# Лабораторная работа №3

**Тема**: ПРОГРАММА УПРАВЛЕНИЯ НАПРЯЖЕНИЕМ НА пинах микроконтроллеров

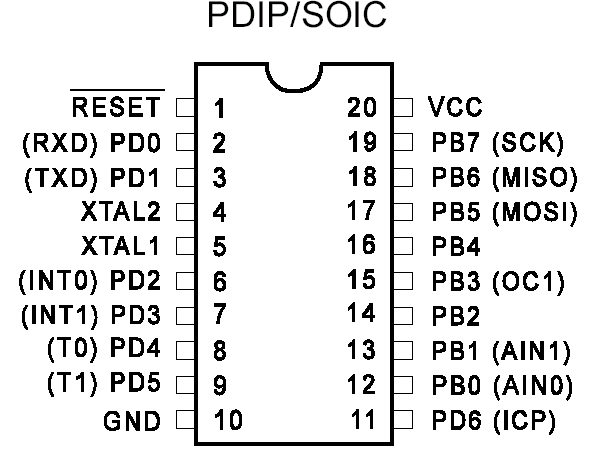
**Цель**: Разработать устройство с дискретными входами и выходами.

# теоретические сведения

Для микроконтроллеров существуют средства подавать на отдельные лапки микросхемы низкой или высокое напряжение. Для процессоров AVR для этого используют порты параллельного ввода-вывода, где можно управлять каждым отдельным битом. Порты ввода-вывода обозначают последовательными буквами латинского алфавита. Мы рассматривать порт В.

Управление портом В используют тройку регистров, которые изображены в оперативную память по адресам:

|  |  |  |
| --- | --- | --- |
| 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 |

По причине, что порты ввода-вывода изображены в оперативной памяти, можно воспользоваться записью out 18h, R24, для записи содержимого 24-го регистра к PORTB, или записать то же значение к участку памяти по адресу 38h.

Расположение лапок процессора и их обозначения можно увидеть на следующем рисунке:

  Рис. 1 - Корпус DIP Attiny2313 и его обозначения выводов

Также принцип объединённой регистровой памяти, регистров ввода-вывода показано на следующей таблице:

Таблица 1 - Организация памяти 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 | |

Из таблицы видно, что объем свободной оперативной памяти составляет $ DF- $ 60 + 1 = 128 байтов. Нужно распорядиться ею осторожно, потому что она должна содержать стек и все переменные, которые будут использованы в программе. Также становится понятным, что использовав этот микроконтроллер нельзя обработать большие массивы данных. Но несмотря на малый объем оперативной памяти, процессора хватает для решения широкого круга практических задач.

Рассмотрим более подробно регистры управления DDRB, PORTB, PINB. Регистры DDRB (указывает направление движения информации), PORTB (используется для вывода информации и управления резисторами подтяжки) используются для чтения и записи, регистр PINB (состояние напряжения на лапках порта) можно использовать только для чтения.

DDRB это 8-ми разрядный регистр, с помощью которого настраивают лапку порта на вывод информации или на ее принятие. Каждая из лапок порта имеет свой порядковый номер от 0 до 7, изображающая номера битов двоичного числа:

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

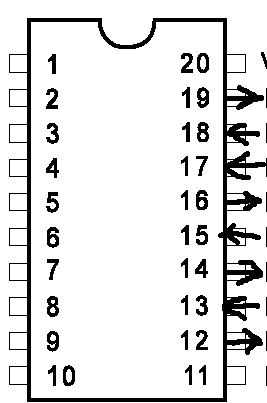
В качестве примера показано число 100101012 = $ 75. Бинарный код означает, что единичка отвечает за вывод из микросхемы информации, ноль - микроконтроллер будет только определять напряжение с другой подключенного устройства (например, кнопки). Поэтому команда DDRB = 0x95; настроит процессор так:

Рис. 2 - Результат настройки DDRB = 0x95;

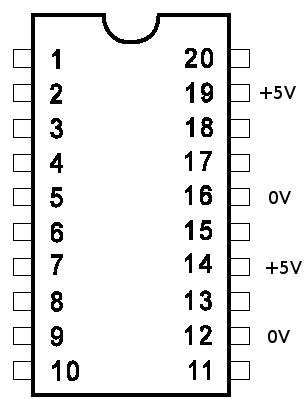
Для лапок микросхемы, которые настроены на вывод информации, можно изменить их состояние на подачу напряжения или снятия ее. Каждая лапка может выдавать или потреблять ток до 20мА, что достаточно для питания светодиодов без дополнительных ключей и усилителей тока. Управлять состоянием напряжения на лапках вывода информации можно через регистр PORTB. Результаты записи PORTB = 0x84; показано на следующем рисунке:

Рис. 3 - Выходное напряжение при подаче команды PORTB = 0x84;

Благодаря такой схеме можно управлять напряжением на каждой лапке микросхемы (кроме питания и некоторых служебных). Прочитав результат из регистра PINB мы всегда за единичным битом можем определить напряжения на лапках микросхемы, независимо от того, где это напряжение было установлено: программой с микроконтроллера или внешним устройством.

С целью иллюстрации использования полученных знаний напишем программу, которая будет следить за состоянием входа и при низком напряжении, будет мигать светодиодом. Код программы с комментариями приведены ниже:

#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--; }

Задание:

1. Каждый по последней цифре номера в списке по журналу определяет свой вариант. (Для заочной формы обучения выбирайте вариант по последней цифре номера зачетной книжки.)

2. Разработайте блок-схему алгоритма работы.

3. Создайте программное обеспечение, и проверьте работу программы на симуляторе.

4. Выпишите asm-код с дебаггера главной функции программы main ().

5. Дайте ответы на контрольные вопросы.

Варианты задания:

№ Задача:

0 Посчитайте в переменной K количество поданных импульсов напряжения с внешнего устройства к BP3.

1 Подайте "1" на BP2, если произойдет ситуация, что на BP0 и BP6 будет "1" одновременно.

2 Выведите на BP4 значение противоположное значению на BP2.

3 Реализуйте булеву операцию BP2 = BP1 and BP0.

4 Реализуйте булеву операцию BP3 = BP1 or BP4.

5 Реализуйте булеву операцию BP7 = not (BP1 and BP0).

6 Реализуйте булеву операцию BP7 = not (BP3 or BP2).

7 Реализуйте булеву операцию BP1 = BP2 xor BP0.

8 Реализуйте булеву операцию BP4 = not (BP5 xor BP1).

9 Реализуйте булеву операцию BP5 = BP4 and BP6.

Контрольные вопросы:

1. Запишите ассемблерный код программы, где значение записывается (читается) в (из) регистра PORTB. Прокомментируйте этот код:

LDI R24,0x46 // загрузка константы в регистр общего назначения

OUT 0x17,R24 // перемещения регистра в регистр порта

2. Какие выводы микропроцессора иметь настройки на принятие информации после выполнения команды DDRB = 0xF0?

PORTB = ob11110000;

3. Содержит ли ассемблерный код неоптимальные участки, как их можно усовершенствовать?

Да, ассемблерный код может содержат неоптимальные участки, усовершенствование кода прямо зависит от алгоритма

4. Для имитации бегущей огонька на восьми светодиодах, какие значения (в бинарном виде) нужно по циклу с паузами писать в PORTB?

Включение одной из ножек, временная задержка и отключение.

5. Как вы бы реализовали задержку в 0.5 секунды в исполнении кода (без текста программы, только принцип)?

Узнать частоту работы микроконтроллера, и сделать цикл с вставкой ассемблерного кода, или воспользовался бы функцией заголовочного файла (<avr/delay.h>)

Вариант №1

#include <avr/io.h>

int main(void)

{

DDRB = 0x46;

while(1)

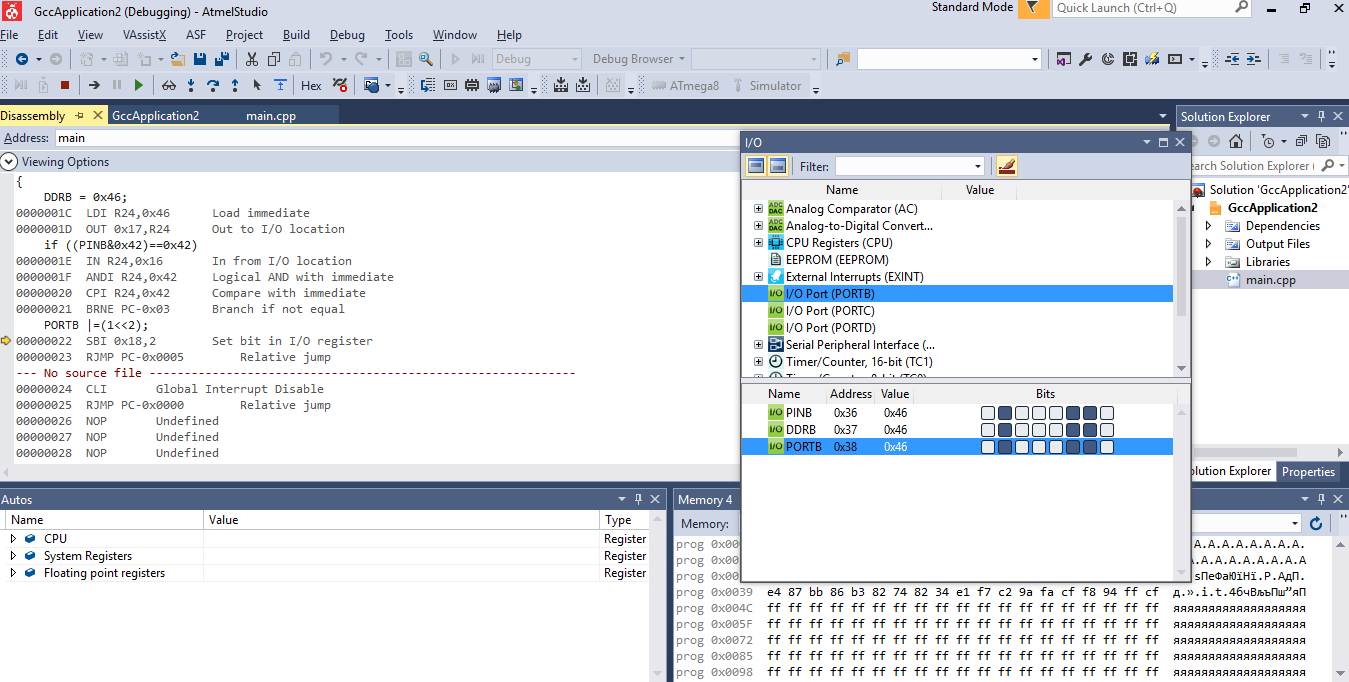
{

if ((PINB&0x42)==0x42)

PORTB |=(1<<2);

}

}



Считывание

Обработка

Листинг дизассемблированной программы

-- c:\users\aska\Documents\Atmel Studio\7.0\GccApplication2\GccApplication2\Debug/.././main.cpp

DDRB = 0x46;

0000001C LDI R24,0x46 Load immediate

0000001D OUT 0x17,R24 Out to I/O location

if ((PINB&0x42)==0x42)

0000001E IN R24,0x16 In from I/O location

0000001F ANDI R24,0x42 Logical AND with immediate

00000020 CPI R24,0x42 Compare with immediate

00000021 BRNE PC-0x03 Branch if not equal

PORTB |=(1<<2);

00000022 SBI 0x18,2 Set bit in I/O register

00000023 RJMP PC-0x0005 Relative jump