ENGINE

Teaching online electronics, microcontrollers and programming in Higher Education

Programing of embedded systems

8. Joystick analogowy

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 Where other published and unpublished source materials have been used, these have been acknowledged.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

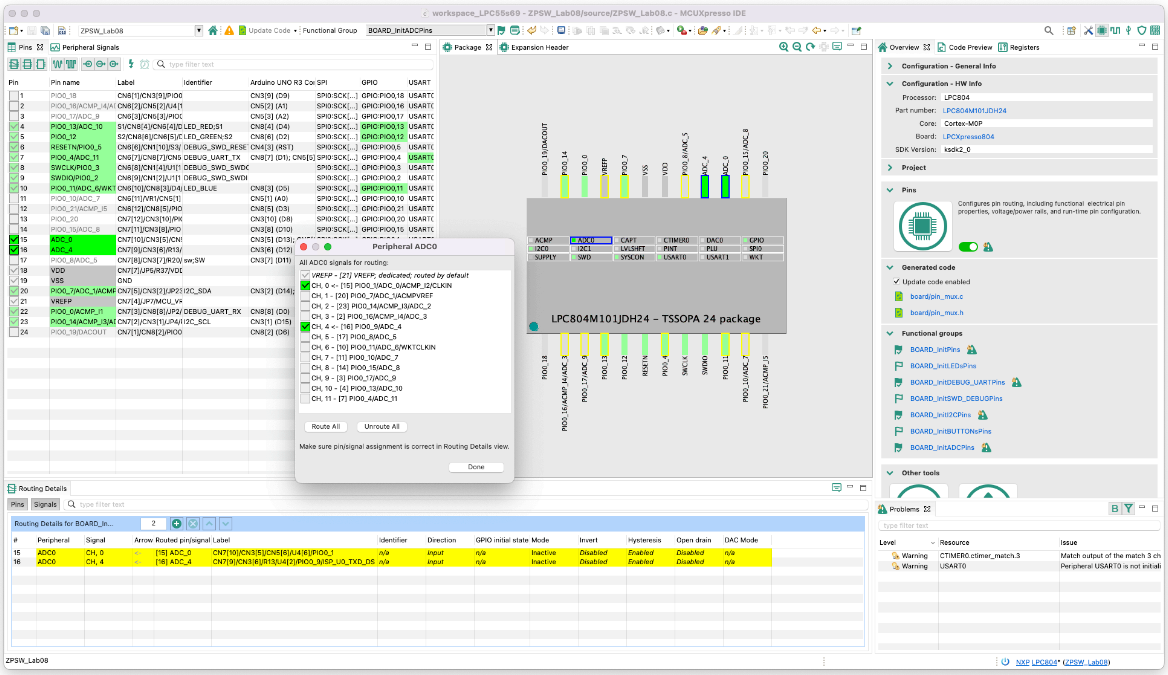
This project has been funded with support from the European Commission. This report reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Programing of embedded systems

8. Joystick analogowy

I. Przetwornik A/C

1. Skopiuj projekt z poprzednich zajęć i nazwij go np. *ZPSW_Lab08*.
2. Przejdź do *Config Tool* -> *Pins* i otwórz preset *BOARD_InitADCPins*. Kliknij w blok *ADC* i do istniejącego sygnału *ADC0* (wyprowadzenie *PIO0_1*), analogicznie dodaj sygnał *ADC4* (wyprowadzenie *PIO0_9*):



Peripheral ADC0

All ADC0 signals for routing:

- ☑ *VREFP - [21] VREFP; dedicated; routed by default*
- ☑ CH, 0 <- [15] PIO0_1/ADC_0/ACMP_I2/CLKIN
- □ CH, 1 - [20] PIO0_7/ADC_1/ACMPVREF
- □ CH, 2 - [23] PIO0_14/ACMP_I3/ADC_2
- □ CH, 3 - [2] PIO0_16/ACMP_I4/ADC_3
- ☑ CH, 4 <- [16] PIO0_9/ADC_4
- □ CH, 5 - [17] PIO0_8/ADC_5
- □ CH, 6 - [10] PIO0_11/ADC_6/WKTCLKIN
- □ CH, 7 - [11] PIO0_10/ADC_7
- □ CH, 8 - [14] PIO0_15/ADC_8
- □ CH, 9 - [3] PIO0_17/ADC_9
- □ CH, 10 - [4] PIO0_13/ADC_10
- □ CH, 11 - [7] PIO0_4/ADC_11

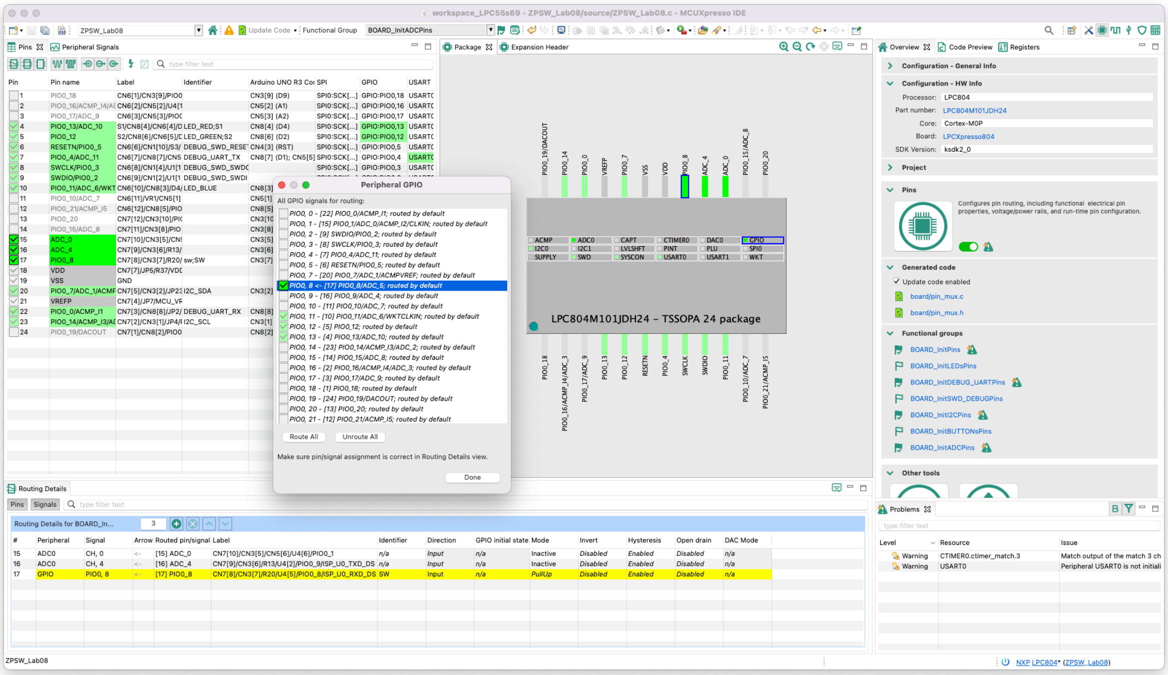
Make sure pin/signal assignment is correct in Routing Details view.

Routing Details for BOARD_In... 2

#	Peripheral	Signal	Arrow	Routed pin/signal	Label	Identifier	Direction	GPIO initial state	Mode	Invert	Hysteresis	Open drain	DAC Mode
15	ADC0	CH, 0	<-	[15] ADC_0	CN7[10]/CN3[5]/CN5[6]/U4[6]/PIO0_1	n/a	Input	n/a	Inactive	Disabled	Enabled	Disabled	n/a
16	ADC0	CH, 4	<-	[16] ADC_4	CN7[9]/CN3[6]/R13/U4[2]/PIO0_9/ISP_U0_TXD_DS	n/a	Input	n/a	Inactive	Disabled	Enabled	Disabled	n/a

workspace_LPC55s69 - ZPSW_Lab08/source/ZPSW_Lab08.c - MCUXpresso IDE
LPC804M101JDH24 – TSSOPA 24 package
Route All
Unroute All
Done

3. Dodaj wyprowadzenie *PIO0_8* jako wejściowe z *PullUp* i dodaj identyfikator SW:



Peripheral GPIO

All GPIO signals for routing:

- □ *PIO0, 0 - [22] PIO0_0/ACMP_I1; routed by default*
- □ *PIO0, 1 - [15] PIO0_1/ADC_0/ACMP_I2/CLKIN; routed by default*
- □ *PIO0, 2 - [9] SWDIO/PIO0_2; routed by default*
- □ *PIO0, 3 - [8] SWCLK/PIO0_3; routed by default*
- □ *PIO0, 4 - [7] PIO0_4/ADC_11; routed by default*
- □ *PIO0, 5 - [6] RESETN/PIO0_5; routed by default*
- □ *PIO0, 7 - [20] PIO0_7/ADC_1/ACMPVREF; routed by default*
- ☑ *PIO0, 8 <- [17] PIO0_8/ADC_5; routed by default*
- □ *PIO0, 9 - [16] PIO0_9/ADC_4; routed by default*
- □ *PIO0, 10 - [11] PIO0_10/ADC_7; routed by default*
- ☑ *PIO0, 11 - [10] PIO0_11/ADC_6/WKTCLKIN; routed by default*
- ☑ *PIO0, 12 - [5] PIO0_12; routed by default*
- ☑ *PIO0, 13 - [4] PIO0_13/ADC_10; routed by default*
- □ *PIO0, 14 - [23] PIO0_14/ACMP_I3/ADC_2; routed by default*
- □ *PIO0, 15 - [14] PIO0_15/ADC_8; routed by default*
- □ *PIO0, 16 - [2] PIO0_16/ACMP_I4/ADC_3; routed by default*
- □ *PIO0, 17 - [3] PIO0_17/ADC_9; routed by default*
- □ *PIO0, 18 - [1] PIO0_18; routed by default*
- □ *PIO0, 19 - [24] PIO0_19/DACOUT; routed by default*
- □ *PIO0, 20 - [13] PIO0_20; routed by default*
- □ *PIO0, 21 - [12] PIO0_21/ACMP_I5; routed by default*

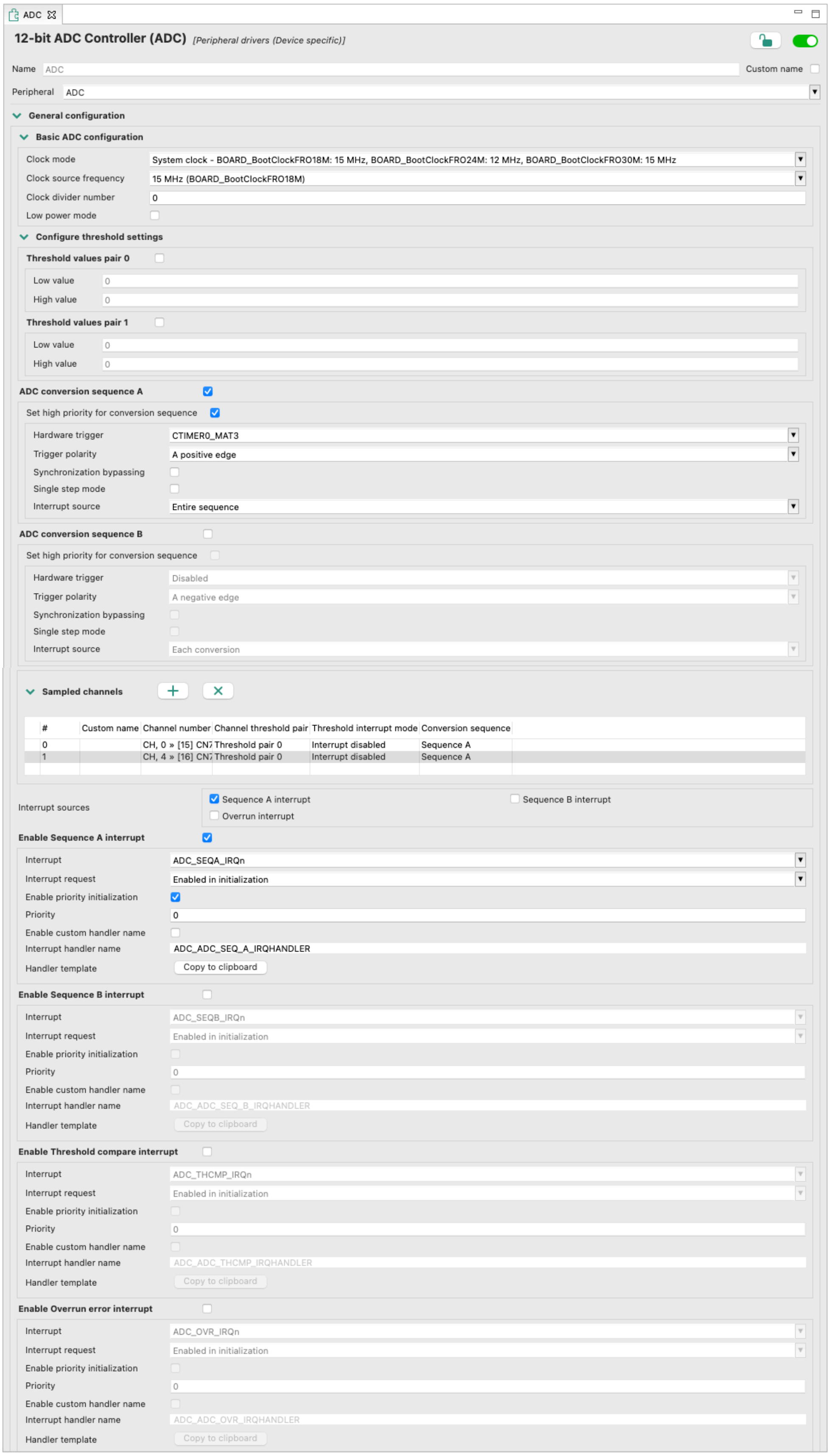
Make sure pin/signal assignment is correct in Routing Details view.

Routing Details for BOARD_In... 3

#	Peripheral	Signal	Arrow	Routed pin/signal	Label	Identifier	Direction	GPIO initial state	Mode	Invert	Hysteresis	Open drain	DAC Mode
15	ADC0	CH, 0	<-	[15] ADC_0	CN7[10]/CN3[5]/CN5[6]/U4[6]/PIO0_1	n/a	Input	n/a	Inactive	Disabled	Enabled	Disabled	n/a
16	ADC0	CH, 4	<-	[16] ADC_4	CN7[9]/CN3[6]/R13/U4[2]/PIO0_9/ISP_U0_TXD_DS	n/a	Input	n/a	Inactive	Disabled	Enabled	Disabled	n/a
17	GPIO	PIO0, 8	<-	[17] PIO0_8	CN7[8]/CN3[7]/R20/U4[5]/PIO0_8/ISP_U0_RXD_DS	SW	Input	n/a	PullUp	Disabled	Enabled	Disabled	n/a

workspace_LPC55s69 - ZPSW_Lab08/source/ZPSW_Lab08.c - MCUXpresso IDE
LPC804M101JDH24 – TSSOPA 24 package
Route All
Unroute All
Done

4. Przejdź do ustawień przetwornika *ADC* i zmień jego konfigurację przez dodanie dodatkowego kanału (*CH 4*):



ADC

12-bit ADC Controller (ADC) *[Peripheral drivers (Device specific)]*

Name	ADC	Custom name	□
Peripheral	ADC		

General configuration

Basic ADC configuration

Clock mode	System clock - BOARD_BootClockFRO18M: 15 MHz, BOARD_BootClockFRO24M: 12 MHz, BOARD_BootClockFRO30M: 15 MHz
Clock source frequency	15 MHz (BOARD_BootClockFRO18M)
Clock divider number	0
Low power mode	□

Configure threshold settings

Threshold values pair 0	□
Low value	0
High value	0
Threshold values pair 1	□
Low value	0
High value	0

ADC conversion sequence A	☑
Set high priority for conversion sequence	☑
Hardware trigger	CTIMER0_MAT3
Trigger polarity	A positive edge
Synchronization bypassing	□
Single step mode	□
Interrupt source	Entire sequence
ADC conversion sequence B	□
Set high priority for conversion sequence	□
Hardware trigger	Disabled
Trigger polarity	A negative edge
Synchronization bypassing	□
Single step mode	□
Interrupt source	Each conversion

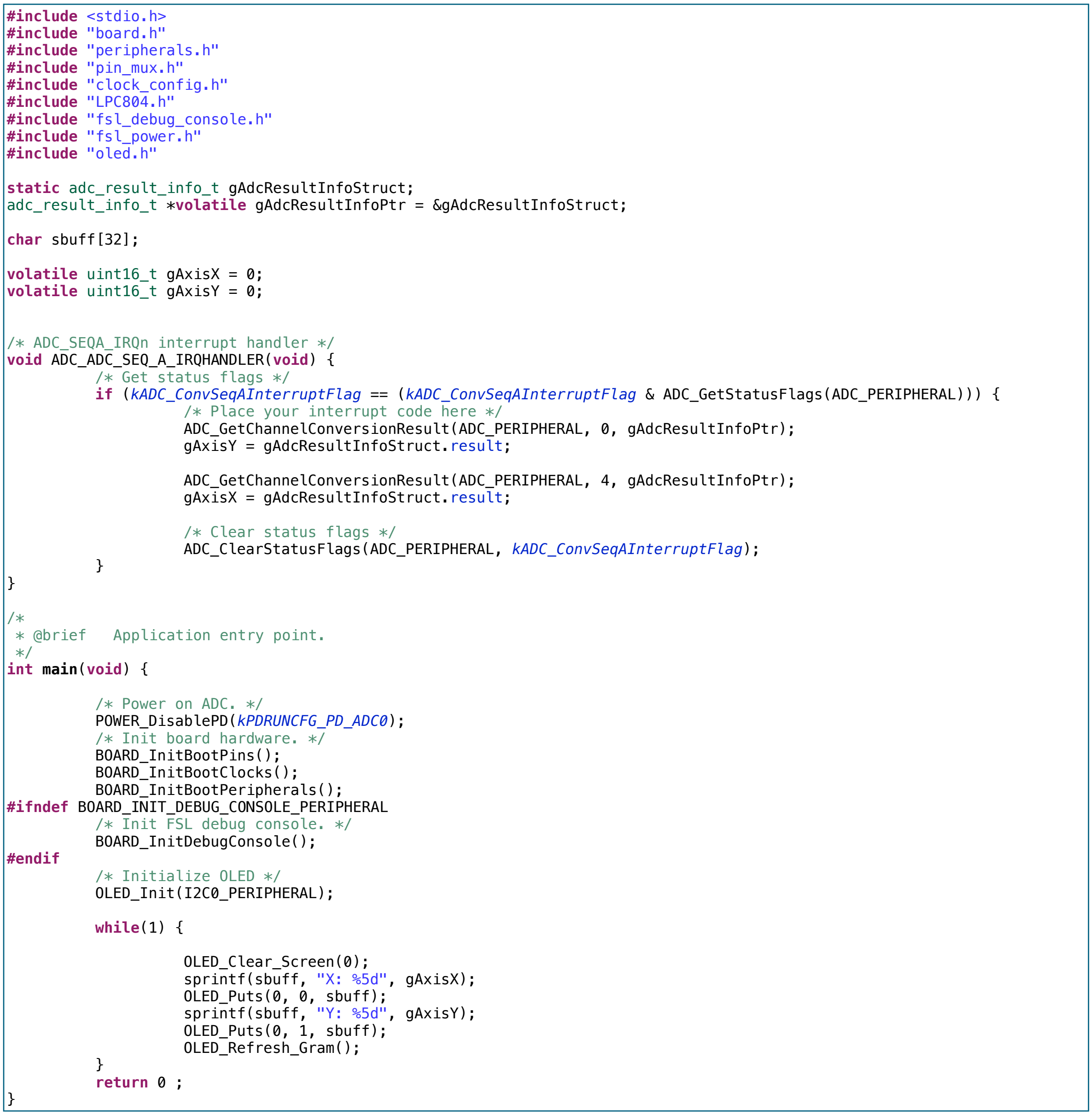
Sampled channels + ×

#	Custom name	Channel number	Channel threshold pair	Threshold interrupt mode	Conversion sequence
0		CH, 0 » [15] CN7	Threshold pair 0	Interrupt disabled	Sequence A
1		CH, 4 » [16] CN7	Threshold pair 0	Interrupt disabled	Sequence A

Interrupt sources	☑ Sequence A interrupt □ Overrun interrupt	□ Sequence B interrupt

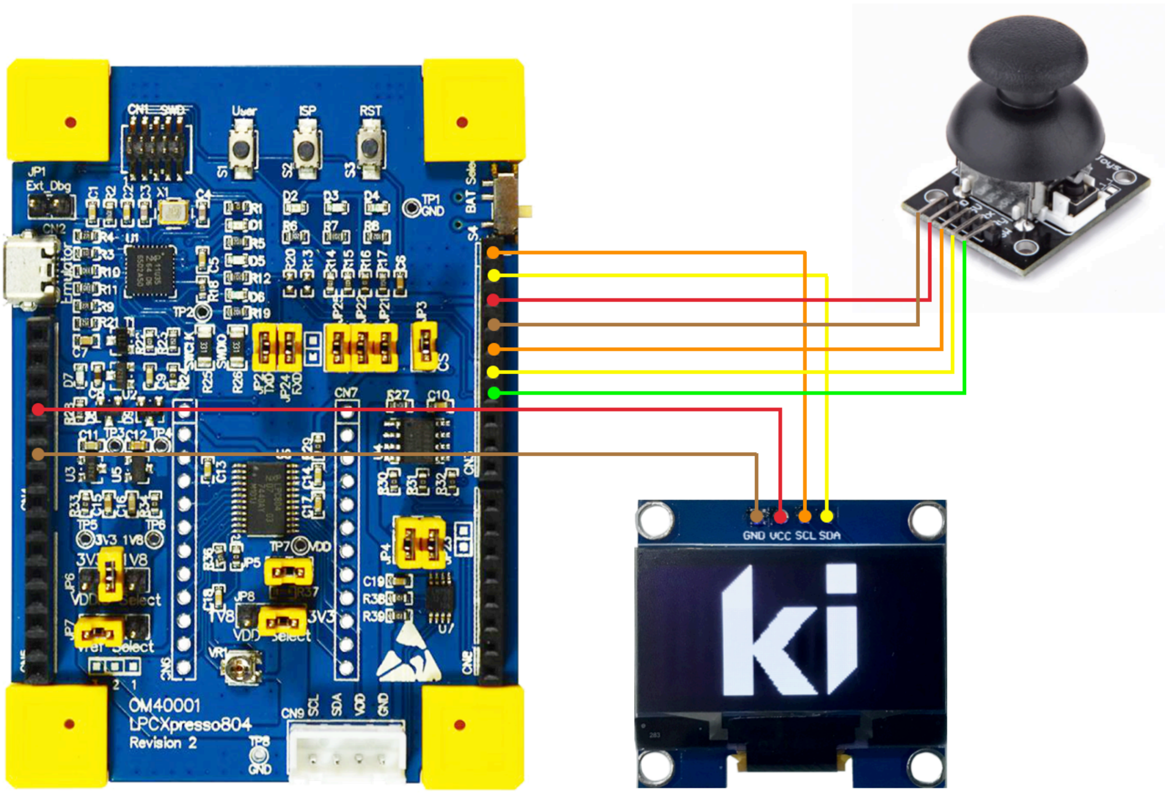
Enable Sequence A interrupt	☑
Interrupt	ADC_SEQA_IRQn
Interrupt request	Enabled in initialization
Enable priority initialization	☑
Priority	0
Enable custom handler name	□
Interrupt handler name	ADC_ADC_SEQ_A_IRQHANDLER
Handler template	Copy to clipboard
Enable Sequence B interrupt	□
Interrupt	ADC_SEQB_IRQn
Interrupt request	Enabled in initialization
Enable priority initialization	□
Priority	0
Enable custom handler name	□
Interrupt handler name	ADC_ADC_SEQ_B_IRQHANDLER
Handler template	Copy to clipboard
Enable Threshold compare interrupt	□
Interrupt	ADC_THCMP_IRQn
Interrupt request	Enabled in initialization
Enable priority initialization	□
Priority	0
Enable custom handler name	□
Interrupt handler name	ADC_ADC_THCMP_IRQHANDLER
Handler template	Copy to clipboard
Enable Overrun error interrupt	□
Interrupt	ADC_OVR_IRQn
Interrupt request	Enabled in initialization
Enable priority initialization	□
Priority	0
Enable custom handler name	□
Interrupt handler name	ADC_ADC_OVR_IRQHANDLER
Handler template	Copy to clipboard

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



```c
#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 gAxisX = 0;
volatile uint16_t gAxisY = 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);
                gAxisY = gAdcResultInfoStruct.result;

                ADC_GetChannelConversionResult(ADC_PERIPHERAL, 4, gAdcResultInfoPtr);
                gAxisX = 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, "X: %5d", gAxisX);
                OLED_Puts(0, 0, sbuff);
                sprintf(sbuff, "Y: %5d", gAxisY);
                OLED_Puts(0, 1, sbuff);
                OLED_Refresh_Gram();
        }
        return 0 ;
}
```

6. Podłącz wyświetlacz oraz joystick do płytki według poniższego schematu:



7. Zaprogramuj układ i sprawdź działanie przykładu.

II. Obsługa przycisku

1. Zmodyfikuj kod projektu przez dodanie obsługi przycisku w osi Z:

```c
#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 gAxisX = 0;
volatile uint16_t gAxisY = 0;
volatile bool     gAxisZ = 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);
                    gAxisY = gAdcResultInfoStruct.result;

                    ADC_GetChannelConversionResult(ADC_PERIPHERAL, 4, gAdcResultInfoPtr);
                    gAxisX = gAdcResultInfoStruct.result;

                    gAxisZ = GPIO_PinRead(BOARD_INITADCPINS_SW_GPIO,
                                        BOARD_INITADCPINS_SW_PORT,
                                        BOARD_INITADCPINS_SW_PIN);

                    /* Clear status flags */
                    ADC_ClearStatusFlags(ADC_PERIPHERAL, kADC_ConvSeqAInterruptFlag);
          }
}
```

```c
/*
 * @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, "X: %5d", gAxisX);
                    OLED_Puts(0, 0, sbuff);
                    sprintf(sbuff, "Y: %5d", gAxisY);
                    OLED_Puts(0, 1, sbuff);
                    sprintf(sbuff, "Z: %5d", gAxisZ);
                    OLED_Puts(0, 2, sbuff);
                    OLED_Refresh_Gram();
          }
          return 0 ;
}
```

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

III. Obsługa kursora

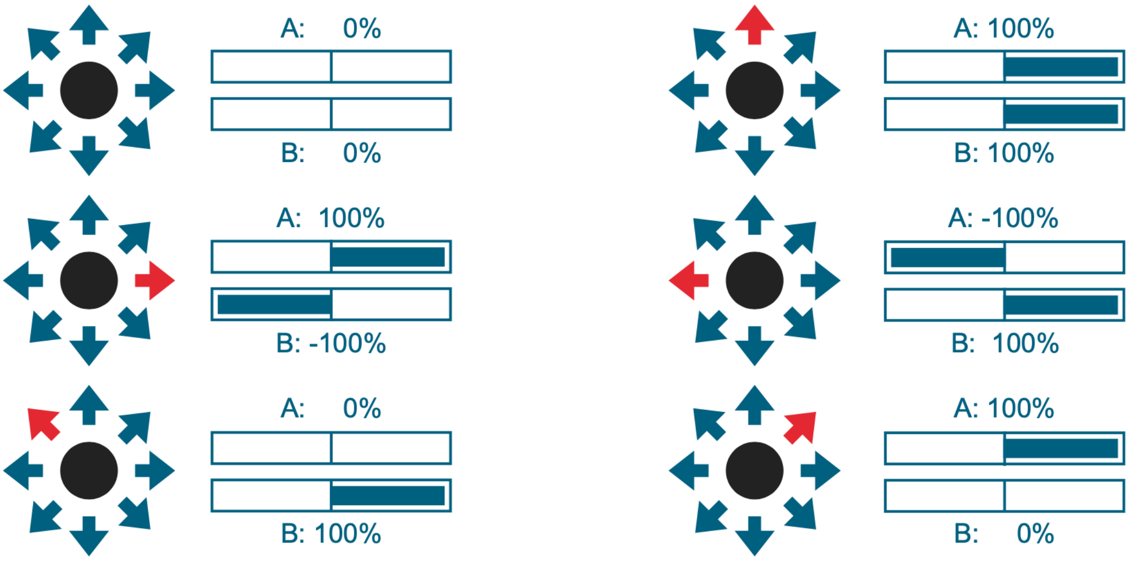
1. Zmodyfikuj kod projektu:

```c
#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 gAxisX = 0;
volatile uint16_t gAxisY = 0;
volatile bool     gAxisZ = 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);
                    gAxisY = gAdcResultInfoStruct.result;

                    ADC_GetChannelConversionResult(ADC_PERIPHERAL, 4, gAdcResultInfoPtr);
                    gAxisX = gAdcResultInfoStruct.result;

                    gAxisZ = GPIO_PinRead(BOARD_INITADCPINS_SW_GPIO,
                                        BOARD_INITADCPINS_SW_PORT,
                                        BOARD_INITADCPINS_SW_PIN);

                    /* Clear status flags */
                    ADC_ClearStatusFlags(ADC_PERIPHERAL, kADC_ConvSeqAInterruptFlag);
          }
}

void setCursor(uint8_t x, uint8_t y, uint8_t size) {

          int8_t a, b;

          a=x-size;
          b=x+size;
          if(a<0) {
                    a=0;
          }
          OLED_Draw_Line(a, y, b, y);
          a=y-size;
          b=y+size;
          if(a<0) {
                    a=0;
          }
          OLED_Draw_Line(x, a, x, b);
}

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

          uint8_t cx, cy;

          /* 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) {

                    cx = gAxisX/32;    // width: 128
                    cy = 63-gAxisY/64; // height: 64

                    OLED_Clear_Screen(0);
                    sprintf(sbuff, "X:%3d Y:%2d Z:%d", cx, cy, gAxisZ);
                    OLED_Puts(0, 0, sbuff);

                    setCursor(cx, cy, 5);
                    if(!gAxisZ) {
                              OLED_Draw_Circle(cx, cy, 8);
                    }
                    OLED_Refresh_Gram();
          }
          return 0 ;
}
```

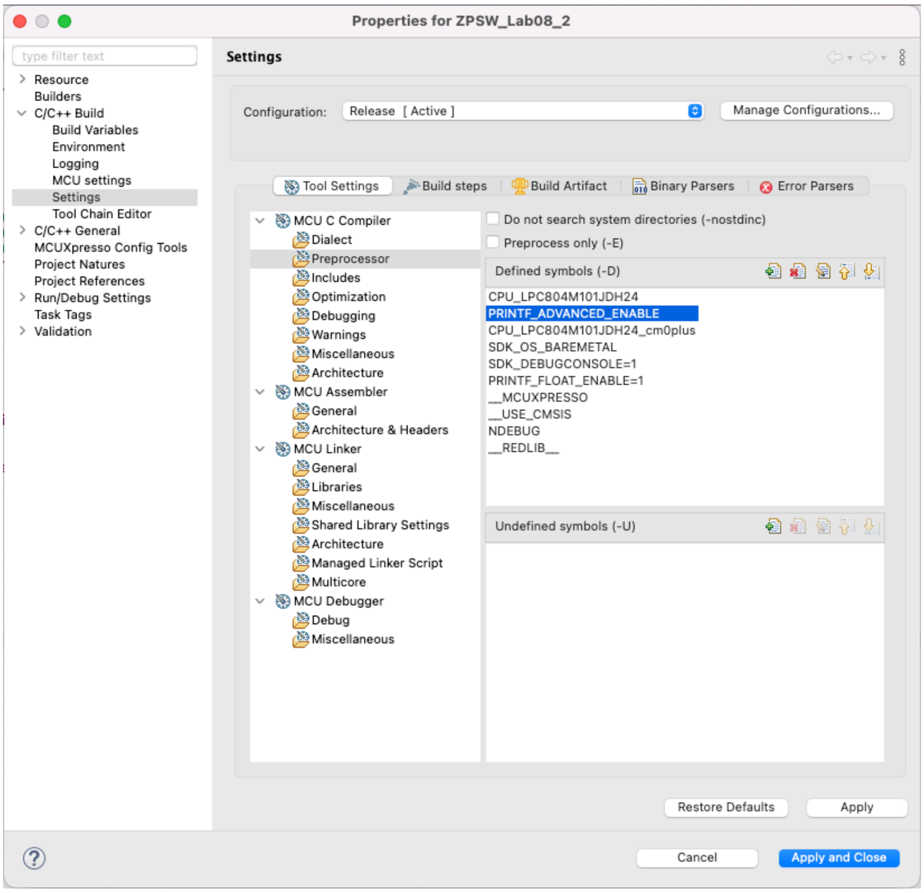
2. Zbuduj projekt w trybie *Release*, zaprogramuj układ i sprawdź działanie przykładu.

IV. Zadania

1. Napisz funkcję *PowerControl* umożliwiającą generowanie sygnałów sterujących dla 2 silników pojazdu gąsienicowego w zależności od położenia joysticka. Funkcja powinna prezentować obliczone sterowanie w postaci dwóch pasków postępu lub wskaźników wychyłowych (jak na poprzednich zajęciach) oraz wyświetlać wartości mocy w procentach. Przykładowe ustawienia joysticka:



W celu wyświetlenia ujemnych wartości, zmiennych całkowitych funkcjami *printf*, *sprint* itp. należy dodać stałą *PRINTF_ADVANCED_ENABLE* w ustawieniach preprocesora:



Properties for ZPSW_Lab08_2

Configuration: Release [Active]

□ Do not search system directories (-nostdinc)
□ Preprocess only (-E)

Defined symbols (-D)

```
CPU_LPC804M101JDH24
PRINTF_ADVANCED_ENABLE
CPU_LPC804M101JDH24_cm0plus
SDK_OS_BAREMETAL
SDK_DEBUGCONSOLE=1
PRINTF_FLOAT_ENABLE=1
__MCUXPRESSO
__USE_CMSIS
NDEBUG
__REDLIB__
```

Undefined symbols (-U)