Sterowanie wskaźnikami ciekłokrystalicznymi

Sterownik wyświetlacza graficznego

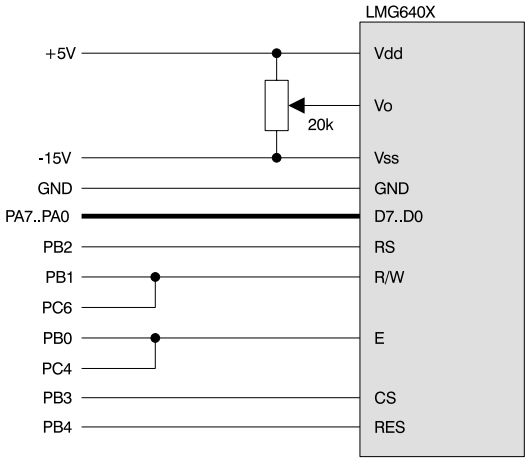
W drugiej części artykułu poświęconego wyświetlaczom LCD omawiamy zagadnienia związane ze sterowaniem wyświetlaczy graficznych. Ze względu na znaczną objętość artykułu, jego dokończenie przedstawimy za miesiąc.

W drugiej części artykułu o wskaźnikach ciekłokrystalicznych zajmiemy się obsługą graficznych. Wyświetlacze graficzne składają się z dwuwymiarowej matrycy punktów, które indywidualnie sterowane umożliwiają wyświetlanie nie tylko kilku lub kilkunastu wierszy tekstu, ale również obrazów graficznych. Nowoczesne moduły graficzne dorównują rozdzielczością kartom graficznym typu SVGA i 15" monitorom kolorowym. Do zastosowań kontrolno-pomiarowych w zupełności wystarczą małe jednokolorowe wyświetlacze o rozdzielczości rzędu 100x100 punktów.

Istotną, praktyczną wadą dostępnych modułów wyświetlaczy jest konieczność stosowania skomplikowanych układów sterujących. Obraz na wyświetlaczu ciekłokrystalicznym, przechowywany w pamięci obrazu RAM, podobnie jak w zwykłym monitorze komputerowym generowany jest liniami. W zależności od konstrukcji wyświetlacza możliwe jest jednoczesne odświeżanie kilku sekcji ekranu. Niestety producenci wyświetlaczy graficznych stosują indywidualne specyfikacje do swoich produktów, co więcej, informacje te przynajmniej do niedawna były skrzętnie ukrywane, co nie ułatwiało indywidualnym konstruktorom życia.

Obecnie dostępne są na rynku dwie kategorie modułów: sterowane szeregowo i z zintegrowaną elektroniką. Moduły sterowane szeregowo są znacznie tańsze, ale - jak wcześniej wspomniano - wymagają zewnętrznych układów sterujących. Do szybkich zastosowań, w ocenie piszącego te słowa, nadają się jedynie wskaźniki zintegrowane z elektroniką sterującą.

W skład układu sterującego wyświe-tlaczem wchodzi kontroler, sterowniki linii i kolumn samej matrycy graficznej oraz pamięć RAM ekranu. Jako kontroler zintegrowanych sterowników najczęściej stosuje się układ HD61830B (lub zgodny) firmy Hitachi. Dostępne są również inne interesujące układy jak na przykład HD63645, który całkowicie emuluje MC6845, czyli klasyczny sterownik terminali alfanumerycznych i graficznych.



Rys. 1.

W dalszej części opracowania przedstawiony zostanie sposób obsługi sterownika wyświetlacza LMG6400PLGR firmy Hitachi.

Wyświetlacze z serii LMG640X posiadają zintegrowany sterownik HD61830B i umożliwiają wyświetlanie w trybie tekstowym do 40 znaków w 16 wierszach lub jednokolorowej grafiki o rozdzielczości 240 na 128 punktów. Wewnętrzny generator znaków umożliwia wyświetlanie do 160 glifów na matrycy 5x7 lub 32 na matrycy 5x11. Generator znaków jest zgodny z ASCII (ISO646). Znaki o kodach powyżej 128, podobnie jak w przypadku kontrolera HD44780, zawierają wybór glifów Kata-kana i greckich. Sterownik wyświetlacza LMG640X posiada również zintegrowaną pamięć RAM o pojemności 8kB. Cały moduł ma wymiary 159x101x9,5mm z ekranem 126x71mm i kwadratowym punktem o rozmiarze 0,47mm.

Wyświetlacz oprócz sygnałów sterujących wymaga dwu napięć zasilających oraz napięcia sterującego kontrastem. Do zasilania części cyfrowej wyświetlacza stosuje się napięcie +5V, natomiast do zasilania matrycy LCD wymagane jest stałe napięcie -15V. Napięcie sterujące kontrastem uzyskuje się z suwaka potencjometru o wartości 10..20kΩ. Sposób podłączenia napięć zasilających wyświetlacz i potencjometru kontrastu przedstawiony jest na **rys. 1**.

W przypadku sterowania wskaźnikiem z płyty głównej komputera PC uzyskanie napięcia -15V nie jest wprost możliwe. Przeprowadzone próby pokazały jednak, że wyświetlacz działa poprawnie, gdy polaryzuje się go dostępnym z zasilacza napięciem -12V. Sygnały sterujące sterownikiem wyś-wietlacza LMG640X wyprowadzone są na 20-stykowe złącze, do którego można podlutować pojedynczą szynę z kołkami Berga i złączem do kabla wstęgowego.

Interfejs układu HD61830B

Podobnie jak w przypadku wcześniej opisywanego układu HD44780, interfejs układu HD61830B przeznaczony jest do bezpośredniej współpracy z szyną procesorów 6502/6800. Podobne są więc sekwencje sygnałów sterujących. Procedura zapisu do układu HD61830 rozpoczyna się od ustawienia linii RS (RS=1, dane lub RS=0, rozkaz) i linii R/W (R/W=0, zapis lub RW=1, odczyt).

Każdy cykl rozpoczyna się rosnącym zboczem sygnału zezwalającego E. Dane do rejestru wewnętrznego wpisywane są zboczem opadającym E. Podobnie cykl odczytu inicjowany jest ustawieniem linii RS i R/W, a sam odczyt dokonywany jest w czasie aktywności linii E. Wyświetlacz LMG640X, oprócz przedstawionych sygnałów RS, RW i E, dodatkowo posiada wejście wybierające CS (CS=0: układ aktywny, CS=1: układ nieaktywny) oraz sygnał zerowania RES (RES=0 - zerowanie kontrolera, RES=1 - wkład pracuje). Podobnie jak w przypadku kontrolera HD44780 obowiązują ograniczenia dynamiczne cyklów dostępu. Sygnały CS, RS i RW muszą być stabilne na co najmniej 140ns przed pojawieniem się rosnącego zbocza E. Czas trwania aktywności sygnału E nie może być krótszy niż 450ns, a cały cykl zapisu lub odczytu musi trwać co najmniej 1µs.

Przebiegi czasowe cyklów zapisu i odczytu przedstawione na **rys. 2** i **3** realizowane

Listing 1. Kod źródłowy biblioteki HD61830.h

```
/*******************************************************
*  HD61830.h
*  biblioteka  obsługi  HD61830  za  pośrednictwem  PPI  8255
*
*  wykorzystywane  linie  8255:
*  PA7  PA6  PA5  PA4  PA3  PA2  PA1  PA0  tryb  2  dwukierunkowy
*  D7    D6  D5    D4  D3    D2    D1  D0
*
*  PB7  PB6  PB5  PB4  PB3  PB2  PB1  PB0  tryb  0  wyjście
*    x    x    x  res  cs    rs    rw  e
*
*  PC7  PC6  PC5  PC4  PC3  PC2  PC1  PC0  (*)  zarezerwowane
*  *    *    *    *    *    x    x    x
*
*  (c)  Janusz  J.  Młodzianowski
*******************************************************/
#include  <dos.h>
#include  <stdlib.h>
#include  <stdio.h>

/*  konfiguracja  8255  */
#define  BASE              0x300
#define  PORTA             BASE
#define  PORTB             BASE+1
#define  PORTC             BASE+2
#define  PORTR             BASE+3
#define  MODE8255          0xC9

/*  sterowanie  interfejsem  HD61830*/
#define  LCGRESET          0x0c        /*  res=0  */
#define  LCGDATA           0x10        /*  rs=0  */
#define  LCGCMD            0x14        /*  rs=1  */
#define  LCGRD             0x12        /*  rw=1  */
#define  LCGWR             0x10        /*  rw=0  */
#define  LCGEN             0x11        /*  e=1  */
#define  LCGDIS            0x10        /*  e=0  */

#define  EDELY                      1
/*  min  540ns  */

/*  rozkazy  HD61830  */
#define  SETLCGMODE        0x00
#define  SETCHPITCH        0x01
#define  SETCHARNUM        0x02
#define  SETTIMEDIVS       0x03
#define  SETCURSPOS        0x04
#define  SETHOMLADDR       0x08
#define  SETHOMHADDR       0x09
#define  SETCURLADDR       0x0a
#define  SETCURHADDR       0x0b
#define  LCGWRDATA         0x0c
#define  LCGRDATA          0x0d
#define  LCGCLEARBIT       0x0e
#define  LCGSETBIT         0x0f

/*  parametry  trybu  tekstowego  */
#define  CURSORON          0x14
#define  BLINK             0x18
#define  DISPON            0x30
#define  MODEGRAPH         0x12

/*  prototypy  funkcji  */
void  lcgwait(void);
void  lcgout(int  where,  int  value);
char  lcgin(void);
void  lcgtext(void);
void  lcdgraph(void);
void  lcgtxtcls(void);
void  lcggraphcls(void);
void  lcggotoxy(int  x,int  y);
void  lcgputc(char  c);
void  lcgputs(char  *s);
void  lcgline(int  x1,int  y1,int  x2,int  y2,int  c);
void  lcgrectangle(int  x1,int  y1,int  x2,int  y2,int  c);
void  lcgellipse(int  xc,int  yc,int  a0,int  b0,int  c);
void  lcgcircle(int  x,int  y,int  r,int  c);

/*******************************************************
*  oczekiwanie  na  zwolnienie  kontrolera
*******************************************************/
void  lcgwait(void)
{
    char  busy;

    outportb(PORTB,LCGCMD|LCGRD|LCGDIS);
    do{
        outportb(PORTB,LCGCMD|LCGRD|LCGEN);
        delay(EDELY);
        busy=inportb(PORTA)&0x80;
        outportb(PORTB,LCGCMD|LCGRD|LCGDIS);
        delay(EDELY);
    }while(busy);
}

/*******************************************************
*  cykl  zapisu
*******************************************************/
void  lcgout(int  where,  int  value)
{
    lcgwait();
    outportb(PORTB,LCGCMD|LCGWR|LCGDIS);
    outportb(PORTB,LCGCMD|LCGWR|LCGEN);
    outportb(PORTA,where);
    delay(EDELY);
    outportb(PORTB,LCGCMD|LCGWR|LCGDIS);
    lcgwait();
    outportb(PORTB,LCGDATA|LCGWR|LCGDIS);
    outportb(PORTB,LCGDATA|LCGWR|LCGEN);
    outportb(PORTA,value);
    delay(EDELY);
    outportb(PORTB,LCGDATA|LCGWR|LCGDIS);
}

/*******************************************************
*  cykl  odczytu  pamięci  VRAM
*******************************************************/
char  lcgin(void)
{
    char  tmp;

    lcgwait();
    outportb(PORTB,LCGCMD|LCGWR|LCGDIS);
    outportb(PORTB,LCGCMD|LCGWR|LCGEN);
    outportb(PORTA,LCGRDATA);
    delay(EDELY);
    outportb(PORTB,LCGCMD|LCGWR|LCGDIS);

    lcgwait();
    outportb(PORTB,LCGDATA|LCGRD|LCGDIS);
    outportb(PORTB,LCGDATA|LCGRD|LCGEN);
    delay(EDELY);
    tmp=inportb(PORTA);
    outportb(PORTB,LCGDATA|LCGRD|LCGDIS);
    return(tmp);
}

/*******************************************************
*  inicjacja  trybu  tekstowego
*******************************************************/
void  lcgtext(void)
{
    outportb(PORTR,MODE8255);      /*  ustawienie  8255*/
    outportb(PORTB,LCGRESET);      /*  reset  kontrolera  */
    delay(10);
    outportb(PORTB,LCGCMD|LCGWR|LCGDIS);

    lcgout(SETLCGMODE,0x3c);       /*  tryb  tekstowy*/
    lcgout(SETCHPITCH,0x75);       /*  znak  5x7*/
    lcgout(SETCHARNUM,39);         /*  40  znaków/wiersz*/
    lcgout(SETTIMEDIVS,127);       /*  cykl  wypełnienia*/
    lcgout(SETCURSPOS,7);          /*  położenie  kursora*/
    lcgout(SETHOMLADDR,0);         /*  adres  HOME  */
    lcgout(SETHOMHADDR,0);
    lcgout(SETCURLADDR,0);         /*  kursor  w  HOME*/
    lcgout(SETCURHADDR,0);
    lcgtxtcls();
}

/*******************************************************
*  inicjacja  trybu  graficznego
*******************************************************/
void  lcdgraph(void)
{
  int  i;

    outportb(PORTR,MODE8255);      /*  ustawienie  8255*/
    outportb(PORTB,LCGRESET);      /*  reset  kontrolera*/
    delay(10);
    outportb(PORTB,LCGCMD|LCGWR);

    lcgout(SETLCGMODE,0x32);       /*  tryb  graficzny*/
    lcgout(SETCHPITCH,0x77);       /*  8  punktów  z  bajtu*/
    lcgout(SETCHARNUM,29);         /*  30  bajtów/linia*/
    lcgout(SETTIMEDIVS,127);       /*  cykl  wypełnienia*/
    lcgout(SETHOMLADDR,0);         /*  adres  HOME*/
    lcgout(SETHOMHADDR,0);
    lcgout(SETCURLADDR,0);         /*  kursor  w  HOME*/
    lcgout(SETCURHADDR,0);
    lcggraphcls();
}

/*******************************************************
*  kasowanie  ekranu  tekstowego
*******************************************************/
void  lcgtxtcls(void)
{
    int  i;

    lcgout(SETCURLADDR,0);         /*  HOME*/
    lcgout(SETCURHADDR,0);
    for(i=0;i<16*40;i++)  lcgout(LCGWRDATA,0);
    lcgout(SETCURLADDR,0);         /*  HOME*/
    lcgout(SETCURHADDR,0);
}

/*******************************************************
*  kasowanie  ekranu  graficznego
*******************************************************/
void  lcggraphcls(void)
{
    int  i;

    lcgout(SETCURLADDR,0);         /*  HOME*/
    lcgout(SETCURHADDR,0);
    for(i=0;i<240*128/8;i++)  lcgout(LCGWRDATA,0);
    lcgout(SETCURLADDR,0);         /*  HOME*/
    lcgout(SETCURHADDR,0);
}

/*******************************************************
*  ustawienie  kursora  tekstowego  w    x,y
```

```
*********************************************************/
void  lcggotoxy(int  x,int  y)
{
    int  n=40*y+x;

    lcgout(SETCURLADDR,n&0xff);
    lcgout(SETCURHADDR,n>>8);
}

/*********************************************************
*  zapisanie  znaku  na  ekran  tekstowy
*********************************************************/
void  lcgputc(char  c)
{
    lcgout(LCGWRDATA,c);
}

/*********************************************************
*  zapis  ciągu  znaków  na  ekran  tekstowy
*********************************************************/
void  lcgputs(char  *s)
{
    while(*s)  lcgout(LCGWRDATA,*s++);
}

/*********************************************************
*  ustawianie/kasowanie  punktu  na  ekranie  graficznym
*********************************************************/
void  lcgpixel(int  x,int  y,int  c)
{
    int  loc;
    char  pixel;

    loc=(y*240+x)/8;
/*  adres  w  VRAM*/
    pixel=(y*240+x)%8;            /*  numer  bitu*/

    lcgout(SETCURLADDR,loc&0xff); /*  ustawienie  kursora*/
    lcgout(SETCURHADDR,loc>>8);
    if(c==0)
        lcgout(LCGCLEARBIT,pixel);
    else
        lcgout(LCGSETBIT,pixel);
}

/*********************************************************
*  rysowanie  linii,  algorytm  Bresenham'a
*********************************************************/
void  lcgline(int  x1,int  y1,int  x2,int  y2,int  c)
{
    int  d,dx,dy;
    int  ainc,binc,yinc;
    int  x,y;


    if(x1>x2){
        swap(&x1,&x2);
        swap(&y1,&y2);
    }

    if(x2==x1){
        if(y1>y2)  swap(&y1,&y2);
        for(d=y1;d<y2;d++)  lcgpixel(x1,d,1);
        return;
    }

    if(y2>y1)  yinc=1;  else  yinc=-1;

    dx=x2-x1;
    dy=abs(y2-y1);
    d=2*dy-dx;
    ainc=2*(dy-dx);
    binc=2*dy;
    x=x1;
    y=y1;
    lcgpixel(x,y,c);
    x=x1+1;

    do{
        if(d>=0){
        y+=yinc;
        d+=ainc;
    }else
        d+=binc;
    lcgpixel(x,y,c);
    }while(++x<x2);
}

/*********************************************************
*  rysowanie  prostokąta
*********************************************************/
void  lcgrectangle(int  x1,int  y1,int  x2,int  y2,int  c)
{
    lcgline(x1,y1,x2,y1,c);
    lcgline(x1,y2,x2,y2,c);
    lcgline(x1,y1,x1,y2,c);
    lcgline(x2,y1,x2,y2,c);
}

/*********************************************************
*  funkcja  pomocnicza  grafiki
*********************************************************/
void  swap(int  *a,int  *b)
{
    int  c;

    c=*a;
    *a=*b;
    *b=c;
}

/*********************************************************
*  funkcja  pomocnicza  grafiki
*********************************************************/
void  lcg4pixels(int  x,int  y,int  xc,int  yc,int  c)
{
    lcgpixel(xc+x,yc+y,c);
    lcgpixel(xc-x,yc+y,c);
    lcgpixel(xc+x,yc-y,c);
    lcgpixel(xc-x,yc-y,c);
}

/*********************************************************
*  rysowanie  elipsy,  algorytm  Bresenham'a
*********************************************************/
void  lcgellipse(int  xc,int  yc,int  a0,int  b0,int  c)
{
    int  x=0;
    int  y=b0;
    float  a=a0;
    float  b=b0;
    float  asqr=a0*a0;
    float  twoasqr=2*a0*a0;
    float  bsqr=b0*b0;
    float  twobsqr=2*b0*b0;
    float  d=bsqr-asqr*b+asqr/4;
    float  dx=0;
    float  dy=twoasqr*b;

    while(dx<dy){
        lcg4pixels(x,y,xc,yc,c);
        if(d>0){
            y--;
            dy-=twoasqr;
            d-=dy;
        }
        x++;
        dx+=twobsqr;
        d=d+bsqr+dx;
    }
    d=d+(3*(asqr-bsqr)/2-(dx+dy))/2;
    while(y>=0){
        lcg4pixels(x,y,xc,yc,c);
        if(d<0){
            x++;
            dx+=twobsqr;
            d+=dx;
        }
      y--;
      dy-=twoasqr;
      d=d+asqr-dy;
    }
}

/*********************************************************
*  rysowanie  okręgu
*********************************************************/
void  lcgcircle(int  x,int  y,int  r,int  c)
{
    lcgellipse(x,y,r,r,c);
}
```

Listing 2. Przykład programu obsługi wskaźnika LMG640X.

```
#include  "HD61830,h"

void  main(void)
{
    lcdgraph();

    lcgrectangle(0,0,239,127,1);
    lcgline(1,1,239,127,1);
    lcgline(239,0,0,127,1);
    lcgline(0,63,239,63,1);
    lcgline(120,0,120,127,1);
    lcgrectangle(80,23,160,103,1);
    lcgcircle(120,63,40,1);
    lcgellipse(120,63,80,47,1);
}
```

są za pomocą funkcji *lcgwr()* i *lcgrd()* zebranych w bibliotece *HD61830.h* i przedstawionych na **list. 1**.

Funkcja *lcgwait()*, wykonywana na początku cyklów dostępu, sprawdza stan gotowości kontrolera. W standardowych konfiguracjach szyn ISA komputerów PC, cykle zapisu i odczytu (instrukcje *outportb* i *inportb*) trwają 1µs. Możliwa jest więc pewna optymalizacja funkcji *lcgwr()* i *lcgrd()*. Funkcje *lcgwait()* i *delay()* mogą być pominięte. Powyższa optymalizacja wymaga jednak dokonania prób na konkretnym sprzęcie.

Na **list. 2** znajduje się przykładowy program obsługi wskaźnika LMG640X. Dokładny opis sposobu programowania układu HD61830B przedstawimy w następnej części artykułu.

Sterowanie układem HD61830B poprzez kartę prototypową i układ 8255

Interfejs sprzętowy kontrolera z komputerem PC w opisywanym rozwiązaniu zrealizowany został za pomocą standardowej karty prototypowej z układem PPI typu 8255. Łącze z komputerem może być również zrealizowane, podobnie jak w przypadku kontrolera HD44780, za pomocą portu drukarkowego.

Koniecznym jednak będzie modyfikacja funkcji komunikacyjnych. Podłączenie układu 8255 do komputera i jego programowanie było już wielokrotnie opisywane w poprzednich numerach EP (np. 3/94).

Ograniczymy więc rozważania do najistotniejszych szczegółów. Układ peryferyjny PPI 8255 posiada trzy 8-bitowe porty danych (PORTA, PORTB i PORTC) oraz rejestr konfiguracyjny (PORTR). Na standardowej karcie prototypowej PPI jest to równoważne czterem kolejnym lokacjom w przestrzeni adresowej układów wejścia/wyjścia. Adres bazowy (PORTA) układu 8255 zwykle ustawiony jest na wartość 0x300. Aby układ mógł być poprawnie użytkowany należy go skonfigurować poprzez wpisanie do rejestru konfiguracyjnego (PORTR, adres bazowy+3) odpowiedniego słowa kontrolnego. Możliwe są trzy tryby pracy portów PPI: proste wejście/wyjście (tryb 0), strobowane wejście/wyjście (tryb 1) oraz dwukierunkowa magistrala danych (tryb 2, jedynie dla PORTA).

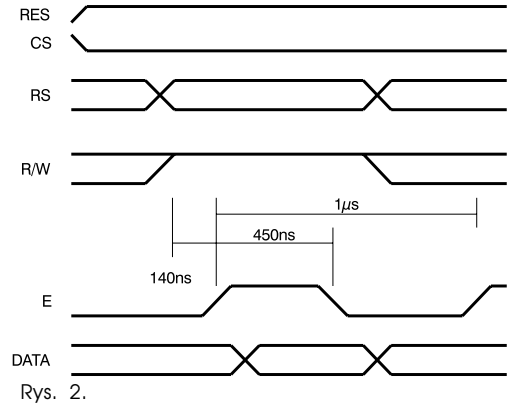
Ponieważ obsługa kontrolera wymaga zarówno zapisu do, jak i odczytu z rejestrów wewnętrznych, to układ 8255 obsługiwany jest w trybie 2. W trybie dwukierunkowej magistrali danych część bitów PORTC służy do obsługi szyny PORTA. W szczególności bit PC6 (ACK) otwiera wyjściowy bufor trójstanowy PORTA (ACK=1 - bufor zamknięty, ACK=0 - bufor otwarty). Bit PC4 (STB) spełnia rolę sygnału zatrzaskującego informację w wejściowym buforze PORTA (aktywne zbocze opadające). Bity PC3, PC5 i PC7 są zarezerwowane i mogą służyć do generacji przerwań (PC3) oraz dodatkowego sygnalizowania stanu transmisji. Bity PC2, PC1 i PC0 mogą być programowane jako proste wejście lub wyście.

W opisywanym interfejsie wyświetlacza PORTA steruje magistralą danych układu HD61380B, PORTB steruje sygnałami RES, CS, RS, RW i E (odpowiednio bity PB4, PB3, PB2, PB1 i PB0). Pozostałe bity nie są wykorzystane.

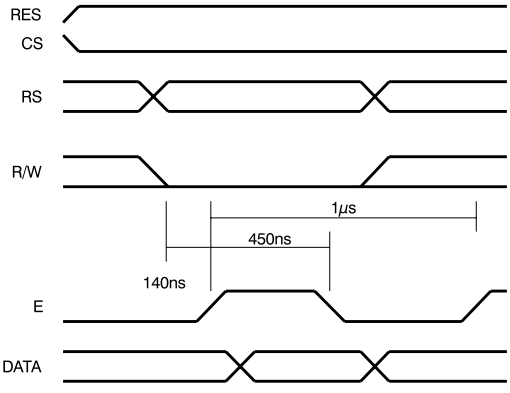
W celu zapewnienia poprawnego sterowania PORTA sygnał RW dołączony jest bezpośrednio do ACK, a sygnał E do STB. Schemat interfejsu wyświetlacza przedstawiony jest na rys. 1.

Janusz J. Młodzianowski

Karta katalogowa (PDF) jest dostępna pod adresem http://www.ep.com.pl/ftp/lcdgraf.pdf



Rys. 2.



Rys. 3.