2.能做即時控制。
- 缺點：1.需額外電路來處理多週邊同時需求。
  2.程式複雜度與成本較高。

中斷處理流程

Step1: CPU 執行原工作

Step2:若週邊裝置(I/O)有需求，則對CPU發出中斷信號

Step3:待CPU知道後，暫停目前工作，對週邊發出中斷認可信號(INTA)

Step4: CPU將目前PC值壓入stack，依中斷來源種類，跳至適當中斷服務常式(Interrupt Service Routine，ISR)

Step5: CPU執行I/O動作

Step6:完畢後，從stack取出舊的PC值，繼續原來工作。

中斷處理流程示意圖

直接記憶體存取(Direct Memory Access，DMA)

- 第一與第二種方式，須藉助CPU介入彼此間連繫。所謂DMA，即允許週邊與記憶體兩者直接傳送，不必CPU介入，完全交給DMA控制器處理。
- 優點：資料傳送速度快，一般用在大量資料傳送，如磁碟機與記憶體或記憶體與記憶體之間。
- 缺點：1.需額外電路、成本高。
  2.程式規劃複雜。

中斷種類(依來源分)

- 外部中斷(external interrupt)
  - 由外界電路所產生中斷，例如鍵盤、計時器等。
- 內部中斷(internal interrupt)
  - 有時也稱程式中斷(program interrupt)，又稱陷(trap)。當程式中的指令或資料不合法或錯誤等產生，例如除以零、堆疊溢位、保護入等。
- 軟體中斷(software interrupt)
  - 本中斷來自執行程式中的一種特殊呼叫指令，例如當程式要求系統OS做某些工作，執行某些監督者呼叫(supervisor call)。如同在個人電腦上呼叫BIOS(基本輸入輸出系統)或DOS。

中斷與副程式呼叫之不同

- 來源不同
- 發生時間不同
- 處理事項不同
- 服務位址不同

中斷種類(依CPU是否需處理分)

- 不可遮罩中斷(NonMaskable Interrupt，NMI)
  - 不可藉助軟體抑制，強迫CPU一定要理的中斷。
- 可遮罩中斷(Maskable Interrupt Request，INT，IRQ)
  - 可以藉助軟體(例如8088之CLI，STI指令)，控制 CPU是否要處理中斷信號。
    註：在CPU內有一旗號暫存器，其中有一中斷旗號(Interrupt FLAG ，IF)，若IF=1，則當INT 中斷信號觸發，ＣＰＵ才會處理。CLI指令即清除IF=0，STI指令即設定IF=1。

中斷服務常式(ISR)位址決定

- 非向量式中斷(non vectored interrupt)
  - ISR寫在固定位址，例如Z-80 CPU，模式1(IM1)中斷，固定跳至0038H位址執行。8048之外界中斷，固定跳至0003H位址執行。Z-80 CPU之NMI中斷，跳至0066 H。例如8051 CPU，INT0中斷跳至0003H位址
- 向量式中斷(vectored interrupt)
  - ISR可寫在任意位址，在中斷過程中，不僅送中斷信號給CPU，並將一組辨識碼(identifier)或中斷向量傳給CPU，而CPU則藉此找到適當的中斷服務程式(ISR)。

微處理機對I/O介面定址方式

- 隔離式I/O(isolated I/O)又稱I/O對映I/O(I/O mapped I/O)。即I/O空間(I/O space)與記憶體空間(memory space)互相獨立。
  - 優點：
    - 1.I/O不佔用memory空間，memory真正可用空間較大。
    - 2.有專屬輸出入指令，如IN，OUT指令，程式容易區別I/O動作。
    - 3.一般I/O空間較小，所以I/O解碼定址較簡單，快速。
  - 缺點：
    - 1.一般CPU中必須有一支接腳，以資區別是要存取I/O，或memory(在8088中即IO/M接腳)。
    - 2.程式較無效率，在I/O埠上只能做簡單IN，OUT動作，不能處理，必須將資料讀入後，再做處理(例如，測試某一位元，與某暫存器相加)，再將結果OUT到I/O上。
- 記憶體對映I/O(memory mapped I/O)即I/O佔用memory space一部份，利用記憶體來對映I/O埠位址。
  - 優點：
    - 1.I/O位址即一記憶體位址，凡是在記憶體上可做的運算， 在I/O上也可以，所以所寫程式較有效率。
    - 2.不必M/IO區別接腳。
  - 缺點：
    - 1.無IN，OUT指令，程式中不易區別何者在執行I/O動作。
    - 2.解碼較慢
    - 3.全部可用的記憶空間減少。

SPCE061A中斷類型

- 軟體中斷 ：軟體中斷是由軟體指令break產生的中斷。軟體中斷的向量位址為FFF5H
- 異常中斷 ：異常中斷表示為非常重要的事件，一旦發生，CPU必須立即進行處理。目前SPCE061A定義的異常中斷只有‘重置Reset’一種。通常，SPCE061A系統重置可以由以下三種情況引起：上電、看門狗計數器溢位以及系統電源低於電壓低限。不論什麼情況引起重置，都會使重置腳的電位變低，進而使程式指標PC指向由一個重置向量（FFF7H）所指的系統重置程式入口位址。
- 事件中斷：事件中斷（可簡稱“中斷”，以下提到的“中斷”均為事件中斷）一般產生於片內部元件或由外部中斷輸入腳引入的某個事件。這種中斷的開通/禁止，由相應獨立使能和相應的IRQ或FIQ總致能控制。

事件中斷

- SPCE061A的事件中斷可採用兩種方式：快速中斷請求即FIQ中斷和中斷請求即IRQ中斷。這兩種中斷都有對應的致能控制。

SPCE061中斷來源

- 中斷系統有14個中斷源分為
  - 兩個計時器溢出中斷、
  - 兩個外部中斷、
  - 一個串列口中斷、
  - 一個觸鍵喚醒中斷、
  - 7個時基信號中斷、
  - 一個PWM輸出中斷。

中斷控制指令

- FIQ ON
- FIQ OFF
- IRQ ON（IRQ的總中斷允許開）
- IRQ OFF
- INT

中斷源表

中斷源	中斷優先順序	中斷向量	保留字
Fosc/1024 溢出信號 PWM INT	FIQ/IRQ0	FFF6H/ FFF8H	_FIQ/_IRQ0
TimerA 溢出信號	FIQ /IRQ1	FFF6H/ FFF9H	_FIQ/_IRQ1
TimerB 溢出信號	FIQ /IRQ2	FFF6H/ FFFAH	_FIQ/_IRQ2
外部時脈源輸入 信號 EXT2	IRQ3	FFFBH	_IRQ3
外部時脈源輸入 信號 EXT1			
觸鍵喚醒信號			
4096Hz 時基信號	IRQ4	FFFCH	_IRQ4
2048Hz 時基信號			
1024Hz 時基信號			
4Hz 時基信號	IRQ5	FFFDH	_IRQ5
2Hz 時基信號			
選頻信號 TMB1	IRQ6	FFFEH	_IRQ6
選頻信號 TMB2			
UART 傳輸中斷	IRQ7	FFFFH	_IRQ7
BREAK	軟體中斷		

中斷優先順序和中斷向量

中斷向量	中斷優先順序別
FFF7H(重置向量)	**RESET**
FFF6H	**FIQ**
FFF8H	**IRQ0**
FFF9H	**IRQ1**
FFFAH	**IRQ2**
FFFBH	**IRQ3**
FFFCH	**IRQ4**
FFFDH	**IRQ5**
FFFEH	**IRQ6**
FFFFH	**IRQ7**

相關SFR

- P_INT_Ctrl暫存器
- P_INT_Clear暫存器

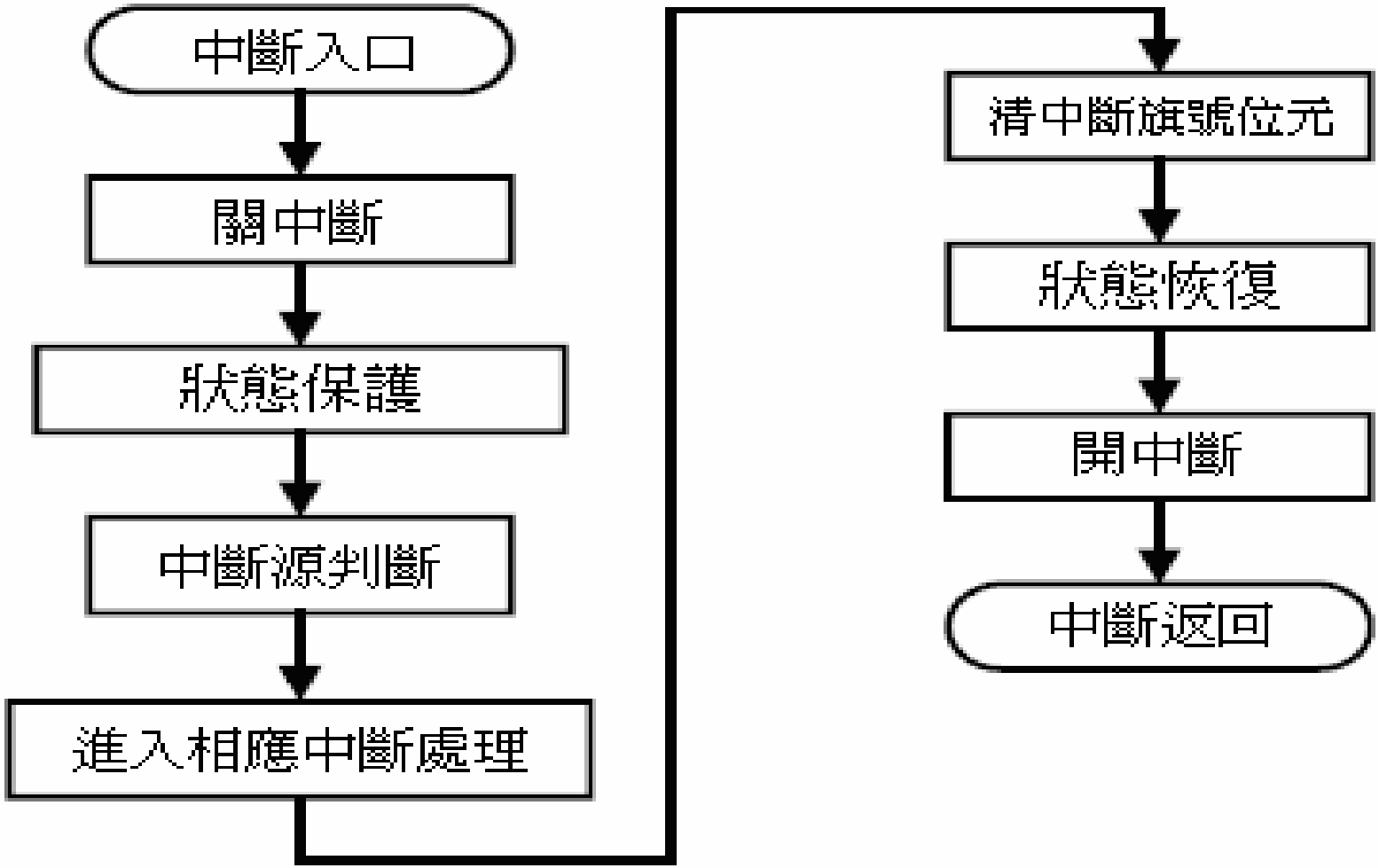
P_INT_Ctrl暫存器

b7	b6	b5	b4	b3	b2	b1	b0
IRQ3_KEY	IRQ4_4KHz	IRQ4_2KHz	IRQ4_1KHz	IRQ5_4Hz	IRQ5_2Hz	IRQ6_TMB1	IRQ6_TMB2
b15	b14	b13	b12	b11	b10	b9	b8
FIQ_Fosc/ 1024	IRQ0_Fosc/ 1024	FIQ_TMA	IRQ1_TMA	FIQ_TMB	IRQ2_TMB	IRQ3_EXT2	IRQ3_EXT1

P_INT_Clear暫存器

b7	b6	b5	b4	b3	b2	b1	b0
IRQ3_KEY	IRQ4_4KHz	IRQ4_2KHz	IRQ4_1KHz	IRQ5_4Hz	IRQ5_2Hz	IRQ6_TMB1	IRQ6_TMB2
b15	b14	b13	b12	b11	b10	b9	b8
FIQ_Fosc/ 1024	IRQ0_Fosc/ 1024	FIQ_TMA	IRQ1_TMA	FIQ_TMB	IRQ2_TMB	IRQ3_EXT2	IRQ3_EXT1

中斷服務副程式流程(ISR寫法)



中斷控制的內建常數

C_IRQ6_TMB2	Timer B IRQ6	(b0)
C_IRQ6_TMB1	Timer A IRQ6	(b1)
C_IRQ5_2Hz	IRQ5 2 Hz	(b2)
C_IRQ5_4Hz	IRQ5 4 Hz	(b3)
C_IRQ4_1KHz	1024Hz IRQ4	(b4)
C_IRQ4_2KHz	2048Hz IRQ4	(b5)
C_IRQ4_4KHz	4096Hz IRQ4	(b6)
C_IRQ3_KEY	Key Change IRQ3	(b7)
C_IRQ3_EXT1	Ext1 IRQ3	(b8)
C_IRQ3_EXT2	Ext2 IRQ3	(b9)
C_IRQ2_TMB	Timer B IRQ2	(b10)
C_FIQ_TMB	Timer B FIQ	(b11)
C_IRQ1_TMA	Timer A IRQ1	(b12)
C_FIQ_TMA	Timer A FIQ	(b13)
C_IRQ0_PWM	PWM IRQ0	(b14)
C_FIQ_PWM	PWM FIQ	(b15)

規劃允許2Hz時基中斷

```
asm("INT OFF");          // 關閉所有中斷
*P_INT_Ctrl = C_IRQ5_2Hz;
   //允許2 Hz 中斷
asm("INT IRQ");     //允許所有IRQ中斷
```

規劃允許外部中斷1

```
asm("INT OFF");      // 關閉所有中斷
Init_B_Port();       //規劃B Port bit
   2為具上拉電阻的輸入
*P_INT_Ctrl = C_IRQ3_EXT1;
  //允許 EXT1 中斷
 asm("INT IRQ");  //允許所有IRQ中斷
```

規劃允許外部中斷1與2

```
asm("INT OFF");          // 關閉所有中斷
*P_INT_Ctrl = C_IRQ3_EXT1 | C_IRQ3_EXT2;
   //允許 EXT1及EXT2 中斷
asm("INT IRQ");   //允許所有IRQ中斷
```

規劃計時器A定時2秒中斷

```
asm("INT OFF"); // 關閉所有中斷
 *P_TimerA_Ctrl = C_SourceA_8192Hz+C_SourceB_1;

   //TimerA:8192Hz
*P_TimerA_Data =  0xffff -0x4000 ;

                //2sec時間常數
*P_INT_Ctrl=C_IRQ1_TMA; //允許
   TimerA_IRQ1中斷
 __asm("INT IRQ");  //允許所有IRQ
   中斷
```

中斷副程式

```
void IRQ5(void) __attribute__ ((ISR));
 void IRQ5(void)
 {
     if ( *P_INT_Ctrl & 0x0004 )
    {       //IRQ5_2Hz
         *P_IOA_Data = data;
         data ^= 0xffff;
         *P_INT_Clear = 0x0004;      // 清除中斷旗號
     }
     else
     {       //IRQ5_4Hz
         *P_INT_Clear=0x0008;      // 清除中斷旗號
    }
 }
```

範例

- 7-1 2Hz時基中斷
- 7-2 外部1中斷
- 7-3 外部1及外部2中斷
- 7-4 外部1及外部2中斷與上下數
- 7-5 觸鍵喚醒中斷
- 7-6 TimerA—IRQ1中斷
- 7-7 多來源—相同進入點
- 7-8 多來源—不同進入點