時鐘
RTC

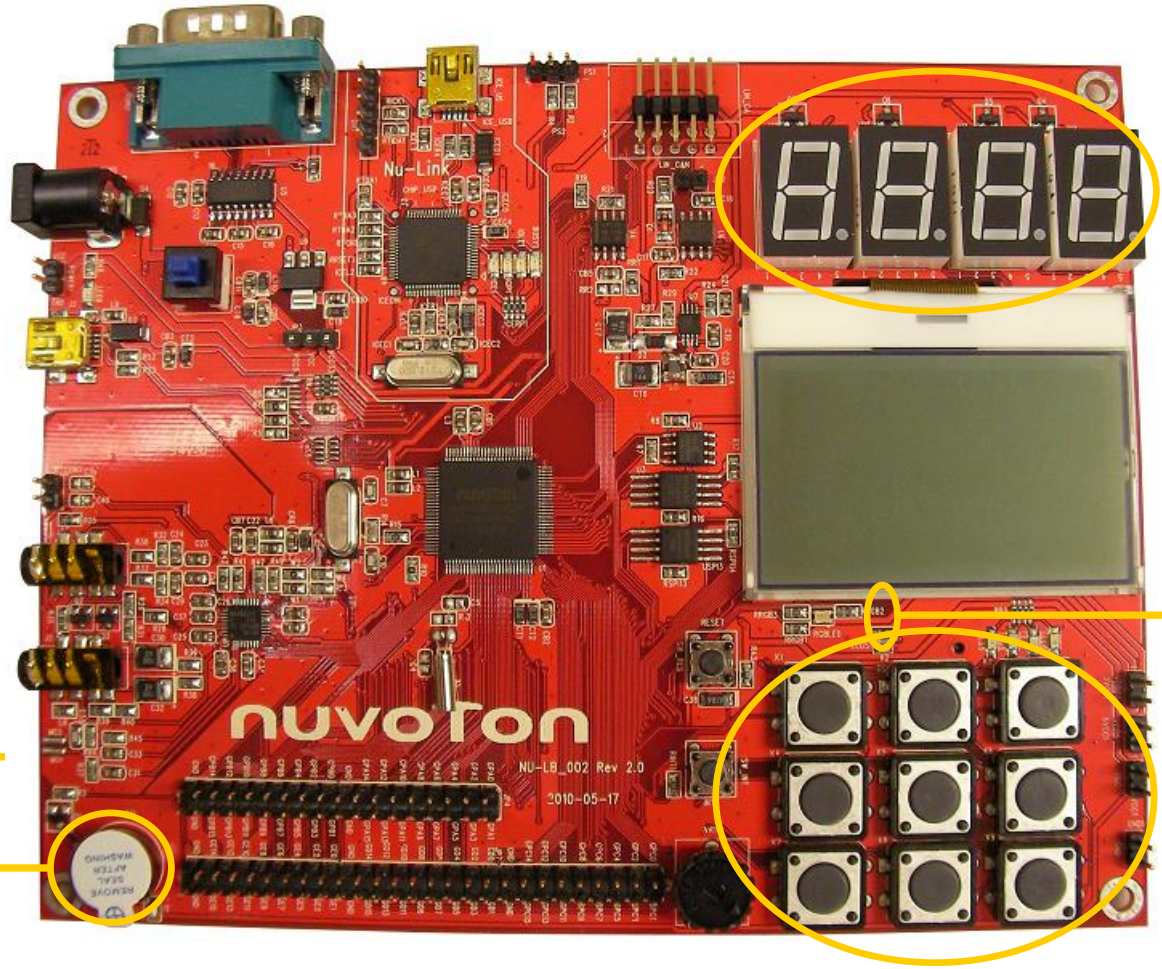
- 2013/4/8

課程大綱 (Lesson Outline)

- **Cortex-M0 MCU 計時時鐘 設計原理**
  - 時鐘(RTC)
- **實習範例 : Test_RTC_RGB**

- **範例練習 : RTC計時時鐘**
  - 利用範例修改出使用RTC時鐘計時之應用

NU-LB-NUC140 開發板



7 segment LEDs

RGB LED
GPA12,13,14

Keypad

SW Int
GPB15

Buzzer
GPB11

RTC Features

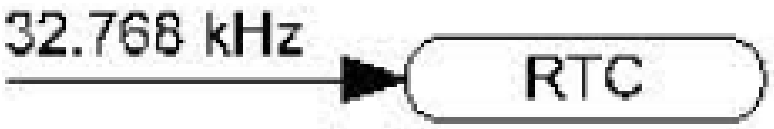
- RTC counter
  - Time counter (hour, minute, second)
    TLR=093015 for 09:30:15
  - Calendar counter (year , month, day)
    CLR=130301 for 2013/03/01
  - Day of week
  - Leap year
  - 12-hour or 24-hour mode
- A set of Alarm
  - TAR=093025 for 09:30:25
  - CAR=130301 for 2013/03/01

RTC Features

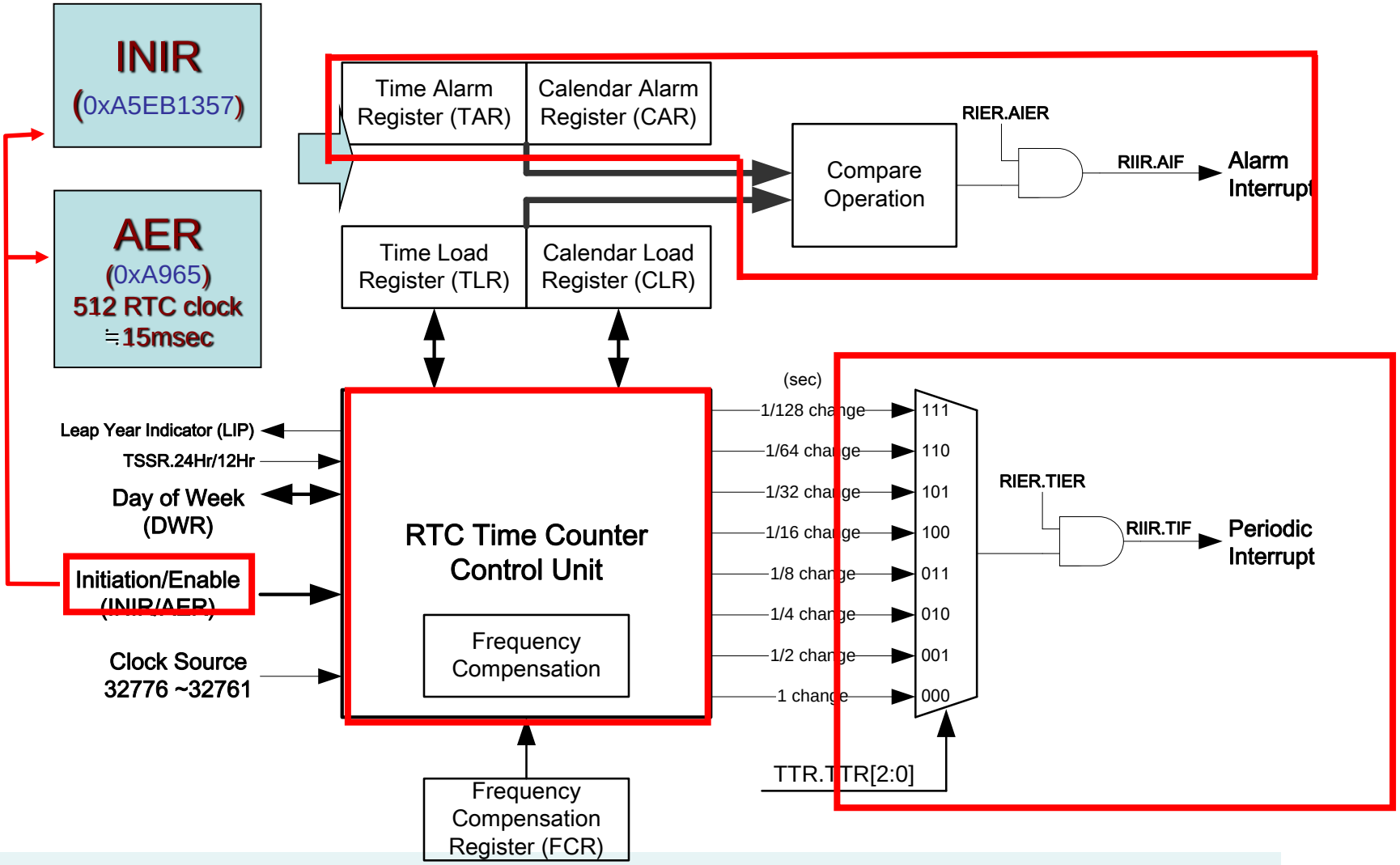
- 8 period time tick interrupt: 1, ½ , ¼ , 1/8, 1/16, 1/32, 1/64, 1/128
- Alarm and time tick interrupt
- Wakeup function
- 32KHz compensation

RTC Clock Source

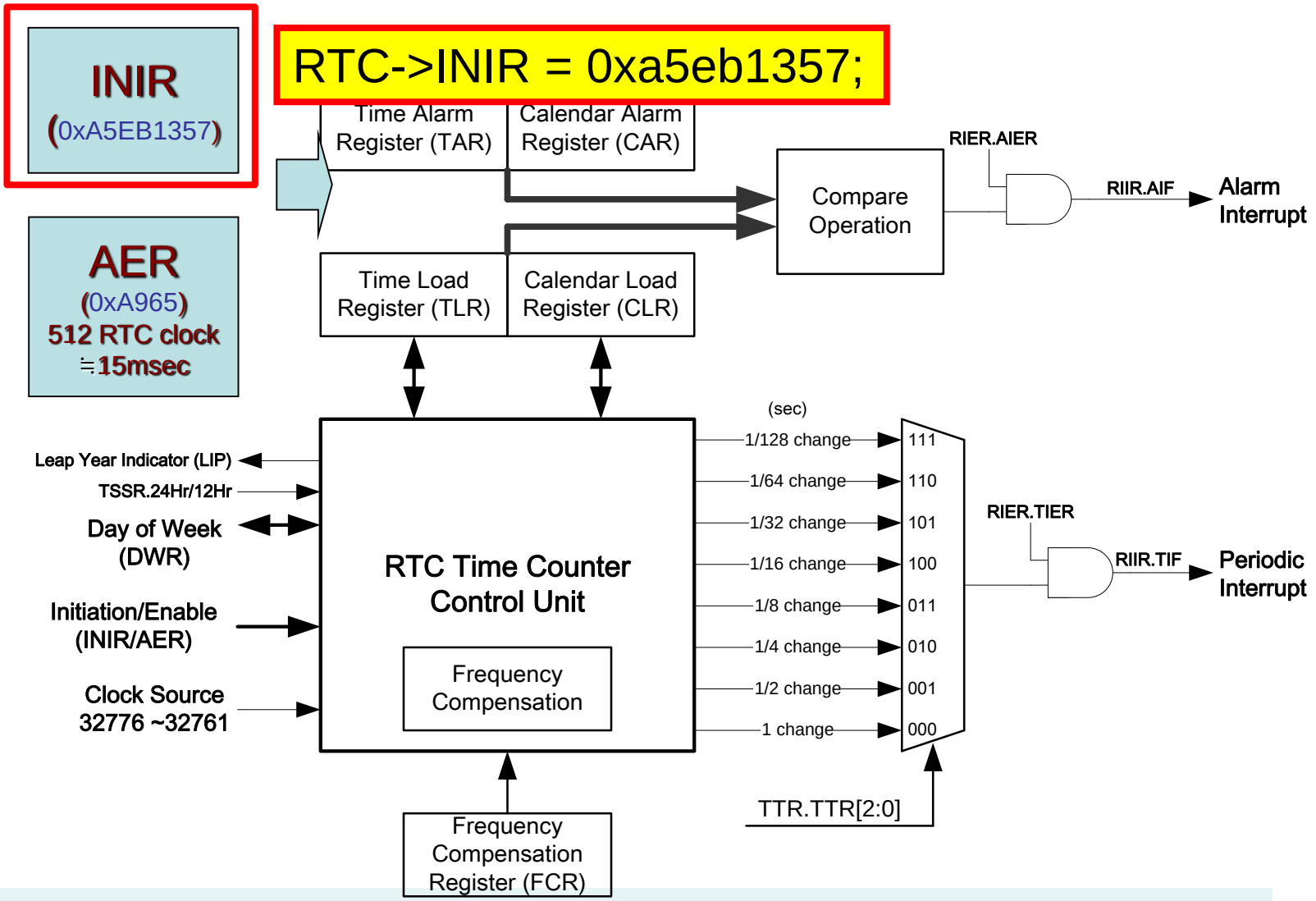
```
//: Enable 32Khz clock source for RTC
SYSCLK->PWRCON.XTL32K_EN = 1;
//: Enable RTC clock source
SYSCLK->APBCLK.RTC_EN =1;
```



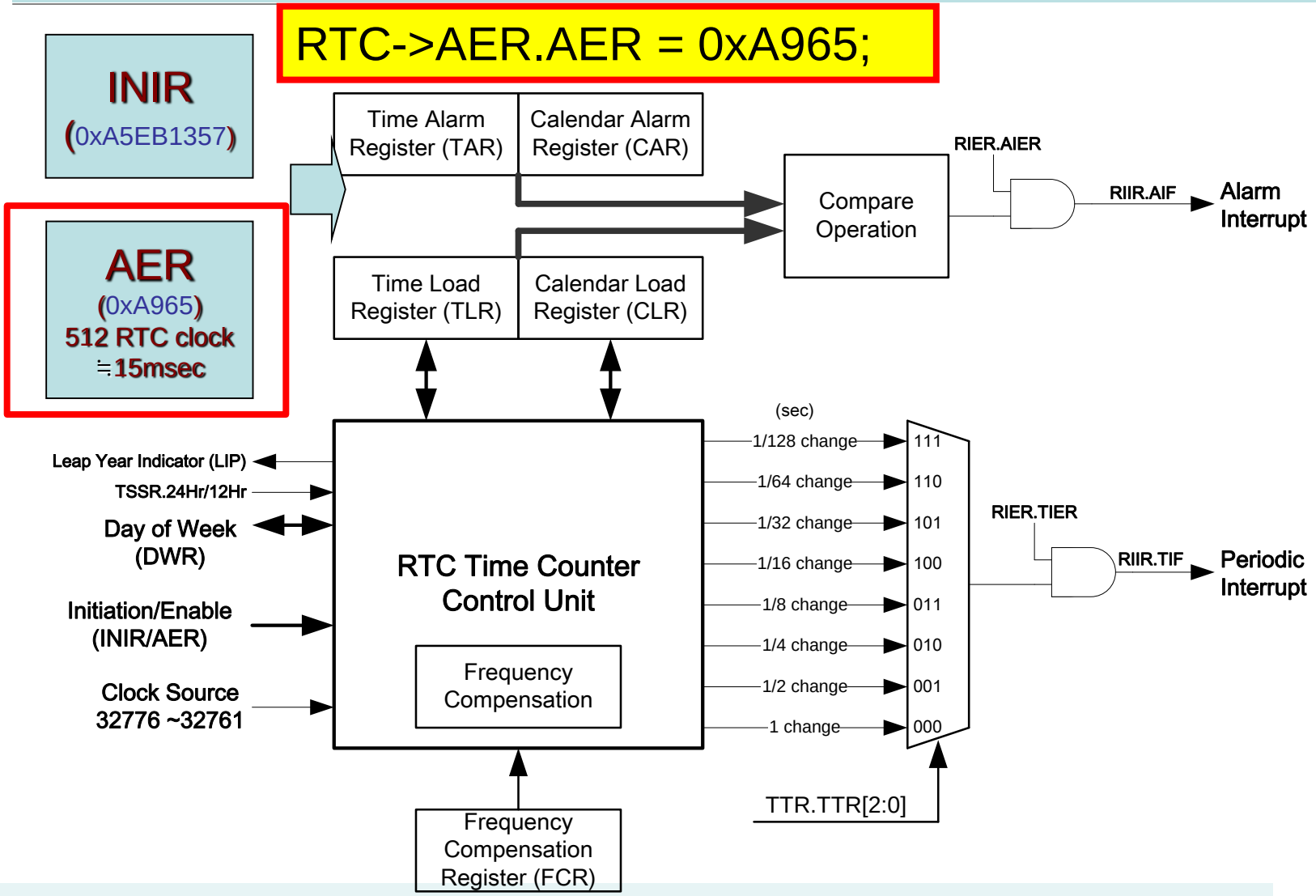
RTC Block Diagram



Set RTC active state (unreset)



Set RTC access enable



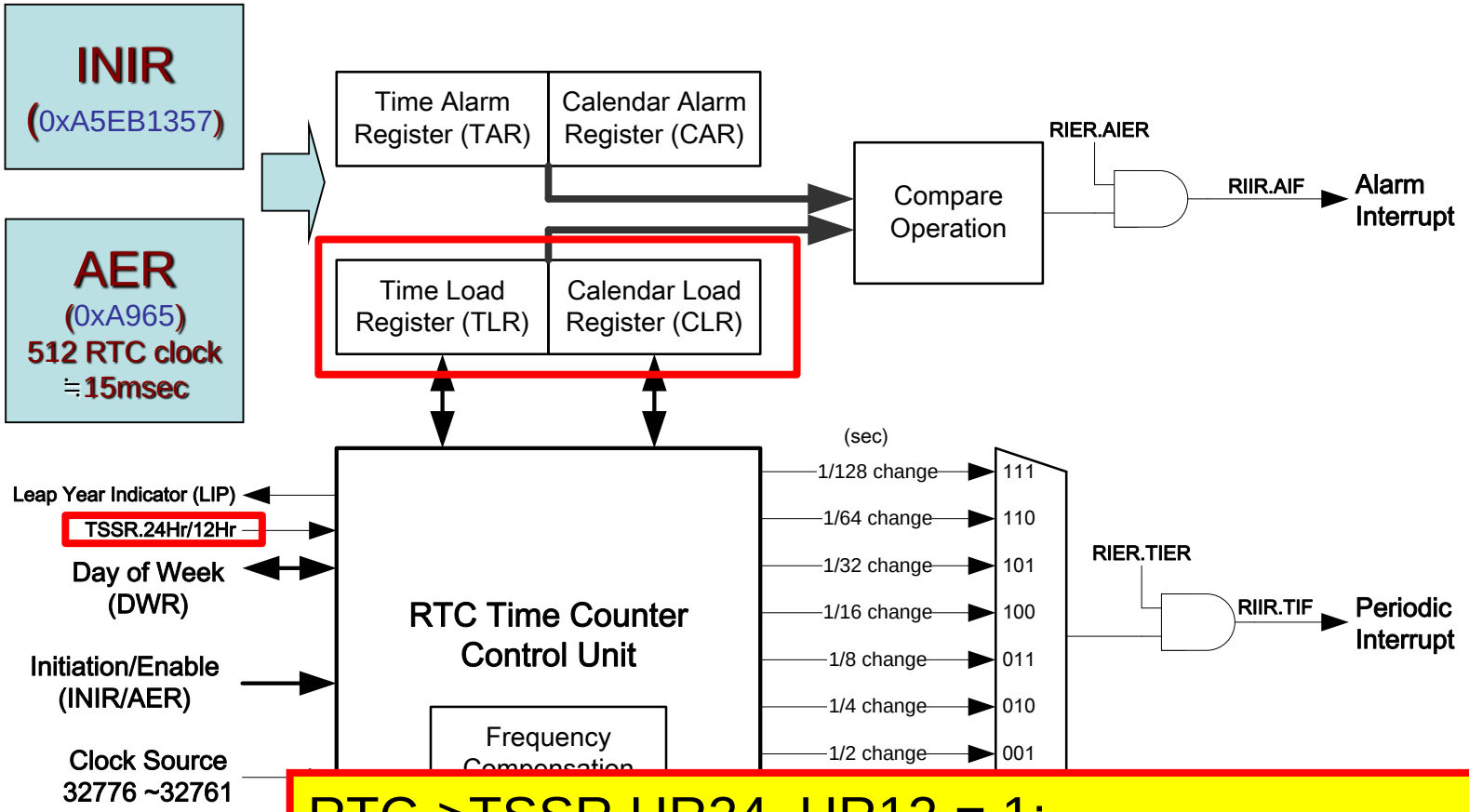
RTC Access

Register	Offset	R/W	Description	Reset Value	AER[16]=0
RTC_BA = 0x4000_8000					
INIR	RTC_BA+0x000	R/W	RTC Initiation Register(A5EB1357)	0x0000_0000	R/W
AER	RTC_BA+0x004	R/W	RTC Access Enable Register(A965)	0x0000_0000	R/W
FCR	RTC_BA+0x008	R/W	RTC Frequency Compensation Register	0x0000_0700	-
TLR	RTC_BA+0x00C	R/W	Time Loading Register	0x0000_0000	R
CLR	RTC_BA+0x010	R/W	Calendar Loading Register	0x0005_0101	R
TSSR	RTC_BA+0x014	R/W	Time Scale Selection Register	0x0000_0001	R/W
DWR	RTC_BA+0x018	R/W	Day of the Week Register	0x0000_0006	R
TAR	RTC_BA+0x01C	R/W	Time Alarm Register	0x0000_0000	-
CAR	RTC_BA+0x020	R/W	Calendar Alarm Register	0x0000_0000	-
LIR	RTC_BA+0x024	R	Leap year Indicator Register	0x0000_0000	R
RIER	RTC_BA+0x028	R/W	RTC Interrupt Enable Register	0x0000_0000	R/W
RIIR	RTC_BA+0x02C	R/C	RTC Interrupt Indicator Register	0x0000_0000	R/C
TTR	RTC_BA+0x030	R/W	RTC Time Tick Register	0x0000_0000	-

Access enable

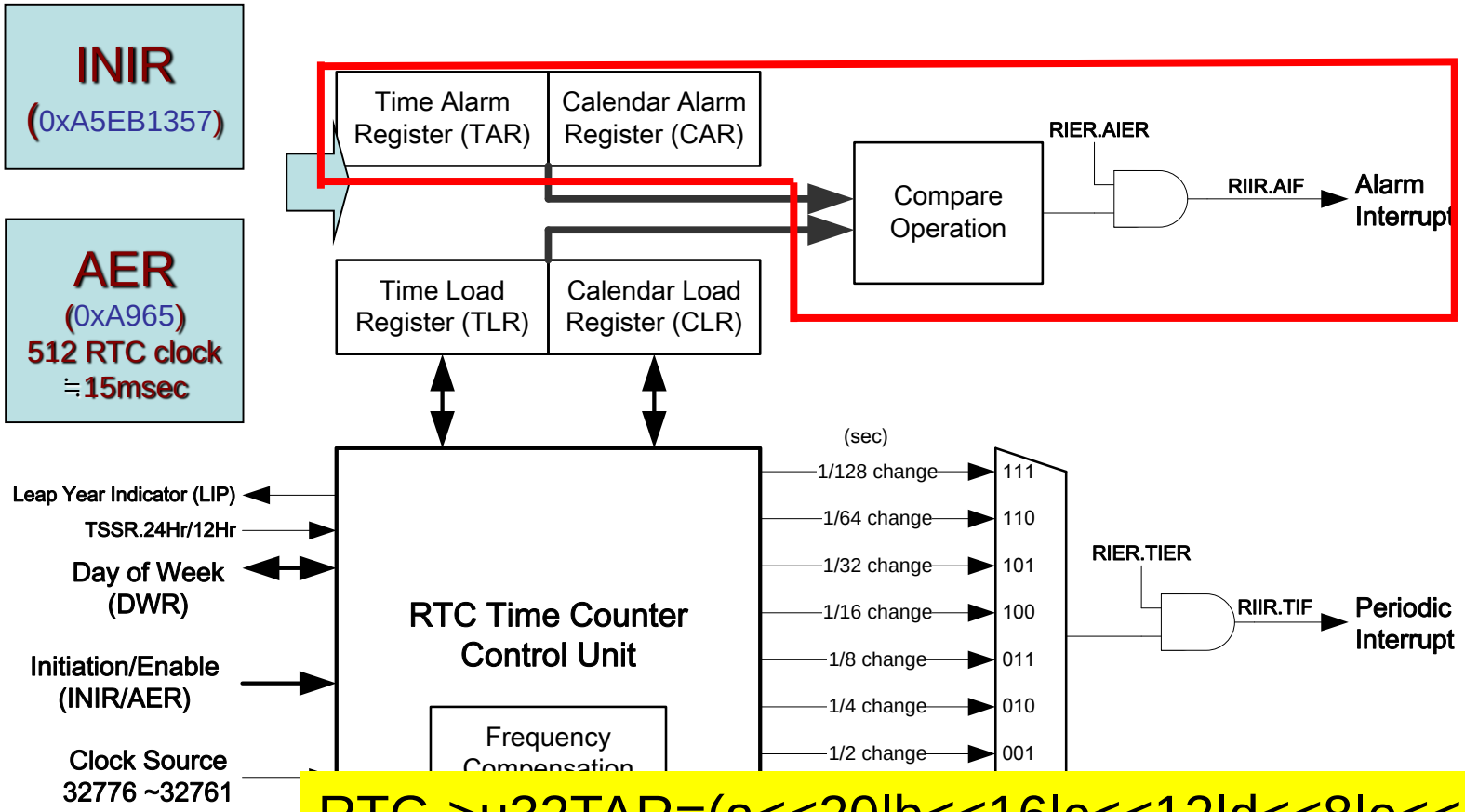
Access disable

Set TLR, CLR, 24h/12hr



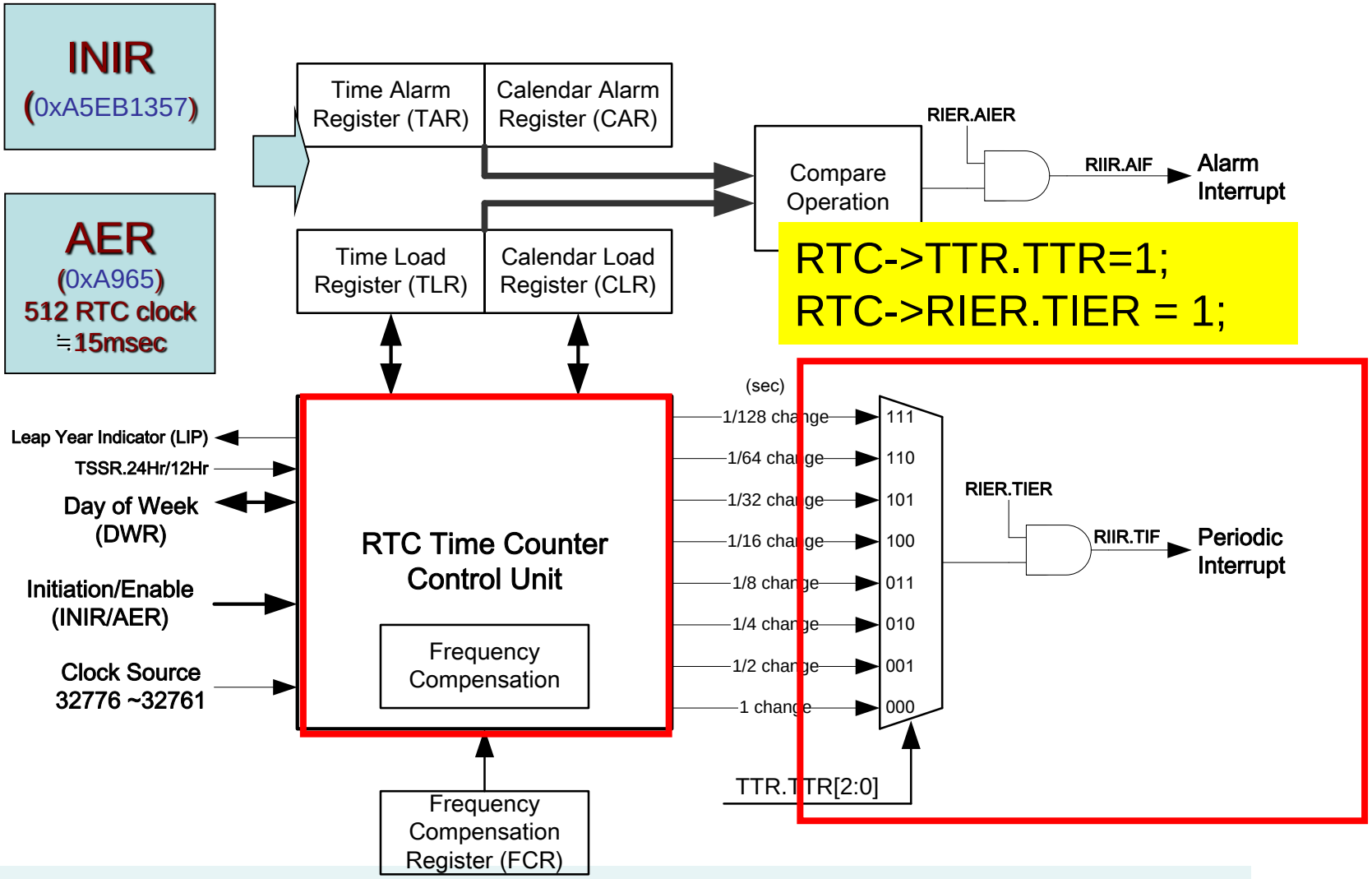

```
RTC->TSSR.HR24_HR12 = 1;
RTC->u32TLR=(a<<20|b<<16|c<<12|d<<8|e<<4|f);
RTC->u32CLR=(a<<20|b<<16|c<<12|d<<8|e<<4|f);
```

RTC Block Diagram




```
RTC->u32TAR=(a<<20|b<<16|c<<12|d<<8|e<<4|f);
RTC->u32CAR=(a<<20|b<<16|c<<12|d<<8|e<<4|f);
RTC->RIER.AIER = 1;
```

RTC Block Diagram



RTC Driver

DrvRTC_SetFrequencyCompensation	Set Frequency Compensation Data
DrvRTC_IsLeapYear	According to current time , return this year is leap year or not.
DrvRTC_GetIntTick	The function is used to get current Software tick count after enable tick interrupt.
DrvRTC_ResetIntTick	The function is used to reset the tick count counting in interrupt.
DrvRTC_WriteEnable	Access Password to AER to make access other register enable
DrvRTC_Init	Initial RTC. It consists of clear callback function pointer, enable 32K clock and RTC clock and write initial key to let RTC start count.

RTC Driver

DrvRTC_SetTickMode	The function is used to set time tick period for periodic time tick Interrupt.
DrvRTC_EnableInt	The function is used to enable specified interrupt and install callback function..
DrvRTC_DisableInt	The function is used to disable specified interrupt and remove callback function.
DrvRTC_Open	Set Current time (Year/Month/Day, Hour/Minute/Sec and day of week)
DrvRTC_Read	Read current date/time or alarm date/time from RTC setting
DrvRTC_Write	Set current date/time or alarm date/time to RTC
DrvRTC_Close	Disable NVIC channel of RTC and both tick and alarm interrupt..
DrvRTC_GetVersion	Return the current version number of driver.

24-hour/12-hour Mapping Table

BCD Format

24-hour time scale	12-hour time scale	24-hour time scale	12-hour time scale (PM time + 0x20)
0x00	0x12(AM12)	0x12	0x32(PM12)
0x01	0x01(AM01)	0x13	0x21(PM01)
0x02	0x02(AM02)	0x14	0x22(PM02)
0x03	0x03(AM03)	0x15	0x23(PM03)
0x04	0x04(AM04)	0x16	0x24(PM04)
0x05	0x05(AM05)	0x17	0x25(PM05)
0x06	0x06(AM06)	0x18	0x26(PM06)
0x07	0x07(AM07)	0x19	0x27(PM07)
0x08	0x08(AM08)	0x20	0x28(PM08)
0x09	0x09(AM09)	0x21	0x29(PM09)
0x10	0x10(AM10)	0x22	0x30(PM10)
0x11	0x11(AM11)	0x23	0x31(PM11)

=AM time

=PM time+0x20

RTC Example

- RTC sample code with driver
  - Enable RTC and set date and time
    - Date: 2013.1.19
    - Time:13.20.00
  - Enable alarm interrupt and set alarm date and time
    - Alarm date:2013.1.19
    - Alarm Time:13.20.10
- RTC sample code
  - The same flow

RTC初始化(RTC Initial Routine) 1/x

```
void InitRTC(void)
{
        UNLOCKREG();
        //: Enable 32Khz clock source for RTC
        SYSCLK->PWRCON.XTL32K_EN = 1;
        //: Enable RTC clock source
        SYSCLK->APBCLK.RTC_EN =1;
         //: set RTC active (unreset) state
        RTCActiveState();
        //: enable RTC access
        RTCAccessEnable();
```

RTC初始化(RTC Initial Routine) 2/x

```
//: Set 24hour mode
RTC->TSSR.HR24_HR12 = 1;
//: set date YY/MM/DD=2013/01/25
set_CLR(1,3,0,1,2,6); //:
//: set time hh:mm:ss=08:12:30
set_TLR(0,8,1,2,3,1); //:
//: set alarm date YY/MM/DD=2013/01/25
set_CAR(1,3,0,1,2,6); //:
//: set alarm time hh:mm:ss=08:12:43
set_TAR(0,8,1,2,3,6); //:
```

RTC初始化(RTC Initial Routine) 3/x

```
    //: RTC alarm Interrupt enable
    RTC->RIER.AIER = 1;
    //: set time tick period, 1/(2^1)
    RTC->TTR.TTR=1;
    //: RTC time tick Interrupt enable
    RTC->RIER.TIER = 1;
     //: (core_cm0.h)RTC IRQ enable
    NVIC_EnableIRQ(RTC_IRQn);
}
```

RTC初始化(RTC Initial Routine) 4/x

```
void RTCActiveState (void)
{
    while(1)
      {
          //: (RTC.INIR)set RTC active state
          RTC->INIR = 0xa5eb1357;
          //: (RTC.INIR) check RTC state?
          //: 1 = RTC is at normal active state.
          if(RTC->INIR) break;
      }
}
```

RTC初始化(RTC Initial Routine) 5/x

```
void RTCAccessEnable(void)
{
    while(1)
     {
         //: (RTC.AER) enable RTC access
         RTC->AER.AER = 0xA965;
         //: (RTC.AER) check RTC access enable?
         //: 1 = RTC register read/write enable
         if(RTC->AER.ENF) break;
     }
}
```

RTC初始化(RTC Initial Routine) 6/x

```
void set_TLR (int32_t a,int32_t b,int32_t c,int32_t d,int32_t e,int32_t f)
{
    //: Time Loading Register (second, minute, hour)
    //: [21:20]10HR         :10-Hour Time Digit of Alarm Setting (0~2)
    //: [19:16]1HR          :1-Hour Time Digit of Alarm Setting (0~9)
    //: [14:12]10MIN        :10-Min Time Digit of Alarm Setting (0~5)
    //: [11:8]1MIN          :1-Min Time Digit of Alarm Setting (0~9)
    //: [6:4]10SEC          :10-Sec Time Digit of Alarm Setting (0~5)
    //: [3:0]1SEC           :1-Sec Time Digit of Alarm Setting (0~9)
    //outpw(&RTC->TLR, a<<20|b<<16|c<<12|d<<8|e<<4|f)         ;
    RTC->u32TLR=(a<<20|b<<16|c<<12|d<<8|e<<4|f);
}
```

RTC初始化(RTC Initial Routine) 7/x

```
void set_CLR (int32_t a,int32_t b,int32_t c,int32_t d,int32_t e,int32_t f)
{
    //: Calenar Loading Register (day, month, year)
    //: [23:20]10YEAR: 10-Year Calendar Digit (0~9)
    //: [19:16]1YEAR: 1-Year Calendar Digit (0~9)
    //: [12] 10MON: 10-Month Calendar Digit (0~1)
    //: [11:8] 1MON: 1-Month Calendar Digit (0~9)
    //: [5:4] 10DAY: 10-Day Calendar Digit (0~3)
   //: [3:0] 1DAY: 1-Day Calendar Digit (0~9)
    //outpw(&RTC->CLR, a<<20|b<<16|c<<12|d<<8|e<<4|f)          ;
    RTC->u32CLR=(a<<20|b<<16|c<<12|d<<8|e<<4|f);
}
```

RTC初始化(RTC Initial Routine) 8/x

```
void set_TAR(int32_t a,int32_t b,int32_t c,int32_t d,int32_t e,int32_t f)
{
   //: Time Alarm Register (second, minute, hour)
   //: [21:20]10HR        :10-Hour Time Digit of Alarm Setting (0~2)
   //: [19:16]1HR         :1-Hour Time Digit of Alarm Setting (0~9)
   //: [14:12]10MIN       :10-Min Time Digit of Alarm Setting (0~5)
   //: [11:8]1MIN         :1-Min Time Digit of Alarm Setting (0~9)
   //: [6:4]10SEC         :10-Sec Time Digit of Alarm Setting (0~5)
   //: [3:0]1SEC          :1-Sec Time Digit of Alarm Setting (0~9)
   //outpw(&RTC->TAR, a<<20|b<<16|c<<12|d<<8|e<<4|f)        ;
   RTC->u32TAR=(a<<20|b<<16|c<<12|d<<8|e<<4|f);
}
```

RTC初始化(RTC Initial Routine) 9/x

```
void set_CAR(int32_t a,int32_t b,int32_t c,int32_t d,int32_t e,int32_t f)
{
    //: Calenar Alarm Register (day, month, year)
    //: [23:20]10YEAR: 10-Year Calendar Digit (0~9)
    //: [19:16]1YEAR: 1-Year Calendar Digit (0~9)
    //: [12] 10MON: 10-Month Calendar Digit (0~1)
    //: [11:8] 1MON: 1-Month Calendar Digit (0~9)
    //: [5:4] 10DAY: 10-Day Calendar Digit (0~3)
   //: [3:0] 1DAY: 1-Day Calendar Digit (0~9)
    //outpw(&RTC->CAR, a<<20|b<<16|c<<12|d<<8|e<<4|f)        ;
    RTC->u32CAR=(a<<20|b<<16|c<<12|d<<8|e<<4|f);
}
```

RTC中斷處理常式 (RTC Interrupt Routine) 1/x

```
void RTC_IRQHandler(void)
{
    //RTC Time Tick Interrupt Flag
    if(RTC->RIIR.TIF)
    {
            //callback function
            RTC_TickCallBackfn();
            //clear RIIT time tick interrupt flag
            //outpw(&RTC->RIIR,2);
            RTC->RIIR.TIF=1;
    }
```

RTC中斷處理常式 (RTC Interrupt Routine) 2/x

```
    // RTC Alarm Interrupt Flag
    //if(inpw(&RTC->RIIR)&0x1)
    if(RTC->RIIR.AIF) //CLR==CAR
    {
            //callback function
            RTC_AlarmCallBackfn();
            //clear RTC alarm interrupt flag
            //outpw(&RTC->RIIR,1);
            RTC->RIIR.AIF=1;
    }
}
```

RTC中斷處理常式 (RTC Interrupt Routine) 3/x

```
void RTC_TickCallBackfn(void)
{
        //flash green LED, GPIOA[13]
        GPIOA->DOUT ^= (1<<13);
}

void RTC_AlarmCallBackfn(void)
{
        //tuen on red LED,GPIOA[14]=0
        GPIOA->DOUT &= ~(1<<14);
}
```

RTC Sample Code with Driver (1/2)

```
/*---------------------------------------------------------------------------
  MAIN function
  ---------------------------------------------------------------------------*/
volatile int32_t   g_bAlarm  = FALSE;
int32_t main (void)
{
    S_DRVRTC_TIME_DATA_T sInitTime;
    /* RTC Initialize */
    DrvRTC_Init();
    /* Time Setting */
    sInitTime.u32Year      = 2009;
    sInitTime.u32cMonth    = 1;
    sInitTime.u32cDay      = 19;
    sInitTime.u32cHour     = 13;
    sInitTime.u32cMinute   = 20;
    sInitTime.u32cSecond   = 0;
    sInitTime.u32cDayOfWeek = DRVRTC_MONDAY;
    sInitTime.u8cClockDisplay = DRVRTC_CLOCK_24;
    while(DrvRTC_Open(&sInitTime) !=E_SUCCESS);

    /* Get the current time */
    S_DRVRTC_TIME_DATA_T sCurTime;
    DrvRTC_Read(DRVRTC_CURRENT_TIME, &sCurTime);
```

RTC Sample Code with Driver (2/2)

```
    /* Set Alarm call back function */
    sCurTime.pfnAlarmCallBack = DrvRTC_AlarmISR;

    /* The alarm time setting */
    sCurTime.u32cSecond = sCurTime.u32cSecond + 10;
    /* Set the alarm time (Install the call back function and enable the alarm interrupt)*/
    DrvRTC_Write(DRVRTC_ALARM_TIME,&sCurTime);

    NVIC_EnableIRQ(RTC_IRQn);
    while(!g_bAlarm);
    /* Disable RTC Clock */
    DrvRTC_Close();
}
void DrvRTC_AlarmISR(void)
{
    g_bAlarm = TRUE;
}
```

RTC Sample Code (1/3)

```
/*---------------------------------------------------------------------------
  MAIN function – 4 steps
 ---------------------------------------------------------------------------*/
int32_t main (void)
{
    UNLOCKREG();
    /* Step 1. Enable and Select RTC clock source */
     //Enable 32Khz for RTC clock source
    SYSCLK->PWRCON.XTL32K_EN = 1;
     //Enable RTC clock source
    SYSCLK->APBCLK.RTC_EN =1;

    /* Step 2. Initiate and unlock RTC module */
    //Initiate RTC module
    RTC->INIR = 0xa5eb1357;
    //Unlock RTC register
    RTC->AER.AER = 0xA965;

    /* Step 3. Initiate Time and Calendar  setting */
    //Set 24hour mode
    RTC->TSSR.HR24      =1;
    //Set time and calendar for loading register, Calendar→ 09/1/19, Time→ 13:20:00
    set_CLR(0,9,0,1,1,9);
    set_TLR(1,3,2,0,0,0);
```

RTC Sample Code (2/3)

```
    /* Step 4. Set alarm interrupt */
    //Set time and calendar for alarm register , Calendar→ 09/1/19, Time→ 13:20:10
    set_CAR(0,9,0,1,1,9);
    set_TAR(1,3,2,0,1,0);
    //Enable alarm interrupt
    RTC->RIER.AIER = 1;
    NVIC_EnableIRQ(RTC_IRQn);
    while(1);
}
/*---------------------------------------------------------------------------
Interrupt subroutine
---------------------------------------------------------------------------*/
 void RTC_IRQHandler(void)
{   /* tick */
    if(inpw(&RTC->RIIR)&0x2)
    {
          outpw(&RTC->RIIR,2);
    }
    /* alarm */
    if(inpw(&RTC->RIIR)&0x1)
    {
          outpw(&RTC->RIIR,1);
    }
}
```

RTC Sample Code (3/3)

```
void set_TLR (int32_t a,int32_t b,int32_t c,int32_t d,int32_t e,int32_t f)
{
   outpw(&RTC->TLR, a<<20|b<<16|c<<12|d<<8|e<<4|f)   ;
}
void set_CLR (int32_t a,int32_t b,int32_t c,int32_t d,int32_t e,int32_t f)
{
   outpw(&RTC->CLR, a<<20|b<<16|c<<12|d<<8|e<<4|f)   ;
}
void set_TAR(int32_t a,int32_t b,int32_t c,int32_t d,int32_t e,int32_t f)
{
   outpw(&RTC->TAR, a<<20|b<<16|c<<12|d<<8|e<<4|f)  ;
}
void set_CAR (int32_t a,int32_t b,int32_t c,int32_t d,int32_t e,int32_t f)
{
   outpw(&RTC->CAR, a<<20|b<<16|c<<12|d<<8|e<<4|f)  ;
}C
```

範例: Smpl_Timer

```
int32_t main (void)
{
	UNLOCKREG();
	SYSCLK->PWRCON.XTL32K_EN = 1;//Enable 32Khz for RTC clock source
	SYSCLK->PWRCON.XTL12M_EN = 1;
	SYSCLK->CLKSEL0.HCLK_S = 0;
	LOCKREG();

	Initial_panel();  //call initial pannel function
	clr_all_panel();

	InitTIMER0();
	InitTIMER1();
	InitTIMER2();


	while(1)
	{
		__NOP();
	}
}
```

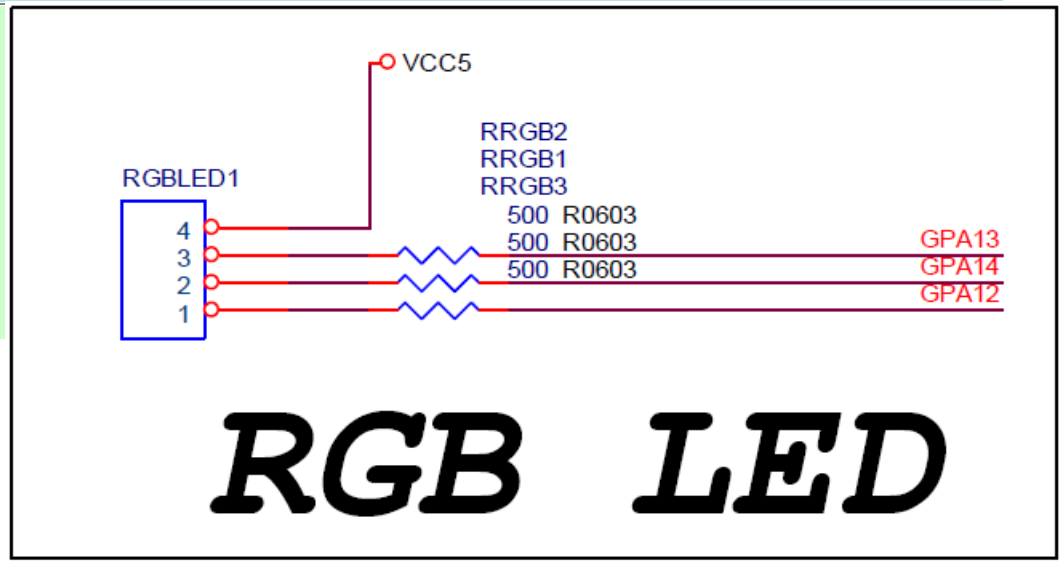
RTC01：Test_RTC

- RGB LED : GPA12,13,14

GPA12 : Blue　0 = on, 1 = off

GPA13 : Green　0 = on, 1 = off

GPA14 : Red　0 = on, 1 = off



寫一程式輪流顯示藍色，綠色，紅色LED，每個LED亮0.5秒。
1. 使用RTC控制時間，每0.5秒產生一次中斷。
2. 在RTC中斷時，切換顯示的LED。
3. 藍色LED :GPA[12]，綠色LED :GPA[13]，紅色LED :GPA[14]

RTC01： Test_RTC 1/x

```
int main (void)
{
    // Initial RTC
    Init_RTC();
    // Open RTC
    Write_RTC(DRVRTC_CURRENT_TIME,2013,4,8,9,10,5,1,1);
    //: set time tick period for periodic time tick Interrupt
    DrvRTC_SetTickMode (DRVRTC_TICK_1_2_SEC) ;
     //: enable specified interrupt and install callback function
     DrvRTC_EnableInt(DRVRTC_TICK_INT, RTC_TickCallBack);
```

RTC01： Test_RTC 2/x

```
    // Initial LED: set GPIO GPA12,13,14 to output mode
    DrvGPIO_Open(E_GPA, 12, E_IO_OUTPUT);
    DrvGPIO_Open(E_GPA, 13, E_IO_OUTPUT);
    DrvGPIO_Open(E_GPA, 14, E_IO_OUTPUT);
    // Main loop
    while (1)
    {
    }
}
```

RTC01： Test_RTC 3/x

```
// Initial RTC
void Init_RTC(void)
{
        UNLOCKREG();
        DrvRTC_Init() ;   //: (DrvRTC.c) Initial RTC
        //: Enable 32Khz clock source for RTC
        //SYSCLK->PWRCON.XTL32K_EN = 1;
        //: Enable RTC clock source
        //SYSCLK->APBCLK.RTC_EN =1;
        //RTC->INIR = 0xa5eb1357;
        LOCKREG();
}
```

RTC01： Test_RTC 4/x

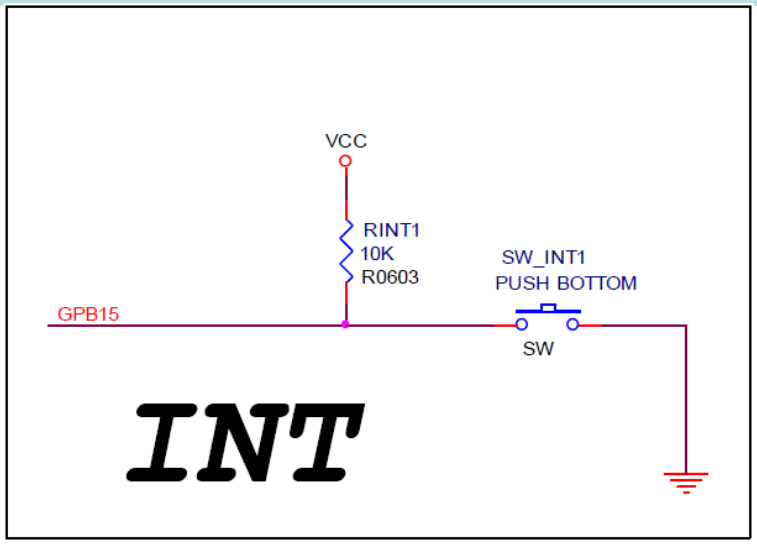
```
void Write_RTC(E_DRVRTC_TIME_SELECT eTime,uint16_t yy,uint8_t
mm,uint8_t dd,uint8_t hh,uint8_t mn,uint8_t ss,uint8_t wd,uint8_t h24)
{
        S_DRVRTC_TIME_DATA_T RTCTime;     // Initial set date and time
        RTCTime.u32Year = yy;
        RTCTime.u32cMonth = mm;
        RTCTime.u32cDay = dd;
        RTCTime.u32cHour = hh;
        RTCTime.u32cMinute = mn;
        RTCTime.u32cSecond = ss;
        RTCTime.u32cDayOfWeek = wd; //=1 for monday
        RTCTime.u8cClockDisplay = h24; //=1 for 24 hour mode
        DrvRTC_Write(eTime,&RTCTime);
}
```

RTC01： Test_RTC 5/x

```
// Display a number on 4-digit 7-seg.
void RTC_TickCallBack(void)
{
        // set RGBled lighted in turn
        DrvGPIO_SetBit(E_GPA,light_led++);
        //GPIOA->DOUT &= ~(0x1<<12);
        if(light_led == 15) light_led=12;
        DrvGPIO_ClrBit(E_GPA,light_led);
        //GPIOA->DOUT |= (1<<13);
}
```

RTC02：RTC_Interrupt_SWInt_RGBled

- RGB LED : GPA12,13,14
  - **GPA12** : Blue 0 = on, 1 = off
  - **GPA13** : Green 0 = on, 1 = off
  - **GPA14** : Red 0 = on, 1 = off
- SW Int : GPB15
  - **GPB15**: 0=pressed, 1= not pressed



寫一程式，LED閃爍亮0.5秒，暗0.5秒。當按鍵按下時，顯示紅色LED。按鍵鬆開時，顯示綠色LED。

GPIOB[15]作為一般輸入，不使用中斷。

RTC02： Test_RTC 1/x

```
int main (void)
{
    // Initial RTC
    Init_RTC();
    // Open RTC
    Write_RTC(DRVRTC_CURRENT_TIME,2013,4,8,9,10,5,1,1);
    //: set time tick period for periodic time tick Interrupt
    DrvRTC_SetTickMode (DRVRTC_TICK_1_128_SEC) ;
     //: enable specified interrupt and install callback function
     DrvRTC_EnableInt(DRVRTC_TICK_INT, RTC_TickCallBack);
```

RTC02： Test_RTC 2/x

```
        // Initial LED: set GPIO GPA13,14 to output mode
        DrvGPIO_Open(E_GPA, 13, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPA, 14, E_IO_OUTPUT);
        // Initial SW EINT1
        DrvGPIO_Open(E_GPB, 15, E_IO_INPUT);
        // Main loop
        while (1)
        {
        }
}
```

RTC02： Test_RTC 3/x

```
// Initial RTC
void Init_RTC(void)
{
        UNLOCKREG();
        DrvRTC_Init() ;   //: (DrvRTC.c) Initial RTC
        //: Enable 32Khz clock source for RTC
        //SYSCLK->PWRCON.XTL32K_EN = 1;
        //: Enable RTC clock source
        //SYSCLK->APBCLK.RTC_EN =1;
        //RTC->INIR = 0xa5eb1357;
        LOCKREG();
}
```

RTC02： Test_RTC 4/x

```
void Write_RTC(E_DRVRTC_TIME_SELECT eTime,uint16_t yy,uint8_t
mm,uint8_t dd,uint8_t hh,uint8_t mn,uint8_t ss,uint8_t wd,uint8_t h24)
{
        S_DRVRTC_TIME_DATA_T RTCTime;     // Initial set date and time
        RTCTime.u32Year = yy;
        RTCTime.u32cMonth = mm;
        RTCTime.u32cDay = dd;
        RTCTime.u32cHour = hh;
        RTCTime.u32cMinute = mn;
        RTCTime.u32cSecond = ss;
        RTCTime.u32cDayOfWeek = wd; //=1 for monday
        RTCTime.u8cClockDisplay = h24; //=1 for 24 hour mode
        DrvRTC_Write(eTime,&RTCTime);
}
```

RTC02： Test_RTC 5/x

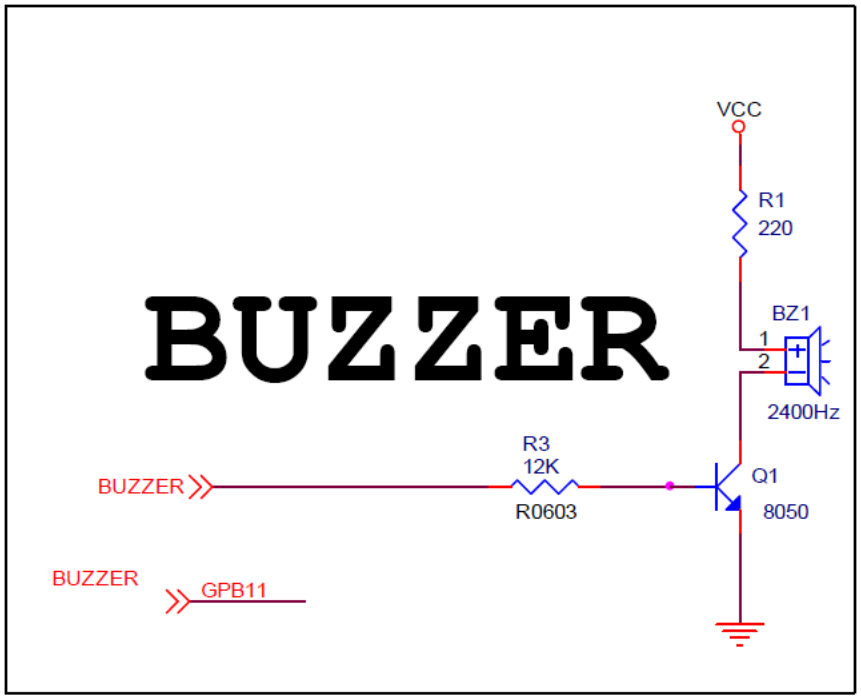
```
void RTC_TickCallBack(void)
{       // check key pressed
        if(DrvGPIO_GetBit(E_GPB,15)==0)
        //while((GPIOB->PIN & (1<<15))==0)
        {
                light_led =14; //red LED
                DrvGPIO_SetBit(E_GPA,13); //turn off green LED
        }
        else
        {
                light_led =13;  //green LED
                DrvGPIO_SetBit(E_GPA,14); //turn off red LED
        }
```

RTC02： Test_RTC 6/x

```
        // set RGBled lighted in turn
        if(!light_led_time--) //wait time 0.5sec=64*(1/128)
        {
                light_led_time=64; // flash every 0.5sec
                GPIOA->DOUT ^= (0x1<<light_led); //1->0, 0->1
        }
}
```

RTC03：RTC_SWInt_Buzzer

- BUZZER : GPB11
  - **GPB11** : 0 = on, 1 = off
- R1電阻太大，聲音太小聽不到。將R1短路，可聽到聲音。
- SW Int : GPB15
  - **GPB15**: 0=pressed, 1= not pressed



寫一程式，當按鍵按下時，蜂鳴器發出聲音。鬆開則沒有聲音。

GPIO pin GPB11=0 turn on BUZZER

GPIO pin GPB11=1 turn off BUZZER

RTC03： Test_RTC 1/x

```
int main (void)
{
    // Initial RTC
    Init_RTC();
    // Open RTC
    Write_RTC(DRVRTC_CURRENT_TIME,2013,4,8,9,10,5,1,1);
    //: set time tick period for periodic time tick Interrupt
    DrvRTC_SetTickMode (DRVRTC_TICK_1_128_SEC) ;
     //: enable specified interrupt and install callback function
     DrvRTC_EnableInt(DRVRTC_TICK_INT, RTC_TickCallBack);
```

RTC03： Test_RTC 2/x

```
        // Initial SW EINT1
        DrvGPIO_Open(E_GPB, 15, E_IO_INPUT);
        // Initial buzzer
        DrvGPIO_Open(E_GPB, 11, E_IO_OUTPUT);
        // Main loop
        while (1)
        {
        }
}
```

RTC03： Test_RTC 3/x

```
// Initial RTC
void Init_RTC(void)
{
        UNLOCKREG();
        DrvRTC_Init() ;   //: (DrvRTC.c) Initial RTC
        //: Enable 32Khz clock source for RTC
        //SYSCLK->PWRCON.XTL32K_EN = 1;
        //: Enable RTC clock source
        //SYSCLK->APBCLK.RTC_EN =1;
        //RTC->INIR = 0xa5eb1357;
        LOCKREG();
}
```

RTC03： Test_RTC 4/x

```
void Write_RTC(E_DRVRTC_TIME_SELECT eTime,uint16_t yy,uint8_t
mm,uint8_t dd,uint8_t hh,uint8_t mn,uint8_t ss,uint8_t wd,uint8_t h24)
{
        S_DRVRTC_TIME_DATA_T RTCTime;     // Initial set date and time
        RTCTime.u32Year = yy;
        RTCTime.u32cMonth = mm;
        RTCTime.u32cDay = dd;
        RTCTime.u32cHour = hh;
        RTCTime.u32cMinute = mn;
        RTCTime.u32cSecond = ss;
        RTCTime.u32cDayOfWeek = wd; //=1 for monday
        RTCTime.u8cClockDisplay = h24; //=1 for 24 hour mode
        DrvRTC_Write(eTime,&RTCTime);
}
```

RTC03： Test_RTC 5/x

```
void RTC_TickCallBack(void)
{       // if key pressed, turn on buzzer
        if(DrvGPIO_GetBit(E_GPB,15)==0)      // pressed
        {       // GPB11 = 0, turn on Buzzer
                DrvGPIO_ClrBit(E_GPB,11);
                //GPIOB->DOUT &= ~(0x1<<11);
        }
        else       // turn off buzzer
        {       // GPB11 = 1, turn off Buzzer
                DrvGPIO_SetBit(E_GPB,11);
                //GPIOB->DOUT |= (1<<11);
        }
}
```

RTC03： Test_RTC 6/x

```
        // set RGBled lighted in turn
        if(!light_led_time--)
        {
                light_led_time=64;
                GPIOA->DOUT ^= (0x1<<light_led); //1->0, 0->1
        }
}
```

RTC04：RTC_7seg–七段顯示器

每個7段顯示器有8個LED，分別標示為a,b,c,d,e,f,g,p。

如果將a,b,c,d,e,f,g,p分別對應到GPIO的bit 0,1,2,3,4,5,6,7，則可由GPIO控制LED的明暗。

共陽極7段顯示器由低電位控制，共陰極7段顯示器由高電位控制。

4個7段顯示器，使用4個GPIO來控制，每次亮1個，1=on,0=off。

題目：顯示0000-9999，間隔0.5秒。

RTC04： Test_RTC 1/x

```
int main (void)
{
    // Initial RTC
    Init_RTC();
    // Open RTC
    Write_RTC(DRVRTC_CURRENT_TIME,2013,4,8,9,10,5,1,1);
    //: set time tick period for periodic time tick Interrupt
    DrvRTC_SetTickMode (DRVRTC_TICK_1_128_SEC) ;
     //: enable specified interrupt and install callback function
     DrvRTC_EnableInt(DRVRTC_TICK_INT, RTC_TickCallBack);
```

RTC04： Test_RTC 2/x

```
        // Initial 7-seg.
        seven_segment_open(); // Main loop
        // Display 0000-9999
        seg_data(num9999);
        while (1)
        {
        }
}
```

RTC04： Test_RTC 3/x

```
// Initial RTC
void Init_RTC(void)
{
        UNLOCKREG();
        DrvRTC_Init() ;   //: (DrvRTC.c) Initial RTC
        //: Enable 32Khz clock source for RTC
        //SYSCLK->PWRCON.XTL32K_EN = 1;
        //: Enable RTC clock source
        //SYSCLK->APBCLK.RTC_EN =1;
        //RTC->INIR = 0xa5eb1357;
        LOCKREG();
}
```

RTC04： Test_RTC 4/x

```
void Write_RTC(E_DRVRTC_TIME_SELECT eTime,uint16_t yy,uint8_t
mm,uint8_t dd,uint8_t hh,uint8_t mn,uint8_t ss,uint8_t wd,uint8_t h24)
{
        S_DRVRTC_TIME_DATA_T RTCTime;     // Initial set date and time
        RTCTime.u32Year = yy;
        RTCTime.u32cMonth = mm;
        RTCTime.u32cDay = dd;
        RTCTime.u32cHour = hh;
        RTCTime.u32cMinute = mn;
        RTCTime.u32cSecond = ss;
        RTCTime.u32cDayOfWeek = wd; //=1 for monday
        RTCTime.u8cClockDisplay = h24; //=1 for 24 hour mode
        DrvRTC_Write(eTime,&RTCTime);
}
```

RTC04： Test_RTC 5/x

```
void seg_data(int16_t value)
{
        // seperate number to single digit
                digit[3] = value / 1000;
                value = value - digit[3] * 1000;
                digit[2] = value / 100;
                value = value - digit[2] * 100;
                digit[1] = value / 10;
                value = value - digit[1] * 10;
                digit[0] = value; //4 states
}
```

RTC04： Test_RTC 6/x

```
void seg_display(void)
{
        close_seven_segment(); //clear GPIOC bit4-7
        //GPIOC->DOUT &= ~(0xF<<4);//clear GPIOC[7:4] bit4-7
        show_seven_segment(light_7seg,digit[light_7seg]);
        //GPIOE->DOUT &= ~(0xFF);  //clear 7 seg.,
        //GPIOE->DOUT |= SEG_BUF[digit[3]];//show 7 seg.,
        //GPIOC->DOUT |= (1<<(3+4));//display 4th 7 seg.
        if(!light_7seg--)light_7seg=3;
}
```

RTC04： Test_RTC 7/x

```
void RTC_TickCallBack(void)
{
        if(!t1sec--)  //wait 0.5sec
        {
                t1sec=64;  // reset counter for 0.5sec
                num9999++; //reset dispaly number 0-9999
                if(num9999 == 10000)num9999=0;
                seg_data(num9999); //display number
        }
        seg_display();    // display one 7-segment
}
```

RTC04： Test_RTC 8/x

```
// set GPIOx for 7-seg.
void seven_segment_open(void)
{
        //Initial GPIOE [7:0] to output mode for 7-seg.
        DrvGPIO_Open(E_GPE, 0, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 1, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 2, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 3, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 4, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 5, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 6, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 7, E_IO_OUTPUT);
```

RTC04： Test_RTC 9/x

```
    //Initial GPIOC [7:4] to output mode for nth 7-seg.
    DrvGPIO_Open(E_GPC, 4, E_IO_OUTPUT);
    DrvGPIO_Open(E_GPC, 5, E_IO_OUTPUT);
    DrvGPIO_Open(E_GPC, 6, E_IO_OUTPUT);
    DrvGPIO_Open(E_GPC, 7, E_IO_OUTPUT);
}
```

RTC05：RTC_7seg_Keypad

Control Pins used for 3x3 Keypad

Column control : GPA2, 1, 0

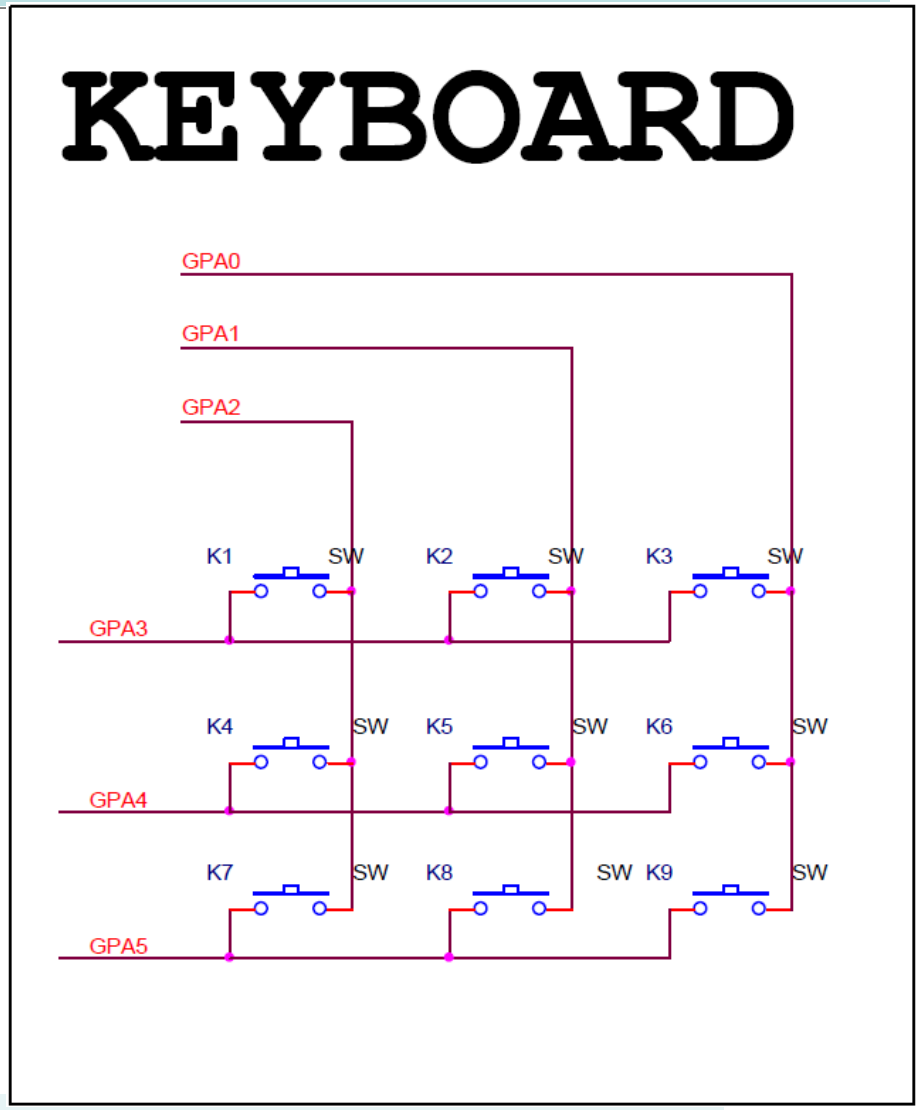
Raw control : GPA 3, 4, 5

- Key1 = GPA3 + GPA2
- Key2 = GPA3 + GPA1
- Key3 = GPA3 + GPA0
- Key4 = GPA4 + GPA2
- Key5 = GPA4 + GPA1
- Key6 = GPA4 + GPA0
- Key7 = GPA5 + GPA2
- Key8 = GPA5 + GPA1
- Key9 = GPA5 + GPA0

寫一程式，當按鍵按下時，顯示最後出現的4個數字。

GPA[2:0]依序輸出低電位(011, 101, 110)供按鍵讀取

GPA[5:3]讀取低電位



RTC05：RTC_7seg_Keypad

- 鍵盤按下時輸入信號為0，鬆開時輸入信號為1
- 通常按下或鬆開時間都會大於40ms，彈跳時間在10ms以內就會結束
- 若每5ms紀錄一次按鍵的狀態，當有按鍵完成時(按下_鬆開)，會出現00xx11的狀態，xx為彈跳期間。
- 程式設計1：
- 1.timer每1/128sec=7.8ms中斷一次，
- 2.檢查keypad是否有按鍵。若有按鍵，記錄按鍵的數字，和其狀態為0；否則紀錄狀態1。
- 3.檢查累積的狀態是否=00xx11。是：紀錄對應的數字，reset鍵盤狀態。
- 4.顯示下一個7seg.的數字
- 缺點：按鍵鬆開才能判斷一個數字。

RTC05： Test_RTC 1/x

```
int main (void)
{
    // Initial RTC
    Init_RTC();
    // Open RTC
    Write_RTC(DRVRTC_CURRENT_TIME,2013,4,8,9,10,5,1,1);
    //: set time tick period for periodic time tick Interrupt
    DrvRTC_SetTickMode (DRVRTC_TICK_1_128_SEC) ;
     //: enable specified interrupt and install callback function
     DrvRTC_EnableInt(DRVRTC_TICK_INT, RTC_TickCallBack);
```

RTC05： Test_RTC 2/x

```
        // Initial 7-seg.
        seven_segment_open();
        close_seven_segment();
        // Initial keypad
        OpenKeyPad(); //: (ScanKey.h)
        // Main loop
        while (1)
        {
        }
}
```

RTC05： Test_RTC 3/x

```
// Initial RTC
void Init_RTC(void)
{
        UNLOCKREG();
        DrvRTC_Init() ;   //: (DrvRTC.c) Initial RTC
        //: Enable 32Khz clock source for RTC
        //SYSCLK->PWRCON.XTL32K_EN = 1;
        //: Enable RTC clock source
        //SYSCLK->APBCLK.RTC_EN =1;
        //RTC->INIR = 0xa5eb1357;
        LOCKREG();
}
```

RTC05： Test_RTC 4/x

```
void Write_RTC(E_DRVRTC_TIME_SELECT eTime,uint16_t yy,uint8_t
mm,uint8_t dd,uint8_t hh,uint8_t mn,uint8_t ss,uint8_t wd,uint8_t h24)
{
        S_DRVRTC_TIME_DATA_T RTCTime;     // Initial set date and time
        RTCTime.u32Year = yy;
        RTCTime.u32cMonth = mm;
        RTCTime.u32cDay = dd;
        RTCTime.u32cHour = hh;
        RTCTime.u32cMinute = mn;
        RTCTime.u32cSecond = ss;
        RTCTime.u32cDayOfWeek = wd; //=1 for monday
        RTCTime.u8cClockDisplay = h24; //=1 for 24 hour mode
        DrvRTC_Write(eTime,&RTCTime);
}
```

RTC05： Test_RTC 5/x

```
void seg_keypad(void)
{
        uint8_t irow;
        //: store keypad number to 7-seg
        if(keybufferptr)
        {
                for (irow=3; irow >0; irow--)
                        digit_7seg[irow]=digit_7seg[irow-1];
                digit_7seg[0]=keybuffer[0];
        }
}
```

RTC05： Test_RTC 6/x

```
void seg_display(void)
{
        close_seven_segment(); //clear GPIOC bit4-7
        //GPIOC->DOUT &= ~(0xF<<4);//clear GPIOC[7:4] bit4-7
        show_seven_segment(light_7seg,digit[light_7seg]);
        //GPIOE->DOUT &= ~(0xFF);  //clear 7 seg.,
        //GPIOE->DOUT |= SEG_BUF[digit[3]];//show 7 seg.,
        //GPIOC->DOUT |= (1<<(3+4));//display 4th 7 seg.
        if(!light_7seg--)light_7seg=3;
}
```

RTC05： Test_RTC 7/x

```
void RTC_TickCallBack(void)
{
        //286clock,12.9us(22M),23.8us(12M)
        Scankeypad();
        //30clock,1.36us(22M),2.5us(12M)
        seg_keypad();
        //813clock,36.8us(22M),67.8us(12M)
        seg_display();
        while(keybufferptr)keybufferptr--;
}
```

RTC05： Test_RTC 8/x

```
// set GPIOx for 7-seg.
void seven_segment_open(void)
{
        //Initial GPIOE [7:0] to output mode for 7-seg.
        DrvGPIO_Open(E_GPE, 0, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 1, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 2, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 3, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 4, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 5, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 6, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 7, E_IO_OUTPUT);
```

RTC05： Test_RTC 9/x

```
    //Initial GPIOC [7:4] to output mode for nth 7-seg.
    DrvGPIO_Open(E_GPC, 4, E_IO_OUTPUT);
    DrvGPIO_Open(E_GPC, 5, E_IO_OUTPUT);
    DrvGPIO_Open(E_GPC, 6, E_IO_OUTPUT);
    DrvGPIO_Open(E_GPC, 7, E_IO_OUTPUT);
}
```

RTC05： Test_RTC 10/x

```
void Scankeypad(void)
{
        int8_t irow,keyno=0;
        for (irow=1; irow <4; irow++)
        {
                GPIOA->DOUT |= (0x7);          //xxxx-x111
                GPIOA->DOUT &= ~(1<<(3-irow));  //011, 101, 110
                //scan row1 GPIOA[3] for 1,2,3
                if((GPIOA->PIN & (0x1<<3)) == 0)keyno = irow;
                //scan row2 GPIOA[4] for 4,5,6
                if((GPIOA->PIN & (0x1<<4)) == 0)keyno = irow+3;
                //scan row3 GPIOA[5] for 7,8,9
                if((GPIOA->PIN & (0x1<<5)) == 0)keyno = irow+6;
        }
```

RTC05： Test_RTC 11/x

```
//record key status
if (keyno)
{    // key pressed
     if (keyno == keynumber )                    // same key
          keystatus = (keystatus<<1 );           // record key status=0
     else //different key
          {
               keystatus = 0xFE;// initial key status=HHHHHHHL
               keynumber=keyno;// record key number
          }
}
else  // key not pressed
{     keystatus = (keystatus<<1 | 1 );// record key status=1
}
```

RTC05： Test_RTC 12/x

```
// if keystatus = LLxxHH, the key is pressed and released
        if( (keystatus & 0x33) == 0x3)
        {
                keybuffer[keybufferptr++]=keynumber;
                keystatus = 0xFF; //reset key status=HHHHHHHH
                keynumber=0;      //reset key number
                keypress=0;       //key released
        }
}
```

RTC06：RTC_7seg_Keypad

Control Pins used for 3x3 Keypad

Column control : GPA2, 1, 0

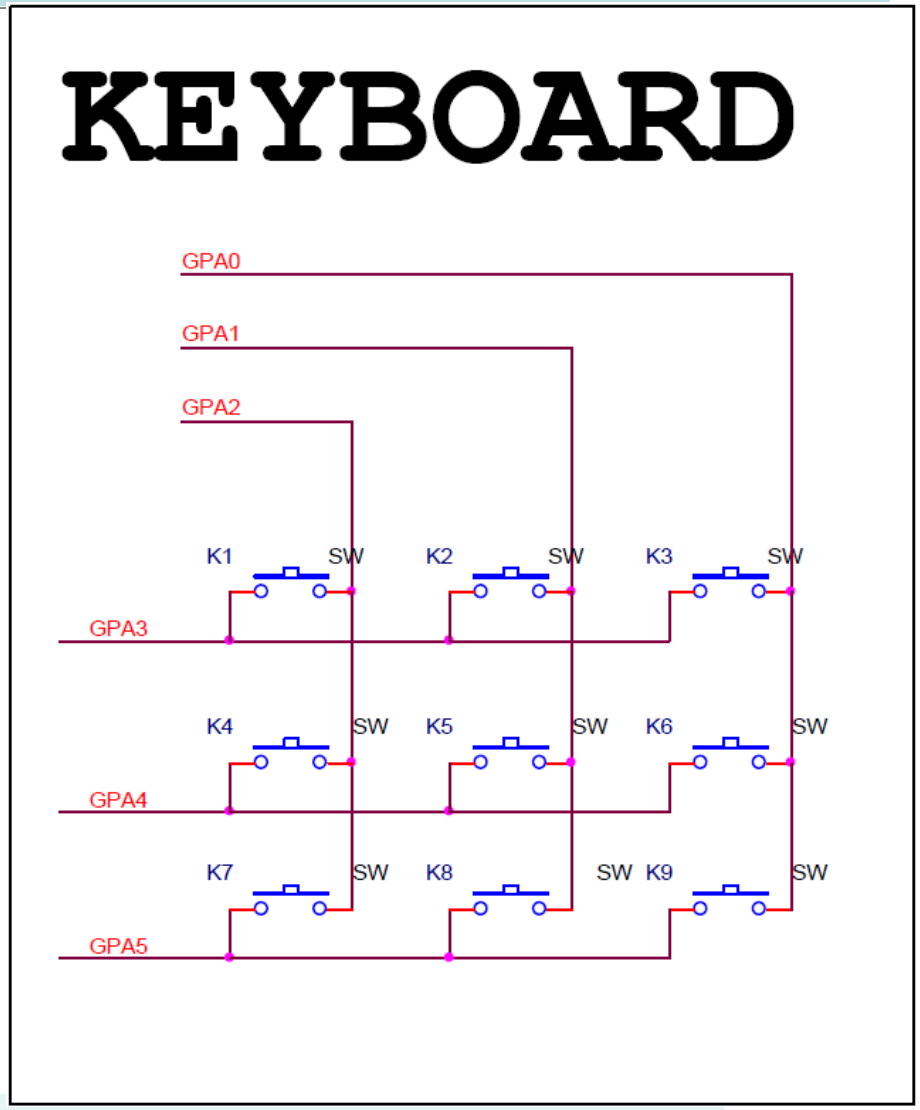
Raw control : GPA 3, 4, 5

- Key1 = GPA3 + GPA2
- Key2 = GPA3 + GPA1
- Key3 = GPA3 + GPA0
- Key4 = GPA4 + GPA2
- Key5 = GPA4 + GPA1
- Key6 = GPA4 + GPA0
- Key7 = GPA5 + GPA2
- Key8 = GPA5 + GPA1
- Key9 = GPA5 + GPA0

寫一程式，當按鍵按下時，顯示最後出現的4個數字。

GPA[2:0]依序輸出低電位(011, 101, 110)供按鍵讀取

GPA[5:3]讀取低電位



RTC06：RTC_7seg_Keypad

- 鍵盤按下時輸入信號為0，鬆開時輸入信號為1
- 通常按下或鬆開時間都會大於40ms，彈跳時間在10ms以內就會結束
- 若每5ms紀錄一次按鍵的狀態，當有按鍵完成時(按下_鬆開)，若出現0000表示按鍵已經按下，出現11表示按鍵鬆開。
- 程式設計2：
- 1.timer每1/128=7.8ms中斷一次，
- 2.檢查keypad是否有按鍵。若有按鍵，記錄按鍵的數字，和其狀態為0；否則紀錄狀態1。
- 3.檢查按鍵狀態。若按鍵未按下(狀態=0)且累積的狀態=0000，則按鍵按下(紀錄對應的數字，按鍵狀態=1)。若按鍵已按下(狀態=1)且累積的狀態=11，則按鍵鬆開(按鍵狀態=0)。
- 4.顯示下一個7seg.的數字

RTC06： Test_RTC 1/x

```
int main (void)
{
    // Initial RTC
    Init_RTC();
    // Open RTC
    Write_RTC(DRVRTC_CURRENT_TIME,2013,4,8,9,10,5,1,1);
    //: set time tick period for periodic time tick Interrupt
    DrvRTC_SetTickMode (DRVRTC_TICK_1_128_SEC) ;
     //: enable specified interrupt and install callback function
     DrvRTC_EnableInt(DRVRTC_TICK_INT, RTC_TickCallBack);
```

RTC06： Test_RTC 2/x

```
        // Initial 7-seg.
        seven_segment_open();
        close_seven_segment();
        // Initial keypad
        OpenKeyPad(); //: (ScanKey.h)
        // Main loop
        while (1)
        {
        }
}
```

RTC06： Test_RTC 3/x

```
// Initial RTC
void Init_RTC(void)
{
        UNLOCKREG();
        DrvRTC_Init() ;   //: (DrvRTC.c) Initial RTC
        //: Enable 32Khz clock source for RTC
        //SYSCLK->PWRCON.XTL32K_EN = 1;
        //: Enable RTC clock source
        //SYSCLK->APBCLK.RTC_EN =1;
        //RTC->INIR = 0xa5eb1357;
        LOCKREG();
}
```

RTC06： Test_RTC 4/x

```
void Write_RTC(E_DRVRTC_TIME_SELECT eTime,uint16_t yy,uint8_t
mm,uint8_t dd,uint8_t hh,uint8_t mn,uint8_t ss,uint8_t wd,uint8_t h24)
{
        S_DRVRTC_TIME_DATA_T RTCTime;     // Initial set date and time
        RTCTime.u32Year = yy;
        RTCTime.u32cMonth = mm;
        RTCTime.u32cDay = dd;
        RTCTime.u32cHour = hh;
        RTCTime.u32cMinute = mn;
        RTCTime.u32cSecond = ss;
        RTCTime.u32cDayOfWeek = wd; //=1 for monday
        RTCTime.u8cClockDisplay = h24; //=1 for 24 hour mode
        DrvRTC_Write(eTime,&RTCTime);
}
```

RTC06： Test_RTC 5/x

```
void seg_keypad(void)
{
        uint8_t irow;
        //: store keypad number to 7-seg
        if(keybufferptr)
        {
                for (irow=3; irow >0; irow--)
                        digit_7seg[irow]=digit_7seg[irow-1];
                digit_7seg[0]=keybuffer[0];
        }
}
```

RTC06： Test_RTC 6/x

```
void seg_display(void)
{
        close_seven_segment(); //clear GPIOC bit4-7
        //GPIOC->DOUT &= ~(0xF<<4);//clear GPIOC[7:4] bit4-7
        show_seven_segment(light_7seg,digit[light_7seg]);
        //GPIOE->DOUT &= ~(0xFF);  //clear 7 seg.,
        //GPIOE->DOUT |= SEG_BUF[digit[3]];//show 7 seg.,
        //GPIOC->DOUT |= (1<<(3+4));//display 4th 7 seg.
        if(!light_7seg--)light_7seg=3;
}
```

RTC06： Test_RTC 7/x

```
void RTC_TickCallBack(void)
{
        //286clock,12.9us(22M),23.8us(12M)
        Scankeypad();
        //30clock,1.36us(22M),2.5us(12M)
        seg_keypad();
        //813clock,36.8us(22M),67.8us(12M)
        seg_display();
        while(keybufferptr)keybufferptr--;
}
```

RTC06： Test_RTC 8/x

```
// set GPIOx for 7-seg.
void seven_segment_open(void)
{
        //Initial GPIOE [7:0] to output mode for 7-seg.
        DrvGPIO_Open(E_GPE, 0, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 1, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 2, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 3, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 4, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 5, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 6, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 7, E_IO_OUTPUT);
```

RTC06： Test_RTC 9/x

```
    //Initial GPIOC [7:4] to output mode for nth 7-seg.
    DrvGPIO_Open(E_GPC, 4, E_IO_OUTPUT);
    DrvGPIO_Open(E_GPC, 5, E_IO_OUTPUT);
    DrvGPIO_Open(E_GPC, 6, E_IO_OUTPUT);
    DrvGPIO_Open(E_GPC, 7, E_IO_OUTPUT);
}
```

RTC06： Test_RTC 10/x

```
void Scankeypad(void)
{
        int8_t irow,keyno=0;
        for (irow=1; irow <4; irow++)
        {
                GPIOA->DOUT |= (0x7);          //xxxx-x111
                GPIOA->DOUT &= ~(1<<(3-irow));  //011, 101, 110
                //scan row1 GPIOA[3] for 1,2,3
                if((GPIOA->PIN & (0x1<<3)) == 0)keyno = irow;
                //scan row2 GPIOA[4] for 4,5,6
                if((GPIOA->PIN & (0x1<<4)) == 0)keyno = irow+3;
                //scan row3 GPIOA[5] for 7,8,9
                if((GPIOA->PIN & (0x1<<5)) == 0)keyno = irow+6;
        }
```

RTC06： Test_RTC 11/x

```
//record key status
if (keyno)
{    // key pressed
     if (keyno == keynumber )                    // same key
          keystatus = (keystatus<<1 );           // record key status=0
     else //different key
          {
               keystatus = 0xFE;// initial key status
               keynumber=keyno;// record key number
          }
}
else  // key not pressed
{     keystatus = (keystatus<<1 | 1 );// record key status=1
}
```

RTC06： Test_RTC 12/x

```
        // if get 4 continuous low level, the key is pressed.
        if( !keypress && !(keystatus & 0x0F) )
        {
                keypress=1;
                keybuffer[keybufferptr++]=keynumber;
        }
        // check key released
        if(keypress && ((keystatus & 0x3)==3) )
        {
                keystatus = 0xFF; //reset key status
                keynumber=0;      //reset key number
                keypress=0;       //key released
        }
}
```

RTC07：show results of 2 digits sum

- 輸入2個2位數，顯示結果。
- 1.使用3x3鍵盤輸入1-9的數字，前面2個數字作為第1個2位數。後面2個數字作為第2個2位數。
- 2.將輸入的數字顯示在7段顯示器。
- 3.將輸入的數字顯示在LCD。
- 4.將相加的結果顯示在LCD。

RTC07：show results of 2 digits sum 1/x

```
int32_t main (void)
{
        uint8_t suma,sumb,sumc;
        char text[4] ,text16[16];
        text[3]=0x0;
        // Initial RTC
        Init_RTC();
        // Open RTC
        Write_RTC(DRVRTC_CURRENT_TIME,2013,4,8,9,10,5,1,1);
        // set time tick period and callback function
        DrvRTC_SetTickMode (DRVRTC_TICK_1_128_SEC) ;
```

RTC07：show results of 2 digits sum 2/x

```
        //: enable specified interrupt and install callback function
        DrvRTC_EnableInt(DRVRTC_TICK_INT,
RTC_TickCallBack);
        Initial_pannel();           // Initial LCD
        while(1)
        {
                suma=read2digit();     // read 2 digits
                uint8_ascii (suma,text);  // convert to ASCII
                clr_all_pannal();          //clear LCD
                print_lcd(0, text);        // show on LCD
                text16[0]=0;               // initial string
                strcat(text16, text);      // copy string to text16[]
                strcat(text16,"+");        // copy string to text16[]
```

RTC07：show results of 2 digits sum 3/x

```
                sumb=read2digit();        //read a 2-digits number
                uint8_ascii(sumb,text);   //conver number to ASCII
                print_lcd(1, text);       //show on LCD
                strcat(text16, text);     // copy string to text16[]
                strcat(text16, "=");      // copy string to text16[]

                sumc=suma+sumb;
                uint8_ascii(sumc,text);   //conver number to ASCII
                strcat(text16, text);     // copy string to text16[]
                print_lcd(2, text16);      //show on LCD
        }
}
```

RTC07：show results of 2 digits sum 4/x

```
//  read a 2-digits number
uint8_t read2digit(void)
{
        uint8_t d10,d1;

        keybufferptr=0;                 //reset keypad input buffer
        while(!keybufferptr);           //wait for keypad input
        d10=keybuffer[--keybufferptr];  //take a digit
        while(!keybufferptr);           //wait for keypad input
        d1=keybuffer[--keybufferptr];   //take a digit
        return 10*d10+d1;               //return number
}
```

RTC07：show results of 2 digits sum 5/x

```
//  convert 0-9,A-F to ascii
char bin2ascii(uint8_t number)
{
        if(number <10)
                return number+'0';          //value 0-9
        else
                return number+'A';          //value 10-15
}
```

RTC07：show results of 2 digits sum 6/x

```
//  convert  a 2-digits number  to ascii
void uint8_ascii(uint8_t value,char text[])
{
        uint8_t d1,d0;
        int8_t ia;
        ia=2;
        for (ia=2; ia>=0; ia--)
        {
                d1=value/10;
                d0=value-d1*10;
                text[ia]=bin2ascii(d0);
                value=d1;
        }
}
```

RTC07：show results of 2 digits sum 7/x

```
// Initial RTC
void Init_RTC(void)
{
        UNLOCKREG();
        DrvRTC_Init() ;   //: (DrvRTC.c) Initial RTC
        //: Enable 32Khz clock source for RTC
        //SYSCLK->PWRCON.XTL32K_EN = 1;
        //: Enable RTC clock source
        //SYSCLK->APBCLK.RTC_EN =1;
        //RTC->INIR = 0xa5eb1357;
        LOCKREG();
}
```

RTC07：show results of 2 digits sum 8/x

```
void Write_RTC(E_DRVRTC_TIME_SELECT eTime,uint16_t yy,uint8_t
mm,uint8_t dd,uint8_t hh,uint8_t mn,uint8_t ss,uint8_t wd,uint8_t h24)
{
        S_DRVRTC_TIME_DATA_T RTCTime;      // Initial set date and time
        RTCTime.u32Year = yy;
        RTCTime.u32cMonth = mm;
        RTCTime.u32cDay = dd;
        RTCTime.u32cHour = hh;
        RTCTime.u32cMinute = mn;
        RTCTime.u32cSecond = ss;
        RTCTime.u32cDayOfWeek = wd; //=1 for monday
        RTCTime.u8cClockDisplay = h24; //=1 for 24 hour mode
        DrvRTC_Write(eTime,&RTCTime);
}
```

RTC07：show results of 2 digits sum 9/x

```
void seg_keypad(void)
{
        uint8_t irow;
        //: store keypad number to 7-seg
        if(keybufferptr)
        {
                for (irow=3; irow >0; irow--)
                        digit_7seg[irow]=digit_7seg[irow-1];
                digit_7seg[0]=keybuffer[0];
        }
}
```

RTC07：show results of 2 digits sum 10/x

```
void seg_display(void)
{
        close_seven_segment(); //clear GPIOC bit4-7
        //GPIOC->DOUT &= ~(0xF<<4);//clear GPIOC[7:4] bit4-7
        show_seven_segment(light_7seg,digit[light_7seg]);
        //GPIOE->DOUT &= ~(0xFF);  //clear 7 seg.,
        //GPIOE->DOUT |= SEG_BUF[digit[3]];//show 7 seg.,
        //GPIOC->DOUT |= (1<<(3+4));//display 4th 7 seg.
        if(!light_7seg--)light_7seg=3;
}
```

RTC07：show results of 2 digits sum 11/x

```
void RTC_TickCallBack(void)
{
        //286clock,12.9us(22M),23.8us(12M)
        Scankeypad();
        //30clock,1.36us(22M),2.5us(12M)
        seg_keypad();
        //813clock,36.8us(22M),67.8us(12M)
        seg_display();
        while(keybufferptr)keybufferptr--;
}
```

RTC07：show results of 2 digits sum 12/x

```
// set GPIOx for 7-seg.
void seven_segment_open(void)
{
        //Initial GPIOE [7:0] to output mode for 7-seg.
        DrvGPIO_Open(E_GPE, 0, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 1, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 2, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 3, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 4, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 5, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 6, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 7, E_IO_OUTPUT);
```

RTC07：show results of 2 digits sum 13/x

```
        //Initial GPIOC [7:4] to output mode for nth 7-seg.
        DrvGPIO_Open(E_GPC, 4, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPC, 5, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPC, 6, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPC, 7, E_IO_OUTPUT);
}
```

RTC07：show results of 2 digits sum 14/x

```
void Scankeypad(void)
{
        int8_t irow,keyno=0;
        for (irow=1; irow <4; irow++)
        {
                GPIOA->DOUT |= (0x7);          //xxxx-x111
                GPIOA->DOUT &= ~(1<<(3-irow));  //011, 101, 110
                //scan row1 GPIOA[3] for 1,2,3
                if((GPIOA->PIN & (0x1<<3)) == 0)keyno = irow;
                //scan row2 GPIOA[4] for 4,5,6
                if((GPIOA->PIN & (0x1<<4)) == 0)keyno = irow+3;
                //scan row3 GPIOA[5] for 7,8,9
                if((GPIOA->PIN & (0x1<<5)) == 0)keyno = irow+6;
        }
```

RTC07：show results of 2 digits sum 15/x

```
//record key status
if (keyno)
{    // key pressed
     if (keyno == keynumber )                    // same key
          keystatus = (keystatus<<1 );           // record key status=0
     else //different key
          {
               keystatus = 0xFE;// initial key status
               keynumber=keyno;// record key number
          }
}
else  // key not pressed
{     keystatus = (keystatus<<1 | 1 );// record key status=1
}
```

RTC07：show results of 2 digits sum 16/x

```
        // if get 4 continuous low level, the key is pressed.
        if( !keypress && !(keystatus & 0x0F) )
        {
                keypress=1;
                keybuffer[keybufferptr++]=keynumber;
        }
        // check key released
        if(keypress && ((keystatus & 0x3)==3) )
        {
                keystatus = 0xFF; //reset key status
                keynumber=0;      //reset key number
                keypress=0;       //key released
        }
}
```

RTC08：show results of 2 digits sum

- 輸入2個2位數，顯示結果。
- 1.使用3x3鍵盤輸入1-9的數字，前面2個數字作為第1個2位數。後面2個數字作為第2個2位數。
- 2.將輸入的數字顯示在7段顯示器。
- 3.將輸入的數字顯示在LCD。
- 4.將相加的結果顯示在LCD。

RTC08：show results of 2 digits sum 1/x

```
int32_t main (void)
{
        uint8_t suma,sumb,sumc;
        char text[4],text16[16],text16a[16];
        text[3]=0x0;
        // Initial RTC
        Init_RTC();
        // Open RTC
        Write_RTC(DRVRTC_CURRENT_TIME,2013,4,8,9,10,5,1,1);
        // set time tick period and callback function
        DrvRTC_SetTickMode (DRVRTC_TICK_1_128_SEC) ;
```

RTC08：show results of 2 digits sum 2/x

```
        //: enable specified interrupt and install callback function
        DrvRTC_EnableInt(DRVRTC_TICK_INT,
RTC_TickCallBack);
        Initial_pannel();           // Initial LCD
        while(1)
        {
```

RTC08：show results of 2 digits sum 3/x

```
clr_all_pannal();           //clear LCD
text16a[0]=0;               // initial string
strcat(text16a,"input:\0"); // initial string
print_lcd(0,text16a);
suma=read2digit();          //read a 2-digits number
uint8_ascii(suma,text);     //conver number to ASCII
strcat(text16a, text);      // copy string to text16[]
print_lcd(0, text16a);      //show on LCD
text16[0]=0;                // initial string
strcat(text16, text);       // copy string to text16[]
strcat(text16,"+");         // copy string to text16[]
```

RTC08：show results of 2 digits sum 3/x

```
text16a[0]=0;                 // initial string
strcat(text16a,"input:\0"); // initial string
print_lcd(1,text16a);
sumb=read2digit();
uint8_ascii(sumb,text);
strcat(text16a, text);        // copy string to text16[]
print_lcd(1, text16a);        //show on LCD
strcat(text16, text);
strcat(text16, "=");
```

RTC08：show results of 2 digits sum 3/x

```
                sumc=suma+sumb;
                uint8_ascii(sumc,text);
                strcat(text16, text);
                print_lcd(2, text16);

                keybufferptr=0;          //reset keypad input buffer
                while(!keybufferptr);    //wait for keypad input
        }
}
```

RTC08：show results of 2 digits sum 4/x

```
//  read a 2-digits number
uint8_t read2digit(void)
{
        uint8_t d10,d1;

        keybufferptr=0;                 //reset keypad input buffer
        while(!keybufferptr);           //wait for keypad input
        d10=keybuffer[--keybufferptr];  //take a digit
        while(!keybufferptr);           //wait for keypad input
        d1=keybuffer[--keybufferptr];   //take a digit
        return 10*d10+d1;               //return number
}
```

RTC08：show results of 2 digits sum 5/x

```
//  convert 0-9,A-F to ascii
char bin2ascii(uint8_t number)
{
        if(number <10)
                return number+'0';          //value 0-9
        else
                return number+'A';          //value 10-15
}
```

RTC08：show results of 2 digits sum 6/x

```
// convert a 2-digits number to ascii
void uint8_ascii(uint8_t value,char text[])
{
        uint8_t d1,d0;
        int8_t ia;
        ia=2;
        for (ia=2; ia>=0; ia--)
        {
                d1=value/10;
                d0=value-d1*10;
                text[ia]=bin2ascii(d0);
                value=d1;
        }
}
```

RTC08：show results of 2 digits sum 7/x

```
// Initial RTC
void Init_RTC(void)
{
        UNLOCKREG();
        DrvRTC_Init() ;   //: (DrvRTC.c) Initial RTC
        //: Enable 32Khz clock source for RTC
        //SYSCLK->PWRCON.XTL32K_EN = 1;
        //: Enable RTC clock source
        //SYSCLK->APBCLK.RTC_EN =1;
        //RTC->INIR = 0xa5eb1357;
        LOCKREG();
}
```

RTC08：show results of 2 digits sum 8/x

```
void Write_RTC(E_DRVRTC_TIME_SELECT eTime,uint16_t yy,uint8_t
mm,uint8_t dd,uint8_t hh,uint8_t mn,uint8_t ss,uint8_t wd,uint8_t h24)
{
        S_DRVRTC_TIME_DATA_T RTCTime;     // Initial set date and time
        RTCTime.u32Year = yy;
        RTCTime.u32cMonth = mm;
        RTCTime.u32cDay = dd;
        RTCTime.u32cHour = hh;
        RTCTime.u32cMinute = mn;
        RTCTime.u32cSecond = ss;
        RTCTime.u32cDayOfWeek = wd; //=1 for monday
        RTCTime.u8cClockDisplay = h24; //=1 for 24 hour mode
        DrvRTC_Write(eTime,&RTCTime);
}
```

RTC08：show results of 2 digits sum 9/x

```
void seg_keypad(void)
{
        uint8_t irow;
        //: store keypad number to 7-seg
        if(keybufferptr)
        {
                for (irow=3; irow >0; irow--)
                        digit_7seg[irow]=digit_7seg[irow-1];
                digit_7seg[0]=keybuffer[0];
        }
}
```

RTC08：show results of 2 digits sum 10/x

```
void seg_display(void)
{
        close_seven_segment(); //clear GPIOC bit4-7
        //GPIOC->DOUT &= ~(0xF<<4);//clear GPIOC[7:4] bit4-7
        show_seven_segment(light_7seg,digit[light_7seg]);
        //GPIOE->DOUT &= ~(0xFF);  //clear 7 seg.,
        //GPIOE->DOUT |= SEG_BUF[digit[3]];//show 7 seg.,
        //GPIOC->DOUT |= (1<<(3+4));//display 4th 7 seg.
        if(!light_7seg--)light_7seg=3;
}
```

RTC08：show results of 2 digits sum 11/x

```
void RTC_TickCallBack(void)
{
        //286clock,12.9us(22M),23.8us(12M)
        Scankeypad();
        //30clock,1.36us(22M),2.5us(12M)
        seg_keypad();
        //813clock,36.8us(22M),67.8us(12M)
        seg_display();
        while(keybufferptr)keybufferptr--;
}
```

RTC08：show results of 2 digits sum 12/x

```
// set GPIOx for 7-seg.
void seven_segment_open(void)
{
        //Initial GPIOE [7:0] to output mode for 7-seg.
        DrvGPIO_Open(E_GPE, 0, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 1, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 2, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 3, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 4, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 5, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 6, E_IO_OUTPUT);
        DrvGPIO_Open(E_GPE, 7, E_IO_OUTPUT);
```

RTC08：show results of 2 digits sum 13/x

```
    //Initial GPIOC [7:4] to output mode for nth 7-seg.
    DrvGPIO_Open(E_GPC, 4, E_IO_OUTPUT);
    DrvGPIO_Open(E_GPC, 5, E_IO_OUTPUT);
    DrvGPIO_Open(E_GPC, 6, E_IO_OUTPUT);
    DrvGPIO_Open(E_GPC, 7, E_IO_OUTPUT);
}
```

RTC08：show results of 2 digits sum 14/x

```
void Scankeypad(void)
{
        int8_t irow,keyno=0;
        for (irow=1; irow <4; irow++)
        {
                GPIOA->DOUT |= (0x7);          //xxxx-x111
                GPIOA->DOUT &= ~(1<<(3-irow));  //011, 101, 110
                //scan row1 GPIOA[3] for 1,2,3
                if((GPIOA->PIN & (0x1<<3)) == 0)keyno = irow;
                //scan row2 GPIOA[4] for 4,5,6
                if((GPIOA->PIN & (0x1<<4)) == 0)keyno = irow+3;
                //scan row3 GPIOA[5] for 7,8,9
                if((GPIOA->PIN & (0x1<<5)) == 0)keyno = irow+6;
        }
```

RTC08：show results of 2 digits sum 15/x

```
//record key status
if (keyno)
{    // key pressed
     if (keyno == keynumber )                    // same key
          keystatus = (keystatus<<1 );           // record key status=0
     else //different key
          {
               keystatus = 0xFE;// initial key status
               keynumber=keyno;// record key number
          }
}
else  // key not pressed
{     keystatus = (keystatus<<1 | 1 );// record key status=1
}
```

RTC08：show results of 2 digits sum 16/x

```
        // if get 4 continuous low level, the key is pressed.
        if( !keypress && !(keystatus & 0x0F) )
        {
                keypress=1;
                keybuffer[keybufferptr++]=keynumber;
        }
        // check key released
        if(keypress && ((keystatus & 0x3)==3) )
        {
                keystatus = 0xFF; //reset key status
                keynumber=0;      //reset key number
                keypress=0;       //key released
        }
}
```

RTC11：Test_RTC

- 寫一程式，設定RTC的年月日和時分秒。在4個7段顯示器顯示目前的分和秒。
- 1.將分和秒的值儲存在一個陣列，對應到4個7段顯示器。
- 2.每一個7段顯示器輪流亮2.5ms，輪完一圈約10ms。
- 3.使用函數DrvSYS_Delay()延遲2500us=2.5ms。
- 缺點：CPU無法在等待2.5ms的時間去做其他的工作。

RTC11： Test_RTC 1/x

```
uint8_t digit_7seg[4];     // store display data for 7-seg.
uint8_t light_7seg=3;      // lighted 7-seg

int32_t main (void)
{
        Init_RTC();
        //=1 for monday
        //=1 for 24 hour mode
       Write_RTC(DRVRTC_CURRENT_TIME,2013,4,8,9,10,5,1,1);
```

RTC11： Test_RTC 2/x

```
    while(1)
    {       // store minute ans second to 7 seg. Data array
            seg_minsec();
            // display data on 7 seg.
            seg_display();
            // delay 2500us
            DrvSYS_Delay(2500); //delay 2500us=2.5ms
            }
}
```

RTC11： Test_RTC 3/x

```
// Initial RTC
void Init_RTC(void)
{
        UNLOCKREG();
        DrvRTC_Init() ;          //: (DrvRTC.c) Initial RTC
        //: Enable 32Khz clock source for RTC
        //SYSCLK->PWRCON.XTL32K_EN = 1;
        //: Enable RTC clock source
        //SYSCLK->APBCLK.RTC_EN =1;
        //RTC->INIR = 0xa5eb1357;          //: RTC Initiation
        LOCKREG();
}
```

RTC11： Test_RTC 4/x

```
void Write_RTC(E_DRVRTC_TIME_SELECT eTime,uint16_t yy,uint8_t
mm,uint8_t dd,uint8_t hh,uint8_t mn,uint8_t ss,uint8_t wd,uint8_t h24)
{
        S_DRVRTC_TIME_DATA_T RTCTime; // Initial set date and time
        RTCTime.u32Year = yy;
        RTCTime.u32cMonth = mm;
        RTCTime.u32cDay = dd;
        RTCTime.u32cHour = hh;
        RTCTime.u32cMinute = mn;
        RTCTime.u32cSecond = ss;
        RTCTime.u32cDayOfWeek = wd; //=1 for monday
        RTCTime.u8cClockDisplay = h24; //=1 for 24 hour mode
        DrvRTC_Write(eTime,&RTCTime);
}
```

RTC11： Test_RTC 5/x

```
// Display minute and second on 4-digit 7-seg.
void seg_minsec(void)
{
        digit_7seg[3]=RTC->TLR.MIN10;
        digit_7seg[2]=RTC->TLR.MIN1;
        digit_7seg[1]=RTC->TLR.SEC10;
        digit_7seg[0]=RTC->TLR.SEC1;
}
```

RTC11： Test_RTC 6/x

```
// display digit on 7-seg.
void seg_display(void)
{
        //: show a digit on 7-seg.
        close_seven_segment();
        show_seven_segment(light_7seg,digit_7seg[light_7seg]);
        if(!light_7seg--)light_7seg=3;
}
```

RTC12：Test_RTC

- 寫一程式，設定RTC的年月日和時分秒。在4個7段顯示器顯示目前的分和秒。
- 1.將分和秒的值儲存在一個陣列，對應到4個7段顯示器。
- 2.使用RTC的Time Tick每1/128秒(7.8ms)產生中斷，用來輪流亮7段顯示器。每個7段顯示器亮7.8ms，下次中斷亮下一個，依序循環。
- 優點：CPU可以去做其他的工作，中斷時亮下一個7段顯示器。
- 缺點：受限於RTC，中斷的時間為1，1/2，1/4，1/8，1/16，1/32，1/64，1/128秒。

RTC12：Test_RTC 1/x

```
int32_t main (void)
{
        Init_RTC();
        Write_RTC(DRVRTC_CURRENT_TIME,2013,4,8,9,10,5,1,1);
        //: set time tick period for periodic time tick Interrupt
        //: DRVRTC_TICK_1_128_SEC : Time tick is 1/128 second
        DrvRTC_SetTickMode (DRVRTC_TICK_1_128_SEC) ;

        //: enable specified interrupt and install callback function
        //: DRVRTC_TICK_INT : Tick interrupt
        DrvRTC_EnableInt(DRVRTC_TICK_INT, RTC_TickCallBack);
```

RTC12：Test_RTC 2/x

```
        while(1)
        {
                __NOP();
        }
}
```

RTC12：Test_RTC 3/x

```
void RTC_TickCallBack(void)
{
        seg_minsec();
        seg_display();
}
```

RTC12：Test_RTC 4/x

```
// Initial RTC
void Init_RTC(void)
{
        UNLOCKREG();
        DrvRTC_Init() ;          //: (DrvRTC.c) Initial RTC
        //: Enable 32Khz clock source for RTC
        //SYSCLK->PWRCON.XTL32K_EN = 1;
        //: Enable RTC clock source
        //SYSCLK->APBCLK.RTC_EN =1;
        //RTC->INIR = 0xa5eb1357;          //: RTC Initiation
        LOCKREG();
}
```

RTC12：Test_RTC 5/x

```
void Write_RTC(E_DRVRTC_TIME_SELECT eTime,uint16_t yy,uint8_t mm,uint8_t dd,uint8_t hh,uint8_t mn,uint8_t ss,uint8_t wd,uint8_t h24)
{
        S_DRVRTC_TIME_DATA_T RTCTime; // Initial set date and time
        RTCTime.u32Year = yy;
        RTCTime.u32cMonth = mm;
        RTCTime.u32cDay = dd;
        RTCTime.u32cHour = hh;
        RTCTime.u32cMinute = mn;
        RTCTime.u32cSecond = ss;
        RTCTime.u32cDayOfWeek = wd; //=1 for monday
        RTCTime.u8cClockDisplay = h24; //=1 for 24 hour mode
        DrvRTC_Write(eTime,&RTCTime);
}
```

RTC12：Test_RTC 6/x

```
// Display minute and second on 4-digit 7-seg.
void seg_minsec(void)
{
        digit_7seg[3]=RTC->TLR.MIN10;
        digit_7seg[2]=RTC->TLR.MIN1;
        digit_7seg[1]=RTC->TLR.SEC10;
        digit_7seg[0]=RTC->TLR.SEC1;
}
```

RTC12：Test_RTC 7/x

```
// display digit on 7-seg.
void seg_display(void)
{
        //: show a digit on 7-seg.
        close_seven_segment();
        show_seven_segment(light_7seg,digit_7seg[light_7seg]);
        if(!light_7seg--)light_7seg=3;
}
```

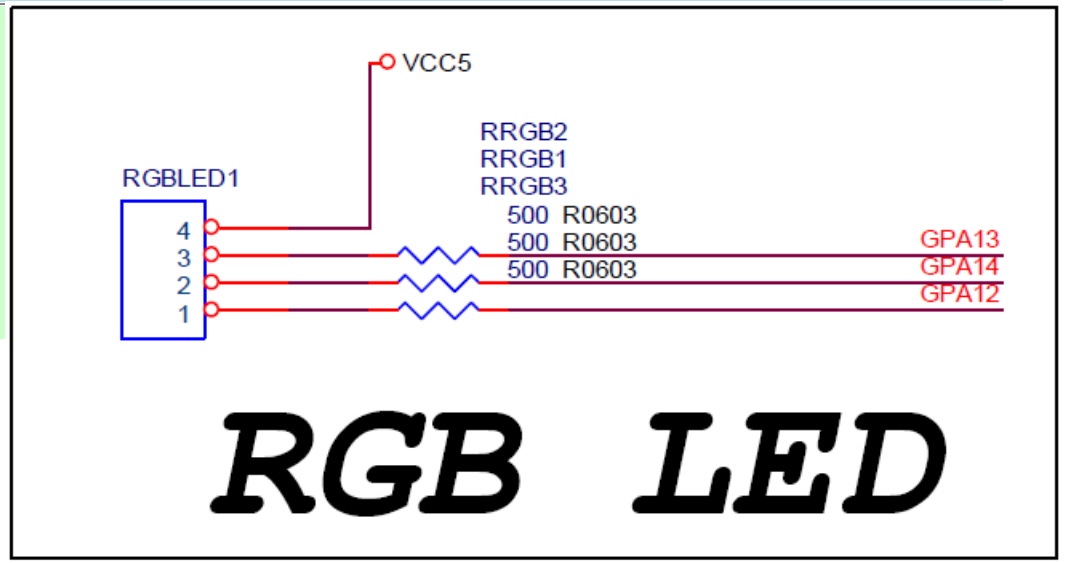
RTC13：Test_RTC

- RGB LED : GPA12,13,14

GPA12 : Blue　0 = on, 1 = off

GPA13 : Green　0 = on, 1 = off

GPA14 : Red　0 = on, 1 = off



- 寫一程式，在LCD顯示目前時間的日期和時間。在7段顯示器顯示分和秒。

RTC13：Test_RTC 1/x

```
int32_t main (void)
{
        char text0[16],text1[16];
        text0[2]='/';
        text0[5]='/';
        text0[8]=0x0;
        text1[2]=':';
        text1[5]=':';
        text1[8]=0x0;
```

RTC13：Test_RTC 2/x

```
// initial RTC
Init_RTC();
//=1 for monday
//=1 for 24 hour mode
Write_RTC(DRVRTC_CURRENT_TIME,2013,4,8,9,10,5,1,1);
//: set time tick period for periodic time tick Interrupt
DrvRTC_SetTickMode (DRVRTC_TICK_1_128_SEC) ;
//: enable specified interrupt and install callback function
DrvRTC_EnableInt(DRVRTC_TICK_INT, RTC_TickCallBack);
```

RTC13：Test_RTC 3/x

```
        Initial_pannel();  //initial LCD
        clr_all_pannal();  //clear LCD
        while(1)
        {
                get_date(text0);  //get year/month/day
                print_lcd(0, text0);
                get_time(text1); //get hour/minute/second
                print_lcd(1, text1);
        }
}
```

RTC13：Test_RTC 4/x

```
// get year,month,day
void get_date(char date[])
{
        date[0]=RTC->CLR.YEAR10+'0'; //year convert to ASCII code
        date[1]=RTC->CLR.YEAR1+'0';
        date[3]=RTC->CLR.MON10+'0'; //month convert to ASCII code
        date[4]=RTC->CLR.MON1+'0';
        date[6]=RTC->CLR.DAY10+'0'; //day convert to ASCII code
        date[7]=RTC->CLR.DAY1+'0';
}
```

RTC13：Test_RTC 5/x

```
void get_time(char time[])
{
        time[0]=RTC->TLR.HR10+'0'; //hour convert to ASCII code
        time[1]=RTC->TLR.HR1+'0';
        time[3]=RTC->TLR.MIN10+'0'; //minute convert to ASCII code
        time[4]=RTC->TLR.MIN1+'0';
        time[6]=RTC->TLR.SEC10+'0'; //second convert to ASCII code
        time[7]=RTC->TLR.SEC1+'0';
}
```

RTC13：Test_RTC 6/x

```
// Display a number on 4-digit 7-seg. Every 1/128=7.8ms
void RTC_TickCallBack(void)
{
        //70clock,3.16us(22M),5.83us(12M)
        seg_minsec();
        //817clock,36.9us(22M),68.1us(12M)
        seg_display();
}
```

RTC13：Test_RTC 7/x

```
// Initial RTC
void Init_RTC(void)
{
        UNLOCKREG();
        DrvRTC_Init() ;          //: (DrvRTC.c) Initial RTC
        //: Enable 32Khz clock source for RTC
        //SYSCLK->PWRCON.XTL32K_EN = 1;
        //: Enable RTC clock source
        //SYSCLK->APBCLK.RTC_EN =1;
        //RTC->INIR = 0xa5eb1357;          //: RTC Initiation
        LOCKREG();
}
```

RTC13：Test_RTC 8/x

```
void Write_RTC(E_DRVRTC_TIME_SELECT eTime,uint16_t yy,uint8_t mm,uint8_t dd,uint8_t hh,uint8_t mn,uint8_t ss,uint8_t wd,uint8_t h24)
{
        S_DRVRTC_TIME_DATA_T RTCTime; // Initial set date and time
        RTCTime.u32Year = yy;
        RTCTime.u32cMonth = mm;
        RTCTime.u32cDay = dd;
        RTCTime.u32cHour = hh;
        RTCTime.u32cMinute = mn;
        RTCTime.u32cSecond = ss;
        RTCTime.u32cDayOfWeek = wd; //=1 for monday
        RTCTime.u8cClockDisplay = h24; //=1 for 24 hour mode
        DrvRTC_Write(eTime,&RTCTime);
}
```

RTC13：Test_RTC 9/x

```
// Display minute and second on 4-digit 7-seg.
void seg_minsec(void)
{
        digit_7seg[3]=RTC->TLR.MIN10;
        digit_7seg[2]=RTC->TLR.MIN1;
        digit_7seg[1]=RTC->TLR.SEC10;
        digit_7seg[0]=RTC->TLR.SEC1;
}
```

RTC13：Test_RTC 10/x

```
// display digit on 7-seg.
void seg_display(void)
{
        //: show a digit on 7-seg.
        close_seven_segment();
        show_seven_segment(light_7seg,digit_7seg[light_7seg]);
        if(!light_7seg--)light_7seg=3;
}
```

RTC14：Test_RTC

- 寫一程式，顯示目前時間的分和秒。
- 到達RTC alarm時間，亮紅色LED，在LCD顯示alarm時間。
- 延遲2秒，關閉紅色LED，亮綠色LED，在LCD顯示alarm時間。
- 延遲5秒，關閉綠色LED。
- 說明：
- 1. 設定時間 2013/3/19 13:20:0
- 2. Alarm time 2013/3/19 13:20:5
- 3. 13:20:5 亮紅色LED
- 4. 13:20:7 關紅色LED，亮綠色LED
- 5. 13:20:12關綠色LED

RTC14：Test_RTC 1/x

```
int32_t main (void)
{
        // Initial LED: set GPIO GPA12,13,14 to output mode
        Init_RGBLED();
        // Initial RTC
        Init_RTC();
        // Open RTC
        Write_RTC(DRVRTC_CURRENT_TIME,2013,4,8,9,10,5,1,1);
        // set Alarm date and time
        Write_RTC(DRVRTC_ALARM_TIME,2013,4,8,9,10,10,1,1);
```

RTC14：Test_RTC 2/x

```
        // set Time Tick period and interrupt
        //: DRVRTC_TICK_1_128_SEC : Time tick is 1/128 second
        DrvRTC_SetTickMode (DRVRTC_TICK_1_128_SEC) ;
        //: enable specified interrupt and install callback function
        DrvRTC_EnableInt(DRVRTC_ALARM_INT,
RTC_AlarmCallBack);
        DrvRTC_EnableInt(DRVRTC_TICK_INT,
RTC_TickCallBack);
        // Initial LCD
        Initial_pannel();
        clr_all_pannal();
```

RTC14：Test_RTC 2/x

```
        while(1)
        {
                //get_date(&yy,&y1,&mm,&m1,&dd,&d1);
                get_date(date_array);
                //92297clock,4173us(22M),7691us(12M)
                print_lcd(0, date_array);
                //1108clock,50.1us(22M),92.3us(12M)
                get_time(time_array);
                //93284clock,4217us(22M),7774us(12M)
                print_lcd(1, time_array);
        }
}
```

RTC14：Test_RTC 3/x

```
void RTC_TickCallBack(void)
{
        // insert minute and second to 7 segment display
        seg_minsec();
        // show 7 segment display
        seg_display();
}
```

RTC14：Test_RTC 4/x

```
void RTC_AlarmCallBack(void)
{
    S_DRVRTC_TIME_DATA_T curTime;  //variable
    DrvGPIO_ClrBit(E_GPA,14); //turn on red LED
    // get current time
    DrvRTC_Read(DRVRTC_CURRENT_TIME, &curTime);
    delaytime(&curTime, 2); //add 2 seconds to current time
    // set alarm time
    DrvRTC_Write(DRVRTC_ALARM_TIME,&curTime);
    // enable alarm interrupt
    DrvRTC_EnableInt(DRVRTC_ALARM_INT,
RTC_AlarmCallBack1);
}
```

RTC14：Test_RTC 5/x

```
// RTC Alarm Routine 1
void RTC_AlarmCallBack1(void)
{
        S_DRVRTC_TIME_DATA_T curTime;  // variable
        DrvGPIO_SetBit(E_GPA,14);    // turn off red LED
        DrvGPIO_ClrBit(E_GPA,13);    //turn on green LED
        DrvRTC_Read(DRVRTC_CURENT_TIME, &curTime);
        delaytime(&curTime, 5);  // add 5 seconds to current time
        DrvRTC_Write(DRVRTC_ALARM_TIME,&curTime);
        DrvRTC_EnableInt(DRVRTC_ALARM_INT,
RTC_AlarmCallBack2);
}
```

RTC14：Test_RTC 6/x

```
// RTC Alarm Routine 2
void RTC_AlarmCallBack2(void)
{
        // turn off green LED
        DrvGPIO_SetBit(E_GPA,13);
}
```

RTC14：Test_RTC 7/x

```
void delaytime(S_DRVRTC_TIME_DATA_T *timePt, uint16_t addsec)
{
        timePt->u32cSecond += addsec;  // add seconds
        if(timePt->u32cSecond > 59)  // next minute
        {
                timePt->u32cSecond -= 60;
                timePt->u32cMinute ++;
                if(timePt->u32cMinute > 59)  //next hour
                {
                        timePt->u32cMinute -= 60;
                        timePt->u32cHour ++;
```

RTC14：Test_RTC 8/x

```
if(timePt->u32cHour > 23)   //next day
{
        timePt->u32cHour -= 24;
        timePt->u32cDay ++;
        // in February with 28 days
        if((timePt->u32cDay == 29) && (timePt->u32cMonth == 2) &&
(timePt->u32Year % 4))
        {
                timePt->u32cDay =1;
                timePt->u32cMonth ++;
        }
```

RTC14：Test_RTC 9/x

```
        if((timePt->u32cDay == 30) && (timePt->u32cMonth == 2) &&
!(timePt->u32Year % 4))          // in February with 29 days
        {
                timePt->u32cDay =1;
                timePt->u32cMonth ++;
        }
        if((timePt->u32cDay == 31) && ((timePt->u32cMonth ==
4)||(timePt->u32cMonth == 6)||(timePt->u32cMonth == 9)||(timePt-
>u32cMonth == 11)) )   // in {April, June, September, November}
        {
                timePt->u32cDay =1;
                timePt->u32cMonth ++;
        }
```

RTC14：Test_RTC 10/x

```
        if(timePt->u32cDay == 32)
        {
                timePt->u32cDay =1;
                timePt->u32cMonth ++;
                if(timePt->u32cMonth == 13)
                {
                        timePt->u32cMonth =1;
                        timePt->u32Year++;
                }
        }
}
```

RTC14：Test_RTC 11/x

```
void Init_RGBLED(void)
{
                // initial GPIO GPA pin  12 to output mode
                DrvGPIO_Open(E_GPA, 12, E_IO_OUTPUT);
                //GPIOA->PMD.PMD12 = 0x01;
                // initial GPIO GPA pin  13 to output mode
                DrvGPIO_Open(E_GPA, 13, E_IO_OUTPUT);
                //GPIOA->PMD.PMD13 = 0x01;
                // initial GPIO GPA pin  14 to output mode
                DrvGPIO_Open(E_GPA, 14, E_IO_OUTPUT);
                //GPIOA->PMD.PMD14 = 0x01;
}
```

RTC14：Test_RTC 12/x

```
void Init_RTC(void)
{
        UNLOCKREG();
        DrvRTC_Init() ;             //: (DrvRTC.c) Initial RTC
        //: Enable 32Khz clock source for RTC
        //SYSCLK->PWRCON.XTL32K_EN = 1;
        //: Enable RTC clock source
        //SYSCLK->APBCLK.RTC_EN =1;
        //: RTC Initiation
        //RTC->INIR = 0xa5eb1357;
        LOCKREG();
}
```

RTC14：Test_RTC 13/x

```
void Write_RTC(E_DRVRTC_TIME_SELECT eTime,uint16_t yy,uint8_t mm,uint8_t dd,uint8_t hh,uint8_t mn,uint8_t ss,uint8_t wd,uint8_t h24)
{
        S_DRVRTC_TIME_DATA_T RTCTime;
        RTCTime.u32Year = yy;
        RTCTime.u32cMonth = mm;
        RTCTime.u32cDay = dd;
        RTCTime.u32cHour = hh;
        RTCTime.u32cMinute = mn;
        RTCTime.u32cSecond = ss;
        RTCTime.u32cDayOfWeek = wd; //=1 for monday
        RTCTime.u8cClockDisplay = h24; //=1 for 24 hour mode
        DrvRTC_Write(eTime,&RTCTime);
}
```

General Disclaimer

The Lecture is strictly used for educational purpose.

MAKES NO GUARANTEE OF VALIDITY

- **The lecture cannot guarantee the validity of the information found here.** The lecture may recently have been changed, vandalized or altered by someone whose opinion does not correspond with the state of knowledge in the relevant fields. Note that most other encyclopedias and reference works also have similar disclaimers.

No formal peer review

- The lecture is not uniformly peer reviewed; while readers may correct errors or engage in casual peer review, they have no legal duty to do so and thus all information read here is without any implied warranty of fitness for any purpose or use whatsoever. Even articles that have been vetted by informal peer review or *featured article* processes may later have been edited inappropriately, just before you view them.

No contract; limited license

- Please make sure that you understand that the information provided here is being provided freely, and that no kind of agreement or contract is created between you and the owners or users of this site, the owners of the servers upon which it is housed, the individual Wikipedia contributors, any project administrators, sysops or anyone else who is in *any way connected* with this project or sister projects subject to your claims against them directly. You are being granted a limited license to copy anything from this site; it does not create or imply any contractual or extracontractual liability on the part of Wikipedia or any of its agents, members, organizers or other users.
- There is **no agreement or understanding between you and the content provider** regarding your use or modification of this information beyond the Creative Commons Attribution-Sharealike 3.0 Unported License (CC-BY-SA) and the GNU Free Documentation License (GFDL);

General Disclaimer

Trademarks

- Any of the trademarks, service marks, collective marks, design rights or similar rights that are mentioned, used or cited in the lectures are the property of their respective owners. Their use here does not imply that you may use them for any purpose other than for the same or a similar informational use as contemplated by the original authors under the CC-BY-SA and GFDL licensing schemes. Unless otherwise stated , we are neither endorsed by nor affiliated with any of the holders of any such rights and as such we cannot grant any rights to use any otherwise protected materials. Your use of any such or similar incorporeal property is at your own risk.

Personality rights

- The lecture may portray an identifiable person who is alive or deceased recently. The use of images of living or recently deceased individuals is, in some jurisdictions, restricted by laws pertaining to personality rights, independent from their copyright status. Before using these types of content, please ensure that you have the right to use it under the laws which apply in the circumstances of your intended use. *You are solely responsible for ensuring that you do not infringe someone else's personality rights.*