Kurs programowania mikrokontrolerów XMEGA (7)

Komparator analogowy

Moim zdaniem między mikrokontrolerami z rodzin ATmega i XMEGA jest przepaść technologiczna – widać to również w obszarze bloków analogowych. Do dyspozycji mamy zaawansowane, wielokanałowe przetworniki analogowo-cyfrowe, cyfrowo-analogowe oraz kilka komparatorów analogowych (a nie jeden, jak w ATmega). W tym artykule zajmiemy się komparatorem analogowym.

W mikrokontrolerach XMEGA wyróżnia się porty cyfrowe i analogowe. Dzięki portom cyfrowym są dostępne wyjścia generatorów PWM, interfejsy komunikacyjne itp. Przyzwyczailiśmy się do tego, że peryferia cyfrowe są powielone na każdym porcie. Dla peryferiów analogowych jest tak samo – przetworniki i komparatory również są powielone. Każdy port analogowy ma dwa komparatory, a procesor ATxmega128A3U na płytce eXtrino XL ma dwa takie porty, zatem mamy do dyspozycji aż cztery komparatory analogowe – oznaczone ACA.AC0, ACA.AC1, ACB.AC0, ACB.AC1.

Komparatory w XMEGA

Komparator jest układem analogowym porównującym napięcia doprowadzone do dwóch wejść: nieodwracającego (+) oraz odwracającego (–). Jeśli napięcie doprowadzone do wejścia nieodwracającego (+) jest wyższe niż napięcie na wejściu odwracającym (–), to na wyjściu komparatora pojawi się poziom logiczny wysoki, a w przeciwnym razie – niski.

Budowę komparatorów analogowych oraz otaczających ich układów przestawiono na **rysunku 1**.

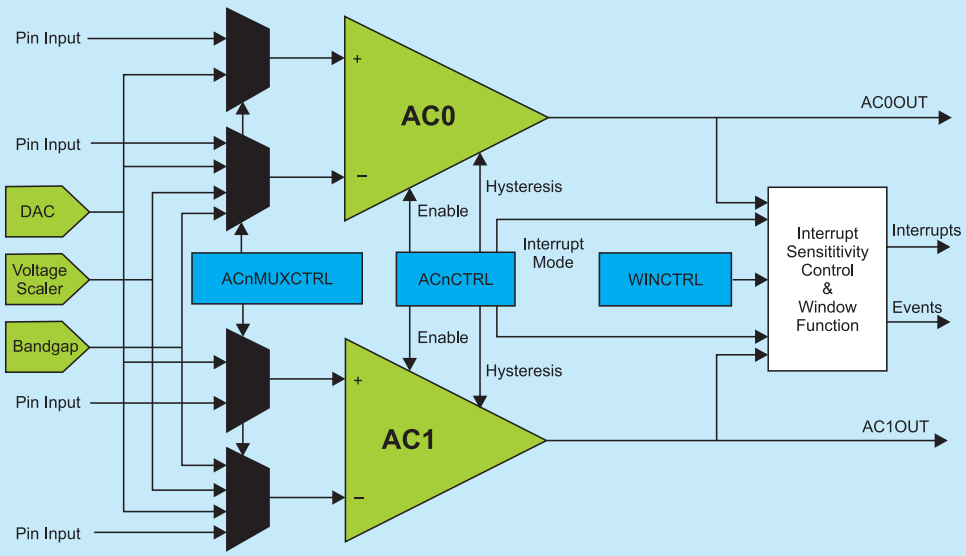
W starych układach ATtiny i ATmega, wejścia komparatora były na sztywno połączone z dwoma ustalonymi pinami i nie można było w żaden sposób tego zmienić. XMEGA jest zdecydowanie bardziej elastyczna. Jako wejście nieodwracające może nam służyć dowolny pin od 0 do 6, a wejściem odwracającym może być pin 0, 1, 3, 5 lub 7.

W praktyce bardzo często porównujemy badany sygnał z ustalonym stałym napięciem referencyjnym. Projektanci mikrokontrolerów XMEGA umożliwili wybór aż trzech źródeł napięcia referencyjnego:

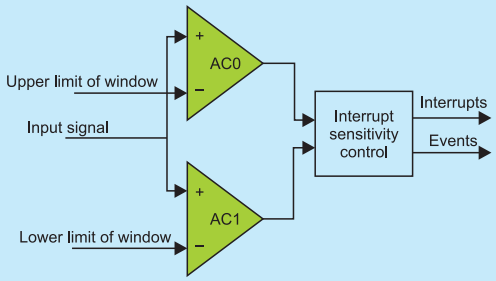
- 1 V bandgap reference,
- 12-bitowy przetwornik C/A,
- 64-stopniowy dzielnik napięcia zasilającego.

Przetwornik C/A można wewnętrznie połączyć z wejściem odwracającym lub nieodwracającym, a bandgap i dzielnik napięcia mogą być połączone tylko z wejściem odwracającym. Zdolność do tworzenia wewnętrznych połączeń, bez żadnych zewnętrznych elementów elektronicznych, daje bardzo ciekawe możliwości. Na przykład bardzo łatwo jest sprawić, by procesor monitorował swoje własne napięcie zasilające, a w razie spadku poniżej wyznaczonego progu, komparator wygeneruje przerwanie (EP 12/2013) lub zdarzenie (EP 3/2014).

Zdarza się, że różnica napięć doprowadzonych do obu wejść komparatora jest bardzo niewielka. Może to powodować chaotyczne, szybkie i wielokrotne przełączenia stanu wyjścia. Aby temu zapobiec, dobrze jest do-



Rysunek 1 Komparatory w portach analogowych XMEGA



Rysunek 2. Komparator okienkowy

dać do komparatora dodatnie sprzężenie zwrotne, przez co utworzy się histereza. W XMEGA wystarczy ustawić dwa bity w odpowiednim rejestrze, by włączyć histerezę małą (30 mV) lub dużą (60 mV).

Wyjście komparatora możemy także połączyć z przeznaczonym do tego pinem mikrokontrolera. Niestety, nie mamy tutaj możliwości wyboru – wyjście komparatora 0 połączone jest z pinem 7 odpowiadającego mu portu analogowego, a wyjście komparatora 1 połączono z pinem 6.

W obrębie jednego portu analogowego dwa komparatory mogą współpracować tworząc komparator okienkowy. Do takiego układu doprowadzamy jedno napięcie badane oraz dwa napięcia referencyjne. Komparator okienkowy umożliwia stwierdzenie czy badany sygnał jest pomiędzy napięciami referencyjnymi (w oknie), poniżej lub powyżej okna. Schemat takiego rozwiązania przedstawiono na **rysunku 2**.

Prosty program „na rozgrzewkę"

W tym odcinku kursu przygotujemy dwa programy. Jeden bardzo prosty, żeby zaznajomić się z podstawami działania komparatora. Drugi będzie nieco bardziej praktyczny i będzie wykorzystywał peryferia, omówione w poprzednich odcinkach kursu – zobaczymy jak zrobić bardzo prosty miernik pojemności kondensatorów.

Pierwszy program będzie porównywać dwa napięcia:

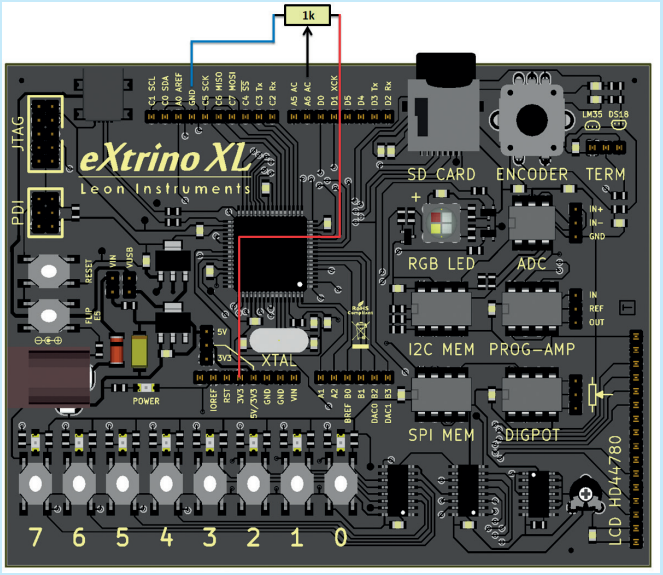
- Napięcie z wbudowanego dzielnika doprowadzamy do wejścia odwracającego (–), a za pomocą przycisków 0 i 1 na płytce eXtrino XL będziemy mogli to napięcie regulować.
- Do wejścia nieodwracającego (+) dołączamy potencjometr (o rezystancji z zakresu 1…100 kΩ), którym będziemy mogli regulować napięcie. Środkowe wyprowadzenie potencjometru należy połączyć z pinem A6, a wyprowadzenia skrajne do zasilania 3,3 V oraz masy GND.

Wynik porównania oraz ustalone napięcie z wbudowanego dzielnika będziemy monitorować na wyświetlaczu (użyłem wyświetlacza VFD 4×20, ale można zastosować dowolny wyświetlacz ze sterownikiem HD44780, co opisałem w EP 11/2013). Przejdźmy do programu, którego kod przedstawiono na **listingu 1**, a schemat układu przedstawiono na **rysunku 3.**

Pierwsze dwie linijki funkcji *main()* to standardowa inicjalizacja wyświetlacza oraz *PORTX*. Następnie musimy skonfigurować komparator, wpisując odpowiednie bity do zaledwie trzech rejestrów. W rejestrze *ACA.AC0MUXCTRL* konfigurujemy, z którymi pinami lub źródłami napięcia referencyjnego mają być połączone wejście odwracające i nieodwracające komparatora o numerze 0. Wejście nieodwracające łączymy z pinem A6, natomiast wejście odwracające będzie połączone z wewnętrznym dzielnikiem napięcia.

Listing 1. Kod programu do pierwszego ćwiczenia

```
#define F_CPU 2000000UL
#include <avr/io.h>
#include „extrino_portx.h"
#include „hd44780.h"

int main(void) {

    PortxInit();                                    // inicjalizacja PORTX
    LcdInit();                                      // inicjalizacja wyświetlacza

    // konfiguracja komparatora 0 w porcie A
    ACA.AC0MUXCTRL    =    AC_MUXPOS_PIN6_gc |      // wejście + PIN A6
                           AC_MUXNEG_SCALER_gc;     // wejście - dzielnik napięcia
    ACA.CTRLB         =    32;                      // początkowe ustawienie dzielnika napięcia
    ACA.AC0CTRL       =    AC_ENABLE_bm;            // włączenie komparatora

    while(1) {

        // wyświetlanie wyniku z komparatora
        Lcd1;
        Lcd(„Vref = „);
        LcdDecComma((330 * (ACA.CTRLB + 1)) / 64, 2);
        Lcd(„V");
        Lcd(„    „);
        Lcd2;
        if(ACA.STATUS & AC_AC0STATE_bm) {           // badanie wyjścia komparatora
            Lcd(„Vin > Vref");
        } else {
            Lcd(„Vin < Vref");
        }

        // zmiana napięcia referencyjnego
        if(PORTX.IN & PIN0_bm) {                    // jeżeli wciśnięto przycisk 0
            if(ACA.CTRLB > 0) ACA.CTRLB--;          // zmniejszanie Vref
            PORTX.OUTCLR    =    PIN0_bm;           // wyłączenie diody LED 0
            _delay_ms(100);                         // czekanie 100ms
        }

        if(PORTX.IN & PIN1_bm) {                    // jeżeli wciśnięto przycisk 1
            if(ACA.CTRLB < 63 ) ACA.CTRLB++;        // zwiększanie Vref
            PORTX.OUTCLR    =    PIN1_bm;           // wyłączenie diody LED 1
            _delay_ms(100);                         // czekanie 100ms
        }

        PORTX.OUTSET    =    PIN1_bm | PIN0_bm;    // włączenie diod LED 1 i 0
    }
}
```



Rysunek 3. Schemat układu demonstrującego działanie komparatora

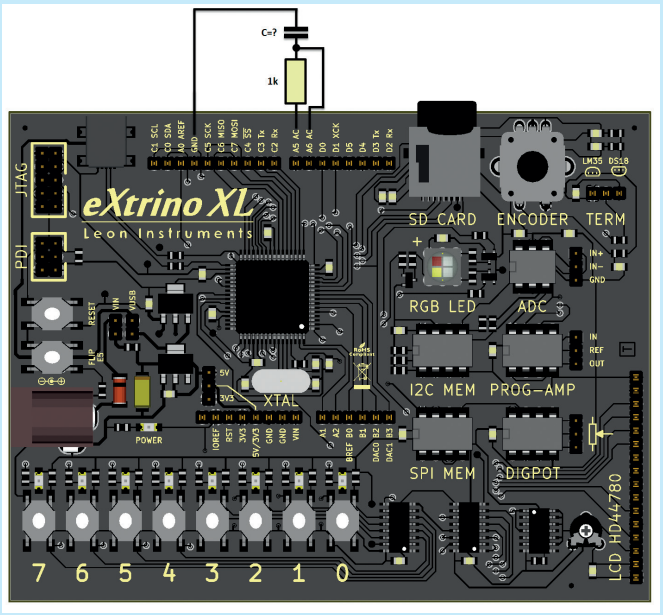
W rejestrze *ACA.CTRLB* ustalamy 64-stopnowy dzielnik napięcia zasilającego. Możemy wybrać liczbę z przedziału od 0 do 63. Im wyższa to będzie liczba, tym wyższe napięcie uzyskamy. Uzyskana napięcie można wyliczyć z prostego wzoru:

$$V_{ref} = \frac{V_{cc} \cdot (CTRLB + 1)}{64}$$

Wynika stąd, że maksymalne napięcie, jakie możemy ustawić, wynosi Vcc, czyli 3,3 V. Natomiast napięcie minimalne to 1/64 napięcia Vdd, czyli ok. 0,05 V. Dostępny jest jeden dzielnik napięcia na oba komparatory w jednym porcie.

W ostatnim rejestrze, *ACA.AC0CTRL*, uruchamiamy komparator. Pamiętajmy przy tym, że w XMEGA komparatory są domyślnie wyłączone, w przeciwieństwie do starych AVR-ów, gdzie po włączeniu zasilania komparator był domyślnie włączony.

Następnie, w pętli *while(1)* sprawdzamy, czy wciśnięte są przyciski 0 i 1 (dołączone do *PORTX*). Jeśli tak, to odpowiednio zwiększamy lub zmniejszamy zawartość rejestru *CTRLB*, pamiętając, że jego zawartość może być liczbą z przedziału od 0 do 63. Dodatkowo, diody LED numer 0 i 1 będą się świecić, a kiedy odpowiadający im przycisk zostanie wciśnięty, diody będą mrugać.



Rysunek 4. Schemat miernika pojemności kondensatorów

Pomiar komparatorem pojemności kondensatora

Istnieje wiele metod pomiaru pojemności kondensatora, a najprostsza z nich polega na zbudowaniu z wykorzystaniem jednego lub dwóch komparatorów oscylatora o częstotliwości zależnej od dołączonej pojemności. Wszystko, co tylko możliwe, „upchniemy" wewnątrz procesora, a jedyne co do niego musimy dołączyć, to rezystor 1 kΩ oraz badany kondensator. Schemat układu przedstawia **rysunek 4**, a jego zdjęcie widoczne jest na **fotografii 5**.

Jeśli do kondensatora przyłożymy pewne stałe napięcie zasilające, doprowadzone przez rezystor, to napięcie na kondensatorze zacznie rosnąć w sposób wykładniczy, stopniowo zbliżając się do napięcia zasilającego. Kiedy elektrody kondensatora zewrzemy rezystorem, napięcie na nim będzie stopniowo spadać, również w sposób wykładniczy, aż w końcu osiągnie wartość zerową.

W prezentowanej metodzie, wykorzystuje się pomiar czasu, jaki jest potrzebny na naładowanie i rozładowanie kondensatora pomiędzy dwoma, charakterystycznymi napięciami granicznymi. Napięcie na kondensatorze będzie się naprzemiennie zwiększać się i zmniejszać pomiędzy tymi dwoma napięciami. Wykresy tych napięć przedstawiono na **rysunku 6**.

Czas ładowania i rozładowania jest tym dłuższy, im większa jest pojemność kondensatora i co najważniejsze jest to zależność liniowa. Tak więc znając czas, wystarczy go przemnożyć lub podzielić przez odpowiedni współczynnik, by poznać pojemność.

Każdy student elektroniki miał (lub będzie miał) całkiem zaawansowaną matematyczną teorię na ten temat, więc pozwolę sobie o niej nie pisać i przejdę od razu do sedna problemu. Spójrz na schemat z rysunku 6 oraz na kod programu z **listingu 2**.

Potrzebujemy dwa źródła napięć granicznych oraz dwa komparatory. Do tego celu idealnie nadaje się tryb okienkowy, pozwalający na sprzężenie dwóch komparatorów. Jakie powinny być wartości napięć granicznych? Ich wartość nie jest krytyczna, ale najważniejsze jest, by były one stabilne. Im większa będzie między nimi różnica, tym większy będzie również czas ładowania i rozładowania kondensatora. W praktycznych rozwiązaniach często ustala się je na 1/3 i 2/3 napięcia zasilającego. W tym przypadku będą to ok. 1 V i 2 V. Zapewne niejeden czytelnik dostrzeże analogię tego rozwiązania do nieśmiertelnego układu NE555.

W celu uzyskania napięć granicznych, do dyspozycji mamy dzielnik napięcia zasilającego, źródło referencyjne (bandgap) oraz

przetwornik analogowo-cyfrowy (DAC). Ze względu na stabilność, najlepiej byłoby użyć bandgap oraz przetwornik DAC, jednak w kursie nie był on jeszcze omawiany, dlatego zastosowałem dzielnik napięcia zasilającego.

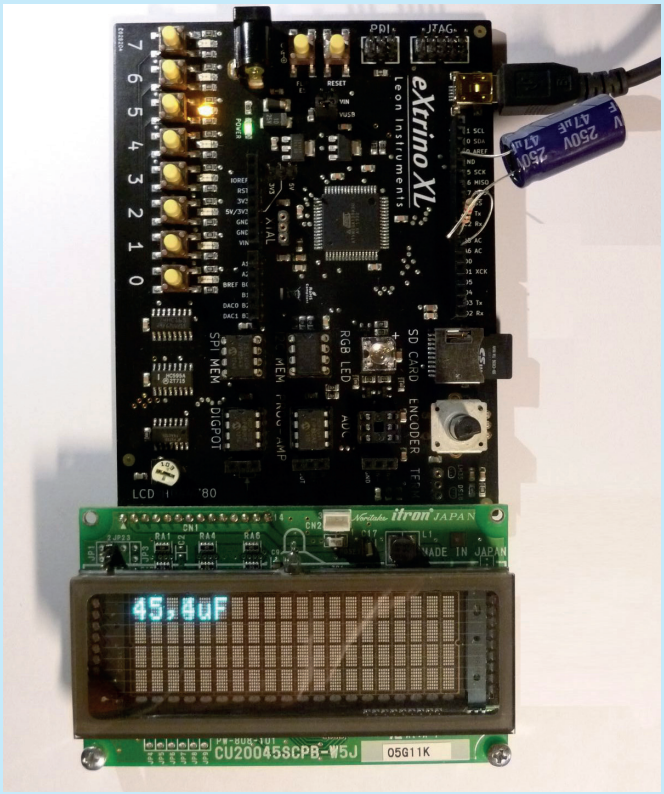
Kiedy napięcie na kondensatorze osiągnie wartość graniczną, układ nadzorujący komparatory wygeneruje odpowiednie zdarzenie i przerwanie komparatora o nazwie *ACA_ACW_vect*. Przerwanie ma na celu zmianę stanu pinu A5. Pin ten naprzemiennie zasila kondensator i rozładowuje go.

Pomiar czasu jest realizowany za pomocą timera C0. W chwili rozpoczęcia ładowania/rozładowywania timer jest zerowany i zaczyna liczyć w górę. Kiedy napięcie na kondensatorze osiągnie wartość graniczną, wówczas jest generowane zdarzenie (równocześnie z przerwaniem!). Zdarzenie jest przekazywane przez system zdarzeń do timera, który jest tak skonfigurowany, by dokonywał przechwycenia stanu licznika do rejestru *CCA*.

Jest tu pewien szkopuł. Mianowicie czas ładowania i rozładowywania nie jest taki sam. Dzieje się tak dlatego, że kondensator jest ładowany/rozładowywany nie

Listing 2. Kod miernika częstotliwości

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include „extrino_portx.h”
#include „hd44780.h”

int main(void) {

    PortxInit();                                // inicjalizacja portu X
    LcdInit();                                  // inicjalizacja wyświetlacza LCD

    // konfiguracja komparatora 0 w porcie A
    ACA.AC0MUXCTRL    =    AC_MUXPOS_PIN6_gc |    // wejście + PIN A6
                           AC_MUXNEG_SCALER_gc;   // wejście - dzielnik napięcia
    ACA.CTRLB         =    38;                    // 2/3 napięcia Vcc jako odniesienie
    ACA.AC0CTRL       =    AC_HYSMODE_LARGE_gc|   // duża histereza
                           AC_ENABLE_bm;          // włączenie komparatora

    // konfiguracja komparatora 1 w porcie A
    ACA.AC1MUXCTRL    =    AC_MUXPOS_PIN6_gc |    // wejście + PIN A6
                           AC_MUXNEG_BANDGAP_gc;  // 1V bandgap źródło nap odniesienia
    ACA.AC1CTRL       =    AC_HYSMODE_LARGE_gc|   // duża histereza
                           AC_ENABLE_bm;          // włączenie komparatora

    // konfiguracja komparatora okienkowego
    ACA.WINCTRL       =    AC_WEN_bm|             // uruchomienie trybu okienkowego
                           AC_WINTMODE_INSIDE_gc| // wybór przerwania
                           AC_WINTLVL_LO_gc;      // wybór priorytetu przerwania

    // pin do ładowania/rozładowania kondensatora
    PORTA.DIRSET      =    PIN5_bm;               // A5 jako wyjście
    PORTA.OUTSET      =    PIN5_bm;               // stan niski w celu rozładowania kondensatora

    // system zdarzeń
    EVSYS.CH0MUX      =    EVSYS_CHMUX_ACA_CH1_gc;// zdarzenie komparatora jest przesyłane kanałem 0 systemu
zdarzeń

    // konfiguracja timera
    TCC0.CTRLD        =    TC_EVACT_FRQ_gc |      // tryb pomiaru częstotliwości
                           TC_EVSEL_CH0_gc;       // zdarzenie wywołuje kanał 0
    TCC0.CTRLB        =    TC0_CCAEN_bm|          // przechwycenie do rejestru CCA
                           TC_WGMODE_NORMAL_gc;   // zwykły tryb zliczania impulsów
    TCC0.CTRLA        =    TC_CLKSEL_DIV8_gc;     // preskaler i uruchomienie timera
    TCC0.INTCTRLA     =    TC_OVFINTLVL_LO_gc;    // pritrytet przerwania od przepełnienia

    // konfiguracja przerwań
    PMIC.CTRL         =    PMIC_LOLVLEN_bm;       // włączenie przerwań o priorytecie LO
    sei();

    uint16_t wynik;                             // zmienna przechowująca pojemność kondensatora

    while(1) {

        // wyświetlanie wyniku
        Lcd1;                                   // kursor na początek 1 linii
        wynik = TCC0.CCA / 31;                  // komentarz w tekście (1)
        LcdDecComma(wynik,1);                   // wyświetlenie liczby stałoprzecinkowej
        Lcd(„uF       „);

        // wyświetlenie stanu pinu A5 na diodach PORTX
        PORTX.OUT     =    PORTA.OUT & PIN5_bm;
    }
}

ISR(ACA_ACW_vect) {                             // przerwanie komparatora

    if(PORTA.OUT & PIN5_bm) {                   // jeżeli teraz jest ładowanie
        ACA.WINCTRL   =    AC_WEN_bm|
                           AC_WINTMODE_BELOW_gc|
                           AC_WINTLVL_LO_gc;
    } else {                                    // jeżeli teraz jest rozładowywanie
        ACA.WINCTRL   =    AC_WEN_bm|
                           AC_WINTMODE_ABOVE_gc|
                           AC_WINTLVL_LO_gc;
    }

    PORTA.OUTTGL      =    PIN5_bm;             // zmiana stanu pinu A5 na przeciwny
}

ISR(TCC0_OVF_vect) {                            // przepełnienie timera C0
    PORTA.OUTTGL      =    PIN5_bm;             // zmiana stanu pinu A5 na przeciwny
}
```



Fotografia 5. Sposób podłączenia rezystora i kondensatora

tylko przez rezystor podłączony do płytki, ale także przez rezystancję pinu procesora, która jest inna w stanie wysokim i inna w stanie niskim. Powodowałoby to naprzemienne pokazywanie dwóch wyników, nieznacznie różniących się od siebie. Na szczęście projektanci mikrokontrolerów XMEGA dostrzegli ten problem – aby go wyeliminować, należy włączyć tryb pomiaru częstotliwości. Dodatkowym bonusem jest to, że po pełnym cyklu timer zostanie automatycznie wyzerowany.

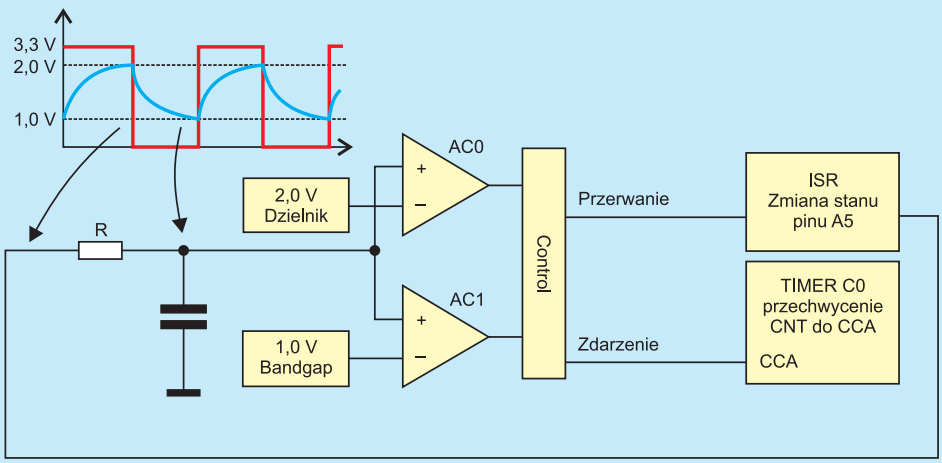
Istnieje jednak ryzyko, że układ nie wskoczy w rytm naprzemiennego ładowania i rozładowywania kondensatora. Może się tak zdarzyć np. po restarcie procesora. Wtedy kondensator naładuje się do napięcia zasilania lub rozładuje się do zera, a program będzie czekał w nieskończoność. Aby temu zapobiec, najprościej zastosować przerwanie od przepełnienia timera *TCC0_OVF_vect* – w jego procedurze znajduje się jedynie polecenie zamiany stanu pinu A5. Dzięki temu układ zawsze będzie działał.

W pętli głównej, musimy jedynie cyklicznie wyświetlać wynik. Jest to zadanie bardzo proste (wyświetlacz LCD omówiono w EP 2013/11). Jedyny problem to przeliczenie cykli zegarowych, zmierzonych przez timer, na pojemność w dowolnej jednostce (ja wybrałem mikrofarady) – musimy skalibrować nasz miernik. Jako kondensator wzorcowy użyłem zwyczajnego kondensatora elektrolitycznego 100 µF. Na wyświetlaczu uzyskałem wynik ok. 31000. Dlatego w kodzie programu, w linii oznaczonej numerem (1), wartość rejestru *TCC0.CCA* dzielę przez 31. Uzyskany w ten sposób wynik stanowi pojemność w mikrofaradach, ale pomnożoną przez 10, aby za pomocą funkcji *LcdDecComma* uzyskać jedno miejsce po przecinku. Dzięki takiemu dość pokrętnemu przeliczeniu, udało się uniknąć stosowania liczb zmiennoprzecinkowych typu float oraz funkcji printf, które przerażająco marnują zasoby procesora. Zachęcam do takiego kombinowania, gdyż przy odrobinie wprawy obliczenia z liczbami ułamkowymi nie będą stanowić problemu, a brak zmiennych float oznacza często wiele kilobajtów zaoszczędzonej pamięci procesora!

Dodatkowo, w każdym obiegu pętli głównej wpisujemy stan pinu A5 do rejestru wyjściowego *PORTX.OUT*, aby widzieć, kiedy kondensator jest ładowany, a kiedy rozładowywany. Można zaobserwować, że dioda mruga tym szybciej, im mniejsze kondensatory podłączamy do układu.

Miernik pojemności zbudowany tym sposobem jest w stanie mierzyć kondensatory o pojemności od 1 µF do 150 µF. Aby zwiększyć zakres pomiarowy, należy zmienić preskaler timera, by mierzyć mniejsze lub większe interwały czasowe. Czas ładowania zależy również od rezystora. Dobrym pomysłem jest zastosowanie kilku rezystorów o różnych wartościach, podłączonych do różnych pinów i napisanie algorytmu, który by automatycznie wybierał rezystor. Dokładność miernika zależy także od stabilności generatora sygnału zegarowego. My w tym odcinku wykorzystaliśmy wbudowany generator RC, a wiele lepszym pomysłem, lecz niewiele bardziej skomplikowanym, jest zastosowanie generatora kwarcowego (opisanego w EP 1/2014).

Dominik Leon Bieczyński
www.leon-instruments.pl



Rysunek 6. Schemat układu mierzącego pojemność i przebiegi w charakterystycznych miejscach

Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: ftp://ep.com.pl, user: 26526, pass: 841uhx54