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)** | F2021 |

| | |
|---|---|
| **Instructor:** | Sunbal Cheema |

| | |
|---|---|
| *Assignment/Lab Number:* | 1 |
| *Assignment/Lab Title:* | Introduction to uVision and ARM Cortex M3 |

| | |
|---|---|
| *Submission Date :* | Tuesday, September 21, 2021 |
| *Due Date:* | Tuesday, September 21, 2021 |

| **Student LAST Name** | **Student FIRST Name** | **Student Number** | **Section** | **Signature*** |
|---|---|---|---|---|
| Fahmy | Ahmad | 500913092 | 4 | [illegible] |

*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

## 1. Introduction:

This lab was done to provide the student with an introduction to uVision with Blinky.c and IRQ.c as well as the tools available for analysis, Input-Output, Debugging etc. Additionally, we were tasked with using the joystick peripheral to accomplish our objective.

## 2. Objective:

The objective of this lab was to add the joystick peripheral to our project and use the Joystick peripheral's files and methods to print the direction of the joystick.

## 3. Methods:

**Main()**

- The first function/method that is executed and is tasked with holding the core operations of what is desired
- Code: See appendix for code

**Joystick_Initialize()**

- Responsible for initializing the joystick peripheral by enabling the GPIO clock and configuring the pins that will be used
- Code:

```
int32_t Joystick_Initialize (void) {
  uint32_t n;

  /* Enable GPIO clock */
  GPIO_PortClock     (1U);

  /* Configure pins */
  for (n = 0U; n < JOYSTICK_COUNT; n++) {
    PIN_Configure (JOYSTICK_PIN[n].Portnum, JOYSTICK_PIN[n].Pinnum, PIN_FUNC_0, 0U, 0U);
    GPIO_SetDir   (JOYSTICK_PIN[n].Portnum, JOYSTICK_PIN[n].Pinnum, GPIO_DIR_INPUT);
  }
  return 0;
}
```

**Joystick_Stats()**

- Responsible for reading and returning the current position of the joystick in integer form
- Code:

```
uint32_t Joystick_GetState (void) {
  uint32_t val;

  val  = 0U;
  if (!(GPIO_PinRead (JOYSTICK_PIN[0].Portnum, JOYSTICK_PIN[0].Pinnum))) val |= JOYSTICK_CENTER;
  if (!(GPIO_PinRead (JOYSTICK_PIN[1].Portnum, JOYSTICK_PIN[1].Pinnum))) val |= JOYSTICK_UP;
  if (!(GPIO_PinRead (JOYSTICK_PIN[2].Portnum, JOYSTICK_PIN[2].Pinnum))) val |= JOYSTICK_DOWN;
  if (!(GPIO_PinRead (JOYSTICK_PIN[3].Portnum, JOYSTICK_PIN[3].Pinnum))) val |= JOYSTICK_LEFT;
  if (!(GPIO_PinRead (JOYSTICK_PIN[4].Portnum, JOYSTICK_PIN[4].Pinnum))) val |= JOYSTICK_RIGHT;

  return val;
}
```

**Printf()**

- Responsible for printing the desired information in the Serial Monitor
- Code:

```
extern _ARMABI int printf(const char * __restrict /*format*/, ...) __attribute__((__nonnull__(1)));
```

## 4. Figures:

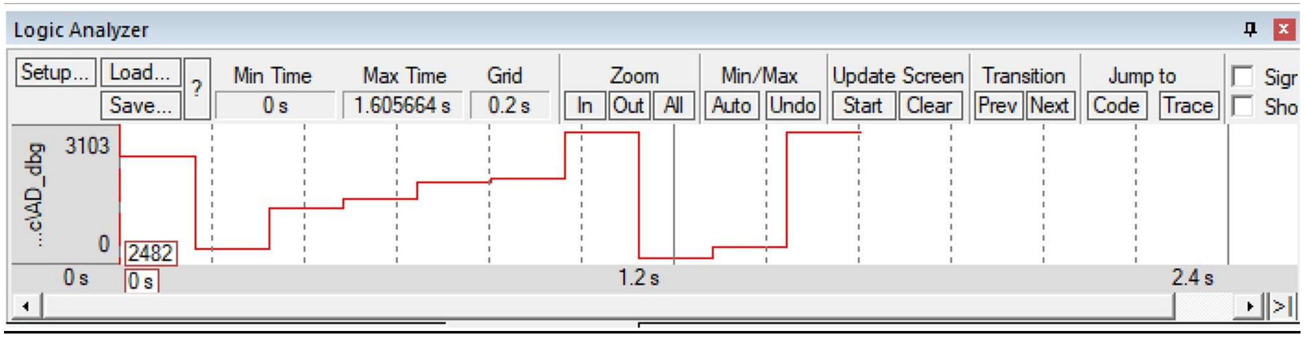
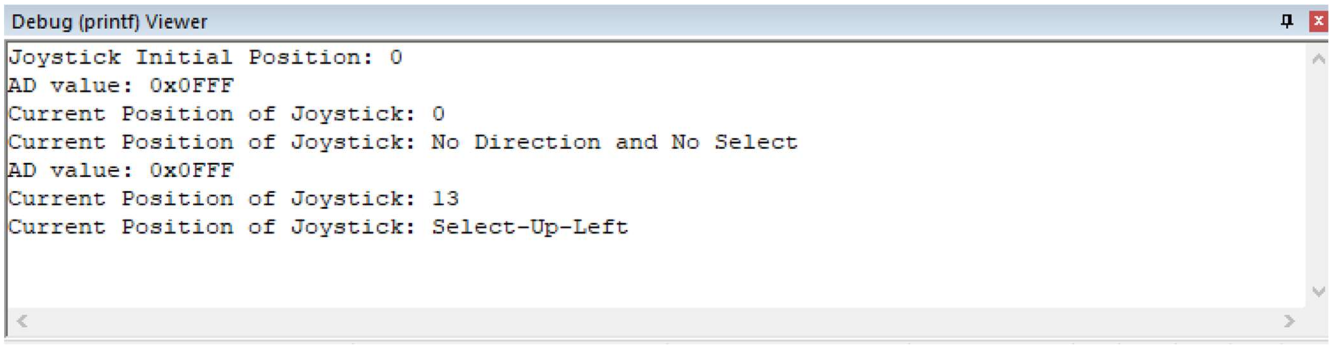
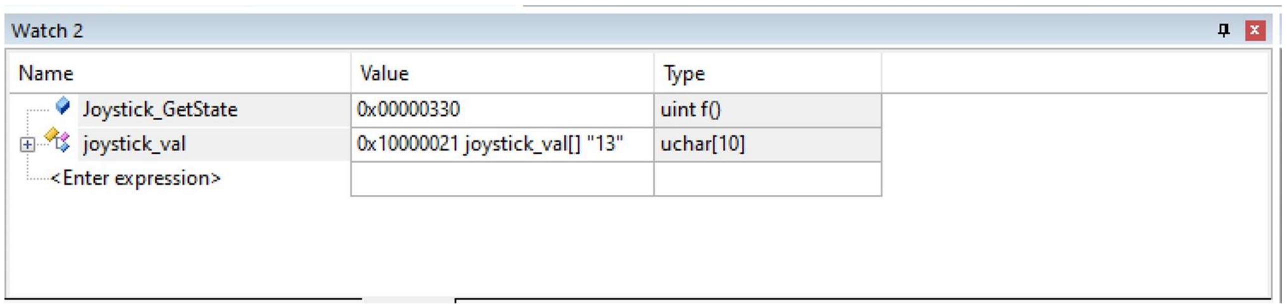


Figure 1: pre-lab student recreation of Fig.22 in Lab Manual – Simulating the Port and A/D Conversion using Logic Analyzer



```
Joystick Initial Position: 0
AD value: 0x0FFF
Current Position of Joystick: 0
Current Position of Joystick: No Direction and No Select
AD value: 0x0FFF
Current Position of Joystick: 13
Current Position of Joystick: Select-Up-Left
```

Figure 2: Example Output



| Name | Value | Type |
|---|---|---|
| Joystick_GetState | 0x00000330 | uint f() |
| joystick_val | 0x10000021 joystick_val[] "13" | uchar[10] |
| <Enter expression> | | |

Figure 3: Watch Window

| GPIO | Joystick_Stats() Value | Corresponding Direction |
|---|---|---|
| N/A | 0 | Nothing Selected |
| 20 | 4 | Select |
| 23 | 8 | Up |
| 24 | 2 | Right |
| 25 | 16 | Down |
| 26 | 1 | Left |
| 23, 24 | 10 | Up-Right |
| 23, 26 | 9 | Up-Left |
| 25, 26 | 17 | Down-Left |
| 24, 25 | 18 | Down-Right |
| 20, 23 | 12 | Select-Up |
| 20, 24 | 6 | Select-Right |
| 20, 25 | 20 | Select-Down |
| 20, 26 | 5 | Select-Left |
| 20, 23, 24 | 14 | Select-Up-Right |
| 20, 23, 26 | 13 | Select-Up-Left |
| 20, 25, 26 | 21 | Select-Down-Left |
| 20, 24, 25 | 22 | Select-Down-Right |
| Impossible Configuration (ex. 24, 26 or 20, 23, 25) | 3, 7, 11, 15, 19, 23, 24, 25, 26, 27, 28, 29, 30, 31 (Ex. 3 and 28) | Examples: Right-Left and Select-Up-Down |

Figure 4: Table Displaying Joystick Peripheral Mapping

## 5. <u>**Conclusion:**</u>

In order to use the Joystick Peripheral one must power up the peripheral with "LPC_SC->PCONP |= (1 << 15);" and initialize the peripheral by using the Joystick_Initialization() function which enables the GPIO clock and configures the pins. Additionally, to read the current position and state of the Joystick we used Joystick_Stats() which returned the current position of the joystick in integer form. Further, to then print this value and its corresponding positions; the joystick position variations were mapped to their corresponding integer values (see Figure 4) and put into a switch…case statement to print the corresponding directions using the printf() function, while physically impossible variations (such as Up-Down or Select-Left-Right) were sent to the 'default' portion of the switch statement as mapping them would've been a waste of system memory, data, and lines of code.

## 6. <u>**Appendix (Code):**</u>

```
/*----------------------------------------------------------------------------
 * Copyright (c) 2004-2020 Arm Limited (or its affiliates). All
 * rights reserved.
 *
 * SPDX-License-Identifier: BSD-3-Clause
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *   1.Redistributions of source code must retain the above copyright
 *     notice, this list of conditions and the following disclaimer.
 *   2.Redistributions in binary form must reproduce the above copyright
 *     notice, this list of conditions and the following disclaimer in the
 *     documentation and/or other materials provided with the distribution.
 *   3.Neither the name of Arm nor the names of its contributors may be used
 *     to endorse or promote products derived from this software without
 *     specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED. IN NO EVENT SHALL COPYRIGHT HOLDERS AND CONTRIBUTORS BE
 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGE.
 *---------------------------------------------------------------------------
 * Name:    Blinky.c
 * Purpose: LED Flasher for MCB1700
 *---------------------------------------------------------------------------*/

#include <stdio.h>
#include "Blinky.h"
#include "LPC17xx.h"                    // Device header
#include "Board_LED.h"                  // ::Board Support:LED
#include "Board_ADC.h"                  // ::Board Support:A/D Converter
#include "Board_Joystick.h"             // ::Board Support:Joystick


static char text[10];
static char joystick_val[10];

// variable to trace in LogicAnalyzer (should not read to often)
static volatile uint16_t AD_dbg;

uint16_t ADC_last;                      // Last converted value

/*---------------------------------------------------------------------------
  Main function
 *---------------------------------------------------------------------------*/
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_Initialize();                    // LED Initialization
  ADC_Initialize();                     // ADC Initialization
Joystick_Initialize();                 // Joystick Initialization

      LPC_SC->PCONP                            |= (1 << 15);                                    // Powering Up Joystick

/* 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 ids input */
     LPC_GPIO1->FIODIR &= ~((1<<20) | (1<<23) | (1<<24) | (1<<25) | (1<<26));



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

printf("Joystick Initial Position: %d\r\n", Joystick_GetState()); // Printing initial position of joystick

  while (1) {                          // Loop forever

                             //

    // 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
    }

    // Print message with AD value every second
    if (clock_1s) {
      clock_1s = 0;

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

                  sprintf(joystick_val, "%d", Joystick_GetState());
                          printf("Current Position of Joystick: %s\r\n", joystick_val);

                          switch(Joystick_GetState())
                                  {
                                                          case 0:
                                                                printf("Current Position of Joystick: No Direction and No
Select\n");
                                                          break;

                                                          case 8:
                                                                printf("Current Position of Joystick: Up\n");
                                                          break;

                                                          case 2:
                                                                printf("Current Position of Joystick: Right\n");
                                                          break;
```

```
case 16:
      printf("Current Position of Joystick: Down\n");
 break;

 case 1:
      printf("Current Position of Joystick: Left\n");
 break;

 case 4:
      printf("Current Position of Joystick: Select\n");
 break;

case 10:
      printf("Current Position of Joystick: Up-Right\n");
 break;

 case 9:
      printf("Current Position of Joystick: Up-Left\n");
 break;

case 17:
      printf("Current Position of Joystick: Down-Left\n");
 break;

case 18:
      printf("Current Position of Joystick: Down-Right\n");
 break;

case 12:
      printf("Current Position of Joystick: Select-Up\n");
 break;

 case 6:
      printf("Current Position of Joystick: Select-Right\n");
```

```
                                                    break;

                                                   case 20:
                                                        printf("Current Position of Joystick: Select-Down\n");
                                                    break;

                                                    case 5:
                                                        printf("Current Position of Joystick: Select-Left\n");
                                                    break;

                                                   case 14:
                                                        printf("Current Position of Joystick: Select-Up-Right\n");
                                                    break;

                                                   case 13:
                                                        printf("Current Position of Joystick: Select-Up-Left\n");
                                                    break;

                                                   case 21:
                                                        printf("Current Position of Joystick: Select-Down-Left\n");
                                                    break;

                                                   case 22:
                                                        printf("Current Position of Joystick: Select-Down-Right\n");
                                                    break;

                                                   default:
                                                        printf("Physically Impossible combination, try using non-
opposing positions\n");

                           }

   }
  }
```