Analogowy STM32, czyli jak zrobić sobie zasilacz

Konstruowanie zasilaczy sterowanych cyfrowo jest popularną dyscypliną wśród elektroników. Zastosowanie mikrokontrolera zwiększa komfort użytkowania przy jednoczesnym zachowaniu dobrych parametrów użytkowych. Najczęściej są to układy znane z realizacji analogowych, w których za pomocą przetwornika C/A zmienia się wartość napięcia odniesienia i tym samym napięcie wyjściowe. Mikrokontroler pozwala na łatwą regulację, pomiar i wyświetlenie napięcia wyjściowego, pomiar i wyświetlenie prądu wyjściowego i realizację zabezpieczenia nadprądowego. Można też ograniczać moc strat w tranzystorze regulacyjnym poprzez programowe przełączanie odczepów transformatora zasilającego.

Jeszcze do niedawana trzeba było wykonywać przetwornik C/A z wykorzystaniem przebiegu PWM i filtra dolnoprzepustowego ze wzmacniaczem operacyjnym lub stosować układ scalony przetwornika. Obecnie w blok przetwornika C/A są wyposażane nawet tanie mikrokontrolery 8-bitowe.

Mikrokontrolery STM32 Value Line mają wbudowane dwa przetworniki A/C i 2 C/A. Niska cena i bardzo dobre wyposażenie powodują, że są to elementy idealnie nadające się do budowy sterownika zasilacza o regulowanym napięciu wyjściowym.

Nieco teorii

Jednym z podstawowych bloków funkcjonalnych zasilaczy stabilizowanych jest źródło napięcia odniesienia. W stabilizatorze to napięcie jest porównywane z napięciem wyjściowym. Im napięcie odniesienia ma lepsze parametry (stabilność temperaturowa, szumy, impedancja różniczkowa), tym lepsze będą parametry napięcia wyjściowego.

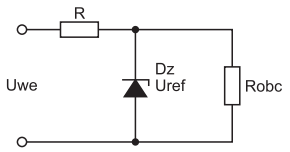
Typowo najczęściej taki obwód jest budowany w oparciu na diodzie Zenera. Na **rysunku 1** pokazano najprostszy układ stabilizatora. Napięcie niestabilizowane jest podawane przez rezystor R na stabilizator Dz. Współczynnik stabilizacji napięciowej to: $\Delta Uwe/\Delta Uref=1+R/Rz$, gdzie Rz to rezystancja różniczkowa diody

Ponieważ zależy nam na jak największym współczynniku stabilizacji, będziemy dążyli do zwiększenia wartości rezystora R lub zmniejszenia wartości Rz. Rezystancja różniczkowa Rz jest odwrotnie proporcjonalna do prądu płynącego przez diodę. Ale tego prądu nie można zwiększać z powodu ograniczonej mocy strat diody. Ponadto zwiększanie prądu diody wiąże się ze zmniejszaniem rezystancji R i tym samym zmniejszaniem współczynnika stabilizacji. Zwiększenie rezystancji R powoduje spadek prądu diody Zenera i tym samym wzrost Rz, dlatego nie można zwiększyć współczynnika stabilizacji, zwiększając rezystancję R przy stałym napięciu Uwe.

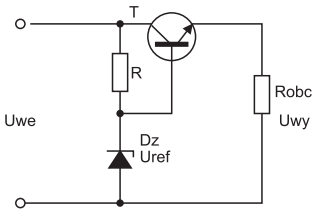
Diody mogą mieć różne napięcia stabilizacji (ze względu na tolerancję wykonania) i wydaje się, że wystarczy dobrać odpowiednią diodę i sprawa napięcia odniesienia jest załatwiona. W rzeczywistości, jeżeli stabilizator ma mieć dobre parametry, użycie diody Zenera o dowolnym napięciu może być problematyczne. Po pierwsze, napięcie stabilizacji dość mocno zależy od temperatury. Poza tym elementy te mają spore szumy własne, a napięcie szumów wzrasta przy niewielkim prądzie przewodzenia. W praktyce diody Zenera mają najlepsze parametry przy napięciu przebicia równym 6…7 V. Aby uzyskać lepsze parametry dla innych napięć, zaczęto stosować ulepszone diody zawierające w swojej strukturze dodatkowe elementy (złącza p-n) kompensujące dryft temperaturowy.

Z powodu problemów z parametrami, w scalonych stabilizatorach napięcia zamiast diod Zenera stosuje się w strukturze krzemowej odpowiednią liczbę połączonych szeregowo i skompensowanych temperaturowo złączy E-B. Problem odpowiedniej jakości napięcia odniesienia dotyczy też przetworników A/C i C/A. Dlatego dla bardziej wymagających zastosowań stosuje się specjalizowane, scalone źródła napięcia o bardzo dokładnej wartości, minimalnych szumach, dokładnie skompensowane temperaturowo.

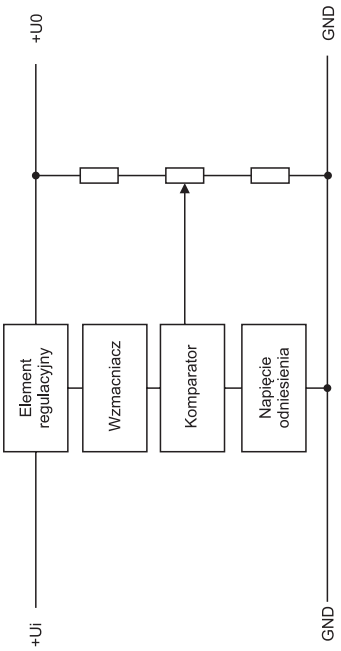
Parametry stabilizatora z rysunku 1 można poprawić, dodając element regulacyjny (wtórnik emiterowy) wprowadzający prądowe sprzężenie zwrotne, jak na **rysunku 2**. Taki układ znacznie podwyższa prąd wyjściowy stabilizatora, jednak tu również parametry diody Zenera mają decydujący wpływ na jakość napięcia wyjściowego (napięcie wyjściowe ma wartość $U_{wy}=U_{ref}-U_{be}$). Dalsze polepszenie właściwości stabilizatora jest możliwe przez wprowadzenie dodatkowego ujemnego sprzężenia zwrotnego. Schemat takiego rozwiązania pokazano na **rysunku 3**. Napięcie z wyjścia stabilizatora po podzieleniu (sygnał regulacji) trafia do komparatora, w którym jest porównywane z napięciem



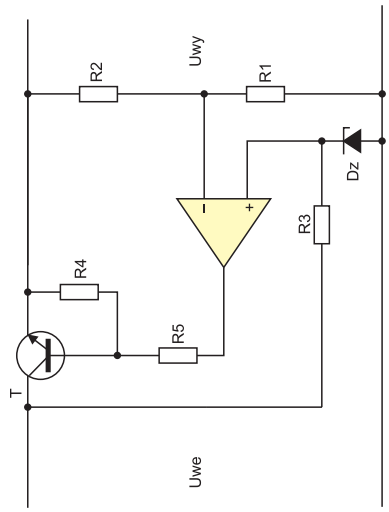
Rysunek 1. Najprostszy stabilizator napięcia z diodą Zenera



Rysunek 2. Stabilizacja za pomocą wtórnika emiterowego



Rysunek 3. Schemat blokowy stabilizatora



Rysunek 4. Praktyczna realizacja stabilizatora z regulowanym napięciem wyjściowym

odniesienia Uref. Sygnał błędu z wyjścia komparatora jest wzmacniany i steruje elementem regulacyjnym. Układ próbkuje napięcie wyjściowe i jeżeli nie jest ono równe zadanemu, to sygnał błędu z wyjścia komparatora tak steruje elementem wykonawczym, żeby tę różnicę zniwelować. Poza dobrymi parametrami taki układ pozwala na prostą regulację napięcia wyjściowego w szerokich granicach za pomocą potencjometru. Praktyczna realizacja układu z rys. 3 może wyglądać tak, jak na **rysunku 4**.

Wzmacniacz operacyjny z tranzystorem T pracują jako wzmacniacz operacyjny mocy objęty pętlą ujemnego sprzężenia zwrotnego przez dzielnik rezystancyjny złożony z rezystorów R1 i R2. Napięcie na jego wyjściu ma wartość Uwy=(1+R2/R1)Uref, gdzie Uref jest napięciem diody Zenera. Z tego prostego wzoru wynika, że napięcie wyjściowe Uwy stabilizatora można regulować przez zmianę stopnia podziału R2/R1 lub zmianę napięcia referencyjnego Uref. W typowych stabilizatorach napięcie referencyjne jest stałe, a napięcie wyjściowe zmienia się np. potencjometrem umieszczonym na wyjściu (zmiana podziału R2/R1).

W zasilaczu sterowanym cyfrowo spróbujemy odmiennego podejścia. Załóżmy, że (1+R2/R1) będzie mało stałą wartość równą 10. Napięcie wyjściowe według naszego wzoru Uwy=10×Uref. 12-bitowy przetwornik C/A pracujący jako źródło odniesienia w mikrokontrolerze na płytce *STM32Value Line* może na wyjściu generować napięcie z zakresu 0…3 V. Jeżeli zatem zamiast źródła napięcia odniesienia o stałej wartości zastosujemy zmieniane cyfrowo w zakresie 0…3 V źródło napięcia odniesienia, to przy założeniu, że Uwy=10×Uref, otrzymamy regulowane napięcie na wyjściu zasilacza w zakresie 0…30 V.

Żeby się o tym przekonać w praktyce, zmontowałem układ testowy ze wzmacniaczem 741 (taki akurat miałem pod ręką) i tranzystorem TIP110 (Darlington NPN). Za pomocą potencjometru ustaliłem podział 1/10, a całość zasiliłem napięciem +26 V. Zmienne napięcie z wyjścia przetwornika C/A symulował zasilacz o regulowanym napięciu wyjściowym. Początkowo układ nie chciał działać, bo do wyprowadzenia ujemnego napięcia zasilania dołączyłem masę układu. Jak się okazało, w takiej konfiguracji wzmacniacz 741 nie chciał działać. Po zasileniu wzmacniacza operacyjnego napięciami +Vcc=+26 V i –VCC=–5 V układ zaczął działać, tak jak wynika to z rozważań teoretycznych.

STM32 DAC

Mamy już układ analogowy regulowanego zasilacza spełniający podstawowy warunek: napięcie wyjściowe Uwy=10×Uref. Trzeba teraz za pomocą przetwornika C/A programowo ustawiać napięcie w zakresie 0…3 V.

Moduł przetwornika C/A w STM32F100 może być skonfigurowany do pracy w rozdzielczości 12- lub 8-bitowej i ma 2 niezależne konwertery. W praktyce są to 2 niezależne przetworniki. Ich wyjścia są dostępne na wyprowadzeniach PA4 (DAC1) i PA5 (DAC2). Zależnie od wybranej rozdzielczości, dane cyfrowe do konwersji są wpisywane do:

- 8 najmłodszych bitów rejestru DAC_DHR8Rx[7:0],
- 12 bitów dosuniętych do lewej w rejestrze DAC_DHR12Lx[15:4],
- 12 bitów dosuniętych do prawej w rejestrze DAC_DHR12Rx[11:0].

Dane z zapisywanych rejestrów DAC_DHR są automatycznie zapisywane do rejestrów przetwornika DAC_DOR w czasie jednego cyklu magistrali APB1, ale tylko w wypadku, kiedy użytkownik nie wybrał wyzwalania sprzętowego konwersji. Kiedy takie wyzwalanie zostało wybrane i warunek wyzwalania nastąpił, to przepisanie pomiędzy DAC_DHR i DAC_DOR zajmuje 3 cykle magistrali APB1. Po zapisaniu rejestru DAC_DOR napięcie na wyjściu zmienia się w czasie *Tsettling*, zależnym od napięcia zasilania i obciążenia wyjścia przetwornika C/A. Warto wspomnieć, że godząc się na pewną dodatkową nieliniowość, na wyjście można włączyć wbudowany w mikrokontroler analogowy układ buforujący i wtedy ma ono niską impedancję wyjściową. Napięcie wyjściowe przetwornika o rozdzielczości 12-bitowej można wyznaczyć ze wzoru:

DACotput=Uref (DOR/4095) [V], gdzie:

- Uref – napięcie referencyjne przetwornika,
- DOR – zawartość rejestru DAC_DOR.

Dla napięcia referencyjnego 3 V napięcie na wyjściu zmienia się w zakresie 0…3 V.

Przetwornik C/A jest dość rozbudowanym modułem peryferyjnym. Ma możliwość generowania szumu pseudolosowego i przebiegu trójkątnego. Połączenie możliwości wyzwalania od liczników i wejścia zewnętrznego z kanałami DMA daje możliwości zastosowania przetwornika do generowania zdefiniowanych przebiegów lub sygnałów audio. W naszym zastosowaniu te zaawansowane właściwości nie będą potrzebne i nie będę ich tutaj opisywał.

Przetwornik C/A w STM32

Do skonfigurowania przetwornika użyjemy funkcji standardowej biblioteki CMSIS dla mikrokontrolerów STM32. Do sterowania zasilaczem zostanie wykorzystany przetwornik DAC1 z wyjściem napięciowym na wyprowadzeniu PA4. W pierwszym kroku zadeklarujemy PA4 jako wyprowadzenie analogowe (**listing 1**).

Aby przetwornik pracował poprawnie, musi być odpowiednio skonfigurowany. Najpierw wyłączamy możliwość wyzwalania. Napięcie na wyjściu ma się zmienić po zapisaniu nowej wartości do rejestru przetwornika przez program użytkownika. Procedurę konfiguracyjną pokazano na **listingu 2**. Następnie trzeba zapewnić taktowanie modułowi ADC1 z magistrali APB1 oraz trzeba go włączyć, co pokazano na **listingu 3**. Zapis

Listing 1. Konfigururacja wyjścia PA4

```
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE); //włączenie taktowania GPIOA
  GPIO_InitStructure.GPIO_Pin =  GPIO_Pin_4;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
  GPIO_Init(GPIOA, &GPIO_InitStructure);
```

Listing 2. Konfiguracja przetwornika

```
void DACInit(void)
{
  DAC_InitTypeDef  DAC_InitStructure;
//wyłacz zewnętrzne wyzwalanie
  DAC_InitStructure.DAC_Trigger = DAC_Trigger_None;
//bufor dołączony na wyjście
  DAC_InitStructure.DAC_OutputBuffer = DAC_OutputBuffer_Enable;
//konfiguracja przetwornika
  DAC_Init(DAC_Channel_1, &DAC_InitStructure);
}
```

Listing 3. Włączenie taktowania i samego przetwornika ADC1

```
//taktowanie z magistrali APB1
RCC_APB1PeriphClockCmd(RCC_APB1Periph_DAC, ENABLE);
DAC_Cmd(DAC_Channel_1, ENABLE);//włącz przetwornik
```

rejestru DAC_DHR i w rezultacie DAC_DOR wykonuje funkcja *DAC_SetChannel1Data*. Jej argumentami są:

Wielkość i format danych do konwersji. My będziemy wpisywać dane 12-bitowe, dosunięte do prawej.

Cyfrowa wartość DOR do konwersji na napięcie analogowe zgodna z zależnością Uwy=Uref×(DOR/4095) [V].

Mamy teraz program umożliwiający ustawianie napięcia w zakresie 0…3 V na wyjściu PA4. W testowym układzie z **rysunku 4** usuwamy diodę Zenera i rezystor R3, a w miejsce tych elementów włączamy wyjście DAC, jak na **rysunku 5**. Zapisując różne wartości do konwersji w funkcji z **listingu 4,** ustawiamy różne napięcie wyjściowe zasilacza. Taki układ może pracować w praktycznej realizacji układu stabilizacji napięcia. Żeby zabezpieczyć się przed ewentualnym wzbudzaniem się na wysokich częstotliwościach, można ograniczyć pasmo przenoszenia wzmacniacza poprzez dołączenie pomiędzy wejściem odwracającym i wyjściem wzmacniacza rezystora 1 MΩ i równolegle do niego kondensatora 4,7 nF.

Przetwornik A/C

W tym momencie zasadnicza część zasilacza, czyli układ cyfrowego ustalania napięcia wyjściowego. Ale w konstrukcjach zasilaczy często wykorzystuje się inny, dużo bardziej popularny, analogowy moduł peryferyjny – przetwornik cyfrowo-analogowy (A/C). Mając go do dyspozycji, można na przykład mierzyć napięcie wyjściowe. Można też mierzyć prąd wyjściowy po zastosowaniu konwertera I/U. Ten pomiar pozwala na zaimplementowanie funkcji zabezpieczenia nadprądowego. Po przekroczeniu określonej wartości prądu wyjściowego można wyłączyć obwód wyjściowy za pomocą przekaźnika lub włączyć ograniczenie prądowe, wykorzystując dodatkowy układ analogowy sterowany przez przetwornik C/A.

Pomiar napięcia wyjściowego

Napięcie wyjściowe jest zwykle o wiele wyższe niż to, które może mierzyć przetwornik A/C. Dlatego jest ono wstępnie dzielone za pomocą dzielnika rezystancyjnego i podawane na wejście przetwornika. Następnie trzeba przeskalować wartość otrzymaną w wyniku konwersji, zmieniając ją na napięcie w woltach i wyświetlić. Jeżeli założymy, że napięcie wyjściowe nie jest większe niż 30 V, a napięcie referencyjne przetwornika ma wartość +3 V, to najlepiej, aby dzielnik wejściowy dzielił napięcie w stosunku 1:10. Na **rysunku 6** pokazano dzielnik o rezystancji całkowitej ok. 100 kΩ. Dzielnik ma w obwodzie potencjometr, bo ze względu na tolerancję wykonania rezystorów, precyzyjne dobranie rezystorów o wymaganej rezystancji jest kłopotliwe. Należy zastosować dobry potencjometr wieloobrotowy.

Pomiar prądu wyjściowego

Ponieważ pomiar napięcia jest najłatwiejszy, natężenie prądu mierzy się metodą pośrednią, techniczną. W mierzony obwód włącza się rezystor szeregowy o jak najmniejszej rezystancji (posobnik). Prąd płynący przez rezystor wywołuje na nim spadek napięcia, który z prawa Ohma jest równy U=I×R. Znając rezystancję i spadek napięcia łatwo wyznaczyć natężenie prądu ze wzoru I=U/R. Nie wolno przy tym zapomnieć o stratach mocy na rezystancji.

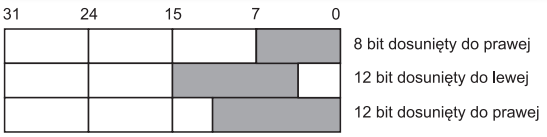
Załóżmy, że nasz zasilacz będzie dostarczał prąd o natężeniu maksymalnym 2 A, a bocznik będzie miał rezystancję 0,1 Ω. Maksymalny spadek napięcia U=2 A×0,1 Ω=0,2 V. Z jednej strony 200 mV spadku napięcia wyjściowego to sporo jak na zasilacz, ale z drugiej dość trudno będzie dokładnie zmierzyć to napięcie przetwornikiem o zakresie napięcia wejściowego 0…3 V. Najprostszym rozwiązaniem będzie wzmocnienie napięcia stałego odkładającego się na boczniku za pomocą wzmacniacza operacyjnego pracującego w konfiguracji wzmacniacza nieodwracającego. Wzmocnienie takiego układu dobiera się za pomocą dzielnika złożonego z 2 rezystorów (**rysunek 7**). Wzmocnienie układu wzmacniacza wynosi Ku=1+R2/R1. Jeżeli przyjmiemy, że dla 2 A będziemy potrzebowali napięcia wyjściowego na poziomie 2 V, to wzmocnienie musi być 10-krotne, a stosunek R1/R1 równy 9. Rzeczywisty układ pomiarowy może wyglądać jak na **rysunku 8**. Dokładne regulowanie wzmocnienia umożliwi potencjometr połączony szeregowo z R2.

Rezystor pomiarowy powinien mieć rezystancje tak małą, jak to możliwe, szczególnie dla większych prądów. Jednak zmniejszanie rezystancji powoduje konieczność większego wzmocnienia spadku napięcia. Przy dużych wzmocnieniach pojawiają się problemy ze stabilnością i trzeba mieć duże doświadczenie w aplikowaniu takich rozwiązań. Poza tym dostępność precyzyjnych rezystorów o małych rezystancjach jest niewielka i są to elementy kosztowne. Dlatego konstruktor musi przyjąć rozsądny kompromis pomiędzy rezystancją posobnika i wzmocnieniem wzmacniacza pomiarowego. Można też zastosować specjalizowane układy scalone z wbudowanym, kalibrowanym rezystorem pomiarowym i wzmacniaczem operacyjnym.

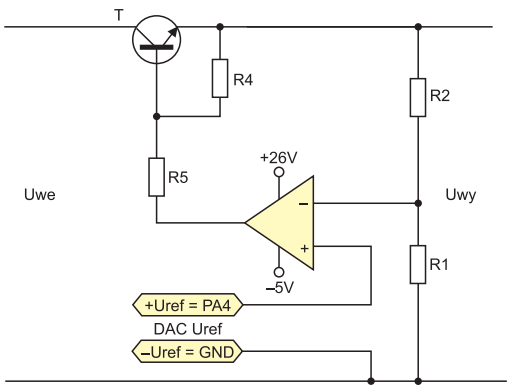
W naszym rozwiązaniu można przyjąć, że wzmocnienie może sięgać granicy 50.

Listing 4. Zapisanie 12-bitowej danej do konwersji

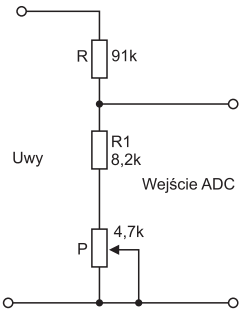
```
DAC_SetChannel1Data(DAC_Align_12b_R, 2048);
```



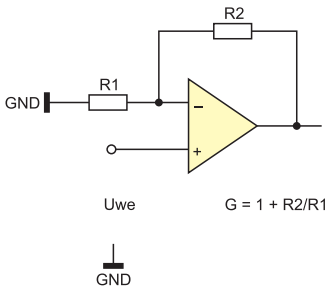
Rysunek 5. Umieszczenie danych do konwersji



Rysunek 6. Układ do testowania zasilacza z napięciem referencyjnym z przetwornika C/A



Rysunek 7. Dzielnik 1:10 do pomiaru napięcia wyjściowego



Rysunek 8. Wzmacniacz nieodwracający

Żeby układ był bardziej stabilny, można mu ograniczyć pasmo od góry przez równoległe dołączenie do R2 kondensatora o pojemności kilku nF. Ja zmontowałem układ próbny ze wzmocnieniem x10 i w czasie testów sprawował się zupełnie poprawnie.

W mikrokontrolerach rodziny STM32F100 są wbudowane dwa niezależne przetworniki o rozdzielczość 12 bitów. Napięcie podawane na wejście analogowe można przypisać do jednej z 2 grup: grupy podstawowej (*regular channel*) i grupy, dla której trudno znaleźć odpowiednik polskiej nazwy – *injected channel*. Grupa *injected* ma wysoki priorytet wyzwalania konwersji. Jeżeli pojawi się sygnał wyzwalania dla tej grupy kanałów,

Listing 5. Konfiguracja wejść analogowych
```
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
GPIO_InitStructure.GPIO_Pin =  GPIO_Pin_0 | GPIO_Pin_1;
 GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;//wejscie analogowe
 GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOA, &GPIO_InitStructure);
```

Listing 6. Konfiguracja taktowania i włączenie przetwornika
```
RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE); //włącz taktowanie ADC1
RCC_ADCCLKConfig(RCC_PCLK2_Div2);//taktowanie przetwornika =12MHz
ADC_Cmd(ADC1, ENABLE);
```

Listing 7. Kalibracja przetwornika
```
ADC_ResetCalibration(ADC1);             //Reset rejestrow kalibracyjnych ADC1
while(ADC_GetResetCalibrationStatus(ADC1)); //Odczekanie na wykonanie resetu
ADC_StartCalibration(ADC1);              //Kalibracja ADC1
while(ADC_GetCalibrationStatus(ADC1)) //odczekanie na wykonanie kalibracji
```

i włączyć. Każdy z pomiarów będzie wyzwolony programowo. Dlatego trzeba wyłączyć pomiar ciągły i wyzwalanie zewnętrzne. Czas trwania konwersji zależy od częstotliwości taktowania modułu przetwornika. Przetwornik może mierzyć napięcia z 16 wejść analogowych przełączanych za pomocą multipleksera analogowego. My potrzebujemy zmierzyć napięcie wyjściowe i napięcie z konwertera prąd/napięcie (czyli prąd wyjściowy). Ustalmy, że napięcie będzie mierzone w kanale 0 (linia PA0), a prąd w kanale 1 (PA1). Linie PA0 i PA1 trzeba skonfiguro-

Listing 8. Konfiguracja przetwornika ADC
```
void ADCInit(void)
{
ADC_InitTypeDef ADC_InitStructure;//struktura konfiguracji ADC
ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;                  //Jeden przetwornik, praca niezalezna
ADC_InitStructure.ADC_ScanConvMode = DISABLE;                       //Pomiar jednego kanalu, skanowanie kanalow nie
potrzebne
ADC_InitStructure.ADC_ContinuousConvMode = DISABLE;                 //Pomiar w trybie jednokrotnym
ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;  //Brak wyzwalania zewnetrznego
ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;               //Wyrownanie danych do prawej - 12 mlodszych bitow
znaczacych
ADC_InitStructure.ADC_NbrOfChannel = 1;                              //Liczba uzywanych kanalow =1
ADC_Init(ADC1, &ADC_InitStructure);                                 //Incjalizacja przetwornika
}
```

Listing 9. Pomiar i wyświetlenie napięcia i prądu wyjściowego
```
while (1)
  {
//pomiar napięcia
DispLcd("   Napiecie  ",0,0);
DispLcd("             ",0,1);
ADC_RegularChannelConfig(ADC1, ADC_Channel_0, 1, ADC_SampleTime_28Cycles5);  //kanał 0 pomiar napięcia
ADC_SoftwareStartConvCmd(ADC1, ENABLE);                              //wyzwolenie pojedynczego pomiaru
 while (!ADC_GetFlagStatus(ADC1,ADC_FLAG_EOC));          //odczekaj na zakonczenie konwersji

pomiar  = ADC_GetConversionValue(ADC1);             //pobiez zmierzona wartosc
pomiar =  pomiar * 7324/10000;               //przelicz wartosc wyrazona jako calkowita, 12-bit na rzeczywistą
pomiar= pomiar*10; //zakres do 30V
sprintf((char *)txt, „%d,%02d V\0”, pomiar / 1000, pomiar % 1000);//konwersja na ASCII
Poz(4,2);
for(i=0;i<7;i++)
WriteChar(txt[i]);

//pomiar prądu
DispLcd("    Prad  ",0,4);
ADC_RegularChannelConfig(ADC1, ADC_Channel_1, 1, ADC_SampleTime_28Cycles5);
ADC_SoftwareStartConvCmd(ADC1, ENABLE);                              //wyzwolenie pojedynczego pomiaru
 while (!ADC_GetFlagStatus(ADC1,ADC_FLAG_EOC));          //odczekaj na zakonczenie konwers   ji
pomiar  = ADC_GetConversionValue(ADC1);             //pobiez zmierzona wartosc
pomiar =  pomiar * 7324/10000;               //przelicz wartosc wyrazona jako calkowita, 12-bit na rzeczywista
sprintf((char *)txt, „%d,%03d A\0”, pomiar / 1000, pomiar % 1000);
Poz(4,5);
for(i=0;i<7;i++)
WriteChar(txt[i]);
  }
```

to po zakończeniu bieżącego cyklu zostanie zatrzymane przetwarzanie w grupie *regular* 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 jeszcze 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 dla wszystkich. W naszym zasilaczu, przy stosunkowo dużej wydajności rdzenia i szybkich przetwornikach, możemy użyć grupy *regular* do pomiaru napięcia i prądu wyjściowego.

Sygnał taktujący przetwornikiem ADCCLK może mieć częstotliwość maksymalną 14 MHz. 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 pobierany z szyny APB2 i dzielony przez wstępny preskaler o podziale przez 2 (ADCCLK=12 MHz).

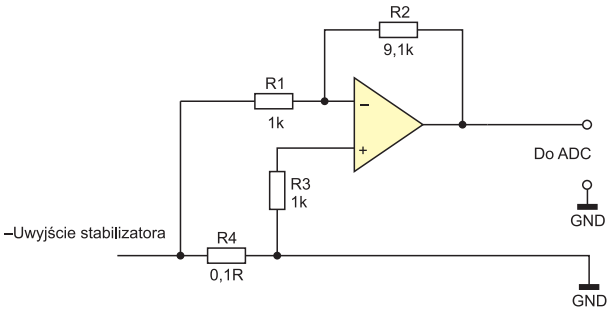
Znając częstotliwość taktowania, można określić, jak długo będzie trwała konwersja:
$T = sampling\ time + 12{,}5 \times TADCCLK$
(TADCCLK – okres sygnału taktowania przetwornika)

Czas 12,5×TADCCLK wynika z przetwarzania 12-bitowego przy metodzie sukcesywnej aproksymacji. Czas *sampling time* jest programowany i może mieć jedną z wartości: 1,5; 7,5; 13,5; 28,5; 41,5; 55,5; 71,5 lub 239,5×TADCCLK.

Moduł przetwornika trzeba przed użyciem skonfigurować, zapewnić taktowanie wać jako wejścia analogowe (**listing 5**). Na **listingu 6** pokazano konfigurację taktowania i włączenie przetwornika. Taktowany i włączony moduł A/C można poddać procedurze kalibracji, co poprawia dokładność konwersji (**listing 7**).

Teraz kiedy przetwornik jest taktowany, zasilony i skalibrowany, trzeba go jeszcze skonfigurować. Moduł A/C jest dość rozbudowany i może pracować w różnych trybach. Nas będzie interesowała praca w trybie *regular* z pomiarem w 2 kanałach. Jest możliwe włączenie automatycznego przemiatania kanałów pomiarowych. Wtedy po zakończeniu pomiaru z jednego kanału jest automatycznie inicjowany pomiar w następnym kanale. Jednak jest z tym problem, bo w trybie *regular* wynik konwersji jest zapisywany do jednego rejestru. Użytkownik po włączeniu skanowania nie wie, z którego kanału odczytuje

pomiar. Tę niedogodność można wyeliminować, stosując przesyłanie wyników poszczególnych pomiarów przez kanał DMA do tablicy wyników umieszczonej w pamięci RAM. Ja postanowiłem zastosować typową metodę polegającą na konfigurowaniu jednego kanału pomiarowego przed pomiarem, wykonaniu i zapisaniu pomiaru, a następnie skonfigurowaniu następnego kanału i wykonaniu pomiaru z tego kanału. Do tego celu zdefiniowałem funkcję *ADCInit* konfigurującą przetwornik do pracy niezależnej i bez wyzwalania zewnętrznego. Pomiar jest wykonywany w pojedynczym kanale, w trybie jednokrotnym (skanowanie kanałów jest wyłączone). Po skonfigurowaniu kanał, w którym będzie wykonywany pomiar jest wybierany funkcją *ADC_RegularChannelConfig* z dwoma argumentami. Pierwszy argument określa kanał pomiarowy, a drugi czas konwersji. Wykonanie jednego pomiaru w konkretnym kanale wygląda następująco:



Rysunek 9. Układ do pomiaru prądu

- Wybieramy kanał, w którym będziemy wykonywać pomiar (funkcja *ADC_RegularChannelConfig*).
- Wyzwalamy programowo jeden pomiar (funkcja *ADC_SoftwareStartConvCmd*).
- Czekamy na zakończenie konwersji (funkcja *ADC_GetFlagStatus*).
- Po zakończeniu pomiaru odczytujemy rejestr wyniku (funkcja *ADC_GetConversionValue*).

Na **listingu 9** pokazano pętlę pomiaru i wyświetlenia napięcia wyjściowego i prądu.

Podsumowanie

Bogate wyposażenie w peryferyjne moduły analogowe, atrakcyjna cena i dostępność powodują, że stosowanie 32-bitowych mikrokontrolerów do budowy sterowników analogowych zasilaczy stabilizowanych jest jak najbardziej uzasadnione. Mikrokontrolery 8-bitowe, najczęściej stosowane w sterownikach zasilaczy, cierpią głównie na brak przetworników C/A. Tu nie ma tego problemu. Oczywiście można się zastanawiać, czy w takim dość nieskomplikowanym programowo zastosowaniu potrzebny jest „aż" 32-bitowy rdzeń. Jeżeli jednak popatrzymy na to ze strony praktycznej, nie ma to znaczenia. Ważne, że mamy do dyspozycji mikrokontroler z potrzebnymi zasobami w cenie jednostki 8-bitowej.

Tomasz Jabłoński, EP