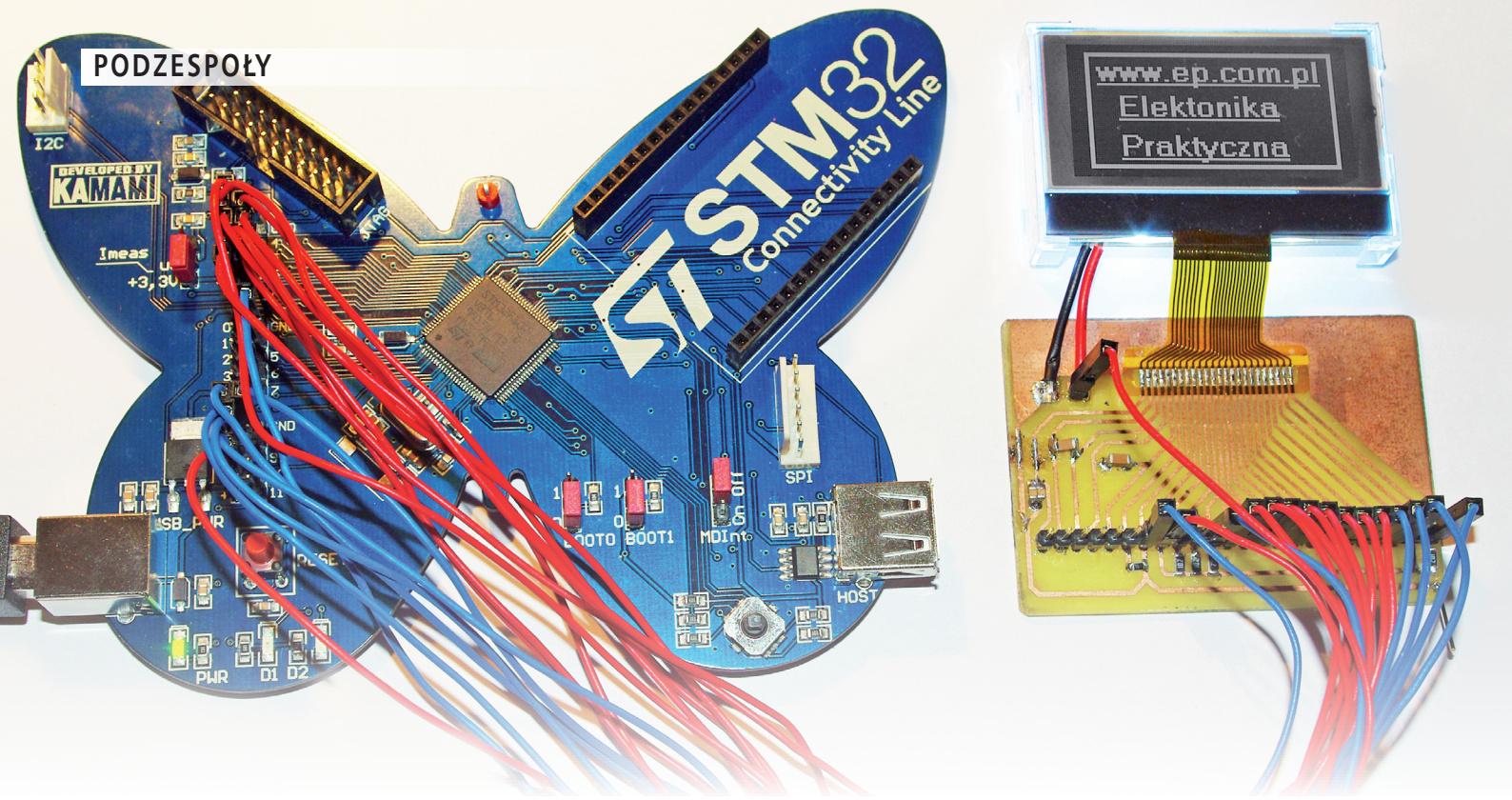


Wyświetlacz graficzny LCD-CGG 128064

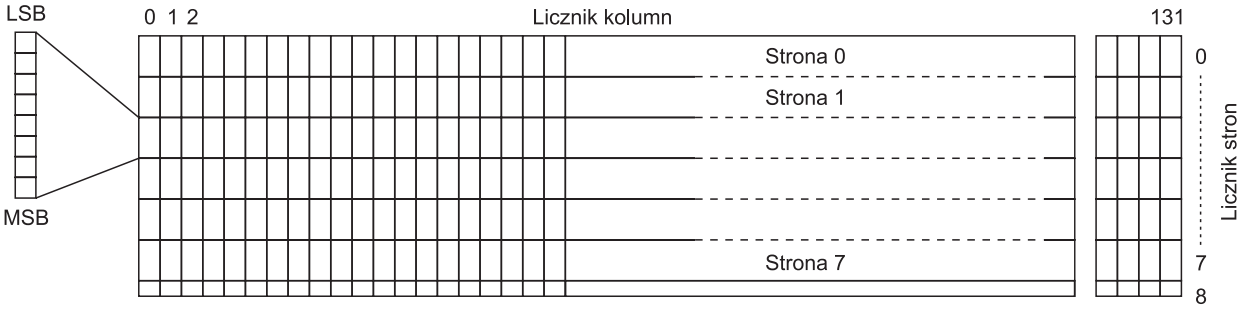
Wyświetlacz LCD-CGG 128064 ma nowoczesną matrycę FFSTN nazywaną przez producenta BlackLine. Matryca ma rozdzielczość 128×64 pikseli. Wyświetlacz zawiera wbudowany kontroler UC1601 produkowany przez firmę UltraChip.

U1601 został zaprojektowany z myślą o sterowaniu monochromatycznych matryc FSTN zorganizowanych w 65 wierszy po 132 segmenty. Drivery matrycy mogą być zasilane z wewnętrznej przetwornicy napięcia i nie ma konieczności stosowania dodatkowego źródła zasilania. Przetwornica do prawidłowego działania wymaga tylko kilku kondensatorów zewnętrznych, a jej napięcie wyjściowe można programować w szerokim zakresie, od ok. 4,8 V do 11,5 V. Współczynnik multipleksowania może być ustawiany w zakresie od 64 do 128, a programowany współczynnik temperaturowy pozwala na utrzymywanie prawidłowego kontrastu w szerokim zakresie temperatur. Dane i komendy są przesyłane magistralą równoległą zgodną ze standardem Intel 8080, Motorola 6800 oraz szeregowymi magistralami SPI lub I²C.

Organizacja pamięci obrazu

Pamięć obrazu zaimplementowana w sterowniku ma wielkość 65×132=8580 bitów i organizację bajtową. Każdemu pikselowi matrycy jest przyporządkowany jeden bit pamięci. Jeżeli bit ma wartość 1, to odpowiedni piksel jest zapalony. Wyzerowanie bitu powoduje wygaszenie piksela. Przy bajtowej strukturze pamięci można zapisywać tylko cały bajt. Zapisanie bajta odpowiada wyświetleniu pionowego paska składającemu się z 8 pikseli, tak jak pokazano na **rys. 1**.

Pamięć wyświetlacza jest adresowana przez licznik stron i licznik kolumn. Adres strony pamięci określa położenie bajta w jednej z 8 poziomych linijek o szerokości 8 pikseli i dodatkowo jednej linijce o szerokości 1 piksela. Stron jest 9, bo 8×8=64 piksele w pionie plus dodatkowo 1 piksel strony dziewiątej. Licznik kolumn określa położenie bajta (czyli paska o szerokości 8 pikseli) na stronie pamięci (czyli odpowiadającej stronie poziomej linijce). Liczniki stron i kolumn są zapisywane odpowiednimi komendami wysyłanymi przez zewnętrzny ste-



Rys. 1. Adresowanie pamięci obrazu sterownika UC1601

Tab. 1. Zestawienie komend sterownika UC1601

Komenda	C/D	W/R	D7	D6	D5	D4	D3	D2	D1	D0	Operacja
Write Data Byte	1	0	#	#	#	#	#	#	#	#	Zapis 1 bajta
Read Data Byte	1	1	#	#	#	#	#	#	#	#	Odczyt 1 bajta
Get Status	0	1	ID	MX	MY	WA	DE	0	0	0	Odczytanie statusu
			Product Code				Ver	0	0	0	
Set Kolumn Address LSB	0	0	0	0	0	0	CA3	CA2	CA1	CA0	Ustaw CA[3:0] młodszą część licznika kolumn
Set Kolumn Address MSB	0	0	0	0	0	1	CA7	CA6	CA5	CA4	Ustaw CA[7:4] starszą część licznika kolumn
Set Temperature compensation	0	0	0	0	1	0	0	1	TC1	TC0	Ustaw TC1, TC0
Set Power Control	0	0	0	0	1	0	1	PC2	PC1	PC0	Ustaw PC[2:0]
Set Adv. Program Config.	0	0	0	0 0	1 0	1 0	0 0	0 0	0 0	R 0	Nie używać
Set Scroll Line	0	0	0	1	SL5	SL4	SL3	SL2	SL1	SL0	Ustaw SL[5:0]
Set Vbias Potentiometer	0	0	1 PM7	0 PM6	0 PM5	0 PM4	0 PM3	0 PM 2	0 PM 1	1 PM0	Ustaw PM[7:0]
Set RAM Address Control	0	0	1	0	0	0	1	AC2	AC1	AC0	Ustaw AC[2;0]
Set All Pixes On	0	0	1	0	1	0	0	1	0	DC[1]	Zapal wszystkie piksele
Set Inverse Display	0	0	1	0	1	0	0	1	1	DC[0]	Wyświetl negatyw
Set Display Enable	0	0	1	0	1	0	1	1	1	DC[2]	Włącz wyświetlanie
Set Page Address	0	0	1	0	1	1	PA3	PA2	PA1	PA0	Ustawienie licznika stron
Set LCD Mapping Control	0	0	1	1	0	0	0	LC2	LC1	0	Ustawienie LC[2:1]
System Reset	0	0	1	1	1	0	0	0	1	0	Zerowanie
NOP	0	0	1	1	1	0	0	0	1	1	Nic nie rób
Set LCD Bias Ratio	0	0	1	1	1	0	1	0	BR [1]	BR[0]	Ustaw BR[1:0]
Set Partial Display Control	0	0	1	0	0	0	0	1	0	LC[4]	Ustawienie LC[4]
Set Frame Rate	0	0	1	0	1	0	0	0	0	LC[3]	Ustawienie LC[3]
Set All Pixel ON	0	0	1	0	1	0	0	1	0	DC[1]	Włączenie wszystkich pikseli
Set COM End	0	0	1 -	1 CEN 6	1 CEN 5	1 CEN 4	1 CEN 3	0 CEN 2	0 CEN 1	1 CEN0	Ustawienie CEN[6:0]
Set Partial Display Start	0	0	1 -	1 DST 6	1 DST 5	1 DST 4	1 DST 3	0 DST 2	1 DST 1	0 DST0	Ustawienie DST[6:0]
Set Partial Display Start	0	0	1 -	1 DEN 6	1 DEN 5	1 DEN 4	1 DEN 3	0 DEN 2	1 DST 1	0 DEN0	Ustawienie DEN[6:0]

Tab. 2. Komendy adresowanie pamięci obrazu

a) Komendy ustawiania licznika kolumn – komendy Set Kolumn Address LSB i MSB

Akcja	C/D	W/R	D7	D6	D5	D4	D3	D2	D1	D0
Ustawienie 4 młodszych bitów licznika kolumn CA[3:0]	0	0	0	0	0	0	CA3	CA2	CA1	CA0
Ustawienie 4 starszych bitów licznika kolumn CA[7:4]	0	0	0	0	0	1	CA7	CA6	CA5	CA4

b) Komenda ustawiania licznika strony pamięci

Akcja	C/D	W/R	D7	D6	D5	D4	D3	D2	D1	D0
Ustawienie licznika stron PA[3:0]	0	0	0	0	0	1	PA3	PA2	PA1	PA0

rownik do UC1601. Po każdym zapisie do pamięci licznik kolumn jest inkrementowany. Po osiągnięciu maksymalnej wartości licznik kolumn jest zerowany, a licznik stron zwiększany o 1. Jeżeli na przykład jest zaprogramowane inkrementowanie licznika kolumn i osiągnie on na stronie 0 wartość 131, to po następnym wpisie do pamięci licznik kolumn się wyzeruje, a licznik kolumn zwiększy o 1. Ten mechanizm jest bardzo wygodny przy zapisywaniu bitmap, pod warunkiem że matryca ma szerokość 132 pikseli.

Sterowanie funkcjami sterownika odbywa się przez wysyłanie do niego komend i odczytywania rejestru statusowego (**tab. 1**). Sterownik interpretuje przesyłane dane jako komendę, kiedy na linii C/D jest stan niski. Na **rys. 2** pokazane są wszystkie komendy sterownika i operacje zapisywania i doczytywania danych (C/D=1).

Do adresowania pamięci obrazu wykorzystywane są 2 komendy: zapisywania licznika kolumn: Set Column Address LSB, Set Column Address MSB, oraz zapisywania licznika stron: Set Page Address (**tab. 2**).

Licznik kolumn może mieć maksymalną wartość 131. Dla komend przyjmuje się zasadę, że starsze bity są kodem komendy, a młodsze argumentem. Ponieważ wartość licznika kolumn (argument komendy) zajmuje 8 bitów, jego zapis trzeba podzielić na 2 części. Komenda *Set Column Address LSB* zapisuje 4 młodsze bity licznika kolumn, komenda *Set Column Address MSB* 4 starsze bity. Argument komendy (numer strony) *Set Page Address* jest zapisany na 4 najmłodszych bitach. Sposób modyfikacji liczników adresowych przy wpisywaniu danych do pamięci obrazu jest programowany komendą *Set RAM Address Control* (**tab. 3**).

Ustawienie bitu AC0 powoduje, że liczniki kolumn i stron pamięci są inkrementowane automatycznie (zależnie od stanu bitu AC1) po każdym zapisie do pamięci obrazu. Jeżeli AC0 jest wyzerowany, to inkrementa-

Tab. 3. Komenda Set RAM Address Control

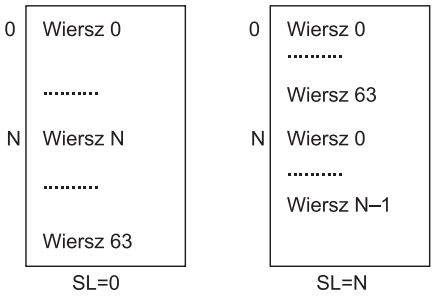
Akcja	C/D	W/R	D7	D6	D5	D4	D3	D2	D1	D0
Sposób modyfikacji liczników adresowych pamięci obrazu	0	0	1	0	0	0	1	AC2	AC1	AC0

Tab. 4. Komenda SetLcd Mapping Control

Akcja	C/D	W/R	D7	D6	D5	D4	D3	D2	D1	D0
Pozycjonowanie wyświetlanej informacji]	0	0	1	1	0	0	0	MY	MX	0

Tab. 5. Komenda ustawiania pierwszej linii obrazu

Akcja	C/D	W/R	D7	D6	D5	D4	D3	D2	D1	D0
Ustawienie pierwszej linii wyświetlacza	0	0	0	1	SL5	SL4	SL3	SL2	SL1	SL0



cja jest zatrzymywana po osiągnięciu przez licznik kolumn lub stron wartości maksymalnej. Dalsze wpisywanie do pamięci zatrzymuje inkrementację.

Kiedy bit AC1 jest wyzerowany, to po każdym wpisie do pamięci inkrementowany jest licznik kolumn, a po jego przepełnieniu licznik stron. Ustawienie bitu AC1 powoduje, że po każdym wpisie do pamięci jest inkrementowany licznik stron, a po jego przepełnieniu jest inkrementowany licznik kolumn. Każda zmiana bitu AC2 ze stanu niskiego na wysoki powoduje zmianę kierunku inkrementacji licznika stron pamięci. Powiązanie pamięci obrazu z pikselami matrycy programuje się komendą *Set LCD Mapping Control* (**tab. 4**).

Matryca wyświetlacza składa się z 64 wierszy. Każdy z wierszy ma 128 pikseli. Wiersze są ponumerowane od 1 do 64. Kiedy bit MY jest wyzerowany, to pierwszemu bajtowi w pamięci jest przyporządkowany pierwszy wiersz, drugiemu drugi itd. Ustawienie MY powoduje lustrzane odbicie w przyporządkowaniu wierszy. Pierwszy bajt odpowiada 64. wierszowi, drugi 63. itd. Podobnie jest z przyporządkowaniem kolumn. Dla wyzerowanego bitu MX pierwszy bit najmłodszego bajta pamięci odpowiada pierwszej kolumnie. Programowanie lustrzanego odbicia umożliwia zmianę położenia wyświetlanej informacji bez konieczności fizycznego przestawiania ekranu wyświetlacza (**fot. 2**, **fot. 3**).

Komenda *Set Start Line* (**tab. 5**) określa, któremu wierszowi matrycy będą odpowiadały pierwsze lokacje w pamięci obrazu. Jeżeli standardowo wpiszemy do SL5:SL0 same zera, to wyświetlanie rozpocznie się od skrajnego górnego wiersza. Wpisanie dowolnej wartości N od 0 do 63 (wyświetlacz ma 64 linii w pionie) spowoduje, że wyświetlanie rozpocznie się od linii o numerze N i będzie się „zawijało" wokół wiersza o numerze 63 (**fot. 4**). Ten mechanizm umożliwia przesuwanie (skrolowanie) wyświetlanej informacji w pionie.

Korekcję temperaturową programuje się komendą *Set Temperature Compensation* (**tab. 6**). Korekcja temperaturowa jest wykorzystywana do regulacji napięcia zasilania matrycy w funkcji temperatury otoczenia (**tab. 7**).

Napięcie wyjściowe przetwornicy z pompą ładunku wbudowanej w wyświet-

Tab. 6. Komenda ustawienia korekcji temperaturowej

Akcja	C/D	W/R	D7	D6	D5	D4	D3	D2	D1	D0
ustawianie korekcji temperaturowej	0	0	0	0	1	0	0	1	TC1	TC0

Tab. 7. Nastawy współczynnika korekcji temperaturowej

TC	0	1	2	3
%/°C	–0,05	–0,10	–0,15	0

Tab. 8. Komenda ustawienia napięcia wstępnego BR[1:0], 00b=6, 01b=7, 10b=8, 11b=9

Akcja	C/D	W/R	D7	D6	D5	D4	D3	D2	D1	D0
ustawianie współczynnika Biasu	0	0	1	1	1	0	1	0	BR1	BR0

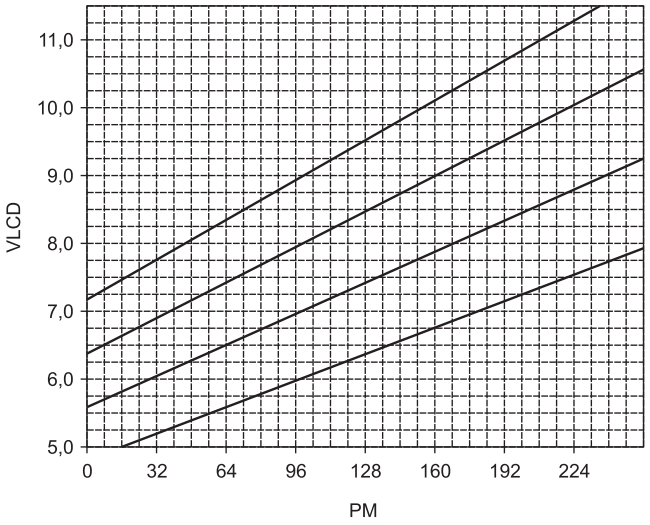
Tab. 9. Komenda Set Vbias Potentiometer

Akcja	C/D	W/R	D7	D6	D5	D4	D3	D2	D1	D0
Kod komendy	0	0	1	0	0	0	0	0	0	1
Ustawienie wzmocnienia i rejestru potencjomentru	0	0	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0

Fot. 2. Wyświetlanie dla MX=1, MY=0

Fot. 3. Wyświetlanie dla MX=0, MY=1

Fot. 4. Działanie komendy Set Start Line



Rys. 5. Zależność napięcia zasilania matrycy od PM i BR

lacz zależy od napięcia polaryzacji wstępnej, zawartości rejestru wewnętrznego potencjometru i opisanego wcześniej współczynnika temperaturowego. Jest wyliczane według zależności:

$$V_{LCD} = (C_{V0} + C_{PM} \times PM) \times (1 + (T - 25) \times C_T\%)$$

gdzie:

- Cvo, Cpm: stałe zależne od zaprogramowanego napięcia wstępnego i współczynnika wzmocnienia Gain,
- PM: zawartość rejestru potencjometru,
- T: temperatura otoczenia,
- CT: współczynnik temperaturowy (**tab. 7**).

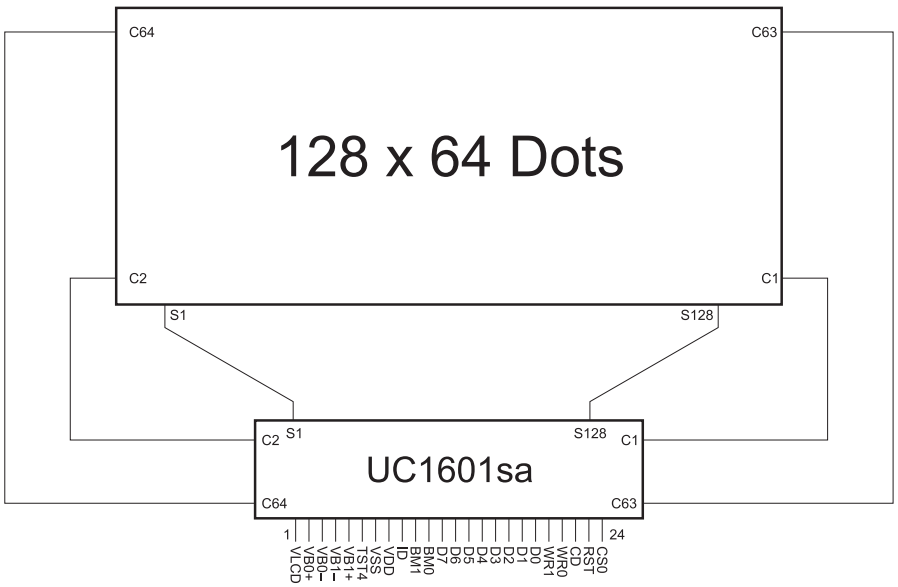
Mnożnik napięcia polaryzacji wstępnej programuje się komendą *Set Bias Ratio* (**tab. 8**), a rejestr potencjometru komendą *Set Vbias Potentiometer* (**tab. 9**). Ta ostatnia komenda jest dwubajtowa. Najpierw jest wysyłany 8-bitowy kod komendy 0x81, a potem 8-bitowy argument z zawartością.

Współczynniki Cvo i Cpm wylicza się na podstawie wartości BR (*Bias Ratio*). W **tab. 10** pokazano gotowe, wyliczone wartości tych stałych i zakresy napięcia zasilającego matrycę VLCD w zależności od wartości BR i rejestru potencjometru.

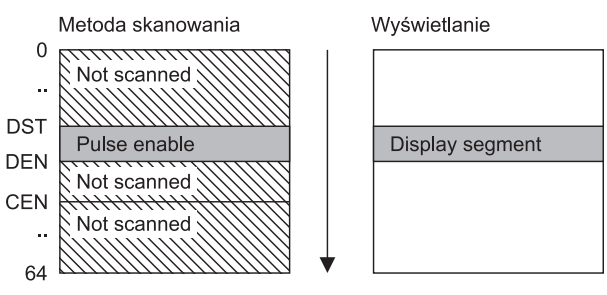
Na **rys. 5** pokazano zależność wartości napięcia zasilającego matrycę LCD (Vlcd) od współczynnika BR (niebieskie linie) i wartości PM zapisanej w rejestrze potencjometru. W dokumentacji wyświetlacza jest podane napięcie zasilania matrycy. Mając tę wartość i tab. 10 lub rys. 5, można szybko oszacować wartość PM i BR, tak aby napięcie było trochę niższe niż zalecane, a potem dokładnie je wyregulować, zmieniając zawartość rejestru potencjometru. Zostanie to dokładniej omówione przy okazji omawiania procedury inicjalizacyjnej. Producent zaleca, by napięcie wyjściowe przetwornicy nie było większe niż 11,5 V w całym zakresie temperatur.

Ostatnią komendą związaną z układem zasilania matrycy jest *Set Power Control* (**tab. 12**). Jeżeli zasilanie matrycy jest podłączone do wyjścia wewnętrznej przetwornicy, to bity PC1 i PC2 muszą być ustawione. Jeżeli znana jest pojemność matrycy, to trzeba ją uwzględnić, programując bit PC0. W innym przypadku wartość tego bitu można próbować ustalić eksperymentalnie.

Dwubajtowa komenda *Set COM End* (**tab. 13**) jest używana dla wyświetlaczy, których matryce mają mniej niż 64 wiersze. W naszym przypadku wyświetlacz ma 64 wierszy i nie musimy tej komendy używać (domyślna wartość to 64).



Rys. 7. Wyprowadzenia wyświetlacza



Rys. 6. Wydzielenie części obszaru do wyświetlania

Komendy *Set Partial Display Start* i *Set Partial Display End* (**tab. 14**, **tab. 15**) są przeznaczone do wydzielania części ekranu z operacji wyświetlania. Używając ich, można wydzielić tylko część wierszy, które będą wyświetlane – **rys. 6**.

Podłączenie modułu

Moduł wyświetlacza ma wyprowadzoną giętką taśmę przyłączeniową zakończoną cynowanymi polami kontaktowymi (**rys. 7**, **fot. 8**). Pola kontaktowe lutuje się bezpośrednio do ocynowanych kontaktów na płytce drukowanej (**fot. 9**). Dane i komendy są przesyłane do sterownika interfejs równoległy zorganizowany według standardów Intel 8080 lub Motorola 6800 oraz przez interfejs szeregowy SPI lub I²C.

Przed napisaniem procedur obsługi wyświetlacza trzeba zdecydować, czy będzie używany interfejs równoległy, czy szeregowy i według jakiego standardu będzie pracować. W **tab. 16** umieszczono opis poszczególnych linii równoległej magistrali danych. Linie BM0 i BM1 służą do wybierania rodzaju

i długości magistrali. Dla magistral szeregowych wybór dodatkowych wariantów jest wykonywany przez wymuszanie stanów na liniach DB7 i DB6 – dodatkowe informacje można znaleźć w dokumentacji sterownika.

W przykładach programowej obsługi wyświetlacza wybrano 8-bitową magistralę Intel 8080. Wtedy wyprowadzenie BM1 musi być w stanie wysokim, a BM0 w stanie niskim. Przebiegi czasowe dla wybranej magistrali zgodnej ze standardem Intel8080 pokazano na **rys. 10**.

Sterownik ma wbudowany mechanizm poprawnego zerowania po włączeniu zasilania (POR). Dlatego można nie wykorzystywać wejścia zerowania RST (musi być wtedy podłączone do plusa zasilania – Vdd). Zerowanie w trakcie normalnej pracy wykonuje się komendą zerowania programowego *System Reset*.

Program sterujący

Do testowania działania wyświetlacza wykorzystano płytkę ewaluacyjną *STM32 Butterfly* z programatorem *JTAG ST-Link ZL30PRG*. Zamontowany na płytce ewaluacyjnej procesor STM32F107VBT6 ma 32-bitowy rdzeń Cortex M3 taktowany zegarem 72 MHz, wbudowane 128 kB pamięci programu Flash i 48 kB wewnętrznej pamięci SRAM. Program sterujący napisano w środowisku uVision V4 dla mikrokontrolerów rdzeniem ARM.

Magistrala standardu Intel8080 jest emulowana za pomocą linii portów GPIOC i GPIOD. Pierwszą czynnością, którą należy wykonać, jest skonfigurowanie portów jako wyjściowe. Na **list. 1** pokazano definicje używane w procedurach manipulacji liniami portów. Modyfikując te definicje, można w prosty sposób użyć innych portów bez konieczności wykonywania zmian w wielu miejscach programu.

Do konfiguracji linii portów zostaną wykorzystane funkcje standardowej biblioteki API bezpłatnie udostępnionej przez producenta mikrokontrolera – firmę STM. Ponieważ w mikrokontrolerach STM32 każdy blok peryferyjny ma oddzielny system taktowania, należy go również skonfigurować przed użyciem portów. Na **list. 2** pokazano sposób konfiguracji używanych linii portów GPIOC i GPIOD.

Konfigurację systemu taktowania wykonuje funkcja API: *RCC_APB2PeriphClockCmd*, a konfigurację modułu portów funkcja API: GPIO_Init. Porty skonfigurowano jako wyjściowe typu push-pull, pracujące z maksymalną częstotliwością 50 MHz. Przy emulacji magistrali będą potrzebne funkcje zmiany stanów pojedynczych linii, ale też funkcja wysłania bajta na port. Funkcje API ustawiania i zerowania linii pokazano na **list. 3**. Zapis 8-bitowej danej wykonuje funkcja *GPIO_Write(PORT_data,data)*.

Tab. 10. Wyliczone współczynniki Cvo i Cpm oraz napięcie wyjściowe VLCD

BR	Cvo (V)	Cpm (Mv)	PM	Zakres Vlcd
6	4,800	12,24	0	4,80
			255	7,92
7	5,600	14,28	0	5,60
			255	9,24
8	6,400	16,32	0	6,40
			255	10,56
9	7,200	18,36	0	7,20
			255	11,50

Tab. 12. Komenda Set Power Control
PC0 programuje zakres pojemności matrycy ob<=15nF , 1b>15...24nF
PC[2:1] zasilanie matrycy 00b=zewnętrzne VLCD 11b VLCD z wbudowanej matrycy

Akcja	C/D	W/R	D7	D6	D5	D4	D3	D2	D1	D0
ustawianie zasilania matrycy	0	0	0	0	1	0	1	PC2	PC1	PC0

Tab. 13. Komenda ustawienia liczby wierszy matrycy

Akcja	C/D	W/R	D7	D6	D5	D4	D3	D2	D1	D0
Kod komendy	0	0	1	1	1	1	0	0	0	1
Ustawienie ilości kolumn	0	0	–	CEN[6:0]						

Tab. 14. Komenda ustawienia wiersza początkowego

Akcja	C/D	W/R	D7	D6	D5	D4	D3	D2	D1	D0
Kod komendy	0	0	1	1	1	1	0	0	0	1
Ustawienie początkowego wiersza	0	0	–	DST[6:0]						

Tab. 15. Komenda ustawienia wiersza końcowego

Akcja	C/D	W/R	D7	D6	D5	D4	D3	D2	D1	D0
Kod komendy	0	0	1	1	1	1	0	0	0	1
Ustawienie końcowego wiersza	0	0	–	DEN[6:0]						

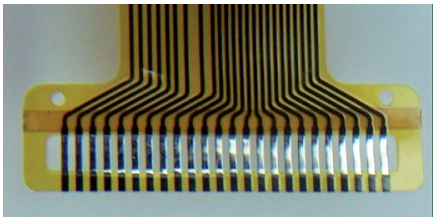
Tab. 16. Funkcje wyprowadzeń magistrali

Nazwa	Typ	opis
BM[1:0]	I	Wybór magistrali 11b 8-bitowa Motorola 6800 00 b 8-bitowy SPI 10b 8-bitowa Intel 8080 01b 9-bitowa SPI lub I²C
CS0	I	Wybór układu – układ aktywny, kiedy CS jest w stanie niskim
RST	I	RST=0b zerowanie sterownika i inicjowanie rejestrów wartościami domyślnymi
CD	I	Wybór komendy/dane CD=0b – komendy CD=1b dane do pamięci obrazu
WR0 WR1	I I	Intel 8080 linia zapisu WR Intel 8080 linia odczytu RD
D[7:0]	I/O	Dwukierunkowa magistrala danych dla magistrali równoległych i linie danych, zegarowe i sterujące dla magistral szeregowych

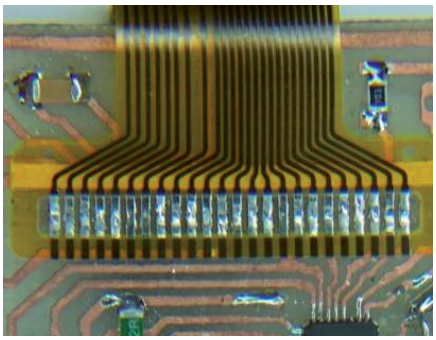
List. 1. Definicje linii portów

```
//definicja portu linii sterujących
#define PORT_ctrl                    GPIOC
#define RCC_APB2Periph_ctrl    RCC_APB2Periph_GPIOC
//definicja portu magistrali danych
#define PORT_data                    GPIOD
#define RCC_APB2Periph_data    RCC_APB2Periph_GPIOD

//definicja linii sterujących
#define WR GPIO_Pin_4       //WR PC4
#define RD GPIO_Pin_5       //RD PC5
#define CE GPIO_Pin_6       //CE PC6 - linia CS0
#define CD GPIO_Pin_7       //CD PC7
#define RES GPIO_Pin_8      //RES PC8
//definicje linii magistrali
#define D0 GPIO_Pin_0
#define D1 GPIO_Pin_1
#define D2 GPIO_Pin_2
#define D3 GPIO_Pin_3
#define D4 GPIO_Pin_4
#define D5 GPIO_Pin_5
#define D6 GPIO_Pin_6
#define D7 GPIO_Pin_7
```



Fot. 8. Taśma przyłączeniowa z ocynowanymi polami kontaktowymi

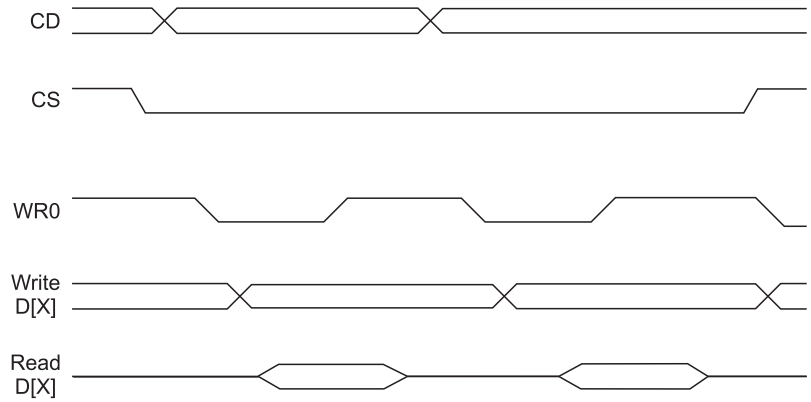


Fot. 9. Złącze przylutowane do płytki drukowanej

Mamy już wszystko, co konieczne, żeby napisać procedury wysyłania do sterownika komend i danych do pamięci obrazu. Przesyłanie danych pamięci obrazu i komend różni się tylko stanem linii C/D – pozostałe sekwencje są takie same. Procedura *wr_lcd_bus* (**list. 4**) wykonuje sekwencję zapisu do sterownika 8-bitowej danej. Używając jej, można napisać dwie proste funkcje zapisu danej (C/D=1) i komendy (C/D=0) – **list. 5**.

Tab. 17. Inicjacja rejestrów po włączeniu zasilania

Nazwa	Bity	Wartość domyślna	Opis
SL	6	00h	SL[5:0] początkowa linia wyświetlania
CA	8	00h	Licznik kolumn
PA	4	00h	Licznik stron pamięci obrazu
BR	2	03h	Współczynnik bias =9
AC	3	01h	AC[0]=1, AC[1]=0, AC[2]=0
PM	8	C0h	Rejestr potencjometru
DC	6	00h	Start wyświetlania
PC	3	06h	Zasilanie matrycy LCD<15nF, wewnętrzne napięcie VLCD
TC	2	00h	Kompensacja temperaturowa –0,05%/°C



Rys. 10. Przebiegi czasowe dla magistrali Intel 8080

Po zapisaniu komendy sterownik może być zajęty jakiś czas jej realizacją. Stan zajętości można stwierdzić, odczytując rejestr statusowy lub przeczekać, stosując programowe opóźnienia. Jednak okazało się, że we wszystkich testowanych procedurach zastosowanie programowego opóźnienia wystarczyło i odczytywanie statusu nie było konieczne. Z tego powodu nie jest ono zaimplementowane.

Po włączeniu zasilania układ POR (Power On Reset) zeruje sterownik i inicjuje część rejestrów wartościami domyślnymi (**tab. 17**).

Jeżeli wartości domyślne rejestrów są odpowiednie dla zastosowanej matrycy, to można ich nie inicjować ponownie programowo, ale ze względu na przejrzystość programu lepiej dla używanych opcji to zrobić. W procedurze inicjalizacji pokazanej na **list. 6** wszystkie potrzebne rejestry są inicjalizowane.

Inicjalizacja rozpoczyna się od sprzętowego zerowania sterownika przez wymuszenie stanu niskiego na linii RES. Potem wszystkie linii sterujące są ustawiane w stan początkowy wysoki i wysyłane są kolejno komendy:

- *Set Temperature Compensation* TC=0,0%/C;
- *Set LCD Mapping Control* z parametrami MY=1, MX=0;
- *Set LCD Bias Ratio* ze współczynnikiem BIAS=9;
- *Set Vlcd Potentiometer* z wartością 0x60;
- *Set Display Enable* włączająca zasilanie matrycy.

Po wysłaniu tych komend do pamięci obrazu wyświetlacza wpisywanych jest 1188 bajtów zerowych. Zerowanie pamięci jest konieczne, bo po włączeniu zasilania w pamięci są wartości przypadkowe.

Komenda *Set LCD Mapinng Control* ustala położenie wyświetlanej informacji. Jeżeli z jakichś względów konieczne jest obrócenie wyświetlanej informacji o 180°, można to zrobić programowo, wysyłając komendę *Set LCD Mapping Control* z parametrami MY=1 i MX=0 (fot. 2 i fot. 3).

Najczęściej dla konkretnego typu wyświetlacza podawana jest wartość optymalnego napięcia zasilania matrycy. Niestety w tym przypadku w momencie pisania tego artykułu napięcie nie było mi znane. Byłem zmuszony, kierując się doświadczeniem, dobrać napięcie Vlcd, tak aby kontrast wyświetlanej informacji był jak najlepszy.

List. 2. Konfiguracja linii portów

```
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_ctrl, ENABLE);
  GPIO_InitStructure.GPIO_Pin = WR | RD | CE | CD | RES ;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_Init(PORT_ctrl, &GPIO_InitStructure);

  RCC_APB2PeriphClockCmd(RCC_APB2Periph_data, ENABLE);
  GPIO_InitStructure.GPIO_Pin = D0 | D1 | D2 | D3 | D4| D5 | D6 | D7  ;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_Init(PORT_data, &GPIO_InitStructure);
```

List. 3. Funkcje ustawienia i zerowania linii

```
// funkcja ustawienia linii
void SetBit(int bits)
{
  GPIO_SetBits(PORT_ctrl, bits);
}
//funkcja zerowania linii
void ClrBit(int bits)
{
  GPIO_ResetBits(PORT_ctrl, bits);
}
```

List. 4. Zapisanie do sterownika 8-bitowej danej

```
//zapisanie danej na magistrale
void wr_lcd_bus(unsigned char data){
  GPIO_Write(PORT_data,data);//8 bitowa dana na magistrali
  ClrBit(CE);;//linia CE=0
  ClrBit(WR);delay();delay();//WR=0 - zapisanie danej
  SetBit(WR);//WR=1
  SetBit(CE);//CE=1
}
```

Dla zaprogramowanego BR=9 Cpm wynosi 18,36 mV, a napięcie zasilania matrycy 7,20...11,50 V (tab. 10 i tab. 11). Dolne napięcie zakresu uzyskuje się dla wyzerowanego rejestru potencjometru. Po wpisaniu do rejestru potencjometru wartości 96 (0x60) uzyskujemy zwiększenie napięcia o 96×18,36 mV=1,763 V, czyli napięcie zasilania matrycy VLCD=7,20+1,763=8,963 [V]. Tę wartość można zmieniać w pewnych granicach, modyfikując zawartość rejestru potencjometru.

Procedurę inicjalizacji kończy wysłanie komendy włączającej sterowanie matrycy i zapisanie całej pamięci obrazu zerami. Po wykonaniu procedury inicjalizacyjnej wyświetlacz jest gotowy do pracy. Liczniki wierszy i kolumn są wyzerowane i zapisywane do pamięci obrazu dane będą wyświetlane od lewego górnego rogu wyświetlacza. W wielu przypadkach konieczne jest ustalenie konkretnej pozycji, od której rozpocznie się wyświetlanie.

Do pozycjonowania początku wyświetlania jest przeznaczona pokazana na **list. 7** procedura Poz. Jej argumentami są wartości *x*, czyli numer kolumny w wierszu i *y*, czyli numer wiersza. Wartość *x* jest wpisywana do licznika kolumn komendami *Set Column Addres LSB* i *Set Column Addres MSB*, a wartość *y* komendą *Set Page Address*.

Wyświetlanie tekstu

Sterownik wyświetlacza nie ma typowego trybu tekstowego. Żeby wyświetlać tekst, trzeba sobie samemu zdefiniować wzorce znaków w zewnętrznym sterowniku. Ponieważ wyświetlacz jest graficzny, nie ma ograniczenia rozmiarów czcionki, ale trzeba pamiętać, że w pionie możemy wyświetlać „porcjami" po 8 pikseli. Najbardziej praktyczne są znaki o wysokości równej wielokrotności 8 pikseli, czyli na przykład 6×8 czy 12×16 itp.

Na **list. 8** pokazano funkcję wyświetlającą znaki alfanumeryczne o rozmiarze 6×8 pikseli zdefiniowane w tablicy *rom_gen[]*. Kod ASCII znaku pomnożony przez 6 daje adres początku wzorca w tablicy. Potem jest pobieranych i zapisywanych do pamięci obrazu 6 kolejnych bajtów.

Przy tak skonstruowanej tablicy wzorców znaków łatwo jest wyświetlać napisy. Funkcja *DispTxt* (**list. 9**) wyświetla napis umieszczony w argumencie od pozycji określonej przez *x* i *y*. Pozycjonowanie wykonuje *PozCh* (**list. 10**). Różni się ona od pozycjonowania *Poz()* tym, że współrzędna *x* jest mnożona przez 6. Pozycja to numer wyświetlanego wiersza i numer znaku, a nie numer kolumny, jak w przypadku funkcji *Poz()*. Efekt działania funkcji pokazano na **fot. 10**.

Żeby wyświetlić bitmapę na wyświetlaczu monochromatycznym, trzeba ją najpierw przekonwertować na bitmapę 1-bitową, a potem na tablicę w języku C. W przypadku bitmap kolorowych i monochromatycznych ze skalą szarości jednemu pikselowi odpowiada co najmniej 1 bajt. W bitmapach 1-bitowych jednemu pikselowi odpowiada jeden bit i bardzo ważne jest, jak te bity są przypisane pikselom w bitmapie. Możliwości jest kilka. Kolejne bajty mogą odpowiadać poziomym paskom pikseli. Jeżeli paski są pionowe, to mogą być adresowane tak, że kolejne paski nie tworzą poziomych linijek o szerokości 8 pikseli, ale pionowe paski o szerokości 1 piksela.

List. 5. Funkcje zapisania danej i komendy

```
//zapisanie danej do sterownika
void lcd_write_data(unsigned char data){
  SetBit(CD);//linia CD=1 - zapisanie danej
  SetBit(RD);//linia RD=1
  wr_lcd_bus(data);//zapisanie na magistrale
}
//zapisanie komendy do sterownika
void lcd_write_cmd(unsigned char cmd){
  ClrBit(CD);//linia CD=0 - zapisanie komendy
  SetBit(RD);//linia RD=1
  wr_lcd_bus(cmd);//zapisanie na magistrale
}
```

List. 6. Inicjalizacja wyświetlacza

```
void lcd_init(void){
int i;

  for(i=0;i<10000;i++)
  delay();
  ClrBit(RES);//reset wyswietlacza aktywny
  for(i=0;i<8000;i++)
  delay();
  SetBit(RES);//reset nieaktywny
  SetBit(WR);//te linie w stan wysoki
  SetBit(RD);
  SetBit(CD);
  SetBit(CE);
  lcd_write_cmd(0x27); delay1(); //temp compens. 0,00%/C
  lcd_write_cmd(0xc4); delay1();//LCD Mapping
  lcd_write_cmd(0xa0); delay1();//frame rate
  lcd_write_cmd(0xeb); delay1();//bias ratio
  lcd_write_cmd(0x81); lcd_write_cmd(0x60); delay1();// potencjometr
  lcd_write_cmd(0xaf);//display EN
  for(i=0;i<1188;i++)
  lcd_write_data(0);
}
```

List. 7. Procedura pozycjonowania na wyświetlaczu

```
//ustalenie pozycji na wyświetlaczu
void Poz(char x, char y)
{
  lcd_write_cmd(y|0xB0);//ustawienie licznika wierszy
  lcd_write_cmd(x&0x0F);//ustawienie licznika kolumn
  lcd_write_cmd(((x&0xF0)>>4)|0x10);
}
```

List. 8. Wyświetlanie znaku o rozmiarze 8×6 pikseli

```
//wyswietlenie zaku ascii 8×6 pixeli
void WriteChar(char code)
{
  int code_point;
  code_point=((int)code*6); //wyliczenie adresu początku wzorca
  lcd_write_data(rom_gen[code_point++]);
  lcd_write_data(rom_gen[code_point++]);
  lcd_write_data(rom_gen[code_point++]);
  lcd_write_data(rom_gen[code_point++]);
  lcd_write_data(rom_gen[code_point++]);
  lcd_write_data(rom_gen[code_point]);
}
```

List. 9. Wyświetlanie tekstu

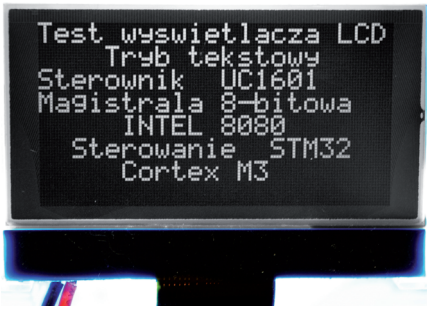
```
void DispTxt(const char *napis,char x,char y)
{
  PozCh(x,y);//pozycja tekstu na wyswietlaczu
  while(*napis!=0)
  {WriteChar(*napis);
  ++napis;}
}
```

List. 10. Pozycjonowanie tekstu

```
//ustalenie pozycji znaku na wyświetlaczu
void PozCh(char x, char y)
{
  x=x*6;
  lcd_write_cmd(y|0xB0);//ustawienie licznika wierszy
  lcd_write_cmd(x&0x0F);//ustawienie licznika kolumn
  lcd_write_cmd(((x&0xF0)>>4)|0x10);
}
```

List. 11. Zapisanie pełnowymiarowej bitmapy

```
void WriteBitmap(void)
{
  unsigned char i,j;
  short k;
  k=0;
  for(i=0;i<8;i++){
  Poz(0,i); //pozycja wiersza
  for(j=0;j<128;j++)
  lcd_write_data(bmp[k++]); //wpisanie 128bajtów bitmapy.
  }
}
```



Rys. 11. Wyświetlanie tekstu

Jeżeli do edycji bitmapy chcemy używać programów graficznych, a potem uzyskać tablicę z bajtami ułożonymi tak, by pasowały do organizacji pamięci wyświetlacza, to trzeba się posłużyć programem do konwersji. Można skorzystać np. ze znanego programu AsystentLCD autorstwa Radosława Kwietnia. Oprócz konwersji bitmap można za jego pomocą tworzyć wzorce znaków alfanumerycznych 8×8 i 16×16 pikseli. Inny bardzo dobry program to *bmp2c* autorstwa Jerzego Szczesiula.

Obsługa tego programu nie jest trudna. Najpierw wybieramy rozmiar bitmapy w pikselach, a potem po wciśnięciu przycisku BMP wczytuje się bitmapę. Po zaznaczeniu opcji V-8 i naciśnięciu przycisku C w schowku Windows jest zapisywana gotowa tablica w języku C, którą wystarczy wkleić do pliku i skompilować. Nie da się tak przygotowanej bitmapy wyświetlić poprzez wpis kolejnych 1024 bajtów do pamięci sterownika wyświetlacza, bo pamięć jest przygotowana na rozmiar matrycy 132×64, a nasz wyświetlacz ma rozmiar 128×64. Trzeba ją wyświetlać wierszami po 128 bajtów (po usunięciu 2 pierwszych bajtów). Po zapisaniu 128 bajtów, funkcją *Poz()* ustawiamy pierwszą pozycję w kolejnym wierszu (banku pamięci) i tak aż do zapisania wszystkich 8 banków.

Podsumowanie

Przedstawione tutaj procedury pozwalają na zainicjowanie wyświetlacza i wyświetlanie napisów w trybie tekstowym oraz wyświetlanie bitmap. Jak już wspomniałem, nie używałem odczytywania rejestru statusowego. W procedurach inicjalizacji zastosowałem niewielkie opóźnienia, a przy zapisywaniu pamięci obrazu nawet one nie były konieczne. W bardziej rozbudowanych aplikacjach można też wykorzystać możliwość odczytywania zawartości pamięci obrazu.

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