STM32 dla użytkowników 8-bitowców (2)

Obsługa przetwornika A/C

Przetwornik A/C jest jednym z najbardziej popularnych modułów peryferyjnych, w które są wyposażane mikrokontrolery. Zazwyczaj jest to jeden układ przetwornika z dołączonym do jego wejścia analogowym multiplekserem. Użytkownik może przełączać wejścia i mierzyć kilka wielkości analogowych – oczywiście nie w tym samym momencie. W artykule omówiono podstawy obsługi przetwornika A/C w STM32.

Dobierając przetwornik A/C do swoich potrzeb zwracamy uwagę na kilka najważniejszych parametrów. Pierwszy z nich to rozdzielczość bitowa – najbardziej eksponowana przez działy marketingu producenta. Obecnie nawet nieskomplikowane mikrokontrolery są wyposażane w przetworniki o rozdzielczości 12-bitowej. Więcej bitów to większa elastyczność pomiaru. Jeżeli jest potrzebna mniejsza rozdzielczość, na przykład 10-bitowa, to można ją zrealizować programowo poprzez odrzucenie 2 najmłodszych bitów wyniku konwersji. Zdarza się tez, że przetwornik można skonfigurować do pracy z mniejszą rozdzielczością.

Moduł przetwornika nie jest idealny i zależnie od technologii wykorzystywanej przez producenta i jego doświadczenia charakteryzuje większymi lub mniejszymi się błędami konwersji. Pomijam tu naturalne błędy kwantyzacji. Nieliniowość przetwarzania, źle zaprojektowany tor analogowy czy konfiguracja niepozwalająca na dołączenie precyzyjnego zewnętrznego napięcia referencyjnego mogą spowodować, że z 12 bitów w rzeczywistości uzyskamy 9. Dobrze jest, jeżeli producent uczciwie podaje, z jakimi niedoskonałościami mamy do czynienia. Na przykład dla 12-bitowego przetwornika mikrokontrolerze PIC16F1782 jest podawanych kilka różnych błędów przetwarzania wyrażonych w jednostkach LSB.

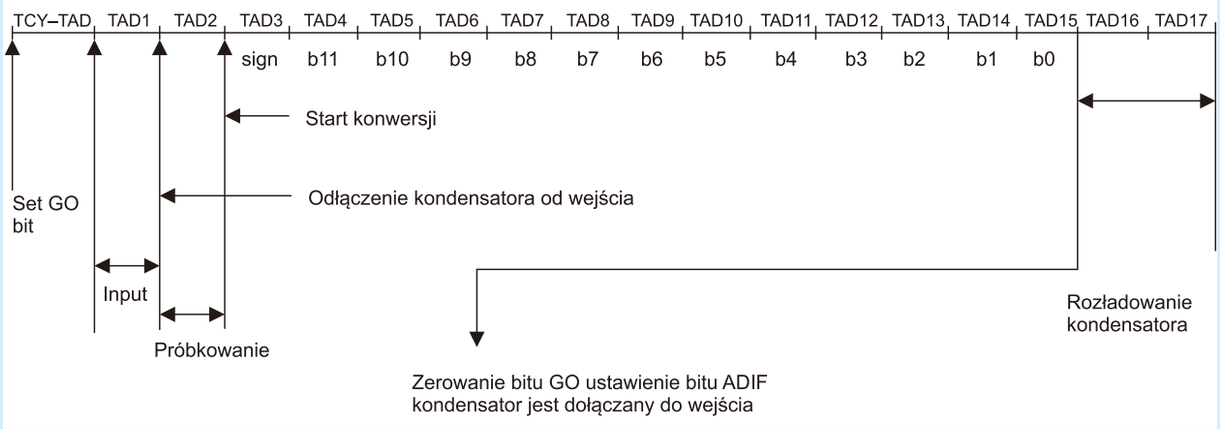
Kolejnym parametrem jest maksymalna częstotliwość, z którą przetwornik może próbkować sygnał analogowy. W przypadku 8-bitowców nie jest on aż tak istotny. Próbkowanie sygnałów z określonymi częstotliwościami to domena technik przetwarzania sygnałów. Co by nie twierdzić o coraz lepszych mikrokontrolerach 8-bitowych, to do przetwarzania sygnałów się nie nadają. Współcześnie są dostępne tanie, specjalizowane 16- i 32-bitowe mikrokontrolery zaprojektowane do tego celu. Żeby przyjrzeć się jak może wyglądać migracja z mikrokontrolera 8-bitowego na 32-bitowy załóżmy, że A/C będzie konwertował napięcia stałe lub wolnozmienne. Może to być pomiar napięcia stałego, prądu stałego (konwertera U/I) czy napięć wyjściowych konwerterów wielkości fizycznych na napięcie.

Zobaczmy jak wygląda działanie i konfigurowanie przetwornika we wspomnianym 8-bitowym mikrokontrolerze PIC16F1782. Moduł jest taktowany sygnałem zegarowym o programowanej częstotliwości. Okres TAD tego zegara nie może być krótszy niż 1 µs. Na **rysunku 1** pokazano zależności czasowe podczas konwersji, a na **rysunku 2** schemat blokowy modułu przetwornika.

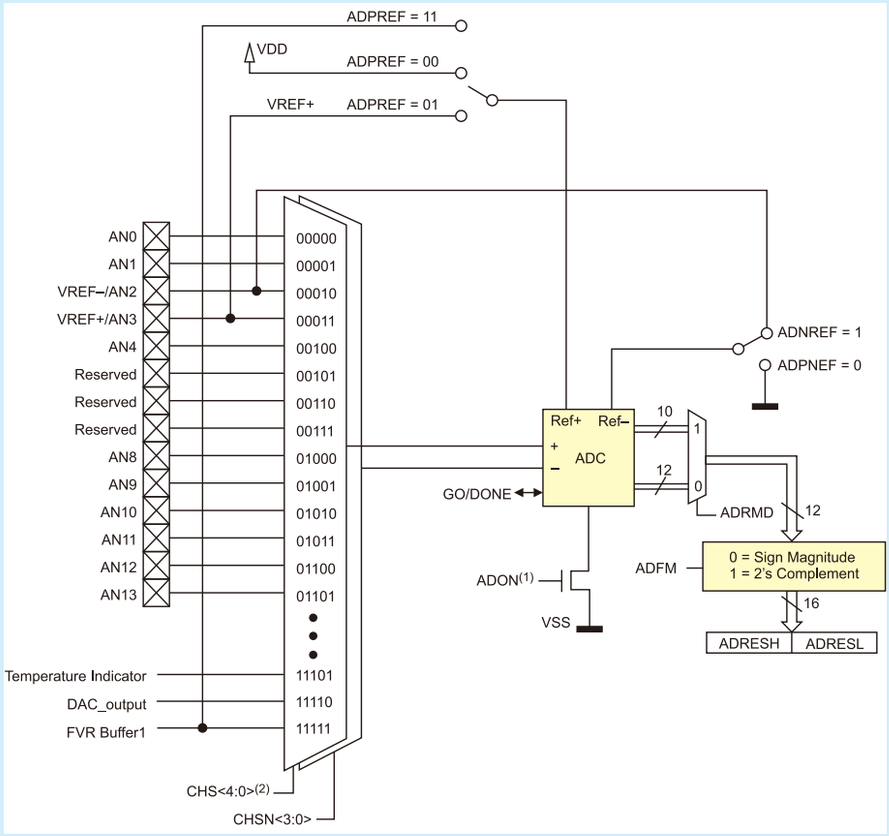
Start konwersji jest wykonywany po ustawieniu bitu GO. W kolejnych cyklach TAD jest wykonywane ładowanie kondensatora wejściowego, odłączenie kondensatora od analogowego wejścia mikrokontrolera, próbkowanie napięcia na kondensatorze i rozpoczęcie konwersji metodą sukcesywnej aproksymacji. Ta metoda charakteryzuje się wydłużeniem czasu konwersji i podwyższoną rozdzielczością bitową.

Aby przetwornik mógł prawidłowo pracować, należy go skonfigurować. Proces konfigurowania będzie polegał na:

- Włączeniu modułu przetwornika.
- Ustawieniu linii portu jako wejścia analogowe. Mikrokontrolery firmy Microchip po restarcie dołączają wszystkie wejścia przetwornika. Dlatego trzeba pozostawić w tym trybie wejścia używane przez aplikację jako



Rysunek 1 Konwersja w przetworniku Microchipa



Rysunek 2. Schemat blokowy modułu ADC PIC16F1782

Listing 1. Przykładowa konfiguracja przetwornika A/C mikrokontrolera PIC16F1782

```
void ADC_Init(void){
  TRISA0=1;//RA0 wejściowy
  ANSELA=0;//RA0 wejście analogowe
  //napięcie wejściowe single ended
  ADPREF0=0;//dodatnie +VREF=VDD
  ADPREF1=0;
  ADNREF=0;//ujemne -VREF=GND
  ADCS0=1; ADCS1=1; ADCS2=1;//taktowanie ADC wewnętrznym
oscylatorem RC
  ADRMD=0;//przetwarzanie 12 bitowe
  ADFM=0;//format binarny
  ADON=1;//włączenie modułu ADC
}
```

Listing 2. Pętla pomiaru

```
while(1)
{
  ADIF=0; //zeruj znacznik przerwania
  GO=1; //start konwersji
  while(!ADIF); //czekaj na zakończenie konwersji
  adc=ADRESH; //zapisz wynik do zniennej adc.
  adc=(adc<<8);
  adc=adc||ADRESL;
}
```

analogowe, a pozostałe przełączyć w tryb wejść/wyjść cyfrowych za pomocą rejestrów TRIS i ANSEL

- Wybraniu, które wejście analogowe jest dołączone do przetwornika.
- Zdefiniowaniu rodzaju sygnału analogowego: single ended lub symetryczny.
- Dołączeniu wybranego napięcia referencyjnego.
- Zaprogramowaniu źródła i częstotliwości zegara taktującego moduł A/C.
- Zaprogramowaniu układu przerwań (jeżeli będzie używany) .
- Ustaleniu formatu danych wyjściowych.

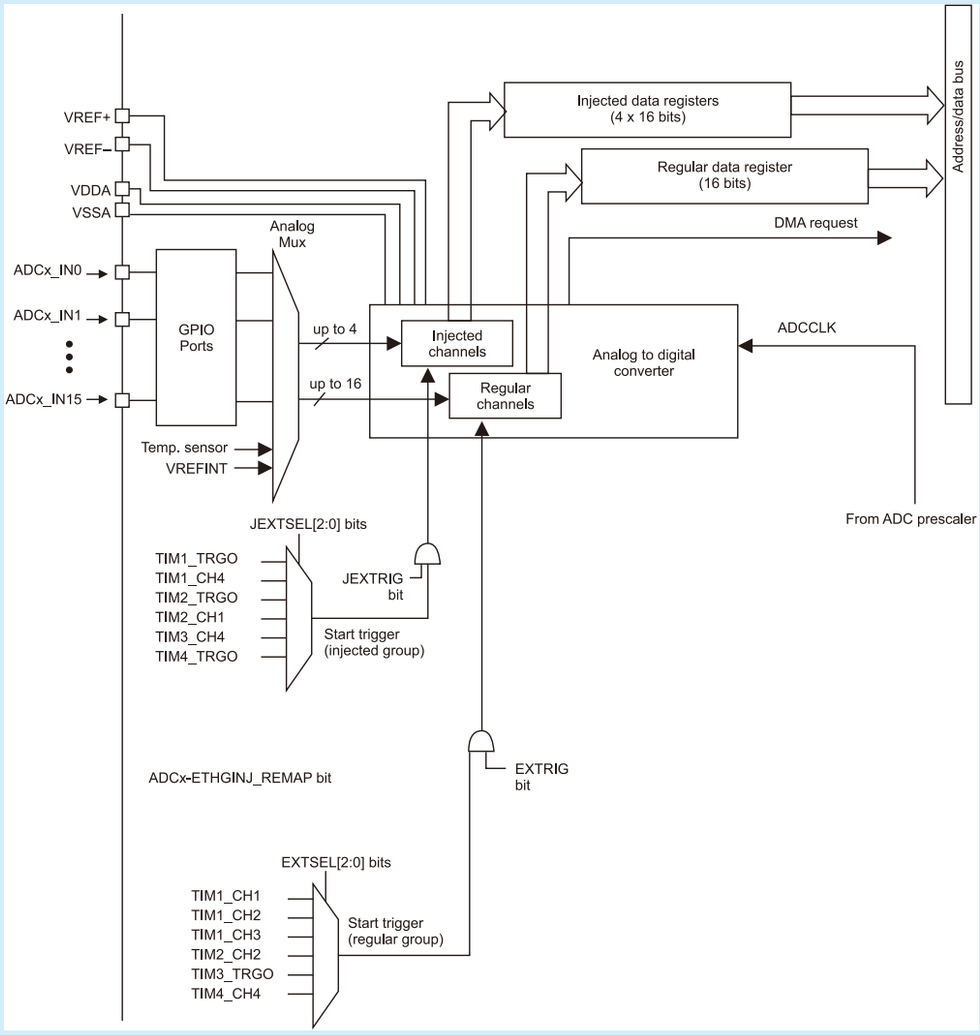
Jak widać to sporo czynności jak dla 8-bitowca, ale jak pokażę dalej – mniej niż dla mikrokontrolerów 32-bitowych. Trochę bardziej rozbudowana konfiguracja jest wynikiem wyposażania peryferii układów 8-bitowych (w tym wypadku A/C) w coraz więcej funkcji. Przykładową konfigurację pokazano na **listingu 1**.

Po skonfigurowaniu przetwornik jest gotowy do pracy. Konwersja rozpoczyna się po ustawieniu bitu GO/!DONE. Po jej zakończeniu konwersji bit GO/!DONE jest zerowany i jednocześnie jest ustawiany bit zgłoszenia przerwania ADIF. Jeżeli przerwanie jest odblokowane, to zostanie zgłoszone i trzeba je obsłużyć. Jednak nawet wtedy, gdy przerwanie jest zablokowane, to bit zostanie ustawiony i można go testować. Pętlę cyklicznego pomiaru napięcia pokazano na **listingu 2**.

Implementacja modułu przetwornika A/C w mikrokontrolerze z rdzeniem 32-bitowym ARM zależy od producenta mikrokontrolera i nie jest związana ze standardem. Z racji o większych możliwości obliczeniowych rdzenia należy spodziewać się, że ten moduł będzie miał lepsze parametry i większe możliwości, niż w mikrokontrolerze 8-bitowym z rdzeniem PIC16F. Jednak załóżmy, że na tym etapie nie potrzebujemy lepszego przetwornika. Będzie nas interesowało tylko to, co trzeba zrobić, by po zamianie z jednostki 8-bitowej na 32-bitowy STM32 układ działał tak samo lub podobnie. Schemat blokowy przetwornika A/C mikrokontrolera STM32 pokazano na **rysunku 3**.

Można tu znaleźć kilka podobieństw do układu z rysunku 2. Analogowy sygnał wejściowy jest podawany na wejście poprzez multiplekser analogowy, a przetwornik ma dodatkowe wyprowadzania dla napięcia referencyjnego. Konwersja analogowo-cyfrowa ma rozdzielczość 12-bitów, czyli jest taka sama, jak w PIC16F1782. Tu również wykorzystano metodę sukcesywnej aproksymacji. Jednak są też wyraźne różnice. Sygnały analogowe poddawane konwersji można przypisać do jednej z 2 grup: podstawowej (*regular channel*) i o nazwie *injected channel*. Grupa *injected* ma wysoki priorytet wyzwalania konwersji. Jeżeli pojawi się sygnał wyzwalania dla tej grupy kanałów, to zostanie zatrzymane przetwarzanie w grupie *regular* (po zakończeniu cyklu) i przetwornik będzie przetwarzał sygnały z wejść grupy *injected*. Ten mechanizm został stworzony po to, aby zapewnić jak najszybszą konwersję dla sygnałów krytycznych z punktu widzenia algorytmów przetwarzania sygnałów. Oprócz priorytetu są też inne różnice. Grupa *injected* może przetwarzać sygnały z maksymalnie 4 wejść, ale do każdego kanału jest przypisany osobny rejestr wyniku. Do grupy *regular* można przypisać maksymalnie 16 wejść, a rejestr wyniku jest wspólny. Z punktu widzenia konstruktora chcącego zastosować STM32F100 zamiast mikrokontrolera 8-bitowego nie ma znaczenia, którą grupę wybierzemy. Chyba, że mamy mierzyć sygnały z więcej niż 4 wejść. Wtedy naturalnym wyborem będzie grupa *regular channel*.

Jak można domyślić się, inicjalizowanie rozbudowanego modułu przetwornika nie jest banalne. Żeby sobie ułatwić pracę, skorzystamy z gotowych funkcji biblioteki *Standard*



Rysunek 3. Schemat blokowy przetwornika A/C mikrokontrolera STM32

Peripherial Library. Na **listingu 3** pokazano kompletną procedurę inicjalizacyjną dla jednego przetwornika pracującego w trybie niezależnym. Trzeba pamiętać, że mikrokontroler ma dwa przetworniki mogące pracować w różnych trybach. My założyliśmy, że konfigurujemy i będziemy używali jednego przetwornika w tak prostej konfiguracji, jak to tylko możliwe.

Przetwornik ma mierzyć napięcie tylko z jednego kanału pomiarowego. Każdy z pomiarów musi być wyzwolony programowo. Dlatego blokowany jest pomiar ciągły i wyzwalanie zewnętrzne. Czas trwania jednaj konwersji zależy od częstotliwości taktowania modułu przetwornika:

T= sampling time + 12,5 TADCCLK (TADCCLK – okres sygnału taktowania przetwornika)

Czas 12,5TADCCLK wynika z przetwarzania 12-bitowego za pomocą aproksymacji sukcesywnej. Czas próbkowania (*sampling time*) jest programowany i może mieć jedną z wartości: 1,5; 7,5; 13,5; 28,5; 41,5; 55,5; 71,5 i 239,5 ×TADCCLK. Funkcja ADC_RegularChannelConfig ustala długość czasu próbkowania (tu 28,5 cyklu ADDCLK), ale oprócz tego konfiguruje przetwornik do pracy w grupie *regular* i określa kanał, na którym będzie mierzone napięcie (kanał zerowy).

Maksymalna częstotliwość sygnału ADCCLK taktującego moduł przetwornika wynosi 14 MHz. Przetwornik A/C jest dołączony do APB2 taktowanej z maksymalną częstotliwością 72 MHz. Ale wiemy, że rdzeń STM32F100 może być taktowany z częstotliwością nie większą niż 24 MHz i z taką maksymalną częstotliwością pracują obie magistrale APB1 i APB2. Sygnał taktujący przetwornikiem jest dzielony przez wstępny dzielnik – preskaler o programowanym podziale przez 2, 4, 6 i 8.

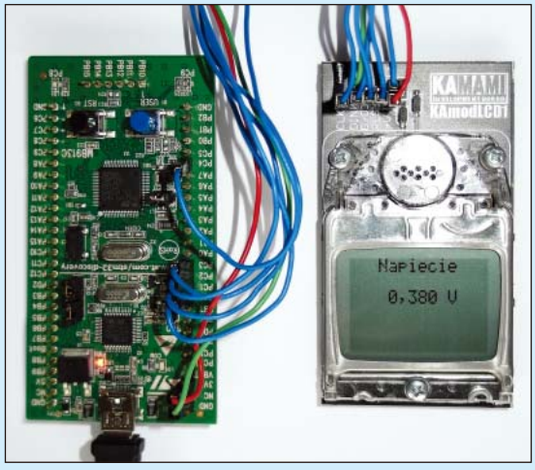
Dla szyny APB2 taktowanej z częstotliwością 24 MHz wybieramy podział przez 2. Częstotliwość ADCCLK ma wartość 12 MHz. Dla STM32F100 jest to maksymalna prędkość pracy A/C. Aby przetwornik mógł pracować, trzeba mu najpierw włączyć taktowanie z magistrali APB2:

Listing 3 Procedura inicjalizacyjna

```
void ADCInit(void)
{
  ADC_InitTypeDef ADC_InitStructure;//struktura
konfiguracji ADC
  //tylko jeden przetwornik
  ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
  //Pomiar z jednego kanału, skanowanie wylaczaone
  ADC_InitStructure.ADC_ScanConvMode = DISABLE;
  //Pomiar jednokrotny
  ADC_InitStructure.ADC_ContinuousConvMode = DISABLE;
  //wyzwalanie zewnetrzne zablokowane
  ADC_InitStructure.ADC_ExternalTrigConv = ADC_
ExternalTrigConv_None;
// dosuniecie wyniku do prawej - 12 mlodszych bitow
znaczacych
  ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
  ADC_InitStructure.ADC_NbrOfChannel = 1; //Liczba
kanalow =1
  ADC_Init(ADC1,&ADC_InitStructure); //Incjalizacja
przetwornika
//kanal 0, time sampling = 28,5cykla
  ADC_RegularChannelConfig(ADC1, ADC_Channel_0, 1, ADC_
SampleTime_28Cycles5);
  ADC_Cmd(ADC1, ENABLE); włączenie modułu przetwornika
}
```

Listing 4. Konfigurowanie PA0 jako wejście analogowe
```
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);//
włączenie taktowania portu A
GPIO_InitStructure.GPIO_Pin =  GPIO_Pin_0;//
inicjalizacja PA0
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;//wejście
analogowe
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOA, &GPIO_InitStructure);//incjalizacja
portu A
```

Listing 5. Kalibrowanie przetwornika
```
ADC_ResetCalibration(ADC1); //zerowanie rejestrów
kalibracyjnych
while(ADC_GetResetCalibrationStatus(ADC1)); //czekaj na
wykonanie zerowania
ADC_StartCalibration(ADC1); //Kalibrowanie ADC1
while(ADC_GetCalibrationStatus(ADC1) //czekaj na
zakończenie kalibracji
```



Fotografia 4. Układ testowy

RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE); //włącz taktowanie ADC1

A potem podzielić częstotliwość taktowania magistrali przez 2, tak by otrzymać częstotliwość 12 MHz:

RCC_ADCCLKConfig(RCC_PCLK2_Div2); //częstotliwość taktowania ADC = 24MHz/2=12MHz

Według dokumentacji mikrokontrolera kanał zerowy multipleksera analogowego A/C jest połączony z wyprowadzeniem PA0. Musi ono być skonfigurowane jako wejście analogowe (**listing 4**).

Przetworniki wbudowane w STM32 mają funkcję kalibracji pomiaru. Można ją uruchomić programowo, co zgodnie z dokumentacją producenta poprawia dokładność przetwarzania (**listing 5**):

Jeżeli program zatrzymuje się na warunku zakończenia kalibracji i nie chce się dalej wykonywać to oznacza, że moduł przetwornika nie został prawidłowo skonfigurowany i trzeba zwrócić uwagę na ustawienia dotyczące taktowania (dołączenie do magistrali APB2 i dzielnik częstotliwości taktowania).

Pojedynczy pomiar jest wyzwalany programowo za pomocą funkcji ADC_SoftwareStartConvCmd(ADC1, ENABLE). Po zakończenie konwersji moduł A/C ustawia znacznik ADC_FLAG_EOC, a wartość zmierzonego napięcia zwraca funkcja ADC_GetConversionValue(ADC1).

Listing 6. Pętla pomiaru napięcia
```
while (1)
{
//wyzwolenie pojedynczego pomiaru
  ADC_SoftwareStartConvCmd(ADC1, ENABLE);
//czekaj na zakonczenie konwersji
  while (!ADC_GetFlagStatus(ADC1,ADC_FLAG_EOC));
  pomiar = ADC_GetConversionValue(ADC1); //pobierz
zmierzona wartość
  pomiar = pomiar * 7324/10000; //przelicz wartość
  sprintf((char *)txt, „%d,%03d V\0”, pomiar / 1000,
pomiar % 1000);
  Poz(4,2); //pozycja na wyświetlaczu
  for(i=0;i<7;i++) //wyświetlenie zawartości bufora
  WriteChar(txt[i]);
}
```

Mikrokontroler zamontowany w STM32 Value Line Discovery jest zasilany napięciem +3 V. Napięcie zasilania jest jednocześnie napięciem referencyjnym przetwornika. Przy pomiarze o rozdzielczości 12-bitowej zmiana wartości najmniej znaczącego bitu odpowiada 0,0007324 V. Aby przekonwertować 12-bitową wartość odczytaną z rejestru wyniku przetwornika na napięcie w woltach, trzeba ją pomnożyć przez 0,0007324. Można do tego celu wykorzystać reprezentację zmiennoprzecinkową, ale lepiej jest operować na liczbach całkowitych. Wynik pomiaru przepisujemy do zmiennej o długości 32 bitów, a potem mnożymy przez 7324. Tak otrzymaną wartość dzielimy przez 10000 i wyniku otrzymujemy całkowitą liczbę reprezentująca mierzone napięcie z dokładnością do 3 miejsc po przecinku. Żeby sprawdzić poprawność działania procedur konfiguracji i pomiaru napięcia, można wynik pomiaru wyświetlić. Ja w tym celu dołączyłem do modułu STM32 Value Line Discovery mały wyświetlacz od telefonu Nokia 3310 (**fotografia 4**). Wartość binarną wyniku – po konwersji na napięcie – zamieniłem na znaki ASCII w formacie dziesiętnym za pomocą funkcji sprintf. Kompletną pętlę pomiaru pokazano na **listingu 6**.

Podsumowanie

Moduły cyfrowo analogowe (przetworniki A/C, C/A, komparatory) są z racji konieczności wykonania toru analogowego układami peryferyjnymi najtrudniejszymi do implementacji wewnątrz struktury mikrokontrolera. Z tego powodu ich parametry mogą różnić się zależnie od determinacji i doświadczenia producenta. Może się zdarzyć, że spotkamy bardzo dobry przetwornik w jednostce 8-bitowej i przeciętny w 32-bitowej. Samo użycie wydajnego rdzenia 32-bitowego nie gwarantuje dobrej jakości peryferii. Dlatego przed podjęciem decyzji o zmianie sprawdzonego mikrokontrolera 8-bitowego na nowy, 32-bitowy trzeba dokładnie przejrzeć dane katalogowe i ewentualnie wykonać potrzebne pomiary w rzeczywistym układzie. Jednak z drugiej strony, znani i doświadczeni producenci będą starali się, by układy peryferyjne dotrzymywały kroku możliwościom rdzenia. Należy spodziewać się, że przetworniki A/C w „mocnych” mikrokontrolerach z rdzeniem ARM będą miały wyższą maksymalną częstotliwość wykonywania pomiarów, możliwości wyzwalania czy konfigurowania różnych trybów pracy.

Przedstawione tu porównanie modułów A/C 8-bitowego mikrokontrolera z rodziny PIC16 i 32-bitowego STM 32 pokazuje, że dla prostych aplikacji niewymagających specjalnego wyzwalania i dużej prędkości wykonywania pomiarów, obie jednostki poradzą sobie równie dobrze. PIC16F ma z oczywistych względów prostszą budowę i co za tym idzie konfigurację. Microchip jest znany z układów peryferyjnych o bardzo dobrej jakości i należy spodziewać się, że praca przetwornika A/C nie będzie budziła podejrzeń. Przetwornik A/C z STM32F100 jest bardziej skomplikowany i trudniejszy w konfigurowaniu. Programistom znacząco pomagają w tym funkcje biblioteki Standard Peripherial Library, bo dla większości z nich „grzebanie” w rejestrach konfigurujących peryferia, a w szczególności te skomplikowane, jest zajęciem niezbyt lubianym. Wbudowanie mechanizmów kalibracyjnych sugeruje, że STM również poważnie podchodzi do jakości konwersji użytego przetwornika. Rozbudowana topologia i konfiguracja jest rekompensowana większymi możliwościami w wypadku, kiedy w przyszłości zechcemy wykonać bardziej zaawansowane pomiary.

Tomasz Jabłoński, EP