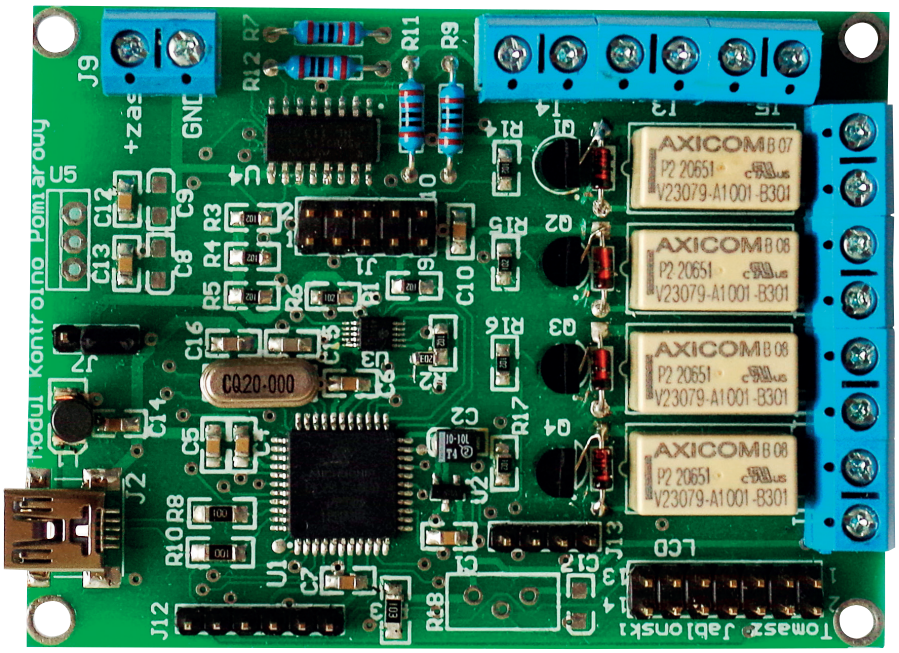
MKP – moduł kontrolno pomiarowy z interfejsem USB

AVT 5425

Przez elektroników często projektowane i wykonywane są urządzenia sterujące mające możliwość wykonywania pomiarów. Zależnie od realizowanych przez nie zadań, mogą być bardzo rozbudowane, z wbudowanymi algorytmami sterowania i wyposażone w możliwość zaawansowanej konfiguracji . Oczywiście, takie sterowniki mają olbrzymie możliwości, ale zwykle są też bardzo drogie. Istnieją jednak aplikacje, gdzie nie jest potrzebna rozbudowana funkcjonalność – do takich przyda się moduł opisywany w artykule.

Rekomendacje: *moduł przyda się w układach automatyki i kontrolno-pomiarowych wymagających możliwości włączenia/wyłączenia oraz regulacji napięcia.*



Opisywany moduł początkowo miał być wielokanałowym woltomierzem prądu stałego. Jednak w trakcie pracy nad założeniami do projektu został uzupełniony o kolejne funkcje. Ostatecznie, postanowiłem zaprojektować i wykonać moduł, który spełniałby 4 główne funkcje:

- Pomiaru napięcia stałego w 4 niezależnych kanałach.
- Sterowania załącz-wyłącz stykami 4 przekaźników.
- Odczytywanie 4 dwustanowych wejść: wejście wyłączone (obwód sygnalizacji otwarty) i wejście załączone (obwód sygnalizacji zamknięty).
- Ustawianie napięcia wyjściowego w 4 niezależnych kanałach.

Układy sygnalizacji i sterowania miały być kompletne, to znaczy – dla większości zastosowań nie będą konieczne dodatkowe układy dołączane do modułu. Sterowania są realizowane przez zwieranie lub rozwieranie izolowanych styków przekaźników o obciążalności ok. 1 A DC. Jeśli taka obciążalność byłaby za mała, można stykami przekaźników modułu sterować zasilaniem dodatkowych, zewnętrznych przekaźników mocy. Układy wejściowe są optoizolowane i przez dobranie rezystorów ograniczających prąd diody LED transoptorów można ustawić napięcie wejściowe w zakresie od 5…48 V DC. Dla wielkości analogowych – mierzonych napięć wejściowych i ustawianych napięć wyjściowych – może być konieczne dobudowanie zewnętrznych układów dodatkowych: zabezpieczeń przed przepięciami, buforów, dzielników wejściowych itp. Sam moduł umożliwia bezpośredni dostęp do wejścia analogowego przetwornika A/C i wyjścia analogowego przetwornika C/A.

Jako interfejs użytkownika miała być użyta aplikacja uruchamiana na komputerze PC z systemem Windows. Moduł i komputer łączą się poprzez interfejs USB. Napięcie zasilające może być pobierane z gniazda USB lub z zewnętrznego źródła zasilania.

Wybór mikrokontrolera – moduł USB

Aby sprostać zadaniom stawianym w założeniach projektowych potrzebny będzie mikrokontroler z wbudowanym przetwornikiem A/C, pewną liczbą wolnych linii GPIO oraz sprzętowym modułem USB. Dobrze, aby miał też wbudowany przetwornik C/A.

Znalezienie mikrokontrolera spełniającego pierwsze 3 warunki nie jest specjalnie

W ofercie AVT*

AVT-5425 A	AVT-5425 B
AVT-5425 C	AVT-5425 UK

Podstawowe informacje:

- 4 wejścia analogowe (0…5 V), 4 wyjścia analogowe (0…5 V).
- 4 wyjścia przekaźnikowe, 4 optoizolowane wejścia cyfrowe.
- Zasilanie +5 V DC ok. 150 mA (z zewnętrznego zasilacza 12 V DC lub z USB).
- Sterowanie za pomocą USB.

Dodatkowe materiały na CD lub FTP:

ftp://ep.com.pl, user: 28585, pass: 410ugxs3

- wzory płytek PCB
- karty katalogowe i noty aplikacyjne elementów oznaczonych w Wykazie elementów kolorem czerwonym

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

AVT-5368	Programowalny moduł przekaźników (EP 11/2012)
AVT-5353	Moduł przekaźników z interfejsem USB (EP 7/2012)
AVT-5233	3-kanałowy woltomierz z USB (EP 5/2010)
AVT-5182	Wielokanałowy rejestrator napięć (EP 4/2009)
AVT-925	Karta przekaźników na USB (EP 4/2006)
AVT-414	Uniwersalna karta portów na USB (EP 9/2005)
AVT-5086	Programowany 4-kanałowy komparator/woltomierz (EP 11/2002)

* Uwaga:
Zestawy AVT mogą występować w następujących wersjach:

AVT xxxx UK	to zaprogramowany układ. Tylko i wyłącznie. Bez elementów dodatkowych.
AVT xxxx A	płytka drukowana PCB (lub płytki drukowane, jeśli w opisie wyraźnie zaznaczono), bez elementów dodatkowych.
AVT xxxx A+	płytka drukowana i zaprogramowany układ (czyli połączenie wersji A i wersji UK) bez elementów dodatkowych.
AVT xxxx B	płytka drukowana (lub płytki) oraz komplet elementów wymienionych w załączniku pdf
AVT xxxx C	to nic innego jak zmontowany zestaw B, czyli elementy wlutowane w PCB. Należy mieć na uwadze, że o ile nie zaznaczono wyraźnie w opisie, zestaw ten nie ma obudowy ani elementów dodatkowych, które nie zostały wymienione w załączniku pdf
AVT xxxx CD	oprogramowanie (nieczęsto spotykana wersja, lecz jeśli występuje, to niezbędne oprogramowanie można ściągnąć, klikając w link umieszczony w opisie kitu)

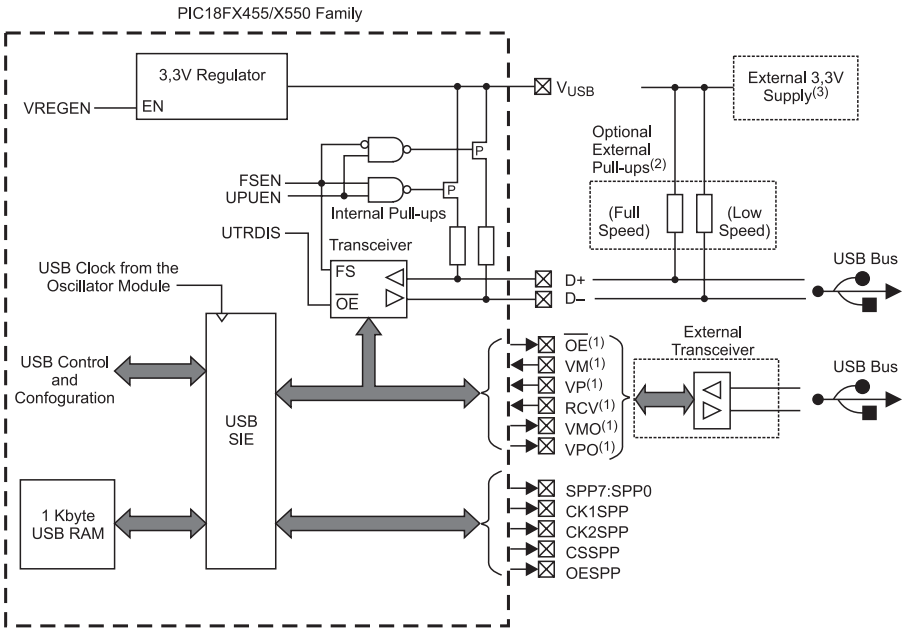
Nie każdy zestaw AVT występuje we wszystkich wersjach! Każda wersja ma załączony ten sam plik pdf! Podczas składania zamówienia upewnij się, którą wersję zamawiasz! (UK, A, A+, B lub C). *http://sklep.avt.pl*

Wykaz elementów

Rezystory:
R1…R6, R14…R17: 1 kΩ (SMD 0805)
R7, R9, R11, R12: 1 kΩ (THT, 0,6 W)
R8, R10: 47Ω R13: 47 kΩ (SMD 1206)
R18: 47 kΩ (potencjometr)
R19: 10 kΩ (SMD 0805)
Kondensatory:
C1, C3…C7, C10, C12…C14: 100 nF (SMD 0805)
C2: 1 µF/10 V (elektrolit., 3216)
C8: 4,7 µF/50 V (elektrolit., 3216)
C9, C17: 10 µF/10 V (elektrolit., 3216)
C15, C16: 27 pF (SMD 0805)
Półprzewodniki:
D1…D4: 1N4148
U1: PIC18F4550-I/PT (TQFP-PT44_N)
U2: MCP1541T-I/TT (SOT-23)
U3: MCP4728
U4: TLP281-4
U5: 7805
Q1…Q4: BC547 (tranzystor NPN)
Inne:
J1: goldpin 5 pin×2 rzędy
J2: złącze USB mini, do druku
J3…J10: złącza AR2/500
J7: goldpin 3 pin×1 rząd+zworka
J12: golpin 6 pin×1 rząd
J13: golpin 4 pin×1 rząd
L1: 10 µH/ (SMD 3528)
LCD: goldpin 7 pin×2 rzędy
RE1…RE4: przekaźnik z cewką na 5 V DC
X1: 20 MHz (rezonator kwarcowy)

trudne. Takie wyposażenie ma wiele mikrokontrolerów, od jednostek 8-bitowych po 32-bitowe. Z przetwornikiem C/A jest o wiele trudniej, ale są dostępne mikrokontrolery, które mają również 4 kanały wyjść C/A. Jednak zasoby to nie wszystko. W założeniach projektu jest połączenie przez interfejs USB, a to oznacza, ze trzeba będzie się zmierzyć z oprogramowaniem transmisji w urządzeniu (module MKP) i w hoście (komputerze PC). Głównie z tego ostatniego powodu zdecydowałem się na użycie mikrokontrolera PIC18F4550. Jest to dobrze znany, 8-bitowy „klasyk", powszechnie stosowany w układach interfejsem USB. Z powodu dość długiej obecności tego mikrokontrolera na rynku jest dostępnych wiele rozwiązań pomagających w tworzeniu oprogramowania transmisji USB. Najlepszym wsparciem jest firmowy stos USB i szereg działających przykładów, bezpłatnie publikowanych na stronach Microchipa.

Mikrokontroler PIC18F4550 jest 8-bitowcem dobrze wyposażonym w moduły peryferyjne a przy tym dość szybkim. Może być zasilany napięciem z zakresu 2,0…5,5 V. Z oczywistych względów, nas najbardziej będzie interesował wbudowany moduł USB. Jest on zgodny ze standardem USB V2.0 i może pracować w trybach *Low Speed* (1,5 Mb/s) i *Full Speed* (12 Mb/s). Sygnał symetryczny danych ze złącza USB jest dołączany bezpośrednio do wyprowadzeń *D+* (RC5) i *D-* (RC4). Można również wykorzystać układ zewnętrznego trasceivera



Rysunek 1. Schemat poglądowy modułu USB

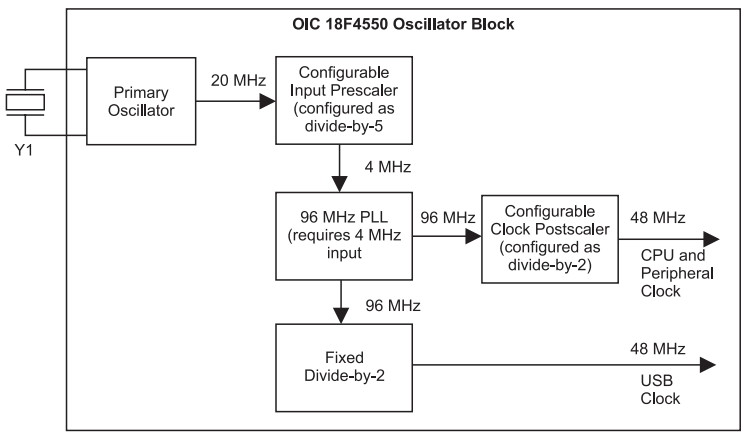
(**rysunek 1**). Interfejs USB mikrokontrolera ma wbudowaną pamięć RAM o pojemności 1 kB, mapowaną w bankach od 4 do 7 obszaru pamięci RAM GPR. Dostęp do niej ma blok SIE USB, który może w niej zapisywać i odczytywać dane bez udziału CPU. Jednocześnie, ten sam obszar CPU „widzi" jako zwykły obszar GPR. Jeżeli moduł USB jest włączony, to bank RB4 nie powinien być używany jako zbiór rejestrów ogólnego przeznaczenia.

W aplikacjach z USB mikrokontroler jest zasilany napięciem +5 V, ale obwody transceivera muszą być zasilane napięciem +3,3 V dostarczanym ze stabilizatora wewnętrznego lub za pomocą doprowadzenia VUSB. Źródło napięcia jest przełączane programowo. Konstrukcja modułu USB umożliwia też sprzętowe wsparcie przesyłania danych na zewnątrz poprzez interfejs do szybkiej komunikacji równoległej SPP (dwukierunkowa, równoległa, 8-bitowa szyna danych z sygnałami potwierdzenia).

Aby uzyskać obie dostępne prędkości transmisji, mikrokontroler wyposażono w rozbudowany system źródeł zegara taktującego oparty na układzie pętli PLL. W trybie *Low Speed* zegar taktujący modułem musi mieć częstotliwość 6 MHz, a w trybie Full Speed 48 MHz. Na **rysunku 2** pokazano system generowania sygnału zegarowego, gdy mikrokontroler jest taktowany oscylatorem stabilizowanym kwarcem o częstotliwości 20 MHz.

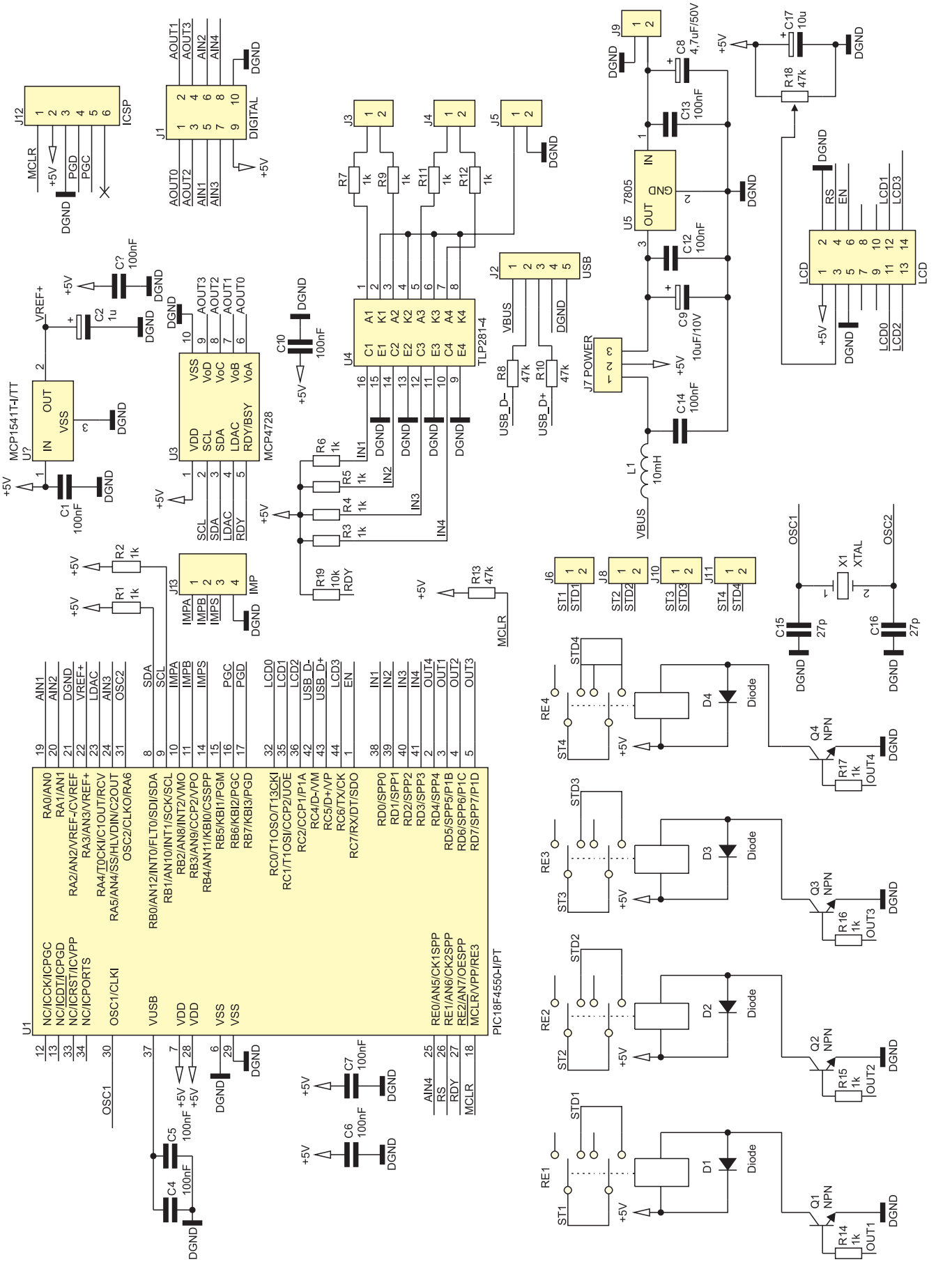
Schemat ideowy modułu

Schemat modułu pokazano na **rysunku 3**. Mikrokontroler PIC16F4550 (U1) jest taktowany generatorem stabilizowanym oscylatorem kwarcowym o częstotliwości 20 MHz (X1). Sygnały danych magistrali USB z gniazda mini A (J2), przez rezystory R8 i R10, doprowadzano do linii RC4/D- i RC5/D+. Napięcie wyjściowe z wewnętrznego stabilizatora +3,3 V musi być filtrowane kondensatorem ceramicznym o pojemności około 200 nF. Rolę filtra pełnią pojemności C4 i C5 dołączone do wyprowadzenia VUSB. Rezystory R1 i R2 dołączone do linii SDA i SCL magistrali I²C wykorzystywanej do komunikacji z przetwornikiem C/A MCP4728 (U3) realizują wymagane przez specyfikację



Rysunek 2. Generowanie zegara systemowego

magistrali podciąganie do plusa zasilania. Obwody wejść dwustanowych są dołączane do złącz J3 i J4. Do pinu 1 złącza J5 jest doprowadzony wspólny obwód katod diod poczwórnego transoptora TLP281-4 (U4). Może tu być podłączony izolowany od reszty układu minus napięcia sygnalizacyjnego. Przez zwarcie pinów 1 i 2 złącza J5 można połączyć ten obwód z masą układu. Rezysto-



Rysunek 3. Schemat ideowy modułu MKP

ry R3…R6 pracują w obwodach kolektorów tranzystorów transoptorów.

Cewki przekaźników RE1…RE4 są sterowane za pomocą tranzystorów Q1…Q4. Rezystory R14…R17 ograniczają prąd płynący w obwodzie bazy po ustawieniu linii sterujących OUT1….OUT4. Diody D1…D4 tłumią przepięcia powstające na cewkach przekaźników, których styki doprowadzono do złącz: J6, J8, J10, J11.

Mikrokontroler i reszta układu są zasilane napięciem +5 V. Układy z interfejsem USB najwygodniej jest zasilać napięciem +5 V ze złącza USB hosta pod warunkiem, że nie pobierają one więcej niż 100 mA, a w szczególnych przypadkach – 500 mA. My mamy do wyboru zasilanie z pinu VBUS lub z zewnętrznego napięcia uzyskanego ze stabilizatora 7805 (U5). Wyboru źródła zasilania dokonuje się przez przestawienie zworki na złączu J7.

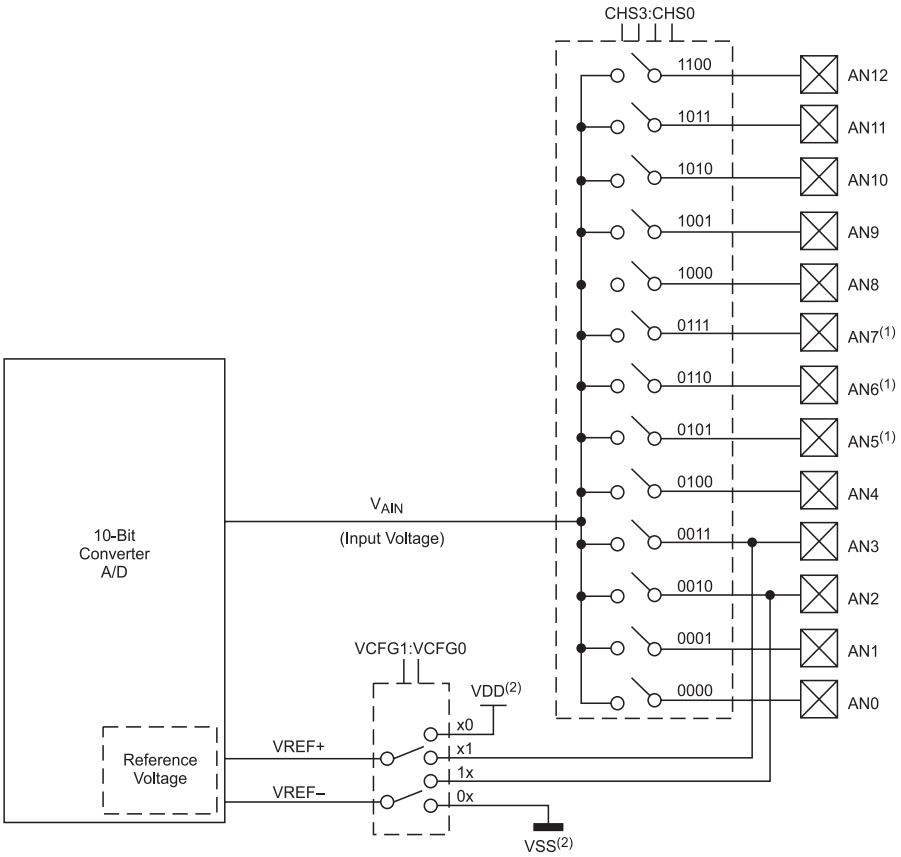
Zasilanie ze złącza USB hosta jest wygodne, ale to napięcie zasilające nie jest zbyt dobrej jakości, ponieważ jego źródłem jest komputer. Przez to charakteryzuje się ono dużą zawartością zakłóceń impulsowych. Żeby ograniczyć te zakłócenia zastosowałem prosty filtr LC składający się z dławika L1 i cewki C14.

Moduł musi wykonywać pomiary z wykorzystaniem wbudowanego w mikrokontroler U1 przetwornika A/C. Jeżeli przyjmiemy, że napięcie zasilania jest jednocześnie napięciem referencyjnym przetwornika, to takie pomiary będą obarczone sporym błędem. Aby temu zapobiec, zastosowałem źródło napięcia referencyjnego MCP1542T.

Na płytce modułu zostały też umieszczone złącza: LCD – dla alfanumerycznego wyświetlacza LCD, J13 dla impulsatora oraz dodatkowe elementy R18 i C17. W tej wersji modułu nie są one wykorzystywane.

Działanie układu – pomiar napięcia

Pomiar napięcia jest wykonywany przez wbudowany w mikrokontroler 10-bitowy przetwornik A/C z umieszczonym na wejściu analogowym multiplekserem pozwalającym na sekwencyjny pomiar napięcia z 13 wejść analogowych AN0…AN12. Wejścia analogowe są współdzielone z liniami cyfrowych portów PORTA, PORTB i PORTE. Warto wspomnieć, że po włączeniu mikrokontrolera wszystkie współdzielone linie są automatycznie ustawiane jako analogowe wejścia przetwornika. Źródłem napięcia referencyjnego może być napięcie zasilania mikrokontrolera, lub napięcie podawane na wyprowadzenie Vref- (AN2) i Vref+ (AN3) – **rysunek 4**. My tę możliwość wykorzystamy do doprowadzenia napięcia wyjściowego ze źródła napięcia referencyjnego MCP1541T. Napięcie z wyjścia jest połączone z wejściem



Rysunek 4. Schemat blokowy przetwornika A/C

AN3 zgodnie z rys. 4. Aby przetwornik używał napięcia z wejścia AN3 jako napięcia referencyjnego, trzeba go skonfigurować przez zapisanie bitów konfiguracyjnych VCFG1 i VCFG0 rejestru konfiguracyjnego ADCON1. Po wyzerowaniu VCFG1, minus zasilania układu referencyjnego jest łączony z masą, a po ustawieniu bitu VCFG0 plus zasilania układu referencyjnego z wyprowadzeniem AN3.

Po włączeniu zasilania wszystkie linie mikrokontrolera, które mogą być wejściami analogowymi są automatycznie ustawiane jako wejścia analogowe. Potrzebujemy wejść od AN0 do AN4, więc pozostałe wejścia trzeba przełączyć w tryb wejść cyfrowych (dwustanowych) za pomocą bitów PCFG3…. PCFG0 rejestru ADCON1.

Znając wartość napięcia referencyjnego można określić wartość otrzymywaną na wyjściu przetwornika:

$$Wartość\ wyjściowa = \frac{2^n \cdot Vin}{Vref}$$

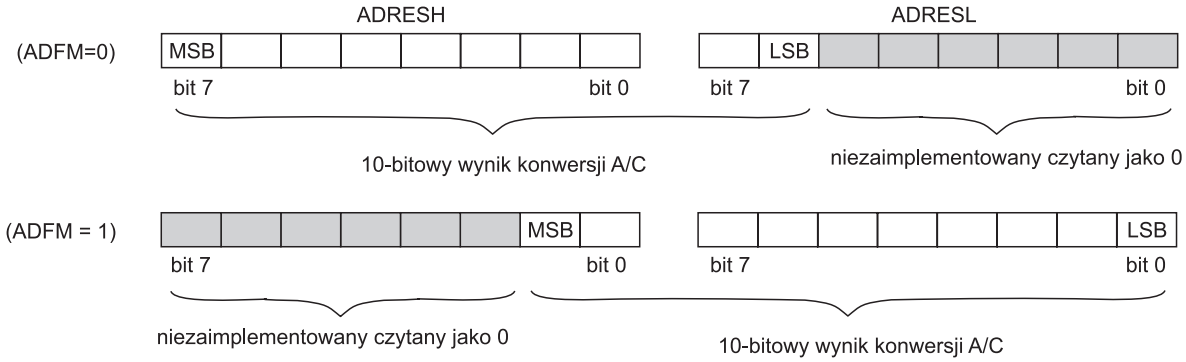
gdzie:

- n – rozdzielczość bitowa przetwornika,
- Vref – napięcie referencyjne,
- Vin – napięcie wejściowe.

Dla rozdzielczości 10-bitowej $2^n=2^{10}=1024$. Jeżeli przyjęlibyśmy Vref=1,024 V, to zakres pomiarowy wynosiłby 1,024 V, a zmiana wartości najmłodszego bitu odpowiadałaby zmianie napięcia na wejściu o 1,024 V/1024=0,001 V=1 mV. Uzyskujemy wtedy uproszczone i łatwe obliczenia przy przeliczaniu wartości cyfrowej z wyjścia przetwornika na mierzone napięcie. Jednak zakres pomiarowy 1 V to zbyt mało. Dlatego stosuje się napięcia referencyjne o wartości 2,048 V lub 4,096 V. Ta ostatnia wartość jest idealna dla mikrokontrolerów zasilanych napięciem +5 V. Zmiana na najmłodszym bicie dla przetwornika

Listing 1. Konfigurowanie przetwornika A/C

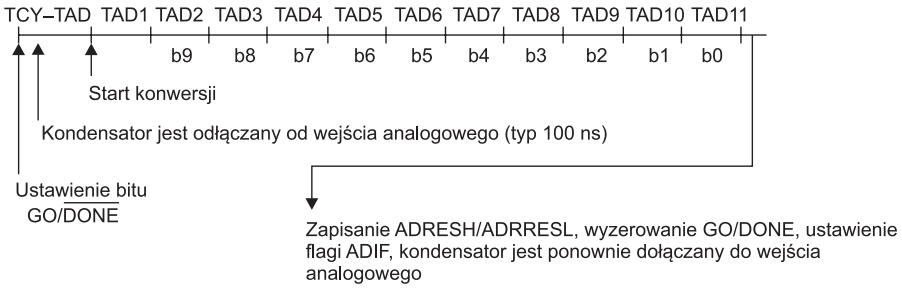
```
//inicjalizacja przetwornika ADC
//wejscia analogowe AN0, AN1, AN4, AN5
//napiecie referncyjne 4,096V na AN3
//wyjscie 10 bitów right justified
void InitADC(void)
{
  TRISA=0xff;//PORTA wszystkie linie wejściowe
  TRISEbits.RE0=1;//RE0 (AN4)wejście
//AN0...AN5 wejścia analogowe
  ADCON1bits.PCFG3=1;
  ADCON1bits.PCFG2=0;
  ADCON1bits.PCFG1=0;
  ADCON1bits.PCFG0=1;
//napięcie odniesienia
  ADCON1bits.VCFG1=0; //VREF-=GND
  ADCON1bits.VCFG0=1;//VREF+=AN3
  ADCON0=1;//ustawienie bitu ADON - włączenie modułu przetwornika
  ADCON2 = 0b10100101; //Right justified, 8TAD, FOSC/16
}
```



Rysunek 5. Formatowanie wyniku konwersji

Listing 2. Pomiar napięcia przetwornikiem A/C

```
//wybranie i odczytanie kanału przetwornika ADC
//kanały 0...3
//wyjscie 10 bitowy pomiar
int GetADC(char channel)
{
  if(channel==0)
  SelChanConvADC(ADC_CH0);//ustawienie wejścia analogowego
  if(channel==1) SelChanConvADC(ADC_CH1);
  if(channel==2) SelChanConvADC(ADC_CH4);
  if(channel==3) SelChanConvADC(ADC_CH5);
  ConvertADC();//start konwersji
  while(BusyADC());//czekaj na zakończenie konwersji
  return(ReadADC());//zwróć wynik konwersji
}
```

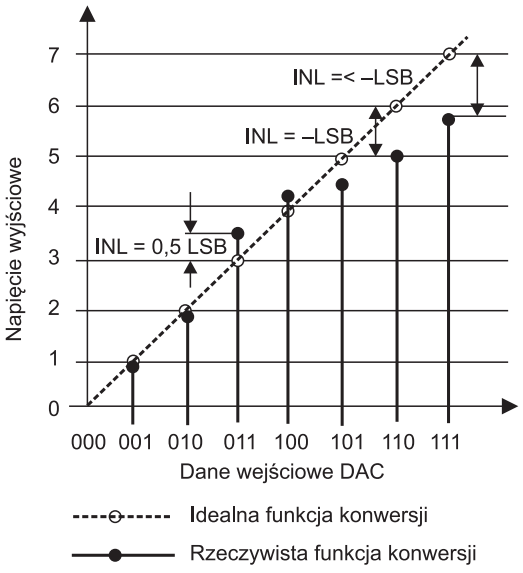


Rysunek 6. Przebieg konwersji A/C

10-bitowego odpowiada zmianie napięcia na wyjściu o 4,096 V/1024=4 mV.

W module zastosowano źródło napięcia odniesienia 4,096 V – układ scalony MCP1541T. Może on pracować w zakresie temperatury od -40°C, do +85° zapewniając dryft temperaturowy nie większy niż ±50 ppm/°C. Napięcie wyjściowe zmienia się w zależności od napięcia wejściowego ze współczynnikiem 350 µV/V w zakresie napięcia wejściowego od +4,3…5,5 V.

Moduł przetwornika trzeba przed użyciem włączyć (po restarcie jest automatycznie wyłączany) poprzez ustawienie bitu ADON w rejestrze ADCON0. Na **listingu 1** pokazano procedurę inicjalizacji modułu przetwornika A/C użytego w module MKP. Konwersja rozpoczyna się po ustawieniu bitu GO/!DONE w rejestrze ADCON0, ale przedtem trzeba ustalić, z którego wejścia będzie wykonywany pomiar przez zapisanie bitów CHS3…CHS0 rejestru ADCON0. Kondensator CHOLD (*Charge Holding Capacitor*) jest odłączany od wejścia analogowego i od tego momentu przetwornik mierzy napięcie na jego okładkach. Czas tego pomiaru (czas akwizycji) jest programowany za pomocą bitów ACQT2:0. Kiedy konwersja jest zakończona, to moduł przetwornika zeruje bit GO/DONE, ustawia flagę przerwania ADIF i zapisuje rejestry ADRESH i ADRESL. Sposób umieszczenie wyniku w rejestrach jest zależny od bitu ADFM z rejestru ADCON2 (**rysunek 5**).



Rysunek 7. Błędy konwersji przetwornika

Wyliczania pozwalające oszacować potrzebny czas akwizycji można znaleźć w dokumentacji mikrokontrolera. Na **rysunku 6** jest pokazany przebieg konwersji dla ACQT=000. Na **listingu 2** zamieszczono procedurę odczytywania kanałów przetwornika. Na podstawie wartości argumentu *channel* procedura biblioteczna *SelChanConvADC* przełącza multiplekser wejściowy przetwornika, tak aby napięcie było mierzone na wybranym wejściu. Funkcja *ConvertADC()* rozpoczyna pomiar przez ustawienie bitu *GO/!DONE*, a funkcja *BusyADC()* czeka na zakończenie pomiaru (wyzerowanie *GO/!DONE*). Po zakończeniu pomiaru, funkcja *ReadADC()* odczytuje rejestry *ADRESH* i *ADRESL* i zwraca ich zawartość w postaci słowa 16-bitowego.

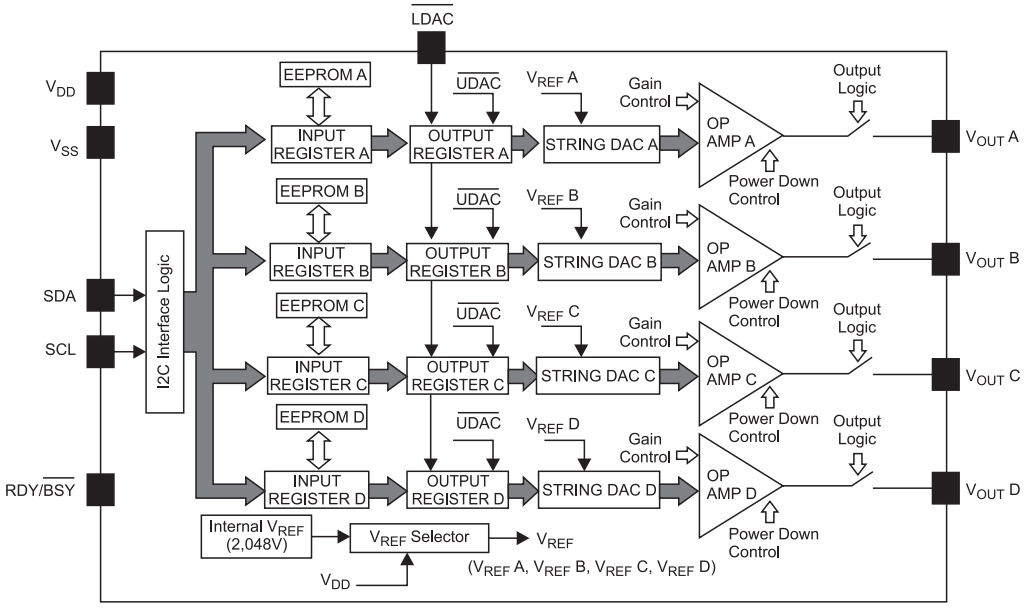
Ustawianie napięcia, przetwornik C/A

Mikrokontroler PIC18F4550 nie ma wbudowanego modułu przetwornika C/A. Dlatego – aby mieć możliwość ustawiania stałego napięcia wyjściowego – zastosowałem zewnętrzny przetwornik C/A typu MCP4728 produkowany przez firmę Microchip.

Układ MCP4728 to 4-kanałowy, 12-bitowy przetwornik C/A z szeregową magistralą I²C. Może być zasilany napięciem pojedynczym z zakresu 2,7…5,5 V. W strukturę układu jest wbudowane precyzyjne źródło napięcia referencyjnego 2,048 V. Przy rozdzielczości 12-bitowej zmiana wartości cyfrowej na najmłodszym bicie odpowiada

zmianie napięcia wyjściowego o Vref/2^{12}=2,048 V/2^{12}=0,5 mV. To są oczywiście wyliczenia teoretyczne. W rzeczywistości przetwornik charakteryzuje się pewną nieliniowością.

Na **rysunku 7** zostało pokazane odchylenie rzeczywistej wartości napięcia wyjściowego od linii prostej (parametr INL – *Integral Nonlinearity*). Oprócz tych błędów mogą występować błędy: offsetu, wzmocnienia, błędy pełnego zakresu (*full scale error*), przesłuchu pomiędzy kanałami DAC itp. O klasie przetwornika decyduje to, na jakim poziomie są te błędy. Dla MCP4728 producent podaje błąd INL na poziomie ±2LSB, błąd offsetu typowo 5 mV dla liczby wejściowej równej 0x000, błąd pełnego zakresu równy 0,4% pełnego napięcia wyjściowego.

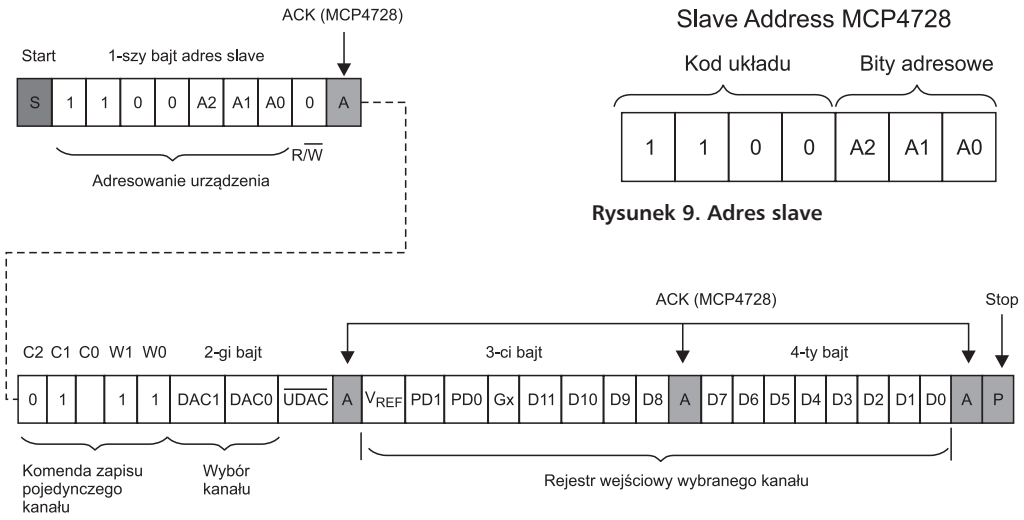


Rysunek 8. Schemat blokowy MCP4728

Dokładność napięcia wyjściowego zapewnia wbudowane, stabilne źródło napięcia odniesienia. Jak już wspomniałem, napięcie odniesienia wynosi 2,048 V, a maksymalne napięcie z wyjścia przetwornika jest równe 2,048 V (wynika z tego możliwość zasilania układu scalonego przetwornika napięciem minimalnym +2,7 V). My jednak zasilamy przetwornik napięciem +5 V z hosta USB i dobrze byłoby, gdyby to napięcie wyjściowe przetwornika A/C mogło być wyższe. Ma on możliwość użycia napięcia zasilania jako napięcia referencyjnego, ale tu takie rozwiązanie nie jest możliwe ze względu na wymagania odnośnie do jakości pomiarów. Na wyjściu każdego z przetworników jest umieszczony niskoszumny wzmacniacz rail-to-rail z programowanym wzmocnieniem równym 1 lub 2. Dla napięcia zasilającego +5 V±10% i dla Verf=2,048 V, współczynnik tłumienia tętnień zasilania PSSR jest równy -57 dB. Jeżeli tor analogowy będzie miał wzmocnienie równe 2, to maksymalne napięcie na wyjściu będzie miało wartość 4,096 V, czyli taką samą, jak maksymalne napięcie wejściowe w torze pomiarowym.

Schemat blokowy przetwornika pokazano na **rysunku 8**. Dane do konwersji z modułu obsługującego magistralę I²C trafiają do rejestru wejściowego *Input Register*. Po jego zapisaniu wartość jest automatycznie zapisywana do przypisanej mu komórki nieulotnej pamięci EEPROM. Jest to przydatna cecha, bo po włączeniu zasilania zawartość komórek EEPROM jest automatycznie wpisywana do rejestrów wyjściowych i na wyjściach analogowych pojawiają się takie napięcia, jakie były przed wyłączeniem zasilania. W trakcie normalnej pracy przetwornika przepisanie ich do buforów wyjściowych jest możliwe, gdy wyprowadzenie sterujące *!LDAC* lub bit *!UDAC* przesyłany z danymi po magistrali I²C są wyzerowane.

Każdy układ dołączony do magistrali I²C musi mieć swój unikalny, 7-bitowy adres slave. Adres MCP4728 składa się z 4-bitowego kodu układu i 3 bitów adresowych (**rysunek 9**). Zazwyczaj w układach z magistralą I²C stan bitów adresowych w adresie slave odpowiada poziomom na wejściach adresowych. MCP4728 nie ma takich wyprowadzeń, ale producent przewidział możliwość



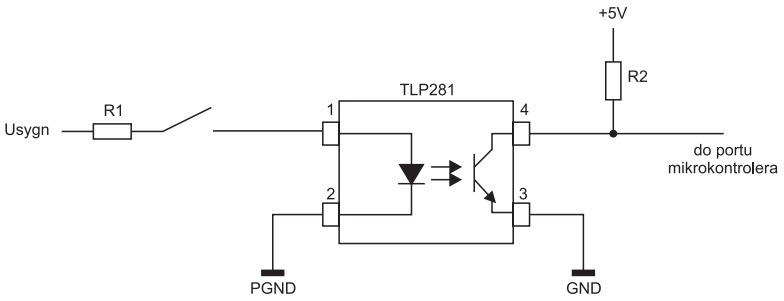
Rysunek 10. Sekwencja zapisu pojedynczego kanału

Rysunek 9. Adres slave

Bit	opis	
DAC1:DAC0	00 =Kanał A 01 =kanał B 10 =kanał C 11 =kanał D	Wybór kanału
!UDAC	1 = dane nie mogą być zapisane do rejestru wyjściowego 0=Dane są przepisane do rejestru wyjsciowego	Zezwolenie na wpis do rejestru wyjściowego
Vref	1= wewnętrzne Vref (2,048V) 0= Vref=Vdd	Wybór źródła napięcia referencyjnego
PD1:PD0	00 – normalna praca Pozostałe kombinacje stan Power down	Stan Power Down
Gx	0= Gx=1 1= Gx=2	Wybór wzmocnienia wzmacniacza wyjściowego

Rysunek 11. Bity sterujące pracą przetwornika

Listing 3. Zapisanie kanału przetwornika C/A
```
void SetDAC(unsigned char channel, unsigned int data)
{
  unsigned char pom,chan;
  channel&=3;
  chan=channel;
  StartI2C();  //sekwencja Start
  pom=SSPBUF;
  WriteI2C(0xc0);  //adres Slave R/W=0
  pom=0x58;  //kod komendy =0x58, oraz !UDAC=0
  channel<<=1;
  pom=pom|channel;  //zapisanie do kodu bitów wyboru kanału
  WriteI2C(pom);  //zapisanie 2-giego bajtu
  pom=(data>>8);  //4  starsze bity danej
  pom=pom|0x90;  //Vref=1, Gx=1, PD1: PD0=00
  WriteI2C(pom);  //zapisanie 3-ciego bajtu
  WriteI2C(data);  //zapisanie 4-tego bajtu -8 młodszych bitów
  StopI2C();
  while(!PORTEbits.RE2); //czekanie na koniec zapisu w pamięci eeprom
}
```



Rysunek 12. Układ sygnalizacji z optoizolacją

zmiany bitów adresu A2…A0 przez przeprogramowanie wewnętrznej pamięci EEPROM, w której są one zapisane. Służy do tego celu specjalna komenda i algorytm, który wymaga znajomości aktualnej wartości zmienianych bitów – jest to dokładnie opisane w dokumentacji. Układy mają domyślnie wyzerowane bity adresowe i jeżeli do magistrali jest dołączony jeden przetwornik, to nie ma potrzeby ich zmiany.

Jak już wiemy, przetwornik może pracować z wewnętrznym napięciem referencyjnym lub z napięciem referencyjnym równym napięciu zasilania. Można też zmieniać wzmocnienie wyjściowego wzmacniacza analogowego. Poza tym, trzeba jakoś kierować dane do poszczególnych kanałów przetwornika. Sterowanie tymi funkcjami odbywa się przez wysyłanie odpowiednich komend. Komendy sterujące mogą być wysyłane autonomicznie lub poprzedzać dane 12-bitowe. Pokażę to na przykładzie wykorzystywanej w projekcie modułu komendy zapisu pojedynczego kanału przetwornika – **rysunek 10**. Jako pierwszy jest wysyłany adres slave z wyzerowanym bitem R/W. Drugi bajt zawiera kod komendy oraz dodatkowe bity wyboru kanału, do którego są przesyłane dane i wspominany już bit !UDAC sterujący przepisywaniem danych z rejestrów wejściowych do wyjściowych. Cztery starsze bity trzeciego bajta sekwencji komendy zapisu zawierają kolejne bity sterujące: wyborem napięcia referencyjnego, stanu Power Down i wzmocnienia wzmacniacza wyjściowego. Cztery młodsze bity trzeciego bajta i czwarty bajt, to 12 bitów wartości wpisywanej do przetwornika. Dokładny opis bitów sterujących jest zamieszczony na **rysunku 11**.

W naszym projekcie musimy ustawić Gx=1 (wzmocnienie równe 2) oraz Vref=1 (napięcie z wewnętrznego źródła o napięciu 2,048 V). Na **listingu 3** pokazano procedurę zapisu kanału przetwornika C/A. Zapisywany kanał jest określony przez argument *channel*, a 12-bitowa dana jest zawarta w argumencie *data*. Procedura zapisu kończy się testowaniem w pętli stanu wyprowadzenia RDY/BSY dołączonego do linii PORTE2. Kiedy wyprowadzenie RDY/BSY jest wyzerowane, to oznacza, że trwa zapisywanie pamięci EEPROM i kolejne dane nie zostaną wpisane do rejestru. Zapisywanie trwa kilkadziesiąt milisekund i program obsługi przetwornika musi to uwzględniać. Mimo że MCP4728 obsługuje cały zestaw komend dokładnie opisanych w dokumentacji, to w praktyce okazało się, że dla naszej aplikacji wystarczyło obsłużenie tylko opisanej powyżej sekwencji zapisu pojedynczego kanału.

Listing 4. Odczytanie stanu linii portów
```
unsigned char GetBIN(void)
{
  unsigned char port,rel;
  port=PORTD;
  rel=PORTD;
  port=~port;  //zanegowanie wszystkich bitów
  port=port&0x0f;  //maska na 4 starsze bity
  rel=rel&0xf0;  //"odzyskanie "4 starszych bitów
  port=port|rel;
  return(port);
}
```

Odczytywanie poziomów wejść dwustanowych

Bezpośrednie połączenie styków sygnalizacyjnych do linii portów mikrokontrolera nie jest dobrym pomysłem. Na długich przewodach mogą występować zaburzenia impulsowe, które w przypadkach skrajnych mogą spowodować uszkodzenie mikrokontrolera. Dlatego – w celu ochrony mikrokontrolera – zwykle stosuje się optoizolację z wykorzystaniem transoptorów. Wtedy obwód sygnalizacyjny ma charakter prądowy, przez co jest bardziej odporny na zaburzenia. Na **rysunku 12** pokazano typowy układ z izolacją transoptorową. Obwód sygnalizacyjny jest odizolowany galwanicznie od mikrokontrolera, ale musi mieć swoje dodatkowe napięcie sygnalizacyjne *Usygn*. Rezystor R1 dobiera się zależnie od wartości *Usygn* i natężenia prądu wymaganego do zasilania diody LED transoptora, tak aby jego tranzystor uległ nasyceniu. Zazwyczaj przyjmuje się, że jest to od 10…20 mA. Przy spadku napięcia na diodzie transoptora o około 1,5 V i przy *Usygn=+12 V*, rezystancję R1 wyliczamy z prawa Ohma R1=10,5 V/10 mA=1050 Ω. Z typoszeregu wybieramy rezystor 1 kΩ o mocy co najmniej $P=I^2\times R=0{,}1$ W. W projekcie zastosowano rezystory metalizowane 1 kΩ o mocy 0,6 W. Wykonując takie obliczenia można dobrać rezystory dla innych wartości Usygn. Jeżeli nie przewidujemy dużego poziomu zaburzeń, to można połączyć masy PGND i GND, a jako USYGN użyć napięcia +5 V. Prądowy charakter wejścia sygnalizacyjnego powinien znacząco zredukować ewentualne zakłócenia. Przy dużym poziomie zakłóceń jest lepiej rozdzielić masy i użyć oddzielnego źródła *Usygn*.

Obwód z rys. 12 odwraca fazę sygnału. Kiedy na wejściu jest podawane napięcie sygnalizacyjne, to tranzystor transoptora nasyca się i na linii portu mikrokontrolera panuje poziom niski. Program odczytujący wejścia neguje poziomy odczytywane z linii RD0….RD3 portu PORTD po to, aby zachować naturalne odwzorowanie poziomów na

B7	B6	B5	B4	B3	B2	B1	B0
CM3	CM2	CM1	CM0	ARG3	ARG2	ARG1	ARG0

Rysunek 13. Kod komendy

wejściach. Odczytywanie stanu linii wejściowych realizuje procedura *GetBIN()* pokazana na **listingu 4**.

Sterownie przekaźnikami

Układ sterownia przekaźnikami nie wymaga specjalnego komentarza. Tranzystor sterujący w układzie wspólnego emitera wchodzi w nasycenie po ustawieniu linii portu. Wtedy na cewce włączonej w obwodzie kolektora pojawia się pełne napięcie +5 V i przekaźnik załącza się. Programowo sterowanie jest podzielone na 2 procedury: jedna wykonuje operacje załączenia, a druga wyłączenia przekaźnika. Tor sterowania jest określony przez argument *channel* (**listing 5**).

Po każdej operacji sterowania stan linii sterujących PD4…PD7 jest zapisywany w pamięci EEPROM po to, aby po włączeniu zasilania można było tak wysterować przekaźniki, jak były wysterowane przed jego wyłączeniem.

Komunikacja host – urządzenie

Wymiana informacji pomiędzy hostem (komputerem PC) a urządzeniem (modułem MKP) odbywa się w oparciu o protokół transmisyjny *master – slave*, gdzie *master* jest hostem, a urządzenie układem *slave*. W takiej organizacji urządzenie slave niczego nie przesyła bez komendy od mastera. Ma to tę zaletę, że umożliwia czytelną i jednoznaczną organizację wykonywania szeregu różnych poleceń: pytania o poziomy wejściowe, pytania o pomiary, wysyłanie komend sterujących i nastaw C/A. Spontaniczne, niewymuszone przesyłanie informacji od urządzenia do hosta jest możliwe, ale jest nieużywane, bo w pewnym stopniu komplikowałoby oprogramowanie hosta.

Podstawowym elementem protokołu jest 8-bitowy kod komendy pokazany na **rysunku 13**. Podzielono go na dwa pola: 4 starsze bity (CM3…CM0), to kod komendy, a 4 młodsze bity (ARG3…ARG0) to jej argument. Zestawienie komend pokazano na **rysunku 14**. Wymiana informacji rozpoczyna się od wysłania przez hosta kodu komendy. Komenda 0x10 jest rozszerzona o 2 bajty z 12-bitową wartością zapisywaną do przetwornika C/A. Po odebraniu kodu moduł MKP dekoduje go i wykonuje odpowiednią czynność. Po jej realizacji typowo jest odsyłane potwierdzenie w postaci kodu komendy z wyzerowanym argumentem, dla kodu 0x40 – dwóch bajtów z wartością pomiaru, dla kodu 0x60 – dwóch bajtów z ustawioną wartością przetwornika C/A. Odebranie kodu komendy przez hosta kończy sekwencję wymiany informacji przez interfejs USB. Na **rysunku 15** pokazano wymianę informacji dla komendy zapisującej dane do kanału B przetwornika C/A. Host

Listing 5. Procedury sterowania na załącz i na wyłącz

```
//steruj na załacz
void CtrON(char channel)
{
  char ctrl;
  ctrl=ReadEE(ADR_REL);
  switch (channel)
  {
    case 4:
      ctrl|=1;
      break;
    case 0:
      ctrl|=2;
      break;
    case 1:
      ctrl|=4;
      break;
    case 2:
      ctrl|=8;
      break;
  }
  PORTD=(ctrl<<4);
  WriteEE(ADR_REL, ctrl);
}

//steruj na wylacz
void CtrOFF(char channel)
{
  char ctrl;
  ctrl=ReadEE(ADR_REL);
  switch (channel)
  {
    case 4:
      ctrl&=(~1);
      break;
    case 0:
      ctrl&=(~2);
      break;
    case 1:
      ctrl&=(~4);
      break;
    case 2:
      ctrl&=(~8);
      break;
  }
  PORTD=(ctrl<<4);
  WriteEE(ADR_REL, ctrl);
}
```

Kod – 4 starsze bity	Funkcja	Argument 4 młodsze bity
0x10	Wyślij dane do przetwornika C/A. W argumencie kanał przetwornika na przykład 0x12 – wyślij dane do kanału C przetwornika C/A	Kanał A Kanał B Kanał C Kanał D
0x20	Załącz. W argumencie numer kanału liczony od zera np. 0x20 załącz kanał 1	0-kanał 1 1-kanał 2 2-kanał 3 3-kanał 4
0x30	Wyłącz. W argumencie numer kanału liczony od zera np. 0x33 wyłącz kanał 4	0-kanał 1 1-kanał 2 2-kanał 3 3-kanał 4
0x40	Zapytanie o pomiary. W argumencie numer kanału pomiarowego	0-kanał 1 1-kanał 2 2-kanał 3 3-kanał 4
0x50	Zapytanie o wejście cyfrowe	brak
0x60	Zapytanie o ustawienie przetwornika DAC	Kanał A Kanał B Kanał C Kanał D

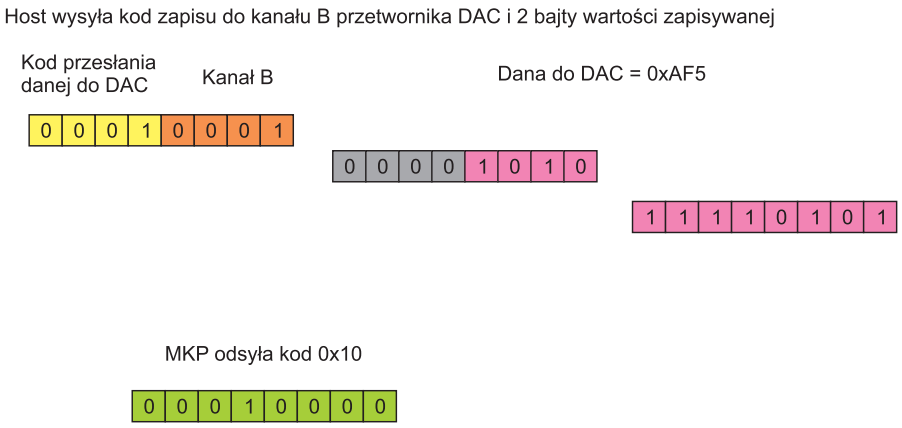
Rysunek 14 Zestawienie komend

wysyła kod 0x10 z numerem kanału (bajt 0x11) i 2 bajty zawierające 12-bitową wartość zapisywaną do rejestru przetwornika. Po wykonaniu zapisu, MKP odsyła kod 0x10 potwierdzający realizację komendy.

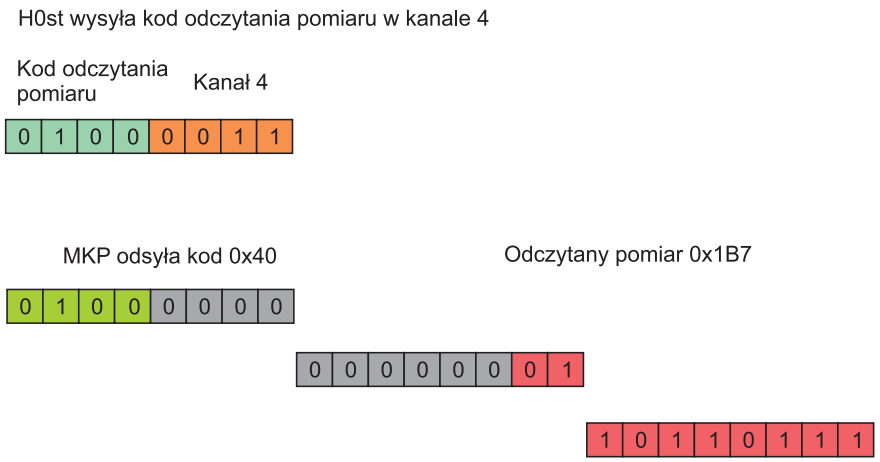
Komenda zapytania o pomiar w kanale 4 (**rysunek 16**) rozpoczyna się od wysłania rozkazu 0x40 (bajt 0x43). Moduł, po odebraniu tej komendy, wykonuje pomiar i odsyła kod komendy 0x40 i 10-bitową wartość odczytaną z przetwornika A/C.

Transfer danych USB

Moduł kontrolno-pomiarowy wykonano w oparciu o mikrokontroler PIC18F4550. Program zapisany w pamięci mikrokontrolera obsługuje wszystkie opisane już funkcje. Dla umożliwienia połączenia przez interfejs USB z hostem, MKP musi prawidłowo obsługiwać proces enumeracji i mieć zdefiniowane przynajmniej 2 endpointy: jeden wejściowy IN (z punktu widzenia hosta) dla danych przesyłanych z urządzenia do hosta i drugi



Rysunek 15. Wymiana informacji dla zapisu danej do przetwornika C/A



Rysunek 16. Wymiana informacji dla odczytu pomiaru w kanale 4

wyjściowy OUT dla danych przesyłanych z hosta (PC) do urządzenia (MKP).

Podczas enumeracji host pobiera informację o urządzeniu i przygotowuje go do wymiany informacji. Oprogramowanie urządzenia odbiera wtedy od hosta szereg standardowych żądań (*request*) i musi każde z nich zidentyfikować oraz odpowiedzieć hostowi. Oprócz odpowiedzi, urządzenie powinno wykonać czynności „zlecone" przez hosta w trakcie obsługi żądania. Kiedy system Windows wykonuje enumerację, to nie jest konieczne żadne dodatkowe działanie aplikacji użytkownika uruchomionej na komputerze PC. Po uruchomieniu system Windows musi zlokalizować plik INF, który zawiera nazwy i położenie (lokację) driverów USB. Jeżeli plik INF i drivery są dostępne oraz działają prawidłowo, to cały proces jest niewidoczny dla użytkownika. Po zakończeniu procesu enumeracji może się rozpocząć przesyłanie danych pomiędzy aplikacją użytkownika uruchomioną na komputerze (hoście) pod kontrolą systemu Windows i aplikacją uruchomioną w urządzeniu peryferyjnym.

Endpoint to najczęściej bufor w pamięci RAM zawierający specyficzne dane. Dane umieszczone w endpoincie mogą być danymi odebranymi lub czekającymi na wysłanie. Również host ma bufor z danymi do wysłania lub odebranymi, ale tego bufora nie nazywa się endopintem.

Specyfikacja USB definiuje endpoint jako unikalną część adresowaną urządzenia USB, która jest źródłem i przeznaczeniem danych w komunikacji pomiędzy hostem a urządzeniem. W naszym wypadku endpointy to obszary danych ze zdefiniowanymi zmiennymi dla 8-bitowego kodu komendy (*endpoint OUT* i *endpoint IN*) i 16-bitowej danej przesyłanej z hosta do MKP (*endpoint OUT* – dane do C/A) oraz 16-bitowej danej z pomiarami odczytanymi z A/C (*endpoint IN*). Z punktu widzenia USB, moduł MKP jest zdefiniowany jako urządzenie klasy HID i tak zgłasza się hostowi w procesie enumeracji. Ta klasa wykorzystuje transfer typu *Interrupt* i *Control*.

Transfer *Intetrrupt* jest przeznaczony dla urządzeń, które muszą cykliczne odbierać polecenia od hosta. Jest to jedyny transfer, który pozwala na przesyłanie danych w trybie *Low Speed*. W naszym wypadku będzie to cykliczne przesyłanie komend do MKP i odbieranie wysyłanych przez niego potwierdzeń. Transferu *Interrupt* można używać z każdą prędkością transmisji. Ogólnie urządzenia nie mają obowiązku obsługiwania transferu *Interrupt*, ale urządzenia klasy HID muszą to robić.

Transfer *Control* jako jedyny ma funkcje zdefiniowane przez specyfikację USB. Konfiguruje hosta do odczytywania informacji o urządzeniu, nadaje adres i wykonuje jego konfigurację i inne ustawienia. Ponadto, może wysłać żądanie podania informacji o producencie/wykonawcy urządzenia (*vendor*) oraz wysyła i odbiera dane. Wszystkie urządzenia dołączone do interfejsu USB muszą wspierać transfer *Control*.

Aplikacja sterująca

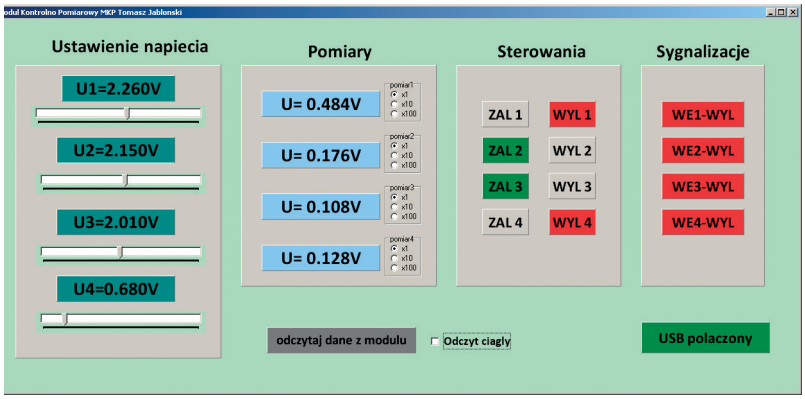
Aplikacja sterująca jest uruchamiana na komputerze PC. Jak wspomniałem, mechanizmy wbudowane w system operacyjny Windows pozwalają na automatyczne wybranie i zainstalowanie drivera obsługującego transfer danych przez USB z urządzeniem klasy HID. Jeżeli popatrzylibyśmy na model warstwowy interfejsu USB, to system operacyjny – łącznie ze sprzętowym portem USB – po zainstalowaniu drivera daje skonfigurowane i gotowe do pracy wszystkie warstwy, oprócz warstwy aplikacji. W przeciwieństwie do oprogramowania modułu MKP użytkownik nie musi się martwić o wiele rzeczy, bo robi to za niego system operacyjny z działającym driverem obsługującym klasę HID. Pozostaje „tylko" zaprojektowanie jakiegoś interfejsu graficznego i zapewnienie mechanizmu pozwalającego na wysyłanie danych do *endpointa IN* i odczytywanie z *endpointa OUT* modułu MKP.

Do projektowania takich interfejsów użytkownika przeznaczone są specjalistyczne programu narzędziowe do programowania obiektowego. Ja użyłem niezbyt już nowego, ale za to znanego i uznanego pakietu projektowego *C++ Builder 6* firmy *Borland* (obecnie *CodeGear*). Jest on przeznaczony do szybkiego tworzenia aplikacji (*Rapid Application Development*). Rzeczywiście, tworzenie nieskomplikowanych aplikacji w trybie graficznym polega na przenoszeniu na okno główne gotowych komponentów z palety dołączonej do *Buildera*. Potem można – za pomocą inspektora obiektów – zmieniać właściwości przypisane do obiektu. Wielką zaletą *C++ Builder* jest to, że przynajmniej na początku nie trzeba znać języka C++. Można się posługiwać starym poczciwym C, oczywiście – godząc się na brak wielu funkcjonalności oferowanych przez C++.

Projektując aplikację sterującą modułem MKP przyjąłem pewne założenia. Po pierwsze, ma być tak prosta, jak tylko się tylko da, ale nie kosztem funkcjonalności. Program nie korzysta z żadnych plików konfiguracyjnych. Wszelkie wartości początkowe są odczytywane z modułu MKP w czasie uruchamiania programu sterującego. Ponieważ MKP obsługuje cztery różne funkcje, to ekran podzielono na cztery obszary:

- Obszar ustawiania 4 napięć przetwornika C/A.
- Obszar wyświetlania 4 odczytywanych pomiarów z przetwornika A/C.
- Obszar sterownia 4 przekaźnikami.
- Obszar wyświetlania stanu wejść cyfrowych.

Okno programu sterującego pokazano na **rysunku 17**.



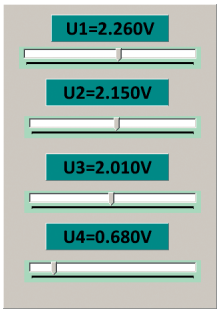
Rysunek 17. Ekran aplikacji sterującej

Ustawianie napięcia wyjściowego przetwornika C/A

Do tego celu jest przeznaczony obszar nazwany „Ustawienie napięcia", pokazany na **rysunku 18**. Każdemu kanałowi jest przypisany jeden element *TrackBar* z palety Win32. *TrackBar* to element graficzny w postaci suwaka przesuwanego za pomocą myszy. W oknie inspektora obiektu zdefiniujemy właściwości *Properties* → *Max=409* i *Properties* → *Min=0*. Wtedy – w trakcie przesuwania suwakiem – właściwość *Position* będzie się zmieniała w zakresie 0…409. Przy każdej zmianie położenia suwaka aplikacja wysyła do MKP wartość *Position* przemnożoną przez 10, a więc do odpowiedniego rejestru kanału przetwornika C/A jest wpisywana wartość z zakresu 0…4090 z krokiem co 10. Ta wartość po konwersji jest też wyświetlana w postaci cyfrowej jako napięcie ustawiane w kanale C/A, na panelu przypisanym do suwaka. Jest to dość wygodny sposób, bo zmiana położenia suwaka powoduje wyświetlenie ustawianego napięcia. Po ustawieniu napięć we wszystkich kanałach i zamknięciu aplikacji, nie jest pamiętane położenie suwaków, bo – jak wspomniano – program nie korzysta z żadnych plików konfiguracyjnych. Dlatego po ponownym uruchomieniu każdy z kanałów jest odpytywany komendą 0x60. Ponieważ MKP ma zapisane w pamięci EEPROM wartości wpisywane do rejestrów kanałów C/A, to po włączeniu zasilania potrafi odesłać do aplikacji ostatnio ustawioną wartość. Aplikacja po jej odczytaniu ustawia suwak w odpowiednim miejscu przez modyfikację właściwości *Position* i wyświetla ustawioną wartość napięcia na panelu przypisanemu do suwaka.

Na **listingu 6** pokazano przykładową obsługę zdarzenia zmiany położenia suwaka. Najpierw do zmiennej *ctrl* jest wpisywana komenda 0x10 powodująca zapis danej do kanału A przetwornika C/A. Potem do zmiennej *DAC_out* jest wpisywana wartość *Trackbar* → *Position* przemnożona przez 10. Funkcja *WriteAllReports()* wysyła wartość zmiennych *ctrl* i *DAC_out* przez USB do modułu MKP. Z przeprowadzonych już wyliczeń wynikało, że dla napięcia referencyjnego 2,048 V, wzmocnienia 2 i rozdzielczości 12-bitowej, ziarno wynosi 2 mV. Aby przekonwertować wartość binarną wysyłaną do C/A na wartość ustawianego napięcia, trzeba ją podzielić przez 1000. Wartość zmiennej *DAC_out* jest przepisywana do zmiennej pom typu *float*, aby było możliwe dzielenie zmiennoprzecinkowe. Potem wynik jest konwertowany na łańcuch ASCII z przecinkiem dziesiętnym za pomocą funkcji *sprintf()*.

Zawartość bufora *display* jest przekazywana do właściwości *Caption panelu21*. Po wysłaniu i wyświetleniu danych do rejestru kanału C/A procedura czeka na potwierdzenie z MKP o ustawieniu napięcia wyjściowego przetwornika. Moduł MKP jako potwierdzenie odsyła kod 0x10 zapisywany do zmiennej *pom_id*. Dane przesyłane z *endpointa OUT* modułu do komputera są odczytywane przez funkcję *ReadAllReports()*. Odczytywanie odbywa się w pętli nieskończonej. Wyjście z niej następuje w dwóch przypadkach: jeżeli do zmiennej *pom_i* zostanie wpisane 0x10 lub po wyzerowaniu zmienna *i*. Do zmiennej *i* jest zapisywana wartość 200. Maksymalnie przez tyle



Rysunek 18. Obszar ustawiania napięcia w kanałach C/A

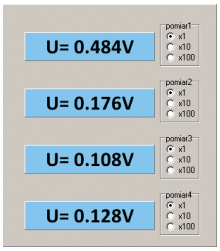
operacji odczytów endpointa czekamy na potwierdzenie. W praktyce, potwierdzenie przychodzi po 20…30 operacjach odczytu, ale przy większym obciążeniu magistrali może się wydłużyć. Jest to zabezpieczenie przed zablokowaniem się aplikacji w wypadku odłączenia USB w trakcie czekania na potwierdzenie komendy z MKP. Ten prosty mechanizm jest stosowany we wszystkich pętlach oczekiwania na potwierdzeni komendy i jego skuteczność została potwierdzona testami.

Listing 6. Obsługa zdarzenia zmiany położenia suwaka komponentu TrackBar1

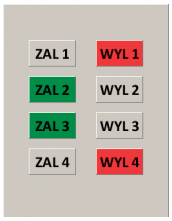
```
void __fastcall TMainForm::TrackBar1Change(TObject *Sender)
{
  int i;
  float pom;
  char display[]={"U1= 0000"};
  MKP[0]->ctrl->UnScaledValue = 0x10;// - komenda wyslij dane do DAC kanał A
  MKP[0]->DAC_out->UnScaledValue= TrackBar1->Position*10;//wartość położenia
suwaka *10
  MKP[0]->WriteAllReports();//wyslij przez USB wartość ctrl I DAC_out
  pom=MKP[0]->DAC_out->UnScaledValue; //wyświetl ustawiona wartość
  pom=pom/1000;
  sprintf(display+3,"%2.3f",pom);
  strcat(display,"V");
  Panel21->Caption=display;
  i=200;
  while(1)
  {
    if(MKP[0]->ReadAllReports()==0) break;
    if (MKP[0]->pom_id->UnScaledValue==0x10) break;
    if(--i==0)
    break;}
}
```

Listing 7. Fragment procedury wyświetlającej wynik pomiaru

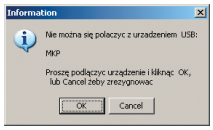
```
pom=  MKP[DevNum]->ADC_in->UnScaledValue;
pom=pom/250;
if(adc_chan==0)
{
  if(pomiar1->ItemIndex==0) sprintf(display+3,"%2.3f",pom);
  if(pomiar1->ItemIndex==1)
  {
    pom=pom*10;
    sprintf(display+3,"%3.2f",pom);
  }
  if(pomiar1->ItemIndex==2) {pom=pom*100;
  sprintf(display+3,"%3.1f",pom);}
  strcat(display,"V");
  Panel1->Caption=display;}
```



Rysunek 19. Obszar wyświetlania pomiarów



Rysunek 20. Obszar sterowania



Rysunek 21. Okno ostrzeżenia o baraku połączenia USB

Obszar wyświetlania pomiarów

Każdemu pomiarowi jest przypisany jeden panel, na którym jest wyświetlana wartość odczytywanych pomiarów i komponent RadioGroup przeznaczony do ustawiania mnożnika (**rysunek 19**). Wartość pomiaru odczytaną z endpointa urządzenia MKP jest zapisywana w zmiennej *ADC_in*. Liczba 10-bitowa reprezentująca wartość pomiaru jest dzielona zmiennoprzecinkowo przez 250 i poddawana konwersji na łańcuch ASCII z kropką dziesiętną, podobnie jak w wypadku wyświetlania napięcia zadawanego w kanałach przetwornika C/A (**listing 7**).

Zależnie od wyboru dokonanego w obiekcie *RadioGroup* (właściwość *ItemIndex*) wynik jest wyświetlany bezpośrednio, mnożony przez 10 lub przez 100. Jest to dodatkowe udogodnienie, gdy napięcie mierzone jest wstępnie dzielone przez 10 lub przez 100. Ustawienie *RadioGroup* nie ma wpływu na działanie modułu MKP – na przykład poprzez wysterowanie dodatkowego przekaźnika zmieniającego zakres pomiaru.

Listing 8. Wykonanie sterowania załącz w torze 3

```
void __fastcall TMainForm::Panel9Click(TObject *Sender)
{
  int i;
  MKP[0]->ctrl->UnScaledValue = 0x22;//wyślij komendę sterowania na załącz tor
3
  MKP[0]->WriteAllReports();
  i=200;
  while(1)
  {
    if(MKP[0]->ReadAllReports()==0)
    {
      Error_ctrl=1;//sygnalizacja błedu sterowania
      break;
    }
    if (MKP[0]->pom_id->UnScaledValue==0x20) break;  //odebrano potwierdzenie
    if(--i==0) break;  //wystąpił błąd przekroczenia czasu
  }
  if(MKP[0]->pom_id->UnScaledValue==0x20)
  {
    Panel9->Color=clGreen; //zamiana koloru panelu ZAL na zielony
    Panel10->Color=clBtnFace;}//zmiana koloru panelu ODL na szary.
  }
}
```

Listing 9. Fragment procedury odtwarzającej stan (kolor)paneli obszaru sterowania

```
if((MKP[0]->idx->UnScaledValue&0x20)==0x20)
{
  Panel5->Color=clGreen;
  Panel6->Color=clBtnFace;
}
else
{
  Panel6->Color=clRed;
  Panel5->Color=clBtnFace;
}
if((MKP[0]->idx->UnScaledValue&0x40)==0x40)
{
  Panel7->Color=clGreen;
  Panel8->Color=clBtnFace;
}
else
{
  Panel8->Color=clRed;
  Panel7->Color=clBtnFace;
}
if((MKP[0]->idx->UnScaledValue&0x80)==0x80)
{
  Panel9->Color=clGreen;
  Panel10->Color=clBtnFace;
}
else
{
  Panel10->Color=clRed;
  Panel9->Color=clBtnFace;
}
if((MKP[0]->idx->UnScaledValue&0x10)==0x10)
{
  Panel11->Color=clGreen;
  Panel12->Color=clBtnFace;
}
else
{
  Panel12->Color=clRed;
  Panel11->Color=clBtnFace;
}
```

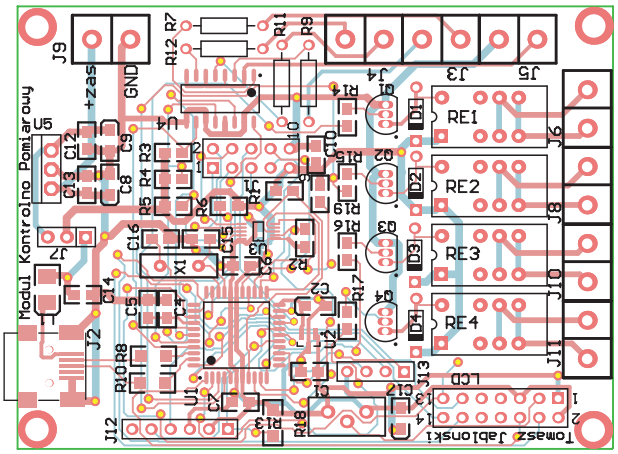
Listing 10. Sygnalizacja prawidłowego połączenia USB

```
void __fastcall TMainForm::HIDConnected(TObject *Sender, long NumConnected,
BSTR NewDevName)
{
  char info[]={"USB polaczony"};
  Panel26->Caption=info;
  Panel26->Color= clGreen;
  if (!JustStarting) HIDagent->OpenAllMatchingIntfs();
}
```

Listing 11. Sygnalizacja rozłączenia połączenia USB z MKP

```
void __fastcall TMainForm::HIDDisconnected(TObject *Sender, long NumConnected,
BSTR NewIntfName)
{
  char info[]={"USB rozlaczony"};
  Panel26->Caption=info;
  Panel26->Color= clRed;
}
```

Wymiana informacji pomiędzy komputerem a modułem MKP odbywa w konfiguracji *master – slave*. Dlatego aby zmierzyć napięcie w kanale pomiarowym A/C trzeba wysłać komendę odczytu z A/C – inaczej mówiąc – zażądać wykonania pomiaru. Można to zrobić na dwa sposoby. Pierwszy, to odczytanie wszystkich danych z modułu przez kliknięcie na panel „*Odczytaj dane z modułu*". Procedura związana ze zdarzeniem kliknięcia na ten panel wysyła jednokrotne zapytanie o wszystkie dane przesyłane z modułu MKP, czyli o pomiary we wszystkich 4 kanałach i poziomy na wejściach cyfrowych. Drugi sposób to zaznaczenie opcji „*Odczyt ciągły*". Wtedy żądanie pomiarów i poziomów na wejściach cyfrowych jest wysyłane cyklicznie. Jeżeli coś mierzymy i/lub zależy nam na ciągłym monitorowaniu stanu wejść cyfrowych, to powinniśmy zaznaczyć tę opcję. Wtedy mamy coś w rodzaju pracy on line. Czas odświeżania pomiarów i odczytu wejść jest zależny od częstotliwości wysyłania komend. W naszym wypadku jest to około 100 ms.



Rysunek 22. Schemat montażowy modułu kontrolno – pomiarowego

Obszar sterowania

Obszar sterowania umożliwia wykonanie sterowania włącz/wyłącz czterech przekaźników oraz sygnalizuje poziomy wyjść sterujących. Są one sygnalizowane kolorami paneli sterujących: stan załączony – zielony i stan wyłączony – czerwony (**rysunek 20**).

Sterowanie przekaźnikami odbywa się przez kliknięcie na panel odpowiadający żądanej akcji. Do modułu jest wysyłana wtedy komenda 0x20 dla operacji „załącz" lub komenda 0x30 dla operacji „wyłącz". Moduł MKP odbiera komendę, wykonuje i odsyła kod potwierdzenia. Procedura czeka na potwierdzenie i kiedy je odbierze, to zmienia kolor panelu, stosowanie do wykonywanej operacji sterowania. Zmiana koloru odbywa się przez zmianę właściwości *Color* panelu. Obsługa operacji sterowania na załącz toru 3 pokazano na **listingu 8**.

Ponieważ moduł MKP po włączeniu zasilania pamięta poprzednio wykonywane sterowania i inicjuje ich stan, to aplikacja po uruchomieniu odczytuje z modułu poprzez USB stany, w jakich są przekaźniki i odpowiednio koloruje panele sterujące. Fragment procedury odświeżającej stan paneli obszaru sterowania jest pokazany na **listingu 9**.

Sygnalizacja połączenia z MKP

Aplikacja automatycznie wykrywa i sygnalizuje czy MKP jest dołączony do portu USB. Po prawidłowo wykonanej inicjalizacji połączenia USB driver wywołuje procedurę *HIDConnected* pokazaną na **listingu 10**. Jest ona między innymi wykorzystana do modyfikowania wyglądu panel *Panel26*. Jeżeli *HIDConnected* jest wywołana, to kolor panelu zmienia się na zielono, a opis na „*USB połączony*".

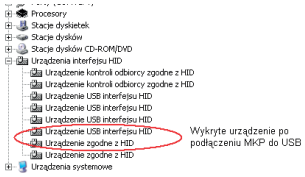
Sygnalizacja zerwania połączenia jest wykonywana przez funkcję HIDDisconnected, wywoływaną w momencie wykrycia przez driver braku połączenia USB. Kolor panelu *Panel26* zmienia się czerwony i opis na „USB rozłączany" (**listing 11**). W momencie uruchamiania aplikacji moduł MKP musi być zaprogramowany i dołączony do magistrali USB. Jeżeli tak nie jest, to jest wyświetlane ostrzeżenie (**rysunek 21**). Dalsze uruchomienie programu nie będzie możliwe do momentu podłączenia MKP i kliknięciu na „OK".

Montaż i uruchomienie modułu

Moduł MKP został zmontowany na dwustronnej płytce drukowanej, której schemat montażowy pokazano na **rysunku 22**. Montaż nie powinien sprawić większych trudności, może poza układem przetwornika C/A, który ma mały rozstaw wyprowadzeń.

Po przylutowaniu trzeba zasilić i zaprogramować mikrokontroler. Zasilanie +5 V w trakcie programowania może być doprowadzone poprzez złącze USB. Trzeba wtedy zewrzeć styki 1 i 2 złącza J7 *POWER*. Programator dołączamy do złącza J12 o nazwie ICSP. Ma ono wyprowadzenia zgodne z programatorem PIC Kit-3. Po zaprogramowaniu i dołączeniu układu do magistrali USB, system operacyjny Windows powinien wykryć moduł MKP jako urządzenie klasy HID i automatycznie zainstalować dla niego odpowiednie sterowniki. Jeżeli tak nie jest, to należy sprawdzić poprawność montażu, zaprogramowania mikrokontrolera i połączenie do portu USB. Na **rysunku 23** pokazano fragment okna *Menadżera* Urządzeń po dołączeniu prawidłowo działającego modułu MKP.

Po poprawnej instalacji sterowników możemy uruchomić aplikację sterującą i w trak-



Rysunek 23. Fragment okna Menadżera Urządzeń

cie jej uruchamiania nie powinno się pojawić okno ostrzeżenia z rys. 20. Jeżeli jeszcze panel kontroli połączenia będzie miał kolor zielony i będzie opisany jako „*USB połączony*”, to mamy pewność, że połączenie USB działa prawidłowo.

Uruchomienie działania poszczególnych funkcji modułu możemy rozpocząć od kanałów pomiarowych. W tym celu zaznaczamy okienko „*Odczyt ciągły*” i podajemy na wejścia AIN1…AIN4 napięcie stałe z zakresu 0…4 V. Wartość napięcia ustawionego w sprawdzanym kanale pomiarowym modułu powinna być równa napięciu wyświetlanemu w obszarze „*Pomiary*”, przy zaznaczonym mnożniku 1. Nietestowane wejścia pomiarowe najlepiej jest zewrzeć z masą.

Kolejną czynnością będzie sprawdzenie ustawiania napięcia w obszarze „*Ustawienie napięcia*” Do wyjścia AOUT0 dołączamy woltomierz i suwakiem kanału 1 ustawiamy napięcie wyjściowe. Mierzone napięcie powinno być równe napięciu wyświetlanemu w kanale 1. Takie same testy powtarzamy dla pozostałych kanałów.

Testowanie sterowania będzie polegało na klikaniu na panele *ZAL* i *WYL* kolejnych przekaźników i sprawdzeniu czy polecenia wysyłane z aplikacji są wykonywane przez moduł MKP.

Przy sprawdzaniu ostatniej funkcji z obszaru „*Sygnalizacje*” powinno być zaznaczane okienko „*Odczyt ciągły*”. Podanie napięcia na wejście sygnalizacyjne 1 powoduje zmianę koloru panelu z czerwonego na zielony i opisu z *WE1-WYL* na *WE1-ZAL*. Odłączenie napięcia sygnalizacyjnego spowoduje powrót do wyświetlania panelu w kolorze czerwonym z opisem *WE1-WYL*.

Pomiary i stan wejść sygnalizacyjnych jest przesyłany cyklicznie po zaznaczeniu okienka „*Odczyt ciągły*”. Jeżeli nie jest zaznaczone, to można je odczytać z modułu klikając na panel „*Odczytaj dane z modułu*”.

Podsumowanie

Przedstawiony tu moduł jest przykładem urządzenia mikroprocesorowego, które z powodzeniem wykorzystuje port USB do komunikacji z komputerem PC. Bardzo dużą pomocą w zaprojektowaniu MKP było wsparcie firmy Microchip. To przede wszystkim gotowy i dobrze opisany stos USB, ale też rozwiązania sprzętowe w postaci wbudowanego modułu USB i systemu generowania zegara taktującego. Ze względu na stawiane wymagania, zastosowałem bardzo dobrze znany mikrokontroler 8-bitowy. Przeprowadziłem też szereg prób z 32-bitowymi mikrokontrolerami z rodziny PIC32. W ich wypadku również nie ma problemu z połączeniem przez magistralę USB.

Mając do dyspozycji program narzędziowy do programowania obiektowego (tu *Builder C++*) możemy korzystać z dużych zasobów dostępnych we współczesnych komputerach. Jeżeli nie są istotne ograniczenia magistrali USB: małe odległości przesyłania danych czy trudności z izolacją galwaniczną komputera – urządzenie (może to mieć znaczenie dla pomiarów), to połączenie urządzenie mikroprocesorowe – komputer PC daje konstruktorowi olbrzymie możliwości. Jak już wspomniałem MKP wymaga rozbudowania o układy zabezpieczające przed przepięciami, dzielniki napięciowe, bufory itp. w torach analogowych zarówno wejściowych (A/C) jak i wyjściowych (C/A). Celowo nie zostały tutaj zaprojektowane, by można było je dostosować do konkretnych wymagań aplikacji.

Tomasz Jabłoński, EP