King Mongkut's University of Technology Thonburi

Department of Electronics and Telecommunication Engineering Faculty of Engineering

EIE/ENE 335 Digital Circuit and Microprocessor Lab for the 3rd year student

Experiment: RTC

Objectives

- How to use
    - the NuMicro™ NUC100 series driver to do the fast application software development
    - RTC

Background Theory

RTC

Real Time Clock (RTC) controller provides user the real time and calendar message. The RTC controller provides the time message (second, minute, hour) in Time Loading Register (TLR) as well as calendar message (day, month, year) in Calendar Loading Register (CLR). The data message is expressed in BCD format. It also offers alarm function that user can preset the alarm time in Time Alarm Register (TAR) and alarm calendar in Calendar Alarm Register (CAR).

The RTC controller supports periodic Time Tick and Alarm Match interrupts. The periodic interrupt has 8 period options 1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2 and 1 second which are selected by TTR (TTR[2:0]). When RTC counter in TLR and CLR is equal to alarm setting time registers TAR and CAR, the alarm interrupt flag (RIIR.AIF) is set and the alarm interrupt is requested if the alarm interrupt is enabled (RIER.AIER=1). Both RTC Time Tick and Alarm Match can cause chip wakeup from power down mode if wake-up function is enabled (TWKE (TTR[3])=1).

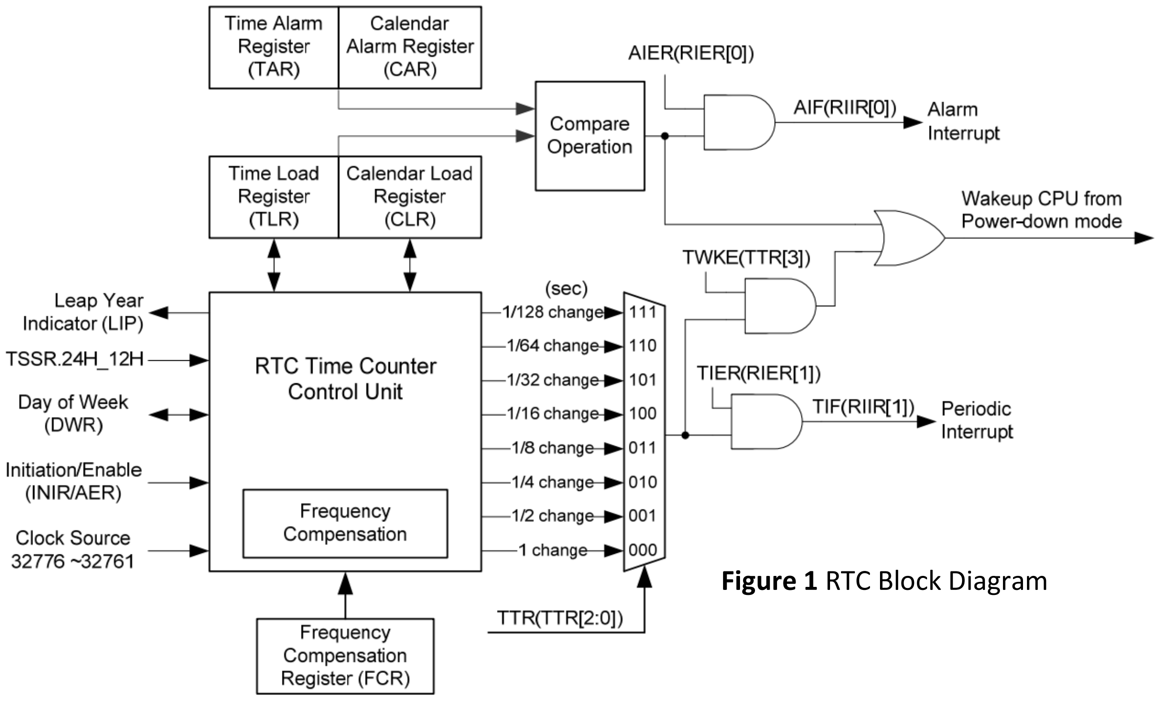


Figure 1 RTC Block Diagram

Equipment required

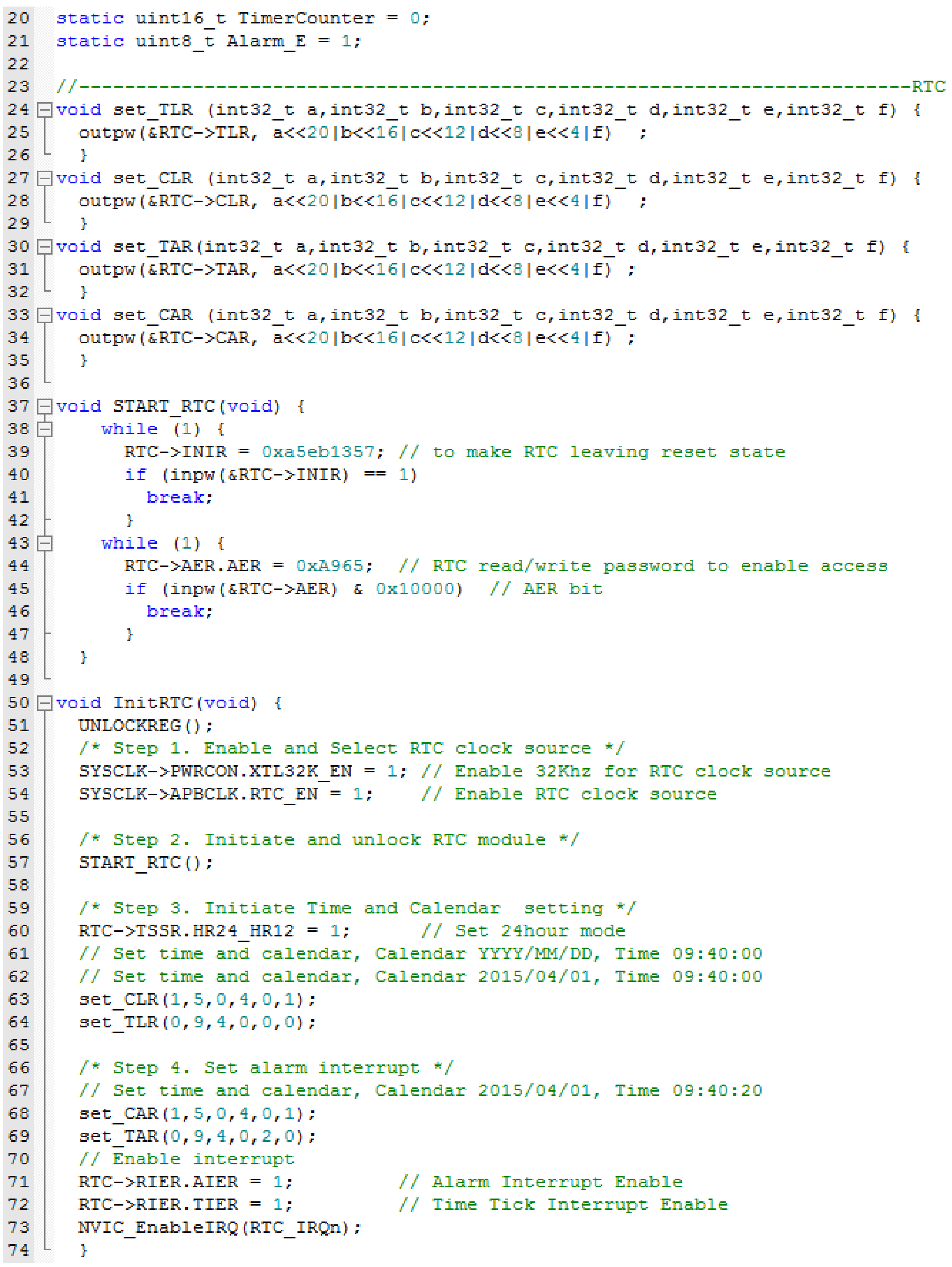
- Nu_LB-002 (Nuvoton learning board)

Reference:

1. Nu_LB-002 Rev 2.1 User's Manual
2. NuMicro™ NUC130_140 Technical Reference Manual EN V2.02
3. NuMicro™ NUC100 Series Driver Reference Guide V1.05.002

Procedure 1: RTC

1. Replace the content of the '**Smpl_Start_Kit.c**' with the '**RTC**' lab file.
2. Compile the project, and run the program.
3. Study the program and work on assignments in the class.



```c
static uint16_t TimerCounter = 0;
static uint8_t Alarm_E = 1;

//-----------------------------------------------------------------------RTC
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) ;
  }

void START_RTC(void) {
    while (1) {
      RTC->INIR = 0xa5eb1357; // to make RTC leaving reset state
      if (inpw(&RTC->INIR) == 1)
        break;
      }
    while (1) {
      RTC->AER.AER = 0xA965;  // RTC read/write password to enable access
      if (inpw(&RTC->AER) & 0x10000)   // AER bit
        break;
      }
  }

void InitRTC(void) {
  UNLOCKREG();
  /* Step 1. Enable and Select RTC clock source */
  SYSCLK->PWRCON.XTL32K_EN = 1; // Enable 32Khz for RTC clock source
  SYSCLK->APBCLK.RTC_EN = 1;    // Enable RTC clock source

  /* Step 2. Initiate and unlock RTC module */
  START_RTC();

  /* Step 3. Initiate Time and Calendar  setting */
  RTC->TSSR.HR24_HR12 = 1;      // Set 24hour mode
  // Set time and calendar, Calendar YYYY/MM/DD, Time 09:40:00
  // Set time and calendar, Calendar 2015/04/01, Time 09:40:00
  set_CLR(1,5,0,4,0,1);
  set_TLR(0,9,4,0,0,0);

  /* Step 4. Set alarm interrupt */
  // Set time and calendar, Calendar 2015/04/01, Time 09:40:20
  set_CAR(1,5,0,4,0,1);
  set_TAR(0,9,4,0,2,0);
  // Enable interrupt
  RTC->RIER.AIER = 1;         // Alarm Interrupt Enable
  RTC->RIER.TIER = 1;         // Time Tick Interrupt Enable
  NVIC_EnableIRQ(RTC_IRQn);
  }
```

```c
//-----------------------------------------------------------------RTC_IRQ
void RTC_IRQHandler(void) {   // default every 1 s.
  uint32_t clock;
  uint32_t date;
  char lcd_line0[15] = "Clock:";
  char lcd_line1[15] = "Date:20";

  /* tick */
  if (inpw(&RTC->RIIR) & 0x2) { // TIF = 1?
    clock = inpw(&RTC->TLR) & 0xFFFFFF;
    sprintf(lcd_line0+6, "%02x", (clock >> 16) & 0xFF);
    sprintf(lcd_line0+9, "%02x", (clock >> 8) & 0xFF);
    sprintf(lcd_line0+12, "%02x", (clock & 0xFF));
    lcd_line0[8] = ':';
    lcd_line0[11] = ':';
    Show_Word(0,13, ' ');
    print_lcd(0, lcd_line0);

    date = inpw(&RTC->CLR) & 0xFFFFFF;
    sprintf(lcd_line1+7, "%02x", (date >> 16) & 0xFF);
    sprintf(lcd_line1+10, "%02x", (date >> 8) & 0xFF);
    sprintf(lcd_line1+13, "%02x", date & 0xFF);
    lcd_line1[9] = '/';
    lcd_line1[12] = '/';
    Show_Word(1, 13, ' ');
    print_lcd(1, lcd_line1);

    outpw(&RTC->RIIR, 2); // clear RTC Time Tick Interrupt Flag
  }

  /* alarm */
  if (inpw(&RTC->RIIR) & 0x1) { // AIF = 1?
    print_lcd(1, "Alarm!!!!");
    GPIOC->DOUT &= 0xFF;        // LED5-8 on
    Alarm_E = 0;

    outpw(&RTC->RIIR, 1);       // clear RTC Alarm Interrupt Flag
  }
}
//---------------------------------------------------------------------WDT
void InitWDT(void) {
  UNLOCKREG();
  /* Step 1. Enable and Select WDT clock source */
  SYSCLK->CLKSEL1.WDT_S = 3;  // Select 10kHz for WDT clock source
  SYSCLK->APBCLK.WDT_EN = 1;  // Enable WDT clock source
  /* Step 2. Select Timeout Interval */
  WDT->WTCR.WTIS = 5;         // 1.63 - 1.74 s.
  /* Step 3. Disable Watchdog Timer Reset function */
  WDT->WTCR.WTRE = 0;
  /* Step 4. Enable WDT interrupt */
  WDT->WTCR.WTIF = 1;         // clear watchdog Timer Interrupt Flag
  WDT->WTCR.WTIE = 1;
  NVIC_EnableIRQ(WDT_IRQn);
  /* Step 5. Enable WDT module */
  WDT->WTCR.WTE = 1;          // Enable WDT
  WDT->WTCR.WTR = 1;          // Clear WDT counter
  LOCKREG();
  }
//-----------------------------------------------------------------WDT_IRQ
void WDT_IRQHandler(void) {
  UNLOCKREG();
  WDT->WTCR.WTIF = 1;         // clear watchdog Timer Interrupt Flag
  WDT->WTCR.WTR = 1;          // reset the contents of watchdog timer
  UNLOCKREG();
  print_lcd(3, "WDT interrupt");
  }
```

```
//-------------------------------------------------------------------TIMER0
void InitTIMER0(void) {
  /* Step 1. Enable and Select Timer clock source */
  SYSCLK->CLKSEL1.TMR0_S = 0; // Select 12Mhz for Timer0 clock source
  SYSCLK->APBCLK.TMR0_EN = 1; // Enable Timer0 clock source

  /* Step 2. Select Operation mode */
  TIMER0->TCSR.MODE = 1;      // Select periodic mode for operation mode

  /* Step 3. Select Time out period =
  (Period of timer clock input) * (8-bit Prescale + 1) * (24-bit TCMP)*/
  TIMER0->TCSR.PRESCALE = 255;  // Set Prescale [0~255]
  TIMER0->TCMPR = 2765;         // Set TCMPR [0~16777215]
  // (1/12000000)*(255+1)*(2765)= 125.01usec or 7999.42Hz

  /* Step 4. Enable interrupt */
  TIMER0->TCSR.IE = 1;
  TIMER0->TISR.TIF = 1;       // Write 1 to clear TIF
  NVIC_EnableIRQ(TMR0_IRQn);  // Enable Timer0 Interrupt

  /* Step 5. Enable Timer module */
  TIMER0->TCSR.CRST = 1;    // Reset up counter
  TIMER0->TCSR.CEN = 1;     // Enable Timer0
  }
//---------------------------------------------------------------Timer0_IRQ
void TMR0_IRQHandler(void) {  // Timer0 interrupt subroutine
  char lcd_line2[12] = "Timer0:";
  TimerCounter += 1;
  sprintf(lcd_line2+7, "%d", TimerCounter);
  print_lcd(2, lcd_line2);

  TIMER0->TISR.TIF = 1;      // Write 1 to clear TIF
  }

//---------------------------------------------------------------------MAIN
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_pannel();  //call initial pannel function
  clr_all_pannal();

  InitTIMER0();
  InitRTC();
  InitWDT();

  while (Alarm_E) {
    UNLOCKREG();
    WDT->WTCR.WTR = 1;  // Reset the contents of WDT
    LOCKREG();
  }
  while (1) {
    __NOP();
    }
  }
```

Assignment(s)

Summarize what you suppose to learn in this class.