ENGINE

Teaching online electronics, microcontrollers and programming in Higher Education

Programing of embedded systems

3. Układy czasowo-licznikowe

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 Where other published and unpublished source materials have been used, these have been acknowledged.

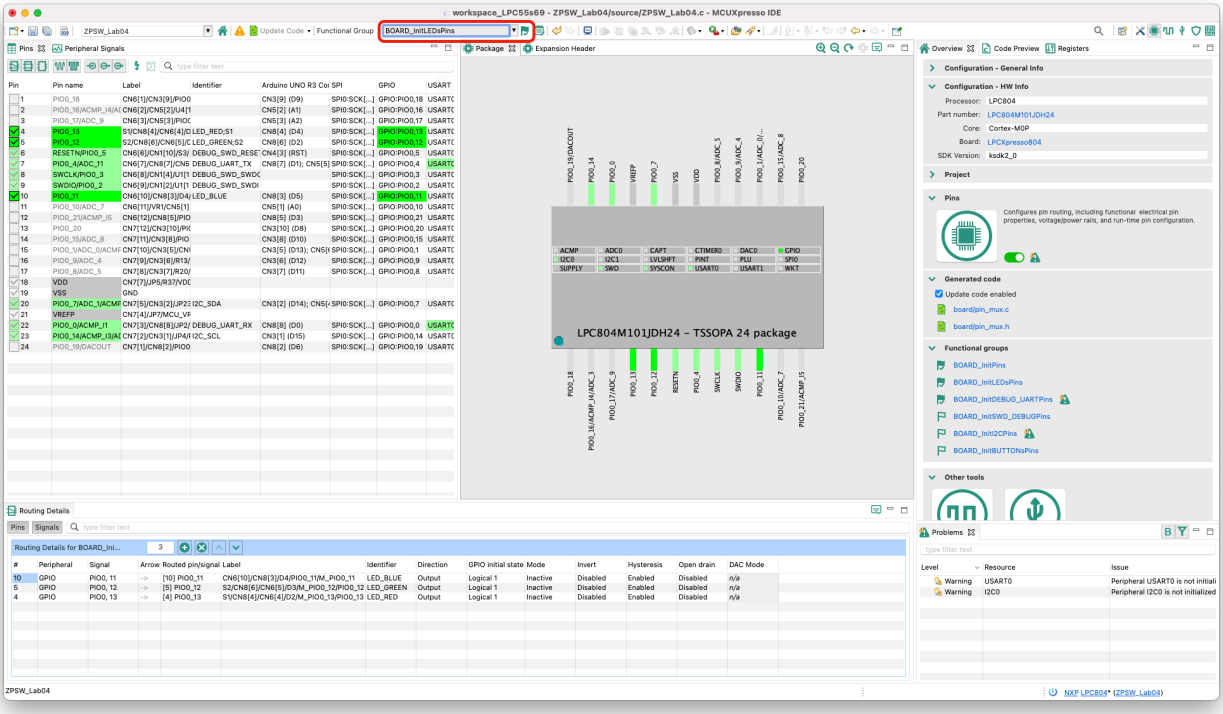
Copyright

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.

I. Timer systemowy

1. Stwórz nowy projekt dla płyty *LPCXpresso804*, tak jak na poprzednich zajęciach i nazwij go *ZPSW_Lab04*.
2. Skonfiguruj trzy linie *GPIO* do sterowania diodami RGB. W tym celu, przejdź do *Config Tool -> Open Pins* a następnie z menu *Functional Group* wybierz preset *BOARD_InitLEDsPins* i aktywuj go zaznaczając ikonę flagi po prawo stronie:



3. Naciśnij *Update Code*.
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"

bool g_pinState = false;

void SysTick_Handler(void) {

          GPIO_PinWrite(BOARD_INITLEDSPINS_LED_RED_GPIO,
                        BOARD_INITLEDSPINS_LED_RED_PORT,
                        BOARD_INITLEDSPINS_LED_RED_PIN,
                        g_pinState ^= true);
}

/*
 * @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

          SysTick_Config(SystemCoreClock / 10U); // 10 Hz
```

```
        while(1) {

        }
        return 0 ;
}
```

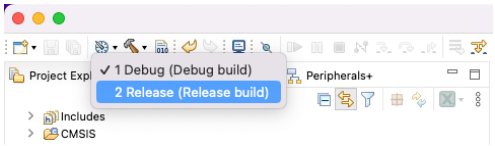
5. Zbuduj projekt, zaprogramuj układ i sprawdź działanie programu.

II. Funkcja „delay"

1. Stwórz nowy projekt dla płyty *LPCXpresso804* i nazwij go *ZPSW_Lab04_2*.
2. Tak jak poprzednio, skonfiguruj trzy linie *GPIO* do sterowania diodami RGB.
3. 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"

bool g_pinState = false;
uint32_t g_systickCounter;

void SysTick_Handler(void) {

        if (g_systickCounter) {
                g_systickCounter--;
        }
}
void delay_ms(uint32_t n) {

        g_systickCounter = n;
        while (g_systickCounter)
                ;
}

/*
 * @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

        SysTick_Config(SystemCoreClock / 1000U); // 1 kHz

        while(1) {

                GPIO_PinWrite(BOARD_INITLEDSPINS_LED_RED_GPIO,
                              BOARD_INITLEDSPINS_LED_RED_PORT,
                              BOARD_INITLEDSPINS_LED_RED_PIN,
                              g_pinState ^= true);

                delay_ms(500);
        }
        return 0 ;
}
```

4. Zbuduj projekt, zaprogramuj układ i sprawdź działanie. Dioda LED powinna zmieniać stan 2 razy na sekundę (1 błysk co sekundę).
5. Przebuduj projekt w konfiguracji *Relase*, zmieniając ustawienia w rozwijanym menu przy ikonie *Build*:



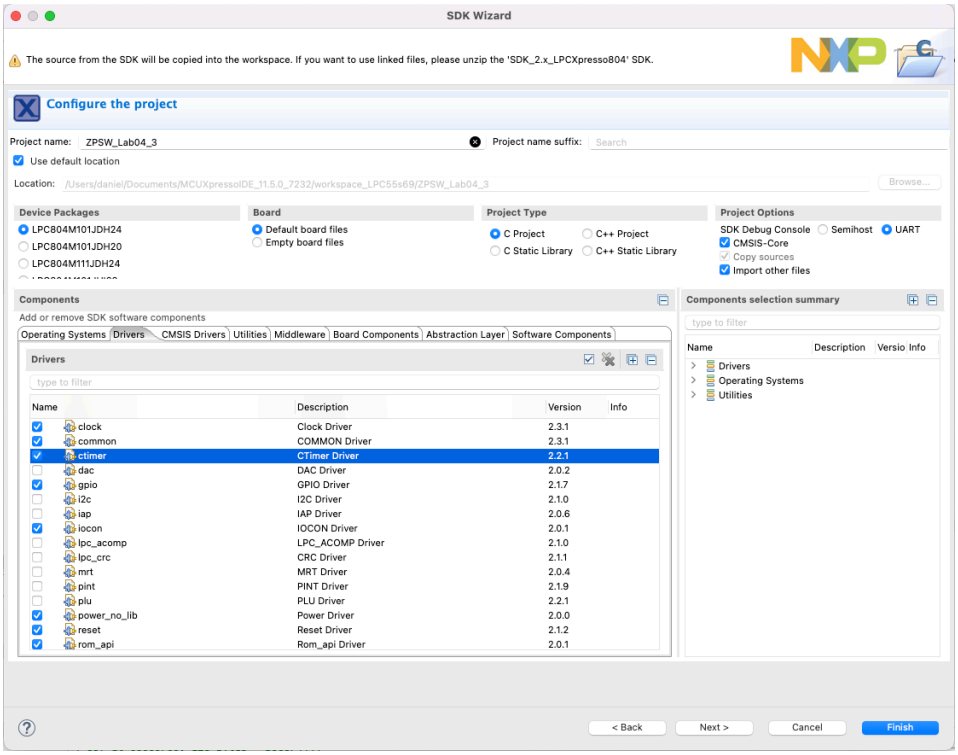
6. Zbuduj projekt, zaprogramuj układ i sprawdź działanie. Ze względu na optymalizację kompilatora, zmienna *g_systickCounter* nie jest „odświeżana" w pętli *while*, znajdującej się wewnątrz funkcji *delay_ms*. Co za tym idzie, dioda LED przestanie błyskać.
7. W celu wymuszenia każdorazowego „odświeżenia" wartości zmiennej *g_systickCounter* dodaj modyfikator *volatile*:

```
#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"

bool g_pinState = false;
volatile uint32_t g_systickCounter;

void SysTick_Handler(void) {

          if (g_systickCounter) {
                    g_systickCounter--;
          }
}
void delay_ms(uint32_t n) {

          g_systickCounter = n;
          while (g_systickCounter)
                    ;
}

/*
 * @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

          SysTick_Config(SystemCoreClock / 1000U); // 1 kHz

          while(1) {

                    GPIO_PinWrite(BOARD_INITLEDSPINS_LED_RED_GPIO,
                                        BOARD_INITLEDSPINS_LED_RED_PORT,
                                        BOARD_INITLEDSPINS_LED_RED_PIN,
                                        g_pinState ^= true);

                    delay_ms(500);
          }
          return 0 ;
}
```

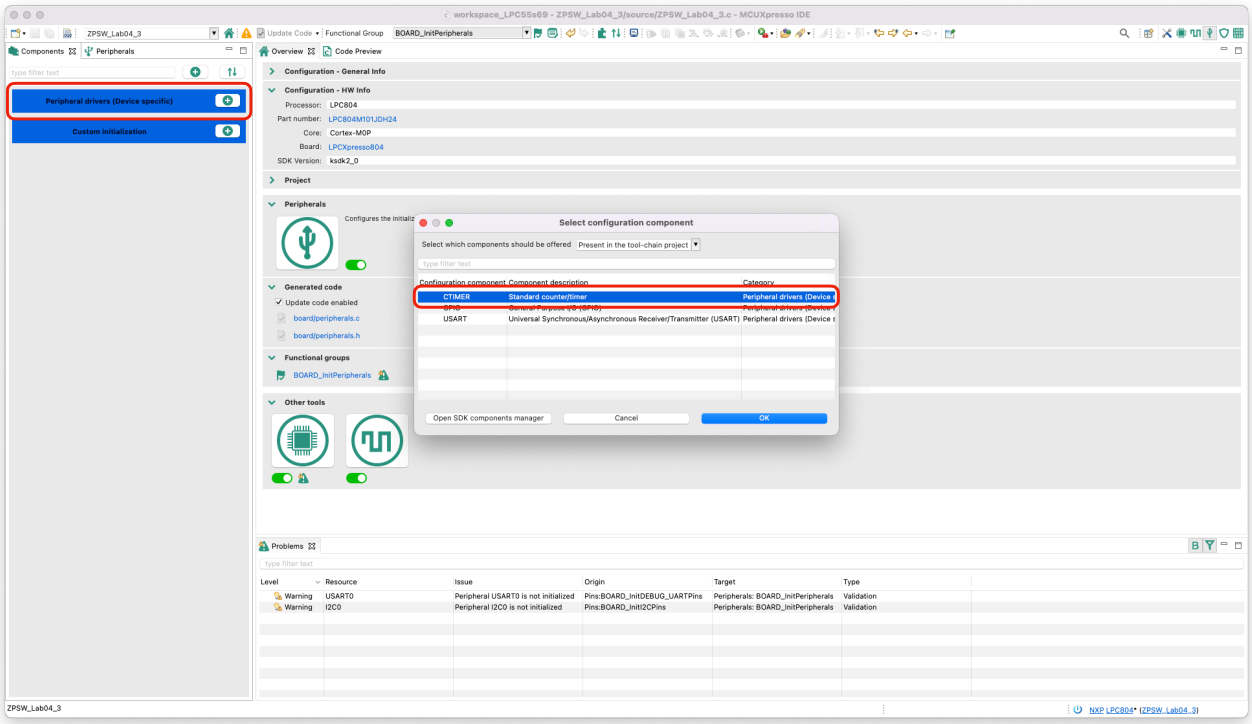
8. Zbuduj projekt, zaprogramuj układ i sprawdź działanie. Dioda led powinna zmieniać stan 2 razy na sekundę (1 błysk co sekundę), jak miało to miejsce w trybie *Debug*.

III. Układ CTIMER - tryb Match

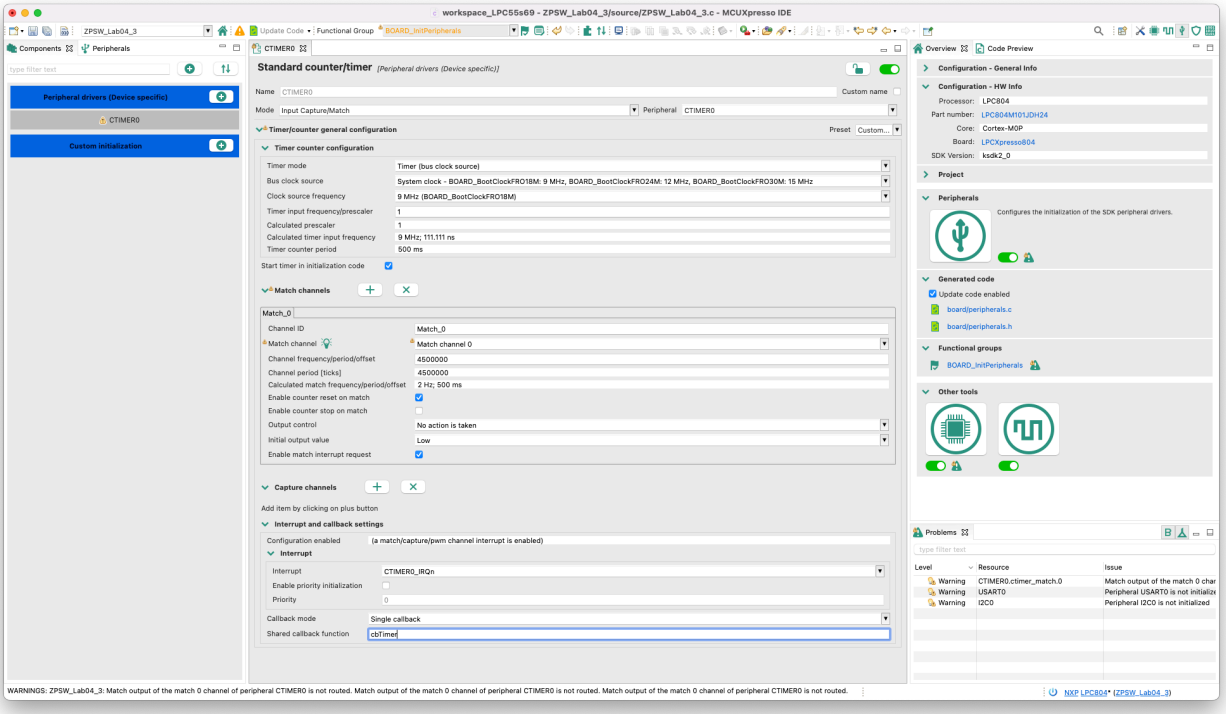
1. Stwórz nowy projekt dla płyty *LPCXpresso804* i nazwij go *ZPSW_Lab04_3*.
2. Dodaj sterownik *ctimer*:



3. Przejdź do *Config Tools -> Peripherals* i dodaj konfigurację sterownika układów *CTIMER*:



4. Skonfiguruj układ *CTIMER0*:

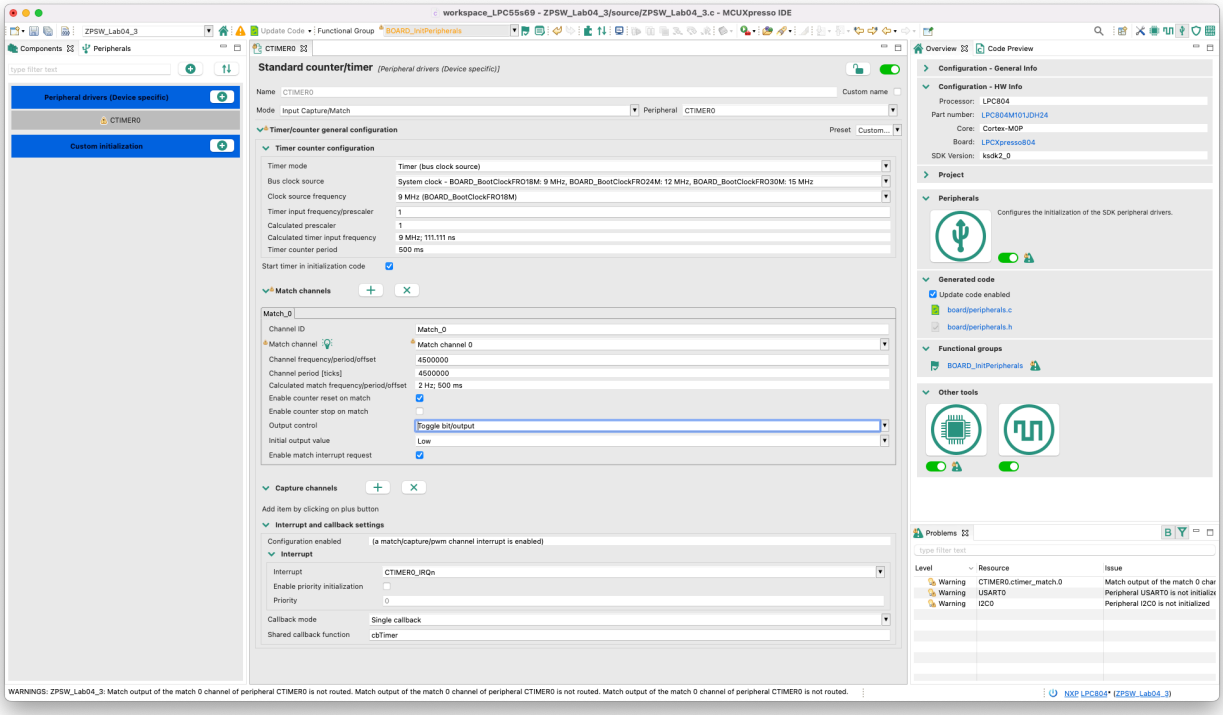


5. Naciśnij *Update Code*.
6. Przejdź do głównego pliku projektu i zmodyfikuj kod jak poniżej:

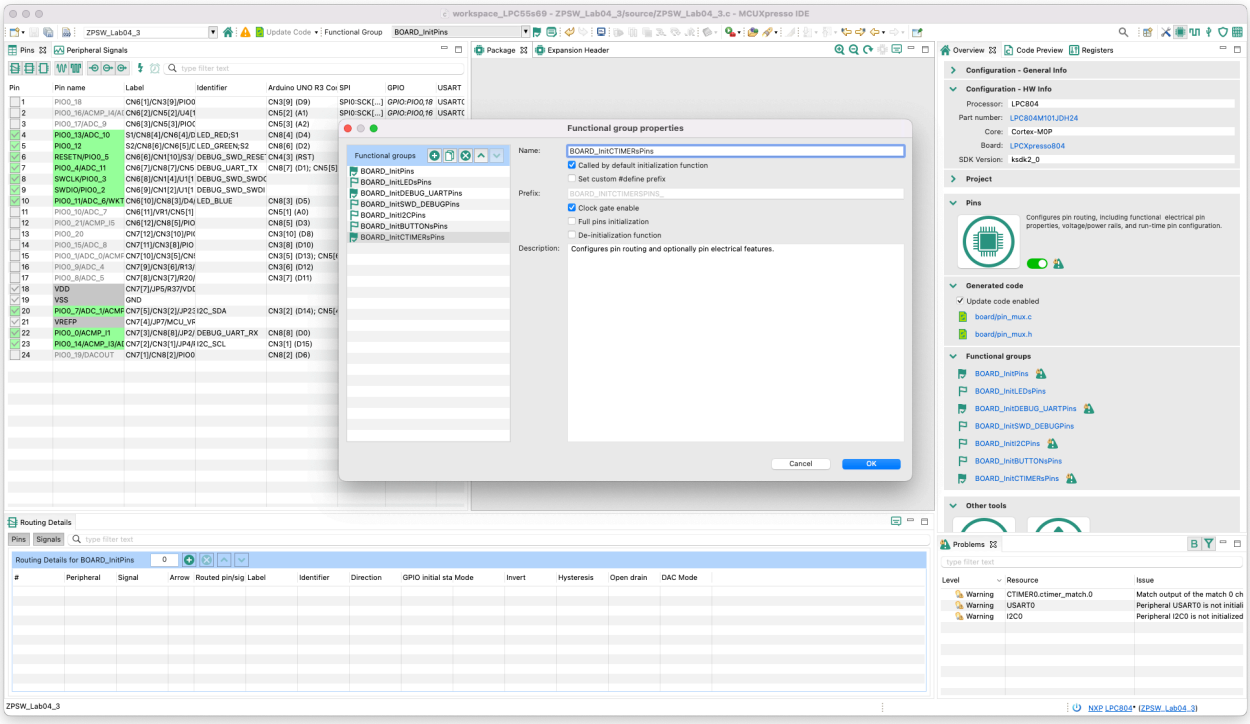
```
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "LPC804.h"
#include "fsl_debug_console.h"

volatile uint32_t step=0;

void cbTimer(uint32_t flags) {

          PRINTF("Timer INT: %d\r\n", step++);
}

/*
 * @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) {

    }
    return 0 ;
}
```

7. Zbuduj projekt, zaprogramuj układ.
8. Uruchom terminal i sprawdź działanie programu w konsoli debuggera.

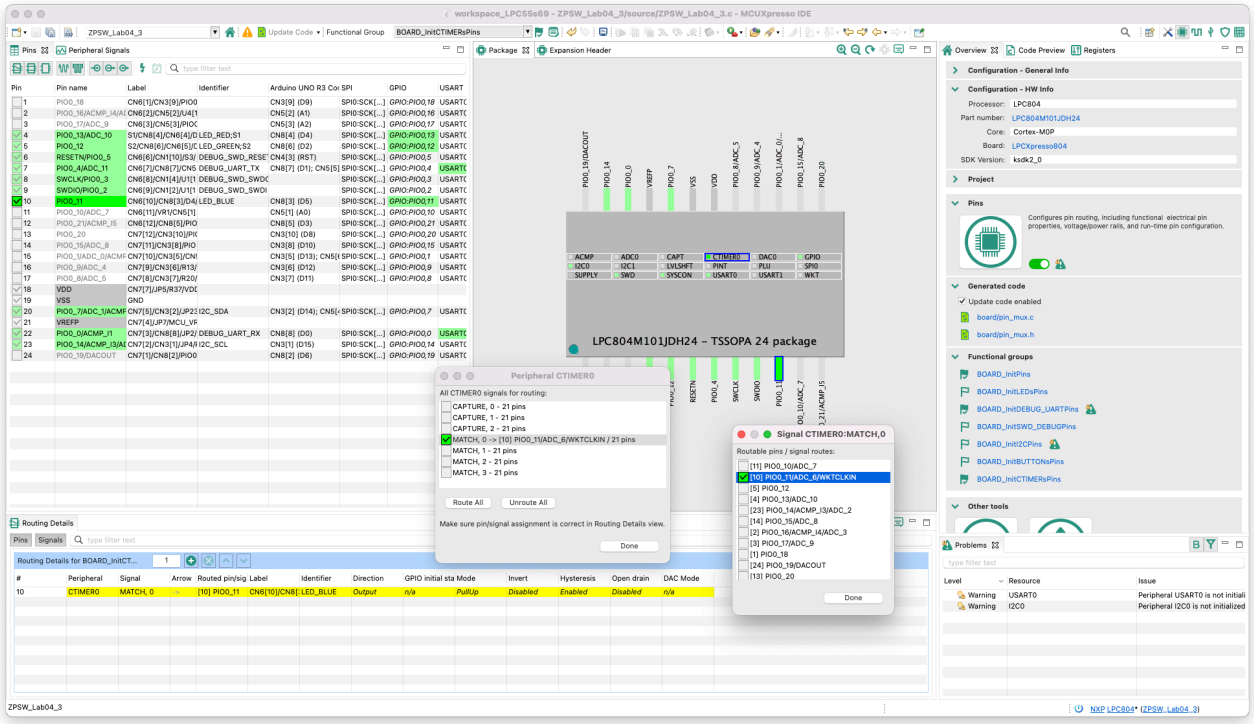
9. Przejdź do *Config Tools -> Peripherals* i aktywuj sprzętowe wyjście bloku *Match*:



10. Przejdź do *MCUXpresso Config Tools-> Pins* i w *Functional group properties* a następnie stwórz własny preset o nazwie *BOARD_InitCTIMERsPins*:



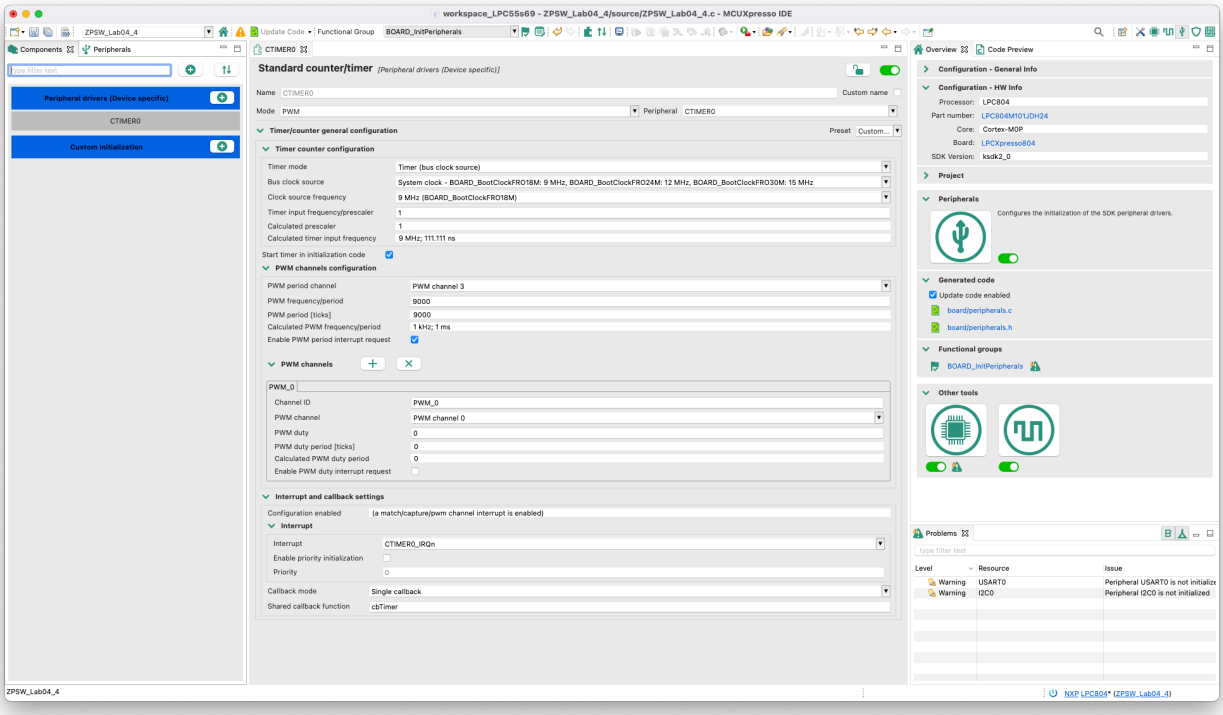
11. Na rysunku przedstawiającym mikrokontroler kliknij na *CTIMER*.
12. W otwartych oknach dialogowych wybierz odpowiednio *MATCH,0* a następnie *PIO0_11*:



13. Naciśnij *Done* w poszczególnych oknach dialogowych a następnie *Update Code*.
14. Zbuduj projekt, zaprogramuj układ i sprawdź działanie. Dioda *LED* (niebieska) powinna zmieniać stan 2 razy na sekundę (1 błysk co sekundę).

IV. Układ CTIMER - tryb PWM

1. Przejdź do *Peripherals* i zmień konfigurację *CTIMER0* na *PWM* i ustaw wartości jak poniżej:



2. Naciśnij *Update Code* i zmodyfikuj kod programu:

```
#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"

volatile uint8_t pwmDuty0=0;

void cbTimer(uint32_t flags) {

          CTIMER_UpdatePwmDutycycle(CTIMER0_PERIPHERAL,
                                    CTIMER0_PWM_PERIOD_CH,
                                    CTIMER0_PWM_0_CHANNEL,
                                    100-pwmDuty0); // LED is active low

}

/*
 * @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");

          char c;

          while(1) {

                    c=GETCHAR();

                    switch(c) {
```

```
                    case 'a':
                              if(pwmDuty0 < 100) {

                                        pwmDuty0++;
                              }
                              PRINTF("PWM0: %d\r\n", pwmDuty0);
                              break;

                    case 'z':
                              if(pwmDuty0 > 0) {

                                        pwmDuty0--;
                              }
                              PRINTF("PWM0: %d\r\n", pwmDuty0);
                              break;
                    }

          }
          return 0 ;
}
```

3. Zbuduj projekt i zaprogramuj układ.
4. Uruchom terminal i sprawdź sterowanie jasnością niebieskiej diody LED za pomocą klawiatury.

   a: Blue PWM ++
   z: Blue PWM --

V. Zadania

1. Skonfiguruj dodatkowe kanały PWM (*PWM_1* i *PWM_2*) dla *CTIMER0*.
2. Podłącz ich wyjścia odpowiednio do *PIO0_12* (*Green LED*) oraz *PIO0_13* (*Red LED*).
3. Napisz program sterujący niezależnie jasnością każdej z 3 diod LED za pomocą terminala. Wysyłane znaki:

   a: Blue PWM ++
   z: Blue PWM --
   s: Green PWM ++
   x: Green PWM --
   d: Red PWM ++
   c: Red PWM --