DSP w praktyce

Generator PWM i interfejs SCI

W poprzednim numerze EP przedstawiony został przetwornik A/C w jaki wyposażone są mikrokontrolery czasu rzeczywistego Piccolo z zestawu startowego controlSTICK. Aby móc wykorzystać przynajmniej część z możliwości, jaki oferują te ciekawe układy należy się zapoznać jeszcze z układem generatora sygnału PWM, oraz żeby nawiązać komunikację ze światem zewnętrznym – z interfejsem szeregowym SCI.

Celem szczegółowego przedstawiania układów peryferyjnych: przetwornika A/C (poprzedni numer EP8/09), generatora sygnału ePWM oraz interfejsu szeregowego SCI, jest zbudowanie podstaw do uruchomienia nieskomplikowanego stanowiska do tworzenia wbudowanych aplikacji korzystających z algorytmów DSP. Na **rys. 1** przedstawiono schemat blokowy systemu będącego narzędziem umożliwiającym pisanie i testowanie algorytmów DSP uruchamianych w mikrokontrolerze czasu rzeczywistego TMS320F28027 z zestawu controlSTICK.

W kolejnych numerach EP przedstawiony zostanie sposób uruchomienia takiego stanowiska oraz kilka aplikacji wykorzystujących algorytmy cyfrowego przetwarzania sygnałów. W niniejszym artykule skupimy się na generatorze ePWM i układzie SCI.

Przed przystąpieniem do pracy z układem SCI należy najpierw do tego odpowiednio przygotować zestaw controlSTICK.

Aktualizacja konfiguracji FT2232D

Mikrokontroler z zestawu controlSTICK komunikuje się z komputerem PC za pomocą układu FT2232D. Jest to podwójny konwerter USB <=> UART/JTAG. Pierwszy kanał konwertera jest wykorzystywany jako programator/debuger JTAG mikrokontrolera, natomiast drugi jest na płytce zestawu podłączony do układu komunikacji szeregowej SCI (*Serial Communication Interface*). Nasuwa się zatem myśl, że można ów drugi kanał wykorzystać do celów budowanej aplikacji.

Niestety nowy zestaw controlSTICK nie oferuje możliwości skorzystania z tego kanału. Aby było to możliwe, należy zaktualizować pamięć EEPROM, w której zapisana jest konfiguracja układu mostu FT2232D. Procedura aktualizacji została opisana w temacie *Important -- C2000 Experimenter Kit „Dual RS232" Fix*, na forum dyskusyjnym dotyczącym mikrokontrolerów TMS320C2000 (*http://e2e.ti.com/forums/35.aspx*) na stronie internetowej Texas Instruments. Najpierw dokonujemy instalacji zgodnie z instrukcją dołączoną przez producenta do zestawu startowego, a dopiero następnie wykonujemy czynności opisane w wyżej wymienionym wątku.

Po zaktualizowaniu pamięci EEPROM w systemowym *Menedżerze urządzeń* pojawi się dodatkowy port szeregowy. Od tego momentu można we własnych aplikacjach używać układ SCI do komunikacji z komputerem przez standardowy port szeregowy.

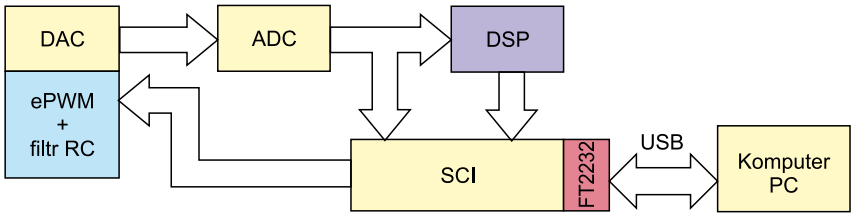
Port szeregowy SCI

Układ SCI (*Serial Communication Interface*) jest asynchronicznym portem szeregowym, odpowiednikiem portu UART w innych systemach mikroprocesorowych. Mikrokontroler TMS320F28027 ma wbudowane dwa układy transmisji szeregowej – SCIA oraz SCIB. Linie nadawcza SCITXD (GPIO29) oraz odbiorcza SCIRXD (GPIO28) układu SCIA zostały na płytce startowej controlSTICK podłączone do mostu FT2232D.

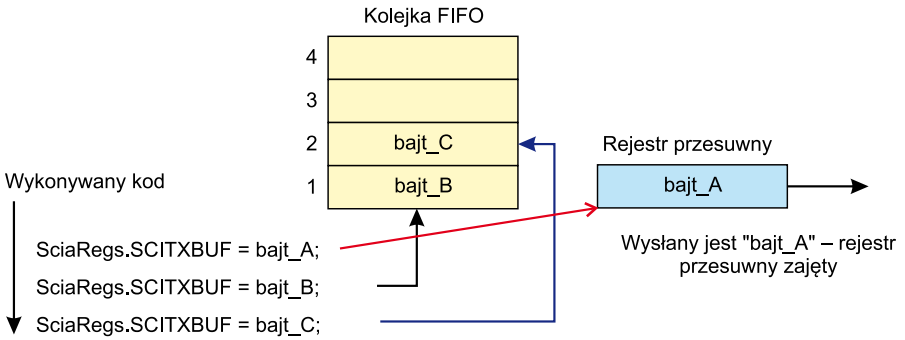
W porównaniu do standardowych portów UART, układy SCI zostały wyposażone w kilka dodatkowych funkcjonalności. Są to m. in. dodatkowe bufory w postaci dwóch, dla danych wysyłanych i odbieranych, 4 – poziomowych kolejek FIFO. Kolejki FIFO pozwalają maksymalnie wykorzystać przepustowość szeregowego kanału komunikacyjnego, przy niewielkim zaangażowaniu CPU, ponieważ można do układu SCI wysyłać kolejne porcje danych wtedy, kiedy inne dane są jeszcze w trakcie wysyłania. Odwrotna sytuacja dotyczy procesu odbierania informacji. Bufor odbiorczy (poszczególne segmenty kolejki FIFO) jest stopniowo zapełniany, zatem aplikacja może „hurtowo" odczytywać odebrane dane. Odbiorcze kolejki FIFO minimalizują także ryzyko utracanie danych przez nadpisanie.

Długość ramki danych może być konfigurowana, w celu wyjaśnienia zasady działania układu SCI z włączonymi kolejkami FIFO zakładamy konfigurację ramki danych na osiem bitów (bajt). Aplikacja przedstawiona w dalszej części artykułu również będzie używała ramki danych o długości ośmiu bitów.

Kolejne bity są wysyłane z wykorzystaniem rejestru przesuwnego. Jeżeli rejestr przesuwny wystawiający kolejne bity na linię TX jest w danym momencie pusty, to wtedy zapis do bufora nadawczego TXBUF powoduje w następnym cyklu zegarowym przepisanie jego wartości do rejestru przesuwnego. Z tego powodu pierwsze wysłanie bajta danych (jeżeli rejestr przesuwny jest pusty) do układu SCI nie powoduje zapisania elementu kolejki FIFO. Stąd też nie następuje tutaj zwiększenie licznika zajętych segmentów kolejki FIFO (bity TXFFST rejestru SCIFFTX). Dopiero, jeśli kolejny bajt zostanie wysłany do układu SCI, a poprzedni jeszcze nie zostanie w całości wysłany, ten pierwszy jest zapisywany w kolejce FIFO – ilustruje to **rys. 2**. Podsumowując: jeżeli w chwili zapisu bajta do SCI rejestr przesuwny był pusty to można od razu wysłać do SCI pięć kolejnych, przeznaczonych do wysłania danych. Taka sytuacja ma miejsce zawsze po uruchomieniu układu SCI oraz wtedy, kiedy układ komunikacji szeregowej zdąży wysłać



Rys. 1. Schemat blokowy systemu do testowania algorytmów DSP



Rys. 2. Zasada działania kolejek FIFO w układzie SCI

List.1. Program konfigurujący SCI do pracy z przerwaniami i inicjalizujący wysyłanie łańcucha znaków

```
void main(void)
{
        DeviceInit(); // Podstawowa konfiguracja mikrokontrolera
        InitSciaGpio(); // Konfiguracja wyprowadzeń do pracy z SCI

        EALLOW;  // Zezwolenie na dostęp do rejetrów
        // Przerwanie od RX obsługiwane przez sciaRxFifoIsr()
        PieVectTable.SCIRXINTA = &sciaRxFifoIsr;
        // Przerwanie od TX obsługiwane przez sciaTxFifoIsr()
        PieVectTable.SCITXINTA = &sciaTxFifoIsr;
        EDIS;   // Wyłączenie dostępu do rejestrów chronionych

        // Konfiguracja SCIA:
        SciaRegs.SCICCR.bit.SCICHAR = 7; // Dane 8 bitowe
        SciaRegs.SCICTL1.bit.TXENA = 1; // Linia TX włączona
        SciaRegs.SCICTL1.bit.RXENA = 1; // Linia RX włączona
        SciaRegs.SCICTL2.bit.TXINTENA = 1; // Przerwanie TX
        SciaRegs.SCICTL2.bit.RXBKINTENA = 1;       // Przerwanie RX

        // LSPCLK = 15MHz, prędkość 9600bodów, a więc:
        // SCIBAUD = 15MHz/(9600*8) - 1 = ok 194 = 0x00C2
        SciaRegs.SCIHBAUD = 0x0000;
        SciaRegs.SCILBAUD = 0x00C2;

        // Konfiguracja rejestru nadawczego bufora FIFO (SCIFFTX)
        SciaRegs.SCIFFTX.bit.SCIRST = 1;
        SciaRegs.SCIFFTX.bit.SCIFFENA = 1; // Włączenie FIFO TX
        SciaRegs.SCIFFTX.bit.TXFFIENA = 1; // Przerwanie od FIFO TX

        // Konfiguracja rejestru odbiorczego bufora FIFO (SCIFFRX)
        SciaRegs.SCIFFRX.bit.RXFFIENA = 1;
        SciaRegs.SCIFFRX.bit.RXFFIL = 4;

        SciaRegs.SCIFFCT.all = 0x0;
        SciaRegs.SCICTL1.all = 0x0023;  // SW Reset

        // Zerowanie kolejek FIFO (TX i RX)
        SciaRegs.SCIFFTX.bit.TXFIFOXRESET = 1;
        SciaRegs.SCIFFRX.bit.RXFIFORESET = 1;

        // Konfiguracja PIE
        PieCtrlRegs.PIECTRL.bit.ENPIE = 1;   // Włączenie PIE
        PieCtrlRegs.PIEIER9.bit.INTx1 = 1;     // INT9.1 (SCIRXINTA)
        PieCtrlRegs.PIEIER9.bit.INTx2 = 0;     // INT9.2 (SCITXINTA)
        IER = 0x100;     // Włącznie przerwań CPU
        EINT;

        ptr_msg = msg;
        PieCtrlRegs.PIEIER9.bit.INTx2=1; // Włączenie przerwania od TX

        while(1){}

}
```

zawartość kolejki FIFO, a rejestr przesuwny zostanie opróżniony.

Rodzina Piccolo rośnie w siłę!
Firma Texas Instruments znacznie poszerza swoją ofertę mikrokontrolerów z rodziny Piccolo. W sumie w najbliższym czasie dostępnych będzie aż 26 układów o różnych parametrach. Najbardziej rozbudowane mikrokontrolery będą wyposażone w pamięć Flash 128 kB, 20 kB pamięci RAM, 14 kanałowy PWM, oraz jednostkę CLA (*Control Law Accelerator*) – jest to 32 – bitowy, zmiennoprzecinkowy, procesor, który zwiększa możliwości rdzenia C28x o możność uruchomienia równoległych procesów. Aktualnie dostępnych jest sześć układów: TMS320F28035-PN, TMS320F28035-PAG, TMS320F28027-PT, TMS320F28027-DA, TMS320F28023-PT, TMS320F28023-DA.

Informacja, ile w danym momencie segmentów kolejki FIFO jest zajętych przez dane, jest zapisana, jak już było wspomniane, na bitach TXFFST w rejestrze SCIFFTX. Odczytując te bity można sprawdzać, czy kolejka jest pusta, czy zapełniona, a na podstawie uzyskanej informacji można podjąć decyzję o wysłaniu kolejnych bajtów danych do układu SCI.

Układ SCI w trybie z obsługą kolejek FIFO może generować przerwania w zależności od poziomu zapełnienia kolejek. Przerwania od linii nadawczej TX są konfigurowane za pomocą bitów TXFFIL w rejestrze SCIFFTX. Przerwanie zostanie zgłoszone, jeśli wartość odpowiadająca bitom informującym o zajętości kolejki FIFO (TXFFST) będzie taka sama, lub mniejsza, jak wartość bitów TXFFIL wymienionych powyżej. Należy pamiętać, że bity TXFFIL nie mogą mieć nadanej wartości większej od 4, ponieważ kolejka FIFO może być maksymalnie o długości czterech komórek.

Fragment kodu, którego zadaniem jest konfiguracja układu SCI do wysyłania i dobierania danych z obsługą przerwań i z wykorzystaniem kolejek FIFO, a następnie wysyłający łańcuch znaków, przedstawiono na **list. 1**. Wstępna konfiguracja mikrokontrolera do pracy odbywa się wraz z wywołaniem funkcji *DeviceInit()*, która jest wykorzystywana w każdym przykładowym projekcie dołączonym do zestawu controlSTICK. Ustawienie parametrów wyprowadzeń do współpracy z układem SCI jest zadaniem funkcji *InitSciaGpio()*, jej ciało zamieszczono na **list. 2**. Po przypisaniu funkcjom *sciaRxFifoIsr()* oraz *sciaTxFifoIsr()* roli obsługi prze-

Każdy moduł (kanał) generatora ePWM jest podzielony na osiem submodułów:
Submoduł podstawy czasu, Time – Base (TB) Submodule – Definiuje częstotliwość generowanego przebiegu oraz tryb zliczania (w górę, w dół, tryb góra/dół). Zapewnia synchronizację programową lub sprzętową, np. z pozostałymi modułami ePWM.
Submoduł porównawczy, Counter – Compare (CC) Submodule – Zadaniem tego submodułu jest nieustanne porównywanie stanu licznika generatora z wartościami zapisanymi w rejestrach porównawczych CMPA i CMPB. Jeżeli wartość licznika będzie równa zawartości któregoś z tych rejestrów, to zostanie o tym poinformowany submoduł AQ. W związku z tym, submoduł porównawczy może służyć do ustalania współczynnika wypełnienia generowanego przebiegu PWM.
Submoduł akcji, Action – Qualifier (AQ) Submodule – Decyduje jakie akcje, ustawienie stanu wysokiego lub niskiego na wyjściach modułu, zostaną podjęte na podstawie przychodzących zdarzeń (np. z *Time – Base lub Counter – Compare*). Bezpośrednio kształtuje generowany sygnał.
Generator czasu martwego, Dead – Band Generator (DB) Submodule – Pozwala wprowadzić opóźnienia pomiędzy zboczami sygnałów na wyjściach EPWMxA i EPWMxB. Wykorzystywany w aplikacjach sterujących dużymi prądami.
Submoduł modulacji PWM, PWM – Chopper (PC) Submodule – Umożliwia modulację przebiegiem PWM sygnału o częstotliwości większej od generowanego przebiegu PWM. Wykorzystywany np. w impulsowych sterownikach transformatorów.
Submoduł zabezpieczający, Trip – Zone (TZ) Submodule – Konfiguracja tego submodułu pozwala ustalić, jak mają się zachowywać wyjścia w sytuacjach krytycznych.
Submoduł zarządzania zdarzeniami, Event – Trigger (ET) Submodule – Zarządza generowaniem przerwań, w zależności od skonfigurowania.
Submoduł komparatora cyfrowego, Digital Compare (DC) Submodule – Na podstawie wyniku porównania wartości liczbowych może głaszać zdarzenia dla submodułu zabezpieczającego, podstawy czasu, zarządzania zdarzeniami lub też dla przetwornika A/C.

rwań następuje konfiguracja układu SCI oraz włączenie bloku multipleksera przerwań od układów peryferyjnych PIE (*Peripheral Interrupt Expansion*). Linie: nadawcza TX i odbiorcza RX są wewnętrznie przyporządkowane do przerwań odpowiednio INT9.2 INT9.1. Przerwanie od linii odbiorczej RX jest w tym miejscu włączane, w odróżnieniu od przerwania linii TX. Póki co nie chcemy jeszcze nic wysyłać, stąd też to przerwanie pozostaje wyłączone. Wyczerpujące informacje na temat przerwań w mikrokontrolerze TMS320F27028 są zamieszczone w nocie katalogowej *„TMS320F2802x Piccolo System Control and Interrupts - SPRUFN3B"*.

Gdy wszystko jest skonfigurowane, program z list. 1 inicjuje wysyłanie komunikatu przez port szeregowy. Adres początku tablicy z łańcuchem znaków, przeznaczonym do wysłania (tablica *msg[]*) jest kopiowany do wskaźnika *ptr_msg*, po czym jest włączana obsługa przerwania TX. Wskaźnik *ptr_msg* jest dalej wykorzystywany przez funkcję obsługi przerwania od części nadawczej TX układu SCI, którą przedstawiono na **list. 3**.

Przerwanie od linii TX zostaje wygenerowane, a więc następuje wywołanie funkcji *sciaTxFifoIsr()*, zawsze, kiedy kolejka FIFO jest pusta. Zadaniem kodu funkcji *sciaTxFifoIsr()*, jest zapełnienie całej kolejki danymi, chyba, że napotkany zostanie znak końca łańcucha '\0'. Jeśli ten znak zostanie znaleziony w wysyłanym łańcuchu, to wyłączane jest przerwanie od linii odbiorczej TX, ponieważ cały komunikat został wysłany (ściślej – jego ostatnie cztery bajty zostały załadowane do kolejki FIFO).

Układ ePWM

Mikrokontrolery należące do rodziny Piccolo zostały wyposażone w bardzo rozbudowany generator sygnału PWM. Z jego złożoności wynika możliwość generowania wielu skomplikowanych sygnałów. Przykładowymi aplikacjami mogą być: sterowanie wielofazowymi silnikami oraz układy przetwornic. Cały proces sterowania odbywa się przy tym przy minimalnym zaangażowaniu rdzenia.

Charakterystyczną cechą generatora ePWM jest wielopoziomowa konstrukcja modułowa. Każdy z modułów ePWM reprezentuje jeden niezależny kanał generatora z dwoma wyjściami: EPWMxA i EPWMxB. Układ TMS320F27028 ma wbudowane cztery takie moduły, czyli w sumie dysponuje ośmioma wyjściami PWM – EPWM1A, EPWM1B, EPWM2A, EPWM2B, itd. Dodatkowo moduły są podzielone na osiem submodułów, z których każdy jest odpowiedzialny za konfigurację innej funkcjonalności generatora. Zestaw startowy controlSTICK ma na złącze szpilkowe wyprowadzone wszystkie osiem wyjść, przy czym EPWM1A jest filtrowane przez pasywny dolnoprzepustowy filtr RC o częstotliwości granicznej około 34 kHz.

Noty katalogowe wyczerpująco opisujące przedstawione układy peryferyjne:
- generator ePWM – *„TMS320x2802x, 2803x Piccolo Enhanced Pulse Width Modulator (ePWM) Module – SPRUGE9B"*
- układ komunikacji szeregowej SCI – *„TMS320x2802x, 2803x Piccolo Serial Communications Interface (SCI) – SPRUGH1A"*
- multiplekser przerwań PIE – *„TMS320F2802x Piccolo System Control and Interrupts – SPRUFN3B"*

List. 2. Funkcja konfigurująca wyprowadzenia GPIO28 i GPIO29 do pracy z układem SCI

```
void InitSciaGpio()
{
        EALLOW;  // Zezwolenie na dostęp do rejetrów
        GpioCtrlRegs.GPAPUD.bit.GPIO28 = 0;        // Włączenie pull-up
(SCIRXDA)
        GpioCtrlRegs.GPAPUD.bit.GPIO29 = 1;        // Wyłączenie pull-up
(SCITXDA)
        GpioCtrlRegs.GPAQSEL2.bit.GPIO28 = 3;      // GPIO28 (SCIRXDA) we
asynchr.
        GpioCtrlRegs.GPAMUX2.bit.GPIO28 = 1;       // GPIO28 jako SCIRXDA
        GpioCtrlRegs.GPAMUX2.bit.GPIO29 = 1;       // GPIO29 jako SCITXDA
        EDIS;    // Wyłączenie dostępu do rejestrów chronionych
}
```

List. 3. Funkcja obsługi przerwania od nadajnika TX układu SCI

```
interrupt void sciaTxFifoIsr(void)
{
        // Wykonuj dopóki kolejka nie zapełniona i nie koniec komunikatu:
        while(SciaRegs.SCIFFTX.bit.TXFFST < 4 && *(ptr_msg+i) != ,\0')
        {
                SciaRegs.SCITXBUF = *(ptr_msg+i);
                i++;
        }

        // Jeśli koniec komunikatu, zeruj licznik i oraz wyłącz
        // przerwanie od TX:
        if(*(ptr_msg+i) == ,\0')
        {
                i = 0;
                PieCtrlRegs.PIEIER9.bit.INTx2 = 0;
        }

        SciaRegs.SCIFFTX.bit.TXFFINTCLR = 1;       // Czyść flagę przerwanaia
        // Grupa 9 gotowa na następne przerwanie:
        PieCtrlRegs.PIEACK.all |= 0x100;

}
```

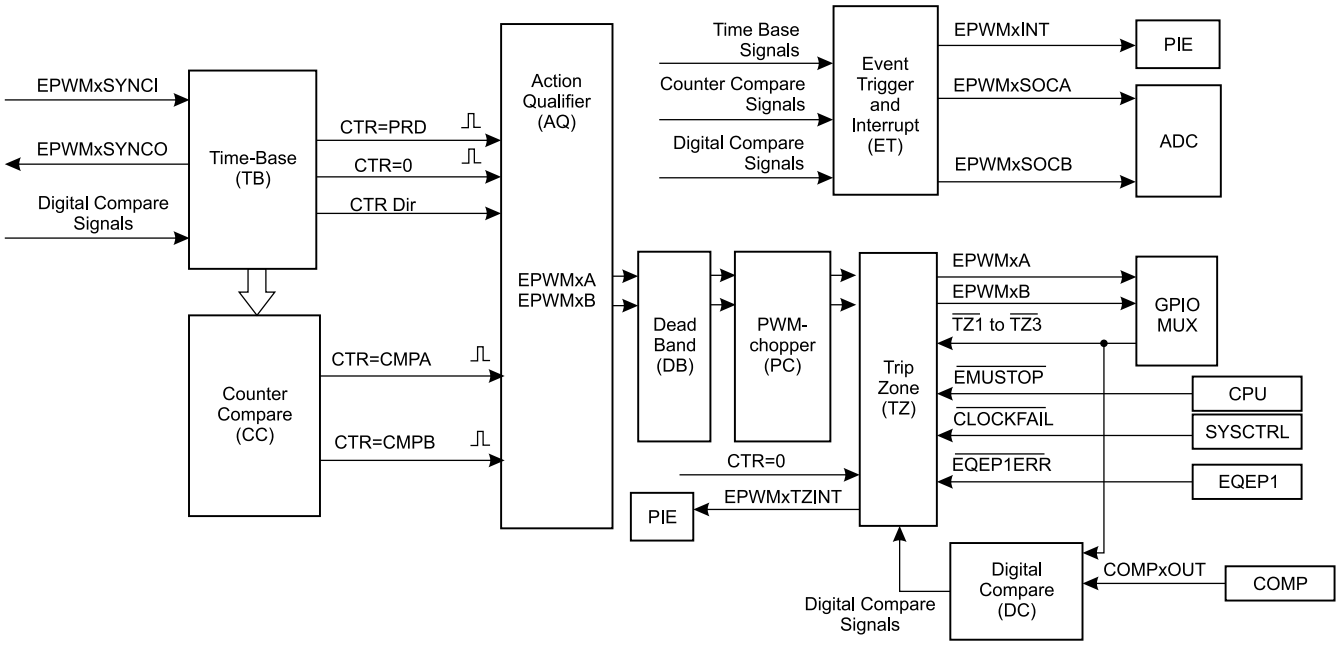
List. 4. Konfiguracja układu ePWM do generowania pojedynczego przebiegu o wypełnieniu 25%

```
// Konfiguracja submodułu podstawy czasu (Time-Base)
EPwm1Regs.TBPRD = 5999;                          // Licznik generatora zlicza
                                                    do 5999
EPwm1Regs.TBPHS.all = 0;                         // Sychronizacja fazowa
                                                    wyłączona
EPwm1Regs.TBCTR = 0;                             // Zerowanie licznika
EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;           // Rejestry-cienie wł. dla
                                                    okresu
EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;       // Zliczanie w górę
EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;          // Brak sychronizacji
EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;  // Wyj. synchr. wyłączone
EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;         // TBCLK = SYSCLK = 60 MHz
EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;

// Konfiguracja submodułu porównawczego
EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;      // Rejestry-cienie wł. dla
                                                    CMPA
EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;    // aktualizacja, gdy CTR=0
EPwm1Regs.CMPA.half.CMPA = 1500;                 // Wypełnienie 25%

// Konfiguracja submodułu akcji
EPwm1Regs.AQCTLA.bit.ZRO = AQ_SET;               // Wy PWM1A=1, gdy CTR=0
EPwm1Regs.AQCTLA.bit.CAU = AQ_CLEAR;             // Wy PWM1A=0, gdy CTR=CMPA
```

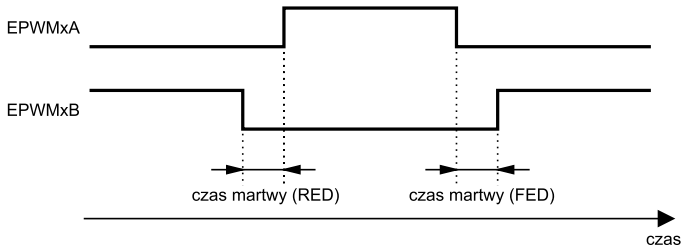
Modułowa konstrukcja umożliwia szybkie zrozumienie jak układ ePWM działa, a ponadto kod konfigurujący urządzenie do pracy jest klarowny i stosunkowo łatwy do napisania – biorąc pod uwagę mnogość opcji, jakie można ustawiać. Schemat blokowy, ilustrujący wzajemne zależności i położenie submodułów, przedstawiono na **rys. 3**, natomiast krótkie omówienie każdego z submodułów zamieszczono w ramce. Bardziej istotne cechy submodułów są szczegółowiej omówione w dalszej części artykułu przy okazji prezentacji sposobu konfiguracji układu ePWM do generowania różnych przebiegów.

Rejestry – cienie (*Shadow register*)

Parametry przebiegu PWM powinny być zmieniane tylko w określonym czasie, innymi słowy wprowadzane zmiany powinny być synchronizowane z generowanym przebiegiem. Jako przykład można tutaj podać zmianę wartości licznika generatora (*Pe-*



Rys. 3. Modułowa budowa układu peryferyjnego ePWM



Rys. 4. Przebieg komplementarny z czasem martwym

riod). Jeżeli ma występować synchronizacja pomiędzy sygnałem PWM, a wprowadzaną zmianą okresu, to należy wykorzystać mechanizm rejestrów – cieni (*Shadow Register*). Włączenie mechanizmu rejestrów – cieni powoduje automatyczne wykorzystanie przez mikrokontroler dwóch rejestrów: aktywnego i rejestru – cienia. Zmiana zawartości rejestru aktywnego (możliwa w trybie pracy w wyłączonym omawianym mechanizmem) wywołuje natychmiastową zmianę w parametrach pracy układów peryferyjnych. W trybie pracy w włączonymi rejestrami – cieniami miejsce rejestru aktywnego zajmuje rejestr – cień o takiej samej nazwie. Aktualizacja rejestru aktywnego, czyli zapis do niego wartości z rejestru – cienia, jest wykonywana całkowicie automatycznie w określonej chwili, np. gdy wartość licznika generatora wynosi 0.

Podstawowa konfiguracja ePWM

Podstawowe parametry sygnału PWM to jego okres i wypełnienie. Okres jest definiowany za pomocą submodułu podstawy czasu (*Time – Base Submodule*) poprzez określenie źródła sygnału taktującego licznik generatora, oraz przez wartość do jakiej (od jakiej) ten licznik będzie zliczał (*Period*).

Przykład ilustrujący sposób konfiguracji układu ePWM do wytwarzania zwykłego przebiegu o zmiennym wypełnieniu na wyjściu EPWM1A przedstawiono na **list. 4**. Generator będzie taktowany z częstotliwością zegara systemowego (domyślnie 60 MHz), natomiast licznik będzie zliczał w górę do wartości 5999. Częstotliwość generowanego przebiegu będzie zatem wynosić:

$$f_{PWM} = \frac{TBCLK}{TBPRD+1} = \frac{60[MHz]}{5999+1} = 10[kHz]$$

Licznik generatora jest zerowany, a sygnały synchronizacji zostają wyłączone, Rejestry – cienie dla okresu licznika (*Period*) zostają włączane.

Submoduł porównawczy (*Coutner-Compare Submodule*) wymaga przede wszystkim ustawienia wartości rejestru porównawczego CMPA, który jest podczas normalnej pracy na bieżąco porównywany z aktualną wartością licznika generatora. Gdy obie wartości są sobie równe, to wtedy jest zgłaszane zdarzenie do submodułu akcji (*Action-Quailfier Submodule*). Również dla rejestru porównawczego CMPA jest włączany mechanizm rejestrów – cieni, aktualizacja wartości będzie następować przejściu licznika przez 0, co jest ustawiane poprzez zapis bitów LOADAMODE w rejestrze CMPCTL.

Współczynnik wypełnienia generowanego przebiegu ma wynosić 25%, stąd też do rejestru porównawczego CMPA wpisywana jest wartość (5999+1)×25%=1500. Komentarza w tym miejscu może wymagać jeszcze sposób zapisu rejestru CMPA. Definicja tego rejestru zapewnia w istocie dostęp do dwóch rejestrów: CMPA oraz CMPAHR. Ostatni jest związany z generowaniem sygnału o zwiększonej rozdzielczości (*HRPWM*). Składnia z **list. 4** powoduje zapis tylko standardowego rejestru porównawczego CMPA.

Gdy submoduł podstawy czasu i submoduł porównawczy są poprawnie skonfigurowane, pozostaje jeszcze ustalić, w jaki sposób sygnał wyjściowy ma być formowany. Odpowiada za to submoduł akcji. Wyjście PWM1A generatora ePWM będzie ustawiane w stan wysoki w chwili, gdy licznik generatora będzie miał wartość 0, natomiast w stan niski – kiedy wartość rejestru porównawczego CMPA będzie równa wartości licznika.

Czas martwy

Aplikacje, w których przełączane są duże prądy za pomocą układów mostkowych, wymagają takiego sterowania tranzystorami, aby w tym samym czasie obydwa nie miały otwartego kanału. Jeśli zostanie wykorzystana zwykła para komplementarna sygnałów, to wtenczas, ze względu na ładunek, jaki gromadzi się w bramce tranzystorów w chwilach występowania zboczy w sygnałach sterujących przewodzić będą oba tranzystory. Aby temu zjawisku zapobiec wprowadza się zwłokę – czas martwy – pomiędzy sygnały komplementarne, tak, że zbocza (narastające i opadające) nie występują w tym samym czasie, ale w pewnych odstępach. Przykład przebiegów komplementarnych z czasem martwym (dead-band) przedstawiono na **rys. 4**. Czas martwy do przebiegów komplementarnych może być wprowadzany przez submoduł *Dead-band*, którego charakteryzuje duża elastyczność ustawianych parametrów.

Fragment kodu, który konfiguruje układ ePWM do pracy z czasem martwym przed-

List. 5. Konfiguracja submodułu generatora czasu martwego

```
// Konfiguracja submodułu czasu martwego (Dead-Band)
EPwm1Regs.DBCTL.bit.IN_MODE = DBA_ALL;              // Na podstawie EPWM1A będą obliczane opóźnienia
EPwm1Regs.DBCTL.bit.OUT_MODE = DB_FULL_ENABLE;      // Oba opóźnienia (FED i RED) włączone
EPwm1Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC;           // Wyj komplementane z aktywnym stanem wys. (AHC)
EPwm1Regs.DBRED = 60;                               // Opóźnienie dla zbocza narastającego (RED) 1 us
EPwm1Regs.DBFED = 60;                               // Opóźnienie dla zbocza opadającego (FED) 1 us
```

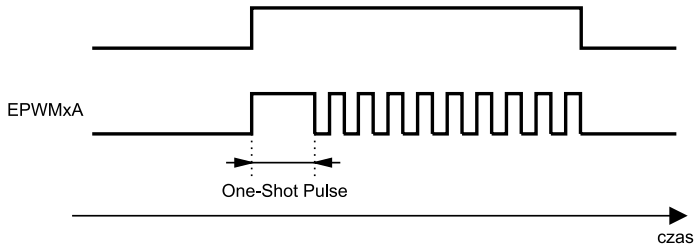
stawiono na **list. 5**. Wykorzystywany jest tryb pracy z sygnałami komplementarnymi, opóźnienia są obliczane na podstawie przebiegu z wyjścia EPWM1A. Wartość opóźnienia jest niezależnie regulowana przez dwa rejestry DBRED oraz DBFED, odpowiednio dla zbocza narastającego RED (*Rising Edge Delayed*) i opadającego (*Falling Edge Delayed*) – oba parametry zostały zaznaczone na **rys. 4**. Jeśli częstotliwość taktująca układ ePWM wynosi tyle ile zegar systemowy (60 MHz) to wartość 60 wpisana tych rejestrów wywołuje opóźnienia 1 µs.

Modulacja sygnałem PWM

Sterowniki transformatorów impulsowych wymagają sterowania przebiegiem, który jest zmodulowany sygnałem PWM. Przykład takiego przebiegu zamieszczono na **rys. 5**. Przedstawiony na rysunku efekt można osiągnąć z pomocą submodułu modulacji PWM (*PWM – Chopper*). Za sterowanie pracą tego submodułu odpowiada tylko jeden rejestr PCCTL, fragment kodu z przykładową konfiguracją

List. 6. Konfiguracja submodułu *PWM – Chopper*

```
EPwm1Regs.PCCTL.bit.CHPDUTY = CHP4_8TH;     // Wypełnienie 50%
EPwm1Regs.PCCTL.bit.CHPFREQ = CHP_DIV2;     // Częstotliwość 3.75 MHz
EPwm1Regs.PCCTL.bit.OSHTWTH = 6;            // t = 1/(7.5 MHz) * 6 = 800 ns
EPwm1Regs.PCCTL.bit.CHPEN = CHP_ENABLE;
```



Rys. 5. Przykład przebiegu sygnału generowanego, gdy submoduł *PWM – Chopper* jest włączony

zamieszono na **list. 6**. Oprócz ustawienia podstawowych parametrów sygnału (wypełnienia, częstotliwości) włączana jest tutaj opcja dodawania jednego dłuższego impulsu (*One – Shot Pusle*), został on zaznaczony na **rys. 5**.

Częstotliwość modulowanego sygnału może wynosić maksymalnie tyle ile zegar systemowy podzielony przez osiem, czyli maksymalnie 7,5 MHz. Wynika to z zastosowania na wejściu omawianego submodułu sprzętowego dzielnika częstotliwości zegara systemowego.

Krzysztof Paprocki
paprocki.krzysztof@gmail.com

R E K L A M A