POLITECHNIKA POZNAŃSKA

WYDZIAŁ AUTOMATYKI, ROBOTYKI I ELEKTROTECHNIKI

INSTYTUT ROBOTYKI I INTELIGENCJI MASZYNOWEJ

ZAKŁAD STEROWANIA I ELEKTRONIKI PRZEMYSŁOWEJ

SPRAWOZDANIE

SYSTEMY MIKROPROCESOROWE (LABORATORIUM)
[WARIE_2021-22_AIR_DZ_1_5_D_LUCZAK_21/22]

PROGRAMOWALNE LICZNIKI
(TIM, UKŁAD FAZOWEGO STEROWANIA ŻARÓWKĄ)

(TEMAT ZAJĘĆ)

KAROL DĘBSKI

(AUTOR I: KAROL.DEBSKI@STUDENT.PUT.POZNAN.PL)

FORMA ZAJĘĆ: LABORATORIUM

PROWADZĄCY:

DR INŻ. DOMINIK ŁUCZAK

DOMINIK.LUCZAK@PUT.POZNAN.PL

POZNAŃ *27-10-2021* 16-50

(*DATA I GODZINA ZAJĘĆ*)

Spis treści

Zadanie #1

1.1 Specyfikacja

Program co 500 ms zmienia stan diody z wykorzystaniem licznika. Do weryfikacji poprawnego działania programu zostanie użyte narzędzie SWV.

1.2 Implementacja

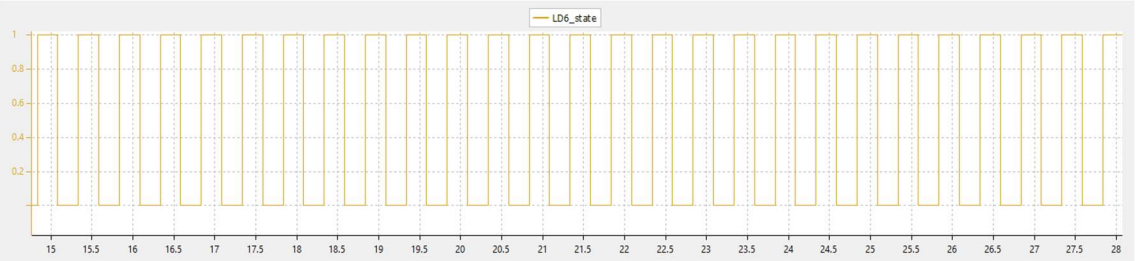
Listing 1 Główna pętla programu

```
while (1)
  {
          if(__HAL_TIM_GET_FLAG(&htim1,TIM_FLAG_UPDATE)){
                    __HAL_TIM_CLEAR_FLAG(&htim1,TIM_FLAG_UPDATE);
                    HAL_GPIO_TogglePin(LD6_GPIO_Port, LD6_Pin);
                    LD6_state = HAL_GPIO_ReadPin(LD6_GPIO_Port, LD6_Pin);
          }
    /* USER CODE END WHILE */
    MX USB HOST Process();

    /* USER CODE BEGIN 3 */
  }
```

Po inicjalizacji timera, program sprawdza czy timer zwraca flage doliczenia do założonego czasu. Jeśli tak to resetuje tą flagę i przełącza stan LED na przeciwny do poprzedniego, dodatkowo jest zapisywany stan diody.

1.3 Wynik testów



Rys. 2 Podgląd SWV

1.4 Wnioski

Program działa poprawnie, odstępy czasu pomiędzy włączeniem a wyłączeniem diody są na oko inżynierskie równe 500 ms.

Zadanie #2

1.1 Specyfikacja

Program przełącza stan diody z poprzedniego na przeciwny co 1 sekunde z użyciem przerwania na liczniku. Do weryfikacji poprawnego działania programu zostanie użyte narzędzie SWV.

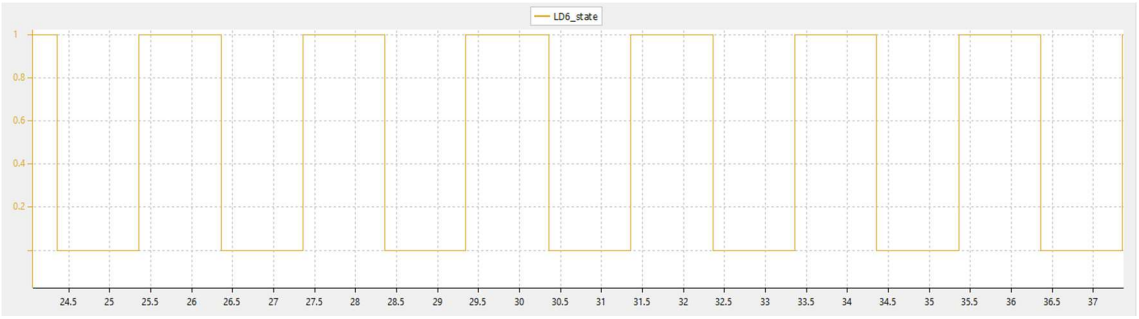
1.2 Implementacja

Listing 2 Funkcja zwrotna dla przerwania na TIM10

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
	if(htim -> Instance == TIM10)
	{
		HAL_GPIO_TogglePin(LD6_GPIO_Port, LD6_Pin);
		LD6_state = HAL_GPIO_ReadPin(LD6_GPIO_Port, LD6_Pin);
	}
}
```

Po inicjalizacji timera, program działa w nieskończonej pętli w oczekiwaniu na przerwanie z TIM10. Po wywołaniu przerwania uruchamiana jest funkcja zwrotna która sprawdza czy przerwanie zostało zainicjowane na timerze TIM10. Jeśli tak to program zmienia stan diody na przeciwny do poprzedniego. Następnie zapisywany jest stan diody.

1.3 Wynik testów



Rys. 2 Podgląd SWV

1.4 Wnioski

Program działa poprawnie, diody przełącza się zgodnie z założeniem o 1 sekunde.

Zadanie #3

1.1 Specyfikacja

Program za pomocą dwóch przycisków dostraja okres timera za pomocą przerwań na nich. Do weryfikacji poprawnego działania programu zostanie użyte narzędzie SWV.

1.2 Implementacja

Listing 3 Funkcja zwrotna dla przerwania na TIM10

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
        if(htim->Instance == TIM10){
                HAL GPIO TogglePin(LD4 GPIO Port, LD4 Pin);
                state_LED=HAL_GPIO_ReadPin(LD4_GPIO_Port, LD4_Pin);
        }
}
```

Listing 4 Funkcja zwrotna dla przerwania na dwóch przyciskach

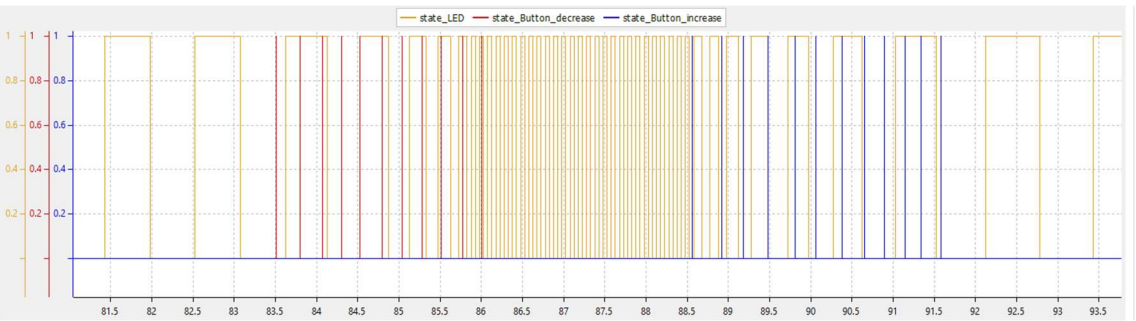
```
void HAL_GPIO_EXTI_Callback ( uint16_t GPIO_Pin ){
        if(GPIO_Pin==Button_decrease_Pin){
                state Button decrease=1;
                if(period_of_timer>1000){
                        period of timer-=1000;
                }
                state_Button_decrease=0;
        }
        if(GPIO_Pin==Button_increase_Pin){
                state Button increase=1;
                if(period_of_timer<64535){
                        period_of_timer+=1000;
                }
                state_Button_increase=0;
        }
}
```

Listing 5 Główna pętla programu

```
while (1)
  {
      HAL TIM SET AUTORELOAD(&htim10, period of timer);
    /* USER CODE END WHILE */
    MX_USB_HOST_Process();
    /* USER CODE BEGIN 3 */
  }
```

Gdy dochodzi do przerwania na timerze TIM10 jest zmieniany stan diody na przeciwny do poprzedniego. Zapisywany jest także stan diody (Listing 3). Gdy zostanie wywołane przerwanie na przycisku Button_decrease, wartość zmiennej globalnej period_of_timer zmiejsza się (Listing 4). Gdy przerwanie zostanie wywołane na przycisku Button_increase, wartość zmiennej globalnej period_of_timer zwiększa się (Listing 4). Podczas działania programu w nieskończonej pętli while ustawiana jest wartość rejestru AutoReload timera z wykorzystaniem zmiennej globalnej period_of_timer. Ta manipulacja ma bezpośredni wpływ na wartość okresu timera (Listing 5).

1.3 Wynik testów



Rys. 3 Podgląd SWV

1.4 Wnioski

Z wykresu wynika że gdy przycisk od zmniejszania okresu timera jest wciskany to okres się zmniejsza a gdy przycisk od zwiększania okresu timera jest wciskany to zwiększa się okres. Wniosek jest jeden – program działa poprawnie.

Zadanie #4

1.1 Specyfikacja

Program uruchamia timer gdy zostanie wciśnięty odpowiedni przycisk. Do regulacji okresu timera będą odpowiadać 2 przyciski. Do weryfikacji poprawnego działania programu zostanie on uruchomiony na mikrokontrolerze podczas kręcenia nagrania video.

1.2 Implementacja

Listing 6 Główna pętla programu

```
while (1)
  {
    if(Button_set_timer_ && !timer_is_on){
          HAL_TIM_Base_Start_IT(&htim10);
          timer_is_on=1;
    }
    __HAL_TIM_SET_AUTORELOAD(&htim10, period_of_timer);
    /* USER CODE END WHILE */
    MX USB HOST Process();
    /* USER CODE BEGIN 3 */
  }
```

Listing 7 Funkcja zwrotna przerwania na timerze

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
          if(htim->Instance == TIM10){
                    HAL_GPIO_TogglePin(LD4_GPIO_Port, LD4_Pin);
                    state_LED=HAL_GPIO_ReadPin(LD4_GPIO_Port, LD4_Pin);
          }
}
```

Listing 8 Funkcja zwrotna przerwania na 3 przyciskach

```
void HAL_GPIO_EXTI_Callback ( uint16_t GPIO_Pin ){
          if(GPIO_Pin==Button_decrease_Pin){
                    state Button decrease=1;
                    if(period_of_timer>1000){
                              period_of_timer-=1000;
                    }
                    state Button decrease=0;
          }
          if(GPIO_Pin==Button_increase_Pin){
                    state_Button_increase=1;
                    if(period_of_timer<64535){
                              period_of_timer+=1000;
                    }
                    state Button increase=0;
          }
          if(GPIO_Pin==Button_set_timer_Pin){
                    if(Button_set_timer_==0){
                              Button_set_timer_=1;
                    }
          }
}
```

Po wciśnięciu przycisku Button_set_timer_Pin uruchamiana jest procedura inicjalizacji timera, dochodzi do tego tylko raz (Listing 6, 8). Gdy dochodzi do przerwania na timerze TIM10 jest zmieniany stan diody na przeciwny do poprzedniego. Zapisywany jest także stan diody (Listing 7). Gdy zostanie wywołane przerwanie na przycisku Button_decrease, wartość zmiennej globalnej period_of_timer zmiejsza się (Listing 8). Gdy przerwanie zostanie wywołane na przycisku Button_increase, wartość zmiennej globalnej period_of_timer zwiększa się (Listing 8). Podczas działania programu w nieskończonej pętli while ustawiana jest wartość rejestru

AutoReload timera z wykorzystaniem zmiennej globalnej period_of_timer. Ta manipulacja ma bezpośredni wpływ na wartość okresu timera (Listing 6).

1.3 Wynik testów

https://drive.google.com/file/d/1q-kAB2m8uLv7aTqkieccn9iFNS4r5Zp6/view?usp=sharing

(25 sekund)

1.4 Wnioski

Program działa poprawnie, timer uruchamia się po przyciśnięciu przycisku a jego okres można modyfikować kolejnymi dwoma przyciskami.

Podsumowanie

Zadanie 5 i 6 nie zostało ukończone ze względu na brak płytki pcb z żarówką. Zadanie 4 wymagało nagrania filmu ze względu na bug związany z SWV. Wartość zmiennej sięgała w górę do wartości 200 gdzie jej wartość była albo 0 albo 1. Powodowało to zawieszanie procesu debugowania.