

Podstawy systemów mikroprocesorowych

Wykład nr 3

Przerwania i liczniki

dr inż. Dariusz Tefelski

na podstawie wykładu dr hab. Piotra Fronczaka

http://fizyka.if.pw.edu.pl/~labe/index.php/Wyklad_PSM

Przerwanie

- Warunek lub zdarzenie, które przerywa normalny ciąg instrukcji w programie
- Przerwanie wywołuje między instrukcjami skok do procedury obsługi przerwania
- Gdy procedura obsługi przerwania kończy się, wznowiany jest normalny ciąg instrukcji w programie

Źródła przerwań

- Przerwania dzielimy w ogólności na
 - › wewnętrzne lub zewnętrzne
 - › Programowe lub sprzętowe
- Przerwanie zewnętrzne jest wywoływane przez urządzenie/układ poza mikrokontrolerem.
- Przerwanie wewnętrzne jest wywoływane przez jeden z podukładów mikrokontrolera

Źródła przerwań

- Przerwanie sprzętowe zachodzi w wyniku zmiany stanu jednego z układów fizycznych
- Przerwanie programowe zachodzi w wyniku wykonania instrukcji programu

System przerwań w mikrokontrolerze AVR

- ATMega32 reaguje na 21 różnych rodzajów przerwań
- Przerwania są numerowane zgodnie z ich priorytetem od 1 do 21
- Przerwanie RESET ma numer 1
- Każde przerwanie powoduje skok do specyficznego dla niego adresu w pamięci programu
 - › Procedura przerwania RESET ma adres \$0000

Wektory przerwań

- Procedura przerwania k jest umieszczona pod adresem $2(k-1)$ w pamięci programu
 - › Adres \$0000 – przerwanie RESET
 - › Adres \$0002 przerwanie zewnętrzne 0
 - › Adres \$0004 przerwanie zewnętrzne 1
- Ponieważ jest miejsce tylko na jedną lub dwie instrukcje, procedura przerwania rozpoczyna się od skoku w inne miejsce pamięci programu, gdzie znajduje się reszta kodu.

Vector No.	Program Address ⁽²⁾	Source	Interrupt Definition
1	\$000 ⁽¹⁾	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	\$002	INT0	External Interrupt Request 0
3	\$004	INT1	External Interrupt Request 1
4	\$006	TIMER2 COMP	Timer/Counter2 Compare Match
5	\$008	TIMER2 OVF	Timer/Counter2 Overflow
6	\$00A	TIMER1 CAPT	Timer/Counter1 Capture Event
7	\$00C	TIMER1 COMPA	Timer/Counter1 Compare Match A
8	\$00E	TIMER1 COMPB	Timer/Counter1 Compare Match B
9	\$010	TIMER1 OVF	Timer/Counter1 Overflow
10	\$012	TIMER0 OVF	Timer/Counter0 Overflow
11	\$014	SPI, STC	Serial Transfer Complete
12	\$016	USART, RXC	USART, Rx Complete
13	\$018	USART, UDRE	USART Data Register Empty
14	\$01A	USART, TXC	USART, Tx Complete
15	\$01C	ADC	ADC Conversion Complete
16	\$01E	EE_RDY	EEPROM Ready
17	\$020	ANA_COMP	Analog Comparator
18	\$022	TWI	Two-wire Serial Interface
19	\$024	INT2	External Interrupt Request 2
20	\$026	TIMER0 COMP	Timer/Counter0 Compare Match
21	\$028	SPM_RDY	Store Program Memory Ready

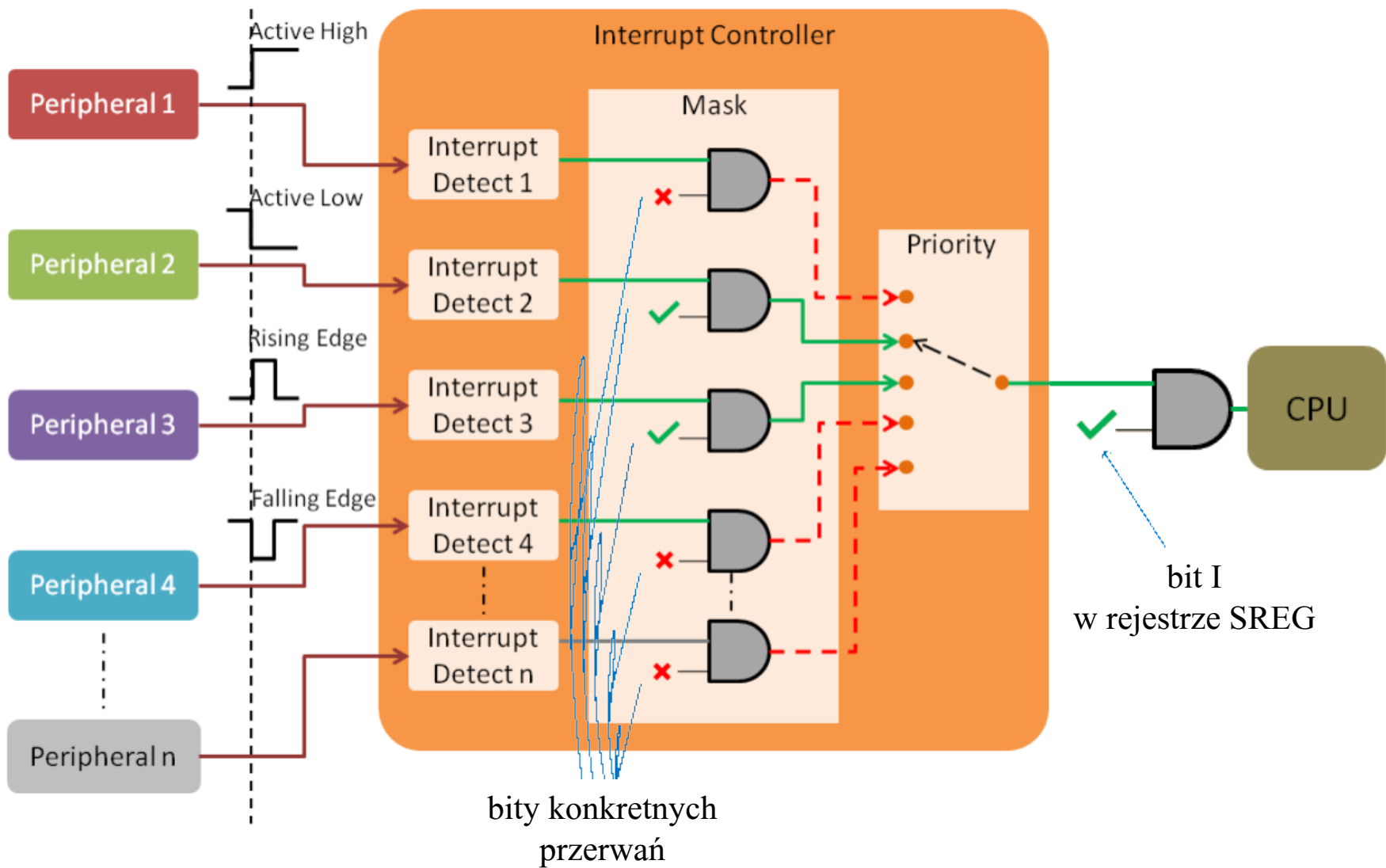
Włączanie przerwań

- Każde potencjalne źródło przerwań można indywidualnie włączać i wyłączać
- Wyjątek – przerwanie RESET – nie można wyłączyć
- By przerwania mogły działać, trzeba ustawić globalny bit przerwań w rejestrze SREG
- Przerwanie RESET zachodzi niezależnie od ustawienia tego bitu.

Bit	7	6	5	4	3	2	1	0
SREG	I							

Działanie przerwań

- Jeżeli
 - › Globalny bit przerwań jest ustawiony
 - › ORAZ bit konkretnego przerwania jest ustawiony
 - › ORAZ zaszedł warunek przerwania
- To wystąpi przerwanie
- Co się właściwie dzieje?
 - › Po zakończeniu aktualnej instrukcji,
 - Aktualna wartość licznika programu PC odkładana jest na stos
 - Kasowany jest globalny bit przerwań
 - Właściwy adres procedury obsługi przerwania jest umieszczany w liczniku programu

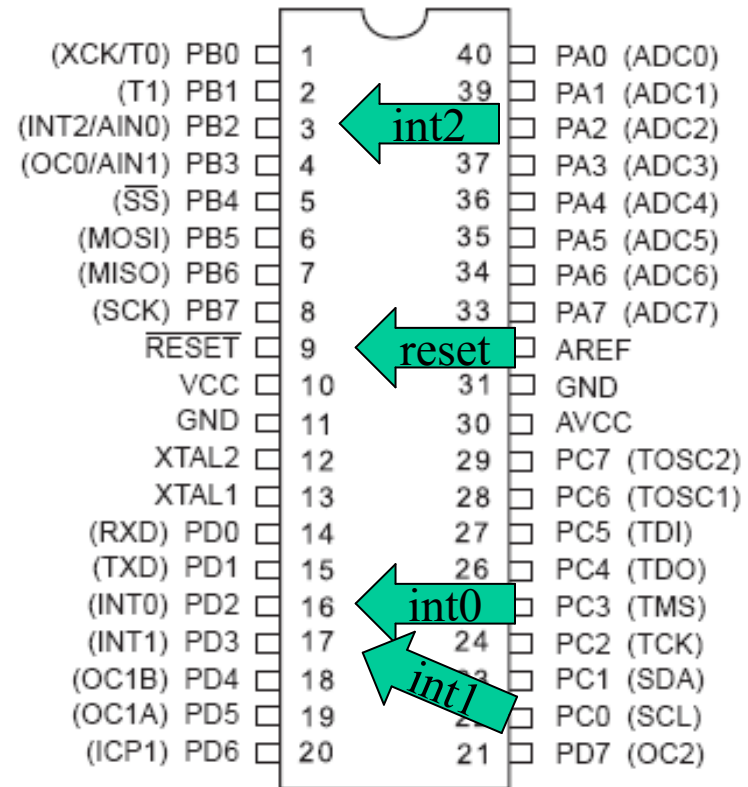


Powrót z procedury przerwania

- Po zakończeniu procedury obsługi przerwania
 - › Adres ze stosu jest umieszczany w liczniku programu
 - › Ustawiany jest globalny bit przerwań
- To powoduje kontynuację uprzednio przerwanego programu
 - › Przynajmniej jedna instrukcja zostanie wykonana zanim nastąpi inne przerwanie

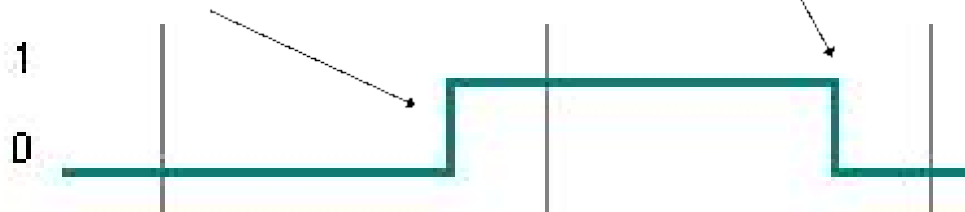
Przerwania zewnętrzne

- ATMega32 reaguje na 4 różne przerwania zewnętrzne – sygnały wprowadzone na odpowiednie wejścia mikrokontrolera
- RESET (pin 9)
- INT0 (pin 16 – PD2)
- INT1 (pin 17 – PD3)
- INT2 (pin 3 – PB3)



Konfiguracja przerwań zewnętrznych

- Wyzwalane poziomem
 - › Gdy stan wejścia jest niski (logiczne 0)
- Wyzwalane zboczem
 - › Stan wejścia zmienił się
 - › Zboczem opadającym (z 1 do 0)
 - › Zboczem rosnącym (z 0 do 1)



MCUCR (MCU Control Register)

MCUCR – MCU Control
Register

7	6	5	4	3	2	1	0
				ISC11	ISC10	ISC01	ISC00

ISC – Interrupt Sense Control bits

INT0

INT1

ISC01	ISC00	opis
0	0	Poziom niski na INT0 generuje przerwanie
0	1	Zmiana stanu logicznego na INT0 generuje przerwanie
1	0	Opadające zbocze na INT0 generuje przerwanie
1	1	Narastające zbocze na INT0 generuje przerwanie

Wyzwalanie poziomem

- Procesor próbuje poziomy na pinach INT0 i INT1 w każdym cyklu zegara
- Impulsy krótsze niż jeden cykl mogą być niezauważone
- Przerwanie niskim poziomem nastąpi tylko, gdy niski stan na wejściu będzie po zakończeniu aktualnie wykonywanej instrukcji

MCUCSR – rejestr kontroli i statusu mikrokontrolera (Control and Status Register)

MCUCSR – MCU Control and Status Register

7	6	5	4	3	2	1	0
	ISC2						

INT2

Jedynie możliwości sposobu przerwań INT2

- › 0 – zbocze opadające
- › 1 – zbocze rosnące

GICR - główny rejestr kontroli przerwań (General Interrupt Control Register)

Tu włączamy przerwania zewnętrzne

GICR – General Interrupt Control Register

7	6	5	4	3	2	1	0
INT1	INT0	INT2					

GIFR – główny rejestr flag przerwań (General Interrupt Flag Register)

GICR – General Interrupt Flag Register

7	6	5	4	3	2	1	0
INTF1	INTF0	INTF2					

- Ustawiona flaga (1) oznacza, że wystąpiło żądanie przerwania przez urządzenie zewnętrzne.
- Te flagi używane są tylko dla przerwań zboczem.
- Gdy mikrokontroler przechodzi do procedury obsługi, następuje zerowanie znacznika.
- Gdy nie korzystamy z przerwań, flagi nadal mogą być ustawiane. Choć samo żądanie jest ignorowane, to możemy wykorzystać tę informację w programie (odpytywanie – pooling). Zerowanie trzeba wykonać manualnie (UWAGA: PRZEZ WPISANIE 1!!!!).

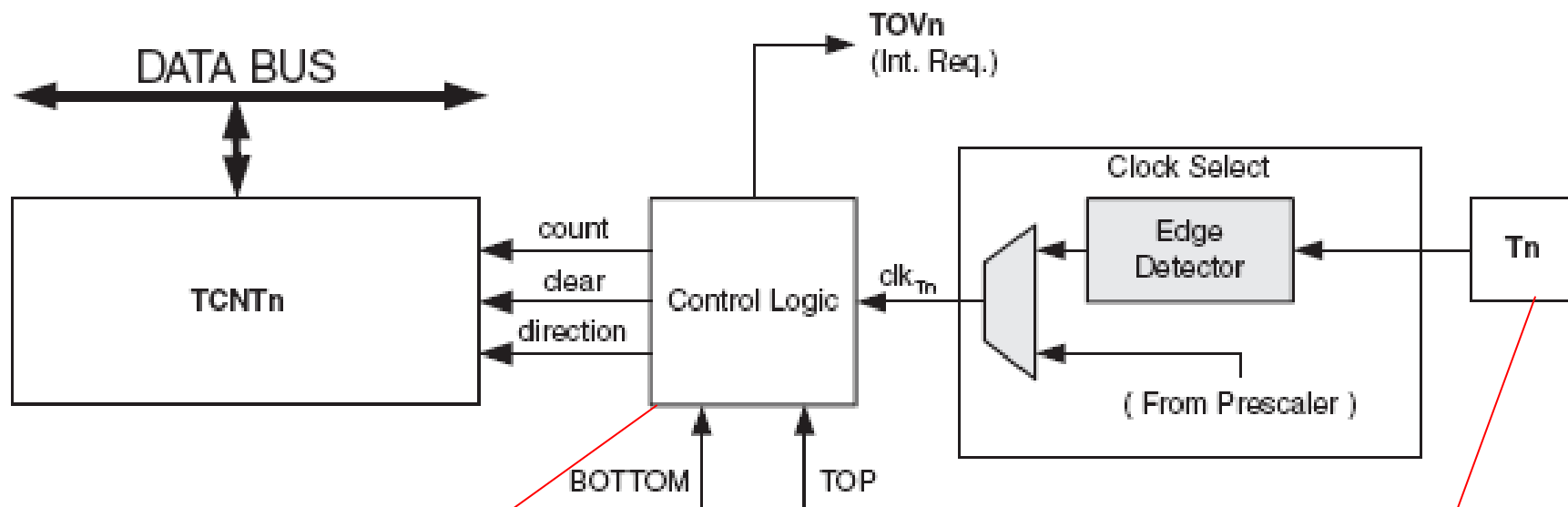
Przerwania programowe

- Jeśli piny przerwań zewnętrznych są skonfigurowane jako wyjścia, możemy wpisać do nich 0 i 1 programowo.
 - › To może wyzwolić przerwania zgodnie z ustawieniami dla przerwań zewnętrznych.

Liczniki (timers/counters)

- W mikrokontrolerze ATmega32 dostępne są trzy moduły liczników (timerów), dwa 8-bitowe i jeden 16-bitowy. Oprócz tych istnieje także specjalizowany licznik tzw. Watchdog Timer.
- Stan każdego licznika jest przechowywany w odpowiednim rejestrze
- Licznik może być taktowany zewnętrznym lub wewnętrznym sygnałem zegarowym.
- Licznik może stanowić źródło przerwań (przepełnienie, zrównanie)

8-bitowy licznik



Jednostka zliczająca:

Główną częścią licznika jest 8-bitowa programowalna dwukierunkowa jednostka logiczna. W zależności od trybu pracy licznik jest zwiększany, zmniejszany lub zerowany w każdym cyklu zegara (clk_{Tn}). Bit przepełnienia ($TOVn$) może być użyty do generowania przerwań.

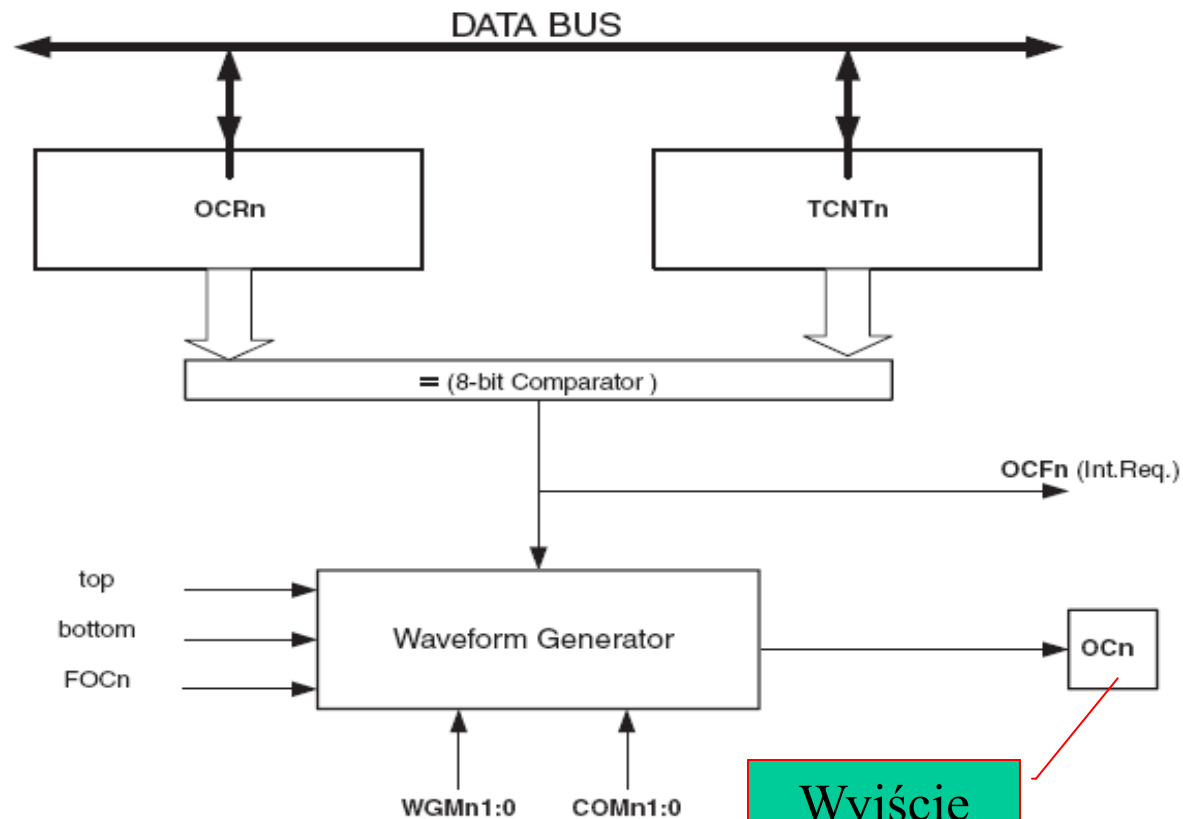
Sygnal zegara
zewnętrznego

Zdarzenia licznika

- Przekroczenie zakresu (overflow)
 - › Przekroczenie zakresu występuje, gdy zliczana wartość przekracza 0xFF i staje się równa 0x00
- Osiągnięcie wartości zadanej (compare match)
 - › Gdy wartość licznika równa jest zawartości odpowiedniego rejestru specjalnego

Komparator (compare unit)

Wyjście rejestru komparatora (OCRn) jest stale porównywane z zawartością licznika. Wynik porównania może być użyty przez generator przebiegu do wytwarzania PWM lub sygnału o zmiennej częstotliwości na wyjściu porównania (OCn). Wynik porównania będzie również ustawiał Bit Porównania (OCFn), który może być użyty do wytwarzania przerwania (od porównania).



Wyjście
na pin

Odpytywanie liczników (pooling)

- Stan licznika można badać w programie głównym poprzez tzw. pooling
- Odczyt flag w rejestrze TIFR (Timer Interrupt Flag Register)
- Przekroczenie zakresu lub osiągnięcie wartości zadanej zmieniają stan odpowiednich bitów w rejestrze TIFR
 - TOVn oraz OCFn ($n=0, 1$, lub 2)
 - Licznik 1 (16-bitowy) ma dwa rejestry porównań OCR: 1A oraz 1B

Automatyczne funkcje liczników

- Liczniki 0 i 2 (8-bitowe) można skonfigurować tak, by automatycznie kasowały, ustawiały, lub odwracały stan odpowiednich wyjść mikrokontrolera, gdy nastąpi przyrównanie do rejestru OCR_n
 - › Nie wymaga to czasu obliczeniowego CPU ani obsługi przerwania – wszystko na poziomie sprzętowym
 - › Odpowiedni pin OC_nx musi być ustawiony jako wyjście. Normalne działanie tego pinu jest niemożliwe
 - OC0 (PB3) OC2 (PD7)
 - OC1A (PD5) OC1B (PD4)

Źródła zegarowe liczników

- Liczniki do zliczania mogą wykorzystać zegar wewnętrzny lub zewnętrzny sygnał zegarowy.
- Częstotliwość zegara wewnętrznego może być zredukowana (preskalowana), dzięki czemu liczniki są taktowane wolniej
 - › Możliwy podział częstotliwości przez 8, 64, 256, 1024
 - › Timer2 umożliwia również preskalowanie sygnału zewnętrznego.

Rejestr kontrolny licznika 0

(TCCR0 Timer/Counter 0 Control Register)

7	6	5	4	3	2	1	0
FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00

bity 6,3 – WGM01:0 – rodzaje generowanych przebiegów:

bity te kontrolują sposób zliczania licznika: wartość maksymalną do której dąży licznik i rodzaj wytwarzanego przebiegu. Tryby pracy licznika: zwykły, zerowanie licznika podczas porównania i dwa rodzaje PWM (modulacji szerokości impulsu).

tryb	WGM01 (CTC0)	WGM00 (PWM0)	Rodzaj pracy licznika	Górna wartość licznika
0	0	0	Zwykły licznik	0xFF
1	0	1	PWM z poprawną fazą	0xFF
2	1	0	CTC	W rejestrze OCR0
3	1	1	Szybki PWM	0xFF

Rejestr kontrolny licznika 0

(TCCR0 Timer/Counter 0 Control Register)

7	6	5	4	3	2	1	0
FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00

bity 5, 4 – COM01:0 – rodzaj wyjścia dla trybu porównania

określają zachowanie się wyjścia komparatora licznika (końcówka OC0). Jeżeli jeden z bitów lub oba bity są ustawione to wyjście układu porównującego w liczniku zostaje dołączone do końcówki portu I/O. Rejestr kierunku portu (DDR) musi ustawić port jako wyjście. Gdy OC0 jest dołączone do końcówek portu, funkcje bitów COM01:0 zależą od trybu pracy licznika (czyli od ustawienia bitów WGM01:0).

Tryb zwykły i porównania

COM01	COM00	Określenie
0	0	Zwykły tryb pracy portu, OC0 odłączone
0	1	Porównanie zmienia OC0
1	0	Porównanie zeruje OC0
1	1	Porównanie ustawia OC0

Rejestr kontrolny licznika 0

(TCCR0 Timer/Counter 0 Control Register)

7	6	5	4	3	2	1	0
FOC0	WGMO0	COM01	COM00	WGM01	CS02	CS01	CS00

bity 2:0 – CS02:0 – wybór sygnału zegarowego:

te trzy bity decydują o źródle sygnału zegarowego dla licznika TC0. Jeżeli ustawione jest zewnętrzne źródło zegara dla licznika TC0, licznik będzie mógł być napędzany nawet jeżeli port ustawiony jest jako wyjście. Umożliwia to programowe taktowanie licznika (przez zmianę stanu końcówki portu).

CS02	CS01	CS00	Określenie
0	0	0	Brak sygnału zegarowego – licznik stoi
0	0	1	$\text{clk}_{I/O}$ (pełna prędkość – bez preskalera)
0	1	0	$\text{clk}_{I/O} / 8$ (z preskalerem)
0	1	1	$\text{clk}_{I/O} / 64$ (z preskalerem)
1	0	0	$\text{clk}_{I/O} / 256$ (z preskalerem)
1	0	1	$\text{clk}_{I/O} / 1024$ (z preskalerem)
1	1	0	Zewnętrzny sygnał zegarowy na końcówce T0 – takt przy zboczu opadającym
1	1	1	Zewnętrzny sygnał zegarowy na końcówce T0 – takt przy zboczu narastającym

Przerwania licznika

- TIMSK - Timer/Counter Interrupt Mask

TIMSK Timer/Counter Interrupt Mask
Register

7	6	5	4	3	2	1	0
OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0

Licznik 0

Output Compare Interrupt
Enable
Timer Overflow Interrupt
Enable

- TIFR - Timer/Counter Interrupt Flags

7	6	5	4	3	2	1	0
OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0

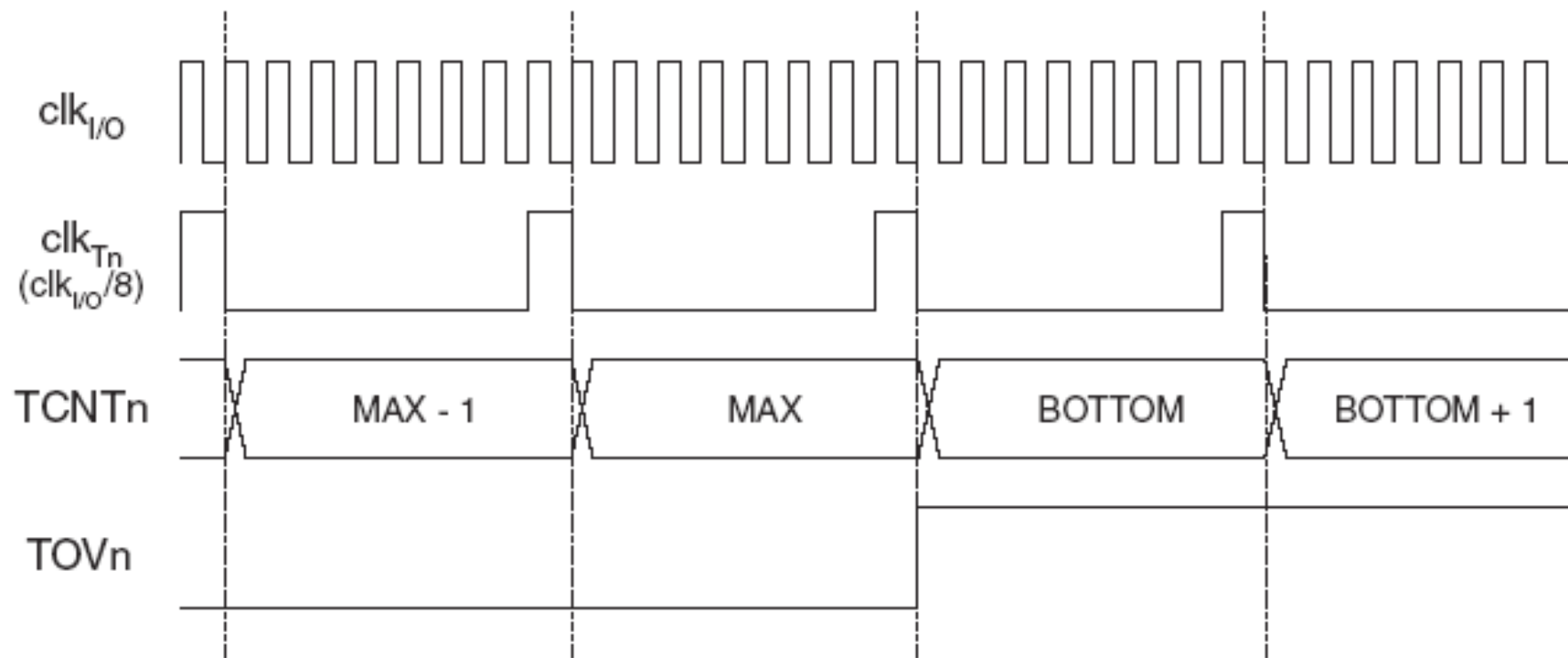
Licznik 0

Output Compare Flag
Timer Overflow Flag

Tryby pracy licznika

- Tryb normalny
 - › Licznik liczy do góry,
 - › Flaga przerwania ustawiona, gdy licznik osiągnie 0
- Tryb porównania (CTC)
 - › Licznik liczy do góry, aż osiągnie wartość rejestru OCR0
 - › W następnym kroku licznik przyjmuje wartość 0 i ustawiana jest flaga przerwania

Tryb normalny (preskaler 8)



Generacja fali prostokątnej ($T=512\mu s$)

```
#include<avr/io.h>
#define F_CPU 8000000
void delay()
{
    TCCR0=(1<<CS01); // Timer Clock = CLK/8 (starts timer too)
    while(!(TIFR&(1<<TOV0))); // Wait Until Overflow
    TIFR=TIFR|(1<<TOV0); // Clear TOV0
    TCCR0=0x00; // Stop Timer0
}
int main(void)
{
    DDRA=0xFF;
    PORTA=0x00;
    TCCR0=0x00;
    TCNT0=0x00;
    while(1){
        PORTA|=(1<<PA0);
        delay();
        PORTA&=~(1<<PA0);
        delay();
    }
}
```

Generacja fali prostokątnej (T=400us)

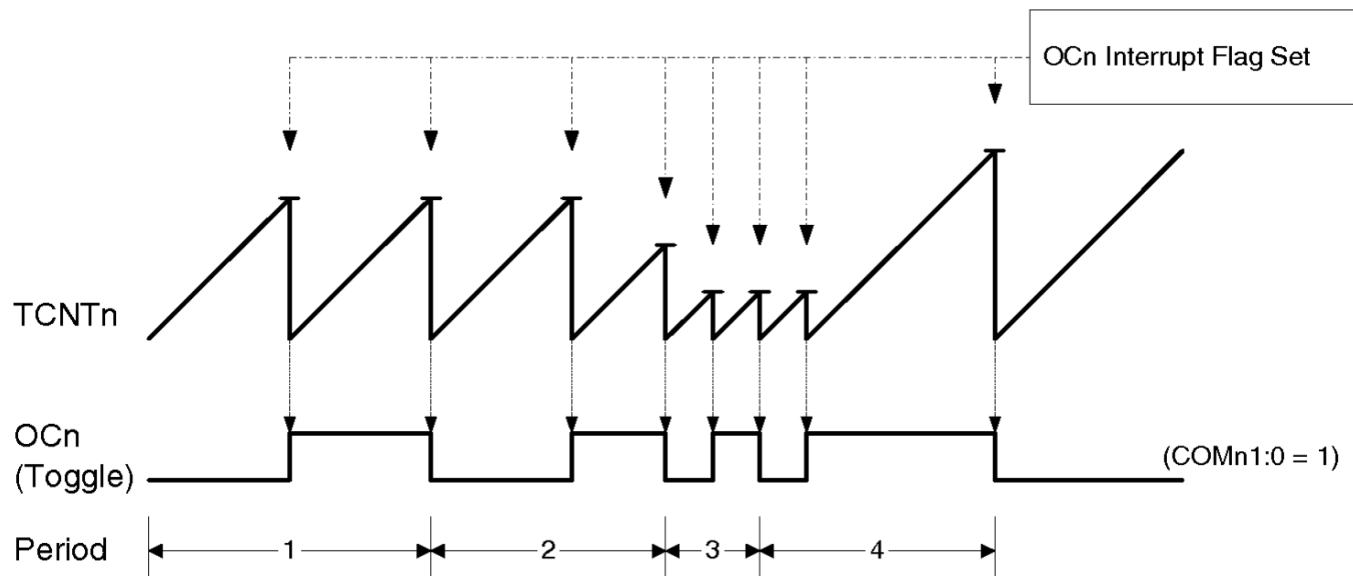
```
void delay()  
{  
    TCNT0=0x38; //TCNT0=56  
    TCCR0=(1<<CS01);  
    while(! (TIFR&(1<<TOV0)) );  
    TIFR=TIFR|(1<<TOV0);  
    TCCR0=0x00;  
}
```

Tryb CTC

(Clear Timer on Compare Match)

- W trybie tym (ustawienie bitów **WGM01:0 = 2**) jest używany rejestr **OCR0**.
- W tym trybie licznik jest zerowany, gdy zliczy do takiej wartości, jaka jest wpisana do rejestru **OCR0**.
- Rejestr ten definiuje największą wartość, do której może doliczyć licznik i w ten sposób decyduje o jego rozdzielczości.

Generowanie przebiegu prostokątnego w trybie CTC



- Przerwanie może być generowane podczas przepełniania się licznika przez użycie flagi **OCF0**.
- Przerwanie może zmienić wartość szczytów dla licznika.
- Ostrożnie ze zmianą wartości szczytowej zbliżonej do zera, podczas gdy licznik pracuje z wyłączonym preskalerem. Licznik będzie musiał zliczyć do wartości maksymalnej (0xFF) i po wyzerowaniu zliczyć do wartości, przy której wystąpi porównanie.
- Generowanie przebiegu prostokątnego w trybie CTC na wyjściu **OC0** uzyskuje się przez ustawienie bitów trybu Wyjścia Porównania (**COM01:0 = 1**).

Tryb CTC

- Generowany przebieg osiągnie najwyższą częstotliwość równą

$$f_{OC0} = f_{CLK_I/O} / 2,$$

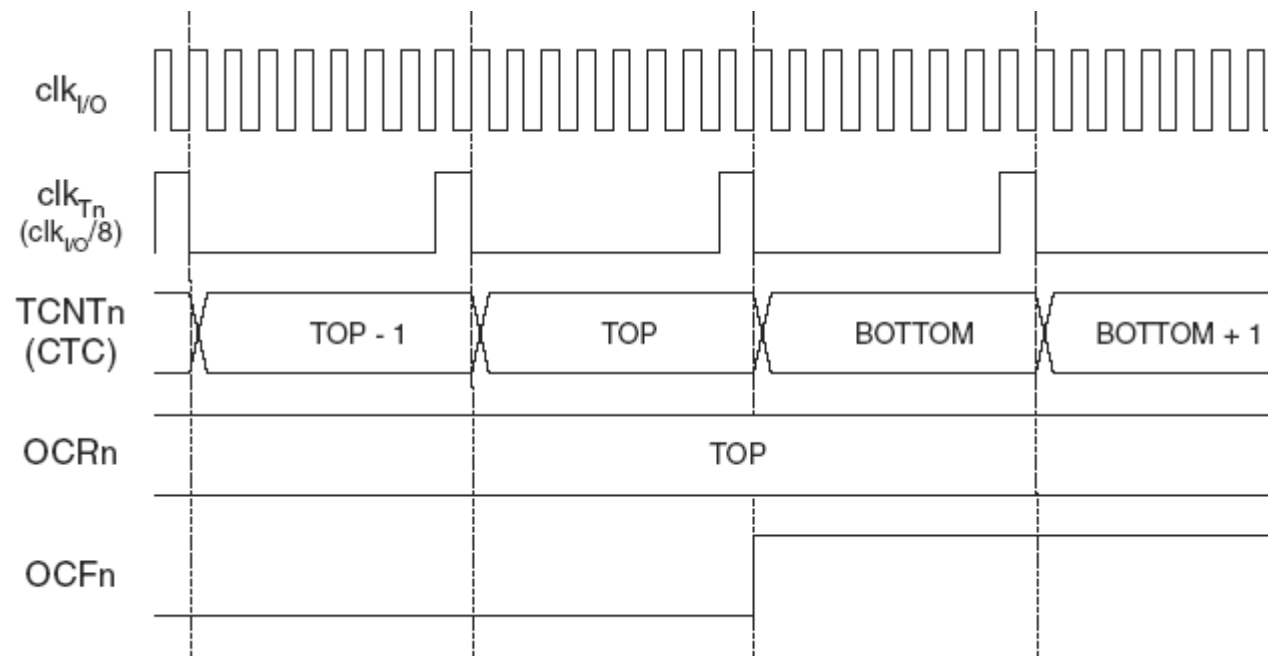
gdy do **OCR0** wpisane jest zero.

Częstotliwość przebiegu wyrażona jest wzorem:

$$f_{OCn} = \frac{f_{clk_I/O}}{2 \cdot N \cdot (1 + OCRn)}$$

N jest podziałem preskalera (przyjmuje wartości 1, 8, 64, 256 lub 1024).

Tryb CTC (preskaler 8)

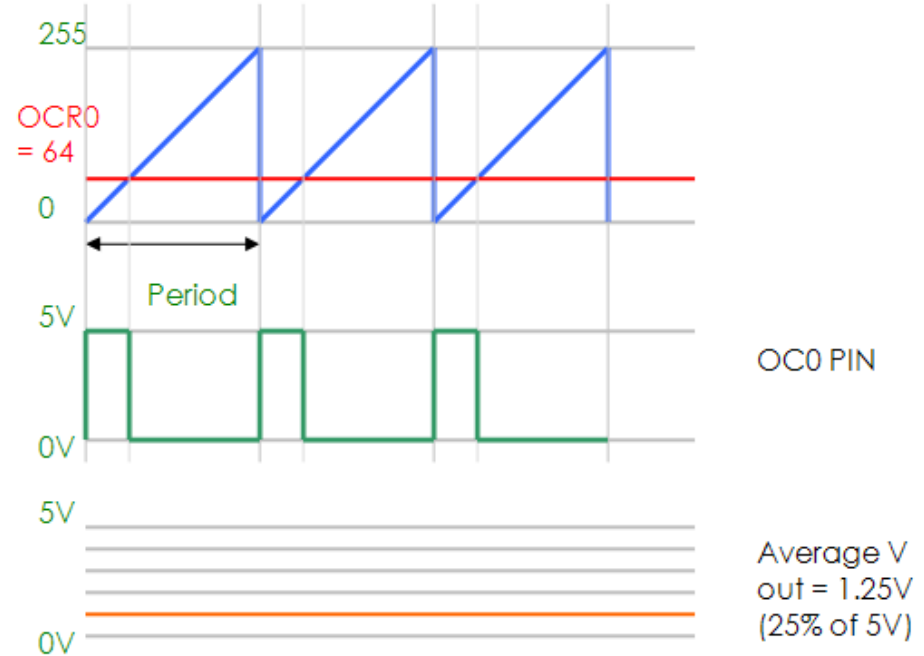


Generacja fali prostokątnej ($f=2\text{kHz}$) w trybie CTC

```
#include <avr/io.h>
#define F_CPU 8000000
int main(void)
{
    DDRB=0xFF;
    PORTB=0x00;
    // Mode: CTC
    TCCR0=(1<<WGM01);
    // CLK/8
    TCCR0|=(1<<CS01);
    // toggle OC0 on compare match
    TCCR0|=(1<<COM00);
    //OCR0=249 f=8000000/(2*8*250)=2KHz
    OCR0=0xF9;
    while (1);
}
```

Szybki tryb Modulacji Szerokości Impulsu (Fast PWM)

- Licznik zlicza od zera do maksymalnej (0xFF) i ponownie zaczyna od zera.
- Wyjście porównania **OC0** jest zerowane, gdy wartość licznika jest większa lub równa wartości w komparatorze i ustawiane gdy licznik jest zerowany.
- Tryb ten jest odpowiedni do regulacji mocy, przetwarzania cyfrowo-analogowego.
- Wysoka częstotliwość PWM umożliwia zmniejszenie rozmiarów zewnętrznych elementów współpracujących – cewek czy kondensatorów i obniża koszty wykonania.
- Bit przepełnienia (**TOV0**) jest ustawiany podczas osiągnięcia przez licznik wartości maksymalnej. Jeżeli przerwania są aktywne, to procedura przerwania może zmienić wartość rejestru porównania.
- Częstotliwość nośna obliczana jest ze wzoru:



Copyright 2007-2009 www.eXtremeElectronics.co.in

$$f_{OCn} = \frac{f_{CLK_110}}{N \cdot 256}$$

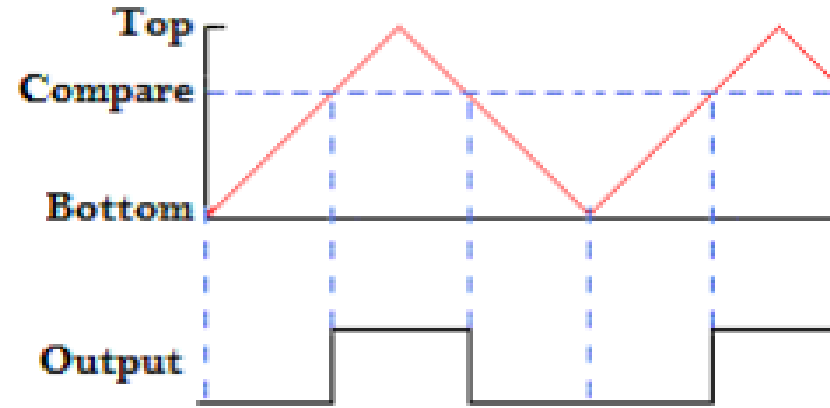
gdzie N jest podziałem preskalera (przyjmuje wartości 1, 8, 64, 256 lub 1024).

Tryb PWM z Korekcją Fazy

- Licznik powtarza zliczanie od wartości zerowej do maksymalnej i potem w dół do zera.
- **OC0** jest zerowany przy porównaniu pomiędzy **TCNT0** a **OCR0** podczas zliczania w dół i ustawiany podczas porównania przy zliczaniu w górę.
- System podwójnego zliczania daje mniejszą częstotliwość niż we wcześniejszym trybie.
- Dzięki uzyskanej w ten sposób symetrii tryb ten jest zalecany do sterowania silnikami.
- Częstotliwość licznika określona jest z zależności :

$$f_{OCnPCPPWM} = \frac{f_{clk_1 / 0}}{N \cdot 510}$$

gdzie N jest podziałem preskalera (przyjmuje wartości 1, 8, 64, 256 lub 1024).

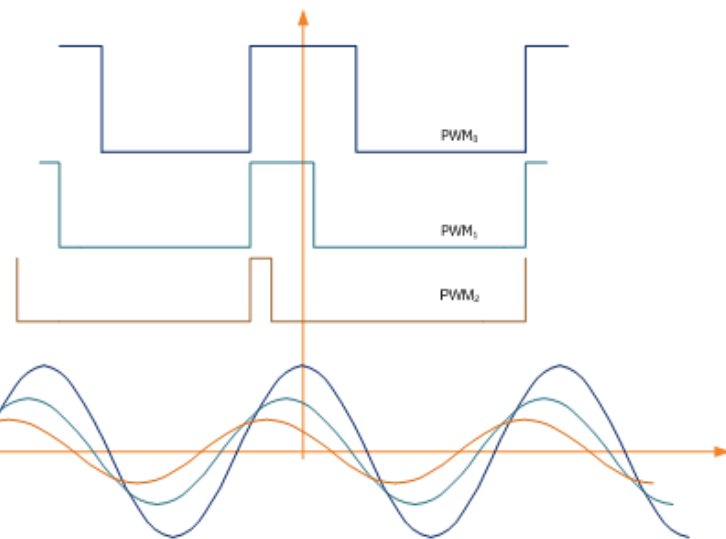


STANDARD PWM

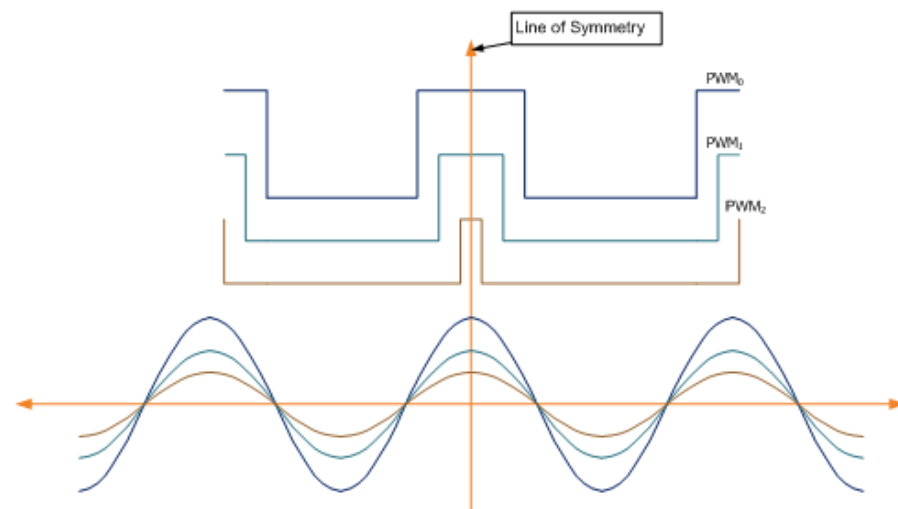
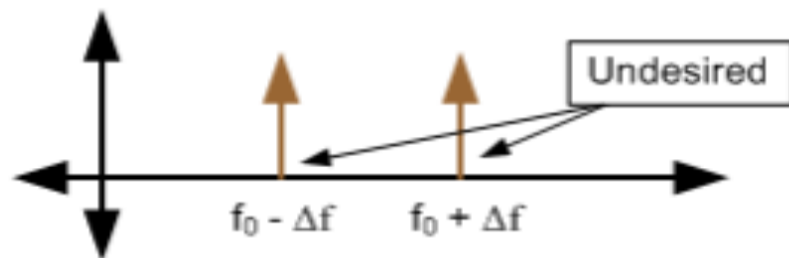


Phase Correct PWM

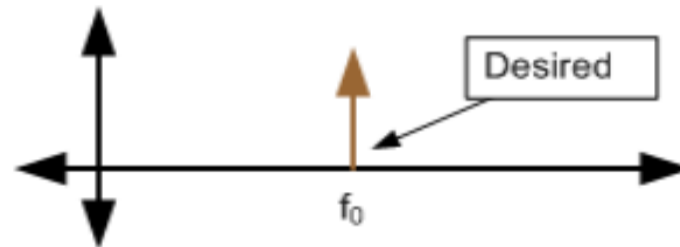




Asymmetric Frequency Spectra



Symmetric Frequency Spectra



Licznik 1

- Licznik 16-bitowy
- Rejestry licznika też 16-bitowe
- TCNT1H, TCNT1L – wartość licznika (0-65535)
- OCR1AH, OCR1AL – wartość porównania A
- OCR1BH:OCR1BL – wartość porównania B
- ICR1H:ICR1L – czas zdarzenia zewnętrznego

Eliminacja drgań styków

- Naciśnięcie przycisku może wywołać wiele zmian stanów logicznych
- Efekt ten trwa zwykle nie dłużej niż 10ms
- W celu eliminacji, należy wykorzystać przerwanie licznika, by odczytywać stan przycisku tylko raz na 10ms (nie używamy funkcji `delay_ms`!).
- Stan przycisku zapisać w zmiennej globalnej dostępnej w programie głównym

