表格內容

專題八

電子計算機(中斷式 Interrupt)

實驗目的

輪詢法與中斷法分別是 CPU 偵知週邊發生事件的兩種方式，由於中斷法是由週邊主動通知 CPU，因此比較不會發生漏掉事件偵測的情況。本實驗重覆使用鍵盤及七節管 ASA 介面卡，利用前一單元的功能方塊函式，以中斷服務函式寫作方式再實作出四則運算電子計算機。

預習讀物

1. 網頁維基百科關於 C 言語
2. http://en.wikipedia.org/wiki/C(programming_language)
3. C 語言學習筆記網頁 http://openhome.cc/Gossip/CGossip/
4. ATmega128(L) Complete.pdf

儀器材料

1. PC 電腦
2. ASA M128 單板電腦 www.viewmove.com
3. WinAVR Programmer's Notepad 程式下載軟體
4. Windows 內建附屬應用程式超級終機
5. ASA-7S00 七節管模組 ×1
6. ASA-KB00 4X4 鍵盤模組 ×1
7. ASA-排線 X1
8. 單位元跳線 X1

背景知識

中斷的意義：

中斷的意義，與日常生活的認知並無太大的差距。也就是在做一件事情的同時發

生了另一件更緊急的事需要先處理，於是暫停原來處理中的工作，轉而處理新的緊急工作，待處理完這件新的緊急工作後，再處理原工作的整個過程，即所謂的中斷。例如說老師在改做業，學生敲門來問問題，就是一件中斷事件。

與中斷相關的工作：

分析中斷的處理過程，可以發覺要處理中斷有許多相關的事情要注意。首先就是要被中斷的人是不是正在處理一些不能被中斷的工作，例如說在老師在出題時，是不會也不應被學生中斷，這叫做中斷致能與否(Interrupt enable，disable)。其次雖然在出題時不方便為學生中斷，但是一同出題的老師，卻可能必須要能中斷，也就是說中斷有不同的中斷源(Interrupt source)，而不同的中斷源又能夠有不同的中斷優先順序。(Interrupt priority)

再來就是，當學生來問問題時，老師為了清出桌面讓學生能夠在桌面上推導問題，必需把桌面上原先批改的作業堆在一邊，而在完成問答之後再將作業重新還原。這動作，稱為推入堆疊，拉出堆疊(Push，Pop，Stack)。

上述例子，老師很難分辨敲門的是學生還是老師要中斷，若是老師對每一種中斷設一道門，門口並設有電鈴，而且區分老師與學生有不同的門及電鈴，並可以設定那一個電鈴會響。則不但可以表現出中斷的優先順序以及致能的觀念，同時更可以表現中斷向量表(Interrupt vector)的觀念。我們可以將不同的門及門鈴對應不同的中斷向量，只有致能的中斷會有中斷通知(門鈴聲)，而在得知中斷時(聽到門鈴時)，會依不同中斷源(不同門鈴聲)，而到不同的中斷向量表讀取中斷服務常式 Interrupt service routing(到不同門口開門聽取要求)。如果中斷服務常式太長(有太多事情要談)則跳到較大的記憶體位置(一起找個地方座下來談)。

綜合上述，要完成一個中斷工作，必須有以下兩類動作：

- 設定中斷方式的相關動作：這些動作應在進入中斷程式前完成
    - 設定中斷優先權：定義中斷之優先，可避免較不急的事件干擾較急的事件之處理，而同時又允許較急的事件再中斷急的事件。這個設定如不做時，會依原訂優先順序安排，故有時會省略。尤其是現代的微控器速度較以前快許多，且中斷中執行工作的觀念愈來愈明確，在中斷中再巢狀中斷愈來愈沒必要，因此中斷優先權，已經愈來愈多是單純以先中斷先處理為標準優先順序，不再依中斷源分優先。
    - 設定中斷觸發方式：設定中斷源以何種方式通知有事件發生，可以是邊緣觸發或則是準位觸發。

- 設定中斷致能：開放中斷源的訊息可以到達 CPU，請求中斷。
- 設定中斷源輸出入：現在微控器為省接腳，外部中斷源會與其它輸出入功能共用接腳，要使用它們做為中斷訊號源需設定它們為輸入，允計訊號可以進入微控器內部電路。

- 提供中斷服務內容的相關動作：這些動作應在中斷服務常式中完成
  - 提供中斷服務常式：提供電腦在中斷中處理事件的程序。
  - 提供中斷服務常式與中斷向量對應：讓中斷處理程式與中斷源之間產生關聯，中斷發生時才有辦法找到中斷處理程式所在。
  - 中斷服務前處理：在正式開始處理中斷服務常式前，先將原處理中之相關資料推入堆疊中，以備將來回復中斷前狀態之用。
  - 中斷服務後處理：在完成中斷服務常式之後，將原處理中之相關資料由堆疊中拉回，回復未中斷前之狀態。

中斷源硬體電路：

觸發方式

中斷是外部電路主動告知電腦有一個事件發生需要中斷處理，而告知的方式是外部電路提供一個準位的變動，而造成電路準位變化的元件有兩類，一類是指撥開關方式，一類則是按鍵。前者指撥後會使得準位變化，而按鍵開關則可偵測其準位的上昇或下降緣，每按一次算有一個下降緣，一個上昇緣，多次輸入可得多個按鍵輸入。中斷觸發配合按鍵及指撥開關的不同特性，而有邊緣觸發方式及準位觸發方式兩種。準位中斷觸發電路，只要用指撥開關即可實現。

彈跳現像及反彈跳

理論上按鍵開關能直接實現邊緣觸發訊號的提供電路，然而在由於 CPU 對於上下緣偵測很敏銳快速，每按一次按鍵即產生多次 ON/OFF，這種現像被稱為彈跳現像(BOUNSING)，而產生多次中斷。由於輪詢法，偵測上下緣是採取掃瞄，並比對前後兩次掃瞄的準位不同來決定是否有上下緣，彈跳現像因為是在極短時間內多次 ON/OFF，兩次輪詢之間，老早就穩定下來，所以並不會影響。但以硬體偵測輸入的中斷上下緣偵測方式則不同，因為偵測迅速，有可能會在按鍵被按一次的就偵測到好多個上昇緣及下降緣，為此用於中斷邊緣觸發用的按鍵電路，我們有時需要加上反彈跳機制(DeBounsing)。常用的反彈跳電路，包括有正反器(即暫存器的單一 BIT)，以及觸發正反器取樣的振盪電路。利用正反器會

由取樣時脈造成同步的作用，去除不穩定的彈跳訊號只輸出穩定後的準位，只留一個上昇緣，一個下降緣。

數位通訊中斷

本單元的中斷，是兩個經由串列通訊埠聯動的電腦以主僕關係互動。主電腦可以主動去輪詢僕電腦，僕電腦大部份狀況下是被動接受主電腦的輪詢監控。但當有特殊事件發生時，若仍維持等待主電腦來詢，可能會延誤了處理時機。這時便需要用到中斷機制，由僕電腦發出中斷需求，主電腦的硬體中斷電路偵測到便能馬上處理。

ASA BUS 介面卡中斷

最新版本之 ASA BUS 的 D1 訊號線，兼有 ASA BUS 介面卡共用外部中斷之功能，只要 ASA BUS 上插有需要能夠中斷 M128 單板電腦能力的介面卡時，就要從 ASA BUS 接頭跳線到 EXT_INT4~INT7 中任一外部中斷。各具有中斷功能之 ASA BUS 介面，會以 WIRE AND 的方式串接在一起，任一介面卡有中斷需求，將此接腳拉到低準位時，D1 訊號線就會是低準位，並經跳線送到外部中斷接腳，產生一個低準位中斷訊號。ASA-M128 單板電腦，可以設定為低準位觸發中斷模式，一旦有中斷訊號觸發中斷服務函式時，就要輪詢 ASA BUS 上所有具有中斷能力之介面卡。以進一步確認誰送進中斷需求，並處理它。

直接讀寫暫存器方式設定及執行中斷服務

執行中斷相關設定

若沒有，或則不使用前述外部中斷驅動函式來執行中斷模式，中斷禁致能設定，則必需要使用硬體暫存器直接寫入的方式來設定。AVR 與其它的電腦一樣，也遵循相同的方式，提供中斷功能。在本課程實驗中，我們使用了 AVR GNU C 語言的語法。以下我們也按各個動作的施行方式來說明如何利用 C 來完成中斷的設定及程式安排。

在 AVR C 中要設定中斷方式，只要以寫入指令，將所要設定之適當數值寫入相應之暫存器中即可。

以程式為例來說明會較容易

```
main()
{
    /* 宣告區塊 */
    int i;
    ...
    /* 設定區塊 */
    /*AVR 中斷優先固定不變無法設定*/
    EICRA=0x??;  /*設定外部中斷 3:0 偵知方式，非外部中斷不需要，*/
    EICRB=0x??;   /*設定中斷 7:4 偵知方式*/
    DDRD=0x??; /*設定中斷輸出入源方向*/
    EIMSK=0x??;  /*設定中斷致能*/

    :
    sei();          /*設定所有中斷均致能*/
    …
    /*正式程式區塊*/
    …
}
```

- 設定中斷優先順序：AVR 中斷向量數量很多，各中斷優先權固定沒有設定的空間。由於現代的微控器運算速度都很快，若在中斷中只執行有限輸出入動作，則優先權的影響有限。只要在中斷服務常式中，不要執行太長的程式，應該不必優先權區隔，也不會出問題。
- 設定中斷觸發方式：AVR 有兩個中斷感測方式設定暫存器，EICRA 及 EICRB。分別管理外部中斷 3:0，以及 7:4 的觸發方式。其中 3:0 能夠設定的觸發方式有三種，分別為準位觸發，上緣觸發，以及下緣觸發。7:4 能夠設定的觸發方式，除了前述三種以外，又多了任意上下緣觸發。每一個中斷，佔用設定暫存器 2 個旗標，分別為 ISCn1,ISCn0 (Interrupt sensing control)，其中 n 代表中斷編號。下圖為各旗標佔用暫存器的位置。
- AVR 外部中斷接腳及其輸出入方向設定：ATMEGA128 的 8 個外部中斷接腳，會佔用 PORT D，PORT E 的部份接腳，做為中斷訊號輸入腳 。PORT D 的 PD3:0 四支腳與 INT3:0 四個中斷分享。PORT E 的 PE7:4 四支腳與 INT7:4 四個中斷分享。由於中斷訊號是輸入訊號，因此要設定所選用的中斷源，其 DDRXn 為輸入。中斷訊號才能夠進入 AVR。

- 設定中斷致能：AVR 的中斷遮罩暫存器名稱為 EIMSK。External Interrupt Mask。八個位元每一個對應一個外部中斷。當中斷總體致能旗標，在 Status Register SREG 的 I bit。設成 1，且主管對特定中斷的旗標也設為 1 時，即致能該特定中斷。

Bit	7	6	5	4	3	2	1	0	
	ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
	ISC71	ISC70	ISC61	ISC60	ISC51	ISC50	ISC41	ISC40	EICRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

下表為 ISC 旗標值設定值對應的觸發方式

Table 48. Interrupt Sense Control(1)

ISCn1	ISCn0	Description
0	0	The low level of INTn generates an interrupt request.
0	1	Reserved
1	0	The falling edge of INTn generates asynchronously an interrupt request.
1	1	The rising edge of INTn generates asynchronously an interrupt request.

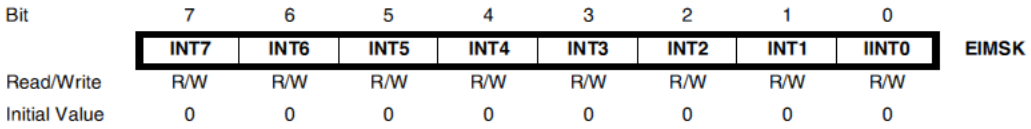
Note: 1. n = 3, 2, 1or 0.
When changing the ISCn1/ISCn0 bits, the interrupt must be disabled by clearing its Interrupt Enable bit in the EIMSK Register. Otherwise an interrupt can occur when the bits are changed.

Table 50. Interrupt Sense Control(1)

ISCn1	ISCn0	Description
0	0	The low level of INTn generates an interrupt request.
0	1	Any logical change on INTn generates an interrupt request
1	0	The falling edge between two samples of INTn generates an interrupt request.
1	1	The rising edge between two samples of INTn generates an interrupt request.

Note: 1. n = 7, 6, 5 or 4.
When changing the ISCn1/ISCn0 bits, the interrupt must be disabled by clearing its Interrupt Enable bit in the EIMSK Register. Otherwise an interrupt can occur when the bits are changed.

EIMSK 內部各 BITS 圖如下



Bit	7	6	5	4	3	2	1	0	
	INT7	INT6	INT5	INT4	INT3	INT2	INT1	IINT0	EIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

AVR C 提供了函式 `sei()`，`cli()` 可以執行整體中斷禁致能工作。

GCC 官方方式之中斷服務撰寫及鏈結

GCC 官方有自己一套，撰寫並將函式連結到中斷的方式。中斷函式登錄驅動函式內部要用官方使用的方式來實現中斷與中斷服務函式連結。

有關於這一部份，以例子說明更為清楚：

```
#include <avr/interrupt.h>

ISR(INT4_vect)

{

        // user code here

}
```

- 提供中斷服務常式：上述的類似函式提供的即為中斷服務常式內容，AVR 所有中斷，都用相同的一個名稱，只用傳參來分辨中斷源。
- 提供中斷服務常式與中斷向量對應：上述的類似函式括號內的傳參，即為對應到中斷向量的名稱。 ATmega128 有 35 個中斷源，提供各週邊硬體主動觸發 MCU 處理程序其中斷向量表如下表

Vector No.	Program Address(2)	Source	Interrupt Definition
1	$0000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$0002	INT0	External Interrupt Request 0
3	$0004	INT1	External Interrupt Request 1
4	$0006	INT2	External Interrupt Request 2
5	$0008	INT3	External Interrupt Request 3
6	$000A	INT4	External Interrupt Request 4
7	$000C	INT5	External Interrupt Request 5
8	$000E	INT6	External Interrupt Request 6
9	$0010	INT7	External Interrupt Request 7
10	$0012	TIMER2 COMP	Timer/Counter2 Compare Match

表 1 ATmega128 的第 1 到 10 中斷向量

11	$0014	TIMER2 OVF	Timer/Counter2 Overflow
12	$0016	TIMER1 CAPT	Timer/Counter1 Capture Event
13	$0018	TIMER1 COMPA	Timer/Counter1 Compare Match A
14	$001A	TIMER1 COMPB	Timer/Counter1 Compare Match B
15	$001C	TIMER1 OVF	Timer/Counter1 Overflow
16	$001E	TIMER0 COMP	Timer/Counter0 Compare Match
17	$0020	TIMER0 OVF	Timer/Counter0 Overflow
18	$0022	SPI, STC	SPI Serial Transfer Complete
19	$0024	USART0, RX	USART0, Rx Complete
20	$0026	USART0, UDRE	USART0 Data Register Empty

表 1(續) Atmega128 的第 11 到 20 中斷向量

21	$0028	USART0, TX	USART0, Tx Complete
22	$002A	ADC	ADC Conversion Complete
23	$002C	EE READY	EEPROM Ready
24	$002E	ANALOG COMP	Analog Comparator
25	$0030(3)	TIMER1 COMPC	Timer/Countre1 Compare Match C
26	$0032(3)	TIMER3 CAPT	Timer/Counter3 Capture Event
27	$0034(3)	TIMER3 COMPA	Timer/Counter3 Compare Match A
28	$0036(3)	TIMER3 COMPB	Timer/Counter3 Compare Match B
29	$0038(3)	TIMER3 COMPC	Timer/Counter3 Compare Match C
30	$003A(3)	TIMER3 OVF	Timer/Counter3 Overflow

表 1(續) Atmega128 的第 21 到 30 中斷向量

31	$003C(3)	USART1, RX	USART1, Rx Complete
32	$003E(3)	USART1, UDRE	USART1 Data Register Empty
33	$0040(3)	USART1, TX	USART1, Tx Complete
34	$0042(3)	TWI	Two-wire Serial Interface
35	$0044(3)	SPM READY	Store Program Memory Ready

表 1(續) Atmega128 的第 31 到 35 中斷向量

表中 Source 為訊號源名稱，GCC 將訊號源名稱中的空白及逗點等以下引線'_'取代，再於名稱最後接'_vect'，作為 C 程式中的訊號名稱。例如 INT4 的變數名稱為 `INT4_vect`， 而 TIMER2 OVF 的變數名稱為 `TIMER2_OVF_vect`， `USART0,RX` 變數名稱為 `USART0_RX_vect`。

- 中斷服務前處理及後處理：在中斷前後，所必需推入及拉出堆疊的執行狀態暫存器 C Compiler 都會主動幫助你處理。

以呼叫驅動函式方式設定及執行中斷服務

執行中斷相關設定

ASA M128 函式庫內，有外部中斷 EXT_INT，驅動函式 ASA_EXT_set()可供呼叫設定中斷相關的模式及禁致能。同一函式，可以用來禁致能中斷，設定中斷觸發模式，以及設定開發觸發源接腳為訊號輸入等。針對不同的外部中斷，以及不同的呼叫目的，函式內會提供了傳參，LSByte, Mask, Shift, Data 應該選用的值，並在最後整理成快查表。依據前述中斷前設定項目，依據使用到的外部中斷以及要設定的項目，從快查表中找出傳參，以傳參呼叫 ASA_EXT_set()，逐一設定完成各項中斷前述應該設定及禁致能的動作，即準備好可以接受外部中斷。

```
char M128_EXT_set(char LSByte, char Mask, char Shift,
char Data);
```

簡介：呼叫本函式可以執行外部中斷設定工作，可以設定工作包括以下：

1. 禁致能外部中斷 `INT0~INT7：200(EIMSK)`

2. 選擇外部中斷 INT0~INT7 觸發模式:設定選擇上緣，下緣，雙緣或準位觸發 201(EICRA),202(EICRB)(ASA M128 未支援 201)
3. 允許中斷源輸入設定 INT0~INT7：設定觸發源腳為輸入 。204(DDRD) (ASA M128 未支援)205(DDRE)

這些設定旗標占用軟體暫存器及所對應 AVR 硬體暫存器如下列：
200(EIMSK),201(EICRA),202(EICRB),204(DDRD),205(DDRE)

參數的選擇，請參考後面**禁致中斷傳參表。**

回應 RETURN：是否成功設定，回傳 0 時代表成功，其餘代表失敗

0：成功

1：輸入參數所選之設定本系統不支援。
2：警告，輸出入方向設定，已變更先前已有之其他設定驅動函式之方向設定，可能造成其他硬體驅動異常。

禁致能中斷傳參表

		LSByte	Mask	Shift	Data
INT0	禁致能中斷	200	0x01	0	禁能：0 致能：1
	中斷源觸發模式	201	0x03	0	低準：0 下緣：2 上緣：3
	允許中斷源輸入設定	204	0x01	0	允許：0 不允許：1
INT1	禁致能中斷	200	0x02	1	禁能：0 致能：1
	中斷源觸發模式	201	0x0C	2	低準：0 下緣：2 上緣：3
	允許中斷源輸入設定	204	0x02	1	允許：0 不允許：1
INT2	禁致能中斷	200	0x04	2	禁能：0 致能：1
	中斷源觸發模式	201	0x30	4	低準：0 下緣：2 上緣：3
	允許中斷源輸入設定	204	0x04	2	允許：0 不允許：1

INT3	禁致能中斷	200	0x08	3	禁能：0 致能：1
	中斷源觸發模式	201	0xC0	6	低準：0 下緣：2 上緣：3
	允許中斷源輸入設定	204	0x08	3	允許：0 不允許：1
INT4	禁致能中斷	200	0x10	4	禁能：0 致能：1
	中斷源觸發模式	202	0x03	0	低準：0 上下緣：1 下緣：2 上緣：3
	允許中斷源輸入設定	205	0x10	4	允許：0 不允許：1
INT5	禁致能中斷	200	0x20	5	禁能：0 致能：1
	中斷源觸發模式	202	0x0C	2	低準：0 上下緣：1 下緣：2 上緣：3
	允許中斷源輸入設定	205	0x20	5	允許：0 不允許：1
INT6	禁致能中斷	200	0x40	6	禁能：0 致能：1
	中斷源觸發模式	202	0x30	4	低準：0 上下緣：1 下緣：2 上緣：3
	允許中斷源輸入設定	205	0x40	6	允許：0 不允許：1
INT7	禁致能中斷	200	0x80	7	禁能：0 致能：1
	中斷源觸發模式	202	0xC0	6	低準：0 上下緣：1 下緣：2 上緣：3
	允許中斷源輸入設定	205	0x80	7	允許：0 不允許：1

中斷服務內容撰寫及鏈結

中斷服務函式的寫作，以及宣告方式，由於 ANSI C 並無規定，因此各家 C 語言編釋器要如何處理，有很大的自由度。例如，KEIL C 與 GCC 的方式便有很大的不同。經由用硬體功能驅動函式的包裝，便能夠把各家不同的處理方法差異包在驅動函式內，讓應用程式可以跨不同微控制器，不同 IDE 上使用。

ASA-M128 函式庫內有，外部中斷 EXT_INT 登錄函式 ASA_EXT_isr()，可以用來將無傳參變數，無回傳變數的函式，登錄成為外部中斷的服務函式，ISR。使中斷能夠觸發行所登錄的函式。

```
char M128_EXT_isr(char Number, void (*function)(void));
```

簡介：呼叫本函式，可以將無傳參函式，登記成為編號指定的外部中斷之 ISR。並在往後中斷時執行。

傳值變數:

Number：EXT 外部中斷編號，ATMega128 有八個外部中斷，編號 7:0 分別代表 INT7:0。M128 單板則只接出 INT4~INT7 四個外部中斷。

1. 使用 INT0 外部中斷函式：Number=0(ASA M128 不支援)
2. 使用 INT1 外部中斷函式：Number=1(ASA M128 不支援)
3. 使用 INT2 外部中斷函式：Number=2(ASA M128 不支援)
4. 使用 INT3 外部中斷函式：Number=3(ASA M128 不支援)
5. 使用 INT4 外部中斷函式：Number=4
6. 使用 INT5 外部中斷函式：Number=5
7. 使用 INT6 外部中斷函式：Number=6
8. 使用 INT7 外部中斷函式：Number=7

傳址變數:

(*function)(void))：先宣告定義出一個無傳參無回應的函式，再取用這個函式的指標住址，做為傳址變數，以供中斷時找到住址並執行。

例如：

```
//宣告定義無傳參無回應的 EXTfunc()函式
```

```
void EXTfunc(void){
:
}
:

int main(){
:
//去除函式之()的名稱如 EXTfunc 即其指標住址，以下登錄函式呼叫，將它登錄為 INT0 的 ISR。
M128_EXT_isr(0,EXTfunc);
    :
}
```

回應 RETURN：是否成功設定，回傳 0 時代表成功，其餘代表失敗

0：成功

1： 輸入參數所選之設定本系統不支援。

輪詢方式處理中斷事件

中斷事件出現時，如果設定了中斷致能，並提供了中斷服務函式，則硬體會自動執行中斷事件後該做的事，這方式對中斷事件的處理最快最即時。

然而有些應用，只要中斷事件發生後，軟體會去處理這個事件所引發該做的事就好，並不在意要即時快速處理時。這類的應用，可以使用輪詢中斷旗標的方，來判斷是否該處理中斷事件後續事宜。ASA-M128 單板電腦函式庫中，有讀取中斷旗標的旗標取得函式 M128_EXT_fgt()

ASA M128 EXT_INT 旗標輸入函式 ASA_EXT_fgt()

```
char M128_EXT_fgt(char LSByte, char Mask,  char Shift,
void *Data_p);
```

簡介：呼叫本函式可讀取 EXT 外部中斷旗標，以了解是否有 EXT 外部中斷。

1. 讀取 INT0~INT7 中斷旗標：203(EIFR)bit0~bit7

傳值變數：

LSByte：讀取 INT0~INT7 中斷旗標：分別涉及 203(EIFR)bit0~bit7 故 LSByte=203

Mask：讀取 INT0~INT7 中斷旗標：分別涉及 203(EIFR)bit0~bit7 故

INT0：Mask=b0000 0001=0x01,

INT1：Mask=b0000 0010=0x02,

INT2：Mask=b0000 0100=0x04

INT3：Mask=b0000 1000=0x08

INT4：Mask=b0001 0000=0x10

INT5：Mask=b0010 0000=0x20

INT6：Mask=b0100 0000=0x40

INT7：Mask=b1000 0000=0x80

Shift：讀取 INT0~INT7 中斷旗標：分別涉及 203(EIFR)bit0~bit7 故

INT0：Shift=0

INT1：Shift=1

INT2：Shift=2

INT3：Shift=3

INT4：Shift=4

INT5：Shift=5

INT6：Shift=6

INT7：Shift=7

傳址變數：

Data_p：應用程式先宣告定義一個變數，並提供變數指標位址做為傳參，以備存入遮罩及平移處理後之旗標值。

```
char EXTin;//宣告並定義變數
:
```

M128_EXT_fgt(..,&EXTin);//呼叫旗標讀取設置

//收到資料若為 0 代表中斷旗標為立起，外部中斷未觸發

//1 代表中斷旗標立起，外部中斷已觸發

回應 **RETURN**：是否成功設定，回傳 0 時代表成功，其餘代表失敗

0：成功

1：錯誤：輸入參數所選之設定本系統不支援。

M128_EXT_fgt()中斷旗標取得傳參表

	LSByte	Mask	Shift	Data_p 回傳值
INT0	203	0x01	0	0：無中斷, 1：有中斷
INT1	203	0x02	1	0：無中斷, 1：有中斷
INT2	203	0x04	2	0：無中斷, 1：有中斷
INT3	203	0x08	3	0：無中斷, 1：有中斷
INT4	203	0x10	4	0：無中斷, 1：有中斷
INT5	203	0x20	5	0：無中斷, 1：有中斷
INT6	203	0x40	6	0：無中斷, 1：有中斷
INT7	203	0x80	7	0：無中斷, 1：有中斷

輪詢方式處理中斷方式，硬體不會自動清除中斷旗標，而必需要在處理完成之後以軟體清除。M128 函式庫提供了 M128_EXT_put()函式可以呼叫用於清除中斷旗標。

ASA M128 EXT_INT 旗標輸入函式 ASA_EXT_fpt()

```
char M128_EXT_fpt(char LSByte, char Mask,  char
Shift,  char Data);
```

簡介：在輪詢處理完中斷後,呼叫本函式可用來清除 EXT 外部中斷旗標，以備後續偵測新中斷。

2. 清除 INT0~INT7 中斷旗標：203(EIFR)bit0~bit7

傳值變數：

LSByte：清除 INT0~INT7 中斷旗標：分別涉及 203(EIFR)bit0~bit7 故 LSByte=203

Mask：清除 INT0~INT7 中斷旗標：分別涉及 203(EIFR)bit0~bit7 故

INT0：Mask=b0000 0001=0x01,

INT1：Mask=b0000 0010=0x02,

INT2：Mask=b0000 0100=0x04

INT3：Mask=b0000 1000=0x08

INT4：Mask=b0001 0000=0x10

INT5：Mask=b0010 0000=0x20

INT6：Mask=b0100 0000=0x40

INT7：Mask=b1000 0000=0x80

Shift：清除 INT0~INT7 中斷旗標：分別涉及 203(EIFR)bit0~bit7 故

INT0：Shift=0

INT1：Shift=1

INT2：Shift=2

INT3：Shift=3

INT4：Shift=4

INT5：Shift=5

INT6：Shift=6

INT7：Shift=7

傳址變數：

Data：不論是那一個中斷，呼叫本函式以清除中斷旗標，

Data=1。

回應 **RETURN**：是否成功設定，回傳 0 時代表成功，其餘代表失敗

0：成功

1：錯誤：輸入參數所選之設定本系統不支援。

M128_EXT_fpt()中斷旗標清除傳參表

	LSByte	Mask	Shift	Data
INT0	203	0x01	0	1
INT1	203	0x02	1	1
INT2	203	0x04	2	1
INT3	203	0x08	3	1
INT4	203	0x10	4	1
INT5	203	0x20	5	1
INT6	203	0x40	6	1
INT7	203	0x80	7	1

實驗步驟/程式

本次實驗，將以外部中斷，讀取鍵盤按鍵的方式來完成計算機，借以學習外部中斷服務函式的撰寫。

1. 硬體配線：新版 ASA-KB00 鍵盤，(如為舊版，需更新軟體，並以細銀線跳線，增加 DO)，由 DIO-1 線可由 ASA 介面卡送低準位中斷需求訊號。此一訊號線需由連接到 40 PIN IO 的

INT4(與 PE4 共腳)，做為 INT4 的外部中斷訊號。

- 以 ASA BUS 灰排串接 M128 單板電腦，ASA-KB00，ASA-7S00 介面卡。
- 以 USB 線連接 M128 單板電腦與 PC。
- 在 ASA BUS 灰排上，詳見專題 5 的灰排實體照片 ，找一個空的母座接頭，在母座接頭上找到 DIO1 接孔，插上單 PIN 針。以跳線，一端接單 PIN 針，另一端接 40 PIN IO 的 INT4(與 PE4 共腳)。

2. 輪詢中斷旗標式中斷處理
   - 主程式：
     - 呼叫 M128_EXT_set()設定 INT4 中斷觸發模式為準位觸發。
     - 呼叫 M128_EXT_set()設定 INT4 中斷源為輸入。
     - 不致能 EXT 中斷(不致能 INT4)
     - 輪詢 while()迴路起始
       - 呼叫 M128_EXT_fgt()取得 INT4 中斷旗標
       - 呼叫 ASA_KB00 讀取鍵入值
       - 執行前單元計算機程式
       - 呼叫 M128_EXT_fpt()清除 INT4 中斷旗標。(準位觸發可能不用做)
     - 輪詢 while()迴路結束
   - 無外部中斷函式

3. 中斷服務函式登錄式中斷處理
   - 無傳參無回傳副函式：
     - 執行前單元計算機程式
   - 主程式：
     - 呼叫 `M128_EXT_set()` 設定 INT4 中斷觸發模式為準位

觸發。

- 呼叫 M128_EXT_set()設定 INT4 中斷源為輸入。
- 呼叫 M128_EXT_isr()登錄 INT4 中斷觸發前述已定義無傳參無回傳副函式。
- 呼叫 M128_EXT_set()，致能 INT4 中斷
- 輪詢 while()迴路起始
  - 其他應用程式碼。
- 輪詢 while()迴路結束

4. 直接寫入暫存器式中斷處理(不計分，有興趣者自行練)
   - 撰寫 INT4 中斷服務函式 ISR(INT4_vect)：
     - 執行前單元計算機程式
   - 主程式：
     - 直接寫暫存器設定 INT4 中斷觸發模式為準位觸發。
     - 直接寫暫存器設定 INT4 中斷源為輸入。
     - 直接寫暫存器致能 INT4 中斷
     - 輪詢 while()迴路起始
       - 其他應用程式碼。
     - 輪詢 while()迴路結束

軟硬體介面規格(略)

提示

實驗報告內容

工作日誌

程式流程/虛擬碼/內部變數資料結構/程式列表(縮印)

請提供本次實驗的程式虛擬碼，以及程式列表。

實驗數據

實測四則運算之等式。確認程式正確。

問題與討論

1. 在本次實驗中你學到了什麼？
2. 若改變 INT4 為 INT5 要有那些變動?
3. 為何中斷服務常式一開始就要禁能中斷，若不如此會發生什麼狀況?
4. 請問你，是否可能不將掃描程式寫在中斷中? 如果可以應如何寫，有何優缺點?

驗收標準

助教依以下步驟要求學生確實執行及修改，以確認學生能夠獨立完成計算器之程式：

1. 學生從關機狀態開始，由開機到燒錄已完成計算器確認本實驗之程式已完成。
2. 助教接手學生開發系統，隨機刪除，修改或對調程式順序後重新編譯出不可執行之錯誤程式，交還學生，在指定限時 10 分鐘之內，獨立不參考其它現成程式，除錯恢復原程式。
3. 任意指定 INT4~INT7 任一個，要求學生改變配線並變更程式讓最系統能完成完全相同之動作。(如果有困難做不出來，學生應該要能夠講出為什麼)

附錄：

```
char M128_EXT_set(char LSByte, char Mask, char Shift, char Data);
```

簡介：呼叫本函式可以執行外部中斷設定工作，可以設定工作包括以下：

4. 禁致能外部中斷 INT0~INT7：200(EIMSK)
5. 選擇外部中斷 INT0~INT3 觸發模式:設定選擇上緣，下緣，雙緣或準位觸發 201(EICRA)(ASA M128 未支援)

6. 選擇外部中斷 INT4~INT7 觸發模式:設定選擇上緣，下緣，雙緣或準位觸發,202(EICRB)
7. 外部中斷 INT0~INT3 觸發源 I/O 設定：設定觸發源腳為輸入 。204(DDRD) (ASA M128 未支援)
8. 外部中斷 INT4~INT7 觸發源 I/O 設定：設定觸發源腳為輸入 。205(DDRE)

這些設定旗標占用軟體暫存器及所對應 AVR 硬體暫存器如下列：
200(EIMSK),201(EICRA),202(EICRB),204(DDRD),205(DDRE)

傳值變數：

LSByte：依據不同的工作，所涉及的暫存器，需使用不同 LSByte。

1. 禁致能外部中斷 INT0~INT7：涉及 200(EIMSK)，LSByte=200
2. 選擇外部中斷 INT0~INT3 觸發模式:涉及 201(EICRA)，LSByte=201 (ASA-M128 不支援)
3. 選擇外部中斷 INT4~INT7 觸發模式:涉及 202(EICRB)，LSByte=202
4. 外部中斷 NT0~INT3 觸發源 I/O 設定：涉及 204(DDRD), LSByte=204 (ASA-M128 不支援)
5. 外部中斷 INT4~INT7 觸發源 I/O 設定：涉及 205(DDRE),LSByte=205。

Mask：使用 set 函式可以設定各種功能工作模式，以及禁致能工作。依據目的之不同，會開放部份的旗標，並遮罩住其他旗標。依目的之不同 Mask 參數會有不同值。

1. 禁致能外部中斷 INT0~INT7：INT0~INT7 各自佔用 200(EIMSK)的 bit0~bit7,故針對不同外部中斷 MASK 分別為

INT0: MASK=b0000 0001=0x01(ASA M128 未支援)

INT1: MASK=b0000 0010=0x02(ASA M128 未支援)

INT2: MASK=b0000 0100=0x04(ASA M128 未支援)

INT3: MASK=b0000 1000=0x08(ASA M128 未支援)

INT4: MASK=b0001 0000=0x10

INT5: MASK=b0010 0000=0x20

INT6: MASK=b0100 0000=0x40

INT7: MASK=b1000 0000=0x80

2. 選擇外部中斷 INT0~INT3 觸發模式:INT0~INT3 分別佔用 201(EICRA)暫存器的 bit1:0,bit3:2,bit5:4,bit7:6 位元。故針對不同外部中斷觸發 MASK 分別為

   INT0: MASK=b0000 0011=0x03(ASA M128 未支援)

   INT1: MASK=b0000 1100=0x0C(ASA M128 未支援)

   INT2: MASK=b0011 0000=0x30(ASA M128 未支援)

   INT3: MASK=b1100 0000=0xC0(ASA M128 未支援)

3. 選擇外部中斷 INT4~INT7 觸發模式:INT4~INT7 分別佔用 202(EICRB)暫存器的 bit1:0,bit3:2,bit5:4,bit7:6 位元。故針對不同外部中斷觸發 MASK 分別為

   INT4: MASK=b0000 0011=0x03

   INT5: MASK=b0000 1100=0x0C

   INT6: MASK=b0011 0000=0x30

   INT7: MASK=b1100 0000=0xC0

4. 外部中斷 NT0~INT3 觸發源 I/O 設定：觸發源腳 INT0~INT3 分別佔用 204(DDRD)bit0~bit3。故針對不同外部中斷觸發源腳 MASK 分別為

   INT0: MASK=b0000 0001=0x01(ASA M128 未支援)

   INT1: MASK=b0000 0010=0x02(ASA M128 未支援)

   INT2: MASK=b0000 0100=0x04(ASA M128 未支援)

   INT3: MASK=b0000 1000=0x08(ASA M128 未支援)

5. 外部中斷 INT4~INT7 觸發源 I/O 設定：觸發源腳 INT0~INT3 分別佔用 205(DDRE)bit4~bit7。故針對不同外部中斷觸發源腳 MASK 分別為

   INT4: MASK=b0001 0000=0x10

INT5: MASK=b0010 0000=0x20

INT6: MASK=b0100 0000=0x40

INT7: MASK=b1000 0000=0x80

Shift：使用 set 函式可以設定各種功能工作模式，以及禁致能工作。依據目的之不同，最左邊位元向左平移位數不同。依目的之不同 Shift 參數會有不同值。

- 禁致能外部中斷 INT0~INT7：INT0~INT7 各自佔用 200(EIMSK)的 bit0~bit7,故針對不同外部中斷 Shift 分別為

  INT0: Shift=0(ASA M128 未支援)

  INT1: Shift=1(ASA M128 未支援)

  INT2: Shift=2(ASA M128 未支援)

  INT3: Shift=3(ASA M128 未支援)

  INT4: Shift=4

  INT5: Shift=5

  INT6: Shift=6

  INT7: Shift=7

- 選擇外部中斷 INT0~INT3 觸發模式:INT0~INT3 分別佔用 201(EICRA)暫存器的 bit1:0,bit3:2,bit5:4,bit7:6 位元。故針對不同外部中斷觸發 Shift 分別為

  INT0: Shift=0 (ASA M128 未支援)

  INT1: Shift=2 (ASA M128 未支援)

  INT2: Shift=4 (ASA M128 未支援)

  INT3: Shift=6 (ASA M128 未支援)

- 選擇外部中斷 INT4~INT7 觸發模式:INT4~INT7 分別佔用 202(EICRB)暫存器的 bit1:0,bit3:2,bit5:4,bit7:6 位元。故針對不同外部中斷觸發 SHIFT 分別為

  INT4: Shift=0

INT5: Shift=2

INT6: Shift=4

INT7: Shift=6

- 外部中斷 NT0~INT3 觸發源 I/O 設定：觸發源腳 INT0~INT3 分別佔用 204(DDRD)bit0~bit3。故針對不同外部中斷觸發源腳 Shift 分別為

  INT0: Shift=0(ASA M128 未支援)

  INT1: Shift=1(ASA M128 未支援)

  INT2: Shift=2(ASA M128 未支援)

  INT3: Shift=3(ASA M128 未支援)

- 外部中斷 INT4~INT7 觸發源 I/O 設定：觸發源腳 INT0~INT3 分別佔用 205(DDRE)bit4~bit7。故針對不同外部中斷觸發源腳 Shift 分別為

  INT4: Shift=4

  INT5: Shift=5

  INT6: Shift=6

  INT7: Shift=7

Data：使用 set 函式可以設定各種功能工作模式，以及禁致能工作。依據目的之不同，有不同的 Data 值。

- 禁致能外部中斷 INT0~INT7：不論那一個，視設定為禁致能目標使用不同 Data
  - 禁能：Data=0 (預設值)
  - 致能：Data=1
- 選擇外部中斷 INT0~INT3 觸發模式:不論那一個，視觸發模式決定 Data
  - 低準位觸發：Data=0 (預設值)
  - 下降緣觸發：Data=2
  - 上升緣觸發：Data=3
- 選擇外部中斷 INT4~INT7 觸發模式:不論那一個，視觸發模式決定

Data

- 低準位觸發：Data=0 (預設值)
- 上下緣均觸發： Data=1
- 下降緣觸發：Data=2
- 上升緣觸發：Data=3

◦ 外部中斷 NT0~INT3 觸發源 I/O 設定：不論那一個，視輸出入決定 Data

- 輸入：Data=0 (開啟外部中以時應設也是預設)
- 輸出：Data=1

◦ 外部中斷 INT4~INT7 觸發源 I/O 設定：不論那一個，視輸出入決定 Data

- 輸入：Data=0 (開啟外部中以時應設也是預設)
- 輸出：Data=1

回應 RETURN：是否成功設定，回傳 0 時代表成功，其餘代表失敗

0：成功

1：輸入參數所選之設定本系統不支援。
2：警告，輸出入方向設定，已變更先前已有之其他設定驅動函式之方向設定，可能造成其他硬體驅動異常。

禁致中斷傳參表

中斷源觸發模式設定傳參表

中斷源輸出入設定傳參表