

**Katedra Biosensorów i Przetwarzania Sygnałów
Medycznych
Wydział Inżynierii Biomedycznej**

POLITECHNIKA ŚLĄSKA



**SPRAWOZDANIE Z PRZEDMIOTU
Systemy Wbudowane i Mobilne w Biomedycynie**

Laboratorium 1

Sekcja 5: Joanna Chwał, Michał Letniowski, Bryan Mierzwa

Prowadzący: mgr inż. Marek Czerw

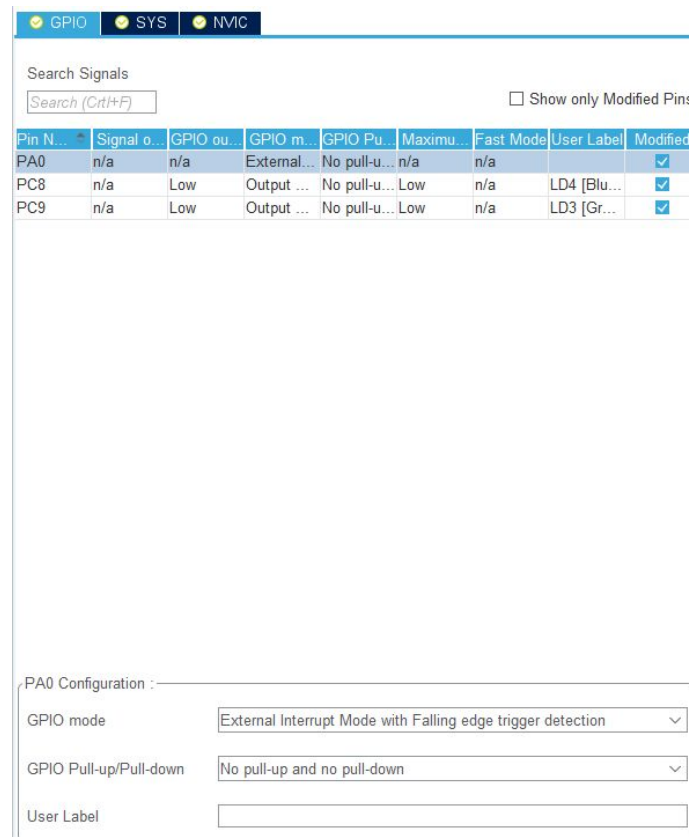
Sekcja: PiAIB

Zabrze 2020

1. Realizacja laboratorium

Celem laboratorium było załączenie świecenia obu diod Led jednocześnie z pomocą przycisku USER a następnie zgaszenie ich po 1 sekundzie.

Aby zrealizować cel laboratorium wykorzystano przerwanie zewnętrzne wywoływane na przyciśnięcie przycisku USER. Do wywołania przerwania zewnętrznego należało odpowiednio skonfigurować GPIO.



Rys. 1 Konfiguracja GPIO PA0 dla przerwania zewnętrznego

Efekt wciśnięcia przycisku USER jest spadek napięcia na linii PA0, w związku z tym ustawiono wywołanie przerwania zewnętrznego przy zboczu opadającym - **Falling edge trigger detection**.

W efekcie powyższego został wygenerowany następujący fragment kodu w pliku **main.c** obsługujący wywołanie przerwania zewnętrznego:

```
/* EXTI interrupt init*/
HAL_NVIC_SetPriority(EXTI0_1_IRQn, 0, 0);
HAL_NVIC_EnableIRQ(EXTI0_1_IRQn);
```

Rys. 2 Implementacja przerwania zewnętrznego w pliku main.c

Aby umożliwić sterowanie diodami dodano odpowiedni kod w pliku **main.c** w celu inicjalizacji obu diod.

```
/* Initialize all configured peripherals */
MX_GPIO_Init();
/* USER CODE BEGIN 2 */
BSP_LED_Init(LED_BLUE);
BSP_LED_Init(LED_GREEN);
/* USER CODE END 2 */
```

Rys. 3 Implementacja obu diod

Zapalenie diod oraz ich gaszenie zostało zaimplementowane w pliku **stm32f0xx_it.c**. Na wywołanie przerwania zewnętrznego, w pierwszej kolejności, zapalane są obie diody poprzez komendę **ON**, następnie jest ustawiany licznik na wartość 1001, co odpowiada 1001 ms czyli w przeliczeniu 1,001 s

```
void EXTI0_1_IRQHandler(void)
{
    /* USER CODE BEGIN EXTI0_1_IRQn 0 */
    BSP_LED_On(LED_BLUE);
    BSP_LED_On(LED_GREEN);
    TimingDelay = 1001;
    /* USER CODE END EXTI0_1_IRQn 0 */
    HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_0);
    /* USER CODE BEGIN EXTI0_1_IRQn 1 */

    /* USER CODE END EXTI0_1_IRQn 1 */
}
```

Rys. 4 Metoda przerwania zewnętrznego

Ponieważ częstotliwość taktowania STM32F030R8 wynosi 48 MHz, dzieląc to przez 1000 otrzymamy 48000 taktów, czyli czas w którym upłynie 1 ms:

```
/* Configure the system clock */
SysTick_Config(SystemCoreClock / 1000)
```

Rys. 5 Konfiguracja SysTick

Dzięki powyższej konfiguracji **SysTick** możemy wykorzystać licznik zliczający w dół od zadanej wartości przy wywołaniu przerwania (1001). W ten sposób co 1ms **SysTick_Handler** wywołuje metodę **TimingDelay_Decrement()**.

```

void SysTick_Handler(void)
{
    /* USER CODE BEGIN SysTick_IRQn 0 */
    TimingDelay_Decrement();

    /* USER CODE END SysTick_IRQn 0 */
    HAL_IncTick();
    /* USER CODE BEGIN SysTick_IRQn 1 */
    /* USER CODE END SysTick_IRQn 1 */
}

```

Rys. 6 Metoda obsługująca wywołanie SysTick

Metoda **TimingDelay_Decrement()** przy każdym wywołaniu sprawdza czy licznik jest różny od zera i jeśli jest to prawdą wykonuje operację dekrementacji. Na końcu działania funkcji sprawdzane jest czy następna dekrementacja spowoduje wyzerowanie licznika, jeżeli tak diody są gaszone.

```

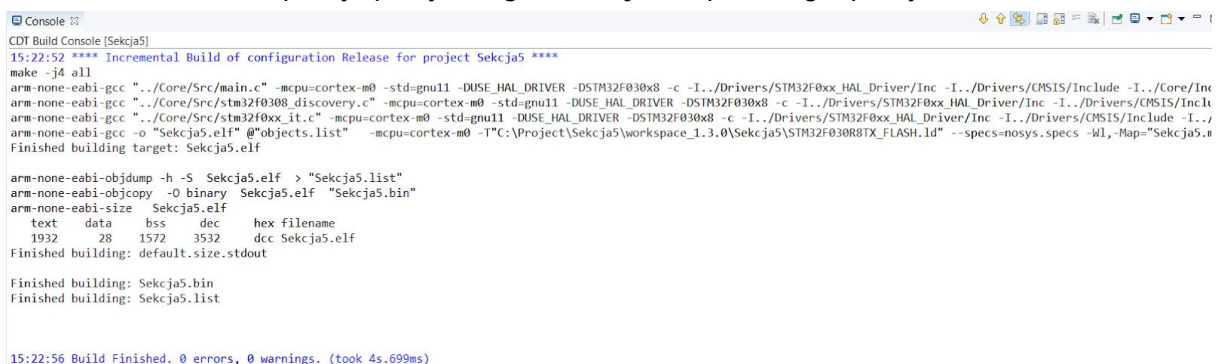
/* Private user code -----
/* USER CODE BEGIN 0 */
uint32_t TimingDelay = 0;

void TimingDelay_Decrement(void)
{
    if (TimingDelay != 0x00)
    {
        TimingDelay--;
        if(TimingDelay-1==0x00){
            BSP_LED_Off(LED_GREEN);
            BSP_LED_Off(LED_BLUE);
        }
    }
}
/* USER CODE END 0 */

```

Rys. 7 Metoda TimingDelay_Decrement

Budowa i kompilacja powyższego rozwiązania przebiegła pomyślnie.



```

Console
CDT Build Console [Sekcja5]
15:22:52 **** Incremental Build of configuration Release for project Sekcja5 ****
make -j4 all
arm-none-eabi-gcc -I../Core/Src/main.c -mcpu=cortex-m0 -std-gnu11 -DUSE_HAL_DRIVER -DSTM32F030x8 -c -I../Drivers/STM32F0xx_HAL_Driver/Inc -I../Drivers/CMSIS/Include -I../Core/Inc
arm-none-eabi-gcc -I../Core/Src/stm32f030x8_discovery.c -mcpu=cortex-m0 -std-gnu11 -DUSE_HAL_DRIVER -DSTM32F030x8 -c -I../Drivers/STM32F0xx_HAL_Driver/Inc -I../Drivers/CMSIS/Include
arm-none-eabi-gcc -I../Core/Src/stm32f0xx_it.c -mcpu=cortex-m0 -std-gnu11 -DUSE_HAL_DRIVER -DSTM32F030x8 -c -I../Drivers/STM32F0xx_HAL_Driver/Inc -I../Drivers/CMSIS/Include -I../
arm-none-eabi-gcc -o "Sekcja5.elf" @"objects.list" -mcpu=cortex-m0 -I"C:\Project\Sekcja5\workspace_1.3.0\Sekcja5\STM32F0308TX_FLASH.ld" --specs=nosys.specs -Wl,-Map="Sekcja5.map"
Finished building target: Sekcja5.elf

arm-none-eabi-objdump -h -S Sekcja5.elf > "Sekcja5.list"
arm-none-eabi-objcopy -O binary Sekcja5.elf "Sekcja5.bin"
arm-none-eabi-size Sekcja5.elf
   text    data     bss     dec      hex filename
   1932     28    1572    3532      dcc Sekcja5.elf
Finished building: default.size.stdout

Finished building: Sekcja5.bin
Finished building: Sekcja5.list

15:22:56 Build Finished. 0 errors, 0 warnings. (took 4s.699ms)

```

Rys. 8 Rezultat kompilacji programu