Sprawozdanie na przedmiot Systemy wbudowane

Dominik Lewczyński

155099

IO grupa 3

Zadanie 1 oraz Zadanie 2

Było wykonane

z Rafałem Olszewskim

Olsztyn 2022

Spis treści

Zadanie 1

W tym zadaniu należało stworzyć program w którym przełączamy się pomiędzy 9 podprogramami, które wykonują odpowiednie zadanie oraz wgrać go do mikrokontrolera PIC24

Obsługa programu:

- Przycisk RD13 – następny program
- Przycisk RD6 – poprzedni program

Żeby przejście miedzy podprogramami działało poprawnie stworzono na początek zmienna typu **int** o nazwie **pom** i nadano jej wartość początkową 1, co oznaczało pierwszy podprogram, następnie napisano instrukcje warunkową **if**, która wygląda na stepująca

```
if (PORTDbits.RD13 == 0)
{
        pom++;
}
else if(PORTDbits.RD6 == 0)
{
        pom--;
}
```

Działanie tej instrukcji wygląda następująco: Po naciśnięciu przycisku **RD13** wartość zmiennej **pom** jest zwiększona o 1, natomiast po naciśnięciu przycisku **RD6** Wartość zmiennej **pom** jest zmniejszona o 1.

Następnie stworzono kilka instrukcji **if**, w których sprawdzano wartość zmiennej **pom** np.: jeśli zmienna będzie miała wartość 2 ma później nastąpić wywołanie funkcji, która jest jednym z podprogramu. Przykład takiej instrukcji wygląda następująco:

```
if(pom==2) display=BIN2(display);
```

8 bitowy licznik binarny zliczający w górę (0...255)

```
int BIN1(unsigned char display)
{
        display=display+1;
        return display;
}
```

Funkcja przyjmuje jako argument aktualny stan zmiennej **display** i zawraca kolejną liczbę w postaci zapisu binarnego zwiększając ją o 1

8 bitowy licznik binarny zliczający w dół (255...0)

```
int BIN2(unsigned char display)
{
        display=display-1;
        return display;
}
```

Funkcja ta działa identycznie jak poprzednie, czyli przyjmuje jako argument aktualny stan zmiennej **display** i zawraca kolejną liczbę w postaci zapisu binarnego, ale tym razem zmniejsza ją o 1

8 bitowy licznik w kodzie Graya zliczający w górę (repr. 0...255) oraz 8 bitowy licznik w kodzie Graya zliczający w dół (repr. 255...0)

Uznano by stworzyć jedną funkcje dla licznika zliczającego w górę jak i w dół. Funkcja wygląda następująco:

```
int GRAY(unsigned char display, int x)
{
        display=x^(x>>1);
        return display;
}
```

Funkcja przyjmuje jako argument aktualny stan zmiennej **display** oraz zmienna **x** typu **int**. Funkcja zwraca kolejną liczbę w postaci kodu **Gray'a**. W głównej funkcji main stworzono zmienną **x**, które nadano wartość początkową 0.

```
if(pom==3)
{
        display=GRAY(display,x);
        x++;
}
if(pom==4)
{
        display=GRAY(display,x);
        x--;
        if(x<0)x=255;
}
```

Jak można zauważyć kiedy zmienną **pom** przyjmie wartość 3 następuje wywołanie funkcji **GRAY** a następnie zmienna **x** jest zwiększana o 1 . W przypadku **pom = 4** sytuacja jest podobna tylko zmienna **x** jest zmniejszana o 1 następnie dodano instrukcje **if**, która ma zamienić wartość **x** z 0 na 255 ponieważ początkową wartość nadano 0 a ma zliczać od 255 do 0.

2x4 bitowy licznik w kodzie BCD zliczający w górę (0...99) oraz 2x4 bitowy licznik w kodzie BCD zliczający w dół (99...0)

Uznano by stworzyć jedną funkcje dla licznika zliczającego w górę jak i w dół. Funkcja wygląda następująco:

```
int BCD(unsigned char display, int x)
{
        display = ((x/10)*16)+(x%10);
        return display;
}
```

Funkcja przyjmuje jako argument aktualny stan zmiennej **display** oraz zmienna **x** typu **int**. Funkcja zwraca kolejną liczbę w postaci kodu **BCD**. W głównej funkcji main stworzono zmienną **x** które nadano wartość początkową 0.

```
if(pom==5)
{
        display=BCD(display ,x);
        x++;
}
if(pom==6)
{
        display=BCD(display, x);
        x--;
        if(x<0)x=99;
}
```

Jak można zauważyć kiedy zmienną **pom** przyjmie wartość 5 następuje wywołanie funkcji **BCD** a następnie zmienna **x** jest zwiększana o 1 . W przypadku **pom = 6** sytuacja jest podobna tylko zmienna **x** jest zmniejszana o 1 następnie dodano instrukcje **if** która ma zamienić wartość **x** z 0 na 99 ponieważ początkową wartość nadano 0 a ma zliczać od 99 do 0

3 bitowy wężyk poruszający się lewo-prawo

```
unsigned char SNAKE(unsigned char display, int pom){
        int l = 7;
        int i=0;
        unsigned char arr[6] = {l,l*2,l*4,l*8,l*16,l*32};

        for(i; i<6; i++)
        {
                if(display==arr[i]&&pom==0){
                        return display<<1;
                }
                else if(display==arr[i]&&pom==1){
                        return display>>1;
                }
        }
        return l;
}
```

Funkcja ma za zadanie zrobić 3 bitowego węża poruszającego się w lewo i w prawo. Funkcja przyjmuję jako argument aktualny stan zmiennej **display** oraz zmienna **pom** typu **int**. Tworzymy zmienną **l** typu **int** która przechowują wartość 7, która odpowiada zapalonym 3 diodom. Następnie tworzymy tablice **arr[6]**, która przechowuję informacje o diodach.

W pętli **for** iterujemy po tablicy i następnie w instrukcji warunkowej **if** sprawdzamy czy elementy tablicy jest równy wartości **display** oraz czy **pom** jest równy 0. Jeśli warunek jest spełniony zwracany jest wartość **display << 1** oznaczające przesuniecie bitowe o 1 czyli 3 diody poruszają się w lewo, natomiast jeśli wartość tablicy jest równy **display** oraz **pom** ma wartość 1 następuje zwrócenie wartości **display >> 1** oznaczającej przesuniecie bitowe w prawo o 1 czyli 3 diody przesuwają się w prawo.

Kolejka

```
unsigned char QUEUE(unsigned char display)
{
	unsigned char i=0,j;
	unsigned char arr[7] = {254,252,248,240,224,192,128};
	for(i; i<7; i++){
		if(display==255) return 1;
		if(display>=arr[i]){
			j=display%arr[i];
			if(j==0) return display|1;
			return display-j|j<<1;
		}
	}
	return display<<1;
}
```

Funkcja ma za zadanie wygenerować kolejkę. Przyjmuje jako argument aktualny stan zmiennej **display**. Następnie jest tworzona tablica **arr** zawierająca 7 elementów, którymi są odpowiednie dane o diodach np.: liczba 128 w DEC oznacza diody zapalone w postaci 1000 000 w BIN. W pętli **for** iterujemy po tablicy i sprawdzane są warunki. Kiedy stan **display** osiągnie wartość 255 ma zwrócić 1, czyli wszystkie diody będą zapalone i następnie mają zgasnąć i ma się zapalić pierwsza dioda. Jeśli warunek nie został spełniony przechodzimy do kolejnej instrukcji **if** sprawdzającej czy aktualny stan zmiennej **display** jest większy bądź równy elementowi z tablicy.

Jeśli warunek jest spełniony, wyliczana jest reszta z dzielenia **display** przez element z tablicy i wartość jest zapisywana w zmiennej **j**, którą utworzono wcześniej, potem sprawdzane jest czy reszta z dzielenia jest równa zero i jeśli się zgadza ma zwrócić wartość z działania **display | 1** czyli mają zostać zapalone aktualnie diody a potem ma się zapalić pierwsza dioda. Natomiast jeśli warunek nie będzie spełniony po prostu ma zwrócić wartość działania **display-j | j<<1**, czyli następuje przesuwanie pierwszej diody do momentu aż dotrze do wartości odpowiadającej tej w tablicy **arr** . Kiedy przeszło się przez całą tablice funkcja ma zwrócić wartość działania **display <<1** i ta sytuacja występuje wyłącznie wtedy, kiedy mamy do czynienia ze stanem **display** mniejszym niż 128.

6 bitowy generator liczb pseudolosowych oparty o konfigurację 1110011

```
unsigned char PRNG_LFSR_1110011(unsigned char display){
        unsigned char xor=1&((display)^(display>>1)^(display>>4)^(display>>5));
        return display>>1|xor<<5;
}
```

Funkcja ma za zadanie losować liczby pseudolosowe oparte o konfiguracje 1110011. Funkcja przyjmuję jako argument aktualny stan zmiennej **display**, a następnie wykonuje działanie **xor=1&((display)^(display>>1)^(display>>4)^(display>>5))**. Po otrzymaniu wyniku funkcja zwraca wartość działania **display>>1|xor<<5**.

Zadanie 2

W tym zadaniu należało stworzyć program imitujący alarm. Należało wgrać ten program do mikrokontrolera PIC24. Źródłem danych jest czujnik temperatury TC 1047. Alarm ma się załączyć po przekroczeniu temperatury 25 stopni potem ma migać dioda i po 3 sekundach mają się zapalić wszystkie. Po naciśnięciu przycisku RD6 na mikrokontrolerze ma nastąpić wyłączenie diod i ponowne działanie programu.

Obsługa programu:

- Czujnik temperatury TC 1047 – Rejestrowanie temperatury
- Przycisk RD6 – wyłączenie alarmu

```
unsigned int czas = 0;
unsigned char lampka = 0;
unsigned int wylacznik = 0;
while (1) {
        PORTA=(unsigned int) display;
        for (i = szybkosc * SCALE; i > 0; i--) Nop();
        temp = readADC(TSENS);
        if(wylacznik == 0)
        {
                if(temp > 250)
                {
                        czas += szybkosc;
                        lampka = 1 - lampka;
                        display = lampka;
                        if(czas >= 3000)
                        {
                                display = 255;
                        }
                }
```

```
                else
                {
                        display = 0;
                        czas = 0;
                }
        }
        if(PORTDbits.RD6 == 0)
        {
                wylacznik = 1 - wylacznik;
                display = 0;
                czas = 0;
        }
}
```

Do poprawnego działania stworzono 3 zmienne **czas**, **lampka** oraz **wylacznik** i nadano im wartości początkowe równe 0. W nieskończonej pętli umieszczono instrukcje, która działa następująco. Kiedy **wylacznik** ma wartość zero, czyli nie jest wciśnięty żaden przycisk wtedy działa czujnik i pobiera dane o temperaturze. Kiedy wartość **temp** przekroczy wartość 250, która odpowiada 25 stopniom, ma nastąpić przyspieszenie migania diody. Potem po 3 sekundach maja się zapalić wszystkie diody. Kiedy przycisk RD6 zostanie wciśnięty następuje wyzerowanie oraz ponownie następuje rejestrowanie temperatury.

Zadanie 3

Zadanie polegało na stworzeniu reklamy o dowolnej tematyce w edytorze MPLAB X IDE v6.00, które ma się wyświetlać na wyświetlaczu LCD 16x2, a następnie wgrać program do symulatora, który symuluje mikrokontroler PIC18F4620.

Obsługa programu:

- Nie ma obsługi przycisków. Reklama działa w nieskończonej pętli

Reklama została napisana za pomocą komend pokazanych na zajęciach. Reklama wyświetla się prawidłowo. Składa się z 77 linijek kodu. Kod programu jest dostępny w pliku „main.c” w folderze Zadanie3 oraz znajduje się tam plik „Zadanie3.X.production.hex”, który można wgrać do symulatora.

Fragment kodu reklamy wyglądu następująco:

```
delay(500);
lcd_cmd(L_CLR);
lcd_cmd(L_L1); //Ustawienie karetki w pierwszej linii
lcd_str("|  Zapraszamy  |"); //napis
lcd_cmd(L_L2); //Przej??ie do drugiej linii
lcd_str("|         do        |");
```

Zadanie 4

Zadanie polegało na napisaniu programu w edytorze MPLAB X IDE v6.00, który ma imitować działanie kontrolera kuchenki mikrofalowej. Program należy wgrać do symulatora który symuluje mikrokontroler PIC18F4620. Komunikaty mają się wyświetlać na wyświetlaczu LCD 16x2.

Obsługa programu:

- RB5 –wybór mocy 800W, 600W, 350W, 200W
- RB4 –dodanie czasu 1min
- RB3 –dodanie czasu 10s
- RB2 –Start/Stop
- RB1 –Reset

```
int moc = 800;
int czas = 0;
int stop = 0;
```

Zostały utworzone 3 zmienne: moc, czas oraz stop i przekazano im wartości początkowe moc = 800, czas = 0 oraz stop = 0. W nieskończonej pętli while zostały utworzone instrukcje warunkowe odpowiadające za działania przycisków oraz za działanie samego zliczania czasu oraz kod umożliwiający wyświetlenie wartości na wyświetlaczu LCD. Kod programu jest dostępny w pliku „main.c” w folderze Zadanie4 oraz znajduje się tam plik „Zadanie4.X.production.hex”, który można wgrać do symulatora.

```
if (PORTBbits.RB5 == 0) {
        moc =
                moc == 800 ? 600 :
                moc == 600 ? 350 :
                moc == 350 ? 200 :
                moc == 200 ? 800 :
                800;
}
```

Ta instrukcja warunkowa odpowiada za zmianę wartości wat poprzez wciśniecie przycisku RB5. Kiedy przycisk RB5 jest wciśnięty zmienna moc przyjmuje odpowiednia wartość np.: jeśli miała wartość 800 przyjmie wartość 600 itd.

```
if (PORTBbits.RB4 == 0) {
        czas += 60;
        if (czas > 3600) {
           czas = 0;
        }
        continue;
}
```

Ta instrukcja warunkowa odpowiada za dodanie 60 sekund do czasu poprzez wciśniecie przycisku RB4. Kiedy przycisk RB4 jest wciśnięty, do zmiennej czas zostanie dodane 60 sekund. Kiedy wartość czas będzie większa niż 3600 sekund (1 godzina) to czas przyjmie wartość 0.

```
if (PORTBbits.RB3 == 0) {
        czas += 10;
        if (czas > 3600) {
           czas = 0;
        }
        continue;
}
```

Ta instrukcja warunkowa odpowiada za dodanie 10 sekund do czasu poprzez wciśniecie przycisku RB3. Kiedy przycisk RB3 jest wciśnięty, do zmiennej czas zostanie dodane 10 sekund. Kiedy wartość czas będzie większa niż 3600 sekund (1 godzina) to czas przyjmie wartość 0.

```
if (PORTBbits.RB2 == 0) {
        if (stop == 0) {
                stop = 1;
        } else {
                stop = 0;
        }
        continue;
}
```

Ta instrukcja warunkowa odpowiada za uruchomienie procesu poprzez wciśniecie przycisku RB2. Kiedy przycisk RB2 jest wciśnięty następuje sprawdzenie czy stop ma wartość 0 czyli proces jest wyłączony i jeśli tak jest to stop przyjmuje wartość 1 czyli zostaje uruchomiony proces. Po ponownym naciśnięciu przycisku następuje wyłączenie procesu.

```
if (PORTBbits.RB1 == 0) {
        if (stop == 0) {
           moc = 800;
           czas = 0;
        }
        continue;
}
```

Ta instrukcja warunkowa odpowiada za reset wartość moc oraz czas do wartości początkowych poprzez wciśniecie przycisku RB1. Kiedy przycisk RB1 jest wciśnięty następuje sprawdzenie czy stop ma wartość 0 czyli proces jest wyłączony i jeśli tak jest zmienne moc oraz czas przyjmują wartości początkowe moc = 800 oraz czas = 0.

```
if (stop == 1) {
        if (czas > 0) {
           czas--;
           delay(800);
        } else {
           stop = 0;
        }
}
```

Ta instrukcja warunkowa odpowiada za odliczanie czasu w momencie kiedy uruchomimy proces. Kiedy stop ma wartość 1 czyli proces jest włączony przechodzimy do kolejnej instrukcji if. W tej instrukcji sprawdzamy czy czas jest większy od 0 i jeśli tak jest czas zostaje zmniejszany o 1. W przeciwnym wypadku stop przyjmuje wartość 0 czyli proces zostaje wyłączony.

```
char pierwsza_linijka[] = "MOC:          W";
pierwsza_linijka[12] = moc / 100 + '0';
pierwsza_linijka[13] = (moc / 10 - (moc / 100 * 10)) + '0';
pierwsza_linijka[14] = (moc - (moc / 100 * 100) - ((moc / 10 - (moc / 100 * 10)) * 10)) + '0';

int minutes = czas / 60;
int seconds = czas - (minutes * 60);

char druga_linijka[] = "Czas:          ";
druga_linijka[7] = stop ? ' ' : '|';
druga_linijka[8] = stop ? ' ' : '|';
druga_linijka[11] = minutes / 10 + '0';
druga_linijka[12] = minutes - (minutes / 10 * 10) + '0';
druga_linijka[13] = ':';
druga_linijka[14] = seconds / 10 + '0';
druga_linijka[15] = seconds - (seconds / 10 * 10) + '0';

lcd_cmd(L_L1);
lcd_str(pierwsza_linijka);
lcd_cmd(L_L2);
lcd_str(druga_linijka);
```

Ten fragment kodu odpowiada za wyświetlenie wartości mocy oraz czasu na wyświetlaczy LCD. Zostają utworzone dwie tablice pierwsza_linijka, która przechowuje wyświetlenie mocy oraz druga_linijka, która przechowuje wyświetlenie czasu. W tablicy pierwsza_linijka na indeksach 12, 13 i 14 znajdują się informacje o wyświetlaniu mocy w postaci liczby setek dziesiątek oraz jedności. Powstały dwie zmienne minutes oraz seconds przechowujące informacje o minutach oraz sekundach. W tablicy druga_linijka na indeksach 7 i 8 zostaje wyświetlone || w zależności czy odliczanie jest wyłączone czy włączone. W tablicy druga_linijka na indeksach 11 - 15 zostaje wyświetlone minuty oraz sekundy w odpowiedniej postaci.

Źródła wykorzystane w czasie pracy

- W zadaniach również wykorzystaliśmy zapiski z tablicy
- Kod Graya: Kod Graya – Wikipedia, wolna encyklopedia
- Kod BCD: arduino - BCD to Decimal And Decimal to BCD - Stack Overflow
- Operatory bitowe: Operatory bitowe (helion.pl)

Problemy w czasie implementacji

Problemem było przejście z węża do kolejki, a dokładniej chodzi o to, że po przejściu kolejka otrzymuje aktualny stan display, który znajdował się podczas przełączania z węża. Nie zauważyliśmy tego problemu w porę co skutkowało tym, iż nie zdążyliśmy go poprawić.

Osoby, które były pomocne

- Paweł Majorowski pomógł w problemie z czujnikiem temperatury.
- W zadaniu 4 miałem problem dodaniem int do tablicy char. W rozwiązaniu tego pomógł mi Rafał Olszewski, który podpowiedział mi, aby użyć + '0'.