Alicja Kapiszka
Paweł Adamski
Informatyka N1, gr. 30C

Sprawozdanie nr 2

Obsługa wyświetlacza siedmiodegmentowego LED

Wstęp

Przedmiotem laboratoriów był 8-bitowy mikrokontroler 8051 z rodziny MCS-51, wykorzystywany już na poprzednich laboratoriach. Do przetestowania po raz kolejny wykorzystany został zestaw uruchomieniowy ZL2MCS51. Używanymi elementami płyty były diody LED (podłączone do portu P2) oraz 4 wyświetlacze siedmiosegmentowe LED (złożone z 7 diód LED tworzących kształt cyfry 8), które podłączono do portu 0. Podobnie jak na poprzednich zajęciach, wykorzystano oprogramowanie MIDE-51 i Flip. Program pisano jednak już nie w Asemblerze, a w języku C, przez co obliczenia czasu potrzebnego do wykonania napisanego kodu wymagały wykorzystania układu czasowo-licznikowego mikrokontrolera i układu przerwań, co było głównym celem niniejszych laboratoriów.

Mikrokontroler 8051 posiada dwa układy czasowo-licznikowe – TIMER 0 i TIMER 1. Każdy z nich taktowany jest 1/12 sygnału zegarowego i może pracować w 4 trybach, przy czym na laboratoriach korzystano z trybu licznika 16-bitowego. Z układem czasowym związane są 3 rejestry specjalnego przeznaczenia: TMOD (określający tryb pracy układu czasowo-licznikowego, który ustawia się za pomocą masek), TCON (rejestr sterujący, ustawiany bitowo) oraz rejestry licznikowe (TH0, TL0, TH1, TL1), w których przechowywana jest wartość odliczonego czasu (TL0 – młodszy bajt licznika 0, TH0 – starszy bajt tego licznika, analogiczna dla licznika 1).

W ćwiczeniu wykorzystano również kontroler przerwań. Źródłem przerwań mogą być sygnały zewnętrzne lub wewnętrzne, jak układ czasowo-licznikowy. Każde źródło przerwania jest określone symbolem i numerem przerwania (np. dla układu czasowo-licznikowego 0 jest to TF0 i numer przerwania 1) oraz ma przypisany stały adres procedury obsługi przerwania. Ponadto kontroler przerwań umożliwia za pośrednictwem rejestru IE blokowanie przerwań – w całości (ustawienie bitu EA na 0 lub 1) lub oddzielnie (np. ET0 dla licznika nr 0).

Celem laboratoriów było właściwe podłączenie zestawu uruchomieniowego oraz wyświelacza LED, a następnie wykonanie zadań związanych z obsługą wyświetlacza, wykorzystując układ czasowo-licznikowy mikrokontrolera oraz system przerwań. Kontrola czasowa jest potrzebna w celu regulacji częstotliwości świecenia diod wyświetlaczy tak, aby wyświetlane symbole wydawały się ciągłe dla ludzkiego oka (w rzeczywistości cyfry powinny zmieniać się cyklicznie z dużą częstotliwością).

Przebieg ćwiczenia:

- Zestaw uruchomieniowy podłączono do komputera za pomocą złącza RS-232 oraz zasilono, podłączając do stacji NI ELVIS.
- Podłączono wyświetlacz LED – złącze JP2 (sygnały sterujące segmentami wyświetlaczy – informacje o tym, co ma zostać wyświetlone) połączono ze złączem JP10 (port 0), a złącze JP3

(sygnały sterujące wzmacniaczami elektrod wspólnych) – ze złączem JP4 (port 1), przy czym w tym przypadku wykorzystano tylko 4 bity.

- Na laboratoriach pracowano, wykorzystując TIMERy jako licznik 16-bitowy, w związku z tym ustawiono rejestr TMOD (za pomocą maski) na wartość odpowiadającą trybowi 1, czyli M1 = 0 i M0 = 1.
- W rejestrze TCON zmieniono wartości TR0 i TR1 na wartość 1, czyli włączono programowe włączanie układów czasowo-licznikowych.
- W celu umożliwienia wyświetlania cyfr dziesiętnych, utworzono tablicę odpowiadających im wartości szesnastkowych (wyliczonych z wartości 0 lub 1, które należy podać na odpowiednie diody wyświetlacza tak, by otrzymać daną liczbę).
- W celu umożliwienia programowania mikrokontrolera 8051 w języku C, do programów dołączono plik nagłówkowy „8051.h”.

Zadanie 1.

Należało zaimplementować gotowy kod i określić jego funkcje, a następnie sprawdzić wpływ stałych TH0_RELOAD, TL0_RELOAD i TIK na częstotliwość migania diody.

- Zadaniem programu było wykorzystanie układu czasowo-licznikowego 8051 (TH0 i TL0) oraz kontrolera przerwań (TIK) w celu uzyskania migania diody na płytce.
- Zwiększenie wartości TH0 i TL0 powodowało zwiększenie częstotliwości migania (aż do ciągłego świecenia się diody).
- Zwiększenie wartości TIK powodowało zmniejszenie częstotliwości migania (gdyż dioda świeciła się tylko co któreś przerwanie).

Zadanie 2.

Należało napisać program, którego efektem miało być wyświetlenie na wyświetlaczu LED bieżącego roku. W tym celu należało odpowiednio dobrać częstotliwość układu czasowo-licznikowego, aby zminimalizować migotwanie wyświetlaczy.

```
#include "8051.h" // zbiór definiujący rejestry procesora

#define TH0_RELOAD 0x00 //czas odliczania
#define TL0_RELOAD 0x00
#define TIK 5

void timer0_init(void)
{
      TH0 = TH0_RELOAD;
      TL0 = TL0_RELOAD;
      TMOD = TMOD | 0x01; // tryb nr 1 układu TIMER 0
      TR0 = 1; // TIMER 0 start
      ET0 = 1; // odblokowanie przerwań od TIMER 0
}

void timer_isr (void) __interrupt (1) __using (0)
{
      static int count=0;
      TH0 = TH0_RELOAD;
      TL0 = TL0_RELOAD;
      count++;
```

```
      if (count==TIK) // dodatkowy licznik przerwań (dod. opóźnienia)
      {
            count=0;
            P2_7=!P2_7;
      }
}
void main(void)
{
      EA = 0; // zablokowanie przerwań
      timer0_init(); // przygotowanie układu Timer0
      EA = 1; // odblokowanie przerwań
      P2=0xFF; // wygaszenie wszystkich diod
      while(1)
      {
            P2_6 = P3_6; // obsługa przycisku i diody

            P0=0x5B; // wyświetlenie cyfry 2
            P1=0x08; // na wyświetlaczu nr 4
            for(i=0; i<255; i++); // pętla opóźniająca

            P0=0x3F; // wyświetlenie cyfry 0
            P1=0x04; // na wyświetlaczu nr 3
            for(i=0; i<255; i++);

            P0=0x06; // wyświetlenie cyfry 1
            P1=0x08; // na wyświetlaczu nr 2
            for(i=0; i<255; i++);

            P0=0x07; // wyświetlenie cyfry 7
            P1=0x08; // na wyświetlaczu nr 1
            for(i=0; i<255; i++);

      }
}
```

Zadanie 3.

Zadanie polegało na takiej modyfikacji programu z poprzedniego zadania, by uzyskać efekt wędrowania cyfr (zapalania się po kolei każdej cyfry).

```
#include "8051.h"

#define TH0_RELOAD 0xF7 // czas odliczania
#define TL0_RELOAD 0x00
#define TIK 1
int wysw = 1 // nr aktywnego wyświetlacza

void timer0_init(void)
{
      TH0 = TH0_RELOAD;
      TL0 = TL0_RELOAD;
      TMOD = TMOD | 0x01;
      TR0 = 1;
      ET0 = 1;
}

void timer_isr (void) __interrupt (1) __using (0)
{
```

```
	static int count=0;
	TH0 = TH0_RELOAD;
	TL0 = TL0_RELOAD;
	count++;
	if (count==TIK)
	{
		count=0;
		P2_7=!P2_7;
		wysw *= 2; // zmiana aktywnego wyswietlacza na kolejny
		if(wysw == 16) wysw = 1;
	}
}
void main(void)
{
	EA = 0;
	timer0_init();
	EA = 1;
	P2=0xFF;
	while(1)
	{
		P2_6 = P3_6;

		P1 = wysw;
		switch(P1)
		{
			case 1:
				P0=0x07;
				break;
			case 2:
				P0=0x06;
				break;
			case 4:
				P0=0x3F;
				break;
			case 8:
				P0=0x5B;
				break;
		}
	}
}
```

Zadanie 4.

Zadanie polegało na takiej modyfikacji programów z poprzednich zadań, by wyświetlić na wyświetlaczu 4 cyfry, których wartości zwiększałyby się po naciśnięciu odpowiedniego przycisku.

```
#include "8051.h"

#define TH0_RELOAD 0xF7
#define TL0_RELOAD 0x00
#define TIK 1
int wysw = 1 // nr aktywnego wyświetlacza

void timer0_init(void)
{
	TH0 = TH0_RELOAD;
	TL0 = TL0_RELOAD;
	TMOD = TMOD | 0x01;
```

```
	TR0 = 1;
	ET0 = 1;
}

void timer_isr (void) __interrupt (1) __using (0)
{
	static int count=0;
	TH0 = TH0_RELOAD;
	TL0 = TL0_RELOAD;
	count++;
	if (count==TIK)
	{
		count=0;
		P2_7=!P2_7;
		wysw *= 2; // zmiana aktywnego wyswietlacza na kolejny
		if(wysw == 16) wysw = 1;
	}
}

char nums[10] = {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F,
0x6F}

void main(void)
{
	int num0 = 0; // zainicjowanie wyswietlanych cyfr
	int num1 = 2;
	int num2 = 3;
	int num3 = 4;

	EA = 0;
	timer0_init();
	EA = 1;
	P2=0xFF;

	while(1)
	{
		P2_6 = P3_6;

		if(!P3_0) // zwiększenie wartości po wciśnięciu przycisku 0
		{
			num0++;
			if(num0>9)
				num0=0;
		}
		if(!P3_1)
		{
			num1++;
			if(num1>9)
				num1=0;
		}
		if(!P3_2)
		{
			num2++;
			if(num2>9)
				num2=0;
		}
		if(!P3_3)
		{
			num3++;
			if(num3>9)
				num3=0;
```

```
            }

            P1 = wysw;
            switch(P1)
            {
                  case 1:
                        P0=nums[num0];
                        break;
                  case 2:
                        P0= nums[num1];
                        break;
                  case 4:
                        P0= nums[num2];
                        break;
                  case 8:
                        P0= nums[num3];
                        break;
            }
      }
}
```

Otrzymano efekt zwiększania się wartości cyfr po wcisnięciu przycisku odpowiadającemu danemu segmentowi wyświetlacza, jednak niepożądanym zjawiskiem było „przeskakiwanie” cyfr o kilka wartości na raz. Było to spowodowane drganiem ze styków, które zostanie wyeliminowane na kolejnych laboratoriach.