Лабораторна робота №3

Тема: ПРОГРАМА КЕРУВАННЯ НАПРУГОЮ НА ПІНАХ МІКРОКОНТРОЛЕРА

Мета: Розробити пристрій з дискретними входами та виходами.

Теоретичні відомості

Для мікроконтролерів існують засоби подавати на окремі лапки мікросхеми низьку чи високу напругу. Для процесорів AVR для цього використовують порти паралельного вводу-виводу, де можна керувати кожним окремим бітом. Порти вводу-виводу позначають послідовними літерами латинського алфавіту. Ми розглядатиме порт B.

Керування портом B використовують трійку регістрів, які зображені в оперативну пам’ять за адресами:

Порт (ОПЗ)	Назва регістру	Призначення регістру
$18($38)	PORTB	Data Register, Port B
$17($37)	DDRB	Data Direction Register Port B
$16($36)	PINB	Input pins, Port B

З причини, що порти вводу-виводу зображені в оперативній пам’яті, можна скористатися записом **`out 18h,R24`**, для запису вмісту 24-го регістра до PORTB, або записати те саме значення до ділянки пам’яті за адресою **`38h`**.

Розташування лапок процесора та їх позначення можна побачити на наступному рисунку:

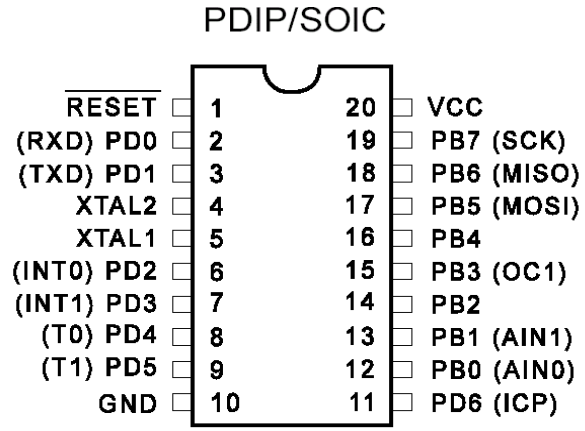

Рис. 1 — Корпус DIP Attiny2313 та його позначення виводів

Також принцип об'єднаності регістрової пам’яті, регістрів вводу-виводу зображено на наступній таблиці:

Таблиця 1 — Організація пам’яті Attiny2313

Регістровий файл	Область адрес даних
R0	$00
R1	$01
…	…
R30	$1E
R31	$1F

Регістри вв./вив	
$00	$20
$01	$21
…	…
$3E	$5E
$3F	$5F

Вбудоване ОЗУ
$60
$61
…
$DE
$DF

З таблиці видно, що обсяг вільної оперативної пам’яті є $DF-$60+1=128 байтів. Потрібно розпорядитися нею обережно, бо вона повинна містити стек та всі змінні, що будуть використані в програмі. Також стає зрозумілим, що використавши цей мікроконтролер не можна обробити великі масиви даних. Та незважаючи на малий обсяг оперативної пам’яті, процесора вистачає для вирішення широкого кола практичних задач.

Розглянемо більш детально регістри керування DDRB, PORTB, PINB. Регістри DDRB (вказує напрямок руху інформації), PORTB (використовується для виводу інформації та керування резисторами підтяжки) використовуються для читання та запису, регістр PINB (стан напруги на лапках порту) можна використати лише для читання.

DDRB це 8-ми розрядний регістр, за допомогою якого налаштовують лапку порту на вивід інформації чи на її прийняття. Кожна з лапок порту B має свій порядковий номер від 0 до 7, що зображає номера бітів двійкового числа:

7	6	5	4	3	2	1	0
1	0	0	1	0	1	0	1

Як приклад показано число 10010101_2=$75. Бінарний код означає, що одиничка відповідає за вивід з мікросхеми інформації, нуль — мікроконтролер буде лише визначати напругу з іншого під'єднаного пристрою (наприклад кнопки). Тому команда **`DDRB = 0x95;`** налаштує процесор так:

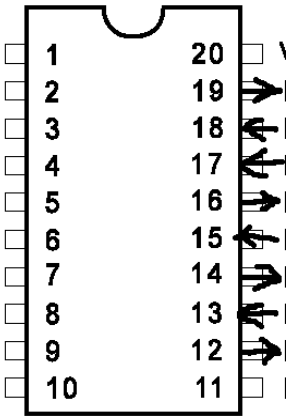

Рис. 2 — Результат налаштування **`DDRB = 0x95;`**

Для лапок мікросхеми, які налаштовані на вивід інформації, можна змінити їх стан на подачу напруги чи зняття її. Кожна лапка може видавати або споживати струм до 20мА, що достатньо для живлення світлодіодів без додаткових ключів та підсилювачів струму. Керувати станом напруги на лапках виводу інформації можна через регістр PORTB. Результати запису **`PORTB=0x84;`** показано на наступному малюнку:

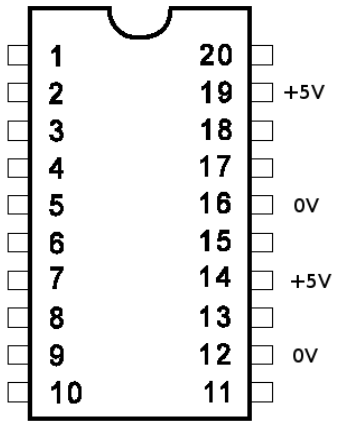

Рис. 3 — Вихідна напруга при поданні команди **`PORTB=0x84;`**

Завдяки такій схемі можна керувати напругою на кожній лапці мікросхеми (крім живлення та деяких службових). Прочитавши результат з регістру PINB ми завжди за одиничним бітом можемо визначити напруги на лапках мікросхеми, незалежно від того де ця напруга була встановлена: програмою з мікроконтролера чи зовнішнім пристроєм.

З метою ілюстрації використання отриманих знань напишемо програму, яка слідкуватиме за станом входу і при низькій напрузі, буде миготіти світлодіодом. Код програми з коментарями наведено нижче:

```
#include <avr/io.h>
 //Бібліотека з адресами портів

void my_delay(unsigned char p);
//прототип паузи

int main(void)
//Початок програми
{



    DDRB = 0x01; //Налаштовуємо PB0 як вихід
    while(1){
 //Цикл до вимкнення живлення
        if( (PINB&0x02)==0x02 )
//Якщо на PB1 є +5В
        {
 //то кнопка натиснута
          if( (PORTB&0x01)==0x01 )
 //Якщо на вихід
          {
```

```
 // вже одиниця
            PORTB &= !0x01; //то видаємо туди “0”

          }else{

            PORTB |= 0x01;
 //інакше засвітимо
              //світлодіод подавши “1”
          }

        }else{
 //Кнопка не натиснута
          PORTB &= !0x01;
 // видаємо “0”
        }

        my_delay(255);
 //Пауза перед зміною стану
        my_delay(255);
 //світлодіода
    }

    return 0;

}



void my_delay(unsigned char p){

    unsigned char i,j;
```

```
 //Для паузи робимо
                          //просто пусті цикли
    for(i=0;i<p;i++){

       j=255;

       while(j>0) j--;

    }

}
```

Завдання:

1. Кожен за останньою цифрою номеру в списку по журналу визначає свій варіант. (Для заочної форми навчання обирайте варіант за останньою цифрою номеру залікової книжки.)

2. Розробіть блок-схему алгоритму роботи.

3. Створіть програмне забезпечення, та перевірте роботу програми на симуляторі.

4. Випишіть asm-код з дебагера головної функції програми main().

_____

_____

_____

_____

_____

_____

_____

_____

_____

5. Дайте відповіді на контрольні питання.

(Повний перелік асемблерних команд процесора надано в файлі atmega8.pdf на сервері з матеріалами до предмету).

Варіанти завдання:

№	Завдання:
0	Порахуйте в змінній K кількість поданих імпульсів напруги з зовнішнього пристрою до BP3.
1	Подайте “1” на BP2, якщо відбудеться ситуація, що на BP0 та BP6 буде “1” одночасно.
2	Виведіть на BP4 значення протилежне значенню на BP2.
3	Реалізуйте булеву операцію **`BP2 = BP1 and BP0`**.
4	Реалізуйте булеву операцію **`BP3 = BP1 or BP4`**.
5	Реалізуйте булеву операцію **`BP7 = not(BP1 and BP0)`**.
6	Реалізуйте булеву операцію **`BP7 = not(BP3 or BP2)`**.
7	Реалізуйте булеву операцію **`BP1 = BP2 xor BP0`**.
8	Реалізуйте булеву операцію **`BP4 = not(BP5 xor BP1)`**.
9	Реалізуйте булеву операцію **`BP5 = BP4 and BP6`**.

Контрольні питання:

1. Запишіть асемблерний код програми, де значення записується(читається) до(з) регістру PORTB. Прокоментуйте цей код:

_____

_____

_____

_____

_____

2. Які виводи мікропроцесора матимуть налаштування на прийняття інформації після виконання команди DDRB=0xF0?

_____

_____

_____

_____

3. Чи містить асемблерний код неоптимальні ділянки, як їх можна вдосконалити?

_____

_____

_____

_____

_____

4. Для імітації біжучого вогника на восьми світлодіодах, які значення (в бінарному вигляді) потрібно по циклу з паузами писати в PORTB?

_____

_____

_____

_____

_____

_____

5. Як ви б реалізували затримку в 0.5 секунди у виконанні коду (без тексту програми, лише принцип)?

_____

_____

_____

_____

_____

_____