Exercise #09 REPORT

Name: 송예린

Student ID: 2171023

1. **Exercise Objective (10%)**

- Extension Board 를 사용해서 Joystick 를 Control 할 수 있다.
- ADC 가 무엇인지 알고 아날로그 신호를 디지털 신호로 바뀌는 과정을 이해할 수 있다.
- Joystick 으로 ADC 값을 얻어서 LDC 이미지를 이동시킬 수 있다.

1.1 이론적 배경

Extension Board 에서 Joystick 은 적게 움직이면 LCD 이미지가 적게 움직이고 많이 움직이면 이미지가 많이 움직인다. 디지털 컴퓨터는 binary number 밖에 이해하지 못하기 때문에 Analog Signal 을 값을 이해할 수 있는 Digital Signal 변환해서 사용된다. X,Y 축으로 움직이면서 저항의 길이를 조절하고 저항의 변화에 따라 전압의 길이를 조절할 수 있다. 변화되는 Voltage 값을 기준으로 Digital Signal 로 바꿔서 조이스틱이 얼마나 움직였는지 판단할 수 있는 것이다.

ADC Converters 는 Analog Signal 에 들어오는 Voltage 값을 Digital Signal 로 변환해준다

Resolution: 변화하는 값들을 얼마나 섬세하는 게 측정하는 지 알 수 있음

Sample rate: 변화하는 값들을 얼마나 자주 측정하는 지 알 수 있음

String 에 값을 저장하고 sprintf 를 하면 String 에 저장된 값을 출력할 수 있다.

2. **Implementation (60%)**

사용할 변수,string 을 선언하고 display 를 초기화해준 후 API 를 사용해서 LCD 의 텍스트 방향 축이 0°가 되도록 설정하고 배경색상은 하얀색, 글자색은 빨강색이 되도록 한다. ADC 를 input 으로 사용하기 위해서 GPIO Pin 6.0 Pin 4.4 를 입력모드로 바꾼다. Pin 6, Pin 4 DIR Register 핀 0,4 에 각각 0 을 넣고 다른 핀번호 1 을 넣는다. Pin 6, Pin 4 DIR Register 핀 0,4 번에 각각 1 값을 주어서 pull up/down 을 활성화시켜주고 Pin 6, Pin 4 Out register 핀 0,4 에 각각 1 을 넣어줘서 pull up 을 enable 해주고(API 를 활용함) 조이스틱을 초기화해준다.x,y Location 을 가져와 string 에 저장을 하기 위해서 ADC14_getResult 를 이용한다. ADC MEM0 을 Loading 하면 X Location, ADC MEM0 을 Loading 하면 y Location,을 불러올 수 있다. 불러온 값을 string 에 각각 저장한다. 번수 x,y 에 저장된 값들을 넣어주고 Graphics_drawStringCentered 로 *을 출력한다. X,Y Location 값은 16000 까지 나타나기 때문에 125 로 나누어서 128 칸으로 움직일 수 있게 해준다. 조이스틱을 위, 아래로 움직여보면 방향이 반대인데 이를 해결하기 위해서 128-resultsBuffer[1]/125 해준다. 다시

x,y location 을 체크해서 string 에 저장을 하고 과거의 x,y 값인 변수 x,y 위치를 흰색으로 덮어주고 x 의 위치: resultsBuffer[0]/125 y 의 위치: 128-resultsBuffer[1]/125 에 '*'을 출력한다.
만약 조이스틱을 움직여서 . X,Y Location 이 변경됐다면 과거의 위치의 글자를 지우고 새로운 위치에 '*'이 출력될 것이다. 실습은 성공하였지만 한가지 문제가 보였는데 글자가 너무 깜빡였다. 그 이유는 위치 값이 변경되지 않아도 지워졌다가 써졌다가 반복하기 때문이다. 문제를 해결하기 위해서 실습이 끝난 코드에 if 문을 삽입하였다. 위치가 변경될 때만 글자를 지우고 새로운 위치에 글자가 써질 것이다.

3. Discussion & Conclusion (10%)

Extension Board 의 Joystick 를 Control 해서 LCD 의 이미지를 움직여보았다. ADC 는 아날로그 신호를 디지털 신호로 변환해서 값을 코딩에 사용할 수 있도록 한다.
소프트웨어적으로 장치를 control 하여 하드웨어적으로 출력됨을 체험하기 위해서 Joy Stick 을 움직여 LCD 이미지를 이동시키는 실습을 하였다. 실습을 진행하다가 게임을 하던 경험이 떠올랐다. 닌텐도 스위치 동물의 숲, 야숨 등의 게임에서 위치뿐만 아니라 화면의 각도도 조절할 수 있었던 것이 생각났다. X,Y 축에 Z 축을 추가하고 다른 조이스틱으로 Z 축의 위치를 움직인다면 각도도 조절할 수 있을 것 같다. 야숨 같은 3D 게임의 경우 x,y 의 조이스틱만 움직여도 화면의 각도가 움직이는 것을 볼 수 있는데 x,y 의 위치가 바뀌면 z 의 위치도 조금씩 바뀌는 것으로 생각된다.

4. Reference(s)

5. Code (20%)

FILE NAME (ex: main.c)

```
#include <ti/devices/msp432p4xx/inc/msp.h>
#include <ti/devices/msp432p4xx/driverlib/driverlib.h>
#include <ti/grlib/grlib.h>
#include "LcdDriver/Crystalfontz128x128_ST7735.h"
#include <stdio.h>
#include "LcdDriver/msp432p4111_classic.h"

/*
 * Main function
 */
void main(void)
{
```

```
    int x,y;
    /* Graphic library context */
    Graphics_Context g_sContext;

    /* ADC results buffer */
    uint16_t resultsBuffer[2];

    /* string buffer for sprintf() */
    char string[10];

    /* Joystick, S1, S2 buttons in boostxl-edumkii flag */
    int buttonPressed,bs1Pressed,bs2Pressed;

    /* Initializes display */
    Crystalfontz128x128_Init();

    /* Set default screen orientation */
    Crystalfontz128x128_SetOrientation(LCD_ORIENTATION_UP);

    /* Initializes graphics context */
    Graphics_initContext(&g_sContext, &g_sCrystalfontz128x128, &g_sCrystalfontz128x128_funcs);
    Graphics_setForegroundColor(&g_sContext, GRAPHICS_COLOR_RED);
    Graphics_setBackgroundColor(&g_sContext, GRAPHICS_COLOR_WHITE);
    GrContextFontSet(&g_sContext, &g_sFontFixed6x8);
    Graphics_clearDisplay(&g_sContext);

    /* Configures Pin 6.0 and 4.4 as ADC input */
    MAP_GPIO_setAsPeripheralModuleFunctionInputPin(GPIO_PORT_P6, GPIO_PIN0,
GPIO_TERTIARY_MODULE_FUNCTION);
    MAP_GPIO_setAsPeripheralModuleFunctionInputPin(GPIO_PORT_P4, GPIO_PIN4,
GPIO_TERTIARY_MODULE_FUNCTION);

    /* Initializing ADC (ADCOSC/64/8) */
    MAP_ADC14_enableModule();
    MAP_ADC14_initModule(ADC_CLOCKSOURCE_ADCOSC, ADC_PREDIVIDER_64,
ADC_DIVIDER_8, 0);

    /* Configuring ADC Memory (ADC_MEM0 - ADC_MEM1 (A15, A9)  with repeat)
         * with internal 2.5v reference */
    MAP_ADC14_configureMultiSequenceMode(ADC_MEM0, ADC_MEM1, true);
    MAP_ADC14_configureConversionMemory(ADC_MEM0,
            ADC_VREFPOS_AVCC_VREFNEG_VSS,
            ADC_INPUT_A15, ADC_NONDIFFERENTIAL_INPUTS);

    MAP_ADC14_configureConversionMemory(ADC_MEM1,
            ADC_VREFPOS_AVCC_VREFNEG_VSS,
            ADC_INPUT_A9, ADC_NONDIFFERENTIAL_INPUTS);
```

```
    /* Setting up the sample timer to automatically step through the sequence
     * convert.
     */
    MAP_ADC14_enableSampleTimer(ADC_AUTOMATIC_ITERATION);

    /* Triggering the start of the sample */
    MAP_ADC14_enableConversion();
    MAP_ADC14_toggleConversionTrigger();


    while(1)
    {
        /* Store ADC14 conversion results */
        resultsBuffer[0] = ADC14_getResult(ADC_MEM0);
        resultsBuffer[1] = ADC14_getResult(ADC_MEM1);
        x=resultsBuffer[0];
        y=resultsBuffer[1];
        Graphics_drawStringCentered(&g_sContext,
                                    (int8_t *)"*",
                                    8,
                                    resultsBuffer[0]/125,
                                    128-resultsBuffer[1]/125,
                                    OPAQUE_TEXT);
        resultsBuffer[0] = ADC14_getResult(ADC_MEM0);
        resultsBuffer[1] = ADC14_getResult(ADC_MEM1);
        if(x!=resultsBuffer[0]||y!=resultsBuffer[1]){
        Graphics_drawStringCentered(&g_sContext,
                                    (int8_t *)" ",
                                    8,
                                    x/125,
                                    128-y/125,
                                    OPAQUE_TEXT);
        Graphics_drawStringCentered(&g_sContext,
                                    (int8_t *)"*",
                                    8,
                                    resultsBuffer[0]/125,
                                    128-resultsBuffer[1]/125,
                                    OPAQUE_TEXT);
}
    }
}
```

실험이 끝난 후 추가된 코드는 빨간색으로 표시했습니다.