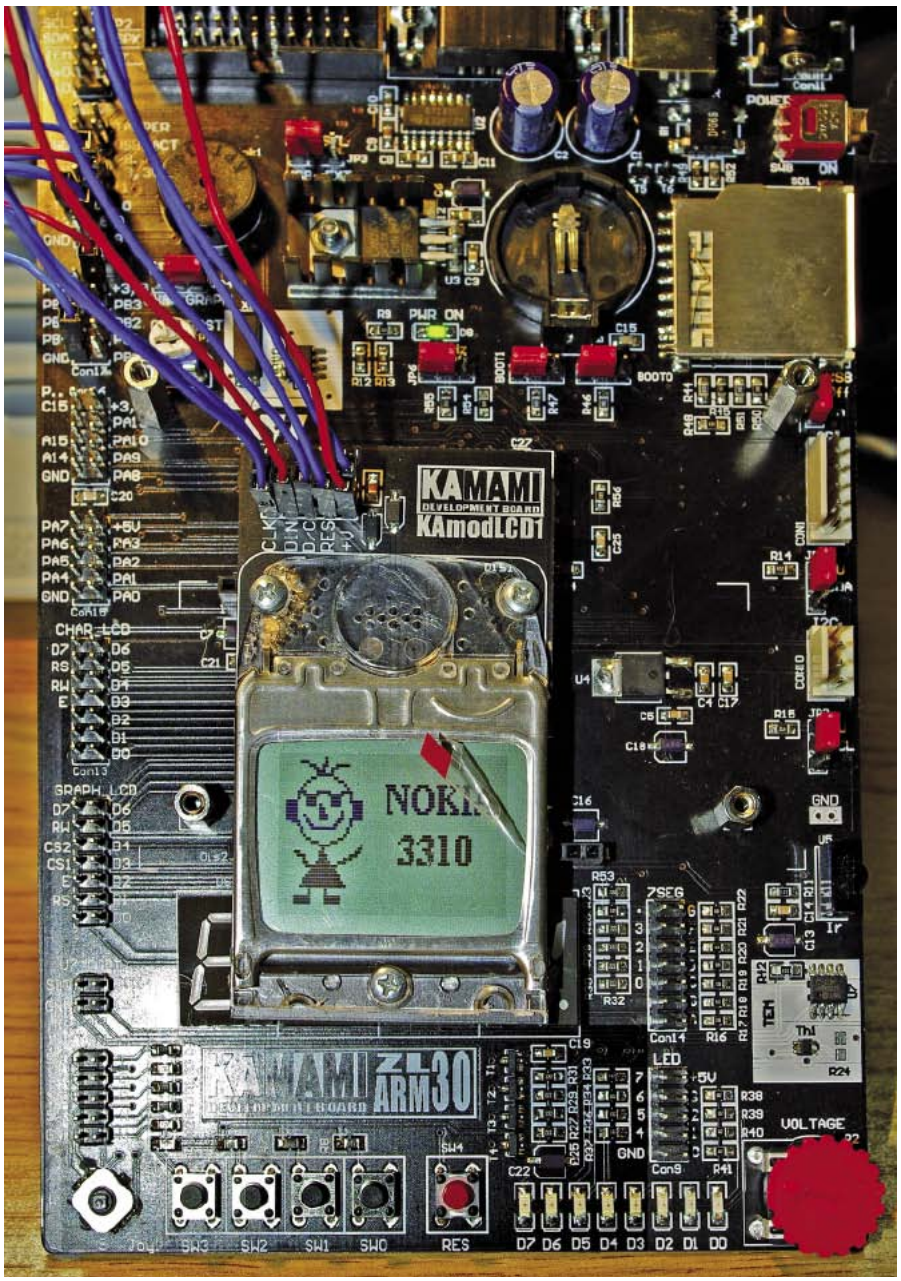
Wyświetlacz graficzny do Nokii 3310

Stary wyświetlacz, nowe problemy...

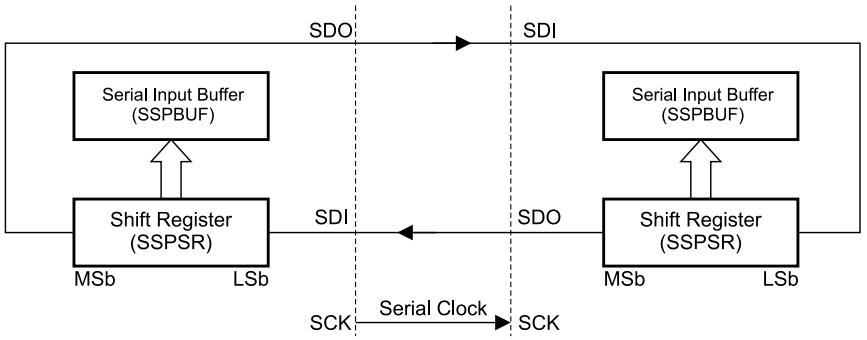
Wyświetlacze LCD od telefonów komórkowych idealnie się nadają do zastosowania w układach z mikrokontrolerem. Telefon Nokia 3310 nie jest już oferowany przez operatorów sieci komórkowych, ale olbrzymia liczba sprzedanych egzemplarzy spowodowała, że można już za około 10 złotych kupić do niego wyświetlacz oferowany jako część zamienna. Niestety, okazuje się, że mimo deklarowanej zgodności z wcześniejszymi modelami wyświetlaczy, przy próbach użycia niektórych z nich mogą pojawić się pewne problemy.



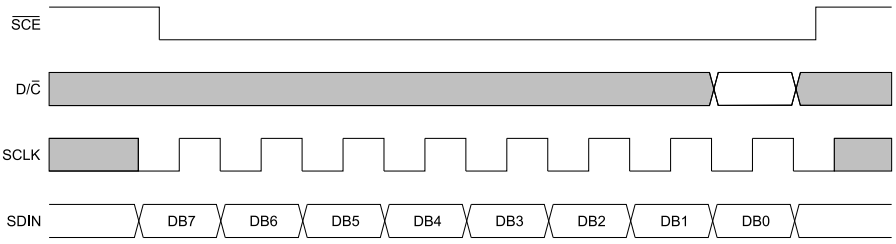
Matryca wyświetlacza do Nokii 3310 ma rozdzielczość 84×48 piksele. W trybie tekstowym może wyświetlić 14 znaków w 6 liniach. W porównaniu z możliwościami popularnych wyświetlaczy alfanumerycznych (najczęściej 2×16 do 4×20 znaków) jest to całkiem sporo. Wyświetlacz został zaprojektowany do urządzenia przenośnego, więc obszar wyświetlania matrycy LCD nie jest zbyt duży – ma wymiary 30×24 mm, a grubość całego modułu jest również niewielka i wynosi ok. 3 mm.

Sterowanie wyświetlaczem tego typu było już opisywane łamach Elektroniki Praktycznej. Również w Internecie można znaleźć sporo opisów z gotowymi procedurami sterującymi napisanymi dla różnych mikrokontrolerów i w różnych językach programowania. Sam stosowałem ten wyświetlacz jako część interfejsu użytkownika w kilku swoich projektach. I pewnie jego popularność nie byłaby niczym zagrożona, gdyby nie to, że pojawiły się problemy ze sterowaniem niektórych egzemplarzy. Ostatnio w moje ręce trafiło kilka wyświetlaczy, z których część zachowywała się zupełnie poprawnie, a część nie chciała działać sterowana moimi, wielokrotnie sprawdzonymi procedurami. Po wykonaniu serii prób modyfikacji tych procedur zacząłem szukać w Internecie czy ktoś jeszcze nie ma podobnych problemów. Okazało się, że jest to znany problem i co gorsza, nikt nie potrafi podać sposobu jego rozwiązania.

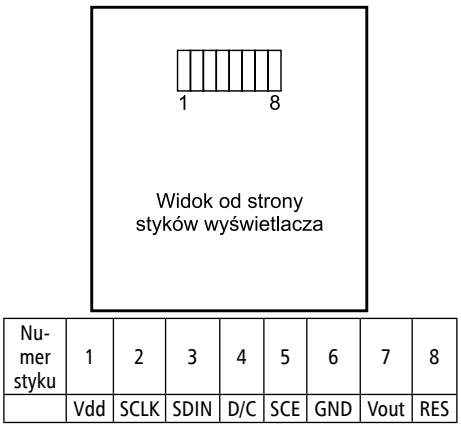
Wszystkie znane mi opisy wyświetlacza od telefonu Nokia 3310 podają, że ma wbudowany sterownik NXP typu PCD8544 i na



Rys. 1. Połączenie dwóch układów interfejsem SPI



Rys. 2. Przesłanie bajtu do wyświetlacza.



Numer styku	1	2	3	4	5	6	7	8
	Vdd	SCLK	SDIN	D/C	SCE	GND	Vout	RES

Rys. 3. Widok wyprowadzeń wyświetlacza

podstawie dokumentacji do tego sterownika oparte są procedury sterujące. Dla ugruntowania wiedzy przypomnijmy pokrótce budowę i sposób działania układu PCD8544.

Działanie sterownika wyświetlacza

Sterownik PCD8544, w który wyposażono wyświetlacz, ma wbudowane w swojej strukturze:

- interfejs przystosowany do sterowania matrycy LCD, zawierający wzmacniacze kolumn i wierszy, układ wytwarzania napięć zasilających matrycę i układ kompensacji ustawienia kontrastu w funkcji temperatury,
- pamięć RAM,
- interfejs szeregowy.

Nas będzie interesował przede wszystkim fizyczny interfejs użytkownika, organizacja pamięci RAM i sposób wymiany informacji pomiędzy mikrokontrolerem, a sterownikiem.

Jak to działa?

Przesyłanie danych pomiędzy sterownikiem wyświetlacza i mikrokontrolerem odbywa się przez interfejs szeregowy SPI. Długość słowa danych wynosi 8 bitów. Większość dostępnych mikrokontrolerów ma wbudowany sprzętowy interfejs SPI. Również programowa implementacja interfejsu nie jest skomplikowana.

Interfejs SPI jest zbudowany się z 3 linii: wejścia DIN, wyjścia DOUT, zegara SCK. Czasami do sterowania funkcjami interfejsu używa się również SS (*Slave Select*). Typowo transmisja przeprowadzana z punktu do punktu, w konfiguracji układ zarządzający (*Master*) – układ podrzędny (*Slave*).

Na **rys. 1** pokazano dwa układy połączone za pomocą SPI. Przy podłączeniu wyświetlacza układem nadrzędnym, generującym sygnał SCK, jest mikrokontroler, natomiast sterownik wyświetlacza jest układem podrzędnym. Ponieważ dane są tylko wpisywane do sterownika, to wykorzystywana jest tylko linia danych wejściowych.

Interfejs szeregowy PCD8544 został uzupełniony o dodatkowe linie:

- SCE służącą do uaktywnienia układu (odpowiednik *Chip Select*)
- D/C do wyboru miejsca przeznaczenia wysyłanych danych: dla D/C=1 jest to pamięć RAM wyświetlacza (dane), natomiast dla D/C=0 rejestr sterujący (rozkazy).
- RES służąca do zerowania sterownika wyświetlacza.

Na **rys. 2** pokazano przebiegi czasowe przy przesyłaniu bajtu do wyświetlacza.

Transmisja rozpoczyna się od uaktywnienia linii SCE (stan niski). Dane doprowadzone do SDIN są wpisywane do rejestru sterownika przy narastającym zboczu zegara SCK. Na rys. 1, po wpisaniu pojedynczego bajtu, SCE staje się nieaktywne. Sterownik akceptuje wpisywanie bloku bajtów z aktywnym SCE (bez konieczności przechodzenia w stan wysoki tej linii po każdym przesłanym bajcie). Przyśpiesza to przesyłanie danych, ale przy większych blokach i dużej prędkości transmisji może powodować przekłamania. Każde opadające zbocze SCE synchronizuje transmisję i jeżeli pojawiają się problemy, to lepiej zastosować sterowanie linią SCE, tak jak na rys. 2.

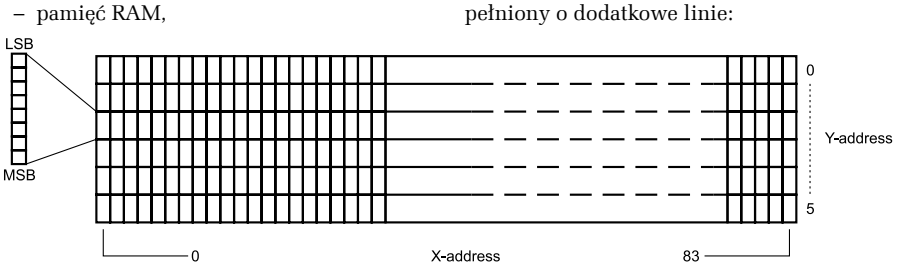
Wyprowadzenia wyświetlacza telefonu Nokia 3310 pokazano na **rys. 3**. Napięcie zasilające wyświetlacz (styk Vdd) może mieć wartość z zakresu 2,7...3,3 V. Napięcie zasilające matrycę wyświetlacza (6...8 V), wytwarzane w wewnętrznej przetwornicy sterownika, jest wyprowadzone na styk Vout. Do tego styku podłącza się zewnętrzny kondensator filtrujący o pojemności 1 μF.

Wbudowana w sterownik pamięć RAM jest zapisywana po wymuszeniu stanu wysokiego na linii D/C. Matryca wyświetlacza może wyświetlić 48 linii. Każda linia ma 84 piksele. Pamięć jest zorganizowana w 6 banków po 84 bajty każdy. Taka organizacja wymusza logiczny podział pola wyświetlacza na 6 wierszy po 84 kolumny (**rys. 4**). Wiersze są numerowane od 0 do 5, a kolumny od 0 do 83.

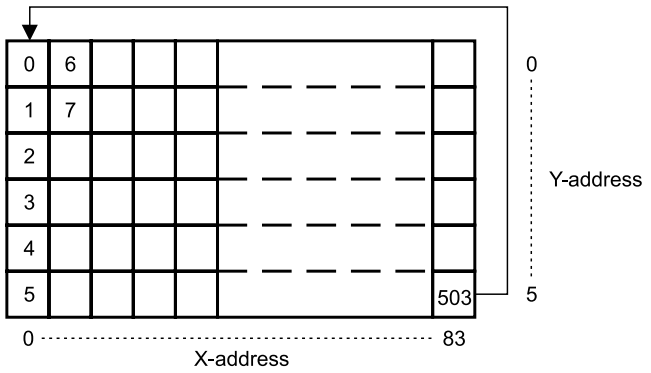
W czasie wpisywania danej do pamięci najpierw określamy numer banku (zmienna Y określająca wiersz na rys. 4), a następnie numer bajtu w wierszu (zmienna X określająca kolumnę na rys. 4). Zapisanie całego banku odpowiada wyświetleniu jednego wiersza na wyświetlaczu. W trybie tekstowym, odpowiada to wyświetleniu pojedynczej linijki tekstu.

Każdy bajt pamięci RAM jest wyświetlany jako pionowy pasek składający się z 8 pikseli. Najwyżej położony piksel w pasku odpowiada bitowi najmniej znaczącemu, a najniżej – najbardziej znaczącemu. W standardowym trybie wyświetlania, ustawienie bitu w bajcie pamięci odpowiada zaświeceniu, a wyzerowanie zgaszeniu piksela.

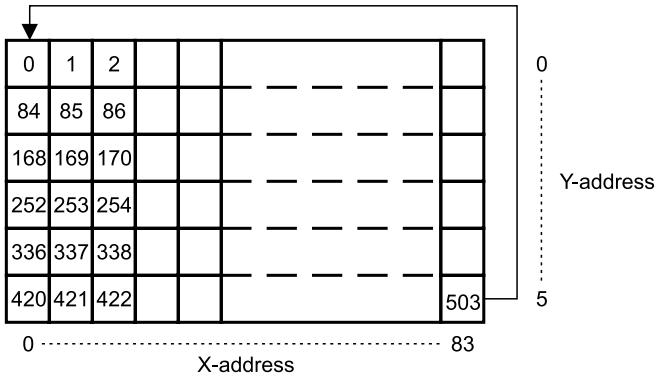
Przy wpisywaniu bajtów do pamięci RAM wykonywana jest przez sterownik PCD8544 automatyczna inkrementacja



Rys. 4. Organizacja pamięci RAM PCD8544



Rys. 5. Adresowanie pionowe



Rys. 6. Tryb adresowania poziomego

liczników wierszy i kolumn. Dostępne są ustawiane komendą *Function Set* dwa tryby automatycznego modyfikowania liczników: pionowy i poziomy.

Tryb pionowy pokazano na **rys. 5**. W tym trybie, po każdym zapisaniu bajtu do pamięci licznik wierszy jest inkrementowany. Po osiągnięciu maksymalnego numeru wiersza licznik wierszy jest zerowany, a licznik kolumn jest inkrementowany. Wysłanie 6 bajtów 0xFF, rozpoczynając od wiersza 0, spowoduje wyświetlenie po lewej stronie wyświetlacza pionowego paska ciągnącego się przez całą wysokość wyświetlacza. Tryb adresowania pionowego można stosować np. przy wpisywaniu pełnoekranowej bitmapy.

Tab. 1. Wykaz komend akceptowanych przez sterownik wyświetlacza Nokii 3310

Instrukcja	D/C	Bajt komendy								Opis
		DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
H=0 lub H=1										
NOP	0	0	0	0	0	0	0	0	0	Nic nie rób
Function Set	0	0	0	1	0	0	PD	V	H	Tryb obniżonego poboru energii (PD), adresowania (V) i komendy rozszerzone (H)
Write Data	1	D7	D6	D5	D4	D3	D2	D1	D0	Zapisanie pamięci RAM
H=0 podstawowy zestaw komend										
Rezerwa	0	0	0	0	0	0	1	X	X	Nie używać
Display Control	0	0	0	0	0	1	D	0	E	Konfiguracja wyświetlacza
Rezerwa	0	0	0	0	1	x	x	X	X	Nie używać
Set Y address	0	0	1	0	0	0	Y2	Y1	Y0	Ustawienie licznika wierszy 0…5
Set X address	0	1	X6	X5	X4	X3	X2	X1	X0	Ustawienie licznika kolumn 0…83
H=1 Rozszerzony zestaw komend										
Rezerwa	0	0	0	0	0	0	0	0	1	Nie używać
Rezerwa	0	0	0	0	0	0	0	1	X	Nie używać
Temperature Control	0	0	0	0	0	0	1	TC1	TC0	Współczynnik temperaturowy
Rezerwa	0	0	0	0	0	1	x	X	X	Nie używać
Bias System	0	0	0	0	1	0	BS2	BS1	BS0	Ustawienie BIAS
Rezerwa	0	0	1	X	X	X	x	X	X	Nie używać
Set Vop	0	1	Vop6	Vop5	Vop4	Vop3	Vop2	Vop1	Vop0	Napięcie zasilania matrycy (kontrast)

Tryb adresowania poziomego pokazano na **rys. 6**. Po zapisaniu każdego bajtu jest inkrementowany licznik kolumn. Po osiągnięciu maksymalnego numeru kolumny, licznik kolumn jest zerowany, a licznik wierszy jest inkrementowany. Tryb adresowania poziomego jest bardzo wygodny przy pracy w trybie tekstowym.

Tryby adresowania można dowolnie przełączać w trakcie normalnej pracy wyświetlacza. Nie ma to wpływu na wyświetlaną informację.

Zestawy komend

Sterownik akceptuje dwa zestawy komend: standardowy i rozszerzony (**tab. 1**). Komenda Function Set jest przeznaczona do programowania wyświetlacza w tryb obniżonego poboru energii, sposobu adresowania pamięci RAM i ustawiania zestawu komend (normalny lub rozszerzony). Kontrast wyświetlacza zależy od napięcia zasilającego matrycę LCD i jest ustawiany komendą *Set Vop*. Można ustawić 128 wartości, ale użyteczny zakres regulacji to nastawy 30...90.

Komenda *Temperature Control* programuje jedną z 4 predefiniowanych charakterystyk temperaturowej kompensacji kontrastu. Dla większości zastosowań odpowiedni jest współczynnik 2 (TC1=1, TC0=0). Dla wyświetlacza od telefonu Nokia 3310 *BIAS* powinien mieć wartość 3.

Problemy

Do testowania wyświetlaczy użyłem modułu KAmodLCD1 (jednej z tych prostych rzeczy, które ułatwiają życie) i płyty ewaluacyjnej ZL30ARM z procesorem STM32.

Pierwszym problemem jaki napotkałem było niewłaściwe zerowanie pamięci wyświetlacza. Po włączeniu zasilania sterownik nie inicjalizuje pamięci i dlatego są w niej pewne nieokreślone wartości. Dlatego zerowanie jest jednym ze składników procedury inicjalizacji.

Pamięć sterownika PCD8544 ma 504 bajty (rys. 5 i 6). Wystarczy wpisać do niej kolejno 504 bajty równe 0, aby ekran wyświetlacza po inicjalizacji był czysty (**list. 1**), jednak po wykonaniu takiego zerowania nie wszystkie piksele na wyświetlaczu zostały zgaszone (**fot. 7**). Wyglądało na to, że nie cała pamięć została wyzerowana. Skoro tak, to w pierwszej próbie zacząłem zwiększać liczbę wpisywanych danych. Zwiększanie ilości wpisywanych danych o niewielką wartość nie dawało żadnych efektów. Dopiero zapełnienie pamięci 550 zerami spowodowało, że obszar na dole zaczynał być gaszony. Próbowałem więc dalej. Po wpisaniu 612 bajtów ekran został wyczyszczony, ale nie cały. Je-

List. 1. Zerowane pamięci wyświetlacza

```
for(i=0;i<504;i++)
WriteData(0x00); //zerowanie pamięci
RAM wyświetlacza
```

Fot. 7. Problem z czyszczeniem ekranu wyświetlacza

Fot. 8. Nieprawidłowe wyświetlania bitmapy

żeli dokładnie przyjrzeć się fot. 8, to widać, że obszar z przypadkowymi wartościami ma szerokość 11 pikseli. Pozostał wąski fragment o szerokości 3 pikseli. Wpisanie kolejnych 102 bajtów zlikwidowało również i ten pasek.

Z tych doświadczeń wynikały pewne wnioski. Pierwszy to taki, że sterownik wyświetlacza ma inną organizację pamięci niż PCD8544, a zatem nie może to być PCD8544. Z wyliczeń wpisywanych bajtów w trakcie zerowania wynika, że pamięć wyświetlacza tego sterownika odpowiada poziomej rozdzielczości równej 102 piksele. Najprawdopodobniej sterownik może sterować matrycami o rozmiarze 102×64 lub 102×65 pikse-

List. 2. Zmodyfikowana funkcja zapisu bitmapy

```
//ustawia pozycje wyświetlania (14 kolumn, 6 wierszy)
void WriteBmp(const unsigned char *buffer)
{
  char j,k;
  for(k=0;k<6;k++){
    Poz(0,k);//pierwsza linijka
    for(j=0;j<84;j++) //wpisywanie 84 bajtów bitmapy
    WriteData(*buffer++);
  }
}
```

List. 3. Zmodyfikowana funkcja pozycjonowania na wyświetlaczu

```
void Poz(char x, char y)
{
  WriteCmd(y|0x40);//zerowanie licznika wierszy
  WriteCmd(x|0x80);//zerowanie licznika kolumn
}
```

Fot. 9. Przesunięcie w pionie

le. Wyjaśniałoby to też, dlaczego procedury wyświetlania bitmap dla PCD8544 nie działają w z tym wyświetlaczem (**fot. 8**).

Procedura wyświetlania bitmapy działa podobnie jak zerowanie wyświetlacza. Z tablicy, w której zapisana jest bitmapa są pobierane i kolejno wpisywane do pamięci 504 bajty. Jeżeli liczniki wierszy się inkrementują po zapisaniu 102 bajtów, a nie 84 bajtów jak w PCD8544, to wyświetlanie bitmap w ten sposób nie mogło się udać.

Mając świadomość, że sterownik ma inna organizację można zmodyfikować procedurę tak, aby bitmapy były wyświetlane prawidłowo. Po ustawieniu licznika kolumn i wierszy na wartości zerowe trzeba wpisać pierwsze 84 bajty bitmapy. Potem trzeba wyzerować licznik kolumn, inkrementować licznik wierszy i wpisać kolejne 84 bajty bitmapy, i tak dalej. Zmodyfikowaną procedurę wpisywania bitmap pokazano na **list. 2**, a procedurę pozycjonowania na wyświetlaczu na **list. 3**.

Wróćmy jeszcze do zerowania wyświetlacza. Po wpisaniu 612 bajtów pozostał nie wyzerowany pasek o szerokości 3 pikseli. Biorąc pod uwagę organizację pamięci taki pasek był dość niepokojący, bo wpisanie każdego bajtu zajmuje pasek o szerokości 8 pikseli, co dla ekranu o wysokości 48 pikseli daje równo 6 pasków (wierszy) i nie ma tutaj miejsca na wiersz o innej szerokości niż 8 pikseli. Po próbach z wyświetlaniem tekstu okazało się ,że rzeczywiście w pionie jest przesunięcie o 3 piksele pomiędzy informacją na ekranie , a zawartością pamięci sterownika (**fot. 9**). Wyglądało to na kolejny problem programowy, ale tym razem o wiele trudniejszy do pokonania.

Aby upewnić się, że wyświetlacz nie jest uszkodzony, podłączyłem go do telefonu Nokia 3310. Zgodnie z oczekiwaniami współpraca z telefonem przebiegała bez zarzutu. Było oczywiste, że sterownik wyświetlacza potrzebuje uzupełnienia o komendę lub komendy w trakcie inicjalizacji. Przeprowa-

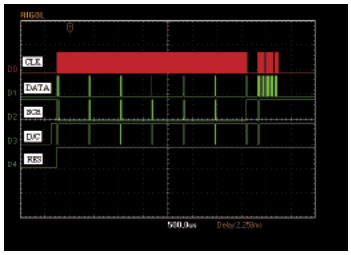
Tab. 2. Znaczenie bitów komend

Bit	0	1
PD	Układ aktywny	Tryb obniżonego poboru mocy
V	Adresowanie poziome	Adresowanie pionowe
H	Podstawowy zestaw instrukcji	Rozszerzony zestaw instrukcji
D E 0 0 0 1 1 0 1 1	Wyświetlacz wygaszony Tryb normalny Wszystkie segmenty zapalone Wyświetlanie w inwersji	
TC1 TC0 0 1 0 1 1	VLCD współczynnik temperaturowy 0 VLCD współczynnik temperaturowy 1 VLCD współczynnik temperaturowy 2 (standardowy) VLCD współczynnik temperaturowy 3	

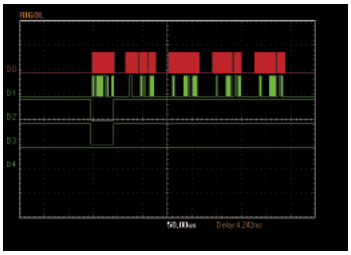
Fot. 10. Zestaw do odczytywania komend wysyłanych do sterownika wyświetlacza

dzane próby wykazały, że prawidłowo działają wszystkie komendy z zestawy komend PCD8544, oprócz prawidłowego pozycjonowania w pionie pierwszej linijki. Ponadto sterownik nie chciał wejść w tryb adresowania pionowego.

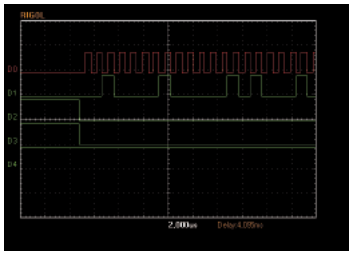
Ponieważ nie wiadomo jaki sterownik zastosowano w wyświetlaczu, to aby nie szukać przysłowiowej igły w stogu siana pozostał tylko jeden sposób trącający szpiegostwem przemysłowym. Do pól kontaktowych telefonu przylutowałem kable zakończone zaciskanym złączem IDC10. Pomiędzy płytkę wyświetlacza, a kabel od telefonu została włączyłem przejściówkę, do której podłączyłem kable kanałów analizatora stanów logicznych LH1116 oscyloskopu cyfrowego Rigol DS1052D (**fot. 10**)



Rys. 11a. Ekran analizatora stanów logicznych: proces inicjalizacji z zerowaniem pamięci



Rys. 11b. Ekran analizatora stanów logicznych: fragment wysyłania komend inicjalizacyjnych (bez zerowania)



Rys. 11c. Ekran analizatora stanów logicznych: szczegóły wysyłania komendy

Do analizatora podłączono wszystkie linie sterujące wyświetlaczem. Na **rys. 11** pokazano zrzuty ekranu oscyloskopu w trakcie analizy sygnałów wysyłanych

Na rys. 11a widać, że zerowanie pamięci jest przeprowadzane przed jej inicjowaniem, wykonywane jest w 6 krokach i przebiega podobnie, jak zapis bitmapy z list. 2. Zamiast wpisywać do sterownika 612 bajtów równych 0 można wykonać zerowanie w wierszach po 84 bajty, tak jak to pokazano na list. 4. Takie zerowanie będzie działało także w przypadku, gdy sterownik wyświetlacza będzie miał jeszcze inną rozdzielczość, niż 102 piksele w poziomie.

Analiza sygnałów odczytanych przez analizator dała w końcu odpowiedź na pytanie o prawidłową inicjalizację sterownika. Telefon wysyła do sterownika wyświetlacza następującą sekwencję komend:

0x21 – komendy rozszerzone, adresowanie poziome

0x05 – współczynnik temperaturowy lub nieznana komenda

0x14 – komenda ustawienia współczynnika multipleksowania (BIAS)

0xBF – komenda ustawiania kontrastu (napięcia zasilania matrycy)

0x20; – zestaw komend podstawowych

0x0C – konfiguracja wyświetlacza (tryb normalny).

Jedyną różnicą pomiędzy inicjalizacją PCD8544 stosowaną przeze mnie a inicjalizacją wysłaną przez telefon była inna komenda współczynnika temperaturowego. Przy wysłaniu komendy 0x06 wyświetlacz zachowywał się jak na fot. 8, ale po wysłaniu 0x05 przesunięcie zniknęło. Podejrzewam, że w badanym sterowniku wyświetlacza

Fot. 12. Wyświetlanie bitmapy

Fot. 13. Wyświetlanie tekstu

List. 4. Zmodyfikowana funkcja zerowania pamięci

```
void ClrDisp (void)
{
  char j,k;
  for(k=0;k<6;k++){
    Poz(0,k);//pierwsza linijka
    for(j=0;j<84;j++) //wpisywanie
84 bajtów zerowych
    WriteData(0);
   }
}
```

komenda o kodzie 0x05 ma inne znaczenie, niż ustawienie współczynnika temperaturowego.

Na **fot. 12** i **fot. 13** pokazano wyświetlanie bitmapy i tekstu przy prawidłowo zainicjalizowanym wyświetlaczu. Samą funkcję inicjalizacji pokazano na **list. 5**. Wyświetlacz

List. 5. Zmodyfikowana funkcja inicjalizująca wyświetlacz

```
//inicjalizacja sterownika LCD
void InitDisNok(void)
{
int i;
  WriteCmd(0x21);//komendy rosrzerzone
  WriteCmd(0x05);//komenda
  WriteCmd(0xc8);//Ustawienie Vop
  WriteCmd(0x06);//korekcja temperatury dla PCD8544
  WriteCmd(0x14);//współczynnik multipleksowania
  WriteCmd(0x20);//komendy standardowe - adresowanie poziome
  WriteCmd(0x01);
  WriteCmd(0x0c);// tyb wyświetlania stndard mode
  WriteCmd(0x40);//zerowanie licznika wierszy
  WriteCmd(0x80);//zerowanie licznika kolumn
  ClrDisp();//zerowanie pamięci
  //for(i=0;i<612;i++)//612
  //WriteData(0x00);//zerowanie pamieci RAM wyswietlacza
}
```

zainicjalizowany zgodnie z list. 5 pracuje poprawnie z zarówno jeśli wyposażony jest w sterownik PCD8544, jak i ten tajemniczy, badany odpowiednik.

Podsumowanie

Przykład tego wyświetlacza pokazuje, że nawet powszechnie znane dane urządzeń stosowanych w technice profesjonalnej nie zawsze są kompletne. Takie elementy jak wyświetlacze od telefonów są oferowane tylko jako podzespoły przeznaczone dla serwisów. Wykorzystanie ich w innym celu może powodować problemy, takie jak tutaj opisywane. Na szczęście, w tym przypadku, udało się je wszystkie rozwiązać.

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