

ENGINE

Teaching online electronics, microcontrollers and programming in Higher Education

Programing of embedded systems

2. Konsola debuggera i GPIO

Lead Partner: Warsaw University of Technology

Authors: Daniel Król

University of Applied Sciences in Tarnow

Programing of embedded systems

2. Konsola debuggera i GPIO

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 Where other published and unpublished source materials have been used, these have been acknowledged.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

© Copyright 2021 - 2023 the ENGINE Consortium

Warsaw University of Technology (Poland)

International Hellenic University (IHU) (Greece)

European Lab for Educational Technology- EDUMOTIVA (Greece)

University of Padova (Italy)

University of Applied Sciences in Tarnow (Poland)

All rights reserved.



This document is licensed to the public under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

Funding Disclaimer

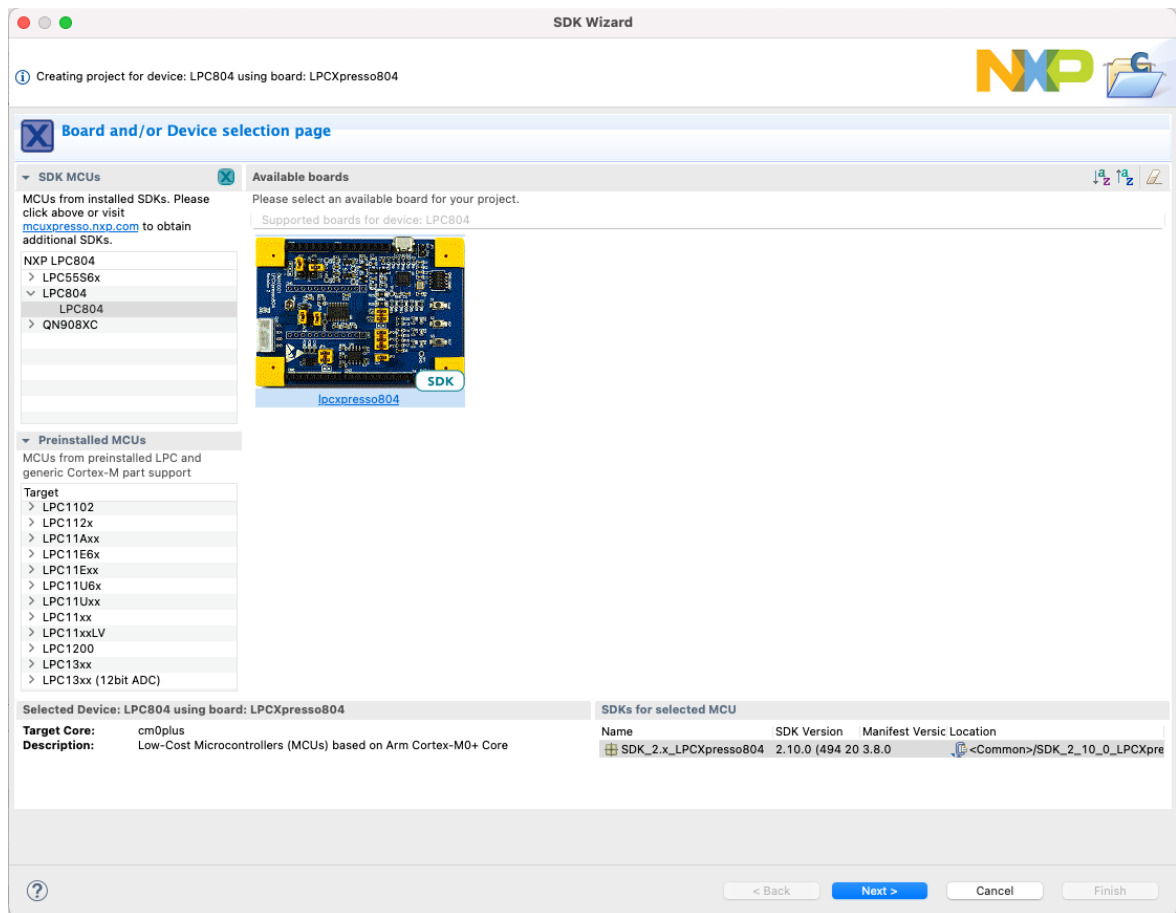
This project has been funded with support from the European Commission. This report reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Programming of embedded systems

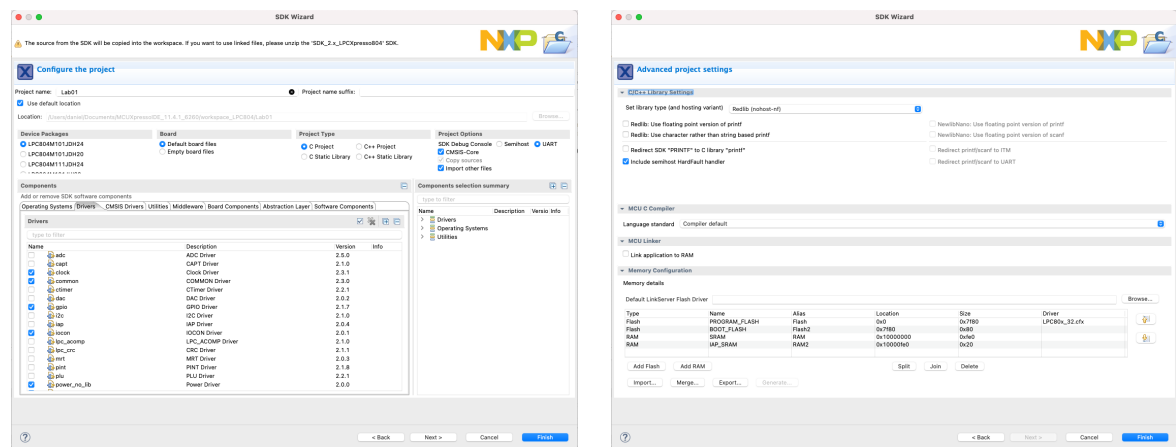
2. Konsola debuggera i GPIO

I. Nowy projekt i konsola debuggera

1. Stwórz nowy projekt dla płyty LPCXpresso804:



2. Nazwij projekt np. *Lab01* i pozostaw domyślną konfigurację:



Programing of embedded systems

2. Konsola debugera i GPIO

3. Zostanie wygenerowany szkielet kodu:

```
/**
 * @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. */

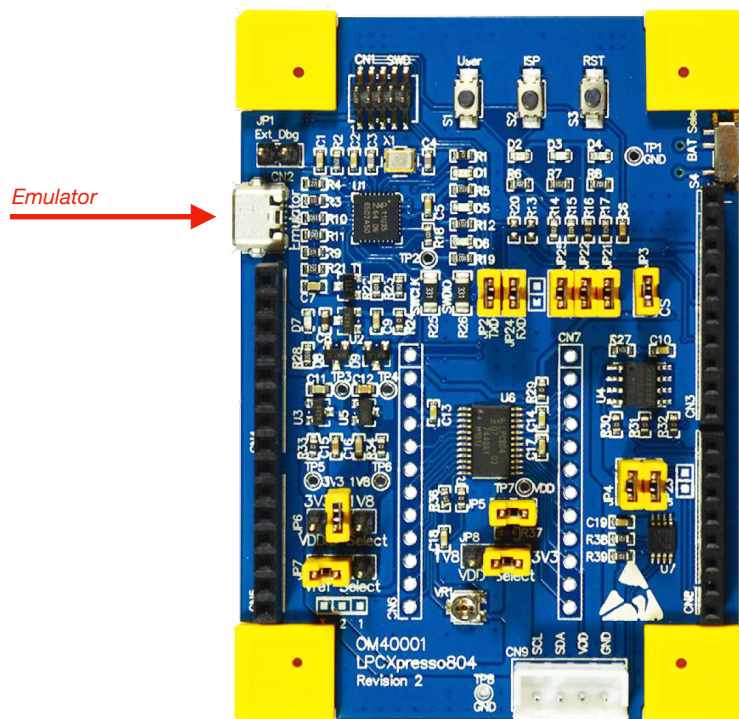
/**
 * @brief   Application entry point.
 */
int main(void) {
    /* Init board hardware. */
    BOARD_InitBootPins();
    BOARD_InitBootClocks();
    BOARD_InitBootPeripherals();
#ifdef BOARD_INIT_DEBUG_CONSOLE_PERIPHERAL
    /* Init FSL debug console. */
    BOARD_InitDebugConsole();
#endif

    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;
}
```

Dodaj znacznik „\r” na końcu tekstu w funkcji *PRINTF*.

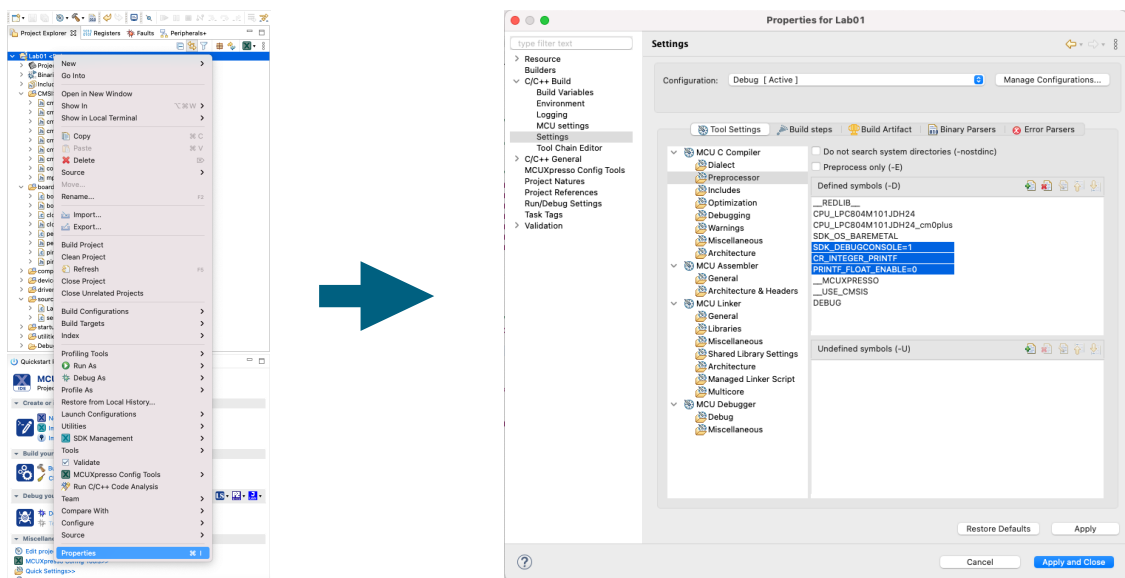
4. Podłącz płytę *LPCXpresso804* interfejsem *USB* do komputera:



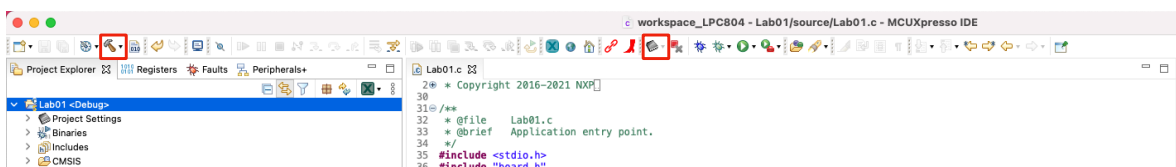
Programing of embedded systems

2. Konsola debuggera i GPIO

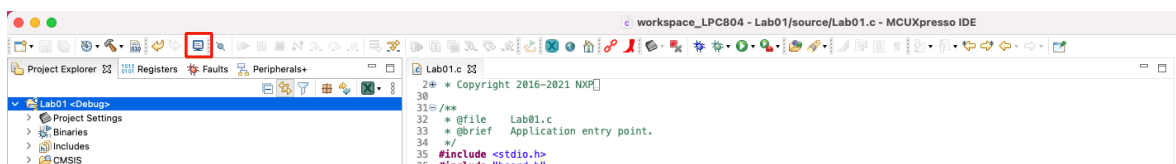
5. Kliknij prawym przyciskiem na nazwie projektu, wybierz *Properties* a następnie przejdź do *C/C++ Build -> Settings* i wybierz *Preprocessor*. Sprawdź wartość stałej `SDK_DEBUGCONSOLE`, która powinna być ustawiona na 1 (w przeciwnym wypadku - ustaw 1), jak na poniższym obrazku. Stała aktywuje konsolę debuggera, wykorzystującą *UART*. Pozostałe dwie stałe domyślnie dezaktywują obsługę liczb rzeczywistych (wykorzystywana jest ograniczona wersja biblioteki, która zajmuje mniej pamięci). Aby wyświetlać typy rzeczywiste w konsoli należy usunąć stałą `CR_INTEGER_PRINTF` oraz zmienić wartość stałej `PRINTF_FLOAT_ENABLE` na 1.



6. Zbuduj projekt klikając *Build* a następnie zaprogramuj układ klikając *GUI Flash Tool*, pozostawiając domyślne ustawienia w kolejnych oknach programatora:



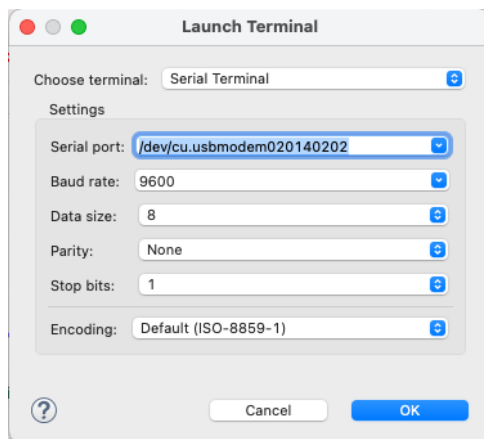
7. Uruchom terminal:



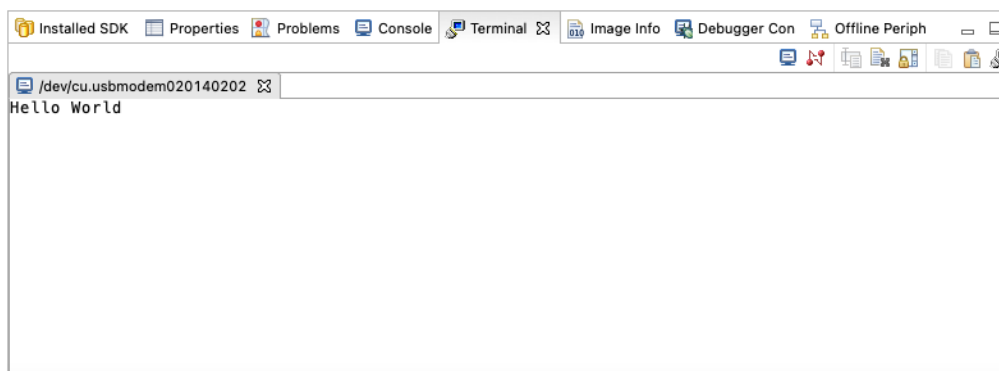
Programing of embedded systems

2. Konsola debuggera i GPIO

8. W ustawieniach wybierz *Serial Terminal* i prędkość transmisji *9600*. W polu serial port wybierz z listy dostępnych portów */dev/cu.usbmodemXXXXXXX*. Uwaga, w zależności od wersji płyty ewaluacyjnej, w miejscu „X” może pojawić nie inny kod niż na poniższym obrazku:



9. Naciśnij przycisk *RESET* na płycie *LPCXpresso55s69*. W oknie terminala powinien wyświetlić się tekst wysłany funkcją *PRINTF*:



10. Napisz prosty program „echo” wypisujący w konsoli odebrane znaki, poprzedzone tekstem „Znak: ”. W tym celu zmodyfikuj kod w funkcji main jak poniżej:

```
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 ;  
}
```

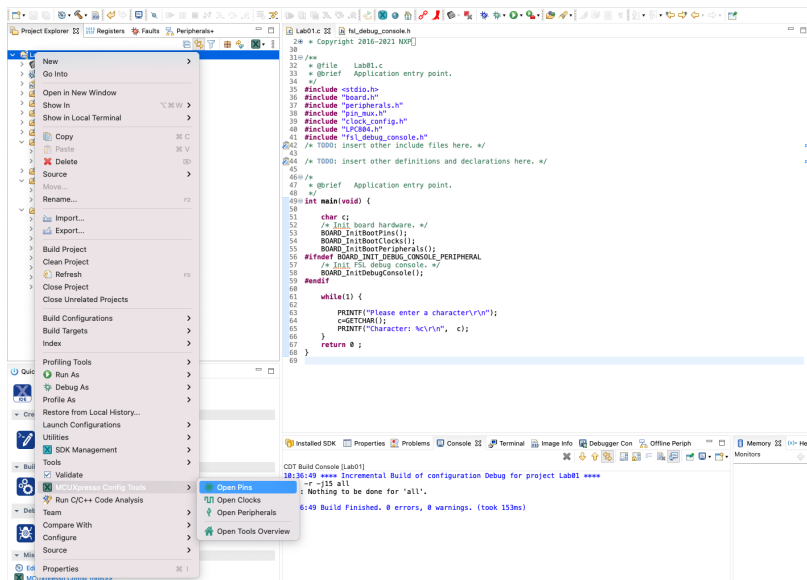
Zbuduj projekt, zaprogramuj układ i sprawdź działanie programu w konsoli terminala.

Programming of embedded systems

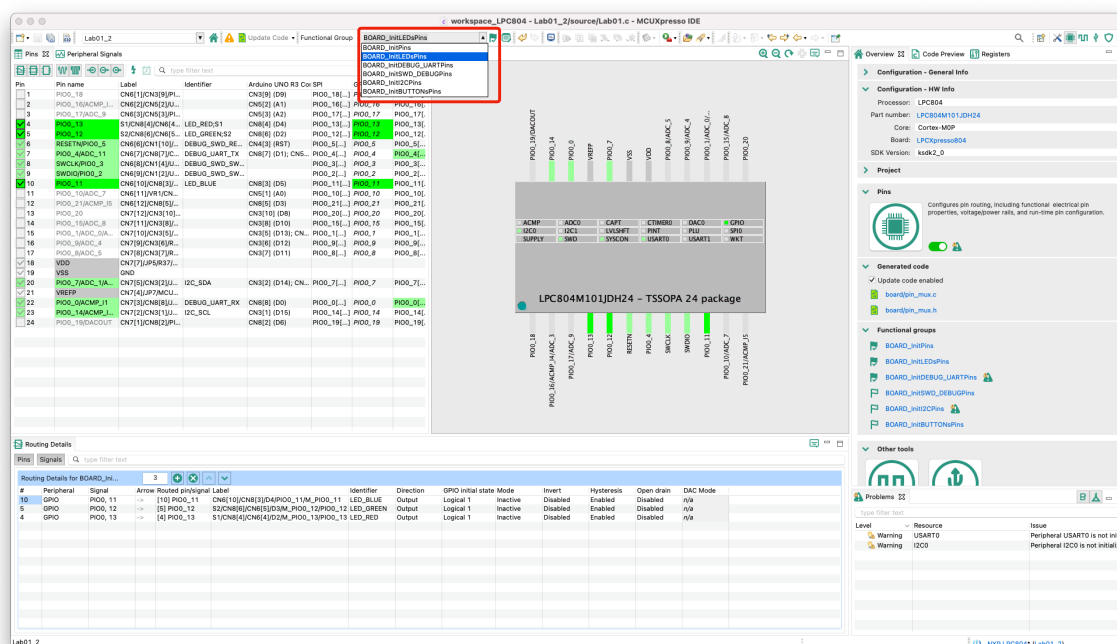
2. Konsola debuggera i GPIO

II. Sterowanie diodami LED

1. Stwórz nowy projekt dla płyty *LPCXpresso804* i nazwij go np. *Lab01_2*.
2. Należy skonfigurować 3 linie *GPIO* do sterowania poszczególnymi diodami *RGB* diody. W tym celu kliknij prawym przyciskiem na nazwie projektu i wybierz *MCUXpresso Config Tools -> Open Pins*, jak na poniższym obrazku:



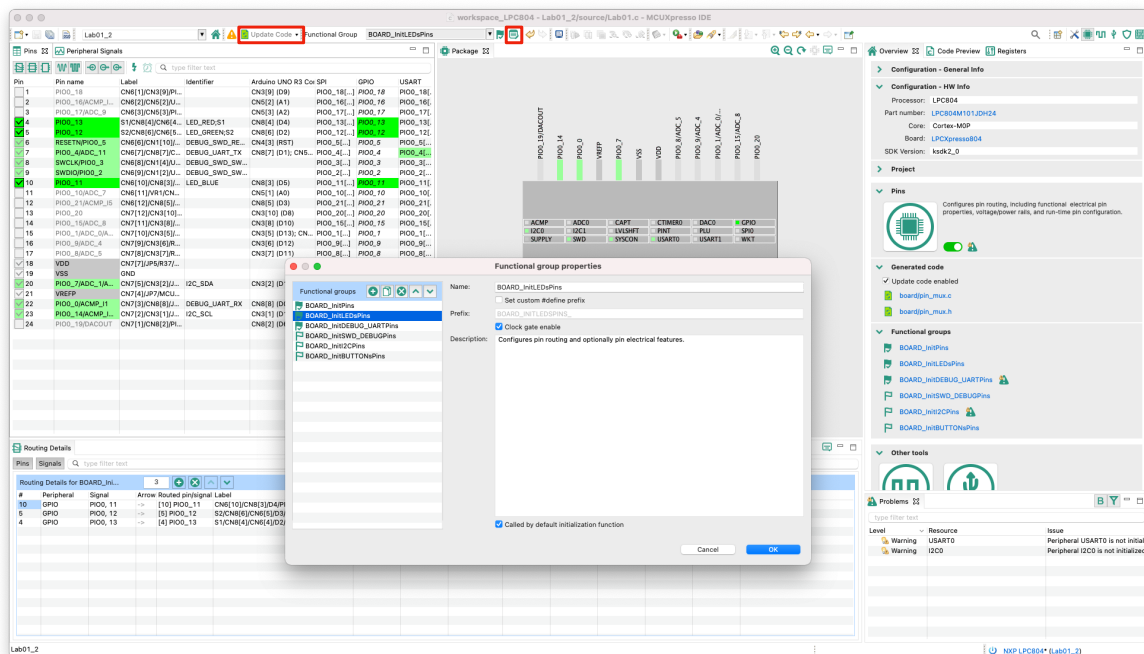
3. W oknie widoczne są automatycznie skonfigurowane linie *TXD* i *RXD* interfejsu *UART (USART0)*, wykorzystywanego przez konsolę debuggera. Z menu *Functional Group* wybierz preset *BOARD_InitLEDsPins*, a następnie aktywuj go zaznaczając ikonę flagi po lewej stronie:



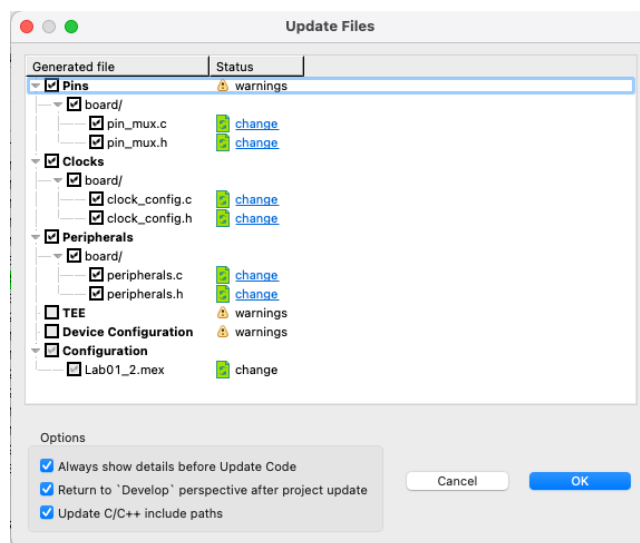
Programing of embedded systems

2. Konsola debuggera i GPIO

4. Aktywne presety można sprawdzić klikając ikonkę *Functional group properties*. W otwartym oknie widoczna jest lista presetów. Można je edytować oraz dodawać własne:



5. Wybierz *Update Code* (powyższy rysunek) w celu wygenerowania kodu na podstawie wprowadzonej konfiguracji. Kod zostanie dodany do plików oznaczonych ikoną „change”:



Przez kliknięcie w *change* można zobaczyć jakie zmiany zostaną wprowadzone w poszczególnych plikach z kodem źródłowym.

Programing of embedded systems

2. Konsola debuggera i GPIO

6. Zaakceptuj zmiany przyciskiem OK.
7. Stałe opisujące poszczególne linie, sterujące diodą RGB, zostały wygenerowane w pliku `board/pin_mux.h`:

```
Lab01.c | pin_mux.h
28 /*
29 * @brief Configures pin routing and optionally pin electrical features.
30 *
31 */
32 void BOARD_InitPins(void); /* Function assigned for the Cortex-M0P */
33
34 #define IOCON_PIO_HYS_EN 0x20u /* @brief Enable hysteresis */
35 #define IOCON_PIO_INV_DI 0x00u /* @brief Input not invert */
36 #define IOCON_PIO_MODE_INACT 0x00u /* @brief No addition pin function */
37 #define IOCON_PIO_OD_DI 0x00u /* @brief Disables Open-drain function */
38
39 /* @name PIO0_11 (number 10), CN6[10]/CN8[3]/D4/PIO0_11/M_PIO0_11
40 @{ */
41
42 /* Symbols to be used with GPIO driver */
43 #define BOARD_INITLEDSPINS_LED_BLUE_GPIO GPIO /* @brief GPIO peripheral base pointer */
44 #define BOARD_INITLEDSPINS_LED_BLUE_GPIO_PIN_MASK (1U << 11U) /* @brief GPIO pin mask */
45 #define BOARD_INITLEDSPINS_LED_BLUE_PORT 0U /* @brief PORT device index: 0 */
46 #define BOARD_INITLEDSPINS_LED_BLUE_PIN 11U /* @brief PORT pin number */
47 #define BOARD_INITLEDSPINS_LED_BLUE_PIN_MASK (1U << 11U) /* @brief PORT pin mask */
48 /* @} */
49
50 /* @name PIO0_12 (number 5), S2/CN8[6]/CN6[5]/D3/M_PIO0_12/PIO0_12
51 @{ */
52
53 /* Symbols to be used with GPIO driver */
54 #define BOARD_INITLEDSPINS_LED_GREEN_GPIO GPIO /* @brief GPIO peripheral base pointer */
55 #define BOARD_INITLEDSPINS_LED_GREEN_GPIO_PIN_MASK (1U << 12U) /* @brief GPIO pin mask */
56 #define BOARD_INITLEDSPINS_LED_GREEN_PORT 0U /* @brief PORT device index: 0 */
57 #define BOARD_INITLEDSPINS_LED_GREEN_PIN 12U /* @brief PORT pin number */
58 #define BOARD_INITLEDSPINS_LED_GREEN_PIN_MASK (1U << 12U) /* @brief PORT pin mask */
59 /* @} */
60
61 /* @name PIO0_13 (number 4), S1/CN8[4]/CN6[4]/D2/M_PIO0_13/PIO0_13
62 @{ */
63
64 /* Symbols to be used with GPIO driver */
65 #define BOARD_INITLEDSPINS_LED_RED_GPIO GPIO /* @brief GPIO peripheral base pointer */
66 #define BOARD_INITLEDSPINS_LED_RED_GPIO_PIN_MASK (1U << 13U) /* @brief GPIO pin mask */
67 #define BOARD_INITLEDSPINS_LED_RED_PORT 0U /* @brief PORT device index: 0 */
68 #define BOARD_INITLEDSPINS_LED_RED_PIN 13U /* @brief PORT pin number */
69 #define BOARD_INITLEDSPINS_LED_RED_PIN_MASK (1U << 13U) /* @brief PORT pin mask */
70 /* @} */
71
72 /*
73 * @brief Configures pin routing and optionally pin electrical features.
74 *
75 */
76 void BOARD_InitLEDsPins(void); /* Function assigned for the Cortex-M0P */
77
```

8. Zmodyfikuj kod w funkcji `main`, tak aby wysłanie znaku „a” powodowało zaświecenie diody LED na czerwono. Z kolei, wysłanie znaku „z” powinno zgasić składową czerwoną:

```
#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();
#ifdef 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 ;
}
```

Programing of embedded systems

2. Konsola debugera i GPIO

Zbuduj projekt, zaprogramuj układ i sprawdź działanie programu.

III. Obsługa przycisków

1. Stwórz nowy projekt dla płyty *LPCXpresso804* i nazwij go np. *Lab01_3*.
2. Otwórz ponownie narzędzie do konfiguracji wyprowadzeń: *MCUXpresso Config Tools* -> *Open Pins*. Ponieważ na płytce prototypowej diody LED i przyciski współdzielili wyprowadzenie mikrokontrolera, odznacz preset *BOARD_InitLEDsPins* i zaznacz preset *BOARD_InitBUTTONsPins* a następnie kliknij *Update Code*.
3. Zmodyfikuj kod w funkcji *main*, tak aby wciśnięcie przycisku *S1* wypisywało w konsoli komunikat „S1” natomiast wciśnięcie przycisku *S2* ma wypisać komunikat „S2”:

```
#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();
#ifdef 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 ;
}
```

Zbuduj projekt, zaprogramuj układ i sprawdź działanie programu.

4. Zmodyfikuj kod w funkcji *main*, tak aby wciśnięcie przycisku *S1* oraz *S2* wypisywało w konsoli odpowiedni komunikat tylko raz (detekcja zbocza opadającego):

```
#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) {
```

Programing of embedded systems

2. Konsola debuggera i GPIO

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

/* Init board hardware. */
BOARD_InitBootPins();
BOARD_InitBootClocks();
BOARD_InitBootPeripherals();
#ifdef 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 ;
}
```

Zbuduj projekt, zaprogramuj układ i sprawdź działanie programu.

IV. Exercises

1. Zmodyfikuj program sterowania LED tak, aby możliwe było sterowanie trzema diodami RGB. Wysyłając znak:
 - a: Red-On
 - z: Red-Off
 - s: Green-On
 - x: Green-Off
 - d: Blue-On
 - c: Blue-Off
2. Napisz ten sam program, używając instrukcji switch-case.
3. Zmodyfikuj program obsługi przycisków, aby można było wykryć zwolnienie przycisku.