Programowanie mikrokontrolerów

Przerwania i liczniki

Marcin Engel Marcin Peczarski

Instytut Informatyki Uniwersytetu Warszawskiego

6 stycznia 2012

Przerwania — terminologia

- Przerwanie — obsługiwane sprzętowo przez mikrokontroler przerwanie wykonania aktualnego programu i przekazanie sterowania do procedury obsługi przerwania.
- Zdarzenie lub stan wyzwalający przerwanie — zewnętrzne lub wewnętrzne zdarzenie lub stan, którego zajście może spowodować przerwanie.
- Przerwanie jest wyłączone, jeśli zdarzenie lub stan wyzwalający nie powoduje powstania przerwania.
- Przerwanie jest włączone, jeśli zdarzenie lub stan wyzwalający powoduje powstanie przerwania.

Przerwania — terminologia, cd.

- Mikrokontroler rejestruje (ustawiając odpowiednie bity) zdarzenia wyzwalające (pewne) przerwania niezależnie od tego, czy dane przerwanie jest włączone, czy nie.
- Do przerwania dochodzi jedynie wówczas, gdy jest ono włączone i zaszło zdarzenie wyzwalające to przerwanie.
- Przerwanie oczekuje, jeśli jest wyłączone, a mikrokontroler zarejestrował zdarzenie wyzwalające go.

Przerwania

- ATmega16 ma 21 źródeł przerwań.
- Przerwania umożliwiają:
  - asynchroniczną obsługę różnych zdarzeń,
  - efektywne wykorzystanie urządzeń peryferyjnych,
  - obsługę „w tle” w stosunku do programu głównego.

Zdarzenia wyzwalające przerwania

- Zmiana stanu pewnych wejść (przerwania INT0, INT1, INT2, ICP1).
- Określony stan pewnych wejść (przerwania INT0, INT1).
- Przepełnienie licznika (TIMER0 OVF, TIMER1 OVF, TIMER2 OVF).
- Osiągnięcie przez licznik zadanej wartości (TIMER0 COMP, TIMER1 COMPA, TIMER1 COMPB, TIMER2 COMP).
- Zakończenie przetwarzania analogowo-cyfrowego (ADC).
- Zakończenie transmisji przez interfejs szeregowy (SPI STC, USART TXC, TWI).
- Odebranie danych przez interfejs szeregowy (USART RXC).
- Gotowość pamięci EEPROM (EE_RDY).
- Zmiana stanu wyjścia komparatora (ANA_COMP).

Tablica przerwań

- W standardowej konfiguracji mikrokontrolera 42 najniższe adresy w pamięci programu zajmuje tablica przerwań.
- Im mniejszy adres, tym wyższy priorytet przerwania.
- Najwyższy priorytet ma RESET, potem przerwanie zewnętrzne INT0 itd.
- Przerwania są numerowane od 1 do 21.
- Obsługując przerwanie o numerze x, mikrokontroler ładuje do licznika programu wartość $2(x-1)$.
- Powoduje to wykonanie rozkazu spod tego adresu w pamięci programu.
- Najczęściej jest to skok (`rjmp` lub `jmp`) do właściwej procedury obsługi przerwania.
- Uwaga, ATmega16 i ATmega32 obsługują ten sam zbiór przerwań, ale są one inaczej ponumerowane.

Konfigurowanie przerwań

- Po włączeniu zasilania wszystkie przerwania są wyłączone. (z wyjątkiem RESET oczywiście).
- W rejestrze znaczników `SREG` znajduje się znacznik `I`, którego wyzerowanie blokuje (wyłącza) wszystkie przerwania.
- Znacznik `I` ustawia się za pomocą rozkazu `sei`, a zeruje za pomocą — `cli`.
- Z każdym źródłem przerwania jest związany bit uaktywniający (w odpowiednim rejestrze wejścia-wyjścia).
- Przerwanie jest włączone wtedy i tylko wtedy, gdy jest ustawiony jego bit uaktywniający oraz znacznik `I`.

Przerwania wyzwalane zdarzeniami

- Z każdym takim przerwaniem jest związany znacznik wystąpienia ustawiany sprzętowo, gdy zajdzie zdarzenie wyzwalające.
- Znacznik ten jest ustawiany niezależnie od tego, czy przerwanie jest włączone.
- Znacznik wystąpienia jest sprzętowo zerowany w trakcie obsługi przerwania (jeśli przerwanie jest włączone).
- Znacznik wystąpienia można wyzerować także programowo, ustawiając go na . . . 1!
- Jeśli zdarzenie wyzwalające zaszło, gdy przerwanie było wyłączone, to po jego włączeniu zostanie wywołana procedura jego obsługi, o ile znacznik wystąpienia nie został uprzednio wyzerowany programowo.

Przerwania wyzwalane stanem

- Nie mają znacznika wystąpienia.
- Dochodzi do nich tak długo, jak długo trwa stan wyzwalający.
- Jeśli stan wyzwalający pojawił się przy wyłączonym przerwaniu i zaniknął przed jego włączeniem, do przerwania nie dochodzi.

Przykład

- Przerwanie zewnętrzne INT0 jest wyzwalane zdarzeniem lub stanem w zależności od konfiguracji.
- Jeśli INT0 skonfigurujemy, aby było wyzwalane zmianą stanu wyprowadzenia INT0, to jest to przerwanie wyzwalane zdarzeniem — mikrokontroler rejestruje wszystkie zdarzenia wyzwalające.
- Jeśli INT0 skonfigurujemy, aby było wyzwalane niskim stanem na wyprowadzeniu INT0, to jest to przerwane wyzwalane stanem i:
  - nie jest rejestrowane,
  - jeśli jest włączone, pojawia się tak długo, jak długo utrzymuje się stan niski na wyprowadzeniu.

Obsługa przerwania

- Gdy pojawi się zdarzenie wyzwalające przerwanie o numerze x, sprzęt ustawia znacznik wystąpienia tego przerwania (jeśli jest).
- Jeśli przerwanie jest włączone, to:
  - jeśli zdarzenie pojawiło się w trakcie wykonania rozkazu złożonego z kilku cykli, to mikrokontroler najpierw kończy wykonanie tego rozkazu,
  - obsługa wszystkich przerwań jest wyłączana,
  - aktualna wartość licznika rozkazów jest odkładana na stos i ustawiana na wartość $2(x-1)$,
  - znacznik wystąpienia przerwania jest zerowany,
  - wykonuje się kod obsługi przerwania zakończony rozkazem `reti`,
  - rozkaz `reti` powoduje zdjęcie ze stosu adresu powrotu, załadowanie go do licznika rozkazów i włączenie przerwań.

Uwagi

- Jeśli zdarzenie wyzwalające pojawi się równocześnie z wykonaniem rozkazu `cli`, do przerwania nie dojdzie.
- Po rozkazie `sei` przed obsługą oczekujących przerwań wykona się co najmniej jeden rozkaz.
- Oczekujące przerwania są obsługiwane w kolejności priorytetów.
- Po zakończeniu obsługi przerwania (`reti`) mikrokontroler zawsze powraca do programu głównego i wykonuje co najmniej jeden jego rozkaz.
- Wykonanie `sei` w procedurze obsługi przerwań umożliwia przerwania zagnieżdżone.
- Gdy dochodzi do przerwania, mikrokontroler nie odkłada na stos rejestru stanu (ani żadnego innego rejestru oprócz licznika rozkazów).

Rozważmy następujący program

- Gdzieś w RAM definiujemy 1-bajtową zmienną.

```
.dseg
.org 0x60 ; adres poczatku RAM
value: .byte 1
```

- Wektor przerwań o adresie 0 musi zawierać instrukcję skoku do właściwego początku programu.

```
.cseg
.org 0
        rjmp    start
```

Rozważmy następujący program

- Rozpoczynamy, jak zwykle, od skonfigurowania stosu.

```
.org 0x2A ; pierwszy adres za tablicą przerwań
start:
        ldi     r17, high(RAMEND)
        ldi     r16, low(RAMEND)
        out     sph, r17
        out     spl, r16
```

- Następnie inicjujemy i konfigurujemy wyświetlacz LCD.

```
        rcall   LCD_init
        rcall   LCD_config
```

- Ustawiamy wartość początkową zmiennej.

```
        ldi     r16, 0
        sts     value, r16
```

- Instrukcja `sts` zapisuje wartość z rejestru pod wskazanym adresem w RAM.

Rozważmy następujący program

- W pętli głównej wyświetlamy wartość naszej zmiennej.

```
forever:
    lds    r16, value
    rcall  LCD_number; wyświetl wartość r16
    ldi    r16, 0
    rcall  LCD_goto; ustaw kursor pod adresem r16
    rjmp   forever
```

- Instrukcja `lds` ładuje rejestr wartością spod podanego adresu w RAM.
- Ten program nie robi nic mądrego.
- Posłuży nam do obserwacji zmian `value`.

Przerwania zewnętrzne

- Są trzy przerwania zewnętrzne: INT0, INT1, INT2.
- Są one wyzwalane zdarzeniami na wyprowadzeniach INT0, INT1, INT2.
- Wyprowadzenie INT0 to PD2, INT1 to PD3, a INT2 to PB2.
- Przerwanie INT2 jest wyzwalane zmianą stanu na wyprowadzeniu PB2.
- Przerwania INT0 i INT1 mogą być wyzwalane zmianą stanu lub stanem niskim na odpowiednich wyprowadzeniach.
- Wyzwalanie przerwań działa także wtedy, gdy wyprowadzenie jest skonfigurowana jako wyjście.
- Przerwanie INT2 oraz przerwanie wyzwalane stanem niskim na INT0 i INT1 są asynchroniczne (do ich wykrycia nie jest potrzebny sygnał zegara).
- Przerwania asynchroniczne można wykorzystać do wybudzenia mikrokontrolera ze stanu oszczędzania energii — powiemy o tym później.

Jak skonfigurować przerwania INT0 i INT1?

- Odpowiednio ustawić cztery młodsze bity w rejestrze `MCUCR`.
- Ustawić bit uaktywniający przerwanie INT0 i/lub INT1 w rejestrze `GICR`.
- Pod adresem `INT0addr` (i/lub `INT1addr`) w pamięci programu umieścić rozkaz skoku do procedury obsługi przerwania.
- Włączyć przerwania rozkazem `sei`.

- Jeśli zdarzeniem wyzwalającym przerwanie INT0/INT1 jest zmiana stanu wejścia, to mikrokontroler ustawia bity INTF0/INTF1 w rejestrze `GIFR`.
- Bity są zerowane programowo przez ustawienie ich na 1 lub sprzętowo w trakcie obsługi przerwania.

Jak skonfigurować przerwanie INT2

- Odpowiednio ustawić bit 6 w rejestrze `MCUCSR`.
- Ustawić bit aktywujący przerwanie INT2 w rejestrze `GICR`.
- Pod adresem `INT2addr` w pamięci programu umieścić rozkaz skoku do procedury obsługi przerwania.
- Włączyć przerwania rozkazem `sei`.

- Zdarzenie wyzwalające przerwanie INT2 powoduje ustawienie bitu INTF2 w rejestrze `GIFR`.
- Bit jest zerowany programowo przez ustawienie go na 1 lub sprzętowo w trakcie obsługi przerwania.

Rejestr MCUCR

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

- Bity 0–1 – zdarzenie wyzwalające przerwanie INT0:

ICS01	ISC00	zdarzenie wyzwalające
0	0	niski stan wyprowadzenia INT0
0	1	każda zmiana poziomu INT0
1	0	zbocze opadające na INT0 (zmiana z 1 na 0)
1	1	zbocze narastające na INT0 (zmiana z 0 na 1)

- Bity 2–3 – zdarzenie wyzwalające przerwanie INT1 (analogicznie do INT0).
- Pozostałe bity nie dotyczą przerwań zewnętrznych i na razie ich nie omawiamy, ale musimy zadbać, aby nie zmieniać ich wartości.

Rejestr MCUCR — uwagi

- Przy wyzwalaniu zmianą poziomu impulsy krótsze niż jeden cykl zegara nie gwarantują przerwania.
- Aby doszło do przerwania wyzwalanego niskim stanem, musi on utrzymać się do zakończenia aktualnie wykonywanego rozkazu.

Rejestr MCUCSR

7	6	5	4	3	2	1	0
	ISC2						

- Bit 6 – sposób zgłaszania przerwania INT2:
  - 0 – przy opadającym zboczu (zmiana z 1 na 0),
  - 1 – przy narastającym zboczu (zmiana z 0 na 1).
- Pozostałe bity nie dotyczą INT2 i na razie ich nie omawiamy, ale musimy zadbać, aby nie zmieniać ich wartości.
- Impulsy krótsze niż 50 ns mogą nie wywołać przerwania.

Rejestr GICR

7	6	5	4	3	2	1	0
INT1	INT0	INT2					

- Bit 5 – bit aktywujący przerwanie INT2,
- Bit 6 – bit aktywujący przerwanie INT0,
- Bit 7 – bit aktywujący przerwanie INT1:
  - 0 – przerwanie wyłączone,
  - 1 – przerwanie włączone (jeśli także znacznik I ustawiony).
- Pozostałe bity nie dotyczą przerwań zewnętrznych i na razie nie będziemy ich omawiać, ale musimy zadbać, aby nie zmieniać ich wartości.

Rejestr GIFR

7	6	5	4	3	2	1	0
INTF1	INTF0	INTF2					

- Bit 5 – znacznik wystąpienia zdarzenia wyzwalającego INT2,
- Bit 6 – znacznik wystąpienia zdarzenia wyzwalającego INT0,
- Bit 7 – znacznik wystąpienia zdarzenia wyzwalającego INT1:
  - 0 – zdarzenia nie było,
  - 1 – zdarzenie wystąpiło.
- Pozostałe bity nie dotyczą przerwań zewnętrznych i nie będziemy na razie ich omawiać, ale musimy zadbać, aby nie zmieniać ich wartości.
- Jeśli INT0/INT1 są wyzwalane niskim stanem wejścia, to znaczniki INTF0/INTF1 nie są ustawiane.

Przykładowa konfiguracja przerwania INT2

- Wciśnięcie klawisza wywołuje zbocze opadające.

```
in      r16, MCUCSR
cbr     r16, 1 << ISC2
out     MCUCSR, r16
```

- Uaktywniamy przerwanie.

```
in      r16, GICR
sbr     r16, 1 << INT2
out     GICR, r16
```

- Uwaga, drugim argumentem instrukcji `cbr` i `sbr` jest maska bitowa, a nie numer bitu, jak w instrukcjach `cbi` i `sbi`.

Przykładowa konfiguracja przerwania INT2

- Zerujemy znacznik przerwania, wysyłając 1. Jest to zalecane przy zewnętrznych źródłach przerwań.

```
ldi     r16, 1 << INTF2
out     GIFR, r16
```

- Uwaga, wpisanie do rejestru `GIFR` wartości `1 << INTF2` nie zmienia pozostałych bitów tego rejestru.
- Uaktywniamy system przerwań.

```
sei
```

Obsługa przerwania INT2

- W wektorze przerwań umieszczamy skok do procedury obsługi.

```
.org INT2addr
        rjmp    interrupt2
```

- Procedura obsługi musi zachować na stosie rejestr znaczników i wszystkie używane rejestry robocze.

```
interrupt2:
        push    r16
        in      r16, SREG
        push    r16
```

Obsługa przerwania INT2

- Właściwa obsługa polega na zwiększeniu `value` o 1.

```
lds     r16, value
inc     r16
sts     value, r16
```

- Odtwarzamy rejestr znaczników i rejestry robocze.

```
pop     r16
out     SREG, r16
pop     r16
```

- Kończymy obsługę przerwania, odblokowujemy obsługę przerwań.

```
reti
```

- Faktyczna realizacja musi zadbać o eliminację drgania styków!

Propozycje ćwiczeń

- Przećwiczyć obsługę przerwania:
  - wersja bez oczekiwania na ustabilizowanie styków,
  - wersja z oczekiwaniem na ustabilizowanie styków,
  - różne inne wersje ze schematów z poprzedniego tygodnia.

Liczniki

- Układy sprzętowe wyposażone w wewnętrzny rejestr zwiększany o 1 przy odpowiedniej zmianie stanu wejścia sterującego.
- Na wejście sterujące licznika można podać:
  - sygnał zegarowy,
  - przeskalowany sygnał zegarowy (z preskalera),
  - sygnał z pewnego wyprowadzenia mikrokontrolera.
- Licznik może służyć do:
  - cyklicznego zgłaszania przerwania,
  - generowania różnego rodzaju przebiegów na pewnym wyprowadzeniu mikrokontrolera,
  - zliczania zdarzeń zewnętrznych,
  - odmierzania czasu między pewnymi zdarzeniami zewnętrznymi.

Liczniki w ATmega16 i ATmega32

- ATmega16 i ATmega32 mają trzy liczniki:
  - dwa 8-bitowe Timer0 i Timer2,
  - jeden 16-bitowy Timer1.
- Każdy z liczników ma od 4 (Timer0) do 16 (Timer1) różnych trybów pracy.
- Poszczególne liczniki mają różne zestawy trybów pracy.
- Na razie zajmiemy się dwoma najprostszymi trybami pracy licznika 0.

Kiedy dochodzi do zwiększenia licznika 0?

W zależności od konfiguracji:

- przy zboczu rosnącym każdego cyklu zegara,
- przy zboczu rosnącym co 8-ego, 64-ego, 256-ego lub 1024-tego cyklu zegara,
- przy zboczu rosnącym na wyprowadzeniu T0 (PB0),
- przy zboczu opadającym na wyprowadzeniu T0 (PB0).

Rejestry licznika 0

- Rejestr `TCNT0` — Timer/Counter Register 0:
  - zawiera aktualną wartość licznika,
  - zapis zmienia wartość licznika.
- `OCR0` — Output Compare Register 0:
  - zawiera wartość, która jest ciągle porównywana z wartością licznika,
  - wykrycie zgodności może być użyte do zmiany poziomu na wyjściu licznika OC0 (PB3) ...
  - ...lub jako zdarzenie wyzwalające przerwanie o symbolicznym adresie `OC0addr`.
- Zapis do `TCNT0` blokuje potencjalne wykrycie zgodności w następnym cyklu licznika.
- Zapis do `OCR0` jest buforowany — w pewnych trybach pracy licznika jego zmiana nie ma natychmiastowego wpływu na licznik.

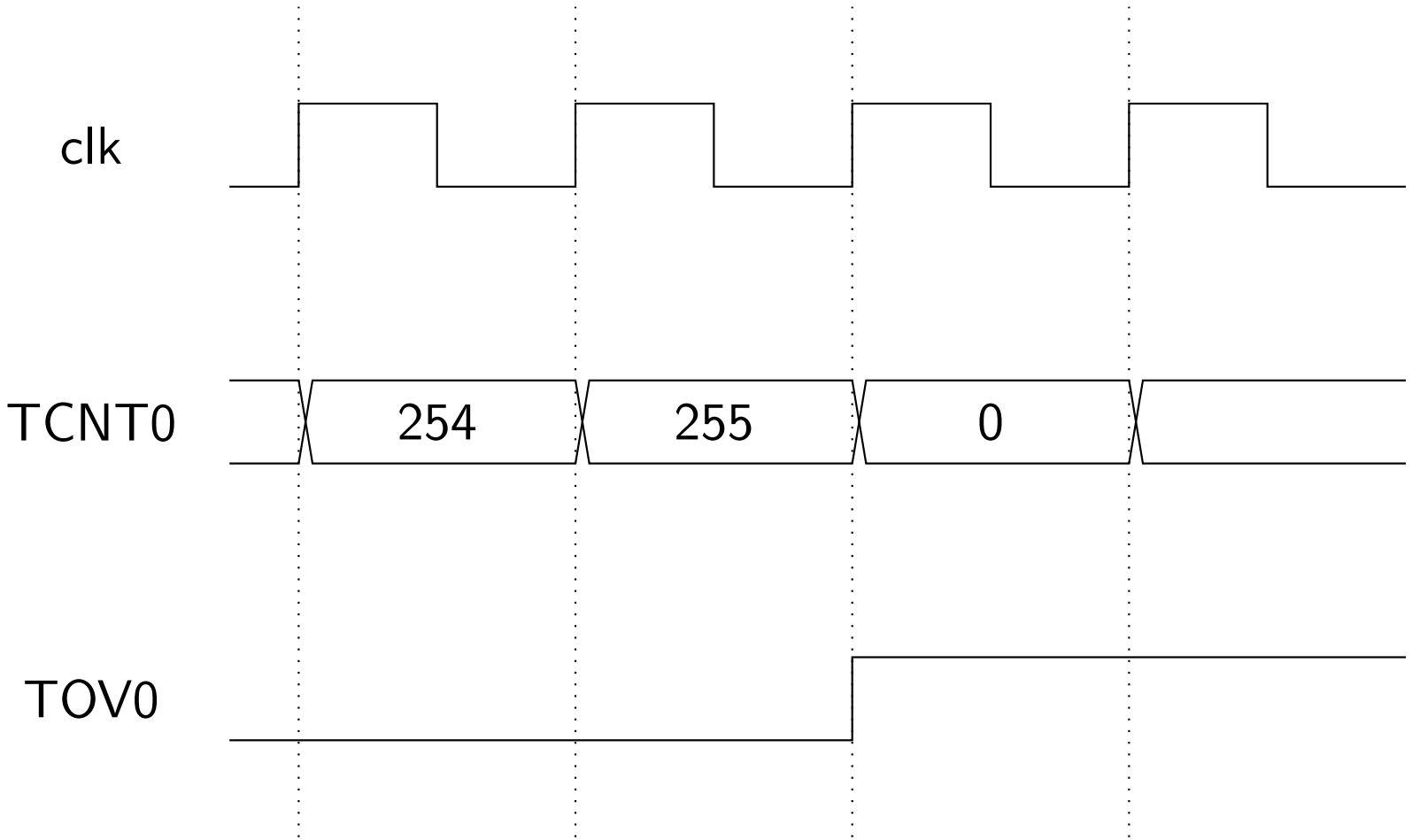
Jakie są tryby pracy licznika 0?

- Tryb zwykły: licznik zlicza od 0 do 255 i potem znów od zera.
- Tryb CTC (Clear Timer on Compare): licznik zlicza od 0 do `OCR0` i potem znów od zera.
- Dwa tryby PWM — o nich za dwa tygodnie.

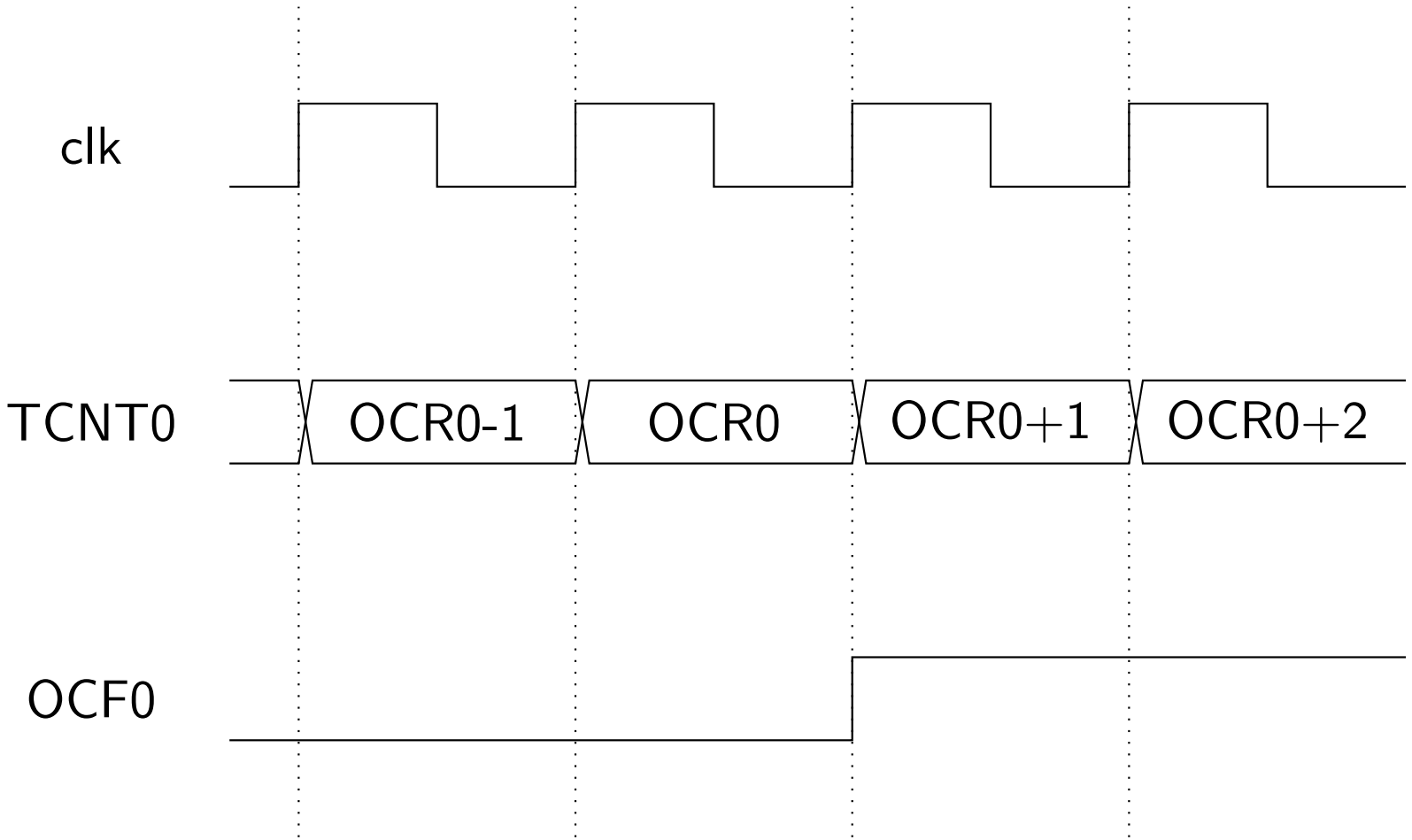
Zarówno w trybie zwykłym jak i w CTC:

- rejestr `OCR0` jest modyfikowany natychmiast,
- jeśli licznik osiągnie wartość równą wartości z rejestru `OCR0`, jest ustawiany znacznik `OCF0`,
- w przypadku przepełnienia jest ustawiany znacznik `TOV0` (w trybie CTC może do tego nigdy nie dochodzić),
- ustawienie znacznika `OCF0` lub `TOV0` jest zdarzeniem wyzwalającym przerwanie.

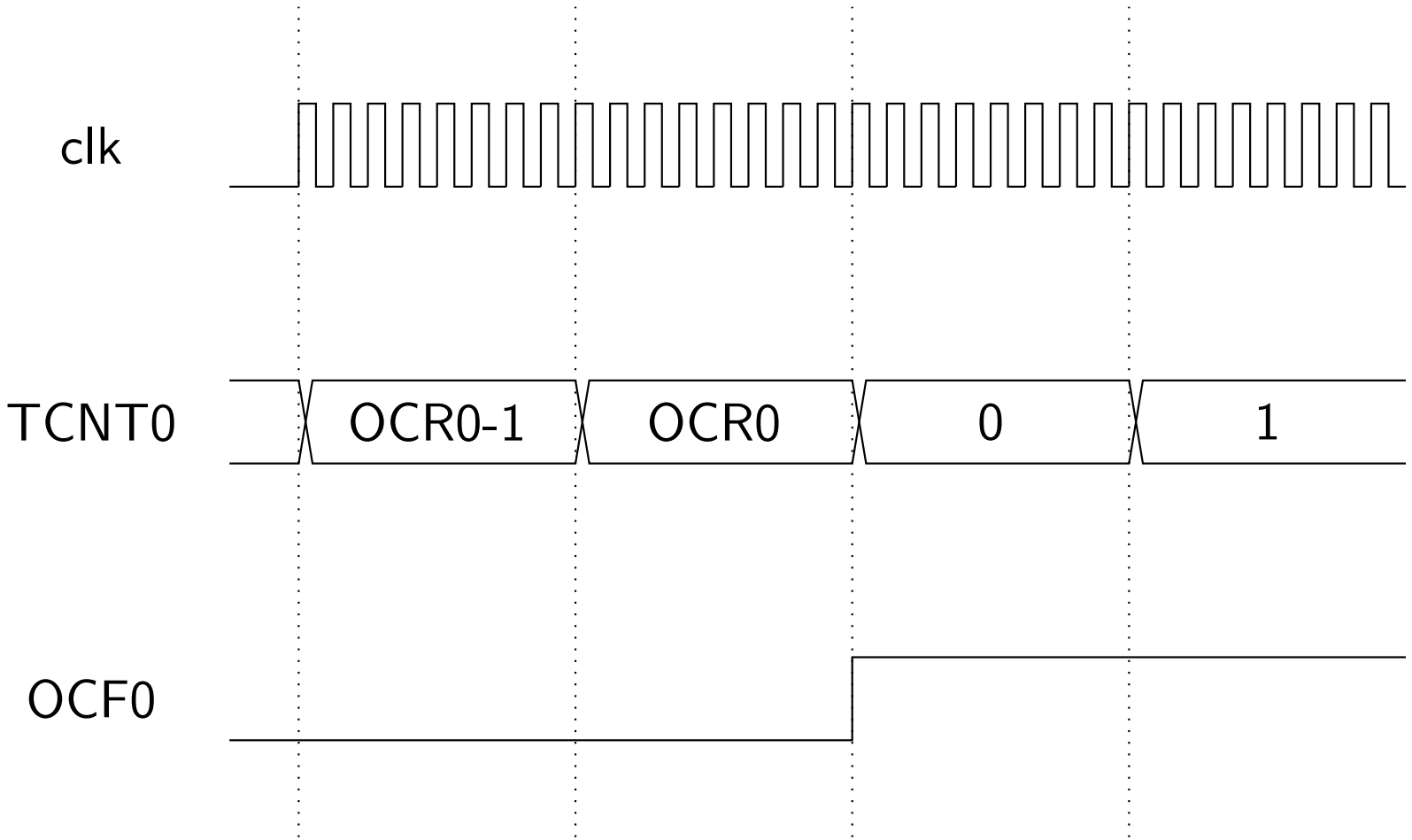
Przebieg czasowy, tryb zwykły bez preskalera



Przebieg czasowy, tryb zwykły bez preskalera



Przebieg czasowy, tryb CTC, preskaler /8



Jak podać wyjście licznika na nogę mikrokontrolera?

- Po odpowiednim skonfigurowaniu wyjście OC0 licznika może być podawane na wyprowadzenie PB3.
- Wyprowadzenie PB3 musi być skonfigurowane jako wyjście (ustawiony trzeci bit w `DDRB`).
- Po włączeniu zasilania stan OC0 jest niski.
- Stan OC0 jest zupełnie niezależny od stanu trzeciego bitu portu B mikrokontrolera. W szczególności:
  - jeśli OC0 przyłączymy na PB3, to trzeci bit rejestru `PORTB` nie ma żadnego wpływu na jej stan,
  - trzeci bit rejestru `PORTB` może być wtedy różny niż trzeci bit rejestru `PINB`.
- VMLAB źle symuluje tę niezależność!

Jak sprzętowo generować przebiegi za pomocą licznika 0?

- Wyjście OC0 może zmieniać się na cztery sposoby:
  - każde wykrycie zgodności zmienia stan wyprowadzenia OC0 na przeciwny niż dotychczasowy,
  - wykrycie zgodności ustawia stan niski na wyprowadzeniu OC0,
  - wykrycie zgodności ustawia stan wysoki na wyprowadzeniu OC0,
  - OC0 nie zmienia się.

Konfigurowanie licznika 0

- Ustaw tryb pracy — bity `WGM00` i `WGM01` w rejestrze `TCCR0`.
- Ustaw preskaler lub zewnętrzne źródło zegara — bity `CS02`, `CS01` i `CS00` w rejestrze `TCCR0`.
- Zdecyduj, czy i jak licznik ma sterować wyprowadzeniem OC0 — bity `COM01`, `COM00` rejestru `TCCR0`.
- Zdecyduj, czy przepełnienie licznika lub wykrycie zgodności mają powodować przerwania — bity `OCIE0` i `TOIE0` rejestru `TIMSK`.

Rejestr TCCR0

7	6	5	4	3	2	1	0
FOC0	WGM00 (PWM0)	COM01	COM00	WGM01 (CTC0)	CS02	CS01	CS00

- Bity 6, 3 – ustalają tryb pracy,
- Bity 5, 4 – sposób sterowania wyjściem OC0,
- Bity 2, 1, 0 – konfiguracja preskalera,
- Bit 7 – omówimy później.

Tryby pracy licznika 0

7	6	5	4	3	2	1	0
FOC0	WGM00 (PWM0)	COM01	COM00	WGM01 (CTC0)	CS02	CS01	CS00

- WGM01 = 0, WGM00 = 0 – tryb normalny,
- WGM01 = 1, WGM00 = 0 – tryb CTC.

Preskaler licznika 0

CS02	CS01	CS00	źródło zegara
0	0	0	licznik zatrzymany
0	0	1	clk
0	1	0	clk/8
0	1	1	clk/64
1	0	0	clk/256
1	0	1	clk/1024
1	1	0	zewnętrzny z nogi T0, zbocze opadające
1	1	1	zewnętrzny z nogi T0, zbocze narastające

Przebieg na nodze OC0

COM01	COM00	stan wyjścia OC0
0	0	OC0 nie zmienia się
0	1	OC0 zmienia swój stan po wykryciu zgodności
1	0	OC0 jest zerowany po wykryciu zgodności
1	1	OC0 jest ustawiany po wykryciu zgodności

- We wszystkich trybach z wyjątkiem pierwszego wyprowadzenie OC0 jest przyłączone do PB3.

Rejestr TIMSK

7	6	5	4	3	2	1	0
						OCIE0	TOIE0

- Bit 1 – aktywacja przerwania, gdy porównanie rejestrów `TCNT0` i `OCR0` wypadło pozytywnie, tzn. gdy ustawiony znacznik `OCF0`:
  - 1 – przerwanie włączone (jeśli ustawiony znacznik `I` w `SREG`),
  - 0 – przerwanie wyłączone.
- Bit 0 – aktywacja przerwania, gdy wystąpiło przepełnienie licznika, tzn. gdy ustawiony znacznik `TOV0`:
  - 1 – przerwanie włączone (jeśli ustawiony znacznik `I` w `SREG`),
  - 0 – przerwanie wyłączone.
- Pozostałe bity dotyczą liczników 1 i 2. Nie będziemy ich na razie omawiać, ale powinniśmy dbać, aby nie zmieniać ich wartości.

Rejestr TIFR

7	6	5	4	3	2	1	0
						OCF0	TOV0

- Bit 1 – znacznik ustawiany, gdy porównanie rejestrów `TCNT0` i `OCR0` wypadło pozytywnie.
- Bit 0 – znacznik ustawiany, gdy wystąpiło przepełnienie licznika.
- Jeśli odpowiednie przerwanie jest włączone, to znacznik jest automatycznie zerowany w trakcie obsługi przerwania.
- Znaczniki mogą być zerowane programowo, przez zapisanie do nich wartości 1.
- Pozostałe bity dotyczą liczników 1 i 2. Nie będziemy ich na razie omawiać, ale powinniśmy dbać, aby nie zmieniać ich wartości.

Propozycje ćwiczeń

- Zaprogramować zwiększanie `value` z częstotliwością ok.: 25 Hz, 10 Hz, 4 Hz, 1 Hz.
- W trybie normalnym przepełnienie występuje z częstotliwością

$$\frac{\text{clk}}{256 \cdot \text{N}},$$

- W trybie CTC zgodne porównanie zachodzi z częstotliwością

$$\frac{\text{clk}}{(\text{OCR0} + 1) \cdot \text{N}},$$

- gdzie
  - clk – częstotliwość zegara,
  - N – współczynnik podziału preskalera.

Propozycje ćwiczeń, cd.

- Wykonać prosty jednogłosowy instrument muzyczny:
  - Przyciski od lewej do prawej to dźwięki gamy: C, D, E, F, G, A, H, C
  - Przybliżone częstotliwości poszczególnych dźwięków:

C	262 Hz
D	294 Hz
E	330 Hz
F	350 Hz
G	392 Hz
A	440 Hz
H	494 Hz
C	523 Hz

  - Głośnik do pobrania u prowadzących laboratorium.

Przykładowa konfiguracja licznika 0, tryb normalny

- Definiujemy tryb pracy licznika.

```
.equ TIMER0_RUNNING_NORMAL = 1 << CS02
```

- Konfigurujemy tryb pracy.

```
ldi     r17, TIMER0_RUNNING_NORMAL
out     TCCR0, r17
```

- Włączamy obsługę przerwania przy przepełnieniu.

```
in      r16, TIMSK
sbr     r16, 1 << TOIE0
out     TIMSK, r16
sei
```

Przykładowa konfiguracja licznika 0, tryb CTC

- Definiujemy tryb pracy licznika.

```
.equ TIMER0_TOP_VALUE = 155
.equ TIMER0_RUNNING_CTC = 1 << CTC0 | 1 << CS02
```

- Konfigurujemy tryb pracy.

```
ldi     r16, TIMER0_TOP_VALUE
ldi     r17, TIMER0_RUNNING_CTC
out     OCR0, r16
out     TCCR0, r17
```

- Włączamy obsługę przerwania przy osiągnięciu wartości z rejestru `OCR0`.

```
in      r16, TIMSK
sbr     r16, 1 << OCIE0
out     TIMSK, r16
sei
```

Obsługa przerwań licznika 0

- W tablicy przerwań umieszczamy skoki do odpowiedniech procedur obsługi.

```
.org OVF0addr
        rjmp    timer0_overflow
.org OC0addr
        rjmp    timer0_compare_match
```

Obsługa przerwań licznika 0

- Procedura obsługi podobna do obsługi przerwania INT2, ale nie ma problemu drgania styków.

```
timer0_compare_match:
timer0_overflow:
        push    r16
        in      r16, SREG
        push    r16

        lds     r16, value
        inc     r16
        sts     value, r16

        pop     r16
        out     SREG, r16
        pop     r16
        reti
```

Propozycje ćwiczeń, cd.

- Napisz program elektronicznej ruletki.
- Nieparzyste wciśnięcie klawisza powoduje wystartowanie licznika – na wyświetlaczu LCD bardzo szybko pojawiają się kolejne wartości od 0 do 37.
- Parzyste wciśnięcie klawisza powoduje zatrzymanie licznika – ostatnia wartość pozostaje na wyświetlaczu.
- Napisz procedury obsługi multipleksowanego 4-cyfrowego wyświetlacza 7-segmentowego.
- Zaimplementuj ruletkę, używając jako wyjścia wyświetlacza 7-segmentowego.

Jak zmienić stan wyjścia OC0 licznika?

7	6	5	4	3	2	1	0
FOC0	WGM00 (PWM0)	COM01	COM00	WGM01 (CTC0)	CS02	CS01	CS00

- Bit 7 — jego ustawienie symuluje wystąpienie zgodności:
  - zmienia stan na wyjściu OC0 zgodnie z ustawionym trybem pracy,
  - ale nie zmienia znaczników wystąpienia przerwania,
  - ani nie zeruje licznika w trybie CTC,
  - jest zerowany sprzętowo,
  - próba jego odczytu daje 0.