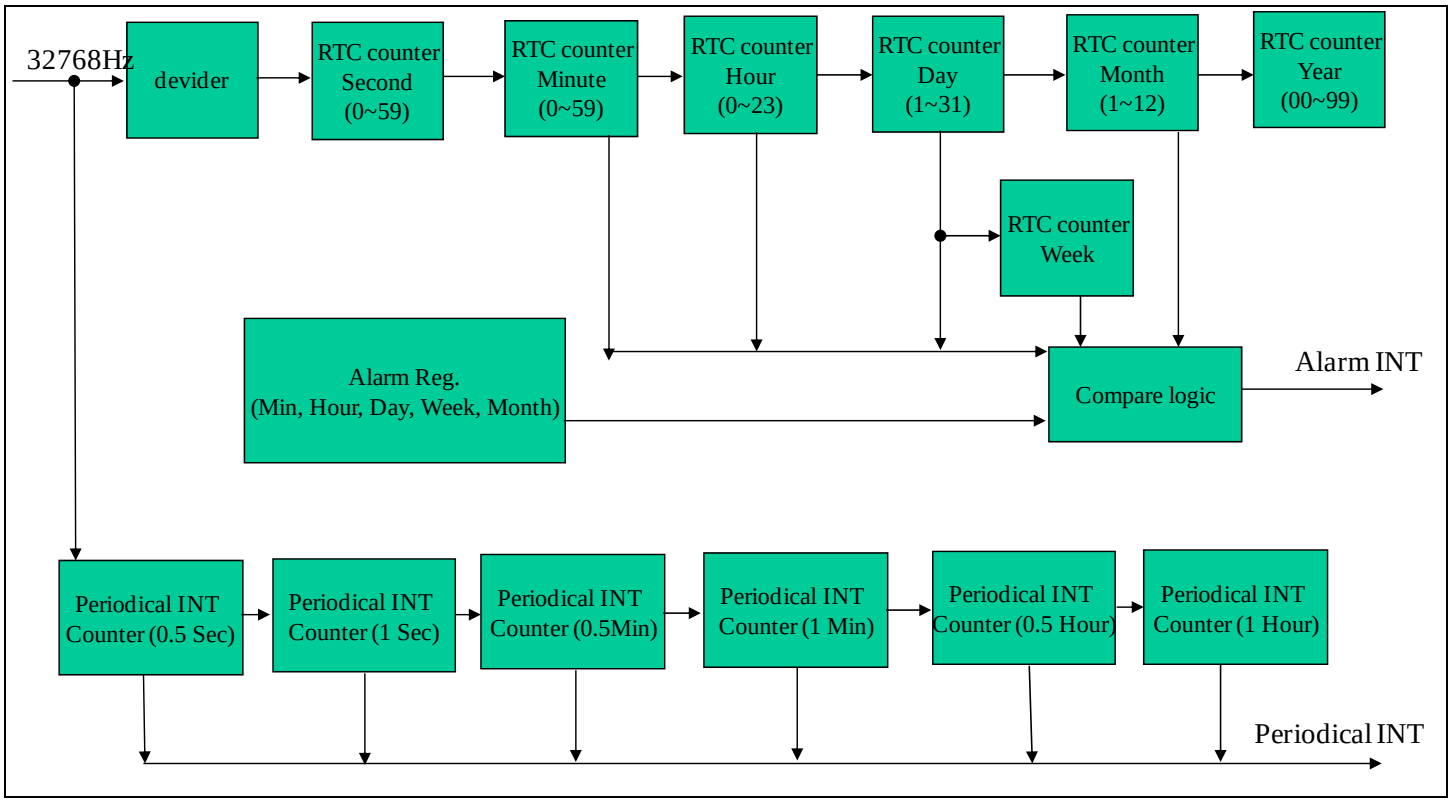
RTC(實時時鐘)功能使用方法

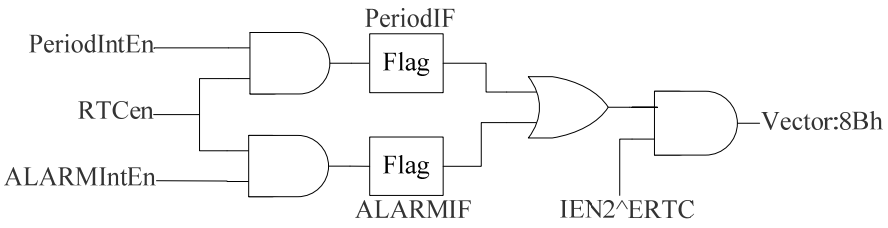
1 適用產品：SM59R16A5/ SM59R09A5/ SM59R05A5

2 RTC 使用說明：

1. 建內一組 RTC，時間具有秒、分、時、日、星期、月、年。
2. **RTC 晶振選用 32.768KHz，匹配電容建議使用 25pF，或依實際晶振做調教。**
3. 二種中斷，分別，爲 period interrupt 和 ALARM interrupt，且共用 interrupt vector(8Bh)。

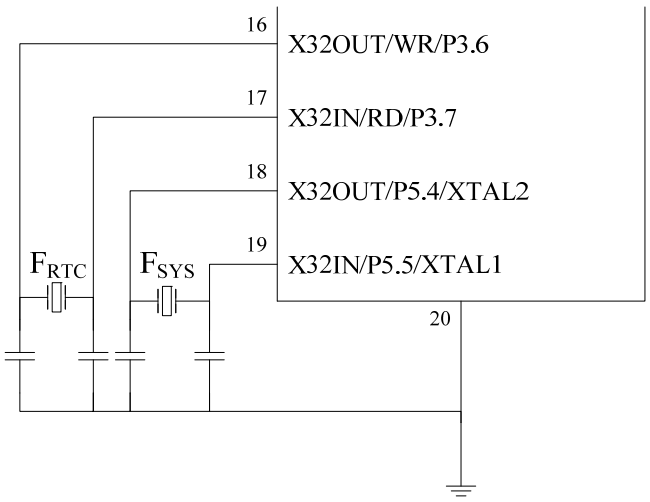
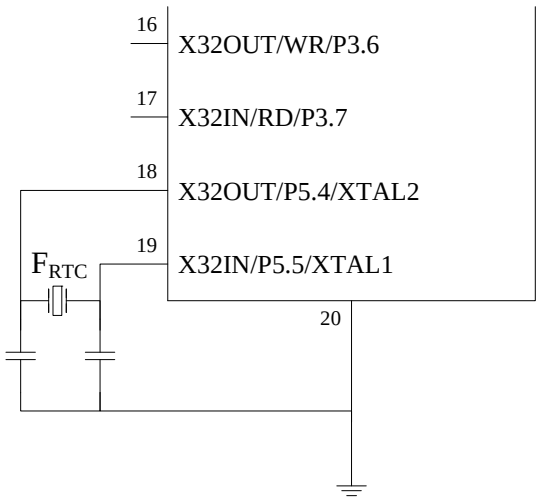


實時時鐘與相關暫存器控制圖(一)



實時時鐘與相關暫存器控制圖(二)

4. RTC 功能在硬件規劃時必須注意，依系統選用內振和外振會有差異，如下說明：

使用 RTC 功能，且系統使用外部晶振時：	使用 RTC 功能，且系統使用內部晶振時：
■ RTC 晶振則使用#16 及#17 ■ 系統晶振腳位使用#18 和#19 ■ 建議電路如下圖：	■ RTC 晶振則使用#18 及#19 ■ 建議電路如下圖：
	

3 RTC 相關暫存器：

Mnemonic	Description	Direct	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	RESET
RTC											
RTCADDR	RTC 地址	9Eh	-	-	-	-	RTCADDR[3:0]				00H
RTCDATA	RTC 數據	9Fh	RTCDATA[7:0]								00H

Mnemonic	Description	Address	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	RESET
RTC function											
RTCCTRL	RTC 控制暫存器	00h	RTCen	PeriodIF	ALARMInten	ALARMIF	PeriodIntEn	INT_SEL[2:0]			00h
SEC	RTC 秒暫存器	01h	-	SEC[6:4]			SEC[3:0]				00h
MIN	RTC 分暫存器	02h	-	MIN[6:4]			MIN[3:0]				00h
HOUR	RTC 時暫存器	03h	-	-	HOUR[5:4]		HOUR[3:0]				00h
DAY	RTC 日暫存器	04h	-	-	DAY[5:4]		DAY[3:0]				01h
WEEK	RTC 星期暫存器	05h	-	-	-	-	-	WEEK[2:0]			06h

MONTH	RTC 月暫存器	06h	-	-	-	MONTH[4]	MONTH[3:0]		01h
YEAR	RTC 年暫存器	07h	YEAR[7:4]				YEAR[3:0]		00h
ALMMIN	RTC 分示警	08h	MINen	ALMMIN[6:4]			ALMMIN[3:0]		00h
ALMHOUR	RTC 時示警	09h	HOURen	-	ALMHOUR[5:4]		ALMHOUR[3:0]		00h
ALMDAY	RTC 日示警	10h	DAYen	-	ALMDAY[5:4]		ALMDAY[3:0]		00h
ALMWEEK	RTC 星期示警	11h	WEEKen	-	-	-	-	ALMWEEK[2:0]	00h
ALMMONTH	RTC 月示警	12h	MONTHen	-	-	ALMMONTH[4]	ALMMONTH[3:0]		00h

Mnemonic: RTCCTRL **Address: 00H**

7	6	5	4	3	2	1	0	Reset
RTCen	PeriodIF	ALARMInten	ALARMIF	PeriodIntEn	INT_SEL[2:0]			00h

RTCen: RTC 電路啓動位元
0: 禁止
1: 啓動
PeriodIF: 週期性中斷旗標位元
1: RTC 中斷由週期性中斷旗標位元產生.他必須由軟體清零.
ALARMInten: 示警中斷啓動位元
當示警數值與RTC數值相等時將產生示警中斷.
0: 示警中斷禁止
1: 示警中斷啓動
ALARMIF: 示警中斷旗標位元
1: RTC 中斷由示警中斷產生.他必須由軟體清零.
PeriodIntEn: 週期性中斷啓動位元
0: 禁止
1: 啓動
INT_SEL[2:0]: RTC 週期性中斷之週期由INT_SEL[2:0]選擇
000: 設定爲每經過0.5秒產生週期性中斷
001: 設定爲每經過1秒產生週期性中斷
010: 設定爲每經過0.5分產生週期性中斷
011: 設定爲每經過1分產生週期性中斷
100: 設定爲每經過0.5小時產生週期性中斷
101: 設定爲每經過1小時產生週期性中斷
附注: 週期性中斷的計數將從RTC暫存器寫入的值開始計數.
舉例來說，現在時間爲 3:23，我們設定INT_SEL[2:0]爲0.5小時；第一個週期性中斷將在3:53產生，不是3:30。

Mnemonic: SEC **Address: 01H**

7	6	5	4	3	2	1	0	Reset
	SEC[6:4]			SEC[3:0]				00H

SEC[6:0]: 以BCD格式顯示目前秒數,區間爲0~59秒.
SEC[6:4]: 此3位代表秒之十位.
SEC[3:0]: 此4位代表秒之個位.

Mnemonic: MIN **Address: 02H**

7	6	5	4	3	2	1	0	Reset
	MIN[6:4]			MIN[3:0]				00H

MIN[6:0]: 以BCD格式顯示目前分鐘數,區間爲0~59分鐘.
MIN[6:4]: 此3位代表分鐘之十位.
MIN[3:0]: 此4位代表分鐘之個位.

Mnemonic: HOUR **Address: 03H**

7	6	5	4	3	2	1	0	Reset
		HOUR[5:4]		HOUR[3:0]				00H

HOUR[5:0]: 以BCD格式顯示目前時數,區間爲0~23小時.
HOUR[5:4]: 此2位代表時之十位.
HOUR[3:0]: 此4位代表時之個位.

Mnemonic: DAY **Address: 04H**

7	6	5	4	3	2	1	0	Reset
		DAY[5:4]		DAY[3:0]				00H

DAY[5:0]: 以BCD格式顯示目前日期,區間爲0~31日.
DAY[5:4]: 此2位代表日期之十位.
DAY[3:0]: 此4位代表日期之個位.

Mnemonic: WEEK **Address: 05H**

7	6	5	4	3	2	1	0	Reset
					WEEK[2:0]			00H

WEEK[2:0]: 顯示目前星期.
此暫存器必須由使用者自行填入,並不會於填入年月日後自動產生.
000: 星期日
001: 星期一
010: 星期二
011: 星期三
100: 星期四
101: 星期五
110: 星期六

Mnemonic: MONTH **Address: 06H**

7	6	5	4	3	2	1	0	Reset
			MONTH[4]	MONTH[3:0]				00H

MONTH[4:0]: 以BCD格式顯示目前月份,區間爲1~12月.
MONTH[4]: 此位代表月之十位.
MONTH[3:0]: 此4位代表月之個位.

Mnemonic: YEAR **Address: 07H**

7	6	5	4	3	2	1	0	Reset
YEAR[7:4]				YEAR[3:0]				00H

YEAR[7:0]: 以BCD格式顯示目前年份,區間爲0~99年.
YEAR[7:4]: 此4位代表年之十位.
YAER[3:0]: 此4位代表年之個位.

Mnemonic: ALMMIN **Address: 08H**

7	6	5	4	3	2	1	0	Reset
MINen	ALMMIN[6:4]			ALMMIN[3:0]				00H

MINen: 分示警啓動位元
0: 禁止; ALMMIN[6:0]填入值不需與RTC值比較,也不會產生示警中斷(ALARM INT).
1: 啓動; ALMMIN[6:0]填入值需與RTC值比較,並產生示警中斷(ALARM INT).
ALMMIN[6:0]: 以BCD格式設定示警中斷之分鐘數,區間爲0~59分鐘.
ALMMIN[6:4]: 此3位設定示警中斷分鐘之十位.
ALMMIN[3:0]: 此4位設定示警中斷分鐘之個位.

Mnemonic: ALMHOUR **Address: 09H**

7	6	5	4	3	2	1	0	Reset
HOUREn		ALMHOUR[5:4]		ALMHOUR[3:0]				00H

HOUREn: 時示警啓動位元
0: 禁止; ALMHOUR[5:0]填入值不需與RTC值比較,也不會產生示警中斷(ALARM INT).
1: 啓動; ALMHOUR[5:0]填入值需與RTC值比較,並產生示警中斷(ALARM INT).
ALMHOUR[5:0]: 以BCD格式設定示警中斷之時,區間爲0~23時.
ALMHOUR[5:4]: 此2位代表示警中斷時之十位.
ALMHOUR[3:0]: 此4位代表示警中斷時之個位.

Mnemonic: ALMDAY **Address: 10H**

7	6	5	4	3	2	1	0	Reset
DAYen	-	ALMDAY[5:4]		ALMDAY[3:0]				00H

DAYen: 日示警啓動位元
0: 禁止; ALMDAY[5:0]填入值不需與RTC值比較,也不會產生示警中斷(ALARM INT).
1: 啓動; ALMDAY[5:0]填入值需與RTC值比較,並產生示警中斷(ALARM INT).
ALMDAY[5:0] 以BCD格式設定示警中斷之日期,區間爲0~31日.
ALMDAY[5:4]: 此2位代表示警中斷日期之十位.
ALMDAY[3:0]: 此4位代表示警中斷日期之個位.

Mnemonic: ALMWEEK **Address: 11H**

7	6	5	4	3	2	1	0	Reset
WEEKen					ALMWEEK[2:0]			00H

WEEKen: 星期示警啓動位元
0: 禁止; ALMWEEK[2:0]填入值不需與RTC值比較,也不會產生示警中斷(ALARM INT).

1: 啓動; ALMWEEK[2:0]填入值需與RTC值比較,並產生示警中斷(ALARM INT).

ALMWEEK[2:0] 示警星期設定

000: 星期日
001: 星期一
010: 星期二
011: 星期三
100: 星期四
101: 星期五
110: 星期六

Mnemonic: ALMMONTH **Address: 12H**

7	6	5	4	3	2	1	0	Reset
MONTHen	-	-	ALMMONTH[4]	ALMMONTH[3:0]				00H

MONTHen: 月示警啓動位元

0: 禁能; ALMMONTH[4:0]填入值不需與RTC值比較,也不會產生示警中斷(ALARM INT).
1: 啓動; ALMMONTH[4:0]填入值需與RTC值比較,並產生示警中斷(ALARM INT).

ALMMONTH[4:0]: 以BCD格式設定示警中斷之月份,區間爲1~12月.
ALMMONTH[4]: 此位代表示警中斷月之十位.
ALMMONTH[3:0]: 此4位代表示警中斷月之個位.

4 RTC 中斷

4.1 中斷向量表(Interrupt vectors table)：

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
RTC/ALARM interrupt	***008Bh***	***17***

Comparator interrupt	0093h	18

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

4.2 中斷相關暫存器(Interrupt SFR)

Mnemonic	Description	Direct	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	RESET
Interrupt											
IEN0	Interrupt Enable 0 register	A8h	**EA**	-	ET2	ES0	ET1	EX1	ET0	EX0	00h
IEN1	Interrupt Enable 1 register	B8h	EXEN2	-	IEIIC	IELVI	IEKBI	IEADC	IESPI	IEPWM	00h
IEN2	Interrupt Enable 2 register	9Ah	-	-	-	-	-	ECmpI	**ERTC**	ES1	00h
IRCON	Interrupt request register	C0H	EXF2	TF2	IICIF	LVIIF	KBIIF	ADCIF	SPIIF	PWMIF	00H
IRCON2	Interrupt request register 2	97H	-	-	-	-	-	CmpIF	**RTCIF**	-	00H
IP0	Interrupt priority level 0	A9h	-	-	IP0.5	IP0.4	IP0.3	IP0.2	IP0.1	IP0.0	00h
IP1	Interrupt priority level 1	B9h	-	-	IP1.5	IP1.4	IP1.3	IP1.2	IP1.1	IP1.0	00h

Interrupt Enable 0 register(IEN0)

Mnemonic: IEN0 **Address: A8h**

7	**6**	**5**	4	3	2	1	0	Reset
EA	**-**	ET2	ES0	ET1	EX1	ET0	EX0	00h

EA: EA = 0 : disable all interrupt.

EA = 1 : enable all interrupt.

ET2: ET2 = 0 : 禁能 Timer 2 溢位中斷及 Timer 2 T2EX 外部中斷。

(disable Timer 2 overflow interrupt or Timer 2 external reload interrupt.)

ET2 = 1 : 致能 Timer 2 溢位中斷及 Timer 2 T2EX 外部中斷。

(enable Timer 2 overflow interrupt or Timer 2 external reload interrupt.)

ES0: ES0=0 – Disable Serial channel 0 interrupt.

ES0=1 – Enable Serial channel 0 interrupt.

ET1: ET1=0 – Disable Timer 1 overflow interrupt.
ET1=1 – Enable Timer 1 overflow interrupt.
EX1: EX1=0 – Disable external interrupt 1.
EX1=1 – Enable external interrupt 1.
ET0: ET0=0 – Disable Timer 0 overflow interrupt.
ET0=1 – Enable Timer 0 overflow interrupt.
EX0: EX0=0 – Disable external interrupt 0.
EX0=1 – Enable external interrupt 0.

Interrupt Enable 2 register(IEN2)

Mnemonic: IE2 **Address: 9Ah**

7	6	5	4	3	2	1	0	Reset
-	-	-	-	-	ECmpI	***ERTC***	ES1	00h

ECmpI: ECmpI =0 – Disable Comparator interrupt.
ECmpI =1 – Enable Comparator interrupt(include comparator_0 and comparator_1).
ERTC: ERTC =0 – Disable RTC interrupt.
ERTC =1 – Enable RTC interrupt(include Periodical and Alarm Int).
ES1: ES1=0 – Disable Serial channel 1 interrupt.
ES1=1 – Enable Serial channel 1 interrupt.

Interrupt request register 2(IRCON2)

Mnemonic: Address: 97h

7	6	5	4	3	2	1	0	Reset
-	-	-	-	-	CmpIF	**RTCIF**	-	00H

RTCIF RTC interrupt flag(include Periodical and Alarm Int).
HW will clear this flag automatically when enter interrupt vector.
SW can clear this flag also.(in case RTC INT disable)
CmpIF Comparator interrupt flag,
HW will clear this flag automatically when enter interrupt vector.
SW can clear this flag also.(in case analog comparator INT disable)

5 Timer 2 中斷應用參考程序及設定：

(1) 中斷致能設定：

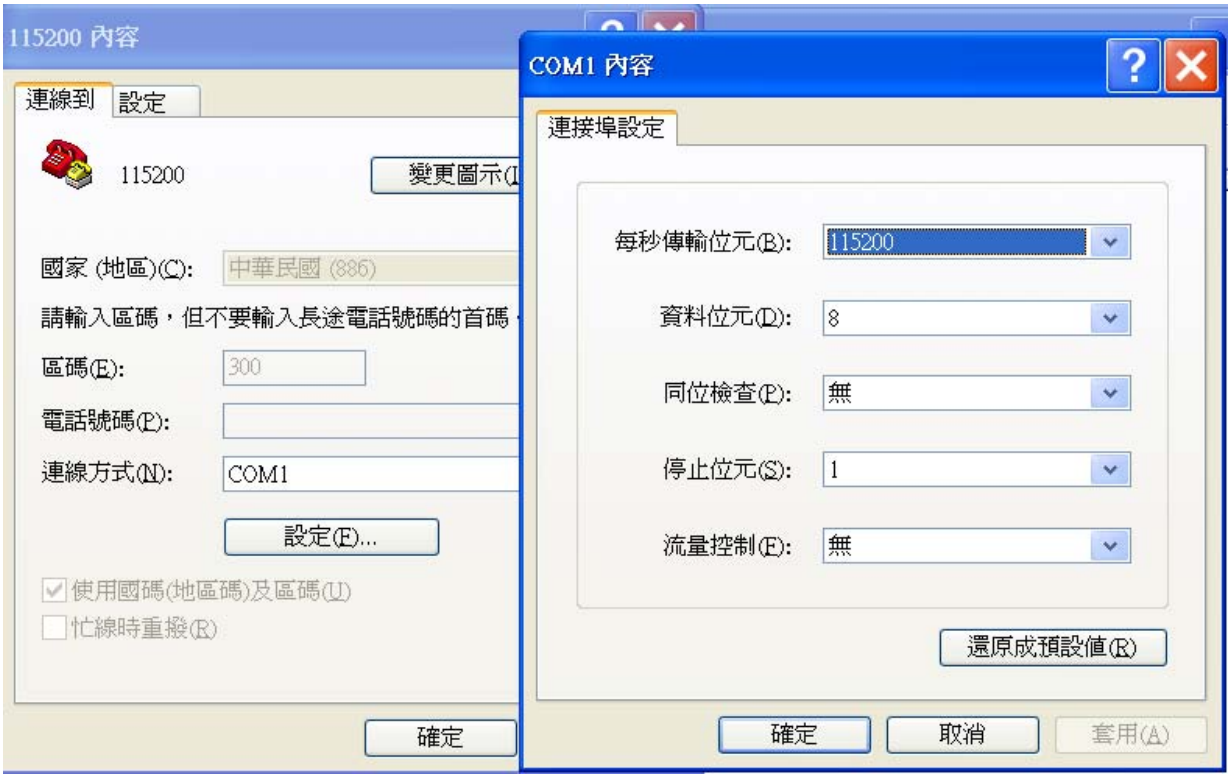
```
IEN2    |= 0x02;    // Enable RTC_INT.
EA      = 1;       // All interrupt enable
RTCADDR = RTCCTRL;
RTCDATA = 0xA8;    // RTCen=1, ALARMInten=1, PeriodIntEn=1, Period=0.5 sec
```

(2) 中斷程序表示：

```
void ISR_RTC(void) interrupt 17
{
    RTCADDR = RTCCTRL;
    if(RTCDATA & PeriodIF)
    {
        // Must clear by software write "0".
        RTCADDR = RTCCTRL;      RTCDATA &= (~PeriodIF);
    }
    else if(RTCDATA & ALARMIF)
    {
        // Must clear by software write "0".
        RTCADDR = RTCCTRL;      RTCDATA &= (~ALARMIF);
    }
    // HW will clear this flag automatically.
    //IRCON2 &= (~RTCIF);
}
```

6 RTC 範例程式說明:

6.1 此範例可使用超機終端機做通訊，並依據操作說明輸入選項及設定即可，超機終端機請設定如下：



7 RTC 應用的範例程式(一)

Description	
Define	//以下定義已加入 SM59R16A5.h /* RTC Define */ #define RTCCTRL 0 #define RTCSEC 1 #define RTCMIN 2 #define RTCHOUR 3 #define RTCDAY 4 #define RTCWEEK 5 #define RTCMONTH 6 #define RTCYEAR 7 #define RTCALMMIN 8 #define RTCALMHOUR 9 #define RTCALMDAY 10 #define RTCALMWEEK 11 #define RTCALMMONTH 12 #define RTCen 0x80 #define PeriodIF 0x40 #define ALARMInten 0x20 #define ALARMIF 0x10 #define PeriodIntEn 0x08

	`#define RTCIF 0X02`
	
	`#define RTC_half_sec 0`
	`#define RTC_one_sec 1`
	`#define RTC_half_min 2`
	`#define RTC_one_min 3`
	`#define RTC_half_hour 4`
	`#define RTC_one_hour 5`

Main program	`#include <stdio.h>`
	`#include <SM59R16A5.h>`
	
	`xdata char CMD_001=0, CMD_002=0;`
	`xdata char RTC_BFU[13];`
	`xdata char RTC_Period_target=5;`
	`xdata char RTC_Period_count=0;`
	
	`#define Pause_RTC_INT() IEN2 &= 0xFD; // Disable RTC_INT.`
	`#define Gono_RTC_INT() IEN2 |= 0x02; // Enable RTC_INT.`
	
	`/*void delay(int dd)`
	`{`
	`while(dd--);`
	`} */`
	
	`void Reset_MCU(void)`
	`{`
	`TAKEY = 0x55;`
	`TAKEY = 0xAA;`
	`TAKEY = 0x5A;`
	`SWRES = 0xFF;`
	`}`
	`void Show_BCD(char dat)`
	`{`
	`putchar((dat>>4) | 0x30);`
	`putchar((dat&0x0F) | 0x30);`
	`}`
	
	`void Show_Manual(void)`
	`{`
	`printf ("\n");`
	`printf ("SM59R16A5 RTC manual: \n");`
	`printf ("1. RTC now \n");`
	`printf ("2. SET RTC time\n");`
	`printf ("3. SET Period \n");`
	`printf ("4. ON/OFF Period \n");`
	`printf ("5. SET ALARM \n");`
	`printf ("6. ON/OFF ALARM \n");`
	`printf ("Other. System Reset!! \n");`
	`printf ("Enter:");`
	`}`
	
	`void Write_RTC_SFR(char addr, char dat)`
	`{`
	`RTCADDR = addr; RTCDATA = dat;`
	`}`
	
	`char Read_RTC_SFR(char addr)`

```
{
    char dat;
    RTCADDR = addr; dat = RTCDATA;
    return dat;
}

void Read_RTC_Now_Time(void)
{
    RTC_BFU[RTCYEAR]    = Read_RTC_SFR(RTCYEAR);
    RTC_BFU[RTCMONTH]   = Read_RTC_SFR(RTCMONTH);
    RTC_BFU[RTCDAY]     = Read_RTC_SFR(RTCDAY);
    RTC_BFU[RTCWEEK]    = Read_RTC_SFR(RTCWEEK);
    RTC_BFU[RTCHOUR]    = Read_RTC_SFR(RTCHOUR);
    RTC_BFU[RTCMIN]     = Read_RTC_SFR(RTCMIN);
    RTC_BFU[RTCSEC]     = Read_RTC_SFR(RTCSEC);
}

void Show_RTC_Now_Time(void)
{
    printf("RTC Now:\t");
    Read_RTC_Now_Time();

    Show_BCD( 0x20 );               // year
    Show_BCD( RTC_BFU[RTCYEAR] );
    putchar(0x2F);                  // /
    Show_BCD( RTC_BFU[RTCMONTH] );  // month
    putchar(0x2F);                  // /
    Show_BCD( RTC_BFU[RTCDAY] );    // day
    putchar(0x20);                  // SPACE

    putchar(0x28);                  // (
    Show_BCD( RTC_BFU[RTCWEEK] );   // week
    putchar(0x29);                  // )
    printf("\t");                   // tab

    Show_BCD( RTC_BFU[RTCHOUR] );   // hour
    putchar(0x3A);                  // :
    Show_BCD( RTC_BFU[RTCMIN] );    // min
    putchar(0x3A);                  // :
    Show_BCD( RTC_BFU[RTCSEC] );    // sec
    printf("\n");
}

void Show_RTC_ALARM_Set(void)
{
    RTCADDR = RTCCTRL;
    if(RTCDATA&0x20)    printf("ALARM Status:\tON\n");
    else                printf("ALARM Status:\tOFF\n");

    printf("ALARM Setting:\tYYYY/MM/DD(week)hr:min\n");

    printf("\t\t");     Show_BCD( 0x20 );           // year
    RTCADDR = RTCYEAR;  Show_BCD( RTCDATA );
    putchar(0x2F);                                  // /

    Show_BCD(Read_RTC_SFR(RTCALMMONTH) & 0x1F);     // month
    putchar(0x2F);                                  // /
```

```
    Show_BCD(Read_RTC_SFR(RTCALMDAY) & 0x3F );      // day
    putchar(0x20);                                  // SPACE

    putchar(0x28);                                  // (
    Show_BCD(Read_RTC_SFR(RTCALMWEEK) & 0x07 );     // week
    putchar(0x29);                                  // )
    printf("\t");                                   // tab

    Show_BCD(Read_RTC_SFR(RTCALMHOUR) & 0x3F );     // hour
    putchar(0x3A);                                  // :
    Show_BCD(Read_RTC_SFR(RTCALMMIN) & 0x07 );      // min
    printf("\n");
}

void SET_Period(void)
{
//  char h_dat, l_dat;
    Show_RTC_Now_Time();
    Show_RTC_ALARM_Set();

    printf ("\t");  printf ("1. 01 sec \n");
    printf ("\t");  printf ("2. 05 sec \n");
    printf ("\t");  printf ("3. 10 sec \n");
    printf ("\t");  printf ("4. 20 sec \n");
    printf ("\t");  printf ("5. 30 sec \n");
    printf ("\t");  printf ("6. 60 sec \n");
    printf ("\t");  printf ("other. Exit!! \n");
    printf ("\t");  printf ("Enter:");

    CMD_002 = getchar();

    switch (CMD_002)
    {
        case 0x31:  RTC_Period_target= 1;// 01 sec
            break;
        case 0x32:  RTC_Period_target= 5;// 05 sec
            break;
        case 0x33:  RTC_Period_target=10;// 10 sec
            break;
        case 0x34:  RTC_Period_target=20;// 20 sec
            break;
        case 0x35:  RTC_Period_target=30;// 30 sec
            break;
        case 0x36:  RTC_Period_target=60;// 60 sec
            break;
        default:
            break;
    }// end switch
    printf ("\n");
}

void ON_OFF_Period(void)
{
    printf ("\t");  printf ("1.    Set period interrupt ON  \n");
    printf ("\t");  printf ("other. Set period interrupt OFF \n");
    printf ("\t");  printf ("Enter:");
```

```
    CMD_002 = getchar();    printf ("\n");
    RTCADDR = RTCCTRL;      RTCDATA &= 0xF7;    // PeriodIntEn=0
    if(CMD_002==0x31)       RTCDATA |= 0x08;    // PeriodIntEn=1
}

void ON_OFF_ALARM(void)
{
    printf ("\t");  printf ("1.    Set ALARM ON  \n");
    printf ("\t");  printf ("other. Set ALARM OFF \n");
    printf ("\t");  printf ("Enter:");

    CMD_002 = getchar();    printf ("\n");
    RTCADDR = RTCCTRL;      RTCDATA &= 0xDF;    // ALARMIntEn=0
    if(CMD_002==0x31)       RTCDATA |= 0x20;    // ALARMIntEn=1
}

void SET_RTC_Time(void)
{
    char h_dat, l_dat;

    while(1)
    {
        Show_RTC_Now_Time();

        printf ("\t");  printf ("<Which one is your choice?>\n");
        printf ("\t");  printf ("1. Year  \n");
        printf ("\t");  printf ("2. Month  \n");
        printf ("\t");  printf ("3. Day    \n");
        printf ("\t");  printf ("4. Week   \n");
        printf ("\t");  printf ("5. Hour   \n");
        printf ("\t");  printf ("6. Minute \n");
        printf ("\t");  printf ("7. Second \n");
        printf ("\t");  printf ("other. Exit!!\n");
        printf ("\t");  printf ("Enter:");

        CMD_002 = getchar();    printf ("\n");
        if(CMD_002<0x31 | CMD_002>0x37) break;

        switch (CMD_002)
        {
            case 0x31:
                printf ("\tYear:");
                h_dat   = getchar()-0x30;
                l_dat   = getchar()-0x30;
                Write_RTC_SFR(RTCYEAR, (h_dat<<4) | l_dat );
                printf ("\n");
                break;
            case 0x32:
                printf ("\tMonth:");
                h_dat   = getchar()-0x30;
                l_dat   = getchar()-0x30;
                Write_RTC_SFR(RTCMONTH, (h_dat<<4) | l_dat );
                printf ("\n");
                break;
            case 0x33:
                printf ("\tDay:\n");
                h_dat   = getchar()-0x30;
```

```
                l_dat   = getchar()-0x30;
                Write_RTC_SFR(RTCDAY, (h_dat<<4) | l_dat );
                printf ("\n");
                break;
            case 0x34:
                printf ("\tWeek:");
                Write_RTC_SFR(RTCWEEK, (getchar()-0x30) | 0x80);
                printf ("\n");
                break;
            case 0x35:
                printf ("\tHour:");
                h_dat   = getchar()-0x30;
                l_dat   = getchar()-0x30;
                Write_RTC_SFR(RTCHOUR, (h_dat<<4) | l_dat | 0x80);
                printf ("\n");
                break;
            case 0x36:
                printf ("\tMinute:");
                h_dat   = getchar()-0x30;
                l_dat   = getchar()-0x30;
                Write_RTC_SFR(RTCMIN, (h_dat<<4) | l_dat | 0x80);
                printf ("\n");
            case 0x37:
                printf ("\tSecond:");
                h_dat   = getchar()-0x30;
                l_dat   = getchar()-0x30;
                Write_RTC_SFR(RTCSEC, (h_dat<<4) | l_dat | 0x80);
                printf ("\n");
                break;
            default:
                break;
        }// end switch
    }// end while(1)
}

void SET_ALARM(void)
{
    char h_dat, l_dat;

    while(1)
    {
        Show_RTC_Now_Time();
        Show_RTC_ALARM_Set();
        printf ("\t");  printf ("<Which one is your choice?>\n");
        printf ("\t");  printf ("1. Month  \n");
        printf ("\t");  printf ("2. Day    \n");
        printf ("\t");  printf ("3. Week   \n");
        printf ("\t");  printf ("4. Hour   \n");
        printf ("\t");  printf ("5. Minute \n");
        printf ("\t");  printf ("Other. Exit!!\n");
        printf ("\t");  printf ("Enter:");

        CMD_002 = getchar();    printf ("\n");
        if(CMD_002<0x31 | CMD_002>0x35) break;

        switch (CMD_002)
        {
```

```
            case 0x31:
                printf ("\tMonth:");
                h_dat   = getchar()-0x30;
                l_dat   = getchar()-0x30;
                Write_RTC_SFR(RTCALMMONTH, (h_dat<<4) | l_dat | 0x80);
                printf ("\n");
                break;
            case 0x32:
                printf ("\tDay:");
                h_dat   = getchar()-0x30;
                l_dat   = getchar()-0x30;
                Write_RTC_SFR(RTCALMDAY, (h_dat<<4) | l_dat | 0x80);
                printf ("\n");
                break;
            case 0x33:
                printf ("\tWeek:");
                Write_RTC_SFR(RTCALMWEEK, (getchar()-0x30) | 0x80);
                printf ("\n");
                break;
            case 0x34:
                printf ("\tHour:");
                h_dat   = getchar()-0x30;
                l_dat   = getchar()-0x30;
                Write_RTC_SFR(RTCALMHOUR, (h_dat<<4) | l_dat | 0x80);
                printf ("\n");
                break;
            case 0x35:
                printf ("\tMinute:");
                h_dat   = getchar()-0x30;
                l_dat   = getchar()-0x30;
                Write_RTC_SFR(RTCALMMIN, (h_dat<<4) | l_dat | 0x80);
                printf ("\n");
                break;
            default:
                break;
        }// end switch
    }// end while(1)
}

void Init_UART(void)
{
    SCON = 0x50;        //SCON: serail mode 1, 8-bit UART, enable receive

   TMOD |= 0x20;      /* TMOD: timer 1, mode 2, 8-bit reload       */
    PCON = 0x80;        //SMOD = 1;
//  TH1 =   0xD0;       //Baud:2400 fosc=22.1184MHz     OK!
//  TH1 = 0xFE;     //Baud:57600 fosc=22.1184MHz    OK!
    TH1 = 0xFF;     //Baud:115200 fosc=22.1184MHz   OK!
//  TH1 = 0xFF;     //Baud:57600 fosc=11.0592MHz    OK!
    TCON= 0x40;         //timer 1 run
    TI    = 1;        /*TI:   set TI to send first char of UART   for
Printf("")*/
//  EA = 1;             //all interrupt enable
//  ES = 1;             //UART enable
}

void Init_RTC(void)
```

```
{
    IEN2    |= 0x02;    // Enable RTC_INT.
//  IEN2    &= 0xFD;    // Disable RTC_INT.
    EA      = 1;        // All interrupt enable
//  RTCADDR = RTCCTRL;
//  RTCDATA = 0x81;     // RTCen=1, ALARMInten=0, PeriodIntEn=0, Period=  1
sec
    Write_RTC_SFR(RTCCTRL, RTCen | RTC_one_sec );

    Write_RTC_SFR( RTCYEAR,     (0x10));
    Write_RTC_SFR( RTCMONTH,    (0x05));
    Write_RTC_SFR( RTCDAY,      (0x01));
    Write_RTC_SFR( RTCWEEK,     (0x06));
    Write_RTC_SFR( RTCALMMONTH, (0x05));
    Write_RTC_SFR( RTCALMDAY,   (0x01));
    Write_RTC_SFR( RTCALMWEEK,  (0x06));
}

void ISR_RTC(void) interrupt 17
{
    RTCADDR = RTCCTRL;
    if(RTCDATA & PeriodIF)
    {
        RTC_Period_count++;
        if(RTC_Period_count >= RTC_Period_target)
        {
            printf ("\n<RTC Period ISR occur> ");
            Show_RTC_Now_Time();
            RTC_Period_count =0;
        }
        RTCADDR = RTCCTRL;
        RTCDATA &= (~PeriodIF);     // Must clear by software write "0".
    }
//  else if(RTCDATA & ALARMIF)
    if(RTCDATA & ALARMIF)
    {
        printf("\n<RTC ALARM ISR occur>");
        Show_RTC_Now_Time();
        Show_Manual();
        RTCADDR = RTCCTRL;
        RTCDATA &= (~ALARMIF);      // Must clear by software write "0".
    }
    //IRCON2 &= (~RTCIF); //RTCIF=0;// HW will clear this flag automatically.
}

void main(void)
{
    Init_UART();
    putchar(0x31);  printf("\n");
    putchar(0x32);  printf("\n");
    putchar(0x33);  printf("\n");

    Init_RTC();
    Show_RTC_Now_Time();
    Show_RTC_ALARM_Set();

    while(1)
```

```
    {

        Show_Manual();
        CMD_001 = getchar();    printf ("\n\n");

        Pause_RTC_INT();
        switch (CMD_001)
        {
            case 0x31:
                Show_RTC_Now_Time();    break;

            case 0x32:
                SET_RTC_Time();         break;

            case 0x33:
                SET_Period();           break;

            case 0x34:
                ON_OFF_Period();        break;

            case 0x35:
                SET_ALARM();            break;

            case 0x36:
                ON_OFF_ALARM();         break;

            default:
                Reset_MCU();            break;

        }//end switch
        //printf("\n");
        CMD_001=0;
        Gono_RTC_INT();
    }
}
```