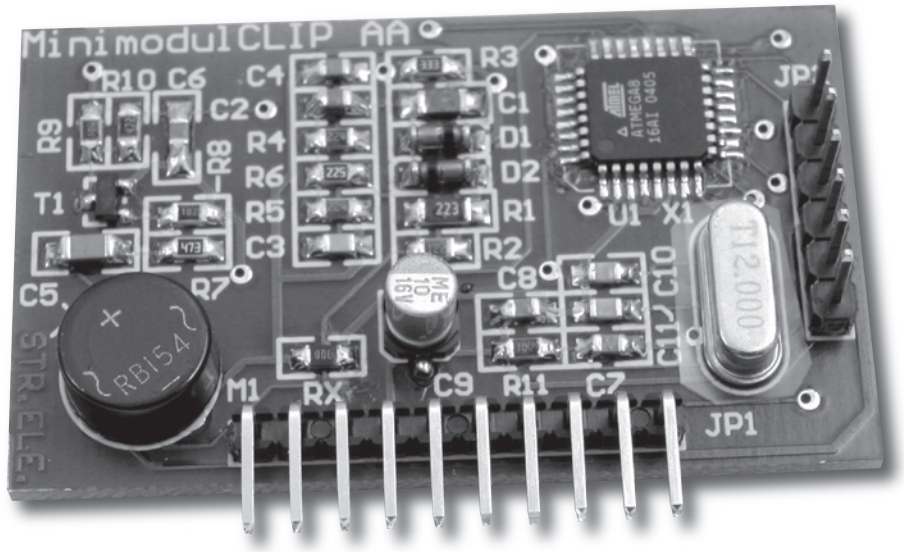
Minimoduł CLIP, część 2

AVT–955

Opublikowany w EP2 i 3/2005 opis „Programowego dekodera CLIP (FSK)” wzbudził bardzo duże zainteresowanie Czytelników Elektroniki Praktycznej. Otrzymałem wiele pytań o możliwość wykorzystania pomysłu programowego dekodowania sygnału CLIP do czegoś więcej niż budowa dekodera pokazującego numer na wyświetlaczu LCD. Wielu Czytelników chciałoby zbudować system, w którym informacja o zdekodowanym numerze przekazywana byłaby do komputera PC za pośrednictwem interfejsu RS232.

Rekomendacje:

prezentowany CLIP polecamy Czytelnikom zainteresowanym budową układów elektronicznych, w których wykorzystuje się identyfikację numeru abonenta dzwoniącego (CLIP). W artykule opisany jest układ, który dołączony do analogowej linii telefonicznej potrafi zdekodować informację CLIP oraz przesłać ją do dowolnego systemu nadrzędnego za pomocą prostego interfejsu cyfrowego.



Program dekodujący

Oprogramowanie dekodujące sygnał CLIP zostało oparte na poprzedniej wersji programu dekodera. Zostało ono skompilowane za pomocą znanego Czytelnikom kompilatora AVR–GCC w wersji 2002––06–25.

W spoczynku mikrokontroler pozostaje w trybie *powerdown*, przy

List. 1. Określenie poziomu odniesienia i warunku wyzwolenia procesu dekodowania

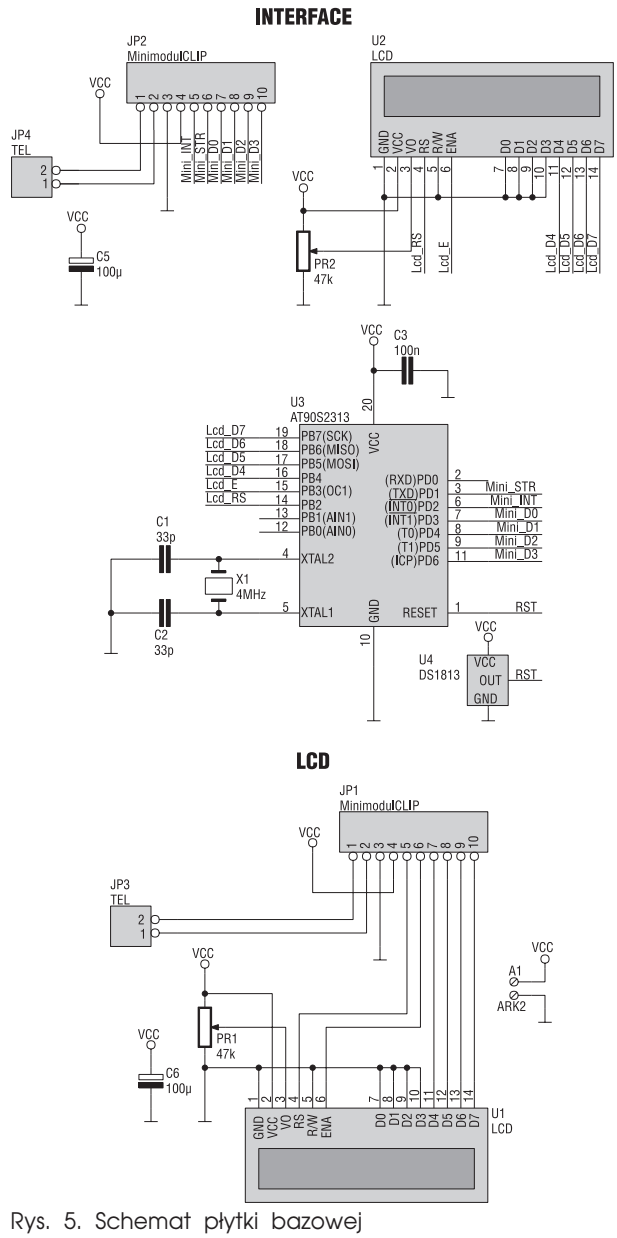
```
fTimeout=0;
TCNT1H=0xDA;
TCNT1L=0x1C;
TCCR1B=0x05;         //Timer1 ON, f=CLK/1024, no CTC

//getting reference woltage - here is no CLIP signal yet
for(iref=0;iref<5;iref++)
{
   ADCSR|=0x40;         //start conversion - here is no CLIP signal yet
   while((ADCSR & 0x40)!=0);
   adc_valL=ADCL;       //read LSB first
   adc_valH=ADCH;       //read MSB (2 bits)
   iConst=(adc_valH<<8)|adc_valL;  //reference voltage

   //Correct value
   if((iConst>460) && (iConst<560))
      break;

   Waitms(5);
}

if(iref==5)
{
   InterfaceOut_rom(PSTR(„**BIG NOISE**”));
   Waitms(500);
   continue;
}

while(1)
{
   i=TCNT1L;      //get LSB first
   i=TCNT1H;
   if(i<0xDA)     //timer1 overflow performed
   {
      fTimeout=1;
      break;
   }
   ADCSR|=0x40;      //start conversion
   while((ADCSR & 0x40)!=0);
   adc_valL=ADCL;    //read LSB first
   adc_valH=ADCH;    //read MSB (2 bits)
   iAdcVal=(adc_valH<<8)|adc_valL;
   if((iAdcVal>(iConst+5)) || (iAdcVal<(iConst-5)))
      break;
}
TCCR1B=0x00;         //Timer1 OFF

//checking timeout
if(fTimeout!=0)
{
   InterfaceOut_rom(PSTR(„**TIMEOUT CLIP**”));
   Waitms(500);
   continue;
}
```

PODSTAWOWE PARAMETRY

- Płytka o wymiarach 51 x 30 mm
- Zasilanie 5 VDC
- Tryby pracy (wymagają odmiennego oprogramowania):
  - minimoduł z interfejsem cyfrowym (magistrala)
  - minimoduł z interfejsem dla LCD

List. 2. Zasada dekodowania

```
SIGNAL (SIG_OUTPUT_COMPARE1A)
{
  ...

  if(fFirstSample)
  {
     //Initialization of static variables - counters etc.
     fFirstSample=0;
     cou=0;
     if(iAdcVal>iConst)
        tmp=1;
     else
        tmp=0;
     ClipState=CLIPSTATE_SMMR;
     SmmrLongPulseCounter=0;
     TabIndex=0;
     AvPulseCounter=0;
     AvPulseSumma=0;
     TimeoutCounter=0;

     ADCSR|=0x40;
     return;
  }

  //Sampling
  adc_valL=ADCL;            //read LSB first
  adc_valH=ADCH;            //read MSB (2 bits)
  iAdcVal=(adc_valH<<8)|adc_valL;
  if(iAdcVal>iConst)
     DigitalInput=1;
  else
     DigitalInput=0;

  ADCSR|=0x40;

  if(tmp==DigitalInput)
  {
     cou++;
  }
  else
  {
     tmp=DigitalInput;
     LastPulseMem=LastPulseTime;
     LastPulseTime=cou;
     cou=0;
  }

  ...
}
```



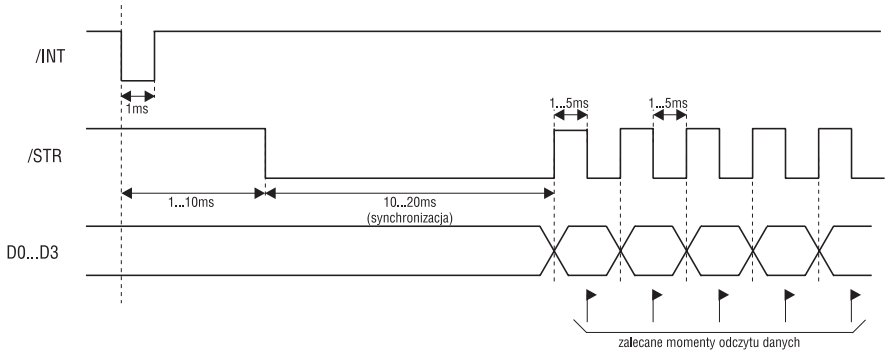
Rys. 5. Schemat płytki bazowej

czym odblokowane jest przerwanie zewnętrzne INT1 wyzwalane poziomem niskim. Pojawienie się sygnału dzwonienia powoduje wyjście procesora ze stanu uśpienia. Sygnał ten jest programowo analizowany pod kątem poprawności czasów trwania stanów wysokiego i niskiego, które powinny wynosić ok. 20 ms. Jeśli są byt długie, mikrokontroler wysyła na wyjście napis „**TIMEOUT CALL**” zakończony bajtem o wartości 0. Wyjściem jest interfejs cyfrowy, albo wyświetlacz LCD, zależnie od wersji oprogramowania jaka została załadowana do pamięci Flash mikrokontrolera U1.

Jeśli sygnał dzwonienia miał poprawną postać, mikrokontroler odczekuje (w sumie) ok. 200 ms od jego zakończenia, po czym przechodzi do wykonania fragmentu programu pokazanego na **list. 1**. W pierwszej kolejności pobierane jest napięcie odniesienia, które jest zapisywane w zmiennej globalnej *iConst*. Wartość zmiennej *iConst* odpowiada napięciu panującemu na wejściu przetwornika A/C tuż przed pojawieniem się w linii sygnału CLIP i wynosi ok. 512, czyli połowę zakresu pomiarowego. Próbkowanie poziomu odniesienia odbywa się co najwyżej pięciokrotnie w odstępach co 5 ms. Jeśli kolejny odczyt da wartość na poziomie połowy zakresu pomiarowego, to uznawany jest on za poprawny. Uznaje się, że napięcie jest na poziomie połowy zakresu pomiarowego, jeśli mieści się przedziale 512 ±10%, czyli 460...560. Taki margines z nawiązką pokrywa błędy związane z tolerancją rezystorów R4 i R5. Jeżeli pięć kolejnych prób odczytu napięcia odniesienia zakończy się niepowodzeniem, to na wyjście minimodułu jest wysyłany napis „**BIG NOISE**”, po czym mikrokontroler odczekuje 500 ms i powraca do stanu uśpienia.

Poprawne pobranie napięcia odniesienia *iConst* powoduje przejście do stanu oczekiwania na pojawienie się w linii sygnału CLIP. Mikrokontroler uznaje, że sygnał ten pojawił się, gdy napięcie na wejściu przetwornika A/C zmieni się w stosunku do poziomu odniesienia o ok. 12 mV (5/1024*2,56 V). Oczekiwanie na sygnał CLIP objęte jest stosownym czasem przeterminowania (*timeout*), który jest odmierzany za pomocą Timera1. Jeśli w tym czasie sygnał CLIP nie pojawi się, mikrokontroler wysyła na wyjście napis „**TIMEOUT CLIP**” zakończony bajtem zerowym.

Jeśli udało się określić poziom odniesienia oraz warunek wyzwolenia wystąpił w odpowiednim przedziale czasu, to mikrokontroler rozpoczyna proces dekodowania sygnału CLIP. Modyfikacja poprzedniej wersji oprogramowania polegała w tym przypadku na dodaniu obsługi przetwornika analogowo–cyfrowego oraz na programowym symulowaniu wyjścia komparatora, który w poprzednim układzie wy-



Rys. 6. Interfejs wyjściowy Minimodułu CLIP

List. 3. Implementacja interfejsu wyjściowego w Minimodule CLIP

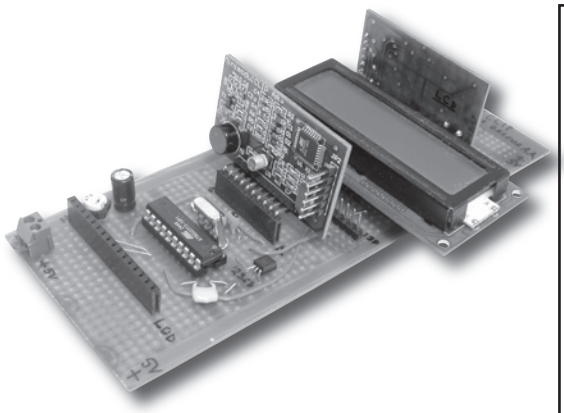
```
//Returns duration[us] on OK
//Returns 0 on Timeout
unsigned int WaitForStrobe_HIGH_End(void)
{
  unsigned int str_cou=0;
  while(STROBE!=0)
  {
     Wait2us(1);

     //Strobe High timeout = 20ms
     if(++str_cou>=10000)
        return 0;
  }

  //2*str_cou contains time of pulse in us
  return (str_cou<<1);
}

//Returns duration[us] on OK
//Returns 0 on Timeout
unsigned int WaitForStrobe_LOW_End(void)
{
  unsigned int str_cou=0;
  while(STROBE==0)
  {
     Wait2us(1);

     //Strobe Low timeout = 40ms
     if(++str_cou>=20000)
        return 0;
  }

  //2*str_cou contains time of pulse in us
  return (str_cou<<1);
}

void SetOutput(unsigned char Val_4bit)
{
  PORTD&=0x0F;
  PORTD|=(Val_4bit<<4);
}

void InterfaceOut(unsigned char *Buffer)
{
  unsigned char HalfByteCounter;
  unsigned char lenght;

  INT_CLR;
  Waitms(1);
  INT_SET;

  lenght=0;
  while(Buffer[lenght++]!=0);

  for(HalfByteCounter=0;HalfByteCounter<2*lenght;HalfByteCounter++)
  {
     unsigned int Duration;

     //Wait for end of Strobe High pulse
     if((Duration=WaitForStrobe_HIGH_End())==0)
     {
        Waitms(2*lenght+20);
        return;
     }

     //Wait for end of Strobe Low pulse
     //if > 10ms then synchronize to begining
     if((Duration=WaitForStrobe_LOW_End())==0)
     {
        Waitms(2*lenght+20);
        return;
     }

     if(Duration>10000)
        HalfByteCounter=0;

     if((HalfByteCounter%2)==0)
     {
        //MS half-byte
        SetOutput(Buffer[HalfByteCounter/2]>>4);
     }
     else
     {
        //LS half-byte
        SetOutput(Buffer[HalfByteCounter/2]&0x0F);
     }
  }
}
```

krywał dodatnie i ujemne połówki sygnału CLIP. Na **list. 2** przedstawiłem najistotniejszy element modyfikacji starego oprogramowania. Konwersja analogowo–cyfrowa inicjowana jest przy każdym występującym co 26 µs przerwaniu od porównania wartości rejestru licznikowego Timera1 z rejestrem OCR1A. Wynik tej konwersji analizowany jest podczas obsługi następnego przerwania. Daje to przetwornikowi A/C czas 26 µs na dokonanie konwersji. Jej wynik zawarty w zmiennej *iAdcVal* jest porównywany z otrzymaną wcześniej wartością napięcia odniesienia *iConst*. W zależności od relacji pomiędzy wartościami zmiennych *iAdcVal* i *iConst*, zmiennej *DigitalInput* nadawana jest wartość 0 albo 1. Zmienna ta, jak sama nazwa wskazuje, symuluje wejście cyfrowe (port) na którym w poprzedniej wersji dekodera pojawiał się sygnał pochodzący z wyjścia wspomnianego komparatora. Jeśli wartość chwilowa przebiegu CLIP jest większa od napięcia spoczynkowego (dodatnia połówka), to zmienna *DigitalInput* przyjmuje wartość 1. W przeciwnym wypadku (ujemna połówka) nadawana jest jej wartość 0. Dzięki temu program jest w stanie zmierzyć chwilową częstotliwość przebiegu CLIP, a co za tym idzie – określić, czy transmitowany bit ma wartość 0, czy 1.

Proces dekodowania objęty jest globalnym przeterminowaniem wynoszącym ok. 1 s, oprócz tego zastosowałem w nim kilka innych zabezpieczeń przed różnymi nieprawidłowościami. Po zakończeniu dekodowania procesor sprawdza za pomocą funkcji *ErrorCheck()*, czy w jego trakcie nie wystąpiły błędy. W szczególności sprawdzana jest suma kontrolna CHECKSUM. Jeśli dekodowanie zakończyło się błędami, na wyjście minimodułu wysyłany jest napis „**ERROR**” zakończony bajtem o wartości 0. W przeciwnym razie trafia tam odczytany numer abonenta dzwoniącego oraz aktualna data i godzina.

Wyjście LCD

Oprogramowanie *Minimodułu CLIP* przygotowano w dwóch wersjach, różniących się sposobem prezentacji zdekodowanego numeru telefonu lub informacji o błędzie. Projekt o nazwie *Mini-*



Fot. 7. Widok płytki bazowej

modulCLIP_LCD umożliwia bezpośrednie dołączenie do minimodułu wyświetlacza alfanumerycznego LCD zgodnego ze specyfikacją interfejsu HD44780 firmy Hitachi. Sposób połączenia minimodułu z wyświetlaczem przedstawiłem na **rys. 5** w części oznaczonej napisem „LCD". Dzięki takiemu rozwiązaniu możliwe jest błyskawiczne zbudowanie prostego dekodera CLIP na bazie opisywanego minimodułu.

Wyjście *Interface*

Drugą wersją oprogramowania Minimodułu CLIP jest projekt *MinimodulCLIP_Interface*. Zamiast sterownika wyświetlacza LCD ma on zaimplementowany prosty interfejs cyfrowy mający charakter magistrali (pliki *Interface.h* i *Interface.c*). Interfejs ten realizowany jest w minimodule przez funkcję *void InterfaceOut(unsigned char *Buffer);* przedstawioną, wraz z funkcjami pomocniczymi, na **list. 3**. Na **rys. 6** przedstawiłem przebiegi czasowe jakie występują w tym interfejsie podczas odczytywania danych z minimodułu przez układ nadrzędny.

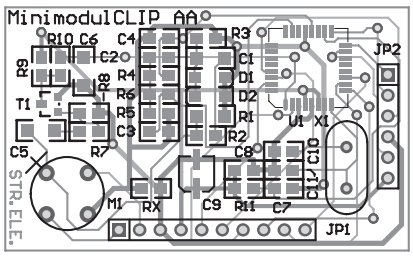
Zasada działania interfejsu jest następująca. Po zakończeniu procesu dekodowania sygnału CLIP lub w sytuacji wystąpienia błędu, minimoduł wywołuje funkcję *InterfaceOut()* z argumentem, którym jest wskaźnik do ciągu znaków jaki ma być przesłany do układu nadrzędnego. Ciąg ten musi być zakończony bajtem o wartości 0. Przykładowo – w przypadku stwierdzenia niewłaściwego sygnału dzwonienia przekazywany ciąg ma postać „**TIMEOUT CALL**"+0x00. Jeśli dekodowanie zakończy się powodzeniem, opisywany ciąg znaków ma następującą postać:

List. 4. Przykładowe oprogramowanie układu nadrzędnego napisane w języku C (AVR-GCC)

```
// Function:  Interface example for CLIP FSK (Caller ID Presentation) Decoder (Minimodul CLIP)
// Target: at90s2313
// Quartz: 4MHz
// Author: Arkadiusz Antoniak @ 2005, Poland

#include <io2313.h>
#include „LCD_44780.h"
#include <progmem.h>
#include <sig-avr.h>

//========================================== PORT ALIASES ========================
=======
#define STROBE_SET PORTD|=(1<<1)
#define STROBE_CLR PORTD&=~(1<<1)

//========================================== MACROS ==============================
=======
#define MAX_NR_DATA_BYTES 64

//========================================== HEADERS =============================
=======
void Strobe(unsigned char msTime);

//========================================== GLOBAL ==============================
=======

//========================================== INTERRUPT HANDLERS ==================
=======
SIGNAL (SIG_INTERRUPT0)
{
  unsigned char DataTable[MAX_NR_DATA_BYTES];
  unsigned char i;

  STROBE_SET;
  Waitms(10);

  //i - byte counter
  for(i=0;i<MAX_NR_DATA_BYTES;i++)
  {
     unsigned char byte;

     //i=0 -> Init transmission
     (i==0)?Strobe(20):Strobe(1);

     //MS half-byte (<<1 = >>3 + <<4)
     byte=(PIND&0x78)<<1;

     //1ms strobe pulse
     Strobe(1);

     //LS half-byte
     byte|=(PIND&0x78)>>3;

     //Store read byte
     DataTable[i]=byte;

     if(byte==0)
        break;
  }

  //Here we use data stored in DataTable
  //For example - let's show it on LCD in 2 lines (separated by semicolon - ';')
  Lcd_clear();
  for(i=0;i<MAX_NR_DATA_BYTES;i++)
  {
     if(DataTable[i]==';')
     {
        DataTable[i]=0;
        Lcd_write(DataTable);
        Second_line();
        Lcd_write(DataTable+i+1);
        Waitms(5000);
        break;
     }

     if(DataTable[i]==0)
     {
        Lcd_write(DataTable);
        Waitms(5000);
        break;
     }
  }

  Lcd_clear();
  Lcd_write_rom(PSTR(„AVR-GCC: waiting"));

  //Clear INT0 flag
  GIFR|=0x40;
}

//========================================== BASIC ROUTINES ======================
========
void Strobe(unsigned char msTime)
{
  //msTime-long STROBE pulse
  STROBE_CLR;
  Waitms(msTime);
  STROBE_SET;
  Waitms(1);
}

void Init(void)
{
```

cd. list. 4.

```
  DDRB=0xFF;
  DDRD=0x83;
  //PORTD|=(1<<2);
  PORTD=0xFF;

  //Write 1 to strobe
  STROBE_SET;

  //INT0 - falling edge
  MCUCR=0x02;

  //Enable INT0
  GIMSK|=1<<INT0;  //Enable interrupts
  asm(„sei”);

  Lcd_init();
}

int main(void)
{
  Init();

  Lcd_clear();
  Lcd_write_rom(PSTR(„’Minimodul
CLIP’”));
  Second_line();
  Lcd_write_rom(PSTR(„AVR-GCC In-
terf.”));

  while(1);

  return 1;
}
```



Rys. 8. Schemat montażowy

„NUMER_TELEFONU;DD.MM hh:mm”+0x00

gdzie:

NUMER_TELEFONU – ciąg cyfr ASCII określających odczytany numer telefonu;

DD – dwie cyfry ASCII określające dzień miesiąca;

MM – dwie cyfry ASCII określające miesiąc;

hh – dwie cyfry ASCII określające godzinę;

mm – dwie cyfry ASCII określające minutę.

Numer telefonu jest oddzielony od informacji o dacie i godzinie za pomocą znaku średnika „;”, zaś informacja o czasie jest oddzielona od daty za pomocą pojedynczego znaku spacji (0x20).

Funkcja *InterfaceOut()*, w celu przesłania tych danych do układu nadrzędnego, wytwarza na wyjściu /INT impuls ujemny o czasie trwania ok. 1 ms. Impuls ten stanowi informację o tym, że nadeszła nowa transmisja danych. Jeśli układem nadrzędnym jest mikrokontroler, końcówkę /INT można połączyć z wejściem przerwania zewnętrznego. Następnie funkcja *InterfaceOut()* oczekuje na ujemne impulsy strobujące na wejściu /STR. Po każdym impulsie na wejściu /STR trwającym krócej niż 10 ms, na linie D0...D3 wystawiany jest kolejny półbajt (4 bity) przesyłanego ciągu znaków. Obowiązuje przy tym zasada „najpierw bardziej znaczący, a potem mniej znaczący półbajt kolejnego bajta”. Jeśli układ nadrzędny wytworzy na wejściu /STR impuls ujemny dłuższy niż 10 ms, to interfejs synchronizuje się i na linie D0...D3 wystawiany jest bardziej znaczący półbajt pierwszego bajta ciągu znaków. Opisana akcja kończy się, gdy przesłany zostanie bajt o wartości 0.

Należy zwrócić uwagę na ograniczenia czasowe jakie funkcja *InterfaceOut()* nakłada na czasy trwania impulsów stanu wysokiego i niskiego na wejściu /STR. Wynoszą one odpowiednio 20 ms i 40 ms. Wytworzenie przez układ nadrzędny impulsów dłuższych jest uznawane za błąd interfejsu i skutkuje zakończeniem funkcji *InterfaceOut()* oraz przejściem minimodułu w stan czuwania.

Na **rys. 5**, w części oznaczonej „INTERFACE”, przedstawiłem prosty przykład układu nadrzędnego w postaci mikrokontrolera AT90S2313. Mikrokontroler ten odczytuje informacje *z Minimodułu CLIP*, po czym – w celu czysto demonstracyjnym – wysyła je na wyświetlacz LCD. Oczywiście praktyczny układ nadrzędny wykorzysta odczytane dane w inny sposób. **List. 4** i **list. 5** zawierają przykładowe programy implementujące interfejs w układzie nadrzędnym z mikrokontrolerem AT90S2313. Zostały one napisane w językach C i BascomAVR. Przykład napisany w języku C może być z niewielkimi zmianami przeniesiony na mikrokontroler nie należący do rodziny AVR.

Montaż i uruchomienie

Schemat montażowy minimodułu pokazano na **rys. 8**. Układ montujemy na niewielkiej płytce drukowanej wykorzystując w większości elementy SMD. Z ich przylutowaniem do płytki nie powinno być najmniejszych problemów. Wystarczy do tego zwyczajna lutownica, taka jak do elementów przewlekanych, cienka cyna (najlepiej 0,5 mm) i odrobina skupienia. Naj-

List. 5. Przykładowe oprogramowanie układu nadrzędnego napisane w języku BascomAVR

```
' Function: Interface example for CLIP
FSK (Caller ID Presentation) Decoder
(Minimodul CLIP)
' Target:   at90s2313
' Quartz:   4MHz
' Compiler: BascomAVR 1.11.7.7 (Demo)
' Author:   Arkadiusz Antoniak @ 2005,
Poland

$regfile = „2313def.dat”
$crystal = 4000000

Strobe_pin Alias Portd.1
Const Max_nr_data_bytes = 64

Declare Sub Strobe(byval Mstime As
Byte)

Dim I As Byte , Tmp As Byte , One_byte
As Byte
Dim Databyte(Max_nr_data_bytes) As
Byte

Config Portb = Output
Config Portd = &H83
Portd = &HFF

Config Lcdpin = Pin , Db4 = Portb.4 ,
Db5 = Portb.5 , Db6 = Portb.6 , Db7 =
Portb.7 , E = Portb.3 , Rs = Portb.2
Config Lcd = 16 * 2

On Int0 Int0_routine
Enable Int0
Enable Interrupts

Cls
Lcd „’Minimodul CLIP’”
Lowerline
Lcd „Bascom Interf.”

Do
Loop
'************************************
******
'
'  INT0 interrupt routine
'
'************************************
******
Int0_routine:
   Strobe_pin = 1
   Waitms 10

   For I = 1 To Max_nr_data_bytes
      If I = 1 Then
         Call Strobe(20)
      Else
         Call Strobe(1)
      End If

      Tmp = Pind And &H78
      One_byte = 2 * Tmp

      Call Strobe(1)

      Tmp = Pind And &H78
      Tmp = Tmp / 8
      One_byte = One_byte Or Tmp

      Databyte(i) = One_byte

      If One_byte = 0 Then Exit For
   Next I

   Cls
   For I = 1 To Max_nr_data_bytes
      If Databyte(i) = 0 Then Exit For

      If Databyte(i) <> Asc( „;”) Then
         Lcd Chr(databyte(i))
      Else
         Lowerline
      End If
   Next I
   Waitms 5000
   Cls
   Lcd „Bascom: waiting”
Return
'************************************
******
'
'  Generating strobe signal
'  mstime - time of strobe in milise-
conds
'
'************************************
******
Sub Strobe(mstime As Byte)
   Strobe_pin = 0
   Waitms Mstime
   Strobe_pin = 1
   Waitms 1
End Sub
```

więcej problemów może sprawić przylutowanie mikrokontrolera U1 zamkniętego w obudowie TQFP32. Jest kilka wypróbowanych metod amatorskiego montażu takich elementów. Ja w układzie modelowym przylutowałem tą kostkę w sposób następujący. Przy użyciu małego pędzelka naniosłem na punkty lutownicze odrobinę kalafonii rozpuszczonej w etanolu (może być denaturat lub spirytus spożywczy). Przy nanoszeniu kalafonii proponuję zrezygnować z rozpuszczania bryłki tej substancji w niewielkiej ilości alkoholu. Zamiast tego należy postąpić podobnie jak robią to malarze malujący swe obrazy. Najpierw moczymy koniuszek pędzelka w spirytusie, a potem nabieramy na niego kalafonię (podobnie jakbyśmy nabierali farbę z palety).

Po naniesieniu kalafonii na punkty lutownicze należy odczekać kilkanaście minut aż alkohol odparuje. Następnie układamy na płytce mikrokontroler i pozycjonujemy go dokładnie w miejscu przeznaczenia. Aby podczas lutowania pozostawał stabilny, lutujemy do płytki cztery skrajne nóżki każdego z czterech rzędów wyprowadzeń. Właściwe lutowanie polega na zrobieniu swego rodzaju minifali za pomocą zwykłego grota lutownicy. W tym celu – w obecności śladowych ilości cyny – przesuwamy grot wzdłuż rzędu wyprowadzeń dotykając nim miejsc, gdzie nóżki układu stykają się z punktami lutowniczymi. Niewielka ilość cyny oraz duża ilość topnika (zwiększenie współczynnika napięcia powierzchniowego płynnego lutowia) zapobiegają zwarciom. Nasączona topnikiem cyna ma tendencję do zbijania się w małe kulki i zwilża jedynie dane wyprowadzenie układu ze znajdującym się pod nim punktem lutowniczym. Wszelkie ewentualne zwarcia z łatwością usuniemy za pomocą specjalnej plecionki do rozlutowywania. Na koniec dokładnie oglądamy pod lupą efekt naszej pracy, badając czy nie pozostały jakiekolwiek zwarcia.

Podczas montażu nie wolno zapomnieć o wlutowaniu rezystora RX, który łączy masy analogową i cyfrową w pobliżu końcówki 3 złącza *Interface*. W tym przypadku proponuję zrezygnować z kropelki cyny i wlutować normalną zworkę (rezystor o nominale 0R) ponieważ pomiędzy punktami lutowniczymi tego elementu poprowadzona jest ścieżka.

Rezonator kwarcowy X1 powinien mieć wysokość 4 mm. Kwestią wyboru jest rodzaj zastosowanego złącza ISP. Ja w modelach wlutowałem rząd sześciu goldpinów aby ułatwić sobie pracę nad oprogramowaniem minimodułu. Czytelnicy, którzy zechcą jedynie wykonać działający egzemplarz układu, będą musieli dokonać programowania kostki U1 jeden raz. W takiej sytuacji można zrezygnować z goldpinów, a podczas programowania pamięci Flash dotknąć stykami złącza programatora do punktów lutowniczych złącza JP2.

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