Министерство образования и науки
Российской Федерации

Федеральное государственное бюджетное образовательное
учреждение высшего образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра конструирования узлов и деталей радиоэлектронной аппаратуры
(КУДР)

А.А. Бомбизов, А.Г. Лощилов

РАБОТА С ПОРТАМИ ВВОДА-ВЫВОДА. ОРГАНИЗАЦИЯ ВЫВОДА ИНФОРМАЦИИ

Методические указания к выполнению
практических занятий и самостоятельной работы
по дисциплине «Программирование микроконтроллеров»

Томск 2017

1 Введение

Настоящий курс направлен на изучение основ программирования микроконтроллеров. Работа будет выполняться с использованием аппаратно-программных средств быстрого проектирования электронных устройств Arduino.

Целью настоящей работы является ознакомление с отладочной платой Arduino и освоение приемов программирования размещенных на ней портов ввода-вывода, настроенных на выход.

2 Краткая теория

Типичный микроконтроллер сочетает на одном кристалле функции процессора и периферийных устройств, а так же содержит ОЗУ и (или) ПЗУ для выполнения и хранения программ. По сути, это однокристальный компьютер, способный выполнять относительно простые задачи. Отличается от микропроцессора интегрированными в микросхему устройствами ввода-вывода, таймерами и другими периферийными устройствами. Как уже было сказано (п. 1), вся работа будет выполняться с использованием средств Arduino. Существует несколько версий платформ Arduino. Последняя версия Leonardo базируется на микроконтроллере ATmega32u4. Uno, как и предыдущая версия Duemilanove построены на микроконтроллере Atmel ATmega328. Старые версии платформы Diecimila и первая рабочая Duemilanoves были разработаны на основе Atmel ATmega168, более ранние версии использовали ATmega8. Arduino Mega2560, в свою очередь, построена на микроконтроллере ATmega2560 [1]. В рамках данного курса в практических работах будет использоваться Arduino UNO, которая обладает 14 цифровыми портами ввода-вывода, шесть из которых имеют возможность выдавать сигнал в виде широтно-импульсной модуляции. Помимо этого имеется 8 портов, обладающих возможностью оцифровывать входной аналоговый сигнал. Микроконтроллер обладает набором таймеров и прерываний. Имеет возможность обеспечить связь по различным интерфейсам типа RS-232, SPI, TWI.

Внешний вид платы изображен на рисунке 1. Краткое описание установленных на ней компонентов и разъемов следующее [2]:

1) Разъем Питания (от батареи) – может использоваться с блоками питания 9–12 Вольт;

2) Разъем USB (USB-порт) – может использоваться для питания схем, а также для связи с компьютером;

3) Индикатор (RX: прием) – используется для индикации приема данных, если конечно это прописано в программе;

4) Индикатор (TX: передача) – используется для индикации передачи данных;

5) Индикатор (порт 13: поиск неисправностей) – во время работы программы показывает правильно ли всё работает;

6) Порты (ARef, Ground, Digital, Rx, Tx) – опорное напряжение, земля, цифровые порты, порты приема и передачи данных.

7) Индикатор (индикатор питания) – сигнализирует о подаче питания на плату Arduino;

8) Reset (сброс) – ручной перезапуск платы Arduino, приводит к перезапуску вашей программы.

9) Разъем ICSP (порт для внутрисхемного программирования) – дает возможность программировать, минуя загрузчик самой платы.

10) Порты (Analog In, Power In, Ground, Power Out, Reset) – аналоговые, входящие, исходящие, питание и общий провод.

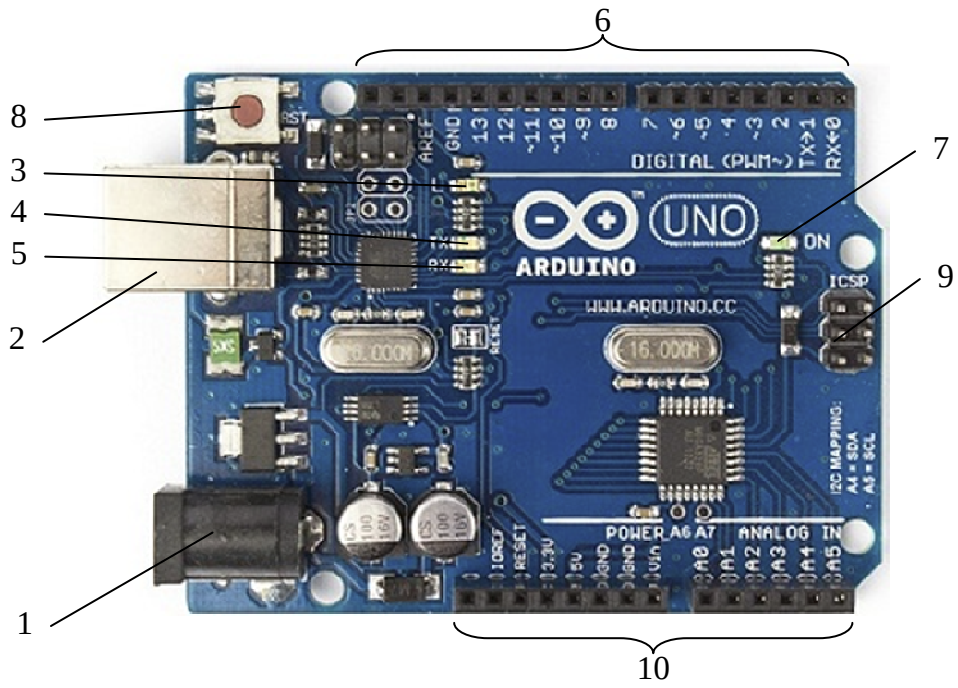

Рисунок 1 – Внешний вид Arduino UNO

В обычном виде плата Arduino мало пригодна для освоения принципов работы микроконтроллера. На ней нет кнопок, индикаторов и других средств взаимодействия с внешней средой. В связи с этим, производителями разработаны платы расширения для увеличения функциональных возможностей существующей отладочной платы. Как правило, большинство таких плат изготавливается под конкретный форм-фактор (в большинстве случаев Arduino UNO) и устанавливаются сверху на существующую отладочную плату. Одним из таких расширений является Multi-Function shield [3], внешний вид которого изображен на рисунке 2.

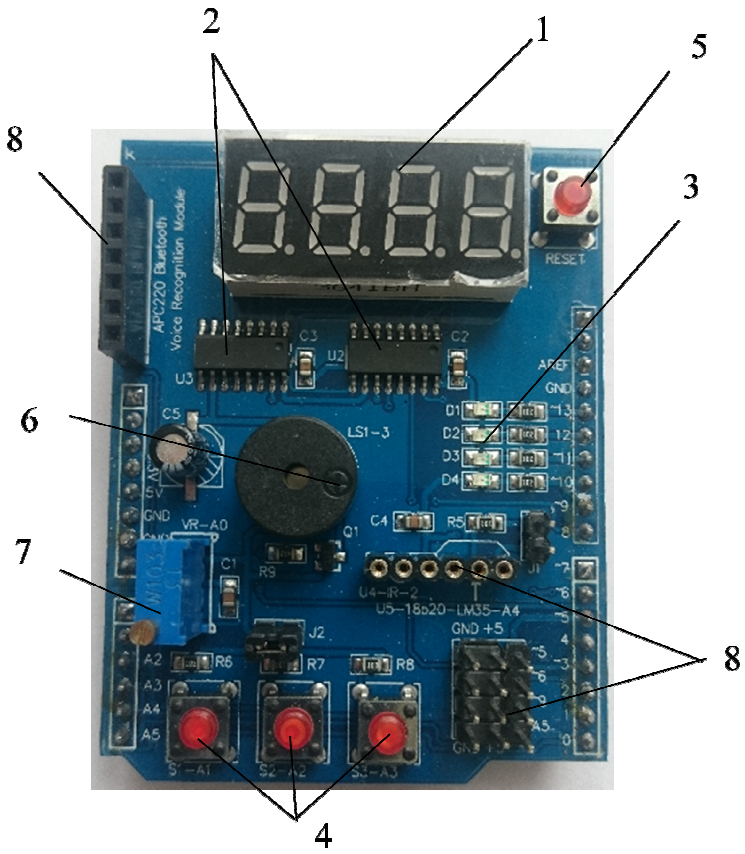

Рисунок 2 – Плата расширения Multi-Function shield

На плате установлены:

1) Четырёхразрядный семисегментный индикатор;
2) Сдвиговые регистры;
3) Индикаторные светодиоды (4 шт.);
4) Кнопки управления (3 шт.);
5) Кнопка сброса;
6) Звуковой излучатель;
7) Потенциометр (10 кОм);
8) Разъемы для подключения датчиков и модулей расширения.

Разработка программного обеспечения выполняется в среде Arduino IDE [4], которая включает в себя текстовый редактор с необходимым набором инструментов для программирования.

Графический интерфейс пользователя содержит следующие основные элементы управления (рисунок 3):

1) Verify (проверить) – компилирует (собирает) проект и проверяет на отсутствие ошибок в коде программы.

2) Upload (загрузить) – загрузить программу в микроконтроллер на плате Arduino;

3) New (новый) – создать новую программу, скетч;

4) Open (открыть) – открывает меню со списком проектов, скетчей;

5) Save (сохранить) – сохраняет активный, текущий проект;

6) Serial Monitor (монитор последовательного порта) – отображает работу COM-порта с текущим скетчем;

7) Sketch Name (имя скетча) – отображает имя текущего проекта (скетча);

8) Code Area (область кода) – область кода программы (скетча);

9) Message Area (область сообщений) – область сообщений и вывода ошибок.

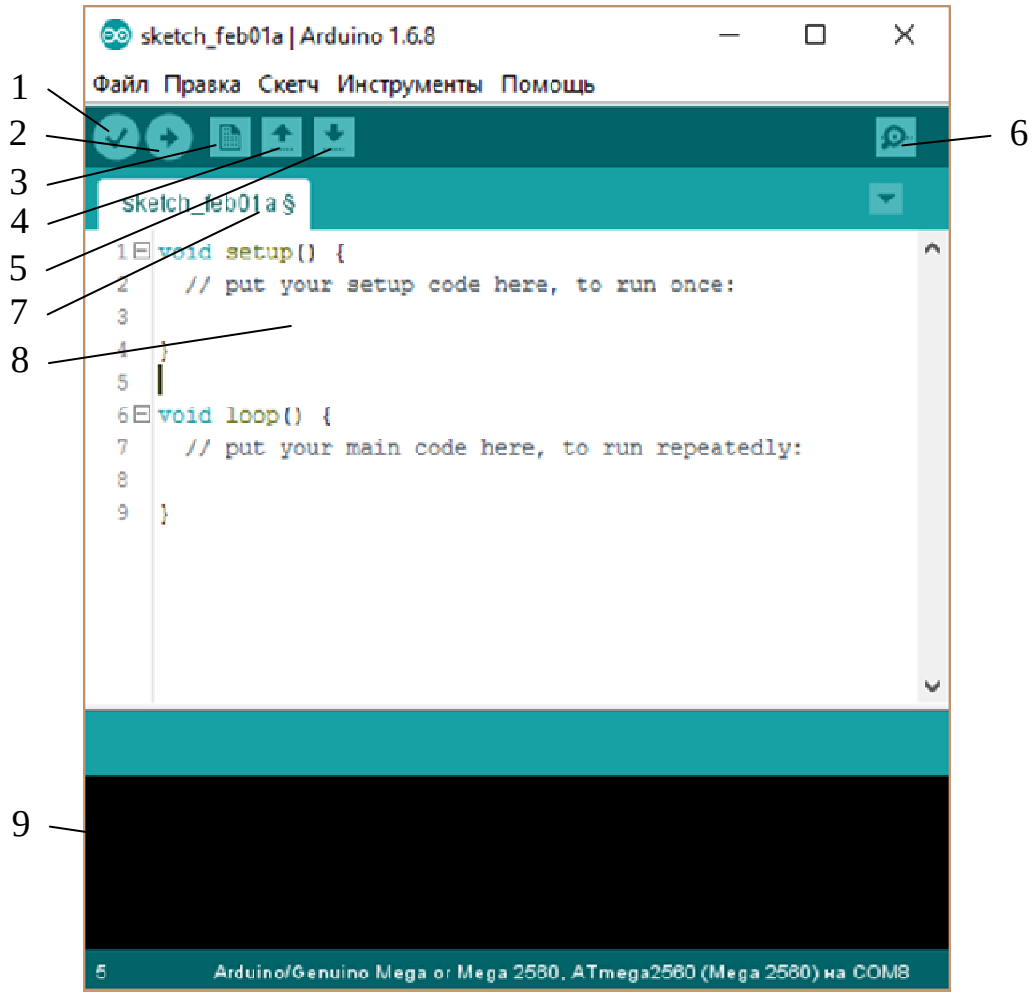

Рисунок 3 – Пользовательский интерфейс Arduino IDE

Проекты в Arduino IDE называются скетчами и реализуются на незначительно измененном языке программирования C/C++. Хотя фактически программы в данной среде пишутся на C/C++ и компилируются и собираются с помощью широко известного компилятора avr-gcc [5]. Особенности среды сводятся к тому, что имеется набор библиотек, включающий в себя различные функции для управления периферией микроконтроллера, а при компиляции программы среда создает временный .cpp файл, в который помимо написанного разработчиком кода, вставляется еще несколько дополнительных инструкций, и полученный результат

передается компилятору, а затем линковщику. Например, пустой проект, созданный в Arduino IDE выглядит следующим образом:

```
void setup()
{
    //здесь размещается код пользовательской инициализации
}
void loop()
{
    //тело основного рабочего цикла
}
```

Базовая заготовка состоит из двух функций:

1) функция *setup* – в ней размещается код для первичной инициализации как пользовательского программного обеспечения, так и периферии микроконтроллера. То есть те инструкции, которые нужно выполнить в программе только один раз при запуске.

2) функция *loop* – здесь размещается основной код разрабатываемого программного обеспечения. Нужно подчеркнуть такую особенность: так как микроконтроллер должен работать условно бесконечно, то есть пока не будет отключено питание, то, как правило, основной код должен быть зациклен и периодически повторяться, например, для опроса состояния датчиков и т.п. Поэтому, по завершению функции *loop* будет выполнен её повторный вызов и так до бесконечности.

Преобразованный средой Arduino IDE перед компиляцией код уже выглядит следующим образом:

```
#include "WProgram.h"      // тут определения всех Arduino ф-ий, констант и т.д.
void setup();              // объявляют ф-ии setup() и loop(), в которых наша
void loop();               // программа для Arduino и пишется
void setup()
{
 }
void loop()
{
}
int main(void)             // здесь как и принято в c/c++ ф-ия main()
{
    init();                // в ней вызывается своя инициализация
    setup();               // затем вызывается наш setup()
    for (;;)               // и в бесконечном цикле вызывается наш loop()
```

loop();

return 0; *// а сюда вообще никогда не попадаем, но для выполнения формальных требований нужно написать*

}

Процесс подключения отладочной платы к компьютеру содержит следующие пункты:

1) Подключите отладочную плату к персональному компьютеру при помощи кабеля USB-A <-> USB-B;

2) Выберите в программном обеспечении тип используемой отладочной платы (рисунок 4);

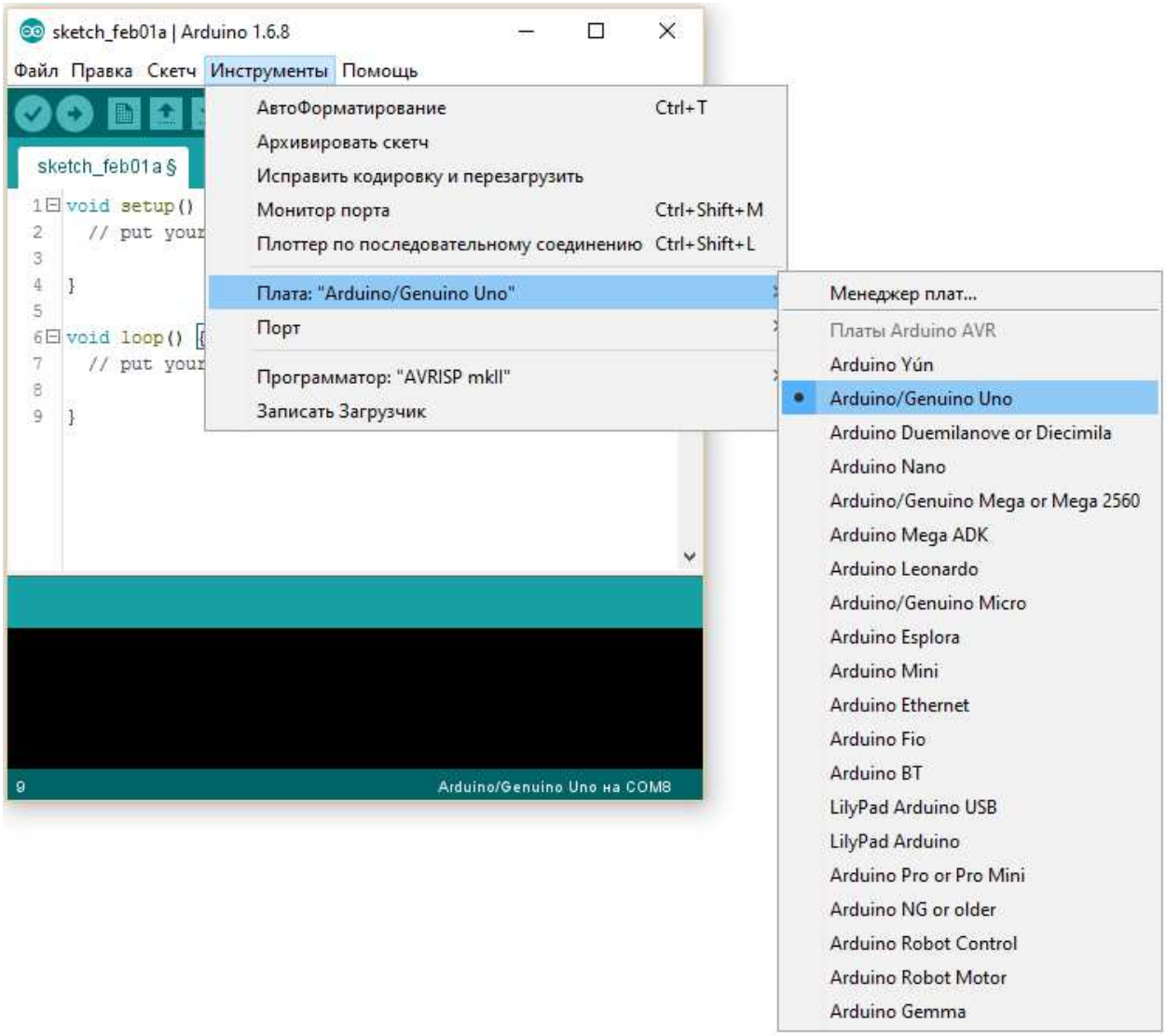

Рисунок 4 – Выбор типа отладочной платы

3) Выберите виртуальный COM-порт, к которому подключена отладочная плата (рисунок 5);

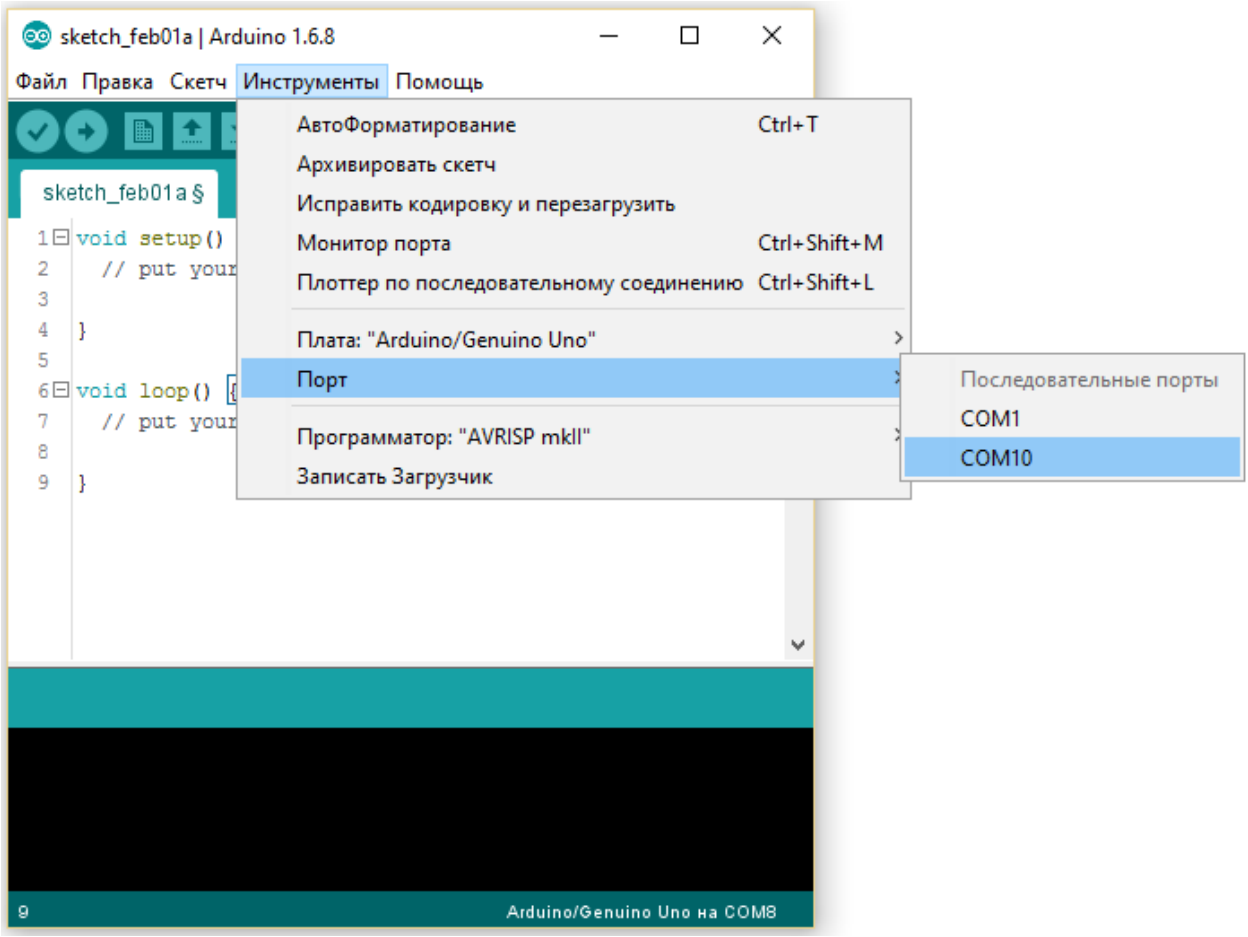

Рисунок 5 – Выбор виртуального COM-порта

Для компиляции программного кода достаточно нажать на кнопку *Verify (проверить)* (см. рисунок 3), а загрузка программы в микроконтроллер осуществляется посредство нажатия на кнопку *Upload (загрузить)*. Допускается сразу нажимать кнопку *Upload (загрузить)* для компиляции и загрузки.

Как было отмечено ранее, на отладочной плате Arduino UNO установлен микроконтроллер ATmega328 [6]. Он основан на восьмибитном процессорном ядре под названием AVR и выполнен с использованием RISC архитектуры [7]. Работает данный микроконтроллер с тактовой частотой 16 МГц. На борту установлены следующие типы памяти:

1) Flash-память 32 КБ – это постоянное запоминающее устройство (ПЗУ) для хранения программы;
2) EEPROM-память 1 КБ – ПЗУ для хранения данных программы;
3) SRAM-память 2 КБ – оперативное запоминающее устройство (ОЗУ) для хранения временных данных во время выполнения программы.

В микроконтроллере размещены различные периферийные блоки:

1) порты ввода-вывода;
2) два восьмибитных таймера;
3) один шестнадцатибитный таймер;

4) часы реального времени;
5) шесть каналов с выходом широтно-импульсной модуляции;
6) восьмиканальный аналого-цифровой преобразователь (АЦП);
7) последовательные интерфейсы USART, SPI, TWI.

С внешним миром микроконтроллер взаимодействует через порты ввода-вывода [6, п. 14]. Структурная схема одного порта изображена на рисунке 6.

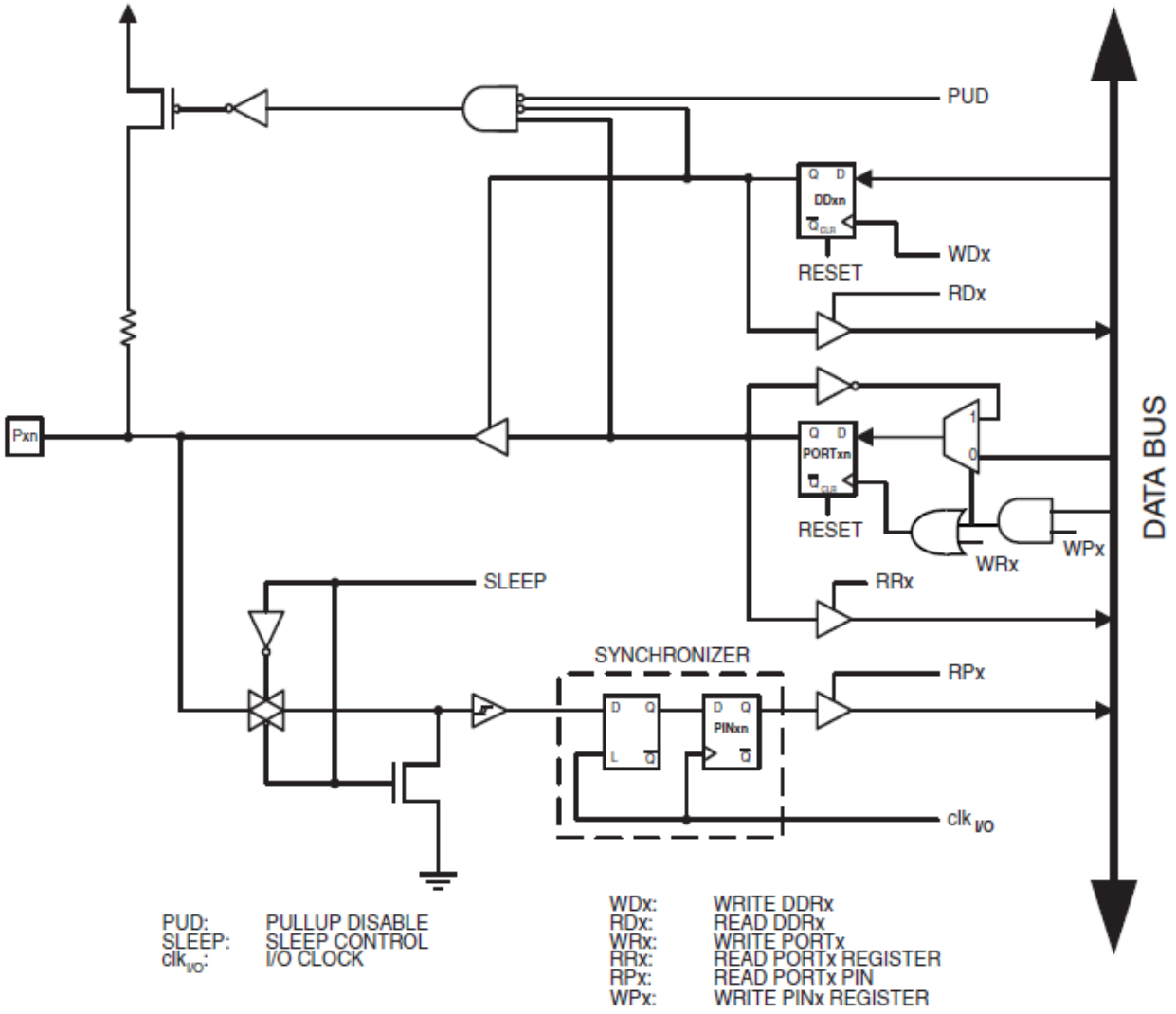

Рисунок 6 – Структурная схема одного вывода порта ввода–вывода микроконтроллера

Согласно рисунку 6 выходной контакт изображен в квадрате с подписью Pxn, где x – буква порта, а n – номер вывода в порте. За управление портами отвечают три регистра (регистр процессора, биты (флаги) которого отражают состояние процессора и свойства результатов предыдущих операций): PORTx, PINx, DDRx. К примеру, за порт «B» отвечают три восьмиразрядных регистра PORTB, PINB, DDRB, а каждый бит в этих регистрах отвечает за соответствующий вывод порта. Это как линейка из восьми переключателей.

DDRx – это регистр направления порта. Порт в конкретный момент времени может быть либо входом, либо выходом. Если определенный бит в регистре DDRx установить в 1, то этот вывод микроконтроллера будет переключен на выход.

PORTx – регистр управления состоянием вывода. Когда вывод настроен на выход, то значение соответствующего бита в регистре PORTx определяет состояние вывода. Если определенный бит в порте установить в 1, то на выводе микроконтроллера будет установлена логическая единица. С нулём обратный эффект.

PINx – регистр чтения будет рассмотрен в следующей части работы

Использоваться описанные регистры на языке Си могут следующим образом:

1) например, настройка четвертого вывода порта А на выход *DDRA|=(1<<4)*, где «|» – операция побитового ИЛИ; «<<» – операция побитового сдвига;

2) переключение вывода на вход выполняется следующим образом *DDRA&=~(1<<4)*, где «&» – операция побитового И; ~ – операция побитовой инверсии; «<<» – операция побитового сдвига;

3) если вывод настроен на выход, то переключение его в логическую единицу осуществляется путём выполнения операции *PORTA|=(1<<4)*;

4) переключение в логический ноль выполняется следующим образом *PORTA&=~(1<<4)*;

Стоит отметить, что PORTx, PINx, DDRx в программной среде Arduino IDE являются константами и содержат адреса соответствующих регистров.

В среде программирования Arduino IDE существует специализированные функции для работы с портами ввода-вывода. Эта библиотека выполнена таким образом, чтобы обращение к определенным портам ввода-вывода было не через порты микроконтроллера, а через обозначения на отладочной плате (см. рисунки 1–2).

int pinMode(int pin, int mode) – устанавливает режим работы вывода [8].

pin – номер вывода или его обозначение (см. рисунок 2: аналоговые входы и кнопки – A0–A5, светодиоды – D1–D4.), которые в данный момент настраивается;

mode – режим одно из двух значение – константа *INPUT* или *OUTPUT*, устанавливает на вход или выход, соответственно.

int digitalWrite(int pin, int value) – устанавливает выбранный вывод в состояние логической единицы или нуля;

value – значение может быть константой *HIGH* или *LOW*.

Использоваться функции могут следующим образом:

1) установка первого вывода на выход *pinMode(1, OUTPUT)*;

2) установка первого вывода на вход *pinMode(1, INPUT)*;

3) установка логической единицы на первом выводе *digitalWrite(1, HIGH)*;

4) установка логического нуля на первом выводе *digitalWrite(1, LOW)*.

В данной и последующих работах может потребоваться функция задержки выполнения операций.

void delay(int ms) – приостанавливает выполнение программы на заданное в параметре количество миллисекунд.

Пример использования:

```
digitalWrite(10, HIGH);   // переключает выход 10 в логическую единицу
delay(1000);              // ожидание секунду
digitalWrite(ledPin, LOW);    // переключает выход 10 в логический ноль
```

3 Порядок выполнения работы

В практической части работы должны быть решены две задачи: освоение работы с портами ввода-вывода, настроенными на режим выхода; закрепление полученных навыков при решении задачи переключения светодиодов в режиме «змейка». Выполнение работы должно быть выполнено в двух вариантах: управление режимами и состояниями выводов путём непосредственного обращения к регистрам микроконтроллера; управление с помощью функций Arduino IDE. Для этого необходимо выполнить следующие действия:

3.1 Изучите предложенный в п. 2 теоретический материал.

Подключение отладочной платы и настройка среды:

3.2 Вставьте плату расширения Multi-Function shield в разъемы Arduino UNO.

3.3 Подключите отладочную плату к компьютеру как описано в теоретической части.

3.4 Создайте новый проект (меню Файл->Новый) и сохраните (меню Файл->Сохранить как…) его в свой рабочий каталог.

Освоение работы с портами ввода-вывода, настроенными на режим выхода:

3.5 Для первой части работы понадобится один светодиод. Выберете светодиод, которые вы собираетесь зажечь, и определите обозначение вывода порта, к которому он подключён.

3.6 С использованием [9] определите букву порта микроконтроллера и номер его вывода, к которому подключен светодиод.

3.7 В функции *setup()* переключите на выход выбранный вывод с подключенным светодиодом, путём изменения состояния необходимого бита конфигурационном регистре.

3.8 В функции *loop()* переключите в «0» вывод для светодиода путём изменения состояния необходимого бита в регистре состояния.

3.9 Установите паузу в 0,5 секунды.

3.10 В функции *loop()* переключите в «1» вывод для светодиода путём изменения состояния необходимого бита в регистре состояния.

3.11 Установите паузу в 0,5 секунды.

3.12 Выполните компиляцию написанного кода нажатием на кнопку *Verify (проверить)* (см. рисунок 3).

3.13 Загрузите программу в микроконтроллер нажатием на кнопку *Upload (загрузить)*.

В результате на плате должен начать моргать выбранный светодиод с частотой 1 Гц.

3.14 Повторите проделанные действия для других светодиодов.

3.15 Создайте новый проект и выполните пункты 3.7 –3.14 , используя функции Arduino IDE.

3.16 Используя оператор *for*, функцию задержки и работу с регистрами, организуйте переключение светодиодов по типу «змейка»: 1) вкл., выкл., выкл., выкл.; выкл., вкл., выкл., выкл.; выкл., выкл., вкл., выкл.; выкл., выкл., выкл., вкл.

3.17 Создайте новый проект и повторите программу «змейка», используя функции Arduino IDE.

3.18 Оформите отчет, содержащий титульный лист и разделы: введение, ход выполнения работы, ответы на контрольные вопросы и выводы.

3.19 Защитите отчет у преподавателя.

4 Контрольные вопросы

4.1 Что такое регистр?

4.2 Какого назначение Multi-Function shield?

4.3 Перечислите основные регистры процессора для работы с портами ввода-вывода.

4.4 Опишите основные подходы для работы с регистрами портов ввода-вывода.

4.5 Опишите основные функции Arduino IDE для работы с портами ввода-вывода.

4.6 Опишите настройку третьего вывода порта А на выход с использованием регистров.

4.7 Опишите настройку третьего вывода порта А на выход с использованием функций Arduino IDE.

4.8 Перечислите основные блоки Arduino UNO.

4.9 Перечислите основные блоки Multi-Function shield.

Список литературы

1. Аппаратная часть платформы Arduino.– URL: http://arduino.ru/Hardware (дата обращения: 20.01.2017).

2. Arduino UNO.– URL: https:\\www.rapidonline.com\pdf\73-4443.pdf (дата обращения: 10.01.2017).

3. Using an Arduino Multi-function Shield.– URL: https:\\www.mpja.com\download\hackatronics-arduino-multi-function-shield.pdf (дата обращения: 10.01.2017).

4. Среда разработки Arduino.– URL: http://arduino.ru/Arduino_environment (дата обращения: 10.01.2017).

5. WinAVR.– URL: http://winavr.sourceforge.net (дата обращения: 10.01.2017).

6. ATMEL 8-BIT MICROCONTROLLER WITH 4/8/16/32KBYTES, IN-SYSTEM PROGRAMMABLE FLASH.– URL: http:\\www.atmel.com\images\Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf (дата обращения: 10.01.2017).

7. Сергеев С. Архитектуры вычислительные систем.– СПб:БХВ-Петербург.– 2010.– 240 с.

8. Справочник языка Ардуино.– URL: http://arduino.ru/Reference (дата обращения 10.01.2017).

9. UNO Schematic – Arduino.– URL: https:\\www.arduino.cc\en\uploads\Main\arduino-uno-schematic.pdf (дата обращения: 10.01.2017).