ENGINE

Teaching online electronics, microcontrollers and programming in Higher Education

Programing of embedded systems

7. A/D Converter

Lead Partner: Warsaw University of Technology

Authors: Daniel Krol

University of Applied Sciences in Tarnow

Erasmus+

GRANT NUMBER: 2020-1-PL01-KA226-HE-095653

Declaration

This report has been prepared in the context of the ENGINE project. Where other published and unpublished source materials have been used, these have been acknowledged.

Copyright

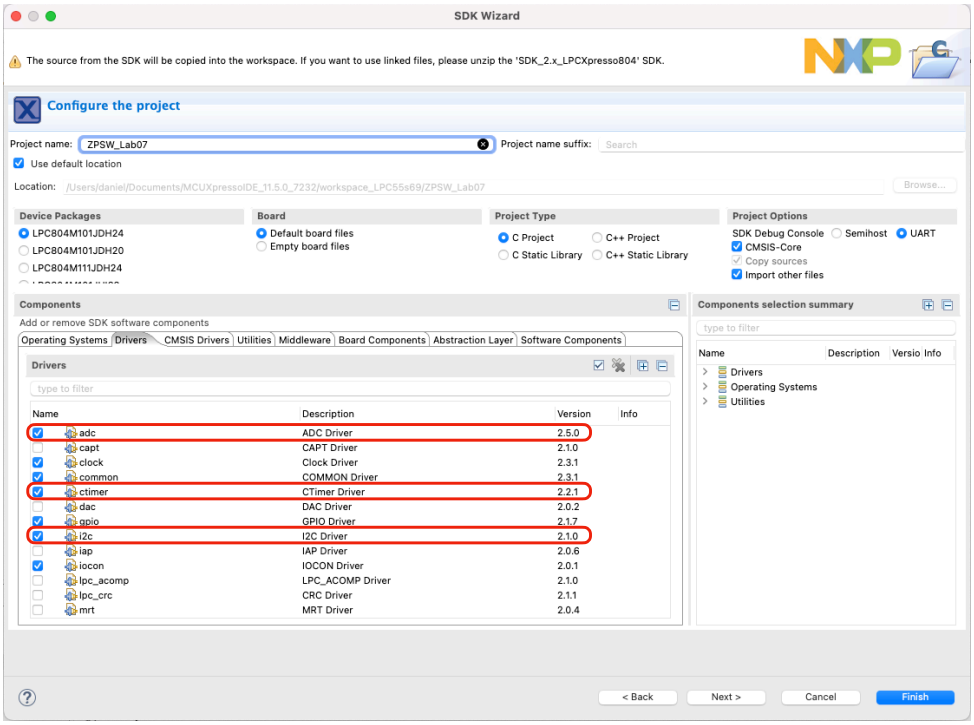
Funding Disclaimer

This project has been funded with support from the European Commission. This report reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Programing of embedded systems
7. A/D Converter

I. OLED Display

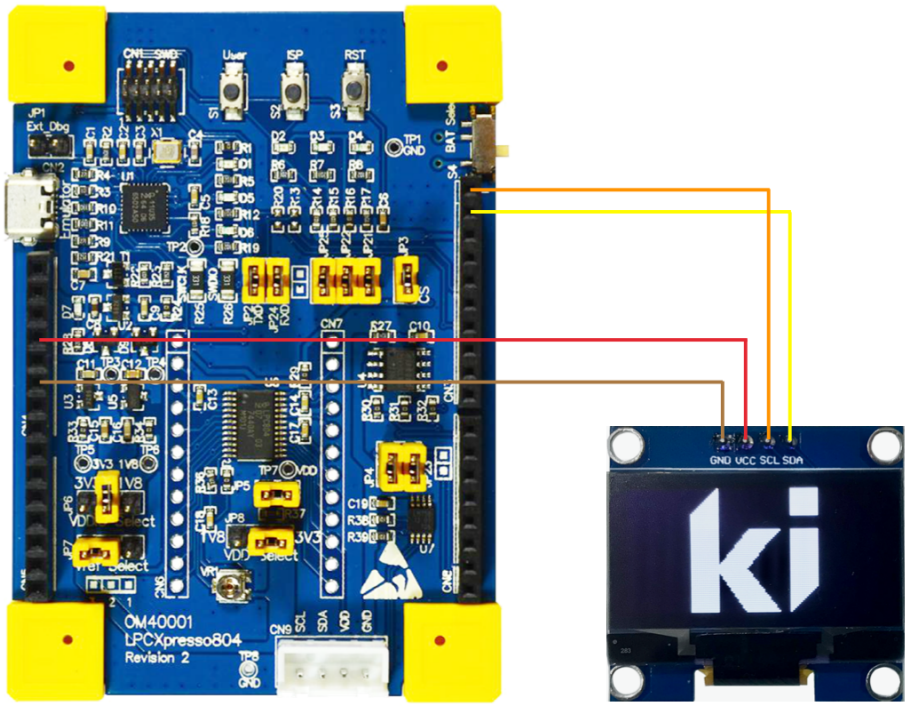
1. Create a new project for the *LPCXpresso804* board and name it eg *Lab07*.
2. Add *ADC*, *CTIMER* and *I2C* drivers:



3. Add the *OLED* library and configure the display operation as in the previous manual.
4. In *Config Tools -> Clocks*, change the frequency of the *FRO_OSC* generator to 30 MHz.
5. Go to the main project file and modify the code as below:

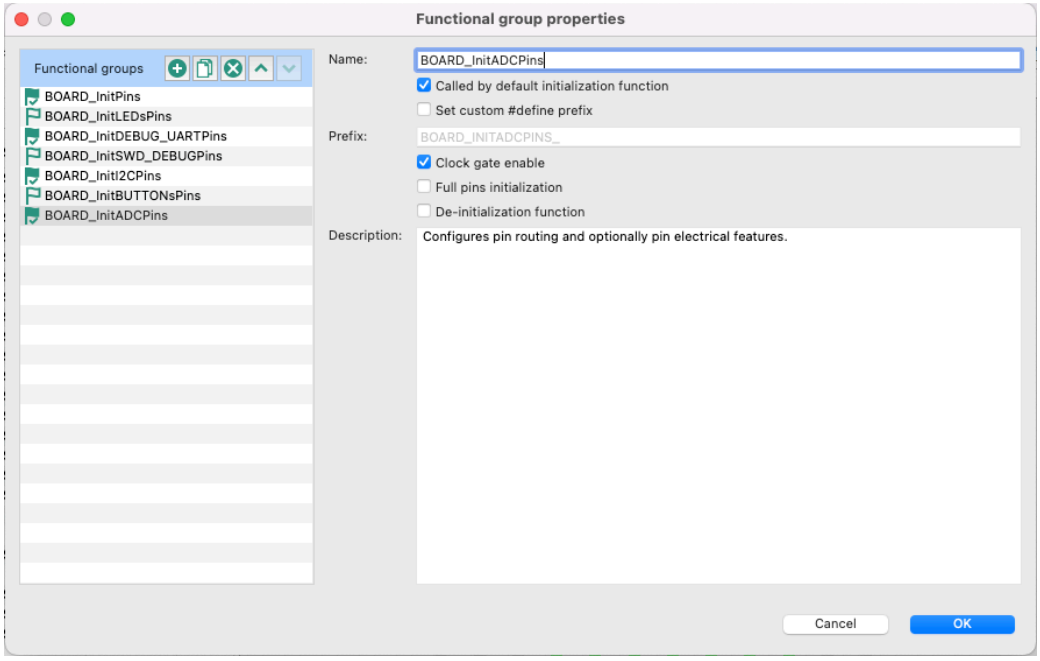
```
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "LPC804.h"
#include "fsl_debug_console.h"
#include "oled.h"

char sbuff[32];
volatile uint16_t adcValue = 0;

/*
 * @brief   Application entry point.
 */
int main(void) {

          /* Init board hardware. */
          BOARD_InitBootPins();
          BOARD_InitBootClocks();
          BOARD_InitBootPeripherals();
#ifndef BOARD_INIT_DEBUG_CONSOLE_PERIPHERAL
          /* Init FSL debug console. */
          BOARD_InitDebugConsole();
#endif
          /* Initialize OLED */
          OLED_Init(I2C0_PERIPHERAL);

          while(1) {

                    OLED_Clear_Screen(0);
                    sprintf(sbuff, "ADC: %5d", adcValue);
                    OLED_Puts(0, 1, sbuff);
                    OLED_Refresh_Gram();
          }
          return 0 ;
}
```

6. Connect the display and check its operation.

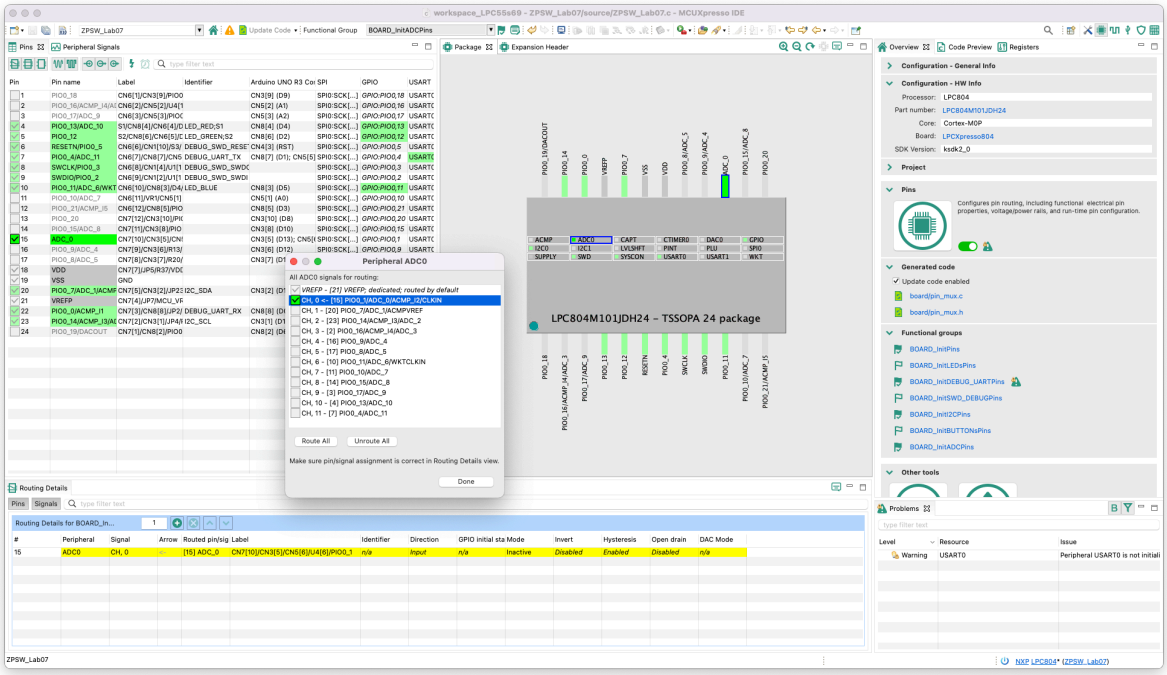


II. A/D Converter

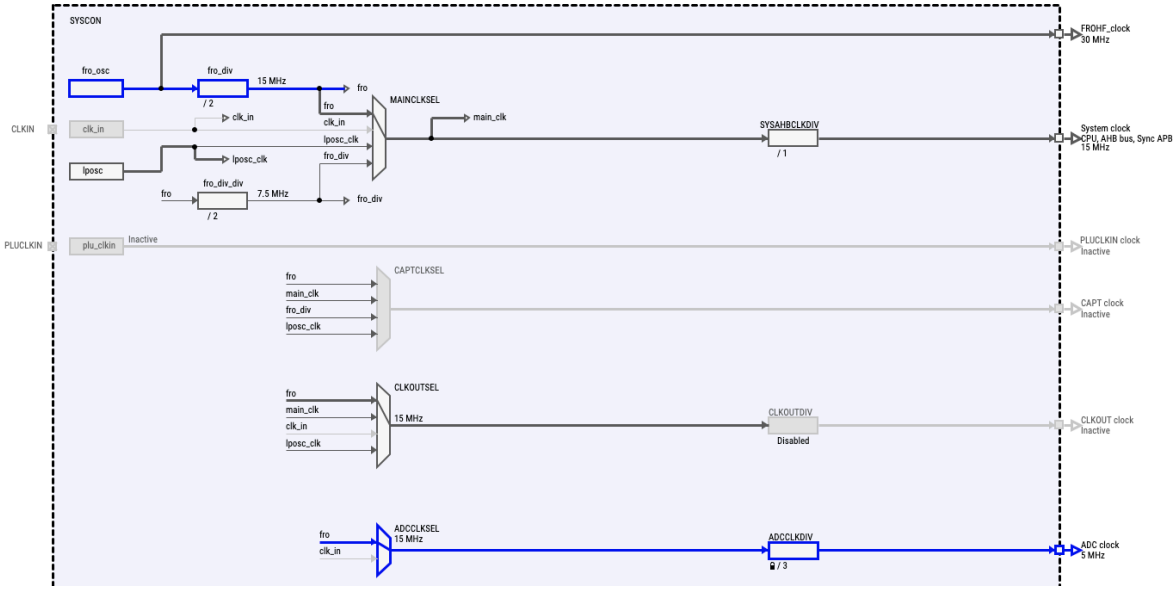
1. Go to *Config Tool -> Pins* and create a new preset called *BOARD_InitADCPins*:



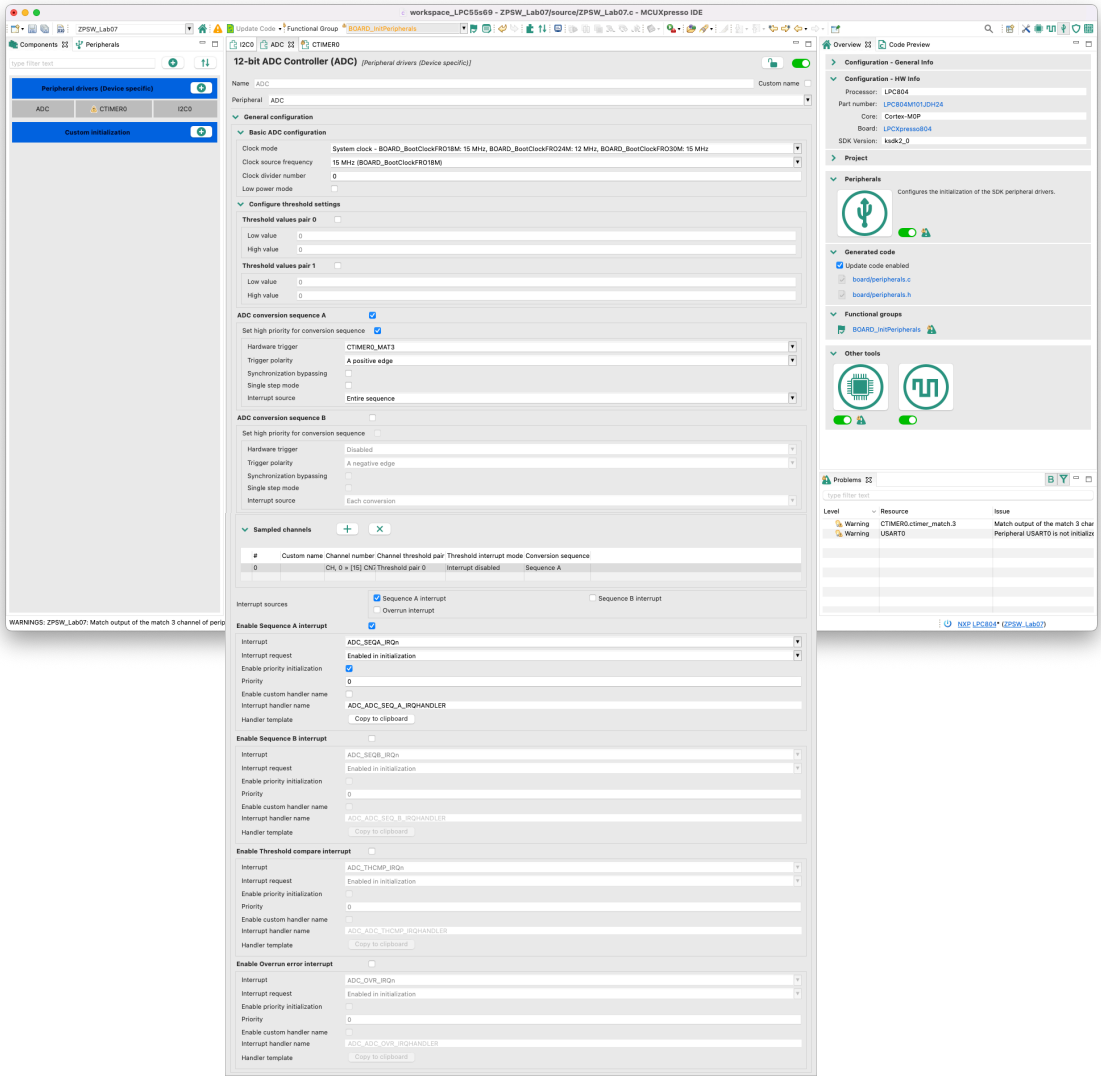
2. Click on the ADC block and connect the *ADC0* signal (*PIO0_1* pin). Disable the default *Pull-Up* by setting the *Mode* field to Inactive:



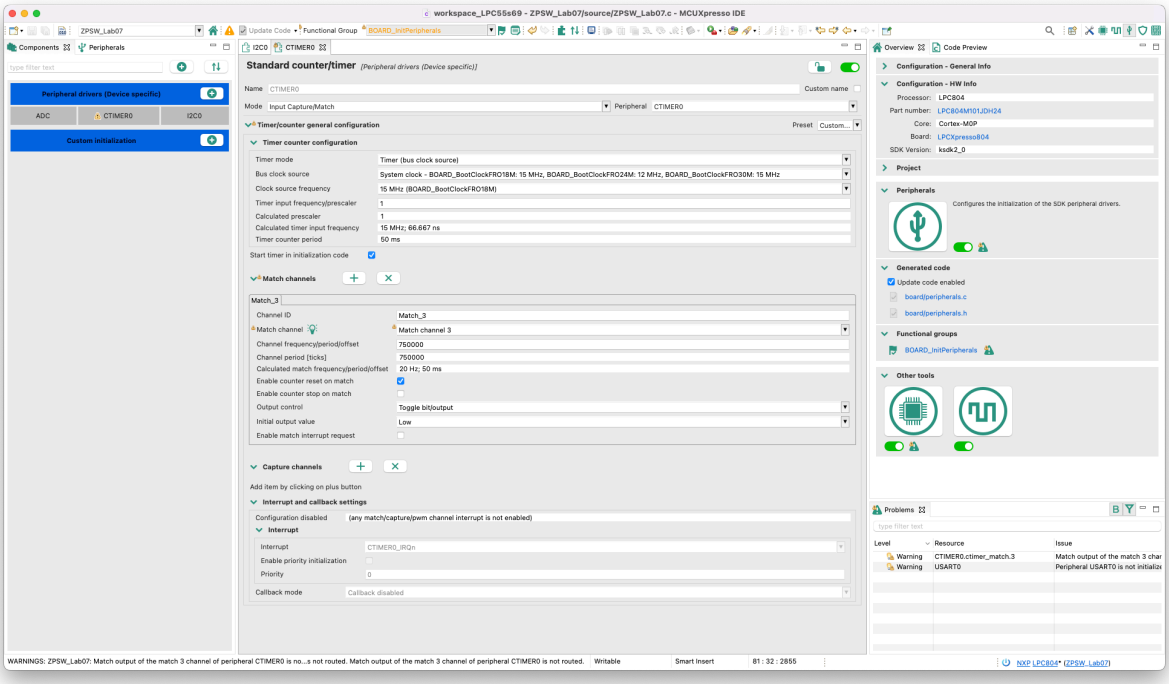
3. Go to Clocks and turn on the ADC clock 5 MHz for the A / D converter:



4. Go to ADC settings and enter the following configuration:



5. Go to Peripherals, select *CTIMER* and configure it to change the state of the output at a frequency of 20 Hz:



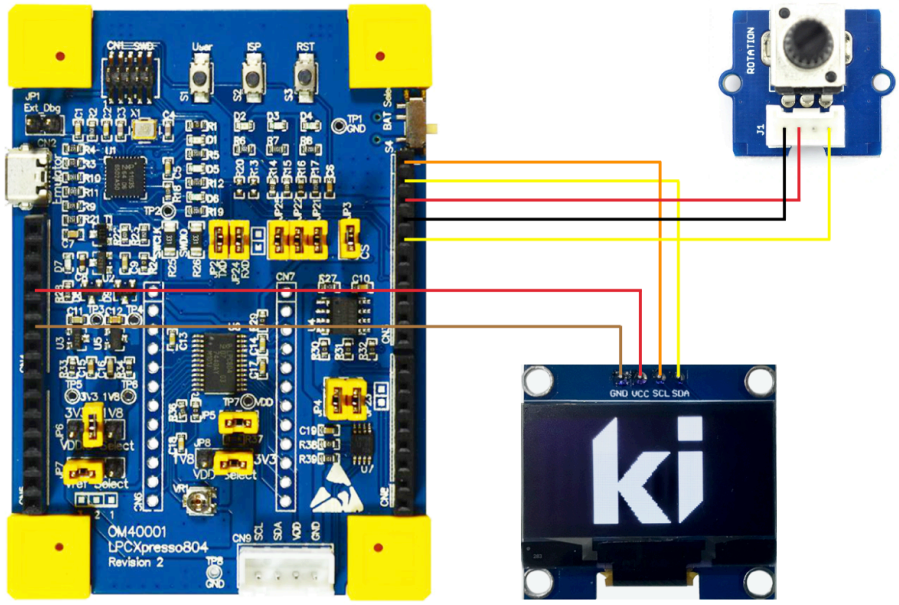
The *ADC* converter will be triggered by only one edge, therefore its sampling frequency will be twice lower - i.e. 10 Hz.

6. Go to the main project file and modify the code as below:

```
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "LPC804.h"
#include "fsl_debug_console.h"
#include "fsl_power.h"
#include "oled.h"

static adc_result_info_t gAdcResultInfoStruct;
adc_result_info_t *volatile gAdcResultInfoPtr = &gAdcResultInfoStruct;
char sbuff[32];
volatile uint16_t adcValue = 0;


/* ADC_SEQA_IRQn interrupt handler */
void ADC_ADC_SEQ_A_IRQHANDLER(void) {
          /* Get status flags */
          if (kADC_ConvSeqAInterruptFlag == (kADC_ConvSeqAInterruptFlag & ADC_GetStatusFlags(ADC_PERIPHERAL))) {
                    /* Place your interrupt code here */
                    ADC_GetChannelConversionResult(ADC_PERIPHERAL, 0, gAdcResultInfoPtr);
                    adcValue = gAdcResultInfoStruct.result;
                    /* Clear status flags */
                    ADC_ClearStatusFlags(ADC_PERIPHERAL, kADC_ConvSeqAInterruptFlag);
          }
}

/*
 * @brief   Application entry point.
 */
int main(void) {

          /* Power on ADC. */
          POWER_DisablePD(kPDRUNCFG_PD_ADC0);
          /* Init board hardware. */
          BOARD_InitBootPins();
          BOARD_InitBootClocks();
          BOARD_InitBootPeripherals();
#ifndef BOARD_INIT_DEBUG_CONSOLE_PERIPHERAL
          /* Init FSL debug console. */
          BOARD_InitDebugConsole();
#endif
          /* Initialize OLED */
```

```
        OLED_Init(I2C0_PERIPHERAL);

        while(1) {

                OLED_Clear_Screen(0);
                sprintf(sbuff, "ADC: %5d", adcValue);
                OLED_Puts(0, 1, sbuff);
                OLED_Refresh_Gram();
        }
        return 0 ;
}
```

7. Connect the potentiometer to the board, program the microcontroller and check the example. By moving the potentiometer axis, the displayed value should change in the range of 0-4095 (12-bit resolution), which corresponds to the input voltage of 0-3.3 V.



III. GUI - a simple analog indicator

1. Modify the project code:

```
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "LPC804.h"
#include "fsl_debug_console.h"
#include "fsl_power.h"
#include "oled.h"

static adc_result_info_t gAdcResultInfoStruct;
adc_result_info_t *volatile gAdcResultInfoPtr = &gAdcResultInfoStruct;
char sbuff[32];
volatile uint16_t adcValue = 0;
float data=0;

void Gauge(uint8_t x0, uint8_t y0, uint8_t radius, float v) {

        float k= (v*270) - 135; // degrees
        float p, q=(2*PI*k)/360.0;
        uint8_t radius0 = radius * 0.9;

        for(int i=-135; i<=135;i+=15) {

                p=(2*PI*i)/360.0;
                OLED_Draw_Line(x0 + radius0*sinf(p), y0 - radius0*cosf(p), x0 + radius*sinf(p), y0 - radius*cosf(p));
        }
        OLED_Draw_Line(x0, y0 , x0 + radius*sinf(q), y0 - radius*cosf(q));
}
```

```
/* ADC_SEQA_IRQn interrupt handler */
void ADC_ADC_SEQ_A_IRQHANDLER(void) {
          /* Get status flags */
          if (kADC_ConvSeqAInterruptFlag == (kADC_ConvSeqAInterruptFlag & ADC_GetStatusFlags(ADC_PERIPHERAL))) {
                    /* Place your interrupt code here */
                    ADC_GetChannelConversionResult(ADC_PERIPHERAL, 0, gAdcResultInfoPtr);
                    adcValue = gAdcResultInfoStruct.result;
                    /* Clear status flags */
                    ADC_ClearStatusFlags(ADC_PERIPHERAL, kADC_ConvSeqAInterruptFlag);
          }
}

/*
 * @brief   Application entry point.
 */
int main(void) {

          /* Power on ADC. */
          POWER_DisablePD(kPDRUNCFG_PD_ADC0);
          /* Init board hardware. */
          BOARD_InitBootPins();
          BOARD_InitBootClocks();
          BOARD_InitBootPeripherals();
#ifndef BOARD_INIT_DEBUG_CONSOLE_PERIPHERAL
          /* Init FSL debug console. */
          BOARD_InitDebugConsole();
#endif
          /* Initialize OLED */
          OLED_Init(I2C0_PERIPHERAL);

          while(1) {

                    OLED_Clear_Screen(0);

                    data=adcValue/4095.0;

                    Gauge(64, 32, 32, data);
                    sprintf(sbuff, "%3d%%", (uint8_t)(data*100));
                    OLED_Puts(50, 7, sbuff);

                    OLED_Refresh_Gram();
          }
          return 0 ;
}
```

2. Build the project in ***Release*** mode, program the microcontroller and check the example.

IV. Exercises

3. Modify the appearance of the analog indicator as you see fit.
4. Write a function that draws the *n-last* samples in the form of a bar graph. The graph is to move across the display screen (horizontally or vertically).