KBI 功能使用方法

1 適用產品：

1.1 SM59R16A2/ SM59R08A2

1.2 SM59R16A5/ SM59R09A5/ SM59R05A5/SM59R16A3/SM59R09A3/SM59R05A3

1.3 SM59R04A2/ SM59R04A1/ SM59R03A1/ SM59R02A1

2 文件說明：SM59R 系列注意 KBI 因架構差異，SM59R16A2/SM59R08A2 於第四節說明。

3 SM59R16A5/ SM59R09A5/ SM59R05A5/ SM59R16A3/ SM59R09A3/ SM59R05A3/ SM59R04A2/ SM59R04A1/ SM59R03A1/ SM59R02A1 KBI 使用說明：

3.1 共有 8 個 I/O 可致能爲 KBI 功能 I/O。

3.2 由 Port 0 或 Port 2 共 8 個 I/O，分別對應至 8 個獨立的 flag(KBF.0~ KBF.7)，且共用同一個中斷向量位置(0x5B)。

3.3 可由程式設定爲高電位或低電位觸發。

3.4 當觸發訊號輸入至 KBI 任一引腳，其對應的旗標將會被設置爲"1"，並且進入中斷副程式。

3.5 KBI 功能主要可做爲 4x4 矩陣式鍵盤掃描，或其它應用。

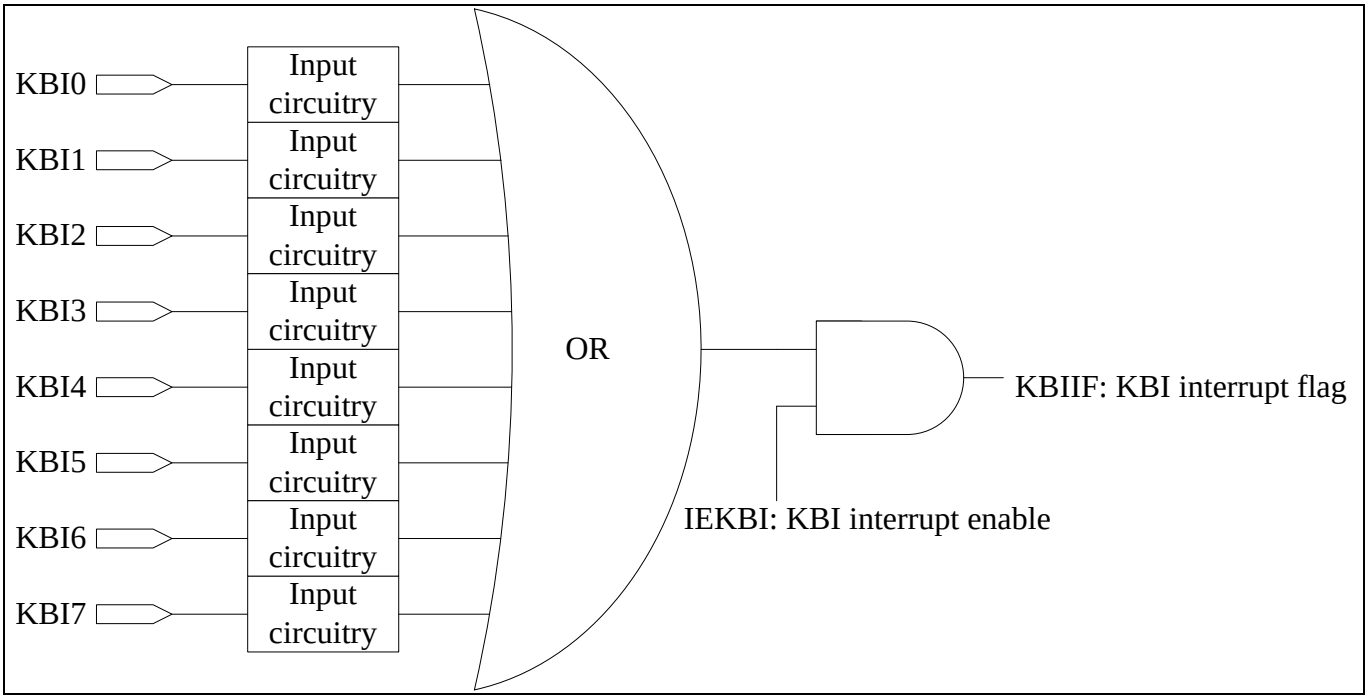

Fig. Interrupts from KBI 8 inputs

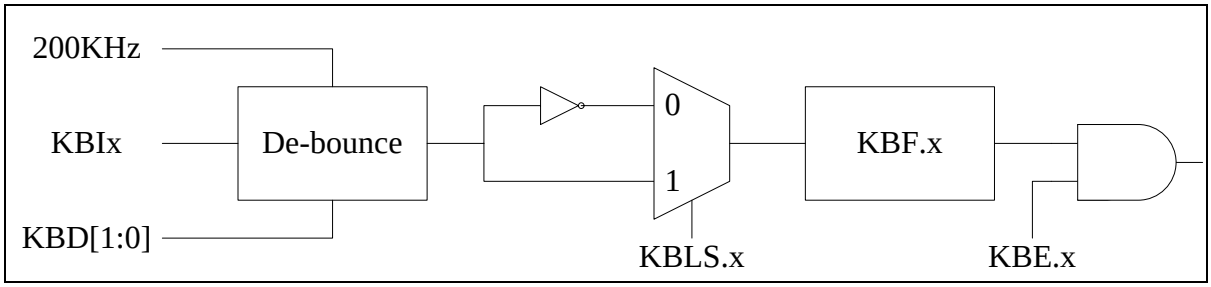

Fig. keyboard input circuitry

3.6 KBI 相關暫存器：

KBI	Description	Direct	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	RESET
KBI function											
AUX	Auxiliary register	91h	BRGS	P4CC	P4SPI	P4UR1	P4IIC	P0KBI	-	DPS	00H
KBLS	KBI level selection	93h	KBLS7	KBLS6	KBLS5	KBLS4	KBLS3	KBLS2	KBLS1	KBLS0	00H
KBE	KBI input enable	94h	KBE7	KBE6	KBE5	KBE4	KBE3	KBE2	KBE1	KBE0	00H
KBF	KBI flag	95h	KBF7	KBF6	KBF5	KBF4	KBF3	KBF2	KBF1	KBF0	00H
KBD	KBI De-bounce control register	96h	KBDEN	-	-	-	-	-	KBD1	KBD0	00H
IEN1	Interrupt Enable 1 register	B8h	EXEN2	-	IEIIC	IELVI	IEKBI	IEADC	IESPI	IEPWM	00h
IRCON	Interrupt request register	C0H	EXF2	TF2	IICIF	LVIIF	KBIIF	ADCIF	SPIIF	PWMIF	00H

Mnemonic: AUX **Address: 91h**

7	6	5	4	3	2	1	0	Reset
BRGS	P4CC	P4SPI	P4UR1	P4IIC	P0KBI	-	DPS	00H

P0KBI: P0KBI = 0 – KBI function on P2.
P0KBI = 1 – KBI function on P0.

Mnemonic: KBLS **Address: 93h**

7	6	5	4	3	2	1	0	Reset
KBLS.7	KBLS.6	KBLS.5	KBLS.4	KBLS.3	KBLS.2	KBLS.1	KBLS.0	00h

KBLS.7: 觸發準位設定位元 (Keyboard Line 7 level selection bit)
0：設定爲低準位觸發
1：設定爲高準位觸發
KBLS.6: 觸發準位設定位元 (Keyboard Line 6 level selection bit)
0：設定爲低準位觸發
1：設定爲高準位觸發
KBLS.5: 觸發準位設定位元 (Keyboard Line 5 level selection bit)
0：設定爲低準位觸發
1：設定爲高準位觸發
KBLS.4: 觸發準位設定位元 (Keyboard Line 4 level selection bit)
0：設定爲低準位觸發

1：設定爲高準位觸發
KBLS.3: 觸發準位設定位元 (Keyboard Line 3 level selection bit)
0：設定爲低準位觸發
1：設定爲高準位觸發
KBLS.2: 觸發準位設定位元 (Keyboard Line 2 level selection bit)
0：設定爲低準位觸發
1：設定爲高準位觸發
KBLS.1: 觸發準位設定位元 (Keyboard Line 1 level selection bit)
0：設定爲低準位觸發
1：設定爲高準位觸發
KBLS.0: 觸發準位設定位元 (Keyboard Line 0 level selection bit)
0：設定爲低準位觸發
1：設定爲高準位觸發

Mnemonic: KBE **Address: 94h**

7	6	5	4	3	2	1	0	Reset
KBE.7	KBE.6	KBE.5	KBE.4	KBE.3	KBE.2	KBE.1	KBE.0	00h

KBE.7: 致能位元 (Keyboard Line 7 enable bit)
0 : KBI 禁能，預設爲一般的出入埠(GPIO)
1 : KBI 致能，當觸發訊號輸入時，KBF.7 = 1，會觸發中斷
KBE.6: 致能位元 (Keyboard Line 6 enable bit)
0 : KBI 禁能，預設爲一般的出入埠(GPIO)
1 : KBI 致能，當觸發訊號輸入時，KBF.7 = 1，會觸發中斷
KBE.5: 致能位元 (Keyboard Line 5 enable bit)
0 : KBI 禁能，預設爲一般的出入埠(GPIO)
1 : KBI 致能，當觸發訊號輸入時，KBF.7 = 1，會觸發中斷
KBE.4: 致能位元 (Keyboard Line 4 enable bit)
0 : KBI 禁能，預設爲一般的出入埠(GPIO)
1 : KBI 致能，當觸發訊號輸入時，KBF.7 = 1，會觸發中斷
KBE.3: 致能位元 (Keyboard Line 3 enable bit)
0 : KBI 禁能，預設爲一般的出入埠(GPIO)
1 : KBI 致能，當觸發訊號輸入時，KBF.7 = 1，會觸發中斷
KBE.2: 致能位元 (Keyboard Line 2 enable bit)
0 : KBI 禁能，預設爲一般的出入埠(GPIO)
1 : KBI 致能，當觸發訊號輸入時，KBF.7 = 1，會觸發中斷
KBE.1: 致能位元 (Keyboard Line 1 enable bit)
0 : KBI 禁能，預設爲一般的出入埠(GPIO)
1 : KBI 致能，當觸發訊號輸入時，KBF.7 = 1，會觸發中斷
KBE.0: 致能位元 (Keyboard Line 0 enable bit)
0 : KBI 禁能，預設爲一般的出入埠(GPIO)
1 : KBI 致能，當觸發訊號輸入時，KBF.7 = 1，會觸發中斷

Mnemonic: KBF **Address: 95h**

7	6	5	4	3	2	1	0	Reset
KBF.7	KBF.6	KBF.5	KBF.4	KBF.3	KBF.2	KBF.1	KBF.0	00h

KBF.7: KBI 觸發旗標 (Keyboard Line 7 flag)

1: 當 KBI7 偵測到輸入訊號時，KBF.7 由硬體設爲"1"，並產生 KBI 中斷；
0: 必須由軟體清除爲"0"

KBF.6: KBI 觸發旗標 (Keyboard Line 6 flag)
1: 當 KBI6 偵測到輸入訊號時，KBF.6 由硬體設爲"1"，並產生 KBI 中斷；
0: 必須由軟體清除爲"0"

KBF.5: KBI 觸發旗標 (Keyboard Line 5 flag)
1: 當 KBI5 偵測到輸入訊號時，KBF.5 由硬體設爲"1"，並產生 KBI 中斷；
0: 必須由軟體清除爲"0"

KBF.4: KBI 觸發旗標 (Keyboard Line 4 flag)
1: 當 KBI4 偵測到輸入訊號時，KBF.4 由硬體設爲"1"，並產生 KBI 中斷；
0: 必須由軟體清除爲"0"

KBF.3: KBI 觸發旗標 (Keyboard Line 3 flag)
1: 當 KBI3 偵測到輸入訊號時，KBF.3 由硬體設爲"1"，並產生 KBI 中斷；
0: 必須由軟體清除爲"0"

KBF.2: KBI 觸發旗標 (Keyboard Line 2 flag)
1: 當 KBI2 偵測到輸入訊號時，KBF.2 由硬體設爲"1"，並產生 KBI 中斷；
0: 必須由軟體清除爲"0"

KBF.1: KBI 觸發旗標 (Keyboard Line 1 flag)
1: 當 KBI1 偵測到輸入訊號時，KBF.1 由硬體設爲"1"，並產生 KBI 中斷；
0: 必須由軟體清除爲"0"

KBF.0: KBI 觸發旗標 (Keyboard Line 0 flag)
1: 當 KBI0 偵測到輸入訊號時，KBF.0 由硬體設爲"1"，並產生 KBI 中斷；
0: 必須由軟體清除爲"0"

Mnemonic: KBD **Address: 96H**

7	6	5	4	3	2	1	0	Reset
KBDEN	-	-	-	-	-	KBD.1	KBD.0	00H

KBDEN: 除彈跳功能致能位元(Enable KBI de-bounce function).
預設爲致能(The default KBI function is enabled).
KBDEN = 0, 致能除彈跳功能
KBDEN = 1, 禁能除彈跳功能

KBD[1:0]: 除彈跳時間設定位元. 若 KBDEN = "0"，則預設爲 320 ms.
KBD[1:0] = 00, 除彈跳時間爲 320 ms.
KBD[1:0] = 01, 除彈跳時間爲 160 ms.
KBD[1:0] = 10, 除彈跳時間爲 80 ms.
KBD[1:0] = 11, 除彈跳時間爲 40 ms.

Mnemonic: IEN1 **Address: B8h**

7	6	5	4	3	2	1	0	Reset
EXEN2	-	IEIIC	IELVI	IEKBI	IEADC	IESPI	IEPWM	00h

IEKBI: KBI interrupt enable.
IEKBI = 0 – Disable KBI interrupt.
IEKBI = 1 – Enable KBI interrupt.

Mnemonic: IRCON **Address: C0h**

7	6	5	4	3	2	1	0	Reset
EXF2	TF2	IICIF	LVIIF	KBIIF	ADCIF	SPIIF	PWMIF	00H

KBIIF: KBI interrupt flag. Must be cleared by software.

3.7 以下是 KBI 相對應的中斷向量表：

Table 11-1: Interrupt vectors

Interrupt Request Flags	Interrupt Vector Address	Interrupt Number *(use Keil C Tool)
IE0 – External interrupt 0	0003h	0
TF0 – Timer 0 interrupt	000Bh	1
IE1 – External interrupt 1	0013h	2
TF1 – Timer 1 interrupt	001Bh	3
RI0/TI0 – Serial channel 0 interrupt	0023h	4
TF2/EXF2 – Timer 2 interrupt	002Bh	5
PWMIF – PWM interrupt	0043h	8
SPIIF – SPI interrupt	004Bh	9
ADCIF – A/D converter interrupt	0053h	10
KBIIF – keyboard Interface interrupt	***005Bh***	***11***
LVIIF – Low Voltage Interrupt	0063h	12
IICIF – IIC interrupt	006Bh	13
RI1/TI1 – Serial channel 1 interrupt	0083h	16

*See Keil C about C51 User's Guide about Interrupt Function description

KBI 中斷應用的參考程式：

(1) 中斷致能設置：

```
void Init_KBI(void)
{
  //KBI function input pin select//
  AUX = 0x00;          //enable all Port 2 as KBI I/O
  //AUX = 0x04;        // enable all Port 0 as KBI I/O

  //de-bounce mechanism select//
  KBD = 0x00;          //enable KBI de-bounce function and set de-bounce time is 320 ms.
  //KBD = 0x02;        // enable KBI de-bounce function and set de-bounce time is 80 ms.
  //KBD = 0x80;        //disable KBI de-bounce mechanism

  //Keyboard level selector register
```

```
    KBLS = 0x00;        //Detect_Lo_level;
    //KBLS = 0xff;      //Detect_Hi_level;

    //Keyboard input enable register
    //KBE = 0x00;       //enable standard I/O pin
    KBE = 0xff;         //enable KBI.KBF to generate interrupt

    IEN0 = 0x80;        //enable all interrupt
    IEN1 = 0x08;        //enable KBI interrupt
    KBF = 0x00;         //clear KBF flag
    IRCON_KBIIF = 0;    //clear KBI flag
}
```

(2) 中斷副程式：

```
void KBI_ISR(void) interrupt 11       //INTERRUPT VECTOR ADDR 0x5B
{
  KBF = 0x00;         //clear KBF flag
  IRCON_KBIIF = 0;    //clear KBI flag
}
```

3.8 KBI 使用組合語言的範例說明

3.8.1 KBI I/O全部致能

3.8.2 當外部觸發訊號至KBI I/O時產生中斷，並將結果顯示於Port1

3.9 KBI 使用組合語言的範例程式

Description	//此範例應用包括

Main program	;===
	;
	; S Y N C M O S T E C H N O L O G Y
	;
	;===
	; KBI function input pin select
	sfr AUX = 0x91;
	; de-bounce mechanism select
	sfr KBD = 0x96;
	;Keyboard level selector register
	sfr KBLS = 0x93;
	;Keyboard input enable register
	sfr KBE = 0x94;
	;Keyboard interrupt flag register
	sfr KBF = 0x95;
	sfr P1 = 0x90;

```
sfr P2      = 0xA0;
sfr IEN0    = 0xA8;
sfr IEN1    = 0xB8;
;sfr    IEN2    = 0x9A;
sfr IRCON   = 0xC0;

    ORG     0000H
    JMP     BEGIN
    ORG     005BH
    JMP     KBI_INT
;================================================
;KBI INTERRUPT INIT ROUTINE
;================================================
BEGIN:
;   MOV     P1,     #00H    ;
    MOV     IEN0,   #80H    ;EA =1
    MOV     IEN1,   #08H    ;IEKBI =1
;   MOV     IEN2,   #00H    ;
    MOV     AUX,    #00H    ; enable all Port 2 as KBI I/O
    MOV     KBD,    #00H    ; enable KBI de-bounce function and set de-bounce
                            ;time is 320 ms.
    MOV     KBLS,   #0FFH   ;High level detect
;   MOV     KBLS,   #000H   ;Low level detect
    MOV     KBE,    #0FFH   ;KBI all pin enable
    MOV     KBF,    #00H    ;KBI all flag clear
    JMP START
;================================================
;MAIN
;================================================
START:
    JMP START
;================================================
;KBI INTERRUPT ROUTINE
;================================================
KBI_INT:
    MOV     P1,     KBF
    MOV     KBF,    #00H
    ANL     IRCON,  #08H
    RETI
END
```

4 SM59R16A2/ SM59R08A2 EEI 使用說明：

4.1 Port1 所有的引腳皆可爲 GPIO，或獨立致能爲 EEI 功能

4.2 由 Port1.0~Port1.7 共八支引腳，分別對應至 8 個獨立的 flag(KBF.0~ KBF.7)，及共用同一個中斷向量位置(0x5B)

4.3 可由程式設定爲高電位或低電位觸發

4.4 當觸發訊號輸入至 Port1 任一引腳，其對應的旗標將會被設置爲"1"，並且進入中斷副程式

4.5 EEI 功能主要可做爲 4x4 矩陣式鍵盤掃描，或其它類似的應用

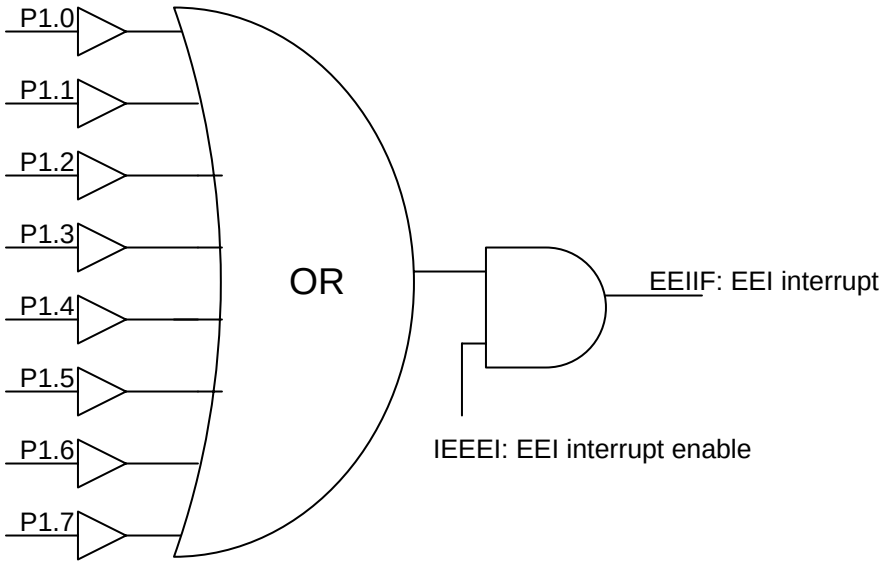

Fig. Interrupts from EEI 8 inputs

4.6 EEI 相關暫存器：

EEI	Description	Direct	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	RESET
EEI function											
KBLS	EEI level selection	93h	KBLS7	KBLS6	KBLS5	KBLS4	KBLS3	KBLS2	KBLS1	KBLS0	00h
KBE	EEI input enable	94h	KBE7	KBE6	KBE5	KBE4	KBE3	KBE2	KBE1	KBE0	00h
KBF	EEI flag	95h	KBF7	KBF6	KBF5	KBF4	KBF3	KBF2	KBF1	KBF0	00h

Mnemonic: KBLS **Address: 93h**

7	6	5	4	3	2	1	0	Reset
KBLS.7	KBLS.6	KBLS.5	KBLS.4	KBLS.3	KBLS.2	KBLS.1	KBLS.0	00h

KBLS.7: EEI line 7 觸發準位設定位元
0：設定爲低準位觸發
1：設定爲高準位觸發
KBLS.6: EEI line 6 觸發準位設定位元
0：設定爲低準位觸發
1：設定爲高準位觸發

KBLS.5: EEI line 5 觸發準位設定位元
0：設定爲低準位觸發
1：設定爲高準位觸發
KBLS.4: EEI line 4 觸發準位設定位元
0：設定爲低準位觸發
1：設定爲高準位觸發
KBLS.3: EEI line 3 觸發準位設定位元
0：設定爲低準位觸發
1：設定爲高準位觸發
KBLS.2: EEI line 2 觸發準位設定位元
0：設定爲低準位觸發
1：設定爲高準位觸發
KBLS.1: EEI line 1 觸發準位設定位元
0：設定爲低準位觸發
1：設定爲高準位觸發
KBLS.0: EEI line 0 觸發準位設定位元
0：設定爲低準位觸發
1：設定爲高準位觸發

Mnemonic: KBE **Address: 94h**

7	6	5	4	3	2	1	0	Reset
KBE.7	KBE.6	KBE.5	KBE.4	KBE.3	KBE.2	KBE.1	KBE.0	00h

KBE.7: EEI line 7 (P1.7)致能位元
0：致能爲一般的出入埠(GPIO)
1：致能爲 EEI 功能，當觸發訊號輸入時，KBF.7 = 1，會觸發中斷
KBE.6: EEI line 6 (P1.6)致能位元
0：致能爲一般的輸出入埠(GPIO)
1：致能爲 EEI 功能，當觸發訊號輸入時，KBF.6 = 1，會觸發中斷
KBE.5: EEI line 5 (P1.5)致能位元
0：致能爲一般的輸出入埠(GPIO)
1：致能爲 EEI 功能，當觸發訊號輸入時，KBF.5 = 1，會觸發中斷
KBE.4: EEI line 4 (P1.4)致能位元
0：致能爲一般的輸出入埠(GPIO)
1：致能爲 EEI 功能，當觸發訊號輸入時，KBF.4 = 1，會觸發中斷
KBE.3: EEI line 3 (P1.3)致能位元
0：致能爲一般的輸出入埠(GPIO)
1：致能爲 EEI 功能，當觸發訊號輸入時，KBF.3 = 1，會觸發中斷
KBE.2: EEI line 2 (P1.2)致能位元
0：致能爲一般的輸出入埠(GPIO)
1：致能爲 EEI 功能，當觸發訊號輸入時，KBF.2 = 1，會觸發中斷
KBE.1: EEI line 1 (P1.1)致能位元
0：致能爲一般的輸出入埠(GPIO)
1：致能爲 EEI 功能，當觸發訊號輸入時，KBF.1 = 1，會觸發中斷
KBE.0: EEI line 0 (P1.0)致能位元
0：致能爲一般的輸出入埠(GPIO)
1：致能爲 EEI 功能，當觸發訊號輸入時，KBF.0 = 1，會觸發中斷

Mnemonic: KBF **Address: 95h**

7	6	5	4	3	2	1	0	Reset
KBF.7	KBF.6	KBF.5	KBF.4	KBF.3	KBF.2	KBF.1	KBF.0	00h

KBF.7: EEI line 7 flag
當觸發訊號輸入時，KBF.7 = 1，會觸發中斷；
必須由軟體清除爲”0”
KBF.6: EEI line 6 flag
當觸發訊號輸入時，KBF.6 = 1，會觸發中斷；
必須由軟體清除爲”0”
KBF.5: EEI line 5 flag
當觸發訊號輸入時，KBF.5 = 1，會觸發中斷；
必須由軟體清除爲”0”
KBF.4: EEI line 4 flag
當觸發訊號輸入時，KBF.4 = 1，會觸發中斷；
必須由軟體清除爲”0”
KBF.3: EEI line 3 flag
當觸發訊號輸入時，KBF.3 = 1，會觸發中斷；
必須由軟體清除爲”0”
KBF.2: EEI line 2 flag
當觸發訊號輸入時，KBF.2 = 1，會觸發中斷；
必須由軟體清除爲”0”
KBF.1: EEI line 1 flag
當觸發訊號輸入時，KBF.1 = 1，會觸發中斷；
必須由軟體清除爲”0”
KBF.0: EEI line 0 flag
當觸發訊號輸入時，KBF.0 = 1，會觸發中斷；
必須由軟體清除爲”0”

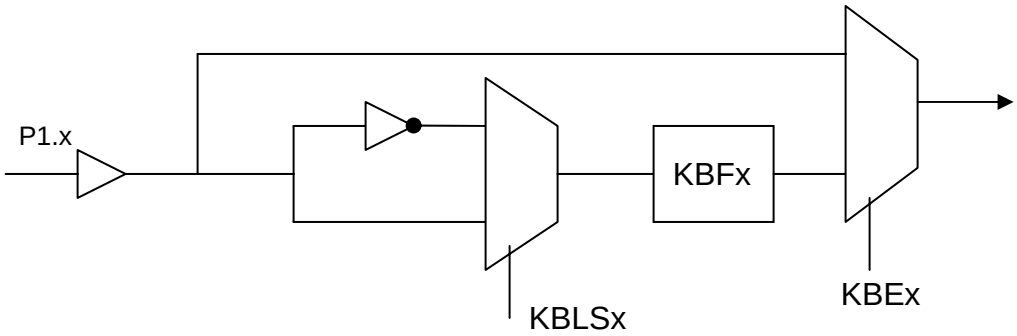

Fig. Block diagram of EEI input

4.7 以下是 **EEI** 相對應的中斷向量表：

Table 11-1: Interrupt vectors

Interrupt Request Flags	Interrupt Vector Address	Interrupt Number *(use Keil C Tool)
IE0 – External interrupt 0	0003h	0

TF0 – Timer 0 interrupt	000Bh	1
IE1 – External interrupt 1	0013h	2
TF1 – Timer 1 interrupt	001Bh	3
RI0/TI0 – Serial channel 0 interrupt	0023h	4
TF2/EXF2 – Timer 2 interrupt	002Bh	5
SPIIF – SPI interrupt	004Bh	9
ADCIF – A/D converter interrupt	0053h	10
EEIIF – Expanded External Interrupt	***005Bh***	***11***
IICIF – IIC interrupt	006Bh	13
RI1/TI1 – Serial channel 1 interrupt	0083h	16

*See Keil C about C51 User's Guide about Interrupt Function description

EEI 中斷應用的參考程式：

(1) 中斷致能設置：

```
void Init_EEI(void)
{
  //Keyboard level selector register
  KBLS = 0x00;        //Detect_Lo_level;
  //KBLS = 0xff;      //Detect_Hi_level;
  //Keyboard input enable register
  //KBE   = 0x00;     //enable standard I/O pin
  KBE = 0xff;     //enable EEI.KBF to generate interrupt
  IEN0 = 0x80;        //enable all interrupt
  IEN1 = 0x08;        //enable EEI interrupt
  KBF  = 0x00;    //clear KBF flag
  IRCON_EEIIF = 0;    //clear EEI flag
}
```

(2) 中斷副程式：

```
void EEI_ISR(void) interrupt 11       //INTERRUPT VECTOR ADDR 0x5B
{
  KBF = 0x00;         //clear KBF flag
  IRCON_EEIIF = 0;    //clear EEI flag
}
```

4.8 EEI 使用組合語言的範例說明

4.8.1 Port1中所有的EEI功能全部致能

4.8.2 當外部觸外訊號至Port1時產生中斷，並將結果顯示於Port2

4.9 EEI 使用組合語言的範例程式

Description	//此範例應用包括

Main program	;===
	;
	; S Y N C M O S T E C H N O L O G Y
	;
	;===
	;Keyboard level selector register
	sfr KBLS = 0x93;
	;Keyboard input enable register
	sfr KBE = 0x94;
	;Keyboard interrupt flag register
	sfr KBF = 0x95;
	sfr P1 = 0x90;
	sfr P2 = 0xA0;
	sfr IEN0 = 0xA8;
	sfr IEN1 = 0xB8;
	;sfr IEN2 = 0x9A;
	sfr IRCON = 0xC0;
	
	ORG 0000H
	JMP BEGIN
	ORG 005BH
	JMP EEI_INT
	;==
	;EEI INTERRUPT INIT ROUTINE
	;==
	BEGIN:
	; MOV P1, #00H ;
	MOV IEN0, #80H ;EA =1
	MOV IEN1, #08H ;IEEEI =1
	; MOV IEN2, #00H ;
	MOV KBLS, #0FFH ;High level detect
	; MOV KBLS, #000H ;Low level detect
	MOV KBE, #0FFH ;EEI all pin enable
	MOV KBF, #00H ;EEI all flag clear
	JMP START
	;==
	;MAIN
	;==
	START:
	JMP START
	;==
	;EEI INTERRUPT ROUTINE
	;==
	EEI_INT:
	MOV P2, KBF
	MOV KBF, #00H
	ANL IRCON, #08H
	RETI
	END

4.10 EEI 使用 C 語言的範例說明(4x4 按鍵掃描)

4.10.1 硬體以SM59R16A2開發板(SDB0001B)爲例，或可參考開發板電路自行製作。

4.10.2 原本開發板的KEYPAD.0~7是接到Port5.0~7，只要依序接至Port1.0~7即可。

4.10.3 副程式Init_EEI(void)中，只有致能Port1.4~7爲EEI功能，Port1.0~3仍爲GPIO使用。

4.10.4 當按下鍵盤時，即觸發中斷副程式EEI_ISR(void)，程式中將根據KBF及shift即可顯示該數值，亦可將main(void)中for迴圈寫在timer中，提升系統的效率。

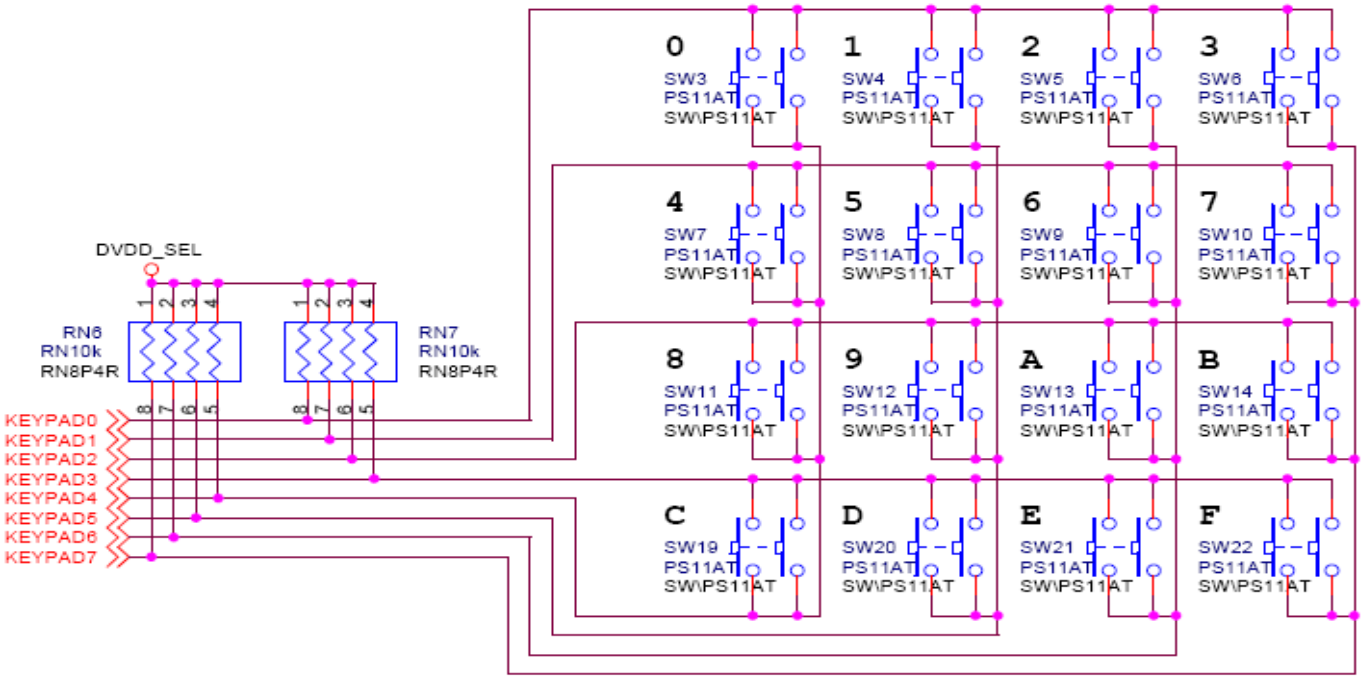

圖. SM59R16A2 開發板 SDB0001B 4x4 鍵盤電路圖

4.11 EEI 使用 C 語言的範例程式(4x4 按鍵掃描)

Description	

Main program	//===
	//
	// S Y N C M O S T E C H N O L O G Y
	//
	//===
	//===
	//INCLUDE FILES
	//===
	#include "..\h\SM59R16A2.h"
	#include "..\h\SM59R16A2_ExtraDef.h"
	#include "..\MISC\delay.h"
	//===
	//EEI INTERRUPT FLAG, MUST CLEAR BY SOFTWARE
	//===
	sbit IRCON_EEIIF = IRCON^3;
	unsigned char code arr[4] = {0xFE,0xFD, 0xFB, 0xF7 };
	unsigned char shift=0;

```
//====================================================================
//EEI INTERRUPT SUBROUTINE
//INTERRUPT NUM.11 = INTERRUPT VECTOR ADDR 0x5B
//====================================================================
void EEI_ISR(void) interrupt 11     //INTERRUPT VECTOR ADDR 0x5B
{
/*KeypadTable[]={
                  0x11, 0x21, 0x41, 0x81,       // 1  2  3  4
                  0x12, 0x22, 0x42, 0x82,       // 5  6  7  8
                  0x13, 0x23, 0x43, 0x83,       // 9  10 11 12
                  0x14, 0x24, 0x44, 0x84        // 13 14 15 16
};*/
    switch(KBF+shift)           // Keyboard Flag
    {
        case 0x11:          P2 = 1;
            break;
        case 0x21:          P2 = 2;
            break;
        case 0x41:          P2 = 3;
            break;
        case 0x81:          P2 = 4;
            break;
        case 0x12:          P2 = 5;
            break;
        case 0x22:          P2 = 6;
            break;
        case 0x42:          P2 = 7;
            break;
        case 0x82:          P2 = 8;
            break;
        case 0x13:          P2 = 9;
            break;
        case 0x23:          P2 = 0x10;
            break;
        case 0x43:          P2 = 0x11;
            break;
        case 0x83:          P2 = 0x12;
            break;
        case 0x14:          P2 = 0x13;
            break;
        case 0x24:          P2 = 0x14;
            break;
        case 0x44:          P2 = 0x15;
            break;
        case 0x84:          P2 = 0x16;
            break;
        default:            P2 = 0x00;
            break;
    }
    Delay10mSec(2);
    //P2 =  KBF+shift;
    P2 = 0x00;          //clear status
    KBF = 0x00;         //clear KBF flag
    IRCON_EEIIF = 0;    //clear EEI flag
}
//====================================================================
//EEI INTERRUPT INITIALIZE
```

```
//=====================================================================
void Init_EEI(void)
{
    //Keyboard level selector register
    KBLS = 0x00;        //Detect_Lo_level;
    //KBLS = 0xff;      //Detect_Hi_level;

    //Keyboard input enable register
    //KBE   = 0x00;     //enable standard I/O pin
    //KBE   = 0xff;     //enable EEI.KBF to generate interrupt
    KBE = 0xf0;         //

    IEN0 = 0x80;        //enable all interrupt
    IEN1 = 0x08;        //enable EEI interrupt
    KBF  = 0x00;    //clear KBF flag
    IRCON_EEIIF = 0;    //clear EEI flag
}

void main(void)
{
    Init_EEI();
    P2 = 0x00;          //clear status
    while(1)
    {
        for(shift=0;shift<4;shift++)
        {
            P1 = arr[shift];
        }
    }
}
```