COE718 - Lab 1

Gabriel Casciano, 500744076
gabriel.casciano@ryerson.ca

Sept, 2019

Blinky.c

```c
/*----------------------------------------------------------------------------
 * Name:    Blinky.c
 * Purpose: LED Flasher
 * Note(s): __USE_LCD   - enable Output on LCD, uncomment #define in code to use
 *          for demo (NOT for analysis purposes)
 *----------------------------------------------------------------------------
 * Copyright (c) 2008-2011 Keil - An ARM Company.
 * Name: Anita Tino
 *----------------------------------------------------------------------------*/

#include <stdio.h>
#include <string.h>
#include "LPC17xx.H"
#include "GLCD.h"
#include "LED.h"
#include "ADC.h"
#include "KDB.h"
#include "Joystick.h"

#define __FI        1                       /* Font index 16x24               */
#define __USE_LCD   1                     /* Uncomment to use the LCD */

//ITM Stimulus Port definitions for printf ////////////////////
#define ITM_Port8(n)    (*((volatile unsigned char *)(0xE0000000+4*n)))
#define ITM_Port16(n)   (*((volatile unsigned short*)(0xE0000000+4*n)))
#define ITM_Port32(n)   (*((volatile unsigned long *)(0xE0000000+4*n)))

#define DEMCR           (*((volatile unsigned long *)(0xE000EDFC)))
#define TRCENA          0x01000000

struct __FILE { int handle;  };
FILE __stdout;
FILE __stdin;

int fputc(int ch, FILE *f) {
  if (DEMCR & TRCENA) {
    while (ITM_Port32(0) == 0);
    ITM_Port8(0) = ch;
  }
  return(ch);
}
////////////////////////////////////////////////////////////

char adc_value[16];
char string[20];
char joystick_value[10];

const unsigned long led_mask1[] = { 1UL<<28, 1UL<<29, 1UL<<31, 1UL<< 2,

                                    1UL<< 3, 1UL<< 4, 1UL<< 5, 1UL<< 6 };

//Use to trace the pot values in Debug
uint16_t ADC_Dbg;

/* Import external variables from IRQ.c file                                  */
extern uint8_t  clock_ms;


```

```c
/*----------------------------------------------------------------------------
  Main Program
 *----------------------------------------------------------------------------*/
int main (void) {
  uint32_t ad_avg = 0;
  uint16_t ad_val = 0, ad_val_ = 0xFFFF;

  LED_Init();                                /* LED Initialization            */
  ADC_Init();                                /* ADC Initialization            */
  KDB_Init();                                /* KDB Initialization            */
  JOYSTICK_Init();                           /* JOYSTICK Initialization       */


#ifdef __USE_LCD
  GLCD_Init();                               /* Initialize graphical LCD (if enabled */

  GLCD_Clear(White);                         /* Clear graphical LCD display   */
  GLCD_SetBackColor(Blue);
  GLCD_SetTextColor(Yellow);
  GLCD_DisplayString(0, 0, __FI, (unsigned char *)"     COE718 Demo    ");
  GLCD_SetTextColor(White);
  GLCD_DisplayString(1, 0, __FI, (unsigned char *)"       Blinky.c     ");
  GLCD_DisplayString(2, 0, __FI, (unsigned char *)"       Something    ");
  GLCD_SetBackColor(White);
  GLCD_SetTextColor(Blue);

#endif

  SysTick_Config(SystemCoreClock/100);       /* Generate interrupt each 10 ms */

  while (1) {                                /* Loop forever                  */

    /* AD converter input                                                     */
    if (AD_done) {                           /* If conversion has finished    */
      AD_done = 0;

      ad_avg += AD_last << 8;                /* Add AD value to averaging     */
      ad_avg ++;
      if ((ad_avg & 0xFF) == 0x10) {         /* average over 16 values        */
        ad_val = (ad_avg >> 8) >> 4;         /* average devided by 16         */
        ad_avg = 0;
      }
    }

    ADC_Dbg = ad_val;

    if (ad_val ^ ad_val_) {                  /* AD value changed              */
        ad_val_ = ad_val;

        sprintf(adc_value, "0x%04X", ad_val);       /* format text for print out     */
    }
    strcpy(string, "ADC: ");
    strcat(string, adc_value);
    GLCD_DisplayString(6, 0, __FI, (unsigned char*)string);
    /* Print message with AD value every 10 ms                                */
    if (clock_ms) {
      clock_ms = 0;

      printf("AD value: %s\r\n", adc_value);
    }
       /* Update Joystick value and displays*/

    strcpy(joystick_value, JOYSTICK_Update());
    strcpy(string, "Joy Value: ");
  if(strcmp(joystick_value, JOYSTICK_UP)){
    LED_Out(0);
  }else if(strcmp(joystick_value, JOYSTICK_DOWN)){
    LED_Out(1);
  }else if(strcmp(joystick_value, JOYSTICK_LEFT)){
    LED_Out(2);
  }else if(strcmp(joystick_value, JOYSTICK_RIGHT)){
    LED_Out(3);
  }else if(strcmp(joystick_value, JOYSTICK_SELECT)){
    LED_Out(4);
  }
    strcat(string, joystick_value);
```

```
    GLCD_DisplayString(7, 0, __FI, (unsigned char*)string);

  }
}
```

ADC.c

```
/*----------------------------------------------------------------------------
 * Name:    ADC.c
 * Purpose: low level ADC functions
 * Note(s): possible defines select the used ADC interface:
 *            __ADC_IRQ   - ADC works in Interrupt mode
 *                        - ADC works in polling mode (default)
 *----------------------------------------------------------------------------
 * This file is part of the uVision/ARM development tools.
 * This software may only be used under the terms of a valid, current,
 * end user licence from KEIL for a compatible version of KEIL software
 * development tools. Nothing else gives you the right to use this software.
 *
 * This software is supplied "AS IS" without warranties of any kind.
 *
 * Copyright (c) 2008-2011 Keil - An ARM Company. All rights reserved.
 *----------------------------------------------------------------------------*/

#include "LPC17xx.H"                        /* LPC17xx definitions           */
#include "ADC.h"

uint16_t AD_last;                           /* Last converted value          */
uint8_t  AD_done = 0;                       /* AD conversion done flag       */

/*----------------------------------------------------------------------------
  Function that initializes ADC
 *----------------------------------------------------------------------------*/
void ADC_Init (void) {

  LPC_SC->PCONP |= ((1 << 12) | (1 << 15));  /* enable power to ADC & IOCON   */

  LPC_PINCON->PINSEL1  &= ~( 3 << 18);
  LPC_PINCON->PINSEL1  |=  ( 1 << 18);      /* P0.25 is AD0.2                */
  LPC_PINCON->PINMODE1 &= ~( 3 << 18);
  LPC_PINCON->PINMODE1 |=  ( 2 << 18);      /* P0.25 no pull up/down         */

  LPC_ADC->ADCR        =  ( 1 <<  2) |      /* select AD0.2 pin              */
                          ( 4 <<  8) |      /* ADC clock is 25MHz/5          */
                          ( 1 << 21);       /* enable ADC                    */

#ifdef __ADC_IRQ
  LPC_ADC->ADINTEN     =  ( 1 <<  8);       /* global enable interrupt       */

  NVIC_EnableIRQ(ADC_IRQn);                 /* enable ADC Interrupt          */
#endif
}


/*----------------------------------------------------------------------------
  start AD Conversion
 *----------------------------------------------------------------------------*/
void ADC_StartCnv (void) {
  LPC_ADC->ADCR &= ~( 7 << 24);             /* stop conversion               */
  LPC_ADC->ADCR |=  ( 1 << 24);             /* start conversion              */
}


/*----------------------------------------------------------------------------
  stop AD Conversion
 *----------------------------------------------------------------------------*/
void ADC_StopCnv (void) {

  LPC_ADC->ADCR &= ~( 7 << 24);             /* stop conversion               */
}


/*----------------------------------------------------------------------------
  get converted AD value
 *----------------------------------------------------------------------------*/
uint16_t ADC_GetCnv (void) {
```

```
70
71 #ifndef __ADC_IRQ
72   while (!(LPC_ADC->ADGDR & ( 1UL << 31)));  /* Wait for Conversion end       */
73   AD_last = (LPC_ADC->ADGDR >> 4) & ADC_VALUE_MAX; /* Store converted value   */
74
75   AD_done = 1;
76 #endif
77
78   return(AD_last);
79 }
80
81
82 /*----------------------------------------------------------------------------
83   A/D IRQ: Executed when A/D Conversion is done
84  *----------------------------------------------------------------------------*/
85 #ifdef __ADC_IRQ
86 void ADC_IRQHandler(void) {
87   volatile uint32_t adstat;
88
89   adstat = LPC_ADC->ADSTAT;               /* Read ADC clears interrupt     */
90
91   AD_last = (LPC_ADC->ADGDR >> 4) & ADC_VALUE_MAX; /* Store converted value   */
92
93   AD_done = 1;
94 }
95 #endif
```

IRQ.c

```
1  /*----------------------------------------------------------------------------
2   * Name:    IRQ.c
3   * Purpose: IRQ Handler
4   * Note(s):
5   *----------------------------------------------------------------------------
6   * This file is part of the uVision/ARM development tools.
7   * This software may only be used under the terms of a valid, current,
8   * end user licence from KEIL for a compatible version of KEIL software
9   * development tools. Nothing else gives you the right to use this software.
10  *
11  * This software is supplied "AS IS" without warranties of any kind.
12  *
13  * Copyright (c) 2011 Keil - An ARM Company. All rights reserved.
14  *----------------------------------------------------------------------------*/
15
16 #include "LPC17xx.H"                         /* LPC17xx definitions           */
17 #include "LED.h"
18 #include "ADC.h"
19
20 uint8_t  clock_ms;                           /* Flag activated every 10 ms    */
21
22
23 /*----------------------------------------------------------------------------
24   Systick Interrupt Handler
25   SysTick interrupt happens every 10 ms
26  *----------------------------------------------------------------------------*/
27 void SysTick_Handler (void) {
28   static unsigned long ticks = 0;
29   static unsigned long timetick;
30   static unsigned int  leds = 0x01;
31
32   if (ticks++ >= 9) {                       /* Set Clock1s to 10ms */
33     ticks    = 0;
34     clock_ms = 1;
35   }
36
37   /* Blink the LEDs depending on ADC_ConvertedValue                           */
38   if (timetick++ >= (AD_last >> 8)) {
39     timetick   = 0;
40     leds     <<= 1;
41     if (leds > (1 << LED_NUM)) leds = 0x01;
42     //LED_Out (leds);
43   }
44
45   ADC_StartCnv();
46 }
```

Joystick.c

```c
/*----------------------------------------------------------------------------
 * Name:    Joystick.c
 * Purpose: Use this class to absract the KBD reading
 * Note(s):
 *----------------------------------------------------------------------------
 */


#include <stdint.h>
#include <string.h>
#include "Joystick.h"
#include "KDB.h"

char JOYSTICK_val[6];
uint32_t KBD_neutral = 0;

/**JOYSTICK_getNeutral
 * Used to get the neutral value of KDB
 */
void JOYSTICK_getNeutral(void){
    KBD_neutral = KDB_button();
}

/**
 * Used to Init KDB, and get initial values
 */
void JOYSTICK_Init(void){
    JOYSTICK_getNeutral();
}


/**JOYSTICK_Update
 *
 * Call this class to update the joystick and possibly get a new
 * position from kdb
 *
 * @return JOYSTICK_val -> type char -> The current value of joystick
 */
char* JOYSTICK_Update(void){
    uint32_t temp = KDB_button() ;
    if(temp != KBD_neutral) {
        if ((temp & KDB_UP) == KDB_UP) {
            strcpy(JOYSTICK_val, JOYSTICK_UP);
        } else if ((temp & KDB_DOWN) == KDB_DOWN) {
            strcpy(JOYSTICK_val, JOYSTICK_DOWN);
        } else if ((temp & KDB_LEFT) == KDB_LEFT) {
            strcpy(JOYSTICK_val, JOYSTICK_LEFT);
        } else if ((temp & KDB_RIGHT) == KDB_RIGHT) {
            strcpy(JOYSTICK_val, JOYSTICK_RIGHT);
        } else if ((temp & KDB_SELECT) == KDB_SELECT) {
            strcpy(JOYSTICK_val, JOYSTICK_SELECT);
        }
    }

    return JOYSTICK_val;
}
```

LED.c

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

 * Name:    LED.c
 * Purpose: low level LED functions
 * Note(s):
 *----------------------------------------------------------------------------
 * This file is part of the uVision/ARM development tools.
 * This software may only be used under the terms of a valid, current,
 * end user licence from KEIL for a compatible version of KEIL software
 * development tools. Nothing else gives you the right to use this software.
 *
 * This software is supplied "AS IS" without warranties of any kind.
 *
 * Copyright (c) 2009-2011 Keil - An ARM Company. All rights reserved.
```

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

#include "LPC17xx.H"                        /* LPC17xx definitions            */
#include "LED.h"

const unsigned long led_mask[] = { 1UL<<28, 1UL<<29, 1UL<<31, 1UL<< 2,
                                   1UL<< 3, 1UL<< 4, 1UL<< 5, 1UL<< 6 };
/*----------------------------------------------------------------------------
  initialize LED Pins
 *----------------------------------------------------------------------------*/

void LED_Init (void) {

  LPC_SC->PCONP     |= (1 << 15);             /* enable power to GPIO & IOCON  */

  LPC_GPIO1->FIODIR |= 0xB0000000;            /* LEDs on PORT1 are output      */
  LPC_GPIO2->FIODIR |= 0x0000007C;            /* LEDs on PORT2 are output      */
}

/*----------------------------------------------------------------------------
  Function that turns on requested LED
 *----------------------------------------------------------------------------*/
void LED_On (unsigned int num) {

  if (num < 3) LPC_GPIO1->FIOPIN |=  led_mask[num];
  else         LPC_GPIO2->FIOPIN |=  led_mask[num];
}

/*----------------------------------------------------------------------------
  Function that turns off requested LED
 *----------------------------------------------------------------------------*/
void LED_Off (unsigned int num) {

  if (num < 3) LPC_GPIO1->FIOPIN &= ~led_mask[num];
  else         LPC_GPIO2->FIOPIN &= ~led_mask[num];
}

/*----------------------------------------------------------------------------
  Function that outputs value to LEDs
 *----------------------------------------------------------------------------*/
void LED_Out(unsigned int value) {
  int i;

  for (i = 0; i < LED_NUM; i++) {
    if (value & (1<<i)) {
      LED_On (i);
    } else {
      LED_Off(i);
    }
  }
}
```

KDB.c

```
#include "LPC17xx.h"
#include "KDB.h"
uint32_t KBD_val  = 0;
/*----------------------------------------------------------------------------   initialize Joystick  *----------------------------------------------------------------------------*/
void KDB_Init (void) {
  LPC_SC->PCONP     |= (1 << 15);/* enable power to GPIO & IOCON  */ /* P1.20, P1.23..26 is GPIO (Joystick) */
  LPC_PINCON->PINSEL3 &= ~((3<< 8)|(3<<14)|(3<<16)|(3<<18)|(3<<20));   /* P1.20, P1.23..26 is input */
  LPC_GPIO1->FIODIR   &= ~((1<<20)|(1<<23)|(1<<24)|(1<<25)|(1<<26));
}
uint32_t KDB_get(void){
  uint32_t kdb_val;
  kdb_val = (LPC_GPIO1->FIOPIN >> 20) & KDB_MASK;
  return(kdb_val);
}

uint32_t KDB_button(void){
  uint32_t val = 0;
  val = KDB_get();
  val = (-val & KDB_MASK);
```

```
20    return (val);
21  }
```