STM32 dla początkujących (i nie tylko)

Liczniki

Trzecia część cyklu związanego z Panelem Edukacyjnym dla STM32F103 zostanie poświęcona licznikom stanowiącym wyposażenie sprzętowe tego mikrokontrolera. Zwięzły opis i proste przykłady pokażą jak ich użyć w praktyce.

Liczniki czy też z angielska *timery* działają jak układy zliczające impulsy. Stanowią wydzieloną sprzętową część struktury mikrokontrolera. Program główny może mieć dostęp do liczników i kontrolować ich działanie poprzez zapis i odczyt ich rejestrów sterujących. Chociaż wszystkie funkcje liczników można zastąpić procedurami, wykorzystanie liczników daje kilka istotnych korzyści. Ponieważ są układami sprzętowymi mogą działać niezależnie nie obciążając programu głównego. Są szybkie, mogą także pełnić dodatkowe funkcje nie tylko zliczające. Liczniki, w które wyposażono STM32F potrafią:

- Zliczać impulsy z różnych źródeł, w górę, w dół a także w zaprogramowanym przedziale od-do.
- Mogą służyć do pomiaru czasu lub okresu zewnętrznych impulsów.
- Generować impulsy o zaprogramowanym czasie trwania.

To są podstawowe, najczęściej wykorzystywane funkcje liczników. W tym miejscu należy dodać, że liczniki STM32F mają dodatkowe mechanizmy rozszerzające ich możliwości. Mogą generować impulsy PWM, współpracować z silnikami krokowymi czy zewnętrznymi sensorami takimi jak czujniki pola magnetycznego Halla. Te właściwości stanowią rozszerzenia trzech podstawowych wymienionych wcześniej funkcji.

Liczniki mikrokontrolera Panelu Edukacyjnego

Zamontowany na Panelu Edukacyjnym mikrokontroler STM32F103RBT6 wyposażony został w 4 liczniki TIM1-TIM4. Każdy z liczników jest 16 bitowy i pozwala zliczać impulsy w przedziale od 0 do 65535. TIM1 jest najbardziej rozbudowany, wyposażony w dodatkowe funkcje. TIM2-4 są identycznymi standardowymi licznikami pozwalającymi zliczać, generować impulsy, mierzyć czas i zostaną dokładniej opisane.

Budowa standardowego licznika STM32F

Dokładny opis liczników można znaleźć w dokumentacji technicznej *Reference manual RM0008* (*CD00171190.pdf*). Jest tam pokazany schemat blokowy licznika. Dla łatwiejszego zrozumienia zasady jego budowy i działania, na **rysunku 1** przedstawiono ten właśnie schemat, ale z pominięciem mniej istotnych szczegółów.

Podstawową częścią każdego timera jest 16-bitowy licznik oznaczony na rysunku jako *CNT counter*. Licznik może naliczać, odliczać, być zatrzymywany i zerowany. W przypadku przewinięcia np. doliczenia do wartości 65535, z rejestru *Autoreload register* do licznika wpisywana jest zaprogramowana wcześniej przez użytkownika wartość początkowa, która nie musi być zerem.

Licznik może współpracować z 4 kanałami oznaczonymi CH1…CH4. Każdy kanał to:

Fizyczne wyprowadzenie będące portem mikrokontrolera TIMx-CH1… TIMx-CH4.

Obwody wejściowe pełniących rolę filtrów i detektorów zboczy.

Preskalery (programowalne podzielniki impulsów).

Rejestr mogący pełnić funkcję *capture*, czyli przechwytywania zawartości licznika głównego, lub *compare* nieprzerwanego porównywania z zawartością tego licznika.

Obwody wyjściowe dołączone do fizycznych wyprowadzeń TIMx-CH1…CH4.

Tak skomplikowana budowa kanałów wynika z tego, że każdy z nich może pracować w jednym z 2 trybów: przechwytywania lub porównywania.

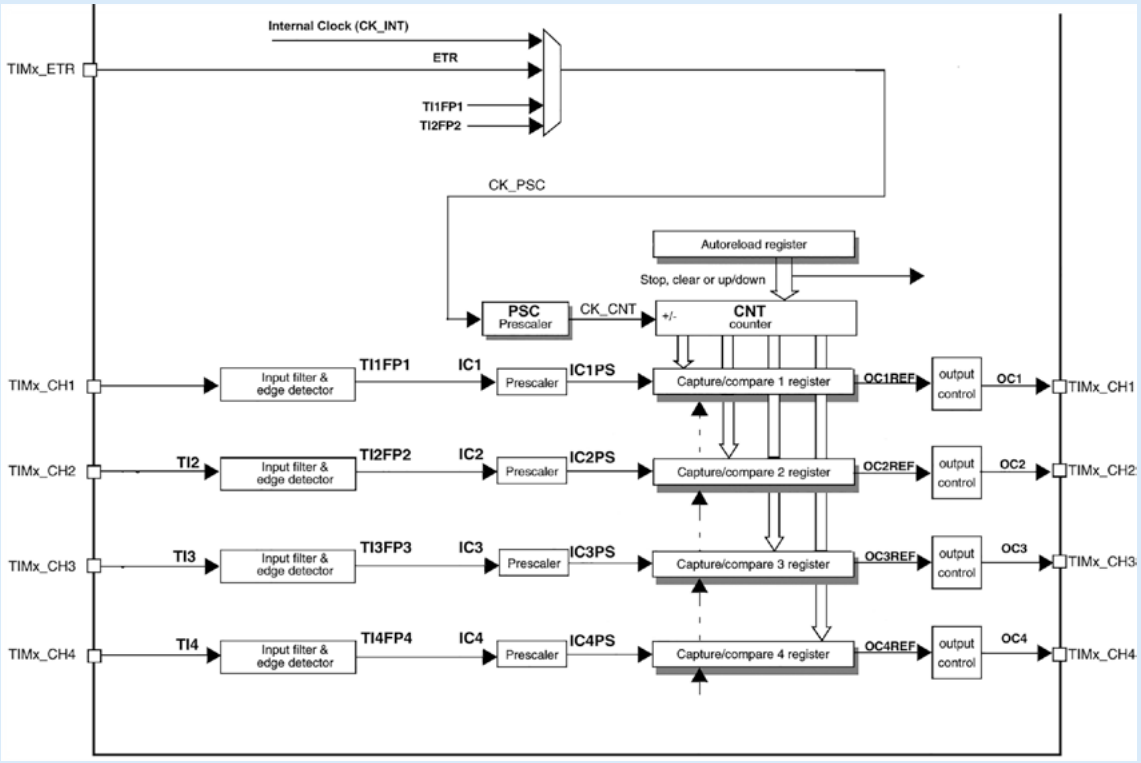
W trybie przechwytywania kanał może być tak zaprogramowany, żeby zapamiętać bieżącą zawartość głównego licznika np. w momencie pojawienia się zbocza impulsu na wejściu TIMx-CH. Wartości zapamiętane w rejestrach *capture* program główny może odczytać i w taki sposób np. obliczyć czas pomiędzy pojawianiem się kolejnych zboczy. W trybie przechwytywania obwody wyjściowe są odłączone od wyprowadzeń TIMx-CH.

W trybie porównywania użytkownik może wpisać do rejestrów *compare* wartość, z którą jest porównywana zawartość licznika głównego. Kiedy obie wartości są sobie równe może nastąpić zaprogramowana wcześniej akcja: np. wyprowadzenie TIMx-CH może zmienić poziom na przeciwny lub na wcześniej określony. W trybie porównywania obwody wejściowe są odłączone od wyprowadzeń TIMx-CH.

W każdej chwili każdy z kanałów może być wyłączony lub zaprogramowany do pracy w jednym z opisanych dwu trybów. Tryb pracy można ustawić indywidualnie dla każdego z 4 kanałów.

- Główny licznik może zliczać impulsy z różnych źródeł i decyduje o tym użytkownik. Mogą to być:
- Impulsy z wewnętrznego zegara mikrokontrolera *Internal Clock*.
- Impulsy podawane z przypisanego do każdego licznika zewnętrznego wyprowadzenia mikrokontrolera *TIMx_ETR*.

Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: ftp://ep.com.pl, user: 43061, pass: 3apmy741



Rysunek 1. Schemat blokowy licznika STM32F

- Impulsy podawane z wejść kanałów 1 lub 2.

Taka budowa sprawia, że standardowe liczniki mogą nie tylko zliczać impulsy, ale i mierzyć czas lub generować przebiegi o zadanym okresie.

Remapowanie wyprowadzeń i STM32CubeMX

Jak to zostało wcześniej napisane do każdego kanału przypisane jest fizyczne wyprowadzenie TIMx-CH. Wyprowadzenie może pełnić rolę wejścia lub wyjścia, reagować na zbocze podawanych impulsów lub ustawiać swój poziom zależnie od stanu głównego licznika i rejestru kanału. Obecność tych wyprowadzeń wzbogaca możliwości timerów o zdolność do interakcji z układami otaczającymi mikrokontroler.

Konstruktorzy STM-a znając realia życia konstruktora przewidzieli możliwość przypisania wyprowadzeń TIMx-CH do różnych portów. Służy do tego funkcja remapowania. O tym, który port można dołączyć do wyprowadzenia TIMx-CH decyduje typ mikrokontrolera jego obudowa a także numer timera. W dokumentacji technicznej można wyszukać tabele *TIMx alternate function remapping* (gdzie x jest numerem timera) i zobaczyć dostępne możliwości zmiany przyporządkowania wyprowadzeń funkcją remapowania. Dodatkowo można się posłużyć remapowaniem pełnym lub częściowym.

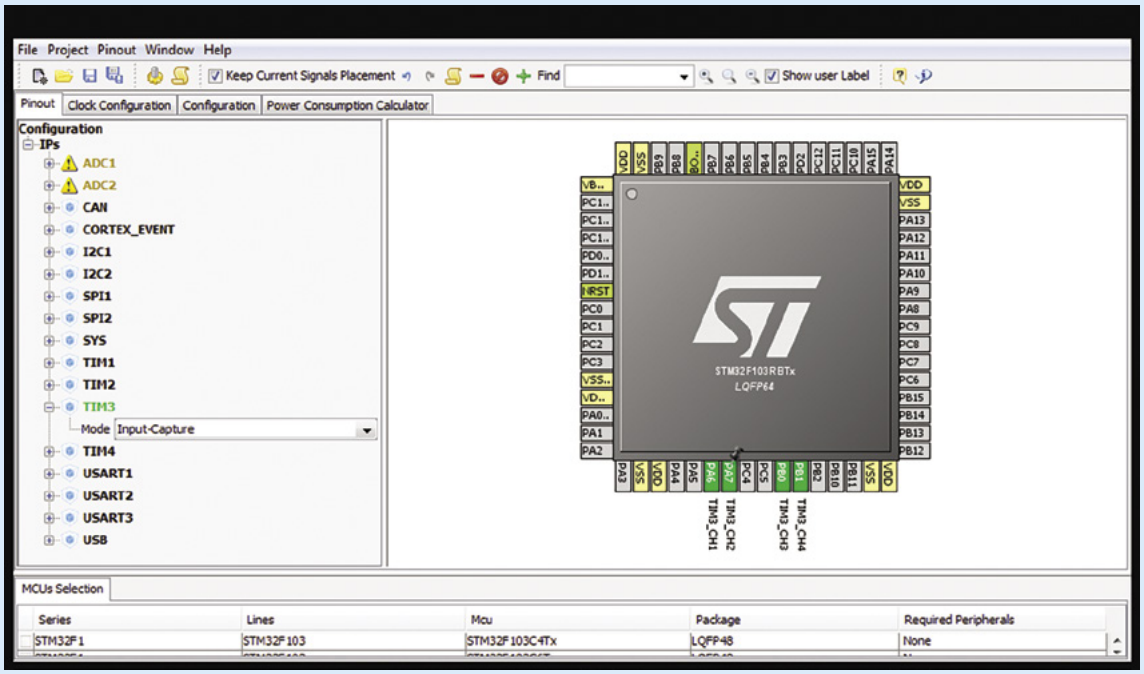
Jak zwykle, poszerzenie swobody powoduje komplikacje w wyborze najkorzystniejszego rozwiązania. Ponieważ niektóre porty mogą pełnić wiele funkcji (być np. wyprowadzeniem kanału timera albo wejściem przetwornika albo wyjściem portu szeregowego itp.), co może powodować konflikty łatwo o błąd. W tej sytuacji wygodnym rozwiązaniem jest posłużenie się narzędziem o nazwie *STM32CubeMX*. Jest to stworzony przez firmę ST darmowy program do przeglądania konfiguracji portów różnych typów mikrokontrolerów STM32 i wykrywania potencjalnych konfliktów. Narzędzie wyewoluowało ze stosunkowo prostego programu w „kombajn" o coraz większych możliwościach wyposażony np. w możliwość automatycznego generowania fragmentów programu w języku C. Wykorzystuje przy tym pliki projektów, które można zapisywać i otwierać podczas kolejnej sesji.

Po otwarciu programu i wskazaniu mikrokontrolera, w naszym wypadku STM32F103RBT6 wyświetlony zostanie pulpit ze schematycznie narysowaną obudową oraz listą dostępnych interfejsów sprzętowych. Na **rysunku 2** przedstawiono sytuację, w której na zielono zostały zaznaczone wyprowadzenia związane z kanałami Timera 3 pracującego w trybie przechwytywania. W przypadku konfliktu z wyprowadzeniami innych interfejsów sprzętowych zostaną one zaznaczone na czerwono. Częściowe konflikty wykluczające tylko niektóre funkcjonalności zaznaczone są wykrzyknikami na żółtym tle. Jeśli wystąpi konflikt należy wybrać inny timer albo spróbować remapować wyprowadzenia. Wskazanie na interesujące nas wyprowadzenie przy jednoczesnym naciśnięciu klawisza *Ctrl* i kliknięciu myszą na niebiesko wyświetli alternatywne porty dostępne po remapowaniu. Użycie opcji *Generate code* spowoduje wygenerowanie szkieletu kodu konfigurującego wybrane porty.

Rejestry konfiguracyjne

Jak zwykle w przypadku układów mikrokontrolera także pracą timerów sterują ustawienia rejestrów konfiguracyjnych. Każdy licznik sterowany jest przez własny zestaw rejestrów. Ze względu na dużą ilość opcji pracy liczników samych rejestrów także jest sporo a jeszcze więcej kombinacji ich ustawień. Funkcje niektórych są dość oczywiste:

- TIMx_CNT – rejestr do odczytu i zapisu zawartości głównego licznika.
- TIMx_CCRx – zależnie od ustawionego trybu rejestr do odczytu przechwyconej wartości licz-



Rysunek 2. Oznaczenie kolorem zielonym wyprowadzeń związanych z kanałami Timera 3 pracującego w trybie przechwytywania

nika głównego (tryb *capture*) lub zapis wartości do porównań (tryb *compare*).

TIMx_PSC -preskaler (podzielnik) impulsów zliczanych przez główny licznik.

Ustawienia innych rejestrów kontrolują działanie poszczególnych kanałów licznika, wybór źródła zliczanych impulsów, ustawiają aktywne zbocza sygnałów itp. Jak zwykle można z poziomu programu głównego samodzielnie ustawiać właściwe rejestry lub wspomóc się biblioteką *STM32F10x Standard Peripherals Firmware Library*.

Ustawianie rejestrów konfiguracyjnych za pomocą biblioteki

Tak jak to opisano w poprzednim odcinku cyklu poświęconym GPIO, w katalogu głównym biblioteki *STM32F10x Standard Peripherals Firmware Library* należy odnaleźć i kliknąć plik menedżera *stm32f10x_stdperiph_lib_um.chm*. W polu wyszukiwań Index należy wpisać *tim* (timers). Na wyświetlonej liście kliknąć na TIM, z prawej strony pojawi się lista dostępnych modułów. Po wybraniu *TIM_Exported_Functions* zostanie wyświetlona obszerna lista dostępnych funkcji związanych z timerami. Standardowe użycie liczników wymaga znajomości tylko niektórych. Zamieszczone dalej przykłady oparte wykorzystują funkcje biblioteczne.

Sterowanie wyświetlaczem alfanumerycznym LCD

Zanim przejdziemy do zwięzłego omówienia przykładów demonstrujących użycie timerów najpierw kilka słów o obsłudze alfanumerycznego wyświetlacza LCD. Na Panelu Edukacyjnym gniazdo J5 przeznaczone zostało do zamontowania typowego wyświetlacza o organizacji 2×16 znaków. W opisanych dalej przykładach wyświetlacz będzie służył do zobrazowania działania liczników.

Do obsługi wyświetlacza wykorzystanych zostało 7 portów mikrokontrolera pełniących następujące funkcje:

- PC11 – sygnał RS, którego poziom decyduje o tym czy do wyświetlacza wysyłany jest rozkaz czy dane do wyświetlenia.
- PC12 – sygnał RW przełączający magistralę danych pomiędzy trybami zapisu i odczytu.
- PC6 – sygnał E zatrzaskujący dane na magistrali danych.
- PA8 – DB4 linia 4 magistrali danych.
- PB7 – DB5 linia 5 magistrali danych.
- PB6 – DB6 linia 6 magistrali danych.
- PA11 – DB7 linia 7 magistrali danych.

W czasie ewentualnej modyfikacji kodu przykładów należy pamiętać, że wykorzystywanie tych portów do innych celów może zakłócić pracę wyświetlacza. Co nie znaczy, że takie wykorzystanie nie jest możliwe. Dobry przykład stanowi linia PC6 współdzielona pomiędzy wyświetlacz (sygnał E) i klawiaturę multipleksowaną. Dla uniknięcia konfliktu na czas obsługi klawiatury wyświetlacz przełączany jest w tryb odczytu. Tym sposobem impulsy na linii PC6 nie powodują wypisywania na wyświetlaczu przypadkowych znaków.

Mikrokontroler STM32F103RBT6 nie posiada sprzętowego interfejsu LCD i obsługa wyświetlacza LCD w całości realizowana jest przy pomocy procedur programowych. Wszystkie znajdują się w pliku *Procedury_LCD.c*. Na początku programu wywoływana jest procedura *LCD_Inicjacja()* która ustawia porty komunikujące się z wyświetlaczem, przeprowadza jego programowy reset i ustawia tryb pracy.

Główną procedurą odpowiedzialną za wyświetlanie tekstów jest *Disp_Wyswietl_txt(char *p_code_txt, char poz_na_disp)*. Przy wywołaniu procedura wymaga podania wskaźnika do początku wyświetlanego tekstu oraz pozycji na wyświetlaczu od której tekst ma zostać wypisany (od 0 do 31). Dodatkowym warunkiem jest zakończenie wyświetlanego tekstu znakiem nowej linii (\n). Podanie miejsca wyświetlenia poza zakresem powoduje usunięcie migającego kursora z wyświetlacza.

PanEduSTM32F_Demo1_TIM użycie timera do zliczania zboczy impulsów zewnętrznych

Pierwszy przykład pokazuje sposób skonfigurowania timera do zliczania doprowadzonych z zewnątrz impulsów lub ich zboczy.

Do zliczania zostały użyte Timer3 i wejście kanału 2. Wejście połączone jest z przełącznikiem S13-1 którego przełączenie zwiera wejście kanału do masy. Te zwarcia czyli zbocza opadające impulsów są zliczane przez Timer3. Zawartość timera jest wyświetlana na wyświetlaczu LCD.

Na początku programu przeprowadzane są inicjacje kolejnych układów w tym portów GPIO. Poprzez remapowanie port PC7 podłączany jest do wejścia kanału 2 TIM3. Procedurę konfigurującą timer *TIMER3_Inicjacja()* w pliku *Procedury_TIMER.c* pokazano na **listingu 1**. Najpierw do Timera3 podłączany jest sygnał zegarowy. Potem wypełniane są kolejne pola struktury inicjującej działanie Timera3:

- TIM_Period – pełny, 16-bitowy zakres zliczania.
- TIM_Prescaler, TIM_ClockDivision – bez wstępnego podziału liczby impulsów wejściowych.
- TIM_CounterMode_Up – licznik dolicza kolejne impulsy.

W następnym kroku wypełniane są pola struktury odpowiedzialnej za programowanie działania kanału 2 timera:

- TIM_Channel – numer programowanego kanału.
- TIM_ICFilter – ustawienie filtru eliminującego krótkie impulsy zakłócające.
- TIM_ICPrescaler – preskaler kanału wyłączony.
- TIM_ICSelection – dołączenie portu do wejścia kanału.

Następnie, procedura *TIM_TIxExternalClockConfig* podłącza kanał 2 jako wejście impulsów Timera 3. W końcu jest włączany sam Timer 3. Odczyt bieżącej zawartości licznika realizuje procedura *uint16_t Odczyt_TIMER3(void)*.

Przykład oparty jest na procedurach biblioteki *STM32F10x Standard Peripherals Firmware Library.* Zachęcam do eksperymentów i doboru własnych ustawień parametrów licznika.

Zwracam uwagę, że Panel Edukacyjny celowo pozbawiony został elementów RC kształtujących sygnały wytwarzane przez elementy zwierne. W miarę pogarszania się ich jakości i generacji w momencie przełączania długich serii impulsów zakłócających nie będą one efektywnie eliminowane przez filtr kanału 2 a licznik może zliczać więcej niż 1 zbocze podczas przełączania styków przełącznika S13.

PanEduSTM32F_Demo2_TIM użycie timera do pomiaru czasu pomiędzy 2 zboczami

W tym przykładzie Timer3 użyty został do mierzenia czasu jaki upływa pomiędzy kolejnymi opadającymi zboczami impulsów na dwóch wejściach. W trakcie pomiaru wyświetlana jest aktualna zawartość Timera 3. Na koniec wyświetlany jest czas w milisekundach.

Do wytworzenia opadających zboczy wykorzystane zostały dwa przełączniki S13-1 i S13-3. Timer3 zlicza impulsy zegarowe o okresie 1ms pomiędzy zwarciem do masy przełącznika S13-1 a zwarciem do masy przełącznika S13-3. S13-1 podłączony jest do wejścia kanału 2, natomiast S13-3 do wejścia kanału 4. Oba kanały pracują w trybie zatrzaskiwania (*capture*). W momencie zwarcia do masy zapisywany jest w rejestrach kanałów stan licznika TIM3. Czas wylicza się z różnicy zapamiętanych wartości. Struktura przykładu jest podobna do poprzedniego z dwiema różnicami: wykorzystywane są dwa kanały a główny licznik TIM3 taktowany jest wewnętrznym zegarem o okresie 1ms (częstotliwość 1 kHz). Tak jak poprzednio na początku inicjowane są wewnętrzne układy. Porty GPIO są remapowane. Procedurę inicjującą TIM3 pokazano na **listingu 2**.

Początek procedury jest podobny do opisanej w poprzednim przykładzie. Tym razem jednak włączany jest preskaler który dzieli częstotliwość wewnętrznego zegara mikrokontrolera tak aby okres impulsów był równy 1ms. Impulsy z preskalera podawane są na wejście liczące głównego licznika TIM3. Następnie inicjowane są dwa kanały: 2 i 4 w trybie zatrzaskiwania (capture). Taktowany sygnałem zegarowym licznik zlicza impulsy zegara do wartości FFFF po czym następuje jego przewinięcie i kontynuacja zliczania od wartości 0. Pomiar czasu polega na obliczeniu różnicy pomiędzy wartościami zatrzaśniętymi w rejestrach kanałw 4 i 2. Oczywiście przy założeniu, że najpierw zadziałał przełącznik S13-1 podłączony do kanału 2 a potem S13-3 podłączony

Listing 1. Procedura konfigurująca Timer 3

```
//---------------------------------
//procedury inicjacji TIMER3
void TIMER3_Inicjacja(void)
{
  TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;
  TIM_ICInitTypeDef  TIM_ICInitStructure;
  // TIM3 clock enable
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);
  // TIM3 disable counter
  TIM_Cmd(TIM3, DISABLE);
  //konfiguracja TIM3 do zliczania zboczy
  TIM_TimeBaseStructure.TIM_Period =0xFFFF;//do zliczania wykorzystano 16 bitów
  TIM_TimeBaseStructure.TIM_Prescaler = 0;
  TIM_TimeBaseStructure.TIM_ClockDivision = 0;
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;//zliczanie „do góry”
  TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);
  TIM_ICInitStructure.TIM_Channel =TIM_Channel_2;
  TIM_ICInitStructure.TIM_ICFilter =0x0F;
  TIM_ICInitStructure.TIM_ICPolarity =TIM_ICPolarity_Falling;
  TIM_ICInitStructure.TIM_ICPrescaler =TIM_ICPSC_DIV1;
  TIM_ICInitStructure.TIM_ICSelection =TIM_ICSelection_DirectTI;
  TIM_ICInit(TIM3, &TIM_ICInitStructure);
  TIM_TIxExternalClockConfig (TIM3, TIM_TIxExternalCLK1Source_TI2, TIM_ICPolarity_Falling, 0x0F);
  //włączenie licznika
  TIM_Cmd(TIM3, ENABLE);
}
```

do kanału 4. Sytuacja komplikuje się, gdy w trakcie pomiaru licznik główny przewinie się i zacznie liczyć od 0. W tej sytuacji najpierw należy wartość rejestru kanału 2 odjąć od liczby FFFF określającej maksymalną pojemność licznika a do różnicy dodać zawartość rejestru kanału 4:

Listing 2. Zmodyfikowana procedura inicjująca Timer 3 (taktowanie z częstotliwością 1 kHz)

```
//--------------------------------
//TIMER3 taktowany wewnętrznym zegarem z częstotliwością
//1kHz dla rozdzielczości pomiaru 1ms
//START pomiaru: zbocze opadające podawane na wejście TIM_Channel_2
//STOP  pomiaru: zbocze opadające podawane na wejście TIM_Channel_4
//procedury inicjacji TIMER3
void TIMER3_Inicjacja(void)
{
  TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;
  TIM_ICInitTypeDef  TIM_ICInitStructure;
  uint16_t PrescalerValue = 0;
  #define TAKTOWANIE_1kHz  1000

  // TIM3 clock enable
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);
  // TIM3 disable counter
  TIM_Cmd(TIM3, DISABLE);
  //konfiguracja TIM3 do pomiaru czasu pomiędzy 2 zboczami
  TIM_TimeBaseStructure.TIM_Period =0xFFFF;  //do pomiaru wykorzystano 16 bitów
  TIM_TimeBaseStructure.TIM_Prescaler = 0;
  TIM_TimeBaseStructure.TIM_ClockDivision = 0;
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;//zliczanie „do góry”
  TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);
  /* Prescaler configuration */
  PrescalerValue = (uint16_t) (SystemCoreClock / (TAKTOWANIE_1kHz)) - 1;
  TIM_PrescalerConfig(TIM3, PrescalerValue, TIM_PSCReloadMode_Immediate);
  //konfiguracja kanału 2 do przechwycenia stanu TIMER3 w momencie podania opadającego zbocza
na wejście PC7
  TIM_ICInitStructure.TIM_Channel =TIM_Channel_2;
  TIM_ICInitStructure.TIM_ICFilter =0x0F;
  TIM_ICInitStructure.TIM_ICPolarity =TIM_ICPolarity_Falling;
  TIM_ICInitStructure.TIM_ICPrescaler =TIM_ICPSC_DIV1;
  TIM_ICInitStructure.TIM_ICSelection =TIM_ICSelection_DirectTI;
  TIM_ICInit(TIM3, &TIM_ICInitStructure) ;
  //konfiguracja kanału 4 do przechwycenia stanu TIMER3 w momencie podania opadającego zbocza
na wejście PC9
  TIM_ICInitStructure.TIM_Channel =TIM_Channel_4;
  TIM_ICInitStructure.TIM_ICFilter =0x0F;
  TIM_ICInitStructure.TIM_ICPolarity =TIM_ICPolarity_Falling;
  TIM_ICInitStructure.TIM_ICPrescaler =TIM_ICPSC_DIV1;
  TIM_ICInitStructure.TIM_ICSelection =TIM_ICSelection_DirectTI;
  TIM_ICInit(TIM3, &TIM_ICInitStructure) ;
  //włączenie licznika
  TIM_Cmd(TIM3, ENABLE);
}
```

Listing 3. Inicjacja portów GPIO połączona z remapowaniem, inicjacja Timera 3 i wyświetlacza LCD

```
#define TAKTOWANIE_1kHz    1000
//--------------------------------
//TIMER3 taktowany wewnętrznym zegarem z częstotliwością
// 1 kHz dla rozdzielczości pomiaru 1ms
//START generowania impulsu: zbocze opadające podawane
// na wejście TIM_Channel_2 procedury inicjacji TIMER3
void TIMER3_Inicjacja(void)
{
TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;
TIM_ICInitTypeDef  TIM_ICInitStructure;
TIM_OCInitTypeDef  TIM_OCInitStructure;
uint16_t PrescalerValue = 0;

  // TIM3 clock enable
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);
  // TIM3 disable counter
  TIM_Cmd(TIM3, DISABLE);
  //konfiguracja TIM3 do generacji impulsów
  TIM_TimeBaseStructure.TIM_Period =1001;
  TIM_TimeBaseStructure.TIM_Prescaler = 0;
  TIM_TimeBaseStructure.TIM_ClockDivision = 0;
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;  //zliczanie „w górę”
  TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);
  /* Prescaler configuration */
  PrescalerValue = (uint16_t) (SystemCoreClock / (TAKTOWANIE_1kHz)) - 1;
  TIM_PrescalerConfig(TIM3, PrescalerValue, TIM_PSCReloadMode_Immediate);
  /* TIM3 PWM2 Mode configuration: Channel4 */
  TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM2;
  TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
  TIM_OCInitStructure.TIM_Pulse = 1;
  TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
  TIM_OC4Init(TIM3, &TIM_OCInitStructure);
  //konfiguracja kanału 2 do wyzwalania TIMER3 w momencie podania opadającego zbocza na wejście PC7
  TIM_ICInitStructure.TIM_Channel =TIM_Channel_2;
  TIM_ICInitStructure.TIM_ICFilter =0x0;
  TIM_ICInitStructure.TIM_ICPolarity =TIM_ICPolarity_Falling;
  TIM_ICInitStructure.TIM_ICPrescaler =TIM_ICPSC_DIV1;
  TIM_ICInitStructure.TIM_ICSelection =TIM_ICSelection_DirectTI;
  TIM_ICInit(TIM3, &TIM_ICInitStructure) ;
  /* One Pulse Mode selection */
  TIM_SelectOnePulseMode(TIM3, TIM_OPMode_Single);
  /* Input Trigger selection */
  TIM_SelectInputTrigger(TIM3, TIM_TS_TI2FP2);
  /* Slave Mode selection: Trigger Mode */
  TIM_SelectSlaveMode(TIM3, TIM_SlaveMode_Trigger);
}
```

Listing 4. Ustawianie czasu trwania pojedynczego impulsu
```
void Ustawianie_Czasu_Imp_ms(uint16_t czas_ms)
{
  TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;
  uint16_t PrescalerValue = 0;
  TIM_Cmd(TIM3, DISABLE);
  TIM_TimeBaseStructure.TIM_Period =czas_ms +1;
  TIM_TimeBaseStructure.TIM_Prescaler = 0;
  TIM_TimeBaseStructure.TIM_ClockDivision = 0;
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;  //zliczanie „w górę”
  TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);
  /* Prescaler configuration */
  PrescalerValue = (uint16_t) (SystemCoreClock / (TAKTOWANIE_1kHz)) - 1;
  TIM_PrescalerConfig(TIM3, PrescalerValue, TIM_PSCReloadMode_Immediate);
}
```

Listing 5. Ustawianie okresu generowanego przebiegu
```
void Ustawianie_Okresu_Imp_ms(uint16_t czas_ms)
{
  TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;
  TIM_OCInitTypeDef  TIM_OCInitStructure;
  uint16_t PrescalerValue = 0;
  TIM_Cmd(TIM3, DISABLE);
  TIM_TimeBaseStructure.TIM_Period =czas_ms;
  TIM_TimeBaseStructure.TIM_Prescaler = 0;
  TIM_TimeBaseStructure.TIM_ClockDivision = 0;
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;//zliczanie „do góry”
  TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);
  TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM2;
  TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
  TIM_OCInitStructure.TIM_Pulse = (czas_ms/2);
  TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
  TIM_OC4Init(TIM3, &TIM_OCInitStructure);
  /* Prescaler configuration */
  PrescalerValue = (uint16_t) (SystemCoreClock / (TAKTOWANIE_1kHz)) - 1;
  TIM_PrescalerConfig(TIM3, PrescalerValue, TIM_PSCReloadMode_Immediate);
}
```

czas_w_ms =(0xFFFF -zawartosc_CAPTURE2) + zawartosc_CAPTURE4 +1;

Jak poprzednio obsługa timera oparta została o procedury z biblioteki STM-a.

PanEduSTM32F_Demo3_TIM użycie timera do generacji pojedynczego impulsu

Trzeci przykład demonstruje użycie licznika do generowania pojedynczego impulsu o programowanym czasie trwania w zakresie od 1 do 60000 ms.

Do ustawiania czasu trwania impulsu używane są przyciski klawiatury multipleksowanej. Po naciśnięciu S12 na wyświetlaczu pojawia się kursor i można zmieniać czas impulsu klawiszami S5 i S8. Do przesuwania kursora służą przyciski S1 i S3. Ponowne naciśnięcie S12 kończy wprowadzanie nowego czasu. Start generowania impulsu następuje po zwarciu do masy przełącznika S13-1. Dodatni impuls pojawia się na wyjściu kanału 4 (PC9). Do sygnalizacji impulsu można użyć którejś z diod LED zamontowanych na Panelu Edukacyjnym. W tym celu należy zewrzeć przewodem zakończonym obustronnie wtykami wyprowadzenia PC9 (złącze J8-4) np. z wyprowadzeniem PB8 (złącze J7-2).

Oprócz wykonania połączenia na Panelu należy:

- Założyć odpowiednią zworę na złączu JP6 (np. jeśli użyjemy do sygnalizacji D1 należy założyć zworę JP6 1-2).
- Założyć wszystkie zwory na złączu JP5 klawiatury.
- Przełącznik S13-3 ustawić w pozycji OFF (rozłączony).

Podobnie jak w poprzednich przykładach na początku programu następuje inicjacja portów GPIO połączona z remapowaniem, inicjacja Timera 3 i wyświetlacza LCD. Procedura inicjacji Timera 3 tym razem wygląda jak na **listingu 3**. Do generowania impulsu wykorzystano tryb PWM2. Timer 3 programowany jest do odliczenia 1001 taktów wewnętrznego zegara o okresie 1 ms. Kanał 4 timera jest programowany do odliczenia krótkiej pauzy o czasie trwania 1 ms. Po jej zakończeniu port PC9 zostanie ustawiony, co oznacza rozpoczęcie generowania impulsu. Po odliczeniu 1001 taktów zegara port PC9 zostanie wyzerowany i generowanie impulsu o długości 1000 ms będzie zakończone. Cały proces inicjuje zwarcie do masy wejścia kanału 2 Timera 3. Kolejny impuls zostanie wygenerowany po ponownym zwarciu do masy przełącznika S13-1 podłączonego do wejścia kanału 2. Ustawienie nowego czasu trwania impulsu następuje w procedurze *Ustawianie_Czasu_Imp_ms(uint16_t czas_ms)* pokazanej na **listingu 4**/

PanEduSTM32F_Demo4_TIM użycie timera do generacji sygnału o wypełnieniu 50%

Czwarty program demonstracyjny pokazuje jak przy pomocy timera można wygenerować przebieg prostokątny o programowanym okresie impulsów i wypełnieniu 50%. Program czwarty stanowi zmodyfikowaną wersję programu trzeciego. W taki sam sposób ustawia się okres sygnału jak w programie trzecim czas trwania impulsu. Należy w identyczny sposób założyć zwory i połączenia oraz pamiętać, że PC9 stanowi wyjście i przełącznik S13-3 powinien być rozwarty. Procedura inicjacji Timera 3 jest bardzo podobna z dwoma wyjątkami:

Jeśli wypełnienie przebiegu ma wynosić 50% parametr TIM_Pulse powinien być równy połowie okresu przebiegu ustawionego w TIM_Period. Dla okresu równego 1000ms ustawienie parametru TIM_Pulse powinno być następujące: *TIM_OCInitStructure.TIM_Pulse = 500;*

Timer 3 ustawiany jest w trybie generowania przebiegu ciągłego: *TIM_SelectOnePulseMode(TIM3, TIM_OPMode_Repetitive);*

Z kolei procedura ustawienia nowego okresu generowanego przebiegu prostokątnego może wyglądać jak na **listingu 5**.

Ryszard Szymaniak, EP