ENGINE

Teaching online electronics, microcontrollers and programming in Higher Education

Programing of embedded systems

7. Przetwornik A/C

Lead Partner: Warsaw University of Technology

Authors: Daniel Król

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 Where other published and unpublished source materials have been used, these have been acknowledged.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

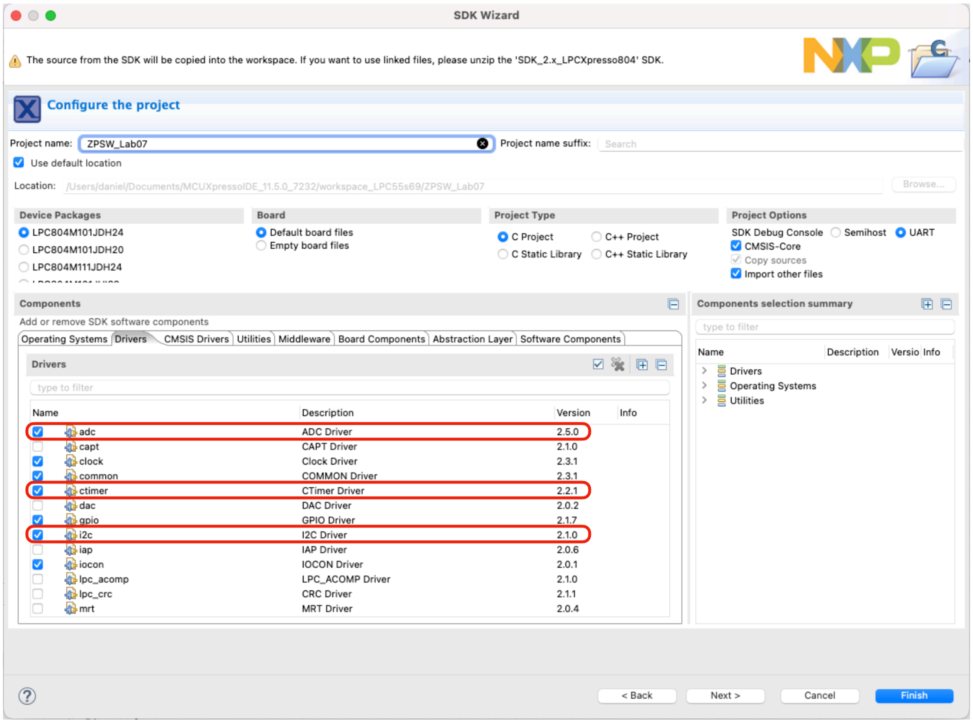
Funding Disclaimer

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. Przetwornik A/C

I. Wyświetlacz OLED

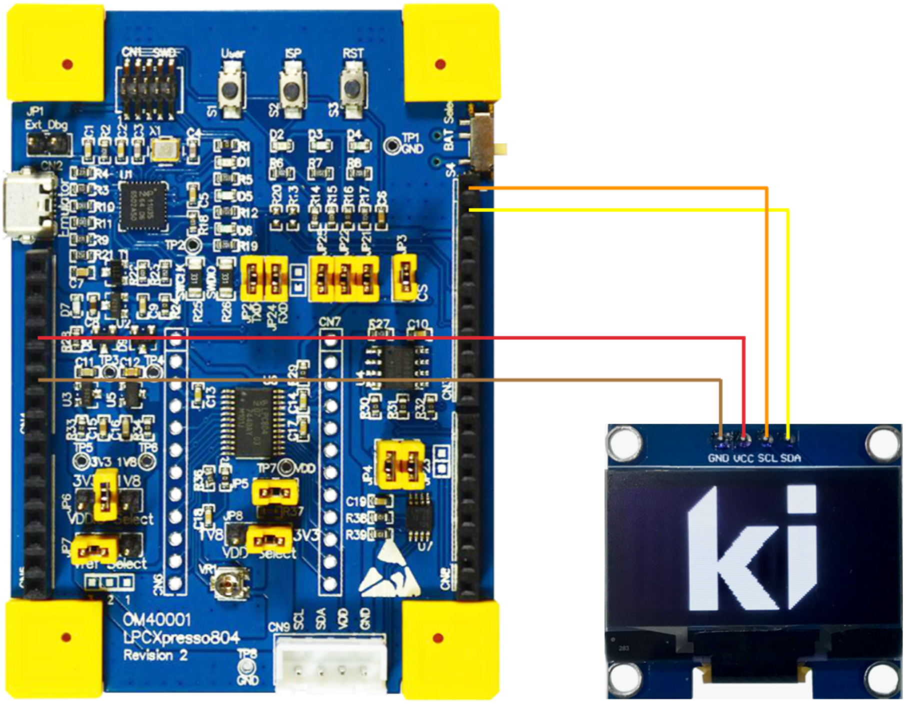
1. Stwórz nowy projekt dla płyty *LPCXpresso804*. Nazwij projekt np. *ZPSW_Lab07* i dodaj sterowniki *ADC*, *CTIMER* oraz *I2C*:



2. Dodaj bibliotekę OLED i skonfiguruj obsługę wyświetlacza jak w poprzedniej instrukcji.
3. W *Config Tools -> Clocks* zmień częstotliwość generatora *FRO_OSC* na 30 MHz.
4. Przejdź do głównego pliku projektu i zmodyfikuj kod jak poniżej:

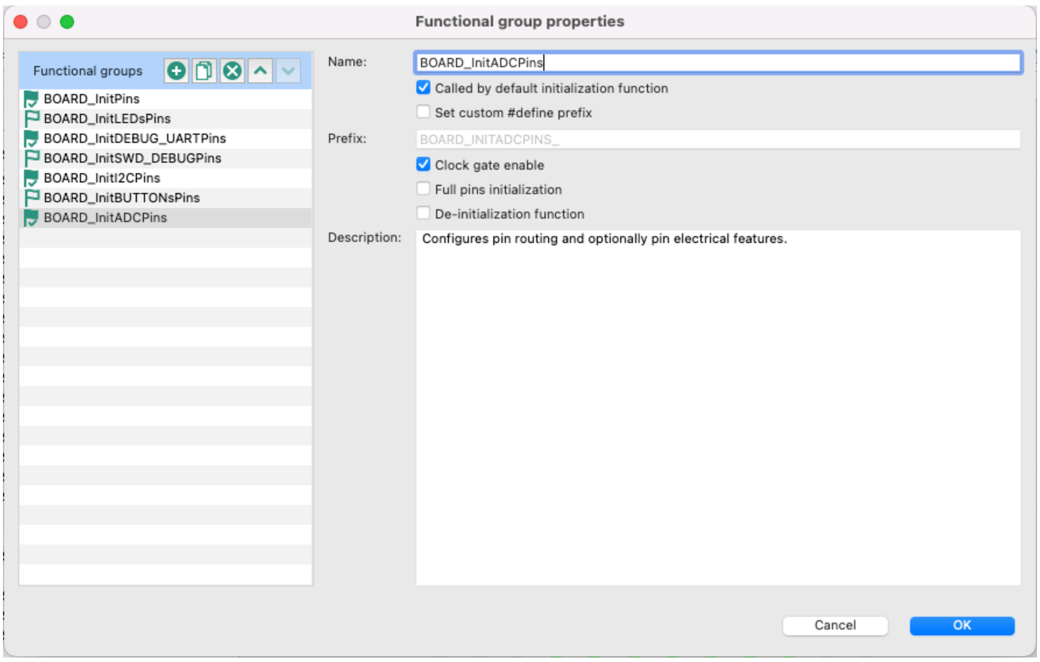
```
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "LPC804.h"
#include "fsl_debug_console.h"
#include "oled.h"

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 ;
}
```

5. Podłącz wyświetlacz i sprawdź jego działanie.

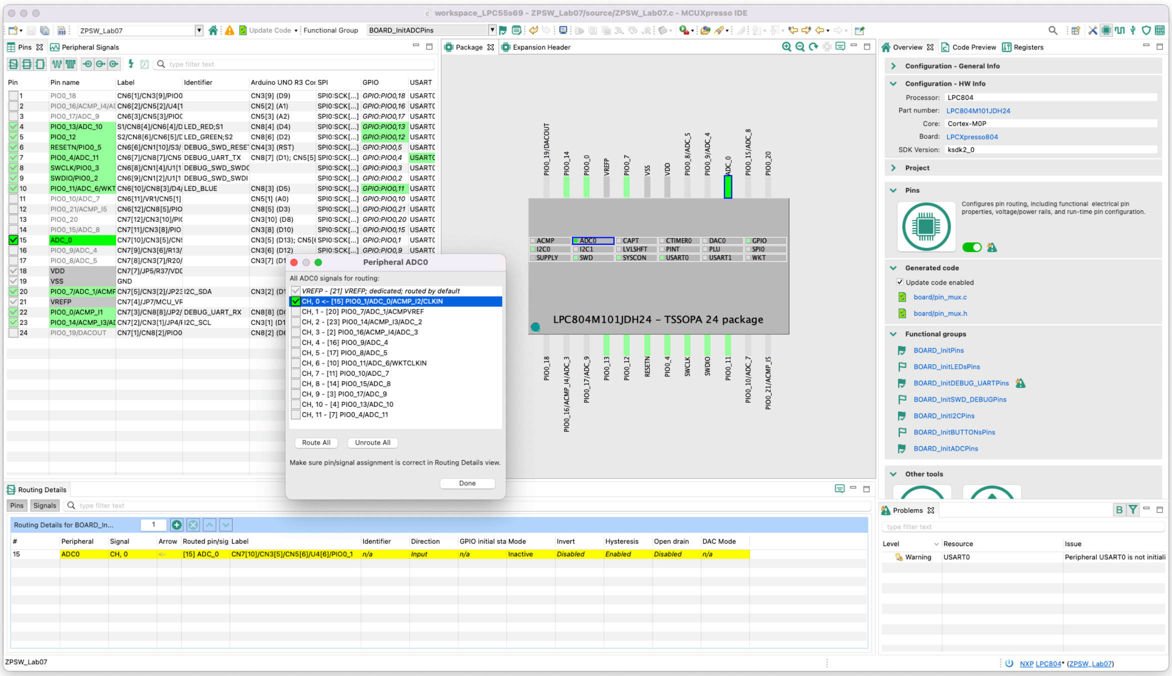


II. Przetwornik A/C

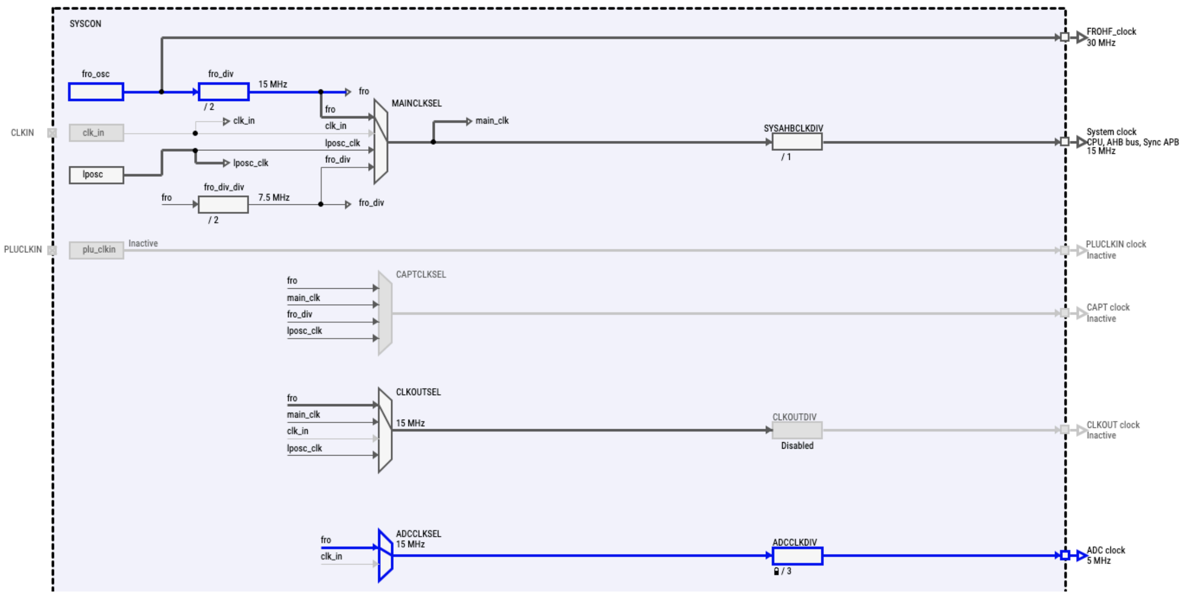
1. Przejdź do Config Tool -> Pins i utwórz nowy preset o nazwie *BOARD_InitADCPins*:



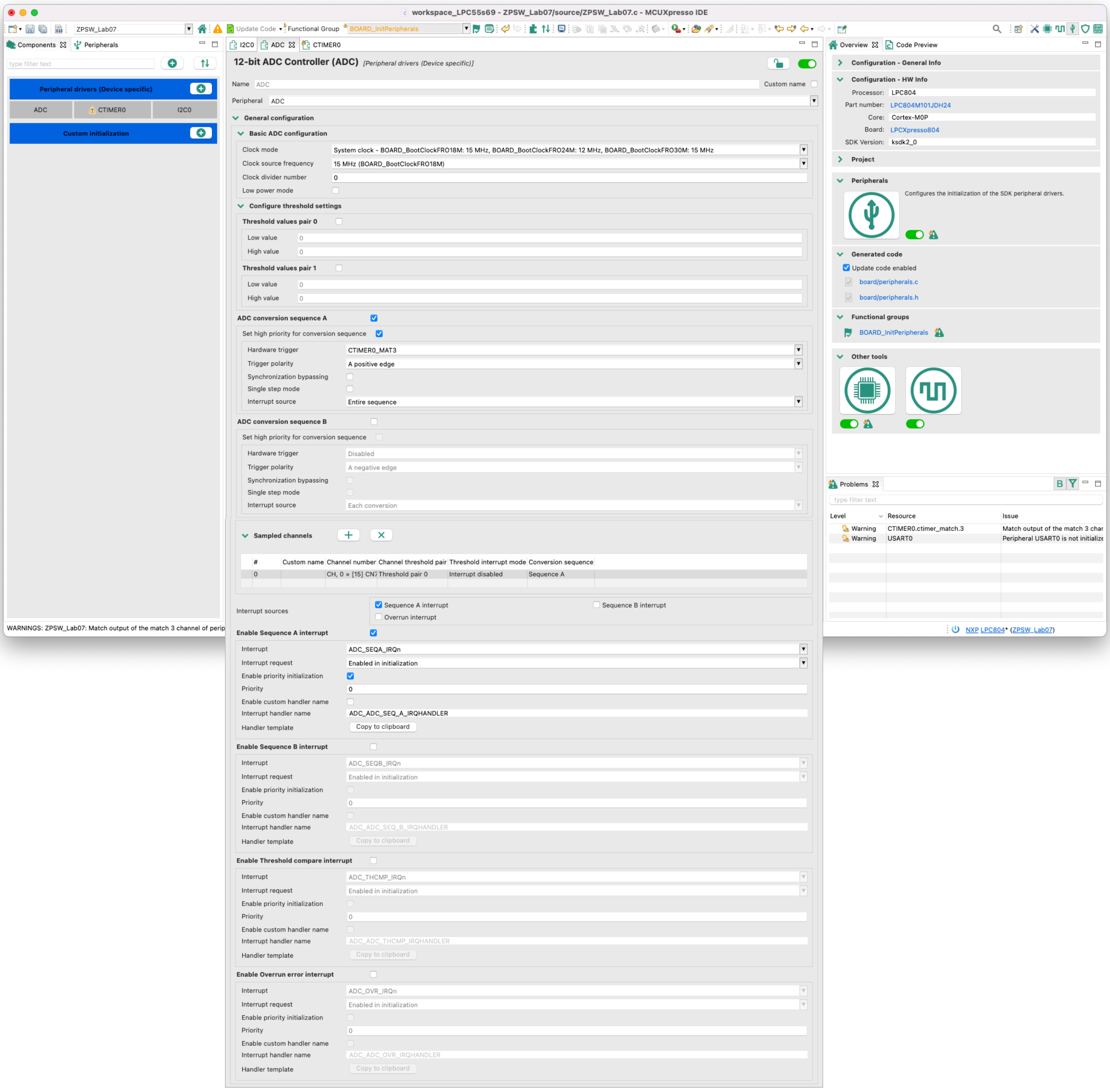
2. Kliknij w blok *ADC* i podłącz sygnał *ADC0* (wyprowadzenie PIO0_1). Wyłącz domyślny *Pull-Up* ustawiając w polu *Mode* wartość *Inactive*:



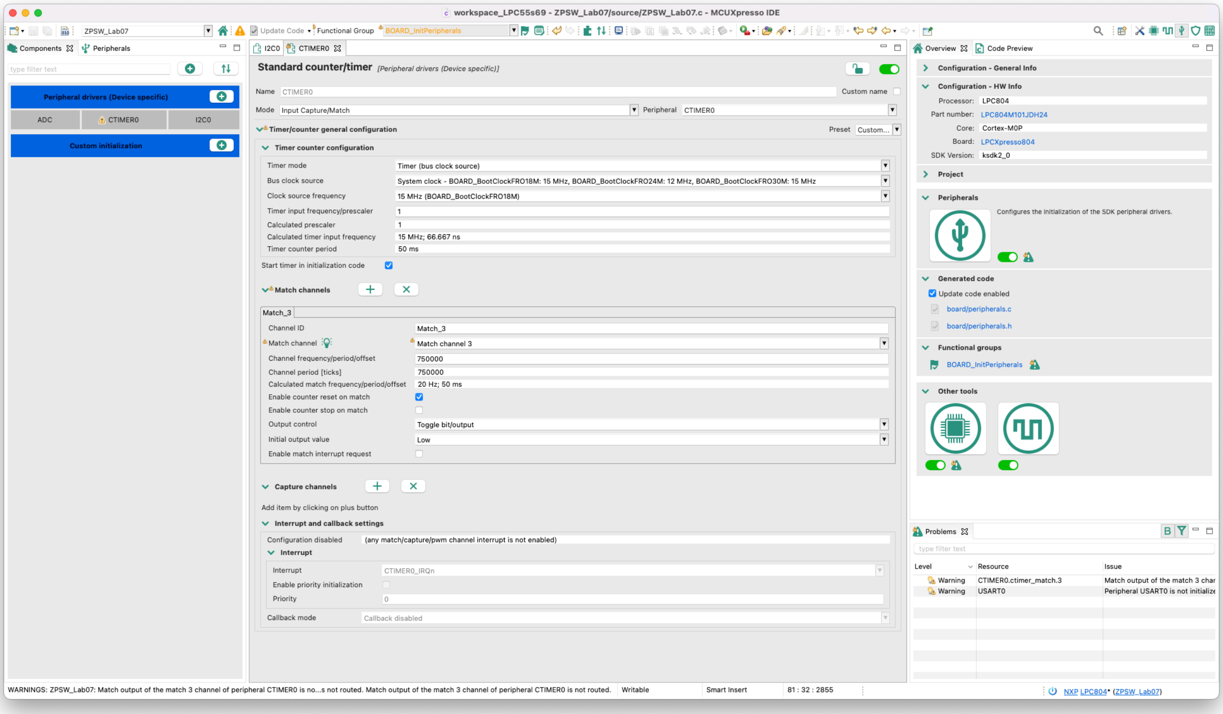
3. Przejdź do *Clocks* i włącz sygnał zegarowy *ADC clock* 5 MHz dla przetwornika A/C:



4. Przejdź do ustawień przetwornika *ADC* i wprowadź poniższą konfigurację:



5. Przejdź do *Peripherals*, wybierz *CTIMER* i skonfiguruj go dla zmiany stanu wyjścia z częstotliwością 20 Hz:



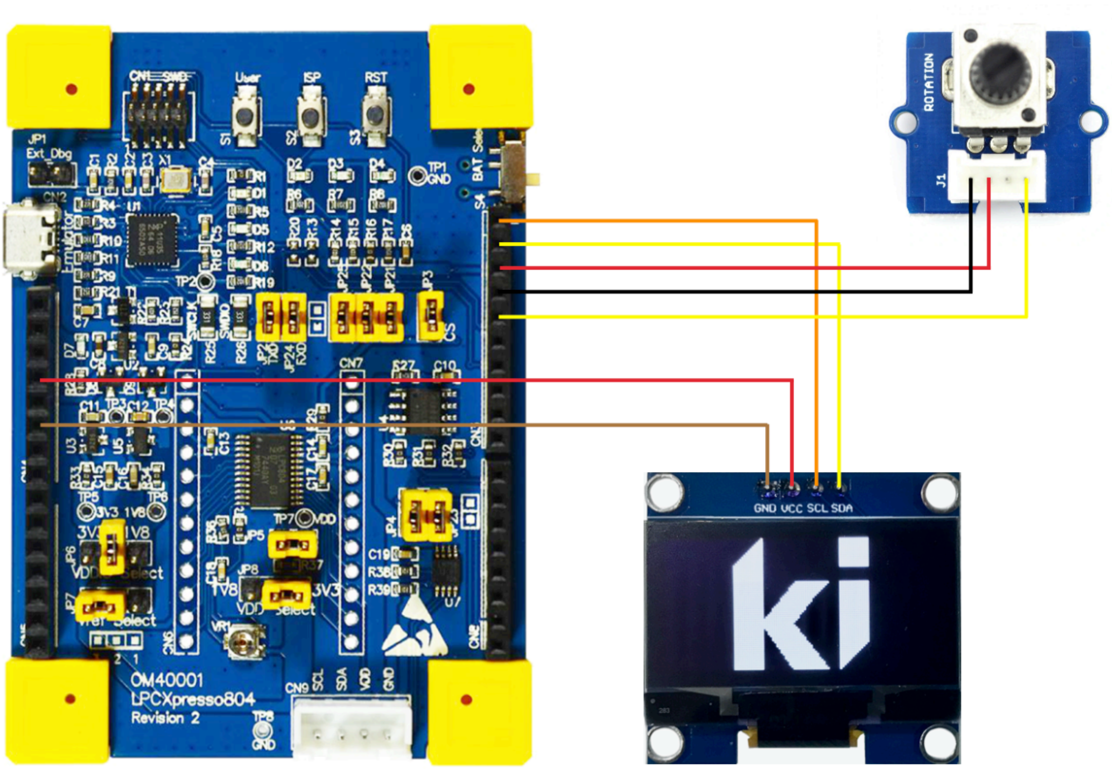
Przetwornik ADC będzie wyzwalany tylko jednym zboczem, dlatego jego częstotliwość próbkowania będzie dwukrotnie niższa - czyli 10 Hz.

6. Przejdź do głównego pliku projektu i zmodyfikuj kod jak poniżej:

```
#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. Podłącz potencjometr do płytki, zaprogramuj układ i sprawdź działanie przykładu. Poruszając osią potencjometru, wyświetlana wartość powinna się zmieniać w zakresie 0-4095 (12-bitowa rozdzielczość) co odpowiada napięciu wejściowemu 0-3.3 V.



III. GUI - prosty wskaźnik analogowy

1. Zmodyfikuj kod projektu:

```
#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. Zbuduj projekt w trybie *Release*, zaprogramuj układ i sprawdź działanie przykładu.

IV. Zadania

1. Zmodyfikuj wygląd wskaźnika analogowego według własnego uznania.
2. Napisz funkcję rysującą *n*-ostatnich próbek w postaci wykresu słupkowego. Wykres ma przesuwać się po ekranie wyświetlacza (poziomu lub pionowo).