Wojskowa Akademia Techniczna im. Jarosława Dąbrowskiego

Laboratorium z przedmiotu:
Interfejsy komputerów cyfrowych

Sprawozdanie z ćwiczenia laboratoryjnego nr 1:
Klawiatura

Prowadzący:
mgr inż. Krzysztof Szajewski

Wykonał: Radosław Relidzyński
Grupa: WCY20IY4S1
Data laboratoriów: 22.03.2021 r.

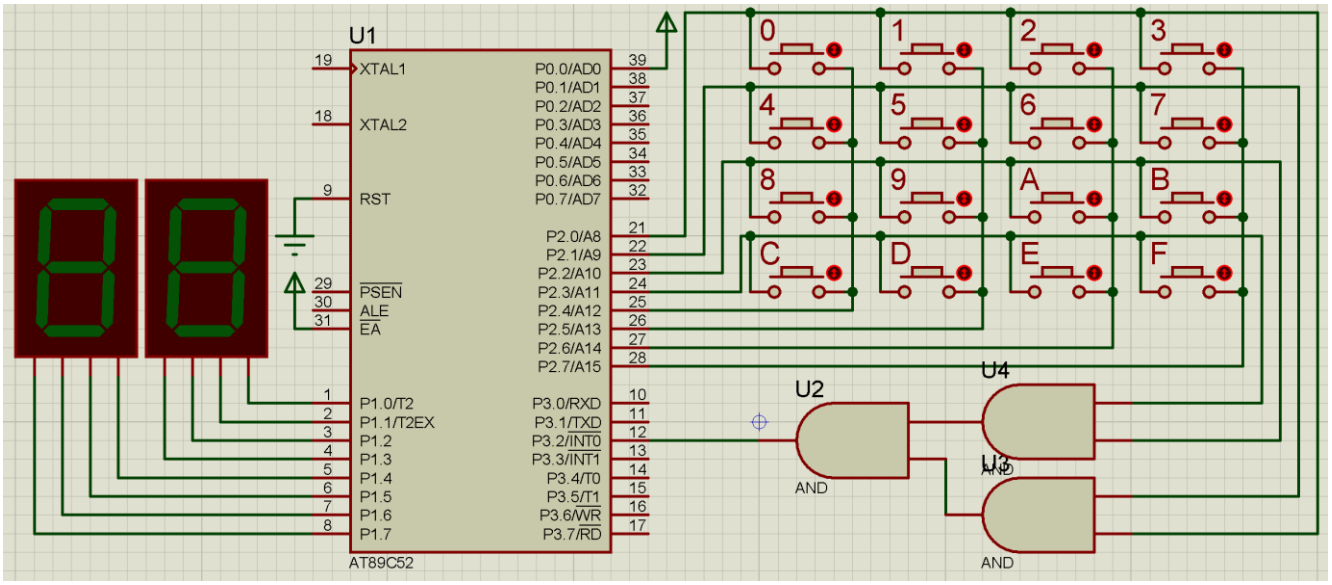
Spis treści

A.Treść zadania

Zbudować w środowisku Proteus klawiaturę z szesnastoma klawiszami opartą na mikrokontrolerze AT89C52.
Należy wzorować się na schemacie poniżej (klawiatura z czterema przyciskami).

- mikrokontroler AT89C52 - (MASTER) - sygnał "1" do pinu P0_0
- po dwa siedmiosegmentowe wyświetlacze podłączone do pinów portu P1 MASTER
- przyciski za pośrednictwem bramki OR aktywnej zerami podłączony do pinu ~INT0 MASTER
- wiersze przycisków podłączone do linii 0,1,2,3 portu P2
- kolumny przycisków podłączone do linii 4,5,6,7 portu P2

Układ powinien być rozwinięciem układu przedstawionego na rysunku:

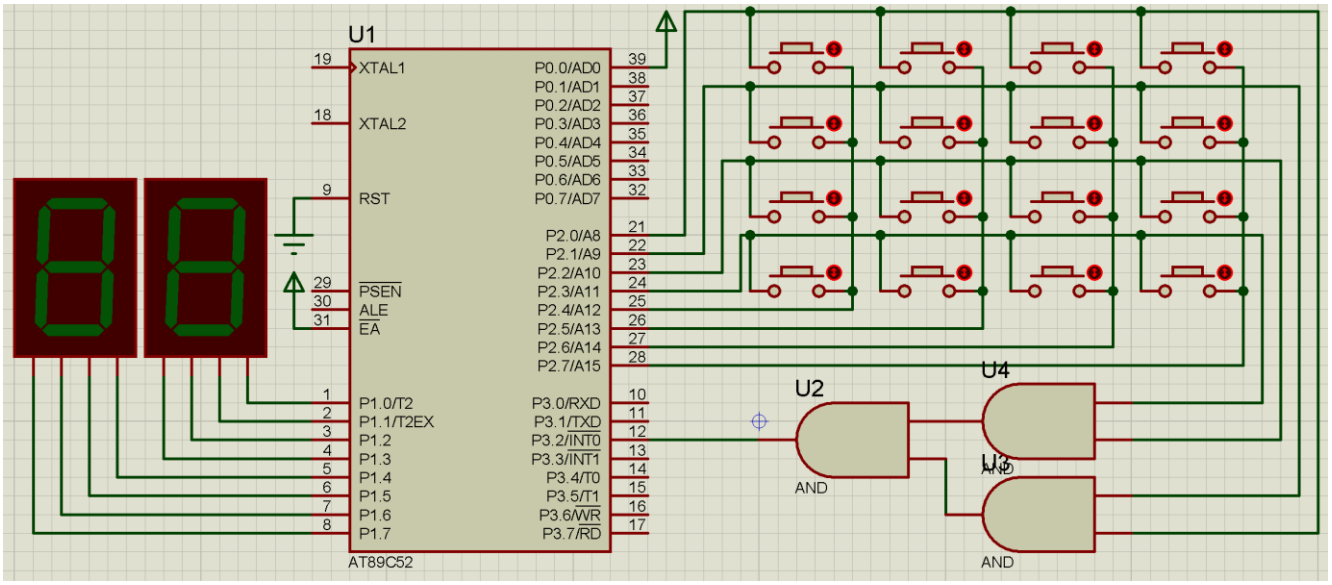


Poniższy link zawiera przykładowy plik, jaki należy opracować w środowisku Keil w celu wygenerowania pliku w formacie hex, dla mikrokontrolera AT89C52. Plik:

```
#include <REGX52.H>

 bit bM = 1;
 int dt = 10;
 int tt = 100;
 volatile unsigned char c;

void init()
{
    IT0 = 1;    //INT0 aktywne zero
    EX0 = 1;    //Wlaczenia INT0
    ES = 1;     //Wlaczenie przerwan od portu szeregowego
    EA = 1;     //Wlaczenie wszstkich przerwan
}

void int0ISR() interrupt 0
 {
    EX0 = 0;
    bM = 0;
    EX0 = 1;
 }

 void generowanieKoduKlawisza()
 {
```

```
    switch(P2)
    {
        case 0x0A:
            P1 = 0x30;  // kod ASCII '0'
            break;
        case 0x06:
            P1 = 0x31;  //kod ASCII '1'
            break;
        case 0x09:
            P1 = 0x32;  //kod ASCII '2'
            break;
        case 0x05:
            P1 = 0x33;  //kod ASCII '3'
            break;
        default:
            P1 = 0xFF;
            break;
    }
 }

void main()
{
    P1 = 0xFF;
    P2 = 0x0B;
    init();
    while(P0_0 == 1)
    {
         while(bM)
         {
             switch(P2)
             {
                case 0x0B:
                    P2 = 0x07;
                    break;
                case 0x07:
                    P2 = 0x0B;
                    break;
             }
             if (P3_2 == 1)
             {
                P1 = 0xFF;
             }
         }
         generowanieKoduKlawisza();
         bM = 1;
    }
    while (P0_0 == 0){}
}
```

B.Schemat układu (Proteus)



C.Program mikrokontrolera (Keil)

```
#include <REGX52.H>

bit bM = 1;
int dt = 10;
int tt = 100;

void init()
{
    IT0 = 1;     //INT0 aktywne zero
    EX0 = 1;     //Wlaczenia INT0
    ES = 1;      //Wlaczenie przerwan od portu szeregowego
    EA = 1;      //Wlaczenie wszstkich przerwan
}

void int0ISR() interrupt 0
{
    EX0 = 0;
    bM = 0;
    EX0 = 1;
}

void generowanieKoduKlawisza()
{
    switch(P2)
    {
        //Wiersz 1
        case 255 - (1 + 16):
            P1 = 0x30;  // kod ASCII '0'
```

```
            break;
        case 255 - (1 + 32):
            P1 = 0x31;  // kod ASCII '1'
            break;
        case 255 - (1 + 64):
            P1 = 0x32;  // kod ASCII '2'
            break;
        case 255 - (1 + 128):
            P1 = 0x33;  // kod ASCII '3'
            break;

        //Wiersz 2
        case 255 - (2 + 16):
            P1 = 0x34;  // kod ASCII '4'
            break;
        case 255 - (2 + 32):
            P1 = 0x35;  // kod ASCII '5'
            break;
        case 255 - (2 + 64):
            P1 = 0x36;  // kod ASCII '6'
            break;
        case 255 - (2 + 128):
            P1 = 0x37;  // kod ASCII '7'
            break;

        //Wiersz 3
        case 255 - (4 + 16):
            P1 = 0x38;  // kod ASCII '8'
            break;
        case 255 - (4 + 32):
            P1 = 0x39;  // kod ASCII '9'
            break;
        case 255 - (4 + 64):
            P1 = 0x41;  // kod ASCII 'A'
            break;
        case 255 - (4 + 128):
            P1 = 0x42;  // kod ASCII 'B'
            break;

        //Wiersz 4
        case 255 - (8 + 16):
            P1 = 0x43;  // kod ASCII 'C'
            break;
        case 255 - (8 + 32):
            P1 = 0x44;  // kod ASCII 'D'
            break;
        case 255 - (8 + 64):
            P1 = 0x45;  // kod ASCII 'E'
            break;
```

```
        case 255 - (8 + 128):
            P1 = 0x46;  // kod ASCII 'F'
            break;

        default:
            P1 = 0xFF;
            break;
    }
}

void main()
{
    P1 = 0xFF;
    P2 = 0x7F;
    init();
    while(P0_0 == 1)
    {
        // Dopoki zaden przycisk nie zostal wlaczony
        while(bM)
        {
            // Przechodzenie wiodacego zera
            switch(P2)
            {
                case 0x7F: // 0111 1111
                    P2 = 0xBF;
                    break;
                case 0xBF: // 1011 1111
                    P2 = 0xDF;
                    break;
                case 0xDF: // 1101 1111
                    P2 = 0xEF;
                    break;
                case 0xEF: // 1110 1111
                    P2 = 0x7F;
                    break;
            }
            // Po odlaczeniu odznacza przycisk
            if (P3_2 == 1)
                P1 = 0xFF;
        }
        generowanieKoduKlawisza();
        bM = 1;
    }
    while (P0_0 == 0){}
}
```

D. Analiza mojego rozwiązania z wnioskami

Klawiatura

Składa się z 16 klawiszy (przycisków), ustawionych w 4 wierszach i 4 kolumnach.

Wciśnięcie danego przycisku powoduje wyświetlenie kodu szesnastkowego jego przypisanego znaku.

Schemat

Zastosowany został mikrokontroler AT89C52, w celu jego podłączenia do wejścia RST (restart) podłączone jest uziemienie (logiczne 0), a do wejścia EA zasilanie (logiczne 1)

Schemat składa się z mikrokontrolera, wyświetlacza oraz klawiatury. Do klawiatury dołączone są bramki logiczne 'AND', służące do obsługi interrupta (w momencie wciśnięcia dowolnego klawisza, bramki przerywają wysyłanie sygnału). Dla programu, wejście będzie na podstawie portu P2, a następnie na podstawie obliczeń będzie wysyłana odpowiednia wartość na port P1.

Kodowanie klawiszy

Kodowanie klawiszy będzie odbywało się zgodnie z poniższą tabelą:

Lp.	Wiersz	Kolumna	Znak	Kod ASCII	Wartość binarna
1	1	1	0	0x30	48
2	1	2	1	0x31	49
3	1	3	2	0x32	48
4	1	4	3	0x33	50
5	2	1	4	0x34	48
6	2	2	5	0x35	51
7	2	3	6	0x36	48
8	2	4	7	0x37	52
9	3	1	8	0x38	48
10	3	2	9	0x39	53
11	3	3	A	0x41	54
12	3	4	B	0x42	48
13	4	1	C	0x43	55
14	4	2	D	0x44	48
15	4	3	E	0x45	56
16	4	4	F	0x46	48

<u>Sposób identyfikacji wejścia i generowania wyjścia:</u>

W kodzie programu zaimplementowana jest funkcja „generowanieKoduKlawisza()". Jej zadaniem jest odebrać wartość portu P2 na jego podstawie zbudować wszystkie elementy instrukcji „switch". Wewnątrz tej instrukcji rozpatrzone są wszystkie możliwe pary wiersza i kolumny.

Wiersze reprezentują wartości [1, 2, 4, 8], a kolumny [16, 32, 64, 128], wynika to z odpowiedniego podpięcia klawiatury do portu P2 mikrokontrolera.

Przypadki skonstruowane są takim wyrażeniem: 255 – (wart_wiersza – wart_kolumny).

Dzięki temu program jest w stanie rozróżnić przyciski na ich podstawie określić wartość wyjścia. W ramach każdego przypadku przypisywana jest odpowiednia wartość heksadecymalna znaku, który ma zostać wyświetlony.

Cykl działania programu

Na początku jest inicjalizacja, tworzone są początkowe wartości, na przykład:

- Początkowa wartość wyjścia (0xFF – oznacza brak wciśniętego przycisku)
- Początkowa wartość wejścia (0x7F – pierwsza wartość do przechodzenia wiodącego zera)

Następnie, dopóki żaden przycisk nie zostanie wciśnięty, program będzie cały czas zmieniał wartość wejścia, zerując cyklicznie wejścia portu dla wierszy. Po wciśnięciu przycisku flaga „bM" zmieni wartość na 0, przez co wspomniany proces się skończy, a uruchomiona zostanie funkcja „generowanieKoduKlawisza()", podająca odpowiednią wartość na port P1. Po odciśnięciu przycisku, pętla zmieniająca wiodące zero się ponownie uruchomi, a na wyjściu będzie podana wartość początkowa „0xFF".

Dzięki zastosowaniu instrukcji „switch", wciśnięcie w tym samym czasie drugiego przycisku nie zmienia wyświetlanej wartości.

Skalowalność programu

Załóżmy, że porty mikrokontrolera mają nieograniczoną liczbę wejść/wyjść. Wtedy dołożenie większej ilości klawiszy byłoby możliwe oraz bardzo proste. Wtedy w ramach programu trzeba jedynie pamiętać, że przejść wiodących zer musi być po każdym wejściu dla kolumny, a ilość warunków do generowania kodu klawisza na wyjście musi być równe iloczynowi wszystkich kolumn oraz wszystkich wierszy.

Wtedy w zależności od liczby wejść P2 równe n, można podłączyć maksymalnie $\left\lfloor \frac{n}{2} \right\rfloor * \left\lceil \frac{n}{2} \right\rceil$ klawiszy.

Również w zależności od liczby wejść P1 równe m, można wyświetlić maksymalnie 2^m różnych wartości.