DSP w praktyce

Konfiguracja przetwornika A/C w TMS320F28027

Pomimo że technika DSP jest dziedziną czysto cyfrową, to procesory sygnałowe pracują na danych pochodzących często z przetworników A/C. Wynik obliczeń też nie jest sztuką dla sztuki, ale jest uzyskiwany w celu dalszego wykorzystania, np. w przetworniku C/A lub do sterowania generatorami PWM. W artykule omówiono przetwornik A/C wbudowany w mikrokontroler czasu rzeczywistego TMS320F28027 (Piccolo).

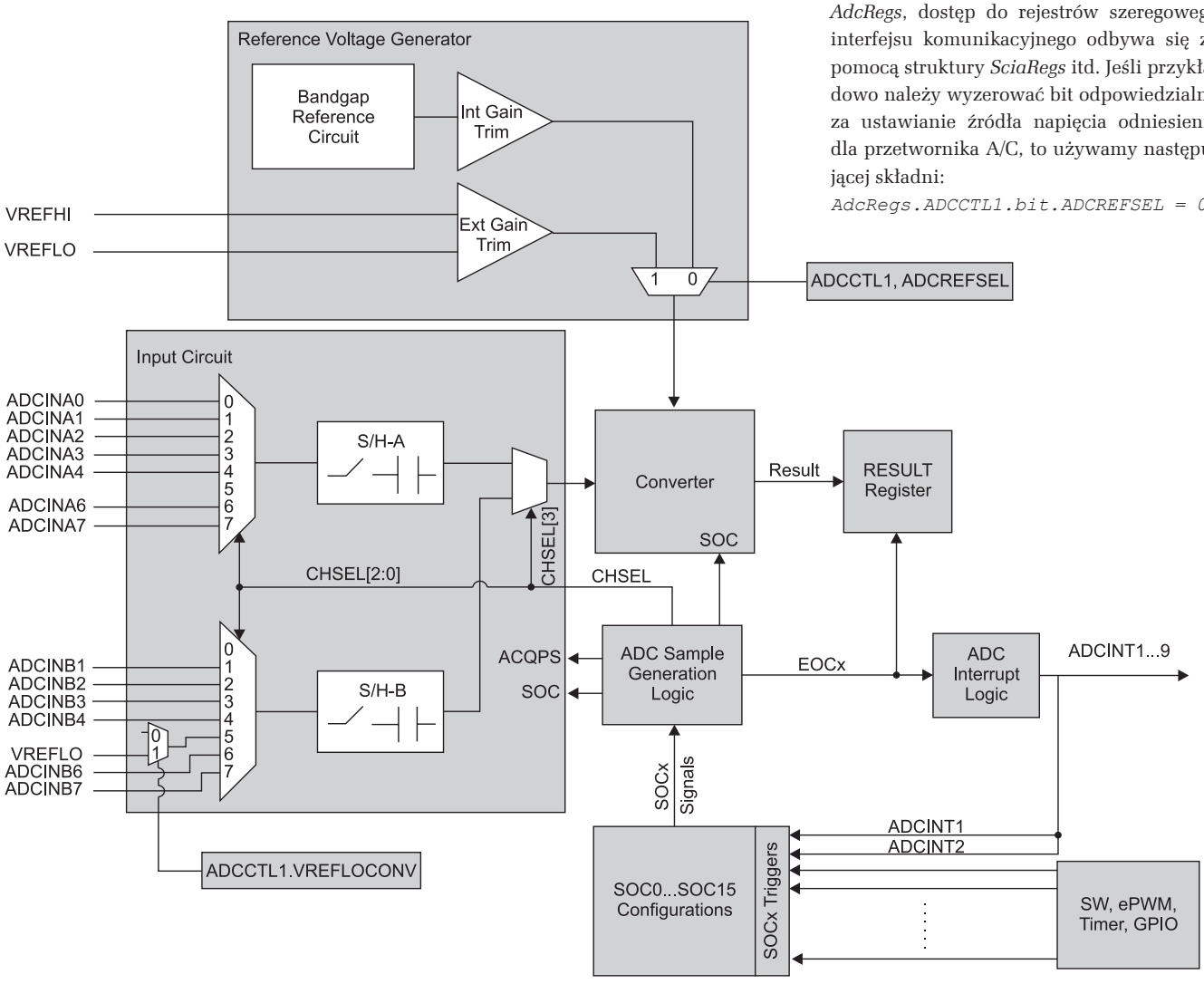
Wiedza o tym, jak konfigurować do pracy przetwornik A/C, jest niezbędna do rozpoczęcia przygody z DSP. Aby móc przetwarzać cyfrowo dane, najpierw trzeba je uzyskać, a następnie umieć zrobić użytek z wyniku obliczeń. Z tego powodu w niniejszym artykule zajmiemy się przedstawieniem przetwornika A/C wbudowanego w mikrokontroler TMS320F28027. Umiejętność wykorzystywania przetwornika A/C umożliwi naukę, a później testowanie w praktyce algorytmów DSP.

Dostęp do rejestrów mikrokontrolera

Przed przystąpieniem do omawiania przetwornika A/C zostanie przedstawiony sposób zapisu i odczytu rejestrów układu TMS320F28027.

W kompilatorze języka C, w który wyposażone jest *Code Composer Studio*, wszystkie rejestry urządzeń peryferyjnych mikrokontrolera TMS320F28027 reprezentowane są przez odpowiednie struktury. Nazwa struktury reprezentującej dane peryferie jest zbudowana według szablonu *układ_peryferyjny-Regs*. Stosując tę zasadę, większość rejestrów przetwornika A/C zawiera się w strukturze *AdcRegs*, dostęp do rejestrów szeregowego interfejsu komunikacyjnego odbywa się za pomocą struktury *SciaRegs* itd. Jeśli przykładowo należy wyzerować bit odpowiedzialny za ustawianie źródła napięcia odniesienia dla przetwornika A/C, to używamy następującej składni:

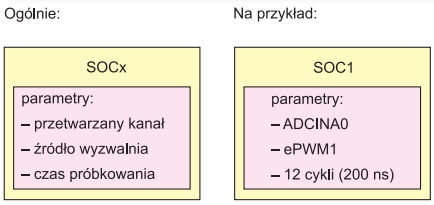
```
AdcRegs.ADCCTL1.bit.ADCREFSEL = 0;
```



Rys. 1. Schemat blokowy przetwornika A/C, w który wyposażone są mikrokontrolery TMS320F28027

List. 1. Włączenie zasilania dla elementów wbudowanego przetwornika A/C

```
EALLOW;                                 // Zezwolenie na dostęp do rejestrów kalibracyjnych
AdcRegs.ADCCTL1.bit.ADCREFSEL   = 0;    // Wybó wewn. źródła napięcia odniesienia
AdcRegs.ADCCTL1.bit.ADCBGPWD    = 1;    // Włączenie zasilania dla poszczególnych
AdcRegs.ADCCTL1.bit.ADCREFPWD   = 1;    // bloków
AdcRegs.ADCCTL1.bit.ADCPWDN     = 1;
AdcRegs.ADCCTL1.bit.ADCENABLE   = 1;    // Włączenie ADC
asm(„ RPT#100 || NOP”)                  // Opóźnienie większe niż 1 ms
```



Rys. 2. Budowa bloku SOC (*Start-Of-Converions*)

ADCCTL1 jest nazwą rejestru, a *ADCREFSEL* nazwą konkretnego bitu.

Budowa przetwornika A/C

Schemat blokowy przetwornika A/C wbudowanego w mikrokontroler TMS320F28027 przedstawiono na **rys. 1**. Trzynaście dostępnych (wyprowadzonych na zewnątrz układu) kanałów jest podzielonych na dwie grupy – A i B. Każda grupa ma swój układ próbkująco-pamiętający i dlatego dwa kanały z różnych grup mogą być **próbkowane** równocześnie. Nie można tutaj napisać, że będą **przetwarzane** równocześnie, ponieważ przetwornik A/C ma tylko jeden układ do konwersji pobranej próbki. Kanały mogą być próbkowane w tym samym czasie, natomiast proces konwersji jest przeprowadzany zawsze w pierwszej kolejności dla kanału z grupy A, a następnie dla kanału z grupy B.

Do przetwornika A/C zaimplementowanego w mikrokontrolerze przypisane są 23 rejestry. Wszystkie zostały dokładnie omówione w dokumencie *SPRUGE5A* dostępnym na stronie internetowej firmy TI. W artykule zostaną przybliżone nieco tylko te rejestry, które będą wykorzystywane w danej konfiguracji przetwornika A/C.

Rejestrów przechowujących wynik pomiaru jest 16, więc nawet więcej niż kanałów. Wydaje się to dziwne i na wyrost, ale tak nie jest. Dlaczego przetwornik A/C wyposażono w 16 rejestrów przechowujących wyniki pomiarów zostało wyjaśnione podczas omawiania bloków SOC.

Zasilanie i sygnał zegarowy

Sygnał zegarowy przetwornika A/C ma tę samą częstotliwość co systemowy sygnał zegarowy. Mikrokontroler TMS320F28027 może pracować z maksymalną częstotliwością wynoszącą 60 MHz. Próbkowanie mierzonego napięcia trwa przez minimum 7 cykli zegarowych, a proces konwersji wymaga 13 cykli. Dodając do siebie obydwie liczby, otrzymamy 20 cykli, co przy częstotliwości taktowania wynoszącej 60 MHz daje około 333 ns i to jest minimalny czas, jaki jest wymagany do przetworzenia (próbkowania i konwersji) jednej próbki.

Przetwornik A/C wymaga taktowania sygnałem zegarowym. Bez tego nie ma możliwości operacji na jego rejestrach. Sygnał taktujący włącza się przez ustawienie bitu *ADCENCLK* w rejestrze *PCKLCR0*, a realizuje to linia kodu: `SysCtrlRegs.PCLKCR0.bit.ADCENCLK = 1;`

Przed rozpoczęciem konfiguracji przetwornika należy jeszcze włączyć zasilanie wszystkich jego bloków i wybrać źródło napięcia odniesienia. Przedstawione zadania realizuje fragment programu z **list. 1**. Pierwszą czynnością jest wybranie źródła napięcia odniesienia, po czym należ włączyć jego zasilanie. Dalej włączane jest zasilanie pozostałych bloków przetwornika A/C i uruchamiany proces konwersji (bit *ADCENABLE* rejestru *ADCCTL1*). Po tych czynnościach należy odczekać minimum 1 ms. Nieco większą zwłokę zapewnia wstawka asemblerowa wykonująca 100 pustych cykli.

Start-of-converions

Konwersja w omawianym przetworniku bazuje na wykorzystaniu tzw. bloków SOC (*Start-Of-Converions*). Każdy blok SOC ma trzy konfigurowalne parametry: kanał, jaki będzie przetwarzany, źródło wyzwalania oraz szerokość okna próbkowania (lub prościej – czas próbkowania). Obrazowo w sposób uproszczony, ale dzięki temu przejrzysty, budowę takiego bloku przedstawia **rys. 2**.

Bloków SOC jest 16 (od SOC0 do SOC15), a wymienione wyżej parametry mogą być dowolnie konfigurowalne. Oznacza to na przykład, że kilka różnych SOC może się zajmować przetwarzaniem tego samego kanału. Jak wynika z powyższego, blok SOC został stworzony po to, aby można było pomiar każdego kanału konfigurować całkowicie niezależnie.

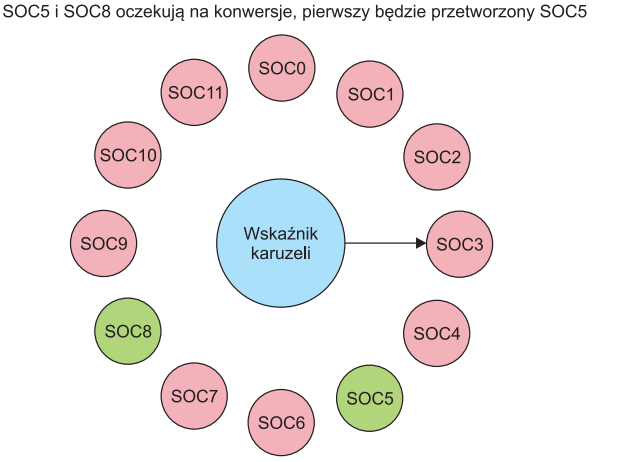
Fragment programu, odpowiedzialny za wstępną konfigurację bloków SOC0, SOC1 i SOC2, do obsługi odpowiednio kanałów A0, A1 i A2 przedstawiono na **list. 2**. Zapisywane są rejestry *ADCSOCxCTL*, a ściślej

List. 2. Wstępna konfiguracja bloków SOC0, SOC1, SOC2 do obsługi kanałów A0, A1, A2

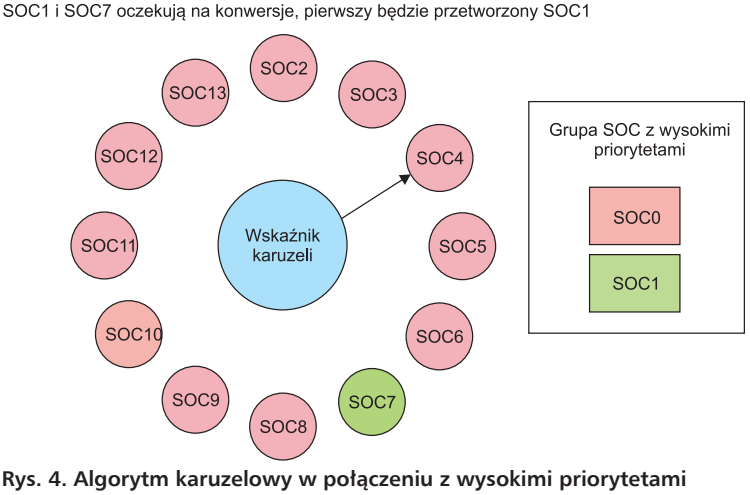
```
AdcRegs.ADCSOC0CTL.bit.CHSEL= 0;      // Blok SOC0 wywołuje konwersję kanału A0
AdcRegs.ADCSOC1CTL.bit.CHSEL= 1;      // Blok SOC1 -> A1
AdcRegs.ADCSOC2CTL.bit.CHSEL= 2;      // Blok SOC2 -> A2
```

List. 3. Różny czas próbkowania dla różnych bloków SOC

```
AdcRegs.ADCSOC0CTL.bit.ACQPS = 6;     // Około 117 ns
AdcRegs.ADCSOC1CTL.bit.ACQPS = 20;    // Około 566 ns
AdcRegs.ADCSOC2CTL.bit.ACQPS = 35;    // 800 ns
AdcRegs.ADCSOC4CTL.bit.ACQPS = 40;    // Około 883 ns
AdcRegs.ADCSOC7CTL.bit.ACQPS = 64;    // Około 1283 ns
```



Rys. 3. Zasada działania algorytmu karuzelowego priorytetów (*Round Robin Priority*)



Rys. 4. Algorytm karuzelowy w połączeniu z wysokimi priorytetami (*High priority*)

List. 4. „Ręczna" kalibracja przetwornika A/C w trybie debugowania
```
EALLOW;
SysCtrlRegs.PCLKCR0.bit.ADCENCLK = 1;      // Załączenie zegara ADC
(*Device_cal)();                           // Automatyczna kalibracja producenta (dane z OTP)
SysCtrlRegs.PCLKCR0.bit.ADCENCLK = 0;      // Przywrócenie oryginalnego zegara ADC
EDIS;
```

cztery bity *CHSEL*. Mamy tutaj do czynienia z przykładem pomiaru sekwencyjnego, jeśli ma się odbywać pomiar jednoczesny dwóch kanałów, to znaczenie bitów *CHSEL* jest nieco inne. Zostało to omówione w dalszej części artykułu.

Każdy blok SOC ma przyporządkowany rejestr, w którym umieszczany jest wynik pomiaru. W odróżnieniu od wszystkich pozostałych rejestrów przetwornika A/C, dostęp do rejestrów z wynikami nie odbywa się za pomocą struktury *AdcRegs*, lecz *AdcResult*. Jeśli odczytany ma być wynik przetwarzania bloku SOC4, to należy w programie umieścić następującą linię kodu: `wynik = AdcResult.ADCRESULT4;`

List. 5. Konfiguracja sygnałów wyzwalania dla bloków SOC przetwornika A/C – praca ciągła
```
// Zdarzenie EOC (End-Of-Conversion)
// bloku EOC10 wywoła przerwanie ADCInterrupt1:
AdcRegs.INTSEL1N2.bit.INT1SEL = 10;

// Przetwarzanie w  trybie ciągłym:
AdcRegs.INTSEL1N2.bit.INT1CONT = 1;

// Włączenie przerwania ADCinterrupt1:
AdcRegs.INTSEL1N2.bit.INT1E = 1;

// Przerwanie ADCInterrupt1 wywoła SOC10, itd.
AdcRegs.ADCINTSOCSEL2.bit.SOC10 = 1;
AdcRegs.ADCINTSOCSEL2.bit.SOC11 = 1;
AdcRegs.ADCINTSOCSEL2.bit.SOC12 = 1;
AdcRegs.ADCINTSOCSEL2.bit.SOC14 = 1;
AdcRegs.ADCINTSOCSEL2.bit.SOC15 = 1;

// Konwersja kanału 10, itd.
AdcRegs.ADCSOC10CTL.bit.CHSEL= 10;
AdcRegs.ADCSOC11CTL.bit.CHSEL= 11;
AdcRegs.ADCSOC12CTL.bit.CHSEL= 12;
AdcRegs.ADCSOC14CTL.bit.CHSEL= 14;
AdcRegs.ADCSOC15CTL.bit.CHSEL= 15;
```

Czas próbkowania

Najkrótszy możliwy czas próbkowania wynosi 7 cykli zegarowych sygnału taktującego przetwornik A/C. Nie zawsze jednak krótkie czasy próbkowania są pożądane. Zdarza się, że wymagane jest stosowanie dłuższych czasów. Do sterowania czasem próbkowania danego bloku SOC służy sześć najmłodszych bitów (*ACQPS*) rejestru *ADCSOCxCTL*. Należy jedynie pamiętać, że nie można tym bitom nadać wartości z przedziału od 0 do 5 (włącznie), ponieważ jak już było wspomniane, minimalny czas próbkowania wynosi 7 cykli zegarowych, co odpowiada wartości 6 nadanej bitom *ACQPS*.

Jak nietrudno się domyślić, maksymalna długość okna próbkowania wynosi 64 cykle zegarowe przetwornika A/C. Jeżeli przykładowo wymagana jest obsługa pięciu bloków SOC, a każdy ma mieć przypisany inny czas próbkowania, to konfiguracja bitów *ACQPS* może wyglądać jak ta umieszczona na **list. 3**. Każdy blok SOC ma indywidualnie przypisywany kanał, z którym będzie współpracował i kanały mogą się powtarzać (np. SOC2 i SOC3 mogą mieć przypisany ten sam kanał). Można zatem tak skonfigurować przetwornik A/C, że ten sam kanał będzie próbkowany z różnym czasem, w zależności od ustawionego źródła wyzwalania.

System priorytetów

W wielu aplikacjach przetwarzanych jest kilka wielkości analogowych i dlatego może dojść do sytuacji, w której w tym samym czasie zostanie zgłoszona potrzeba przetworzenia dwóch kanałów. Aby system mikroprocesorowy był w takich przypadkach przewidywalny, należy dobrze zrozumieć zasadę działania priorytetów, którymi MCU kieruje się w kolejności konwersji.

Przedstawiany przetwornik A/C oferuje dwa możliwe sposoby priorytetów, są to: algorytm karuzelowy (*Round Robin Priority*) oraz rozszerzony o dodatkowy poziom wysokich priorytetów algorytm karuzelowy. Ten ostatni jest nazywany przez producenta po prostu *High Priority*, a w artykule będzie używane określenie „algorytm wysokich priorytetów".

Algorytm karuzelowy priorytetów charakteryzuje się tym, że wyższy priorytet ma zawsze blok SOC o najbliższym numerze w stosunku do tego, na jaki aktualnie wskazuje wskaźnik karuzeli. Zasadę działania dla 12 bloków SOC ilustruje **rys. 3**. W przedstawionej sytuacji wskaźnik karuzeli wskazuje na SOC3, a w tym samym czasie bloki SOC5 i SOC8 zgłosiły potrzebę konwersji. Ponieważ SOC5 znajduje się bliżej bloku SOC3, na który wskazuje wskaźnik, to właśnie kanał przypisany do SOC5 będzie pierwszy poddany konwersji. Jako drugi zostanie przetworzony kanał związany z SOC8. Po wykonaniu konwersji wskaźnik karuzeli wskazuje na SOC8, a więc blok SOC9 ma teraz wyższy priorytet niż na przykład SOC0.

Użycie algorytmu wysokich priorytetów powoduje, że część bloków SOC może być wyłączona z przetwarzania w algorytmie karuzelowym, a tym samym będą one miały zawsze pierwszeństwo przed blokami SOC, które pozostały w karuzeli. Załóżmy, że wysoki priorytet został nadany blokom: SOC0 i SOC1 (**rys. 4**). Do chwili, kiedy bloki SOC z grupy wysokiego priorytetu nie zgłoszą potrzeby przetworzenia, cały mechanizm działa identycznie jak w przypadku zwykłego algorytmu karuzelowego. Jeśli jednak jednocześnie nadejdzie sygnał wyzwolenia dla bloku SOC z karuzeli i z grupy wysokich priorytetów, to zawsze ten ostatni będzie przetworzony pierwszy. Taką właśnie sytuację przedstawia rys. 4. W tym samym czasie zostały wyzwolone bloki SOC0 i SOC4, jednak to kanał z bloku SOC0 będzie obsłużony jako pierwszy. W przypadku, gdy bloki SOC0 i SOC1 zgłoszą potrzebę przetworzenia w tym samym czasie, to blok SOC0 ma pierwszeństwo.

Źródła napięcia odniesienia

Przetwornik A/C, w jaki wyposażony jest mikrokontroler czasu rzeczywistego TMS320F28027, może pracować (zależnie od konfiguracji) z wewnętrznym źródłem napięcia odniesienia lub ze źródłami zewnętrznymi. W pierwszym przypadku brany jest pod uwagę (przetwarzany) zakres 0...3,3 V, natomiast przy współpracy z zewnętrznymi źródłami odniesienia przetwarzany zakres zależy od wartości tych napięć.

Są dwa zewnętrzne źródła napięcia odniesienia: VREFLO i VREFHI, które wyznaczają zakres pomiarowy. Źródło VREFLO wyznacza dolną granicę napięcia, które będzie mierzone, a źródło VREFHI – górną. Innymi słowy, przetwarzane napięcie musi być większe od VREFLO i mniejsze od VREFHI. Badany przedział jest podzielony na 4096 poziomów.

Wyboru źródła napięcia odniesienia dokonuje się przez zapis bitu *ADCREFSEL* w rejestrze *ADCCTL1*: `AdcRegs.ADCCTL1.bit.ADCREFSEL = 0;`

Wartość 0 oznacza wybór wewnętrznego źródła odniesienia, natomiast zapis 1 załącza źródło zewnętrzne.

Kalibracja

Nie ma idealnych przetworników A/C, a wyniki przetwarzania zawsze są obarczone błędami. Pierwszą linią obrony przed nadmiernymi błędami jest kalibracja przetworników. Układ TMS320F28027 oferuje

kilka możliwości zminimalizowania błędów: przesunięcia zera, wzmocnienia oraz błędu wynikającego z prądu wejściowego. Wartości wszystkich poprawek zostały wyznaczone przez producenta i są zapisane w pamięci OTP. Wartości kalibracyjne są pobierane automatycznie podczas startu mikrokontrolera. Jedynie w przypadku tworzenia oprogramowania, gdy przeprowadzany jest proces debugowania, należy zadbać programowo o kalibrację przetwornika A/C. Na **list. 4** został przedstawiony fragment kodu, który realizuje kalibrację. Instrukcja *EALLOW* jest niezbędna, ponieważ rejestry kalibracyjne są chronione przed nieuprawnionym dostępem.

Wyzwalanie

Źródłem sygnału wyzwalania dla bloków SOC może być: wykonywany program, przerwanie od któregoś z timerów (0,1 lub 2), sygnał XINT2 oraz generator ePWM. Każdy blok SOC ma indywidualnie ustalane wyzwalanie. Wyboru dokonuje się przez zapis bitów *TRIGSEL* w rejestrze *ADCSOCxCTL*, gdzie „x" to numer bloku SOC. Przykładowo, jeśli blok SOC0 ma być wyzwalany za pomocą Timera 0, to zapis bitów *TRIGSEL* będzie wyglądał następująco: `AdcRegs.ADCSOC0CTL.bit.TRIGSEL = 1;`

Oczywiście w takim przypadku należy również skonfigurować do pracy Timer 0 oraz system przerwań.

Programowa inicjacja przetwarzania określonego bloku SOC jest realizowana przez zapis 1 do rejestru *ADCSOCFRC1*. Numer bitu rejestru odpowiada numerowi SOC, a więc programowe wymuszenie przetwarzania bloku SOC2 wygląda następująco: `AdcRegs.ADCSOCFRC1.bit.SOC2 = 1;`

Ponadto przetwornik A/C może generować sygnały ADCINT1 oraz ADCINT2, które mogą być tak ustawione, że zakończenie konwersji jednego kanału może wywołać konwersję innego. Obydwa sygnały są konfigurowane przez zapis rejestrów *ADCINTSOCSEL1* i *ADCINTSOCSEL2*. Wymienione rejestry mają wyższy priorytet od bitów *TRIGSEL*, zatem jeśli sygnał ADCINT1 lub ADCINT2 jest aktywny, to wtedy drugie źródło wyzwalania (wyznaczone przez bity *TRIGSEL*) jest ignorowane.

Wyzwalanie przetwarzania przez samego siebie (przez przetwornik A/C) jest mechanizmem pozwalającym na łatwą konfigurację do pracy w trybie ciągłym. Przykład takiej konfiguracji umieszczono na **list. 5**.

Jeśli kilka bloków SOC będzie miało ustawione takie samo źródło sygnału wyzwalania, to bloki te będą przetwarzane sekwencyjnie, począwszy od bloku SOC o najmniejszym numerze. Takie zachowanie gwarantuje system priorytetowania wykorzystujący omówiony wcześniej algorytm karuzelowy.

Próbkowanie dwóch kanałów jednocześnie

Jak już wspomniano przy okazji omawiania budowy przetwornika A/C, można go tak skonfigurować, aby dwa kanały były próbkowane jednocześnie. Służy to tego rejestr *ACDSAMPLEMODE*. Nie można jednak dowolnie dobierać par bloków SOC, które będą próbkowały w tym samym czasie. Pary zostały ustalone przez producenta mikrokontrolera, który zachował logiczną kolejność par zgodną z numeracją bloków SOC, czyli SOC0+SOC1, SOC2+SOC3, itd.

Warto tutaj przypomnieć, że każdy blok SOC może mieć dowolnie wybrany kanał pomiarowy, jednak w przypadku pracy równoległej są narzucone pewne ograniczenia.

Gdy tryb pracy jednoczesnej jest aktywny, wtedy zmienia się znaczenie bitów *CHSEL* w rejestrze *ADCSOCxCT* odpowiedzialnych za wybór przetwarzanego przez blok SOC kanału pomiarowego. Kanały te mogą być przetwarzane tylko w parach ADCINAx/ADCINBx. Umieszczenie w programie linii kodu: `AdcRegs.ADCSOC0CTL.bit.CHSEL = 0;` spowoduje, że zostanie wybrana para ADCINA0/ADCINB0, która będzie próbkowana równolegle.

Podsumowanie

W kolejnym artykule na temat mikrokontrolerów Piccolo zostanie przybliżona budowa i działanie układu generatora sygnału PWM. Posiadając wiedzę pozwalającą na swobodne korzystanie z przetwornika A/C i generatora PWM, można zagłębić się w poznawanie algorytmów przetwarzania sygnałów, których działanie będzie można sprawdzać od razu w praktyce.

Krzysztof Paprocki
paprocki.krzysztof@gmail.com