W ofercie AVT jest dostępna płytka ewaluacyjna umożliwiająca przyłączenie wyświetlacza opisywanego w artykule. Numer kitu AVT-5563 www.sklep.avt.pl

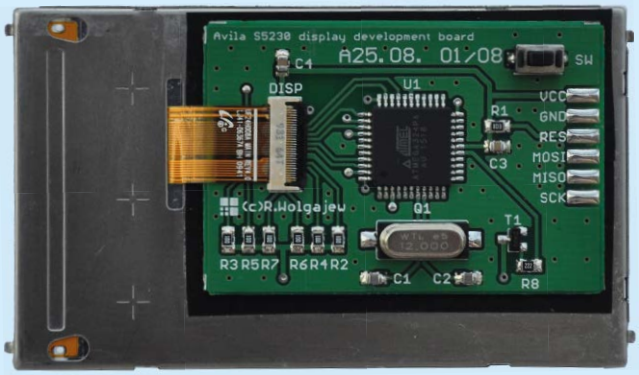
Obsługa kolorowego wyświetlacza TFT z telefonu Samsung GT-S5230 Avila (1)

Wyświetlacze z telefonów komórkowych są bardzo chętnie stosowane z jednego powodu – masowa produkcja jest przyczyną atrakcyjnej ceny. Na łamach „Elektroniki Praktycznej" wielokrotnie prezentowaliśmy drivery programowe i sprzętowe do wyświetlaczy z telefonów marki Nokia i Siemens. Teraz przyszedł czas na wyświetlacz z telefonu marki Samsung, który może przydać się w wielu zastosowaniach.

W trakcie swojej długoletniej przygody z elektroniką wielokrotnie konstruowałem urządzenia wyposażone w wyświetlacze, od tych najprostszych, 7-segmentowych LED, po najbardziej wyszukane, jak choćby kolorowe OLED czy TFT. Każdy wyświetlacz znacznie poszerza możliwości w zakresie interakcji urządzenia z użytkownikiem, dając przy okazji spore pole do popisu programiście czy grafikowi. Muszę przyznać, że najbardziej lubię wyświetlacze TFT, ponieważ pozwalają one na popuszczenie wodzy fantazji w kwestii projektowania graficznego interfejsu użytkownika. Niestety, przy użyciu tych podzespołów pewien problem zawsze stanowiła wysoka cena.

Przed podobnym dylematem stanąłem niedawno, gdy przystępując do zdefiniowania założeń konstrukcyjnych jednego ze swoich projektów, zacząłem poszukiwania taniego, niewielkiego, łatwego w obsłudze wyświetlacza graficznego TFT. Zupełnie przypadkowo natknąłem się przy tym na komponent firmy Samsung w oszałamiająco niskiej cenie 20 złotych brutto! Zapytacie – czy to możliwe? Przecież w tej cenie możemy kupić, co najwyżej, najprostszy i najbardziej popularny model zwykłego, alfanumerycznego wyświetlacza LCD o organizacji 2×16 znaków, a nie nowoczesny wyświetlacz TFT o dużej rozdzielczości. A jednak… Cała tajemnica niskiej ceny wynika z faktu, iż jest to wyświetlacz z telefonu komórkowego, a więc element produkowany masowo, zapewne w milionach sztuk! Mowa o 3" wyświetlaczu TFT o rozdzielczości 240×400 pikseli (155 pikseli na cal) stosowanym w telefonie Samsung GT-S5230 Avila.

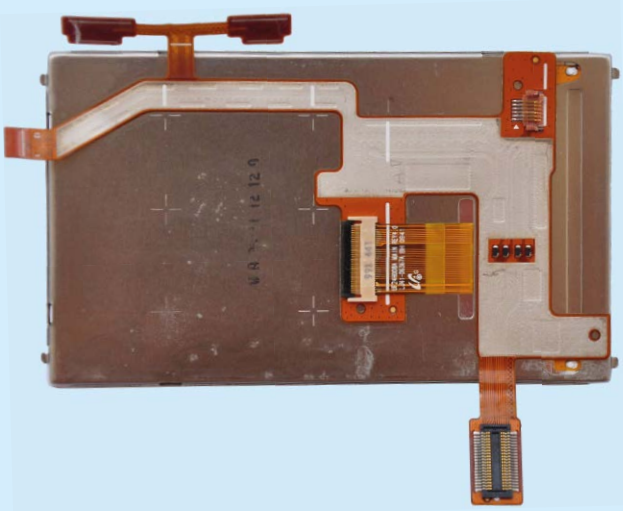
Wyświetlacz wyposażono w zintegrowany sterownik ekranu S6D04D1 (również produkcji Samsunga) pozwalający na wyświetlanie obrazów o 16- i 18-bitowej liczbie kolorów. Wyświetlacz taki na popularnych portalach aukcyjnych można kupić w cenie nawet poniżej 20 zł,



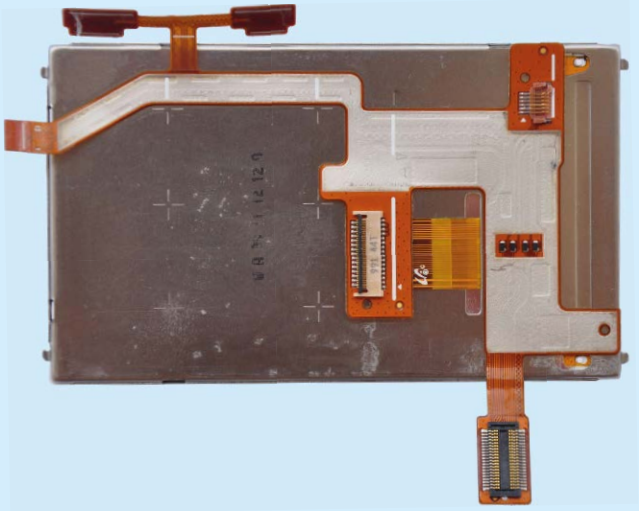
natomiast za granicą jeszcze taniej. Moduł, o którym mowa, wyposażono w 31-pinową taśmę podłączeniową przeznaczoną do wsunięcia w dwustronne złącze ZIF.

W związku z zastosowaniem specjalnego złącza ZIF nasuwa się pytanie, gdzie można je kupić? Po pierwsze, złącze takie możemy kupić w cenie około 5 zł na tych samych portalach aukcyjnych. Drugie rozwiązanie jest zaskakujące, ale ja zdecydowałem się właśnie na nie. Zwyczajnie, w cenie poniżej (!) 20 złotych, kupiłem cały telefon, rozkręciłem i wyciągnąłem z niego sam wyświetlacz, specjalną, giętką taśmę podłączeniową (była do niego przyklejona od spodu), na której znajdowało się złącze ZIF oraz jeszcze kilka innych elementów, które z pewnością przydadzą mi się w przyszłości (kamerka, akumulator). Przy okazji miałem pewność, że wymontowany wyświetlacz jest oryginalnym produktem Samsunga. Oczywiście, tego typu rozwiązanie wymusza na nas wylutowanie złącza ZIF z oryginalnej taśmy telefonu, ale nie jest to zadanie zbyt trudne, zwłaszcza biorąc pod uwagę fakt, że musimy to złącze przylutować ponownie na naszej płytce ewaluacyjnej. Możecie w tym miejscu stwierdzić, że jest to swojego

Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: ftp://ep.com.pl, user: 66532, pass: 8mjjeaa



Fotografia 1. Widok wyświetlacza GT-S5230 (od tyłu) z przyklejoną do niego giętką taśmą podłączeniową i zamkniętym gniazdem ZIF



Fotografia 2. Widok wyświetlacza GT-S5230 (od tyłu) z przyklejoną do niego giętką taśmą podłączeniową i otwartym gniazdem ZIF (i wysuniętą taśmą)

rodzaju marnotrawstwo, ale cóż poradzić? Dzisiaj urządzenia „żyją" krócej niż kiedykolwiek wcześniej. Taka kolej rzeczy...

Wyświetlacz S5230 z przyklejoną do niego giętką taśmą podłączeniową pokazano na **fotografiach 1** i **2** (z zamkniętym i otwartym gniazdem ZIF). W **tabeli 1** pokazano rozkład i opis wyprowadzeń złącza ZIF wyświetlacza GT-S5230.

W wyświetlaczu zastosowano typowe rozwiązanie równoległej magistrali danych standardu 8080 z pomocniczymi sygnałami sterującymi. Możliwości zastosowanego sterownika ekranu S6D04D1 są znacznie większe, gdyż sam układ ma wiele interfejsów danych (szeregowe 3- i 4-przewodowe, równoległe 8-, 9-, 16-, 18-, 24-przewodowe, 24-bitowe RGB oraz MDDI) oraz liczby dostępnych kolorów (16-, 18-, 24-bitowy kolor oraz zredukowany tryb 8 kolorów), jednak w module zastosowanym w telefonie GT-S5230 ograniczono je sprzętowo do wymienionych na wstępie. Warto również wspomnieć, że sterownik S6D04D1 wyposażono w zdecydowanie większą funkcjonalność, aniżeli produkty innych firm przeznaczone do zastosowań ogólnego przeznaczenia. Wynika to z faktu, że jego przeznaczeniem jest implementacja w urządzeniach mobilnych, które stawiają duże wymagania odnośnie do jakości wyświetlanego obrazu.

Jak zwykle, obsługa tego rodzaju peryferii sprowadza się do ich wstępnej konfiguracji, niezbędnej z uwagi na konieczność dopasowania ustawień sterownika ekranu do rodzaju zastosowanego panelu TFT, a następnie do przesyłania pakietów danych stanowiących treść wyświetlanego obrazu. Konfiguracja, o której mowa, możliwa jest dzięki wyposażeniu układu S6D04D1 w szereg rejestrów konfiguracyjnych. Wybór rodzaju przesyłanych danych następuje dzięki wprowadzeniu sygnału RS, którego stan determinuje, czy przesyłany bajt danych jest adresem rejestru sterującego (czyli tzw. rozkazem, RS = 0), czy też wartością (wartościami – dla rejestrów, dla których konieczne jest przesłanie wielu bajtów

Tabela 1. Rozmieszczenie wyprowadzeń złącza ZIF wyświetlacza S5230

Numer	Symbol	Opis
1	GND	Masa zasilania
2	RST	Sygnał zerowania sterownika ekranu
3	NC	Niepodłączone
4	VCC	Zasilanie (2,3 ÷ 3,3 V)
5	VCC	
6	NC	Niepodłączone
7	VGH	Wyjście nieużywane
8	D7	8-bitowa, równoległa magistrala danych
9	D6	
10	D5	
11	D4	
12	D3	
13	D2	
14	D1	
15	D0	
16	RS	Sygnał wyboru rodzaju danych sterownika ekranu (0: rozkaz, 1: dane obrazu lub parametr rozkazu)
17	RD	Sygnał odczytu sterownika ekranu
18	WR	Sygnał zapisu sterownika ekranu
19	CS	Sygnał aktywacji sterownika ekranu
20	FLM	Wyjście nieużywane
21	NC	Niepodłączone
22	LED_A	Anody diod LED podświetlenia ekranu (3 diody z lewej strony ekranu)
23	LED_B	Anody diod LED podświetlenia ekranu (3 diody z prawej strony ekranu)
24	LED_K1	Katody poszczególnych diod LED podświetlenia ekranu
25	LED_K2	
26	LED_K3	
27	LED_K4	
28	LED_K5	
29	LED_K6	
30	NC	Niepodłączone
31	GND	Masa zasilania

Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: **ftp://ep.com.pl**, user: **66532**, pass: **8nnjjeaa**

danych) tegoż rejestru lub „zawartością" ekranu (bajtem danych obrazu, RS = 1). Przejdźmy zatem do szczegółów implementacyjnych.

Listing 1. Plik nagłówkowy modułu obsługi wyświetlacza GT-S5230

```
//Parametry fizyczne - położenie poziome
#define TFT_WIDTH 400
#define TFT_HEIGHT 240
//Konfigurowanie wyprowadzeń. RD=1, CS=0
#define TFT_DATA_PORT PORTA
#define TFT_DATA_DDR DDRA
#define TFT_CTRL_PORT PORTB
#define TFT_CTRL_DDR DDRB
#define RS_PIN PB1  //Wyjście RS: 1->Display data, 0->Command data
#define WR_PIN PB2  //Wyjście WR
#define RD_PIN PB3  //Wyjście RD
#define CS_PIN PB4  //Wyjście CS
#define RES_PIN PB0  //Wyjście RESET
//Makra
#define SET_WR TFT_CTRL_PORT |= (1<<WR_PIN)
#define RESET_WR TFT_CTRL_PORT &= ~(1<<WR_PIN)
#define SET_RS TFT_CTRL_PORT |= (1<<RS_PIN)
#define RESET_RS TFT_CTRL_PORT &= ~(1<<RS_PIN)
#define SET_RES TFT_CTRL_PORT |= (1<<RES_PIN)
#define RESET_RES TFT_CTRL_PORT &= ~(1<<RES_PIN)
//Parametry i komendy układu S6D04D1
#define CMD_SWRESET 0x01  //Zerowanie programowe. Po tej komendzie wymagane opóźnienie 5 lub 120 ms
#define CMD_SLPOUT 0x11  //Wyjście z trybu czuwania
#define CMD_DISPOFF 0x28  //Wyłączenie ekranu
#define CMD_DISPON 0x29  //Załączenie ekranu
#define CMD_RAMWR 0x2C  //Zapisanie pamięci
#define CMD_CASET 0x2A  //Ustawienie adresu kolumny
#define CMD_PASET 0x2B  //Ustawienie adresu strony
#define CMD_COLMOD 0x3A  //Definiowanie formatu danych RGB
    #define RGB_INTERFACE_16BITS (0b101<<4)
    #define RGB_INTERFACE_18BITS (0b110<<4)
    #define RGB_INTERFACE_24BITS (0b111<<4)
    #define CONTROL_INTERFACE_16BITS (0b101)
    #define CONTROL_INTERFACE_18BITS (0b110)
    #define CONTROL_INTERFACE_24BITS (0b111)
#define CMD_MADCTL 0x36  //Definiowanie kierunku skanowania pamięci
    #define PAGE_ORDER_NORMAL (0<<7)
    #define PAGE_ORDER_REVERSED (1<<7)
    #define COLUMN_ORDER_NORMAL (0<<6)
    #define COLUMN_ORDER_REVERSED (1<<6)
    #define PAGE_COLUMN_NORMAL (0<<5)
    #define PAGE_COLUMN_EXCHANGED (1<<5)
    #define VERTICAL_REFRESH_ORDER_NORMAL (0<<4)
    #define VERTICAL_REFRESH_ORDER_REVERSED (1<<4)
    #define COLOR_DIRECTION_RGB (0<<3)
    #define COLOR_DIRECTION_BGR (1<<3)
#define CMD_WRITE_BRIGHTNESS 0x51  //Jasność kontrolowana ręcznie
#define CMD_WRITE_BL_CTRL 0x53  //Komenda BL Control
#define CMD_WRITE_MIE_MODE 0x55  //Komenda MIE Mode
#define CMD_WRITE_MIN_BRIGHTNESS 0x5E  //Jasność minimalna
#define CMD_WRITE_MIE_CTRL1 0xCA  //Zapisanie MIE Control 1
#define CMD_WRITE_BL_CTRL_MODE 0xCB  //Zapisanie BL Control Mode
#define CMD_WRITE_MIE_CTRL2 0xCC  //Zapisanie MIE Control 2
#define CMD_WRITE_MIE_CTRL3 0xCD  //Zapisanie MIE Control 3
#define CMD_DISCTL 0xF2  //Rejestr kontrolny obrazu
#define CMD_PWRCTL 0xF3  //Rejestr kontrolny zasilania
#define CMD_VCMCTL 0xF4  //Rejestr VCOM (napięcie oraz zegar)
#define CMD_SRCCTL 0xF5  //Rejestr Source Output Control Register
    #define CMD_POS_GAMMA_R_CTRL 0xF7  //Korekcja gamma - pozytywny, czerwony
    #define CMD_NEG_GAMMA_R_CTRL 0xF8  //Korekcja gamma - negatywny, czerwony
    #define CMD_POS_GAMMA_G_CTRL 0xF9  //Korekcja gamma - pozytywny, zielony
    #define CMD_NEG_GAMMA_G_CTRL 0xFA  //Korekcja gamma - negatywny, zeilony
    #define CMD_POS_GAMMA_B_CTRL 0xFB  //Korekcja gamma - pozytywny, niebieski
    #define CMD_NEG_GAMMA_B_CTRL 0xFC  //Korekcja gamma - negatywny, niebieski
    #define CMD_GATECTL 0xFD  //Rejestr kontrolny bramki
//Makra pomocnicze
#define SOLID_TEXT 0
#define TRANSPARENT_TEXT 1
#define DOT_POS3 0b11000000
#define DOT_POS2 0b10000000
#define DOT_POS1 0b01000000
#define NO_DOTS 0b00000000
#define DOT_MASK 0b11000000
#define INTERSPACE_MASK 0b00111111
```

Listing 2. Funkcje odpowiedzialne za wysyłanie rozkazów sterujących lub danych do wyświetlacza

```
void writeCommand(uint8_t Command)
{
    RESET_RS;  //komenda
    RESET_WR;
    TFT_DATA_PORT = Command;
    SET_WR;  //TFT czyta dane na zboczu narastającym
    SET_RS;  //RS jest domyślnie 1 dla przyśpieszenia transmisji
}

void writeData(uint8_t Data)
{
    RESET_WR;
    TFT_DATA_PORT = Data;
    SET_WR;  //TFT reads data at the rising edge
}
```

Nie będę w tym miejscu szczegółowo opisywał każdego rejestru konfiguracyjnego sterownika S6D04D1, gdyż po pierwsze, jest to materiał bardzo obszerny (ponieważ

Listing 3. Funkcja inicjalizacyjna sterownika ekranu S6D04D1 wyświetlacza GT-S5230

```
void TFTinit(void)
{
    TFT_DATA_DDR = 0xFF;  //Port jako wyjście, wyzerowany
    TFT_CTRL_PORT |= (1<<WR_PIN)|(1<<RS_PIN)|(1<<RES_PIN)|(1<<RD_PIN);  //Domyślnie CS=0 i RD=1
    TFT_CTRL_DDR |= (1<<WR_PIN)|(1<<RS_PIN)|(1<<RES_PIN)|(1<<RD_PIN)|(1<<CS_PIN);  //Wszystkie linie kontrolne jako wyjściowe
    _delay_ms(100);
    RESET_RES;  _delay_ms(1);  SET_RES;  //Sprzętowy restart wyświetlacza
    _delay_ms(5);
    writeCommand(CMD_SWRESET);  //Programowy restart wyświetlacza
    _delay_ms(5);
//Power Control
    writeCommand(CMD_PWRCTL);
    writeData(0x80);
    writeData(0x00);
    writeData(0x00);
    writeData(0x0b);
    writeData(0x33);
    writeData(0x7f);
    writeData(0x7f);
//VCOM Control
    writeCommand(CMD_VCMCTL);
    writeData(0x6e);
    writeData(0x6e);
    writeData(0x7f);
    writeData(0x7f);
    writeData(0x33);
//Source Output Control
    writeCommand(CMD_SRCCTL);
    writeData(0x12);
    writeData(0x00);
    writeData(0x03);
    writeData(0xf0);
    writeData(0x70);
//Sleep Out
    writeCommand(CMD_SLPOUT);
    _delay_ms(120);
//Memory Data Access Control
    writeCommand(CMD_MADCTL);
    writeData(PAGE_COLUMN_EXCHANGED|COLOR_DIRECTION_BGR);  //Orientacja pozioma 0x48
    writeCommand(CMD_COLMOD);  //Interface Pixel Format
    writeData(RGB_INTERFACE_16BITS|CONTROL_INTERFACE_16BITS);
//RGB565 format 0x55
//Display Control
    writeCommand(CMD_DISCTL);
    writeData(0x14);
    writeData(0x14);
    writeData(0x03);
    writeData(0x03);
    writeData(0x04);
    writeData(0x03);
    writeData(0x04);
    writeData(0x10);
    writeData(0x04);
    writeData(0x14);
    writeData(0x14);
//Gate Control
    writeCommand(CMD_GATECTL);
    writeData(0x22);
    writeData(0x01);
    writeCommand(CMD_WRITE_BRIGHTNESS);  //Jasność sterowana ręcznie
    writeData(0x00);
    writeCommand(CMD_WRITE_MIN_BRIGHTNESS);  //Jasność minimalna
    writeData(0x00);
    writeCommand(CMD_WRITE_MIE_CTRL1);  //MIE Control 1
    writeData(0x80);
    writeData(0x80);
    writeData(0x20);
    writeCommand(CMD_WRITE_BL_CTRL_MODE);  //BL Control Mode
    writeData(0x03);
    writeCommand(CMD_WRITE_MIE_CTRL2);  //MIE Control 2
    writeData(0x20);
    writeData(0x01);
    writeData(0x8F);
    writeCommand(CMD_WRITE_MIE_CTRL3);  //MIE Control 3
    writeData(0x7C);
    writeData(0x01);
//Positive Gamma Control Register - czerowny
    writeCommand(CMD_POS_GAMMA_R_CTRL);
    writeData(0x00);
    writeData(0x23);
    writeData(0x15);
    writeData(0x15);
    writeData(0x1C);
    writeData(0x19);
    writeData(0x18);
    writeData(0x1E);
    writeData(0x24);
    writeData(0x25);
    writeData(0x25);
    writeData(0x20);
    writeData(0x10);
    writeData(0x22);
    writeData(0x21);
//Negative Gamma Control Register - czerwony
    writeCommand(CMD_NEG_GAMMA_R_CTRL);
```

Listing 3. cd.

```
    writeData(0x19);
    writeData(0x00);
    writeData(0x15);
    writeData(0x15);
    writeData(0x1C);
    writeData(0x1F);
    writeData(0x1E);
    writeData(0x24);
    writeData(0x1E);
    writeData(0x1F);
    writeData(0x25);
    writeData(0x20);
    writeData(0x10);
    writeData(0x22);
    writeData(0x21);
//Positive Gamma Control Register - zielony
    writeCommand(CMD_POS_GAMMA_G_CTRL);
    writeData(0x06);
    writeData(0x23);
    writeData(0x14);
    writeData(0x14);
    writeData(0x1D);
    writeData(0x1A);
    writeData(0x19);
    writeData(0x1F);
    writeData(0x24);
    writeData(0x26);
    writeData(0x30);
    writeData(0x1E);
    writeData(0x1E);
    writeData(0x22);
    writeData(0x21);
//Negative Gamma Control Register - zielony
    writeCommand(CMD_NEG_GAMMA_G_CTRL);
    writeData(0x19);
    writeData(0x06);
    writeData(0x14);
    writeData(0x14);
    writeData(0x1D);
    writeData(0x20);
    writeData(0x1F);
    writeData(0x25);
    writeData(0x1E);
    writeData(0x20);
    writeData(0x30);
    writeData(0x1E);
    writeData(0x1E);
    writeData(0x22);
    writeData(0x21);
//Positive Gamma Control Register - niebieski
    writeCommand(CMD_POS_GAMMA_B_CTRL);
    writeData(0x2C);
    writeData(0x23);
    writeData(0x20);
    writeData(0x20);
    writeData(0x23);
    writeData(0x2F);
    writeData(0x30);
    writeData(0x39);
    writeData(0x09);
    writeData(0x09);
    writeData(0x18);
    writeData(0x13);
    writeData(0x13);
    writeData(0x22);
    writeData(0x21);
//Negative Gamma Control Register - niebieski
    writeCommand(CMD_NEG_GAMMA_B_CTRL);
    writeData(0x19);
    writeData(0x2C);
    writeData(0x20);
    writeData(0x20);
    writeData(0x23);
    writeData(0x35);
    writeData(0x36);
    writeData(0x3F);
    writeData(0x03);
    writeData(0x03);
    writeData(0x18);
    writeData(0x13);
    writeData(0x13);
    writeData(0x22);
    writeData(0x21);
//Ustawienie adresu kolumny
    writeCommand(CMD_CASET);
    writeData(0x00);
    writeData(0x00);
    writeData((TFT_WIDTH-1)>>8);
    writeData((uint8_t) TFT_WIDTH-1);
//Ustawienie adresu strony
    writeCommand(CMD_PASET);
    writeData(0x00);
    writeData(0x00);
    writeData((TFT_HEIGHT-1)>>8);
    writeData(TFT_HEIGHT-1);
//Włączenie ekranu
    writeCommand(CMD_DISPON);
//Czyszczenie ekranu
    uint32_t bytesToSend = (uint32_t) TFT_WIDTH*TFT_HEIGHT*2
    writeCommand(CMD_RAMWR);  //Rozpoczęcie zapisu pamięci
    while(bytesToSend--) writeData(0x00);  //Kolor czarny
}
```

liczba opcji przyprawia o zawrót głowy) i dostępny w dokumentacji sterownika, a po drugie, nie wszystkie rejestry wymagają ustawienia w konkretnej aplikacji – wystarczą wartości domyślne. Warto jednak wspomnieć, że sterownik wyposażono w szereg sprzętowych rodzajów funkcjonalności, za których pomocą mamy pełną kontrolę nad jakością wyświetlanych obrazów, w tym obrazów ruchomych. Mowa o takich możliwościach, jak:

- Dynamiczne sterowanie podświetleniem ekranu zależne od treści obrazu.
- Dynamiczne sterowanie nasyceniem kolorów.
- Dynamiczna regulacja ostrości ekranu.
- Możliwość regulacji przejść tonalnych elementów obrazu.
- Korekcja gamma, automatyka przyciemniania ekranu.
- Poprawa wyświetlania obrazów ruchomych.
- Programowanie trybów o obniżonym poborze mocy i wiele, wiele innych.

Listing 4. Ustawienie aktywnego obszaru ekranu

```
void TFTsetActiveWindow(uint16_t X1, uint8_t Y1, uint16_t X2, uint8_t Y2)
{
    writeCommand(CMD_CASET);
    writeData(X1 >> 8);
    writeData(X1 & 0xFF);
    writeData(X2 >> 8);
    writeData(X2 & 0xFF);
    writeCommand(CMD_PASET);
    writeData(Y1 >> 8);
    writeData(Y1);
    writeData(Y2 >> 8);
vwriteData(Y2);
}
```

Lisitng 5. Wyświetlanie wypełnionego i „pustego" prostokąta na ekranie wyświetlacza TFT

```
void TFTdrawFilledBox(uint16_t X1, uint8_t Y1, uint16_t X2, uint8_t Y2)
{
    uint32_t bytesToSend = (X2-X1+1)*(Y2-Y1+1);  //Obliczenie liczby pikseli
    //Definiowanie obszaru aktywnego
    TFTsetActiveWindow(X1, Y1, X2, Y2);
    //Rozpoczęcie zapisu pamięci
    writeCommand(CMD_RAMWR);
    while(bytesToSend--) {writeData(Colour>>8); writeData(Colour&0xFF);}  //(2 bajty/piksel)
}

void TFTdrawRectangle(uint16_t X1, uint8_t Y1, uint16_t X2, uint8_t Y2)
{
    TFTdrawFilledBox(X1 , Y1 , X2 , Y1);
    TFTdrawFilledBox(X1 , Y2 , X2 , Y2);
    TFTdrawFilledBox(X1 , Y1 , X1 , Y2);
    TFTdrawFilledBox(X2 , Y1 , X2 , Y2);
}
```

Listing 6. Wyświetlanie obrazu na ekranie wyświetlacza TFT

```
void TFTdrawPicture(uint16_t X1, uint8_t Y1, const uint16_t *Picture)
{
    register uint16_t pixelData, pixelsToSend;
    register uint8_t Width, Height;
    //Odczytanie wielkości obrazu
    pixelData = pgm_read_word(Picture++);
    //Wyznaczenie wielkości obrazka
    Width = pixelData >> 8; Height = pixelData & 0xFF;
    //Obliczenie liczby pikseli do wysłania
    pixelsToSend = Width * Height;
    //Definiowane obszaru aktywnego dla uproszczenia zapisu
    TFTsetActiveWindow(X1, Y1, X1+Width-1, Y1+Height-1);
    //Rozpoczęcie zapisu pamięci
    writeCommand(CMD_RAMWR);
    while(pixelsToSend--)
    {
        pixelData = pgm_read_word(Picture++);
        writeData(pixelData >> 8); writeData(pixelData & 0xFF);
    }
}
```

Lista jest bardzo długa, w związku z czym ciekawskich czytelników odsyłam do wspomnianej dokumentacji producenta.

W tym miejscu przedstawię plik nagłówkowy modułu obsługi wyświetlacza GT-S5230, który został napisany w taki sposób, aby umożliwić łatwą edycję najistotniejszych parametrów bez potrzeby zaglądania do dokumentacji układu. Plik ten pokazano na **listingu 1**. Warto wspomnieć, że ustawienia portów mikrokontrolera zawarte w tym pliku odpowiadają połączeniom na naszej płytce ewaluacyjnej, więc nie jest tutaj wymagana jakakolwiek edycja.

Pora na dwie funkcje „narzędziowe", które pozwolą na wysłanie do panelu TFT rozkazu sterującego lub też danych obrazu (w podobny sposób jak dane obrazu wysyłamy dane towarzyszące rozkazom sterującym). Wspomniane, podstawowe funkcje zamieszczono na **listingu 2**.

Dysponując tymi podstawowymi funkcjami, możemy przystąpić do inicjalizacji modułu TFT, ponieważ zwykle każdy moduł peryferyjny tego typu wymaga zapisania szeregu nastaw w rejestrach konfiguracyjnych sterownika S6D04D1, które zależą od właściwości obsługiwanego panelu TFT i specyfikacji producenta modułu. Stosowną funkcję inicjalizacyjną odpowiednią dla opisywanego wyświetlacza pokazano na **listingu 3**. Co ważne, brak inicjalizacji sterownika ekranu w zasadzie uniemożliwia poprawne funkcjonowanie modułu TFT, gdyż domyślne ustawienia rejestrów sterujących układu S6D04D1 odbiegają od tych wymaganych przez producenta wyświetlacza TFT.

Dalej, na **listingu 4**, pokazano kolejną funkcję narzędziową odpowiedzialną za ustawienie aktywnego obszaru ekranu, w którego ramach jest wykonywany zapis do pamięci ekranu sterownika S6D04D1, a jej zastosowanie upraszcza i przyśpiesza w sposób znaczący manipulacje w zakresie pamięci obrazu. Czas na funkcje odpowiedzialne za rysowanie elementów graficznych, to jest funkcje umożliwiające wyświetlanie prostokąta lub jego obrysu na ekranie wyświetlacza TFT – te funkcje pokazano na **listingu 5**. Korzystają one z dwóch zmiennych globalnych modułu obsługującego wyświetlacz TFT przechowujących bieżący kolor treści ekranu i kolor tła. Specyfikacja tych zmiennych przedstawia się następująco: *static uint16_t Colour, Background*.

Na **listingu 6** zamieszczono funkcję odpowiedzialną za wyświetlanie obrazków na ekranie wyświetlacza. Jednym z argumentów przekazywanych do wspomnianej funkcji jest wskaźnik na tablicę elementów 16-bitowych umieszczoną w pamięci programu (Flash), która zawiera rozmiar reprezentowanego obrazu (pierwszy element: MSB → szerokość, LSB → wysokość) oraz jego treść (pozostałe elementy, przy czym każdy z nich zawiera kolor kolejnego piksela obrazu w formacie RGB565).

ROBERT WOŁGAJEW, EP