ENGINE

Teaching online electronics, microcontrollers and programming in Higher Education

Programing of embedded systems

3. Timers and counters

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 Where other published and unpublished source materials have been used, these have been acknowledged.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

3. Timers and counters

I. System Timer

1. Create a new project for the *LPCXpresso804* board as in the previous manual and name the project eg *Lab02*.
2. Configure three GPIO lines to control the RGB LEDs. From the Functional Group menu, select the *BOARD_InitLEDsPins* preset, then activate it by selecting the flag icon on the left, as in the previous manual. Select *Update Code* to generate the code based on the entered configuration.
3. Modify the program code by adding system timer support:

```
#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 ;
}
```

Build a project, program the microcontroller and check the operation. The led should change state 10 times per second (5 flashes per second).

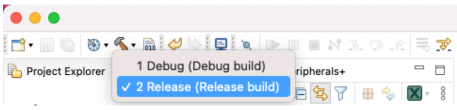
II. Delay function

1. Create a new project for the *LPCXpresso804* board and name it eg *Lab02_2*.
2. *As before, configure three GPIO lines to control the RGB LEDs*. Modify the program code as in the example 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"

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 ms

          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 ;
}
```

Build a project, program the microcontroller and check the operation. The led should change state 2 times per second (1 flash every second).

3. Rebuild the project in the *Release* configuration by changing the settings in the drop-down menu next to the Build icon:



Build a project, program the microcontroller and check the operation. Due to compiler optimization, the *g_systickCounter* variable is not "refreshed" in the while loop inside the *delay_ms* function. Hence, the LED will stop flashing.

4. To force the value of the *g_systickCounter* variable to "refresh" each time, add the *volatile* modifier:

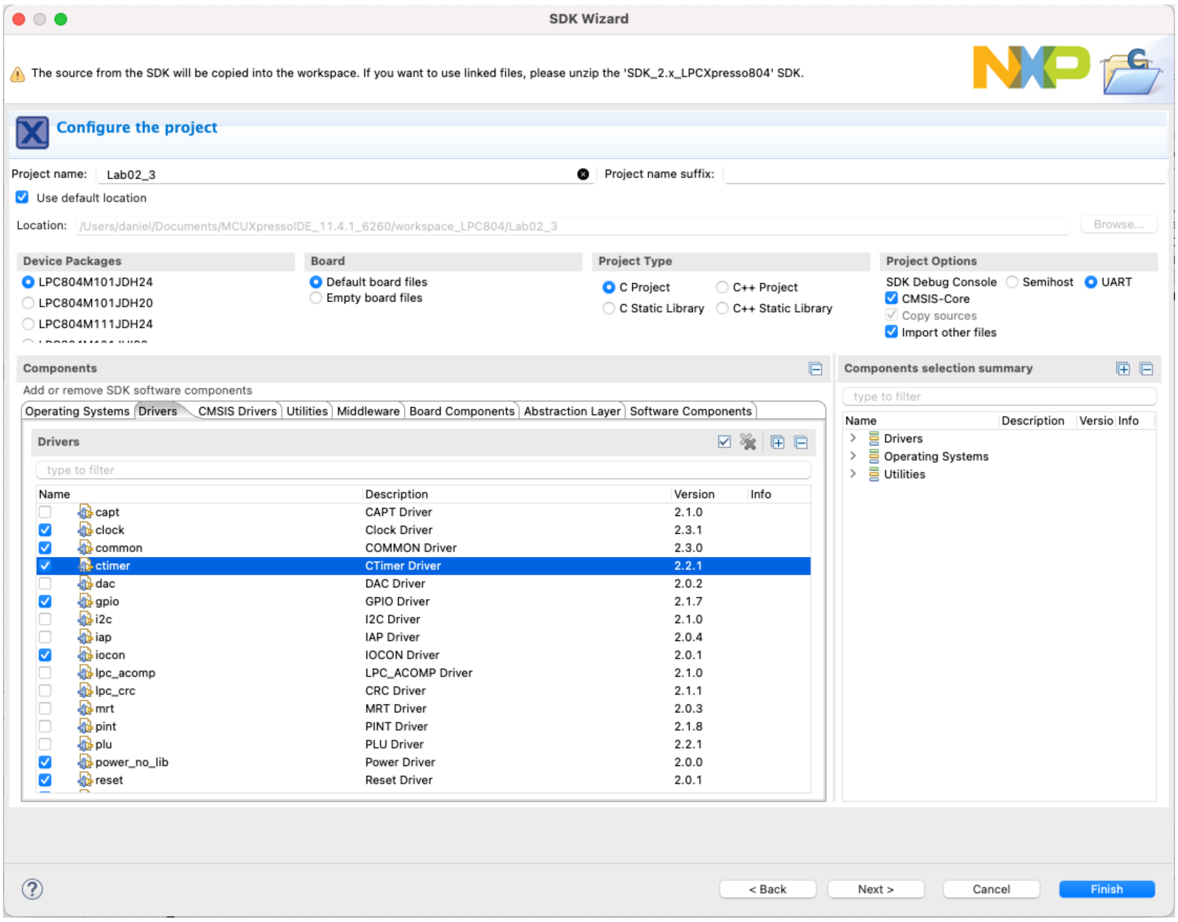
```
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "LPC804.h"
#include "fsl_debug_console.h"

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 ms
```

```
        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 ;
}
```

Build a project, program the microcontroller and check the operation. The led should change state 2 times per second (1 flash every second) as it did in *Debug* mode.

III. CTIMER - Match mode

1. Create a new project for the *LPCXpresso804* board and name it eg *Lab02_3*. Add the *ctimer* 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: Lab02_3 Project name suffix:

☑ Use default location

Location: /Users/daniel/Documents/MCUXpressoIDE_11.4.1_6260/workspace_LPC804/Lab02_3

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

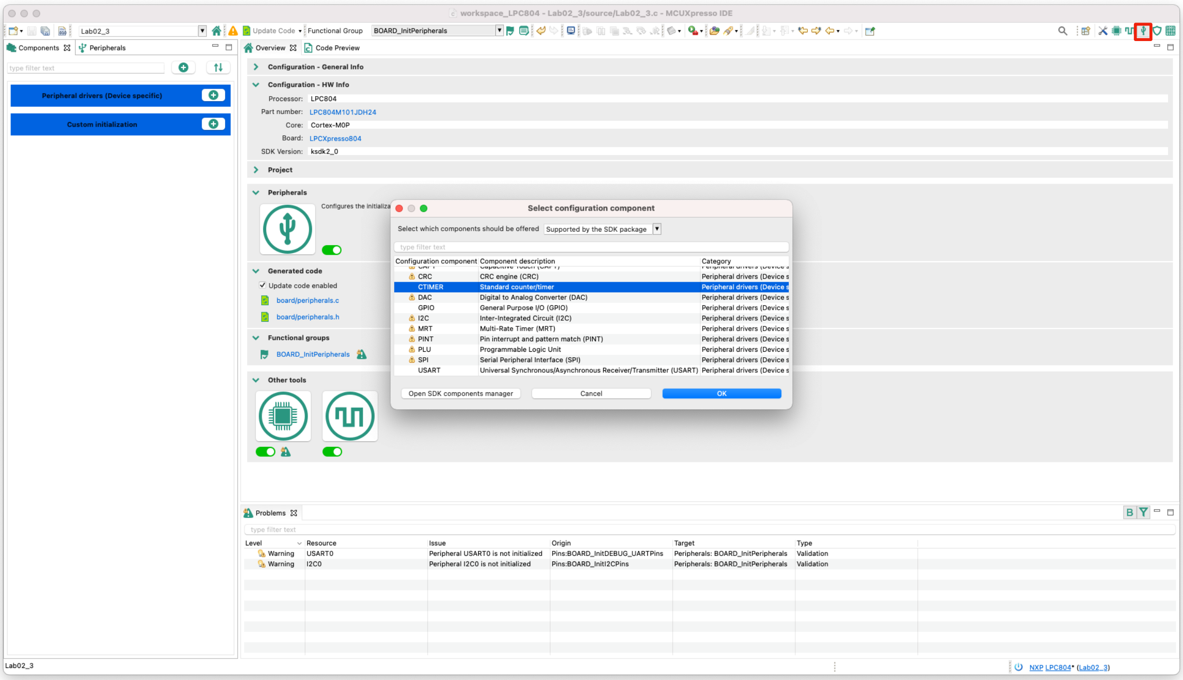
Drivers

	Name	Description	Version	Info
□	capt	CAPT Driver	2.1.0	
☑	clock	Clock Driver	2.3.1	
☑	common	COMMON Driver	2.3.0	
☑	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	2.0.4	
☑	iocon	IOCON Driver	2.0.1	
□	lpc_acomp	LPC_ACOMP Driver	2.1.0	
□	lpc_crc	CRC Driver	2.1.1	
□	mrt	MRT Driver	2.0.3	
□	pint	PINT Driver	2.1.8	
□	plu	PLU Driver	2.2.1	
☑	power_no_lib	Power Driver	2.0.0	
☑	reset	Reset Driver	2.0.1	

Components selection summary

Name	Description	Versio	Info
› Drivers			
› Operating Systems			
› Utilities			

2. To do this, right-click on the project name and select *MCUXpresso Config Tools -> Open Pheriperals*:



workspace_LPC804 - Lab02_3/source/Lab02_3.c - MCUXpresso IDE

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

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

Select configuration component

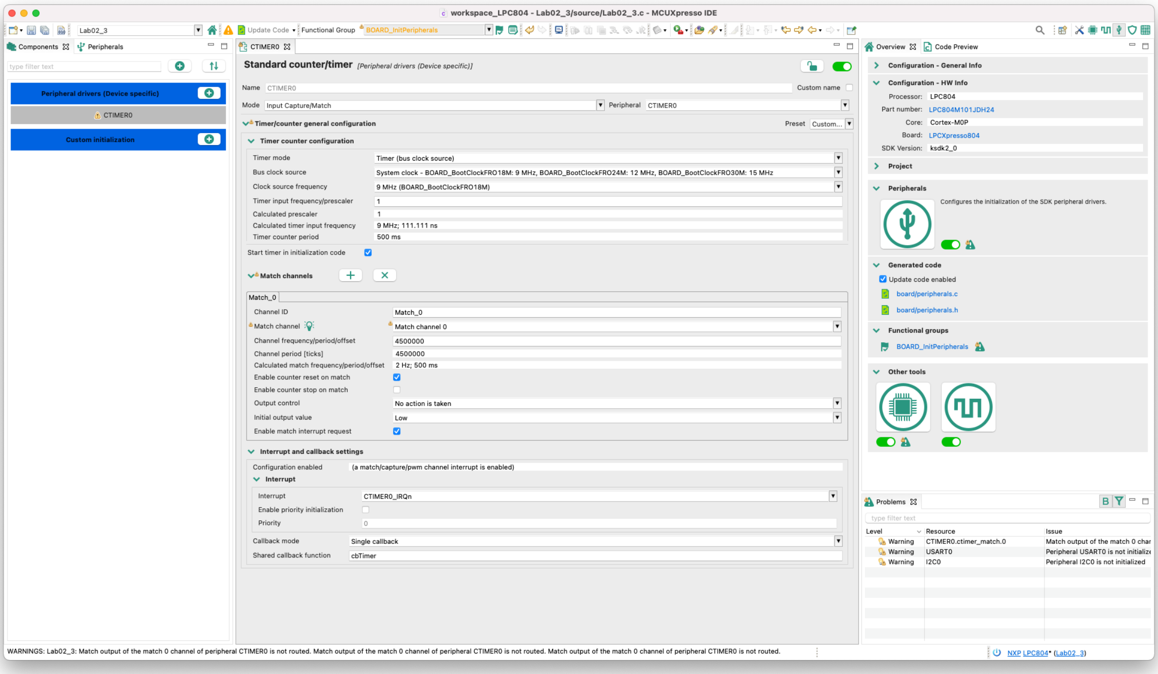
Select which components should be offered: Supported by the SDK package

Configuration component	Component description	Category
CRC	CRC engine (CRC)	Peripheral drivers (Device s
CTIMER	Standard counter/timer	Peripheral drivers (Device s
DAC	Digital to Analog Converter (DAC)	Peripheral drivers (Device s
GPIO	General Purpose I/O (GPIO)	Peripheral drivers (Device s
I2C	Inter-Integrated Circuit (I2C)	Peripheral drivers (Device s
MRT	Multi-Rate Timer (MRT)	Peripheral drivers (Device s
PINT	Pin interrupt and pattern match (PINT)	Peripheral drivers (Device s
PLU	Programmable Logic Unit	Peripheral drivers (Device s
SPI	Serial Peripheral Interface (SPI)	Peripheral drivers (Device s
USART	Universal Synchronous/Asynchronous Receiver/Transmitter (USART)	Peripheral drivers (Device s

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

Lab02_3 NXP LPC804* (Lab02_3)

3. Configure the *CTIMER0* pheriperal:



workspace_LPC804 - Lab02_3/source/Lab02_3.c - MCUXpresso IDE

Standard counter/timer *[Peripheral drivers (Device specific)]*

Name: CTIMER0 Custom name □
Mode: Input Capture/Match Peripheral: CTIMER0

Timer/counter general configuration Preset: Custom...

Timer counter configuration

Timer mode	Timer (bus clock source)
Bus clock source	System clock - BOARD_BootClockFRO18M: 9 MHz, BOARD_BootClockFRO24M: 12 MHz, BOARD_BootClockFRO30M: 15 MHz
Clock source frequency	9 MHz (BOARD_BootClockFRO18M)
Timer input frequency/prescaler	1
Calculated prescaler	1
Calculated timer input frequency	9 MHz; 111.111 ns
Timer counter period	500 ms

Start timer in initialization code ☑

Match channels

Match_0

Channel ID	Match_0
Match channel	Match channel 0
Channel frequency/period/offset	4500000
Channel period [ticks]	4500000
Calculated match frequency/period/offset	2 Hz; 500 ms
Enable counter reset on match	☑
Enable counter stop on match	□
Output control	No action is taken
Initial output value	Low
Enable match interrupt request	☑

Interrupt and callback settings

Configuration enabled (a match/capture/pwm channel interrupt is enabled)

Interrupt

Interrupt	CTIMER0_IRQn
Enable priority initialization	□
Priority	0
Callback mode	Single callback
Shared callback function	cbTimer

Level	Resource	Issue
Warning	CTIMER0.ctimer_match.0	Match output of the match 0 char
Warning	USART0	Peripheral USART0 is not initialize
Warning	I2C0	Peripheral I2C0 is not initialized

WARNINGS: Lab02_3: Match output of the match 0 channel of peripheral CTIMER0 is not routed. Match output of the match 0 channel of peripheral CTIMER0 is not routed. Match output of the match 0 channel of peripheral CTIMER0 is not routed.

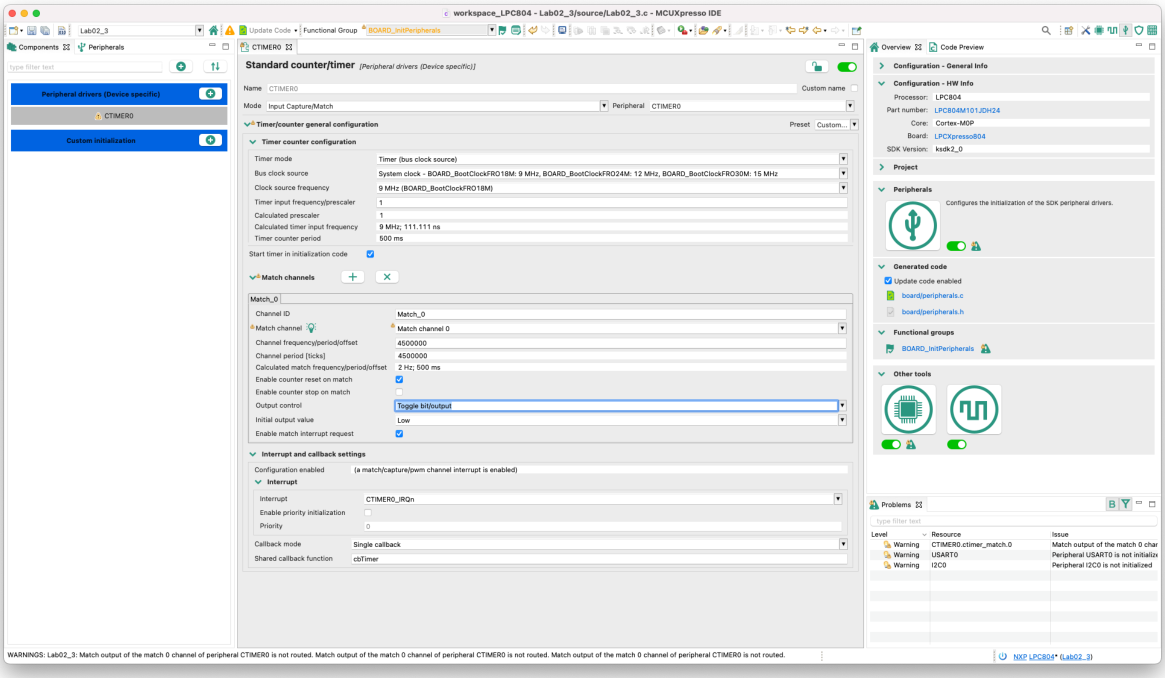
NXP LPC804* (Lab02_3)

4. Press *Update Code* and modify the program code by adding the *cbTimer* function, the definition of which was generated in the file *peripherals.h*:

```
#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"

void cbTimer(uint32_t flags) {

          PRINTF("Timer INT\r\n");
}

/*
 * @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\n");

    while(1) {

    }
    return 0 ;
}
```

Build a project, program the microcontroller, start the debugger console and check the program operation.

5. Go back to *MCUXpresso Config Tools-> Pheriperals* and activate the hardware output of the *Match* block:



Standard counter/timer *[Peripheral drivers (Device specific)]*

Name: CTIMER0 Custom name □
Mode: Input Capture/Match Peripheral: CTIMER0

Timer/counter general configuration Preset: Custom...

Timer counter configuration

Timer mode: Timer (bus clock source)
Bus clock source: System clock - BOARD_BootClockFRO18M: 9 MHz, BOARD_BootClockFRO24M: 12 MHz, BOARD_BootClockFRO30M: 15 MHz
Clock source frequency: 9 MHz (BOARD_BootClockFRO18M)
Timer input frequency/prescaler: 1
Calculated prescaler: 1
Calculated timer input frequency: 9 MHz; 111.111 ns
Timer counter period: 500 ms

Start timer in initialization code ☑

Match channels

Match_0

Channel ID: Match_0
Match channel: Match channel 0
Channel frequency/period/offset: 4500000
Channel period [ticks]: 4500000
Calculated match frequency/period/offset: 2 Hz; 500 ms
Enable counter reset on match ☑
Enable counter stop on match □
Output control: Toggle bit/output
Initial output value: Low
Enable match interrupt request ☑

Interrupt and callback settings

Configuration enabled: (a match/capture/pwm channel interrupt is enabled)

Interrupt

Interrupt: CTIMER0_IRQn
Enable priority initialization □
Priority: 0

Callback mode: Single callback
Shared callback function: cbTimer

Configuration - General Info

Configuration - HW Info

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

Project

Peripherals

Configures the initialization of the SDK peripheral drivers.

Generated code

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

Functional groups

BOARD_InitPeripherals

Other tools

Level	Resource	Issue
Warning	CTIMER0.ctimer_match.0	Match output of the match 0 char
Warning	USART0	Peripheral USART0 is not initialize
Warning	I2C0	Peripheral I2C0 is not initialized

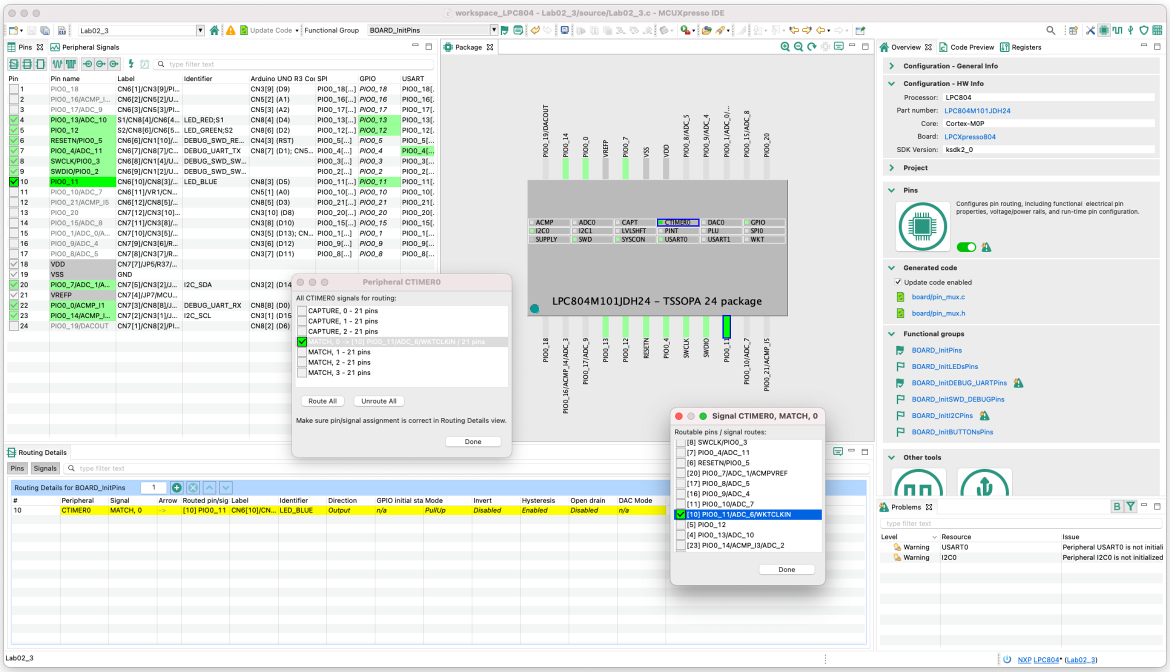
WARNINGS: Lab02_3: Match output of the match 0 channel of peripheral CTIMER0 is not routed. Match output of the match 0 channel of peripheral CTIMER0 is not routed. Match output of the match 0 channel of peripheral CTIMER0 is not routed.

NXP LPC804* (Lab02_3)

Programing of embedded systems

3. Timers and counters

6. In the picture showing the microcontroller, click on *CTIMER*.
7. In the open dialog boxes select MATCH, 0 and then PIO0_11, respectively:



Peripheral CTIMER0

All CTIMER0 signals for routing:

- □ CAPTURE, 0 - 21 pins
- □ CAPTURE, 1 - 21 pins
- □ CAPTURE, 2 - 21 pins
- ☑ MATCH, 0 -> [10] PIO0_11/ADC_6/WKTCLKIN / 21 pins
- □ MATCH, 1 - 21 pins
- □ MATCH, 2 - 21 pins
- □ MATCH, 3 - 21 pins

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

Signal CTIMER0, MATCH, 0

Routable pins / signal routes:

- □ [8] SWCLK/PIO0_3
- □ [7] PIO0_4/ADC_11
- □ [6] RESETN/PIO0_5
- □ [20] PIO0_7/ADC_1/ACMPVREF
- □ [17] PIO0_8/ADC_5
- □ [16] PIO0_9/ADC_4
- □ [11] PIO0_10/ADC_7
- ☑ [10] PIO0_11/ADC_6/WKTCLKIN
- □ [5] PIO0_12
- □ [4] PIO0_13/ADC_10
- □ [23] PIO0_14/ACMP_I3/ADC_2

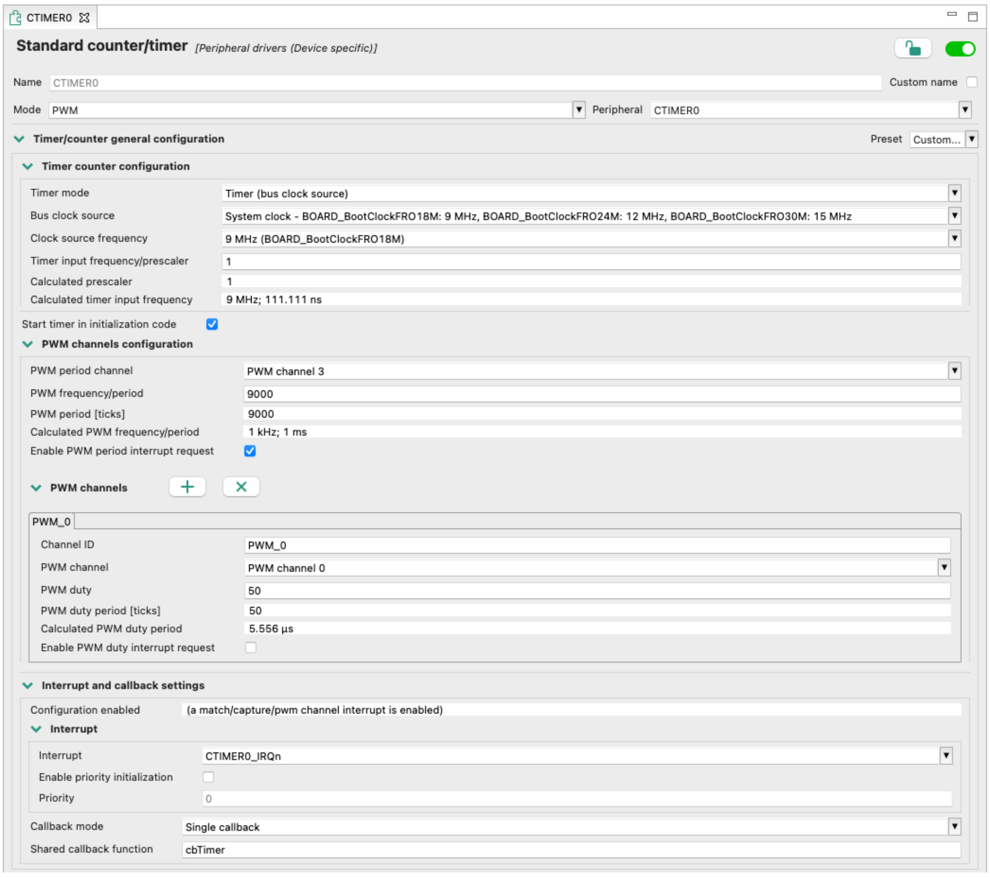
Routing Details for BOARD_InitPins

#	Peripheral	Signal	Arrow	Routed pin/sig	Label	Identifier	Direction	GPIO initial sta	Mode	Invert	Hysteresis	Open drain	DAC Mode
10	CTIMER0	MATCH, 0	->	[10] PIO0_11	CN6[10]/CN...	LED_BLUE	Output	n/a	PullUp	Disabled	Enabled	Disabled	n/a

8. Press *Done* in the individual dialog boxes and then *Update Code*.
9. Build a project, program the microcontroller and check the operation. The LED (blue) should change state 2 times per second (1 flash every second).

IV. CTIMER - PWM mode

1. Go to *Pheriperals* and change the configuration of CTIMER0 to PWM and set the values as below:



CTIMER0

Standard counter/timer *[Peripheral drivers (Device specific)]*

Name: CTIMER0 — Custom name □

Mode: PWM — Peripheral: CTIMER0

Timer/counter general configuration — Preset: Custom...

Timer counter configuration

Timer mode	Timer (bus clock source)
Bus clock source	System clock - BOARD_BootClockFRO18M: 9 MHz, BOARD_BootClockFRO24M: 12 MHz, BOARD_BootClockFRO30M: 15 MHz
Clock source frequency	9 MHz (BOARD_BootClockFRO18M)
Timer input frequency/prescaler	1
Calculated prescaler	1
Calculated timer input frequency	9 MHz; 111.111 ns

Start timer in initialization code ☑

PWM channels configuration

PWM period channel	PWM channel 3
PWM frequency/period	9000
PWM period [ticks]	9000
Calculated PWM frequency/period	1 kHz; 1 ms
Enable PWM period interrupt request	☑

PWM channels

PWM_0

Channel ID	PWM_0
PWM channel	PWM channel 0
PWM duty	50
PWM duty period [ticks]	50
Calculated PWM duty period	5.556 µs
Enable PWM duty interrupt request	□

Interrupt and callback settings

Configuration enabled: (a match/capture/pwm channel interrupt is enabled)

Interrupt

Interrupt	CTIMER0_IRQn
Enable priority initialization	□
Priority	0

Callback mode	Single callback
Shared callback function	cbTimer

2. Press *Update Code* and modify the program 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"

volatile uint8_t pwmDuty0=0;

void cbTimer(uint32_t flags) {

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

/*
 * @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

          EnableIRQ(CTIMER0_TIMER_IRQN); // Fix the BUG in Config Tools

          PRINTF("Start\r\n");

          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 ;
}
```

Adding the *EnableIRQ* function is necessary due to a bug in Config Tools (does not set the interrupt activation flag in the generated code).

Build a project, program the microcontroller and check the operation. Open the terminal and check the LED brightness control operation.

V. Exercises

1. Add an additional PWM channels (PWM_1 and PWM_2) to *CTIMER0*, connect its outputs to PIO0_12 (Green LED) and PIO0_13 (Red LED). Write a program to control LEDs brightness using the terminal. Send a mark:
   a: Blue PWM ++
   z: Blue PWM - -
   s: Green PWM ++
   x: Green PWM - -
   d: Red PWM ++
   c: Red PWM - -