M. Suchenek

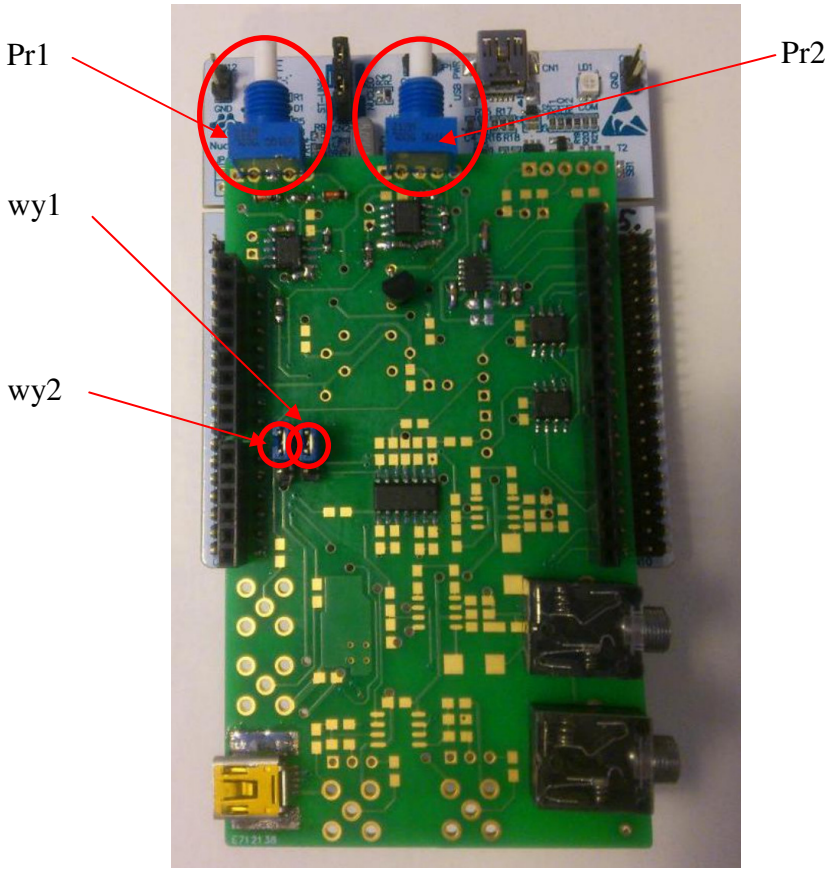
MARM

Laboratorium 3 – system liczników i przerwań

Celem laboratorium jest zapoznanie się z systemem liczników oraz przerwaniami mikrokontrolera STM32.

Zestaw laboratoryjny

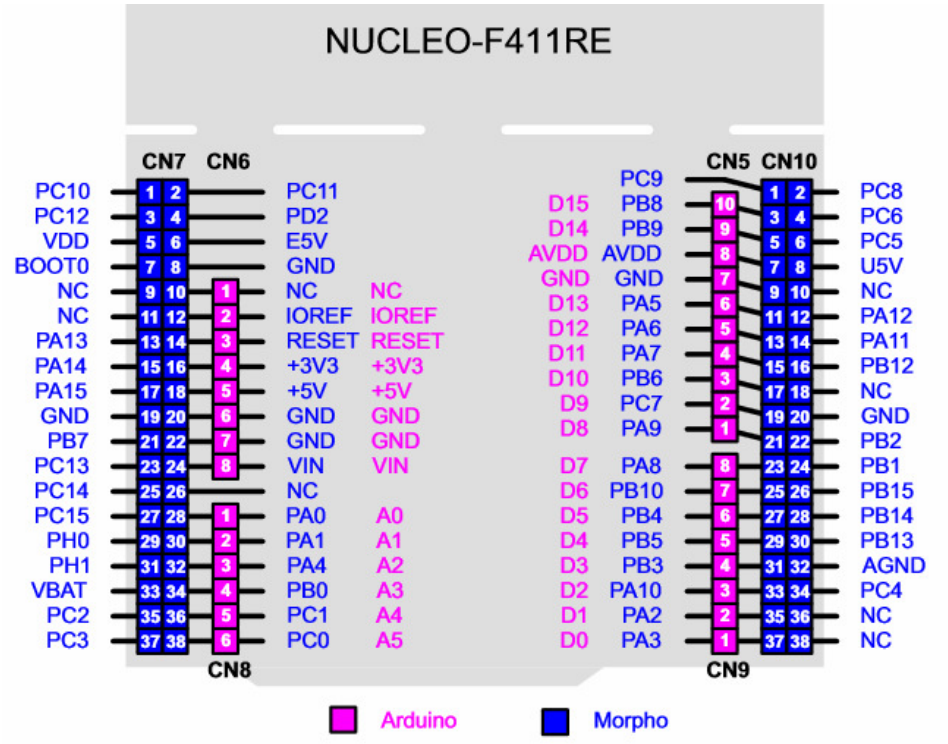
Do płytki NUCLEO-F411RE dołączona jest nakładka z generatorami przebiegów sygnału prostokątnego sterowanego potencjometrami Pr1 służącego do zmiany częstotliwości sygnału wyjścia wy1, potencjometrem Pr2 do zmiany wypełniania sygnału, wyjście wy2. Wygląd płytki wraz z oznaczeniami wyjść przedstawiono na rysunku.



Zdjęcie płytki uruchomieniowej z mikrokontrolerem STM32F411 oraz płytką z generatorami.

Wyjścia generatorów wy1 oraz wy2 można podłączyć do jednego z pinów złącza CN7 bądź CN10 płytki NUCLEO-F411RE przy pomocy linii połączeniowej.

Opis wyprowadzeń płytki uruchomieniowej NUCLEO-F411



1. System liczników

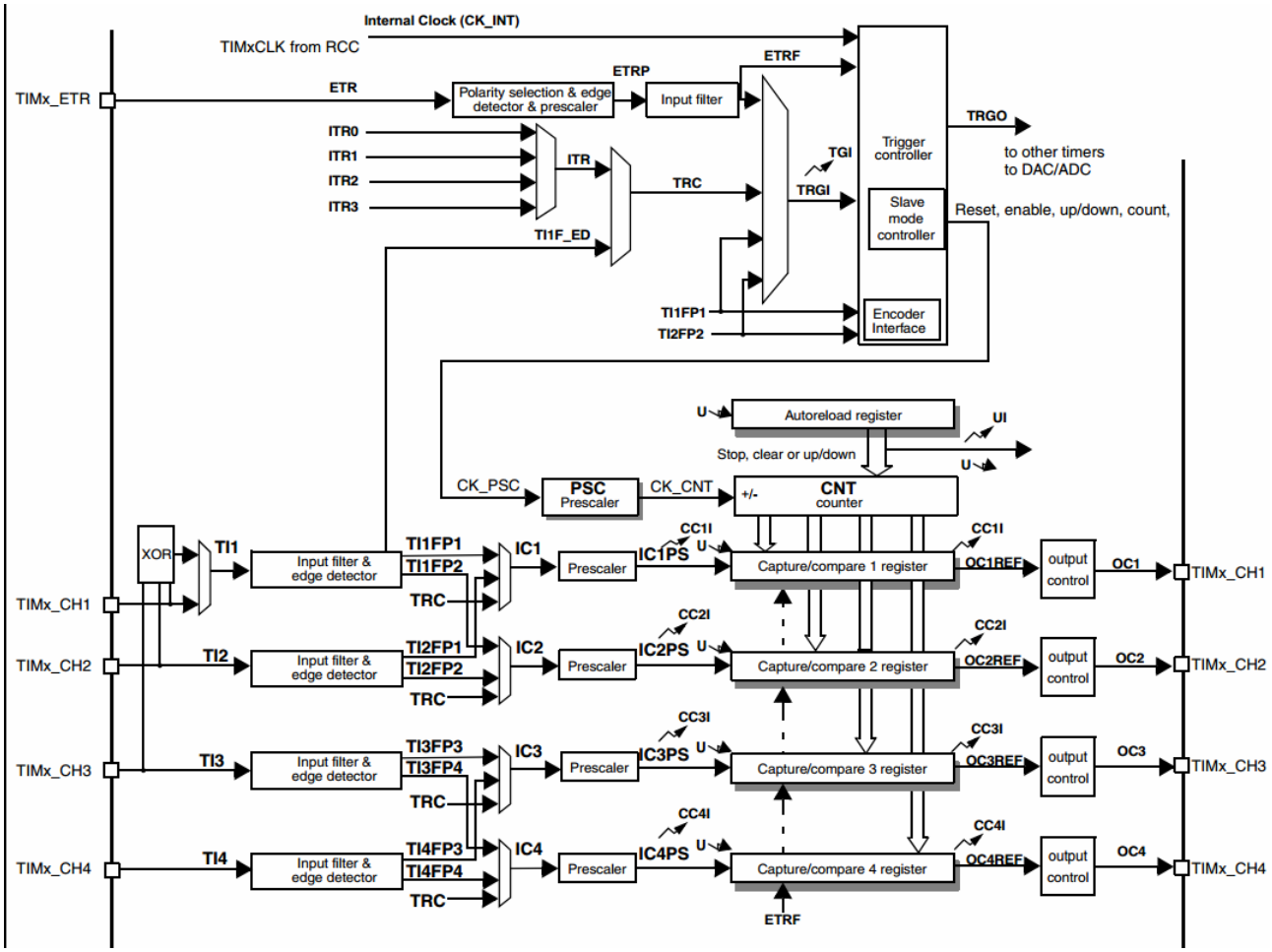
Parametry poszczególnych liczników mikrokontrolera STM32F411RE zamieszczono w tabeli:

Table 4. Timer feature comparison

Timer type	Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/ compare channels	Complementary output	Max. interface clock (MHz)	Max. timer clock (MHz)
Advanced-control	TIM1	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	Yes	100	100
General purpose	TIM2, TIM5	32-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	50	100
	TIM3, TIM4	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	50	100
	TIM9	16-bit	Up	Any integer between 1 and 65536	No	2	No	100	100
	TIM10, TIM11	16-bit	Up	Any integer between 1 and 65536	No	1	No	100	100

Źródło: dokumentacja STM32F411xC STM32F411xE strona 27.

Ogólna struktura licznika mikrokontrolera STM32F411



W zależności od trybu pracy licznik może zliczać impulsy sygnału podłączone do zewnętrznego wejścia mikrokontrolera bądź zliczać impulsy zegara mikrokontrolera (pracować jako czasomierz).

Podstawowa struktura konfiguracji licznika:

```
typedef struct
{
        uint16_t TIM_Prescaler;
        uint16_t TIM_CounterMode;
        uint16_t TIM_Period;
        uint16_t TIM_ClockDivision;
        uint8_t TIM_RepetitionCounter;
} TIM_TimeBaseInitTypeDef;
```

Opis poszczególnych pól struktury:

uint16_t TIM_Prescaler: dzielnik częstotliwości ustawiany w zakresie 0x0000 - 0xFFFF

uint16_t TIM_Period: wartość do której lub od której będzie następowało zliczanie. W zależności czy licznik zlicza w górę czy w dół. Ustawiany w zakresie od 0x0000 do 0xFFFF

uint16_t TIM_CounterMode: tryb pracy licznika

- TIM_CounterMode_Up - licznik zliczający w górę
- TIM_CounterMode_Down - licznik zliczający w dół
- TIM_CounterMode_CenterAligned1 – jeśli licznik zlicza w dół

- TIM_CounterMode_CenterAligned2 – jeśli licznik zlicza w górę
- TIM_CounterMode_CenterAligned3 – licznik zliczający w górę do zadanej wartości, następnie zlicza w dół do wartości 0, można włączyć flagę przerwania przy zmianie kierunku zliczania

uint16_t TIM_ClockDivision - wybór dzielnika zegara dla układu generatora martwego czasu i filtra wejściowego:

- TIM_CKD_DIV1 – brak dzielnika
- TIM_CKD_DIV2 – dzielnik clk/2
- TIM_CKD_DIV4 – dzielnik clk/4

uint8_t TIM_RepetitionCounter: liczba powtórzeń w liczniku powtórzeń, ustawiana w zakresie od 0x00 do 0xFF

Funkcja zapisująca ustawuienia do rejestru
void TIM_TimeBaseInit(TIM_TypeDef* TIMx, TIM_TimeBaseInitTypeDef* TIM_TimeBaseInitStruct)

Struktura konfiguracji wyjścia komparatorów

```
typedef struct
{
        uint16_t TIM_OCMode;
        uint16_t TIM_OutputState;
        uint16_t TIM_OutputNState;
        uint16_t TIM_Pulse;
        uint16_t TIM_OCPolarity;
        uint16_t TIM_OCNPolarity;
        uint16_t TIM_OCIdleState;
        uint16_t TIM_OCNIdleState;
} TIM_OCInitTypeDef;
```

uint16_t TIM_OCMode: tryb pracy kanału komparatora

- TIM_OCMode_Timing - wyjście komparatora nieaktywne
- TIM_OCMode_Active - wyjście komparatora w stanie aktywnym w momencie osiągnięcia przez licznik wartości porównywanej
- TIM_OCMode_Inactive - wyjście komparatora nie aktywne
- TIM_OCMode_Toggle - wyjście komparatora będzie zmieniać się na przemian w momencie osiągnięciu przez licznik porównywanej wartości
- TIM_OCMode_PWM1 - tryb PWM, komparator jest aktywny do momentu osiągnięcia przez licznik wartości porównywanej
- TIM_OCMode_PWM2 - tryb PWM, komparator będzie aktywny od momentu osiągnięcia przez licznik wartości porównywanej

uint16_t TIM_Pulse: wartość porównania dla licznika, może być w zakresie od 0x0000 do 0xFFFF

uint16_t TIM_OutputState/TIM_OutputNState: wyłącza/włącza wyjście komparatora

- TIM_OutputState_Disable
- TIM_OutputState_Enable

uint16_t TIM_OCPolarity/TIM_OCNPolarity: polaryzacja poziomu wyjścia komparatora, aktywny

stan niski bądź wysoki

- TIM_OCIdleState/TIM_OCNIdleState: poziom linii w stanie jałowym
- TIM_OCIdleState_Set/TIM_OCIdleState_Reset – poziom logiczny "1" lub '0'

Funkcja zapisująca ustawienia komparatora do rejestrów
void TIM_OCxInit(TIM_TypeDef* TIMx, TIM_OCInitTypeDef* TIM_OCInitStruct)
void TIM_OCxPreloadConfig(TIM_TypeDef* TIMx, uint16_t TIM_OCPreload)

Przykład konfiguracji liczsnika 2 (TIM2), wyjścia komparatora nr. 2
TIM_OC*x*Init(TIM2, &TIM_OCInitStruct);
TIM_OC*x*PreloadConfig(TIM2, TIM_OCPreload_Enable);
gdzie
x – numer komparatora z zakresu od 1 do 4.

Jeśli wyjście komparatora ma być podłączone do wyprowadzenia mikrokontrolera należy przypisać wyjście mikrokontrolera do licznika funkcją:
void GPIO_PinAFConfig(GPIO_TypeDef *GPIOx, uint16_t GPIO_PinSource, uint8_t GPIO_AF)
Np. GPIO_PinAFConfig(GPIOA, GPIO_PinSource0, GPIO_AF_TIM2);

Struktura konfiguracji wejścia licznika:

```
typedef struct
	{
		uint16_t TIM_Channel;
		uint16_t TIM_ICPolarity;
		uint16_t TIM_ICSelection;
		uint16_t TIM_ICPrescaler;
		uint16_t TIM_ICFilter;
	} TIM_ICInitTypeDef;
```

uint16_t TIM_Channel: wybór komparatora wejściowego TIM_Channel_1 ... 4

uint16_t TIM_ICPolarity: zbocze sygnału względem którego następuje reakcja komparatora

- TIM_ICPolarity_Rising
- TIM_ICPolarity_Falling
- TIM_ICPolarity_BothEdge

uint16_t TIM_ICSelection

- TIM_ICSelection_DirectTI: wejście TIM 1/2/3/4 podłączone do IC 1/2/3/4
- TIM_ICSelection_IndirectTI: wejście TIM 1/2/3/4 podłączone do IC 2/1/4/3
- TIM_ICSelection_TRC wejście: TIM 1/2/3/4 podłączone do TRC

uint16_t TIM_ICPrescaler: dzielnik przechwytywanych zdarzeń (zbocze bądź zbocza)

- TIM_ICPSC_DIV1 - przechwytywane każde zdarzenie
- TIM_ICPSC_DIV2 – przechwytywane co 2 zdarzenie
- TIM_ICPSC_DIV4 – przechwytywane co 4 zdarzenie
- TIM_ICPSC_DIV8 – przechwytywane co 8 zdarzenie

uint16_t TIM_ICFilter: przyjmuje wartości od 0 do 0xFF, długość filtru w ramach którego sprawdzany jest stan wejściowej linii.

Funkcje związane z systemem liczników:

Porównywaną wartość można zmienić poprzez wywołanie funkcji przypisanej do określonego komparatora x – czyli od 1 do 4
void TIM_SetCompare*x*(TIM_TypeDef* TIMx, uint32_t Compare1);

Konfiguracja podrzędnego licznika (jako slave) poprzez funkcje: TIM_SelectInputTrigger(…) oraz TIM_SelectSlaveMode(…)

Podłączenie wejścia zliczającego
void TIM_SelectInputTrigger(TIM_TypeDef* TIMx, uint16_t TIM_InputTriggerSource);
TIMx: określa wybór licznika z zakresu od 1 do 14
TIM_InputTriggerSource – wybór wejścia zliczającego

- TIM_TS_ITR0: wewnętrzny ITR0
- TIM_TS_ITR1: wewnętrzny ITR1
- TIM_TS_ITR2: wewnętrzny ITR2
- TIM_TS_ITR3: wewnętrzny ITR3
- TIM_TS_TI1F_ED: detektor zboczy TI1
- TIM_TS_TI1FP1: filtr wejściowy 1
- TIM_TS_TI2FP2: filtr wejściowy 2
- TIM_TS_ETRF: zewnętrzny ETRF

Konfiguracja licznika w trybie podrzędnym „slave”
void TIM_SelectSlaveMode(TIM_TypeDef* TIMx, uint16_t TIM_SlaveMode)
TIMx: numer licznika od 1 do 9 lub 12
TIM_SlaveMode: tryb pracy

- TIM_SlaveMode_Reset: narastające zbocze sygnału TRGI inicjalizuje licznik zliczający i jego rejestry
- TIM_SlaveMode_Gated: zegar zliczający licznika będzie włączany jeśli TRGI będzie w stanie aktywnym „1”
- TIM_SlaveMode_Trigger: zegar zliczający licznika będzie włączany na narastającym zboczy sygnału TRGI
- TIM_SlaveMode_External1: zegar zliczający licznika podłączony do TRGI

Konfiguracja licznika w trybie nadrzędnym „master”
void TIM_SelectOutputTrigger(TIM_TypeDef* TIMx, uint16_t TIM_TRGOSource);
TIMx: wybór licznika z zakresu od 1 do 8
TIM_TRGOSource: Wybór wyjścia jako sygnału zegarowego dla licznika podrzędnego

- TIM_TRGOSource_OC1Ref: wyjście z komparatora 1 OC1REF jako wyjście TRGO
- TIM_TRGOSource_OC2Ref: wyjście z komparatora 2 OC2REF jako wyjście TRGO
- TIM_TRGOSource_OC3Ref: wyjście z komparatora 3 OC3REF jako wyjście TRGO
- TIM_TRGOSource_OC4Ref: wyjście z komparatora 4 OC4REF jako wyjście TRGO
- TIM_TRGOSource_Reset: bit UG w rejestrze TIM_EGR podłączony jest jako wyjście sygnału TRGO
- TIM_TRGOSource_Enable: Counter Enable CEN podłączony jest jako wyjście sygnału TRGO
- TIM_TRGOSource_Update: zdarzenie podłączony jest jako wyjście sygnału TRGO
- TIM_TRGOSource_OC1: Generowany jest impuls na wyjściu modułu TRGO jeśli flaga CC1IF jest ustawiana

Ustawienie licznika w trybie nadrzędnym (master)
void TIM_SelectMasterSlaveMode(TIM_TypeDef* TIMx, uint16_t TIM_MasterSlaveMode);
TIMx: numer licznika od 1 do 9 lub 12
TIM_MasterSlaveMode: specifies the Timer Master Slave Mode.

- TIM_MasterSlaveMode_Enable: włącza tryb licznika nadrzędnego, synchronizacja poprzez TRGO
- TIM_MasterSlaveMode_Disable: bez wpływu

Konfiguracja zewnętrznego wejścia ETR jako źródła zegara dla licznika
Void TIM_ETRConfig(TIM_TypeDef* TIMx, uint16_t TIM_ExtTRGPrescaler, uint16_t TIM_ExtTRGPolarity, uint16_t ExtTRGFilter);
TIMx: numer licznika 1 - 5, 8
TIM_ExtTRGPrescaler: konfiguracja wejściowego preskalera dla licznika

- TIM_ExtTRGPSC_OFF: bez preskalera
- TIM_ExtTRGPSC_DIV2: podział przez /2
- TIM_ExtTRGPSC_DIV4: podział przez /4
- TIM_ExtTRGPSC_DIV8: podział przez /8

TIM_ExtTRGPolarity: polaryzacja sygnału zewnetrznego

- TIM_ExtTRGPolarity_Inverted: aktywny niskim bądź opadającym zboczem
- TIM_ExtTRGPolarity_NonInverted: aktywny wysokim bądź narastającym zboczem zegara

ExtTRGFilter: długość filtru przeciwzakłóceniowego ustawiana w zakresie od 0x0 do 0x0F.

2. System przerwań

Konfiguracja priorytetów kontrolera przerwań
void NVIC_PriorityGroupConfig(uint32_t NVIC_PriorityGroup)

- NVIC_PriorityGroup_0 - 0 bitów priorytet główny, 4 bity pod priorytet
- NVIC_PriorityGroup_1 - 1 bitów priorytet główny, 3 bity pod priorytet
- NVIC_PriorityGroup_2 - 2 bitów priorytet główny, 2 bity pod priorytet
- NVIC_PriorityGroup_3 - 3 bitów priorytet główny, 1 bity pod priorytet
- NVIC_PriorityGroup_4 - 4 bitów priorytet główny, 0 bity pod priorytet

Np. NVIC_PriorityGroupConfig(NVIC_PriorityGroup_0);

Konfiguracja kontrolera przerwań NVIC poprzez strukturę:

```
typedef struct
{
        uint8_t NVIC_IRQChannel;
        uint8_t NVIC_IRQChannelPreemptionPriority;
        uint8_t NVIC_IRQChannelSubPriority;
        FunctionalState NVIC_IRQChannelCmd;
} NVIC_InitTypeDef;
```

Opis poszczególnych pól struktury:
NVIC_IRQChannel - numer kanału przerwania

NVIC_IRQChannelPreemptionPriority - numer priorytetu głównego, w zależności od modelu przerwania NVIC_PriorityGroup0-4, może przyjąć wartości od 0 do15

NVIC_IRQChannelSubPriority - numer podpriorytetu, w zależności od modelu przerwania NVIC_PriorityGroup0-4, może przyjąć wartości od 0 do 15

NVIC_IRQChannelCmd
ENABLE/DISABLE

Funkcja zapisująca strukturę do rejestrów mikrokontrolera
void NVIC_Init(NVIC_InitTypeDef * NVIC_InitStruct)
np. NVIC_Init(&NVIC_InitStructure);

Konfiguracja wejścia pinu mikrokontrolera jako źródła przerwań poprzez układ EXTI:

```
typedef struct
        {
                uint32_t EXTI_Line;
                EXTIMode_TypeDef EXTI_Mode;
                EXTITrigger_TypeDef EXTI_Trigger;
                FunctionalState EXTI_LineCmd;
        } EXTI_InitTypeDef;
```

Opis poszczególnych pól struktury:
EXTIMode_TypeDef EXTI_Mode: tryb obsługi

- EXTI_Mode_Interrupt – przerwanie
- EXTI_Mode_Event – zdarzenie

EXTITrigger_TypeDef: wystąpienie przerwania zboczem

- EXTI_Trigger_Rising – narastającym
- EXTI_Trigger_Falling – opadającym
- EXTI_Trigger_Rising_Falling – narastającym i opadającym

FunctionalState:
ENABLE/DISABLE

Funkcja zapisująca strukturę do rejestrów mikrokontrolera
void EXTI_Init(EXTI_InitTypeDef * EXTI_InitStruct)
Np. EXTI_Init(&EXTI_InitStructure);

Funkcje związane z przerwaniami:

Sprawdzenie statusu linii EXTI, dopuszczalne wartości od 0 do 22
ITStatus EXTI_GetITStatus(uint32_t EXTI_Line)
Np. EXTI_GetITStatus(EXTI_Line0)

Czyszczenie flagi linii EXTI
EXTI_ClearITPendingBit (uint32_t EXTI_Line)
Np. EXTI_ClearITPendingBit(EXTI_Line0)

MARM 2016L

Protokół z laboratorium nr. 3 z przedmiotu MARM

Temat:
liczniki, system przerwań mikrokontrolera STM32

Imię, nazwisko: .....

Numer płytki: ….....

Data: .....

Zadania do wykonania w trakcie laboratorium:

1) Proszę napisać funkcje odczytPWM - która będzie zwracać aktualną wartość wypełnienia sygnału z zewnętrznego wejścia mikrokontrolera.
2) Proszę rozbudować program z wykładu generujący sygnał typu PWM w taki sposób żeby można było generować sygnał o częstotliwości 10 kHz i tym samym wypełnieniu co sygnał zewnętrzny podłączony do mikrokontrolera.
* zmiana wypełnienia powinna zostać zrealizowana w przerwaniu od licznika sterującego wyjściem PWM

Zadania do realizacji		**Punktacja**	
1	Konfiguracja portów, zegarów	1	
2	Konfiguracja licznika pomiaru wypełnienia	1	
3	Odczyt wypełnienia	1	
4	Generacja sygnału PWM	1	
5	Sterowanie wyjściem w przerwaniu	1	
6	Generacja sygnału 10 kHz o zadanym wypełnieniu	1	
		SUMA	

Uwagi:

1)

2)

3)

4)

5)

6)