ENGINE

Teaching online electronics, microcontrollers and programming in Higher Education

Programing of embedded systems

5. Digital Thermometer I2C

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 Where other published and unpublished source materials have been used, these have been acknowledged.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

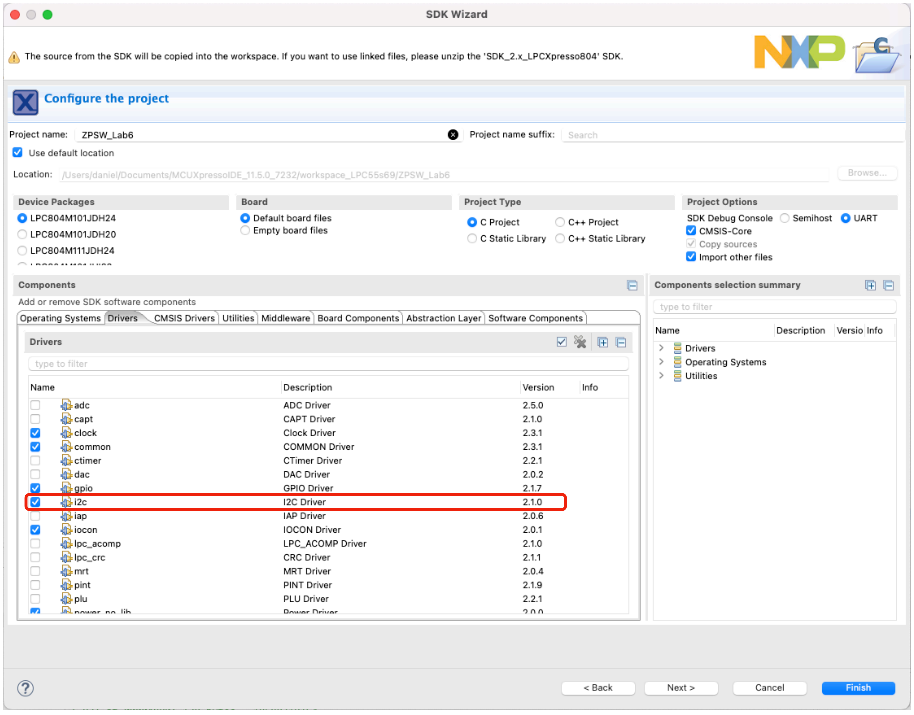
This project has been funded with support from the European Commission. This report reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Programing of embedded systems

5. Digital thermometer I2C

I. Configuration of the I2C interface

1. Create a new project for the *LPCXpresso804* board and name it eg *Lab05*. Add the *i2c* driver:



SDK Wizard

The source from the SDK will be copied into the workspace. If you want to use linked files, please unzip the 'SDK_2.x_LPCXpresso804' SDK.

Configure the project

Project name: ZPSW_Lab6 Project name suffix: Search

☑ Use default location

Location: /Users/daniel/Documents/MCUXpressoIDE_11.5.0_7232/workspace_LPC55s69/ZPSW_Lab6 Browse...

Device Packages: ● LPC804M101JDH24 ○ LPC804M101JDH20 ○ LPC804M111JDH24

Board: ● Default board files ○ Empty board files

Project Type: ● C Project ○ C++ Project ○ C Static Library ○ C++ Static Library

Project Options: SDK Debug Console ○ Semihost ● UART; ☑ CMSIS-Core; ☑ Copy sources; ☑ Import other files

Components

Add or remove SDK software components

Operating Systems | Drivers | CMSIS Drivers | Utilities | Middleware | Board Components | Abstraction Layer | Software Components

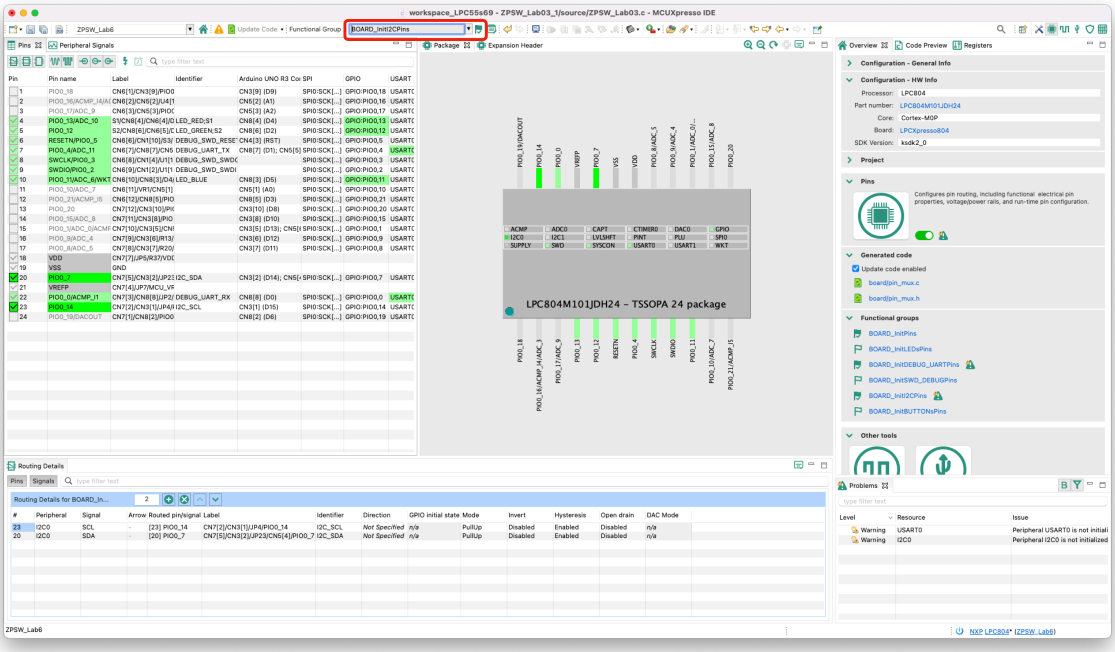
Drivers

	Name	Description	Version	Info
□	adc	ADC Driver	2.5.0	
□	capt	CAPT Driver	2.1.0	
☑	clock	Clock Driver	2.3.1	
☑	common	COMMON Driver	2.3.1	
□	ctimer	CTimer Driver	2.2.1	
□	dac	DAC Driver	2.0.2	
☑	gpio	GPIO Driver	2.1.7	
☑	i2c	I2C Driver	2.1.0	
□	iap	IAP Driver	2.0.6	
☑	iocon	IOCON Driver	2.0.1	
□	lpc_acomp	LPC_ACOMP Driver	2.1.0	
□	lpc_crc	CRC Driver	2.1.1	
□	mrt	MRT Driver	2.0.4	
□	pint	PINT Driver	2.1.9	
□	plu	PLU Driver	2.2.1	
☑	power_no_lib	Power Driver	2.0.0	

Components selection summary

Name	Description	Versio	Info
Drivers			
Operating Systems			
Utilities			

2. Go to Config Tools -> Open Pins. From the *Functional Group* menu select the *BOARD_InitI2CPins* preset, then activate it by selecting the flag icon on the left. The window now shows the automatically configured lines connected to I2C interface:



workspace_LPC55s69 - ZPSW_Lab03_1/source/ZPSW_Lab03.c - MCUXpresso IDE

ZPSW_Lab6 | Update Code | Functional Group: BOARD_InitI2CPins

Pins | Peripheral Signals

Pin	Pin name	Label	Identifier	Arduino UNO R3 Co	SPI	GPIO	USART
□ 1	PIO0_18	CN6[1]/CN3[9]/PIO0		CN3[9] (D9)	SPI0:SCK[...]	GPIO:PIO0,18	USARTC
□ 2	PIO0_16/ACMP_I4/AI	CN6[2]/CN5[2]/U4[1		CN5[2] (A1)	SPI0:SCK[...]	GPIO:PIO0,16	USARTC
□ 3	PIO0_17/ADC_9	CN6[3]/CN5[3]/PIOC		CN5[3] (A2)	SPI0:SCK[...]	GPIO:PIO0,17	USARTC
✓ 4	PIO0_13/ADC_10	S1/CN8[4]/CN6[4]/D	LED_RED;S1	CN8[4] (D4)	SPI0:SCK[...]	GPIO:PIO0,13	USARTC
✓ 5	PIO0_12	S2/CN8[6]/CN6[5]/D	LED_GREEN;S2	CN8[6] (D2)	SPI0:SCK[...]	GPIO:PIO0,12	USARTC
✓ 6	RESETN/PIO0_5	CN6[6]/CN1[10]/S3/	DEBUG_SWD_RESET	CN4[3] (RST)	SPI0:SCK[...]	GPIO:PIO0,5	USARTC
✓ 7	PIO0_4/ADC_11	CN6[7]/CN8[7]/CN5	DEBUG_UART_TX	CN8[7] (D1); CN5[5]	SPI0:SCK[...]	GPIO:PIO0,4	USARTC
✓ 8	SWCLK/PIO0_3	CN6[8]/CN1[4]/U1[1	DEBUG_SWD_SWDC		SPI0:SCK[...]	GPIO:PIO0,3	USARTC
✓ 9	SWDIO/PIO0_2	CN6[9]/CN1[2]/U1[1	DEBUG_SWD_SWDI		SPI0:SCK[...]	GPIO:PIO0,2	USARTC
✓ 10	PIO0_11/ADC_6/WKT	CN6[10]/CN8[3]/D4/	LED_BLUE	CN8[3] (D5)	SPI0:SCK[...]	GPIO:PIO0,11	USARTC
□ 11	PIO0_10/ADC_7	CN6[11]/VR1/CN5[1]		CN5[1] (A0)	SPI0:SCK[...]	GPIO:PIO0,10	USARTC
□ 12	PIO0_21/ACMP_I5	CN6[12]/CN8[5]/PIO		CN8[5] (D3)	SPI0:SCK[...]	GPIO:PIO0,21	USARTC
□ 13	PIO0_20	CN7[12]/CN3[10]/PIC		CN3[10] (D8)	SPI0:SCK[...]	GPIO:PIO0,20	USARTC
□ 14	PIO0_15/ADC_8	CN7[11]/CN3[8]/PIO		CN3[8] (D10)	SPI0:SCK[...]	GPIO:PIO0,15	USARTC
□ 15	PIO0_1/ADC_0/ACMF	CN7[10]/CN3[5]/CN!		CN3[5] (D13); CN5[(	SPI0:SCK[...]	GPIO:PIO0,1	USARTC
□ 16	PIO0_9/ADC_4	CN7[9]/CN3[6]/R13/		CN3[6] (D12)	SPI0:SCK[...]	GPIO:PIO0,9	USARTC
□ 17	PIO0_8/ADC_5	CN7[8]/CN3[7]/R20/		CN3[7] (D11)	SPI0:SCK[...]	GPIO:PIO0,8	USARTC
✓ 18	VDD	CN7[7]/JP5/R37/VDD					
✓ 19	VSS	GND					
☑ 20	PIO0_7	CN7[5]/CN3[2]/JP23	I2C_SDA	CN3[2] (D14); CN5[4	SPI0:SCK[...]	GPIO:PIO0,7	USARTC
✓ 21	VREFP	CN7[4]/JP7/MCU_VR					
✓ 22	PIO0_0/ACMP_I1	CN7[3]/CN8[8]/JP2/	DEBUG_UART_RX	CN8[8] (D0)	SPI0:SCK[...]	GPIO:PIO0,0	USARTC
☑ 23	PIO0_14	CN7[2]/CN3[1]/JP4/F	I2C_SCL	CN3[1] (D15)	SPI0:SCK[...]	GPIO:PIO0,14	USARTC
□ 24	PIO0_19/DACOUT	CN7[1]/CN8[2]/PIO0		CN8[2] (D6)	SPI0:SCK[...]	GPIO:PIO0,19	USARTC

Overview | Code Preview | Registers

Configuration - General Info

Configuration - HW Info

Processor: LPC804

Part number: LPC804M101JDH24

Core: Cortex-M0P

Board: LPCXpresso804

SDK Version: ksdk2_0

Project

Pins

Configures pin routing, including functional electrical pin properties, voltage/power rails, and run-time pin configuration.

Generated code

☑ Update code enabled

board/pin_mux.c

board/pin_mux.h

Functional groups

BOARD_InitPins

BOARD_InitLEDsPins

BOARD_InitDEBUG_UARTPins

BOARD_InitSWD_DEBUGPins

BOARD_InitI2CPins

BOARD_InitBUTTONsPins

Other tools

Routing Details

Pins | Signals

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
23	I2C0	SCL	-	[23] PIO0_14	CN7[2]/CN3[1]/JP4/PIO0_14	I2C_SCL	*Not Specified*	*n/a*	PullUp	Disabled	Enabled	Disabled	*n/a*
20	I2C0	SDA	-	[20] PIO0_7	CN7[5]/CN3[2]/JP23/CN5[4]/PIO0_7	I2C_SDA	*Not Specified*	*n/a*	PullUp	Disabled	Enabled	Disabled	*n/a*

Problems

Level	Resource	Issue
Warning	USART0	Peripheral USART0 is not initiali
Warning	I2C0	Peripheral I2C0 is not initialized

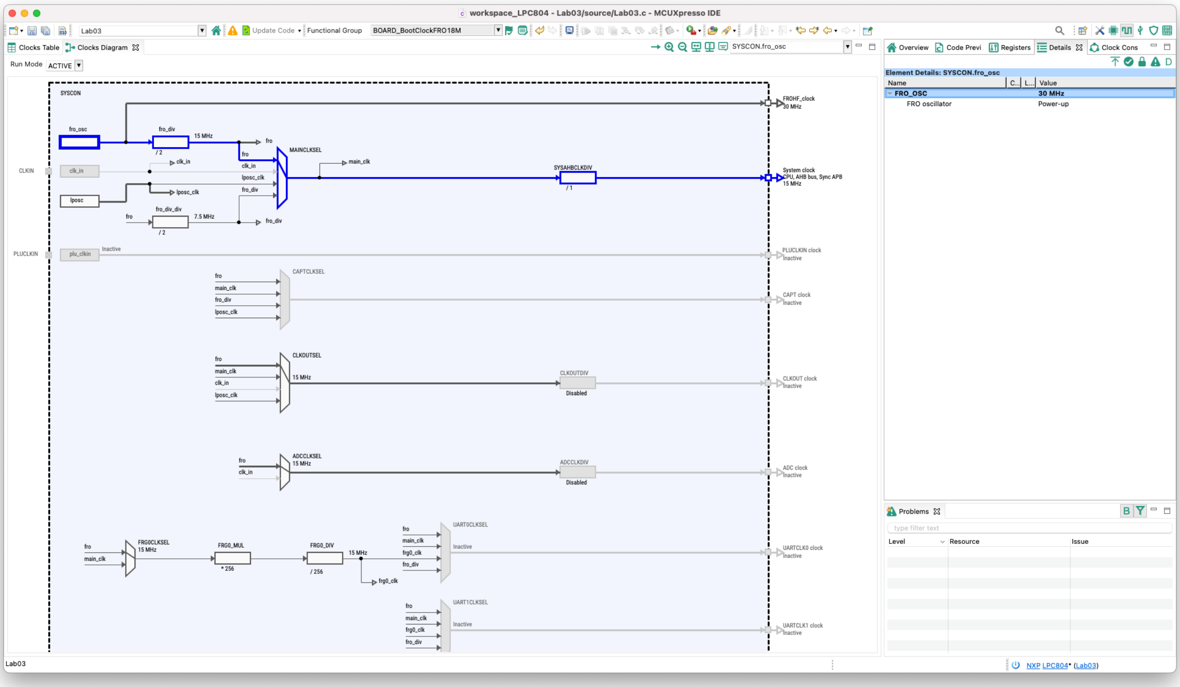
ZPSW_Lab6

NXP LPC804* (ZPSW_Lab6)

Programing of embedded systems

5. Digital thermometer I2C

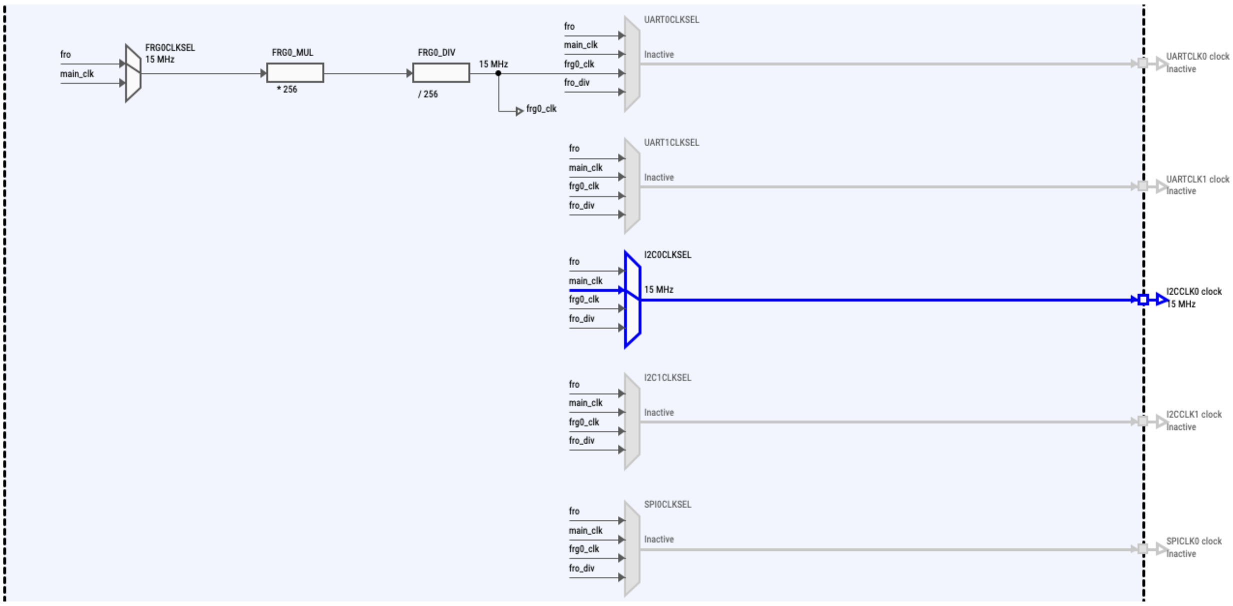
3. Go to the *Clocks* tab and then double-click on the *FRO_OSC* block and change the *FRO_OSC* to 30 MHz clock:



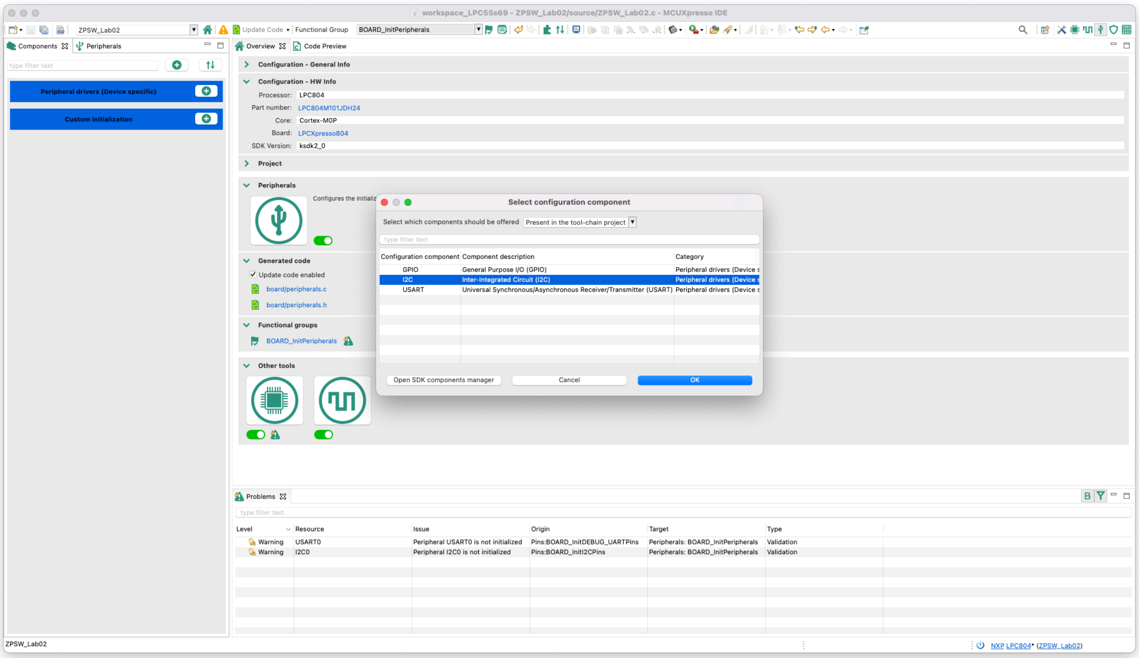
Element Details: SYSCON.fro_osc

Name	C...	L...	Value
FRO_OSC			30 MHz
FRO oscillator			Power-up

4. Next double-click on the *I2C0CLKSEL* block and set the *main_clk* (15 MHz) clock:



5. Go to the Peripherals tab and then click on *Peripheral* drivers and select *I2C* from the list:



Processor: LPC804
Part number: LPC804M101JDH24
Core: Cortex-M0P
Board: LPCXpresso804
SDK Version: ksdk2_0

☑ Update code enabled
board/peripherals.c
board/peripherals.h

Select configuration component

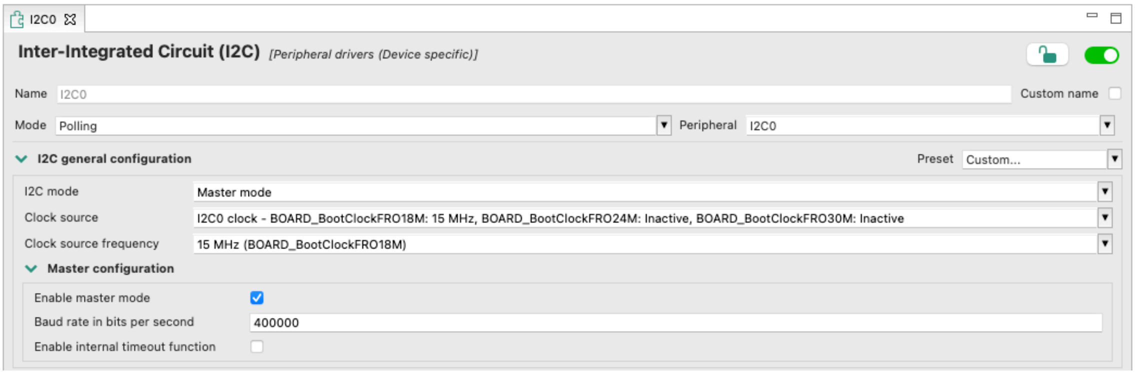
Select which components should be offered: Present in the tool-chain project

Configuration component	Component description	Category
GPIO	General Purpose I/O (GPIO)	Peripheral drivers (Device s
I2C	Inter-Integrated Circuit (I2C)	Peripheral drivers (Device s
USART	Universal Synchronous/Asynchronous Receiver/Transmitter (USART)	Peripheral drivers (Device s

Open SDK components manager | Cancel | OK

Level	Resource	Issue	Origin	Target	Type
Warning	USART0	Peripheral USART0 is not initialized	Pins:BOARD_InitDEBUG_UARTPins	Peripherals: BOARD_InitPeripherals	Validation
Warning	I2C0	Peripheral I2C0 is not initialized	Pins:BOARD_InitI2CPins	Peripherals: BOARD_InitPeripherals	Validation

6. Select the *I2C0* interface and change the default baud rate to 400 000 bps:



Inter-Integrated Circuit (I2C) *[Peripheral drivers (Device specific)]*

Name: I2C0 — Custom name □

Mode: Polling — Peripheral: I2C0

I2C general configuration — Preset: Custom...

I2C mode: Master mode
Clock source: I2C0 clock - BOARD_BootClockFRO18M: 15 MHz, BOARD_BootClockFRO24M: Inactive, BOARD_BootClockFRO30M: Inactive
Clock source frequency: 15 MHz (BOARD_BootClockFRO18M)

Master configuration

Enable master mode: ☑
Baud rate in bits per second: 400000
Enable internal timeout function: □

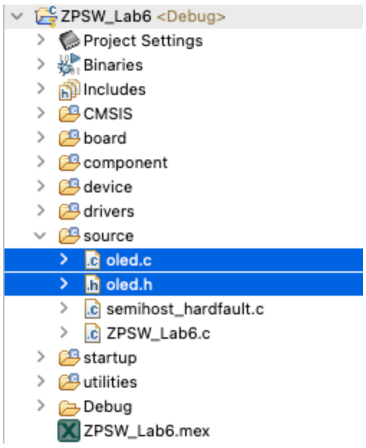
Then click *Update Code* to generate the I2C configuration code.

Programing of embedded systems

5. Digital thermometer I2C

II. Graphic display

1. Add the OLED display library to the project (drag files to workspace):



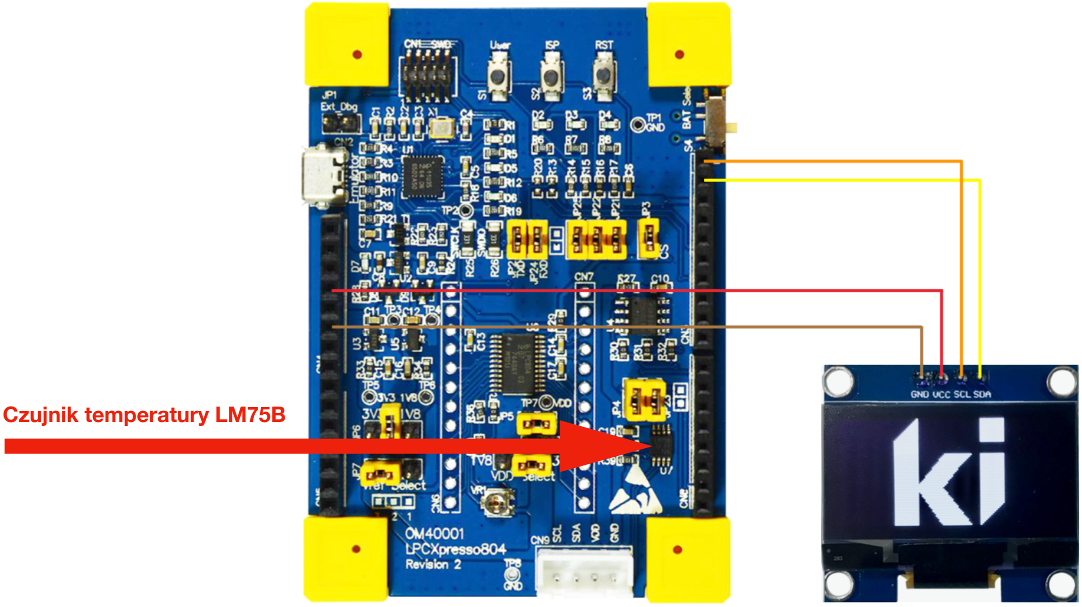
```
ZPSW_Lab6 <Debug>
  Project Settings
  Binaries
  Includes
  CMSIS
  board
  component
  device
  drivers
  source
    oled.c
    oled.h
    semihost_hardfault.c
    ZPSW_Lab6.c
  startup
  utilities
  Debug
  ZPSW_Lab6.mex
```

2. 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"
/*
 * @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);
          OLED_Draw_Bitmap(LogoKI);
          OLED_Refresh_Gram();

          while(1) {
          }
          return 0 ;
}
```

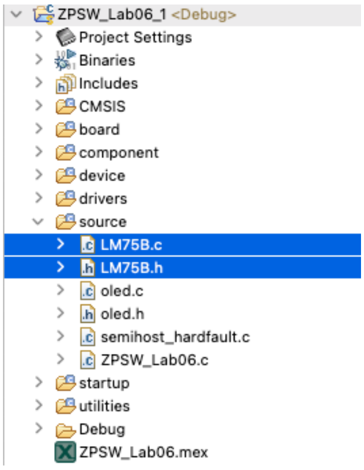
3. Connect the display to the prototype board according to the diagram below:



4. Build a project, program the microcontroller and check the operation.

III. LM75B library

1. Create thermometer library files. To do this, right-click on the source folder in workspace and then select *New-> Header File* and name it *LM75B.h*.
2. Similarly, by right-clicking on the source folder in workspace, select *New-> Source File* and name it *LM75B.c*:



```
ZPSW_Lab06_1 <Debug>
  Project Settings
  Binaries
  Includes
  CMSIS
  board
  component
  device
  drivers
  source
    LM75B.c
    LM75B.h
    oled.c
    oled.h
    semihost_hardfault.c
    ZPSW_Lab06.c
  startup
  utilities
  Debug
  ZPSW_Lab06.mex
```

3. Go to the *LM75.h* file and add the code:

```
#ifndef LM75B_H_
#define LM75B_H_

#include "fsl_i2c.h"

#define LM75_REG_TEMP (0x00) // Temperature Register
#define LM75_REG_CONF (0x01) // Configuration Register
#define LM75_ADDR     (0x48) // LM75 address

void  LM75B_Init(I2C_Type *base);
float LM75B_Read();

#endif /* LM75B_H_ */
```

4. Go to the *LM75.c* file and add the code:

```
#include "LM75B.h"

static I2C_Type *I2C_base=NULL;

void LM75B_Init(I2C_Type *base) {

        I2C_base=base;

        char data_write[2];

        data_write[0] = LM75_REG_CONF;
        data_write[1] = 0x02;

        if (kStatus_Success == I2C_MasterStart(I2C_base, LM75_ADDR, kI2C_Write)) {

                I2C_MasterWriteBlocking(I2C_base, &data_write[0], 2, kI2C_TransferDefaultFlag);
                I2C_MasterStop(I2C_base);
        }
}

float LM75B_Read() {

        char data_read[2];
        char data_write[1];
        float temp;
        int16_t v;

        data_write[0] = LM75_REG_TEMP;

        if (kStatus_Success == I2C_MasterStart(I2C_base, LM75_ADDR, kI2C_Write)) {

                I2C_MasterWriteBlocking(I2C_base, &data_write[0], 1, kI2C_TransferNoStopFlag);
```

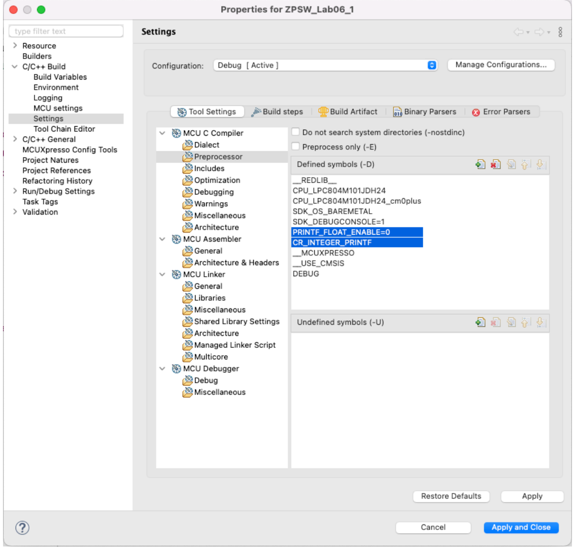
```
                    I2C_MasterRepeatedStart(I2C_base, LM75_ADDR, kI2C_Read);
                    I2C_MasterReadBlocking(I2C_base, &data_read[0], 2, kI2C_TransferDefaultFlag);

                    I2C_MasterStop(I2C_base);
          }

          v= (data_read[0] << 8) | data_read[1];
          temp = v / 256.0; // temperature value in Celsius

          return temp;
}
```

5. Go to project settings. Right-click on the project name, select *Properties* and then *Settings -> Preprocessor*. Change the *PRINTF_FLOAT_ENABLE* flag to 1 and remove the *CR_INTEGER_PRINTF* flag:



Configuration: Debug [Active]

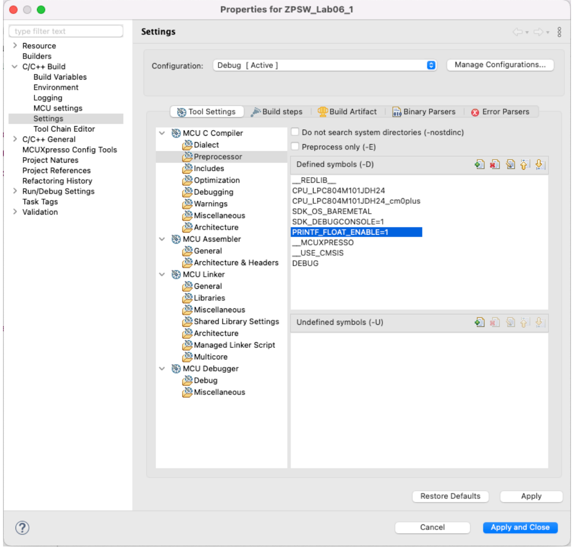
□ Do not search system directories (-nostdinc)

□ Preprocess only (-E)

Defined symbols (-D)

```
__REDLIB__
CPU_LPC804M101JDH24
CPU_LPC804M101JDH24_cm0plus
SDK_OS_BAREMETAL
SDK_DEBUGCONSOLE=1
PRINTF_FLOAT_ENABLE=0
CR_INTEGER_PRINTF
__MCUXPRESSO
__USE_CMSIS
DEBUG
```

Undefined symbols (-U)



Configuration: Debug [Active]

□ Do not search system directories (-nostdinc)

□ Preprocess only (-E)

Defined symbols (-D)

```
__REDLIB__
CPU_LPC804M101JDH24
CPU_LPC804M101JDH24_cm0plus
SDK_OS_BAREMETAL
SDK_DEBUGCONSOLE=1
PRINTF_FLOAT_ENABLE=1
__MCUXPRESSO
__USE_CMSIS
DEBUG
```

Undefined symbols (-U)

6. Go to the main program file and modify the 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 "oled.h"
#include "LM75B.h"

char sbuff[32];
float temp;

/*
 * @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);
          OLED_Draw_Bitmap(LogoKI);
          OLED_Refresh_Gram();

          /* Initialize LM75 */
          LM75B_Init(I2C0_PERIPHERAL);

          while(1) {

                    OLED_Clear_Screen(0);

                    temp = LM75B_Read();

                    sprintf(sbuff, "t: %.3f C", temp);
                    OLED_Puts(0,0, sbuff);

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

7. Build the project, program the chip and check the temperature reading (you can gently touch the LM75B chip to change the temperature).

IV. Simple GUI

1. Add a 7-segment display simulation for temperature display:

```
#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"
#include "LM75B.h"

char sbuff[32];
float temp;

/*
 * @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);

          /* Initialize LM75 */
          LM75B_Init(I2C0_PERIPHERAL);

          while(1) {

                    temp = LM75B_Read();

                    OLED_Clear_Screen(0);

                    OLED_7segf(0, 4, temp, 4, 1, 1);
                    OLED_Puts(105, 1, "C");

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

2. Build the project, program the chip and check the temperature reading.
3. Add a bargraph:

```
#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"
#include "LM75B.h"

char sbuff[32];
float temp;

#define T_MIN       0
#define T_MAX       40


void Bargraph(uint8_t x, uint8_t y, uint8_t w, uint8_t h, float min, float max, float v) {

          if(v<min) {
                    v=min;
          }
          if(v>max) {
                    v=max;
          }
          v = ((v-min)*w)/(max-min);

          OLED_Draw_Rect(x , y, x+w-1, y+h-1, 1);
          OLED_Draw_Fill_Rect(x+2, y+2, x+v-3 , y+h-3, 1);
}

/*
 * @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);

          /* Initialize LM75 */
          LM75B_Init(I2C0_PERIPHERAL);

          while(1) {

                    temp = LM75B_Read();

                    OLED_Clear_Screen(0);

                    OLED_7segf(0, 4, temp, 4, 1, 1);
                    OLED_Puts(105, 1, "C");

                    Bargraph(0, 45, 128, 8, T_MIN, T_MAX, temp);

                    sprintf(sbuff, "%d", T_MIN);
                    OLED_Puts(0, 7, sbuff);
                    sprintf(sbuff, "%3d", T_MAX);
                    OLED_Puts(110, 7, sbuff);

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

4. Build the project, program the chip and check the temperature reading.

V. Exercises

1. Check barograph indications for different ranges *T_MIN* and *T_MAX*..
2. Implement a moving average filter a certain number of measurements given by the equation:

$$y(n) = \frac{1}{N}\sum_{k=0}^{N-1} x(n-k) \text{ dla } n = 0, 1, 2, 3, \ldots$$

in *FilterAVG* function:

```
#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"
#include "LM75B.h"

char sbuff[32];
float temp;

#define T_MIN       0
#define T_MAX       40
#define N 16

float FilterAVG(float x) {


}
```

```c
void Bargraph(uint8_t x, uint8_t y, uint8_t w, uint8_t h, float min, float max, float v) {

        if(v<min) {
                v=min;
        }
        if(v>max) {
                v=max;
        }
        v = ((v-min)*w)/(max-min);

        OLED_Draw_Rect(x , y, x+w-1, y+h-1, 1);
        OLED_Draw_Fill_Rect(x+2, y+2, x+v-3 , y+h-3, 1);
}

/*
 * @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);

        /* Initialize LM75 */
        LM75B_Init(I2C0_PERIPHERAL);

        while(1) {

                temp = LM75B_Read();
                temp = FilterAVG(temp);

                OLED_Clear_Screen(0);

                OLED_7segf(0, 4, temp, 4, 1, 1);
                OLED_Puts(105, 1, "C");

                Bargraph(0, 45, 128, 8, T_MIN, T_MAX, temp);

                sprintf(sbuff, "%d", T_MIN);
                OLED_Puts(0, 7, sbuff);
                sprintf(sbuff, "%3d", T_MAX);
                OLED_Puts(110, 7, sbuff);

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