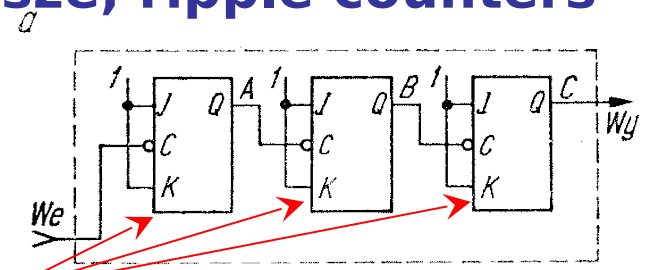
Uniwersalny układ czasowy - licznik – TIMER

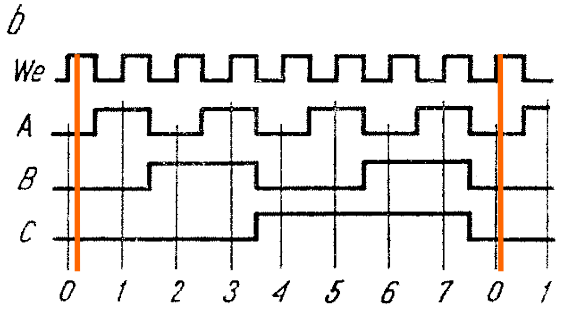
Przeważnie mikrokontroler posiada od jednego/dwóch do kilkunastu takich układów…

- zliczanie impulsów (zboczy sygnałów zewn. i wewn., zdarzeń itp.),
- **odmierzanie czasu (np. wykonania programu, delay, zegar/kalendarz),**
- **generowanie przerwań z ustaloną częstotliwością,**
- **obsługa urządzeń wejściowych np. enkodera obrotowego,**
- **pomiar częstotliwości/okresu/współczynnika wypełnienia fali prostokątnej (LAB),**
- generowanie fali prostokątnej (o danej częstotliwości i wypełnieniu - PWM),
- układ czuwający/zabezpieczający – watchdog.

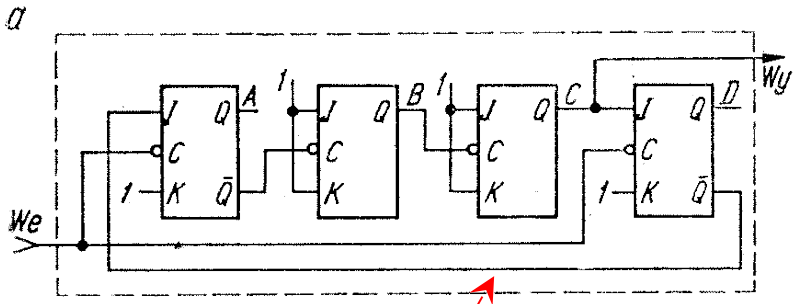
Liczniki – najprostsze, ripple counters

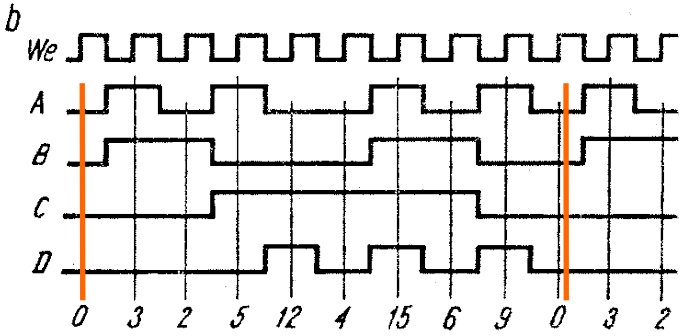
dzielniki częstotliwości impulsów wejściowych





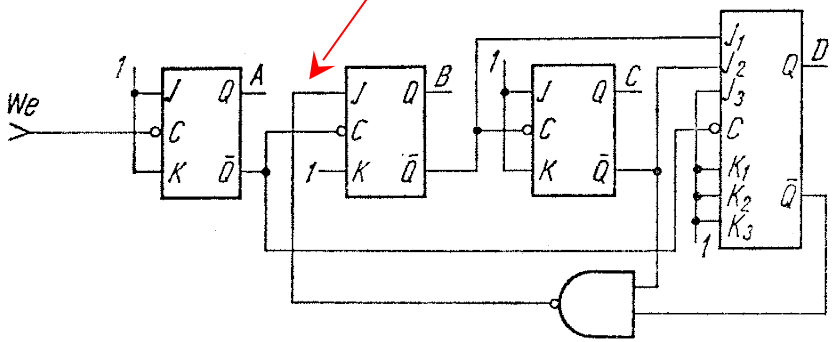
przez 8 = 2^3





przez 9

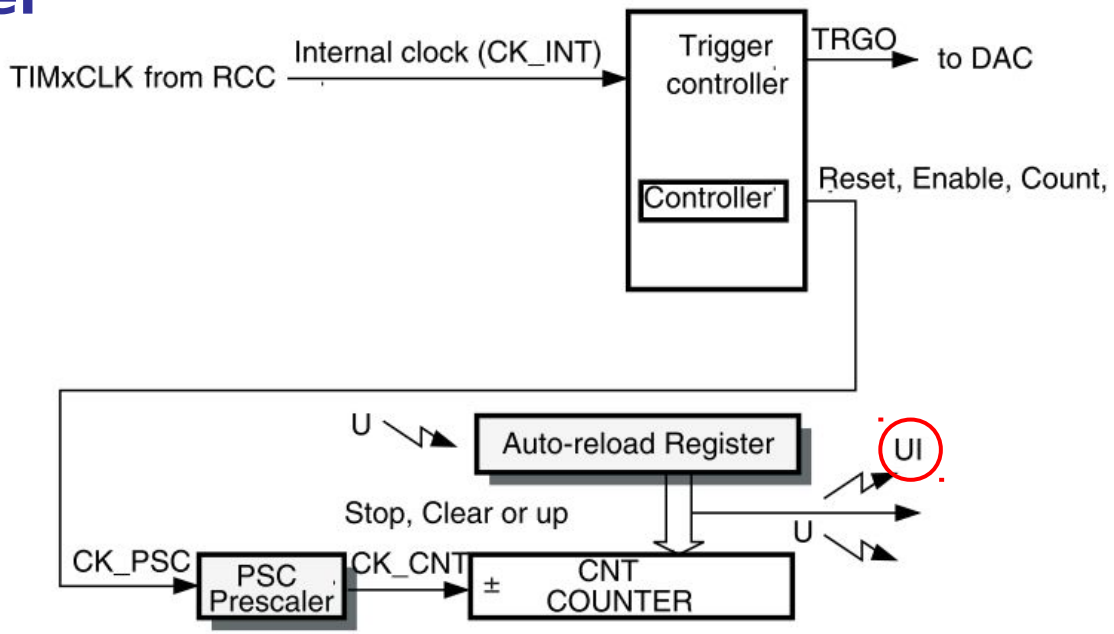
sprzężenia zwrotne



n	D	C	B	A
0	0	0	0	0
9	1	0	0	1
8	1	0	0	0
7	0	1	1	1
6	0	1	1	0
5	0	1	0	1
4	0	1	0	0
3	0	0	1	1
2	0	0	1	0
1	0	0	0	1

licznik zliczający wstecz (modulo 10, od 9 do 0, kod BCD)

STM32F429 – Basic Timer



- Licznik (16-bitowy, 65536 stanów) zlicza impulsy (zbocza sygnału CK_CNT) „w górę” – od 0 do 2^{16} = 65535 lub ustalonej wartości w rejestrze ARR.

- W chwili przepełnienia licznika może być generowane zgłoszenie: przerwania (UI), ew. transferu DMA.

- Po przepełnieniu, w zależności od ustawień, licznik może się zatrzymać albo zostać przeładowany wartością początkową (0) i powtarzać odliczanie.

- W każdej chwili licznik może być programowo przekonfigurowany, zatrzymany uruchomiony, a jego chwilowa wartość odczytana lub zmieniona.

STM32F429 - włączenie/wyłączanie timerów (i innych peryferiów)

Blok **RCC – Reset Clock Control**

Rejestr **APB1ENR** (**APB1 Enable Register**, po resecie same zera)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8 EN	UART7 EN	DAC EN	PWR EN	Reser-ved	CAN2 EN	CAN1 EN	Reser-ved	I2C3 EN	I2C2 EN	I2C1 EN	UART5 EN	UART4 EN	USART 3 EN	USART 2 EN	Reser-ved
rw	rw	rw	rw		rw	rw		rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 EN	SPI2 EN	Reserved		WWDG EN	Reserved		TIM14 EN	TIM13 EN	TIM12 EN	TIM7 EN	TIM6 EN	TIM5 EN	TIM4 EN	TIM3 EN	TIM2 EN
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

Timery podłączone są do jednej z szyn: APB1 lub APB2.

AHB – Advanced High-Performance Bus

APB – Advanced Peripheral Bus

Boundary address	Peripheral	Bus	Register map
0x4000 7400 - 0x4000 77FF	DAC		Section 14.5.15: DAC register map on page 456
0x4000 7000 - 0x4000 73FF	PWR		Section 5.6: PWR register map on page 151
0x4000 6800 - 0x4000 6BFF	CAN2		Section 32.9.5: bxCAN register map on page 1121
0x4000 6400 - 0x4000 67FF	CAN1		
0x4000 5C00 - 0x4000 5FFF	I2C3		Section 27.6.11: I2C register map on page 875
0x4000 5800 - 0x4000 5BFF	I2C2		
0x4000 5400 - 0x4000 57FF	I2C1		
0x4000 5000 - 0x4000 53FF	UART5		Section 30.6.8: USART register map on page 1021
0x4000 4C00 - 0x4000 4FFF	UART4		
0x4000 4800 - 0x4000 4BFF	USART3		
0x4000 4400 - 0x4000 47FF	USART2		
0x4000 4000 - 0x4000 43FF	I2S3ext		Section 28.5.10: SPI register map on page 928
0x4000 3C00 - 0x4000 3FFF	SPI3 / I2S3	APB1	
0x4000 3800 - 0x4000 3BFF	SPI2 / I2S2		
0x4000 3400 - 0x4000 37FF	I2S2ext		
0x4000 3000 - 0x4000 33FF	IWDG		Section 21.4.5: IWDG register map on page 715
0x4000 2C00 - 0x4000 2FFF	WWDG		Section 22.6.4: WWDG register map on page 722
0x4000 2800 - 0x4000 2BFF	RTC & BKP Registers		Section 26.6.21: RTC register map on page 839
0x4000 2000 - 0x4000 23FF	TIM14		Section 19.5.12: TIM10/11/13/14 register map on page 697
0x4000 1C00 - 0x4000 1FFF	TIM13		
0x4000 1800 - 0x4000 1BFF	TIM12		Section 19.4.13: TIM9/12 register map on page 687
0x4000 1400 - 0x4000 17FF	TIM7		Section 20.4.9: TIM6 and TIM7 register map on page 710
0x4000 1000 - 0x4000 13FF	TIM6		

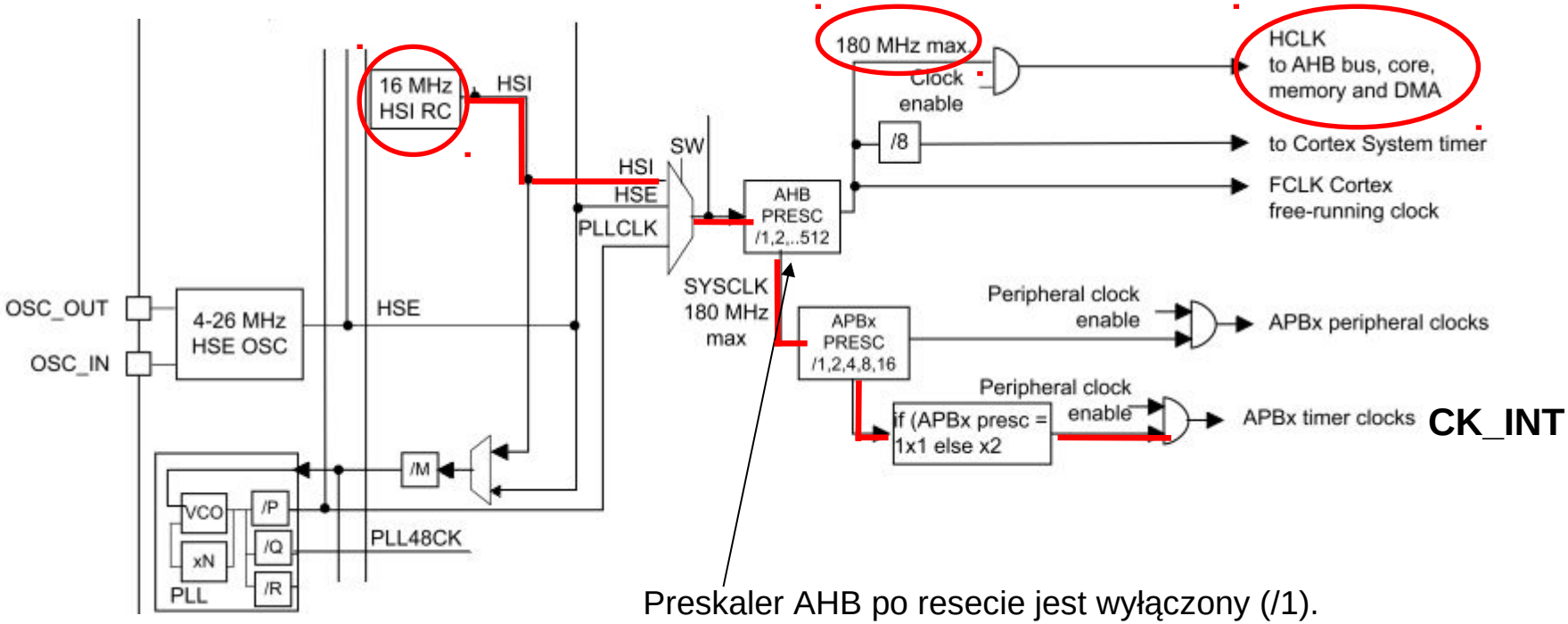
STM32F429 – częstotliwość zliczania

Częstotliwość taktowania timera:

częstotliwość sygnału zegarowego z generatora (lub PLL) podzielona przez:

- preskaler magistrali AHB – domyślnie /1,
- preskaler magistrali APB (jeżeli APB != 1 to APB *= 2), domyślnie /1,
- preskaler Timera (PSC),
- pojemność licznika (ARR).

Domyślnie, po resecie, układ jest taktowany z wewnętrznego generatora RC f = 16 MHz

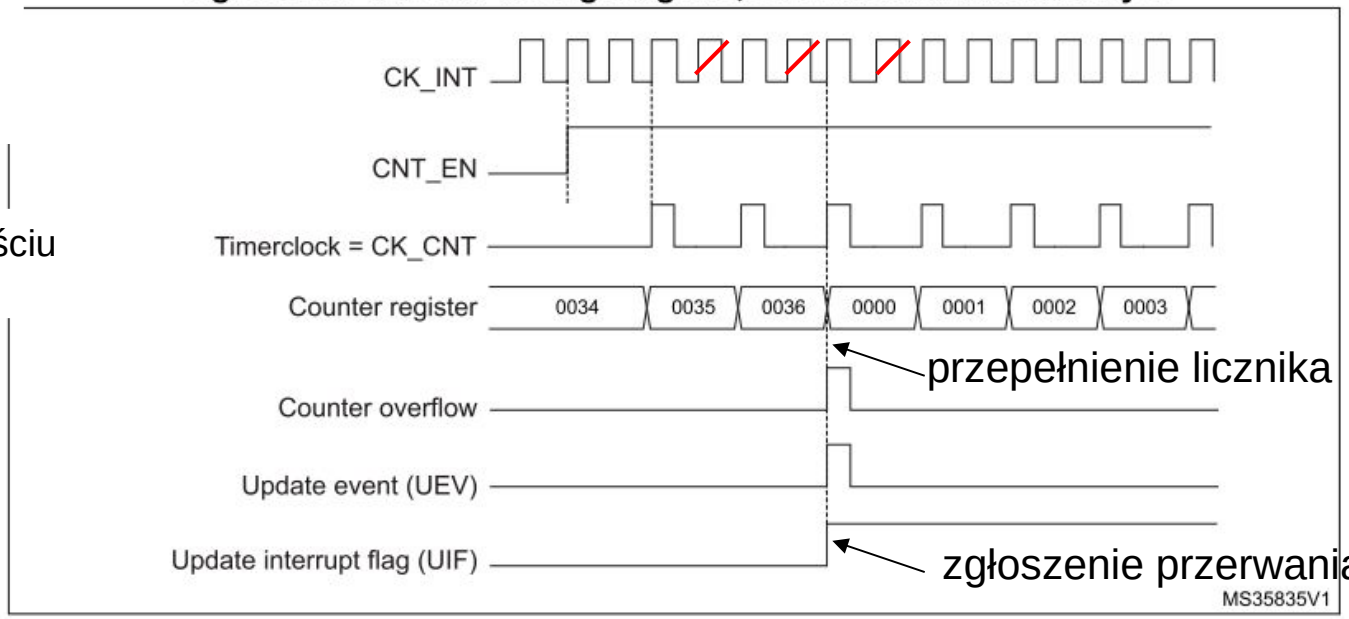


Preskaler AHB po resecie jest wyłączony (/1).

Zasada działania programowalnego preskalera (PSC)

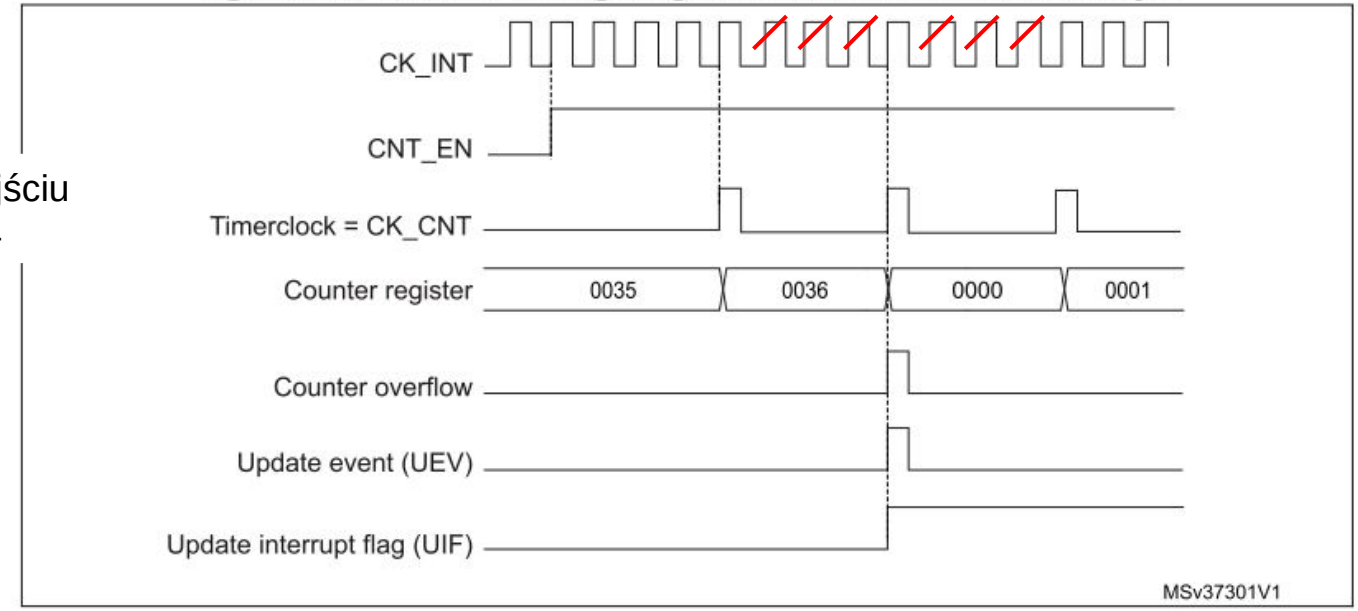
sygnał taktujący po przejściu przez preskaler (PSC) /2

Figure 207. Counter timing diagram, internal clock divided by 2



sygnał taktujący po przejściu przez preskaler (PSC) /4

Figure 208. Counter timing diagram, internal clock divided by 4



ARR = 36
czyli licznik zlicza od 0 do 36.

Preskaler „dzieli” częstotliwość taktowania przez wartość o 1 większą niż wpisana w PSC.

Prosty przykład - delay

Stan po resecie:

- częstotliwość SYSCLK = 16 MHz,
- preskalery AHB i APB wyłączone (/1),
- na wejście timera (CK_INT) trafia sygnał (fala prostokątna) 16 MHz.

```
RCC->APB1ENR |= (1<<5);     // włącz sygnał zegarowy timera TIM7
TIM7->PSC = 15999;          // preskaler = 16000, 16 MHz / 16000 = 1 kHz
                            // licznik CNT zlicza z częstotliwością 1 kHz
TIM7->ARR = 1999;           // ew. ogranicz pojemność licznika do 2000 stanów
                            // (nie jest to niezbędne)
TIM7->CR1 |= 1              // włącz licznik w Control Register 1

//---------------------------------


void del(uint32_t d){       // d - w milisekundach
   TIM7 -> CNT = 0;         // wyzeruj licznik
   while((TIM7 -> CNT) < d); // czekaj aż zliczy d impulsów (1 impuls – 1 ms)
}
```

Wywołanie np. del(500); - zwłoka 0.5 s

Prosty przykład – delay

- W tym przykładzie licznik pracuje cały czas.

- Można wykorzystać również tryb ONE-PULSE.

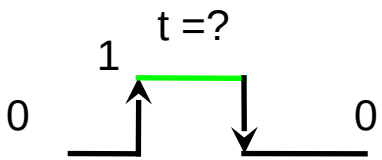
- **Nie uzwględniamy narzutu czasowego**
  - **wywołania funkcji,**
  - **dostępu do I/O: zerowanie i odczyt wartości chwilowej licznika**
  - **sprawdzenia warunku w „while”**

  - **zakłóceń od działających „w tle” przerwań…**

- W STM32f429 liczniki są 16-bitowe lub 32-bitowe.

- Gdyby pojemność jednego licznika (np. maksymalny czas od zerowania do przepełnienia przy danej częstotliwości pracy) była niewystarczająca, liczniki można łączyć kaskadowo (przepełnienie jednego taktuje następny).

Bardziej zaawansowany przykład (laboratoria): pomiar czasu/długości impulsu – tryb timera „input capture”



Zadanie można zrealizować konwencjonalnie, np.:

- zatrzymaj i wyzeruj licznik (ew. jeśli już uruchomiony - niech pracuje cały czas),

- czekaj na narastające zbocze (np. pętla while…),
- z chwilą pojawienia się stanu wysokiego uruchom licznik (ew. wyzeruj),

- czekaj na opadające zbocze (licznik w tym czasie pracuje),

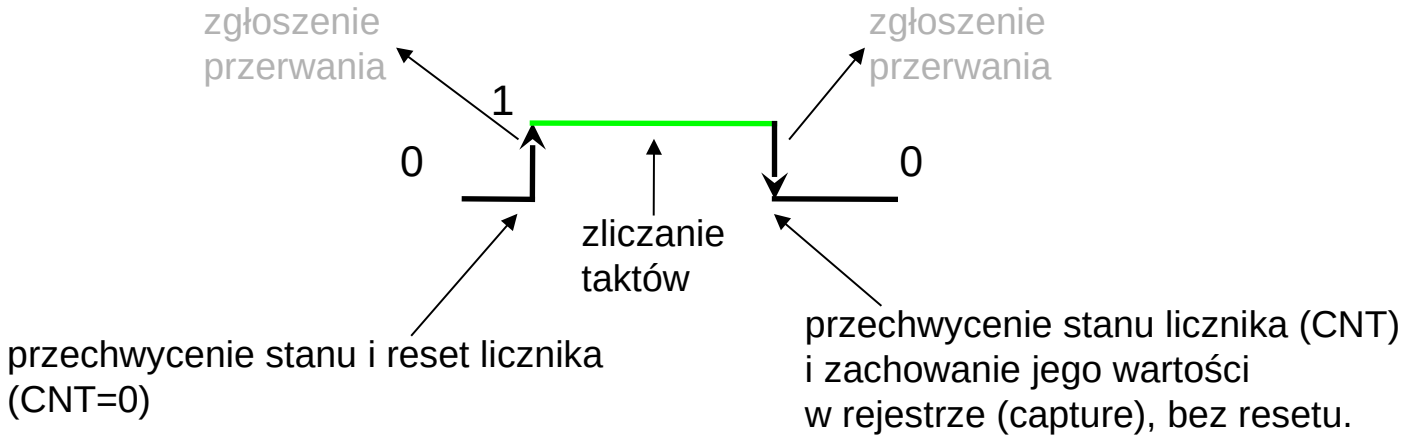
- z chwilą pojawienia się stanu niskiego odczytaj zliczoną wartość (ew. zatrzymaj licznik i potem odczytaj),

- odczytaną zawartość licznika wyskaluj (liczba impulsów * okres sygnału taktującego).

Znowu ten sam problem: każda z tych prostych operacji wymaga czasu i programowego dostępu do I/O. Dodatkowo narzut czasowy może zależeć od częstotliwości taktowania CPU, optymalizacji kodu, obciążenia CPU innymi zadaniami i przerwaniami...

Lepiej i prościej wykorzystać tryb „**input caputure**” (przechwytywania wartości chwilowej licznika) w jaki jest obecnie wyposażona większość timerów.
Idea działania (po odpowiedniej konfiguracji licznika):

• licznik pracuje w sposób ciągły: zlicza impulsy sygnału zegarowego o stałej, znanej częstotliwości.



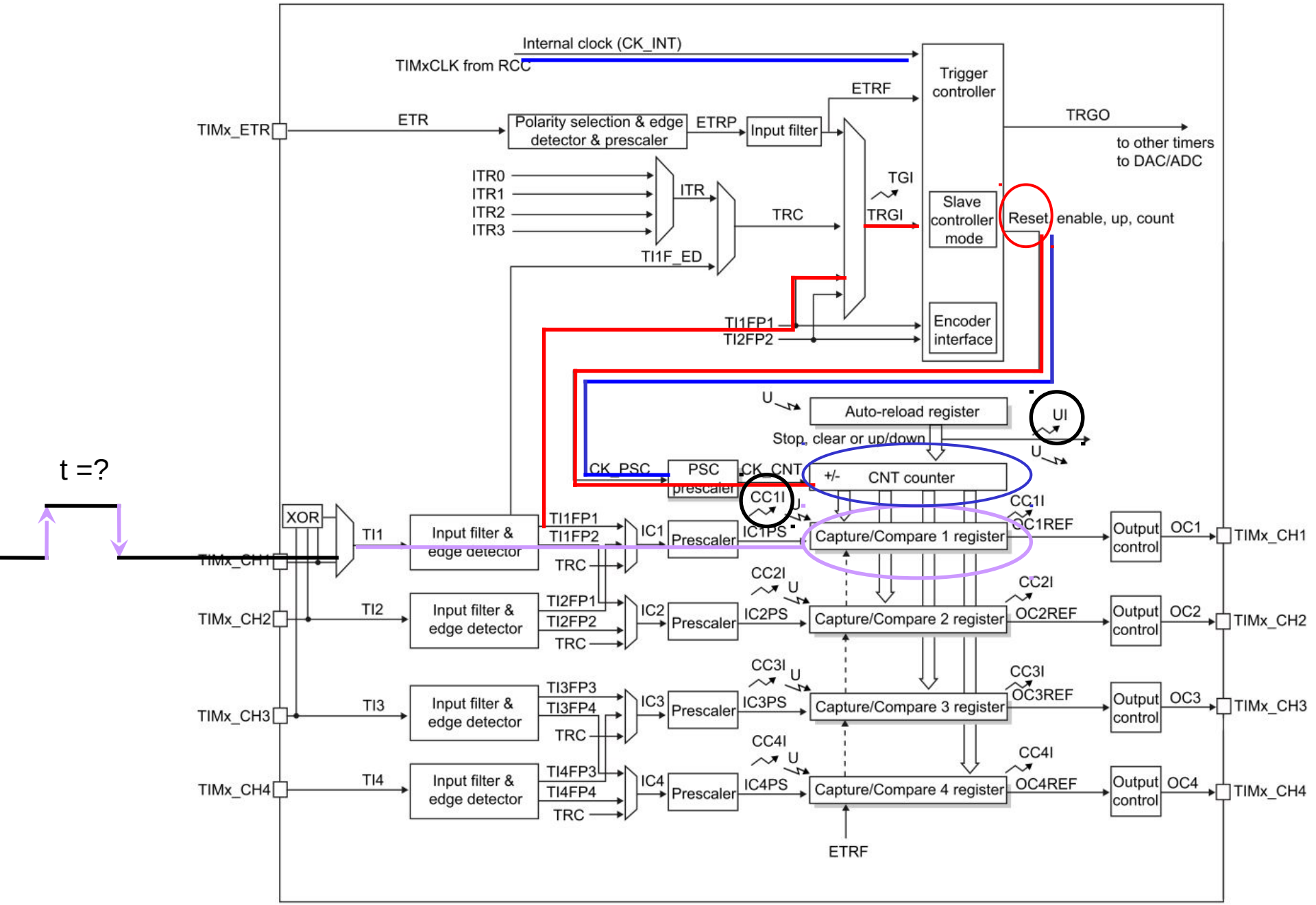
• W chwili wykrycia odpowiedniego zbocza (tutaj: rosnącego) sygnału na wejściu jednego z czterch kanałów, chwilowa wartość licznika jest przechwytywana i zapisywana w dedykowanym rejestrze (capture register). Niezwłocznie układ sterowania (tutaj: slave controller) zeruje licznik.

• Licznik rozpoczyna zliczanie „taktów”, dopóki na wejściu utrzymuje się stan wysoki.

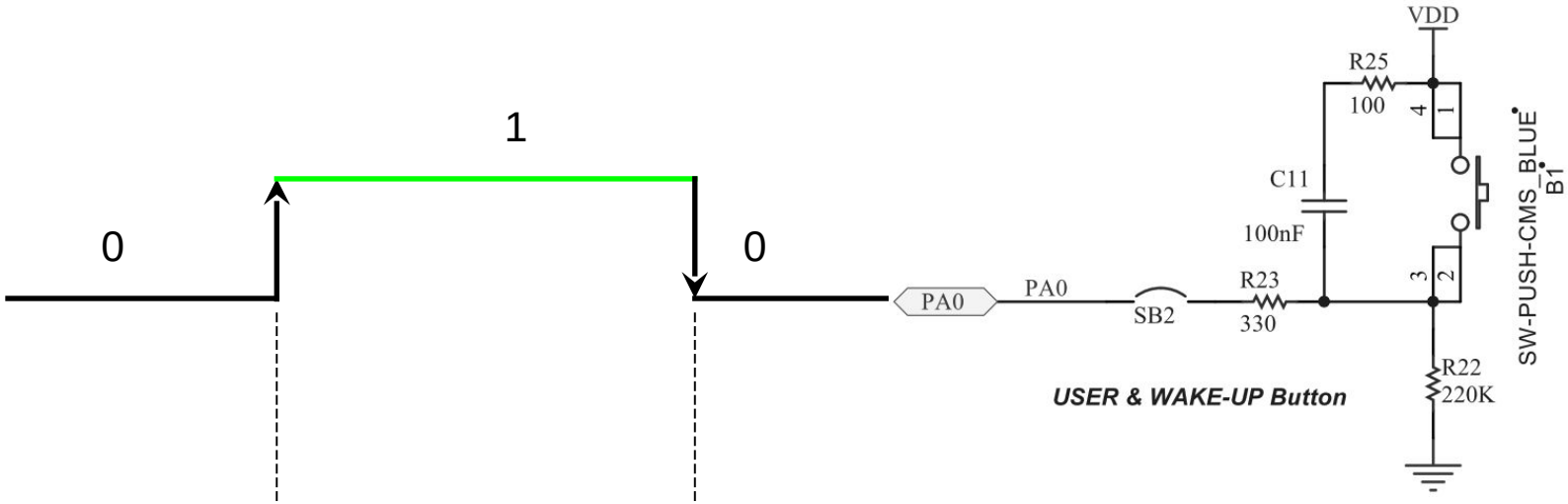
• W chwili powrotu stanu sygnału wejściowego do stanu niskiego, aktualna wartość licznika ponownie zostaje przechwycona i przeniesiona do rejestru capture.

• Podczas przechwytywania, jak i podczas przepełnienia licznika może zostać wygenerowane zgłoszenie obsługi przerwania. W ISR można zrealizować obsługę przekroczenia zakresu pomiarowego.

Pomiar czasu/długości impulsu tryb input capture



Pomiar czasu/długości impulsu tryb input capture



Podstawowe zalety trybu input capture

(w stosunku do programowego sterowania licznikiem i *pollingu*):

- wszystko dzieje się w dedykowanym układzie, poza rdzeniem procesora;
  pomijając konfigurację timera, obsługa programowa jest zredukowana do minimum,

- szybkość działania: czasy propagacji sygnałów w układzie cyfrowym są krótkie i stałe:

  niezależne od np. obciążenia procesora, jego taktowania, liczby i priorytetów przerwań oczekujących na obsługę i sposobu optymalizacji i kompilacji kodu. Zwłoka związana z rozpoczęciem i zakończeniem pomiaru jest minimalna i stała.

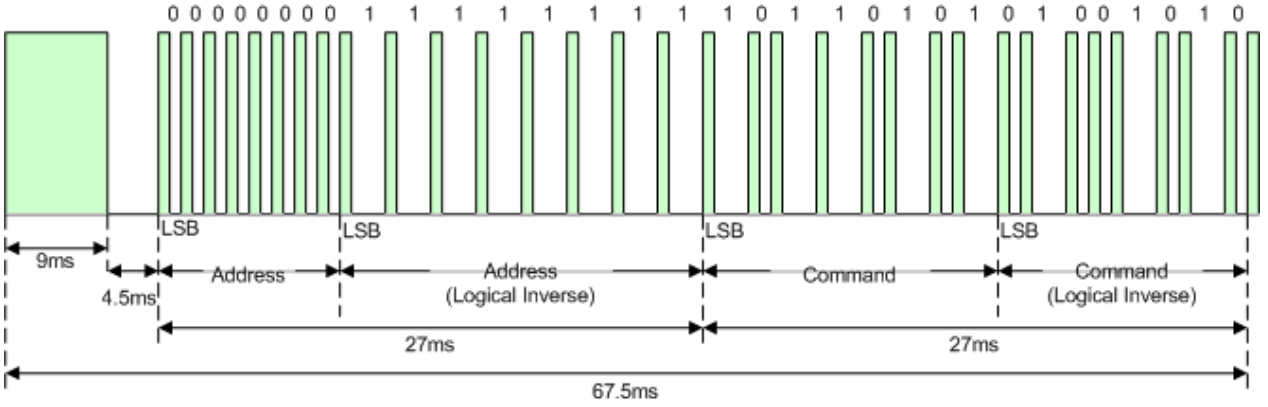
„Przechwycona” wartość licznika może być odczytana w dogodnym momencie. Jest zachowana w dedykowanym rejestrze, więc opóźnienie jej odczytu nie jest krytyczne.

Zastosowania praktyczne trybu input capture

(inne niż pomiar czasu między dwoma kolejnymi zdarzeniami…):

dekodowanie sygnałów wykorzystujących modulację gęstości i szerokości impulsów.

Pilot TV:
transmisja
w podczerwieni
Kodów sterujących RTV.



Protokół firmy NEC.

DCF77 - transmisja radiowa wzorca czasu
f = 77.5 kHz.