Systemy Wbudowane

Laboratorium 4:

Przerwania, liczniki i generowanie sygnałów PWM

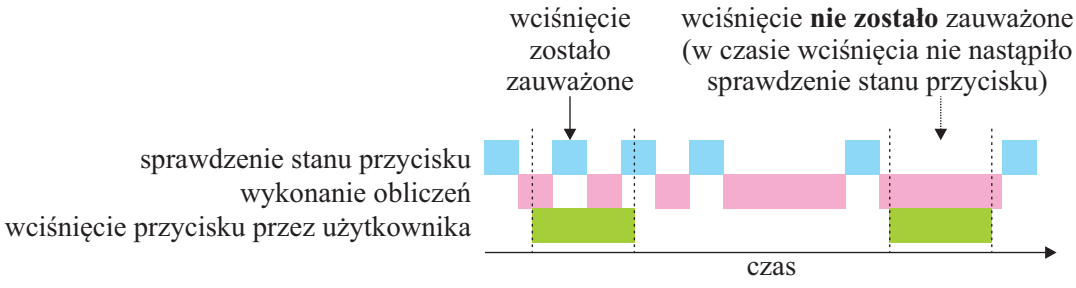
4.1 Metody obsługi zdarzeń

4.1.1 Obsługa przez zapytywanie

Obsługa zdarzeń przez zapytywanie (*ang. polling*) polega na **okresowym** sprawdzaniu statusu urządzeń zewnętrznych (np. przycisków) przez kontroler. Przedstawia to następujący przykład:

```
/* USER CODE BEGIN WHILE */
while (1)
{
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
    if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0) == GPIO_PIN_SET)
    {
        // zdarzenie gdy wykryto wcisniecie klawisza
    }
    Obliczenia(); // wykonanie pewnych obliczeń
}
```

W powyższym przykładzie w pętli jest naprzemiennie odczytywany stan przycisku, a następnie wykonywane są pewne obliczenia. Jeżeli obliczenia trwają krótko to stan przycisku będzie odczytywany na tyle często, że program powinien zareagować na wciśnięcie klawisza przez użytkownika. Jeżeli obliczenia będą trwały długo (dłużej niż czas wciśnięcia przycisku) to program może nie zdążyć odczytać stanu przycisku. Przedstawia to następujący rysunek:



Rysunek 4.1: Obsługa zdarzeń przez zapytywanie

Jednym ze sposobów rozwiązań problemów z obsługą przez zapytywanie jest podzielenie pozostałych obliczeń na mniejsze podzadania. Stosuje się także podejście, w którym program oczekuje na wciśnięcie klawisza i do tego czasu wstrzymuje swoje działanie. Takie podejście jest wykorzytsywane do obsługi prostych menu:

```
// pętla wykonywana do momentu wciśnięcia klawisza z pinu 0 (można oczekiwać na różne kombinacje klawiszy)
while((GPIOA->IDR & GPIO_IDR_IDR_0) == 0) { }
```

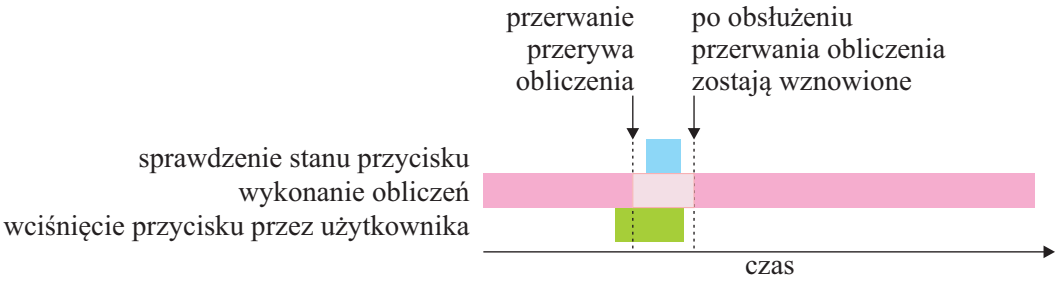
Problemem obsługi zdarzeń przez zapytywanie są też sytuacje w których zdarzenia powinny być wykonywane przez określony czas. Przykład przedstawiono poniżej:

```
if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0) == GPIO_PIN_SET)
{
    HAL_GPIO_WritePin(GPIOG, GPIO_PIN_13, GPIO_PIN_SET); // zapalenie diody
    HAL_Delay(1000); // odczekanie sekundy
    HAL_GPIO_WritePin(GPIOG, GPIO_PIN_13, GPIO_PIN_RESET); // zgaszenie diody
}
Obliczenia(); // wykonanie pewnych obliczeń
```

W powyższym przykładzie po wykryciu wciśnięcia przycisku zostaje zapalona dioda na czas 1 sekundy. Funkcja HAL_Delay(1000) zabiera przez tą sekundę cały czas procesora i nie jest możliwe wykonywanie innych niż przerwania poleceń aż do jej zakończenia.

4.1.2 Obsługa przez przerwania

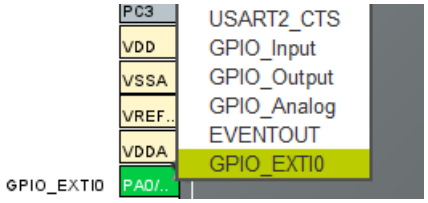
Obsługa zdarzeń przez przerwania (*ang. interrupt*) jest pozbawiona wymienionych powyżej wad. W tym przypadku występujące zdarzenia (np. wciśnięcie przycisku) wywołują przerwania, które zatrzymują działanie całego programu i oczekują na zakończenie ich obsługi:



Rysunek 4.2: Obsługa zdarzeń przez przerwania

Wyróżniamy przerwania wewnętrzne (nazywane wyjątkami i zgłaszane przez procesor dla sygnalizowania sytuacji wyjątkowych) oraz zewnętrzne (pochodzące z zewnętrznych układów - np. przycisków, liczników, zegarów).

Aby skonfigurować przerwanie z GPIO (np. z przycisku) należy odpowiednio skonfigurować wybrany pin:

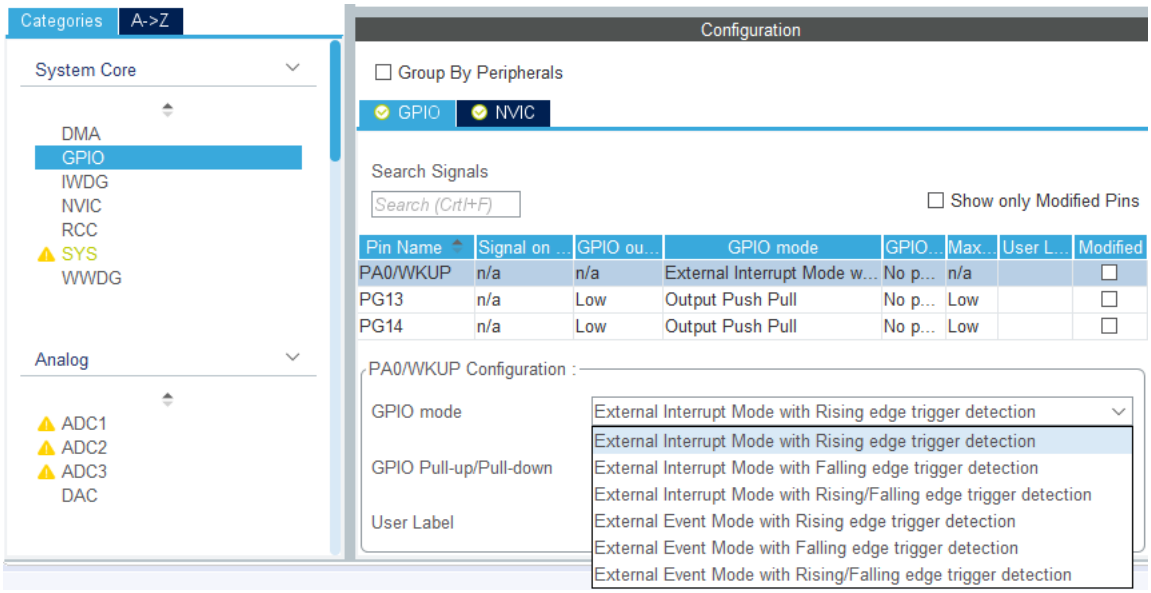


Rysunek 4.3: Konfiguracja pinu w celu obsługi przerwania

Na powyższym rysunku skrót EXTI oznacza *EXTernal Interrupt*, a 0 oznacza numer linii do obsługi przerwania. Płytki rozwojowe STM32F4 posiadają 16 linii obsługi przerwań z GPIO. Przerwania GPIO mogą zostać generowane dla zdarzeń związanych z:

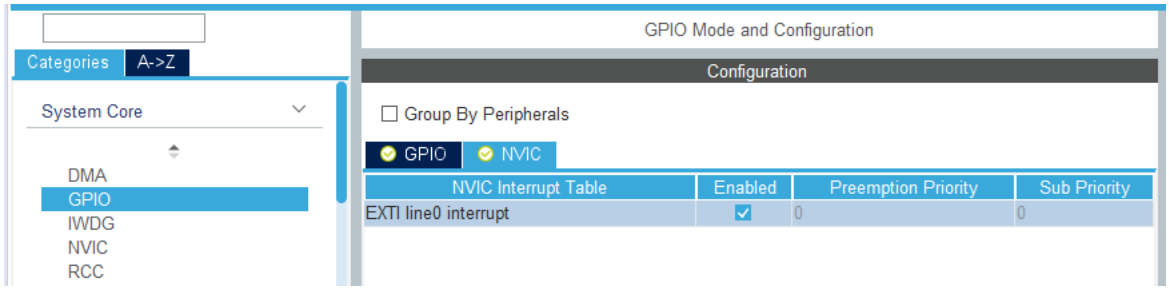
- przejściem stanu linii na stan wysoki (*ang. Rising*) [domyślnie]
- przejściem stanu linii na stan niski (*ang. Falling*)
- przejściem stanu linii na stan wysoki lub niski (*ang. Rising/Falling*)

Konfiguracja obsługi przerwań dostępna jest w sekcji GPIO:



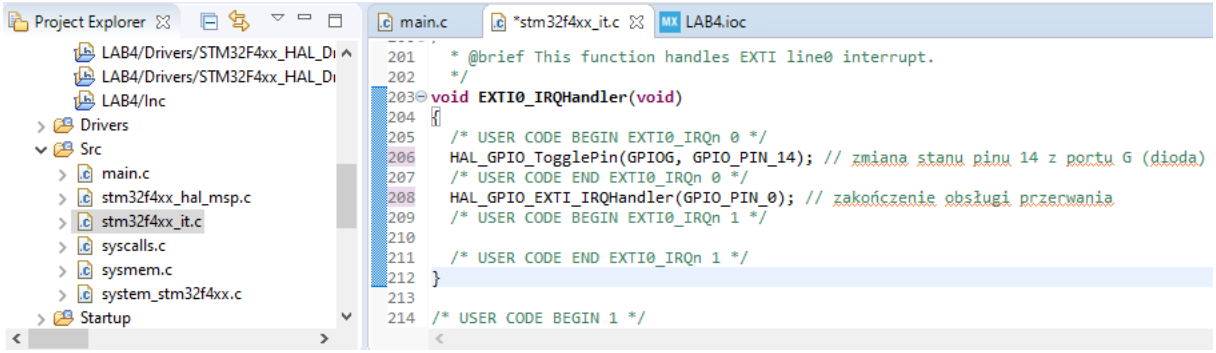
Rysunek 4.4: Szczegółowa konfiguracja przerwań GPIO

Aby automatycznie obsłużyć przerwanie należy jeszcze uaktywnić obsługę przerwania:



Rysunek 4.5: Aktywacja obsługi przerwania

Po wygenerowaniu kodu (Alt+K) metoda obsługi przerwań zostaje wygenerowana w pliku *stm32f4xx_it.c*



Rysunek 4.6: Metoda obsługująca przerwanie

W powyższym przykładzie przerwanie wywołane przez wciśnięcie przycisku zmienia stan diody podpiętej do linii PG13 (oczywiście należy linię PG13 skonfigurować jako wyjście). W obsłudze przerwania warto również sprawdzić czy został wciśnięty dany lub konkretny przycisk. Należy także wspomnieć że przerwania z linii 5-9 oraz 10-15 obsługiwane są za pomocą wspólnych metod: *EXTI9_5_IRQHandler* oraz *EXTI15_10_IRQHandler*.

4.2 Liczniki

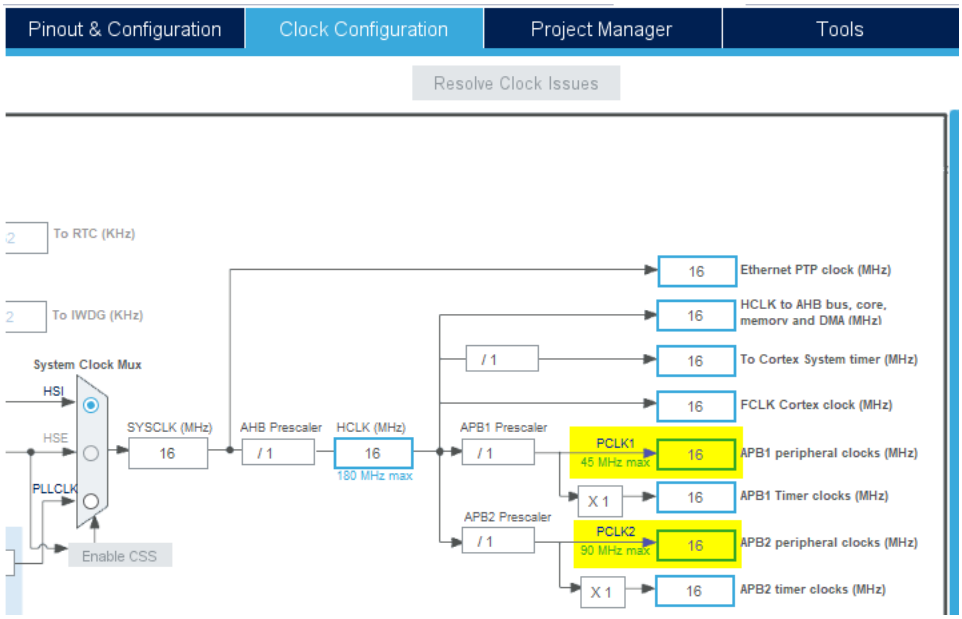
4.2.1 Podstawowa konfiguracja

Liczniki to układy cyfrowe zliczające i pamiętające ilości impulsów podawanych na ich wejścia zliczające. Płytki STM32F429ZI posiadają 14 niezależnie konfigurowalnych liczników TIM1-TIM14. Ich szczegółowy opis można znaleźć w instrukcjach obsługi płytki STM32F429ZI i jej specyfikacji:

Tabela 4.1: Wybrana charakterystyka liczników

Type	Timer	Resolution	Counter	DMA request	Capture channels	Comple--mentary output	Base clock
Advanced--controll	TIM1, TIM8	16-bit	Up/down	Yes	4	Yes	PCLK2
General purpose	TIM2, TIM5	32-bit	Up/down	Yes	4	No	PCLK1
	TIM3, TIM4	16-bit	Up/down	Yes	4	No	PCLK1
	TIM9	16-bit	Up	No	2	No	PCLK2
	TIM10, TIM11	16-bit	Up	No	1	No	PCLK2
	TIM12	16-bit	Up	No	2	No	PCLK1
	TIM13, TIM14	16-bit	Up	No	1	No	PCLK1
Basic	TIM6, TIM7	16-bit	Up	Yes	0	No	PCLK1

Jak widać w powyższej tabeli, bazowym zegarem liczników jest sygnał PCLK1 lub PCLK2. Wartości te można odnaleźć w konfiguracji zegarów:

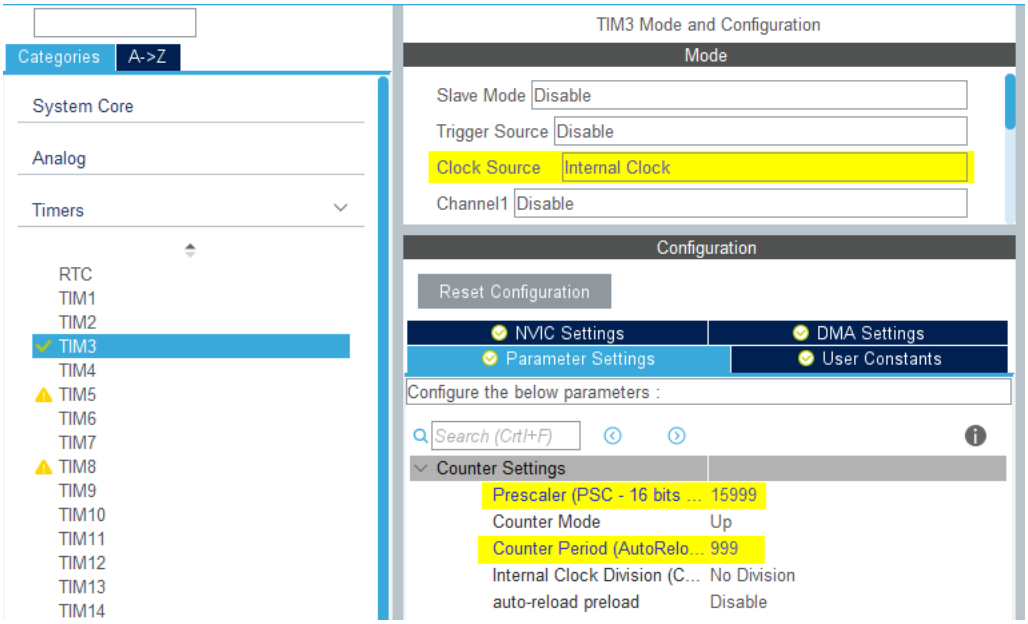


Rysunek 4.7: Konfiguracja zegarów

Przy domyślnej konfiguracji zegarów licznik modyfikowałby (domyślnie zwiększał) swoją wartość 16000000 razy na sekundę (PCLK1 = PCLK2 = 16MHz). Można tą wartość odpowiednio zmniejszyć ustawiając dzielnik PSK danego licznika. Dzielniki wszystkich liczników w STM32F429ZI są 16 bitowe, a zatem można je ustawić na wartość od 1 do 65536. Łatwo zauważyć że ustawiając wartość dzielnika na 15999 (16000 zliczeń od 0 do 15999), licznik będzie modyfikował swoją wartość już tylko 1000 razy na sekundę (co jedną milisekundę). Liczniki posiadają także wartość Counter Period (CP), a zatem wartość przy osiągnięciu której zaczynają liczyć od nowa, wartość CKD (wewnętrznego dzielnika) i wiele różnych innych opcji konfiguracji. Przy zegarze PCLK1 = 16MHz, dzielniku PSK 15999 i wartości CP 999 (1000 zliczeń od 0 do 999), licznik będzie zaczynał liczenie od nowa co sekundę.

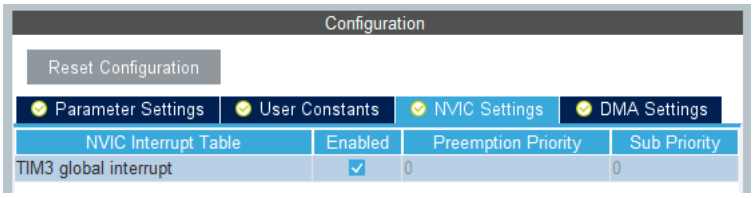
4.2.2 Liczniki a przerwania

Aby skonfigurować licznik należy wybrać źródło zegara i wartość PSC, a także opcjonalnie Counter Period:



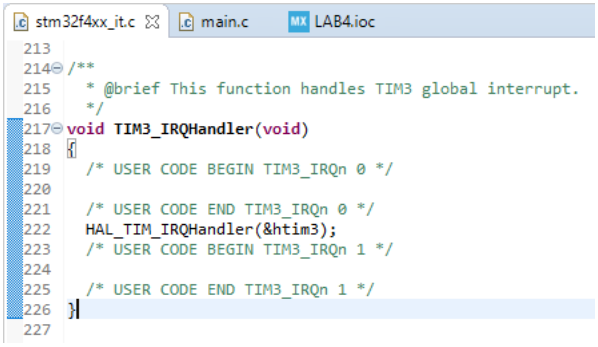
Rysunek 4.8: Konfiguracja licznika TIM3

Po osiągnięciu zadanej wartości Counter Period licznik może wywołać funkcję przerwania, aby to zrobić wystarczy aktywować przerwanie w zakładce *NVIC Settings* licznika:



Rysunek 4.9: Aktywacja wywoływania przerwań przez licznik

Przy powyższej konfiguracji i wygenerowaniu kodu, w pliku projektu *stm32f4xx_it.c* zostanie umieszczona funkcja *TIM3_IRQHanlder(void);*:



Rysunek 4.10: Funkcja obsługi przerwań wywołanych przez licznik TIM3

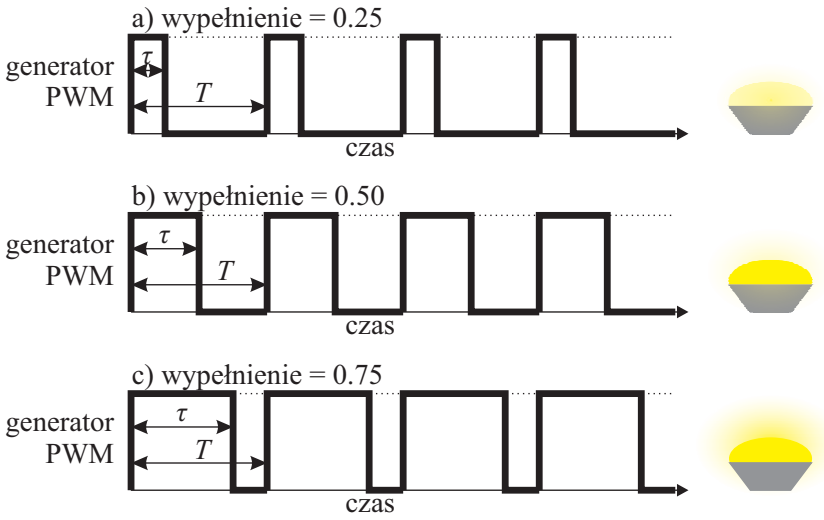
UWAGA: sama konfiguracja licznika nie uruchomi zliczania implusów, należy jeszcze w metodzie main (po konfiguracji licznika) wywołać *metodę HAL_TIM_Base_Start(&htim3);* lub odpowiednio *HAL_TIM_Base_Start_IT(&htim3);* jeżeli chce się uruchomić licznik wywołujący przerwania.

4.3 PWM

4.3.1 Generowanie sygnału PWM

Generatory PWM (*ang. Pulse-Width Modulation* - modulacja szerokości impulsów) generują sygnał o okresie T przez pewien czas τ (*ang. Pulse*) utrzymując stan wysoki sygnału (impuls). Współczynnik wypełnienia takiego sygnału oblicza się następująco: $k_w = \frac{\tau}{T}$.

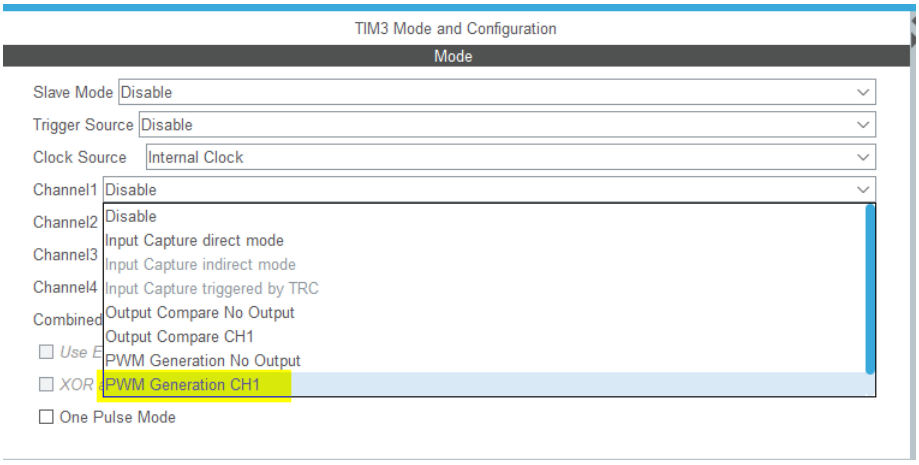
Jeżeli okres T jest dostatecznie krótki, to w zależności od współczynnika wypełnienia można sterować różnymi urządzeniami dostarczając im sygnał PWM. Pokazuje to następujący obrazek (w analogiczny sposób można sterować silnikami elektrycznymi, dźwiękiem, itp.):



Rysunek 4.11: Zmiana wypełnienia i jej wpływ na świecenie żarówką (efekt wygaszania i zapalania żarówki jest niewidoczny dla ludziego oka przy niskiej wartości T).

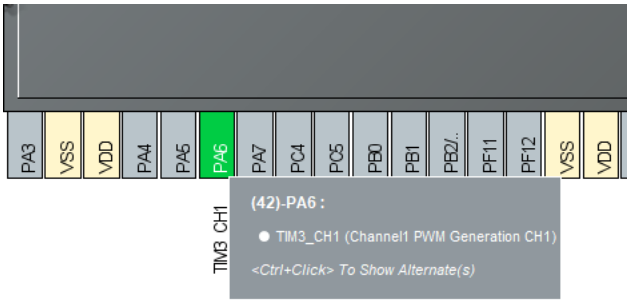
4.3.2 Generowanie sygnału PWM przez liczniki

Sygnał PWM można generować "ręcznie" zmieniając odpowiednio często stany poszczególnych wyjść, jednak o wiele lepszym rozwiązaniem jest wykorzystanie do tego liczników. Konfiguracja polega na wybraniu odpowiedniego kanału i ustalenia jego przeznaczenia jako PWM. Można także automatycznie wystawić wyjście sygnału PWM na linię powiązaną z danym kanałem:



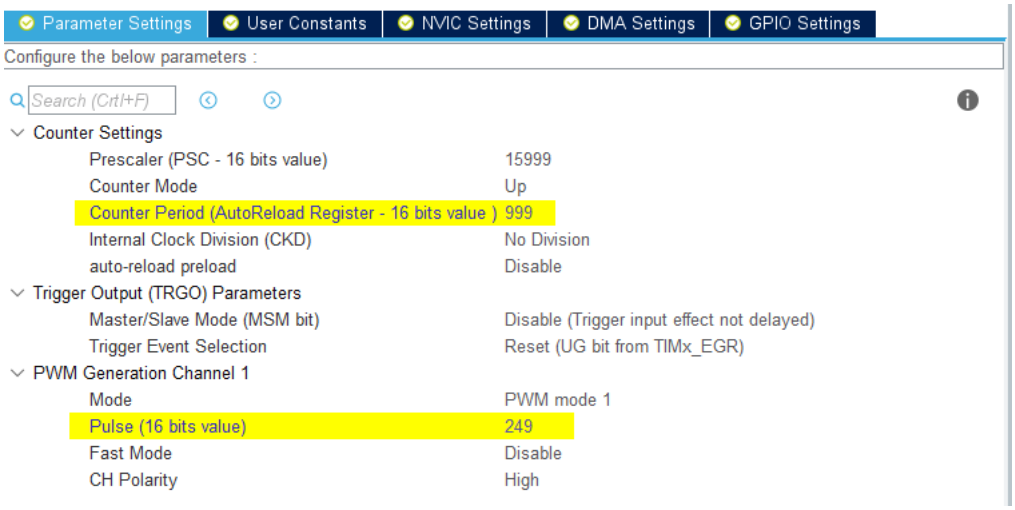
Rysunek 4.12: Generowanie sygnału PWM

Po takiej konfiguracji automatycznie zostanie skonfigurowany pin, na którym będzie generowany dany sygnał (w przypadku TIM3 i CH1-Channel1 będzie to PA6):



Rysunek 4.13: PIN z generowanym sygnałem PWM

Należy jeszcze uwzględnić kwestię wypełnienia sygnału poprzez ustalenie wartości τ. W konfiguratorze można tą wartość ustawić zmieniając wartość pola Pulse (należy pamiętać że w tym przypadku czas okresu T odpowiada wartości Counter Period i przez to τ musi być mniejsza od T):



Rysunek 4.14: Ustawienie wartości T (Counter Period) oraz τ (Pulse)

Po konfiguracji licznika należy jeszcze w funkcji main (po konfiguracji) wywołać funkcję rozpoczynającą generowanie sygnału PWM ustawiając odpowiedni licznik oraz kanał:
HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_1);

Należy pamiętać, że aby czas T był odpowiednio krótki należy zmniejszyć wartość dzielnika licznika. Okres T odpowiada czasowi w którym licznik osiągnie wartość zadaną (Counter Period). Przy 16MHz np. wartość dzielnika równa 15 (liczenie co 16 impuls - od 0 do 15) spowoduje że przy Counter Period = 999 (liczenie od 0 do 999 - 1000 wartości) okres T będzie wynosił jedną milisekundę (16MHz / 16 / 1000 = 1kHz).

4.3.3 Zmiana wypełnienia sygnału

Aby zmienić wypełnienie sygnału w dowolnym czasie za pomocą bibliotek HAL, należy wykorzystać metodę *HAL_TIM_PWM_ConfigChannel(&htim3, &config, CHANNEL)* wywołując ją podobnie jak ma to miejsce w wygenerowanej procedurze *MX_TIM3_Init(void);* w pliku *main.c*.

Można także wykonać to szybciej, zmieniając wartość rejestru CCR1 danego licznika, np:
TIM3->CCR1 = 100; // Pulse = 100

4.4 Ćwiczenie 1

Celem ćwiczenia jest napisanie programu wykorzystującego przerwania. W celu realizacji ćwiczenia należy:

- Stworzyć nowy projekt o nazwie LAB4A dla otrzymanej płytki rozwojowej.
- Skonfigurować piny diód LD3 i LD4 jako GPIO_Output
- Skonfigurować przycisk użytkownika (PA0), tak aby jego wciśnięcie generowało przerwanie na podstawie przykładu umieszczonego w instrukcji w punkcie 4.1.2
- Skonfigurować wybrany przez siebie licznik tak, aby co 500 milisekund było generowane przerwanie na podstawie przykładu umieszczonego w instrukcji w punkcie 4.2.2
- Wygenerować kod projektu i zmodyfikować procedury obsługi przerwań tak, aby przerwanie wywołane wciśnięciem przycisku zmieniało stan diody LD3, natomiast przerwanie wywołane przez licznik zmieniało stan diody LD4
- Skompilować kod programu, wgrać na płytkę i przetestować jego działanie. Napisać wnioski dotyczące działania programu w taki sposób (w porównaniu ze zmianą stanu diod z laboratorium nr. 1)

4.5 Ćwiczenie 2

Celem ćwiczenia jest napisanie programu generującego sygnał PWM. W celu realizacji ćwiczenia należy:

- Zgłosić się do prowadzącego po diodę lub inny element możliwy do sterowania za pomocą PWM
- Stworzyć nowy projekt o nazwie LAB4B dla otrzymanej płytki rozwojowej
- Skonfigurować wybrany przez siebie licznik tak, aby okres T generowania sygnału wynosił około 50kHz (odpowiednie obliczenia zamieścić w sprawozdaniu)
- Odnaleźć pin na którym generowany jest sygnał PWM i podpiąć do niego dane urządzenie
- Napisać kod programu tak, aby program stopniowo zwiększał wypełnienie sygnału odczekując za każdym razem 10 milisekund (można to zrobić w pliku *main.c* i nieskończonej pętli, albo za pomocą innego licznika i obsługi przerwań). Stopniowe zwiększanie wypełnienia sygnału powinno płynnie zwiększać dostrzegalną jasność świecenia diody
- Skompilować kod programu, wgrać na płytkę i przetestować jego działanie
- W przypadku diody opcjonalnie sprawdzić jaki okres T powoduje że człowiek przestaje zauważyć zmiany stanu diody. W sprawozdaniu zamieścić odpowiednie wnioski

4.6 Ćwiczenie 3 (opcjonalne)

Celem ćwiczenia jest odczytanie wartości licznika. W celu realizacji ćwiczenia należy:

- Zmodyfikować program z ćwiczenia pierwszego tak, aby po wciśnięciu przycisku odczytać aktualną wartość licznika (właściwy rejestr lub metodę z biblioteki HAL należy odnaleźć samodzielnie)
- Zastanowić się czy odczytaną wartość można traktować jako wartość losową