Alternatywa dla alfanumerycznych wyświetlaczy LCD

Jednym z ważniejszych elementów urządzeń sterowanych mikrokontrolerem jest interfejs użytkownika. Od tego, z jakich komponentów jest zbudowany i jak jest przemyślany zależy komfort, ale także w wielu przypadkach bezpieczeństwo obsługi urządzenia. W systemach mikroprocesorowych interfejs użytkownika jest najczęściej realizowany z użyciem wszelkiego rodzaju wyświetlaczy.

Rekomendacje:
doskonała propozycja zastąpienia typowego alfanumerycznego wyświetlacza LCD nowocześniejszym wyświetlaczem graficznym, oferującym o wiele większe możliwości prezentacji komunikatów.

Dodatkowe materiały do artykułu publikujemy na CD-EP oraz www.ep.com.pl

Wyświetlacz opisany w artykule udostępniła redakcji firma Artronic S.J., ul. Parkowa 6, 81-549 Gdynia, www.artronic.pl. Telefony do Działu Handlowego: 58-668-57-83, 58-668-57-84.

Jednym z najważniejszych elementów interfejsu użytkownika jest wyświetlacz. W pierwszych systemach z mikroprocesorami i mikrokontrolerami królowały siedmiosegmentowe, numeryczne wyświetlacze LED. Nadawały się znakomicie do wyświetlania danych numerycznych, chociaż próbowano je również (z różnym skutkiem) wykorzystywać do wyświetlania komunikatów tekstowych.

Dopiero zastosowanie wyświetlaczy alfanumerycznych LCD pozwoliło konstruktorom na opracowanie prostego, a jednocześnie czytelnego interfejsu. Taki wyświetlacz ma wbudowany sterownik, pobiera mało energii i potrafi wyświetlić znaki tekstu i cyfry. Początkowo stosowanie wyświetlaczy alfanumerycznych było mocno ograniczane wysoką ceną. Wraz z rozwojem technologii ceny drastycznie spadły i stosowanie wyświetlaczy LCD stało się powszechne. Ich zaletą mającą wpływ na popularność jest również powszechnie stosowany sterownik HD44780.

W niedługim czasie po wyświetlaczach alfanumerycznych pojawiła się następna generacja wyświetlaczy: wyświetlacze graficzne. Można na nich wyświetlać nie tylko cyfry i znaki tekstowe, ale też elementy grafiki: wykresy, ikony, fragmenty schematów itp. Aktualnie jesteśmy w okresie gwałtownego spadku cen wyświetlaczy graficznych, a to oznacza, że konstruktorzy niebawem zaczną je masowo stosować w swoich projektach, mimo trudniejszej obsługi i różnorodności stosowanych sterowników. Sprzyjać temu będą też rosnące zasoby mikrokontrolerów, a szczególnie pamięci programu niezbędnej do przechowywania wyświetlanych bitmap.

Parametrem, który szczególnie decyduje o masowym stosowaniu atrakcyjnych elementów są niskie ceny. Ich użycie w nowym projekcie nie stanowi problemu, jednak w przypadku zastosowania nowego wyświetlacza w starym urządzeniu zachodzi konieczność modyfikacji interfejsu użytkownika. Wyobraźmy sobie sytuację, gdy mamy zaprojektowane kompletne urządzenie z wyświetlaczem alfanumerycznym i chcemy unowocześnić sposób komunikacji z użytkownikiem poprzez wymianę wyświetlacza na graficz-

ny. Trzeba będzie zmodyfikować układ elektryczny (interfejs fizyczny), a więc zmienić projekt płytki i program oraz zmienić obudowę, bo zazwyczaj matryce wyświetlaczy graficznych mają inne wymiary niż stosowane wcześniej wyświetlacze alfanumeryczne. Paradoksalnie, to nie zmiana elektroniki i oprogramowania sprawi nam najwięcej kłopotu. W gotowym urządzeniu często bardzo duży udział w kosztach mają elementy mechaniczne, a szczególnie obudowa. Jeżeli obudowa jest zaprojektowana optymalnie i jest produkowana w większych ilościach, to najlepiej jest jej nie zmieniać. Więc co zrobić, kiedy nie chcemy tworzyć nowego projektu obudowy i jednocześnie użyć wyświetlacza graficznego. Odpowiedzią na to jest zastosowanie opisywanego tutaj wyświetlacza HY–12232 z matrycą o rozmiarach 122x32 piksele. Ekran wyświetlacza ma wymiary 27x65,6 mm, równe lub zbliżone do wymiarów często stosowanych wyświetlaczy alfanumerycznych 2x16 znaków lub 1x16 znaków.

Interfejs wyświetlacza

Wiadomo, że po wymianie wyświetlacza niezbędne mogą się okazać zmiany układowe. Wyświetlacz ze sterownikiem HD44780 może być sterowany magistralą 8–bitową uzupełnioną o 3 linie sterujące: E, R/W i RS. Maksymalnie potrzebnych jest 11 linii sterujących, jednak często jest wykorzystywany tryb pracy z 4 liniami danych i 2 liniami sterującymi E i RS. Linia R/W jest połączona do masy na stałe, a program zamiast odczytywać bit zajętości odmierza konieczne opóźnienia.

Wyświetlacz HY12232 jest sterowany dwoma sterownikami NJU6450. Magistrala sterownika NJU6450 akceptuje dane w formatach magistrali zgodnej z systemami Intel 8080 lub Motorola 6800. Rodzaj magistrali jest ustawiany sprzętowo (wymuszenie stanu na wejściu sterownika). Zazwyczaj producent wyświetlacza ustawia na stałe rodzaj magistrali i nie można jej zmienić. Tak jest i w tym przypadku: wyświetlacz NJU6450 pracuje z magistralą zgodną z Motorola6800.

Interfejs sterujący składa się z 8–bitowej magistrali danych uzupełnionych o linie sterujące A0, E1, E2, RW i RES (**tab. 1**). Magistrala danych może pracować tylko w trybie 8–bitowym. Jak widać nie można zastąpić wyświetlaczy bez zmian układowych (zwiększenia linii portów interfejsu sterującego).

Tab. 1. Opis interfejsu wyświetlacza NJU6450

Linia	Opis
D0	Dwukierunkowa linia danych D0
D1	Dwukierunkowa linia danych D1
D2	Dwukierunkowa linia danych D2
D3	Dwukierunkowa linia danych D3
D4	Dwukierunkowa linia danych D4
D5	Dwukierunkowa linia danych D5
D6	Dwukierunkowa linia danych D6
D7	Dwukierunkowa linia danych D7
A0	Linia wyboru rejestru: „H” rejestr danych, „L” rejestr instrukcji
E1	Linia wyboru sterownika lewej strony panelu LCD
E2	Linia wyboru sterownika prawej strony panelu LCD
R/W	Odczyt („H”) zapis („L”) 8–bitowej danej na magistrali
RES	Zerowanie sterownika

Wszystkie linie oprócz E1 i E2 są równolegle połączone do obu sterowników wyświetlacza. Na **rys. 1** pokazano przebiegi czasowe w trakcie przesyłania danych przez magistralę Motorola 6800. Linie wyboru rejestru A0 i kierunku przepływu danych R/W są ustawiane, kiedy na linii E jest stan niski. Wystawienie stanu wysokiego na linii E powoduje pojawienie się danych do odczytania na magistrali. Dane przeznaczone do zapisania powinny być ustawione i stabilne, kiedy E jest w stanie wysokim Opadające zbocze na linii E kończy cykl zapisu i odczytu danych. Linia RES jest wykorzystywana do zerowania sterownika wyświetlacza (aktywny stan niski).

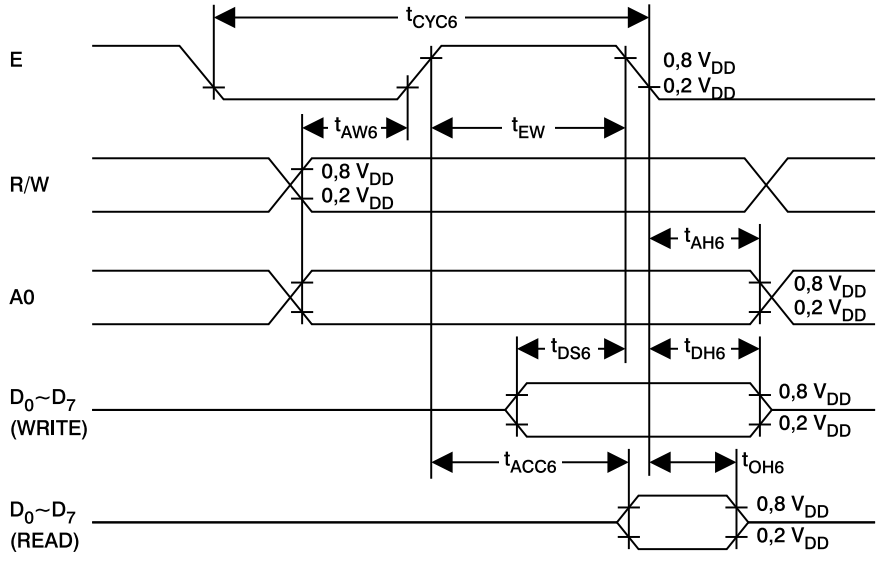
Pamięć RAM wyświetlacza

Pamięć sterownika ma pojemność 2560 bitów i jest zorganizowana w 4 banki (strony pamięci). Każda strona ma pojemność 80 bajtów. Wpisanie bajtu do pamięci wyświetlacza powoduje wyświetlenie pionowego paska o długości 8 pikseli (**rys. 2**).

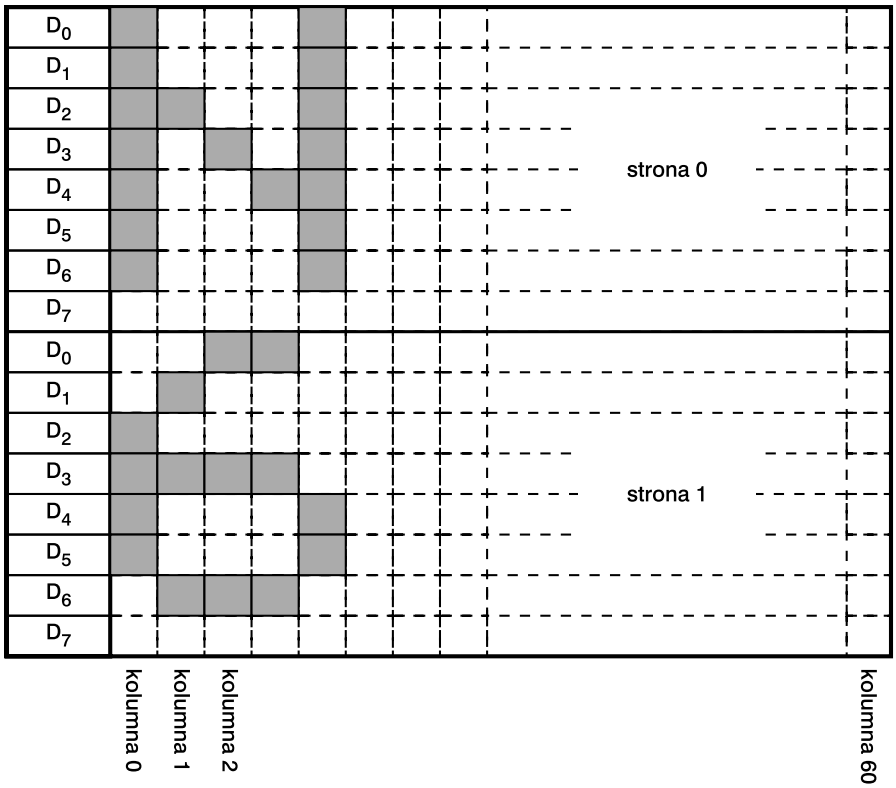
W wyświetlaczu HY12232 w każdej ze stron pamięci sterownika wykorzystywanych jest 61 bajtów. Ponieważ zastosowano tu 2 sterowniki (jeden dla lewej połowy wyświetlacza, drugi dla prawej), to można w ten sposób sterować matrycą o szerokości 122 pikseli.

Po zapisaniu wszystkich 4 stron pamięci wyświetlany jest obszar 122x32 piksele (dla obu sterowników).

Pamięć RAM jest adresowana przez zapisanie licznika kolumn i licznika stron pamięci. Licznik kolumn określa pozycję wyświetla-



Rys. 1. Sygnały na magistrali wyświetlacza podczas przesyłania danych



Rys. 2. Zależność pomiędzy zawartością pamięci i wyświetlaną informacją

nego paska od długości 8 pikseli w linijce wskazywanej przez numer strony.

Komendy sterownika

Sterownie pracą wyświetlacza odbywa się przez wysyłanie do niego komend. Sterownik interpretuje przesyłane do niego dane jako komendę, kiedy linie A0 i R/W są w stanie niskim.

W **tab. 2** pokazano zestaw wszystkich komend sterownika NJU6450A.

Działanie większości komend jest jasne po przestudiowaniu informacji zawartych w tab. 2. Dodatkowych wyjaśnień wymagają komendy *ADC Select* i *Read/Modify/Write*.

Komenda *ADC Select* ustawia zależność pomiędzy adresem kolumny w pamięci RAM wyświetlacza i wyjściem drivera segmentu. W trybie normalnym driver segmentu odpowiadającego skrajnemu lewemu położeniu na ekranie odpowiada pierwszej lokacji w banku pamięci. W trybie inwersji driver segmentu odpowiadającego skrajnemu prawemu położeniu na ekranie odpowiada pierwszej lokacji w banku pamięci.

Komenda ReadModify/Write jest używana do wprowadzania sterownika w tryb ułatwiający modyfikację wyświetlanej informacji na podstawie zawartości wcześniej zapisanej pamięci RAM. Po wysłaniu komendy i odczytaniu danej z pamięci nie jest wykonywana automatyczna inkrementacja licznika kolumn. Kiedy odczytana dana zostanie zmodyfikowana i ponownie zapisana, licznik kolumn jest inkrementowany i wskazuje na następną kolumnę. Ten tryb jest wyłączany po wysłaniu komendy *End*. Kiedy tryb wprowadzany przez komendę *ReadModify/Write* jest wyłączony, to po każdym odczytaniu i zapisaniu danej do pamięci RAM licznik kolumn jest inkrementowany.

Zasilanie i podłączenie do mikrokontrolera

Wyświetlacz HY–12232 jest zasilany napięciem +5 V. Z jednej strony może to stanowić pewną niedogodność w systemach zasilanych napięciem +3,3 V, bo konieczne jest dodatkowe napięcie zasilania. Jednak, jeżeli traktujemy ten wyświetlacz jako zamiennik wyświetlacza alfanumerycznego ze sterownikiem HD44780 standardowo zasilanego +5 V, to takie zasilanie może być zaletą. W zestawie komend nie ma komendy ustawiającej kontrast, ale regulacja napięcia zasilającego matrycę jest wykonywana za pomocą zewnętrznego potencjometru połączonego pomiędzy wyprowadzenie Vo i masę Vss.

Testy wyświetlacza były wykonywane po podłączeniu wyświetlacza do linii portów mikrokontrolera LPC2148 z zestawu ZL9ARM + dipARM2148. Sposób podłączenia został pokazany w **tab. 3**.

List. 1. Definicje linii sterujących magistrali i sekwencja zerowania

```
//definicja linii magistrali
#define E1 1<<17
#define E2 1<<18
#define RW 1<<19
#define A0 1<<16
#define RES 1<<20
//definicja wyboru sterowników
#define IC1 0
#define IC2 1
IO0DIR=0x00FF0000;//linie sterujące P0.16...P0.23 jako wyjścia
  IOCLR1=0;
  IOSET1=0;
  IOCLR0|=RES;    //sekwencja zerowania
  delay();delay();
  IOSET0|=RES;
```

List. 2. Funkcja zapisująca dane do sterownika

```
void LcdBusWrite(unsigned char data, unsigned char chip){
  IOCLR0|=E1;delay();           //E1=0
  IOCLR0|=E2;delay();           //E2=0
  if(chip==IC1)
  IOSET0|=E1;delay();
  if(chip==IC2)
  IOSET0|=E2;delay();

  IOCLR0|=RW;delay();           //RW=0 zapis danej
  IO1DIR|=(0xff<<16);           //bity P1.16....P1.23 wyjscie
  int temp=(data<<16);
  IOCLR1=0x00ff0000;delay();    //zeruj bity P1.16...P1.23
  IOSET1=temp; delay();delay(); //dane na swoim miejscu

  IOCLR0|=E1;delay();           //E1=0
  IOCLR0|=E2;delay();           //E2=0
  delay();
}
```

Tabela 2 Komendy sterownika NJU6450

Komenda	Kod										Opis
	A0	R/W	D7	D6	D5	D4	D3	D2	D1	D0	
Display ON/OFF	0	0	1	0	1			0 1 1 1 0 1			Ekran wyłączony Ekran włączony
Display Start line	0	0	1	1 0 Numer linii początkowej (1...31)							Określa numer linii przypisanej bitowi D0 ze strony 0
Column address set	0	0	0	Numer kolumny (0...79)							Zapisanie licznika kolumn
Page address set	0	0	1	0	1	0	1	0	Numer strony		Zapisanie licznika stron pamięci
Status Read	0	1	Busy	ADC	ON OFF	RESET	0	0	0	0	Odczytanie statusu BUSY: 1 – zajęty, 0 – gotowy ADC: kierunek wyświetlania ON/OFF: 1 – wyświetlacz wyłączony, 0 – włączony RESET: 1 – trwa sekwencja zerowania, 0 – sekwencja zerowania zakończona
Write display data	1	0	Zapisywana dana								Zapisanie pamięci RAM wyświetlacza
Read Display Data	1	0	Odczytywana dana								Odczytywanie zawartości pamięci RAM wyświetlacza.
ADC select	0	0	1	0	1	0	0	0	0	0 1	Tryb normalny Tryb inwersji
Static drive ON/OFF	0	0	1	0	1	0	0	1	0	0 1	Tryb dynamiczny Tryb statyczny
Read/Modify/ Write	0	0	1	1	1	0	0	0	0	0	Inkrementowanie licznika kolumn przy zapisywaniu danych. Przy odczytywaniu brak inkrementacji
End	0	0	1	1	1	0	1	1	1	0	Koniec trybu read/modify/ write
Duty Ratio select	0	0	1	0	1	0	1	0	0	0/1	Programowanie cyklu wyświetlania 0 – 1/16, 1 – 1/32
Reset	0	0	1	1	1	0	0	0	1	0	Programowe zerowanie sterownika
Power save	0 0	0 0	1 1	0 0	1 1	0 0	1 0	1 1	1 0	0 1	Tryb oszczędzania energii

Programowa obsługa wyświetlacza

Przed programową inicjacją sterownika wszystkie linie sterujące połączone do linii portu P0 trzeba zaprogramować jako wyjściowe i wykonać sekwencję zerowania sterownika (**list. 1**). Sekwencję zerowania wykonuje się przez wymuszenie ⎍ impulsu na linii RST.

Magistrala sterująca wyświetlaczem (zgodna z systemem Motorola 6800) jest emulowana programowo. Wszystkie linie magistrali (oprócz linii wyboru sterownika E1 i E2) są połączone równolegle do obu sterowników wyświetlacza. Procedura *LcdBusWrite* pokazana na **list. 2** zapisuje 8–bitowe dane umieszczone w argumencie `data` do sterownika określonego przez argument `chip`. Przed jej wywołaniem musi być odpowiednio ustawiona linia wyboru rejestru A0 (dane do rejestru komend – A0=0 lub pamięci RAM – A0=1).

Zależnie od wartości argumentu `chip`, najpierw jest ustawiana w stanie wysokim linia E1 lub E2. Ponieważ procedura zapisuje dane do sterownika, to linia R/W musi być wyzerowana. Po wystawieniu danej na linie portu P1.16...P1.23 linia wyboru sterownika przechodzi w stan niski kończąc sekwencję zapisu danych.

Przed zapisaniem każdej danej do sterownika NJU6450 trzeba sprawdzić czy nie jest ustawiona flaga BUSY w rejestrze statusowym. Zastosowanie 2 sterowników powoduje, że konieczne jest oddzielne sprawdzanie zajętości w każdym ze sterowników. Procedura *CheckBusy* (**list. 3**) odczytuje rejestr statusowy sterownika określonego w argumencie `chip`. Ponieważ dane będą odczytywane ze sterownika, to linia RW musi być w stanie wysokim, a linia A0 w stanie niskim. Linie E1 i E2 są sterowane zależnie od wartości argumentu `chip`. Procedura zwraca wartość 0x00, kiedy sterownik może akceptować dane i wartość 0x01, kiedy sterownik jest zajęty.

Procedura wysyłania komend (**list. 4**) również rozróżnia, do którego sterownika ma być wysłana (argument `chip`). Przed zapisaniem kodu komendy do sterownika program czeka na wyzerowanie flagi zajętości (procedura *CheckBusy*). Kiedy flaga BUSY jest wyzerowana, to jest zerowana linia A0

Tab. 3. Opis połączeń wyświetlacza z liniami portów LPC2148

Linia LPC2148	Interfejs wyświetlacza
P0.16	A0
P0.17	E1
P0.18	E2
P0.19	R/W
P0.20	RST
P1.16	DB0
P1.17	DB1
P1.18	DB2
P1.19	DB3
P1.20	DB4
P1.21	DB5
P1.22	DB6
P1.23	DB7

i dana jest zapisywana procedurą *LcdBusWrite*.

Sterownik NJU6450 nie może pracować w trybie magistrali 4–bitowej i z tego względu nie wymaga programowej inicjalizacji interfejsu takiej, jak w sterowniku HD44780. Jednak przed rozpoczęciem wyświetlania niezbędne jest wstępne zaprogramowanie sterownika przez wysłanie komend *ADC Select*, *Duty RAtio Select* i *Stanic Drive ON/OFF*. Na **list. 5** pokazano inicjalizację wykonywaną przed zapisywaniem pamięci RAM wyświetlacza. Każda z komend inicjalizacyjnych jest wysyłana oddzielnie do sterownika lewej połowy ekranu (IC1) i prawej połowy ekranu (IC2). Wartość współczynnika duty ratio programuje się na podstawie danych producenta wyświetlacza. Dla HY12232 duty ratio=1/32. Po włączeniu zasilania konieczne jest zerowanie pamięci RAM wyświetlacza, gdyż mogą się tu znaleźć przypadkowe wartości. Służy do tego procedura *LcdCls*.

Zastosowanie dwóch sterowników w jednym wyświetlaczu komplikuje jego obsługę. We wszystkich pokazanych do tej pory procedurach niezbędne było przekazanie informacji o sterowniku (argument `chip`). Użytkownik zapisując pamięć RAM wyświetlacza chciałby traktować tę pamięć jako jedną przestrzeń. Z tego powodu procedura zapisywania danych do pamięci RAM wyświetlacza musi na podstawie numeru kolumny wyświetlacza (nie sterownika) wpisać daną do odpowiedniego sterownika. Pokazana na **list. 6** procedura *LcdWriteData* ma 3 argumenty: daną do wpisania, numer kolumny i numer wiersza (strony pamięci). Numer kolumny zmienia się w zakresie 0...121, a numer wiersza w zakresie 0...3. Na podstawie numeru kolumny określane jest, do którego sterownika będzie konieczny dostęp. Jeżeli numer kolumny ma wartość z zakresu 0...60, to dane będą zapisywane do sterownika lewej części ekranu. Jeżeli numer kolumny jest większy od 60, to dane zostaną zapisane do sterownika prawej części ekranu. Przed wysłaniem danych jest zapisywany numer strony komendą *Page Address Set* (argument `page`). Licznik kolumny zapisuje procedura *LcdColumnWrite*.

Procedura *LcdColumnSet* na podstawie numeru kolumny wyświetlacza określa, w którym ze sterowników zapisać licznik kolumn. Jeżeli numer kolumny jest mniejszy od 61, to argument kolumn jest zapisywany do sterownika lewej części wyświetlacza (**list. 7**).

Wyświetlacz graficzny może wyświetlać elementy grafiki (bitmapy) lub znaki alfanumeryczne. Sterownik NJU6450 nie ma wbudowanego generatora znaków i użytkownik musi go sobie sam zdefiniować w pamięci programu mikrokontrolera. Organizacja pamięci RAM wyświetlacza ułatwia wyświetlanie znaków o wysokości 7 pikseli. Wpisanie jednego bajtu powoduje wyświetlenie pionowej linijki o długości 8 pikseli. Jeżeli przyjmiemy, że znaki będą miały rozmiar 5x7 pikseli, to można je łatwo wyświetlać wpisując 6 kolejnych bajtów z generatora znaków. Jeden piksel z każdego bajtu jest zarezerwowany na odstęp pomiędzy linijkami tekstu, a szósty (zerowy) bajt generatora znaków jest odstępem pomiędzy znakami, (dlatego wpisujemy 6 bajtów dla znaków 7x5 pikseli). Wyświetlacz może wyświetlić 4 linijki po 20 znaków, a więc cał-

List. 3. Sprawdzanie zajętości sterownika

```
unsigned char CheckBusy(unsigned char chip){
  unsigned char data;
  IOCLR0|=E1;delay();   //E1=0
  IOCLR0|=E2;delay();   //E2=0
  IOSET0|=RW;delay();   //RW=1
  IOCLR0|=A0;delay();   //czytaj status
  if(chip==IC1)
  IOSET0|=E1;delay();
  if(chip==IC2)
  IOSET0|=E2;delay();
  IO1DIR&=0xff00ffff;   //bity P0.16...P023 wejscie
  data=(IOPIN1>>16);    //odczytaj dane
  IOCLR0|=E1;delay();   //E1=0
  IOCLR0|=E2;delay();   //E2=0
  return(data>>7);
}
```

List. 4. Zapisanie komendy do rejestru komend sterownika

```
void LcdWriteCmd(unsigned char cmd,unsigned char chip){
  if(chip==IC1)
  while(CheckBusy(IC1));
  if(chip==IC2)
  while(CheckBusy(IC2));
  IOCLR0|=A0;delay(); //zapisywanie polecenia
  if(chip==IC1)
  LcdBusWrite(cmd,IC1);
  if(chip==IC2)
  LcdBusWrite(cmd,IC2);
}
```

List. 5. Inicjalizacja wyświetlacza i zerowania pamięci RAM

```
  LcdWriteCmd(0xae,IC1); //wylacz wyswietlacz
  LcdWriteCmd(0xae,IC2); //wylacz wyswietlacz
  LcdWriteCmd(0xa0,IC1); //tryb norma (ADC)
  LcdWriteCmd(0xa0,IC2); //tryb norma (ADC)
  LcdWriteCmd(0xa4,IC1); //dynamic drive
  LcdWriteCmd(0xa4,IC2); //dynamic drive
  LcdWriteCmd(0xa9,IC1); //1/32 duty
  LcdWriteCmd(0xa9,IC2); //1/32 duty
  LcdCls();
  LcdWriteCmd(0xaf,IC1); //wlacz wyswietlacz
  LcdWriteCmd(0xaf,IC2); //wlacz wyświetlacz

//zerowanie pamięci RAM
void LcdCls(void){
  char i,j;
  for(j=0;j<4;j++){
  for(i=0;i<122;i++)
  LcdWriteData(0,i,j);
  }
}
```

List. 6. Zapisanie pamięci wyświetlacza

```
void LcdWriteData(unsigned char data, unsigned char column, unsigned char
page){
  page&=3;
  if(column>60)
  LcdWriteCmd(page|0xb8,IC2);  //adres strony
  else
  LcdWriteCmd(page|0xb8,IC1);  //adres strony
  LcdColumnSet(column);        //numer kolumny

  if(column>60)
  {while(CheckBusy(IC2));
   IOSET0|=A0;delay();         //zapisanie danej
   LcdBusWrite(data,IC2);      //zapisanie danej do ukladu IC1
  }
  else
  {while(CheckBusy(IC1));
   IOSET0|=A0;delay();         //zapisanie danej
   LcdBusWrite(data,IC1);      //zapisanie danej do układu IC2
  }
}
```

List. 7. Zapisanie licznika kolumn sterownika

```
void LcdColumnSet(unsigned char column){
  if(column>60){
   while(CheckBusy(IC2));
   IOCLR0|=A0;delay();          //zapisywanie polecenia
   LcdBusWrite(column-61,IC2);} //korekcja
  else{
  while(CheckBusy(IC1));
  IOCLR0|=A0;delay();           //zapisywanie polecenia
  LcdBusWrite(column,IC1);}
}
```

List. 8. Wyświetlanie jednego znaku i fragment tablicy `const char rom_gen[]`

```
char WriteChar(char code,unsigned char col,unsigned char page)
{
  int code_point;
  if((col+6)>121)
  return(0); //ten znak sie nie zmiesci
  code_point=((int)code*6);
  LcdWriteData(rom_gen[code_point++],col++,page);
  LcdWriteData(rom_gen[code_point++],col++,page);
  LcdWriteData(rom_gen[code_point++],col++,page);
  LcdWriteData(rom_gen[code_point++],col++,page);
  LcdWriteData(rom_gen[code_point++],col++,page);
  LcdWriteData(rom_gen[code_point],col,page);
  return(1);
}

0x3e,0x51,0x49,0x45,0x3e,0, //kod znaku 0
0,0x42,0x7f,0x40,0,0,       //kod znaku 1
0x42,0x61,0x51,0x49,0x46,0, //kod znaku 2
0x21,0x41,0x45,0x4b,0x31,0, //kod znaku 3
0x18,0x14,0x12,0x7f,0x10,0, //kod znaku 4
0x27,0x45,0x45,0x45,0x39,0, //kod znaku 5
0x3c,0x4a,0x49,0x49,0x30,0, //kod znaku 6
```

List. 9. Wyświetlanie ciągu znaków (tekstu)

```
void LcdTxt(const char *napis,char x,char y)
{
  while(*napis!=0)
  {if(WriteChar(*napis,x,y))
    x+=6;          //poprzedni znak zostal caly wyswietlony
   else
   {y=((y+1)&3);  //nowa strona (linijka)
    x=0;          //od pierwszej kolumny
    --napis;}     //korekcja - nie wyswietlony znak od nowa
    ++napis;}
}
```

List. 10. Wyświetlenie bitmapy o rozmiarze 122x32 piksele

```
void LcdBmp(void){
char i,j;
unsigned short k=0;
  for(j=0;j<4;j++){
   for(i=0;i<122;i++)
   LcdWriteData(bmp[k++],i,j);
  }
  LcdOn();       //wlaczenie wyswietlacza
}
```

kiem sporo jak na takie wymiary. Na **list. 8** pokazano procedurę wyświetlenia jednego znaku o kodzie *ascii* umieszczonym w argumencie `code` i na pozycji określonej przez numer kolumny początkowej (`kolumn`) i przez numer linijki (`page`). Procedura zapisywania RAM–u nie jest wykonywana, jeżeli cały znak się nie zmieści w prawej części ekranu.

Na **list. 9** pokazano procedurę *LcdTxt*, która wyświetla ciąg znaków (tekst) od pozycji określonej w argumentach `x` (numer kolumny) i `y` (numer wiersza). Kiedy kolejny znak nie mieści się w linijce (banku pamięci), to jest przenoszony do następnej linijki.

Bitmapy przygotowywane w programach graficznych muszą być poddane konwersji. Najlepiej, gdy wynikiem konwersji jest plik z tablicą w języku C. Do konwertowania monochromatycznych bitmap dla wyświetlacza ze sterownikami NJU6450 idealnie nadaje się program Asystent LCD przygotowany przez Radosława Kwietnia i umieszczony na jego stronie *http://radzio.dxp.pl/asystentlcd/*. Po uruchomieniu programu trzeba wybrać opcję „Konwersja bitmapy 122x32 SED1520". Wynikiem konwersji jest tablica w języku C przeznaczona dla kompilatora AVR GCC. Dla naszych potrzeb trzeba zmodyfikować deklarację tablicy do postaci `const unsigned char bmp[]`. Tak przygotowaną bitmapę o wymiarach 122x32 piksele wyświetla procedura pokazana na **list. 10**.

Wyświetlacz HY–12232 może być znakomitą alternatywą w aplikacjach używających do tej pory wyświetlaczy alfanumerycznych ze sterownikiem HD44780. Sterowanie wyświetlaczem nie jest skomplikowane. Zamianę ułatwia zasilanie napięciem +5 V i podobna regulacja kontrastu oraz podobne wymiary. Problem większej liczby linii portów można doraźnie rozwiązać stosując ekspandery (np. popularny PCF8574 sterowany magistralą I²C) lub rejestry przesuwne z wyjściem równoległym. W zamian otrzymujemy wyświetlacz o zdecydowanie większych możliwościach wyświetlania informacji

Tomasz Jabłoński, EP
tomasz.jablonski@ep.com.pl