TFT display s řadičem SSD1963 a ATmega162

Publikované: 15.09.2016, Kategória: Mikroprocesory

www.svetelektro.com

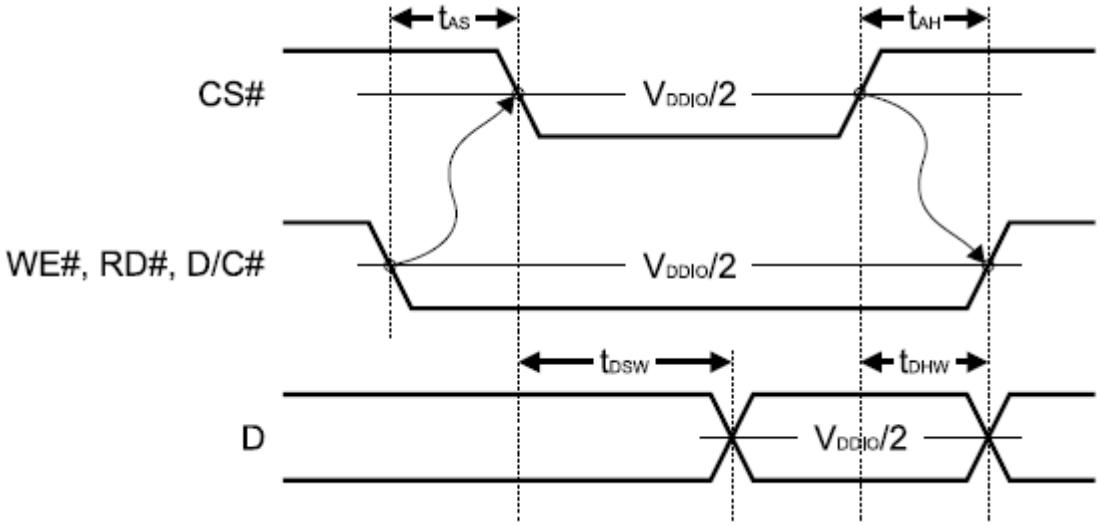
V tomto článku se budu věnovat popisu ovládání TFT displeje, který byl koupen na Ebay.com. TFT display je řízen řadičem SSD1963.

Napájení displeje je 3,3V a se zapnutým podsvícením bude potřeba skoro 300mA – podle toho je potřeba navrhnout zdroj pro první pokusy na nepájivém poli. V první části článku popíšu připojení řadiče k procesoru řady ATmega a inicializaci řadiče. V dalších kapitolách textu budou popsány funkce nutné k vykreslení bodu, čáry, obdélníku a kruhu na TFT display. Řadič displeje neobsahuje žádný alfanumerický font, jako je obvyklé u LCD displejů s řadiči HD44780, nebo T6963. Tam stačilo poslat na displej kód znaku a řadič displeje se postaral o vytvoření znaku ze své paměti. Při práci s řadičem SSD1963 se o tvorbu znaků musí postarat program v procesoru, jednotlivé znaky je potřeba poslat na displej jako pole barevných bodů. Takže v poslední části článku se pokusíme z Windows do paměti procesoru ATmega dostat nějaký font.

Popisovaný TFT display má rozlišení 480 x 272 bodů.

Připojení řadiče SSD1963 k procesoru ATmega644

Komunikace probíhá prostřednictvím 16-ti bitové, nebo 8-mi bitové sběrnice a řídících vodičů. Řadič umožňuje uspořádat časování řídících vodičů tak, jak je zvykem u procesorů řady 6800, nebo u procesorů řady 8080. V displeji který mám k dispozici, je řadič zapojen v režimu 8080, takže se podíváme, jak zařídit komunikaci v tomto režimu:



Obrázek z datového listu obvodu SSD1963

Při zápisu dat do displeje, musí být data připravena na sběrnici, signál /WR musí být aktivní (úroveň L) a data do displeje budou přenesena sestupnou hranou signálu /CS. Nakonec se ukázalo, že pořadí signálu /WR a /CS může být i opačné. Takže pokud na datové sběrnici procesoru není připojen jiný obvod než displej, stačí pro komunikaci signál /WR. Signál /CS, výběr obvodu, může být připojen na úroveň L pořád. Při čtení dat z displeje je nutné, aby datová sběrnice procesoru byla ve stavu vysoké impedance. Data na ní budou k dispozici, když bude signál /RD v úrovni L. Z displeje v tomto článku nebudemečíst nic, takže /RD zůstane pořád neaktivní v úrovni H.

Signál D/C slouží k rozlišení, zda jsou přenášena data, nebo příkazy (command). Pro přenos příkazů stačí 8-mi bitová sběrnice. Pro přenos barev je zapotřebí šest bitů na každou barvu, červenou, zelenou a modrou. To je dohromady 18 bitů. Řadič nabízí několik možností, jak barvy přenášet. Můžeme použít 8-mi bitovou sběrnici a barvu přenést ve třech krocích. Nebo je možné použít devět bitů a potom stačí dva kroky. Pro 16-ti bitový přenos řadič umožňuje ušetřit dva bity z 18-ti. Tento režim je nazván 5-6-5, červená a modrá barva má o jeden bit míň. Protože je nutné data přenášet co nejrychleji – na vykreslení 480 x 272 bodů je potřeba 130560 čísel, použijeme tento režim.

Interface	Cycle	D[23]	D[22]	D[21]	D[20]	D[19]	D[18]	D[17]	D[16]	D[15]	D[14]	D[13]	D[12]	D[11]	D[10]	D[9]	D[8]	D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]
24 bits	1st	R7	R6	R5	R4	R3	R2	R1	R0	G7	G6	G5	G4	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0
18 bits	1st							R5	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B5	B4	B3	B2	B1	B0
16 bits (565 format)	1st									R5	R4	R3	R2	R1	G5	G4	G3	G2	G1	G0	B5	B4	B3	B2	B1
16 bits	1st									R5	R4	R3	R2	R1	R0	X	X	G5	G4	G3	G2	G1	G0	X	X
	2nd									B5	B4	B3	B2	B1	B0	X	X	R5	R4	R3	R2	R1	R0	X	X
	3rd									G5	G4	G3	G2	G1	G0	X	X	B5	B4	B3	B2	B1	B0	X	X
9 bits	1st																R5	R4	R3	R2	R1	R0	G5	G4	G3
	2nd																G2	G1	G0	B5	B4	B3	B2	B1	B0
8 bits	1st																	R5	R4	R3	R2	R1	R0	X	X
	2nd																	G5	G4	G3	G2	G1	G0	X	X
	3rd																	B5	B4	B3	B2	B1	B0	X	X

Obrázek z datového listu obvodu SSD1963

Ještě je potřeba najít příkaz, který ovládá režimy – je to 0xF0.

TFT display jsem připojil k procesoru a napsal makra pro jednotlivé piny portů, které budu potřebovat:

```
#define SSD1963_set_RD    PORTD &=~0x08;   // cteni z displeje, bit 3 portu D
#define SSD1963_clr_RD    PORTD |= 0x08;
#define SSD1963_set_WR    PORTD &=~0x04;   // zapis do displeje, bit 2 portu D
#define SSD1963_clr_WR    PORTD |= 0x04;
#define SSD1963_set_RS    PORTD |= 0x02;   // prenos dat, bit 1 portu D
#define SSD1963_clr_RS    PORTD &=~0x02;   // prenos prikazu
#define SSD1963_DataL     PORTA
#define SSD1963_DataH     PORTC
```

Funkce pro přenos příkazu:

```
void TFT_SendCommand(unsigned int val)
{
   SSD1963_clr_RS;               // nastavi pin pro prenos prikazu
   SSD1963_DataH = val >> 8;      // horni polovina 16-ti bitoveho cisla na port C
   SSD1963_DataL = val;           // dolni polovina 16-ti bitoveho cisla na port A
   SSD1963_set_WR;               // impuls na WR
   SSD1963_clr_WR;
}
```

Funkce pro přenos dat:

```
void TFT_SendData(unsigned int val)
{
   SSD1963_set_RS;               // nastavi pin pro prenos dat
   SSD1963_DataH = val >> 8;      // horni polovina 16-ti bitoveho cisla na port C
   SSD1963_DataL = val;           // dolni polovina 16-ti bitoveho cisla na port A
   SSD1963_set_WR;               // impuls na WR
   SSD1963_clr_WR;
}
```

Ještě se bude hodit funkce, která najednou pošle příkaz a jeho parametr:

```
void TFT_WriteDataChar(unsigned int command, unsigned int val)
{
   TFT_SendCommand( command );
   TFT_SendData( val );
}
```

Na začátku práce, po připojení displeje na napájení je zapotřebí nastavit spoustu registrů v řadiči, aby bylo možné řadičem ovládat TFT panel. Následující funkce provede softwarový reset displeje, nastaví PLL generátor, nastaví počet bodů našířku a na výšku displeje. Poslední část funkce nastaví PWM generátor, kterým je možné řídit jas podsvícení displeje. Kvůli tomu je zapotřebí přemístit jeden odpor – propojku na plošném spoji displeje z pozice R4 na pozici R8. Parametry jednotlivých příkazů jsou převzaty z dokumentace, kterou lze stáhnout ze stránek na ebay.com.

```
void TFT_Init(void)
{
   SSD1963_clr_WR;
   SSD1963_clr_RD;
   TFT_SendCommand( 0x0001 );               // software reset
   _delay_ms( 10 );
   TFT_SendCommand( 0x00E2 );               // PLL multiplier, set PLL clock to 120MHz
   TFT_SendData( 0x002D );
   TFT_SendData( 0x0002 );
   TFT_SendData( 0x0004 );
   TFT_WriteDataChar( 0x00E0, 0x0001 );      // PLL enable
   _delay_ms( 2 );
   TFT_WriteDataChar( 0x00E0, 0x0003 );
   _delay_ms( 5 );
   TFT_SendCommand( 0x0001 );               // software reset
   _delay_ms( 5 );

   TFT_SendCommand( 0x00E6 );               //SET PCLK freq=9.5MHz  ; pixel clock frequency
   TFT_SendData( 0x0000 );                 // 0x03
   TFT_SendData( 0x00FF );
   TFT_SendData( 0x00BE );

   TFT_SendCommand( 0x00B0 );               // set LCD specification
   TFT_SendData( 0x0020 );
   TFT_SendData( 0x0000 );                 //LCD panel mode
   TFT_SendData( 0x0001 );                 //SET horizontal size=480-1 HightByte
   TFT_SendData( 0x00DF );                 //SET horizontal size=480-1 LowByte
   TFT_SendData( 0x0001 );                 //SET vertical size=272-1 HightByte
   TFT_SendData( 0x000F );                 //SET vertical size=272-1 LowByte
   TFT_SendData( 0x0000 );                 //SET even/odd line RGB seq.=RGB

   TFT_SendCommand( 0x00B4 );              //SET Horizontal Period (8 parameters)
   TFT_SendData( 0x0002 );                 //531 HT
   TFT_SendData( 0x0013 );
   TFT_SendData( 0x0000 );                 //43 HPS
   TFT_SendData( 0x002B );
   TFT_SendData( 0x000A );                 //10 HPW
   TFT_SendData( 0x0000 );
   TFT_SendData( 0x0008 );                 //8 LPS
   TFT_SendData( 0x0000 );

   TFT_SendCommand( 0x00B6 );               //SET Vertical Period (7 parameters),
   TFT_SendData( 0x0001 );                 //288 VT
   TFT_SendData( 0x0020 );
   TFT_SendData( 0x0000 );                 //12 VPS
   TFT_SendData( 0x000C );
   TFT_SendData( 0x000A );                 //10 VPW
   TFT_SendData( 0x0000 );                 //4 FPS
   TFT_SendData( 0x0004 );

   TFT_WriteDataChar( 0x0036, 0x0000 );      //rotation!!!!!!!!! toci displejem, prostudovat
   TFT_WriteDataChar( 0x00F0, 0x0003 );      // 565 format barvy
   _delay_ms( 5 );
   TFT_SendCommand( 0x0029 );               // Display ON
   TFT_SendCommand( 0x00BE );               // set PWM for B/L
   TFT_SendData( 0x0006 );
   TFT_SendData( 0x00F0 );                 // hodnota PWM
   TFT_SendData( 0x0001 );
   TFT_SendData( 0x00F0 );
   TFT_SendData( 0x0000 );
   TFT_SendData( 0x0000 );
   TFT_WriteDataChar( 0x00D0, 0x000D );

   TFT_SendCommand( 0x00B8 );              // reset GPIO0
   TFT_SendData( 0x0000...
```

TFT display se podařilo připojit a rozsvítit. Paměť tímto způsobem inicializovaného displeje je naplněna náhodnými čísly. Proto budou i barvy na displeji rozmístěny chaoticky – zrnění všech možných barev. Budeme potřebovat funkci, která celý displej zaplní jednou barvou. No a protože jsou barvy vyjádřeny 16-ti bitovými čísly, bude dobré nadefinovat několik základních barev, abychom si nemuseli pamatovat jejich číselné vyjádření:

```
#define     ColBlack          0x0000       // cerna
#define     ColNavy           0x000F       // namorni modra
#define     ColBlue           0x001F       // modra
#define     ColDarkBlue       0x01CF       // tmave modra
#define     ColDarkGreen      0x03E0       // tmave zelena
#define     ColDarkCyan       0x03EF       // tmave azurova
#define     ColGreen          0x07E0       // zelena
#define     ColCyan           0x07FF       // azurova
#define     ColGrayBlue       0x5458       // seda modra
#define     ColMaroon         0x7800       // hnedocervena
#define     ColPurple         0x780F       // nachova
#define     ColOlive          0x7BEF       // olivova
#define     ColLightBlue      0x7D7C       // svetle modra
#define     ColPorpo          0x801F       //
#define     ColDarkGrey       0x8410       // tmave seda
#define     ColGrey           0x8430       // seda
#define     ColLGreyGreen     0xA651       // svetle sedozelena
#define     ColLightGrey      0xC618       // svetle seda
#define     ColRed            0xF800       // cervena
#define     ColMagneta        0xF81F       // purpurova
#define     ColOrange         0xFC08       // oranz
#define     ColYellow         0xFFE0       // zluta
#define     ColWhite          0xFFFF       // bila
```

TFT display a funkce pro kreslení

Na začátku každé operace na displeji vymezíme prostor – obdélník, ve kterém budeme kreslit. K tomu slouží příkaz 0x2A a 0x2B, které jsem využil v následující funkci:

```
void TFT_WindowSet(unsigned int StartX, unsigned int EndX, unsigned int StartY, unsigned int EndY)
{
   TFT_SendCommand( 0x002A );       // SET page address
   TFT_SendData( StartX >> 8 );     // SET start page address
   TFT_SendData( StartX );
   TFT_SendData( EndX >> 8 );       // SET end page address
   TFT_SendData( EndX );
   TFT_SendCommand( 0x002B );       // SET column address
   TFT_SendData( StartY >> 8 );     // SET start column address
   TFT_SendData( StartY );
   TFT_SendData( EndY >> 8 );       // SET end column address
   TFT_SendData( EndY );
}
```

A tady je funkce, která vyhrazenou plochu displeje zaplní zadanou barvou. Do prostoru v paměti displeje vyhrazeného funkcí TFT_WindowSet, nahraje požadovanou barvu. To, že bude zahájeno nahrávání barev do paměti, oznámí řadiči příkaz 0x2C:

```
void TFT_ColorBox(unsigned int StartX, unsigned int EndX, unsigned int StartY, unsigned int EndY, unsigned int color)
{
   unsigned int x, y;
   TFT_WindowSet( StartX, EndX, StartY, EndY );
   TFT_SendCommand( 0x002C );
   for( x = StartX; x <= EndX; x++ )
   {
      for ( y = StartY; y <= EndY; y++ )
      {
         TFT_SendData( color );            // nastavi barvu
      }
   }
}
```

Další funkce je podobná, ale vyhrazená plocha má velikost jednoho bodu. Takže nakreslíme bod:

```
void TFT_DrawPoint(unsigned int x, unsigned int y, unsigned int color)
{
   TFT_WindowSet( x, x, y, y );
   TFT_SendCommand( 0x002C );
   TFT_SendData( color );
}
```

No a na konec budeme kreslit přímku, obdélník a kružnici:

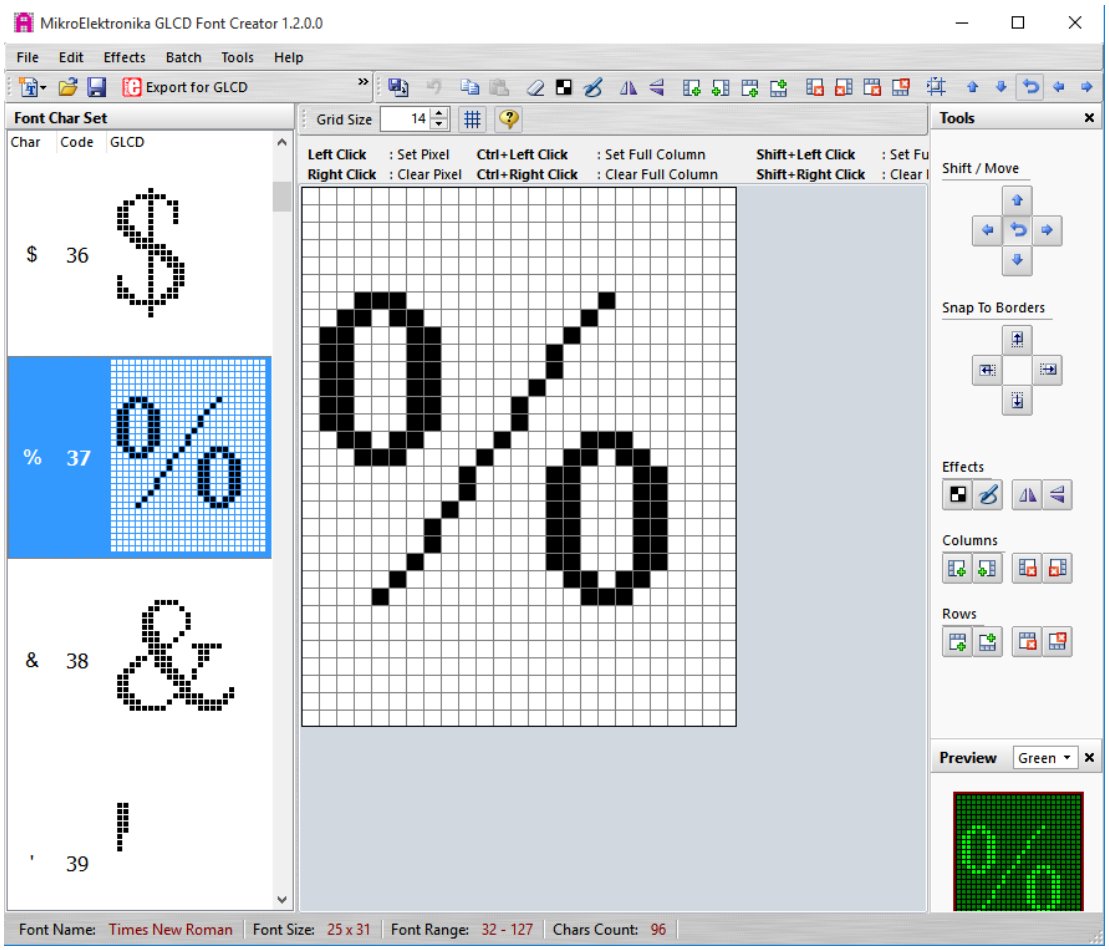
```
void TFT_DrawLine(unsigned int x1, unsigned int y1, unsigned int x2,
unsigned int y2, unsigned int color)
{
   int dx, dy, stepx, stepy, fraction;

   dy = y2 - y1;
   dx = x2 - x1;
   if (dy < 0)
   {
      dy = -dy;
      stepy = -1;
   }
   else stepy = 1;
   if (dx < 0)
   {
      dx = -dx;
      stepx = -1;
   }
   else stepx = 1;
   dx <<= 1;
   dy <<= 1;
   TFT_DrawPoint( x1, y1, color );
   if (dx > dy)
   {
      fraction = dy - (dx >> 1);
      while( x1 != x2 )
      {
         if (fraction >= 0)
         {
            y1       += stepy;
            fraction -= dx;
         }
         x1          += stepx;
         fraction    += dy;
         TFT_DrawPoint( x1, y1, color );
      }
   }
   else
   {
      fraction = dx - (dy >> 1);
      while( y1 != y2 )
      {
         if (fraction >= 0)
         {
            x1           += stepx;
            fraction     -= dy;
         }
         y1           += stepy;
         fraction     += dx;
         TFT_DrawPoint( x1, y1, color );
      }
   }
}
```

```
void TFT_DrawBox(unsigned int x1, unsigned int y1, unsigned int x2,
unsigned int y2, unsigned int color)
{
   TFT_DrawLine( x1, y1, x2, y1, color);    // horni cara
   TFT_DrawLine( x1, y2, x2, y2, color);    // spodni cara
   TFT_DrawLine( x2, y1, x2, y2, color);    // prava cara
   TFT_DrawLine( x1, y1, x1, y2, color);    // leva cara
}
```

```
void TFT_DrawCircle(unsigned int x, unsigned int y, unsigned int radius,
unsigned int color)
{
   int xc = 0;
   int yc, p;
   yc = radius;
   p = 3 - (radius << 1);
   while( xc <= yc)
   {
      TFT_DrawPoint( x + xc, y - yc, color );
      TFT_DrawPoint( x - xc, y - yc, color );
      TFT_DrawPoint( x + yc, y - xc, color );
      TFT_DrawPoint( x - yc, y - xc, color );
      TFT_DrawPoint( x + xc, y + yc, color );
      TFT_DrawPoint( x - xc, y + yc, color );
      TFT_DrawPoint( x + yc, y + xc, color );
      TFT_DrawPoint( x - yc, y + xc, color );
      if (p < 0)
      p += ( xc++ << 2 ) + 6;
      else
      p += (( xc++ - yc-- ) << 2 ) + 10;
   }
}
```

TFT display a fukce pro psaní alfanumerických znaků

Písmenka jsou v paměti procesoru uložena jako sady černých a bílých bodů, kterými je nutné zaplnit obdélník veliký jako písmenko. K vyhrazení obdélníku slouží funkce TFT_WindowSet. Nyní je potřeba přijít na to, kde vzít soubor písmenek, ideálně z fontů, které jsou součástí systému Windows. Našel jsem volně dostupný program MikroElektronika-GLCD. Do něj je možné importovat každý font, který je součástí Windows, v požadované velikosti. Při importu lze zvolit rozsah importovaných znaků (smysl mají znaky 32 – 127), možnosti posouvání znaků k okraji bitmapy dle potřeby. Jednotlivé znaky lze dál upravovat podle vlastní fantazie:



Když jsme s fontem spokojeni, můžeme jej vyexportovat do pole dat, které bude možno použít v AVR studiu:

```
const unsigned char Tahoma11x13[] PROGMEM  =
{
   0x00,
   0x00,
   0x20,0x00,
   0x7F,0x00,
   0x0D,
   0x00,
   0x01,0x88,0x01,0x00,
   0x02,0x95,0x01,0x00,
   0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,          // Code for char num
32
   0x00,0x02,0x02,0x02,0x02,0x02,0x02,0x00,0x02,0x02,0x00,0x00,0x00,              // Code for char num
33
.......
```

Tohle je jenom kousek... datové pole je celkem velké. První řádek plný nul, kód 32, je mezera. Na dalším řádku je zobrazen vykřičník, takže data 0x02 zobrazují šest bodů pod sebou. Potom je mezera a pak dva body pro tečku vykřičníku. Tímto způsobem jsou uložena všechna písmena fontu. Čtyři fonty, které jsou na úvodní fotografii zaberou v paměti procesoru asi 30kb.

Je potřeba si uvědomit, že překladač AVR studia automaticky vytvoří program, který na začátku uloží obsah datového pole do paměti RAM procesoru. No a to je problém, protože paměti RAM v ATmega není moc, musíme šetřit. A tvar znaků během chodu programu není potřeba měnit, takže není žádný důvod, aby pole bylo v RAM. K tomu, aby pole dat zůstalo v paměti programu a odtud bylo používáno, slouží knihovna pgmspace.h. Tu připojíme na začátku programu a potom můžeme využít její funkce. Do názvu pole dat doplníme slovo PROGMEM – data zůstanou v paměti programu a paměť RAM zůstane volná.

Nyní musíme pochopit, jak jsou data uspořádána, abychom je uměli použít. Na začátku datového pole je vyznačen kód prvního a posledního znaku. Začátek je 0x20 hexadecimálně, to je 32 decimálně, podle ASCII tabulky mezera. Konec datového pole je hexadecimálně 0x7F = 127 decimálně, to je kousek za malým Z, ještě je tam ~ a ještě něco. Dál je v datovém poli číslo 0x0D, to bude výška písmenka – počet řádků, pixelů displeje, které jsou zapotřebí pro zobrazení. Pak tu máme vždy čtveřici čísel pro každý znak uložený v datovém poli. První číslo udává počet sloupců, pixelů displeje – znaky mají různou šířku. Další tři čísla označují místo v poli dat, na kterém začíná písmenko. Písmenko je v poli dat zobrazeno po řádcích. Bity jdou zleva do prava, od bitu s nejnižší vahou k nejvyšší. Takže při zobrazování budeme číslem rotovat doprava a číst bit 0. Pro větší znaky bude čísel na jeden řádek potřeba víc, jde jedno za druhým. A když bude jeden řádek zaplněn, nový řádek začíná novým číslem.

V první části funkce pro zobrazení znaku z pole dat přečteme údaje o znaku, který má být zobrazen. Znaky jsou seřazeny tak, jak jdou za sebou v tabulce ASCII, takže od vstupní proměnné ChAscii stačí odečíst 32 a můžeme číst údaje z pole dat:

```
Column = pgm_read_byte(&( Tahoma11x13[( ChAscii * 4 ) + 8  ] ));               // pocet zobrazenych sloupcu
PointerStart =  pgm_read_byte(&( Tahoma11x13[( ChAscii * 4 ) + 10 ] )) * 256;   // misto kde zacina pismenko
PointerStart += pgm_read_byte(&( Tahoma11x13[( ChAscii * 4 ) + 9  ] ));
ChAscii++;
PointerEnd =  pgm_read_byte(&( Tahoma11x13[( ChAscii * 4 ) + 10 ] )) * 256;     // misto kde konci pismenko
PointerEnd += pgm_read_byte(&( Tahoma11x13[( ChAscii * 4 ) + 9  ] ));
```

pgm_read_byte je z knihovny pgmspace.h. V další části funkce je smyčka, která bude z pole dat postupně číst bajty a rotovat jimi. Podle hodnoty nejnižšího bitu bude na displej odeslána barva pozadí, nebo barva znaku – když bit = 1.

```
TFT_WindowSet( x, x + Column + 2, y, y + Row - 1 );             // nastavi pracovni plochu na displeji
TFT_SendCommand( 0x002C );                                      // bude posilat data do pracovni plochy
CounterBits = 8;                                        // pocitadlo vysilanych bitu
CounterColumn = 0;                                          // pocitadlo sloupcu
while( PointerEnd != PointerStart )                             // konec, kdyz jsou vyslany vsechny data znaku
{
   if( CounterBits == 8 )                                   // pri osmem bitu vezme dalsi bajt
   {
      CounterBits = 0;                                  // vynuluje pocitadlo bitu
      Bits = pgm_read_byte(&( Tahoma11x13[ PointerStart ] ));  // veme novy bajt ze spravne znakove sady
      PointerStart++;                                   // pocitadlo bajtu, ktere se maji vyslat
   }
   if( Bits & 0x01 ) TFT_SendData( ColFront );                  // posila barvu bitu
   else TFT_SendData( ColBack );
   Bits = Bits >> 1;                                    // posune se na dalsi bit
   CounterColumn++;                                         // posune se na dalsi sloupec
   CounterBits++;                                       // pocitadlo bitu
   if( CounterColumn == Column )                                // kontrola, jestli je sloupec posledni
   {
      CounterColumn = 0;                                    // vynuluje pocitadlo sloupcu
      TFT_SendData( ColBack );                              // prida slopec za pismeno
      CounterBits = 8;                                  // novych 8 bitu
      TFT_SendData( ColBack );                              // dalsi sloupec za pismenkem
      TFT_SendData( ColBack );
   }
}
Column += 2;
return Column;                                          // vraci pocet spotrebovanych sloupcu
```

Soubory ke stažení

Knihovna pro komunikaci s obvodem SSD1963, včetně funkce pro inicializaci obvodu

TFT_SSD1963.h

TFT_SSD1963.c

Knihovna obsahuje grafické funkce

TFT_GraphicFunctions.h

TFT_GraphicFunctions.c

Knihovna obsahuje alfanumerické funkce pro psaní textu a číslic

TFT_AlphanumericFunctions.c

TFT_AlphanumericFunctions.h

Fonty, které mohou být použity v knihovně alfanumerických funkcí

TFT_Tahoma32x33

TFT_Tahoma30x32

TFT_Tahoma15x16

TFT_Tahoma11x13

TFT_Segoe_Print28x47

TFT_Segmental30x33

TFT_Nixie_One33x32

TFT_Nixie_One38x37

TFT_Nixie_One45x43

TFT_Nixie_One50x47

TFT_Segmental11x13

TFT_Segmental15x17

TFT_Segmental21x25

Autor článku: František OK2JNJ