Sterowanie wyświetlaczem graficznym OLED typu DD12832YW-1A

Wyświetlacze wykonywane w technologii OLED wzbudzają duże zainteresowanie, ponieważ w odróżnieniu od matryc LCD, które do działania potrzebują światła odbitego (podświetlania), OLED-y świecą własnym światłem. Dzięki temu jest możliwe uzyskanie bardzo dobrego kontrastu obrazu nawet w złych warunkach oświetleniowych. W artykule opisano sposób sterowania wyświetlaczem DD12832YW-1A firmy Densitron.

Dodatkowe informacje:
Wyświetlacz DD12832YW-1A firmy Densitron został udostępniony do testów przez firmę Farnell. Więcej informacji na stronie *www.pl.farnell.com*

Świecący na żółto, monochromatyczny wyświetlacz OLED typu DD12832YW-1A ma rozdzielczość 128×32 pikseli. Ponieważ nie jest potrzebne podświetlenie, jest on bardzo cienki. Sygnały sterujące i zasilanie są wyprowadzane za pomocą elastycznej tasiemki. Zakończono ją 24-wyprowadzeniowym polem kontaktowym przystosowanym do złącza zaciskowego typu Molex-52893. Opis wyprowadzeń wyświetlacza zamieszczono w **tabeli 1**.

W wyświetlaczu zastosowano specjalizowany sterownik typu SSD1305 produkcji Solomon Systech. Ma on wbudowane drivery przeznaczone do sterowania matrycy o rozdzielczości 132×64 piksele. Schemat blokowy sterownika SSD1305 pokazano na **rysunku 1**.

Na początku najbardziej będzie nas interesował interfejs MCU. Sterownik może łączyć się z mikrokontrolerem przez interfejsy równoległe zgodne ze standardami przemysłowymi Intel 8080 i Motorola 6800 oraz szeregowe I²C i SPI. Do zastosowań praktycznych interesującą alternatywą jest użycie do sterowania interfejsu szeregowego, ponieważ wymaga ona tylko kilku linii sterujących, a przez to zmniejszają się wymagania dotyczące liczby używanych wyprowadzeń mikrokontrolera i upraszcza projekt płytki drukowanej.

Najważniejsze komendy sterownika zamieszczono w **tabeli 2**. Zestawienie wszystkich komend można znaleźć w dokumentacji sterownika.

Struktura pamięci obrazu

Pamięć obrazu (GDDRAM) jest statyczną pamięcią RAM o pojemności 132×64 bity. Każdy bit pamięci odpowiada jednemu pikselowi matrycy. Pamięć o organizacji 132×64 bity jest trudno adresować, dlatego podzielono ją na 8 stron. Ponieważ sterownik może sterować również wyświetlacz kolorowy, w trybie wyświetlania koloru pierwsze 2 strony zawierają atrybuty piksela (**rysunek 2**). Jeśli sterownik współpracuje z wyświetlaczem monochromatycznym, zawartość wszystkich 8 stron jest bezpośrednio odwzorowywana na ekranie wyświetlacza. Po zapisaniu kolejnych bajtów do strony pamięci GDDRAM tworzy się pozioma linijka o szerokości 8 pikseli (**rysunek 3**).

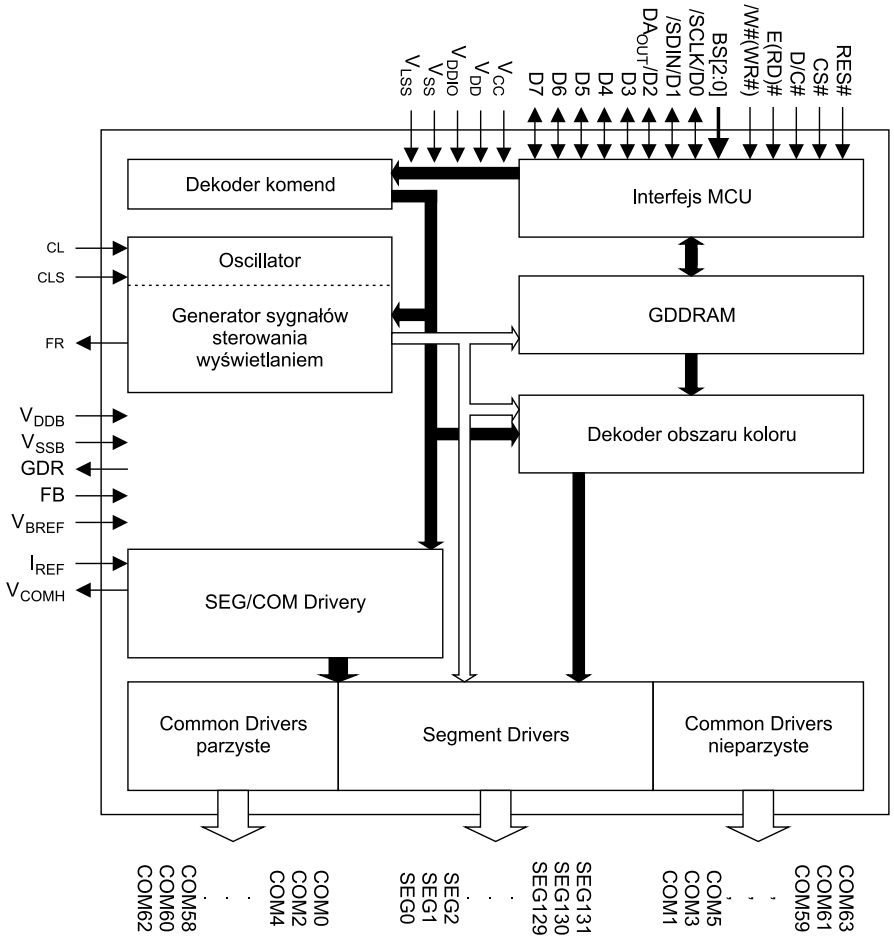
Pamięć obrazu GDDRAM jest adresowana licznikami stron i kolumn. Licznik stron zmienia się od 0 do 7, a licznik kolumn od 0 do 131. Komenda *Set Memory Addresing Mode* umożliwia wybór jednego z trzech trybów adresowania:

Tabela 1. Wyprowadzenia wyświetlacza DD12832YW-1A

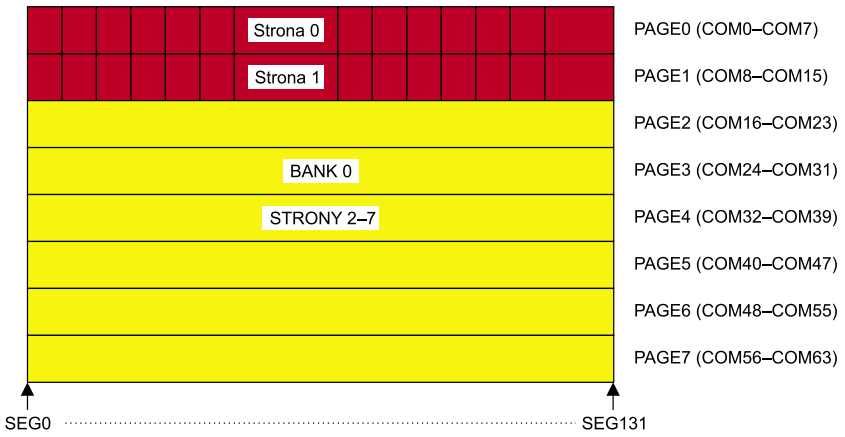
Nr wyprowadzenia	Oznaczenie	Opis
1	NC(GND)	Zarezerwowany (musi być połączone z GND)
2	VLSS	Masa układów analogowych (musi być połączone z GND)
3	VSS	GND
4	NC	–
5	VDD	Zasilanie układów cyfrowych
6	BS1	Wybór rodzaju interfejsu: 00=SPI, 01=I²C, 10=Motorola 6800, 11=Intel 8080
7	BS2	
8	CS#	Chip Select – aktywny poziom niski
9	RES#	Zerowanie – aktywny poziom niski
10	D/C#	Poziom wysoki=dane, poziom niski=komendy
11	R/W#	Int. Motorola 6800=poziom wysoki odczyt, poziom niski=zapis; Intel 8080=sygnał WR (zapis)
12	E/RD#	Int. Motorola 6800 sygnał E (enable); Intel 8080 sygnał RD (odczyt)
13...20	D0...D7	D0...D7 linie danych int. równoległych SPI: D0=CLK, D1=SDIN I²C: D1 i D2 połączone=SDA, D0=SCL
21	IREF	Napięcie odniesienia – regulacja jasności
22	VCOMH	Blokowane (4,7 µF kondensator tantalowy do masy)
23	VCC	Zasilanie driverów matrycy OLED (ok. +12 V)
24	NC	Musi być połączony do GND

– adresowania stron,
– adresowania poziomego,
– adresowania pionowego.

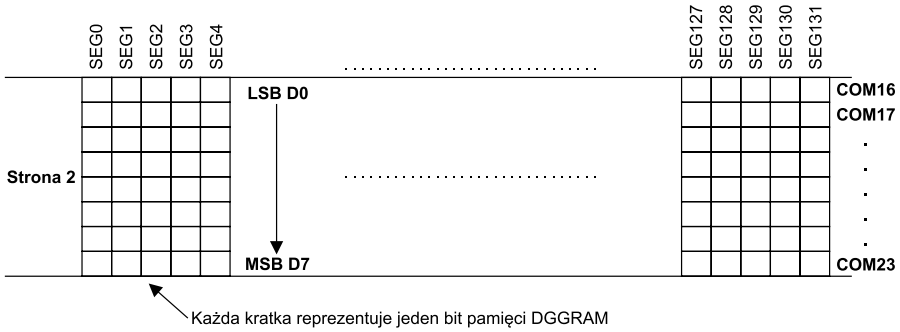
W trybie adresowania stron licznik kolumn jest zwiększany o 1 po każdym zapisie do pamięci GDDRAM. Po osiągnięciu adresu końcowego, licznik kolumn jest zerowany, ale zawartość licznika stron się nie zmienia. Aby zmodyfikować zawartość licznika stron, trzeba użyć komendy „ustawienie licznika stron" (kod B0h...B7h). Licznik kolumn ustawia się komendami „ustaw 4 młodsze bity adresu kolumn" (kod 00h...0Fh) oraz „ustaw 4 starsze bity adresu kolumn" (kod 10h...1Fh).



Rysunek 1. Schemat blokowy sterownika



Rysunek 2. Podział pamięci na strony



Rysunek 3. Zapisywanie bajtów do strony pamięci GDDRAM

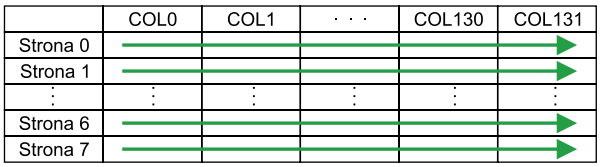
W trybach adresowania poziomego i pionowego zakresy zmian liczników są programowane komendami „ustawienie zakresu adresów kolumn" (kod 21h) i „ustawienie zakresu adresów stron" (kod 22h).

Komenda ustawiania zakresu adresów kolumn jest 3-bajtowa. Pierwszy bajt to kod komendy (21h), drugi to adres początku zakresu, a trzeci to adres końca zakresu. Po zapisaniu tej komendy adres początku jest wpisywany do licznika adresowego kolumn. Na przykład po przesłaniu sekwencji 21h, 02h, 20h, pierwszy bajt danych będzie zapisywany pod adresem odpowiadającym 2 kolumnie.

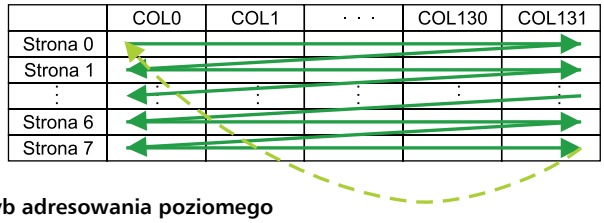
Podobnie działa komenda „ustawienie zakresu adresów stron". Składa się ona z 3 bajtów: kodu, adresu początkowego i adresu końcowego.

Zapisując po sobie te dwie komendy, ustala się adres początkowy zapisywanych danych (kolumna i strona) oraz zakresy zmian, istotne w trybach adresowania poziomego i pionowego. Ideę działania mechanizmu ustalania zakresów zmian adresów dla adresowania poziomego pokazano na **rysunku 7**.

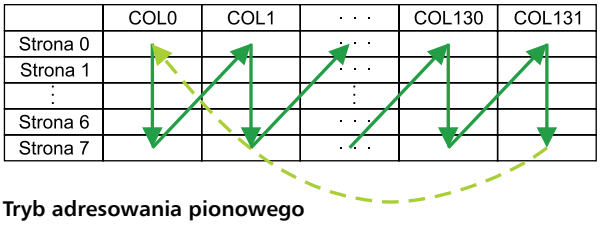
Po kolejnych wpisach do pamięci GDDRAM licznik kolumn jest inkrementowany od wartości początkowej (02d) do wartości końcowej (129d). Po osiągnięciu wartości końcowej, licznik kolumn jest ustawiany na wartość początkową, a licznik stron jest inkrementowany. Jest to powtarzane do momentu, aż oba liczniki osiągną wartości końcowe. Dalsze wpisy powodują ustawienie liczników na wartości początkowe i proces zaczyna się od nowa. W przypadku naszego wyświetlacza tej właściwości sterownika można użyć do wyświetlania niepełnowymiarowych bitmap.



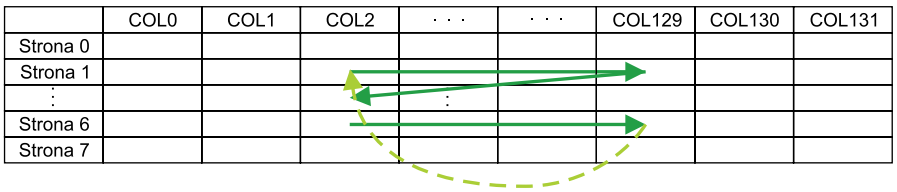
Rysunek 4. Tryb adresowania stron



Rysunek 5. Tryb adresowania poziomego



Rysunek 6. Tryb adresowania pionowego



Rysunek 7. Adresowanie po ustawieniu zakresów dla trybu poziomego

Komenda „ustawienie początkowej linii wyświetlania" pozwala przypisać pierwszą linię wyświetlacza do początkowych lokacji w pamięci GDDRAM. Jeżeli numer linii początkowej jest ustawiony na 0, to najmłodsze bity pierwszej strony pamięci będą odpowiadać pierwszej (w tym przypadku najwyższej) linii kreślonej na wyświetlaczu. Wpisanie przez tę komendę innego numeru linii spowoduje, że najmłodsze bity będą kreślić odpowiadającą mu linię na wyświetlaczu, a obraz będzie się wokół tej linii „zawijał". Jest to praktyczny i prosty sposób na przewijanie (skrolowanie) obrazu w pionie.

Wyświetlany obraz można programowo obracać o 180 stopni, tak aby można było montować wyświetlacz w obudowie w taki sposób, jak jest nam wygodnie. Do tego celu używane są komendy „remapowanie segmentów matrycy" i „ustawienie kierunku skanowania kolumn matrycy".

Kontrast wyświetlacza (prąd segmentów) jest ustawiany w 256 krokach 2-bajtową komendą „ustawienie kontrastu dla banku 0". Prąd segmentów jest wyliczany z wyrażenia $I_{SEG}=Kontrast/256\times I_{REF}\times współczynnik\ skalowania$. Poszczególne zmienne określają:

- Kontrast zmienia się w zakresie 0...255 i jest regulowany komendą ustawiania kontrastu.
- *Współczynnik skalowania* jest ustawiony na 32.
- I_{REF} to prąd referencyjny, którego wartość można ustawić, dołączając zewnętrzny rezystor do wyprowadzenia I_{REF} wyświetlacza. Napięcie na tym wyprowadzeniu wynosi VCC-3 V. Producent podaje, że prąd powinien mieć wartość 10 µA. Dla napięcia Vcc=12 V rezystor dołączony do I_{REF} powinien mieć rezystancję 910 kΩ.

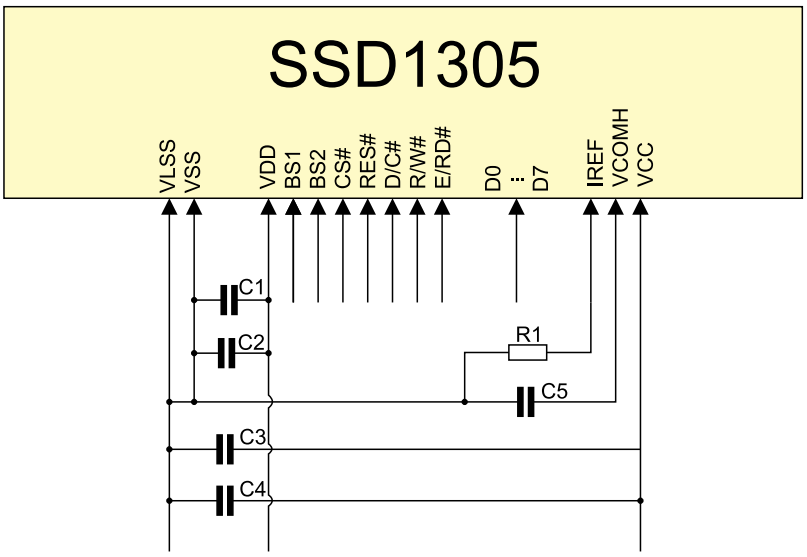
Sterowanie wyświetlaczem

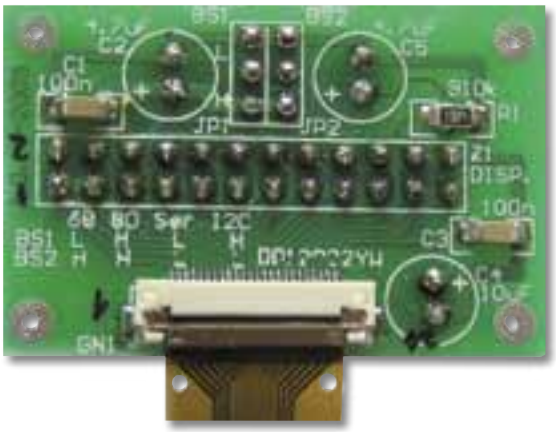
Do połączenia wyświetlacza z zewnętrznym sterownikiem zaprojektowano niewielką płytkę drukowaną, na której umieszczono: złącze Molex, goldpiny, kondensatory elektrolityczne oraz rezystor pomiędzy wyprowadzeniem IREF i masą. Schemat i płytki układu pośredniczącego pokazano na **rysunku 8**. Napięcie Vcc o wartości +12 V z zewnętrznego zasilacza stabilizowanego.

W czasie testowania wyświetlacza do komunikacji wybrano interfejs SPI przez wymu-

Listing 1. Definicja linii interfejsu SPI i zerowania

```
//definicja linii sterujących
#define DATA GPIO_Pin_4     //DATA PC4
#define CLK GPIO_Pin_5      //CLK PC5
#define CS GPIO_Pin_6       //CS PC6
#define RES GPIO_Pin_7      //RES PC7
#define DC GPIO_Pin_8       //DC PC8
#define CTRL GPIO_Pin_9     //CTRL PC9
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_data, ENABLE);
  GPIO_InitStructure.GPIO_Pin = D0 | D1 | D2 | D3 | D4| D5 | D6 | D7  ;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_Init(PORT_data, &GPIO_InitStructure);
```





Rysunek 8. Schemat i płytka układu pośredniczącego

szenie poziomów niskich na wejściach sterujących BS1 i BS2. Sterownikiem była płytka z mikrokontrolerem STM32. Program napisano w języku C za pomocą wersji ewaluacyjnej środowiska programistycznego Keil µVision4.

Interfejs SPI składa się z linii DATA (danych), CLK (zegarowej), CS (wyboru interfejsu) oraz DC (wybór dane/komendy). W programie zadeklarowano również linie: RES zerującą i CTRL VCC sterującą przekaźnikiem załączającym napięcie. Na **listingu 1** zamieszczono deklaracje funkcji portów mikrokontrolera.

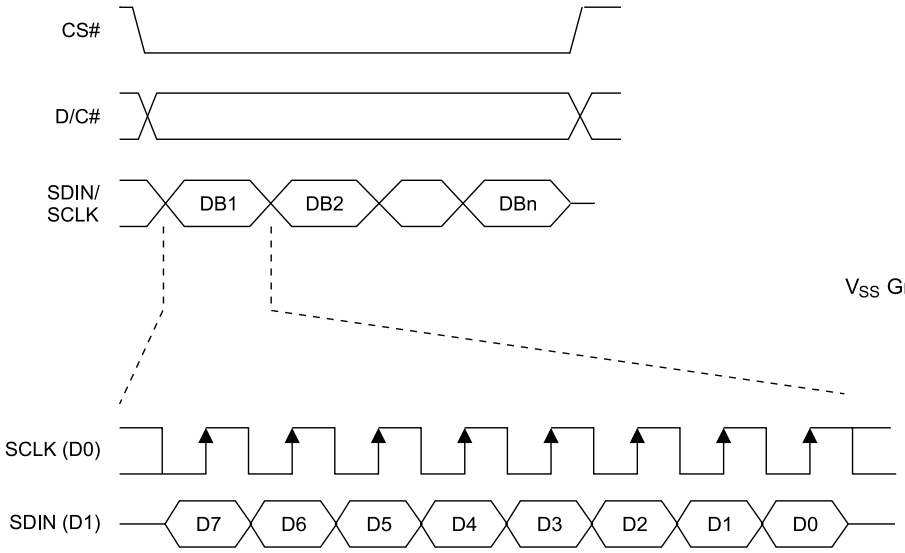
Na **listingu 2** pokazano programową implementację interfejsu SPI, łącznie z funkcjami manipulacji poziomami wyprowadzeń mikrokontrolera.

Do zapisywania komend i danych przeznaczono dwie funkcje: *WriteSpiCommand* i *WriteSPIData*. Obie używają *WriteSpi*, a różnią się tylko poziomem logicznym portu DC (**listing 3**).

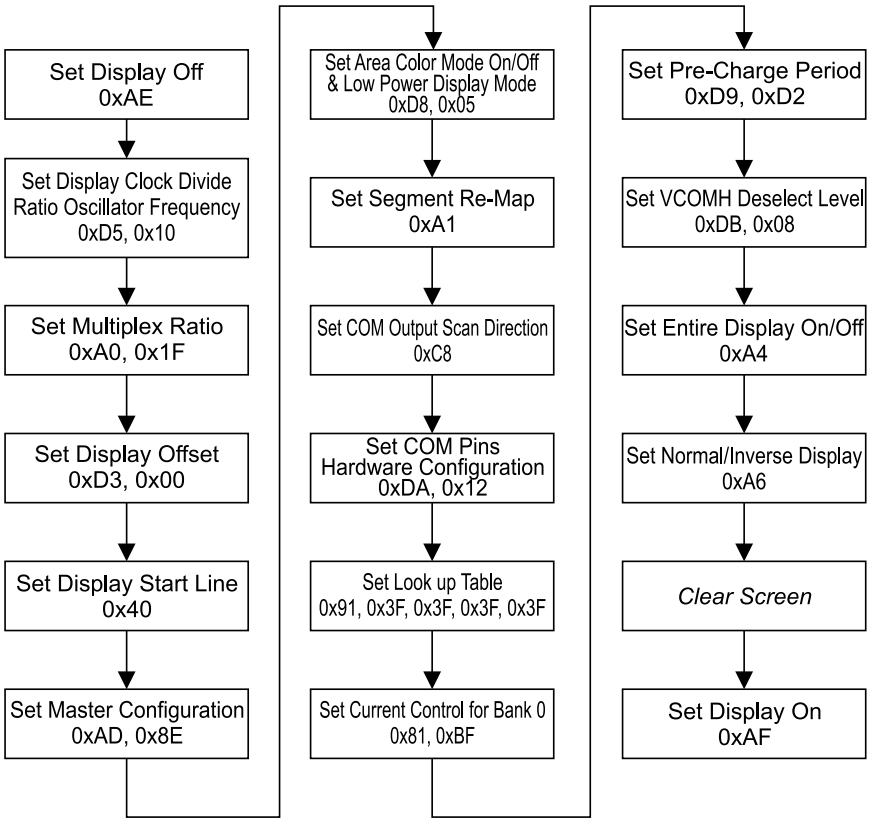
Pierwszą czynnością, jaką należy wykonać przed zapisem danych do pamięci GDDRAM, jest inicjalizacja sterownika wyświetlacza. Producent wyświetlacza znacznie ułatwił pracę programistom, zamieszczając w danych technicznych sekwencje komend, które należy wysłać do sterownika w czasie inicjalizacji. Pokazano je na **rysunku 10**. Prawidłowa inicjalizacja pozwala na maksymalne wydłużenie czasu eksploatacji wyświetlacza. W przy-

Tabela 2. Wykaz najważniejszych komend sterownika SSD1305

D/C	Hex	Dane	Komenda	Opis
0	00-0f	0000 x3 x2 x1 x0	Ustawienie 4 młodszych bitów licznika kolumn dla trybu adresowania stron	Ustawienie 4 młodszych bitów licznika kolumn dla trybu adresowania stron. Po zerowaniu do x3...x0=0000
0	10-1f	0001 x7 x6 x5 x4	Ustawienie 4 starszych bitów licznika kolumn dla trybu adresowania stron	Ustawienie 4 starszych bitów licznika kolumn dla trybu adresowania stron. Po zerowaniu do x7...x4=0000
0 0	20 A[1:0]	0010 0000 **** **A1 A0	Ustawienie trybu adresowania pamięci obrazu	A[1:0] =00 tryb horyzontalny, =01 tryb wertykalny, =10 tryb adresowania stron (domyślny), =11 niedopuszczalny
0 0 0	21 A[7:0] B[7:0]	0010 0001 A7…A0 B7…B0	Ustawienie zakresu adresów kolumn	Ustawienie numeru kolumny początkowej A[7:0] 0...131 i końcowej B[7:0]. 0...131
0 0 0	22 A[7:0] B[7:0]	0010 0010 **** *A2 A1 A0 **** *B2 B1 B0	Ustawienie zakresu adresów stron	Ustawienie numeru kolumny początkowej A[2:0] (0...7) i końcowej B[7:0] (0...7)
0	40-4f	01 x5 x4 x3 x2 x1 x0	Ustawienie początkowej linii wyświetlania	Ustawienie przypisania początku pamięci RAM obrazu do rejestru początkowej linii wyświetlacza (zakres 0...63)
0 0	81 A[7:0]	1000 0001 A7….A0	Ustawienie kontrastu dla banku 0	Ustawienie kontrastu w 256 krokach dla banku 0 pamięci obrazu. Po zerowaniu ustawiane na 80h
0 0 0 0 0	91 X[5:0] A[5:0] B[5:0] C[5:0]	1001 0001 ** X5…X0 **A5…A0 **B5…B0 **C5…C0	Ustawienie tablicy LUT Look Up Table	Ustawienie prądu driverów dla banku 0 (X5...X0) i banków kolorów (A5...A0, B5...B0, C5...C0)
0	A0/A1	1010 000 X0	Remapowanie segmentów matrycy	X0=0 adres kolumny 0 jest mapowany na segment 0 matrycy X0=1 adres kolumny 131 jest mapowany na segment 0 matrycy
0	A4/A5	1010 010 X0	Włączenie wyświetlania pamięci obrazu	X0=0 wyświetlanie włączone X0=1 wyświetlanie wyłączone
0	A6/A7	1010 011 X0	Wyświetlanie normalne/ inwersja	X0=1 normalne 0 w RAM piksel zgaszony 1 w RAM piksel zapalony X0=1 inwersyjne 1 w RAM piksel zgaszony 0 w RAM piksel zapalony
0 0	A8 A[5:0]	1010 1000 ** A5…A0	Ustawienie współczynnika multipleksowania	Ustawienie współczynnika multipleksowania N+1 A[5:0] programuje współczynnik od 16 do 64. Wartości od 0 do 14 są niedozwolone
0 0	AD A[7:0]	1010 1101 1000 111 A0	Master Configuration	A0=0 wybranie zewnętrznego zasilania matrycy A0=1 rezerwa
0 0 0	AC AE AF	1010 11A1 A0	Włączenie/wyłączenie wyświetlania	AC włączenie wyświetlania w trybie dim AE wyłączenie (domyślne po zerowaniu) AF włączenie wyświetlania w trybie normalnym
0	B0-B7	10110 X2 X1 X0	Ustawienie licznika stron	Ustawienie licznika stron (0…7) w trybie adresowania stron
0	C0/C8	1100 X3 000	Ustawienie kierunku skanowania kolumn matrycy	X3=0 tryb normalny (po zerowaniu), skanowanie od 0 do N–1. X3=1 tryb remapowania, skanowanie od N–1 do 0
0	D3	1101 0011 ** A5…A0	Offset wyświetlania	Ustawienie pionowego przesunięcia obrazu (0...63)
0 0	D5 A[7:0]	1101 0101 A7…A0	Ustawienie podzielnika zegara/częstotliwości oscylatora	A[3:0] +1 podzielnik zegara – po zerowaniu ustawiany na 0000 A[7:4] ustawienie częstotliwości zegara – po zerowaniu ustawiany na 0111 (7dec).
0 0	D9 A[7:0]	1101 1000 A7…A0	Ustawienie faz zegara	A[3:0] faza 1 od 1 do 15 wartość 0 niedozwolona A[7:4] faza 2 od 1 do 15 wartość 0 niedozwolona
0 0	DA	1101 1010 00 x5 x4 0010	Sprzętowa konfiguracja wyprowadzeń COM	X[4]=0 konfiguracja sekwencyjna X[4]=1 konfiguracja alternatywna X[5]=0 zablokowanie remapowania X[5]=1 odblokowanie remapowania
0 0	DB A[5:2]	1101 1011 00 A5..A2 00	Ustawienie Vcomh	00hex ~0,43xVcc 34hex ~0,77xVcc 3Chex~0,83xVcc



Rysunek 9. Przebiegi czasowe w trakcie zapisu przez SPI



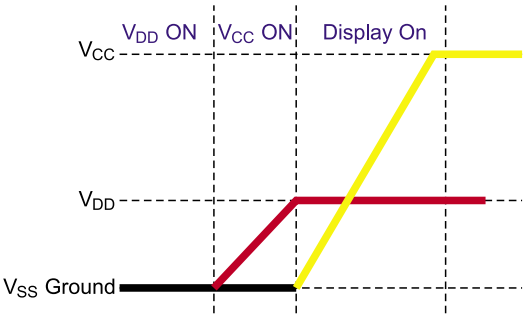
Rysunek 10. Sekwencja inicjalizacji wyświetlacza zalecana przez producenta

Listing 2. Programowa implementacja interfejsu SPI

```
//procedura programowej implementacji SPI
void WriteSpi( unsigned char data){
unsigned char i;
  SetBit(CLK);
    for (i=0;i<8;i++) {
      ClrBit(CLK);
      if ((data&0x80)==0) ClrBit(DATA);
        else SetBit(DATA);
      SetBit(CLK);
      data<<=1;
    }
  SetBit(CS);
}

//ustawienie linii portu
void SetBit(int bits){
  GPIO_SetBits(PORT_ctrl, bits);
}

//zerowanie linii portu
void ClrBit(int bits){
  GPIO_ResetBits(PORT_ctrl, bits);
}
```



Rysunek 11. Sekwencja załączenia zasilania

padku matryc OLED jest to szczególnie ważne.

Na **listingu 4** zamieszczono funkcję inicjalizacji wyświetlacza DD12832YW-1A. Rozpoczyna się ona od zainicjowania wyprowadzeń SPI i wysłania sekwencji zerującej do sterownika SSD1305. Potem kolejno, zgodnie z rysunkiem 10, przesyłane są odpowiednie komendy. Posługując się dokumentacją sterownika, można eksperymentować z komendami odpowiedzialnymi za orientację wyświetlanej informacji i ewentualnie ustawienia kontrastu. Pozostałe komendy należy modyfikować ostrożnie, bo mogą wpłynąć na trwałość wyświetlacza.

Funkcja *Cls()* (**listing 5**) zeruje pamięć GDRAM. Ponieważ wielkość wyświetlacza jest mniejsza od maksymalnej obsługiwanej przez SSD1305, użyto komend ustawiania zakresu zmian liczników kolumn i stron.

Wyjaśnienia wymaga wywołanie ustawienia linii CTRL – *SetBit(CTRL)*. Producent wyświetlacza zaleca, żeby w czasie wykonywania sekwencji inicjalizacji nie było napięcia na doprowadzeniu Vcc. W czasie prób z wyświetlaczem napięcie Vcc było włączane przekaźnikiem sterowanym linią CTRL.

Sekwencja włączania napięcia zasilającego powinna zaczynać się od włączenia napięcia zasilającego układy cyfrowe (Vdd) przy wyłączonym napięciu Vcc. Następnie trzeba wysłać komendę wyłączenia wyświetlacza i sekwencję inicjalizacji, łącznie z zerowaniem pamięci GDDRAM. Po załączeniu Vcc i odczekaniu ok. 100 ms można wysłać komendę załączającą wyświetlacz (**rysunek 11**).

Przy wyłączaniu zasilania trzeba najpierw wyłączyć napięcie zasilania matrycy Vcc, a po upływie ok. 100 ms napięcie za-

Listing 3. Procedury zapisu komend i danych do sterownika

```
//wysłanie komendy
void WriteSpiCommand(unsigned char cmd){
  ClrBit(CS);
  ClrBit(DC);//DC =0 - komenda
  WriteSpi(cmd);
}

//wysłanie danej
void WriteSpiData(unsigned char data){
  ClrBit(CS);
  SetBit(DC);//DC =1 - dane
  WriteSpi(data);
}
```

Listing 4. Inicjalizacja wyświetlacza
```
void InitLcd(void) {
  short i;
  ClrBit(CTRL);//wyłącz VCC
  SetBit(CS);//te linie w poziomie wysokim
  SetBit(CLK);
  SetBit(DATA);
  ClrBit(RES);
  for(i=0;i<8000;i++)
  delay();
  SetBit(RES);
  for(i=0;i<8000;i++)
  delay();
  WriteSpiCommand(0xae);//Display off
  WriteSpiCommand(0xd5);//Clock Divide/oscillator freq
  WriteSpiCommand(0x10);
  WriteSpiCommand(0xA8);//multiplex ratio
  WriteSpiCommand(0x1f);
  WriteSpiCommand(0xd3);//display offset
  WriteSpiCommand(0x00);
  WriteSpiCommand(0x40);//display start line
  WriteSpiCommand(0xad);//master configuration
  WriteSpiCommand(0x8e);
  WriteSpiCommand(0xd8);//area color mode
  WriteSpiCommand(0x05);
  WriteSpiCommand(0xa1);//segment re-map
  WriteSpiCommand(0xc8);//output scan direction
  WriteSpiCommand(0xda);//com pins hardware configurations
  WriteSpiCommand(0x12);
  WriteSpiCommand(0x91);//look up table
  WriteSpiCommand(0x3f);
  WriteSpiCommand(0x3f);
  WriteSpiCommand(0x3f);
  WriteSpiCommand(0x3f);
  WriteSpiCommand(0x81);//current control bank0
  WriteSpiCommand(0xbf);
  WriteSpiCommand(0xd9);//pre-charge period
  WriteSpiCommand(0xd2);
  WriteSpiCommand(0xdb);//deselected level
  WriteSpiCommand(0x08);
  WriteSpiCommand(0xa4);//entire display
  WriteSpiCommand(0xa6);//normal/inverse display
  Cls();
  Delay(0x3ffff);
  SetBit(CTRL);//wlacz VCC
  Delay(0x3ffff);
  WriteSpiCommand(0xaf);//display on
}
```

silania układów cyfrowych sterownika Vdd.

Wyświetlacze graficzne doskonale nadają się do pracy w trybie tekstowym. Przy rozdzielczości 128×64 piksele można wyświetlić 4 linijki każda po 21 znaków 6×8 pikseli. Do pracy w trybie tekstowym potrzebna jest tablica z wzorami czcionek, ponieważ sterownik nie ma wbudowanego generatora znaków. Ja użyłem tablicy *rom_gen[]* pierwotnie zdefiniowanej dla wyświetlacza od telefonu Nokia 3310. Pozycja początkowa tekstu do wyświetlenia jest ustalana przez funkcję *Poz()* zamieszczoną na **listingu 6**.

Zamieszczona na **listingu 7** funkcja *WriteChar* wyświetla znak o kodzie ASCII umieszczonym w argumencie *code*. Zmiennej *code_point* jest nadawana wartość indeksu tablicy *rom_gen* wyliczana na podstawie kodu znaku. Potem 6 kolejnych bajtów jest pobieranych z *rom_gen* i wpisywanych do pamięci GDDRAM. Funkcje z listingów 6 i 7 wykorzystano do praktycznej realizacji funkcji wyświetlającej ciąg znaków od ustalonej pozycji (**listing 8**).

Listing 5. Zerowanie pamięci DDRAM
```
void Cls(void){
  int i;
  WriteSpiCommand(0x20);
  WriteSpiCommand(0x00);
  WriteSpiCommand(0x21);//zakres kolumn
  WriteSpiCommand(0);
  WriteSpiCommand(131 );
  WriteSpiCommand(0x22);//zakres stron
  WriteSpiCommand(0);
  WriteSpiCommand(7);
  for (i=0;i<528;i++) WriteSpiData(0x00);
}
```

Listing 6. Pozycja początkowa tekstu
```
//ustalenie pozycji na wyswietlaczu
void Poz(char x, char y)
{
  x=x*6+5;
  WriteSpiCommand(y|0xB0);//ustawienie licznika wierszy
  WriteSpiCommand(x&0x0F);//ustawienie licznika kolumn
  WriteSpiCommand(((x&0xF0)>>4)|0x10);
}
```

Listing 7. Wyświetlanie znaku 6×8 pikseli
```
//wyświetlenie znaku ASCII 8x6 pixeli
void WriteChar(char code){
  int code_point;
  code_point=((int)code*6);
  WriteSpiData(rom_gen[code_point++]);
  WriteSpiData(rom_gen[code_point++]);
  WriteSpiData(rom_gen[code_point++]);
  WriteSpiData(rom_gen[code_point++]);
  WriteSpiData(rom_gen[code_point++]);
  WriteSpiData(rom_gen[code_point]);
}
```

Listing 8. Wyświetlanie napisu od ustalonej pozycji
```
void DispTxt(const char *napis,char x,char y){
  Poz(x,y);//pozycja tekstu na wyswietlaczu
  while(*napis!=0)
  {WriteChar(*napis);
  ++napis;}
}
```

Listing 9. Przykład procedury wyświetlającej bitmapę
```
void WriteBmp(void){
  char ptr;
  int i;
  i=0;
  Poz(0,0);
  for(ptr=0;ptr<122;ptr++)
  WriteSpiData(bmparray[i++]);
  Poz(0,1);
  for(ptr=0;ptr<122;ptr++)
  WriteSpiData(bmparray[i++]);
  Poz(0,2);
  for(ptr=0;ptr<122;ptr++)
  WriteSpiData(bmparray[i++]);
  Poz(0,3);
  for(ptr=0;ptr<122;ptr++)
  WriteSpiData(bmparray[i++]);
}
```

Jeszcze mniej skomplikowane jest wyświetlanie bitmap, ale pod warunkiem że dysponujemy programem przekształcającym mapę monochromatyczną na tablicę liczb. Konwersja musi być wykonana zgodnie z zasadą pokazaną na rysunku 3. Przykład funkcji wyświetlające bitmapę przygotowaną dla wyświetlacza o rozdzielczości 122×64 piksele pokazano na **listingu 9**. Wykorzystano w niej wyświetlanie linijka po linijce. Rozpoczęcie wyświetlania nowej linijki jest ustalane przez funkcję *Poz()*. Jest to jeden ze sposobów wyświetlania bitmapy mniejszej niż maksymalna rozdzielczość wyświetlacza. Być może bardziej naturalnym sposobem byłoby wykorzystanie komend ustawiania zakresu adresów kolumn i stron, tak jak to zrobiłem w funkcji *Cls()*.

Podsumowanie

Opisywany wyświetlacz ma klarowny i wyraźny obraz. Jest bardzo cienki, a więc na pewno będzie nadawał się do urządzeń, w których jest mało przestrzeni. Dużą zaletą jest możliwość wyboru rodzaju interfejsu komunikacyjnego, w tym aż dwóch szeregowych. Wadą jest brak przetwornicy zasilającej matrycę. Sterownik mógłby mieć wbudowane układy sterowania przetwornicą, tak jak to jest w przypadku SSD1303. Oczywiście taką przetwornicę można łatwo zbudować jako układ zewnętrzny, tym do zasilania potrzebny jest prąd rzędu 20...30 mA. Wyświetlacz jest droższy od standardowych modułów LCS, ale w wielu zastosowaniach sprawdzi się lepiej niż one.

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