ENGINE

Teaching online electronics, microcontrollers and programming in Higher Education

Programing of embedded systems

2. Debbuger console and GPIO

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 Where other published and unpublished source materials have been used, these have been acknowledged.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

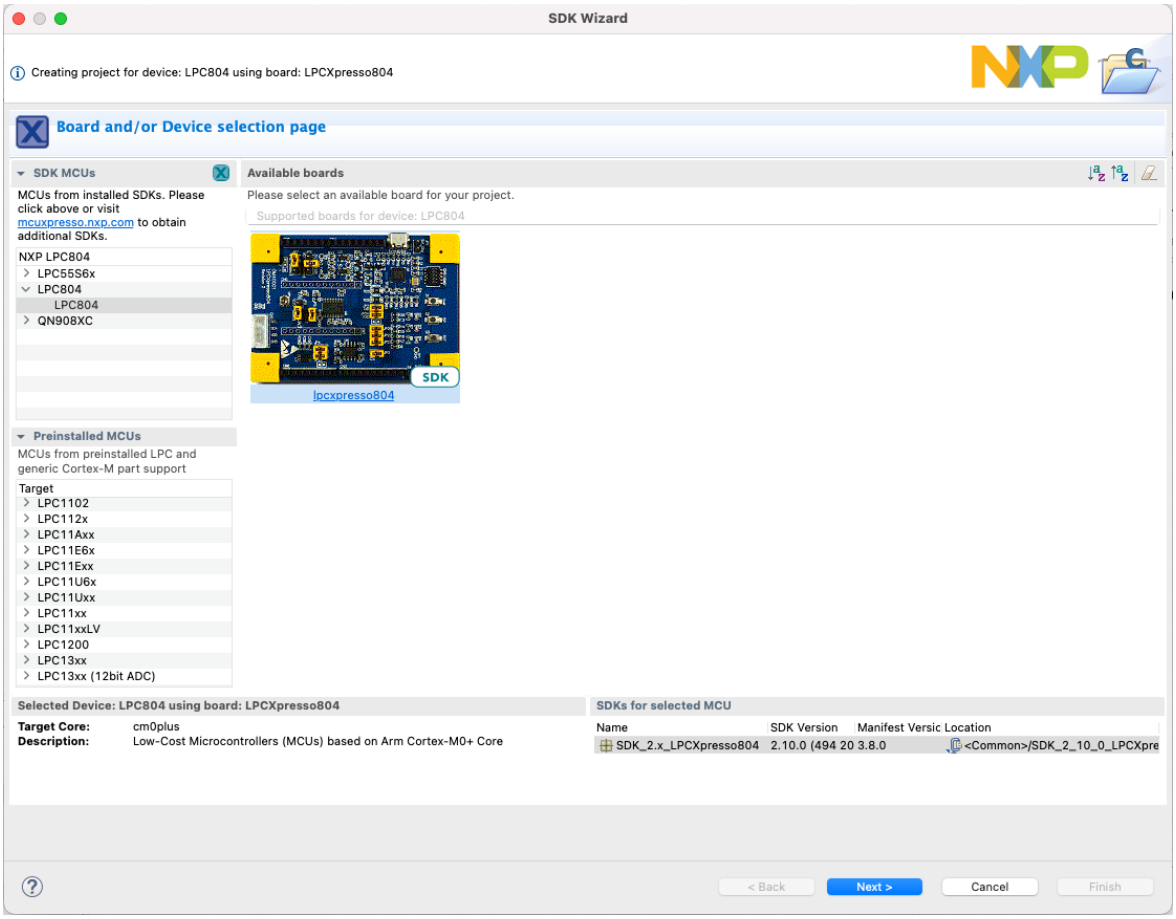
Funding Disclaimer

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.

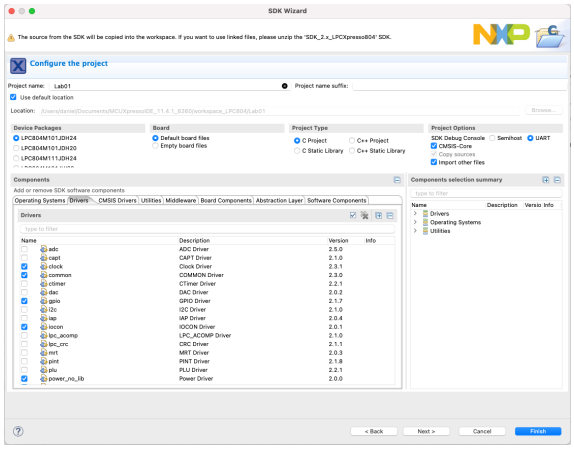
Programing of embedded systems
2. Debbuger console and GPIO

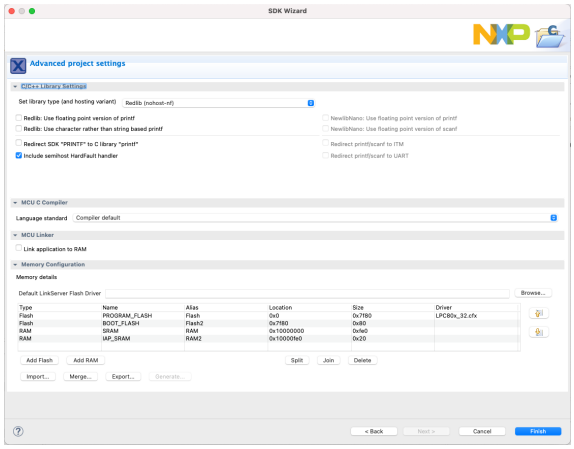
I. New project and debugger console

1. Create a new project for the *LPCXpresso804* board:



2. Name the project eg *Lab01* and keep the default configuration:



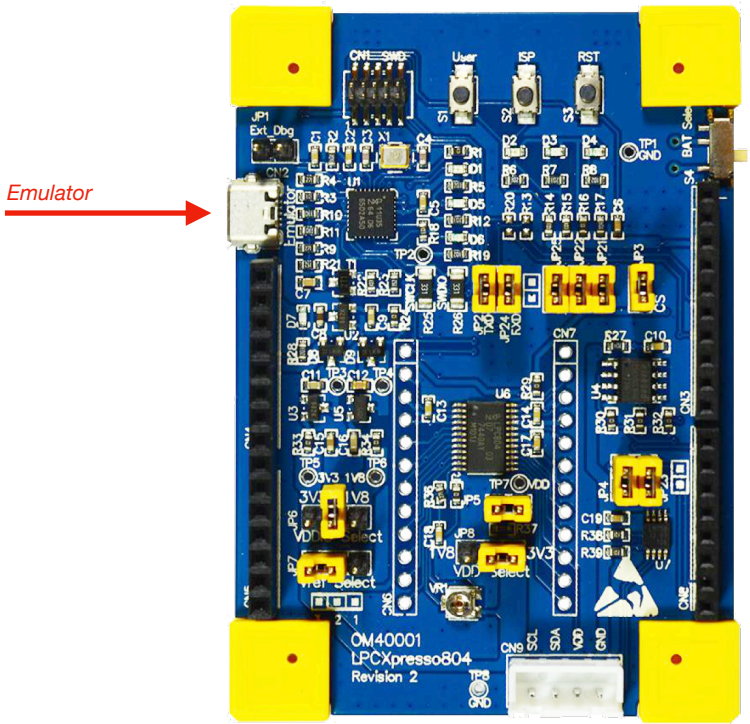


3. A code skeleton will be generated:

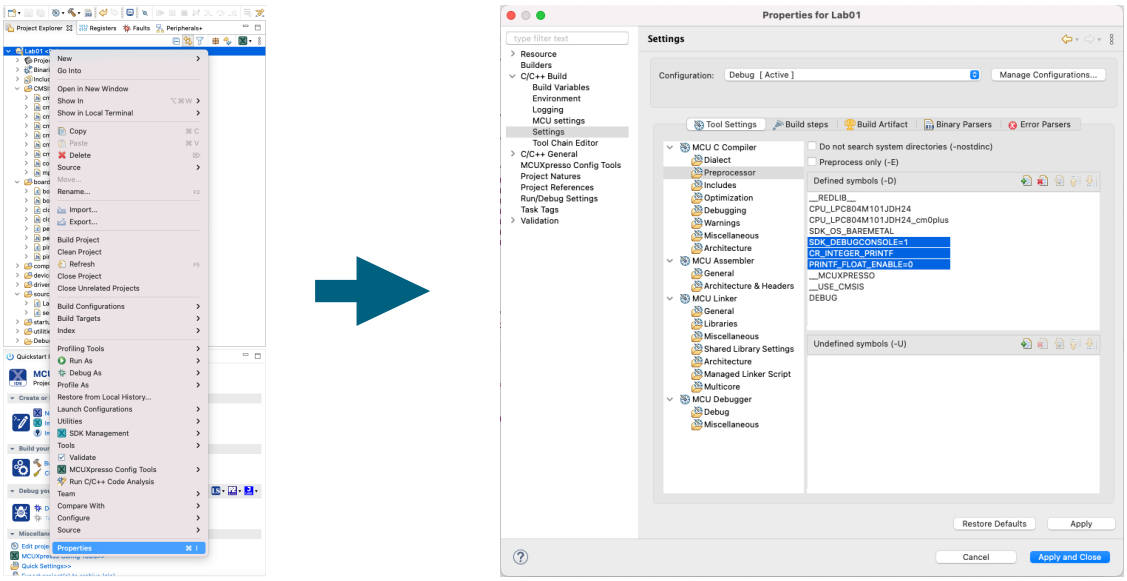
```
/**
 * @file    Lab01.c
 * @brief   Application entry point.
 */
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "LPC804.h"
#include "fsl_debug_console.h"
/* TODO: insert other include files here. */

/* TODO: insert other definitions and declarations here. */

/*
 * @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

    PRINTF("Hello World\n");

    /* Force the counter to be placed into memory. */
    volatile static int i = 0 ;
    /* Enter an infinite loop, just incrementing a counter. */
    while(1) {
        i++ ;
        /* 'Dummy' NOP to allow source level single stepping of
            tight while() loop */
        __asm volatile ("nop");
    }
    return 0 ;
}
```

Add a "\ r" tag to the end of the text in the PRINTF function.

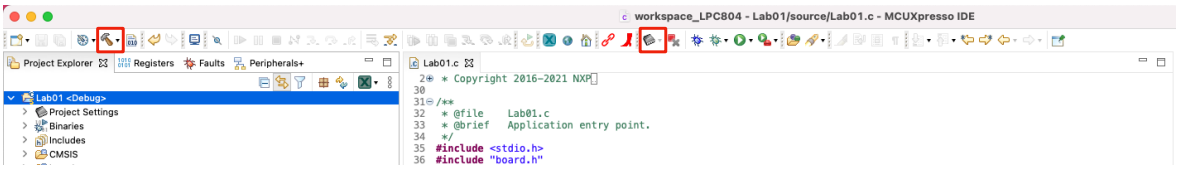
4. Connect the *LPCXpresso804* board with the USB interface labeled *Emulator* to the computer:



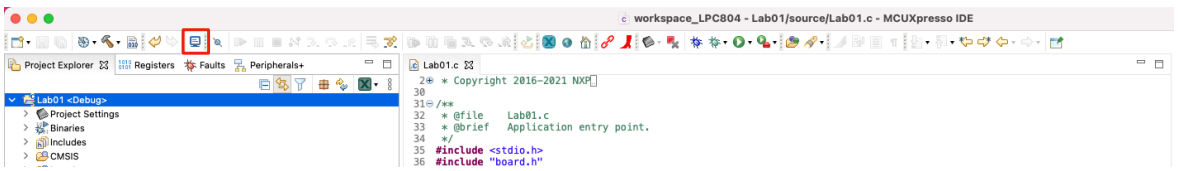
5. Right click on the project name, select Properties and then go to C/C++ Build -> Settings and select Preprocessor. Check the value of the SDK_DEBUGCONSOLE constant, which should be set to 1 (otherwise set to 1), as in the picture below. The constant activates the debugger console that uses UART. The other two constants disable real number support by default (a limited version of the library that takes up less memory is used). To display real types in the console, remove the CR_INTEGER_PRINTF constant and change the value of the PRINTF_FLOAT_ENABLE constant to 1.



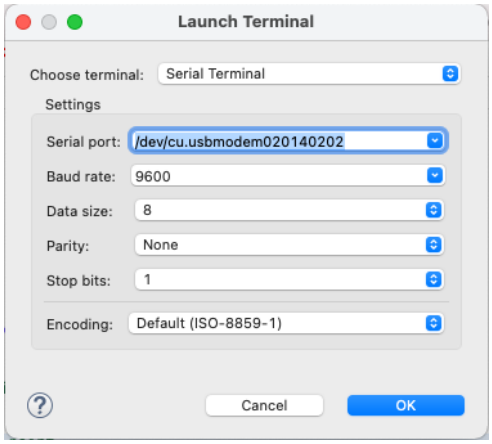
6. Build the project by clicking Build and then program the layout by clicking GUI Flash Tool, leaving the default settings in the following programmer windows:



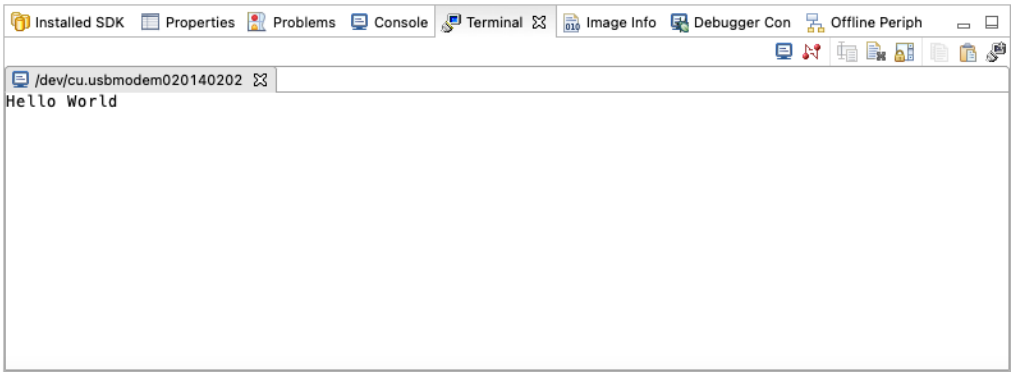
7. Launch the terminal:



8. In the settings, select *Serial Terminal* and the baud rate 9600. In the serial port field, select */dev/cu.usbmodem020140202* from the list of available ports. In Windows, select *COMMx* port. Note, depending on the version of the evaluation board, in the place of "X" there may appear no other code than in the picture below:



9. Press the RESET button on the *LPCXpresso804* board. The text sent with the *PRINTF* function should appear in the terminal window:



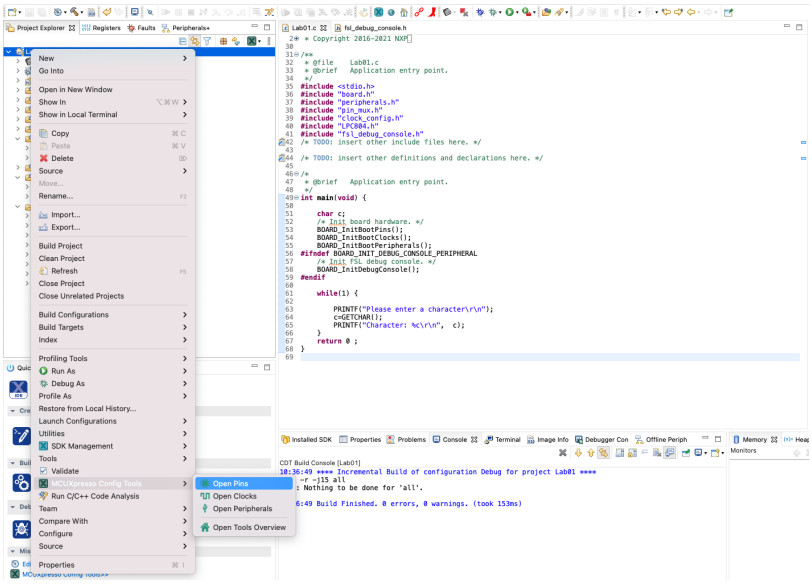
10. Write a simple "echo" program that prints the received characters to the console, preceded by the string "Character:". To do this, modify the code in the main function as below:

```
int main(void) {

          char c;
          /* Init board hardware. */
          BOARD_InitBootPins();
          BOARD_InitBootClocks();
          BOARD_InitBootPeripherals();
#ifndef BOARD_INIT_DEBUG_CONSOLE_PERIPHERAL
          /* Init FSL debug console. */
          BOARD_InitDebugConsole();
#endif

          while(1) {

                    PRINTF("Please enter a character\r\n");
                    c=GETCHAR();
                    PRINTF("Character: %c\r\n",  c);
          }
          return 0 ;
}
```

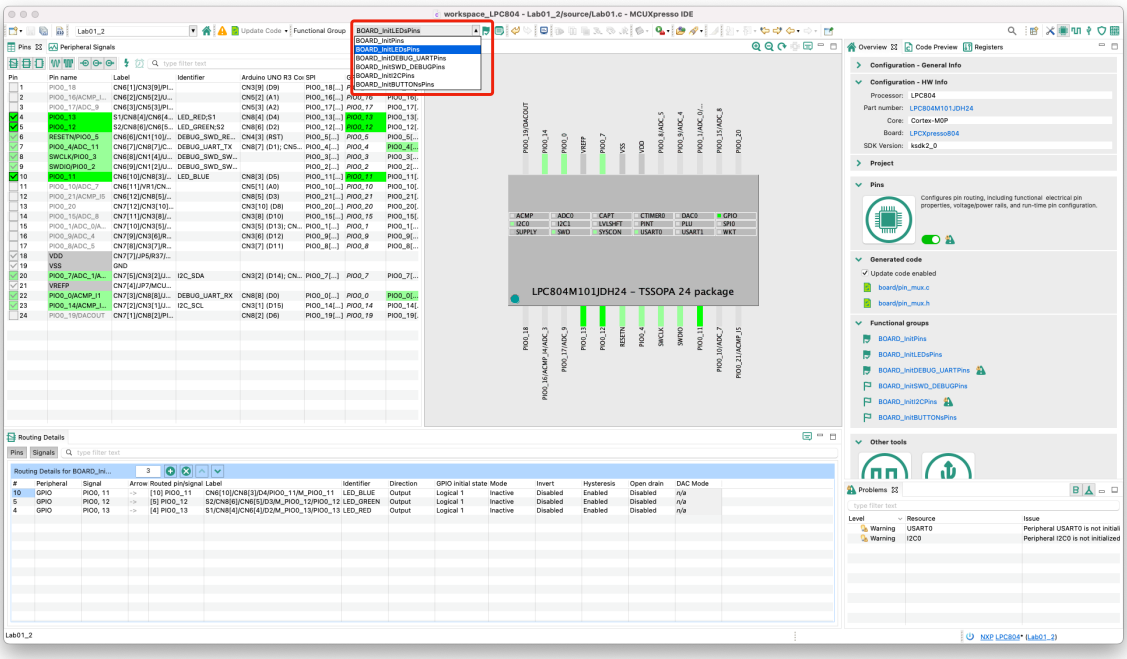
Build a project, program the microcontroller and check the program operation in the terminal console.

II. Controlling of LEDs

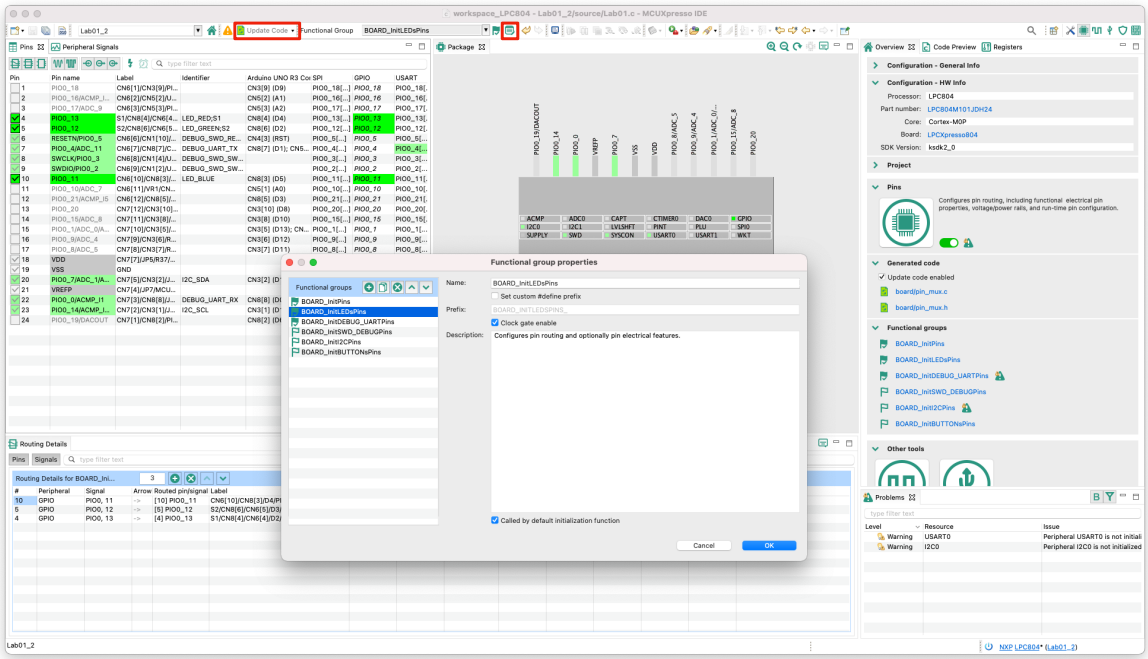
1. Create a new project for the *LPCXpresso804* board and and name it *Lab01_1*.
2. You must configure 3 *GPIO* lines to control the RGB LEDs. To do this, right-click on the project name and select *MCUXpresso Config Tools -> Open Pins*, as in the picture below:



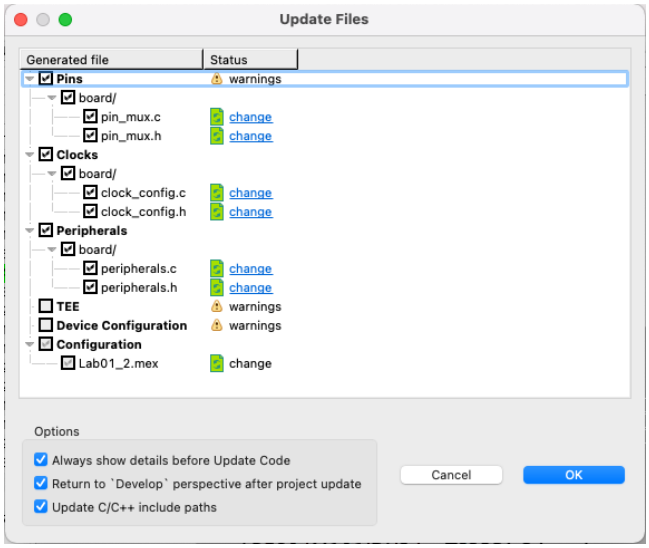
3. From the *Functional Group* menu select the *BOARD_InitLEDsPins* preset, then activate it by selecting the flag icon on the left. The window now shows the automatically configured lines connected to RGB LEDs on the prototype board:



4. Active presets can be checked by clicking the *Functional group* properties icon. The list of presets is displayed in the opened window. You can edit them and add your own:

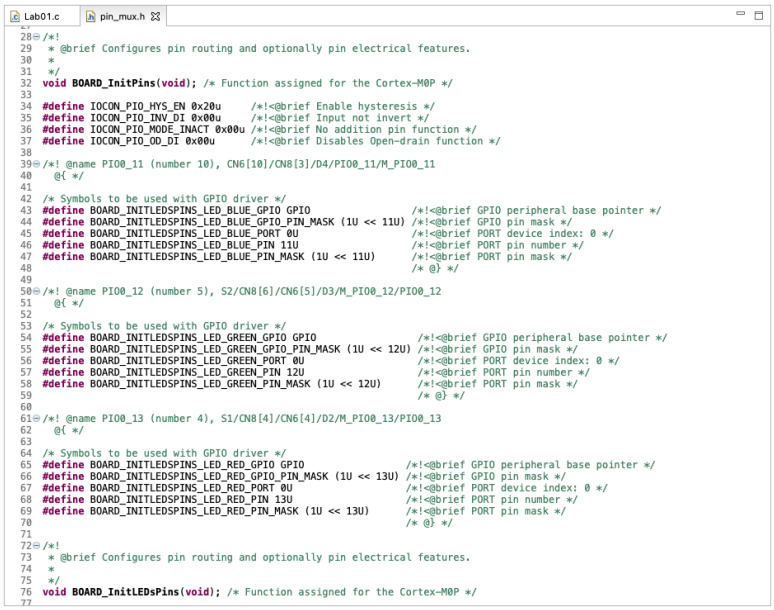


5. Select *Update Code* (the picture above) to generate the code based on the entered configuration. The code will be added to files marked with the "change" icon:



By clicking on change you can see what changes will be made to individual files with the source code.

6. Accept the changes with the OK button.
7. The constants describing the individual lines controlling the RGB LEDs were generated in the *pin_mux.h* file:



8. Modify the code in the main function so that sending "a" causes the LED to light red. In turn, sending the "z" character should extinguish the red component:

```
#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"

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

          char c;
          /* Init board hardware. */
          BOARD_InitBootPins();
          BOARD_InitBootClocks();
          BOARD_InitBootPeripherals();
#ifndef BOARD_INIT_DEBUG_CONSOLE_PERIPHERAL
          /* Init FSL debug console. */
          BOARD_InitDebugConsole();
#endif

          PRINTF("Start\r\n");

          while(1) {

             c=GETCHAR();

             if(c=='a') {
                  // On
                  GPIO_PinWrite(BOARD_INITLEDSPINS_LED_RED_GPIO, BOARD_INITLEDSPINS_LED_RED_PORT,
BOARD_INITLEDSPINS_LED_RED_PIN, 0);
             }

             if(c=='z') {
                 // Off
                 GPIO_PinWrite(BOARD_INITLEDSPINS_LED_RED_GPIO, BOARD_INITLEDSPINS_LED_RED_PORT,
BOARD_INITLEDSPINS_LED_RED_PIN, 1);
             }
          }
          return 0 ;
}
```

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

III. Button operation

1. Create a new project for the *LPCXpresso804* board and and name it *Lab01_3*.
2. Open the pin configuration tool again: *MCUXpresso Config Tools -> Open Pins*. Add the *BOARD_InitBUTTONsPins* preset and then click *Update Code*.
3. Modify the code in the main function so that pressing the S1 and S2 buttons displays the appropriate messages:

```
#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"
/* TODO: insert other include files here. */

/* TODO: insert other definitions and declarations here. */

/*
 * @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

          PRINTF("Start\r\n");

          while(1) {

                if(GPIO_PinRead(BOARD_INITBUTTONSPINS_S1_GPIO, BOARD_INITBUTTONSPINS_S1_PORT,
BOARD_INITBUTTONSPINS_S1_PIN) == 0)
                   PRINTF("S1\r\n");

                if(GPIO_PinRead(BOARD_INITBUTTONSPINS_S2_GPIO, BOARD_INITBUTTONSPINS_S2_PORT,
BOARD_INITBUTTONSPINS_S2_PIN) == 0)
                   PRINTF("S2\r\n");
          }
          return 0 ;
}
```

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

4. Modify the program so that it reacts only to a single button press (falling edge detection):

```
#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"
/* TODO: insert other include files here. */

/* TODO: insert other definitions and declarations here. */

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

          bool sw1=false, tm1=false;
          bool sw2=false, tm2=false;

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

          PRINTF("Start\r\n");

          while(1) {
```

```
                    tm1=sw1;
                    sw1 = GPIO_PinRead(BOARD_INITBUTTONSPINS_S1_GPIO, BOARD_INITBUTTONSPINS_S1_PORT,
BOARD_INITBUTTONSPINS_S1_PIN);


                    tm2=sw2;
                    sw2 = GPIO_PinRead(BOARD_INITBUTTONSPINS_S2_GPIO, BOARD_INITBUTTONSPINS_S2_PORT,
BOARD_INITBUTTONSPINS_S2_PIN);

                    if(sw1 < tm1) {

                       PRINTF("S1\r\n");
                    }

                    if(sw2 < tm2) {

                       PRINTF("S2\r\n");
                    }
          }
          return 0 ;
}
```

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

IV. Exercises

1. Modify the LED control program so that it is possible to control three RGB diodes. Send a mark:
   - *a: Red-On*
   - *z: Red-Off*
   - *s: Green-On*
   - *x: Green-Off*
   - *d: Blue-On*
   - *c: Blue-Off*
2. Write the same program using the switch-case statement.
3. Modify the button program so that it is possible to detect when a button is released.