Toronto Metropolitan University

Faculty of Engineering & Architectural Science

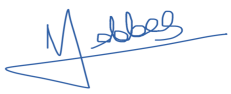
Department of Electrical, Computer and Biomedical Engineering

Course Title:	Embedded Systems Design
Course Number:	COE718 - 072
Semester/Year (e.g. F2016)	F2025

Instructor:	Prof. Asad, Arghavan
Teaching Assistant (TA)	karan hayer- k1hayer@torontomu.ca

Assignment/Lab Number:	Lab 1
Assignment/Lab Title:	Introduction to Keil uVision & ARM Cortex M3

Submission Date:	September 18th, 2025
Due Date:	September 18th, 2025 (6:00 PM)

Student LAST Name	Student FIRST Name	Student Number	Section	Signature*
Maisam Abbas	Syed	501103255	07	

*By signing above you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, an "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at: http://www.ryerson.ca/senate/current/pol60.pdf

Table of Contents

Objectives……..3

Code……..3

Blinky.c……..3

KBD.c. ……..7

KBD.h. ……..9

Conclusions……..10

Objectives

This lab introduces the Keil uVision IDE and some of its features. More specifically the architecture of the ARM Cortex M3. The coding basics of configuring the LEDs and LCDs of the ARM Cortex M3 and its NXP LPC1768 microcontroller will be simulated, debugged, and analyzed. Including how to run a simple program on the MCB1700 dev board. This lab will allow engineers to become familiar with the uVision environment, its simulation capabilities, and the tools needed to assess various CPU performance factors. As the majority of embedded systems use ARM processors for low-power consumption and competitive performance, designers will find the skill sets obtained from this lab especially useful.

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

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 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;



/*----------------------------------------------------------------------------
  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;
int32_t key;

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

#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 Demooo   ");
GLCD_SetTextColor(White);
  GLCD_DisplayString(1, 0, __FI, "       Blinky.c     ");
  GLCD_DisplayString(2, 0, __FI, "  Turn pot for LEDs ");
  GLCD_SetBackColor(White);
  GLCD_SetTextColor(Blue);
  GLCD_DisplayString(5, 0, __FI, "AD value:            ");
GLCD_DisplayString(8, 0, __FI, "Joystick:            ");

#endif

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

  while (1) {                           /* Loop forever                 */

    /* AD converter input                                               */
    // AD converter input
    res = ADC_GetValue();
key = get_button();
    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


#ifdef __USE_LCD
      GLCD_SetTextColor(Red);
      GLCD_DisplayString(6,  9, __FI,  (unsigned char *)text);
GLCD_SetTextColor(Green);
      GLCD_Bargraph (144, 7*24, 176, 20, (AD_value >> 2)); /* max bargraph is 10 bit */
#endif
```

```
    }

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

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

switch (key) {
   case KBD_SELECT:
      printf("Joystick center\n");
      LED_Out(0);             // Turn off all LEDs
      LED_On(4);              // Turn on LED 4 for SELECT
#ifdef __USE_LCD
      GLCD_DisplayString(8, 0, __FI, "Joystick: S          ");
#endif
      break;

   case KBD_DOWN:
      printf("Joystick down\n");
      LED_Out(0);
      LED_On(1);              // Turn on LED 1 for DOWN
#ifdef __USE_LCD
      GLCD_DisplayString(8, 0, __FI, "Joystick: D          ");
#endif
      break;

   case KBD_LEFT:
      printf("Joystick left\n");
      LED_Out(0);
      LED_On(2);              // Turn on LED 2 for LEFT
#ifdef __USE_LCD
      GLCD_DisplayString(8, 0, __FI, "Joystick: L          ");
#endif
      break;

   case KBD_RIGHT:
      printf("Joystick right\n");
      LED_Out(0);
      LED_On(3);              // Turn on LED 3 for RIGHT
#ifdef __USE_LCD
      GLCD_DisplayString(8, 0, __FI, "Joystick: R          ");
#endif
```

```
      break;

   case KBD_UP:
      printf("Joystick up\n");
      LED_Out(0);
      LED_On(0);              // Turn on LED 0 for UP
#ifdef __USE_LCD
      GLCD_DisplayString(8, 0, __FI, "Joystick: U          ");
#endif
      break;

   default:
      printf("Unknown key: %d\n", key);
      LED_Out(0);             // Turn off all LEDs if no valid input
      break;
}

}

 }
```

- **KBD.c**

```
/* P1.20, P1.23..26 is input */
//LPC_GPIO1->FIODIR   &= ~((1<<20)|(1<<23)|(1<<24)|(1<<25)|(1<<26));
/*----------------------------------------------------------------------------
16

 * Name:    KBD.c
 * Purpose: MCB1700 low level Joystick
 * Version: V2.0
 *----------------------------------------------------------------------------
 * 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 Keil - An ARM Company. All rights reserved.
 *----------------------------------------------------------------------------
 * History:
 *          V2.0 - updated by Anita Tino for LPC1768
```

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

#include <LPC17xx.H>                    /* LPC17xx definitions               */
#include "KBD.h"

uint32_t KBD_val  = 0;

/*---------------------------------------------------------------------------
  initialize Joystick
 *---------------------------------------------------------------------------*/
void KBD_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));
}

/*---------------------------------------------------------------------------
  Get Joystick value.. part of get_button
 *---------------------------------------------------------------------------*/
uint32_t KBD_get  (void) {
  uint32_t kbd_val;

  kbd_val = (LPC_GPIO1->FIOPIN >> 20) & KBD_MASK;
  return (kbd_val);
}

/*---------------------------------------------------------------------------
  Get Joystick value
 *---------------------------------------------------------------------------*/
uint32_t get_button (void) {
   uint32_t val = 0;

   val = KBD_get();                     /* read Joystick state    */
   val = (~val & KBD_MASK);             /* key pressed is read as a non '0' value*/
```

```
  return (val);
}
```

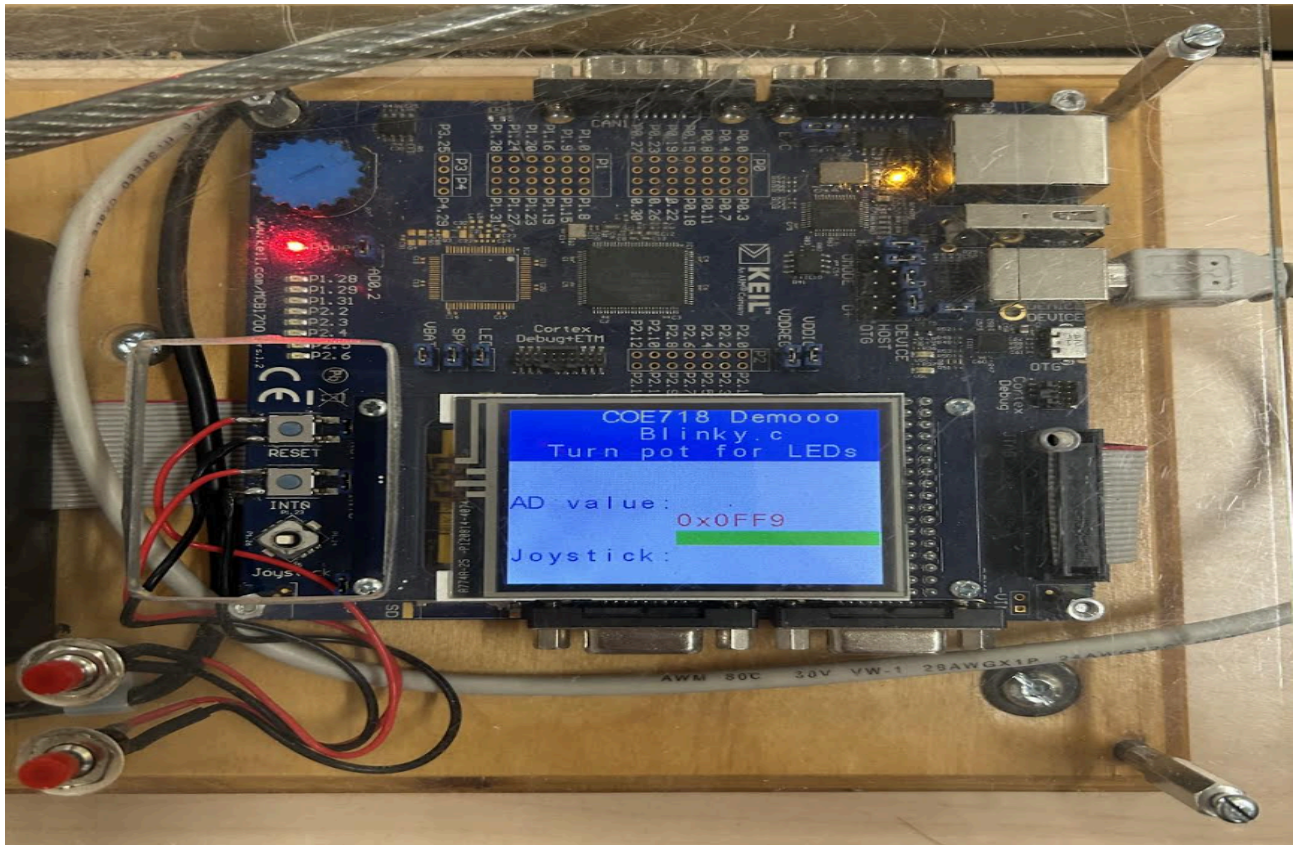
- **KBD.h**

```
/*----------------------------------------------------------------------------
* Name:    KBD.h
* Purpose: MCB1700 low level Joystick definitions
* Version: V2.00
* Note(s): Positioning of Joystick on MCB1700
*
Revised by: Anita Tino
*----------------------------------------------------------------------------
* 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.
*
* Copyright (c) 2008 Keil - An ARM Company. All rights reserved.
*----------------------------------------------------------------------------*/
#include <stdint.h>

#ifndef __KBD_H
#define __KBD_H

#define KBD_SELECT      0x01
#define KBD_UP          0x08
#define KBD_RIGHT       0x10
#define KBD_DOWN        0x20
#define KBD_LEFT        0x40
#define KBD_MASK        0x79

extern uint32_t KBD_val;

extern void     KBD_Init(void);
extern uint32_t KBD_get (void);
extern uint32_t get_button (void);

#endif
```

Conclusions

The following results were observed following the simulation of the Keil uVision software and ARM Cortex M3 device as displayed on the liquid crystal display (LCD).

Figure 1: Demonstration of the output to the LCD and LEDs on the dev board



NOTE: The LEDs ports light up based on the last direction of the joystick represented by the following: U, D, S, L, R.
The LCD also provides the AD value, variable based on the potentiometer, as per the colored segment.

Figure 2: uVision Debug Mode simulation run time using the MCB1700 Dev Board (joystick operations)

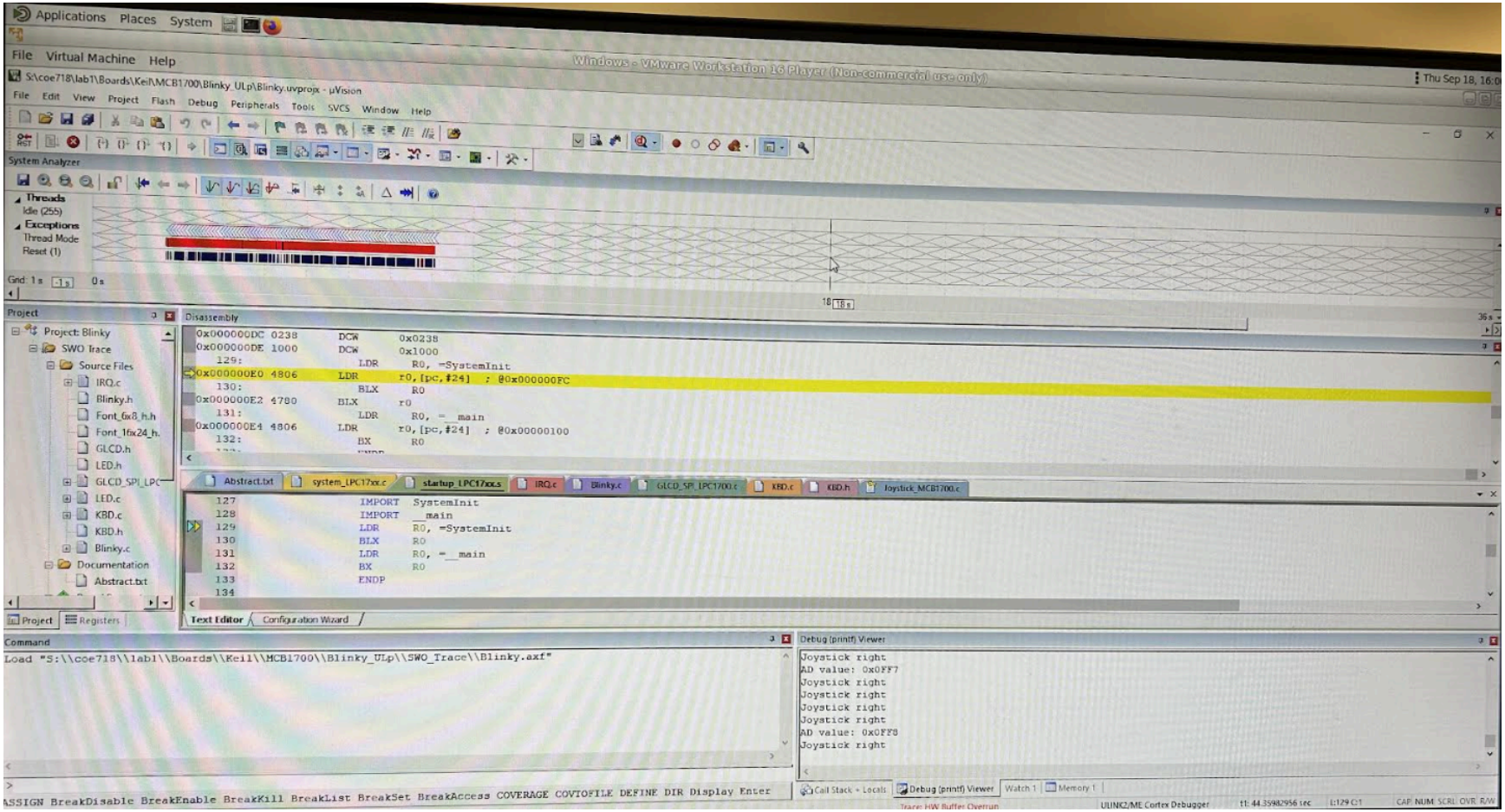


Table 1: Result Observation Table displayed on the LCD (Liquid Crystal Display) through the simulation of the Keil uVision IDE

Operation	Right - R	Left - L	Select - S	Up - U	Down - D
Results