Obsługa wyświetlaczy graficznych z wbudowanym kontrolerem S6B1713, część 2

Przykłady dla mikrokontrolera AT91SAM7S256

Od wielu lat dużą popularnością cieszą się sterowniki wyświetlaczy graficznych typu KS0108 oraz T6963C. Są one stosowane w najpopularniejszych modułach LCD, zazwyczaj produkowanych w technologii Chip–On–Board, polegającej na umieszczeniu układu kontrolera oraz matrycy LCD na klasycznej płytce drukowanej. Rozwiązanie to pociąga za sobą znaczną masę modułu wyświetlacza, poprzez konieczność stosowania metalowych elementów dociskających wyświetlacz do płytki drukowanej oraz jego duże wymiary, czasem znacznie przekraczające rozmiar aktywnego obszaru ekranu. Na rynku są dostępne także nowocześniejsze kontrolery, które ze względu na zintegrowanie w strukturze wszystkich niezbędnych do pracy obwodów pozwalają na wykonanie wyświetlacza w technologii Chip–On–Glass polegającej na umieszczeniu struktury układu kontrolera bezpośrednio na szklanym ekranie wyświetlacza. Przedstawicielem tej grupy kontrolerów jest opisany w artykule układ S6B1713.

Procedury obsługi wyświetlacza

Procedury sterujące wyświetlaczem zostały napisane w języku C dla kompilatora arm–elf–gcc wchodzącego w skład pakietu WinARM. Wyświetlacz został podłączony do zestawu uruchomieniowego ZL11ARM z minimodułem ZL12ARM (*www.kamami.pl*) w wersji z mikrokontrolerem AT19SAM7S256. Schemat podłączenia wyświetlacza przedstawiono na **rys. 1**. Zasadnicza część funkcji sterujących realizujących funkcje bezpośrednio związane ze sterownikiem S6B1713 zawarta jest w pliku *S6B1713.c*. Z plikiem tym jest związany plik nagłówkowy *S6B1713.h*, w którym zdefiniowano linie konfigurujące I/O mikrokontrolera oraz parametry wyświetlacza. Najważniejsze stałe zdefiniowane w tym pliku określają, które linie I/O mikrokontrolera AT91SAM7S256 będą wykorzystane do sterowania poszczególnymi liniami wyświetlacza. Deklaracje linii sterujących przedstawiono na **list. 1**.

Konfiguracja ta może zostać oczywiście dowolnie dostosowana do potrzeb projektowanego układu.

Odczyt bajtu statusowego

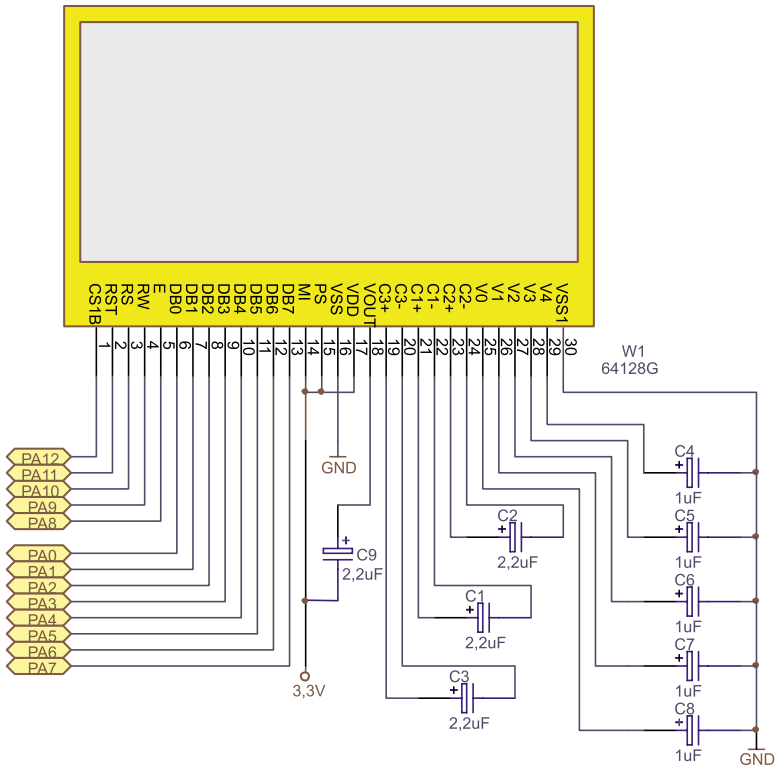
Kontroler wyświetlacza do czasu zakończenia wykonywania poprzedniej instrukcji ignoruje kolejne, dlatego też przed zapisaniem instrukcji należy sprawdzić stan flagi BUSY. W przypadku, gdy znajduje się ona w stanie wysokim należy odczekać na jej wyzerowanie. Kod funkcji odczytu bajtu statusowego przedstawiono na **list. 2**.

Zapis rozkazu

Przed zapisaniem rozkazu sprawdzana jest flaga BUSY, w przypadku gdy jest ona w stanie wysokim program oczekuje na jej wyzerowanie. Kod funkcji zapisującej rozkaz przedstawiono na **list. 3**.

Zapis danej

Również przed zapisaniem danych jest sprawdzana flaga BUSY. W przypadku, gdy jest ona w stanie wysokim program oczekuje na jej wyzerowanie. Każdorazowe zapisanie danej do wyświetlacza powoduje inkrementacją adresu kolumny. Po osiągnięciu wartości maksymalnej dalsza inkrementacja jest



Rys. 1.

wstrzymywana. Kod funkcji zapisu danej przedstawiono na **list. 4**.

Odczyt danej

Przed odczytem danych sprawdzana jest flaga BUSY. W przypadku, gdy jest ona w stanie wysokim program oczekuje na jej wyzerowanie. Po wykonaniu instrukcji adres kolumny jest automatycznie inkrementowany. Po osiągnięciu wartości maksymalnej dalsza inkrementacja jest wstrzymywana. Kod funkcji odczytu danej przedstawiono na **list. 5**.

List. 1.
```
#define S6B_E     AT91C_PIO_PA8          /* pin PA8 -> E */
#define S6B_RW    AT91C_PIO_PA9          /* pin PA9 -> RW */
#define S6B_RS    AT91C_PIO_PA10         /* pin PA10 -> RS */
#define S6B_RST   AT91C_PIO_PA11         /* pin PA11 -> RST */
#define S6B_CS1B  AT91C_PIO_PA12         /* pin PA12 -> CS1B */
#define S6B_D0    0                      /* pierwszym bitem szyny danych jest PA0 */
```

List. 2.
```
unsigned char GLCD_ReadStatus(void)
{
volatile int tmp;
AT91F_PIO_ClearOutput(AT91C_BASE_PIOA, (S6B_CS1B | S6B_RS)); // CS1B = 0; RS = 0
AT91F_PIO_SetOutput(AT91C_BASE_PIOA, S6B_RW | S6B_E); // RW = 1; E = 1
AT91F_PIO_CfgInput(AT91C_BASE_PIOA,(0xFF << S6B_D0));
tmp = ((AT91F_PIO_GetInput(AT91C_BASE_PIOA) >> S6B_D0) & 0xFF);
AT91F_PIO_ClearOutput(AT91C_BASE_PIOA, S6B_E);
AT91F_PIO_SetOutput(AT91C_BASE_PIOA, S6B_CS1B);
return tmp;
}
```

List. 3.
```
void GLCD_WriteCommand(unsigned char commandToWrite)
{
while((GLCD_ReadStatus()&STATUS_BUSY));
AT91F_PIO_ClearOutput(AT91C_BASE_PIOA, S6B_RS | S6B_CS1B | S6B_RW);
AT91F_PIO_CfgOutput(AT91C_BASE_PIOA,(0xFF << S6B_D0));
AT91F_PIO_ForceOutput(AT91C_BASE_PIOA, (unsigned int)(commandToWrite << S6B_D0));
AT91F_PIO_SetOutput(AT91C_BASE_PIOA, S6B_E);
AT91F_PIO_ClearOutput(AT91C_BASE_PIOA, S6B_E);
AT91F_PIO_SetOutput(AT91C_BASE_PIOA, S6B_CS1B);
}
```

List. 4.
```
void GLCD_WriteData(unsigned char dataToWrite)
{
while((GLCD_ReadStatus()&STATUS_BUSY));
AT91F_PIO_ClearOutput(AT91C_BASE_PIOA, S6B_RW | S6B_CS1B);
AT91F_PIO_SetOutput(AT91C_BASE_PIOA, S6B_RS);
AT91F_PIO_CfgOutput(AT91C_BASE_PIOA,(0xFF << S6B_D0));
AT91F_PIO_ForceOutput(AT91C_BASE_PIOA, (unsigned int)(dataToWrite << S6B_D0));
AT91F_PIO_SetOutput(AT91C_BASE_PIOA, S6B_E);
AT91F_PIO_ClearOutput(AT91C_BASE_PIOA, S6B_E);
AT91F_PIO_SetOutput(AT91C_BASE_PIOA, S6B_CS1B);
}
```

Ustawienie współrzędnych ekranowych

Funkcja ustawia współrzędne wyświetlacza, pod którymi wykonana zostanie następna operacja (odczyt, bądź zapis). Współrzędna pionowa może przyjmować wartości z zakresu 0...8 (numer strony). Współrzędna pozioma może przyjmować wartości z zakresu 0...131. Należy jednak pamiętać, że w przypadku wyświetlacza 64128G współrzędne są ograniczone organizacją ekranu wyświetlacza (128x64 piksele). Kod funkcji ustawiającej współrzędne ekranowe przedstawiono na **list. 6**.

Wyświetlenie znaku ASCII

Wyświetlenie znaku polega na zapisie do pamięci obrazu wyświetlacza 5 kolejnych bajtów tworzących dany znak. Ponieważ nie ma sensu marnować miejsca w pamięci programu na znaki niewyświetlane (czyli pierwszych 31 znaków tablicy ASCII) pierwszy znak w tablicy (spacja) ma indeks 0, podczas gdy w tablicy ASCII znak ten posiada kod 32. Od kodu znaku przekazanego do funkcji jako parametr należy więc odjąć liczbę 32 (zdefiniowaną jako stała symboliczna *FONT_OFFSET* w pliku *font5x8.h*). Kod funkcji zapisującej znak ASCII przedstawiono na **list. 7**.

Wyświetlenie łańcucha znaków ASCII

Parametrem funkcji jest wskaźnik do typowego dla języka C ciągu znaków zakończonego zerem. Napis jest wyświetlany począwszy od aktualnej wartości adresów strony i kolumny. Przekroczenie maksymalnej współrzędnej pionowej nie powoduje przeniesienia wyświetlanego tekstu do kolejnej linii. Kod funkcji wyświetlającej tekst przedstawiono na **list. 8**.

Przykładowy tekst wyświetlony na wyświetlaczu przedstawiono na **fot. 2**.

Włączenie piksela

Włączenie piksela polega na ustawieniu odpowiadającego mu bitu w pamięci wyświetlacza. Po-

List. 5.
```
unsigned char GLCD_ReadData(void)
{
unsigned char volatile tmp = 0;
while((GLCD_ReadStatus()&0x80));
AT91F_PIO_ClearOutput(AT91C_BASE_PIOA, S6B_CS1B);
AT91F_PIO_SetOutput(AT91C_BASE_PIOA, S6B_RS | S6B_RW);
AT91F_PIO_SetOutput(AT91C_BASE_PIOA, S6B_E);
AT91F_PIO_ClearOutput(AT91C_BASE_PIOA, S6B_E);
AT91F_PIO_SetOutput(AT91C_BASE_PIOA, S6B_E);
AT91F_PIO_CfgInput(AT91C_BASE_PIOA,(0xFF << S6B_D0));
tmp = ((AT91F_PIO_GetInput(AT91C_BASE_PIOA) >> S6B_D0) & 0xFF);
AT91F_PIO_ClearOutput(AT91C_BASE_PIOA, S6B_E);
AT91F_PIO_SetOutput(AT91C_BASE_PIOA, S6B_CS1B);
return tmp;
}
```

List. 6.
```
void GLCD_GoTo(unsigned char column, unsigned char page)
{
GLCD_WriteCommand(LCD_SET_PAGE | page);
GLCD_WriteCommand(LCD_SET_COL_HI | ((column + COLUMN_OFFSET) >> 4));
GLCD_WriteCommand(LCD_SET_COL_LO | ((column + COLUMN_OFFSET) & 0x0F));
}
```

List. 7.
```
void GLCD_WriteChar(char charCode)
{
unsigned char fontCollumn;
for(fontCollumn = 0; fontCollumn < FONT_WIDTH; fontCollumn++)
 GLCD_WriteData(font5x7[((charCode - FONT_OFFSET) * FONT_WIDTH) + fontCollumn]);
GLCD_WriteData(0);
}
```

List. 8.
```
void GLCD_WriteString(char* string)
{
while(*string)
 {
 GLCD_WriteChar(*string++);
 }
}
```

nieważ możliwy jest odczyt i zapis tylko całego bajtu pamięci obrazu procedura włączenia piksela przebiega następująco:

- ustawiamy współrzędne: poziomą oraz podzieloną przez 8 pionową (ponieważ dostęp do pamięci odbywa się całymi bajtami),
- odczytujemy z wyświetlacza aktualny stan pikseli,
- modyfikujemy odczytany bajt poprzez wykonanie na nim odpowiedniej operacji w zależności od wartości parametru *color*,
- zapisujemy tak zmodyfikowany bajt danych pod odpowiedni adres pamięci wyświetlacza.

Kod funkcji przedstawiono na **list. 9**.

List. 9.
```
void GLCD_SetPixel(int x, int y, int color)
{
unsigned char temp = 0;
GLCD_GoTo(x, (y/8));
temp = GLCD_ReadData();
if(color)
 temp |= (1 << (y % 8));
else
 temp &= ~(1 << (y % 8));
GLCD_GoTo(x, (y/8));
GLCD_WriteData(temp);
}
```

Czyszczenie pamięci obrazu

Wyczyszczenie zawartości pamięci obrazu sprowadza się do zapisu do wszystkich jej komórek bajtów o wartości zerowej, niezależnie od tego, czy wyświetlacz pracuje w trybie pozytywowym czy też negatywowym. Kod funkcji czyszczącej pamięć obrazu przedstawiono na **list. 10**.

Wyświetlenie bitmapy

Funkcja wyświetlająca bitmapę przyjmuje 5 argumentów: wskaźnik do tablicy z bitmapą, współrzędne pod którymi bitmapa ma zostać wyświetlona oraz jej rozmiar. Współrzędna pionowa przyjmuje wartości 0...7 (numer strony). Wysokość bitmapy może przyjmować wartości będące wielokrotnością liczby 8. Natomiast współrzędna pozioma i szerokość bitmapy mogą przyjmować dowolne wartości (mieszczące się oczywiście na ekranie wyświetlacza). Bitmapę można przygotować w dowolnym edytorze graficznym i skonwertować za pomocą programu Asystent LCD (*http://radzio.dxp.pl/asystentlcd/*). Kod funkcji wyświetlającej bitmapę przedstawiono na **list. 11**.

Przykładowa bitmapa przygotowana na komputerze PC i wyświetlona

List. 10.
```
void GLCD_ClearScreen(void)
{
int pageIndex, columnIndex;
for(pageIndex = 0; pageIndex < NUMBER_OF_PAGES; pageIndex++)
 {
 GLCD_GoTo(0, pageIndex);
 for(columnIndex = 0; columnIndex < NUMBER_OF_COLUMNS; columnIndex++)
  GLCD_WriteData(0);
 }
GLCD_GoTo(0,0);
}
```

List. 11.
```
void GLCD_Bitmap(char * bitmap,int left, int top, int width, int height)
{
int pageIndex, columnIndex;
for(pageIndex = 0; pageIndex < height / 8; pageIndex++)
 {
 GLCD_GoTo(left, top + pageIndex);
 for(columnIndex = 0; columnIndex < width; columnIndex++)
  GLCD_WriteData(*(bitmap++));
 }
}
```

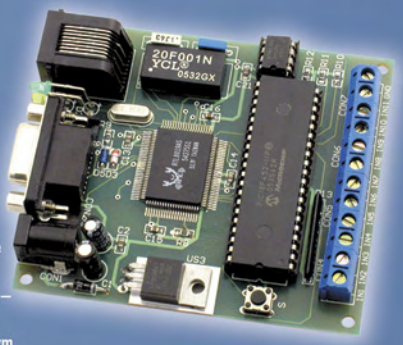
Fot. 2.

SUPER HITY

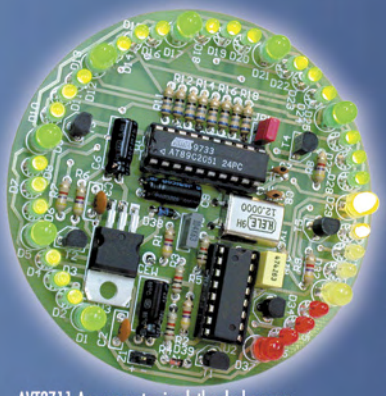
AVT953

Karta wejść z interfejsem Ethernet

Zestaw to karta wejść cyfrowych, umożliwiająca wizualizacje poszczególnych stanów w przeglądarce internetowej. Rozwiązanie to pozwala na niemal nieograniczone kontrolowanie sygnałów cyfrowych – poczynając od wewnętrznej sieci LAN, poprzez sieć przewodową Internet a kończąc na bezprzewodowym dostępie przez telefon komórkowy.
Karta ma 11 wejść i może być zastosowana np. do zdalnej obserwacji wejść centrali alarmowej. Pomimo tego, że generowana strona przystosowana jest do typowej przeglądarki internetowej, może też być wyświetlona na ekranie telefonu komórkowego.



AVT953 A+ - w zestawie płytka drukowana, zaprogramowany układ i dokumentacja. Cena: 69 zł
AVT953 B+ - w zestawie płytka drukowana, komplet elementów i dokumentacja. Cena: 98 zł
AVT953 C - układ zmontowany i uruchomiony. Cena: 220 zł



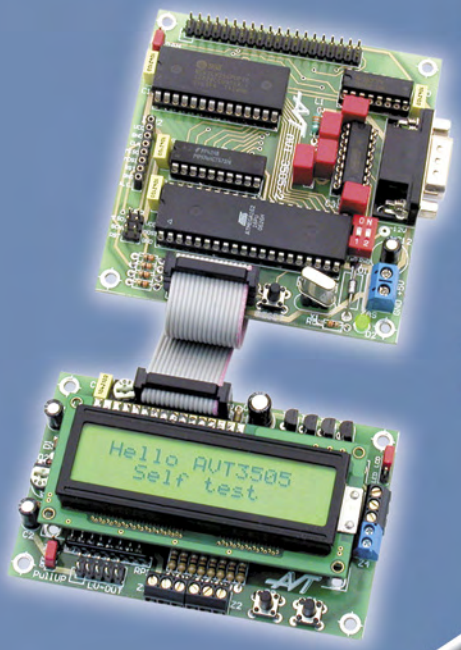
AVT2711

Obrotomierz samochodowy

W nowoczesnych samochodach z doskonale wytłumionymi silnikami zmiana biegów 'na słuch' jest coraz trudniejsza i może prowadzić do przedwczesnego zużycia mechanizmów. Zbyt duże obroty to większy 'apetyt' na paliwo i zużycie silnika. Obrotomierz AVT 2711 jest idealnym rozwiązaniem tych problemów. Jego układ oparty jest na mikroprocesorze co sprawia, że zestaw jest łatwy w montażu i uruchomieniu (nie wymaga kalibracji). Może go wykonać praktycznie każdy.

AVT2711 A+ - w zestawie płytka drukowana zaprogramowany układ i dokumentacja. Cena: 29,6 zł
AVT2711 B+ - w zestawie płytka drukowana, komplet elementów i dokumentacja. Cena: 37 zł

AVT3505



Płytka testowa kursu C

Podstawowym przeznaczeniem kitu jest nauka programowania w C, w oparciu o kurs prowadzony przez miesięcznik EdW. Układ podzielony został na dwie części: płytkę główna i wykonawczą. O ile druga z nich zaprojektowana została głównie z myślą o realizacji projektów przerabianych przez EdW, płytka główna stanowi uniwersalny sterownik o sporych możliwościach. Sercem systemu jest procesor Atmega162. Duże możliwości rozbudowy układu daje złącze EXP. Wyprowadzono na nie całą magistralę zewnętrznej pamięci oraz jedne z portów. Na płytce wykonawczej umieszczono elementy zapewniające komunikację z operatorem – do wyboru wyświetlacz LCD lub LED.

AVT3505 A - w zestawie płytka drukowana i dokumentacja. Cena: 34 zł
AVT3505 B - w zestawie płytka drukowana, komplet elementów i dokumentacja. Cena: 120 zł

AVT2809

Zdalne sterowanie przez telefon

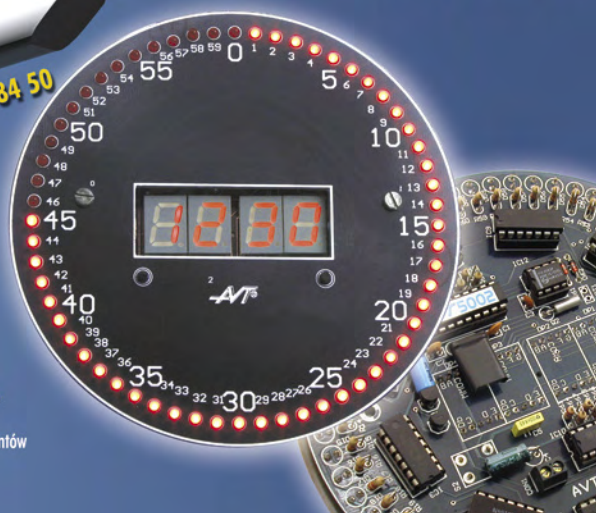
Układ służy do zdalnego sterowania przez telefon. Podłączając urządzenie do linii abonenckiej uzyskujemy możliwość zdalnego łączenia się z nim. Sterownik możesterować dowolnymi układami wykonawczymi (triaki lub przekaźniki). Aby skorzystać z dostępnych funkcji, należy zadzwonić na numer telefonu, do którego dołączony jest moduł. Następnie należy wprowadzić odpowiedni kod sterujący aktywujący bądź dezaktywujący wybrane wyjście.

AVT2809 A - w zestawie płytka drukowana i dokumentacja. Cena: 16 zł
AVT2809 B - w zestawie płytka drukowana, komplet elementów i dokumentacja. Cena: 46 zł
AVT2809 C - układ zmontowany i uruchomiony. Cena: 82 zł

AVT - Korporacja
Sp. z o.o.
03-197 Warszawa
ul. Leszczynowa 11
tel. 022 257 84 50
fax 022 257 84 55
e-mail: handlowy@avt.pl

www.sklep.avt.pl

Zamów pełną ofertę kitów mailem handlowy@avt.pl lub telefonicznie 022 257 84 50



AVT5002

Zegar cyfrowy z sekundnikiem analogowym

Układ łączący w sobie cechy klasycznego zegara wskazówkowego z wyglądem współczesnego zegara cyfrowego. Upływający czas obrazowany jest na wyświetlaczach cyfrowych, a sekundy wyświetlane są analogowo, na kołowym wyświetlaczu zbudowanym 60 diod LED.

AVT5002 A - w zestawie płytka drukowana i dokumentacja. Cena: 64 zł
AVT5002 B - w zestawie płytka drukowana, komplet elementów i dokumentacja. Cena: 100 zł

na wyświetlaczu 64128G przedstawiono na **fot. 3**.

Zmiana współczynnika kontrastu

Funkcja umożliwia zmianę współczynnika kontrastu. Wartość współczynnika musi się zawierać w przedziale 0...63. Kod funkcji przedstawiono na **list. 12**.

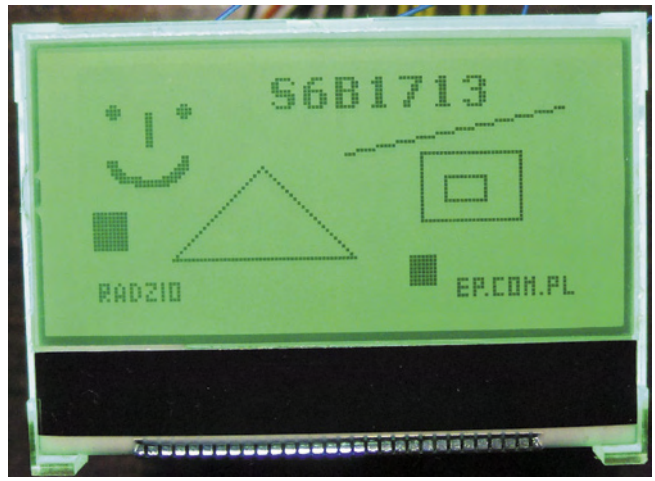
Funkcje graficzne

W pliku *graphic.c* znajdują się podstawowe funkcje graficzne umożliwiające rysowanie podstawowych figur geometrycznych: prostokąta, okręgu oraz linii prostych. Ponieważ wyświetlacz 64128G jest monochromatyczny, na stałe zdefiniowano stałą *color* o wartości 1. Stała ta będzie zawsze przekazywana do funkcji *GLCD_SetPixel*, co wydaje się być zbędne, jednak dzięki temu można przedstawione funkcje w prosty sposób dostosować do współpracy z wyświetlaczami umożliwiającymi wyświetlenie szerszej palety barw.

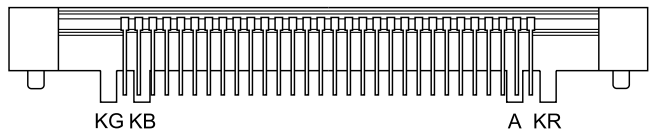
List. 12.
```
void GLCD_SetContrast(unsigned char contrast)
{
GLCD_WriteCommand(LCD_REF_VOLT_MODE);
GLCD_WriteCommand(LCD_REF_VOLT_REG | contrast);
}
```

Rysowanie prostokąta

Funkcja umożliwia narysowanie prostokąta, którego lewy górny róg ma współrzędne (*x*, *y*), a boki długości *a* i *b*. Boki prostokąta muszą być równoległe do odpowiednich osi układu współrzędnych – nie jest możliwe narysowanie prostokąta obróconego. Kod funkcji przedstawiono na **list. 13**.



Fot. 3.



Rys. 4.

List. 13.
```
void GLCD_Rectangle(unsigned char x, unsigned char y, unsigned char b, unsigned char a)
{
 unsigned char j; // zmienna pomocnicza
 // rysowanie linii pionowych (boki)
 for (j = 0; j < a; j++) {
      GLCD_SetPixel(x, y + j, color);
      GLCD_SetPixel(x + b - 1, y + j, color);
   }
 // rysowanie linii poziomych (podstawy)
 for (j = 0; j < b; j++)  {
      GLCD_SetPixel(x + j, y, color);
      GLCD_SetPixel(x + j, y + a - 1, color);
   }
}
```

List. 14.
```
void GLCD_Circle(unsigned char cx, unsigned char cy ,unsigned char radius)
{
int x, y, xchange, ychange, radiusError;
x = radius;
y = 0;
xchange = 1 - 2 * radius;
ychange = 1;
radiusError = 0;
while(x >= y)
 {
 GLCD_SetPixel(cx+x, cy+y, color);
 GLCD_SetPixel(cx-x, cy+y, color);
 GLCD_SetPixel(cx-x, cy-y, color);
 GLCD_SetPixel(cx+x, cy-y, color);
 GLCD_SetPixel(cx+y, cy+x, color);
 GLCD_SetPixel(cx-y, cy+x, color);
 GLCD_SetPixel(cx-y, cy-x, color);
 GLCD_SetPixel(cx+y, cy-x, color);
 y++;
 radiusError += ychange;
 ychange += 2;
 if ( 2*radiusError + xchange > 0 )
  {
  x--;
  radiusError += xchange;
  xchange += 2;
  }
 }
}
```

Rysowanie okręgu

Funkcja umożliwia narysowanie na ekranie wyświetlacza okręgu o środku w punkcie o współrzędnych (cx,cy) oraz o promieniu *radius*. Kod funkcji przedstawiono na **list. 14**.

Podświetlacz RGB

Wyświetlacz 64128H–RGB wyposażono w podświetlenie RGB, umożliwiające uzyskanie praktycznie dowolnego koloru podświetlania. Rozmieszczenie wyprowadzeń podświetlacza przedstawiono na **rys. 4**. Na układ podświetlania składają się po trzy diody LED każdego koloru podstawowego (czerwony, niebieski i zielony) połączone równolegle. Należy zwrócić uwagę na występujący w gałęzi diod czerwonych szeregowy rezystor o wartości kilkunastu omów. Do regulacji jasności świecenia poszczególnych kolorów składowych wykorzystamy występujący w mikrokontrolerze AT91SAM7S256 generator PWM, który pokrótce omówimy w dalszej części artykułu.

Radosław Kwiecień, EP
radoslaw.kwiecien@ep.com.pl