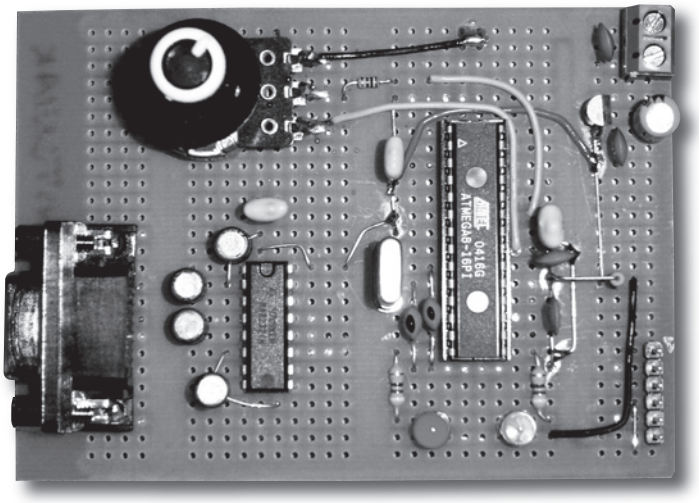
Programowanie portu szeregowego w systemach operacyjnych Linux i Windows, część 2



Umiejętność programowej obsługi interfejsu RS232 od strony komputera PC jest dziś istotnym elementem elektronicznego rzemiosła. W niniejszym kursie piszemy jak w praktyce oprogramować port szeregowy w środowiskach Linux i Windows. Wiele miejsca poświęcamy pisaniu przenośnych aplikacji GUI, które korzystają z interfejsu szeregowego i zachowują się tak samo w systemach Windows jak i Linux. Wszystkie omawiane zagadnienia poparto szczegółowo opisanymi praktycznymi przykładami.

W poprzedniej części kursu omówiłem funkcje służące do realizacji operacji na porcie szeregowym w Linuksie. Przyszedł czas na wykorzystanie zdobytej wiedzy w praktyce. W części drugiej przedstawię budowę platformy sprzętowej, która pozwoli nam testować tworzone oprogramowanie.

Zestaw testowy

Schemat zestawu testowego, który posłuży nam do testowania wszystkich przykładów w dalszych częściach kursu, znajduje się na **rys. 2**. Jego sercem jest dobrze wszystkim znany mikrokontroler ATmega8 firmy Atmel. Wybór tego właśnie mikrokontrolera podyktowany był faktem, że jest to jeden z najmniejszych mikrokontrolerów serii ATmega, który ma „na pokładzie" przetwornik analogowo–cyfrowy. Przetwornik ten wykorzystamy w ostatniej części cyklu do realizacji woltomierza sterowanego przez port szeregowy. Sam wybór rodziny AVR Atmela był oczywisty – są to wszak najpopularniejsze (jeszcze!) w Polsce mikrokontrolery i większość Czytelników zna je co najmniej dobrze. Dzięki temu będą oni mogli skupić swoją uwagę na zagadnieniu programowania RS232C na PC, zaś szczegóły oprogramowania części sprzętowej będą dla nich od początku do końca jasne.

Połączenie procesora U1 z komputerem PC zrealizowane jest typowo, to jest za pośrednictwem układu MAX232. Model łączony jest z komputerem za pośrednictwem kabla „prostego" (bez przeplotu), co wymusiło zastosowanie złącza DB9F (żeńskiego). Czytelnicy, którzy zechcą użyć typowego przewodu z przeplotem muszą wykorzystać złącze DB9M oraz zamienić w nim miejscami końcówki 2 i 3. Rezonator kwarcowy X1 ma „okrągłą" wartość 1,8432 MHz, co pozwala uzyskać większość typowych prędkości transmisji z zerowym (obliczanym) błędem baudrate. We wszystkich przykładach w tym kursie wykorzystana będzie prędkość 19200 bodów i ramka w formacie 8N1 (8 bitów danych, brak kontroli parzystości, 1 bit STOP). Dodatkową zaletą dosyć niskiej częstotliwości pracy mikrokontrolera jest redukcja pobieranego przez zestaw prądu.

Diody LED D1 i D2 stanowią najprostszy interfejs wyjściowy – rola przez nie pełniona będzie zależna od funkcjonalności testowanego przykładu. Elementy U3, C7... C9 tworzą typowy zasilacz stabilizowany zasilający część cyfrową napięciem 5 V. Para R1C3 wytwarza napięcie zasilające część analogową zestawu. Masy: analogowa i cyfrowa są rozdzielone i łączą się elektrycznie niedaleko punktu dołączenia napięcia zasilającego. Nierozdzielenie mas analogowej i cyfrowej mogłoby doprowadzić do podawania na wejście przetwornika A/C impulsów szpilkowych i – w konsekwencji – do jego błędnej pracy. Impulsy te pochodziłyby ze spadków napięcia na ścieżkach masy powodowanych dużymi prądami impulsowymi występującymi w części cyfrowej.

We wspomnianym przykładzie woltomierza cyfrowego przetwornik A/C będzie pracował z wewnętrznym napięciem odniesienia równym (typowo) 2,56 V – napięcie to występuje na końcówce AREF. Jest ono filtrowane przez kondensator C2 i stanowi napięcie wejściowe dla potencjometru PR1. Napięcie z suwaka PR1 jest podawane poprzez filtr R2C1 na wejście ADC0 przetwornika. Napięcie to będzie mierzone przez woltomierz, który zaprojektujemy w dalszej części kursu.

Zestaw testowy można zmontować w dowolny sposób pamiętając jedynie o rozdzieleniu mas i ogólnych zasadach montażu układów elektronicznych. Zamiast opisanego układu można oczywiście użyć dowolnego zestawu laboratoryjnego dostępnego w handlu, byleby tylko miał on podobne możliwości funkcjonalne jak opisany powyżej (przetwornik A/C i interfejs RS232C). Nie musi to być

nawet zestaw z mikrokontrolerem AVR. Oprogramowanie jest napisane w języku C i może być przeniesione na dowolny mikrokontroler, do którego istnieje kompilator tego języka. Oczywiście w takim przypadku trzeba będzie zmienić te fragmenty programu, które są specyficzne dla danego układu (np. konfiguracja modułu USART).

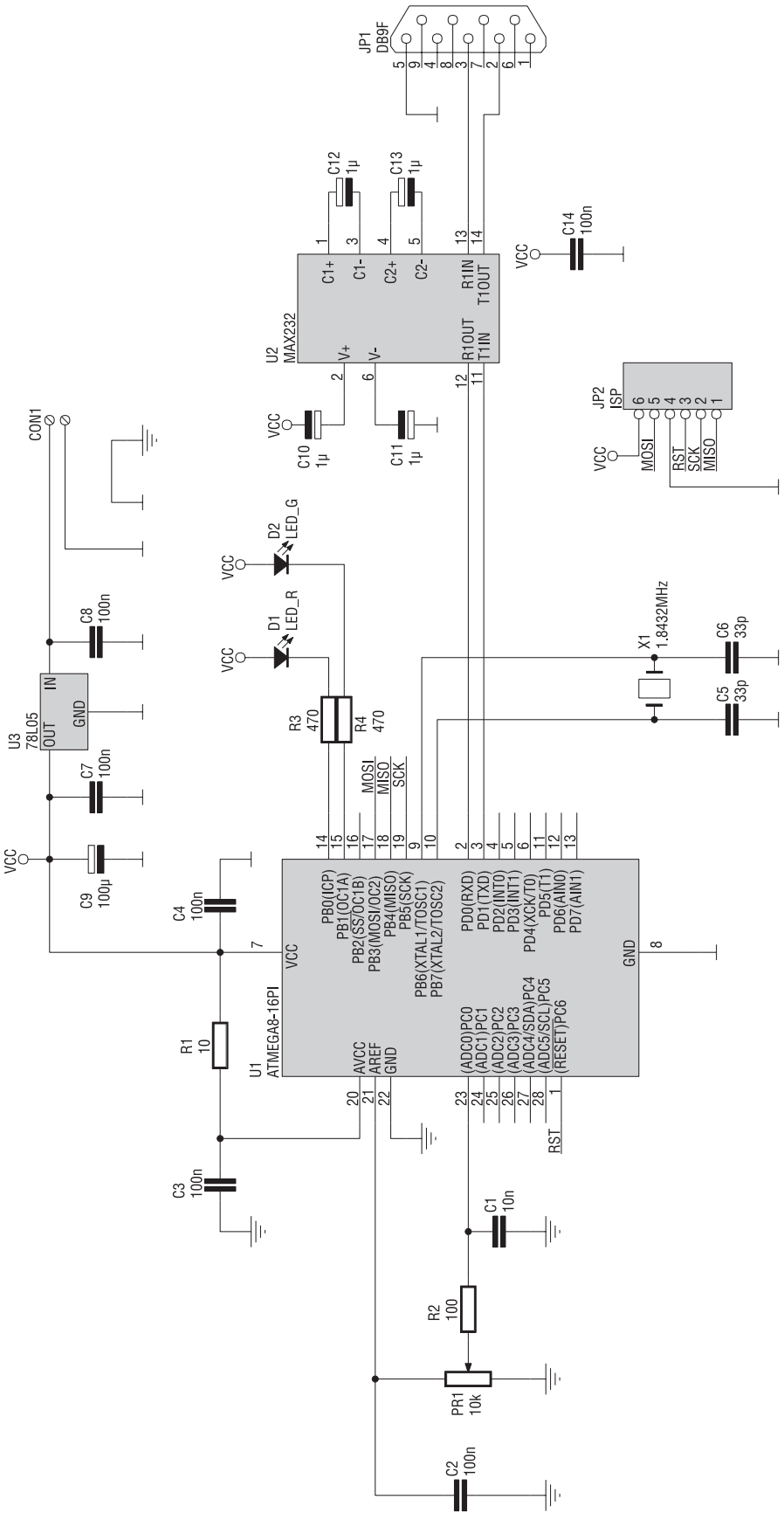
Oprogramowanie mikrokontrolera

Oprogramowanie mikrokontrolera U1 ATmega8 napisałem w języku C używając popularnego kompilatora GCC w wersji 2002–06–25 zintegrowanego z pakietem AVRStudio3.53. Kod źródłowy jest na tyle krótki, prosty i pozbawiony „wodotrysków", że można go z powodzeniem skompilować za pomocą dowolnego kompilatora C dla mikrokontrolerów AVR – oczywiście po dokonaniu stosownych (niewielkich) poprawek.

List. 5. Program testowy Example1.c działający na mikrokontrolerze

```
//*******************************
// MAIN
//*******************************
int main(void)
{
  unsigned char data,cou;
  DDRB=0xFF;
  DDRC=0x00;
  DDRD=0xFE;
  PORTC=0x00;
  PORTD=0xFE;

  Hello();

  Init_UART();

  while(1)
  {
     //Waiting for 1 byte
     UCSRA&=~(1<<RXC);
     while((UCSRA&(1<<RXC))==0);
     data=UDR;

     if(data==0)
     {
        RED_OFF;
        GREEN_OFF;
        SendString(PSTR(„ZERO"));
        SendByte(13); SendByte(10);
     }
     else if(data==1)
     {
        RED_ON;
        GREEN_OFF;
        SendString(PSTR(„ONE"));
        SendByte(13); SendByte(10);
     }
     else if(data==2)
     {
        RED_OFF;
        GREEN_ON;
        SendString(PSTR(„TWO"));
        SendByte(13); SendByte(10);
     }
     else if(data==3)
     {
        RED_ON;
        GREEN_ON;
        SendString(PSTR(„THREE"));
        SendByte(13); SendByte(10);
     }
     else if(data==255)
     {
        cou=0;
        while(1)
        {
           SendByte(cou);
           if(++cou==0) break;
        }
     }
     else
     {
     SendString(PSTR(„FOUR OR MORE"));
     SendByte(13); SendByte(10);
     }
  }
}
```

W **tab. 10** zamieściłem konfigurację *fusebit*–ów niezbędną do poprawnej pracy procesora ATmega8 w opisywanym zestawie. Podczas programowania mikrokontrolera należy zwrócić baczną uwagę na poprawność tych ustawień, gdyż ich zmiana może uniemożliwić pracę syste-



Rys. 2. Schemat zestawu testowego

mu! Podczas kursu będziemy wykorzystywać dwa programy działające na mikrokontrolerze U1. Pierwszym z nich (opisanym niżej) jest prosty system odpowiadający na pytania. Program ten będzie wykorzystywany w dalszych częściach niniejszego kursu. Drugą aplikacją jest oprogramowanie cyfrowego woltomierza, którą weźmiemy na warsztat w ostatniej części kursu.

Wszystkie przykłady oprogramowania na PC (poza woltomierzem) będą współpracować z prostym programem *Example1.c* działającym na mikrokontrolerze. Główna część tego programu (funkcja *main*) została przedstawiona na **list. 5** (pominąłem inicjalizację modułu USART i inne funkcje pomocnicze). Idea tej aplikacji polega na stworzeniu prostego systemu zdolnego odpowiadać na zadawane mu proste pytania. Taki system pozwoli w łatwy sposób testować różne aplikacje działające na komputerze PC.

Sposób działania programu z list. 5 jest następujący:

1. Mikrokontroler czeka na 1–bajtowe zapytanie, które wysyła komputer PC;
2. Po otrzymaniu jednego bajta mikrokontroler odpowiednio do jego wartości zapala diody D1 i D2 oraz wysyła via RS232 stosowną odpowiedź. Odbywa się to według następującej zasady:
   - Jeśli wartość bajta zapytania mieści się w granicach 0...3, to stan diod odzwierciedla tę liczbę (D2 – bit nr 1, D1 – bit nr 0), zaś wysyłana do PC–ta odpowiedź stanowi nazwę tej wartości zapisaną wielkimi literami w języku angielskim zakończoną sekwencją bajtów 0x0D i 0x0A. Na przykład otrzymanie bajta o wartości 1 skutkuje wygaszeniem diody D2, zapaleniem D1 i wysłaniem odpowiedzi „ONE”+0x0D+0x0A. Dla bajta o wartości 3 zostaną zapalone obie diody, a odpowiedzią będzie „THREE”+0x0D+0x0A.
   - Jeśli wartość bajta zapytania mieści się w granicach 4...254, to stan diod nie zmienia się, zaś wysyłana odpowiedź ma postać „FOUR OR MORE”+0x0D+0x0A.
   - Jeśli wartość bajta zapytania wynosi 255, to stan diod nie zmienia się, zaś w odpowiedzi wysyłana jest sekwencja 256 bajtów o wartościach 0...255.

Wywołanie funkcji powitania *Hello* powoduje dwukrotne mignięcie obiema diodami D1 i D2 tuż po włączeniu zasilania zestawu lub zerowaniu mikrokontrolera. Rzecz banalna, ale podczas pierwszego uruchomienia systemu pozwala zorientować się, czy zestaw daje jakiekolwiek „znaki życia”.

WYKAZ ELEMENTÓW

Rezystory

R1: 10 V
R2: 100 Ω
R3, R4: 470 Ω

Kondensatory

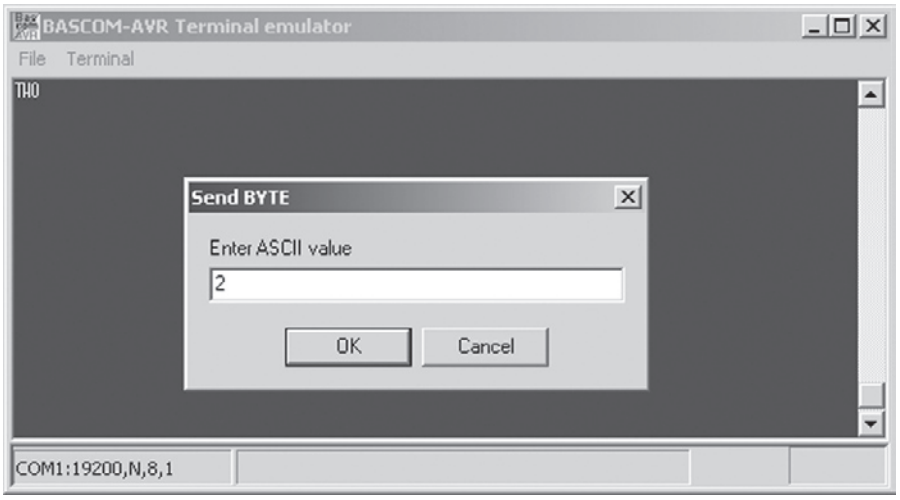
C1: 10nF
C2...C4, C7, C8, C14: 100 nF
C5, C6: 33 pF
C9: 100 μF/16 V
C10...C13: 1...10 μF/16 V

Półprzewodniki

U1: ATmega8
U2: MAX232
U3: 78L05
D1: LED czerwona
D2: LED zielona

Inne

Kwarc: 1,8432 MHz
JP1: złącze DB9F (kabel prosty) lub DB9M (kabel z przeplotem)
JP2: goldpiny (złącze ISP)
PR1: potencjometr 10 kΩ A z pokrętłem



Rys. 3. Okno terminala podczas testowania zestawu

Testowanie zestawu

Poprawnie zmontowany zestaw działa od pierwszego włączenia i nie wymaga żadnych czynności uruchomieniowych. Warto jednak przetestować go przed podjęciem prób współpracy z oprogramowaniem pracującym na PC. Do testów można użyć dowolnego terminala RS232C. Należy w tym celu ustawić odpowiednie parametry transmisji (19200, 8N1), a następnie wysyłać 1–bajtowe zapytania i sprawdzać czy w odpowiedzi wysyłane są poprawne ciągi znaków. Na **rys. 3** przedstawiłem przykładowy test zestawu za pomocą terminala wchodzącego w skład pakietu BascomAVR Demo, pracującego w systemie Windows.

Arkadiusz Antoniak, EP
arkadiusz.antoniak@ep.com.pl
www.antoniak.ep.com.pl

Tab. 10. Ustawienia bezpieczników (fusebit) mikrokontrolera

	Fusebit	Wartość
Fusebit LOW	BODLEVEL	1: BODLEVEL 2,7V
	BODEN	0: BODEN enabled
	SUT1..0	01: fast rising power
	CKSEL3..0	1101: Crystal oscillator
Fusebit HIGH	S8515C	1: MEGA8515 mode
	WDTON	1: Watchdog timer enabled by software
	SPIEN	0: Serial programming enabled
	CKOPT	1: Oscillator option
	EESAVE	1: Erase EEPROM when chip erase
	BOOTSZ1..0	00: 1024 words boot size
	BOOTRST	1: Reset vector is $0000