Wielokanałowy optoizolowany przetwornik A/C

Przetwarzanie analogowo – cyfrowe przy dzisiejszej ofercie rozmaitych przetworników stało się tematem w znacznym stopniu banalnym. Często wystarczy wybrać odpowiedni model mikrokontrolera wyposażony w przetwornik A/C aby szybko i sprawnie zrealizować pomiar wielkości analogowej. Nadal jednak kłopotliwe jest obsługiwanie dużej liczby kanałów pomiarowych. Rozwiązujemy to albo budując każdorazowo docelowy układ z multipleksowaniem wejść, albo składając system z jednostki centralnej i podrzędnych modułów pomiarowych. Prezentowany projekt jest rozwiązaniem pośrednim. Jest to niezależny 32-kanałowy moduł 12-bitowego przetwornika A/C, wyposażony w optoizolację wejść analogowych oraz uniwersalny interfejs odczytowy zgodny ze standardem I²C. Pozwala to na zastosowanie modułu w wielu rozmaitych systemach akwizycji danych.

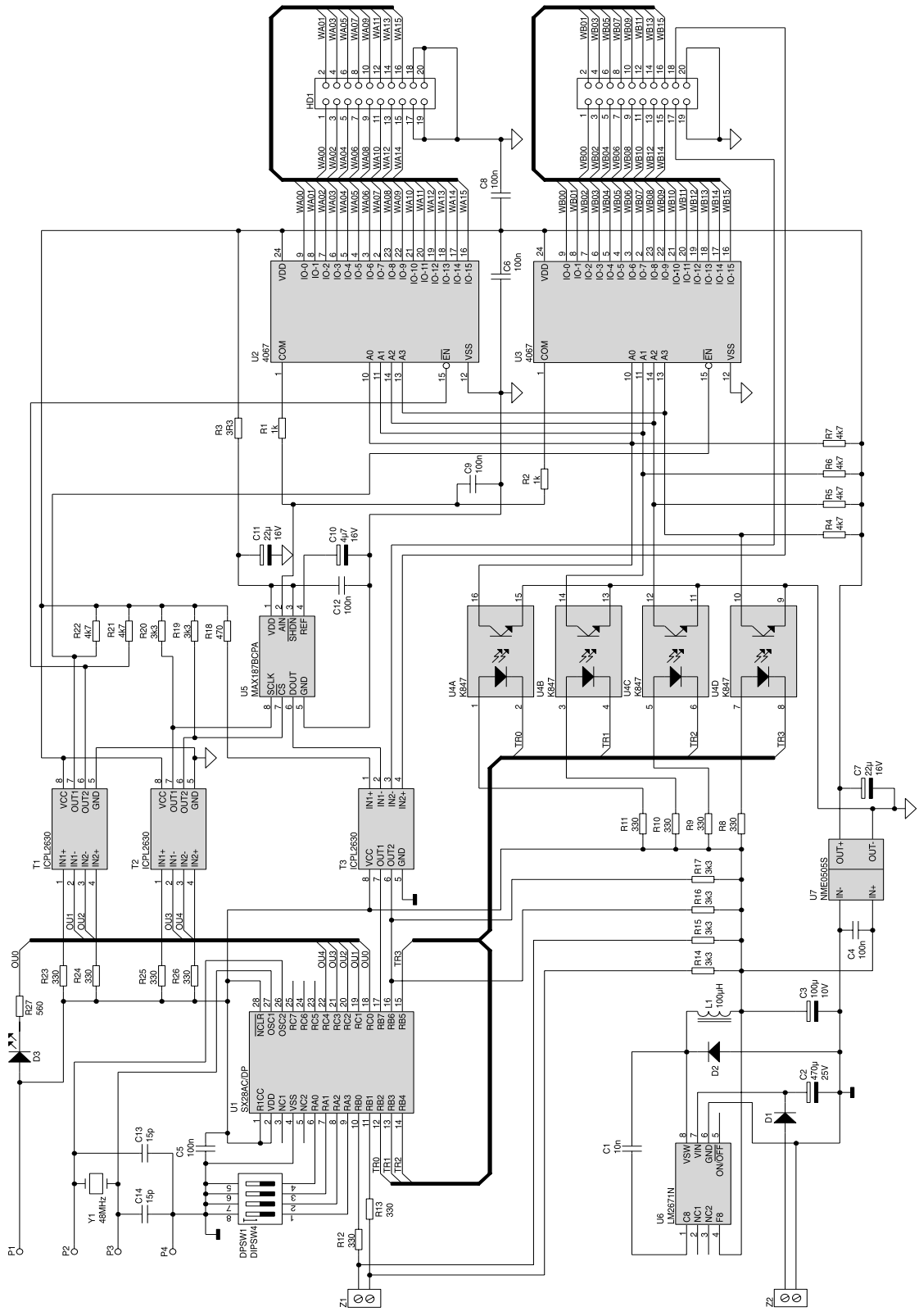
Rekomendacje: *polecamy wszystkim projektantom lubiącym rozwiązania alternatywne i nieco nietypowe, a także Czytelnikom zainteresowanym nowoczesnymi systemami akwizycji danych analogowych.*

Opis układu

Schemat elektryczny modułu przedstawiono na **rys. 1**. Zasada jego działania nie jest skomplikowana: dwa 16-kanałowe multipleksery analogowe z optoizolowanymi liniami sterującymi pozwalają na wybór jednego z 32 wejść napięciowych wyprowadzonych na złącza szpilkowe HD1 i HD2. Wybrany sygnał dociera na wejście jednokanałowego 12-bitowego przetwornika A/C poprzez rezystory R1 i R2 eliminujące ewentualne krótkotrwałe konflikty poziomów w chwili przełączania pomiędzy U2 i U3. Przetwornik U5 jest wyposażony w interfejs szeregowy, co pozwala w prosty sposób zrealizować optoizolowany tor odczytu (według przykładu przedstawionego w nocie katalogowej). Posiada on także wbudowane źródło napięcia referencyjnego 4,096 V. Taki zespół może być sterowany praktycznie dowolnym mikrokontrolerem. W tym projekcie zastosowano stosunkowo mało popularny układ SX28AC, między innymi właśnie w celu praktycznego wypróbowania go. Mikrokontroler dokonuje cyklicznie odczytu kolejno wszystkich kanałów wejściowych oraz realizuje programowy interfejs I²C *Slave*, za pośrednictwem którego przesyła żądane wartości do urządzenia *Master*.

Jako optoizolator stosunkowo powoliprzełączanych linii adresowych A0...A3 multiplekserów służy poczwórny transoptor U4 o całkowicie przeciętnych parametrach. Natomiast w interfejsie szeregowym przetwornika A/C (U5 – MAX187) użyto szybkich transoptorów T2 i T3 (zastosowanie szybkiego T1 do przełączania wejść EN multiplekserów jest nadmiarowe – wynikło z będących do dyspozycji zapasów). Odseparowane galwanicznie zasilanie części analogowej modułu zapewnia przetwornica DC/DC U7. Urządzenie – ze względu na znaczny sumaryczny pobór prądu – jest zasilane za pomocą stabilizatora impulsowego *step-down* (U6 oraz elementy towarzyszące), co umożliwia użycie dość szerokiego zakresu napięć zasilających (projektowo 8...15 V) bez konieczności stosowania dużego radiatora. Dioda D3 ma charakter kontrolny – sygnalizuje pracę modułu i oddaje duże usługi przy uruchamianiu.

Poczwórny DIP-switch dołączony do linii wejściowych mikrokontrolera pozwala na dowolne ustawienie adresu *slave* modułu (moż-

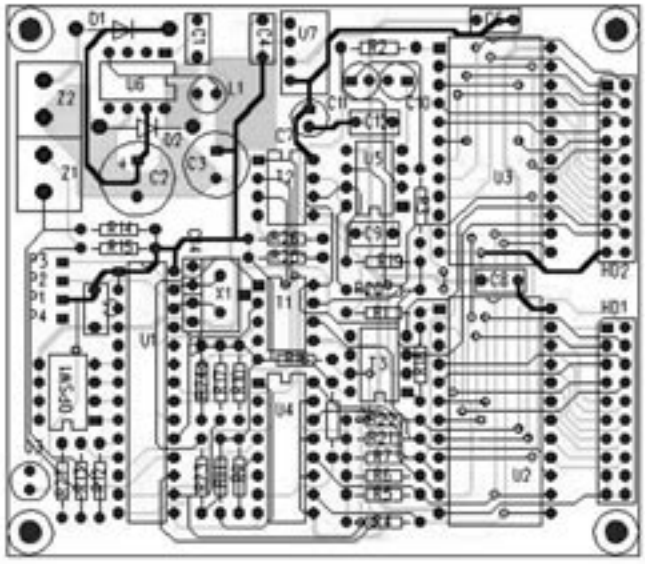


Rys. 1. Schemat elektryczny modułu przetwornika

na więc obsłużyć maksymalnie 16 * 32 = 512 wejść pomiarowych). Złącze szpilkowe P1...P4 służy do podłączenia programatora – mikrokontroler SX28 posiada interfejs ISP i kod programu wpisujemy po zamontowaniu go w układzie.

Kilka słów o rodzinie mikrokontrolerów SX

Mikrokontrolery SX są produkowane przez firmę Ubicom (dawniej Scenix). Są one udoskonaloną i przyśpieszoną wersją popularnych procesorów PIC16C5x. Zgadzają się wyprowadzenia obydwu rodzin (dla wersji 18 i 28), rejestry i banki RAM, pojemność pamięci programu (2048 słów 12-bitowych dla SX28), kody instrukcji i szybkość ich wykonywania w trybie kompatybilności. Różnice obejmują:



Rys. 2. Rozmieszczenie elementów na płytce

– programowanie szeregowe przez wyprowadzenia oscylatora (nie są zajmowane żadne linie I/O),
– zapis kodu programu w pamięci Flash (o trwałości 10000 przeprogramowań),
– wbudowany w każdą kostkę mechanizm wspomagający debugowanie, który umożliwia realizację funkcji ICE bez żadnych dodatkowych urządzeń (używany jest ten sam programator SX-Key podłączany do wyprowadzeń oscylatora),
– dodatkowe mnemoniki dla ułatwienia nawigacji wśród banków RAM i stron Flasha,
– wprowadzenie 4-poziomowego potokowania (*pipelining*), które pozwala w trybie Turbo na 4-krotne przyspieszenie wykonania programu,
– znaczne przyśpieszenie taktowania – częstotliwość zegarowa może wynosić maksymalnie 75 MHz,
– dostępne są dodatkowe bity konfiguracyjne kostki (*fuses*),
– ośmiopoziomowy sprzętowy stos,
– automatyczne zachowywanie kontekstu podczas przerwań (w dodatkowych rejestrach niewidocznych dla programu),
– wbudowany wewnętrzny generator sygnału zegarowego o częstotliwości od 31 kHz do 4 MHz,
– wszystkie linie I/O są dowolnie konfigurowalne (wejście/wyjście, włączanie podciągnięcia) i mają symetryczną wydajność prądową 30 mA.

Dodatkowe ułatwienia aplikacyjne to:
– wbudowany watchdog,
– wbudowany układ zerujący mikrokontroler po włączeniu zasilania,
– możliwość zerowania CPU przy spadku napięcia (*brownout*),
– tryb obniżonego poboru mocy z wybudzeniem zboczem na wybranej linii portu B.

Mikrokontroler SX oferuje całkiem odmienne podejście do konstrukcji i sposobu programowania niż wiele innych mikrokontrolerów. Jest praktycznie pozbawiony często obecnie spotykanych sprzętowych peryferiów (typu SPI, UART, I²C, PWM), natomiast jego szybkość działania pozwala większość tego typu usług i protokołów realizować na drodze programowej. Dotyczy to zwłaszcza wszelkich komunikacji szeregowych – dlatego też serię SX nazywa się również procesorami komunikacyjnymi. Rozwiązanie to producent określił mianem wirtualnych peryferiów (*Virtual Peripherals*) i opracował oraz udostępnił szereg gotowych procedur. Zasoby VP są znaczne (łącznie ze stosem TCP/IP). Obecnie jednak Ubicom promuje zupełnie nowe linie produktów i wsparcie dla SX na firmowej witrynie zostało znacznie zredukowane – trzeba więc raczej szukać w starszych archiwach (dużo materiałów zawierała m.in. jedna z płyt EP) oraz na innych tematycznych stronach WWW.

Realizacja VP jest oparta o przerwanie sprzętowego 8-bitowego licznika (RTCC). Częstotliwość jego występowania możemy w szerokim zakresie dopasowywać do potrzeb aplikacji za pomocą programowanego preskalera. W obsłudze przerwania podejmujemy odpowiednie czynności – często (jak np. w przedstawianym projekcie) jest to realizacja działania kolejnego stanu programowego automatu stanów. Im częściej wyzwalamy przerwanie – tym szybsze procesy możemy obsłużyć. Jest to zarazem atrakcja dla miłośników asemblera – np. przy 50 MHz przerwanie co 1μs daje nam 50 cykli zegarowych (20 ns/cykl) dla zrealizowania obsługi – i pozostawienia jeszcze zapasu na główną pętlę programu. Żadne języki wyższego poziomu nie wchodzą tu już w grę – dopasowanie kodu wymaga wyliczania przebiegu instrukcji co do cyklu (SX ma wszystkie instrukcje całkowicie jednoznaczne czasowo, można więc takie wyliczenia precyzyjnie przeprowadzić). Sprawa jest dodatkowo utrudniona dosyć niewdzięczną strukturą pamięci danych oraz kodu.

W SX28 pamięć kodu jest podzielona na 4 strony (*pages*) po 512 słów, co wynika z 9-bitowego adresowania w instrukcjach skoków. Przejście do odpowiedniej strony musi być jawnie wykonane w programie (instrukcja *page* ustawiająca bity adresu strony w rejestrze statusu) przed realizacją fragmentu kodu. Aby dodatkowo „uprościć" sprawę, instrukcje wywołania procedur (*call*) posługują się adresowaniem tylko 8-bitowym. Wszystkie wejścia do procedur muszą się więc mieścić w pierwszych 256 słowach strony (natomiast oczywiście można wewnątrz procedury wykonać skok do wyższej połówki strony jeśli brakuje miejsca na kod).

Z kolei pamięć danych podzielona jest na banki (8 w SX28). Są to 32-bajtowe strony, jednak dolne adresy (00...0x0F) są wspólne dla wszystkich banków, natomiast górne (0x10...0x1F, 0x30...0x3F itd.) są rozdzielone. Adresowanie bezpośrednie w SX używa tylko 5 bitów (zakres 0...0x1F), czyli za mało dla rozróżnienia banków. Pozostałe 3 bity adresu musimy jawnie ustawić w rejestrze FSR (*file select register*) przed wyko-

List. 1. Kod obsługi przetwornika MAX187

```
org   $400
;*****************************************************************
; Max187 Interrupt Service Routines
;*****************************************************************
   ;*****************************************************************
   ; Function: ADC_ISR
   ; AD multichannel conversion Interrupt-Driven State Machine
   ;*****************************************************************
ADC_isr   mov w,ADC_state   ;1
      add PC,w      ;3 ;Add the state to the program counter
                ;and go to the state in the jump table.

      ;*************************************************************
      ; States for ADC while channel changing
      ;*************************************************************
   ADC_channel      =  $
      jmp ADC_change_channel      ;3 new channel is set
      jmp ADC_change_delay_lo     ;3 some delay to stabilize input
      jmp ADC_change_delay_hi     ;3 some delay to stabilize input
      ;*************************************************************
      ; States for ADC measure cycle
      ;*************************************************************
   ADC_conversion   =  $
      jmp ADC_start               ;3 start conversion
      jmp ADC_startdelay          ;3 wait min. 8,5 us before reading
      jmp ADC_firstpulse          ;3 generate first SCK pulse
      jmp ADC_highpart            ;3 load 4 high conversion bits
      jmp ADC_lowpart             ;3 load 8 low conversion bits
      jmp ADC_end                 ;3
      jmp ADC_store_delay         ;wait for storing values

ADC_init  jmp Do_ADC_init

;*************************************************
;Jump to the next channel and set all the control lines
;*************************************************
ADC_change_channel
      inc channel_number
      snb channel_number.5; is 32 ?
      clr channel_number

      setb lo_channel_mux
      setb hi_channel_mux         ;disable both 4067

      and rb_buff,#%00000011      ;keep scl, sda lines

      snb channel_number.0
      setb rb_buff.2
      snb channel_number.1
      setb rb_buff.3
      snb channel_number.2
      setb rb_buff.4
      snb channel_number.3
      setb rb_buff.5

      sb channel_number.4
      clrb lo_channel_mux         ;enable 0-15 inputs 4067 multiplexer

      snb channel_number.4
      clrb hi_channel_mux         ;enable 16-31 inputs 4067 multiplexer

      inc ADC_state
      retp
;*********************************************
;wait a moment to stabilize input after switching
; we use 5* 200*2,7 us = 2,7 ms
;*********************************************
ADC_change_delay_lo
      dec chan_delay_lo
      sz
      retp      ; state unchanged
      mov chan_delay_lo,#200
      inc ADC_state
      retp
ADC_change_delay_hi
      dec chan_delay_hi
      snz
      jmp :end_delay
      dec ADC_state
      retp      ; state unchanged
:end_delay
      mov chan_delay_hi,#5
      inc ADC_state
      retp
;*********************************************
; Start max187 conversion cycle
;
;*************************************************
ADC_start
      clrb max_cs                 ;start conversion (low level active)
      clr ADC_buf_lo              ;buffers ready for new value
      clr ADC_buf_hi
```

cd. List. 1.

```
      inc ADC_state
      retp

ADC_startdelay
      dec start_delay
      sz
      retp                    ;state unchanged, continue delay

      setb start_delay.2      ;means start_delay = 4,
                              ;ready for the next delay
      inc ADC_state
      retp

ADC_firstpulse
      ;set sclk if low and then reset it
      snb max_sclk
      jmp :done
      setb max_sclk
      retp   ; sclk is set

:done
      setb high_counter.2     ; means high_counter = 4
      setb low_counter.3      ; means low_counter = 8
      clrb max_sclk           ; ask for bit 11.

      inc ADC_state
      retp

ADC_highpart
      ;if sclk is low set it to enable data bit
      snb max_sclk
      jmp :bitvalid
      setb max_sclk;enable data bit
      retp

:bitvalid
                              ;after previous raising sclk edge
                              ;we have valid data bit on max_data_in
                              ;read it and write into buffer
      clc    ;clear carry
      snb max_data_in
      stc    ; set carry if data bit high
      rl ADC_buf_hi

                              ;and now clear sclk again to ask for next bit
      clrb max_sclk

      ;then check if all the 4 high bits are ready
      dec high_counter
      sz
      retp   ; not yet

      inc ADC_state ; ready - go to low bits
      retp

ADC_lowpart
      ; quite the same but 8 times
      snb max_sclk
      jmp :bitvalid
      setb max_sclk;enable data bit
      retp

:bitvalid
      clc    ;clear carry
      snb max_data_in
      stc    ; set carry if data bit high
      rl ADC_buf_lo

      ; and now reset sclk again to ask for next bit
      clrb max_sclk

      ;then check if all the 8 low bits are ready
      dec low_counter
      sz
      retp   ; not yet

      inc ADC_state ; done, sclk remains low
      retp

;*************************************************
; Switch off CS line,
; reload temporary value buffers into target memory
; be ready to change channel
;*************************************************
ADC_end
      setb max_cs          ; switch Max187 off (active low level)
      setb ADC_ready_flag      ; conversion done, main loop will use it
      inc ADC_state
      retp
ADC_store_delay
      clr ADC_state
      retp
```

naniem instrukcji używającej tego trybu adresowania (służy do tego specjalny mnemonik *bank*).

Prezentowane informacje są oczywiście wysoce wyrywkowe, ale wydaje się, że właśnie te specyficzne aspekty konstrukcji oraz programowania SX najbardziej przeszkadzają podczas pierwszego czytania manuali. Uprzedzenie o nich może więc znacznie przyspieszyć zapoznanie się z procesorem oraz ułatwić analizę przykładowych kodów.

Jako środowisko programistyczne do realizacji projektu posłużył pakiet *SX Key* z *www.parallax.com*. Jest on – jako program – bezpłatny, ale niestety współpracuje z komercyjnym sprzętowym programatorem o tej samej nazwie. Zestaw umożliwia zarówno programowanie jak i pracę w trybie debuggera w docelowym układzie oraz dowolne ustawianie częstotliwości zegara, stanowi więc narzędzie bardzo silne i uniwersalne. Za mniejszą cenę można się zaopatrzyć w skromniejszy programator – *SX Blitz* – pozbawiony funkcji debugowania. Oszczędność pinów używanych do programowania wiąże się niestety z dość skomplikowanym i wymagającym protokołem. Chociaż został on opublikowany, nie znajdziemy więc zbyt

wielu alternatywnych amatorskich rozwiązań programatorów. Wydaje się, że jedynym rzeczywiście dopracowanym projektem jest *Fluffy 2* – też jednak oparty na mikrokontrolerze PIC i wymagający sporego nakładu pracy przy złożeniu i uruchomieniu.

Jeśli chcielibyśmy korzystać w mniej wymagających czasowo aplikacjach z języków wyższego poziomu, warto sięgnąć m.in. na stronę *www.picant.com*, gdzie znajdziemy shareware'owe kompilatory C, C++ i Pascala dla PIC oraz SX.

Montaż i wstępne uruchomienie

Cały układ został wykonany na elementach w obudowach przewlekanych (**rys**. **2**). Wynikło to m.in. z chęci wykorzystania starszych, szufladowych zapasów (jak np. multipleksery 4067). Jednak dzięki temu montaż i kontrola nie sprawiają żadnych nietypowych kłopotów ani niespodzianek. Zaczynamy tradycyjnie od obwodów zasilania – po ich złożeniu sprawdzamy działanie oraz dostarczane napięcia przy zewnętrznym zasilaniu zmienianym w zakresie 8...15 V. Wtedy dopiero składamy dalej płytkę. W prototypie uzyskano dokładne +5 V z przetwornicy impulsowej oraz ok. 5,3 V z przetwornicy separującej DC/DC. Wprawdzie wykracza to nieco poza zalecane warunki pracy użytego przetwornika A/C, ale nie przekracza wartości dopuszczalnych i całość działa prawidłowo. Jeśli nie będziemy dalej modyfikować programu wlutowujemy także elementy obwodu oscylatora (C13, C14 oraz rezonator kwarcowy Y1). Pozostawimy je nie wmontowane, gdy mamy w planie dalsze debugowanie za pomocą *SX Key*.

Wpisanie kodu do pamięci Flash wykonujemy przy pomocy jednego ze wspomnianych programatorów *SX Key* albo *SX Blitz* podłączonych do listwy P1...P4 (uwaga na kolejność wyprowadzeń!). Program powinien ruszyć od razu. O pracy procesora świadczy miganie diody kontrolnej D3. Po chwili elementy zasilaczy oraz mikrokontroler SX28 lekko się rozgrzewają – jest to objaw prawidłowy i nie świadczy o żadnej usterce. Pobór prądu (bez programatora) wynosi ok. 100mA.

Dokładniejsze sprawdzenie funkcji przetwornika wymagać będzie zestawienia bardziej rozbudowanego układu odczytowego.

Oprogramowanie

Działanie programu polega na cyklicznym przełączaniu wejść pomiarowych, obsłudze sekwencji konwersji A/C przetwornika MAX 187 i wpisywaniu wyników w odpowiednie miejsca buforów pamięciowych. Jednocześnie monitorowany jest stan linii magistrali I2C:

- po wykryciu sekwencji *start* odbierany jest adres *slave*,
- sprawdzana jest zgodność przesłanego adresu z własnym,
- w przypadku odebrania zgodnego adresu wysyłane jest potwierdzenie ACK,
- jeśli odebrano adres do zapisu – następujący po nim jeden bajt danych określa numer kanału, z którego *master* chce otrzymać wynik; na tej podstawie zostają załadowane do bufora nadajnika odpowiednie pozycje pamięci pomiarów,
- jeśli odebrano adres do odczytu – wysyłane są dwa bajty wyniku pomiaru z żądanego kanału.

Uzupełniająco pracuje 16-bitowy timer, który poprzez przełączanie diody kontrolnej LED pozwala na stwierdzenie poprawnego działania układu. Jako podstawa do napisania programu posłużył firmowy moduł VP – urządzenie *slave* I²C. Jest to programowy automat stanów przełączany w obsłudze przerwania RTCC zgodnie z aktualną sytuacją na magistrali. Kod programu jest dostępny na CD-EP10/2004B.

Z kolei obsługa przetwornika MAX 187 wymagała napisania programu od podstaw. Przebieg pojedynczego cyklu konwersji przetwornika przedstawiono na **rys**. **3**:

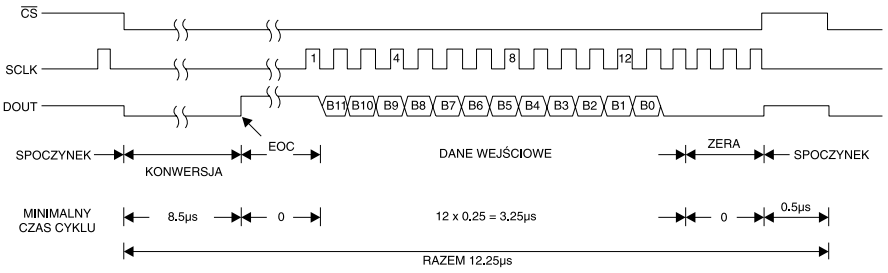
- wyzwolenie konwersji niskim poziomem CS,
- odczekanie min. 8,5 μs do zakończenia konwersji (EOC),
- szeregowy odczyt wyniku (pierwszy takt zegara zwraca zawsze 1, następne 12 taktów pobiera 12 bitów wyniku począwszy od MSB, zera kończące można pominąć).

Do realizacji tego przebiegu został również użyty automat stanów – przedstawia ono go na **list**. **1** – który dodatkowo wykonuje przełączanie kolejnych kanałów z odpowiednim opóźnieniem na ustabilizowanie (przeładowanie pojemności) wejścia. Automat nie zapewnia nam wprawdzie wykorzystania pełnej szybkości odczytu przetwornika, ale w przypadku naszego stosunkowo powolnego urządzenia nie jest to wcale wadą. Zwróćmy też uwagę na prosty ale wydajny mechanizm lokalizacji procedur dla poszczególnych stanów – numer stanu jest dodawany do licznika rozkazów czego efektem jest wybranie odpowiedniej pozycji w tabeli skoków i przejście bezpośrednio do kodu obsługi tego stanu.

Program był uruchamiany tylko do momentu uzyskania stabilnej pracy modułu. Dalsze udoskonalenia i modyfikacje (użycie watchdoga, autoinkrementacja numerów kanałów przy odczycie itd.) są pozostawione do uznania użytkowników.

Przykład sprawdzenia i wykorzystania modułu

W materiałach pomocniczych opublikowanych na CD-EP10/2004B znajdziemy cały projekt Delphi do testowego odczytu naszego modułu. Program odczytuje cyklicznie wszystkie kanały pomiarowe i wyświetla pobrane wartości na 32 bargrafach o zakresach znormalizowanych do przedziału <0...1>. Jako testowy zadajnik napięć posłużyły wieloobrotowe potencjometry montażowe (10 sztuk – czyli używamy tylko wybranych kanałów) ulokowane prowizorycznie na kawałku płytki uniwersalnej. Magistrala I²C jest zrealizowana programowo poprzez sterowanie indywidualnymi wyprowadzeniami portu szeregowego. Służy do tego dodatkowy komponent *TRsPin*, którego źródła są również załączone. Dopasowanie poziomów elektrycznych linii SDA i SCL a także całkowita separacja galwaniczna portu szeregowego od modułu zapewnione są dzięki zastosowaniu optoizolatora magistrali I²C opisanego w miniprojektach EP11/02. Metoda ta wymaga na ogół dopasowania generowanych przebiegów do szybkości używanego PC – w unicie *u_i2c* znajdziemy stałe *DsrDelay* oraz *LineDelay*, które decydują o opóźnieniach i mogą wymagać eksperymentalnych zmian (wpisane w kodzie wartości doty-



Rys. 3. Przebieg cyklu konwersji A/C w MAX187

Rys. 4. Uruchomiony program testowy z wyświetlonymi etykietami wizualizacji

czą Athlona 1,33 GHz). Ponieważ program ma charakter tylko testowo – warsztatowy nie przewidziałem specjalnych kontrolek do nastawy tych parametrów przez użytkownika. Należy je korygować (podobnie jak numer używanego portu) z poziomu Delphi bezpośrednio w kodzie źródłowym.

Można natomiast zmieniać adres *slave* modułu SX dla uzgodnienia z położeniem przełącznika adresowego na płytce. Ekran uruchomionego testu przedstawiono na **rys**. **4**. Dodatkowe etykiety liczbowe widoczne na tle okienka są składową całkiem oddzielnego programu współpracującego z odczytem wartości pomiarowych.

Uniwersalny program do prostej wizualizacji danych

Program do uniwersalnej tekstowej wizualizacji danych powstał wcześniej przy zupełnie innej okazji – teraz zaś nadarzyła się sposobność do jego przykładowego zastosowania. Pozwala on na wyświetlenie do 24 etykiet (kanałów) prezentujących tekstowo wartości liczbowe. Etykiety mają właściwość *stay-on-top* więc zawsze pozostają ponad dowolnym tłem (aplikacją). Każdą z etykiet możemy indywidualnie skonfigurować (jednostka, liczba miejsc po przecinku, mnożnik, filtr, częstotliwość aktualizacji, opis, kolorystyka) i przesunąć myszą w dowolne miejsce ekranu. Dzięki temu możemy w prosty i szybki sposób wyposażyć dowolną grafikę (obraz obiektu technologicznego, schemat elektryczny lub elektroniczny, rysunek przekrojowy pomieszczeń itp.) w dynamiczny opis dowolnych parametrów. Wizualizacja nie jest bowiem powiązana na stałe z żadnym interfejsem pomiarowym i samodzielnie nie dostarcza żadnych danych. Udostępnia natomiast funkcję biblioteczną, za pośrednictwem której wszelkie inne aplikacje bezpośrednio obsługujące pomiar mogą przekazać niezbędne wartości w postaci tablicy liczb typu *double*. Nasz przykładowy czytnik modułu SX pokazuje dokładnie jak to zrobić.

Program wizualizacji załączono w materiałach pomocniczych jako gotowy plik instalacyjny *setup.exe*. Instalator kopiuje wszystkie potrzebne pliki i biblioteki, w tym pomoc, która bardziej szczegółowo opisuje sposób użytkowania. Nie miałem jednak okazji sprawdzić instalatora i programu na platformach NT/2000/XP – prosiłbym o informacje w razie wystąpienia kłopotów.

Jerzy Szczesiul, EP
jerzy.szczesiul@ep.com.pl

Wzory płytek drukowanych w formacie PDF są dostępne w Internecie pod adresem: ***pcb.ep.com.pl*** *oraz na płycie CD-EP10/2004B w katalogu* ***PCB****.*

WYKAZ ELEMENTÓW

Półprzewodniki

U1: mikrokontroler SX28AC/DP Ubicom

U2, U3: multiplekser analogowy CMOS 4067

U4: poczwórny transoptor K847 (lub zbliżony)

U5: 12-bitowy przetwornik AC MAX 187 BCPA

U6: stabilizator impulsowy LM 2671 N – 5.0V

U7: przetwornica DC/DC NME 0505 S (C&D – dawniej Newport Components)

T1...3: szybki podwójny transoptor HCPL (ICPL) 2630

D1: dioda prostownicza 1A

D2: dioda Schottky 1N5817

D3: dioda LED

Rezystory:

wszystkie 1/8 W

R1, R2: 1kΩ

R3: 3,3Ω

R4...7, R16, R17, R21, R22: 4,7kΩ

R8...13, R23...26: 330Ω

R14, R15, R19,R 20: 3,3kΩ

R18: 470Ω

R27: 560Ω

Kondensatory

C1: 10nF monolityczny

C2: kondensator elektrolityczny niskoimpedancyjny 470µF/25V (CapXon)

C3: kondensator elektrolityczny niskoimpedancyjny 100µF/10V (Sanyo OS-CON)

C4, C5, C6, C8, C12: odsprzęgające 100 nF monolityczne

C13, C14: 15 pF ceramiczny

C7, C11: 22µF/16V tantalowy

C10: 4,7µF/25V tantalowy

C9: 1 nF monolityczny (wartość dobrana przy uruchamianiu, 100nF ze schematu nieaktualne)

Elementy indukcyjne

L1: dławik 100µH DSZ6/100/0.8 Feryster

Inne

Listwy goldpin – proste podwójne i kątowe pojedyncze.

Przełącznik DIPswitch 4-pozycyjny.

Y1 – rezonator kwarcowy 50,00 MHz (częstotliwość podstawowa).

Element ten okazał się niestety stosunkowo trudno dostępny. Na schemacie opisano kwarc jako 48,00 MHz – jest to wartość wystarczająca, jednak okazało się, że posiadany w zapasach jest nie podstawowy ale overtonowy co uniemożliwiło prawidłową pracę. Na witrynach poświęconych SX można znaleźć opisy wykorzystania również overtonów, ale wiąże się to z przeróbkami układu. Zwróćmy też uwagę, że programowanie można wykonywać przy dołączonym rezonatorze (kwarcowym lub ceramicznym), natomiast wszelkie inne źródła taktowania – np. oscylatory – muszą być odłączane gdyż zazwyczaj nie wytrzymują używanego do programowania napięcia 12V.