```
/**************************************************************************//**
 * @file     main.c
 * @version  V3.00
 * $Revision: 3 $
 * $Date: 14/01/28 11:44a $
 * @brief    M051 Series GPIO Driver Sample Code
 *
 * @note
 * Copyright (C) 2013 Nuvoton Technology Corp. All rights reserved.
 ******************************************************************************/
#include <stdio.h>
#include "M051Series.h"


#define PLL_CLOCK           50000000

volatile uint32_t g_au32TMRINTCount[1] = {0};
volatile uint32_t u32InitCount;


/**
 * @brief       Port2/Port3/Port4 IRQ
 *
 * @param       None
 *
 * @return      None
 *
 * @details     The Port2/Port3/Port4 default IRQ, declared in startup_M051Series.s.
 */
void GPIOP2P3P4_IRQHandler(void)
{   //display=GPIO_GET_INT_FLAG(P2, BIT4);
    /* To check if P4.5 interrupt occurred */
    if(GPIO_GET_INT_FLAG(P2, BIT4))
    {
        GPIO_CLR_INT_FLAG(P2, BIT4);
        printf("\n P2.4 INT occurred.\n");
    }
    else
    {
        /* Un-expected interrupt. Just clear all PORT2, PORT3 and PORT4 interrupts */
        P2->ISRC = P2->ISRC;
        P3->ISRC = P3->ISRC;
        P4->ISRC = P4->ISRC;
        printf("Un-expected interrupts.\n");
    }
}

void TMR0_IRQHandler(void)
{
    if(TIMER_GetIntFlag(TIMER0) == 1)
    {
        /* Clear Timer1 time-out interrupt flag */
        TIMER_ClearIntFlag(TIMER0);
        //P00 = !P00;
       //printf("\n\nuart1: 115200\n");
    }
}

/**
 * @brief       Timer2 IRQ
 *
 * @param       None
 *
 * @return      None
 *
 * @details     The Timer1 default IRQ, declared in startup_M051Series.s.
 */
volatile uint32_t capture_value[52][2];
volatile uint32_t capture_data[52];
volatile uint32_t capture_count=0;
volatile uint32_t capture_value_count=0;
volatile uint32_t capture_T1=0;
volatile uint32_t capture_T2=0;
volatile uint32_t capture_T3=0;
volatile uint32_t display = 0;
volatile uint32_t TIMER_CAPTURE_EDGE_flag=0;
volatile uint32_t u32AdcIntFlag=0;
```

```c

void TMR2_IRQHandler(void)
{
    if(TIMER_GetCaptureIntFlag(TIMER2) == 1)
    {
        /* Clear Timer2 capture interrupt flag */
        TIMER_ClearCaptureIntFlag(TIMER2);

        if(TIMER_CAPTURE_EDGE_flag==0)  { TIMER_CAPTURE_EDGE_flag=1;TIMER_EnableCapture(TIMER2, TIMER_CAPTURE_FREE_COUNTING_MODE, TIMER_CAPTURE_FALLING_EDGE);}
        else                            { TIMER_CAPTURE_EDGE_flag=0;TIMER_EnableCapture(TIMER2, TIMER_CAPTURE_FREE_COUNTING_MODE, TIMER_CAPTURE_RISING_EDGE); }


        switch(capture_count-capture_count/3*3)
          {
          case  0:
                   capture_T1 = TIMER_GetCaptureData(TIMER2);
          break;

          case  1:
                   capture_T2 = TIMER_GetCaptureData(TIMER2);
          break;

          case  2:
                  capture_T3 = TIMER_GetCaptureData(TIMER2);
                   if(capture_value_count==1)
                     display=0;
                  if(capture_T2 > capture_T1)
                  {
                     capture_value[capture_value_count][0]   = (capture_T2-capture_T1);
                     capture_value[capture_value_count][0]   = capture_value[capture_value_count][0]/12;
                  }
                  else
                  {  capture_value[capture_value_count][0]   = (0xffffff+capture_T2-capture_T1);//width}
                     capture_value[capture_value_count][0]   = capture_value[capture_value_count][0]/12;//width}
                  }

                  if(capture_T3 > capture_T2 )
                  {  capture_value[capture_value_count][1]    = (capture_T3-capture_T2);//low
                     capture_value[capture_value_count][1]    = capture_value[capture_value_count][1]/12;
                  }
                  else
                  {  capture_value[capture_value_count][1]     = (0xffffff+capture_T3-capture_T2);//low
                     capture_value[capture_value_count][1]     = capture_value[capture_value_count][1]/12;
                  }


                  capture_value_count++;

                  capture_count++;
                  capture_T1  = capture_T3;

                  if(capture_value_count==51)
                    capture_value_count=0;

          break;

          default:break;;
          }

          capture_count++;

        }
//    if(TIMER_GetIntFlag(TIMER2)==1)
}

//uint32_t g_u32AdcIntFlag=0;
volatile uint32_t NUM=0;
volatile uint32_t ConversionData[4];
volatile uint32_t i32ConversionData=0;
```

```c
void ADC_IRQHandler(void)
{

    ADC_CLR_INT_FLAG(ADC, ADC_ADF_INT); /* clear the A/D conversion flag */
    ConversionData[NUM] = ADC_GET_CONVERSION_DATA(ADC, 4);
    NUM++;
    if(NUM==4)
    {  NUM=0;u32AdcIntFlag = 1;
       i32ConversionData=(ConversionData[0]+ConversionData[1]+ConversionData[2]+ConversionData[3])>>2;
    }
}

void GPIO_Init(void)
{
  /* Configure P2.4 as Input mode and enable interrupt by FALLING edge trigger */
    GPIO_SetMode(P2, BIT4, GPIO_PMD_INPUT);
    GPIO_EnableInt(P2, 4, GPIO_INT_FALLING);
    NVIC_EnableIRQ(GPIO_P2P3P4_IRQn);
    GPIO_ENABLE_DEBOUNCE(P2, BIT4);//去抖
    GPIO_SET_DEBOUNCE_TIME(GPIO_DBCLKSRC_LIRC, GPIO_DBCLKSEL_16);//16x0.1ms(10k)

  /* Configure P0.0 as OUTPUT mode*/
    GPIO_SetMode(P0, BIT0, GPIO_PMD_OUTPUT);


    GPIO_SetMode(P4, BIT0, GPIO_PMD_INPUT);//t2ex
    //GPIO_EnableInt(P2, 4, GPIO_INT_FALLING);
   // NVIC_EnableIRQ(GPIO_P2P3P4_IRQn);
   // GPIO_ENABLE_DEBOUNCE(P4, BIT0);//去抖
   // GPIO_SET_DEBOUNCE_TIME(GPIO_DBCLKSRC_LIRC, GPIO_DBCLKSEL_16);//16x0.1ms(10k)



    //GPIO_EnableInt(P2, 4, GPIO_INT_FALLING);
    //NVIC_EnableIRQ(GPIO_P2P3P4_IRQn);
    //GPIO_ENABLE_DEBOUNCE(P2, BIT4);//去抖
   // GPIO_SET_DEBOUNCE_TIME(GPIO_DBCLKSRC_LIRC, GPIO_DBCLKSEL_16);//16x0.1ms(10k)

}


void ADC_Init(void)
{

    CLK_EnableModuleClock(ADC_MODULE);

    /* ADC clock source is 22.1184MHz, set divider to 7, ADC clock is 22.1184/7 MHz */
    CLK_SetModuleClock(ADC_MODULE,CLK_CLKSEL1_ADC_S_HXT, CLK_CLKDIV_ADC(96));
                                  //CLK_CLKSEL1_ADC_S_HIRC
   // CLK_EnableModuleClock(ADC_MODULE);


//  /* Disable the P1.0 - P1.4 digital input path to avoid the leakage current */
    GPIO_DISABLE_DIGITAL_PATH(P1,BIT4);
    GPIO_SetMode(P1, BIT4, GPIO_PMD_INPUT);
//
   /* Configure the P1.0 - P1.3 ADC analog input pins */
   SYS->P1_MFP &= ~(SYS_MFP_P14_Msk);
   SYS->P1_MFP |= SYS_MFP_P14_AIN4 ;

   //SYS_ResetModule(ADC_RST);
   ADC_SET_INPUT_CHANNEL(ADC, BIT4);

    /* Set the ADC operation mode as CONTINUOUS, input mode as single-end and enable the analog input channel 4 */
            //ADC_Open(ADC, ADC_ADCR_DIFFEN_SINGLE_END, ADC_ADCR_ADMD_SINGLE,0x1 << 4);
            ADC_Open(ADC, ADC_ADCR_DIFFEN_SINGLE_END, ADC_ADCR_ADMD_CONTINUOUS,BIT4);
            /* Power on ADC module */
            ADC_POWER_ON(ADC);

            /* clear the A/D interrupt flag for safe */
            ADC_CLR_INT_FLAG(ADC, ADC_ADF_INT);
            ADC_DisableHWTrigger(ADC);


            /* Enable the ADC interrupt */
            ADC_EnableInt(ADC, ADC_ADF_INT);
            NVIC_EnableIRQ(ADC_IRQn);
}

void ADC_Process(void)
{

  ADC_START_CONV(ADC);
  while(1)
    {

            CLK_SysTickDelay(2000000);
           // P00 = !P00;

            /* Reset the ADC interrupt indicator and Start A/D conversion */
            //u32AdcIntFlag = 0;
           // ADC_START_CONV(ADC);
             //ADC_DisableInt(ADC, ADC_ADF_INT);
            /* Wait ADC interrupt (g_u32AdcIntFlag will be set at IRQ_Handler function)*/
            if(u32AdcIntFlag == 1)
              {

            /* Disable the ADC interrupt */
            //ADC_DisableInt(ADC, ADC_ADF_INT);
            u32AdcIntFlag = 0;
            /* Get the conversion result of the ADC channel 4 */
            i32ConversionData = ADC_GET_CONVERSION_DATA(ADC, 4);
            printf("Conversion result of channel 2: 0x%X (%d)\n\n", i32ConversionData, 4960*i32ConversionData/4095);
            P00 = !P00;
            }

    }


}


void SYS_Init(void)
{

/*---------------------------------------------------------------------------------------------------------*/
    /* Init System Clock                                                                                   */
/*---------------------------------------------------------------------------------------------------------*/
    /* Enable Internal RC 22.1184MHz clock */
    CLK_EnableXtalRC(CLK_PWRCON_OSC22M_EN_Msk);
    /* Waiting for Internal RC clock ready */
    CLK_WaitClockReady(CLK_CLKSTATUS_OSC22M_STB_Msk);

    /* Switch HCLK clock source to Internal RC and HCLK source divide 1 */
    CLK_SetHCLK(CLK_CLKSEL0_HCLK_S_HIRC, CLK_CLKDIV_HCLK(1));

    /* Enable external XTAL 12MHz clock */
    CLK_EnableXtalRC(CLK_PWRCON_XTL12M_EN_Msk);

    /* Waiting for external XTAL clock ready */
    CLK_WaitClockReady(CLK_CLKSTATUS_XTL12M_STB_Msk);

    /* Set core clock as PLL_CLOCK from PLL */
    CLK_SetCoreClock(PLL_CLOCK);

   /* Peripheral clock source */
   CLK->CLKSEL1 = CLK_CLKSEL1_TMR2_S_HXT;

    SystemCoreClockUpdate();


}



void UART1_Init(void)
{

```

```
/*---------------------------------------------------------------------------------------------------------*/
    /* Init UART                                                                                           */
/*---------------------------------------------------------------------------------------------------------*/
    /* Enable UART module clock */
    CLK_EnableModuleClock(UART1_MODULE);

    /* Select UART module clock source */
    CLK_SetModuleClock(UART1_MODULE, CLK_CLKSEL1_UART_S_HXT, CLK_CLKDIV_UART(1));//CLK_CLKSEL1_UART_S_HXT或则CLK_CLKSEL1_UART_S_PLL

    /* Set P1 multi-function pins for UART1 RXD and TXD */
      SYS->P1_MFP &= ~(SYS_MFP_P13_Msk | SYS_MFP_P12_Msk);
      SYS->P1_MFP |= (SYS_MFP_P12_RXD1 | SYS_MFP_P13_TXD1);

    /* Reset UART */
    SYS_ResetModule(UART1_RST);

    /* Configure UART1 and set UART1 Baudrate */
    UART_Open(UART1, 115200);

}


void Timer_Init(void)
{
 /* Enable peripheral clock */
    CLK_EnableModuleClock(TMR0_MODULE);

 /* Open Timer0 frequency to 1 Hz in periodic mode, and enable interrupt */
    TIMER_Open(TIMER0, TIMER_PERIODIC_MODE, 2000);

    TIMER_EnableInt(TIMER0);

 /* Enable Timer0  NVIC */
    NVIC_EnableIRQ(TMR0_IRQn);

 /* Start Timer0  counting */
    TIMER_Start(TIMER0);


  //init T2EX----------------------------------------------------------------------------
     //Init I/O Multi-function
     SYS->P4_MFP &= ~(SYS_MFP_P40_Msk );
     SYS->P4_MFP |= SYS_MFP_P40_T2EX;

   /* Enable peripheral clock */
    CLK_EnableModuleClock(TMR2_MODULE);

  /* Initial Timer1 default setting */
    TIMER_Open(TIMER2, TIMER_CONTINUOUS_MODE, 1);
    //TIMER_Open(TIMER2, TIMER_PERIODIC_MODE, 1000);

    /* Configure Timer1 setting for external counter input and capture function */
    TIMER_SET_PRESCALE_VALUE(TIMER2, 0);
    TIMER_SET_CMP_VALUE(TIMER2, 0xffffff);//2*24次方/12000=1.3s
    //TIMER_EnableEventCounter(TIMER2, TIMER_COUNTER_FALLING_EDGE);
    TIMER_EnableCapture(TIMER2, TIMER_CAPTURE_FREE_COUNTING_MODE, TIMER_CAPTURE_RISING_EDGE);
    TIMER_EnableCaptureInt(TIMER2);
    //TIMER_EnableInt(TIMER2);
    //TIMER_EnableCaptureDebounce(TIMER2);
    /* Enable Timer1 NVIC */
    NVIC_EnableIRQ(TMR2_IRQn);

    /* Clear Timer2 interrupt counts to 0 */
    //u32InitCount = g_au32TMRINTCount[1] = 0;

    /* Start Timer2 counting */
    TIMER_Start(TIMER2);

}

/*---------------------------------------------------------------------------------------------------------*/
```

```c
/* MAIN function                                                                                   */
/*---------------------------------------------------------------------------------------------------------*/
int main(void)
{
//    int32_t i32Err;

    /* Unlock protected registers */
    SYS_UnlockReg();

    /* Init System, peripheral clock and multi-function I/O */
    SYS_Init();

    /* Lock protected registers */
    SYS_LockReg();

    /* Init UART0 for printf */
    UART1_Init();

    Timer_Init();

    GPIO_Init();

    ADC_Init();

    printf("\n\nCPU @ %d Hz\n", SystemCoreClock);
    printf("\n\nTimer0: Clock source 12 MHz; Periodic mode; Enable interrupt; 2 interrupt tick/sec.\n");
    printf("\n\nuart1: 115200\n");
    //ADC_START_CONV(ADC);


//    printf("+-----------------------------------------------+\n");
//    printf("|    P1.2(Output) and P4.1(Input) Sample Code    |\n");
//    printf("+-----------------------------------------------+\n\n");

    while(1)
{
  ADC_Process();
  //P00 = !P00;
  //CLK_SysTickDelay(2000000);
  //CLK_SysTickDelay(100000);
//  CLK_SysTickDelay(100000);
//  CLK_SysTickDelay(100000);
//  CLK_SysTickDelay(100000);
//  CLK_SysTickDelay(100000);
//  CLK_SysTickDelay(100000);
//  CLK_SysTickDelay(100000);
//  CLK_SysTickDelay(100000);
//  CLK_SysTickDelay(100000);




}//while(1)


}



/*** (C) COPYRIGHT 2013 Nuvoton Technology Corp. ***/

```