Architektura systemów komputerowych

Podczas realizacji poniższych zadań, oprócz instrukcji/dokumentacji podanych na końcu ćwiczeń (sekcja „Materiały"), można korzystać z dowolnych źródeł internetowych. Szczególnie polecany przez prowadzącego jest kurs dostępny na stronie: https://forbot.pl/blog/kurs-stm32-l4-wstep-spis-tresci-dla-kogo-jest-ten-kurs-id48575.

Uwaga: Po wykonaniu każdego zadania należy zapisać, kody i inne informacje do sprawozdania (zrzuty ekranu, odpowiedzi na pytania), o ile są potrzebne w danym zadaniu. Po wykonaniu wszystkich zadań niniejszej instrukcji należy powiadomić prowadzącego.

ARM 3/3

TIM, PWM

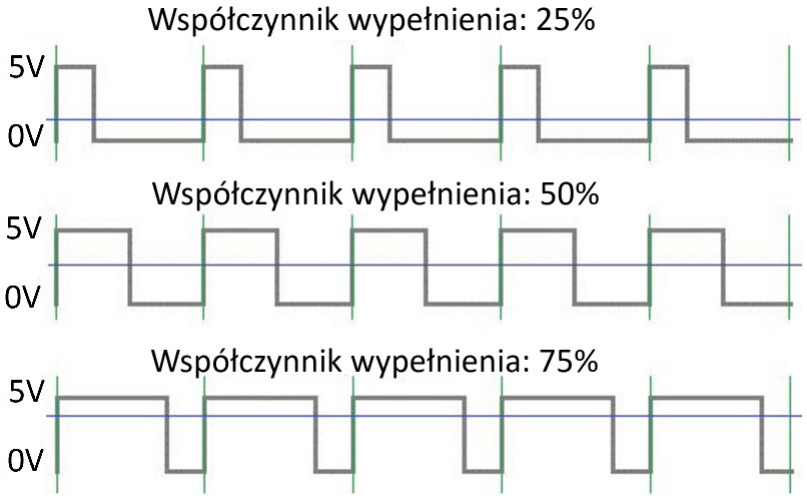
Wstęp

Liczniki sprzętowe są wbudowanymi peryferiami odpowiedzialnymi za różne operacje związane z odmierzaniem czasu, generowaniem sygnałów PWM (Pulse Width Modulation), odmierzaniem impulsów, pomiarem częstotliwości oraz innymi funkcjami związanymi z czasem.

W STM32 istnieją różne typy liczniki:

1. **Liczniki podstawowe (Basic Timers – TIM6, TIM7)**
    - Używane głównie do generowania przerwań w określonych odstępach czasu.
    - Mogą być wykorzystywane np. w systemach czasu rzeczywistego do odmierzania czasu.
2. **Liczniki ogólnego przeznaczenia (General-purpose Timers – np. TIM2, TIM3, TIM4, TIM5)**
    - Mogą pracować w trybie licznikowym, PWM, przechwytywania zdarzeń (input capture) oraz generowania impulsów.
3. **Liczniki zaawansowane (Advanced-control Timers – np. TIM1, TIM8)**
    - Oprócz standardowych funkcji liczników ogólnego przeznaczenia, oferują one dodatkowe opcje związane z kontrolą silników oraz zaawansowaną generacją PWM.
    - Często wykorzystywane w sterowaniu silnikami.
4. **Liczniki watchdog (IWDG – Independent Watchdog, WWDG – Window Watchdog)**
    - Zapobiegają zawieszeniu się systemu poprzez resetowanie mikrokontrolera, jeśli nie zostaną odświeżone w określonym czasie.
5. **Licznik SysTick**
    - Specjalny 24-bitowy licznik używany głównie do odmierzania czasu w systemach operacyjnych (np. w FreeRTOS do planowania zadań).

Sygnał PWM to przebieg prostokątny, w którym zmienia się szerokość impulsu (czas trwania stanu wysokiego) przy stałej częstotliwości. Współczynnik wypełnienia (ang. duty cycle) sygnału PWM to parametr określający procentowy stosunek czasu trwania stanu wysokiego do całkowitego okresu sygnału. Poniższy rysunek przedstawia trzy sygnały PWM o identycznej częstotliwości, z różnymi współczynnikami wypełnienia. Jeśli stan wysoki PWM odpowiada zapalonej diodzie, to za pomocą współczynnika wypełnienia możemy kontrolować jej jasność świecenia (im wyższy współczynnik, tym jaśniej dioda świeci).



Przebieg ćwiczenia

1. W projekcie z poprzednich zajęć aktywuj w konfiguratorze licznik TIM6 (kurs Forbot - lekcja #8). Chcemy, żeby licznik generował przerwanie co jedną sekundę. Ustawiamy więc wartości **Prescaler** na **16999** oraz **Counter Period** na **9999**. Skąd biorą się te wartości? Wyjaśnij. Ustaw możliwość generowania przerwań od licznika w ustawieniach NVIC. Zmniejsz priorytet przerwań **TIM6** z 0 (wartość domyślna) do **10**. Należy pamiętać, że im wyższy numer, tym niższy priorytet.
2. Za pomocą licznika skonfigurowanego w poprzednim zadaniu zrealizuj miganie diody **LD2** z częstotliwością 2Hz.
   Podpowiedź: Wystarczy uruchomić licznik skonfigurowany w ramach poprzedniego zadania i w funkcji obsługi przerwań zmieniać stan diody (po sprawdzeniu, czy przerwanie pochodzi od licznika **TIM6**). Licznik ten generuje przerwania co jedną sekundę, co jest zgodne z częstotliwością migania 2Hz.
   Podpowiedź 2: Nazwę uchwytu licznika można odczytać w kodzie automatycznie wygenerowanej funkcji `MX_TIM6_Init`.
3. W widoku pinów (Pinout view) ustaw pinowi **PA5** (dioda) tryb **TIM2_CH1**, aby uzależnić go od licznika **TIM2**. Aktywuj licznik **TIM2**: ustaw tryb **Clock Source** na **Internal Clock** i **Channel1** na **PWM Generation CH1** (kurs Forbot - lekcja #8). Licznik powinien pracować z częstotliwością 100Hz.
   Podpowiedź: Aby uzyskać częstotliwość 100Hz należy zmniejszyć wartość **Prescaler** 100-krotnie (i obciąć część ułamkową) w stosunku do licznika **TIM6** z zad. 1 i 2.
4. Zrealizuj zwiększanie jasności świecenia diody za pomocą przycisku. W tym celu należy zmieniać wartości współczynnika wypełnienia sygnału PWM (Pulse Width Modulation) generowanego przez licznik **TIM2**. Przyciśnięcie (i puszczenie) przycisku powinno zwiększyć współczynnik wypełnienia o 300 (3%), startując od wartości 100 (1%). Po osiągnięciu wartości 2000 (20%) należy przypisać początkową wartość 100.
   Podpowiedź: W konfiguratorze współczynnik wypełnienia możemy ustalić na stałe ustawiając wartość Pulse (w sekcji PWM Generation Channel 1), natomiast jeśli chcemy zaprogramować jego zmianę, to musimy wykorzystać funkcję `__HAL_TIM_SET_COMPARE`.
   Podpowiedź 2: Należy jednorazowo uruchomić generator PWM za pomocą funkcji `HAL_TIM_PWM_Start`.

5. Wysyłaj co jedną sekundę (za pomocą licznika z zad. 1) wartość temperatury wewnętrznej układu zmierzonej przez czujnik wbudowany w mikrokontroler.
   W tym celu najpierw należy aktywować czujnik temperatury w konfiguratorze:
   Categories -> Analog -> ADC1 -> Mode -> Temperature Sensor Channel (zaznaczamy checkbox)
   Następnie należy zwiększyć częstotliwość próbkowania przetwornika:
   ADC1 -> Configuration -> Parameter Settings -> ADC_Regular_ConversionMode -> Rank -> Sampling Time: 247.5 Cycles.
   Aby program mógł poprawnie mierzyć temperaturę należy dokonać kalibracji ADC (przetwornika analogowo-cyfrowego) jednorazowo, za pomocą instrukcji:
   `HAL_ADCEx_Calibration_Start(&hadc1, ADC_SINGLE_ENDED);`
   Aby odczytać surową wartość czujnika (napięcie), należy za każdym razem wywołać instrukcje:
   ```
   HAL_ADC_Start(&hadc1);
   HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
   rawValue = HAL_ADC_GetValue(&hadc1);
   HAL_ADC_Stop(&hadc1);
   ```
   Temperaturę można następnie obliczyć ze wzoru:

$$temperature = \frac{vSense - V25}{avgSlope} + C25,$$

   gdzie:
   - *V25* – napięcie odniesienia dla 25℃ (domyślnie 0.76);
   - *C25* – wartość 25℃;
   - *avgSlope* – średnie nachylenie charakterystyki czujnika temperatury (domyślnie 0.0025);
   - *vSense* – napięcie, które można przeliczyć z następującego wzoru:

$$vSense = \frac{supplyVoltage \cdot rawValue}{resolutionADC},$$

   gdzie:
   - *supplyVoltage* – napięcie odniesienia używane przez ADC (domyślnie 3.3);
   - *resolutionADC* – rozdzielczość ADC (maksymalna wartość typu `unsigned int`, jaką można zapisać na 12 bitach);
   - *rawValue* – surowa wartość napięcia, odczytana przez czujnik.

   Podpowiedź: Konwersja wartości temperatury typu `float` na łańcuch znakowy `char*` może być wykonana przy użyciu funkcji `snprintf`. Aby umożliwić jej poprawne działanie konieczna będzie zmiana ustawienia MCU. W tym celu postępuj według wskazówki wyświetlanej w edytorze.

Funkcje

- **TIM**
  - `HAL_StatusTypeDef` **`HAL_TIM_Base_Start_IT`**`(TIM_HandleTypeDef *htim)`
  - `void` **`HAL_TIM_PeriodElapsedCallback`**`(TIM_HandleTypeDef *htim)`
  - `HAL_StatusTypeDef` **`HAL_TIM_PWM_Start`**`(TIM_HandleTypeDef *htim, uint32_t Channel)`
  - `HAL_StatusTypeDef` **`__HAL_TIM_SET_COMPARE`**`(TIM_HandleTypeDef *htim, uint32_t Channel, uint32_t pulse)`

    __HAL_TIM_SET_COMPARE jest w rzeczywistości makrodefinicją, ale można ją używać tak, jakby była funkcją o przedstawionej wyżej sygnaturze.

- **ADC**
  - `HAL_StatusTypeDef` **`HAL_ADCEx_Calibration_Start`**`(ADC_HandleTypeDef *hadc, uint32_t SingleDiff)`
  - `HAL_StatusTypeDef` **`HAL_ADC_Start`**`(ADC_HandleTypeDef *hadc)`
  - `HAL_StatusTypeDef` **`HAL_ADC_Stop`**`(ADC_HandleTypeDef *hadc)`
  - `HAL_StatusTypeDef` **`HAL_ADC_PollForConversion`**`(ADC_HandleTypeDef *hadc, uint32_t Timeout)`
  - `uint32_t` **`HAL_ADC_GetValue`**`(const ADC_HandleTypeDef *hadc)`

Materiały

- „STM32CubeIDE - poradnik” – dokument dostępny na stronie prowadzącego w folderze przedmiotu Architektura systemów komputerowych.
- STM32G4 Series advanced Arm®-based 32-bit MCUs - Reference manual
- Datasheet - STM32G491xC STM32G491xE - Arm® Cortex®-M4 32-bit MCU+FPU, 170 MHz / 213 DMIPS, up to 512 KB Flash, 112 KB SRAM, rich analog, math accelerator
- Description of STM32G4 HAL and low-layer drivers - User manual