Ryerson University

Department of Electrical, Computer, & Biomedical Engineering
Faculty of Engineering & Architectural Science

Course Title:	**Embedded Systems Design**
Course Number:	**COE718**
Semester/Year (e.g.F2016)	**F2025**

Instructor:	**Gul Khan**

Assignment/Lab Number:	*Lab 1*
Assignment/Lab Title:	*Lab 1: Introduction to Keil uVision*

Submission Date :	**September 18, 2025**
Due Date:	**September 18, 2025**

Student LAST Name	Student FIRST Name	Student Number	Section	Signature*
Grewal	**Arman**	**501100160**	**03**	**AG**

Table Of Contents

1.0 LAB OBJECTIVE

The objective of this lab is to implement and analyze a joystick-controlled application on the NXP LPC1768 microcontroller using the Keil uVision IDE. The lab demonstrates how joystick inputs can be used to control LEDs and generate corresponding outputs on the LCD display. This process involves configuring GPIOs, integrating peripheral functions, and utilizing debugging and simulation tools to verify functionality and assess performance

2.0 Simulation Results

Joystick Select:

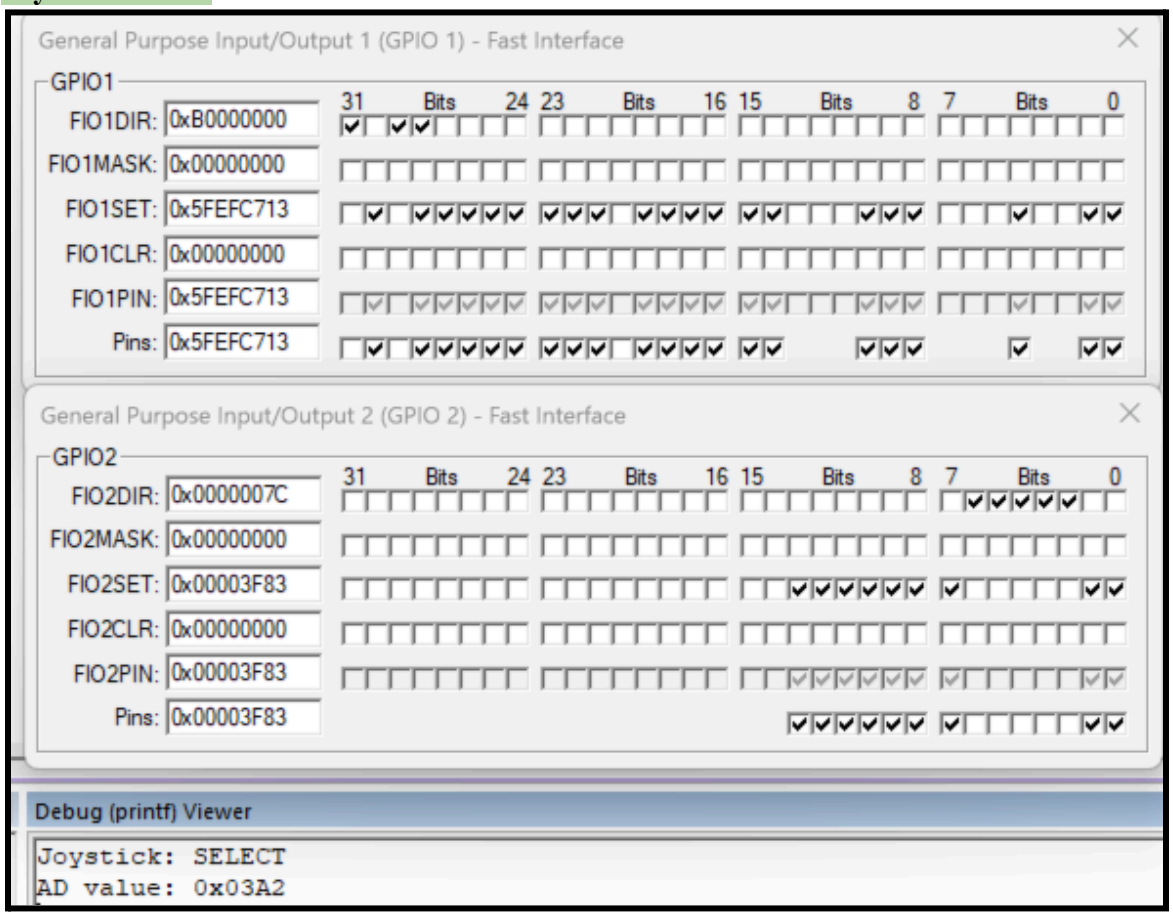

Figure 1.0: Joystick select button enabled

In figure 1.0, the interface is set so the joystick select (clicking the joystick) is set by using **GPIO1.20**. This causes the led at **GPIO1.28** to turn on.

Joystick Left:

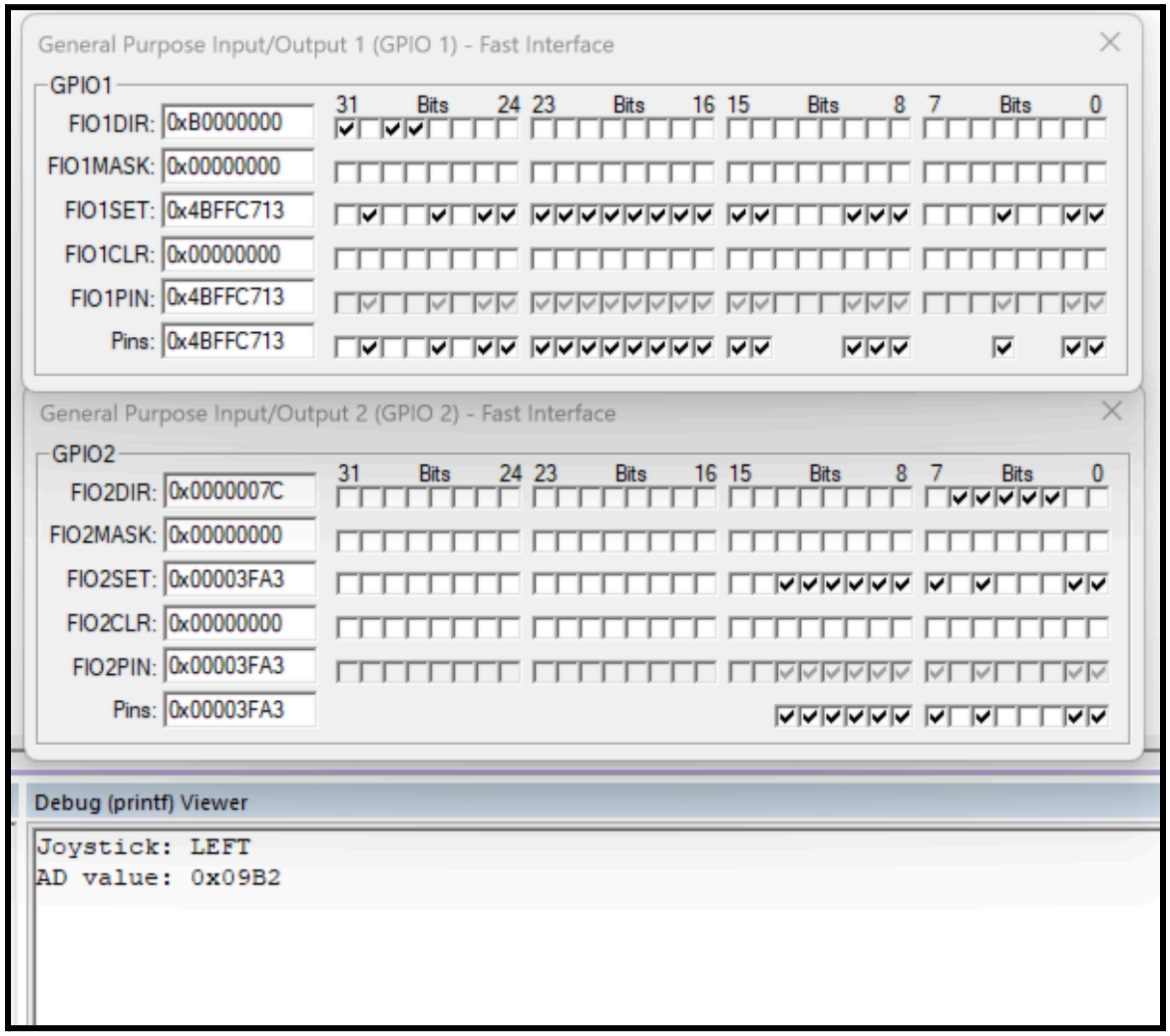

Figure 2.0: Joystick left button enabled

In figure 2.0, the interface is set so the joystick left is set by using **GPIO1.26**. This causes the led at **GPIO2.5** to turn on.

Joystick Right:

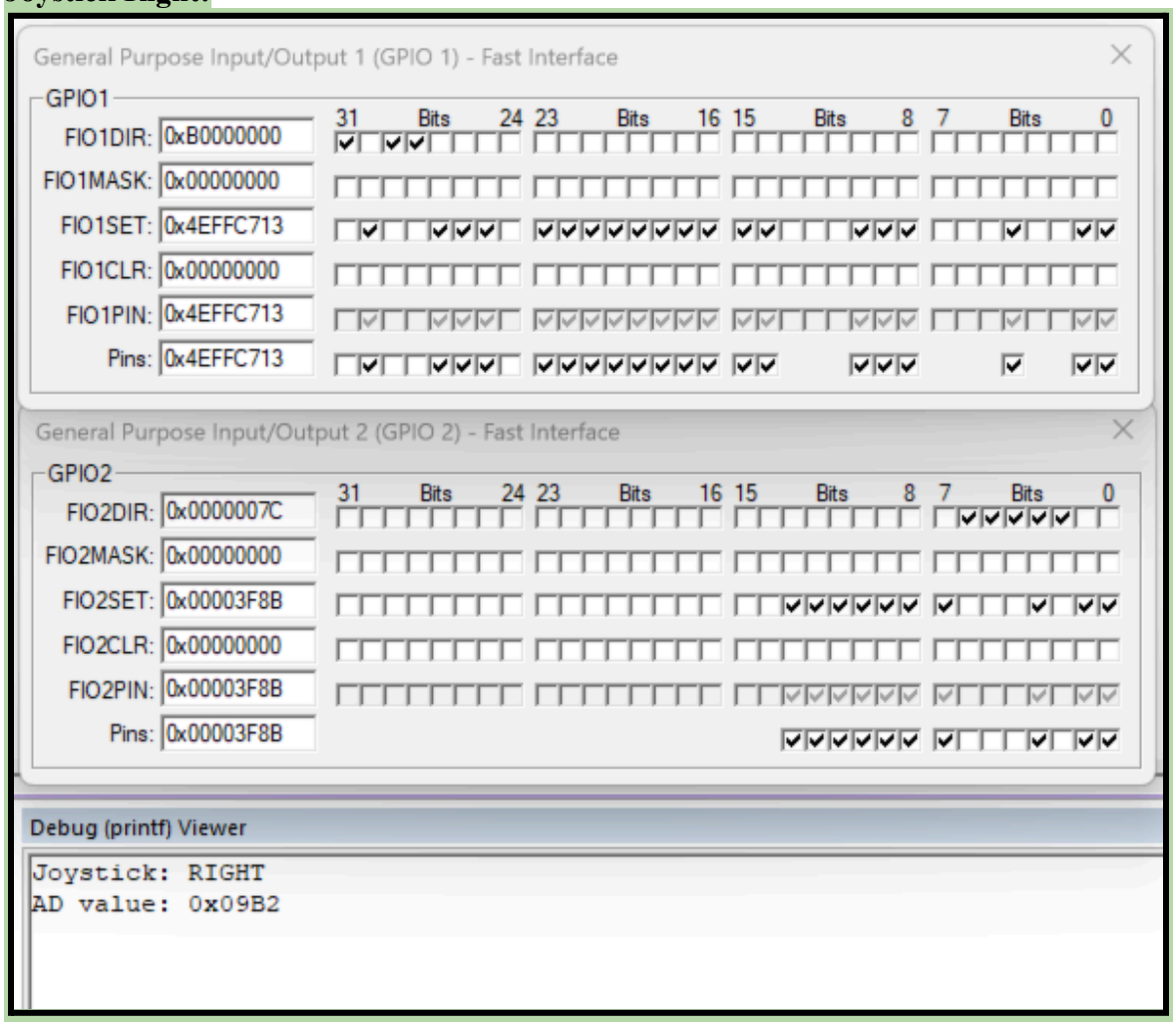

Figure 3.0: Joystick right button enabled

In figure 3.0, the interface is set so the joystick right is set by using **GPIO1.24**. This causes the led at **GPIO2.3** to turn on.

Joystick Up:

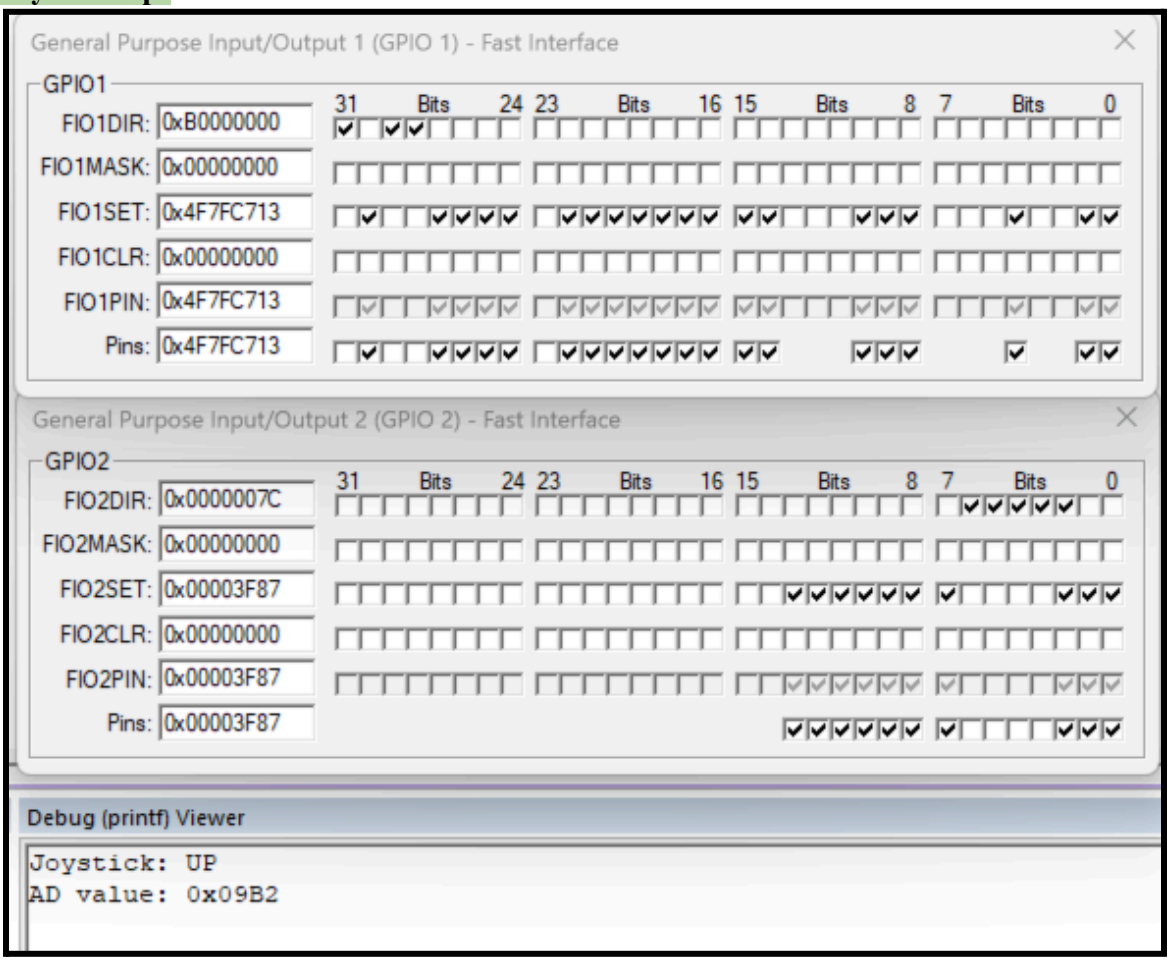

Figure 4.0: Joystick up button enabled

In figure 4.0, the interface is set so the joystick up is set by using **GPIO1.23**. This causes the led at **GPIO2.2** to turn on.

Joystick Down:

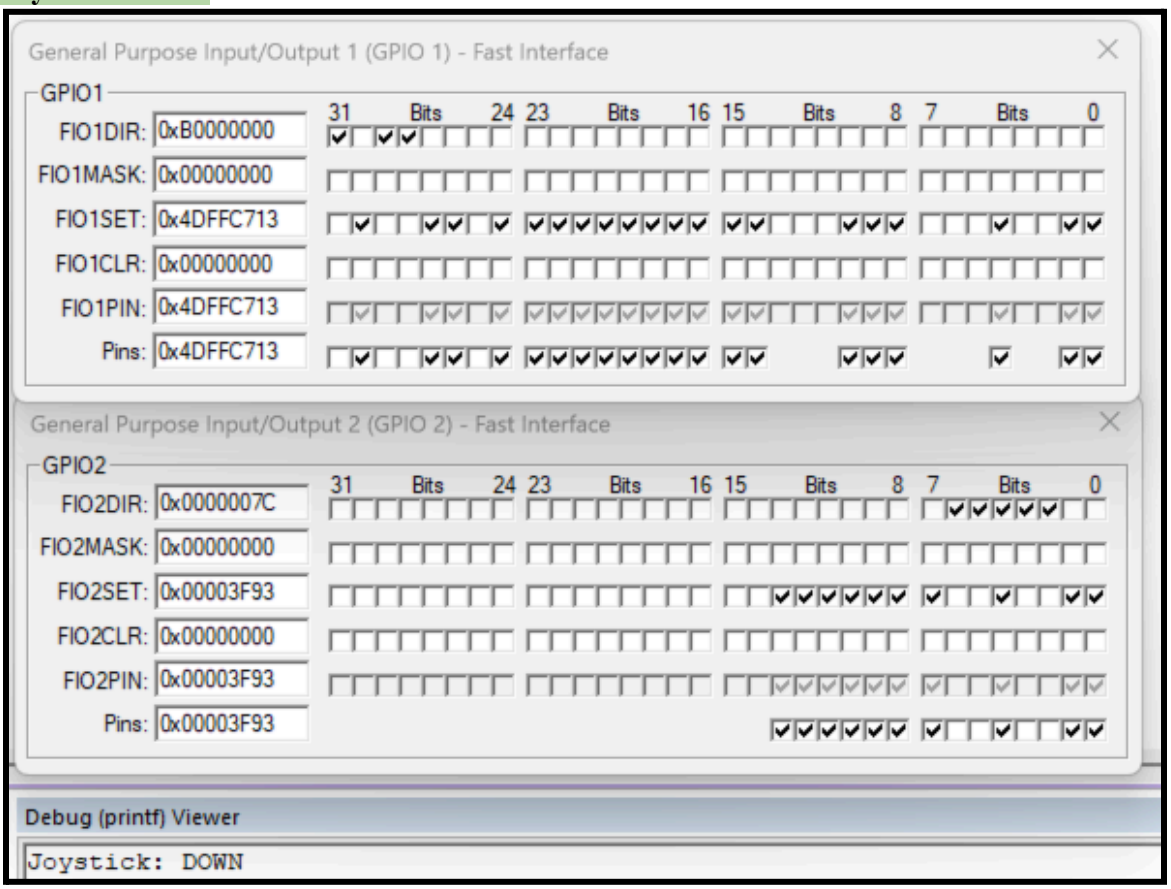

Figure 5.0: Joystick down button enabled

In figure 4.0, the interface is set so the joystick down is set by using **GPIO1.25**. This causes the led at **GPIO2.4** to turn on.

3.0 Conclusion

In conclusion, this lab was completed successfully, with the enabling of our joystick being connected with our leds. Joystick directions were mapped to board LEDs through the LED.c driver (LED_Out and led_mask), and simulation confirmed the expected GPIO activity for SELECT, LEFT, RIGHT, UP, and DOWN. In order to Disable the LCD when in Debug mode, the variable __USE_LCD was used. Overall, the stated objectives were met, and the system behaved as designed.

4.0 Appendix/Code

Blinky.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 "Blinky.h"
#include "LPC17xx.h"
#include "GLCD.h"
#include "LED.h"
#include "Board_ADC.h"
#include "KBD.h"


#define __FI       1                  /* Font index 16x24              */
#define __USE_LCD   0
                  /* 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


//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 text[10];
char text_l[10];

static volatile uint16_t AD_dbg;

uint16_t ADC_last;                  // Last converted value
/* Import external variables from IRQ.c file                          */
extern uint8_t  clock_ms;



/*////////// Joystick Methoods ////////////////////*/

void handleJoystick(uint32_t val) {
   if (val == KBD_MASK) {
      LED_Out(0x00);
      printf("Joystick none\n");
      return;
   }

   LED_Out(val);
}

const char* joystickName(uint32_t val) {
   if (val == KBD_UP)     return "UP";
   if (val == KBD_DOWN)   return "DOWN";
   if (val == KBD_LEFT)   return "LEFT";
   if (val == KBD_RIGHT)  return "RIGHT";
   if (val == KBD_SELECT) return "SELECT";
   return "NONE";
}



/*---------------------------------------------------------------------------
  Main Program
 *---------------------------------------------------------------------------*/
int main (void) {
  int32_t  res;
  uint32_t AD_sum   = 0U;
  uint32_t AD_cnt   = 0U;
  uint32_t AD_value = 0U;
```

```
  uint32_t AD_print = 0U;

  LED_Init();                          /* LED Initialization          */
  ADC_Initialize();                    /* ADC Initialization          */
          KBD_Init();
                                                                       /* Joysticck
Initialization*/

#ifdef __USE_LCD
  GLCD_Init();
  GLCD_Clear(White);
  GLCD_SetBackColor(Blue);
  GLCD_SetTextColor(Yellow);
  GLCD_DisplayString(0, 0, __FI, (unsigned char *)"   Arman's COE718 Joystick
Demo ");
  GLCD_SetTextColor(White);
  GLCD_DisplayString(1, 0, __FI, (unsigned char *)"       LAB 1          ");
  GLCD_SetBackColor(White);
  GLCD_SetTextColor(Blue);
  GLCD_DisplayString(3, 0, __FI, (unsigned char *)"Last Dir:");
  GLCD_SetTextColor(Green);
  GLCD_DisplayString(6, 10, __FI, (unsigned char *)"CENTER    ");
#endif

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

          uint32_t last_buttons = 0U;                                  /*
Last joystick control */

  while (1) {                          /* Loop forever                */

    /* AD converter input                                             */
    // AD converter input
    res = ADC_GetValue();
    if (res != -1) {                  // If conversion has finished
      ADC_last = (uint16_t)res;

      AD_sum += ADC_last;             // Add AD value to sum
      if (++AD_cnt == 16U) {          // average over 16 values
        AD_cnt = 0U;
        AD_value = AD_sum >> 4;       // average devided by 16
        AD_sum = 0U;
      }
    }
```

```
   if (AD_value != AD_print) {
     AD_print = AD_value;            // Get unscaled value for printout
     AD_dbg   = (uint16_t)AD_value;

     sprintf(text, "0x%04X", AD_value); // format text for print out
                  }


                  //Joystick input
                  uint32_t joystick_val = get_button();
                  handleJoystick(joystick_val);




#ifdef __USE_LCD
                        if (joystick_val != KBD_MASK) {
          switch(joystick_val){
             case KBD_UP:
                GLCD_SetTextColor(Green);
                GLCD_DisplayString(6, 10, __FI, (unsigned char *)"UP      ");
                break;
             case KBD_DOWN:
                GLCD_SetTextColor(Red);
                GLCD_DisplayString(6, 10, __FI, (unsigned char *)"DOWN    ");
                break;
             case KBD_LEFT:
                GLCD_SetTextColor(Blue);
                GLCD_DisplayString(6, 10, __FI, (unsigned char *)"LEFT    ");
                break;
             case KBD_RIGHT:
                GLCD_SetTextColor(Yellow);
                GLCD_DisplayString(6, 10, __FI, (unsigned char *)"RIGHT   ");
                break;
             case KBD_SELECT:
                GLCD_SetTextColor(DarkGreen);
                GLCD_DisplayString(6, 10, __FI, (unsigned char *)"CENTER  ");
                break;
                                                        default:
                                                                  break;
          }
       }

#endif

                        /*
```

```
    GLCD_SetTextColor(Red);
    GLCD_DisplayString(6,  9, __FI,  (unsigned char *)text);
                        GLCD_SetTextColor(Green);
    GLCD_Bargraph (144, 7*24, 176, 20, (AD_value >> 2));
                        */


   /* Print message with AD value every 10 ms                    */
   if (clock_ms) {
    clock_ms = 0;

                        printf("Joystick: %s\n", joystickName(joystick_val));

    printf("AD value: %s\r\n", text);
   }
  }
}
```

IRQ.c:

```
/*----------------------------------------------------------------------------
 * Name:    IRQ.c
 * Purpose: IRQ Handler
 * 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) 2011 Keil - An ARM Company. All rights reserved.
 *----------------------------------------------------------------------------*/

#include "LPC17xx.h"                    /* LPC17xx definitions          */
#include "LED.h"
#include "Board_ADC.h"
#include "Blinky.h"

uint8_t  clock_ms;                      /* Flag activated every 10 ms    */
```

```
/*---------------------------------------------------------------------------
  Systick Interrupt Handler
  SysTick interrupt happens every 10 ms
 *---------------------------------------------------------------------------*/
void SysTick_Handler (void) {
  static unsigned long ticks = 0;
  static unsigned long timetick;
  static unsigned int  leds = 0x01;

  if (ticks++ >= 9) {                   /* Set Clock1s to 10ms */
    ticks    = 0;
    clock_ms = 1;
  }

  /* Blink the LEDs depending on ADC_ConvertedValue                        */
  if (timetick++ >= (ADC_last >> 8)) {
    timetick   = 0;
    leds     <<= 1;
                  /*
   if (leds > (1 << LED_NUM)) leds = 0x01;
    LED_Out (leds);
                  */
  }

  ADC_StartConversion();
}
```