Objective

This lab project introduces Keil uVision IDE, and the development boards students will be using throughout the semester. A general survey of features is accomplished by working with an input (joystick), and several outputs (LEDs and LCD screen) of the development board.

Implementation

This implementation elects to use the builtin joystick library – and GPIO libraries, as a dependency – provided by the Keil IDE rather than those provided in the lab manual, in an effort to increase *integration* with the IDE.

After inspecting the project following the lab manual instructions, the joystick header was added to the `Blinky.c` file and initialized alongside the other peripherals in `main()`. Both the builtin joystick library and the provided *KBD* library provide an interface to the hardware, albeit in different ways. After initialization, the joystick is *polled*, returning a `uint32_t` value. While the KBD library defines values for each state that can be compared as a boolean, the builtin library defines masks to read the direction as a bit, also as a `uint32_t`.

By relying on the default C99 boolean behaviour, 0 being *false* and any other integer being *true*, the joystick direction can be determined by masking the current joystick state with each mask, which are defined alongside the joystick functions in the builtin library. For example, if the joystick is pushed to the right, the respective bit in the joystick *word* will be set high. When a logical *and* is performed with the value and mask, every bit other than the desired `JOYSTICK_RIGHT` bit will be 0. Since the joystick is pushed right, this bit will be 1, resulting in a non-zero value C will then treat as true.

This is demonstrated in Figure 1, where the state is fetched and used to set a variable with the current state, and modify the LEDs on the board in a similar fashion. The variable is later used to update the LCD screen.

```
const char* joystick = "NONE ";
while (1) {
  const uint32_t state = Joystick_GetState();

  if      (state & JOYSTICK_LEFT)   { joystick = "LEFT "; LED_Out(1); }
  else if (state & JOYSTICK_RIGHT)  { joystick = "RIGHT"; LED_Out(2); }
  else if (state & JOYSTICK_DOWN)   { joystick = "DOWN "; LED_Out(3); }
  else if (state & JOYSTICK_UP)     { joystick = "UP   "; LED_Out(4); }
  else if (state & JOYSTICK_CENTER) { joystick = "PRESS"; LED_Out(5); }
```

Figure 1: Joystick action control path

From this point, extraneous code related to the ADC function was removed, and any references, such as in `IRQ.c`, were commented out.

Appendix

```
/*----------------------------------------------------------------------------
 * 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 "Blinky.h"

#include <stdio.h>

#include "Board_Joystick.h"
#include "GLCD.h"
#include "LED.h"
#include "LPC17xx.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

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);
}
```

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

int main(void) {
  LED_Init(); /* LED Initialization            */
  Joystick_Initialize();

#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, "     COE718 Lab 1    ");
  GLCD_SetTextColor(White);
  GLCD_DisplayString(1, 0, __FI, "       Blinky.c     ");
  GLCD_DisplayString(2, 0, __FI, "  Try the joystick! ");
  GLCD_SetBackColor(White);
  GLCD_SetTextColor(Blue);
  GLCD_DisplayString(6, 0, __FI, "Direction:");
#endif

  // SystemCoreClockUpdate();
  SysTick_Config(SystemCoreClock / 100);

  const char *joystick = "NONE ";
  while (1) {
    const uint32_t state = Joystick_GetState();

    if (state & JOYSTICK_LEFT) {
      joystick = "LEFT ";
      LED_Out(1);
    } else if (state & JOYSTICK_RIGHT) {
      joystick = "RIGHT";
      LED_Out(2);
    } else if (state & JOYSTICK_DOWN) {
      joystick = "DOWN ";
      LED_Out(3);
    } else if (state & JOYSTICK_UP) {
      joystick = "UP   ";
      LED_Out(4);
    } else if (state & JOYSTICK_CENTER) {
      joystick = "PRESS";
      LED_Out(5);
    }
```

```
#ifdef __USE_LCD
    GLCD_SetTextColor(Green);
    GLCD_DisplayString(6, 11, __FI, (unsigned char *)joystick);
#endif

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

      printf("Joystick: %s\r\n", joystick);
    }
  }
}
```