Celem tego artykułu jest opis funkcjonowania oraz sposobów konfiguracji i wykorzystania timerów w mikrokontrolerach z rodziny AVR. Rozpoczynając od ogólnego opisu, poprzez przykładowe programy, postaram się wytłumaczyć jak wykorzystać timer wbudowany w strukturę mikrokontrolera AVR dla własnych potrzeb. W przykładach programów posługiwałem się mikrokontrolerem AT90S8535.

Obsługa timerów w mikrokontrolerach AVR

część 1

Trudno jest znaleźć polski odpowiednik słowa „timer". Większość konstruktorów, którzy mieli już do czynienia z mikrokontrolerami, doskonale wiedzą jakie funkcje może on spełniać. Pozwólcie więc, że będę się tym słowem posługiwał bez poszukiwania jego odpowiednika w naszym ojczystym języku.

Timer to prosty układ liczący, najczęściej o rozdzielczości 8 lub 16 bitów. Niech nie zwiedzie nas jednak prostota jego budowy - z każdym timerem związany jest bowiem szereg różnych zmiennych (najczęściej są to bity rejestru kontrolnego) wpływających na to w jaki sposób będzie on pracował. Często istnieją więc możliwości nastaw kierunku zliczania (w górę lub w dół) oraz wyboru źródła impulsów zegarowych - czy to z otoczenia mikrokontrolera, czy też z wewnętrznego generatora zegarowego lub dołączonego rezonatora kwarcowego (AVR). Programista - elektronik najczęściej używa timera bądź to do zliczania impulsów, bądź to do pomiaru czasu ich trwania albo też do budowy tak zwanego generatora PWM. Będzie o tym mowa w dalszej części artykułu.

Najczęściej, jeśli timer wykorzystywany jest do pomiaru czasu trwania impulsu, to jako wzorzec wykorzystuje się wewnętrzny generator zegarowy lub wzorcowy, zewnętrzny sygnał odniesienia. Prowadzi to nas do wniosku, że czas trwania impulsu może być mierzony z dokładnością do czasu trwania impulsów wzorcowych. Stanowią one swego rodzaju jednostkę pomiarową. Najważniejszą jednak cechą timera jest ta, że może on funkcjonować niezależnie od reszty procesów obsługiwanych przez jednostkę centralną mikrokontrolera (abstrahując od konfiguracji bitów kontrolujących pracę timera, która musi być wykonana przez CPU).

Struktury współczesnych mikrokontrolerów wyposażane są w 2 lub 3 układy timerów. Mikrokontrolery rodziny AVR (AT90 i ATmega) są wyposażone w dwa timery 8-bitowe i jeden 16-bitowy. W większości zastosowań większe możliwości oferuje timer 16-bitowy, jednak dla wielu aplikacji rozdzielczość 8-bitowa jest wystarczająca. Jest ona też bardziej dopasowana do architektury rdzenia (który jest 8-bitowy) i przez to umożliwia znacznie szybsze wykonywanie operacji arytmetycznych czy porównań ze stałymi lub zmiennymi używanymi przez daną aplikację.

Ze względu na swoją elastyczność, timery mikrokontrolerów AVR mogą być wykorzystywane dla różnych celów. Dalsza część tekstu ma na celu przybliżenie tych zastosowań oraz wytłumaczenie w jaki sposób niezależne układy funkcjonalne komunikują się z CPU mikrokontrolera oraz jak mogą być przezeń wykorzystane.

Tabela 1. Nastawy bitów preskalera

TCCRx			Synchroniczny Timer 0 i Timer 1	Synchroniczny/Asynchroniczny Timer 2
Bit 2	Bit 1	Bit 0	$P_{CK0,1}$ = zegar syst.	P_{CK2} = zegar syst./zegar zewn.
0	0	0	0 (Timer 0/1 zatrzymany)	0 (Timer 2 zatrzymany)
0	0	1	P_{CK} (zegar systemowy)	P_{CK2} (zegar systemowy lub asynchroniczny)
0	1	0	$P_{CK}/8$	$P_{CK2}/8$
0	1	1	$P_{CK}/64$	$P_{CK2}/32$
1	0	0	$P_{CK}/256$	$P_{CK2}/64$
1	0	1	$P_{CK}/1024$	$P_{CK2}/128$
1	1	0	P_{CK} opadające zbocze na wypr.Tx	$P_{CK2}/256$
1	1	1	P_{CK} narastające zbocze na wypr.Tx	$P_{CK2}/1024$

Sygnalizacja zdarzeń

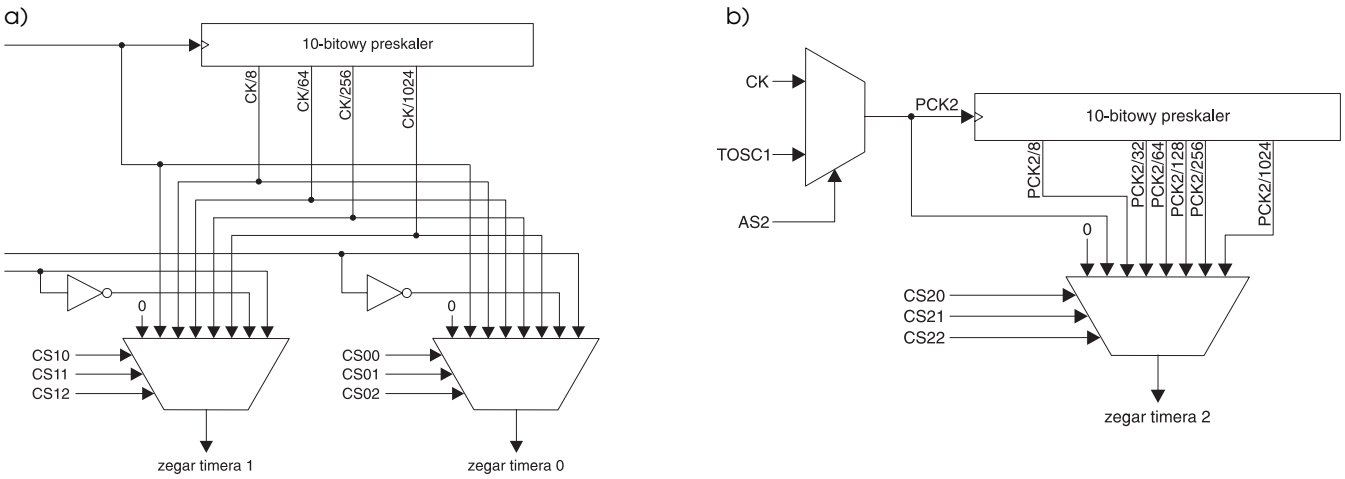
CPU mikrokontrolera AVR może monitorować do 3 zdarzeń powodowanych przez każdy z timerów. Zdarzenia te są sygnalizowane przez ustawienie odpowiednich bitów statusu (tak zwanych flag) w rejestrze TIMSK (*Timer Interrupt Mask*). Tak więc kontrola stanu timera sprowadza się do testowania przez CPU mikrokontrolera maksymalnie 3 bitów sygnalizujących stan timera. Bitami tymi są:

Timer Overflow
(przepełnienie timera)

Ustawienie tego bitu informuje, że timer osiągnął wartość maksymalną i zostanie wyzerowany w następnym cyklu zegarowym. Jak wcześniej wspomniałem, AVR wyposażony jest w dwa timery 8-bitowe oraz jeden 16-bitowy. W praktyce oznacza to dwa timery mogące liczyć do wartości 0xFF oraz jeden liczący do 0xFFFF. Przepełnienie sygnalizowane jest przy pomocy bitu noszącego nazwę Timer Overflow Flag (TOVx) w rejestrze TIFR (*Timer Interrupt Flag Register*).

Compare Match
(spełniony warunek porównania)

W przypadku, gdy nie jest konieczne monitorowanie stanu flagi przepełnienia, może być używane przerwanie typu COMPARE MATCH wywoływane, gdy wartość zapamiętana w rejestrze OCRx (*Output Compare Register*) zgadza się ze zliczoną przez timer. Wskazanie przez timer wartości identycznej z zapisaną w rejestrze OCRx powoduje ustawienie właściwego bitu OCFx (*Output Compare Flag*) w rejestrze TIFR. Timer może być również skonfigurowany w taki sposób, aby jednocześnie z ustawieniem flagi OCFx wartość rejestru liczącego timera była zerowana. Istnieje również możliwość wyboru takiego trybu pracy, dzięki któremu automatycznie, w momencie



Rys. 1. Poglądowy schemat połączeń pomiędzy preskalerem i multiplekserem dla Timerów 0 i 1 (a) oraz 2 (b)

Uwagi:

1. Preskaler pracuje nieprzerwanie - również podczas wprowadzania nastaw timerów. W przypadkach gdy wymagane jest bardzo dokładne odmierzanie czasu, należy samemu zadbać o to, aby timer został zatrzymany i preskaler zaczął podział od wartości 0. W mikrokontrolerach nie przeprowadzających zerowania preskalera może ono zostać przeprowadzone przez detekcję przepełnienia preskalera przez aplikację oraz inicjalizację rejestru TCNTx po tym zdarzeniu.

2. W nowszych mikrokontrolerach posiadających preskaler dzielony pomiędzy kilka timerów, przeprowadzenie sekwencji reset w taki sam sposób wpływa na wszystkie podłączone urządzenia, inicjując je i przeprowadzając odliczanie od wartości 0.

spełnienia warunku porównania, odpowiednim wyprowadzeniom mikrokontrolera może zostać przypisany stan niski, wysoki lub zanegowany. Funkcja ta jest bardzo użyteczna podczas budowy generatorów sygnału prostokątnego o różnej częstotliwości. Oferując szeroki zakres generowanych częstotliwości umożliwia na przykład budowę prostych przetworników cyfrowo - analogowych, jakkolwiek do tego zastosowania bardziej właściwym wydaje się wykorzystanie trybu generatora o modulowanej szerokości impulsu (PWM).

Input Capture

(przechwyt wartości)

Mikrokontrolery AVR posiadają wejście nazywane *Input Capture* (IC). Zmiana stanu na tym wejściu powoduje, że aktualna wartość timera jest odczytywana i zapamiętywana w rejestrze ICRx (*Input Capture Register*). Jednocześnie ustawiana jest flaga ICFx (*Input Capture Flag*) w rejestrze TIFR. Funkcja ta najczęściej wykorzystywana jest do pomiaru czasu trwania impulsu.

Każdy z wyżej wymienionych bitów może wywoływać odpowiedni wektor przerwania. Przerwaniami oraz ich obsługą zajmiemy się w dalszej części artykułu.

Kontrola stanu timera

Są trzy podstawowe metody kontrolowania zdarzeń generowanych przez timer a tym samym powodowania reakcji mikrokontrolera w zależności od stanu timera:

1. Kontrolowanie stanów bitów statusu (flag) w czasie pracy programu poprzez ich testowanie metodą odpytywania (*pooling* - odpytywanie) i podejmowanie akcji odpowiedniej dla danej ich kombinacji.
2. Odpowiednie ustawienie rejestru kontrolującego przerwania a następnie automatyczne przerywanie pracy programu głównego i wykonywanie programów obsług przerwań.
3. Sprzętowa i całkowicie automatyczna zmiana stanu odpowiedniego wyprowadzenia mikrokontrolera.

Kontrola statusu flag korzysta z faktu, że wewnętrzne układy mikrokontrolera ustawiają określone bity powodujące przejście do procedury obsługi przerwania o ile ta nie została zabroniona. Oczywiście warunkiem korzystania z tej metody jest wyłączenie obsługi przerwania, bo inaczej odpytywanie nie miałoby sensu. Kontrola stanu bitów flag, jakkolwiek chyba najłatwiejsza do wykonania, jest jednocześnie mało efektywną bo zajmuje czas mikrokontrolera. Należy również liczyć się z pewnym opóźnieniem przy podejmowaniu akcji, ponieważ CPU zanim zacznie kontrolować stan flag, może być zaangażowane w realizację zupełnie innej części kodu związanej z obsługą całkowicie innych funkcji mikrokontrolera.

Poniższy fragment programu w asemblerze ilustruje użycie tej metody wykorzystanej do kontroli Timera 0:

```
loop:  ;główna pętla programu
   .........
   in r16,TIFR
   ;załadowanie rejestru TIFR
   ;do r16

   sbrs  r16,TOV0
   ;omiń następną instrukcję,
   ;jeśli bit 0 w r16 jest
   ;ustawiony

   rjmp  loop
   ;wykonaj skok do początku
   ;pętli głównej programu
   ;jeśli bit przepełnienia
   ;Timera 0 nie był ustawiony

event:
   .........
   ;tu rozpoczyna się obsługa
   ;zdarzenia "przepełnienie
   ;Timera 0"
```

Linie te powinny być umieszczone w pętli głównej wykonywanego programu a stan flag musi być kontrolowany tak często, jako tylko jest to możliwe.

Najlepszym - moim zdaniem - sposobem kontroli stanu timera jest wykorzystanie systemu przerwań. Jak wcześniej wspomniałem, określone zdarzenia związane ze stanem timera powodują ustawianie flag w rejestrze TIMSK. Powodem ustawienia flagi może być przepełnienie rejestru liczącego, spełnienie warunku porównania czy też zakończenie działania przez funkcję pomiaru czasu trwania impulsu związaną z wejściem ICP. Tyle gwoli przypomnienia. O ile wykonywanie funkcji obsług przerwań jest

dozwolone, CPU mikrokontrolera przerywa wykonywanie bieżącego programu lub wychodzi ze stanu uśpienia i wykonuje skok pod ściśle określony adres związany z danym powodem przerwania. Jednocześnie zapamiętany zostaje stan licznika rozkazów tak, że jest możliwe jego odtworzenie w momencie powrotu do programu głównego.

Jest to metoda bardzo efektywna - oszczędza czas mikrokontrolera angażując CPU tylko wówczas, gdy to jest naprawdę potrzebne, chociaż nastręcza pewne trudności przy implementacji. Program główny jest bowiem przerywany w momencie, który trudno przewidzieć i to programista musi zadbać o to, aby przy wejściu do procedury obsługi przerwania i po jej opuszczeniu program nadal wykonywany był normalnie. Odpowiednie przerwania załączane są przez nastawy bitów w rejestrze TIMSK (*Timer Interrupt Mask*).

Poniższy przykład w asemblerze ilustruje w jaki sposób włączyć procedurę obsługi przerwania na skutek przepełnienia Timera 2:

```
ldi r16,1<<OCIE2
out TIMSK,r16    ;zezwolenie na
      ;przerwanie Output
      ;Compare Timera 2
sei   ;zezwolenie na
      ;przyjmowanie przerwań
```

Tryby pracy, w które wyposażono Timer 1 i Timer 2 umożliwiają również nastawy akcji wykonywanych w sposób sprzętowy, bez konieczności wykonywania żadnego podprogramu. Odpowiednie wyprowadzenie mikrokontrolera może zostać skonfigurowane w taki sposób, aby było ustawiane, zerowane bądź też negowane w momencie spełnienia warunku porównania. W stosunku do dwóch poprzednich rozwiązań ten tryb nie angażuje w żaden sposób CPU mikrokontrolera. Poniższy przykład ilustruje ten sposób konfiguracji z wykorzystaniem Timera 2:

```
ldi r16,(1<<COM20)|(1<<CS20)
out   TCCR2,r16
;OC2 negowany po spełnieniu
;warunku compare/match

;zegar = zegar systemowy
ldi   r16,32
out   OCR2,r16   ;ustawienie
;porównywanej wartości na 32
```

Poziom logiczny wyprowadzenia OC2 jest negowany w momencie spełnienia warunku porównania (gdy licznik Timera 2 osiągnie wartość dziesiętną 32). Zawiera on też sposób ustawienia wartości porównywanej. Konfiguracja timera jest dokonywana przy pomocy ustawienia bitów COMx0 i COMx1 w rejestrze TCCRx - w przypadku użycia Timera 2 są to bity COM20 i COM21 w rejestrze TCCR2.

Należy jednak pamiętać o tym, że wybór trybu pracy timera nie wpływa na ustawienie kierunku linii portu właściwej dla OC2. Aby zezwolić na ustawianie wartości wyprowadzenia OC2, odpowiedni bit konfiguracji kierunku bitu portu musi być ustawiony w taki sposób aby wyprowadzenie to pracowało jako wyjściowe.

Opcje nastaw zegara

Generator zegarowy AVR zawiera preskaler podłączony do multipleksera. Preskaler to dzielnik częstotliwości zegara. Został on zaimplementowany jako licznik z kilkoma wyjściami o różnych stopniach podziału. W przypadku AT90S8535 jest to 10-bitowy licznik używany do wytworzenia czterech (w przypadku Timera 2 sześciu) różnych częstotliwości taktujących timery, wynikających z podziału częstotliwości generatora zegarowego. Multiplekser używany jest do wyboru która z czterech (sześciu) częstotliwości używana jest jako podstawa czasu timera. Alternatywnie multiplekser może być użyty do ominięcia preskalera oraz konfiguracji zewnętrznego wyprowadzenia jako wejściowego dla timera.

Timery 0 i 1 są timerami synchronicznymi i używają zegara systemowego CPU jako źródła sygnału zegarowego. Asynchroniczny Timer 2 wymaga własnego preskalera co czyni go niezależnym od zegara systemowego. Na **rys. 1** pokazano połączenia pomiędzy preskalerem i multiplekserem. Danych na temat konkretnej konfiguracji dla danego mikrokontrolera AVR należy szukać w jego karcie katalogowej. W **tab. 1** zawarto listę możliwych nastaw preskalera. I tu również należy odwołać się do danych zawartych w konkretnej karcie katalogowej, gdzie prawdopodobnie będą one opisane dokładniej i powiązane z konkretnym modelem mikrokontrolera.

Jacek Bogusz, AVT
jacek.bogusz@ep.com.pl