Obsługa wyświetlacza TFT

Sterowanie wyświetlaczy kolorowych z kontrolerem SSD1963 w Bascom AVR

Szybki rozwój elektroniki użytkowej spowodował, iż nawet proste urządzenia są wyposażane w imponujące, kolorowe wyświetlacze TFT będące elementem interfejsu użytkownika, nierzadko zintegrowane z panelem dotykowym. Nic w tym dziwnego, ponieważ poprawiają one w znaczący sposób komunikację z użytkownikiem czyniąc samo urządzenie bardziej atrakcyjnym.

Wymownym i dobrym przykładem tej sytuacji jest widziana ostatnio przeze mnie lampa błyskowa wyposażona w taki interfejs. Czy był niezbędny? Sprawa dyskusyjna, ale interfejs był bardzo atrakcyjny. Oczywiście, co należy szczególnie podkreślić, popularyzacji tego rodzaju rozwiązań sprzyja przede wszystkim niska cena wyświetlaczy oraz ich dostępność. Przykładem takiego nowoczesnego sterownika wyświetlacza TFT jest układ SSD1963 produkowany przez firmę Solomon Systech Limited, a praktycznego jego zastosowania moduł wyświetlacza TFT produkowany przez firmę Winstar, oznaczony symbolem WF57ETIBCDC#000 mający następujące parametry:

- 8-bitowa magistrala danych,
- rozdzielczość 320×240 pikseli,
- przekątna ekranu 5,7 cala,
- 18-bitowa głębia kolorów (tryb 6-6-6),
- zintegrowny, rezystancyjny panel dotykowy.

Dlaczego wybrałem ten konkretny typ panela TFT? Dlatego, że ma sporą przekątną przy niewielkiej (jak na jego rozmiary) rozdzielczości i wystarczającej liczbie kolorów, co czyni zeń optymalne rozwiązanie do zastosowania w systemie mikroprocesorowym z mikrokontrolerem AVR. Oczywiście, przy tej przekątnej i rozdzielczości należy się liczyć, że rozmiar pojedynczego piksela obrazu będzie dość spory, co przekłada się na pewien dyskomfort przy obserwacji wyświetlanych obrazów z niewielkiej odległości, lecz jest to kompromis pomiędzy wymaganiami systemu mikroprocesorowego, a parametrami modułu wyświetlacza TFT. Dla bardziej rozbudowanych systemów należałoby zastosować wyświetlacz o większej rozdzielczości. Przejdźmy, zatem do opisu samego sterownika SSD1963, ponieważ jest on podstawowym elementem takiego modułu.

Sterownik SSD1963, aplikacja wyświetlacza

Sterownik ten charakteryzuje się następującymi, wybranymi parametrami funkcjonalnymi:

- wbudowana pamięć o pojemności 1215 kB umożliwiająca obsługę wyświetlaczy TFT o maksymalnej rozdzielczości 864×480 pikseli i 24-bitowej głębi kolorów,
- obsługa 18- i 24-bitowej magistrali TFT,
- obsługa 8-bitowego interfejsu danych RGB,
- sprzętowa funkcja obrotu obrazu o 0, 90, 180 i 270 stopni,
- sprzętowa funkcja lustrzanego odbicia obrazu w poziomie i w pionie (mirroring),
- sprzętowa funkcja definicji aktywnego obszaru pamięci ekranu z automatyczną inkrementacją (windowing),
- sprzętowa funkcja regulacji parametrów jasności, kontrastu i saturacji ekranu,
- sprzętowa funkcja automatycznej kontroli podświetlenia ekranu TFT (modulacja PWM),
- obsługa interfejsów danych o szerokości 8, 9, 16, 18, i 24-bitów,
- wbudowany oscylator PLL.

Wspomniany kontroler charakteryzuje się sporą elastycznością funkcjonalną a jak zobaczymy później, także dużą prostotą obsługi. Jak wspomniano, układ SSD1963 obsługuje magistrale danych o różnych szerokościach (ustawiane sprzętowo poprzez dedykowane wyprowadzenia chipsetu) a konkretne rozwiązanie zależne jest od producenta modułu wyświetlacza TFT (w wypadku opisywanego wyświetlacza firmy Winstar jest to 8 bitów). Jak łatwo domyślić się, w zależności od wybranej szerokości magistrali danych, jak i wybranej, obsługiwanej liczby kolorów, istnieją różne formaty zapisu informacji o kolorach. Inaczej mówiąc, wspomniane założenia zmieniają organizację pamięci obrazu, co jest ważne z punktu widzenia operacji zapisu/odczytu. Wszystkie dostępne opcje przedstawiono na **rysunku 1** zaczerpniętym z dokumentacji modułu.

Znając już organizację pamięci video naszego sterownika jak i krótką charakterystykę chipsetu SSD1963, warto przedstawić szczegóły konkretnego rozwiązania układowego tj. parametry wyświetlacza firmy Winstar. Wyposażono go w 20-stykowe złącze ZIF o rastrze 1 mm, którego rozkład wyprowadzeń wraz z opisem ich znaczenia dla systemu mikroprocesorowego przedstawiono w **tabeli 1**.

Dodatkowo, moduł ma 2-przewodowe złącze do dołączenia napięcia zasilającego wbudowanego podświetlacza. Typowe parametry źródła napięcia zasilającego dla modułu podświetlacza to U_{LED}=10 V, I_{LED}=140 mA. Przykładową aplikację układową z wykorzystaniem wyświetlacza WF57ETIBCDC#000 zintegrowanego z panelem dotykowym oraz mikrokontrolera Atmega32L przedstawiono na **rysunku 2**.

Wybór mikrokontrolera w tym konkretnym wypadku nie jest elementem krytycz-

Interface	Cycle	D[23]	D[22]	D[21]	D[20]	D[19]	D[18]	D[17]	D[16]	D[15]	D[14]	D[13]	D[12]	D[11]	D[10]	D[9]	D[8]	D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]
24 bits	1st	R7	R6	R5	R4	R3	R2	R1	R0	G7	G6	G5	G4	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0
18 bits	1st							R5	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B5	B4	B3	B2	B1	B0
16 bits (565 format)	1st									R5	R4	R3	R2	R1	G5	G4	G3	G2	G1	G0	B5	B4	B3	B2	B1
16 bits	1st									R5	R4	R3	R2	R1	R0	X	X	G5	G4	G3	G2	G1	G0	X	X
	2nd									B5	B4	B3	B2	B1	B0	X	X	R5	R4	R3	R2	R1	R0	X	X
	3rd									G5	G4	G3	G2	G1	G0	X	X	B5	B4	B3	B2	B1	B0	X	X
9 bits	1st																R5	R4	R3	R2	R1	R0	G5	G4	G3
	2nd																G2	G1	G0	B5	B4	B3	B2	B1	B0
8 bits	1st																	R5	R4	R3	R2	R1	R0	X	X
	2nd																	G5	G4	G3	G2	G1	G0	X	X
	3rd																	B5	B4	B3	B2	B1	B0	X	X

X: Don't Care

Rysunek 1. Organizacja pamięci obrazu

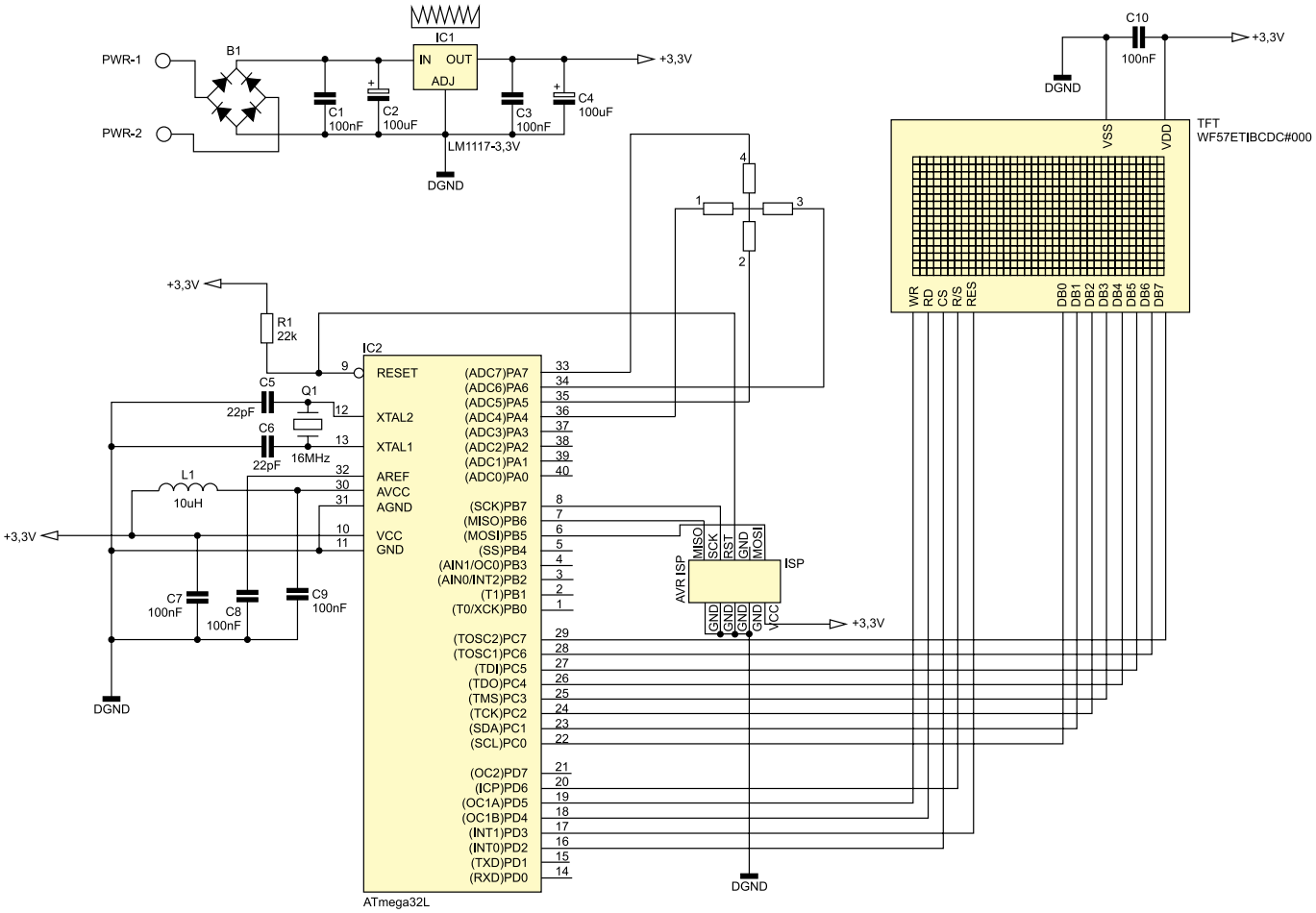
Tabela 1. Rozkład wyprowadzeń wyświetlacza WF57ETIBCDC#000 (w zależności od modelu i producenta mogą występować różnice)

Nr wypr.	Symbol	Opis
1	Vss	Masa zasilania
2	Vdd	Napięcie zasilania (3.3V). Prąd zasilania ok. 300mA.
3	NC	
4	RS	Sygnał wyboru rodzaju danych: Polecenie sterujące/Dana
5	WR	Sygnał żądania operacji zapisu
6	RD	Sygnał żądania operacji odczytu
7	DB0	Magistrala danych
8	DB1	
9	DB2	
10	DB3	
11	DB4	
12	DB5	
13	DB6	
14	DB7	
15	CS	Sygnał wyboru układu (aktywacji chipsetu)
16	RST	Sygnał Reset
17	NC	
18	NC	
19	NC	
20	NC	

nym, jednak należy mieć na uwadze potrzebę zastosowania mikrokontrolera o możliwości taktowania sygnałem o minimalnej częstotliwości rzędu 16...20 MHz, a to z uwagi na dużą liczbę informacji niezbędnych do przesłania, jak i oczekiwaną szybkość działania. Tym bardziej, jeśli zamierzamy wyświetlać obrazy o sporej rozdzielczości czy tez wielkości (tutaj przyda się na pewno spora pamięć RAM niezbędnej np. do implementacji obsługi kart SD). Sytuację dodatkowo poprawiłoby zastosowanie modelu wyświetlacza (jak i odpowiedniego podłączenia) o 16-, 18- czy 24-bitowej organizacji magistrali danych oraz odpowiedniego do tej organizacji mikrokontrolera.

Oprogramowanie

Skupimy się na stronie programowej obsługi wyświetlacza, która jest nieoczekiwanie nieskomplikowana. Sterowanie pracą wyświetlacza opiera się wyłącznie na wysyłaniu komend, za pomocą których ustawiamy pewne wstępne parametry konfiguracyjne korzystając z dedykowanych rejestrów konfiguracyjnych (do których zapis następuje na skutek wykonania komend sterujących), a następnie zapisujemy (lub też odczytujemy) dane do/z pamięci obrazu powodując ich natychmiastowe wyświetlenie. Interpretacja rodzaju danych, które mają być odebrane przez chipset SSD1963 jest zdeterminowana poziomem wyprowadzenia RS. Jest to typowe rozwiązanie w sterowaniu modułami wyświetlaczy różnego typu. Poziom niski na tym wyprowadzeniu decyduje o tym, iż przesyłana wartość zinterpretowana zostanie jako komenda sterująca, zaś wysoki, że będzie zinterpretowana jako parametr polecenia lub dane dla pamięci obrazu (w zależności od przesłanego wcześniej polecenia). W związku z powyższą organizacją procedur sterujących, można wyróżnić dwa rodzaje sekwencji operacji zapisu do układu SSD1963 (operacje odczytu przebiegają



Rysunek 2. Schemat przykładowej aplikacji modułu Winstar

analogicznie, lecz nie będą tematem naszych rozważań):

– sekwencja zapisu komendy sterującej (zapisu danych do rejestrów konfiguracyjnych), którą pokazano na **rysunku 3**,
– sekwencja zapisu danych do pamięci obrazu (**rysunek 4**).

Analizują rys. 3 i rys. 4 możemy wyodrębnić dwa stany pracy magistrali sterującej dla operacji zapisu, zdeterminowane stanem wyprowadzenia RS: wysłanie komendy sterującej oraz wysłanie bajtu danych pamięci obrazu. Na **listingu 1** zamieszczono procedury służące do zapisu danych (dla pokazanej wcześniej aplikacji). W treści samych procedur użyto aliasów dla właściwych nazw portów (**listing 2**), co dodatkowo ułatwia zrozumienie treści programu. Przyjęto także, iż port sterujący sygnałem RD (sygnał żądania operacji odczytu) pozostaje zawsze na poziomie wysokim, gdyż w naszej aplikacji interesuje nas wyłącznie zapis do sterownika SSD1963.

Prezentowane procedury napisano przy użyciu języka Bascom Basic nieprzypadkowo. Pomijając fakt, że można go wykorzystać zupełnie bezpośrednio w aplikacjach rozwijanych z jego użyciem to, w mojej opinii, każdy język o strukturze podobnej do Basica czy też Pascala jest na tyle czytelny, iż łatwo przenieść daną implementację na inną, dowolną platformę programową (np. AVR-GCC).

Przy implementacji procedur sterujących należy, co oczywiste, ustawić odpowiednie kierunki portów sterujących pracą naszego wyświetlacza (w tym przypadku, wszystkie porty ustawiane są jako porty wyjściowe z poziomem wysokim po włączeniu zasilania).

W tym miejscu posiadamy już niezbędną wiedzę w zakresie sterowania pracą naszego wyświetlacza lecz do „pełni szczęścia" potrzebny jest nam szczegółowy opis rejestrów konfiguracyjnych, za pomocą których możemy zainicjować moduł poprzez ustawienie niezbędnych parametrów konfiguracyjnych sterownika SSD1963 oraz możemy zainicjować zapis do pamięci obrazu. Co ważne, w wypadku niektórych rejestrów konfiguracyjnych (zwłaszcza tych, dotyczących zależności czasowych dla pracy sterownika ekranu TFT) producent zaleca wpisanie do nich wartości domyślnych, dostępnych w stosownej dokumentacji bez potrzeby ich zmiany. Pomijając jednak znaczenie tych ustawień, należy podkreślić, iż dla poprawnej inicjalizacji sterownika jest niezbędne określenie szeregu danych związanych bezpośrednio z organizacją zastosowanego wyświetlacza TFT oraz organizacją pamięci obrazu i typem magistrali danych.

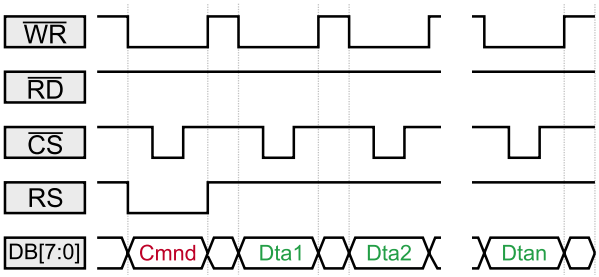
Listę wybranych komend sterujących wraz z opisem ich znaczenia z punktu widzenia aplikacji docelowej zaprezentowano w ramce. Należy podkreślić, że jest to wyłącznie skrót pełnej listy komend, ale jest on wystarczający do przeprowadzenia ważniejszych operacji konfiguracyjnych.

W tym momencie posiadamy już wszystkie, niezbędne informacje potrzebne do napisania własnej, podstawowej biblioteki obsługi wyświetlacza ze sterownikiem SSD1963. Na listingu 3 zaprezentowano procedurę inicjalizującą pracę modułu wyświetlacza TFT. Jej wywołanie powinno nastąpić przed użyciem modułu.

Teraz zaprezentuję kilka podstawowych procedur odpowiedzialnych za wyświetlanie różnych elementów na ekranie wyświetlacza TFT. Procedury te zakładają deklarację pewnych zmiennych bibliotecznych, których potrzeba stosowania wynika z założeń i konstrukcji biblioteki (jest to jedno z wielu możliwych rozwiązań). Deklaracje tych zmiennych i opis ich znaczenie umieszczono na **listingu 4**.

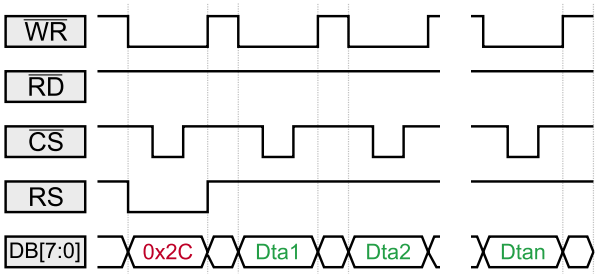
Po wywołaniu procedury inicjalizacyjnej wyświetlacz jest gotowy do odbioru danych (komend sterujących czy danych dla pamięci obrazu).

Na **listingu 5** zamieszczono procedurę definiującą aktywny obszar ekranu dla operacji zapisu. Jest to bardzo użyteczna procedura, której użycie w upraszcza a zarazem znacznie przyspiesza wyświetlanie obrazów, których rozmiar jest inny aniżeli całkowity, fizyczny rozmiar ekranu. Kolejne dwie procedury, zaprezentowane na **listingu 6**, ustalają bieżący kolor piksela obrazu oraz tła korzystając ze składowych RGB będących argumentem ich wywoła-



Cmnd – komenda sterująca
Dta1 – parametr 1 (jeśli występuje)
Dta2 – parametr 2 (jeśli występuje)

Rysunek 3. Przebiegi sygnałów sterujących w przypadku sekwencji wysyłania komendy sterującej do sterownika SSD1963



0x2C – polecenie inicjujące zapis do pamięci video sterownika SSD1963
Dta1 – dana dla bieżącego adresu pamięci video. Bieżący adres pamięci video określają rejestry 0x2A i 0x2B. Postinkrementacja adresu następuje w chwili przesłania kompletu danych przypadających na jeden piksel obrazu (w zależności o wybranego interfejsu danych i liczby kolorów będą to 1, 2 lub 3 bajty)
Dta2 – kolejna dana
Dtan – ostatnia dana

Rysunek 4. Przebiegi sygnałów sterujących w przypadku sekwencji zapisu danych po pamięci obrazu sterownika SSD1963

Listing 1. Procedura wysyłająca polecenia sterujące

```
'zapis komendy
Sub Write_command(byval Addr As Byte)
  Reset Rs_pin
  Reset Wr_pin
  Reset Cs_pin
  Data_port = Addr
  Set Cs_pin
  Set Wr_pin
  'Zakładamy, iż pin Rs_pin (Dane/Komendy) w stanie spoczynku
  'jest ustawiony (1) wskazując na wysyłanie danych - skróci to
  'czas wykonania procedury Write_data
  Set Rs_pin
End Sub

'zapis bajtu do pamięci obrazu
Sub Write_data(byval Dta As Byte)
  Reset Wr_pin
  Reset Cs_pin
  Data_port = Dta
  Set Cs_pin
  Set Wr_pin
End Sub
```

Listing 2. Definicje aliasów nazw portów

```
Data_port Alias Portc
Wr_pin Alias Portd.5
Rd_pin Alias Portd.4
Cs_pin Alias Portd.2
Rs_pin Alias Portd.6
Reset_pin Alias Portd.3
```

Listing 3. Procedura inicjalizująca pracę modułu TFT

```
Sub Initialize_ssd1963
  Reset Reset_pin                       ,Zerowanie sprzętowe sterownika SSD1963
  Waitms 10
  Set Reset_pin
  Waitms 1
  Call Write_command(&H01)              ,Zerowanie programowe sterownika SSD1963
  Waitms 10
  Call Write_command(&He0)              'Uruchomienie oscylatora PLL
  Call Write_data(&H01)
  Waitms 10
  Call Write_command(&He0)              'Aktywacja oscylatora PLL
  Call Write_data(&H03)
  Call Write_command(&Hb0)              'Ustawienie trybu pracy sterownika SSD1963
  Call Write_data(&H0c)                 'Głębia kolorów: 18-bitów
  Call Write_data(&H00)                 'Tryb pracy sterownika: TFT/ Hsync+Vsync+DE
  Call Write_data(&H01)                 ,Rozmiar ekranu w poziomie (320-1), starszy bajt
  Call Write_data(&H3f)                 ,Rozmiar ekranu w poziomie (320-1), młodszy bajt
  Call Write_data(&H00)                 ,Rozmiar ekranu w pionie (240-1), starszy bajt
  Call Write_data(&Hef)                 ,Rozmiar ekranu w pionie (240-1), młodszy bajt
  Call Write_data(&H00)                 ,Porządek danych o kolorach: RGB
  Call Write_command(&Hf0)              ,Rodzaj interfejsu wejściowego: 8 bitów
  Call Write_data(&H00)
  Call Write_command(&H3a)              'Format informacji o kolorach: 18 bitów na pixel (format 6-6-6)
  Call Write_data(&H60)
  Call Write_command(&He2)              ,Ustawienie częstotliwości oscylatora PLL: 113.33 MHz
  Call Write_data(&H22)
  Call Write_data(&H03)
  Call Write_data(&H04)
  Call Write_command(&He6)              ,Ustawienie częstotliwości sterowania pixelami obrazu PCLK: 10 MHz
  Call Write_data(&H01)
  Call Write_data(&H69)
  Call Write_data(&H6c)
  Call Write_command(&Hb4)              ,Ustawienie parametrów czasowych synchronizacji poziomej HBP
  Call Write_data(&H01)                 ,Wartości domyślne z dokumentacji producenta panela TFT
  Call Write_data(&Hb8)
  Call Write_data(&H00)
  Call Write_data(&H44)
  Call Write_data(&H0f)
  Call Write_data(&H00)
  Call Write_data(&H00)
  Call Write_data(&H00)
  Call Write_command(&Hb6)              ,Ustawienie parametrów czasowych synchronizacji pionowej VBP
  Call Write_data(&H01)                 ,Wartości domyślne z dokumentacji producenta panela TFT
  Call Write_data(&H08)
  Call Write_data(&H00)
  Call Write_data(&H13)
  Call Write_data(&H07)
  Call Write_data(&H00)
  Call Write_data(&H00)
  Call Write_command(&H2a)              ,Ustawia współrzędne ekranu w poziomie dostępne z punktu widzenia operacji
                                        ,zapisu/odczytu z automatyczną inkrementacją
  Call Write_data(&H00)  'X1=0 (starszy bajt)
  Call Write_data(&H00)  'X1=0 (młodszy bajt)
  Call Write_data(&H01)  'X2=319 (starszy bajt)
  Call Write_data(&H3f)  'X2=319 (młodszy bajt)
  Call Write_command(&H2b)              ,Ustawia współrzędne ekranu w pionie dostępne z punktu widzenia operacji
                                        ,zapisu/odczytu z automatyczną inkrementacją
  Call Write_data(&H00)  'Y1=0 (starszy bajt)
  Call Write_data(&H00)  'Y1=0 (młodszy bajt)
  Call Write_data(&H00)  'Y2=239 (starszy bajt)
  Call Write_data(&Hef)  'Y2=239 (młodszy bajt)
  Call Write_command(&H29)              'Włączenie ekranu TFT
                                        'Czyszczenie ekranu sterownika TFT
  Call Write_command(&H2c)              'Rozkaz: Write Memory Start
                                        'Rejestry R16:R17:R18 stanowić będą licznik pętli (320x240x3 bajty= 230400
                                        'pętli)
  ser r16
  ldi r17, &H84
  ldi r18, &H03

  Set Rs_pin
  Poczatek:
    Reset Wr_pin
    Reset Cs_pin
    Data_port = 0
    Set Cs_pin
    Set Wr_pin
                                        'Dekrementacja licznika pętli
    subi r16, 1
    sbci r17, 0
    sbci r18, 0
  brne  Poczatek
End Sub
```

nia. Tak naprawdę procedury te wpływają na wartości odpowiednich zmiennych bibliotecznych.

Na **listingu 7** pokazano procedurę wyświetlającą piksel obrazu w kolorze ustalonym aktualnymi wartościami zmiennych jego koloru Jak da się zauważyć, ciała wspomnianych procedur zawierają kod procedury *Write_data*, jednak celowo unika się wywoływania tej ostatniej dla skrócenia czasu zapisu danych do pamięci ekranu. Ponadto, kod wspomnianych procedur jest uproszczony (nie zawiera poleceń sterujących) z uwagi na fakt, iż są one przeznaczone wyłącznie do wykorzystania w pozostałych procedurach bibliotecznych.

Na **listingu 8** umieszczono dwie procedury – rysujące poziomą i pionową linię, korzystają-

Listing 4. Wykaz zmiennych bibliotecznych

```
Dim Red As Byte , Green As Byte , Blue As Byte  ,Składowe RGB aktualnego koloru Piela
Dim Bk_red As Byte , Bk_green As Byte , Bk_blue As Byte  ,Składowe RGB aktualnego koloru tła
Dim Picture_pointer As Word  ,Zmienna - wskaźnik do bieżącej sekcji danych fontów
```

Listing 5. Procedura definiująca aktywny obszar ekranu TFT

```
Sub Set_active_window(byval X1 As Word , Byval Y1 As Byte , Byval X2 As Word , Byval Y2 As Byte)
 Local Msb_lsb As Byte
  Call Write_command(&H2a)                          ,Ustawia współrzędne ekranu w poziomie dostępne z punktu widzenia
                                                    ,operacji zapisu/odczytu z automatyczną inkrementacją
  Msb_lsb = High(x1)
  Call Write_data(msb_lsb)  'X1 starszy bajt
  Msb_lsb = X1
  Call Write_data(msb_lsb)  'X1 młodszy bajt
  Msb_lsb = High(x2)
  Call Write_data(msb_lsb)  'X2 starszy bajt
  Msb_lsb = X2
  Call Write_data(msb_lsb)  'X2 młodszy bajt
  Call Write_command(&H2b)                          ,Ustawia współrzędne ekranu w pionie dostępne z punktu widzenia
                                                    ,operacji zapisu/odczytu z automatyczną inkrementacją
  Call Write_data(0)  'Y1 starszy bajt
  Call Write_data(y1)  'Y1 młodszy bajt
  Call Write_data(0)  'Y2 starszy bajt
  Call Write_data(y2)  'Y2 młodszy bajt
End Sub
```

Listing 6. Procedury ustalające bieżący kolor tła oraz piksela obrazu

```
'Listing procedury ustalającej bieżący kolor pixela obrazu
Sub Set_color(byval R As Byte , Byval G As Byte , Byval B As Byte)
  Red = R
  Green = G
  Blue = B
End Sub

'Listing procedury bieżący kolor tła
Sub Set_bk_color(byval R As Byte , Byval G As Byte , Byval B As Byte)
  Bk_red = R
  Bk_green = G
  Bk_blue = B
End Sub
```

Listing 7. Procedury wyświetlające piksel w ustalonych kolorach

```
'Procedura wyświetlająca piksel
obrazu w kolorze ustalonym
aktualnymi wartościami zmiennych
koloru piksela
Sub Draw_pixel
  Set Rs_pin
  Reset Wr_pin
  Reset Cs_pin
  Data_port = Red
  Set Cs_pin
  Set Wr_pin
  Reset Wr_pin
  Reset Cs_pin
  Data_port = Green
  Set Cs_pin
  Set Wr_pin
  Reset Wr_pin
  Reset Cs_pin
  Data_port = Blue
  Set Cs_pin
  Set Wr_pin
End Sub

'Procedura wyświetlająca piksel
obrazu w kolorze ustalonym
aktualnymi wartościami zmiennych
koloru tła
Sub Draw_bk_pixel
  Set Rs_pin
  Reset Wr_pin
  Reset Cs_pin
  Data_port = Bk_red
  Set Cs_pin
  Set Wr_pin
  Reset Wr_pin
  Reset Cs_pin
  Data_port = Bk_green
  Set Cs_pin
  Set Wr_pin
  Reset Wr_pin
  Reset Cs_pin
  Data_port = Bk_blue
  Set Cs_pin
  Set Wr_pin
End Sub
```

Listing 8. Procedury rysujące linie poziomą i pionową

```
'Listing procedury wyświetlającej poziomą linię
Sub Draw_hline(byval X As Word , Byval Y As Byte , Byval Length As Word )
  Local X2 As Word
  X2 = X + Length
  Decr X2
  Call Set_active_window(x , Y , X2 , Y)
  Call Write_command(&H2c)  'Rozkaz: Write Memory Start
  For X2 = 1 To Length
   Call Draw_pixel  ,Rysujemy kolejne pixele w kolorze określonym zmiennymi
globalnymi Red, Green i Blue
  Next X2
End Sub

'Listing procedury wyświetlającej pionową linię
Sub Draw_vline(byval X As Word , Byval Y As Byte , Byval Length As Byte )
 Local Y2 As Byte
  Y2 = Y + Length
  Decr Y2
  Call Set_active_window(x , Y , X , Y2 )
  Call Write_command(&H2c)  'Rozkaz: Write Memory Start
  For Y2 = 1 To Length
   Call Draw_pixel  ,Rysujemy kolejne pixele w kolorze określonym zmiennymi
globalnymi Red, Green i Blue
  Next Y2
End Sub
```

ce z mechanizmu pozwalającego zdefiniować aktywny obszar ekranu dla operacji zapisu. Dla linii pionowej wspomniany mechanizm jest szczególnie użyteczny, gdyż nie wymaga się każdorazowego ustawiania wskaźnika pamięci ekranu (dla każdej linii wyświetlacza).

Niejako na „deser" przedstawię procedurę pozwalającą na wyświetlanie napisów na ekranie TFT (**listing 9**). Sterownik SSD1963 nie jest wyposażony w generator znaków. Jest to przykładowe, rzekłbym typowe i zarazem edukacyjne rozwiązanie bazujące na definicji czcionki dostarczanej przez producenta kompilatora Bascom. Mowa o czcionce „Color8x8.font", której definicja nie jest niczym innym, jak zbiorem bajtów poszczególnych linii każdego znaku uszeregowanych według ich kodów ASCII. Szczegółów dotyczących struktury pliku typu *.font (w tym wypadku koniecznie w wersji dla wyświetlaczy kolorowych) należy szukać w pliku pomocy programu Bascom. Plik z definicją takiej czcionki należy dołączyć do naszego programu za pomocą instrukcji *$include „color8x8.font"*.

Podsumowanie

Na tym zakończę opis podstawowych procedur związanych z obsługą naszego panela mając jednocześnie nadzieję, iż ten krótki kurs zachęci Czytelników do własnych eksperymentów z peryferium tego typu. Tym bardziej, iż ich cena osiągnęła poziom, który jeszcze do niedawna zajmowały zwykłe, graficzne, monochromatyczne wyświetlacze LCD. Ja ze swojej strony mogę Was tylko gorąco zachęcić do stosowania wyświetlaczy kolorowych zwłaszcza, że możliwości, które ma sterownik SSD1963 są ogromne i trudno byłoby opisać je tutaj szczegółowo. Oczywiście, co należy mieć na względzie, takie moduły wymagają jednocześnie znacznie więcej od systemu mikroprocesorowego, jednak przy obecnych, bardzo niskich cenach dobrze wyposażonych mikrokontrolerów, nie powinno to stanowić żadnego problemu w aplikacji.

Chciałbym podziękować Panu Sławomirowi Szweda z firmy Unisystem za dostarczenie modułów wyświetlaczy firmy Winstar. Szczególne podziękowania chciałbym także złożyć na ręce Pana Aik Hong Tok z firmy Solomon Systech Limited za udzielone wsparcie techniczne.

Robert Wołgajew, EP

Skrócony wykaz komend sterujących pracą modułu wyświetlacza z SSD1963

Komenda: **0x01**
Parametry: Brak
Inicjuje programowe zerowanie sterownika SSD1963 przywracając wszystkim rejestrom sterującym ich wartości domyślne. Po wysłaniu komendy, program aplikacji użytkownika powinien odczekać 5 ms w celu kontynuowania transmisji.
Komenda: **0x28**
Parametry: Brak
Wyłączenie ekranu TFT, przy czym zawartość bufora ramki (treść obrazu) pozostaje niezmieniona.
Komenda: **0x29**
Parametry: Brak
Włączenie ekranu TFT powodując wyświetlenie obrazu znajdującego się w buforze ramki.
Komenda: **0x2A**
Parametry: 4

	D7	D6	D5	D4	D3	D2	D1	D0
Parametr 1	SC15	SC14	SC13	SC12	SC11	SC10	SC9	SC8
Parametr 2	SC7	SC6	SC5	SC4	SC3	SC2	SC1	SC0
Parametr 3	EC15	EC14	EC13	EC12	EC11	EC10	EC9	EC8
Parametr 4	EC7	EC6	EC5	EC4	EC3	EC2	EC1	EC0

Określenie adresu startowego (SC15...0) i końcowego (EC15...0) aktywnego obszaru pamięci ekranu w zakresie współrzędnej X dla operacji zapisu czy też odczytu. Wartość SC15...0 nie może być mniejsza od wartości EC15...0.
Komenda: **0x2B**
Parametry: 4

	D7	D6	D5	D4	D3	D2	D1	D0
Parametr 1	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8
Parametr 2	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0
Parametr 3	EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8
Parametr 4	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0

Określenie adresu startowego (SP15...0) i końcowego (EP15...0) aktywnego obszaru pamięci ekranu w zakresie współrzędnej Y dla operacji zapisu czy też odczytu pamięci. Wartość SP15...0 nie może być mniejsza od wartości EP15...0.
Komendy **0x2A** i **0x2B** pozwalają zdefiniować aktywny obszar pamięci ekranu przeznaczony dla operacji zapisu czy odczytu. W przypadku, gdy w danej chwili chcemy mieć dostępną całą widoczną powierzchnię ekranu to dla wyświetlacza o organizacji 320×240 pikseli będą to odpowiednio zakresy 0…319 dla parametrów SC15…0 i EC15…0 oraz 0…239 dla parametrów SP15…0 i EP15…0. Wprowadzenie możliwości definiowana aktywnego obszaru ekranu znacznie upraszcza zapis do pamięci ekranu, gdyż kolejne dane (dla wspomnianego aktywnego obszaru danych) wyświetlane będą od lewej do prawej i od góry do dołu (wyłącznie w przypadku standardowego trybu adresowania) w ramach aktualnej definicji wykorzystując mechanizm autoinkrementacji adresów bufora ramki bez potrzeby „pilnowania" aktualnego położenia wskaźnika bieżącego adresu. Bazując na tej właściwości możemy dla przykładu w bardzo prosty sposób wyświetlić pionową linię, skalowane czcionki czy też niewielki obrazek na ekranie wyświetlacza bez każdorazowego przemieszczania się po pamięci ekranu (rozumianego jako całości).
Komenda: **0x2C**
Parametry: Brak
Przygotowanie sterownika SSD1963 do operacji zapisu do pamięci ekranu. Kolejno przesyłane bajty danych obrazu umieszczane będą począwszy od lokalizacji określonej wartościami rejestrów **0x2A** i **0x2B**. Dla normalnego trybu adresowania dane pierwszego pixela obrazu umieszczone zostaną w buforze ramki w miejscu określonym parametrami rejestrów SC15…0 i SP15…0. Każdy, kolejny piksel obrazu umieszczany będzie jako kolejny w bieżącej linii obrazu aż do osiągnięcia przez licznik kolumn wartości EC15...0. W tym miejscu licznik linii obrazu zostanie inkrementowany i nastąpi zapis do kolejnej linii. Zapis obrazu kontynuowany będzie do momentu aż licznik linii osiągnie wartość EP15...0. Jeśli przesłana przez mikrokontroler liczba danych obrazu przekracza rozmiar zdefiniowanego, aktywnego obszaru obrazu ((EC15...0 – SC15...0 +1)* (EP15...0 – SP15...0 +1)), dane te zostaną pominięte przez sterownik SSD1963. Inną kolejność zapisywania danych do bufora obrazu zapewniają specjalne ustawienia rejestru konfiguracyjnego **0x36**, dzięki którym możliwe jest np. obracanie obrazu czy też „odbijanie" obrazu w poziomie czy też pionie.
Komenda: **0x3A**
Parametry: 1

	D7	D6	D5	D4	D3	D2	D1	D0
Parametr	0	A6	A5	A4	0	0	0	0

Ustalenie formatu danych informacji o kolorach pikseli obrazu czyli de facto, liczby kolorów możliwych do wyświetlenia. Dostępne są następujące wartości dla bitów **A6**...**A4** [bity/piksel]:
– 000: wartość zarezerwowana, domyślna,
– 001: 3,
– 010: 8,
– 011: 12,
– 100: wartość zarezerwowana,
– 101: 16,
– 110: 18,
– 111: 24.

Uwaga: prezentowany w artykule wyświetlacz TFT wyposażony w 8-bitową magistralę danych obsługuje wyłącznie 18-bitowy format danych (6-6-6) informacji o kolorach pikseli.
Komenda: **0x3C**
Parametry: Brak
Podobna do komendy **0x2C** z tą różnicą, iż dane pierwszego piksela obrazu zostaną umieszczone w buforze ramki w miejscu określonym bieżącą wartością licznika kolumn i linii (ustaloną po ostatnim wywołaniu komendy **0x2C** lub **0x3C**).
Komenda: **0xB0**
Parametry: 7

	D7	D6	D5	D4	D3	D2	D1	D0
Parametr 1	0	0	A5	A4	A3	A2	A1	A0
Parametr 2	B7	B6	B5	0	0	0	0	0
Parametr 3	0	0	0	0	0	HPS10	HPS9	HPS8
Parametr 4	HPS7	HPS6	HPS5	HPS4	HPS3	HPS2	HPS1	HPS0
Parametr 5	0	0	0	0	0	VPS10	VPS9	VPS8
Parametr 6	VPS7	VPS6	VPS5	VPS4	VPS3	VPS2	VPS1	VPS0
Parametr 7	0	0	G5	G4	G3	G2	G1	G0

Pozwala na określenie trybu pracy sterownika SSD1963 i właściwości zastosowanego ekranu TFT (składających się na moduł wyświetlacza). Znaczenie poszczególnych bitów konfiguracyjnych kolejnych parametrów wywołania przedstawia się następująco:
– Bit **A5** determinuje szerokość wewnętrznej magistrali danych: 0: panel 18-bitowy (wartość domyślna), 1: panel 24-bitowy.
– Bit **A4** decyduje o załączeniu funkcji zwiększających głębię palety kolorów: 0: funkcje nieaktywne (wartość domyślna), 1: załączona funkcja FRC lub Dithering.
– Bit **A3** decyduje o rodzaju aktywnej funkcji zwiększającej głębię palety kolorów: 0: aktywna funkcja Dithering (wartość domyślna), 1: aktywna funkcja FRC.
– Bitu **A2** decyduje o aktywnym zboczu sygnału zegarowego sterującego pracą bufora ramki: 0: dane zatrzaskiwane są przy rosnącym zboczu sygnału PCLK (wartość domyślna), 1: dane zatrzaskiwane są przy opadającym zboczu sygnału PCLK.
– Bity **A1** i **A0** decydują o polaryzacjach impulsów sygnałów synchronizacji, odpowiednio: poziomej i pionowej (domyślnie „0").
– Bit **B7** determinuje tryb pracy: 0: tryb pracy Hsync+Vsync+DE (wartość domyślna), 1: Tryb TTL.
– Bity **B6**...**B5** określają tryb pracy: 00, 01: tryb TFT (wartość domyślna), 10: Serial RGB, 11: Serial RGB+dummy.
– Bity **HPS10...0** określają rzeczywisty rozmiar panela dotykowego w poziomie (dla naszego panela będzie to wartość 319, gdyż numery pikseli są liczone są w tej osi od 0 do 319), zaś bity **VPS10...0** określają rzeczywisty rozmiar panela dotykowego w pionie (dla naszego panela będzie to wartość 239, gdyż numery pikseli są liczone w tej osi od 0 do 239).
– Bity **G5**...**G3** określają kolejność informacji o składowych R, G, B poszczególnych pikseli obrazu dla parzystych linii obrazu, zaś bity **G2**... **G0** dla linii nieparzystych: 000: RGB (wartość domyślna), 001: RBG, 010: GRB, 011: GBR, 100: BRG, 101: BGR, 110: wartość zarezerwowana, 111: wartość zarezerwowana.
Komenda: **0xE0**
Parametry: 1

	D7	D6	D5	D4	D3	D2	D1	D0
Parametr	0	0	0	0	0	0	A1	A0

Uruchamia wbudowany generator PLL. Do czasu jego uruchomienia wyświetlacz TFT (jego sterownik SSD1963) pracuje z dołączonym rezonatorem kwarcowym o częstotliwości 10 MHz (typowo). Z tego wynika także, iż do czasu uruchomienia oscylatora PLL maksymalna prędkość transmisji do sterownika SSD1963 nie może przekraczać połowy częstotliwości rezonatora kwarcowego, którym jest taktowany (w tym przypadku 5 MHz). Znaczenie poszczególnych bitów parametru towarzyszącego komendzie **0xE0** jest następujące:
– Bitu **A1** decyduje o źródle sygnału taktującego sterownik SSD1963: 0: sterownik SSD1963 taktowany rezonatorem kwarcowym (wartość domyślna), 1: sterownik SSD1963 taktowany oscylatorem PLL.
– Bit **A0** włącza i wyłącza oscylator: 0: oscylator nieaktywny (wartość domyślna), 1: oscylator włączony.

Uwaga: procedura aktywacji oscylatora PLL wymaga pewnej kolejności transmitowanych danych. Przykładowa sekwencja powinna wyglądać następująco:
1. Wysłanie komendy sterującej **0xE0**,
2. Wysłanie danej **0x01**,
3. Odczekanie 100 μs na ustabilizowanie się oscylatora PLL,
4. Wysłanie danej **0x03**.

Komenda: **0xE2**
Parametry: 3

	D7	D6	D5	D4	D3	D2	D1	D0
Parametr 1	N7	N6	N5	N4	N3	N2	N1	N0
Parametr 2	0	0	0	0	M3	M2	M1	M0
Parametr 3	0	0	0	0	0	C2	0	0

Definiuje częstotliwość sygnału zegarowego na wyjściu oscylatora PLL określając parametry wbudowanego weń podzielnika i mnożnika częstotliwości. Częstotliwość tę obliczamy według wzoru:
$f_{PLL} = (f_{OSC} * N7…0)/M3…0$, gdzie
f_{OSC} – częstotliwość dołączonego rezonatora kwarcowego,
N7…0 – wartość bitów N7…0,
M3…0 – wartość bitów M3…0.
Bit **C2** determinuje wykorzystanie parametrów podzielnika i mnożnika częstotliwości według poniższej specyfikacji: 0: oscylator PLL ignoruje wartości N7...0 i M3...0 (wartość domyślna), 1: oscylator PLL korzysta z wartości N7...0 i M3...0.

Komenda: **0xE6**
Parametry: 3

	D7	D6	D5	D4	D3	D2	D1	D0
Parametr 1	0	0	0	0	FPR19	FPR18	FPR17	FPR16
Parametr 2	FPR15	FPR14	FPR13	FPR12	FPR11	FPR10	FPR9	FPR8
Parametr 3	FPR7	FPR6	FPR5	FPR4	FPR3	FPR2	FPR1	FPR0

Definiuje częstotliwość sygnału zegarowego odpowiedzialnego za odświeżanie pikseli obrazu (PCLK). Częstotliwość tę obliczamy według wzoru:
$f_{PCLK} = (f_{PLL} * FPR19…0)/2^{20}$, gdzie:
f_{PLL} – częstotliwość oscylatora PLL,
FPR19…0 – wartość bitów FPR19…0.
Dla przykładu, przy częstotliwości oscylatora f_{PLL}=120 MHz i oczekiwanej częstotliwości sygnału zegarowego f_{PCLK}= 5,3 MHz, przykładowa sekwencja sterująca powinna wyglądać następująco:
1. Wysłanie komendy sterującej **0xE6**,
2. Wysłanie danej **0x00**,
3. Wysłanie danej **0xB4**,
4. Wysłanie danej **0xE7**.

Komenda: **0xF0**
Parametry: 1

	D7	D6	D5	D4	D3	D2	D1	D0
Parametr	0	0	0	0	0	A2	A1	A0

Ustala szerokość magistrali danych sterownika SSD1963 przeznaczonej do współpracy z systemem mikroprocesorowym (hostem):
– 000: 8-bitowa magistrala danych,
– 001: 12-bitowa magistrala danych,
– 010: 16-bitowa magistrala danych,
– 011: 16-bitowa magistrala danych (format informacji o składowych RGB typu 5-6-5),
– 100: 18-bitowa magistrala danych,
– 101: 24-bitowa magistrala danych (wartość domyślna),
– 110: 9-bitowa magistrala danych,
– 111: wartość zarezerwowana.

Listing 9. Procedura wyświetlająca tekst na zadanych współrzędnych

```
Sub Draw_text(byval X1 As Word , Byval Y1 As Byte , Text As String)
  Local Char_index As Byte
  Local Char As String * 1
  Local Index As Byte
  Local A As Word
  Local B As Byte
  Local Offset As Word
  For Char_index = 1 To Len(tekst)                    ,Pętla po kolejnych znakach zmiennej Text
     Char = Mid(text , Char_index , 1)                ,Pobieramy kolejny znak ze zmiennej tekstowej Text
                                                      ,Obliczamy wskaźnik do miejsca w tablicy fontów, gdzie
                                                      ,znajduje sie początek definicji znaku o odczytanym kodzie
                                                      ,ASCII
     Offset = Asc(char) - 32                          ,Obsługiwane kody Ascii to zakres: 32...127
     Shift Offset , Left , 3                          ,Mnożymy otrzymaną wartość przez 8 gdyż każda definicja
                                                      ,znaku zajmuje 8 bajtów
     Offset = Offset + 4                              ,Pomijamy 4 bajty opisujące specyfikację czcionki, gdyż
                                                      ,używamy wyłącznie czcionki 8x8
                                                      ,Obliczmy współrzędne okna dla wyznaczenia aktywnego
                                                      ,obszaru wyświetlacza aby uprościć zapis bajtów
                                                      ,do pamięci wyświetlacza TFT. W ten sposób poruszamy się
                                                      ,wyłącznie w zakresie wyznaczonego okna pamięci obrazu.
     A = X1 + 7
     B = Y1 + 7
     Call Set_active_window(x1 , Y1 , A , B)
     Call Write_command(&H2c)                         ,Rozkaz: Write Memory Start
     Picture_pointer = Loadlabel(color8x8) + Offset   ,Ustawiamy wskaźnik na sekcję danych czcionki plus
                                                      ,niezbędny offset wynikający z kodu ascii wyświetlanego
                                                      ,znaku
     Lds _DPTRL , {Picture_Pointer}                   ,Do bibliotecznych zmiennych Bascoma (symboliczne nazwy
                                                      ,wybranych rejestrów) wczytujemy wskaźnik do tych danych
     Lds _DPTRH , {Picture_Pointer+1}
     For A = 1 To 8                                   ,Czytamy 8 kolejnych bajtów przypadających na definicję
                                                      ,znaku
        Read B
        For Index = 0 To 7
                                                      ,Rysujemy piksele w kolorze tekstu lub w kolorze tła
                                                      ,zależnie od tego czy kolejny bit w definicji czcionki jest
                                                      ,ustawiony czy wyzerowany
           If B.index = 1 Then Call Draw_pixel Else Call Draw_bk_pixel
        Next Index
     Next A
     X1 = X1 + 8                                      ,Zwiększamy wartości X1 by ustalić miejsce startowe dla
                                                      ,wyświetlenia kolejnych znaków
  Next Char_index
End Sub
```