Temat: Obsługa wyświetlacza LCD systemie STRC51. **Ćwiczenie 5. (sd)**

1.Wyświetlacz LCD.

1.1.Zasada pracy wyświetlaczy LCD i kody sterujące.

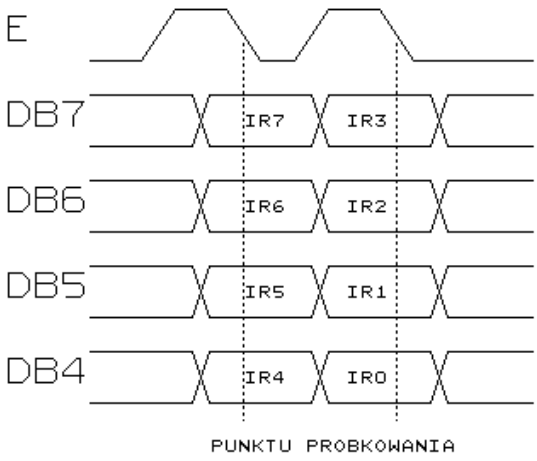
Standardem na rynku wyświetlaczy LCD alfanumerycznych, stały się moduły zawierające kontroler HD44780 firmy Hitachi. Budowa ich jest stosunkowo prosta a możliwości użytkowe niezbyt wyszukane. Ich popularność ugruntowana została ceną i mnogością dostępnych w internecie przykładowych aplikacji. Moduły produkowane przez wielu producentów mają cechy wspólne: układ złącza elektrycznego, sposób programowania (będący niejednokrotnie rozszerzonym zbiorem standardu). W produkcji są wyświetlacze o: 8, 16, 20, 24 znakach w jednej, dwóch lub czterech liniach. Różnią się także wielkością ekranu, poborem prądu i zastosowanym podświetlaniem (lub jego brakiem). W systemie STRC51 zastosowano moduł LCD o szerokości 16 znaków w dwóch liniach, bez podświetlania.

Dla programisty w zależności od użytej konfiguracji (w STRC51 jest to 4-bitowa), moduł widoczny jest jako specyficzna mini-magistrala I/O. Komunikacja po niej przebiega w dwóch trybach:

-poleceń,
-danych.

O tym w jakim trybie pracujemy decyduje linia RS. Dla RS=0, kontroler LCD interpretuje informacje na magistrali jako polecenia do wykonania, dla RS=1 dane są przekazywane do pamięci wewnętrznej kontrolera.

Sygnał E jest impulsem wyzwalania (tzw. strobem) – wprowadza informacje z linii danych do wewnętrznych rejestrów kontrolera modułu. Zasada wpisywania sygnałów polega na podaniu danych na linie danych i wygenerowaniu sekwencji 0-1-0 na linii E (co pokazuje poniższa tabela).



Gdzie:
IR7...IR4 starsze bity przesyłanej danej lub rozkazu,
IR3...IR0 młodsze bity przesyłanej danej lub rozkazu.

Rys.1. Przykład zależności między sygnałem E i danymi na mini-magistrali w trybie 4-bitowym.

Sygnał sterujący na mini-magistrali R/W, umożliwia:

-dla 0 pisanie do modułu LCD,
-dla 1 czytanie z niego.

Dane do i z modułu wystawiane są w na liniach DB7...DB0 dla trybu ośmiobitowego lub dla trybu czterobitowego na DB7...DB4 (patrz rys.1). Wybór trybu jest związany z konfiguracją sprzętową. Poinformowanie modułu o trybie pracy dokonywane jest przez programowe podanie odpowiednich poleceń do LCD.

Transfer informacji w trybie cztero-bitowym polega na przesyłaniu danych w dwóch paczkach (każda potwierdzana sygnałem „strob” - E), najpierw przesyłane są cztery starsze bity na liniach

DB7...DB4 (z zachowaniem kolejności) a następnie cztery młodsze podobnie na liniach DB7...DB4 w taki sposób, że bit 3 przesyłany jest na linii DB7, 2-DB6, 1-DB5, 0-DB4 (patrz rys.1).
W STRC51 na stałe ustawiono linię R/W w stanie 0, co uniemożliwia „czytanie" z modułu LCD. Pociąga to za sobą szczególne rozwiązanie programistyczne. Brak jest możliwości odczytu danych z modułu a co za tym idzie sprawdzenie stanu statusów. Aby rozwiązać ten problem należy z pewnym zapasem (około 10%) przestrzeganie czasów wykonania poszczególnych poleceń przez moduł. Zatem nie wolno wysłać nowego polecenia przed wykonaniem poprzedniego. W STRC51 jedynym rozwiązaniem jest wprowadzanie opóźnień programowych dla poszczególnych poleceń.

1.2.Polecenia HD447800.

Tabela 1. Polecenia, ich znaczenia i czasy wykonania.

Nazwa Polecenia	RS	Dane 7	6	5	4	3	2	1	0	Opis	Czas wykonania
1.Czyszczenie ekranu	0	0	0	0	0	0	0	0	1	Wyczyszczenie pamięci DDRAM i ustawienie wskaźnika zapisu na 0.	1,52ms
2.Powrót do „domu" kursora	0	0	0	0	0	0	0	1	X	Ustawienie wskaźnika zapisu do pamięci ekranu na 0, przywrócenie trybu wprowadzania w stan domyślny (dopisywanie)	1,52ms
3.Ustawienie trybu pracy	0	0	0	0	0	0	1	I/D	S	Ustawienie kierunku poruszania się kursora, oraz trybu wprowadzania danych (dopisywanie, przesuwanie) I/D= 0-dekrementowanie, 1-inkrementowanie S= 1-przesuwanie, 0-dopisywanie.	37us
4.Sterowanie włączeniem/ wyłączeniem	0	0	0	0	0	1	D	C	B	Włączenie / wyłączenie: -wyświetlania: D=1-włączony, 0-wyłączony -kursora: C=1-nie wyświetlany, 0-wyświetlany, -tryb kursora: B=1-blokowy, 0-kursor podkreślenie.	37us
5.Ustalenie trybu poruszania się kursora	0	0	0	0	1	S/C	R/L	X	X	Możliwe jest poruszanie kursora bez zmiany / ze zmianą wyglądu ekranu: S/C-0 – porusza się kursor, S/C-1 – porusza się zawartość ekranu (nie kursor), R/L – kierunek przesuwania się: 1- do prawej, 0- do lewej	37us
6.Parametry funkcjonalne	0	0	0	1	DL	N	F	X	X	Ustalenie: -szerokości magistrali danych: DL=1-8bitów, 0-4bity, -liczby linii: N=0-jedna linia, 1-dwie linie (użyty w STRC51), -zestawu znaków F=1-znaki 5x10, 0-znaki 5x8.	37us

7.Ustawienie wskaźnika CGRAM	0	0	1	A	A	A	A	A	A	Ustalenie wskaźnika CGRAM decydującego o pozycji pisania po pamięci wzoru znaków których kształt wolno zmienić.	37us Uwaga! Przełączanie z DDRAM trwa 150uS.
8.Ustawienie wskaźnika DDRAM	0	1	A	A	A	A	A	A	A	Ustawienie wskaźnika DDRAM – decydującego o pozycji pisania na ekran LCD	37us Uwaga! Przełączanie z CGRAM trwa 150uS.
9.Pisanie do pamięci	1	D	D	D	D	D	D	D	D	Dane z magistrali przenoszone są do odpowiedniego miejsca w pamięci modułu LCD.	41us

Symbol X oznacza stan nieistotny (0 lub 1).

Wyjaśnienia wymagają polecenia: Ustawienie wskaźnika CGRAM i DDRAM. Wskaźnik zapisu może odnosić się do pamięci znaków do przeprogramowania jak i do pamięci ekranu LCD. Pisanie do pamięci znaków następuje po wysłaniu polecenia ustawiającego wskaźnik CGRAM z odpowiednim adresem. Znaki definiowane są w matrycy 8x8. Mimo że znaki wyświetlane są w formacie np.: 5x8 – reszta informacji (trzy starsze bity z każdego bajtu redefinicji znaku) nie jest używana. Dla celów użytkownika można przeprogramować tylko pierwsze 8 znaków (znaki o kodach od 0x00 do 0x07). Redefiniowanie polega na ustawieniu wskaźnika CGRAM na odpowiedni adres (jest on wyliczany według wzoru: adres=numer_znaku*8), a następnie wysłaniu ciągu 8 bajtów redefinicji w trybie danych. Poniższa tabela 2. Przedstawia sposób zapisania znaku 'A' w pamięci:

Tabela 2. Opis wzoru znaku 'A'

Adres	Dane								Opis
	7	6	5	4	3	2	1	0	
X+0	-	-	-	0	0	1	0	0	Pierwsza linia w znaku
X+1	-	-	-	0	1	0	1	0	
X+2	-	-	-	1	0	0	0	1	
X+3	-	-	-	1	0	0	0	1	
X+4	-	-	-	1	1	1	1	1	
X+5	-	-	-	1	0	0	0	1	
X+6	-	-	-	1	0	0	0	1	Ostatnia linia w znaku
X+7	-	-	-	0	0	0	0	0	Linia kursora-podkreślenie Zalecane nie definiowanie tej linii.

Gdzie X jest numerem znaku. Dla podanego w tabeli wyglądu znaku takiego znaku należy wysłać następujący ciąg danych podczas redefinicji:

```
0x04, 0x0a, 0x11, 0x11, 0x1f, 0x11, 0x11, 0x00
```

Należy pamiętać aby po zdefiniowaniu nowego znaku, przejść do trybu ustawiania wskaźnika DDRAM, aby możliwe było używanie nowego znaku.

Aby móc zapisywać znaki w konkretnym miejscu pamięci ekranu, należy wysłać polecenie ustawienia wskaźnika DDRAM. Standardowo znaki wpisywane są od lewego brzegu do prawego od pierwszej linii do następnych. Dzięki ustawianiu wskaźnika DDRAM można umieszczać znaki w dowolnym miejscu pamięci ekranu (w przedziale 0 – 80).

Warto zwrócić uwagę na architekturę użytych modułów w STRC51. Pierwszy znak (lewy górny róg) znajduje się pod lokacją 0x00, ostatni znak na pozycji prawy górny róg, jest pod adresem 0x0F. Znaki w dolnej linii znajdują się pod lokacjami odpowiednio: 0x40...0x4F.

Tabela 3. przedstawia mapę pamięci ekranu LCD

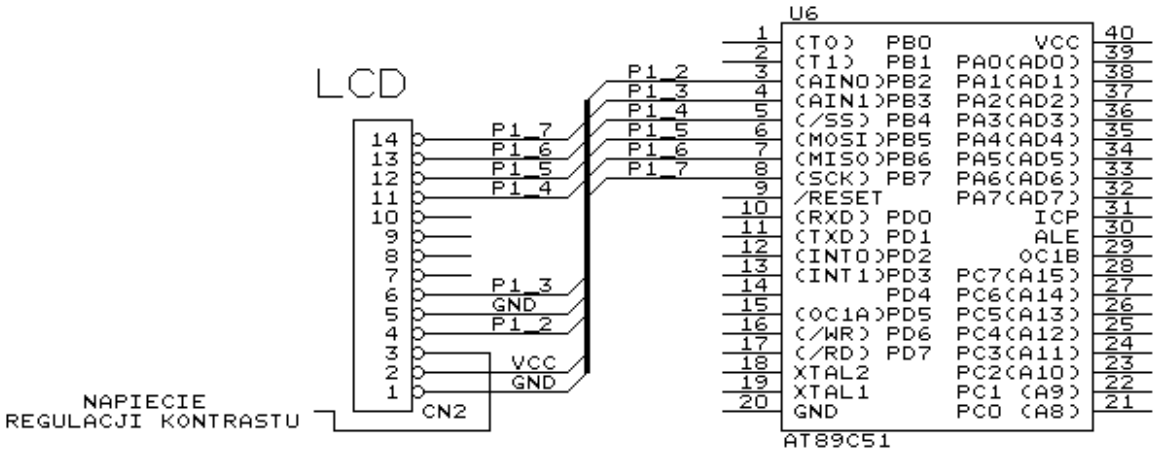
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

Tabela 3. Przedstawia mapę pamięci ekranu wyświetlacza LCD (podane są wartości w zapisie szesnastkowym).

2. Podłączenie wyświetlacza LCD w systemie STRC51 – interfejs w języku „C”.

2.1. Odwołania nisko poziomowe do wyświetlacza.

Od strony programu w „C” moduł wyświetlacza widoczny jest przez port P1. Schemat połączeń pokazuje rysunek 2.



Rys. 2. Schemat połączenia modułu LCD z 80C51

Układ przyporządkowań doprowadzeń jest następujący:

Nazwa symboliczna portu	Numer bitu	Sygnał modułu LCD
P1.2	2	RS
P1.3	3	E
P1.4	4	DB4
P1.5	5	DB5
P1.6	6	DB6
P1.7	7	DB7

Port P1 w języku „C” ma następującą deklaracje (użycie w treści pliku .C wpisu: #include <8051.h> zastępuje tą deklarację):

```
sfr at 0x90 P1;
```

Aby do modułu LCD wysłać w trybie 8 bitowym jakieś polecenie, należy wykonać następujące operacje:

```
P1=polecenie;
P1=polecenie | E;     //gdzie E – jest maską bitową "ustawiającą" linię E – należy ją
                      //          wyznaczyć na podstawie znanego przypisania linii
                      //          sterujących do odpowiednich linii portu P1
P1=polecenie & !E;    //gdzie !E – jest zanegowana maska bitowa, zerującą linie E –
                      //          należy ją wyznaczyć - jak wyżej
```

W przypadku komunikacji w trybie 4 bitowym wysłanie polecenia polega, na wykonaniu następujących czynności:

```
P1=polecenie & 0xf0;                  //najpierw starsza połówka bajtu
P1=(polecenie & 0xf0) | E;            //sygnał „strob”
P1=(polecenie & 0xf0) & !E;
P1=(polecenie << 4) & 0xf0;           //teraz młodsza połówka bajtu
P1=((polecenie << 4) & 0xf0) | E;     //sygnał „strob”
P1=((polecenie << 4) & 0xf0) & !E;
```

Wysłanie w trybie danych, dla linii cztero bitowej powinno wyglądać w następujący sposób:

```
P1=(dana & 0xf0) | RS;
P1=((dana & 0xf0) | E) | RS;
P1=((dana & 0xf0) & !E) | RS;
P1=((dana << 4) & 0xf0) | RS;
P1=(((dana << 4) & 0xf0) | E) | RS;
P1=(((dana << 4) & 0xf0) & !E) | RS;
```

W powyższych przykładach symbol RS, oznacza analogicznie jak symbol E użyty wcześniej, maskę bitową ustawiającą linię RS modułu LCD.
Należy zwrócić uwagę iż podane implementacje nie są optymalne i nie należy ich w identycznym kształcie stosować w swoich programach. W niektórych przypadkach konieczne jest użycia drobnego opóźnienia między wykonaniem czynności w sąsiednich liniach pseudokodu (zależnie od sposobu zaimplementowania kodu).

2.2. Odwołania wysokiego poziomu do modułu LCD.

Inicjacja modułu musi się zacząć w trybie 8 bitowym, by następnie transmisja mogła odbywać się w trybie czterobitowego. Poniższy pseudokod prezentuje proces wymiany danych:

```
1. <WŁĄCZENIE ZASILANIA>
2. <Wstrzymanie na 15ms>
3. (w ośmiobitowym trybie!)
    RS  D7  D6  D5  D4  D3  D2  D1  D0
    0   0   0   1   1   n/c n/c n/c  n/c
    n/c – nie podłączone
4. <Wstrzymanie na 4.1ms>
5. (w ośmiobitowym trybie!)
    RS  D7  D6  D5  D4  D3  D2  D1  D0
    0   0   0   1   1   n/c n/c n/c  n/c
6. <Wstrzymanie na 120us>
7. (w ośmiobitowym trybie!)
    RS  D7  D6  D5  D4  D3  D2  D1  D0
    0   0   0   1   1   n/c n/c n/c  n/c
8. <Wstrzymanie na 4.1ms>
9. (w ośmiobitowym trybie!)
    RS  D7  D6  D5  D4  D3  D2  D1  D0
    0   0   0   1   0   n/c n/c n/c  n/c
    ustawienie trybu transmisji na czterobitowy
10. <Wstrzymanie na 40us>
11. (w czterobitowym trybie)
    RS  D7  D6  D5  D4
    0   0   0   1   0
    RS  D7  D6  D5  D4
```

```
      0   1   F   x   x
      ustawienie czterobitowego transferu,
      F= wybór zestawu znaków (1- 5x11, 0- 5x8)
      x- znaczenie nieistotne
   12.  <Wstrzymanie na 40us>
   13. RS  D7  D6  D5  D4
      0   0   0   0   0
      RS  D7  D6  D5  D4
      0   1   0   0   0
      Wyłączenie ekranu, kursora i trybu migotania (kursora)
   14. <Wstrzymanie na 40us>
   15. RS  D7  D6  D5  D4
      0   0   0   0   0
      RS  D7  D6  D5  D4
      0   1   1   1   0
      Włączenie ekranu, włączenie kursora bez migotania
   16.  <Wstrzymanie na 40us>
   17. RS  D7  D6  D5  D4
      0   0   0   0   0
      RS  D7  D6  D5  D4
      0   0   1   1   0
      Ustalenie mod'u pracy na przesuwanie się w prawo kursora
   18. <Wstrzymanie na 40us>
   19. <Inicjacja kompletna>
```

Uwaga:
n/c - oznaczenie stosowane do zaznaczenia, że dane połączenie jest nie wykorzystywane (rozłączone).
Należy zauważyć, że różne egzemplarze mogą mieć inne wymagane czasy wstrzymania między poleceniami, często trzeba je dobierać dla danej serii wyświetlaczy eksperymentalnie. Jednym ze sposobów opóźniania programu są pętle „bezczynne". Można napisać taką implementacje:

```
void czekaj(int d){
  int i;
  for(i=0; i<d; i++){
    _asm
       nop
   _endasm;
  }
}
```

Tak napisana funkcja w pewnych przypadkach może nie działać poprawnie. Tak może stać się gdy kompilator zbyt „gorliwie" zoptymalizuje taki kod – zalecane jest sprawdzenie dokumentacji do kompilatora (np.: `n:\sdcc\doc\index.html`). Aby sprawdzić czy taka sytuacja (mimo naszych zabiegów) nastąpiła, używając kompilatora `SDCC` można przejrzeć efekt kompilacji w postaci pliku `ASM` (zlokalizowanego z reguły w tym samym katalogu co plik źródłowy). W takim pliku można zlokalizować linię odpowiadającą linii z pliku w języku `C` (linie komentowane są za pomocą znaku `;` a informacja o numerze linii z pliku `C` jest umieszczona w formacie: `;nazwa_pliku_w_c.c:numer_linii: odpowiadający_kod_w_c)`, i sprawdzić czy wersja asemblerowa odpowiada żądanemu kodowi w `C`. W razie problemów z rozszyfrowaniem kodu asemblerowego, można metodą prób i błędów dokonać sprawdzenia efektywnego kodu (w tym opóźnień).