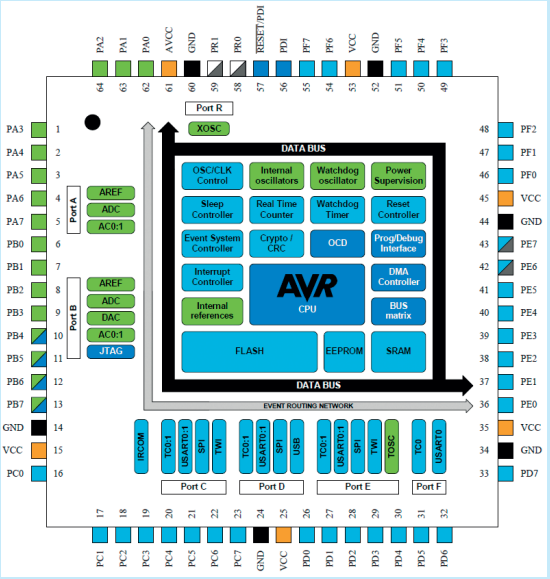
Mikrokontrolery Xmega (4)

Timery

Długo zastanawiałem się, w jaki sposób i jakie tematy poruszyć w artykułach o licznikach. W ATmega liczniki były bardzo rozbudowane i dosyć mocno zagmatwane. W XMEGA bałagan w rejestrach, znany z klasycznych AVR-ów, został uporządkowany, a oprócz tego dołożono szereg różnych możliwości. Dodatkowo, liczniki mogą współpracować z systemem zdarzeń, który sam w sobie udostępnia wiele ciekawych możliwości.



Rysunek 1. Schemat blokowy mikrokontrolera z wyszczególnionymi peryferiami

W mikrokontrolerach Xmega timery są związane z portami I/O. Spójrzmy na schemat blokowy mikrokontrolera ATxmega128A3U pokazany na **rysunku 1**. Timery oznaczone są skrótem TC od *Timer/Counter*. Z portami C, D, E związane są dwa timery o numerach 0 i 1, a w porcie F mamy tylko jeden timer o numerze 0. Numer oznacza typ timera. Aby skonfigurować te timery, musimy odwoływać się do nich używając ich nazw, które są następujące: TCC0, TCC1, TCD0, TCD1, TCE0, TCE1, TCF0. Przyporządkowanie portom oznacza, że wejścia i wyjścia timera, takie jak PWM czy przechwytywanie, dostępne są na określonych pinach portu, który jest związany z timerem.

Wszystkie timery typu 0 i 1 w mikrokontrolerach XMEGA są 16-bitowe, a więc mogą liczyć od 0 do 65535. Korzystając z systemu zdarzeń (który będzie opisany w późniejszych odcinkach) możemy timery łączyć ze sobą, co pozwala na uzyskanie timera o długości nawet 112 bitów. W nowszych modelach Xmega z rodziny AU możemy timery 16-bitowe podzielić na dwa niezależne timery 8-bitowe, które nazywamy timerami typu 2. W ten sposób z 7 timerów możemy uzyskać aż 14 timerów w jednym procesorze (a cały czas omawiamy ATxmega128A3U, modele A1U mają jeszcze więcej peryferiów).

Warto wiedzieć, że każdy timer może liczyć w górę lub w dół, w zależności od naszych potrzeb, a zmianę kierunku liczenia wywołuje się ustawiając odpowiedni bit w rejestrze konfiguracyjnym.

Timery mają szereg rejestrów i zanim zaczniemy pisać programy, poznajmy najważniejsze z nich:

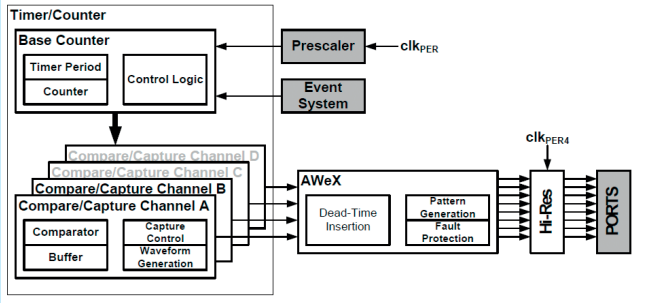
- CNT – jest to rejestr przechowujący aktualną wartość timera, która zwiększa się lub zmniejsza z każdym taktem sygnału sterującego timer (od *counter*).
- PER – rejestr PER (od *period*) wyznacza maksymalną wartość, którą timer może osiągnąć. Zatem, jeśli PER=10, to timer będzie liczył od 0 do 10, a więc przepełnienie będzie następowało co 11 cykli. Po włączeniu zasilania PER domyślnie jest ustawiany na wartość 65535. W przypadku, kiedy timer liczy w dół, rejestr PER określa wartość początkową, od której timer liczy do zera.
- CCx – rejestry te są wykorzystywane do funkcji Compare/Capture czyli porównania i przechwytywania.

Table 32-3. Port C - alternate functions.

PORTC	PIN#	INTERRUPT	TCC0 (1)(2)	AWEXC	TCC1	USART C0 (3)	USART C1	SPIC (4)	TWIC	TWIC w/ext driver	CLOCKOUT (5)	EVENTOUT (6)
PC0	16	SYNC	OC0A	$\overline{OC0ALS}$					SDA	SDAIN		
PC1	17	SYNC	OC0B	OC0AHS		XCK0			SCL	SCLIN		
PC2	18	SYNC/ ASYNC	OC0C	$\overline{OC0BLS}$		RXD0				SDAOUT		
PC3	19	SYNC	OC0D	OC0BHS		TXD0				SCLOUT		
PC4	20	SYNC		$\overline{OC0CLS}$	OC1A			$\overline{SS}$				
PC5	21	SYNC		OC0CHS	OC1B		XCK1	MOSI				
PC6	22	SYNC		$\overline{OC0DLS}$			RXD1	MISO			RTCOUT	
PC7	23	SYNC		OC0DHS			TXD1	SCK			clk_{PER}	EVOUT
GND	24											
VCC	25											

Notes:
1. Pin mapping of all TC0 can optionally be moved to high nibble of port.
2. If TC0 is configured as TC2 all eight pins can be used for PWM output.
3. Pin mapping of all USART0 can optionally be moved to high nibble of port.
4. Pins MOSI and SCK for all SPI can optionally be swapped.
5. CLKOUT can optionally be moved between port C, D and E and be on pin 4 or 7.
6. EVOUT can optionally be moved between port C, D and E and be on pin 4 or 7.

Rysunek 2. Przyporządkowanie wyjść OCx timerów portu C



Rysunek 3. Schematyczna budowa timera

Porównywanie wykorzystuje się do generowania sygnałów PWM. Rejestr CNT jest bezustannie porównywany z CCx, a w zależności od tego który z tych rejestrów ma większą wartość, ustalany jest stan logiczny odpowiedniego pinu OCx (zobacz kolumny TCC0 i TCC1 w tabeli na **rysunku 2**). Przechwycenie polega na zapisaniu wartości rejestru CNT do CCx w chwili wystąpienia określonego zbocza sygnału na wybranej nóżce procesora. Warto zaznaczyć, że XMEGA nie mają dedykowanych do tego celu wejść ICP, tak jak ATmega, lecz możemy wybrać dowolny pin procesora poprzez system zdarzeń. Timery typu 0 mają cztery kanały porównania/przechwycenia o nazwach CCA, CCB, CCC, CCD, a timery typu 1 mają tylko CCA i CCB (schematyczną budowę timera przedstawiono na **rysunku 3**).

- CTRLx – są to rejestry konfiguracyjne.
- INTCTRLx – rejestry konfigurujące przerwania.

Timery w mikrokontrolerach XMEGA mogą być taktowane sygnałem z systemu dystrybucji sygnałów zegarowych lub mogą pochodzić z systemu zdarzeń. System zdarzeń daje bardzo duże możliwości i jest to temat na osobny artykuł. Sygnał zegarowy możemy podzielić preskalerem, podobnie jak w ATmega. Przepełnienie licznika oraz wystąpienie zdarzeń CCx może generować przerwanie lub zdarzenie dla systemu zdarzeń oraz DMA.

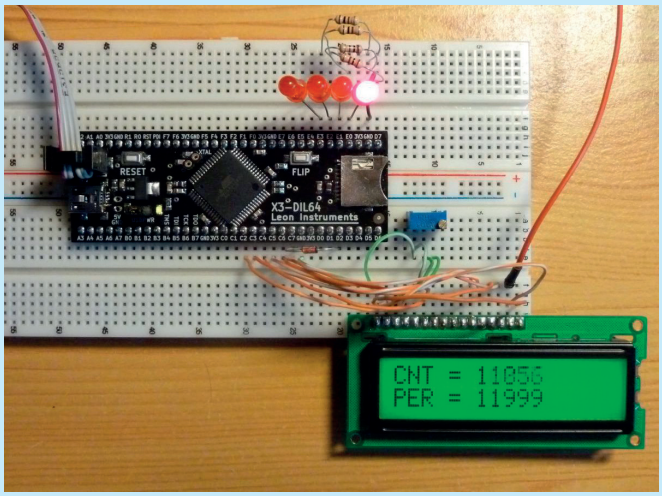
Z bardziej wyrafinowanych możliwości timerów w mikrokontrolerach ATxmega należy wyszczególnić:

- tryby pracy umożliwiające pomiar częstotliwości, okresu i wypełnienia sygnału zegarowego, doprowadzonego do dowolnego pinu procesora,
- PWM z korekcją fazy i częstotliwości,
- współpraca z DMA,
- dodatki rozszerzające rozdzielność PWM,
- sprzętowa obsługa enkoderów,
- układ AWEX umożliwiający generowanie sygnałów np. do sterowania silnikami elektrycznymi, kontrolę czasu martwego,
- XMEGA Custom Logic dostępne w modelach serii E, które są bardzo prostym odpowiednikiem komórek logicznych układów FPGA, umożliwiające uzyskanie np. sygnałów modulowanych.

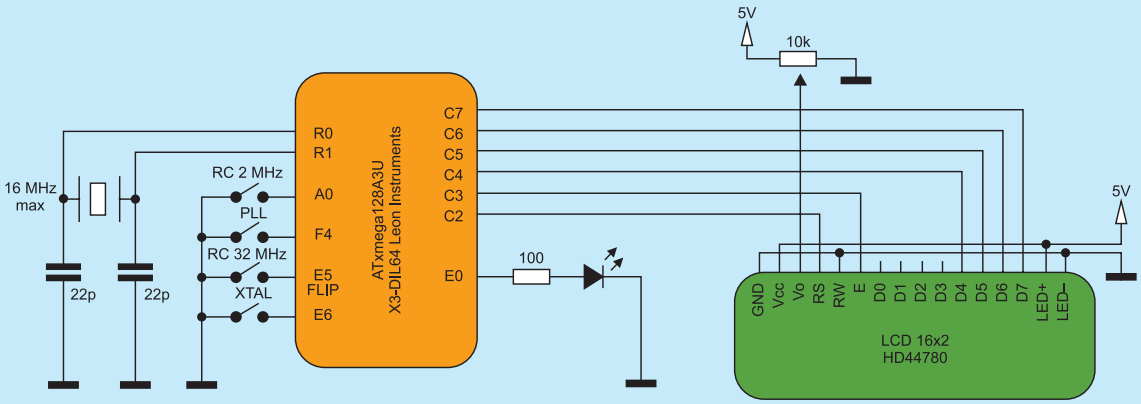
Prostsze możliwości timerów Xmega poznamy w kilku kolejnych odcinkach kursu, a czytelników zainteresowanych wykorzystaniem bardziej zaawansowanych funkcji odsyłam do książki Tomasza Francuza *AVR. Praktyczne projekty*, w której zostały szczegółowo opisane.

Podstawowe operacje i przerwania

Aby zapoznać się z podstawami pracy timerów, napiszemy program demonstrujący działanie timera TCC0 oraz jego rejestru PER i systemu przerwań. Po włączeniu zasilania, rejestr CNT timera TCC0 rozpocznie zliczanie w górę, aż do osiągnięcia wartości PER, która domyślnie jest ustawiona na 65535. Wartość tę można będzie zmienić przyciskiem FLIP, który jest na płytce X3-DIL64. Aby było wiadomo, jakie aktualnie wartości są w rejestrach licznika CNT oraz PER, będziemy odczytywać je z wyświetlacza LCD. Dodatkowo, każde zrównanie się rejestrów CNT oraz PER wywoływać będzie przerwanie przepełnienia OVF, co spowoduje zmianę stanu diody podłączonej do pinu E0.



Fotografia 5. Zdjęcie układu zbudowanego na płytce stykowej

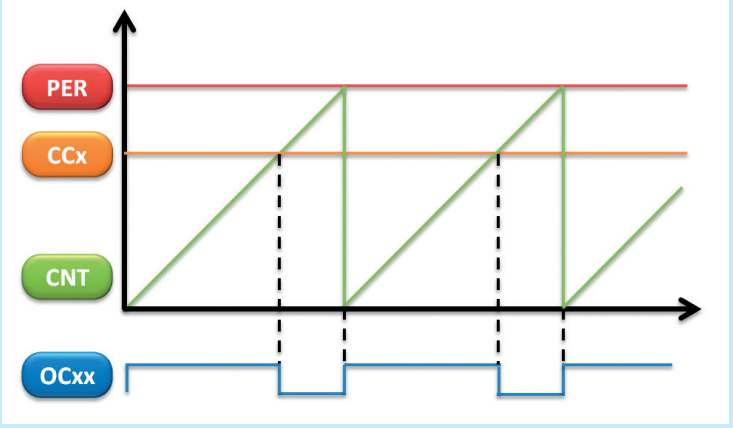


Rysunek 4. Schemat układu demonstrującego pracę timera

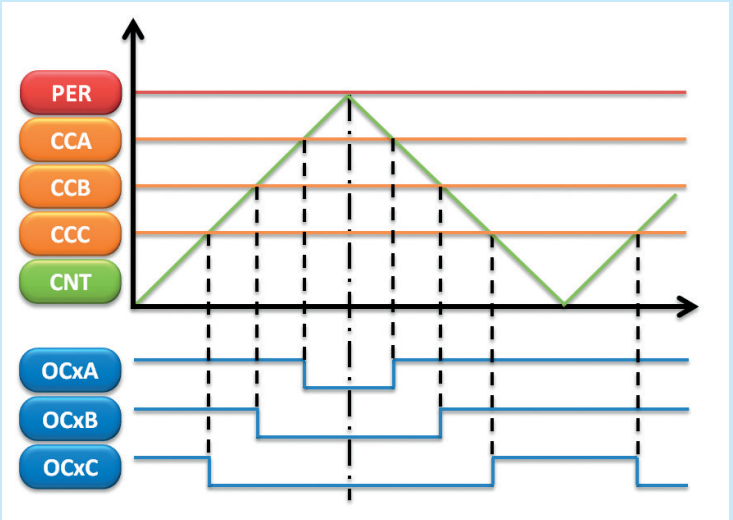
Listing 1. Program demonstrujący podstawowe możliwości timerów w XMEGA

```
#define  F_CPU  2000000UL  // (1)
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include „hd44780.h”

int main(void) {
// przycisk (2)
  PORTE.DIRCLR = PIN5_bm;  // pin E5 jako wejście (przycisk FLIP)
  PORTE.PIN5CTRL = PORT_OPC_PULLUP_gc;  // podciągnięcie do zasilania
// dioda LED
  PORTE.DIRSET = PIN0_bm;  // pin E0 jako wyjście
// wyświetlacz
  LcdInit();
// konfigurowanie przerwań, przepełnienie ma generować przerwanie LO
  TCC0.INTCTRLA = TC_OVFINTLVL_LO_gc;
// odblokowanie przerwań o priorytecie LO
  PMIC.CTRL = PMIC_LOLVLEN_bm;
  sei();  // globalne odblokowanie przerwań
// konfiguracja timera (3)
  TCC0.CTRLB = TC_WGMODE_NORMAL_gc;  // tryb normalny
//  TCC0.CTRLFSET = TC0_DIR_bm;  // liczenie w dół
// ustawienie preskalera i uruchomienie timera
  TCC0.CTRLA = TC_CLKSEL_DIV1024_gc;
// pętla główna (4)
  while(1) {
// wyświetlenie aktualnej wartości licznika CNT i PER
// CNT = ...
// PER = ...
    LcdClear();
    Lcd(„CNT = „);
    LcdDec(TCC0.CNT);
    Lcd2;
    Lcd(„PER = „);
    LcdDec(TCC0.PER);
    _delay_ms(100);  // czekanie 100ms
    if(!(PORTE.IN & PIN5_bm)) {  // jeżeli przycisk FLIP wciśnięty
      TCC0.PER += 1000;  // zwiększ PER o 1000
    }
  }
  ISR(TCC0_OVF_vect) {  // przerwanie przepełnienia TCC0 (5)
  PORTE.OUTTGL = PIN0_bm;  // zamiana stanu diody
}
```



Rysunek 6. PWM single slope



Rysunek 7. PWM dual slope

Zbudujemy układ, którego schemat przedstawiono na **rysunku 4**, a jego realizację na płytce stykowej przedstawia **fotografia 5**. Układ ten posłuży do demonstracji działania generatorów PWM, opisanych w dalszej części artykułu.

Program zamieszczono na listingu 1, a ważniejsze instrukcje zostały opisane. Program zaczynamy, jak zwykle, od dołączenia potrzebnych bibliotek (1). Bibliotekę hd44780 można ściągnąć z serwera FTP Elektroniki Praktycznej lub skopiować z załączonej płyty CD.

Jako że nie będziemy używać żadnych funkcji ani zmiennych globalnych, od razu przechodzimy do pisania funkcji *main()*, w której na początku ustawimy wejście i wyjścia zgodnie z pierwszą częścią kursu, zamieszczoną w EP 2013/11. Fragment odpowiedzialny za inicjalizację podstawowych peryferiów to (2).

Ustawimy kontroler przerwań PMIC. Jak działają przerwania w Xmega opisałem w EP 2013/12. Podobnie jak w każdym innym typie przerwania, możemy ustalić priorytet niski (LO), średni (MED) lub wysoki (HI). Po wybraniu priorytetu dla przerwania przepełniania OVF, musisz również uaktywnić przerwania o tym priorytecie, wpisując *PMIC_xxLVLEN_bm* do rejestru CTRL kontrolera przerwań PMIC.

Przejdźmy do skonfigurowania timera TCC0 (fragment 3 listingu). W rejestrze CTRLB wybieramy normalny tryb pracy. Do rejestru CTRLF nie możemy wpisać danych wprost, jak do innych dotychczas używanych. Można wpisywać do niego wartości poprzez rejestry pomocnicze CTRLFSET oraz CTRLFCLR. Działa to podobnie jak przy ustawianiu portów, co robiliśmy na początku programu. Jedynki wpisane do rejestru SET zostaną wpisane do rejestru CTRLF, a jedynki wpisane do rejestru CLR zostaną w CTRLF wyzerowane. Dzięki takiemu rozwiązaniu nie musimy stosować masek bitowych ani operatorów |= ani &=. Przy pomocy bitu TC0_DIR_bm w rejestrze CTRLF decydujemy, czy timer ma liczyć w górę czy w dół. Ostatnim etapem konfiguracji timera jest wybranie źródła sygnału. Wybieramy sygnał zegarowy o częstotliwości wstępnie podzielonej

przez 1024 za pomocą stałej *TC_CLKSEL_DIV1024_gc*. Po tym poleceniu timer zostaje uruchomiony.

W pętli głównej (4) program będzie wyświetlał bieżącą wartość przechowywaną w rejestrach CNT oraz PER. Dodatkowo, jeśli przycisk FLIP zostanie wciśnięty, to rejestr PER będzie zwiększał się o 1000.

Do opisania pozostała tylko procedura przerwania przepełnienia TCC0_OVF_vect (5). Składa się z zaledwie jednej linijki, która zamienia stan pinu E0. Jeśli przez zgłoszeniem dioda się świeciła, to zgaśnie, a jeśli była wygaszona, to się zaświeci.

Układy PWM

PWM to skrót od *Pulse Width Modulation* czyli modulacja szerokości impulsu. Generowany jest sygnał o stałej amplitudzie i częstotliwości, a zmieniać może się jedynie współczynnik wypełnienia. Dzięki takiemu zabiegowi można bardzo łatwo sterować różnymi urządzeniami: prędkością

Listing 2. Program demonstrujący działanie układów PWM w mir kontrolerach XMEGA

```
#define  F_CPU    2000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include „hd44780.h”

int main(void) {
// przycisk
  PORTE.DIRCLR = PIN5_bm;  // pin E5 jako wejście (przycisk FLIP)
  PORTE.PIN5CTRL = PORT_OPC_PULLUP_gc;  // podciągnięcie do zasilania
// diody od PWM
  PORTE.DIRSET = 0b00001111;  // piny 3..0 jako wyjście
// wyświetlacz
  LcdInit();
// konfiguracja timera (6)
  TCE0.CTRLB = TC_WGMODE_DSBOTH_gc|  // tryb PWM dual-slope
    TC0_CCAEN_bm|
    TC0_CCBEN_bm|
    TC0_CCCEN_bm|
    TC0_CCDEN_bm;
  TCE0.PER = 10000;
  TCE0.CCA = 2000;
  TCE0.CCB = 4000;
  TCE0.CCC = 6000;
  TCE0.CCD = 8000;
// ustawienie preskalera i uruchomienie timera
  TCE0.CTRLA = TC_CLKSEL_DIV1024_gc;
  while(1) {
// wyświetlenie aktualnej wartości licznika CNT i PER
// CNT = ...
// PER = ...
    LcdClear();
    Lcd(„CNT = „);
    LcdDec(TCE0.CNT);
    Lcd2;
    Lcd(„PER = „);
    LcdDec(TCE0.PER);
    _delay_ms(100);
    if(!(PORTE.IN & PIN5_bm)) {  // przycisk FLIP przyspiesza PWM
      TCE0.PER = 1000;
      TCE0.CCA = 10;
      TCE0.CCB = 50;
      TCE0.CCC = 200;
      TCE0.CCD = 500;
      TCE0.CTRLA      =    TC_CLKSEL_DIV1_gc;
    }
  }
}
```

14.12.2 CTRLB – Control register B

Bit	7	6	5	4	3	2	1	0
+0x01	CCDEN	CCCEN	CCBEN	CCAEN	–	WGMODE[2:0]		
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:4 – CCxEN: Compare or Capture Enable**
  Setting these bits in the FRQ or PWM waveform generation mode of operation will override the port output register for the corresponding OCn output pin.
  When input capture operation is selected, the CCxEN bits enable the capture operation for the corresponding CC channel.
- **Bit 3 – Reserved**
  This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.
- **Bit 2:0 – WGMODE[2:0]: Waveform Generation Mode**
  These bits select the waveform generation mode, and control the counting sequence of the counter, TOP value, UPDATE condition, interrupt/event condition, and type of waveform that is generated according to Table 14-4 on page 175.
  No waveform generation is performed in the normal mode of operation. For all other modes, the result from the waveform generator will only be directed to the port pins if the corresponding CCxEN bit has been set to enable this. The port pin direction must be set as output

Table 14-4. Timer waveform generation mode.

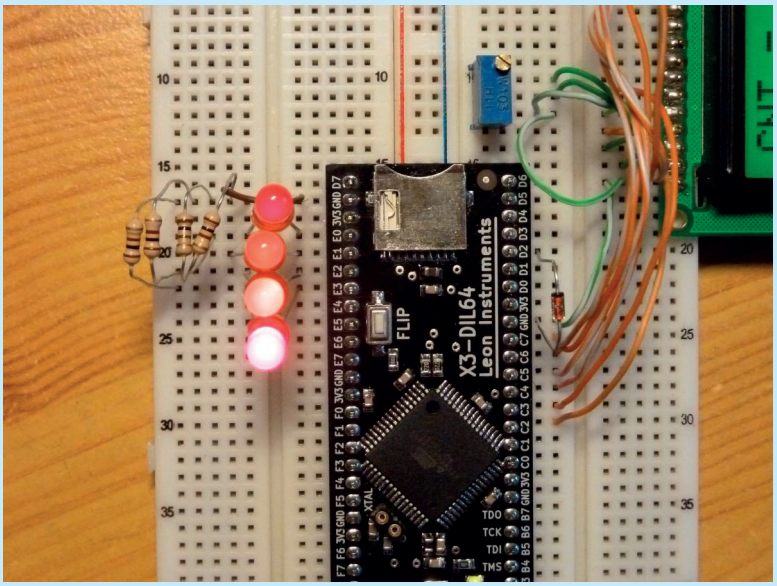
WGMODE[2:0]	Group configuration	Mode of operation	Top	Update	OVFIF/Event
000	NORMAL	Normal	PER	TOP	TOP
001	FRQ	Frequency	CCA	TOP	TOP
010		Reserved	–	–	–
011	SINGLESLOPE	Single-slope PWM	PER	BOTTOM	BOTTOM
100		Reserved	–	–	–
101	DSTOP	Dual-slope PWM	PER	BOTTOM	TOP
110	DSBOTH	Dual-slope PWM	PER	BOTTOM	TOP and BOTTOM
111	DSBOTTOM	Dual-slope PWM	PER	BOTTOM	BOTTOM

Rysunek 8. Opis rejestru CTRLB

silnika, jasnością żarówki lub diody LED.

Mikrokontrolery Xmega mają możliwość wygenerowania jednocześnie bardzo wielu sygnałów PWM. Timer typu 0 może generować cztery takie sygnały, a typ 1 może tylko dwa. Jednak mając do dyspozycji cztery timery typu 0 oraz trzy timery typu 1, możemy uzyskać aż 22 kanały PWM. W nowszych XMEGA (takich jak np. ATxmega128A3U) timery 16-bitowe można podzielić na 8-bitowe timery typu 2, a każdy z nich ma 4 kanały. W końcowym rozrachunku można mieć nawet 32 kanały PWM!

Jeśli wiesz jak działa PWM – przeskocz kilka akapitów i zacznij czytać opis kodu programu.



Fotografia 9. Diody świecące z różną jasnością

Timer generujący PWM może pracować w trybie single slope, czyli licząc zawsze w tym samym kierunku albo może pracować w trybie dual slope, czyli liczyć naprzemiennie w górę i w dół.

W trybie single slope timer zaczyna liczyć od zera do wartości określonej w rejestrze PER. Kiedy rejestr licznika CNT i PER zrównają się, wówczas CNT zostaje wyzerowany i timer zaczyna liczyć od początku. Sygnał PWM dostępny jest na nóżce OCxx. Na początku cyklu, na pinie OCxx jest stan logiczny wysoki. W chwili, kiedy rejestr CNT zrówna się z liczbą wpisaną do rejestru CCX, wówczas pin OCxx ustawia się w stan niski i pozostaje tak do końca cyklu. Wykresy czasowe takiej sytuacji zostały przedstawione na **rysunku 6**.

W trybie dual slope, timer liczy od zera do wartości PER, a potem zmienia kierunek liczenia i wraca do zera. Zmiana stanu pinu OCxx dokonuje się w chwili zrównania się rejestrów CNT i CCx, jednak zależy jeszcze od kierunku liczenia timera. Gdy timer liczy w górę, przy zrównaniu się, stan pinu OCxx zmienia się z wysokiego na niski, a w przypadku liczenia w dół jest odwrotnie. Aby opis był bardziej zrozumiały zamieściłem **rysunek 7**.

Należy mieć na uwadze, że częstotliwość sygnału będzie dwukrotnie mniejsza niż przy identycznie ustawionym trybie single slope, ponieważ cykl pracy jest dwukrotnie dłuższy.

Tryb dual slope czasami nazywany jest określeniem *PWM z korekcją fazy*. Czym on się różni od single slope? Niektórzy twierdzą, że jest lepszy i wynika to z teorii sygnałów… Otóż, jeśli wykorzystujemy tylko jeden kanał timera, to w obu trybach możemy uzyskać całkowicie identyczny sygnał. Różnica jest wtedy, gdy jeden timer kontroluje kilka kanałów PWM, a tym samym kilka odbiorników. W przypadku single slope, na początku cyklu wszystkie odbiorniki, którymi sterujemy, włączają się jednocześnie. Bardzo często PWM wykorzystywany jest do sterowania urządzeniami dużej mocy. Jednoczesne załączenie kilku takich urządzeń może powodować spadki napięć na szynach zasilających. Nie bez powodu mówi się, że *PWM sieje zakłóceniami*. Rozwiązaniem problemu jest zastosowanie PWM dual slope. Spójrz jeszcze raz na rysunek 2 – w tym przypadku w danej chwili otwierany lub zamykany jest tylko jeden z trzech kanałów PWM. Z pewnością ograniczy to zakłócenia generowane przez PWM.

Przejdźmy wreszcie do pisania programu, przedstawionego na **listingu 2**. Po włączeniu zasilania, uruchomione zostaną cztery kanały PWM, do których podłączymy diody LED. Aby zaobserwować jak działa PWM, po starcie programu będzie on skonfigurowany z bardzo dużym preskalerem. Dopiero po wciśnięciu przycisku FLIP zostanie uruchomiona normalna prędkość pracy, a poszczególne diody LED będą świecić się z różną jasnością.

Aby móc skorzystać z dobrodziejstw PWM, musimy nieco inaczej skonfigurować timer, a w szczególności jego rejestr CTRLB – spójrzmy na fragment dokumentacji przedstawiony na **rysunku 8**.

Grupa konfiguracyjna WGMODE odpowiada za tryb pracy timera. Wcześniej wykorzystaliśmy tryb TC_WGMODE_NORMAL_gc. PWM możemy generować przy pomocy czterech trybów – jednego single-slope, w którym timer cały czas zlicza w tym samym kierunku oraz trzech trybów dual-slope. Tryby dual-slope różnią się jedynie chwilami, w których zostanie zgłoszone przerwanie przepełnienia licznika. Ponieważ przerwań w niniejszych przykładzie nie wykorzystujemy, to możemy wybrać dowolny tryb dual-slope (6).

Oprócz tego, w rejestrze CTRLB musimy wybrać, które kanały CCx zamierzamy wykorzystać, wpisując do rejestru odpowiednie stałe *TC0_CCxEN_bm*, gdzie x oznacza wybór kanału A, B, C oraz D. To wystarczy, aby timer zaczął generować sygnały PWM (oczywiście odpowiednie piny muszą być skonfigurowane w rejestrze DIR jako wyjście).

Po uruchomieniu programu, świecą się wszystkie diody, a rejestr CNT licznika jest oczywiście wyzerowany. Zwróć uwagę na wskazania wyświetlacza. Kiedy licznik CNT przekroczy 2000 to zgaśnie dioda podłączona do pinu E0, bo steruje nią kanał A, którego rejestr CCA wynosi 2000. W ten sposób po kolei wszystkie diody mają zgasnąć, aż licznik osiągnie wartość 10000 i zacznie liczyć w dół. Diody będą się po kolei zapalać. Po wciśnięciu przycisku FLIP, preskaler timera zmienimy na 1 zamiast 1024 i ustawimy nieco inne wartości CCx. Dzięki temu będziemy mieć wrażenie, że diody świecą się z różną jasnością, choć rzeczywistości odbywa się podobny proces, jaki oglądaliśmy przed chwilą, ale z dużo większą prędkością. Działanie programu przedstawia **fotografia 9**.

Dominik Leon Bieczyński
www.leon-instruments.pl