

Sterowanie graficznymi wyświetlaczami z 4-bitowym interfejsem równoległym na przykładzie panelu elektroluminescencyjnego firmy Planar



Wiele z dostępnych na rynku czarno-białych wyświetlaczy graficznych jest wyposażonych w 4-bitowy interfejs równoległy. Ponieważ wymagają one aktywnego odświeżania zawartości ekranu, do ich poprawnej obsługi zwykle jest potrzebny oddzielny sterownik. Utrudnia to stosowanie tego typu paneli, zwłaszcza we własnoręcznie budowanych urządzeniach. W artykule pokażę jak w prosty sposób obsługiwać takie wyświetlacze za pomocą mikrokontrolerów z rdzeniem ARM na przykładzie układu AT91SAM7S256 oraz panelu elektroluminescencyjnego firmy Planar typu EL.320.240.36-HB o rozdzielczości 320 na 240 pikseli.

INTERFEJS 4-BITOWY...

Podobnie jak połączenie między kartą graficzną a monitorem, czterobitowy interfejs spotykany w wielu graficznych panelach LCD/EL jest przeznaczony wyłącznie do przesyłania danych opisujących kompletne ramki obrazu. Każda zmiana jaką chcemy wprowadzić wymaga ponownego przesłania treści całego ekranu do wyświetlacza – nie ma możliwości wydawania poleceń w rodzaju *wyczyść ekran* lub *narysuj tekst*. Ponadto większość takich paneli do poprawnej pracy musi być odświeżana z określoną częstotliwością. Jak

Materiały i dodatkowe informacje

Dodatkowe materiały są dostępne na płycie dołączonej do numeru.



za chwilę się przekonamy, nie są to wcale ich wady...

Przedstawione w artykule rozwiązanie wymaga:

- zastosowania mikrokontrolera z synchronicznym portem szeregowym (np. SSC lub SPI) i kontrolerem DMA (*Direct Memory Access* – bezpośredni dostęp do pamięci). Kontroler DMA nie jest konieczny, choć jego wykorzystanie pozwala znacznie zmniejszyć obciążenie procesora. Układy spełniające te wymagania to między innymi seria Atmel AT91SAM7 oraz STMicroelectronics STR91x,
- 10 kilobajtów pamięci RAM,
- jednego wejścia PWM,
- dwóch wolnych wyprowadzeń wejścia wyjścia (oprócz portu SPI/SSC).

Sposób transmisji pojedynczej ramki do wyświetlacza przedstawiono na **rys. 1**. Kolejne linie obrazu są

skanowane od lewej do prawej strony, zaś cały obraz jest odświeżany od góry do dołu ekranu. Przy opadającym zboczku zegara VCLK wyświetlacz zatrzymuje dane czterech pikseli o wartościach podawanych przez sygnały VID0..VID3. Stan niski oznacza piksel zgaszony, stan wysoki – zapalony. Sygnał VID3 reprezentuje piksel znajdujący się najbardziej z lewej strony „czwórki”, VID0 – najbardziej z prawej. Ponieważ nasz wyświetlacz ma rozdzielczość w poziomie równą 320, na jedną linię przypada 80 cykli VCLK. Krótki dodatni impuls sygnału HS oznacza koniec przesyłania bieżącej linii, a sygnału VS – koniec przesyłania danych pierwszej (y = 0) linii obrazu.

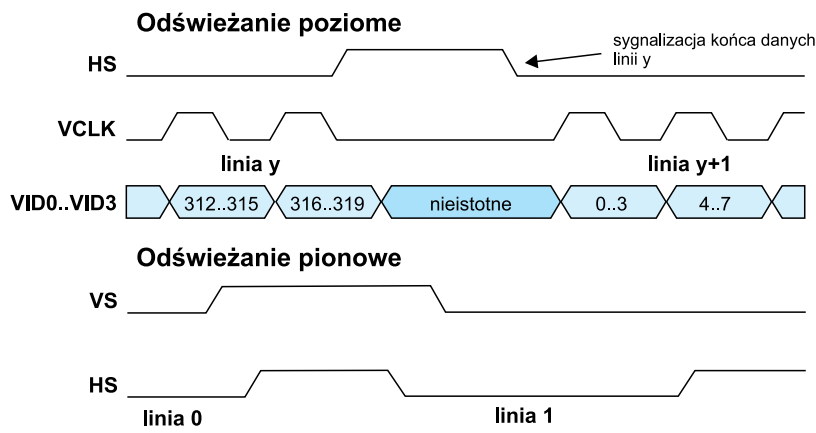
W większości przypadków *timing* sygnałów sterujących musi być dobrany tak, aby uzyskać określoną częstotliwość odświeżania. W przypadku standardowej częstotliwości

(ok. 60 Hz, stosowana w przykładowym sterowniku), przesyłanie jednej ramki trwa około 16 ms, zaś pojedynczej linii – około 69 μ s. Prezentowany w artykule panel ma wbudowany wewnętrzny bufor, który uniezależnia częstotliwość odświeżania ekranu od parametrów czasowych dostarczonych sygnałów sterujących – obraz może być aktualizowany z dowolną częstotliwością z zakresu od 1 do 120 Hz.

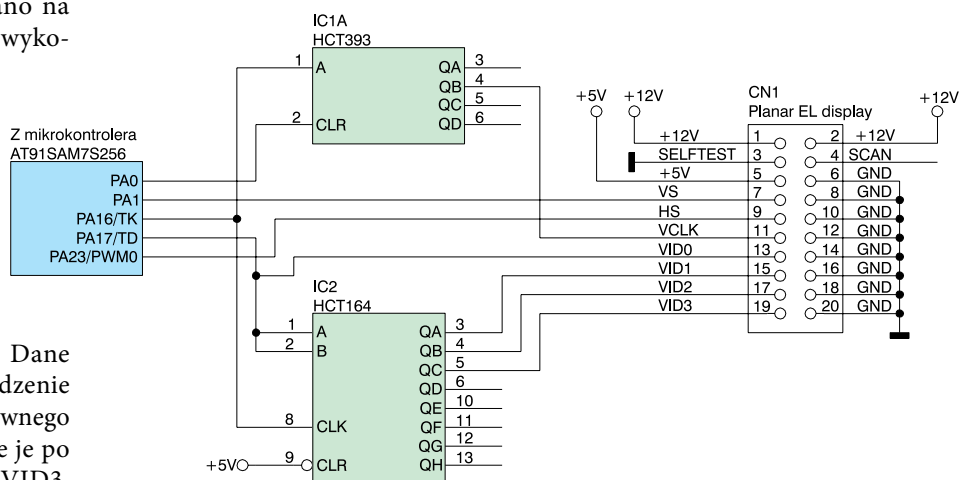
POŁĄCZENIE WYŚWIETLACZA Z MIKROKONTROLEREM

Schemat połączenia wyświetlacza z mikrokontrolerem pokazano na rys. 2. „Serce” sterownika to wykorzystany w dość nietypowy sposób synchroniczny port szeregowy, którego zadaniem jest transmisja danych kolejnych pikseli obrazu. W przykładowym rozwiązaniu wykorzystalem SSC (*Synchronous Serial Controller*) mikrokontrolera AT91SAM7S256. Dane wysłane przez SSC (wyprowadzenie TD) trafiają do rejestru przesuwanego HCT164 (IC2), który grupuje je po 4 piksele na liniach VID0...VID3. W każdym cyklu zegara TK do rejestru jest „wsuwany” 1 bit, zatem co 4 cykle TK na linii TD oraz wyjściach QA, QB i QC układu IC2 (połączonych bezpośrednio z wejściami VID0..VID3 wyświetlacza) znajdują się kolejne 4 piksele odświeżanej linii. Licznik typu HCT393 (IC1A) dzieli sygnał zegarowy interfejsu SSC (TK) przez 4, wytwarzając w ten sposób zegar danych dla wyświetlacza (VCLK). Dzielnik jest zaprojektowany w taki sposób, aby opadające zbocze VCLK wypadało w momencie, gdy na liniach VID0..VID3 znajdują się poprawne dane. Proces ten pokazano na rys. 3.

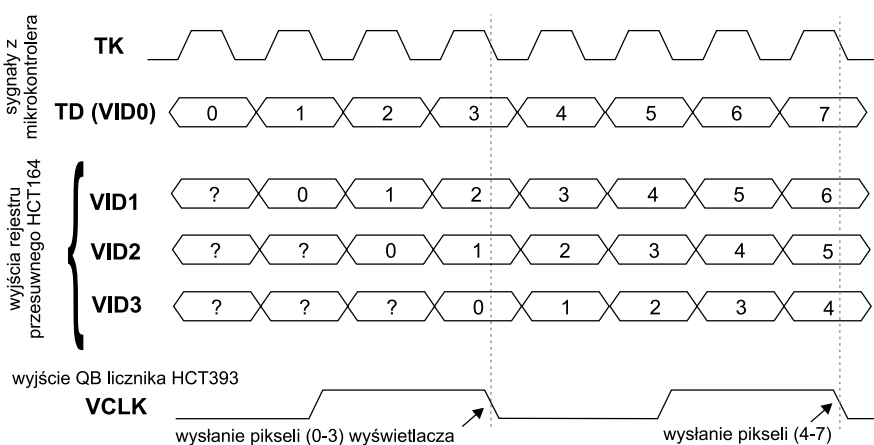
Wyprowadzenie PA0 służy do ustalenia znanego stanu początkowego licznika '393. Pozostałe sygnały sterujące panelem są dołączone bezpośrednio do odpowiednich portów mikrokontrolera. Sygnał synchronizacji poziomej (HS) wytwarza kontroler PWM w postaci krótkich dodatnich impulsów. Odstęp między impulsami HS musi być nieznacznie większy niż czas potrzebny na transmisję poje-



Rys. 1. Transmisja danych obrazu do wyświetlacza



Rys. 2. Połączenie wyświetlacza Planar z mikrokontrolerem AT91SAM7S256



Rys. 3. Konwersja danych z portu SSC na format 4-bitowy

dynczej linii przez SSC. Sygnał VS jest generowany programowo.

STEROWANIE WYŚWIETLACZEM

Aby sterownik mógł działać potrzebne jest jeszcze oprogramowanie, które będzie wytwarzać sygnały syn-

chronizacji (HS i VS) oraz obsługiwać port szeregowy mikrokontrolera. Najistotniejsze fragmenty jego kodu pokazane są na list. 1.

Zacznijmy od tablicy vscreen – bufora ramki (*framebuffer*), przechowującego aktualnie wyświetlany obraz. Zawiera ona wartości pikseli

Funkcje biblioteki graficznej

```
void planar_lcd_init();
```

Inicjalizacja odświeżania wyświetlacza. Należy wywołać przed rysowaniem czegokolwiek.

```
void gfx_putpixel(int x, int y, int p);
```

Zapalenie (p=1) lub zgaszenie (p=0) piksela o współrzędnych (x, y)

```
void gfx_clear(int p);
```

Czyszczenie ekranu. W zależności od wartości parametru p ekran zostanie zgaszony (p=0) lub zapalony (p=1)

```
void gfx_draw_text(font_t *font, int x, int y, const char *str, int p);
```

Rysowanie tekstu *str* czcionką *font*, o lewym górnym rogu w punkcie (x, y). Niezerowa wartość *p* powoduje inwersję koloru tekstu (jasne tło, ciemne znaki). Do biblioteki dołączony jest spory zbiór fontów, ich lista znajduje się w pliku nagłówkowym *fonts.h*.

```
void gfx_draw_rect(int x1, int y1, int w, int h, int p);
```

```
void gfx_draw_box(int x1, int y1, int w, int h, int p);
```

Rysowanie prostokątnej ramki (*rect*) lub wypełnionego prostokąta (*box*) o lewym górnym rogu w punkcie (x1, y1), szerokości *w*, wysokości *h* kolorem *p* (0 – ciemny, 1 – jasny)

```
void gfx_draw_line(int x1, int y1, int x2, int y2, int p);
```

Rysowanie linii od punktu (x1, y1) do (x2, y2) kolorem *p*.

w kolejnych liniach obrazu, przy czym jeden bit odpowiada jednemu pikselowi. Ponieważ bufor jest zbudowany z 32-bitowych liczb typu *int*, pojedyncza linia zajmuje (320 pikseli/32 bity/słowo) = 10 słów, a cały bufor – 240 linii * 10 = 2400 słów (9600 bajtów). Najmłodszy bit w danym słowie odpowiada pikselowi o najmniejszej współrzędnej *x*. Mając współrzędne (x, y) piksela na ekranie, jego po-

łożenie w *vscreen* można obliczyć w następujący sposób:

```
// nr słowa w tablicy vscreen
o = (320 / 32) * y + (x >> 5);
// nr bitu
i = x & 0x1f;
```

Aby zapalić lub zgasić piksel należy ustawić lub wyzerować *i*-ty bit w *o*-tym słowie tablicy *vscreen*:

```
// zapalenie piksela
vscreen[o] |= (1<<i);
// zgaszenie piksela
vscreen[o] &= ~(1<<i);
```

Najistotniejszym elementem programu sterownika jest funkcja – wektor przerwania *horiz_irq()*, wywoływana po wygenerowaniu przez kontroler PWM impulsu synchronizacji poziomej (HS). Jej głównym zadaniem jest wysyłanie kolejnych linii obrazu przez port SSC z wykorzystaniem kontrolera DMA. Polega to na:

– ustawieniu adresu początku danych wysyłanej linii w tablicy *vscreen* w rejestrze *SSC_TPR*:

```
// ustawiamy adres bieżącej linii dla kontrolera DMA interfejsu SSC
*AT91C_SSC_TPR = (unsigned int)
voffset;
//zwiększamy offset, aby wskazywał na następną linię
voffset += 10;
```

– ustawieniu liczby transferów DMA w rejestrze *SSC_TCR*. Ponieważ kontroler SSC zaprogramowany jest do wysyłania ramek o długości 32 bitów, potrzebne jest wykonanie (320 pikseli / 32) = 10 transferów:


```
// liczba transferow = 10
// (320 pikseli / 32 bity)
*AT91C_SSC_TCR = 10;
```

– rozpoczęciu transferu DMA przez ustawienie bitu *TXEN* (8) w rejestrze *SSC_PTCR*:

```
// rozpoczynamy transfer DMA
*AT91C_SSC_PTCR = (1<<8);
```


Dzięki wykorzystaniu DMA, port SSC samoczynnie pobiera kolejne dane do wysłania z pamięci, przez co transmisja praktycznie nie obciąża procesora. Jak wspomniałem wcześniej, zastosowanie DMA nie jest

R E K L A M A



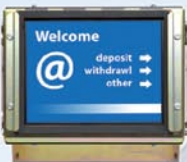

amtek



autoryzowany dystrybutor



PLANAR
www.planar.com

- **Wyświetlacze EL**
 - przekątne obrazu od 3.05" do 10.4"
 - odporne na wibracje i wstrząsy
 - temperatura pracy od -50°C do +85°C
 - technologia ICEBright
- **Wyświetlacze TFT LCD bez obudowy (open frame)**
 - przekątne obrazu od 8" do 19"
 - opcjonalnie ekran dotykowy
 - szyba wandaloodporna

www.amtek.pl

AMTEK spol. s r.o. Sp. z o.o. – oddział w Polsce, ul. Przasnyska 6b / 01-756 Warszawa / tel. 022 866 4140 / fax 022 866 4141 / e-mail amtek@amtek.pl / www.amtek.pl

```

unsigned int vscreen [ 320 / 32 * 240 ]; // Tablica - bufor ekranu.
//Jeden bit odpowiada jednemu pikselowi,
// bufor ma więc rozmiar (320 / 8) * 240 = 9600 bajtow

static volatile int *voffset; // Adres aktualnie wyswietlanej linii
//w buforze ekranu

static volatile int vcnt; // licznik linii

// przerwanie co kazda linie wywoływane po wygenerowaniu przez
//kontroler PWM impulsu synchronizacji poziomej (HSYNC).
//Dla przyspieszenia pracy funkcja jest wykonywana z RAMu
//(atrybut section .fastrun)

static void horiz_irq()
{
    volatile int tmp;

// jesli licznik linii jest >= 0, wysylamy dane linii via SSC,
//VSYNC w stanie niskim.
    if(vcnt>=0)
    {
        if(vcnt == 239)
            *AT91C_PIOA_SODR = (1<<1); // wyswietlamy pierwsza linie - VSYNC = 1
        else
            *AT91C_PIOA_CODR = (1<<1); // wyswietlamy pozostale linie - VSYNC = 0

        *AT91C_SSC_TPR = (unsigned int)voffset; // ustawiamy adres biezacej
        //linii dla kontrolera DMA
        //intefejsu SSC
        *AT91C_SSC_TCR = 10; //liczba transferow = 10 (320 pikseli / 32 bity)
        *AT91C_SSC_PTCR = (1<<8); // rozpoczynamy transfer DMA
        voffset+=10; // zmieniamy adres na nastepna linie

// jesli koniec cyklu, przechodzimy do nast ramki
    } else {
        vcnt=240; // ustawiamy licznik linii
        voffset=vscreen; // .. i jej adres na poczatek ekranu
    }

    vcnt--; // zmniejszamy licznik linii
    tmp=*AT91C_PWMC_ISR; // potwierdzenie obsluzenia przerwania
    // dla kontrolera PWM
}

```

List. 1. Najważniejsze fragmenty oprogramowania sterownika

```

#include „gfx.h”

main()
{
// inicjalizacja wyswietlania :)
    planar_lcd_init();

// czyscimy ekran na czarno
    gfx_clear(0);

// rystujemy rameczke
    gfx_draw_rect(0,0,320,240,1);

// rysujemy tekst
    gfx_draw_text(&font_helv38b, 20, 20, „Hello, world”, 1);

    for(;;);
}

```

List. 2. Prosty przykład wykorzystania biblioteki dołączonej do sterownika

konieczne, ale wówczas przerwanie musiałyby być wywoływane dużo częściej, przez co odświeżanie zużywałoby znacznie więcej mocy obliczeniowej. Przy starannej optymalizacji kodu z obsługą wyświetlacza powinny poradzić sobie nawet mikrokontrolery 8-bitowe (np. AVR).

Funkcja `horiz_irq()` zajmuje się również wytwarzaniem sygnału synchronizacji pionowej (VS) – generuje impuls na linii VS pomiędzy pierwszą a drugą linią obrazu.

Odświeżanie wyświetlacza należy zainicjalizować wywołując na początku programu funkcję `planar_lcd_init()`. Wykonuje ona następujące czynności:

- włącza peryferia mikrokontrolera (PWM, SSC, SCU),
- ustawia kierunki portów wejścia-wyjścia łączących mikrokontroler z wyświetlaczem,
- zeruje licznik HCT393,
- inicjalizuje generator PWM sygnału HS oraz wektor przerwania PWM.

Od tej pory zawartość bufora `vscreen` jest na bieżąco przesyłana do wyświetlacza. Podobnie jak w przypadku karty graficznej w komputerze PC, każda zmiana w `vscreen` jest prawie natychmiast widoczna na ekranie. Dzięki temu obsługa wyświetlacza jest szybka i wygodna.

OPROGRAMOWANIE

Do kodu sterownika dołączyłem prostą bibliotekę umożliwiającą rysowanie podstawowych kształtów geometrycznych oraz tekstu. Funkcje, które zawiera są omówione w oddzielnej ramce. Oprogramowanie zostało opracowane pod systemem Linux z wykorzystaniem GCC w wersji 4.1.0 i powinno bez problemu kompilować się także pod WinARM/Yagarto. Źródła można znaleźć na płycie CD dołączonej do EP+ lub na stronie <http://wlostowski.ep.com.pl>. Oprócz biblioteki dostępne są także kody źródłowe przykładowych programów:

- *hello*, czyli najprostszy możliwy przykład – inicjalizacja odświeżania i wyświetlenie dużego napisu *Hello, world*. Kod źródłowy można zobaczyć na **list. 2**.

- *starwars* – demonstracja nieco bardziej skomplikowanej grafiki.

Implementacja funkcji rysujących nie jest bardzo szybka, optymalizując kod można je przyspieszyć nawet o kilkadziesiąt procent.

NA ZAKOŃCZENIE...

Sterownik powinien (teoretycznie) działać z każdym wyświetlaczem wyposażonym w 4-bitowy interfejs przedstawiony w artykule. Metoda, którą opisałem po drobnych modyfikacjach może być wykorzystana także do generowania czarno-białego obrazu na monitorze VGA (640x480), matrycy LCD lub ekranie telewizora. Dla zainteresowanych mogą udostępnić dokumentację sterownika monitora VGA.

Tomasz Włostowski, EP
tomasz.wlostowski@ep.com.pl

Dodatkowe informacje:
 Amtek Sp. z o.o.
www.amtek.pl
 tel. 22 866 41 40