Dział „Projekty Czytelników" zawiera opisy projektów nadesłanych do redakcji EP przez Czytelników. Redakcja nie bierze odpowiedzialności za prawidłowe działanie opisywanych układów, gdyż nie testujemy ich laboratoryjnie, chociaż sprawdzamy poprawność konstrukcji.

Prosimy o nadsyłanie własnych projektów z modelami (do zwrotu). Do artykułu należy dołączyć podpisane ***oświadczenie, że artykuł jest własnym opracowaniem autora i nie był dotychczas nigdzie publikowany****. Honorarium za publikację w tym dziale wynosi 250,- zł (brutto) za 1 stronę w EP. Przysyłanych tekstów nie zwracamy. Redakcja zastrzega sobie prawo do dokonywania skrótów.*

Pokojowy regulator temperatury ze zdalnym sterowaniem, część 2

Pomysł budowy urządzenia powstał, gdy ze względu na niską temperaturę panującą w pokoju, autor zmuszony był do korzystania z dodatkowego ogrzewania elektrycznego. Użytkowanie typowego ogrzewacza wnętrzowego (popularnej „farelki"), oprócz wysokich kosztów zużytej energii elektrycznej, miało poważną wadę: każdorazowe włączenie i wyłączenie urządzenia wymagało interwencji użytkownika. A przecież jesteśmy tak leniwi...

Projekt 113

Obsługa magistrali jednoprzewodowej

Na **list. 2** przedstawiono procedury asemblerowe realizujące fizyczną komunikację z urządzeniami dołączonymi do magistrali jednoprzewodowej. Każdorazowe rozpoczęcie transmisji na magistrali jednoprzewodowej wymaga najpierw wysłania tzw. impulsu zerującego przez nadrzędny układ nadajnika (tutaj mikrokontroler). Jest to realizowane poprzez wymuszenie przez pewien czas (480...960µs) niskiego poziomu na magistrali, następnie zwolnienie magistrali (normalnie - gdy magistralą nie są transmitowane żadne dane - na magistrali występuje poziom wysoki wymuszany rezystorem podciągającym), odczekanie określonego czasu i sprawdzenie stanu magistrali. Jeżeli wówczas magistrala znajduje się w stanie niskim, oznacza to obecność jednego lub kilku układów dołączonych do magistrali jednoprzewodowej. W takim przypadku układy te, po upływie określonego czasu, zwalniają magistralę jednoprzewodową.

Generowanie impulsu zerującego realizuje bezparametrowa procedura *DSReset*. Na początku dodatkowo blokowane są wszystkie przerwania z wyjątkiem przerwania zewnętrznego - z odbiornika zdalnego sterowania, następnie zerowany jest bit magistrali *DSbit* (tutaj P3.3) i wywoływana jest procedura *Delay5us* realizująca opóźnienie będące wielokrotnością 5µs. Pod adresem bezpośrednim (zmienną) *delay* - parametrem procedury - umieszcza się krotność tego opóźnienia. Po wymuszeniu na magistrali poziomu niskiego procedura odczekuje ok. 600µs, po czym zwalnia magistralę (ustawia *DSbit*), czeka ok. 66µs i sprawdza stan magistrali, po czym odczekuje jeszcze dodatkowo 300µs i kończy swoje działanie. Rejestr R7 zawiera rezultat wykonanej operacji. Jeżeli wystąpił impuls obecności układów dołączonych do magistrali jednoprzewodowej, w tym rejestrze znajduje się 0. W przeciwnym przypadku wartość jest różna od zera.

Procedura *_DSTx* realizuje zapis 8 bitów danych do układu dołączonego do magistrali jednoprzewodowej. Dane do zapisu powinny znajdować się w rejestrze R7 procesora. Zapis każdego bitu na magistrali jednoprzewodowej rozpoczyna się od wymuszenia, przez układ nadrzędny, poziomu niskiego na magistrali trwającego, zgodnie ze specyfikacją, nie dłużej niż 15µs (procedura czeka ok. 13µs). Następnie jeżeli zapisywanym bitem jest „1" - magistrala jest zwalniana (występuje poziom wysoki wymuszony rezystorem podciągającym), jeżeli zaś zapisywanym bitem jest „0", wówczas układ podrzędny podtrzymuje stan niski na magistrali. Całkowita długość szczeliny czasowej zapisu (od momentu wymuszenia poziomu niskiego na magistrali) nie powinna przekraczać 120µs i po tym czasie magistrala powinna być zwolniona. Czas przerwy między nadawaniem kolejnych bitów nie może być krótszy niż 1µs.

Procedura *DSRx* realizuje odczyt 8 bitów danych z magistrali jednoprzewodowej.

List. 2. Procedury obsługi magistrali jednoprzewodowej

```
Delay5us:
   nop
   nop
   nop
   djnz    delay,Delay5us
   ret

DSReset:
   mov r5,IE
   anl IE,#129
   clr DSbit
   mov delay,#119
   lcall   Delay5us
   setb    DSbit
   mov delay,#12
   lcall   Delay5us
   mov r7,#0
   jnb DSbit,next1
   mov r7,#1
next1:
   mov delay,#60
   lcall   Delay5us
   mov IE,r5
   ret

_DSTx:
   clr EA
   mov r6,#8
   mov a,r7
Loop1:
   mov delay,#1
   clr DSbit
   lcall   Delay5us
   clr c
   rrc a
   jnc next2
   setb    DSbit
next2:
   mov delay,#14
   lcall   Delay5us
   setb    DSbit
   nop
   djnz    r6,Loop1
   setbEA
   ret

DSRx:
   mov r6,#8
   mov a,#0
   clr EA
Loop2:
   mov delay,#1
   clr DSbit
   lcall   Delay5us
   setb    DSbit
   nop
   nop
   clr c
   jnb DSbit,next3
   setb    c
next3:
   rrc a
   mov delay,#11
   lcall   Delay5us
   djnz    r6,Loop2
   mov r7,a
   setb    EA
   ret

_CRCupdate:
   mov a,r7
   mov temp,a
   mov r6,#8
CRCLoop:
   xrl a,CRC
   rrc a
   mov a,CRC
   jnc ZERO
   xrl a,#18h
ZERO:
   rrc a
   mov CRC,a
   mov a,temp
   rr  a
   mov temp,a
   djnz    r6,CRCLoop
   ret
```

Szczelina czasowa odczytu wygląda w sposób analogiczny jak to opisano w przypadku szczeliny czasowej zapisu. Odczyt poszczególnych bitów również rozpoczyna się od wymuszenia poziomu niskiego na magistrali przez czas nie dłuższy niż 15µs, następnie magistrala jest zwalniana, procedura odczekuje kilka mikrosekund i próbkowany jest stan magistrali. Jeżeli w wyniku próbkowania odczytano poziom wysoki, oznacza to, że odczytanym bitem jest 1, w przeciwnym przypadku odczytany bit to 0. Odczytany bajt danych znajduje się w rejestrze R7.

Podczas działania obydwu procedur (*_DSTx* i *DSRx*) zabronione jest przyjmowanie jakichkolwiek przerwań (zerowana globalna flaga *EA*), które mogłyby spowodować zakłócenia w precyzyjnym odmierzaniu czasu, określanego przez liczbę cykli maszynowych niezbędnych do wykonania poszczególnych rozkazów.

Procedura *CRCupdate* realizuje obliczanie wartości wielomianu kontrolnego *CRC* zgodnie z równaniem: $x^8+x^5+x^4+1$.

Aktualizowaną wartość wielomianu zawiera zmienna o nazwie *CRC*. Ośmiobitowa dana wejściowa, na podstawie której dokonuje się aktualizacja wartości wielomianu, znajduje się w rejestrze R7.

Wyżej wymienione procedury (pełny listing programu w asemblerze znajduje się w materiałach udostępnionych na *www.ep.com.pl* i płycie CD-EP11/2003B) zostały skompilowane asemblerem A51 z pakietu KEIL i jako plik **.obj* dołączone do listy plików wejściowych linkera L51.

Przekazywanie parametrów z poziomu języka C do procedur asemblerowych odbywa się poprzez rejestry mikroprocesora. Każda procedura asemblerowa odpowiada funkcji w języku C. Jeżeli podprogram asemblerowy wymaga jednego 8-bitowego parametru, wówczas kompilator KEIL wartość argumentu funkcji (zmiennej typu *char*) w języku C zapisze do rejestru R7 mikroprocesora. Przekazanie 8-bitowej wartości z programu w asemblerze do funkcji w języku C (wartość zwracana przez funkcję) również odbywa się za pośrednictwem rejestru R7.

Z poziomu języka C deklaracja funkcji realizujących wymienione wyżej operacje (definicja tych funkcji zawarta jest w kodzie asemblerowym, jako wyżej wymienione procedury) wygląda następująco:

```
uchar DSReset(void);
void DSTx(uchar);
uchar DSRx(void);
void  CRCupdate(uchar);
void ResetCRC(void);
uchar GetCRC(void);
```

przy czym typ *uchar* został zdefiniowany następująco:

```
typedef unsigned char
uchar;
```

Dodatkowo w module asemblerowym zdefiniowano dwie procedury *ResetCRC* i *GetCRC*. Pierwsza z nich zeruje lokalną zmienną *CRC* przechowującą wartość wielomianu kontrolnego, a druga procedura umożliwia pobranie wartości tej zmiennej.

Odczyt temperatury z termometru DS1820

Na **list. 3** przedstawiono kod źródłowy funkcji w języku C realizującej odczyt temperatury z jednego lub kilku termometrów DS1820 dołączonych do magistrali jednoprzewodowej z wykorzystaniem wcześniej zdefiniowanych procedur.

Funkcja *ReadTemp()* posiada dwa parametry. Pierwszy z nich (*dev*) to wskaźnik do tablicy zawierającej 8-bajtowy unikalny numer seryjny danego układu DS1820 wraz z wartością CRC dla tych 8 bajtów (istotne w przypadku dołączania do magistrali jednoprzewodowej więcej niż jednego termometru, jeśli do magistrali dołączony jest tylko jeden układ wówczas wystarczy ten argument zastąpić stałą NULL). Definicja tego parametru może wyglądać następująco:

```
code char
dev1[]={0x10,0x3b,0xe1,
0x38,0x00,0x00,0x00,0xad};
```

List. 3. Funkcja odczytu pamięci podręcznej układu DS1820

```
uchar ReadTemp (char *dev, char *buf)
{  char i;

  if (DSReset()) return 1;
  // impuls zerujący, jeśli brak odpowiedzi koniec funcji

  DSTx(0xcc);
  // Polecenie "przeskocz ROM", czyli transmisja do wszystkich układów
  // dołączonych do magistrali

  //DSTx(0x55);
  //for (i=0;i<8;i++)DSTx(dev[i]);
  //jeżeli chcemy nadawać do konkretnego układu (nie do wszystkich)
  //należy użyć powyższego kodu

  DSTx (0x44);
  // Polecenie "zmierz temperaturę"

  i=0; while (!DSRx())if (i++>10000) return 5;
  // oczekiwanie na zakończenie pomiaru

  if (DSReset()) return 1;
  // znów impuls zerujący

  if (dev!=NULL)
  { DSTx(0x55);
   for (i=0;i<8;i++) DSTx(dev[i]);
   // Polecenie "dopasuj ROM", czyli kolejne polecenia będą
   // akceptowane tylko przez ten układ, którego numer seryjny
   //znajduje się w tablicy dev
  }
  else
   DSTx(0xcc);
  // "przeskocz ROM" - jeśli jest tylko jeden układ

  DSTx(0xBE);
  //Polecenie "odczytaj pamięć podręczną"

  ResetCRC();
  // zerowanie lokalnej zmiennej CRC

  for (i=0;i<8;i++)
  {  buf[i]=DSRx();
    CRCupdate(buf[i]);
  }
  // Odczyt kolejno 8 bajtów pamięci podręcznej, zapis każdego bajtu
  // do bufora buf i aktualizacja CRC

  buf[8]=DSRx();
  // Dziewiąty transmitowany bajt to wartość CRC

  if (DSReset()) return 1;
  // Koniec operacji odczytu pamięci podręcznej - impuls zerujący

  if (GetCRC()!=buf[8]) return 2;
  return 0;
  // Jeżeli wartość CRC obliczona i odczytana z układu DS1820
  // są zgodne funkcja zwraca 0

}
```

Ponieważ numer seryjny wraz z wartością CRC są to stałe, więc można je umieścić w pamięci programu, oszczędzając pamięć danych. Stąd słowo kluczowe *code*.

Drugi parametr *buf* to wskaźnik do 9-bajtowej tablicy, do której zostaną skopiowane kolejne bajty pamięci podręcznej z układu DS1820 zawierające m.in. zmierzoną wartość temperatury. Funkcja zwraca 0 jeżeli pomiar i odczyt temperatury przebiegły pomyślnie i wartość różną od zera w przeciwnym przypadku.

Odczyt temperatury z dokładnością 0,1°C, zgodnie z dokumentacją, jest możliwy po zastosowaniu wzoru:

$$T = temp\_r - 0.25 + \frac{buf[7] - buf[6]}{buf[7]}$$

gdzie *buf* jest tablicą zawierającą (jak wyżej) kopię zawartości pamięci podręcznej układu DS1820, zaś *temp_r* jest wartością *buf[0]*, której wszystkie bity zostały przesunięte o jeden w prawo. Funkcjonalnie odpowiada to dzieleniu całkowitemu:

$$temp\_r = buf[0]/2$$

lub w języku C, z wykorzystaniem operatora przesunięcia bitowego: *temp_r=buf[0]>>1*. Implementacja powyższego równania pozornie wymaga zastosowania arytmetyki zmiennoprzecinkowej (np. zmiennych typu *float*), ponieważ jednak wystarczająca jest dokładność na poziomie jednego miejsca po przecinku - można wykorzystać zmienne całkowite z odpowiednim przesunięciem zakresu (dziesiąte części stają się jednostkami - czyli otrzymana wartość jest w efekcie 10-krotnie większa od wartości rzeczywistej). Taką właśnie ideę realizuje funkcja przedstawiona na **list. 4**.

Rzeczywistą wartość temperatury otrzymuje się przez podzielenie wartości zwracanej przez funkcję przez 10 (np. dla temperatury 21,3°C funkcja zwróci wartość 213).

List. 4. Funkcja obliczająca temperaturę z rozdzielczością 0,1°C

```
int GetTemp(char *buf)
{ int k,temp_r;

    temp_r=buf[0];
    temp_r>>=1;
     // przesuniecie wszystkich bitow o jeden w prawo

    if(buf[1]&0x01) temp_r-=128;
    // jeżeli temperatura ujemna

    k=((buf[7]-buf[6])*100);
    // licznik ułamka pomnożony przez 100

    k/=buf[7];
    // realizacja ułamka

    k-=25;      // odjęcie stałej 0.25 pomnożonej przez 100
    k/=10;      // wystarczy dokładność 1 miejsce po przecinku

    temp_r*=10; temp_r+=k;
    // temp_r zawiera wartosc temperatury 10-krotnie większą
    // od rzeczywistej

    return temp_r;
}
```

Prezentowany układ pokojowego regulatora temperatury można łatwo wzbogacić o dodatkowe funkcje użytkowe. Na przykład poprzez niewielką modyfikację oprogramowania i dodanie drugiego termometru DS1820 można uzyskać pomiar temperatury panującej na zewnątrz budynku. Można też dodatkowo wprowadzić funkcję kalkulacji kosztów zużytej energii elektrycznej.

Zbigniew Hajduk