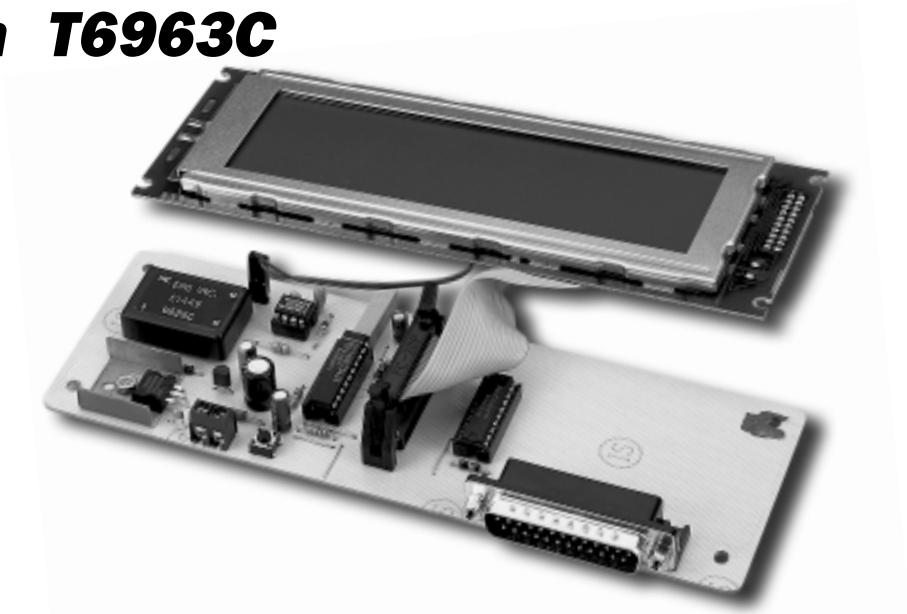
Programowanie sterowników wyświetlaczy graficznych, część 1

Wyświetlacz graficzny LCD z kontrolerem T6963C

Artykuł ten jest dość nietypowy - przynajmniej jak na EP. Opisujemy w nim bowiem konstrukcję urządzenia, ale tylko przy okazji opisu sposobu sterowania graficznych wyświetlaczy LCD. Artykułu tego nie można przegapić! Znajdziecie w nim bowiem komplet informacji o sposobach sterowania graficznych wyświetlaczy oraz doskonałe narzędzie do sterowania nimi.



Gdy budujemy jakiekolwiek urządzenie mikroprocesorowe prawie zawsze zachodzi konieczność wyświetlenia danych: wyników pomiaru, wprowadzonych parametrów, czasu, daty itp. W większości przypadków wystarcza nam wizualizacja znakowa. Jest ona łatwo realizowalna (LED, LCD, wyświetlacze alfanumeryczne), w miarę tania, a także wielokrotnie opisywana i nie stwarzająca większych problemów programowych.

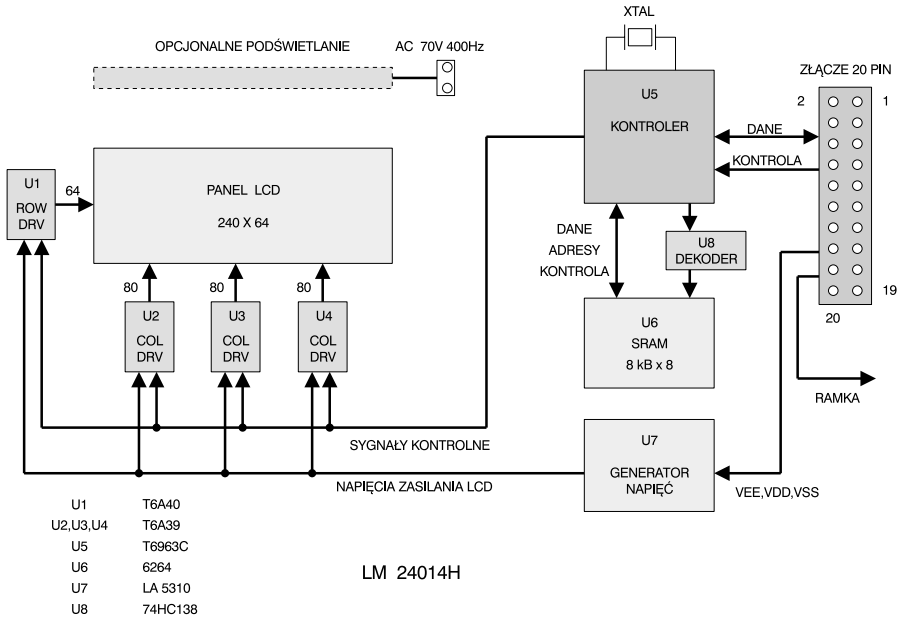
Jednak czasem - chociaż raczej rzadko w domowych, amatorskich aplikacjach - funkcje urządzenia wymagają wizualizacji graficznej. Jest tak na przykład w razie konieczności przedstawiania przebiegów czasowych, zwięzłego i skutecznego zobrazowania przebiegu procesu technologicznego lub stanu urządzeń itp. W takich przypadkach warto sięgnąć po uniwersalne, graficzne wyświetlacze LCD. Są one dostępne w różnych rozmiarach i wykonaniach, począwszy od całkiem niewielkich (128x32 piksele), a kończąc na dużych ekranach używanych w laptopach.

W naszym przykładzie wykorzystania wyświetlacza graficznego użyjemy modelu LM24014H firmy Sharp o wymiarach ekranu 240x64 piksele.

Budowa wyświetlacza

Podobnie jak w wyświetlaczach alfanumerycznych mamy do czynienia ze zintegrowanym zespołem panelu LCD oraz elektroniki sterującej, pozwalającej na stosunkowo proste sterowanie wyświetlaczem. Schemat blokowy urządzenia jest przedstawiony na **rys. 1**.

Ekran panelu LCD jest matrycą 240x64 sterowanych indywidualnie pikseli. Jest wykonany jako transfleksyjny, czyli może pracować zarówno ze światłem zewnętrznym - odbitym, jak i ze światłem przechodzącym, emitowanym przez podświetlającą folię elektroluminescencyjną. Stanem aktywnym (zapaleniem) piksela jest jego zaciemnienie. Indywidu-



Rys. 1. Schemat blokowy wyświetlacza LM24014H.

alne wysterowanie każdego piksela zapewnia zespół driverów (U1 - wiersze, U2, U3, U4 - kolumny) podających w odpowiednie miejsca ekranu napięcie przemienne między elektrodami wiersza i kolumny. Odpowiednie napięcia są formowane z zewnętrznego zasilania przez pomocniczy układ U7.

Wszelkimi operacjami logicznymi oraz komunikacją zajmuje się zespół sterujący: kontroler U5 współpracujący z pamięcią SRAM 8kBx8 - U6, wspomagany dekoderem U8 i rezonatorem ceramicznym XTAL. Jako kontroler pracuje układ Toshiby T6963C. Chociaż w wyświetlaczach graficznych nie ma tak uniwersalnego standardu jaki stanowi na przykład układ Hitachi HD44780 w wyświetlaczach alfanumerycznych - to T6963C jest spotykany dosyć często i opisane dalej zasady jego obsługi mogą się przydać także dla innych typów wyświetlaczy. Łączność kontrolera z otoczeniem zapewnia 20-pinowe typowe złącze (piny widziane od strony panelu - nie od strony gniazda):

1 - FGND - masa obudowy (metalowej ramki mocującej panel LCD)
2 - GND (inaczej Vss) - masa
3 - Vdd - zasilanie układów logiki (+5V)
4 - Vee - zasilanie panelu LCD (regulacja kontrastu - ok. -10V)
5 - WR - zapis (aktywny poziom niski)
6 - RD - odczyt (aktywny poziom niski)
7 - CE - wejście zezwalające (aktywny poziom niski)
8 - C/D - przełączanie: komendy/dane (HI - komenda, LO - dana)
9 - NC - nie podłączone
10 - RESET - zerowanie (poziomem niskim)
11 - 18 - D0..D7 - magistrala danych
19 - FS - wybór szerokości fontu (HI - 6x8, LO - 8x8) (dla trybu tekstowego)
20 - NC - nie podłączone

Operując odpowiednio powyższymi sygnałami realizujemy potrzebne sekwencje zapisu i odczytu komend i danych. Zależności czasowe pomiędzy sygnałami zostały przedstawione na **rys. 2**.

Jak widać, jest tu wiele zbieżności ze sposobem sterowania wyświetlaczem alfanumerycznym, chociaż oczywiście występuje o wiele więcej komend, trybów pracy itp.

Kilka słów o napięciach zasilania. Układy logiczne mają typowe, własne zasilanie +5V. Napięcie dla panelu LCD jest natomiast formowane z napięć Vdd i Vee. Maksymalny zakres Vee to -12V do -6V. Należy zapewnić regulację tego napięcia, gdyż jego optymalna wartość (decydująca o kontraście) zależy (w granicach ok. 1V) od egzemplarza wyświetlacza. Przykład rozwiązania układowego jest podany na schemacie interfejsu sterującego wyświetlaczem (przedstawimy go w EP7/99). Pobór prądu dla poszczególnych napięć wynosi średnio ok. 12mA dla +5V i ok. 1,5mA dla Vee= -10V.

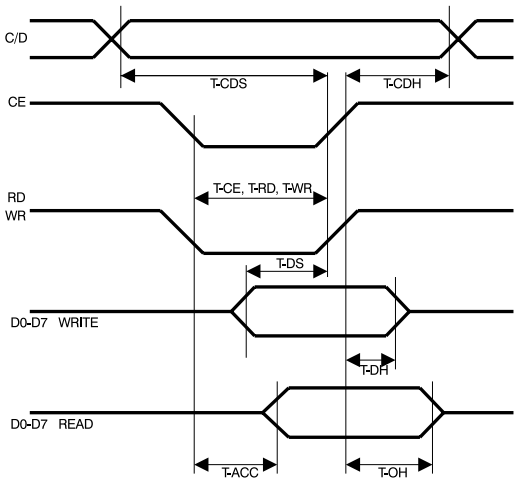
Organizacja procesu wyświetlania

Wyświetlacz dysponuje trybami graficznym i tekstowym, które mogą być niezależnie włączane i wyłączane. W trybie graficznym ekran jest podzielony na 64 linie, a każda linia na 40 zespołów (pattern) po 6 pikseli. Każdy *pattern* odpowiada jednemu bajtowi pamięci, z wykorzystanymi 6 młodszymi bitami. Ustawienie bitu (1) oznacza zapalenie (zaciemnienie) piksela. W trybie tekstowym ekran dzieli się na 8 linii po 40 (font 6x8) lub 30 (font 8x8) znaków. Znaki mogą być pobierane z generatora wbudowanego (CG-ROM) lub ładowanego (CG-RAM). Do dyspozycji mamy też programowo konfigurowany kursor.

Tryby mogą ze sobą współpracować w różnych stylach:

OR - sumowanie - znaki i grafika są wyświetlane niezależnie od siebie,

		CZAS [ns]	
	SYMBOL	MIN.	MAX.
WYPRZEDZENIE USTAWIENIA C/D	T-CDS	100	–
PRZETRZYMANIE C/D	T-CDH	10	–
DŁUGOŚĆ IMPULSU CE	T-CE	80	–
DŁUGOŚĆ IMPULSU WR	T-WR	80	–
DŁUGOŚĆ IMPULSU RD	T-RD	80	–
WYPRZEDZENIE USTAWIENIA DANYCH	T-DS	80	–
PRZETRZYMANIE DANEJ - ZAPIS	T-DH	80	–
CZAS DOSTĘPU - ODCZYT	T-ACC	–	150
PRZETRZYMANIE DANEJ - ODCZYT	T-OH	10	50



Rys. 2. Przebiegi czasowe charakteryzujące interfejs wyświetlacza LM24014H.

FUNKCJA	KOD										
	C/D	RW	WR	D7	D6	D5	D4	D3	D2	D1	D0
KONFIGURACJA PAMIĘCI	1	1	0	0	1	0	0	0	0	A	A
ZAPIS DO REJESTRU	1	1	0	0	0	1	0	0	A	A	A
TRYB WYŚWIETLANIA	1	1	0	1	0	0	0	A	A	A	A
ROZMIAR KURSORA	1	1	0	1	0	1	0	0	A	A	A
ODCZYT/ZAPIS BAJTU DANYCH	1	1	0	1	1	0	0	0	A	A	A
TRYB AUTO	1	1	0	1	0	1	1	0	0	A	A
STYL WYŚWIETLANIA	1	1	0	1	0	0	0	A	A	A	A
ODCZYT EKRANU (PEEK)	1	1	0	1	1	1	0	0	0	0	0
KOPIOWANIE LINII (COPY)	1	1	0	1	1	1	0	1	0	0	0
USTAWIENIE POJEDYNCZEGO BITU	1	1	0	1	1	1	1	A	A	A	A
ODCZYT STATUSU	1	0	1	BAJT STATUSU							
WPIS BAJTU DANYCH	0	1	0	BAJT WPISYWANY							
POBRANIE BAJTU DANYCH	0	0	1	BAJT ODCZYTANY							

A- BITY OKREŚLAJĄCE DZIAŁANIE
DOKŁADNY OPIS W TEKŚCIE

Rys. 3. Format i znaczenie stanów sterujących.

AND - iloczyn - zapalone zostają tylko piksele, które są aktywne w obu trybach,

EXOR - wyłączność - zapalone zostają tylko piksele różniące się w obu trybach,

ATTR - tekst z atrybutami (grafika musi być wyłączona - atrybuty są pobierane z obszaru graficznego pamięci). Wszystkie niezbędne dane są przechowywane w pamięci U6 w wydzielonych obszarach: grafiki, tekstu, generatora CG-RAM, atrybutów tekstu. Adresy tych obszarów są wprowadzone do kontrolera podczas konfiguracji. Każda zmiana zawartości pamięci jest samoczynnie przenoszona na ekran. Ponieważ pamięć ma strukturę liniową - dodatkowo wpisujemy do kontrolera żądaną szerokość pola tekstu lub obrazu (w znakach tekstowych oraz *patternach* graficznych), co pozwala podzielić obszar pamięci na poszczególne wiersze. Lokalizacja poszczególnych obszarów jest dowolna.

Sterowanie wyświetlaczem

Do wyświetlacza wysyłamy komendy konfiguracyjne i sterujące oraz potrzebne dane. Odczytywać możemy status, zawartość pamięci oraz ekranu. W praktyce najbardziej potrzebny jest odczyt statusu, gdyż przed każdym wpisem danej lub komendy do kontrolera musimy sprawdzić czy jest on gotów na jej przyjęcie, o czym świadczy ustawienie odpowiednich bitów w słowie statusu:

STA 0 - gotowość do zapisu instrukcji: 1 - gotów, 0 - zajęty

STA 1 - gotowość do zapisu/odczytu danej: 1 - gotów, 0 - zajęty

STA 2 - gotowość do odczytu danej w trybie AUTO: 1 - gotów, 0 - zajęty

STA 3 - gotowość do zapisu danej w trybie AUTO: 1 - gotów, 0 - zajęty

STA 4 - nieistotny

STA 5 - wskaźnik stanu po zerowaniu (przez ok. 2ms po zerowaniu kontroler stabilizuje zegar i nie wykonuje operacji)

STA 6 - wskaźnik błędu w niektórych operacjach dotyczących obszaru graficznego, błąd (1) jest sygnalizowany przy próbie operacji graficznej pod adresem wykraczającym poza ustawiony obszar graficzny pamięci

STA 7 - wskaźnik migania: 1 - włączony, 0 - wyłączony.

Status odczytujemy przy poziomie wysokim na linii C/D (zestaw sygnałów przy poszczególnych operacjach jest przedstawiony na **rys. 3**).

Komendy mogą być bez argumentów, z argumentem 1-bajtowym lub z argumentem 2-bajtowym. Trochę nietypowy jest sposób wprowadzania argumentów: wpisujemy je w pierwszej kolejności w operacjach zapisu danych, a dopiero potem - w operacji zapisu komendy - wpisujemy kod komendy informując kontroler co ma zrobić z już przesłanymi argumentami.

Dane możemy wprowadzać pojedynczo (komendami zapisu jednego bajtu), dla większych bloków danych należy jednak używać trybu AUTO - kolejne wpisywane bajty danych są w tym trybie przyjmowane bez dodatkowej komendy, z jednoczesną inkrementacją adresu w pamięci RAM.

Przykładowe sekwencje operacji są przedstawione na **rys. 4**. Poniżej zamieszczono dokładne opisy poszczególnych komend.

✓*Konfiguracja pamięci*. Argument dwubajtowy D1, D2, kod 010000AA:

AA=00 - ustawienie adresu początku obszaru tekstowego w pamięci, w D1 młodszy bajt adresu, w D2 starszy bajt adresu;

AA=01 - ustawienie szerokości pola tekstowego (liczby znaków w linii), w D1 liczba kolumn, D2=0;

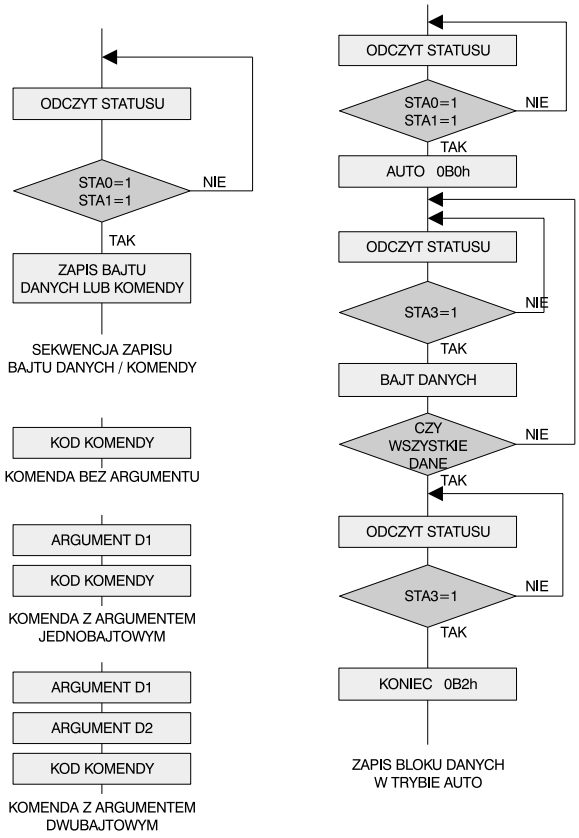
AA=10 - ustawienie adresu początku obszaru graficznego w pamięci, w D1 młodszy bajt adresu, w D2 starszy bajt adresu;

AA=11 - ustawienie szerokości ekranu graficznego (w 6-pikselowych *patternach*) w D1 liczba kolumn, D2=0.

✓*Zapis do wewnętrznych rejestrów*. Argument dwubajtowy D1, D2, kod 00100AAA:

AAA=001 - ustawienie pozycji kursora, w D1 współrzędna pozioma x w zakresie 0..127 (7 bitów), w D2 współrzędna pionowa y (od góry) w zakresie 0..31 (5 bitów);

AAA=010 - ustawienie offsetu ładowalnego generatora znaków (CG-RAM), w D1 offset w zakresie 0..31 (5 bitów), D2=0;



Rys. 4. Algorytmy sterowania wyświetlaczem LM24014H.

Rys. 5. Okno programu testowego.

AAA=100 - ustawienie wskaźnika adresowego, w D1 młodszy bajt adresu, w D2 starszy bajt adresu - następnie wykonywane operacje zapisu lub odczytu będą dotyczyć tego adresu. Dodatkowego wyjaśnienia wymaga offset CG-RAM. Struktura generatora znaków jest następująca:

- ✗ opis każdego znaku jest zawarty w 8-bajtowym obszarze pamięci (kolejne bajty opisują kolejne wiersze znaku),
- ✗ kod znaku podany w obszarze tekstowym ekranu (0..$ff) wskazuje na odpowiedni opis znaku - do ulokowania pełnej tablicy znaków potrzebujemy więc 8x256=2048 bajtów (2kB), których zaadresowanie wymaga 10 bitów.

Offset CG-RAM określa lokację 2 kB generatora w 8 kB pamięci - inaczej mówiąc jest najstarszymi, począwszy od 11, bitami adresu. Na przykład *offset=1* ustawia początek obszaru generatora na $800 (rzecz jasna należy zadbać, aby generator mieścił się w granicach obszaru pamięci!).

✓ *Ustawianie trybu wyświetlania*. Komenda bez argumentu, kod 1001AAAA. W tej komendzie bity 0..3 są użyte jako flagi, czyli mogą być ustawiane niezależnie. Ustawienie (1) oznacza włączenie przypisanego danej fladze trybu.

Bit 0 - miganie kursora,
Bit 1 - kursor widoczny,
Bit 2 - ekran tekstowy widoczny,
Bit 3 - ekran graficzny widoczny.

✓ *Ustawienie rozmiaru kursora*. Komenda bez argumentu, kod 10100AAA:
AAA+1= wysokość kursora (np. AAA=000 - jedna linia od dołu, AAA=111 - kursor na cały obszar znaku).

✓ *Odczyt/zapis bajtu danych*. Komenda bez argumentu przy odczycie, natomiast z argumentem jednobajtowym D1 przy zapisie, kod 11000AAA.
Mechanizm używania jest następujący: najpierw należy ustawić wskaźnik adresu, przy zapisie - wpisać daną D1 i komendę z kodem do zapisu, kontroler ulokuje D1 pod podanym adresem. Przy odczycie - wpisać komendę z kodem do odczytu (kontroler przygotuje zawartość podanego wcześniej adresu) i odczytać daną.
AAA=000 - zapis danej z inkrementacją wskaźnika adresowego,
AAA=001 - odczyt danej z inkrementacją wskaźnika adresowego,
AAA=010 - zapis danej z dekrementacją wskaźnika adresowego,
AAA=011 - odczyt danej z dekrementacją wskaźnika adresowego,
*AAA=1*0* - zapis danej bez zmiany wskaźnika adresowego,
*AAA=1*1* - odczyt danej bez zmiany wskaźnika adresowego (* - stan dowolny).

✓ *Tryb AUTO*. Komenda bez argumentu, kod 101100AA.
Pozwala na załadowanie lub odczyt kolejno większej liczby bajtów danych bez każdorazowego wywoływania poprzedniej komendy dotyczącej jednego bajtu. Schemat użycia do zapisu jest pokazany na rys. 4. Należy pamiętać, że przy włączonym trybie AUTO gotowość kontrolera określają oddzielne bity słowa statusu (STA2 dla odczytu, STA3 dla zapisu).
AA=00 - włączenie trybu AUTO do zapisu,
AA=01 - włączenie trybu AUTO do odczytu,
*AA=1** - wyłączenie trybu AUTO (* - stan dowolny).

✓ *Ustawianie stylu wyświetlania*. Komenda bez argumentu, kod 1000AAAA.
Bity 0,1,2 stanowią przełącznik stylu:
AAA=000 - OR (suma grafiki i tekstu),
AAA=001 - EXOR (wyświetlanie różnicowe grafiki i tekstu),
AAA=011 - AND (iloczyn grafiki i tekstu),
AAA=100 - tekst z atrybutami.

Bit 3 stanowi niezależną flagę wyboru generatora znaków: 1 - ładowany CG-RAM, 0 - wbudowany CG-ROM. Przy włączeniu atrybutów tekstu należy kody atrybutów ulokować w obszarze graficznym pamięci. Jednocześnie oznacza to, że tryb graficzny jest przy użyciu atrybutów nieaktywny. Można albo przeładować obszar graficzny kodami atrybutów, albo wydzielić dla atrybutów oddzielny blok pamięci i odpowiednio przestawić adres obszaru graficznego.

Kody atrybutów wykorzystują 4 najmłodsze bity:
0000 - znak normalny,
0011 - znak wygaszony,
1000 - znak migający.

W dokumentacji sterownika podano także 0101 - inwersja, ale nie wiedzieć czemu nie udało mi się tego uzyskać. Generator CG-ROM pomija znaki sterujące - zaczyna się od spacji $20. Należy więc odpowiednio przeliczyć kody ASCII przy ładowaniu tekstu. Jeśli chcemy używać własnych znaków musimy niestety załadować wszystko - nie można korzystać częściowo z ROM i RAM. Ułatwieniem jest zamieszczony w kodzie źródłowym przykładowy generator z polskimi znakami.

✓ *Zapalanie indywidualnego bitu*. Komenda bez argumentu, kod 1111AAAA.

Bity 0,1,2 określają pozycję bitu w bajcie (wskazywanym przez zawartość wskaźnika adresowego). Bit 3 oznacza: 1 - zapalenie piksela, 0 - zgaszenie piksela.

Typowe operacje

Po zerowaniu lub włączeniu zasilania należy przeprowadzić inicjalizację. Należy zwłaszcza zwrócić uwagę na fakt, że zerowanie nie zmienia rejestru stylu wyświetlania - brak jawnego ustawienia stylu powoduje zagadkowy brak startu wyświetlacza. Sekwencja operacji wygląda np. następująco:

1. Zerowanie i ustawienie:
- stylu, np. OR + CG-RAM,
- adresu pola grafiki,
- szerokości ekranu graficznego,
- adresu pola tekstowego,
- szerokości ekranu tekstowego,
- offsetu CG-RAM,
2. Załadowanie generatora znaków.
3. Zerowanie pamięci tekstowej.
4. Załadowanie pamięci graficznej.
5. Włączenie trybu graficznego.

Ładowanie bloku danych najlepiej - jak już wspomniałem - wykonać z użyciem trybu AUTO (patrz też rys. 4):

1. Ustawienie wskaźnika adresowego na początek ładowanego obszaru.
2. Włączenie trybu AUTO do zapisu.
3. Wpisanie odpowiedniej liczby danych.
4. Wyłączenie trybu AUTO.

Powyższy opis nie objął mniej przydatnych w praktycznym wykorzystaniu wyświetlacza funkcji odczytywania zawartości ekranu, kopiowania linii, czy też używania okna przewijanego - pozostawiam to Czytelnikom do ewentualnych własnych eksperymentów.

Rzecz jasna, w rzeczywistych aplikacjach konieczne jest przygotowanie sobie szeregu - zależnie od potrzeb - funkcji graficznych, np. rysujących linie, ramki, wykresy itp. Bardzo efektownym przykładem takiej profesjonalnej biblioteki jest Window Server w palmtopach Psion. We własnych projektach jednakże na ogół wystarczą znacznie prostsze rozwiązania. W szczególności często możemy mieć do czynienia z ładowaniem pełnoekranowej grafiki (logo firmowe, tło itd.). Do przygotowania grafiki w formacie zgodnym z potrzebami wyświetlacza służy specjalnie napisany program. Pracuje on w środowisku Windows i pozwala na przekodowanie typowych plików *.bmp. Dodatkowo można z jego pomocą wygenerować gotowy tekstowy plik assemblerowy do dołączenia do oprogramowania mikrokontrolera sterującego wyświetlaczem. Wygląd okna programu przedstawia **rys. 5**.

Jerzy Szczesiul,
jerzy.szczesiul@ep.com.pl