Sprawozdanie z laboratorium

Architektury Komputerów

Laboratorium numer: 3

Temat: Wyświetlacz LCD – Hitachi HD 44780

Wykonujący ćwiczenie:

- Patryk Wójtowicz

Studia dzienne I stopnia

Kierunek: Informatyka

Semestr: III Grupa zajęciowa: Lab 15

Prowadzący ćwiczenie:
Dr inż. Mirosław
Omieljanowicz

.....

OCENA

Data wykonania ćwiczenia

25.10.2021r.

.....

Data i podpis prowadzącego

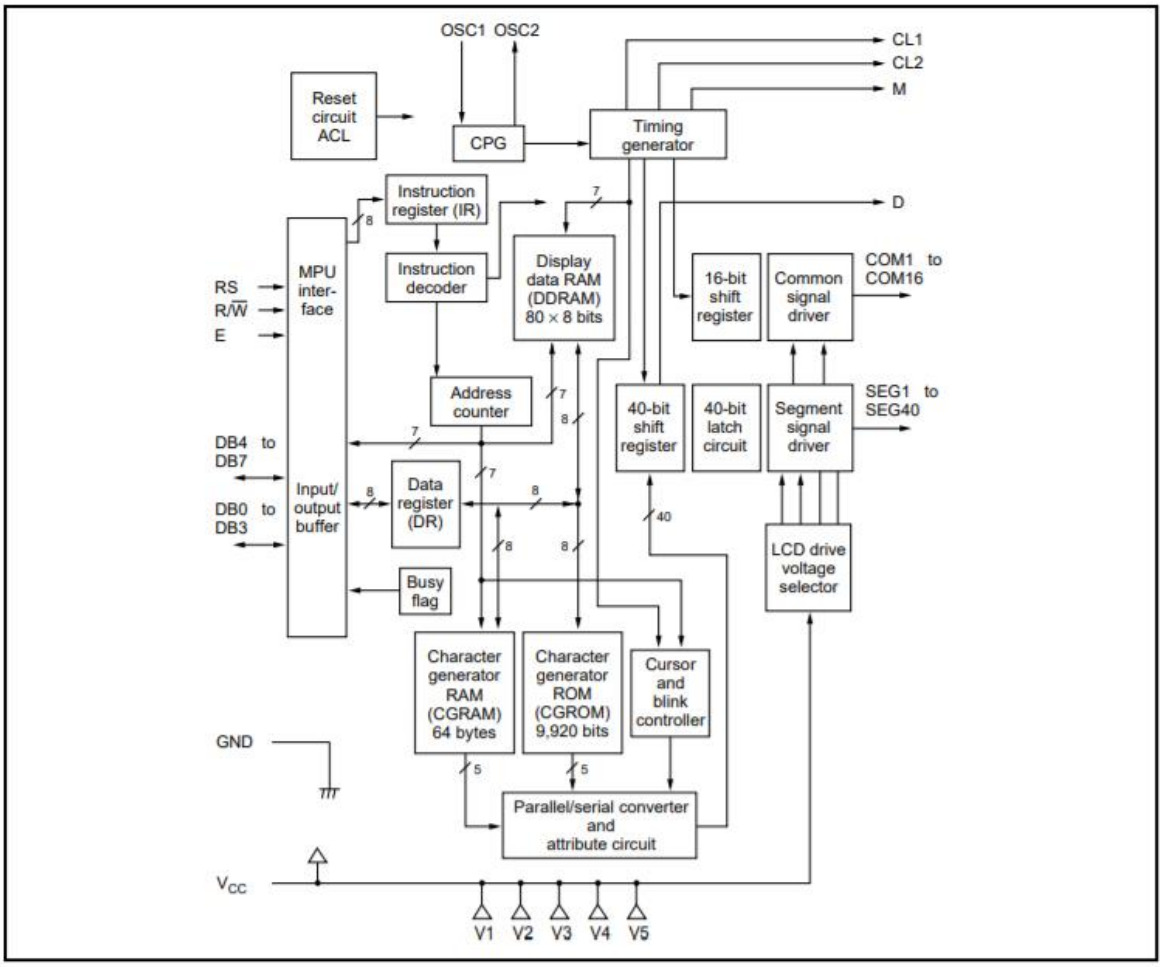
Cel zadania

Zapoznanie się z obsługą wyświetlacza, zarządzanego układem Hitachi HD 44780 przy uzyciu mikrokontrolera MSP430 na płytce EasyWeb2.

Teoria

Płytka EasyWeb2 z mikrokontrolerem MSP430F149 wyposażona jest w układ wyświetlacza LCD opartego o sterownik Hitachi HD 44780. Do obsługi wyświetlacza niezbędne są komendy zarządzające wyświetlaczem oraz dane króre chcemy wyświetlić.

HD44780U Block Diagram



Rysunek 1 Schemat sterownika Hitachi HD 44780 [1]

Wyświetlacz pracuje z ośmio bitową (DB0 -DB7) lub cztero bitową (DB4 -DB7) magistralą danych, dodatkowo wyświetlacza posiada podłączone: linie potwierdzającą E oraz dwie linie specjlane RS , R/W (Read/Write) oraz zasilanie.

- Linia RS, odpowiedzialna jest za interpetowanie danych przychodzących do wyświetlacza. Rozróżniamy dwa stany wysoki i niski, dla niskiego dane są interpretowane jako instrukcje, natomiast dla wysokiego dane interpretowane są jako znaki które mają być wyświetlone na ekranie

a zapisane są w tablicy znaków pamięci ROM.

- Linia R/W (Read/Write) informuje sterownik wyświetlacza o trybie pracy pomiędzy odczytem a zapisem. W płytce EasyWeb2 linia ta jest na stałe podłączona do masy, czyli możliwy jest tylko zapis danych.

- Linia E służy do aktywowania zapisu danych do układu, powinna być aktywna za każdym razem kiedy zostaną podane dane na linie danych.

Flaga BF zapewnia kolejność wykonania instrukcji jedna po drugiej. Flaga ta aktywuje się za każdym razem gdy zostanie przekazana instrukcja, flaga wtedy przyjmuje wartość 1. Wtedy żadna instrukcja nie jest wykonywana, dopóki flaga BF nie wróci do stanu 0. Stan 0 jest przywracany po zakończeniu jednej instrukcji i wtedy można wczytać kolejną instrukcję i ją wykonać.

Sterownik wyświetlacza wyposażony jest w dwa rodzaje pamięci DD_RAM oraz CG_RAM, DD_RAM służy do przechowywania aktualnie zakodowanych znaków, CG_RAM posiada miejsce na specjalnie zakodowane znaki przez użytkownika.

Ważne są także dwa ośmiobitowe rejestry IR oraz DR, IR przyjmuje kody instrukcji takie jak czyszczenie ekranu czy przesunięcia. DR przyjmuje dane do przesłania na wyświetlacz. DR wybrać można poprzez stan wysoki na lini RS a IR można wybrać poprzez stan niski na lini RS.

	Code											Execution Time (max) (when f_{cp} or f_{osc} is 270 kHz)
Instruction	**RS**	**R/$\overline{W}$**	**DB7**	**DB6**	**DB5**	**DB4**	**DB3**	**DB2**	**DB1**	**DB0**	**Description**	
Clear display	0	0	0	0	0	0	0	0	0	1	Clears entire display and sets DDRAM address 0 in address counter.	
Return home	0	0	0	0	0	0	0	0	1	—	Sets DDRAM address 0 in address counter. Also returns display from being shifted to original position. DDRAM contents remain unchanged.	1.52 ms
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies display shift. These operations are performed during data write and read.	37 µs
Display on/off control	0	0	0	0	0	0	1	D	C	B	Sets entire display (D) on/off, cursor on/off (C), and blinking of cursor position character (B).	37 µs
Cursor or display shift	0	0	0	0	0	1	S/C	R/L	—	—	Moves cursor and shifts display without changing DDRAM contents.	37 µs
Function set	0	0	0	0	1	DL	N	F	—	—	Sets interface data length (DL), number of display lines (N), and character font (F).	37 µs
Set CGRAM address	0	0	0	1	ACG	ACG	ACG	ACG	ACG	ACG	Sets CGRAM address. CGRAM data is sent and received after this setting.	37 µs
Set DDRAM address	0	0	1	ADD	ADD	ADD	ADD	ADD	ADD	ADD	Sets DDRAM address. DDRAM data is sent and received after this setting.	37 µs
Read busy flag & address	0	1	BF	AC	AC	AC	AC	AC	AC	AC	Reads busy flag (BF) indicating internal operation is being performed and reads address counter contents.	0 µs

Rysunek 2 Tabela komend wyświetlacza Hitachi HD 44780 [1]

```
#define     CLR_DISP            0x01            // clear display
#define     CUR_HOME            0x02            // return home
#define     ENTRY_INC           0x06            // entry mode increment
#define     ENTRY_INC_ROL       0x07            // entry mode increment with rol data
#define     ENTRY_DEC           0x04            // entry mode decrement
#define     ENTRY_DEC_ROL       0x05            // entry mode decrement witch rol dat
#define     DISP_OFF            0x08            // all display off
#define     DISP_ON             0x0c            // all display on
#define     DATA_ROL_LEFT       0x18            // rol data left
#define     DATA_ROL_RIGHT      0x1c            // rol data right
#define     CUR_SHIFT_LEFT      0x10            // shift coursor left
#define     CUR_SHIFT_RIGHT     0x14            // shift coursor right
#define     DD_RAM_ADDR         0x80            // set DD_RAM 1
#define     DD_RAM_ADDR2        0xc0            // set DD_RAM 2
#define     DD_RAM_ADDR3        0x28            // set DD_RAM 3
#define     CG_RAM_ADDR         0x40            // set CG_RAM
```

Rysunek 3 Zdefiniowane komendy na podstawie tabeli z Rysunku 2

Ważnym aspektem pracy wyświetlacza jest poprawne jego zainicjonowanie, czysto teoretycznie po włączeniu zasilania sterownik inicjalizuje się sam. W praktyce powinno się to robić w programie. Jest to ważne dlatego iż sterownik wyświetlacza musi pracować w trybie pracy magistrali. Uzyskuje się to poprzez wysłanie trzykrotne inicjalizacji wyświetlacza w trybie ośmio bitowym a następnie w trybie cztero bitowym.

Na szczęście biblioteki „***lcd.h***" oraz „***portyLcd.h***" do programowania w języku C posiadają zdekalrowane funkcje przygotowania wyświetlacza do pracy. Najważniejsze to:

- *InitPortsLcd* -> odpowiedzialna za incjalizacje protów wyswietlacza w celu komunikacji ze sterownikiem.

- *InitLCD* -> odpowiedzialna za inicjalizację ekranu LCD w celu poprawnego wyświetlania danych.

- *clearDisplay* -> odpowiedzialna za wyczyszczenie ekranu z danych. Zaleca się wykonać to jako pierwszą czynność po Inicjalizacji portów jak i wyświetlacza LCD.

- *SEND_CMD* -> służy do sterowania czynnościami wyświetlacza, trzy przykładowe parametry funkcji to:

    * *CG_RAM_ADDR* -> służy do wysyłania wpisów o znakach stworzonych przez użytkownika.
    * *DD_RAM_ADDR* -> odpowiedzialny za pisanie w pierwszym wierszu wyświetlacza.
    * *DD_RAM_ADDR2* -> odpowiedzialny za pisanie w drugim wierszu wyświetlacza.

- *SEND_CHAR* -> Wysyła znak na ekran, pzryjmuje jeden parametr, indeks znaku albo jego odwołanie.

```
void clearDisplay() {
    SEND_CMD(CLR_DISP); // Czyszczenie wyświetlacza
    Delayx100us(10);
}
void gotoSecondLine() {
    SEND_CMD(DD_RAM_ADDR2); // Przejście do drugiej lini
}


void Delay (unsigned int a) // Opóźnienie
{
  int k;
  for (k=0 ; k != a; ++k) {
    _NOP();
    _NOP();
    _NOP();
    _NOP();
  }
}

void Delayx100us(unsigned char b) // Opóźnienie w micro sekundach
{
  int j;
  for (j=0; j!=b; ++j) Delay (_100us);
}


void _E(void)
{
        bitset(P2OUT,E);  // Przełączenie linii E
  Delay(_10us);
  bitclr(P2OUT,E);
}
```

Rysunek 4 Zdeklarowane funkcje w bibliotece LCD cz.1 [1]

```
void SEND_CHAR (unsigned char d) // Wysyłanie znaku
{
    int temp;
  Delayx100us(5);                 //.5ms
  temp = d & 0xf0;    //get upper nibble
  LCD_Data &= 0x0f;
  LCD_Data |= temp;
  bitset(P2OUT,RS);               //set LCD to data mode
  _E();                           //toggle E for LCD
  temp = d & 0x0f;
  temp = temp << 4;               //get down nibble
  LCD_Data &= 0x0f;
  LCD_Data |= temp;
  bitset(P2OUT,RS);             //set LCD to data mode
  _E();                           //toggle E for LCD
}

void SEND_CMD (unsigned char e) // Wysyłanie polecenia
{
    int temp;
  Delayx100us(10);                //10ms
  temp = e & 0xf0;    //get upper nibble
  LCD_Data &= 0x0f;
  LCD_Data |= temp;               //send CMD to LCD
  bitclr(P2OUT,RS);               //set LCD to CMD mode
  _E();                           //toggle E for LCD
  temp = e & 0x0f;
  temp = temp << 4;               //get down nibble
  LCD_Data &= 0x0f;
  LCD_Data |= temp;
  bitclr(P2OUT,RS);             //set LCD to CMD mode
  _E();                           //toggle E for LCD
}
```

Rysunek 5 Zdeklarowane funkcje w bibliotece LCD cz.2 [1]

```
void InitLCD(void) // Inicjalizacja LCD
{
    bitclr(P2OUT,RS);
    Delayx100us(250);                   //Delay 100ms
    Delayx100us(250);
    Delayx100us(250);
    Delayx100us(250);
    LCD_Data |= BIT4 | BIT5;            //D7-D4 = 0011
    LCD_Data &= ~BIT6 & ~BIT7;
    _E();                               //toggle E for LCD
    Delayx100us(100);                   //10ms
    _E();                               //toggle E for LCD
    Delayx100us(100);                   //10ms
    _E();                               //toggle E for LCD
    Delayx100us(100);                   //10ms
    LCD_Data &= ~BIT4;
    _E();                               //toggle E for LCD

    SEND_CMD(DISP_ON);
    SEND_CMD(CLR_DISP);
    Delayx100us(250);
    Delayx100us(250);
    Delayx100us(250);
    Delayx100us(250);
}
```

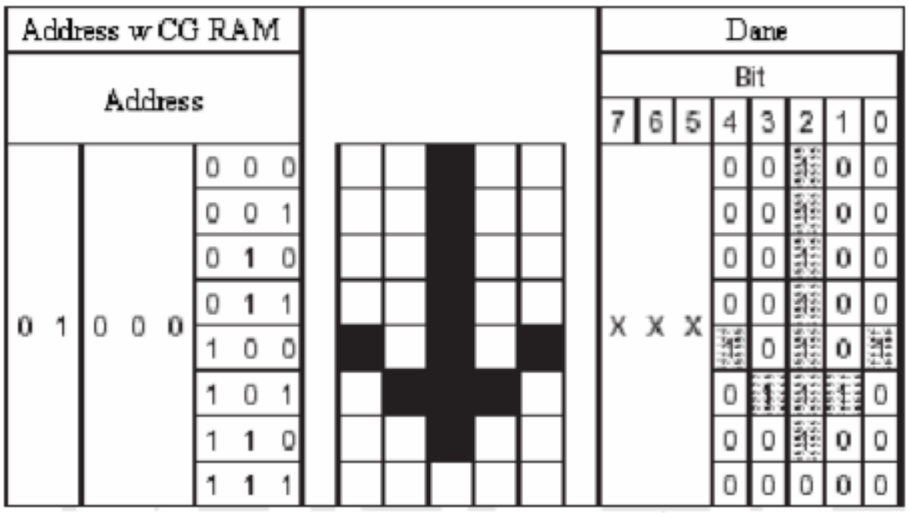
Rysunek 6 Zdeklarowane funkcje w bibliotece LCD cz.3

Przykładowe zainicjonowanie wyświetlacza i jego wyczyszczenie przy zastosowaniu bibliotek zewnętrznych w C:

```
InitPortsLcd();
InitLCD();
clearDisplay();
```

Rysunek 7 Przykładowa inicjalizacja ekranu z wykorzystaniem bibliotek

Wyświetlacz pozwalna na wyświetlenie 16 znaków w jednej linii. Każdy znak składa się z 40 pikseli. Wspomniane było o pamięci CG_RAM umożliwia nam ona definicje własnych znaków. Odbywa się to poprzez odpowiednie ustawienie 8 bajtów w pamięci CG_RAM. Dla wyświetlacza Hitachi HD 44780 ustawiamy bity D0 do D4 ponieważ pojedyńczy wyświetlacz z szesnastu posiada tylko rozmiar 5 na 8 pixeli (5 bajtów , 40 pikseli), dlatego bity D5 – D7 są nieistotne. Poniższy rysunek przedstawia przykładowy sposób zapisu bitów w celu uzyskania strzałki w dół:



Rysunek 8 Tablica rozkładu bitów dla pamięci wyświetlacza i ich zapis

Warto też dodać iż piksele liczone są od prawej do lewej co jest nieintuicyjne i może sprawiać problemy początkującym programistom w liczeniu dziesiętnej reprezentacji liczby gdyż zapisać ją trzeba od tyłu i w łatwy sposób można wtedy przełożyć ją na liczbę dziesiętną. Poniżej przykład zakodowanych znaków:

adres	7	6	5	4	3	2	1	0	HEX
00	0	0	0	0	1	1	0	0	0CH
01	0	0	0	0	0	1	0	0	04H
02	0	0	0	0	0	1	1	0	06H
03	0	0	0	0	0	1	0	0	04H
04	0	0	0	0	1	1	0	0	0CH
05	0	0	0	0	0	1	0	0	04H
06	0	0	0	0	1	1	1	0	0EH
07	0	0	0	0	0	0	0	0	00H
08	0	0	0	0	0	0	1	0	02H
09	0	0	0	0	0	1	0	0	04H
0A	0	0	0	0	1	1	1	0	0EH
0B	0	0	0	1	0	0	0	1	11H
0C	0	0	0	1	0	0	0	1	11H
0D	0	0	0	1	0	0	0	1	11H
0E	0	0	0	0	1	1	1	0	0EH
0F	0	0	0	0	0	0	0	0	00H
10	0	0	0	0	0	0	1	0	02H
11	0	0	0	0	0	1	0	0	04H
12	0	0	0	0	1	1	1	0	0EH
13	0	0	0	1	0	0	0	0	10H
14	0	0	0	0	1	1	1	0	0EH
15	0	0	0	0	0	0	0	1	01H
16	0	0	0	1	1	1	1	0	1EH
17	0	0	0	0	0	0	0	0	00H
18	0	0	0	1	1	1	1	1	1FH
19	0	0	0	0	0	0	0	1	01H
1A	0	0	0	0	0	0	1	0	02H
1B	0	0	0	1	1	1	1	1	1FH
1C	0	0	0	0	1	0	0	0	08H
1D	0	0	0	1	0	0	0	0	10H
1E	0	0	0	1	1	1	1	1	1FH
1F	0	0	0	0	0	0	0	0	00H

Rysunek 9 Przykład kodowania znaków [3]

Założenia zadania

Założeniem zadanie jest demonstracja obsługi wyświetlacza Hitachi HD44780 przy użyciu mikrokontrolera MSP430 na płytce EasyWeb2. Zadanie polega na wyświetleniu danych w dwóch wierszach a nastepnie przy użyciu przycisku pierwszego lub drugiego, sterowanie w lewo lub prawo wyświetlanymi danymi. Dodatkowo zadanie polegało na utworzeniu polskiego znaku w celu nauczenia się obsługi tablicy znaków i ich wyświetlania.

Realizacja zadania:

```
1  #include<msp430x14x.h>                          // Biblioteka MSP430
2  #include "lcd.h"                                // Bliblioteka wyświetlacza
3  #include "portyLcd.h"                           // Biblioteka portów wyświetlacza
4
5  #define KL1 BIT4&P4IN                          // Zdefiniowanie Klawisza pierwszego jako KL1
6  #define KL2 BIT5&P4IN                          // Zdefiniowanie Klawisza drugiego jako KL2
7
8  unsigned char znak1 ,znak2, znak3;             // Zdefiniowanie zmiennej znakowej znak1 , znak2 oraz znak3
9
10 // Przesunięcie o jedną pozycje w prawo lub lewo po kliknięciu klawisza
11 // minimum 4 znaki, w tym jeden polski, wzór -> <=ż=>
12 // Dodatkowo obiekt znajduje się na obu wierszach
13
14 void polski_znaczek()                          // Funkcja wysłania znaku do pamięci
15 {
16     SEND_CMD(CG_RAM_ADDR);                     // Inicjonowanie wysyłania do pamięci
17     int znaczek[8] = {4,0,31,2,4,8,31,0};      // Definicja znaku w systemie dziesiętkowym ale indexy pikseli
18                                                // od prawej liczone w systemie potęg dwójki (1,2,4,8,16)
19   for(int i=0;i<8;i++) SEND_CHAR(znaczek[i]);  // Wysyłanie z tablicy znaku do pamięci
20 }
21
22 void main( void )
23 {
24   znak1='<';                                   // Znak końca lewego
25   znak2='=';                                   // Znak środka
26   znak3='>';                                   // Znak końca prawego
27   WDTCTL=WDTPW+WDTHOLD;                        // Zatrzymanie WDT
28
29   InitPortsLcd();                              // Inicjalizacja portów
30   InitLCD();                                   // Inicjalizacja LCD
31   clearDisplay();                              // Czyszczenie LCD
32
33   polski_znaczek();                            // Wywołanie funkcji inicjującej znaczek
34
35   SEND_CMD(DD_RAM_ADDR);                       // Ustawienie pisania na wiersz pierwszy
36
37   SEND_CHAR(znak1);                            // Wysłanie pierwszego znaku (początek)
38   SEND_CHAR(znak2);                            // Wysłanie drugiego znaku (środek)
```

Rysunek 5 Kod programu cz.1

```
39   SEND_CHAR(8);                                // Wysłanie polskiego znaku (środek), 8 index jest indexem
40                                                // w którym przechowywuje się nasz znak (pierwszy wolny, dla uzytkownika)
41   SEND_CHAR(znak2);                            // Wysłanie drugiego znaku (środek)
42   SEND_CHAR(znak3);                            // Wysłanie trzeciego znaku (koniec)
43
44   SEND_CMD(DD_RAM_ADDR2);                      // Ustawienie pisania na wiersz drugi
45
46   SEND_CHAR(znak1);                            // Wysłanie pierwszego znaku (początek)
47   SEND_CHAR(znak2);                            // Wysłanie drugiego znaku (środek)
48   SEND_CHAR(8);                                // Wysłanie polskiego znaku (środek), 8 index jest indexem
49                                                // w którym przechowywuje się nasz znak (pierwszy wolny, dla uzytkownika)
50   SEND_CHAR(znak2);                            // Wysłanie drugiego znaku (środek)
51   SEND_CHAR(znak3);                            // Wysłanie trzeciego znaku (koniec)
52
53     while (1)                                  // Nieskończona pętla
54     {
55       if((KL1)==0) SEND_CMD(DATA_ROL_LEFT);    // Przesuwanie napisu w lewo w momencie kliknięcia przycisku pierwszego
56       else if((KL2)==0) SEND_CMD(DATA_ROL_RIGHT); // Przesuwanie napisu w prawo w momencie kliknięcia przycisku drugiego
57       Delayx100us(100);                        // Ustawienie opóźnienia na 100ms w celu przesuwania znaków o jedno pole
58   }                                            // oraz umożliwienie szybkiego klikania przycisku (płynność przechodzenia)
59 }
```

Rysunek 6 Kod program cz.2

Wnioski

Programowanie wyświetlacza LCD ze sterownikiem HD44780 firmy Hitachi było bardzo ciekawym doświadczeniem, szczególnie w momencie programowania własnych znaków. Dzięki tej pracy zrozumiałe stało się w jaki sposób mogą działać bilbordy ze zmieniającymi się napisami, które są powszechnym widokiem na naszych ulicach. Niestety rozmiar matrycy mocno ogranicza aspekt wizualny i zmusza nas do tworzenia prostych znaków, nie możliwym jest stworzenie bardziej wymagajacych znaków jak na przykład emotek. W poprawnym wykonaniu ćwiczenia kluczowym było zrozumienie jak działa wyświetlacz, tzn. ile znaków wyświetla się w jednej linii, jakich komend użyć do wypisywania znaków w konkretnej linii, oraz w jaki sposób można zdefiniować znaki specjalne. Po zapoznaniu się z tymi komendami zadanie stało się banalnie proste i przyjemne do realizacji ciekawych przejść jak i przesuwających się napisów.

Bibliografia

[1] Informator Laboratoryjny
[2] https://pl.wikipedia.org/
[3] http://old.piko.avx.pl/