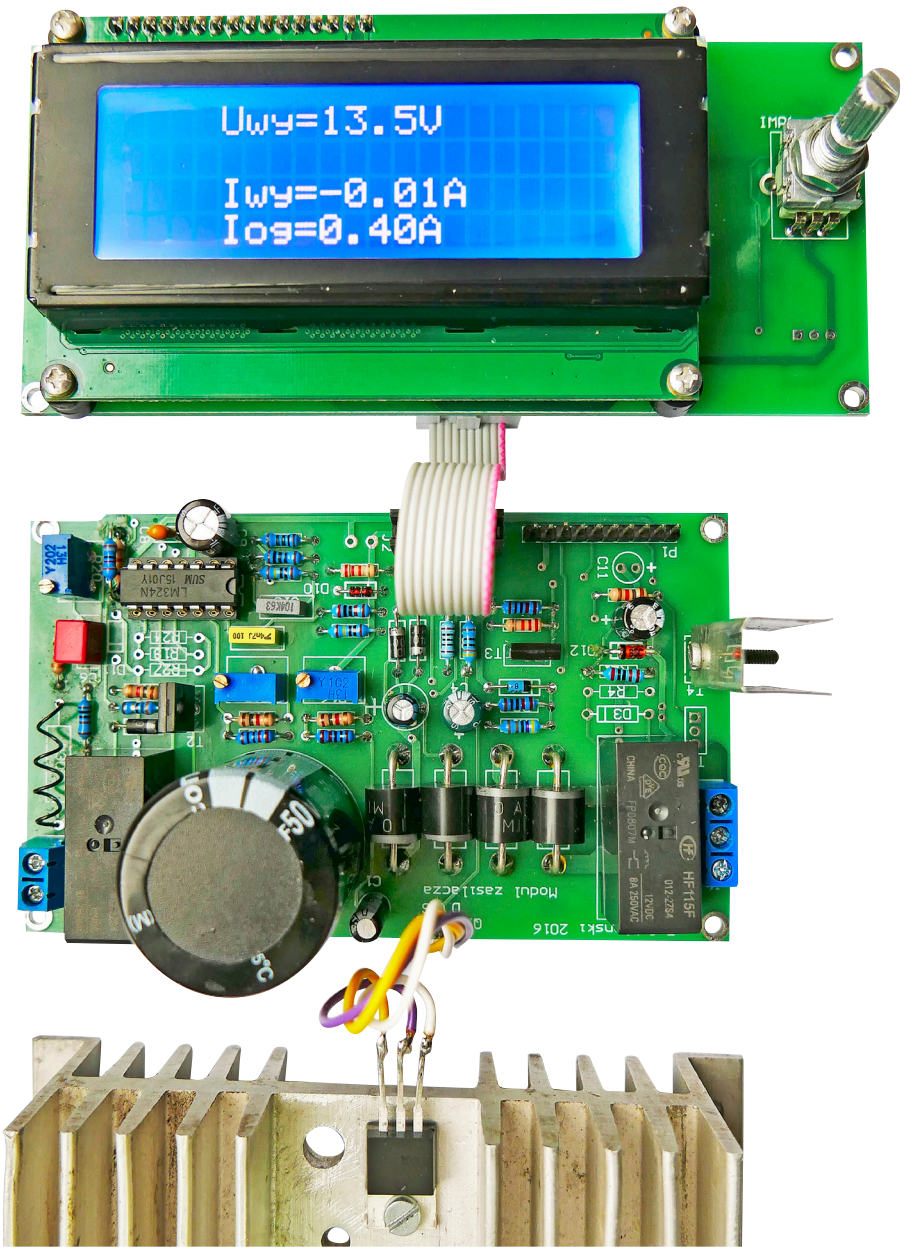
Zasilacz warsztatowy (2)

Klika lat temu opisywałem na łamach „Elektroniki Praktycznej” projekt zasilacza sterowanego cyfrowo. Ten zasilacz jest moim podstawowym zasilaczem i pracuje bezawaryjnie do dzisiaj, jednak postanowiłem zaprojektować i zbudować kolejny zasilacz, o lepszych parametrach elektrycznych, nieco prostszy w konstrukcji. W pierwszej części artykułu omówiłem budowę zasilacza. Teraz czas na opisanie jego oprogramowania, które jest nietrywialne do wykonania.



DODATKOWE MATERIAŁY NA FTP:

ftp://ep.com.pl

USER: 44747, PASS: 3qwdwa8u

W ofercie AVT*

AVT-5579

Podstawowe informacje:

- Napięcie wyjściowe regulowane w zakresie 0...28 V DC.
- Prąd wyjściowy regulowany w zakresie 0...3 A.
- Stabilizator liniowy, analogowy z szeregowym tranzystorem regulacyjnym.
- Pomiar prądu i napięcia wyjściowego.
- Sterowanie zasilaczem za pomocą mikrokontrolera.

Projekty pokrewne na FTP:
(wymienione artykuły są w całości dostępne na FTP)

AVT-1946	Zasilacz napięcia symetrycznego z LM27762 (EP 2/2017)
AVT-1938	Moduł zasilacza z układem KDSN05 (EP 11/2016)
AVT-1895	Uniwersalny moduł zasilający (EP 10/2016)
AVT-1913	Moduł miniaturowego zasilacza (EP 8/2016)
AVT-5546	Stabilizator z kompensacją spadku napięcia na przewodach połączeniowych (EP 7/2016)
AVT-1882	Regulowany zasilacz napięcia symetrycznego (EP 9/2015)
AVT-1865	Dołączany do USB zasilacz napięcia symetrycznego z układem ADP5071 (EP 8/2015)

*** Uwaga! Elektroniczne zestawy do samodzielnego montażu. Wymagana umiejętność lutowania!**
Podstawową wersją zestawu jest wersja [**B**] nazywana potocznie KITem (z ang. zestaw). Zestaw w wersji [**B**] zawiera elementy elektroniczne (w tym [UK] – jeśli występuje w projekcie), które należy samodzielnie wlutować w dołączoną płytkę drukowaną (PCB). Wykaz elementów znajduje się w dokumentacji, która jest podlinkowana w opisie kitu.
Mając na uwadze różne potrzeby naszych klientów, oferujemy dodatkowe wersje:
- wersja [**C**] zmontowany, uruchomiony i przetestowany zestaw [**B**] (elementy wlutowane w płytkę PCB)
- wersja [**A**] płytka drukowana bez elementów i dokumentacja

Kity w których występuje układ scalony wymagający zaprogramowania, posiadają następujące dodatkowe wersje:
- wersja [**A+**] płytka drukowana [**A**] + zaprogramowany układ [**UK**] i dokumentacja
- wersja [**UK**] zaprogramowany układ

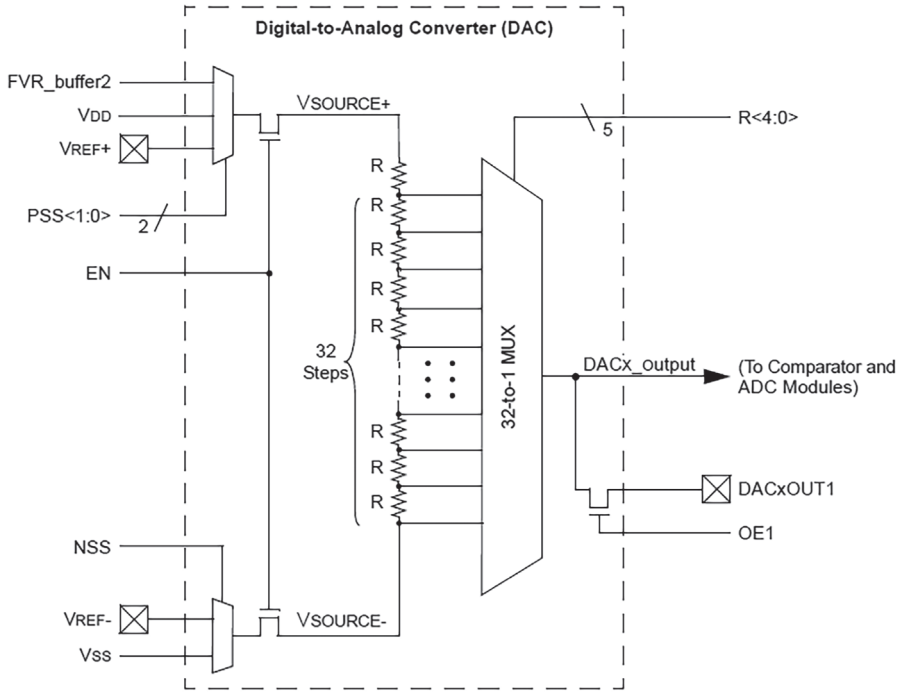
Nie każdy zestaw AVT występuje we wszystkich wersjach! Każda wersja ma załączony ten sam plik pdf! Podczas składania zamówienia upewnij się, którą wersję zamawiasz! ***http://sklep.avt.pl***

Program sterujący napisano w środowisku MPLAB X i skompilowano bezpłatnym kompilatorem MPLAB XC8. Do konfigurowania układów peryferyjnych użyto wtyczki MPLAB Code Configurator (MCC). Wszystkie te narzędzia są dostępne bezpłatnie i można je pobrać ze strony firmy Microchip. Mikrokontroler PIC16F1769 jest stosunkowo nowy, więc trzeba użyć najnowszych wersji oprogramowania wspierających ten układ.

Przetworniki C/A, źródło napięcia odniesienia i wzmacniacze operacyjne

Do regulacji napięcia i prądu wyjściowego jest potrzebne regulowane źródło napięcia odniesienia. W typowych rozwiązaniach stosuje się stabilne (stabilizowane) źródło napięcia dzielone przez precyzyjny potencjometr. W sterownikach mikroprocesorowych funkcję regulowanego źródła napięcia referencyjnego pełni przetwornik C/A. Mikrokontroler PIC16F1769 ma wbudowane dwa 10-bitowe i dwa 5-bitowe przetworniki C/A. W sterowniku użyłem obu przetworników 10-bitowych. Jeden jest źródłem regulowanego napięcia odniesienia i służy do zmiany napięcia wyjściowego, a drugi jest źródłem regulowanego napięcia odniesienia do zmiany natężenia prądu zabezpieczenia prądowego.

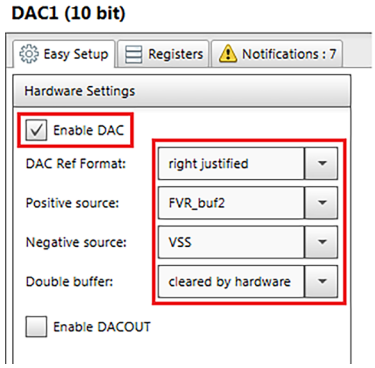
Schemat blokowy przetwornika pokazano na **rysunku 8**. Zasadniczymi elementami modułu są drabinka rezystancyjna



Rysunek 8. Schemat blokowy przetwornika C/A

oraz multiplekser analogowy. Drabinka jest zbudowana z rezystorów o rezystancji ok. 600 Ω. Do ostatniego, „górnego" rezystora dołączono dodatni biegun napięcia referencyjnego określającego maksymalne napięcie na wyjściu przetwornika. Napięciem referencyjnym mogą być: napięcie zasilania mikrokontrolera VDD, napięcie z zewnętrznego źródła podłączone do wyprowadzenia Vref+, napięcie z modułu FVR_buffer2. Ujemny biegun napięcia referencyjnego jest podłączony na stałe do VSS – masy mikrokontrolera.

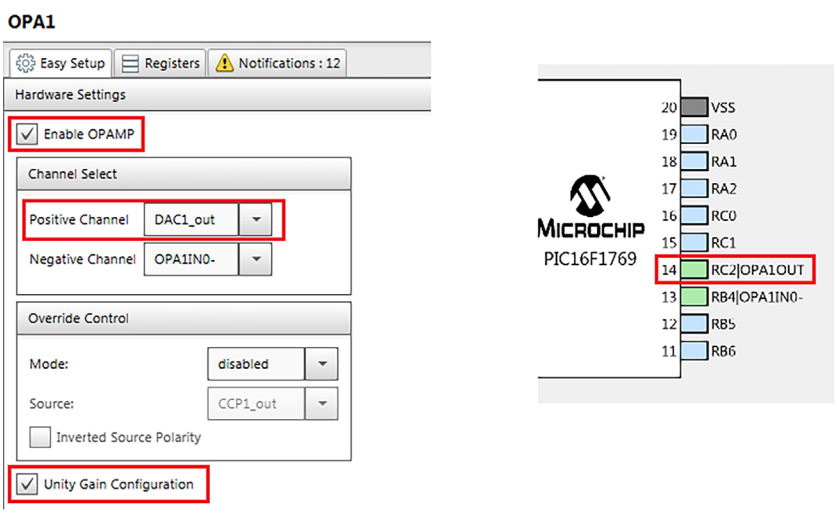
Przed użyciem moduł przetwornika C/A musi być skonfigurowany. Jak już wspomniałem, wszystkie moduły peryferyjne będą konfigurowane za pomocą wtyczki MCC środowiska IDE MPALB X. Konfiguracja przetwornika cyfrowo-analogowego w MCC jest bardzo łatwa. Trzeba włączyć moduł (opcja „Enable DAC"), wybrać dodatnie napięcie referencyjne oraz odblokować wyjście napięcia na wyprowadzenie DACOUT1, jak na **rysunku 9**. W konfiguracji nie zaznaczono „Enable DACOUT". Wyjście przetwornika zostanie wewnętrznie połączone z wejściem wbudowanego wzmacniacza operacyjnego pracującego jako bufor. W tym celu należy skonfigurować moduł wzmacniacza operacyjnego, tak aby wejście nieodwracające było połączone z wyjściem przetwornika, jak na **rysunku 10**. Po zaznaczeniu opcji „Unity Gain Configuration" wzmacniacz zaczyna pracować



Rysunek 9. Konfiguracja modułu przetwornika DAC1 do regulacji napięcia wyjściowego

jako standardowy bufor (**rysunek 11**). Nie ma wtedy znaczenia, co zostanie wybrane jako sygnał doprowadzony do wejścia odwracającego w konfiguracji wzmacniacza operacyjnego.

Buforowanie wyjścia eliminuje wpływ impedancji obciążenia na przetwornik C/A. W konfiguracji przetwornika wybraliśmy jako napięcie referencyjne – opcję *FVR_buf2*. W systemie analogowym używającym przetworników C/A i A/C trzeba się zmierzyć z problemem jakości napięcia odniesienia. Najłatwiej jest użyć napięcia zasilania jako napięcia odniesienia. Ma to jednak wady: ma ono wartość zależną od egzemplarza stabilizatora oraz zwykle jest zaszumione. Szum jest spowodowany impulsowym charakterem poboru prądu przez układy cyfrowe. Jeśli potrzebujemy zmierzyć (lub wygenerować) napięcie o przewidywanej dokładności, trzeba zastosować dodatkowe źródło napięcia odniesienia. Jeżeli napięcie na wyjściu przetwornika musi być ustawiane dokładnie, to trzeba zastosować precyzyjne źródło napięcia referencyjnego o wartości będącej jedną z potęg liczby 2 (1,024 V, 2,048 V itp.). Mikrokontroler PIC16F1769 ma wbudowany programowany moduł napięcia odniesienia mogący dostarczyć 1,024 V,

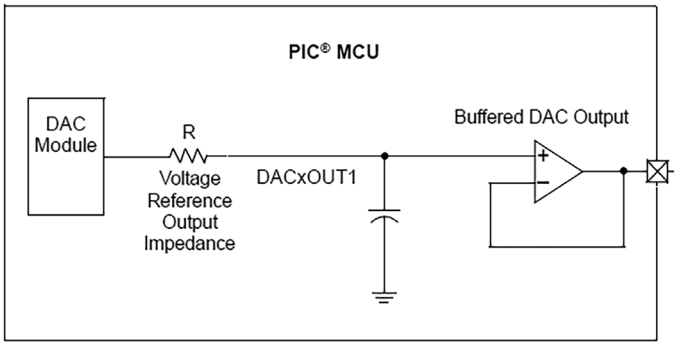


Rysunek 10. Konfiguracja wzmacniacza operacyjnego

2,048 V lub 4,096 V. Ponieważ napięcie zasilania wynosi +5 V, użyjemy napięcia 4,096 V. Konfigurowanie modułu FVR pokazano na **rysunku 12**. Zostaną zastosowane oba bufory FVR. Pierwszy dla przetwornika A/C, a drugi dla przetwornika C/A.

W trakcie opisywania modułu analogowego wspomniałem, że napięcia regulujące napięcie wyjściowe i próg zabezpieczenia prądowego będą się zmieniały w zakresie 0…4 V. Wszystkie obliczenia elementów w torach regulacji były dostosowane do takiego zakresu.

Dla napięcia referencyjnego *FVR_Buffer2* równego 4,096 V rozdzielczość wynosi 4,096/1024=4 mV. Napięcie z modułu FVR jest ustawiane z niepewnością 1…2%, a niepewność samego konwertera wynosi ±1,5 LSB. Dla naszego zastosowania jest to zupełnie wystarczające. Zapisanie do przetwornika wartości 1000 powoduje ustawienia napięcia na wyjściu wzmacniacza równego 4 mV×1000=4 V.



Rysunek 11. Buforowane wyjście przetwornika C/A

To upraszcza obliczenia, które musi wykonać mikrokontroler.

MCC generuje kilka funkcji związanych z obsługą przetwornika DAC1, z których użyjemy tylko dwóch: pokazanej na **listingu 1** *DAC1_Initialize(void)* oraz *DAC1_Load10bitInputData(uint16_t input10BitData)* z **listingu 2**.

Przyjąłem, że napięcie wyjściowe będzie zmieniane z rozdzielczością 0,5 V. Wiemy już, że maksymalne napięcie na wyjściu zasilacza wynosi 28 V i że odpowiadają mu 4 V napięcia regulującego z przetwornika C/A. Dla wygenerowania tego napięcia trzeba wpisać do rejestru przetwornika C/A liczbę 1000. Zatem zmiana na najmłodszym bicie odpowiada zmianie napięcia wyjściowego o 28 V/1000=0,028 V. Żeby zmienić napięcie o 0,5 V, trzeba wpisać do przetwornika wartość 0,5 V/0,028=17,85. Oczywiście, nie da się wpisać do rejestru wartości 17,85.

Listing 1. Inicjalizacja modułu przetwornika DAC1

```
void DAC1_Load16bitInputData(uint16_t input16BitData)
{
    //DAC input reference range should be 16 bit
    //Input data left justified.
    DAC1CON0bits.DAC1FM = 1
    //Loading 16bit data to DAC1
    DAC1REFL  = (uint8_t) input16BitData;
    DAC1REFH  = (uint8_t)(input16BitData >> 8);
    //Loading DAC1 double buffer data to latch.
    DAC1_DoubleBufferLatch();
}
```

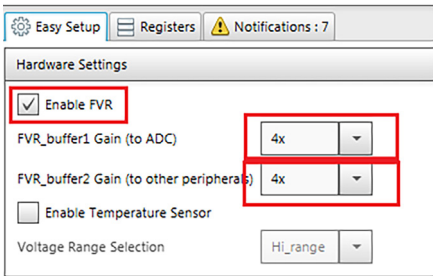
Listing 2. Zapisanie 10-bitowych danych do przetwornika DAC1

```
void DAC1_Load10bitInputData(uint16_t input10BitData)
{
    //DAC input reference range should be 10bit.
    //Input data right justified.
    DAC1CON0bits.DAC1FM = 0;
    //Loading 10bit data to DAC1
    DAC1REFL  = (uint8_t) input10BitData;
    DAC1REFH  = (uint8_t)(input10BitData >> 8);
    //Loading DAC1 double buffer data to latch.
    DAC1_DoubleBufferLatch();
}
/**
 *  Loads data from DAC buffer onto the DAC output
 */
#define DAC1_DoubleBufferLatch() \
    (DACLDbits.DAC1LD = 1)
```

Listing 3. Pętla ustawiania napięcia wyjściowego zasilacza

```
while(1)
{
    if(frtime==1) //cykliczny pomiar napięcia i prądu
    {
        DispNap(PLV,1);
        DispPr(PL,3);
        Frame_ms(500);
    }
    kod=GetEncoder();
    if(kod==KOD_IMP_UP)
    {
        volt=volt+17.85;// zmiana o 0,5V - zwiększenie
        if(volt>1000)
        volt=1000;
        DAC1_Load10bitInputData((unsigned int)volt);//zapis do DAC
        Delay_ms(80);
        DispNap(PLV,1);//zmierz i wyświetl napięcie
        DispPr(PL,3);//zmierz i wyświetl prąd
        Frame_ms(1000);//kolejny pomiar za 1 s
    }
    if(kod==KOD_IMP_DWN)
    {
        volt=volt-17.85;//zmiana o 0,5V - zmniejszenie
        if(volt<35.7)
        volt=35.7;
        DAC1_Load10bitInputData((unsigned int)volt);//zapis do DAC
        Delay_ms(80);
        DispNap(PLV,1); //zmierz i wyświetl napięcie
        DispPr(PL,3); //zmierz i wyświetl prąd
        Frame_ms(1000);//kolejny pomiar za 1 s
    }
    if(kod==KOD_IMP_ST)//przyciśnięcie oski  - ustawienie ograniczenia prądowego
    {
        Delay_ms(200) ;
        SetI();//ustawienie progu zabezpieczenia prądowego
    }
}
```

FVR

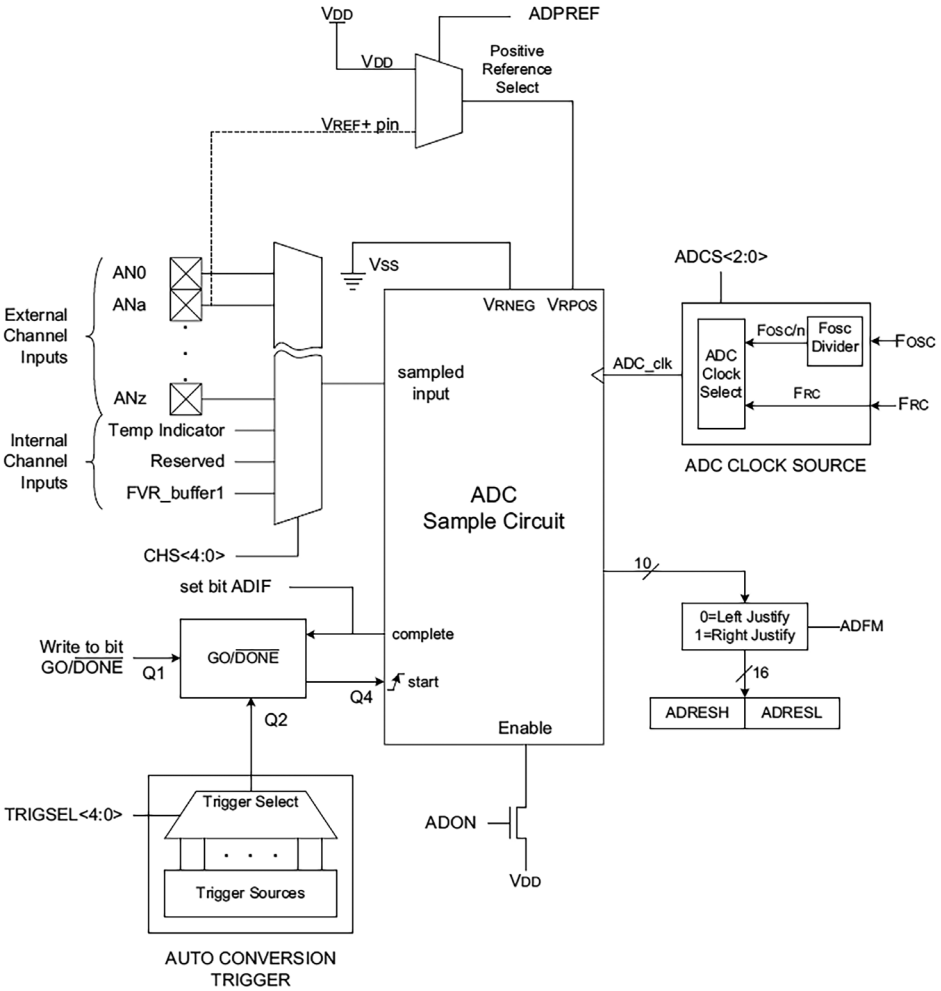


Rysunek 12. Konfigurowanie modułu FVR (źródła napięcia odniesienia)

Można wpisać 17 lub 18, ale w trakcie sekwencyjnej zmiany napięcia w każdym z kroków będziemy dodawać do poprzedniej wartości 17,85. Przy takim postępowaniu regulacja „nie rozjedzie się" po kilku krokach. Program wpisuje do przetwornika wartość zaokrągloną w dół, czyli dla 0,5 V wpisuje 17 i w teorii na wyjściu zasilacza będzie 17*0,028 =0,476 V, a dla 1 V (17,85×2)×0,028= 35,7×0,028, czyli 35×0,028=0,98 V. Gdyby była potrzebna większa dokładność, to trzeba by było zastosować przetwornik o większej rozdzielczości, na przykład 12-bitowy. Trzeba przy tym pamiętać o niepewnościach napięcia referencyjnego i samego przetwornika C/A.

Przetwornik 12-bitowy zastosowano w poprzedniej wersji zasilacza. Dlatego tam było możliwe ustawianie napięcia wyjściowego z rozdzielczością 0,1 V. Jednak ze względu na nieliniowości przetwornika i wewnętrznych wzmacniaczy operacyjnych musiałem zastosować programową korektę, aby napięcie było ustawiane z akceptowaną dokładnością. W praktyce różnica pomiędzy wyliczoną przez arytmetykę zmiennoprzecinkową wartością a wartością wynikającą z zaokrągleń i zastosowanej rozdzielczości 10-bitowej jest pomijalna dla urządzenia typu zasilacz warsztatowy. W zasilaczach z analogowymi miernikami wskazówkowymi napięcia wyjściowego precyzyjne ustawienie napięcia z rozdzielczością 0,5 V będzie o wiele trudniejsze, jeżeli w ogóle możliwe.

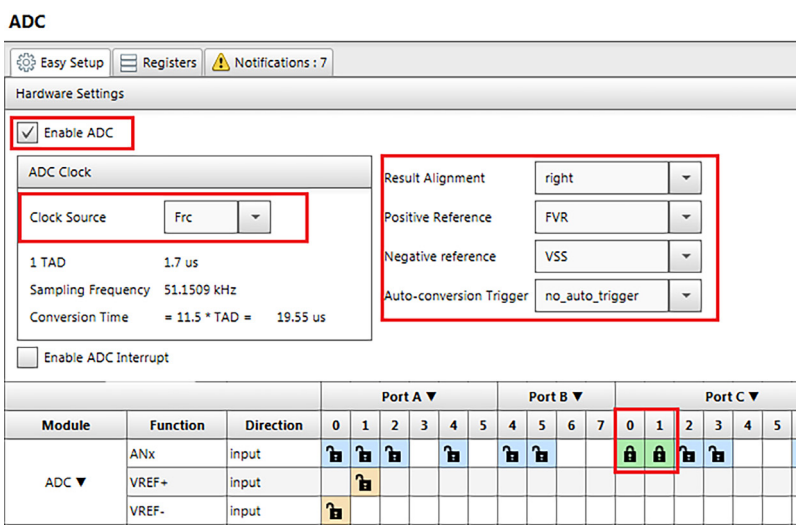
Na **listingu 3** pokazano pętlę główną ustawiania napięcia wyjściowego zasilacza. Po odczytaniu statusu procedury obsługi enkodera i wykryciu stanu obrócenia oski napięcie jest zwiększane lub zmniejszane



Rysunek 13. Schemat blokowy przetwornika A/C

o 0,5 V. Zmienna *volt* typu *float* zawiera wartość modyfikowaną o 17,85 przy każdym obrocie enkodera. Potem ta wartość jest zapisywana do przetwornika poprzez jawne rzutowanie typu *unsigned int*. Następnie program czeka 80 ms na ustabilizowanie się napięcia, mierzy oraz wyświetla napięcie wyjściowe i prąd obciążenia zasilacza. Jeżeli oska enkodera nie jest obracana, to pomiary napięcia i prądu są wykonywane automatycznie co 0,5 sekundy (wyzwalane za pomocą funkcji *Frame_ms(500)*). Ta funkcja wykorzystuje procedury obsługi przerwań od licznika wywoływane co 1 ms. Po odliczeniu 500 ms jest ustawiana flaga *frtime*, wykonywany pomiar napięcia i prądu, następnie flaga *frtime* jest zerowana i cykl się powtarza. Przyciśnięcie oski powoduje wejście do procedury ustawiania progu zabezpieczenia prądowego.

Pomiar napięcia i prądu jest wykonywany przez przetwornik A/C (**rysunek 13**).



Rysunek 14. Konfiguracja przetwornika A/C

W przetworniku zastosowano konwerter SAR. Prędkość konwersji (maksymalnie 100 kSa/s) jest dostosowana do wydajności 8-bitowego rdzenia. Próbkowane napięcie jest doprowadzane przez multiplekser do kondensatora wejściowego o małej pojemności. Po uruchomieniu konwersji jest ustawiany bit bit *GO* w rejestrze *ADCON* lub następuje zdarzenie wyzwalające automatycznie konwersję. Kondensator jest odłączany od wejścia i jest wykonywana konwersja analogowo-cyfrowa. Jej wynik jest zapisywany w 16-bitowym rejestrze ADRES. Źródłem taktowania przetwornika może być wewnętrzny oscylator RC (Frc) o stałej częstotliwości ok. 500 kHz. Taki tryb taktowania pozwala na pracę przetwornika w trybie ograniczonego poboru mocy (sleep), kiedy wszystkie przebiegi taktujące są wyłączone. Do taktowania można też użyć podzielonego przebiegu zegarowego rdzenia. Częstotliwość taktująca przetwornik nie była wyższa niż maksymalna częstotliwość taktowania przetwornika (Tad>=1 μs).

Przy napięciu wyjściowym +28 V na wyjściu dzielnika powinno panować napięcie +4 V, a identycznie jak na wyjściu układu pomiaru prądu przy natężeniu prądu obciążenia wynoszącym 3 A. Przy okazji omawiania modułu źródła napięcia referencyjnego FVR wspomniałem, że bufor *FVR_buf1* jest zaprogramowany na 4,096 V i jego napięcie będzie wykorzystywane jako napięcie referencyjne dla przetwornika analogowo- cyfrowego A/C.

Przetwornik A/C jest konfigurowany za pomocą MCC (**rysunek 14**). Odblokowujemy działanie przetwornika i wybieramy rodzaj taktowania z wewnętrznego oscylatora RC Frc. Jako napięcie referencyjne wybieramy FVR (wyjście FVR_buffer1 zostanie przyłączone automatycznie). 10-bitowy wynik wyjściowy jest w formacie „dosunięty do prawej”. Po skonfigurowaniu przetwornika trzeba jeszcze skonfigurować dwa wejścia analogowe AN4 (RC0)

do pomiaru napięcia i AN5(RC1) do pomiaru prądu. Konfiguracja modułu ADC jest wykonywana przez pokazaną na **listingu 4** funkcję *void ADC_Initialize(void)*. Inicjowanie konwersji analogowo-cyfrowej i odczytywanie wyniku ze wskazanego wejścia analogowego realizuje funkcja *adc_result_t ADC_GetConversion(adc_channel_t channel)*. Jej argumentem jest numer analogowego wejścia przetwornika, a rezultatem 10-bitowy wynik umieszczony w 16-bitowej liczbie z dosunięciem do prawej strony (**listing 5**).

Wyświetlenie pomiaru napięcia w woltach wymaga obliczeń. Napięciu +28 V odpowiada +4 V po podzieleniu przez dzielnik. Czyli inaczej mówiąc, napięciu +28 V odpowiada wynik konwersji równy 1000 dla napięcia referencyjnego 4,096 V. Zmiana napięcia na najmłodszym bicie odpowiada zmianie napięcia wyjściowego o 28 V/1000=0,028 V. Aby obliczyć wartość odczytaną z przetwornika na napięcie w woltach, trzeba ją pomnożyć przez 0,028. Procedura pomiaru napięcia rozpoczyna się od obliczenia średniej z kolejnych 10 pomiarów wykonywanych w odstępie co 2 ms. Potem średnia wartość odczytana z przetwornika jest mnożona przez 0,028, a właściwie mnożona przez 2,8 i dzielona przez 100 (**listingu 6**).

Do sformatowania wyniku pomiaru użyłem funkcji *sprintf* z biblioteki *stdio.h*. Znaki są zapisywane do bufora. Wskaźnik do tego bufora jest argumentem funkcji. Jeżeli napięcie jest mniejsze niż 10 V, to jest wyświetlane w postaci dwóch cyfr: jedności i cyfry po przecinku. Dla napięcia powyżej 10 V wyświetlane są trzy cyfry: dziesiątki, jedności i cyfra po przecinku. Funkcję wyświetlającą napięcie pokazano na **listingu 7**. Jej argumenty to napięcie w formacie zmiennoprzecinkowym oraz współrzędne początku wyświetlania na ekranie.

Analogicznie jest wykonywany pomiar natężenia prądu. Maksymalne napięcie, które występuje na wyjściu układu pomiarowego, powinno wynosić 4 V dla prądu o natężeniu 3 A. Większy prąd nie powinien płynąć, bo na taką wartość ustawiono domyślne, maksymalne zabezpieczenie prądowe. Funkcja pomiaru prądu rozpoczyna się od wykonania 10 pomiarów i obliczenia średniej. Potem następuje przeliczenie odczytanej wartości na ampery. Wartości 3 A odpowiadają 4 V, czyli liczba 1000 z przetwornika. Zmiana wartości na najmłodszym bicie odpowiada 3 A/1000= 0,003 A, więc aby obliczyć natężenie prądu, należy liczbę z przetwornika pomnożyć przez 0,003. W programie ta wartość jest mnożona przez 3 i dzielona przez 1000, jak na **listingu 8**.

Próg zabezpieczenia prądowego jest wykonywany przez funkcję *SetI(void)* – **listing 9**. Po naciśnięciu ośki impulsatora na ekranie w linii wyświetlającej nastawiony próg zabezpieczenia jest wyświetlany komunikat „<–Set”. Kręcąc ośką, zmieniamy nastawiony próg z krokiem 0,1 A. Nastawiana wartość po przeliczeniu jest wysyłana do przetwornika DAC2 i wyświetlana przez procedurę *DispIogr()*. Po odliczeniu 80 ms program mierzy i wyświetla napięcie na wyjściu oraz prąd pobierany z zasilacza. Pozwala to na bezpośrednią obserwację wpływu zmiany progu zabezpieczenia prądowego. Po naciśnięciu ośki funkcja kończy działanie i program przechodzi do pętli głównej, w której można ustawiać napięcie wyjściowe i jest wykonywany pomiar napięcia i prądu.

Listing 4. Inicjalizacja modułu przetwornika A/C

```
void ADC_Initialize(void)
{
    // set the ADC to the options selected in the User Interface
    // ADGO stop; ADON enabled; CHS AN4;
    ADCON0 = 0x11;
    // ADFM right; ADNREF VSS; ADPREF FVR; ADCS Frc;
    ADCON1 = 0xF3;
    // TRIGSEL no_auto_trigger;
    ADCON2 = 0x00;
    // ADRESL 0;
    ADRESL = 0x00;
    // ADRESH 0;
    ADRESH = 0x00;
}
```

Listing 5. Konwersja analogowo-cyfrowa

```
adc_result_t ADC_GetConversion(adc_channel_t channel)
{
    // wybór kanału pomiarowego
    ADCON0bits.CHS = channel;
    // włączenie modułu
    ADCON0bits.ADON = 1;
    // opóźnienie czasu akwizycji
    __delay_us(ACQ_US_DELAY);
    // Start konwersji
    ADCON0bits.ADGO = 1;
    // czekanie na zakończenie konwersji
    while (ADCON0bits.ADGO)
    {
    }
    // zwracanie wyniku pomiaru
    return ((ADRESH << 8) + ADRESL);
}
```

Listing 6. Pomiar napięcia i przeliczenie na napięcie w woltach

```
void DispNap(unsigned char x, unsigned char y)
{
    unsigned char i;
    double vol;
    vol=0;
    for(i=0;i<10;i++          )//wykonanie kolejnych 10 pomiarów
    {
        vol=vol+ADC_GetConversion(4);//konwersja w kanale analogowym AN_4
        Delay_ms(2);
    }
    vol=vol/10;//średnia z 10 pomiarów
    vol=vol*2.8;     //przeliczenie wartości odczytanej z ADC na wolty
    vol=vol/100;
    DispVolt(vol,x,y);  //wyświetlenie pomiaru w woltach
    volp=vol;}
}
```

Listing 7. Wyświetlenie napięcia na wyświetlaczu

```
void DispVolt(double vol,unsigned char x, unsigned char y )
{
    char buf[10];
    if(vol<10)  //dla napięcia poniżej 10V
        sprintf (buf,"%1.1fV „,vol);
        else sprintf (buf,"%2.1fV „,vol);
    PosLcd(x,y);
    DispLcdRam(buf);
}
```

Listing 8. Pomiar prądu zasilacza

```
void DispPr(unsigned char x, unsigned char y)
{
    unsigned char i;
    double pr;
    pr=0;
    for(i=0;i<10;i++)
    {
        pr=pr+ADC_GetConversion(5);
        Delay_ms(2);
    }
    pr=pr/10;
    pr=pr*3;
    pr=pr/1000;
    DispI(pr,x,y);
}

void DispI(double pr,unsigned char x, unsigned char y )
{
    char buf[10];
    if(pr<1)
    sprintf (buf,"%1.2fA  „,pr);
    else
    sprintf (buf,"%2.1fA  „,pr);
    PosLcd(x,y);
    DispLcdRam(buf);
}
```

Listing 9. Ustawianie progu zabezpieczenia prądowego

```
void SetI(void)
{
    unsigned char kod;
    PosLcd(15,4);DispLcd(„<-Set”);
    DispIogr(PL,4);
    while(1)
    {
        kod=GetEncoder();
        if(kod==KOD_IMP_UP)
        {
            iogr=iogr+30.8;
             if(iogr>925)
            iogr=925;
            DAC2_Load10bitInputData((unsigned int)iogr);
            Delay_ms(80);
            DispIogr(PL,4);
            Delay_ms(80);
            DispNap(PLV,1);
            DispPr(PL,3);
        }
        if(kod==KOD_IMP_DWN)
        {
            iogr=iogr-30.8;
            if(iogr<30.8)
            iogr=30.8;
            DAC2_Load10bitInputData((unsigned int)iogr);
            Delay_ms(80);
            DispIogr(PL,4);
            Delay_ms(80);
            DispNap(PLV,1);
            DispPr(PL,3);
        }
        if(kod==KOD_IMP_ST)
        {
            Delay_ms(200);
            PosLcd(15,4);DispLcd(„     „);
            DispIogr(PL,4);
            return;
        }
    }
}
```

Listing 10. Procedura odczytania słowa z pamięci Flash

```
uint16_t FLASH_ReadWord(uint16_t flashAddr)
{
    uint8_t GIEBitValue = INTCONbits.GIE;   // zapisanie stanu bitu GIE

    INTCONbits.GIE = 0;      // Zablokowanie przerwań
    PMADRL = (flashAddr & 0x00FF);
    PMADRH = ((flashAddr & 0xFF00) >> 8);
    PMCON1bits.CFGS = 0;     //
    PMCON1bits.RD = 1;       // zainicjowanie odczytu danych
    NOP();
    NOP();
    INTCONbits.GIE = GIEBitValue;       // Odtworzenie stanu bitu GIE
    return ((PMDATH << 8) | PMDATL);
}
```

Listing 11. Procedura zapisu wiersza do pamięci Flash

```
int8_t FLASH_WriteBlock(uint16_t writeAddr, uint16_t *flashWordArray)
{
    uint16_t  blockStartAddr=(uint16_t)(writeAddr&((END_FLASH-1)^(ERASE_FLASH_BLOCK-
SIZE-1)));
    uint8_t   GIEBitValue=INTCONbits.GIE;   // Save interrupt enable
    uint8_t i;

    // zapis Flash musi się rozpoczynać od początku wiersza
    if( writeAddr != blockStartAddr ) return -1;
    // Zapisanie stanu zezwolenia na  przerwania
    INTCONbits.GIE = 0;
    // sekwencja kasowania wiersza
    FLASH_EraseBlock(writeAddr);
    // sekwencja zapisu wiersza
    PMCON1bits.CFGS = 0;     // Deselect Configuration space
    PMCON1bits.WREN = 1;     // Enable wrties
    PMCON1bits.LWLO = 1;     // Only load write latches
    for (i=0; i<WRITE_FLASH_BLOCKSIZE; i++)
    {
        //8 młodszych bitów adresu
        PMADRL = (writeAddr & 0xFF);
        // 6 starszych bitów adresu
        PMADRH = ((writeAddr & 0xFF00) >> 8);
        // zapisanie danych
        PMDATL = flashWordArray[i];
        PMDATH = ((flashWordArray[i] & 0xFF00) >> 8);
        if(i == (WRITE_FLASH_BLOCKSIZE-1))
        {
            // start zapisu pamięci Flash
            PMCON1bits.LWLO = 0;
        }
        PMCON2 = 0x55;
        PMCON2 = 0xAA;
        PMCON1bits.WR = 1;
        NOP();
        NOP();
        writeAddr++;
    }
    // zablokowanie zapisu
    PMCON1bits.WREN = 0;
    // odtworzenie stanu zezwolenie na  przerwania
    INTCONbits.GIE = GIEBitValue;
    return 0;
}
```

Zmiana każdej nastawy jest zapamiętywana w nieulotnej pamięci Flash mikrokontrolera. Po każdym włączeniu zawartość zapisanych wartości jest odczytywana z pamięci Flash i zasilacz jest ustawiony tak jak przed wyłączeniem. Uwalnia nas to od każdorazowego ustawiania na nowo napięcia wyjściowego i zabezpieczenia prądowego po włączeniu zasilania.

Do zapamiętywania danych użytkownika w pamięci nieulotnej w rodzinach PIC16F Microchip stosował odrębną pamięć EEPROM. W rodzinie PIC16F1xxx zrezygnowano zapewne z powodu kosztów produkcji z pamięci EEPROM, ale w zamian dodano możliwość zapisu danych użytkownika w pamięci programu Flash. Na końcu przestrzeni pamięci programu wydzielono przestrzeń mieszczącą 128 słów, spełniającą funkcję pamięci EEPROM. Tych 128 słów ma podwyższoną liczbę cykli kasowanie/zapis. Kasowanie i zapisywanie danych odbywa się w porcjach (wierszach) mających 32 słowa 14-bitowe. Nie można zapisać jednego słowa – zapisują się od razu 32 słowa całego wiersza. Dlatego w praktyce dane przeznaczone do zapisywania w tej nieulotnej pamięci są przechowywane w buforze RAM mieszczącym 32 słowa 16-bitowe. Również adresowanie podlega pewnym ograniczeniom. Adres początkowy wiersza musi być wielokrotnością liczby 32. Jeżeli tak nie jest, to układy kontrolne zablokują zapis danych. Podobnie jak w pamięci EEPROM, musi być wykonana sekwencja kontrolna polegająca na zapisaniu danych do rejestrów w określonej sekwencji.

Gotowe funkcje zapisu i odczytu są generowane przez wtyczkę MCC. Na **listingu 10** pokazano funkcję odczytującą dane z pamięci Flash. Adres komórki jest argumentem procedury, a odczytana wartość jest zwracana w formie liczby 14-bitowej umieszczonej w rejestrze 16-bitowym. Funkcję zapisującą dane pokazano na **listingu 11**. Pierwszy argument to adres, który musi być początkiem wiersza, a drugi jest wskaźnikiem na bufor z 32 słowami zapisywanymi w jednym cyklu zapisu.

Tomasz Jabłoński, EP