Procedury obsługi wyświetlaczy LCD, część 2

W drugiej i ostatniej części artykułu przedstawiamy procedury obsługi alfanumerycznego wyświetlacza LCD przez dowolny mikrokontroler z rodziny '51.

Procedury obsługi dla MCS51

Do projektów sterowników dołączone są procedury w języku assemblera mikroprocesora 8051. Są one dostępne w postaci źródłowej na stronie internetowej pod adresem: **http://www.ep.com.pl/ftp_lcd_prakt.exe** oraz na płycie CD-EP3/2000. Ze względu na ich dużą objętość nie publikujemy tu pełnego listingu, a ograniczamy się jedynie do procedur.

Procedury realizują następujące funkcje:

InitLCD - inicjacja wyświetlacza i ustawienie jego trybu pracy.

DefChars - pobiera z pamięci programu definicje znaków od adresu Chars do końca definicji określonego jako cyfra 0.

CheckBusyFlag - testuje stan zajętości wyświetlacza i czeka na gotowość do transmisji danych.

PutCharOnLCD - wyświetla znak, o kodzie zapisanym w rejestrze R3, na ekranie LCD na współrzędnych R1=numer wiersza (0..3), R2=numer kolumny (0..19).

WriteCString - wyświetla łańcuch zdefiniowany w pamięci programu; R1=numer wiersza, R2=numer kolumny, DPTR=adres początku łańcucha znaków do wyświetlenia; łańcuch musi być zakończony cyfrą 0.

WriteAString - wyświetla łańcuch zapisany w pamięci RAM; R1=numer wiersza, R2=numer kolumny, R0=adres początku łańcucha znaków; łańcuch musi być zakończony cyfrą 0.

Hex2DecConv - dokonuje konwersji liczby 2-bajtowej zawartej w rejestrach R6 (MSB) i R7 (LSB) na postać dziesiętną; wynik zwracany jest w rejestrach R5, R6 i R7; R5, bity 0..3=najstarsza cyfra liczby dziesiętnej, R7, bity 0..3=najmłodsza.

Dec2AsciiConv - dokonuje konwersji liczby dziesiętnej na postać ASCII i zapisuje ją, w formacie łańcucha gotowego do wyświetlenia, w pamięci RAM w zmiennej LCD_1; R5, R6, R7=liczba dziesiętna (wprost po Hex2DecConv).

ClearLCD - kasowanie wyświetlacza.

Opisu wymaga procedura *PutCharOnLCD*. Działa ona w sposób wolny i może mało elegancki, jednak jest skuteczna i można ją z powodzeniem stosować do różnych modeli wyświetlaczy. W pierwszym kroku wykonywany jest powrót do pozycji 0,0 kursora (ang. HOME), a następnie za pomocą rozkazu powodującego jego przesuwanie przenoszony jest do miejsca, gdzie należy umieścić znak. W czasie przesuwania kursora uwzględniany jest przeplot. Dzieje się tak przy każdym wyświetlanym znaku. Obawiałem się, że wyświetlanie znaków będzie zbyt wolne. Obawy okazały się jednak bezpodstawne. Dużo bardziej elegenckie byłoby co prawda ustawienie adresu pamięci obrazu i zapis znaku pod tenże adres, jednak przy tak małej liczbie znaków do wyświetlenia i dodatkowo jeszcze dużej, tak zwanej bezwładności wyświetlacza LCD, jest to problem drugorzędny. Wyświetlacz i tak nie nadąża ze zmianami obrazu i często procedury wymagają dodatkowych pętli opóźniających.

Procedura zapisująca znaki do wyświetlacza działa asynchronicznie, to znaczy następne dane wysyłane są nie cyklicznie co pewien czas, lecz po zgłoszeniu przez kontroler wyświetlacza gotowości. Oprócz propozycji procedur obsługi wyświetlacza dodatkowo dołączyłem procedury konwersji liczb szesnastkowych - dwubajtowych z zakresu od 0..FFFFH na liczby dziesiętne oraz zamiany liczb dziesiętnych na łańcuch kodów ASCII gotowy do wyświetlenia. Kolejność użycia procedur jest następująca:

```
MOV R6,#LSB
MOV R5,#MSB       ;do R5 i R6 liczba
                  ;szesnastkowa
CALL Hex2DecConv  ;zamiana liczby hex
                  ;na dziesiętną
CALL Dec2AsciiConv
                  ;zamiana liczby
                  ;dziesiętnej na ascii
MOV R1,#1
MOV R2,#0         ;np. 2 wiersz,
                  ;1 kolumna
         ;(współrzędne liczone od 0)
MOV R0,#LCD_1     ;w tej komórce wynik
                  ;konwersji
CALL WriteAString
         ;umieszczenie liczby na
         ;ekranie
```

Procedura *Hex2DecConv* odejmuje od liczby szesnastkowej najpierw 10000, później 1000, 100, 10 i liczy, ile razy można było poszczególne liczby odjąć. Liczniki operacji różnic stają się wagami liczby dziesiętnej. *Dec2AcsiiConv* do każdej z wag dziesiętnych dodaje kod znaku „0" i zapisuje w pamięci RAM w zmiennej LCD_1. Jako ostatni kod łańcucha zapisywana jest cyfra 0, informująca o końcu ciągu do wyświetlenia. Znak ten wymagany jest przez *WriteAString*.

Nieco inaczej używa się pokrewnej *WriteAString* procedury *WriteCString* wypisującej łańcuch znaków z pamięci stałej na ekranie LCD. Tu również ważne jest, aby łańcuch był zakończony cyfrą 0, bo to jest dla procedury informacja o końcu ciągu znaków. Zbyt duża długość łańcucha (więcej niż 80 znaków) nie jest sygnalizowana - po prostu koniec napisu przykryje jego początek.

```
;przykład użycia procedury
WriteCString
Napis db 'Elektronika..Praktyczna ,0
MOV R1,#0
MOV R2,#0
;zacznij od 1|wiersza i|1|kolumny
MOV DPTR,#Napis   ;do dptr adres napisu
                  ;w|rom
CALL WriteCString
                  ;wyświetl napis
JP $              ;koniec programu
```

Na podobnej zasadzie działa procedura *DefChars*. Ona również pobiera z pamięci programu wzorce znaków od adresu *Chars* do napotkania cyfry 0 i wysyła je jako definicje do CG RAM (proszę pamiętać o tym, że najstarsze 3 bity bajtu definicji są odrzucane i pusta linia znaku może być zapisana jako C0h, a nie jako 0h, bo to sygnał o końcu definicji). Przykładowo podaję sposób zdefiniowania kilku polskich „ogonków". Procedura zapisuje definicje znaków w CG RAM od adresu 0, toteż przy wyświetlaniu dostępne są one jako kody od 0 do ostatniej definicji znaku.

```
;przykład użycia DefChars
CALL DefChars     ;wywołanie procedury
                  ;definicji znaków
MOV A,#8
Disp:
MOV R1,#0         ;1 linia
MOV R2,A          ;numer kolumny
                  ;w|akumulatorze
MOV R3,A          ;numer kolumny jest
                  ;również kodem znaku
CALL PutCharOnLCD
                  ;wyświetl znak
SUBB A,#1         ;czy kod mniejszy
                  ;od 0?
JNC Disp          ;nie,następny
                  ;"obrót" pętli
MOV R1,#1         ;wyświetlenie
                  ;napisu przy
                  ;wykorzystaniu
                  ;definicji "ogonków"
MOV R2,#0         ;w 2 linii
                  ;i pierwszej kolumnie
MOV DPTR,#Speed
CALL WriteCString
JP $              ;tak,koniec

;definicje dla DefChars
Char     db
0E0H,0EEH,0E1H,0EFH,0F1H,0EFH,0E4H,0E2H
;litera "ą"
Cc       db
0E2H,0E4H,0EEH,0F0H,0F0H,0F1H,0EEH,0E0H
;litera "ć"
Ce       db
0E0H,0EEH,0F1H,0FFH,0F0H,0EEH,0E4H,0E2H
;litera "ę"
Cl       db
0ECH,0E4H,0E5H,0E6H,0ECH,0E4H,0EEH,0E0H
;litera "ł"
Cni      db
0E4H,0E2H,0F6H,0F9H,0F1H,0F1H,0F1H,0E0H
;litera "ń"
Co       db
0E2H,0E4H,0EEH,0F1H,0F1H,0F1H,0EEH,0E0H
;lietra "ó"
Csi      db
0E2H,0E4H,0EEH,0F0H,0EEH,0E1H,0FEH,0E0H
;litera "ś"
Czi      db
0E2H,0E4H,0FFH,0E2H,0E4H,0E8H,0FFH,0E0H
;litera "ź"
Cz       db
0E4H,0E0H,0FFH,0E2H,0E4H,0E8H,0FFH,0E0H,0
;litera "ż"
;dla przykładu napis "prędkość"
Speed    db       'Pr ,2, dko ,6,1,0
```

Myślę, że uzbrojeni w wiedzę przedstawioną w tym artykule nie będziemy mieli żadnego kłopotu z wyświetleniem komunikatu, o którym była mowa we wstępie. Ktoś, kto buduje własnoręcznie urządzenia elektroniczne, zrozumie co mam na myśli - cała przyjemność zaczyna się wtedy, gdy nasze urządzenie zaczyna do nas „przemawiać". Życzę więc miłej zabawy!

Jacek Bogusz, AVT
jacek.bogusz@ep.com.pl

Listingi programów omawianych w artykule dostępne są pod adresem ***http://www.ep.com.pl/ftp/lcd_prakt.exe*** *oraz na płycie CD-EP3/2000 w katalogu* ***\Noty katalogowe do projektów\LCD****.*

Nota katalogowa sterownika HD44870 dostępna jest pod adresem: ***http://www.ep.com.pl/ftp/hd44780.pdf*** *oraz na płycie CD-EP3/2000 w katalogu* ***\Noty katalogowe do projektów\LCD****.*