# Alfanumeryczne wyświetlacze LCD

## część III

*Inteligentne wyświetlacze alfanumeryczne LCD są elementem coraz częściej spotykanym w sprzęcie powszechnego użytku: od urządzeń gospodarstwa domowego, poprzez systemy alarmowe, na medycznym sprzęcie profesjonalnym skończywszy. Dla wielu elektroników amatorów, chcących wykorzystać te efektowne elementy we własnych konstrukcjach są one często tylko marzeniem. W wielu przypadkach powodem takiego stanu rzeczy nie jest bynajmniej ich cena, lecz brak wiedzy o praktycznych sposobach na zmuszenie tego elementu do wyświetlania „tego co akurat w danej chwili chcemy...”*
*W niniejszym artykule autor w przystępny sposób stara się zapoznać czytelników z tymi elementami, budzącymi często podziw i westchnienie niejednego początkującego konstruktora.*

### Praca ze sterowaniem 4-bitowym

Jak już wiesz, do komunikacji z modułem oprócz sygnałów sterujących (RW,RS,E) służy osiem linii danych: D0...D7. Jednak każdy wyświetlacz tekstowy ze sterownikiem HD44780 może także pracować będąc sterowanym tylko za pomocą 4 linii danych – D7...D4. W takim przypadku linie D3...D0 są ignorowane i mogą w związku z tym pozostać nie podłączone. W tabeli 2 pokazana jest zresztą instrukcja „Function Set” w której bit D4 odpowiada za ustalenie, jaki rodzaj komunikacji wybierze użytkownik po włączeniu zasilania modułu. Jeżeli bit ten ustawimy, moduł komunikować się będzie poprzez 8-bitową szynę danych, jeżeli zaś wyzerujemy go, komunikacja będzie 4-bitowa. W tym ostatnim przypadku, aby przesłać kompletną instrukcję lub daną do wyświetlacza (a także ją odczytać), należy wykonać to w dwóch etapach. W pierwszym należy przesłać starszy półbajt instrukcji lub danej, a następnie młodszy. Tak wiec np. aby wyświetlić literę „S” (kod: 01010011) trzeba najpierw podać na linie D7...D4 kombinację pierwszych czterech bitów czyli: „0101”, a następnie wysłać drugą połowę: „0011”. Oczywiście przed każdą połówką należy odpowiednio ustawić sygnały RS i RW oraz uaktywnić transmisję sygnałem E (także za każdym razem). Różnica polega tylko na tym, że po przesłaniu pierwszego półbajtu nie jest potrzebne sprawdzanie flagi zajętości, lub czekanie przez, podany w tabeli 2, czas, w którym moduł wykonuje wewnętrzną operację. Dzięki temu ogólny czas obsługi interfejsu 4-bitowego, szczególnie w układach mikroprocesorowych, nie jest o wiele dłuższy niż przy pełnym 8-bitowym interfejsie.

Pod koniec opisu modułów LCD pokażę przykładowy sposób podłączenia wyświetlacza poprzez interfejs zarówno 8 jak i 4-bitowy.

### Inicjalizacja – czyli co się dzieje po włączeniu zasilania

Jak każde inteligentne urządzenie moduły LCD, po dołączeniu do nich zasilania, przez jakiś określony czas wykonują automatyczną procedurę inicjalizacji swych „wnętrzności”.

W przypadku modułów z HD44780 czas ten wynosi z reguły 10 ms. Moduł realizuje funkcję „zerowania” dzięki wbudowanemu w moduł układowi „reset”. Podczas tego okresu wyświetlacz nie przyjmuje żadnych rozkazów, toteż w układach sterujących warto przewidzieć możliwość opóźnienia o wspomniany czas (najlepiej z zapasem: 20ms) po ustabilizowaniu się napięcia zasilania na poziomie 4,5V lub więcej. W przypadku kiedy po włączeniu zasilania czas narastania napięcia spełnia wymogi modułu, czyli zawiera się w granicach 0,1ms...10ms przy wzroście od 0,2V do 4,5V, układ resetu pracuje poprawnie zapewniając prawidłową inicjalizację wyświetlacza. W takim przypadku wartości początkowe rejestrów wewnętrznych są następujące:
– panel LCD wygaszony (“display clear”)
– nastawy „Function Set”:
DL=1 : 8-bitowa szyna danych,
N=0 : wyświetlanie w trybie 1 liniowym,
F=0 : matryca znaku 5x7 punktów;
– nastawy „Display ON/OFF”:
D=0 : wyświetlacz wygaszony,
C=0: kursor wygaszony,
B=0 wyłączona funkcja migania kursora lub znaku
– nastawy: „Entry Mode Set”:
I/D=1 : +1 (zwiększanie adresu),
S=0: bez przesuwania
– pamięć wyświetlania DD RAM jest wypełniona spacjami, zaś pamięć CG RAM jest wypełniona przypadkowymi danymi.

Rozpoczęcie pracy w takim przypadku może się odbyć w sposób normalny, czyli np. jak podano w przykładzie z tabeli 4.

Jeżeli zaś czas narastania napięcia po włączenia zasilania modułu nie mieści się w podanych wcześniej granicach, niezbędne jest wykonanie procedury inicjującej przez układ sterujący użytkownika. Kolejne kroki takiej operacji dla interfejsu 4 i 8-bitowego podane są na rysunku 8 i 9.

W praktyce przypadek kiedy zasilanie nie spełnia wymogów umożliwiających prawidłową inicjalizację modułu jest bardzo rzadki, szczególnie gdy w zasilaczu pracuje monolityczny stabilizator napięcia np. 7805, a kondensator filtrujący zasilanie po stronie wtórnej stabilizatora nie jest zbyt duży (47uF...220uF). W każdym razie w przypadku kiedy moduł odmówi „posłuszeństwa” po załączeniu zasilania, pomimo wprowadzania komend i instrukcji w celu wyświetlenia czegoś na displeju LCD, należy zastosować programową procedurę inicjalizacyjną, taką jak omówiona wcześniej.

### Struktura wewnętrznych pamięci modułu

Jak wspomniałem wcześniej, moduł posiada trzy rodzaje wewnętrznej pamięci, powtórzmy je sobie:

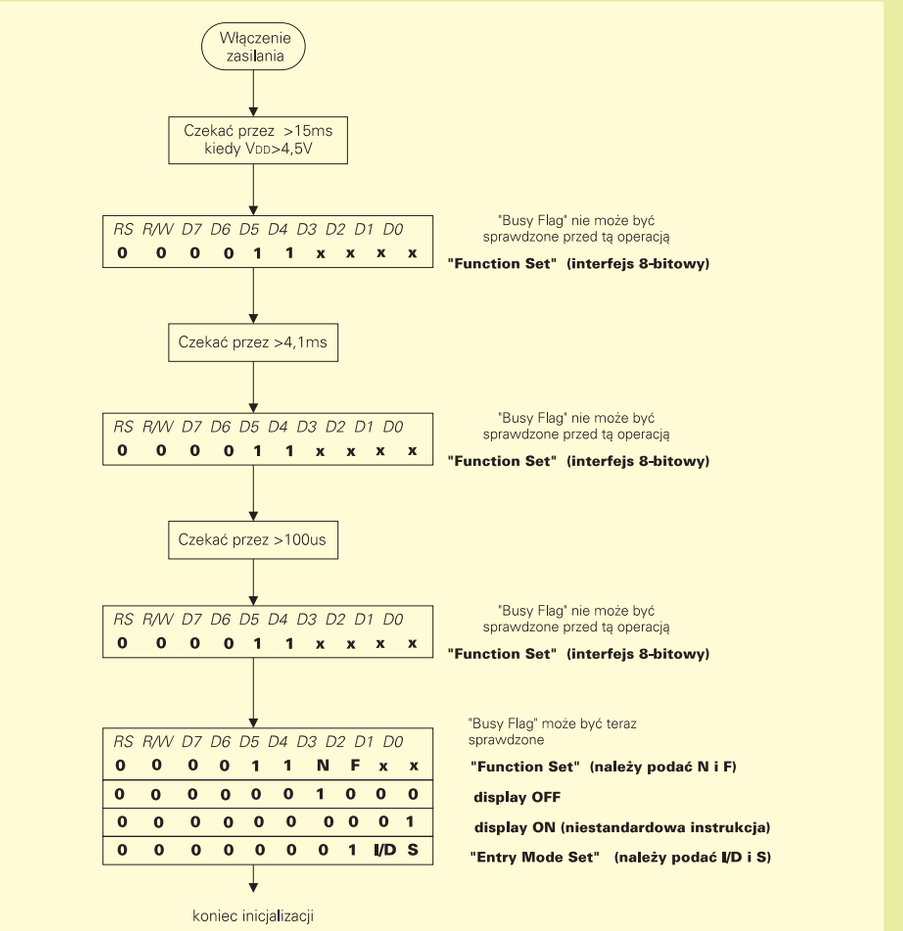
– CG ROM : pamięć generatora znaków, jest to pamięć stała (ROM) z zapisanymi danymi, na podstawie których wyświetlany jest jeden ze 160 znaków. Zawartość CG ROM podałem wcześniej w tabeli na rysunku 4.
– DD RAM : pamięć typu RAM o pojemności 80 znaków (bajtów) przechowująca kody znaków do wyświetlenia na panelu LCD. Pamięć ta może być zapisywane lub odczytywana przez zewnętrzny układ sterujący.
– CG RAM : pamięć typu RAM o pojemności 64 bajtów przeznaczona na definiowanie znaków przez użytkownika.

Poniżej zapoznam cię dokładnie ze strukturą tych pamięci i ich znaczeniem dla pracy modułu.

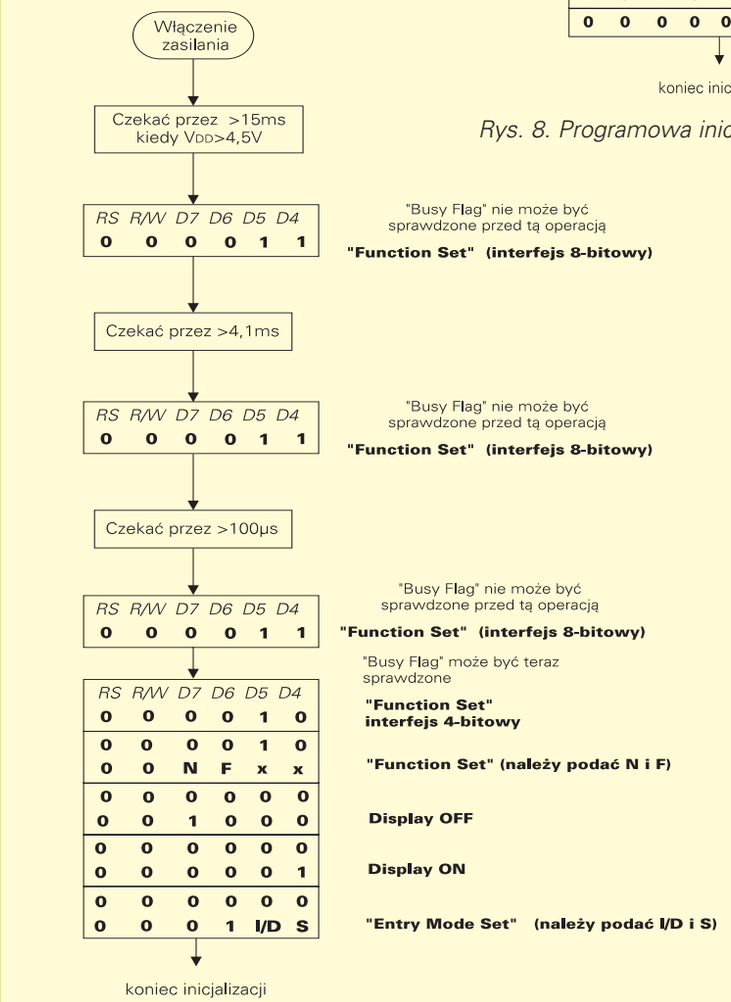
**Pamięć CG ROM.** Znaczenie tej pamięci zostało już wyjaśnione, a struktura została przedstawiona na rysunku 4. W artykule nie podano wyglądu znaków kiedy moduł pracuje w trybie 5x10 punktów. Po pierwsze dlatego, że nie wszystkie moduły mogą pracować w tym trybie, po drugie: różnica w wyglądzie znaków z matrycy 5x7 i 5x10 jest niewielka, różni się tylko ostatnie 16 znaków o kodach: 240...255. Znaki te nie wchodzą w skład alfabetu łacińskiego, a są symbolami z innych zbiorów znaków. Jeżeli kogoś interesuje zawartość i wygląd tych znaków, powinien sprawdzić to w praktyce, kupując wyświetlacz, ewentualnie przestudiować literaturę [1]. Pamięć CG ROM jest adresowana (tak właściwie to jest adresowany cały blok bajtów opisujący konkretny znak) poprzez wpisanie kodu znaku do wyświetlenia przy RS=1 i RW=0. Nie jest możliwa ingerencja i modyfikacja tej pamięci, ani nie jest możliwe dołączenie np. zewnętrznej pamięci tego typu.

Na rynku elektronicznym istnieją wersje wyświetlaczy tekstowych LCD zgodnych programowo z opisywanym tu standardem, lecz zawartość pamięci CG ROM poszczególnych modeli w zakresie kodów: 128...255 może się nieco różnić. Są to jednak znaki spoza naszego alfabetu, toteż nie stanowi to dużego problemu dla użytkownika, szczególnie że może on zawsze zdefiniować swój nietypowy znak korzystając z pamięci CG RAM.

**Pamięć DD RAM.** Pamięć ta służy do przechowywania kodów znaków do wyświetlenia, jej pojemność dla każdego modułu opartego o sterownik HD44780 wynosi 80 bajtów (znaków). Jak już wiesz, w przypadku kiedy panel LCD wy-




*Rys. 8. Programowa inicjalizacja modułu z interfejsem 8-bitowym*




*Rys. 9. Programowa inicjalizacja modułu z interfejsem 4-bitowym*

świetlacza ma mniejsza „długość" , wyświetlana jest tylko część tekstu, zawartego w DD RAM. Pozostała część pamięci może być wykorzystana przez programistę jako pamięć RAM dowolnego wykorzystania. W praktyce jednak zapisuje się do niej maksymalnie dużo komunikatów (które mają być wyświetlone), by potem w razie potrzeby, poprzez zwykłe przesunięcie tekstu (poprzez komendy przesuwające) pokazać żądany tekst. Innym sposobem, szczególnie przydatnym dla początkujących w dziedzinie obsługi wyświetlaczy tego typu, jest zapis do pamięci DD RAM tylko tej części tekstu, która może być wyświetlona. Jeżeli potrzeba pokazać coś innego, wystarczy poleceniem „clear display" wyczyścić pamięć DD RAM a potem zapisać nowy, żądany tekst.

Kolejne komórki DD RAM są oczywiście ponumerowane, w tym przypadku będzie to zakres 0...79 (80 znaków), w zapisie szesnastkowym: 00h...4Fh. Takim zapisem będziemy się posługiwać w dalszej części opisu pamięci wyświetlacza.

Dla trybu jednoliniowego, kiedy w instrukcji „Function Set" (patrz tabela 2) bit N=0 organizacja pamięci DD RAM jest następująca:

| 1 | 2 | 3 | 4 | 5 | ... | 79 | 80 | ← pozycja na displeju |
|---|---|---|---|---|---|---|---|---|
| 00 | 01 | 02 | 03 | 04 | ..... | 4E | 4F | ← adres w DD RAM |

W przypadku, gdy wyświetlacz posiada panel LCD o mniejszej (<80) liczbie pozycji, wyświetlana jest tylko część tekstu, np. dla panelu 1x8 mamy:

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | ← pozycja na displeju |
|---|---|---|---|---|---|---|---|---|
| 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | ← adres w DD RAM |

Podczas wykonania operacji przesuwania tekstu adres podąża jak pokazano poniżej:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | ← przesuw w lewo |
| 4F | 00 | 01 | 02 | 03 | 04 | 05 | 06 | ← przesuw w prawo |

Dla przykładu struktura pamięci 16-znakowego wyświetlacza (1x16) ma postać:

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | ← pozycja na displeju |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | ← adres w DD RAM |

Przy przesuwaniu zawartość pamięci wygląda następująco:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | 10 | ← przesuw w lewo |
| 4F | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | ← przesuw w prawo |

W trybie dwuliniowym, kiedy w instrukcji „Function Set" (patrz tabela 2) bit N=1 organizacja pamięci DD RAM jest nieco inna

| | 1 | 2 | 3 | 4 | 5 | ... | 39 | 40 | ← pozycja na displeju |
|---|---|---|---|---|---|---|---|---|---|
| 1 linia | 00 | 01 | 02 | 03 | 04 | ..... | 26 | 27 | ← adres w DD RAM |
| 2 linia | 40 | 41 | 42 | 43 | 44 | | 66 | 67 | ← adres w DD RAM |

Jak widać w tym przypadku pamięć DD RAM jest podzielona na dwie części, po 40 bajtów pojemności każda. Pierwsza część (linia) zawiera adresy 00...27h (0...39), druga zaś adresy: 40h...67h (64...103).

Kiedy wyświetlacz ma mniej (np. 2x8) znaków w linii niż 40, adresy są oczywiście ułożone w następujący sposób:

W przypadku przesuwania zawartości DD RAM w trybie 2-liniowym adresy w liniach 1 i 2 wyglądają następująco:

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | ← pozycja na displeju |
|---|---|---|---|---|---|---|---|---|---|
| 1 linia | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | ← adres w DD RAM |
| 2 linia | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | ← adres w DD RAM |

przesuw w lewo

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 linia | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 |
| 2 linia | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 |

przesuw w prawo

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 linia | 27 | 00 | 01 | 02 | 03 | 04 | 05 | 06 |
| 2 linia | 67 | 40 | 41 | 42 | 43 | 44 | 45 | 46 |

Dla przykładu w najbardziej popularnym wyświetlaczu LCD 2x16 znaków struktura „widocznej" DD RAM jest następująca:

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | ← pozycja na displeju |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 linia | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | ← adres w DD RAM |
| 2 linia | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 4A | 4B | 4C | 4D | 4E | 4F | ← adres w DD RAM |

Kiedy realizowane jest przesuwanie, struktura DD RAM przyjmuje postać:

przesuw w lewo

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 linia | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | 10 |
| 2 linia | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 4A | 4B | 4C | 4D | 4E | 4F | 50 |

przesuw w lewo

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 linia | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | 10 |
| 2 linia | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 4A | 4B | 4C | 4D | 4E | 4F | 50 |

F

Jak widać, w trybie 2-liniowym, przy przesuwaniu tekst przesuwany jest w obrębie każdej linii osobno, można powiedzieć, że „zapętla" się w obszarze jednej linii. Ta właściwość jest bardzo przydatna szczególnie przy demonstracyjnym przesuwaniu komunikatów wieloliniowych.

Istnieją modele wyświetlaczy 1x16 znaków np. Hitachi typ LM020, w którym pomimo 1 linii wyświetlającej 16 znaków przyporządkowanie poszczególnych matryc na panelu LCD do adresów w DD RAM jest nieco inne:

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | ← pozycja na displeju |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | ← adres w DD RAM |

Dla tego modelu LCD podczas inicjalizacji powinna być wpisana instrukcja „Function set" z bitem N=1, czyli praca w trybie 2-liniowym, inaczej aktywna będzie tylko połowa (8 znaków) wyświetlacza.

Dla większych modułów, wieloliniowych modułów LCD układ adresów w zależności od pozycji znaku i linii jest następujący:

Moduł 2×40 (2 linie po 40 znaków), np. LM107, LM018, LM092

| | | |
|---|---|---|
| 1 linia | A | B |
| 2 linia | C | D |

Moduł 4×40 (4 linie po 20 znaków), np. LM044

| | |
|---|---|
| 1 linia | A |
| 2 linia | C |
| 1 linia | B |
| 2 linia | D |

gdzie: oznaczenia A, B, C, D to następujące kolejne adresy:
A: 00h...13h
B: 40h...53h
C: 14h...27h
D: 54h...67h

Ważną informacją jest to, że każdorazowe wpisanie danej do pamięci CG RAM powoduje automatyczną inkrementację licznika adresu. Dzięki temu nie jest konieczne ustawianie go za każdym razem przed wpisanie kolejnej litery tekstu.

Po tej sporej liczbie przykładów warto poeksperymentować z nawet najtańszym, zakupionym modułem tekstowym LCD. Zapewniam cię, że każdy zgodny z omawianym standardem, tak czy inaczej, da się „opanować" nawet przez mało doświadczonego w tej materii elektronika.

**Pamięć CG RAM.** Przyszła wreszcie pora na przedstawienie zasad wprowadzania zdefiniowanych przez użytkownika zna-

ków. Do tego celu służy właśnie CG RAM. Jak już wiesz, jej pojemność to 64 bajty, co pozwala na zdefiniowanie 8 znaków, każdy opisany jest przez 8 bajtów, a każdy z tych bajtów opisuje 1 wiersz matrycy pojedynczego znaku. Skoro powiedziałem wcześniej, że matryca znaku ma postać w tym wypadku 5x7, to po co ten ósmy bajt? – a no po nic. Jest on po prostu „marnowany". Tak samo jest z bitami w danym bajcie, opisującym wiersz, istotne jest tylko 5 najmłodszych bitów, bo przecież matryca ma 5 kolumn. Jeżeli dwa ostatnie zdania nie są zbyt jasne, pomoże ci z pewnością **rysunek 10**.

Teraz już chyba powinieneś wiedzieć dlaczego powiedziałem wcześniej że część bitów jest „marnowana". Jest to jednak tylko pozorne marnotrawstwo, bowiem dzięki takiemu ułożeniu informacji w pamięci CG RAM możliwe jest logiczne i kolejne adresowanie definiowanych komórek. W **tabeli 5** pokazano strukturę pamięci CG RAM.

Dla przykładu jako pierwszy znak (kod=00h) zdefiniowano „szachownicę", pod kodem 02h znak „karo", ostatnim zdefiniowanym znakiem są cztery poziome linie. Bity oznaczone gwiazdką nie mają znaczenia przy definiowaniu matrycy znaku.

Ósmy wprowadzany kolejno bajt (wiersz) matrycy znaku nie jest istotny, ze względu na zastosowane matryce LED: 5x7, zwykle należy wpisać 00h.

Tak więc, aby zdefiniować np. 4 nowe znaki, należy najpierw wykonać instrukcję ustawienia adresu w CG RAM na pozycję 0 (patrz tabela 2): „Set CG RAM Address" i

| RS | RW | D7...D0 |
|---|---|---|
| 0 | 0 | 01000000 |

Następnie należy kolejno wpisywać wszystkie bajty, po kolei od pierwszego, opisującego pierwszy (górny) wiersz matrycy pierwszego znaku, aż do ostatniego 31-ego (32-gi jest i tak równy „0" więc nie ma potrzeby go wpisywać) bajtu, opisującego ostatni wiersz 4-tej matrycy definiowanego znaku.

Tak samo jak w przypadku pamięci DD RAM po każdym wpisie danej do tej pamięci licznik adresu jest automatycznie inkrementowany, toteż nie jest potrzebne ustawianie adresu za każdym razem za pomocą instrukcji „Set CG RAM Address".

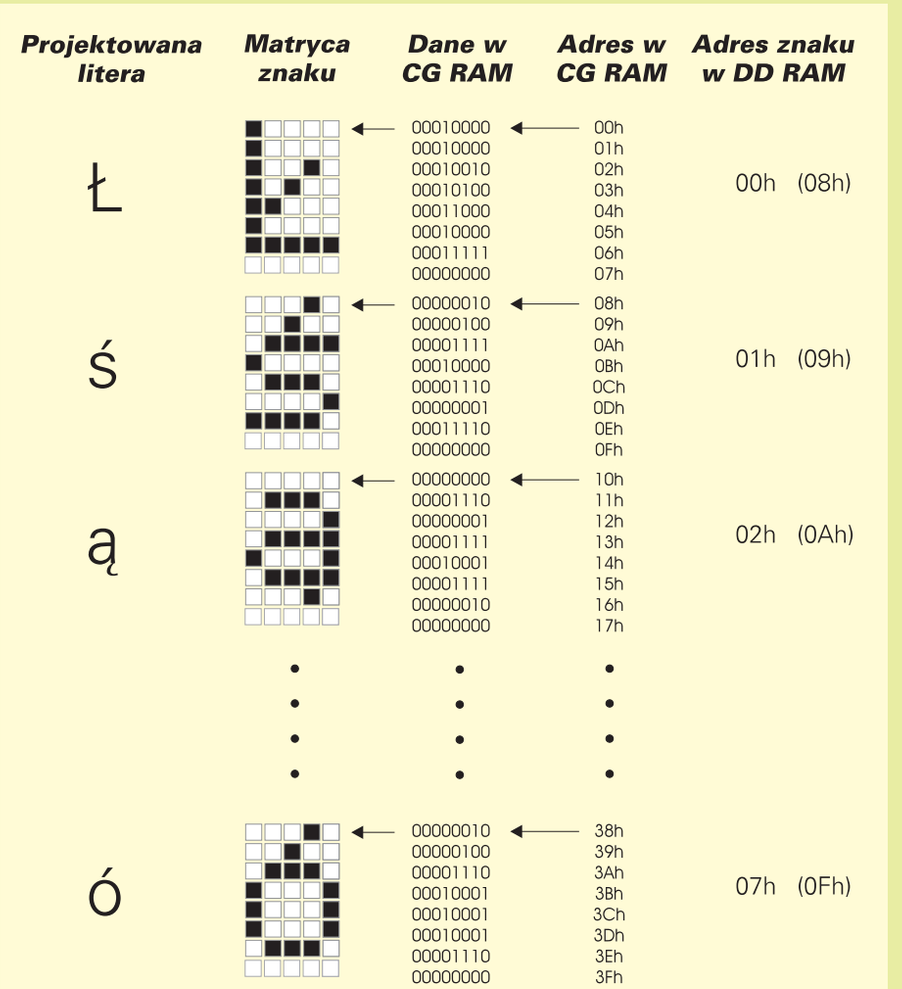
Obok w **tabeli 6** przedstawiam 2 komplety danych (po 64 bajty), umożliwiających zdefiniowanie 16 polskich znaków, w pierwszym przypadku są to litery małe: „ąćęłńóśż", w drugim duże: „ĄĆĘŁŃÓŚŻ".

Dzięki tym danym nie będziesz musiał samodzielnie męczyć się z rysowaniem „kropka po kropce" każdej polskiej litery.

Oczywiście po zdefiniowaniu wszystkich 8 znaków ich kolejne kody będą zgodne z tabelą na rysunku 4, czyli np. znak „ą" będzie miał kod 00h, znak „ć" kod 01, itd., aż do kodu 07.

*Ciąg dalszy w następnym numerze EdW.*

**Sławomir Surowiński**




*Rys. 10. Matryca definiowanego znaku, a bajty ją opisujące*

*Tabela 5*

| Kod znaku (tab. 1) bity: *76543210* | | Adres w CG RAM bity: *543* | *210* | | Matryca znaku bity: *765* | *43210* |
|---|---|---|---|---|---|---|
| 0000*000<br>kod: 00h lub 08h | | 000 | 000<br>001<br>010<br>011<br>100<br>101<br>110<br>111 | | ***<br>***<br>***<br>***<br>***<br>***<br>***<br>*** | 01010<br>10101<br>01010<br>10101<br>01010<br>10101<br>01010<br>00000 |
| 0000*001<br>kod: 01h lub 09h | | 001 | 000<br>001<br>010<br>011<br>100<br>101<br>110<br>111 | | ***<br>***<br>***<br>***<br>***<br>***<br>***<br>*** | 00000<br>00100<br>01110<br>11111<br>01110<br>00100<br>00000<br>00000 |
| itd...aż do kodu 07h | | | | | | |
| 0000*111<br>kod: 07h lub 0Fh | | 111 | 000<br>001<br>010<br>011<br>100<br>101<br>110<br>111 | | ***<br>***<br>***<br>***<br>***<br>***<br>***<br>*** | 11111<br>00000<br>11111<br>00000<br>11111<br>00000<br>11111<br>00000 |

*Tabela 6*

| polski znak | seria 8 kolejnych bajtów (w zapisie dziesiętnym) |
|---|---|
| **ą** | 0, 14, 1, 15, 17, 15, 2, 0 |
| **ć** | 0, 2, 4, 14, 16, 16, 17, 14 |
| **ę** | 0, 14, 17, 31, 16, 14, 2, 0 |
| **ł** | 12, 4, 6, 4, 12, 4, 14, 0 |
| **ń** | 2, 4, 22, 25, 17, 17, 17, 0 |
| **ó** | 2, 4, 14, 17, 17, 17, 14, 0 |
| **ś** | 2, 4, 14, 16, 14, 1, 30, 0 |
| **ż** | 2, 4, 31, 2, 4, 8, 31, 0 |

| polski znak | seria 8 kolejnych bajtów (w zapisie dziesiętnym) |
|---|---|
| **Ą** | 14, 17, 17, 31, 17, 17, 19, 0 |
| **Ć** | 2, 14, 21, 16, 16, 17, 14, 0 |
| **Ę** | 31, 16, 30, 16, 16, 31, 2, 0 |
| **Ł** | 16, 16, 20, 24, 16, 16, 31, 0 |
| **Ń** | 2, 21, 25, 21, 19, 17, 17, 0 |
| **Ó** | 2, 14, 21, 17, 17, 17, 14, 0 |
| **Ś** | 2, 14, 16, 14, 1, 1, 30, 0 |
| **Ż** | 31, 1, 2, 14, 8, 16, 31, 0 |