|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |



ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.03 Прикладная информатика**

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К КУРСОВОЙ РАБОТЕ***

***По дисциплине «Микропроцессорные системы»***

***НА ТЕМУ:***

***МК-система управления приборами жилого помещения***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-75Б |  |  | В.Д. Шульман |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Руководитель |  |  |  | В.Я. Хартов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Оглавление

[Реферат 4](#_Toc28564527)

[Обозначения и сокращения 5](#_Toc28564528)

[Введение 6](#_Toc28564529)

[Основная часть 7](#_Toc28564530)

[1 Конструкторская часть 8](#_Toc28564531)

[1.1 Проектирование МК-системы 8](#_Toc28564532)

[1.1.1 Описание структурно-функциональной схемы микроконтроллерной системы 8](#_Toc28564533)

[1.1.2 Выбор микроконтроллера 10](#_Toc28564534)

[1.1.3 Описание архитектуры и технических характеристик микроконтроллера 11](#_Toc28564535)

[1.1.4 Распределение адресного пространства ATmega8515 13](#_Toc28564536)

[1.1.5 Особенности системы команд микроконтроллера ATmega8515 14](#_Toc28564537)

[1.2 Разработка функциональной схемы 15](#_Toc28564538)

[1.2.1 Используемые модули ATmega8515 15](#_Toc28564539)

[1.2.2 Пульт оператора 15](#_Toc28564540)

[1.2.3 Блок передачи данных к ПЭВМ 18](#_Toc28564541)

[1.2.4 Блок реле 19](#_Toc28564542)

[1.2.5 LCD-дисплей 20](#_Toc28564543)

[1.3 Разработка принципиальной схемы 23](#_Toc28564544)

[1.3.1 Синтез принципиальной схемы 23](#_Toc28564545)

[1.3.2 Конфигурация выводов микроконтроллера 23](#_Toc28564546)

[1.3.3 Схема понижения входного напряжения до 5В 24](#_Toc28564547)

[1.3.4 Подключение матричной клавиатуры 25](#_Toc28564548)

[1.3.5 Подключение LCD-дисплея 27](#_Toc28564549)

[1.4 Расчёт параметров 28](#_Toc28564550)

[1.4.1 Расчет параметров настройки таймеров 28](#_Toc28564551)

[1.4.2 Расчёт сопротивления резисторов для катушек реле 28](#_Toc28564552)

[1.4.3 Расчёт потребляемой мощности 29](#_Toc28564553)

[1.5 Описание алгоритмов функционирования устройства 30](#_Toc28564554)

[1.5.1 Схемы-алгоритмы работы программы МК 30](#_Toc28564555)

[1.5.1 Формат передаваемых данных от сервера расписаний 38](#_Toc28564556)

[2 Технологическая часть 39](#_Toc28564557)

[2.1 Характеристика использованных систем разработки 39](#_Toc28564558)

[2.2 Оценка количества задействованной памяти микроконтроллера ATmega8515 39](#_Toc28564559)

[2.3 Симуляция в Proteus 8 40](#_Toc28564560)

[2.4 Способы программирования памяти микроконтроллера ATmega8515 42](#_Toc28564561)

[2.4.2 Алгоритм последовательного программирования через SPI 43](#_Toc28564562)

[2.4.3 Опрос данных Flash памяти 46](#_Toc28564563)

[2.4.4 Опрос данных EEPROM 47](#_Toc28564564)

[Заключение 49](#_Toc28564565)

[Список использованных источник 50](#_Toc28564566)

[Приложение А – Исходные коды программ 51](#_Toc28564567)

[Приложение Б – Спецификация радиоэлементов схемы 76](#_Toc28564568)

Реферат

Расчётно-пояснительная записка с. 50, рис. 29, табл. 6, источников 4, приложений 2.

МИКРОПРОЦЕССОР, МИКРОКОНТРОЛЛЕР, ATMEGA8515, СЕРВЕР РАСПИСАНИЙ, ПУЛЬТ ОПЕРАТОРА, UART, ТАЙМЕР, ДИСПЛЕЙ.

Объектом разработки курсовой работы является устройство управления приборами, получающее расписание их работы с удаленного сервера.

Цель работы – создание полного комплекса конструкторской документации для устройства управления приборами, создание программного обеспечения для микроконтроллера семейства AVR/

При проектировании решены следующие задачи:

* анализ объекта разработки на функциональном уровне;
* разработка функциональной схемы;
* выбор элементной базы для реализации объекта;
* разработка принципиальной схемы;
* расчет потребляемой мощности;
* разработка алгоритмов работы микроконтроллера;
* написания программного обеспечения для микроконтроллера.

Результатом проектирования является комплекс конструкторской документации для изготовления устройства, исходные коды программ для программирования памяти микроконтроллера.

Спроектированное устройство обладает следующими характеристиками:

1. управление до 8 приборами одновременно;
2. хранение до 127 записей для включения/выключения устройств;
3. получение актуального расписания и времени по UART;
4. управление устройствами вручную через пульт оператора;
5. установка текущего времени и расписания по умолчанию;
6. вывод текущего времени МК-системы на LCD-дисплей

Обозначения и сокращения

МК – микропроцессор

ПЭВМ – персональная электронно-вычислительная машина

СР – сервер расписания

РОН – регистры общего назначения

АЛУ – арифметико-логическое устройство

ПЗУ – постоянно запоминающее устройство

EEPROM – (Electrically Erasable Programmable Memory) электрически стираемое программируемое ПЗУ

Flash – перепрограммируемая память для хранения программ

PC – (Program Counter) программный счетчик

SREG – (Status Register) регистр статуса

MCUCR – (MCU Control Register) регистр управления

TIMSK – (Timer/Counter Interrupt Mask Register) регистр масок прерывания по таймерам/счетчикам

ISP – (In System Programming) внутрисхемное программирования

SPI – (Serial Peripheral Interface) последовательный периферийный интерфейс

UART – (Universal asynchronous receiver/transmitter) универсальный асинхронный приёмопередатчик

Введение

В данной работе на основании учебного плана кафедры ИУ6 производится разработка устройства управления для приборов жилого помещения, которое осуществляет включение и выключение устройств по расписанию, получаемом от удаленного сервера по протоколу асинхронной передачи UART.

Для выполнения поставленной задачи используется высокопроизводительный 8-разрядный контроллер AVR ATmega8515. Внутренняя оперативная память SRAM данного микроконтроллера позволяет хранить до 512 байт данных, чего вполне достаточно для хранения расписания для многократного включения и отключения 8 приборов жилого помещения в течение суток. Модуль USART, 1 8-разрядный и 1 16-разрядный таймеры позволяют обеспечить необходимый функционал устройству для оперативного получения расписания и включения приборов в необходимые временные отрезки.

Для нештатных ситуаций, которые могут возникать в процессе работы устройства и сервера расписаний, в устройстве присутствует пуль управления оператора [ПУО]. ПУО позволяет манипулировать устройством управления напрямую.

Основная часть

В данной курсовой работе было разработано устройство управления 8 приборами жилого помещения на основе 8-разрядного высокопроизводительного микроконтроллера AVR ATmega8515

В техническом задании не предъявлялись специальные требования к выбору микроконтроллера и периферийных микросхем для создаваемого устройства управления. Был выбран контроллер ATmega8515, ввиду его функциональности и высокой частоты работы процессора.

Для осуществления получения расписания с удаленного сервера было принято решение использовать протокол передачи данных RS-232 и модуль микроконтроллера USART. При этом было принято решение использовать асинхронный способ передачи данных по UART ввиду простоты и большей эффективности такого метода по сравнению с синхронной передачей.

Для хранения расписания было принято решение использовать 4-х байтовые сообщение в оперативной памяти SRAM в качестве меток включения или выключения устройств. В эти 4 байта входят номер устройства, время (часы, минуты, секунды), когда необходимо выключить/включить устройство, и флаг, характеризующий выключение или включение устройства.

Для ручного управления устройством был принято решение предусмотреть пульт оператора, с помощью которого можно включать и выключать устройства без расписания. В качестве ПУО используется матричная клавиатура 4x4 с 16 клавишами, 8 из которых – различные команды, ещё 8 – клавиши выбора прибора.

# Конструкторская часть

## Проектирование МК-системы

### Описание структурно-функциональной схемы микроконтроллерной системы

Согласно заданию, нужно разработать устройство управления 8 приборами жилого помещения согласно расписанию, получаемому с удаленного сервера.

Из этого следует, что необходимо использовать модуль для приема и передачи данных USART микроконтроллера.

Для отсчета времени следует использовать имеющиеся таймеры T0 8-разрядный и T1 16-разрядный.

Для обеспечения повышенной точности при работе разрабатываемой системы будет подключен внешний кварцевый генератор с частотой 8 MHz к разъемам XTAL1 и XTAL2.

Для дополнительного контроля над устройством будет возможность управлять им напрямую с помощью пульта оператора, состоящего из 16 кнопок.

Для подключения к ПЭВМ, которая и является сервером расписания, будет использоваться драйвер MAX232.

Для возможности оперативного получения расписания необходимо предусмотреть возможность отправки запроса на сервер расписания для получения последних данных о текущем расписании и времени.

Для отображения текущего времени в составе разрабатываемой МК-системы будет использоваться LCD-дисплей LM016L 16x2, где каждую секунду будет обновляться текущее время микроконтроллера.

Т. к. предполагается, что приборы будут питаться от стандартного напряжения в жилом помещении (220 В), то для включения и отключения питания приборов будет использоваться блок реле.

Итоговое устройство должно выводить на порт управления приборами текущее состояние каждого из приборов, где каждому прибору соответствует один бит, согласно принятому от ПВМ и записанному в оперативную память расписанию работы приборов.

Исходя из вышеперечисленного, итоговое устройство должно состоять из следующих блоков:

микроконтроллер;

блок обмена информацией с ПЭВМ;

пуль оператора;

блок реле для управления питанием приборов

LCD-дисплей

Обобщенная структура проектируемого устройства представлена на рисунке 1.

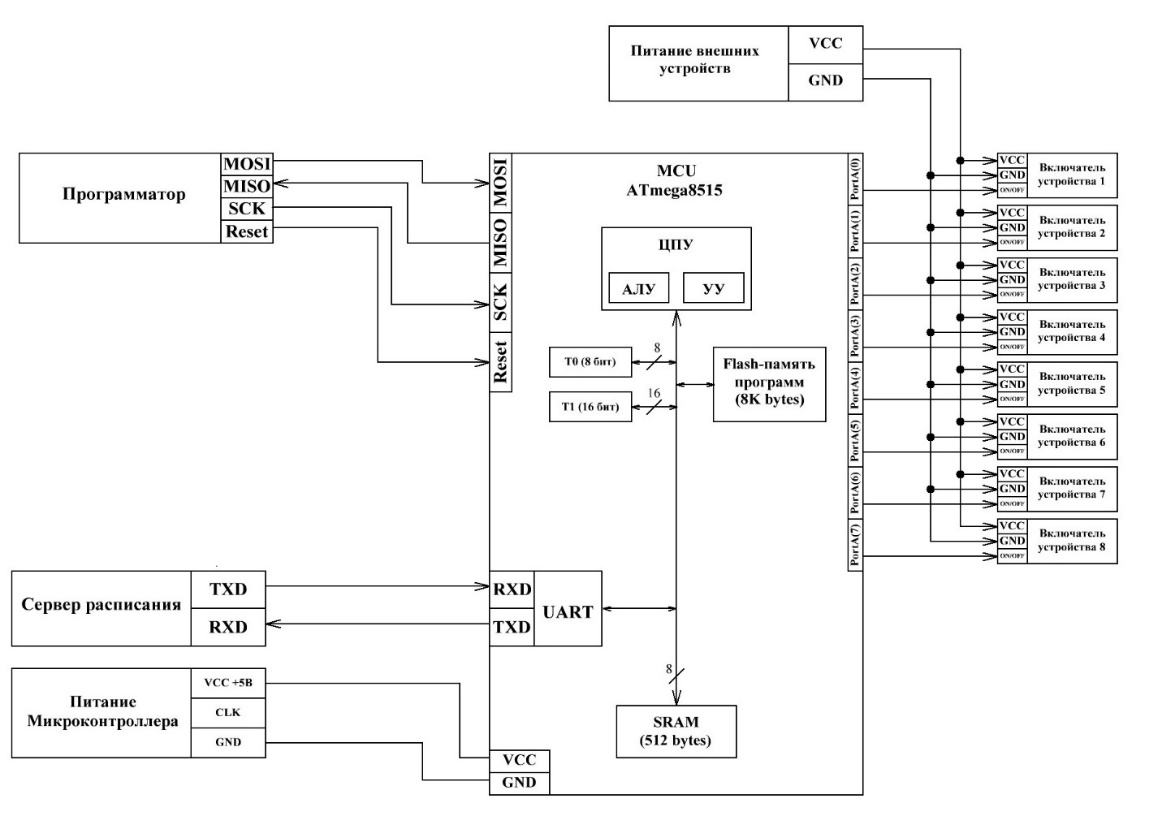


Рисунок – Структурная схема устройства управления приборами жилого помещения

### Выбор микроконтроллера

При выборе микроконтроллера важными параметрами были следующие:

* наличие модуля для асинхронной передачи данных UART;
* частота работы;
* объем оперативной памяти;
* количество выводов
* объем памяти программ
* количество таймеров и их разрядность

В таблице представлено сравнение некоторых микроконтроллеров AVR по важными для данной разрабатываемой системы параметрам.

Таблица – Сводная таблица параметров различных МК AVR

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| МК | Пины | ПЗУ КБ | SRAM Б | Таймеры | Максимальная частота | Наличие модуля USART |
| ATmega8A | 28 | 8 | 1024 | 2x8 бит  1x16 бит | 16 | Да |
| AT90LS2323 | 8 | 2 | 128 | 1x8 бит | 4 | Нет |
| AT90S4433 | 28 | 4 | 128 | 6x10 бит | 8 | Да |
| AT90S2343 | 8 | 2 | 128 | 1x8 бит | 10 | Нет |
| ATmega8515 | 40 | 8 | 512 | 1x8 бит  1x16 бит | 16 | Да |
| ATtiny2313 | 20 | 2 | 128 | 1x8 бит  1x16 бит | 20 | Да |

Исходя из сводной таблицы видно сразу, что для поставленных целей подходят не все из представленных микроконтроллеров.

Однозначно не подходят микроконтроллеры, у которых отсутствует USART, без которого осуществление асинхронного обмена данным с удаленным сервером не представляется возможным.

Микроконтроллер ATtiny2313 подходит по большинству параметров, однако он обладает достаточно небольшим объемом SRAM и малым количеством пинов, что приведет к наложению значительных ограничений при реализации МК-системы с использованием этого микроконтроллера.

Наиболее подходящие кандидаты – это ATmega8A и ATmega8515. В данном случае предпочтение отдается ATmega8515, т.к. он обладает большим количеством выводом. ATmega8A обладает большим объемом SRAM и таймеров, но в контексте поставленной задачи такое количество памяти и счётчиком является избыточным и не будет использоваться в полном объеме.

### Описание архитектуры и технических характеристик микроконтроллера

В проектируемом устройстве используется 8-битный микроконтроллер AVR ATmega8515. Его функциональная схема представлена на рисунке 2.

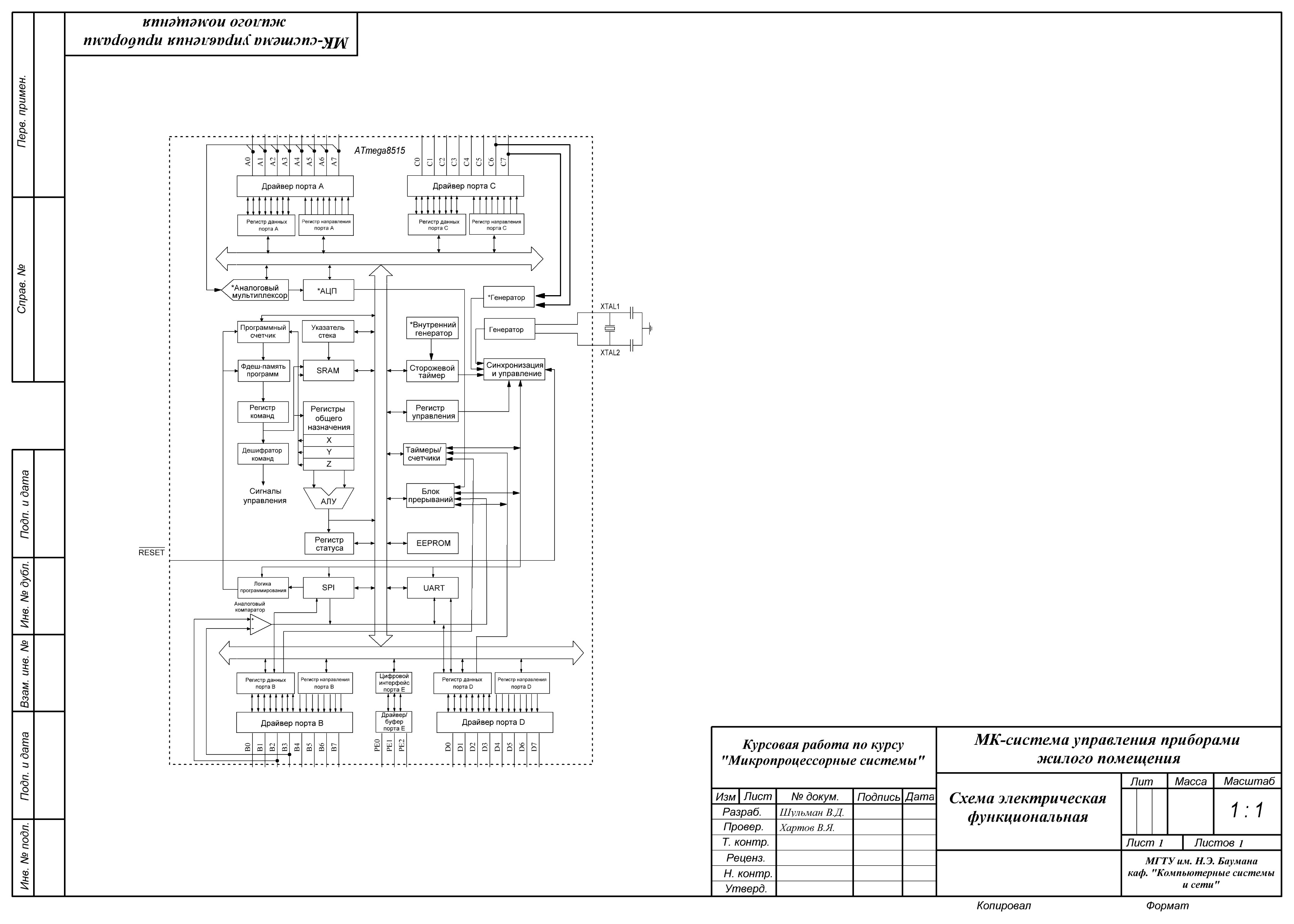


Рисунок – Функциональная схема микроконтроллера ATmega8515

Из функциональной схемы видно, что микроконтроллер обладает четырьмя 8-разрядными портами ввода-вывода, один из которых имеет АЦП с мультиплексором; дополнительный 3-разрядный порт PE; аппаратными интерфейсами USART, SPI; встроенным компаратором, встроенным генератором (осциллятором); счетчиками (один 8-разрядный и один 16-разрядный); сторожевым таймером; блоком прерываний; энергонезависимой и энергозависимой памятью.

Семейство микроконтроллеров Mega – это 8-битные микроконтроллеры, представляющие собой одну из лучших основ для создания экономных и высокопроизводительных устройств различного назначения.

Микроконтроллеры этого семейства изготавливаются по RISC-архитектуре, согласно которой инструкции, выполняемые процессором микроконтроллера, должны быть как можно более простыми. Такой подход позволяет получить оптимальное соотношение между стоимостью, быстродействием и энергопотреблением.

### Распределение адресного пространства ATmega8515

В микроконтроллерах AVR используется Гарвардская архитектура. Согласно этой архитектуре память программ и память данных находится в разных адресных пространствах. Способу адресации и доступа к этим областям также различны. Такая архитектура обеспечивает центральному процессору одновременную работу с памятью программ и с памятью данных. Это существенно повышает производительность МК.

Память данных МК разделена на три части:

* регистровая память;
* оперативная память;
* энергонезависимая память.

Регистровая и оперативная память находится в одном адресном пространстве, в отличии от энергонезависимой, которая обладает собственным адресным пространством.

Изображение адресных пространств МК ATmega8515 представлено на рисунке 3.

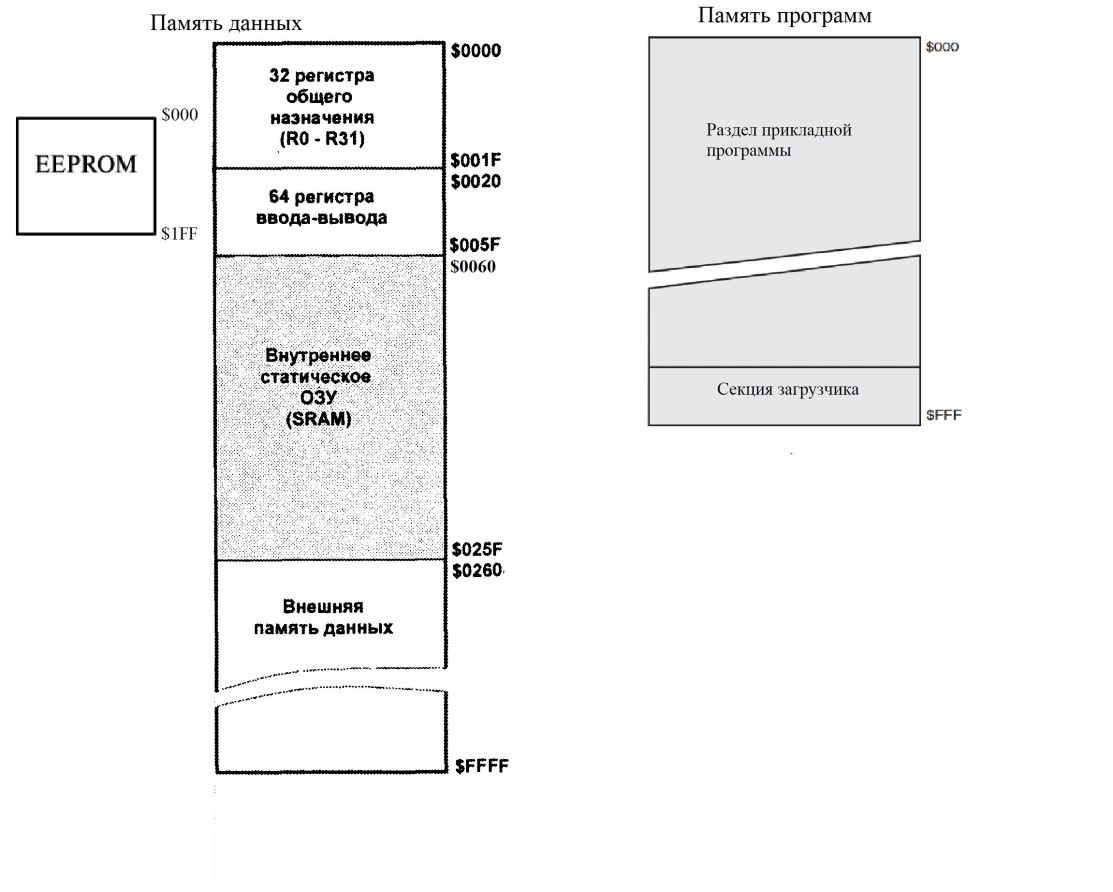


Рисунок – Адресные пространства МК ATmega8515

Регистровая память включает 32 регистра общего назначения и 64 регистра ввода-вывода.

Для хранения данных имеется память RAM объемом 512 байт. Есть возможность подключения внешней памяти SRAM, позволяющее расширить оперативную память микроконтроллера до 64 Кбайт.

Для долгосрочного хранения данных в МК присутствует 512 байт памяти EEPROM.

### Особенности системы команд микроконтроллера ATmega8515

Система команд микроконтроллера ATmega8515 выполнена по RISC архитектуре и состоит из 130 инструкций, большинство из которых выполняются за один такт.

Система команд обладает полностью статическим функционированием. Производительность составляет до 16 млн. операций в секунду при тактовой частоте 16 МГц.

## Разработка функциональной схемы

### Используемые модули ATmega8515

Микроконтроллер ATmega8515 является основным узлом в данной разрабатываемой системе.

В рразработке МК-системы используются следующие компоненты и модули микроконтроллера;

* модуль USART в асинхронном режиме;
* модуль SPI для программирования микроконтроллера;
* таймер T0 для отсчета времени
* система прерываний
* 8 выводов порта А для управления приборами
* 8 выводов порта C для функционирования ПУО
* 3 вывода порта B для программирования
* 2 вывода порта D для передачи данных по USART

Для эффективной и быстрой передачи данных, а также для оперативного включения и отключения приборов используется тактовая частота в 8 МГц.

### Пульт оператора

Пульт оператора представляет собой матричную клавиатуру, которая состоит из 16 кнопок.

Он позволяет оказывать воздействие на работу устройства в случае непредвиденных ситуаций (отказ работы сервера расписания, необходимость срочного включения или отключения одного из приборов и т. п.).

Все 16 клавиш пульта оператора задействованы и несут в себе определенную функцию. Функционирование части клавиш зависит от клавиш, которые были нажаты на пульте оператора в предыдущий момент времени. Функциональное назначение кнопок пульта оператора представлено на рисунке 4.

|  |  |  |  |
| --- | --- | --- | --- |
| 7 | 8 | ВСЕ | ВКЛП |
| 4 | 5 | 6 | ВЫКЛП |
| 1 | 2 | 3 | ЗСР |
| СВПР  /ОТМ | УРПУ | УВВ | ЗСВ |

Рисунок – Пульт управления оператора

Клавиши на пульте оператора можно условно разделить на 2 категории: функциональные клавиши, выполняющие какую-то операцию (ВКЛП, ЗСР, СВПР и т.д.), и контекстные клавиши, предназначенные для выбора прибора, над которым необходимо совершить, выбранное с помощью функциональных клавиш, действие (1, 2, …, 7, ВСЕ).

Подробное описание функции, которую несет каждая из клавиш, представлено в таблице 2.

Таблица – Назначения и расшифровка клавиш пульта оператора

|  |  |  |
| --- | --- | --- |
| Название клавиши | Группа | Расшифровка и назначение |
| ВКЛП | Функциональная | Включить принудительно. Принудительно включает одно из выбранных устройств и переводит его в принудительный режим. |
| ВЫКЛП | Функциональная | Выключить принудительно. Принудительно выключает одно из выбранных устройств и переводит его в принудительный режим |
| ЗСР | Функциональная | Запросить серверное расписание. Запрашивает у сервера новое расписание. |
| ЗСВ | Функциональная | Запросить серверное время. Запрашивает у сервера текущее время. |
| УРПУ | Функциональная | Установить расписание по умолчанию. Устанавливает расписание по умолчанию. |
| УВВ | Функциональная | Установить время вручную. Позволяет оператору вручную установить текущее временя в МК-системе. |
| СВПР/ОТМ | Функциональная | Сброс всех принудительных режимов/Отмена. Выводит все устройства из принудительного режима.  Может использоваться как клавиша отмены операции на этапе выбора прибора. |
| 1 | Контекстная | Выбирает первый прибор |
| 2 | Контекстная | Выбирает второй прибор |
| 3 | Контекстная | Выбирает третий прибор |
| 4 | Контекстная | Выбирает четвертый прибор |
| 5 | Контекстная | Выбирает пятый прибор |
| 6 | Контекстная | Выбирает шестой прибор |
| 7 | Контекстная | Выбирает седьмой прибор |
| 8 | Контекстная | Выбирает восьмой прибор |
| ВСЕ | Контекстная | Выбирает все приборы |

Один из возможных сценариев использования пульта оператора:

возникла необходимость получения более актуального расписания;

оператор нажимает кнопку «ЗСР»;

ответ от сервера расписания не приходит. Оператор решает самостоятельно выключить один из приборов;

оператор нажимает клавишу «ВКЛП»;

оператор выбирает первое устройство нажатием клавиши «1»;

устройство выключается;

оператор снова запрашивает расписание клавишей «ЗСР»;

от сервера приходит новое расписание;

оператор нажимает клавишу «СВПР» для вывода приборов из принудительного режима;

приборы продолжают работать по новому расписанию.

На этапе 5 можно было осуществить нажатие на клавишу «ОТМ», что отменило бы действие клавиши «ВКЛП»

### Блок передачи данных к ПЭВМ

Блок передачи данных состоит из драйвера MAX232 и COM-порта, соединяющего само устройство и удаленный сервер.

Сигнал TxD c выхода микроконтроллера поступает на схему формирования уровней сигналов интерфейса RS-232 (MAX232), далее через разъем, в усиленном состоянии, он уходит на линию связи.

Сигнал RxD, поступающий от сервера расписания, проходит через преобразователь, ослабляется, и попадает на вход микроконтроллера RxD

Усиление и ослабление сигнала необходимо, т. к. передача данных по кабелю требует большего уровня сигнала из-за затухания. Без усиления посылаемого сигнала он может в процессе достижения своей цели настолько ослабнуть, что ПЭВМ его не сможет воспринять. В случае с сигналами, которые поступают на микроконтроллер, их необходимо ослаблять во избежание сгорания микроконтроллера.

Модуль передачи данных USART настроен в данной разрабатываемой МК-системе следующим образом;

* скорость передачи данных 9600 бит в секунду;
* 8 бит данных в кадре;
* 1 стоповый бит;
* бит четности отключен.

Кадр UART изображен на рисунке 5.

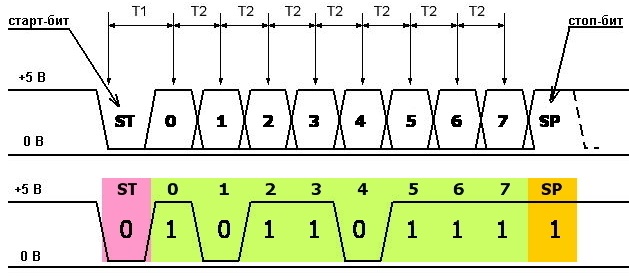


Рисунок – Кадр UART

Отключение бита четности и отключение второго стопового бита обусловлено стремлением к большей скорости передачи данных.

Модуль USART ATmega8515 может принимать в одному кадре до 9 информационных бит, однако в данном случае, для упрощения алгоритмов обработки данных и более наглядного вида передаваемых данных было принято решение использовать 8 бит – размер байта памяти данных.

Получившаяся итоговая конфигурация является достаточно простой для понимания и отладки и одновременно высокопроизводительной.

### Блок реле

Для управление блоком реле используется порт A. Каждый вывод порта A подключён к соответствующему реле для управления питанием прибора. При этом уровень логической единицы означает, что прибор в данный момент находится во включенном состоянии, а уровень логического нуля, соответственно, означает, что прибор в данный момент времени находится в выключенном состоянии.

Для вывода состояния приборов на порт A внутри программы микроконтроллера используется алгоритм, который по записанному в память SRAM расписанию определяет какие приборы в данный моменты времени должны быть включены или выключены.

### LCD-дисплей

LCD дисплей представляет из себя жидкокристаллический индикатор, сделанный на основе жидких кристаллов. С его помощью, в простых устройства, можно отображать простые графические объекты (буквы, цифры, специальные символы и т.д.).

В данной курсовой работе для разработки МК-системы используется LCD-дисплей LM016L с монохромным экраном, сделанный на базе контроллера HD44780. В данном дисплее используется монохромный экран с двумя строками вывода.

Видимая область экрана составляет по 16 символов для каждой строки, а общий объем памяти 40 символов для каждой строки. В данной курсовой работе для вывода текущего времени микроконтроллера достаточно одной видимой области экрана, потому сдвиг зоны отображения информации на экране не производится.

Схема отображения символов из ячеек памяти отображена на рисунке 6. Ячейки пронумерованы в 16-ричной системе счисления.

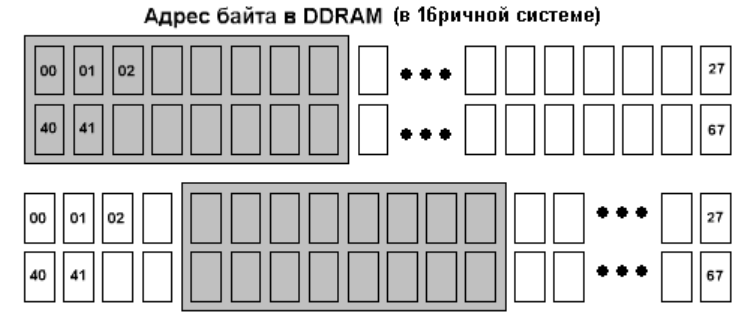


Рисунок – Окно отображения дисплея LM016L

На представленном выше рисунке видно, что окно, сдвигаемое на некоторое количество ячеек в сторону, позволяет отобразить информацию, которая скрыта за областью отображения. Это позволяет хранить информацию в дисплее и отображать ее в случае необходимости.

Данный дисплей обладает 8 информационными и 3 управляющими входами. В зависимости от сигналов управления, последовательность бит, пришедшая на информационные входы, воспринимается как очередной символ или одна из команд.

Перечень управляющих сигналов:

* **E** — стробирующий вход. Отрицательным перепадом напряжения на этой линии мы даем понять дисплею что нужно забирать/отдавать данные с/на шину данных;
* **RW** — определяет в каком направлении у нас движутся данные. Если 1 — то на чтение из дисплея, если 0 то на запись в дисплей;
* **RS** — определяет, что передается команда (RS=0) или данные (RS=1). Данные будут записаны в память по текущему адресу, а команда исполнена контроллером.

Допустимый набор команды для дисплея LM016L представлен в таблице 3.

Таблица – Команды управления дисплеем LM016L



Значения каждых битов следующие:

I/D — инкремент или декремент счетчика адреса. По дефолту стоит 0 — Декремент. Т.е. каждый следующий байт будет записан в n-1 ячейку. Если поставить 1 — будет Инкремент;

S — сдвиг экрана, если выставить 1, то с каждым новым символом будет сдвигаться окно экрана, пока не достигнет конца DDRAM;

D — включить дисплей. Если поставить, 0 то изображение исчезнет, а если 1 – изображение наоборот появится;

С — включить курсор в виде прочерка. Для включения курсора необходимо, чтобы бит был равен 1;

B — сделать курсор в виде мигающего черного квадрата;

S/C сдвиг курсора или экрана. Если стоит 0, то сдвигается курсор. Если 1, то экран. По одному разу за команду;

R/L — определяет направление сдвига курсора и экрана. 0 — влево, 1 — вправо;

D/L — бит определяющий ширину шины данных. 1-8 бит, 0-4 бита;

N — число строк. 0 — одна строка, 1 — две строки;

F — размер символа 0 — 5х8 точек. 1 — 5х10 точек;

AG — адрес в памяти CGRAM;

АD — адрес в памяти DDRAM.

Совокупность посланных информационных сигналов воспринимается как команды в случае, если управляющий вход RS=0.

## Разработка принципиальной схемы

### Синтез принципиальной схемы

По результатам проектирования МК-системы и разработке функциональной схемы, а также анализу возможностей и требований к реализуемому устройству, был сформулирован перечень необходимых компонентов, а также способы их подключения. На основе выделенных компонентов была разработана принципиальная схема.

В результате принципиальная схема может быть представлена следующими основными компонентами:

Микроконтроллер ATmega8515;

Блок реле для управления питанием приборов;

Матричная клавиатура 4x4;

LCD-дисплей для индикации текущего времени МК-системы;

Преобразователь входного напряжения.

### Конфигурация выводов микроконтроллера

Основным узлом разрабатываемого устройства в данной курсовой работе является микроконтроллер ATmega8515. Конфигурация выводов микроконтроллера в корпусе PDIP приведена на рисунке 7.

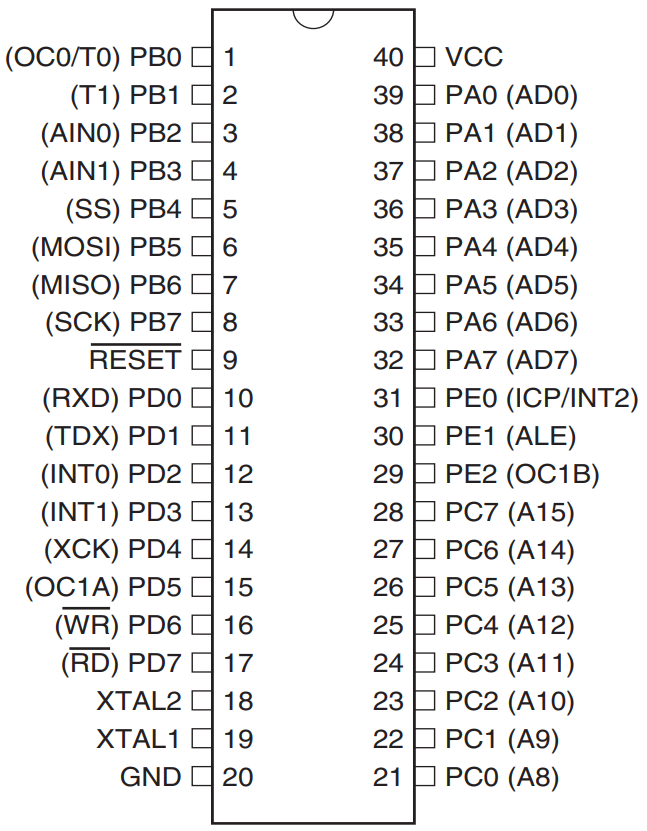


Рисунок – Микроконтроллер ATmega8515 в корпусе PDIP

### Схема понижения входного напряжения до 5В

На схему устройства подается напряжение 12В. Сам микроконтроллер и другие устройства принципиальной схемы работают от напряжения питания 5В. Для преобразования 12В в 5В требуется использовать устройство, понижающее напряжение до необходимого уровня.

Для решения данной задачи используется стабилизатор напряжения KP142EH5A. Для нормальной работы данный стабилизатор шунтируется конденсаторами на входе и выходе. Схема подключения представлена на рисунке 8.

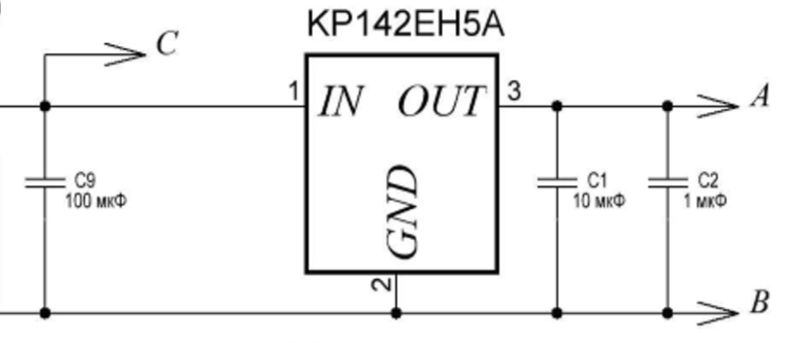


Рисунок – Схема подключения стабилизатора напряжения KP142EH5A

В схеме выше подключение конденсатор C9 используется для сглаживания скачков напряжения питания на входе, а конденсаторы C1 и C2 используются в качестве фильтров, сглаживая подаваемое напряжения на устройства.

### Подключение матричной клавиатуры

Матричная клавиатура представляет из себя блок кнопок, в котором клавиши размещены в виде матрицы на пересечении горизонтальных и вертикальных линий связи.

В данном разрабатываемом устройстве вертикальные линии подключены к входному регистру, а горизонтальный ряд к выходному регистру. На входной регистр подается код, который содержит 0 уровень сигнала на одном из разрядов и 1 на всех остальных. При замыкании кнопки вертикального ряда, на котором в данный момент присутствует сигнал 0, этот сигнал поступает на горизонтальную линию и по ней на выходной регистр. Проверяя состояние выходного регистра, микроконтроллер может идентифицировать строку и номер замкнутой кнопки.

С помощью последовательного опроса кодами вида: 1110, 1101, 1011 и 0111 – можно опросить состояния всех столбцов клавиатуры и установить номер замкнутой кнопки.

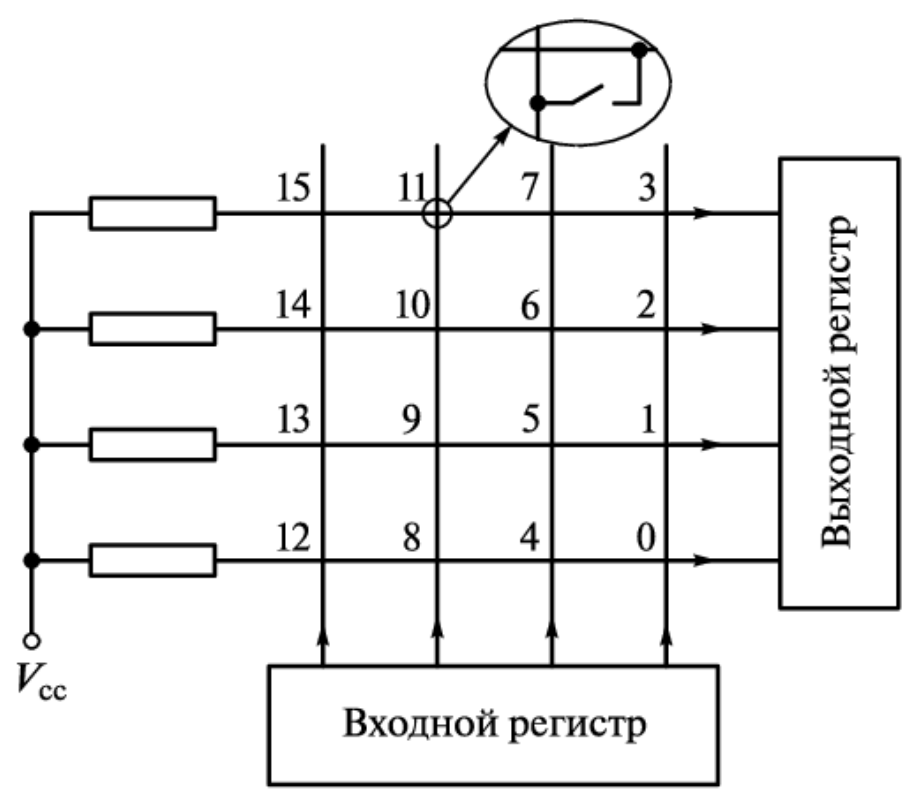


Рисунок – Матричная клавиатура

Входной регистр данной матричной клавиатуры подключается к входам 4, 5, 6, 7 порта C.

Выходной регистр подключается к входам 0, 1, 2, 3 того же порта C микроконтроллера ATmega8515.

Таким образом, все 16 кнопок обрабатываются с помощью 8 пинов порта C микроконтроллера. В случае, если бы каждая кнопка подключалась простейшим образом (1 кнопка – 1 вывод порта), то пришлось бы использовать в 2 раза больше выводов, что могло бы привести к дефициту выводов микроконтроллера.

Подключение клавиатуры матричным способом позволяет довольно сильно экономить на количестве необходимых выводов для считывания состояний кнопок. При этом чем больше размерность матричной клавиатуры, тем более эффективен такой способ подключения.

При этом стоит отметить, что минимальный размер матричной клавиатуры, в котором есть смысл – это 3x2 или 2x3.

### Подключение LCD-дисплея

LCD-дисплей LM016L может работать в 2-х режимах:

8-битный режим. В этом режиме данные передаются сразу одним байтом за один такт. При этом используются сразу 8 контактов;

4-битный режим. В этом режиме данные передаются полубайтами, используя только 4 контакта. Для передачи байта в данном режиме необходимо передавать байт данных в 2 такта.

В целях экономии выводов микроконтроллера в данной курсовой работе используется 4-битный режим.

Схема подключения этого дисплея показана на рисунке 10.

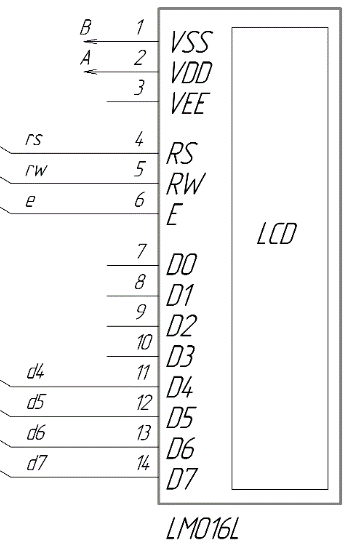


Рисунок – Подключение LCD-дисплея LM016L

Как видно из рисунка выше, информационные входы D0 – D3 не используются, т.к. данный дисплей используется в 4-битном режиме. Вход VEE (контрастность) также не используется, поскольку нет необходимости динамически менять контрастность дисплея в процессе его работы.

## Расчёт параметров

### Расчет параметров настройки таймеров

В разработанном устройстве используется 2 таймера – T0 и T1. 8-разрядный таймер T0 используется для обновления экрана LCD-дисплея с частотой 50 Гц. 16-разрядный таймер T1 используется для счёта текущего времени, вызывая прерывания по переполнению каждую 1 секунду.

Для таймера T0 при частоте микроконтроллера 8 000 000 Гц необходим вызов прерывания 50 раз в секунду, т.е. каждые тактов. Число 160 000 можно разбить на 2 множителя, например 160 000, где 1024 – это делитель частоты CK, а 156 – непосредственно количество отсчитываемых таймером тактов с учетом коэффициента деления.

Для таймера T1 при частоте работы МК 8 МГц необходим вызов прерывания по переполнению каждую секунду, т.е. каждые 8 000 000 тактов. Это количество тактов можно разбить на 2 множителя где 256 – делитель частоты CK, а 31 250 – количество отсчитываемых таймером T1 тактов с учетом коэффициента деления.

Таким образом, для таймера T0 начальное значение должно равняться 256 – 156 = 100, коэффициент деления должен равняться 1024 (CS02 = 1; CS01 = 0; CS00 = 1), а для таймера T1 начальное значение должно равняться 65 536 – 31 250 = 34 286, коэффициент деления должен равняться 256 (CS12 = 1; CS11 = 0; CS10 = 0).

### Расчёт сопротивления резисторов для катушек реле

Для переключения каждого реле W107DIP-3 требуется ток в 10 мА. С учетом того, что сама первичная обмотка обладает сопротивляемостью в 1000 Ом, то для достижения оптимального тока необходимо перед каждым реле поставить токоограничивающий резистор в 200 Ом, при условии, что подаваемое напряжением переключения реле составляет 12 В.

### Расчёт потребляемой мощности

Оценка мощности, потребляемой микроконтроллером, будет производиться при условии пикового режима его работы.

Для подсчета потребляемой мощности спроектированной МК-системы, воспользуемся графиком потребляемого тока микроконтроллера ATmega8515 в зависимости от частоты и питающего напряжения. Сам график приведен на рисунке 11.

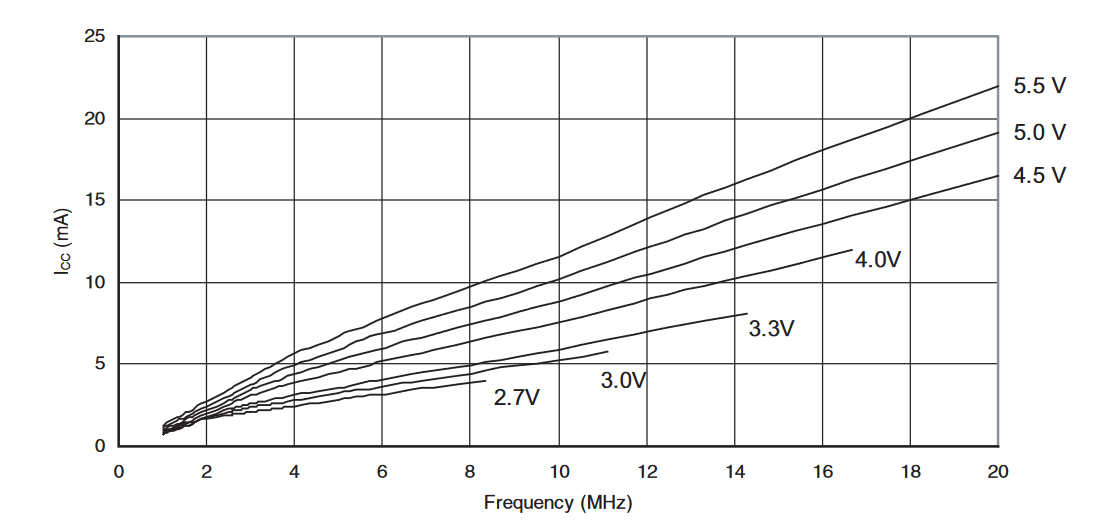


Рисунок – График зависимости потребляемого тока для ATmega8515

По графику определяем ток 8 мА.

Потребление LCD-дисплея LM016L при напряжении питания 5В составляет в пике 3 мА, согласно datasheet. Потребление драйвера MAX232 при передаче данных может составлять максимум 22 мА при скорости передачи в 9600 бод/с. Рассеиваемая мощность на стабилизаторе напряжения KP142EH5A равна падению напряжения на нем 7В, умноженному на суммарный ток, проходящий через него 33 мА. Через каждое реле в активном режиме проходи ток в 10мА.

Потребляемую устройствами мощность можно определить по следующей формуле: P = Icc \* Uпит.

Потребляемая мощность устройств представлена в таблице 4.

Таблица – Потребляемая мощности компонентов МК-системы

|  |  |  |  |
| --- | --- | --- | --- |
| Микросхема | Pone, мВт | Количество | Psumm, мВт |
| ATmega8515 | 40 | 1 | 40 |
| KP142EH5A | 231 | 1 | 231 |
| MAX232 | 110 | 1 | 110 |
| LM016L | 15 | 1 | 15 |
| W107DIP-3 | 120 | 8 | 960 |

Суммарная потребляемая мощность составляет около 1 356 мВт.

Основные потребители – стабилизатор напряжения драйвер, MAX232 и 8 реле W107DIP-3. Однако вычисленная мощность является максимально возможной. В штатном режиме работы интенсивность передачи данных по драйверу MAX232 будет на порядок меньше. Одновременное включение всех устройств также не будет постоянным явлением.

## Описание алгоритмов функционирования устройства

### Схемы-алгоритмы работы программы МК

На рисунке 12 представлена обобщенная схема-алгоритм работы разработанной программы для микроконтроллера ATmega8515. Данная схема дает общее представление о принципах работы устройства управления приборами жилого помещения, и как оператор может взаимодействовать с разработанной МК-системой с помощью пульта оператора.

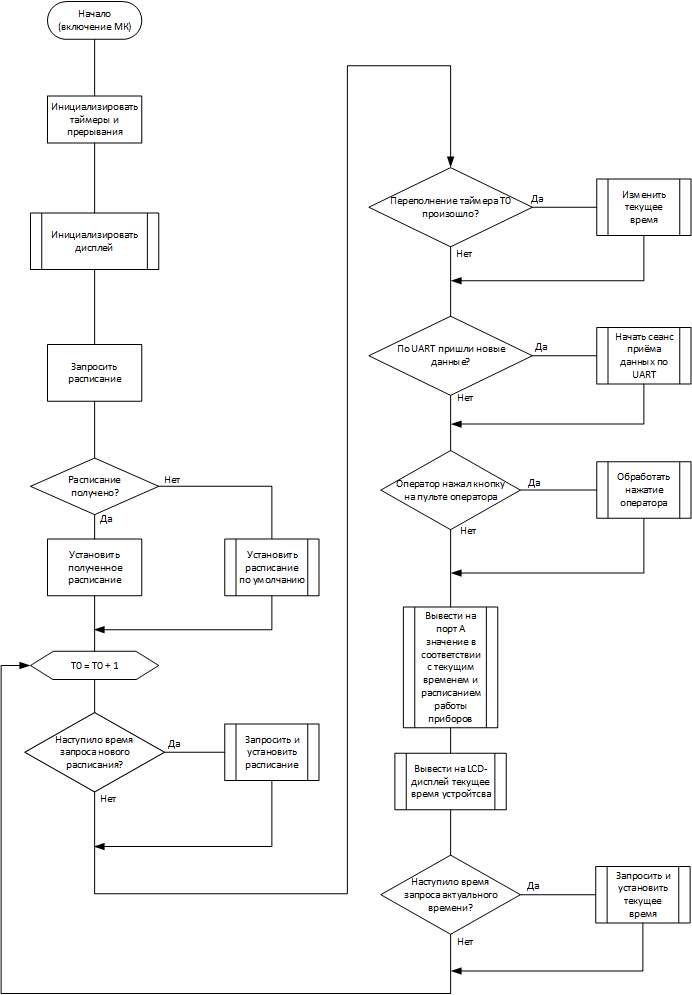


Рисунок – Обобщенная схема-алгоритма работы программы

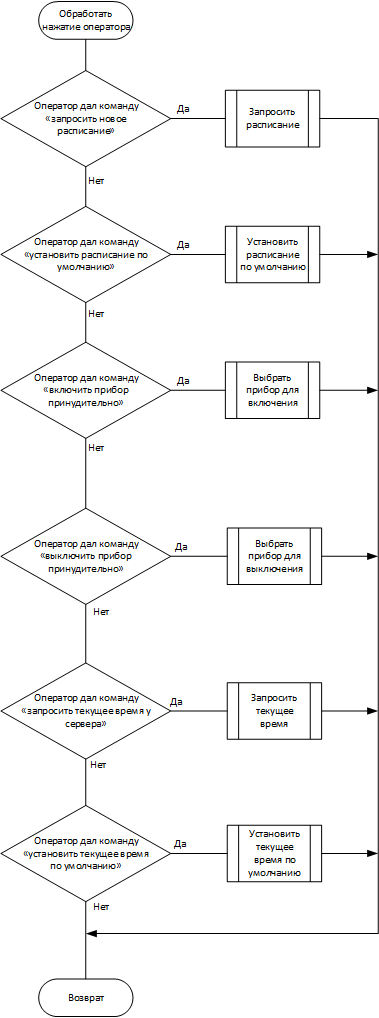


Рисунок – Схема-алгоритм обработки нажатия клавиши на пульте оператора

Алгоритм обработки нажатия кнопки на пульте оператора представлен на рисунке 13. Здесь, в зависимости от нажатой кнопки, производится любо некоторая операция, либо ожидается следующее контекстное нажатие. Не предполагается нажатие сразу нескольких клавиш или их долгое удержание в качестве дополнительной функциональности.

Само срабатывание клавиши происходит в момент отпускания клавиши, а не сразу при её нажатии.

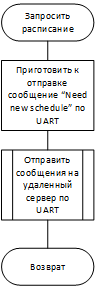


Рисунок – Схема-алгоритм отправки запроса нового расписания



Рисунок – Схема-алгоритм отправки запроса текущего времени

При запросе нового расписания или текущего времени программа не переходит в режим какого-либо ожидания приёма, а продолжает работу в штатном режиме. Сеанс приема расписания начинается в момент начала передачи ответа от сервера.



Рисунок – Схема-алгоритм установки расписания по умолчанию

Установка расписания по умолчанию осуществляется посредством занесения вместо временных меток для включения и выключения приборов по расписанию, получаемому от удаленного сервера, временных меток из памяти программ, определенных заранее на этапе программирования микроконтроллера.

Установка времени по умолчанию осуществляется немного иным образом. Информация о текущем времени в секундах, минутах и часах находится в соответствующих регистрах time\_seconds, time\_minutes и time\_hours. При невозможности получения времени от сервера в эти регистры можно установить расписание, зашитое заранее в память программ на этапе программирования микроконтроллера.

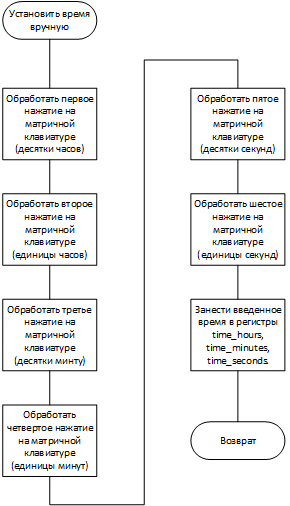


Рисунок – Схема-алгоритм установки времени вручную

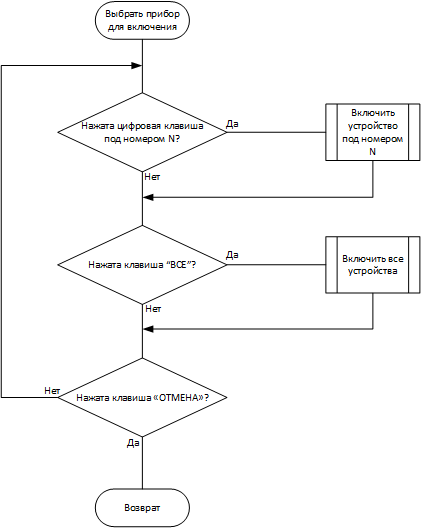


Рисунок – Схема-алгоритм выбора прибора для включения

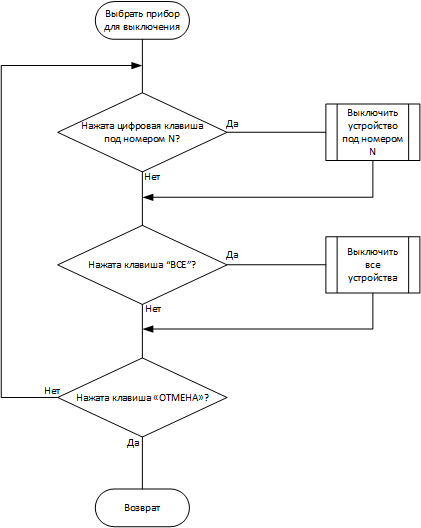


Рисунок – Схема-алгоритм выбора прибора для выключения



Рисунок 20 – Отправка сообщения по UART

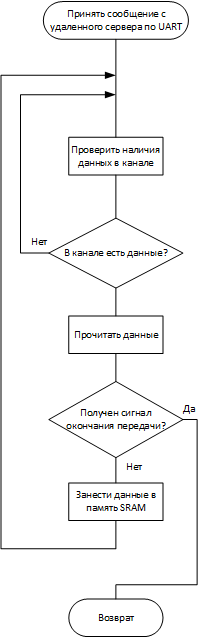


Рисунок – Прием сообщения по UART

### Формат передаваемых данных от сервера расписаний

Данные о переключении состояний приборов передаются в виде набора меток их включения и выключения в различные моменты времени. О начале передачи данных свидетельствует определенный набор бит, пришедших по каналу UART.

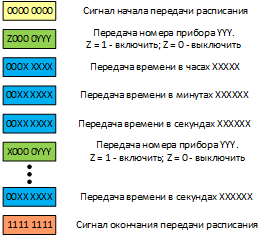


Рисунок – Формат передачи расписания

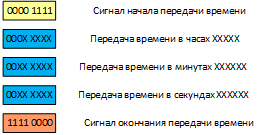


Рисунок – Формат передачи времени

Также рассматривался вариант с передачей диапазонов работы приборов, но было принято решения отказаться от такого подхода, т.к. передача временных диапазонов требует большого количества памяти для хранения, а также алгоритм проверки попадания приборов в рабочий диапазон времени был признан как более сложный, чем алгоритм с использованием меток.

# Технологическая часть

## Характеристика использованных систем разработки

Для проектирования и отладки разрабатываемой МК-системы в качестве средства разработки использованы следующие среды:

AVR Studio 4 – для отладки программного кода на ассемблере;

Proteus 8 Professional – для симуляции работы устройства.

AVR Studio 4 представляет из себя удобную и относительно простую среду для разработки программного обеспечения под микроконтроллеры фирмы AVR на языках C и Assembler.

ISIS Proteus позволяет строить и симулировать спроектированные схемы и прошивать микропроцессорные компоненты созданными программами в AVR Studio и запускать полученную виртуальную модель в режиме симуляции реального времени.

ISIS Proteus позволяет взаимодействовать с элементами ввода-вывода, такими как светодиодные и LCD-дисплеи, а также с исполнительными механизмами (кнопки и различные другие переключатели).

Данный пакет программ специально предназначен для разработки программного обеспечения для микроконтроллеров AVR. Он дает возможность достаточно тщательно отлаживать разработанные программы, предоставляя визуальные инструменты и отображая состояния всех регистров используемого микроконтроллера.

## Оценка количества задействованной памяти микроконтроллера ATmega8515

Среда AVR Studio позволяет определять процент используемой памяти микроконтроллера. Количество задействованной памяти представлено на рисунке 24.

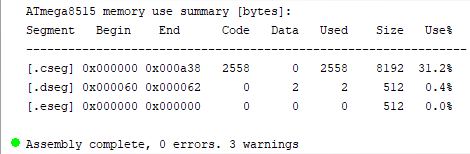


Рисунок – Количество занимаемой памяти программой

## Симуляция в Proteus 8

Отладка с помощью пакета программ ISIS Proteus дает возможность получить наглядный результат моделирования разработанной МК-системы.

Для симуляции работы MK и датчиков построена упрощенная схема в Proteus 8, представленная на рисунке 25. Эта схема состоит из матрицы кнопок, LCD-дисплея, драйвера MAX232, виртуального терминала и блока реле.

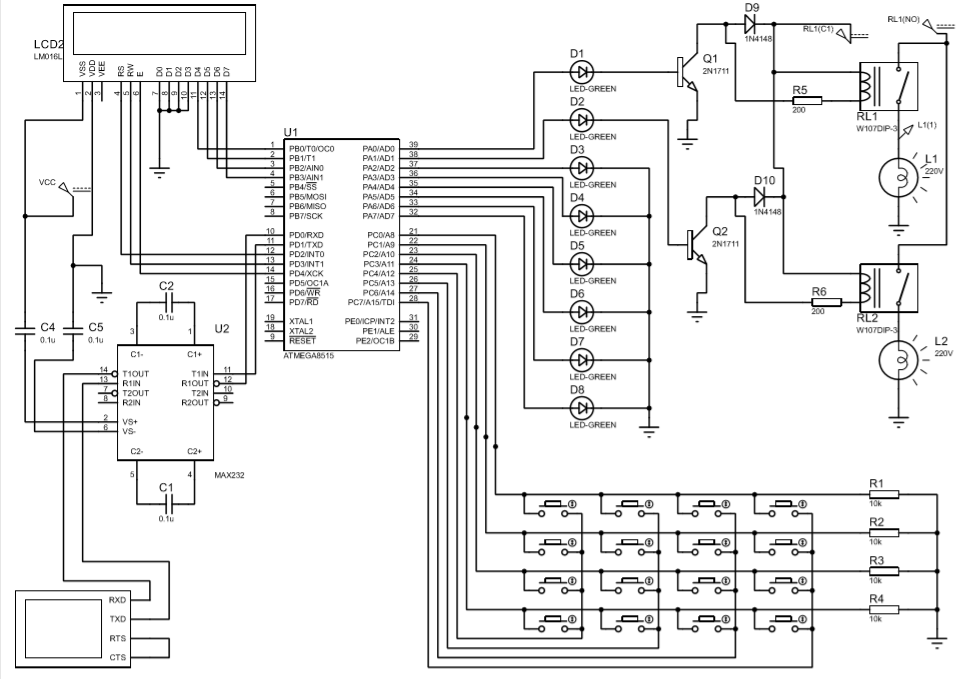


Рисунок – Упрощенная схема разрабатываемой МК-системы

Для того, чтобы не перегружать схему однотипными компонентами, в блоке реле показаны только 2 компонента с реле, остальные заменены простыми светодиодами, которые олицетворяют собой работу приборов.

На рисунке 26 представлен скриншот работающей модели.

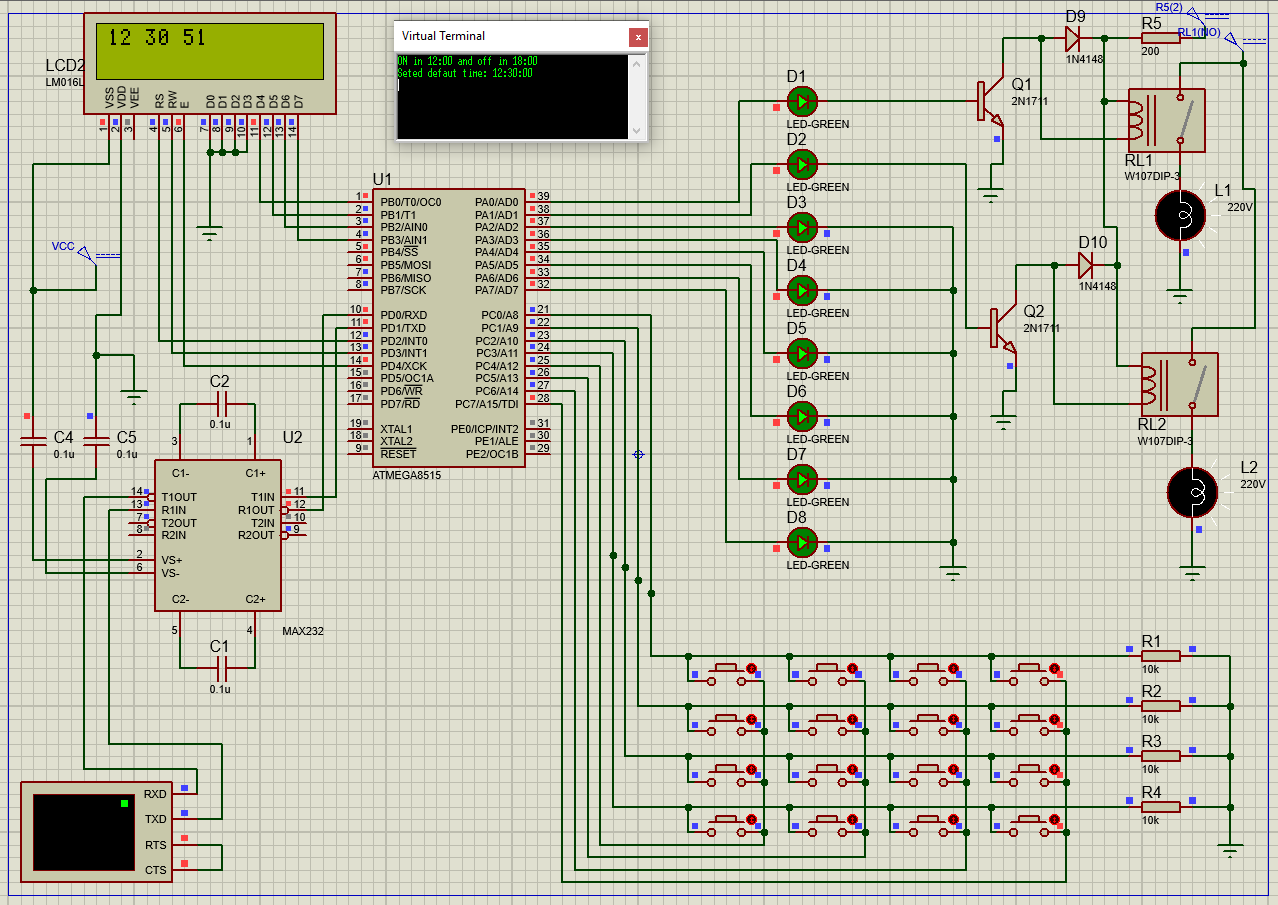


Рисунок – Симуляция в Proteus

Матричная клавиатура представляет из себя систему из 16 кнопок, соединенных с портами порта C определенным образом. Она служит пультом оператора для взаимодействия пользователя с устройством напрямую, без необходимости передавать информацию через сервер расписания.

Драйвер MAX232 представляет собой связующий элемент, позволяющий микроконтроллеру принимать с удаленного сервера информационные сигналы значительного большего уровня напряжения, чем 5В. Кроме того, при отправке данных на удаленный сервер, данный драйвер наоборот усиливает сигнал, чтобы он был успешно доставлен и распознан ПЭВМ.

Виртуальный терминал позволяет симулировать общение между ПЭВМ и разработанной МК-системой с помощью модуля USART микроконтроллера ATmega8515.

Схема из 8 светящихся диодов показывает, какие устройства включены или выключены, посредством вывода их текущего состояния на порт A микроконтроллера. Данная схема позволяет понять во время отладки работы МК-системы, какое из устройств в каком состоянии находится в результате действий оператора или в результате установки расписания с удаленного сервера расписаний.

## Способы программирования памяти микроконтроллера ATmega8515

Для программирования МК ATmega8515 могут быть использованы два способа:

* программирование повышенным напряжением в параллельном формате с использованием дополнительного источника питания +12В;
* внутрисхемное программирование ISP с использованием последовательного периферийного интерфейса SPI.

Рассмотрим внутрисхемное программирование ISP по интерфейсу SPI.

Это самый популярный способ прошивать современные контроллеры. Внутрисхемным данный метод называется потому, что микроконтроллер в этот момент находится в схеме целевого устройства. Для нужд программатора в этом случае выделяется несколько выводов контроллера (обычно 3..5 в зависимости от контроллера). К этим выводам подключается прошивающий шнур программатора и происходит заливка прошивки. После чего шнур отключается, и контроллер начинает работу.

Для подключения по SPI, передачи прошивки по нему в МК AVR и работы программатора нужно четыре линии и питание (достаточно только земли, чтобы уравнять потенциалы земель программатора и устройства):

SCK — тактовые импульсы интерфейса SPI, синхронизирующие все операции обмена данными.

MOSI (Master-Output/Slave-Input) — линия данных от ведущего устройства к контроллеру.

MISO (Master-Input/Slave-Output) — линия данных от контроллера к ведущему.

RESET — сигналом на RESET программатор вводит контроллер в режим программирования, потому что для разрешения прошивки по SPI нужно подать логический «0» на этот вывод.

GND — земля.

Доступны две разные схемы контактов разъёма ISP: 6-контактная и 10-контактная. Для программирования микроконтроллера был выбран Atmel 6-Pin ISP разъём.

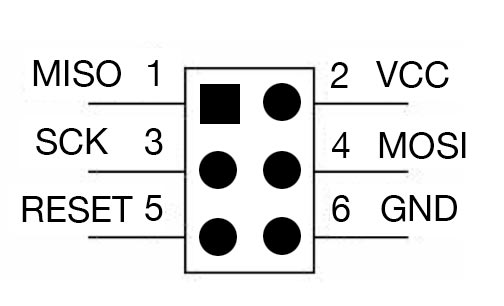


Рисунок – Разъем Atmel 6-pin ISP

### Алгоритм последовательного программирования через SPI

Взаимодействие устройств по интерфейсу SPI требует установки одного из устройств в режим ведущего, а остальных – в режим ведомого. При этом ведущее устройство отвечает за выбор ведомого и инициализацию передачи.

SPI является синхронным интерфейсом: все операции синхронизированы фронтами тактового сигнала (SCK), который вырабатывается ведущим устройством.

Программирование микроконтроллера по SPI осуществляется путём посылки 4-байтовых команд на вывод MOSI МК, в который один или два байта определяют тип операции, остальные – адрес, записываемый байт, установочные биты и биты защиты, пустой байт. При выполнении операции чтения считываемый байт снимается через вывод MISO. Так же можно запрограммировать память данных EEPROM.

Во время последовательной записи в ATmega8515 данные тактируется нарастающим фронтом SCK. Во время чтения данные тактируются спадающим фронтом SCK. Временная диаграмма представлена на рисунке 17.

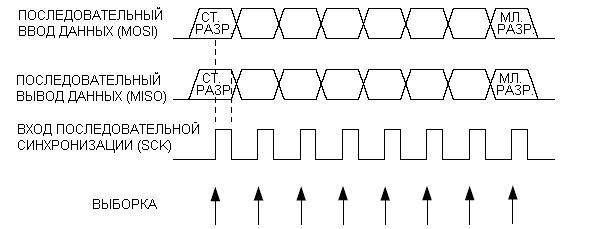


Рисунок – Последовательное программирование по интерфейсу SPI

Последовательность подачи питания: подать напряжение питания между VCC и GND, когда на входах RESET и SCK присутствует лог. 0. В некоторых системах программатор не может гарантировать, что SCK = 0 при подаче питания. В этом случае необходимо сформировать положительный импульс на RESET длительностью не менее двух тактов ЦПУ после того, как для SCK установлено значение «0».

Задержка не менее 20 мс и разрешение последовательного программирования путём записи команды разрешения последовательного программирования через вход MOSI.

Инструкции последовательного программирования не выполняются, если связь не синхронизирована. Когда связь синхронизирована, будет возвращаться значение второго байта ($53) от МК при отправке третьего байта команды разрешения программирования. Независимо от того, корректно или нет принятое значение, все четыре байта инструкции должны быть переданы. Если принятое значение не равно $53, то формируется положительный импульс на входе RESET и вводится новая команда разрешения программирования.

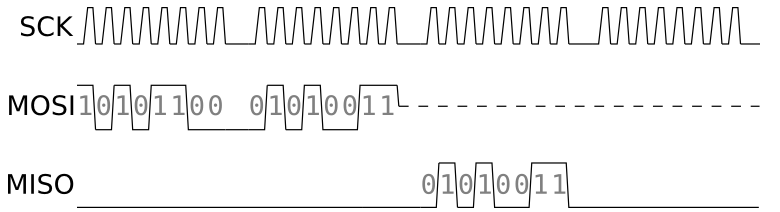


Рисунок – Команда «Program Enable»

Flash память программируется постранично, размер страницы составляет 64 байта. В страницу памяти загружается по одному байту за раз, предоставляя 5 младших бит адреса и данные вместе с командой загрузки страницы памяти программ. Чтобы обеспечить корректную загрузку страницы, сначала необходимо записать младший байт, а затем старший байт данных по данному адресу. Страница памяти программ сохраняется путём загрузки инструкции «запись страницы памяти программ" с 7 младшими битами адреса страницы. Доступ к интерфейсу последовательного программирования до завершения операции записи во Flash может привести к неправильному программированию.

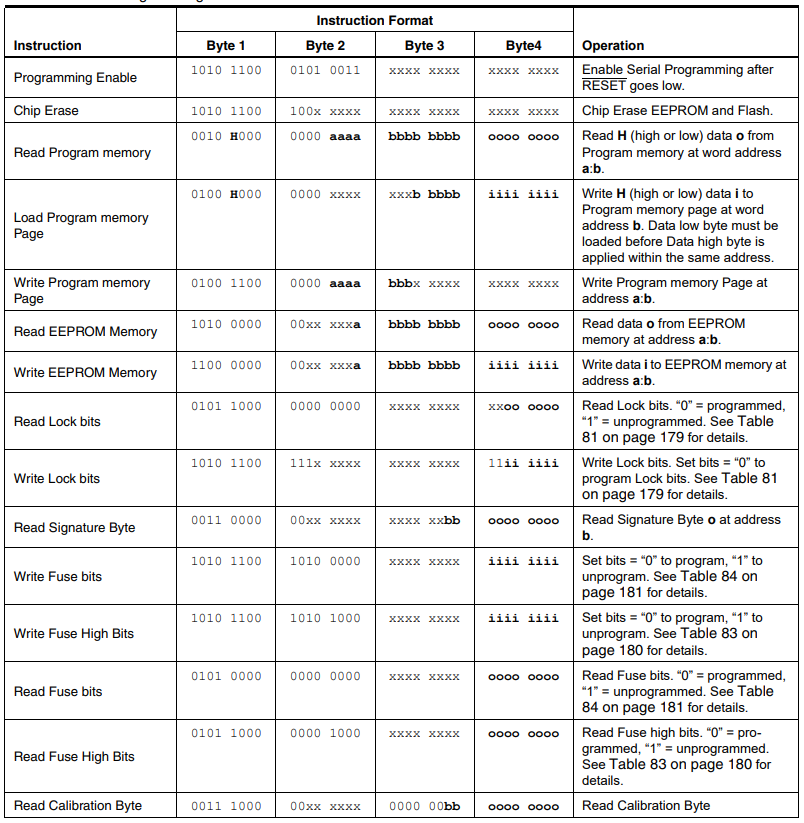
Массив памяти EEPROM программируется побайтно, при этом в инструкции записи указывается адрес и данные. Перед записью новых данных первоначально автоматически стирается адресуемая ячейка EEPROM.

Любую ячейку памяти можно проверить, используя инструкции чтения, которые возвращают содержимое ячейки по указанному адресу путём последовательной передачи на выходе MISO.

По завершении программирования вход RESET должен быть переведён в высокое состояние для возобновления нормальной работы.

Последовательность выключения (при необходимости): установка RESET = "1", отключить питание VCC.

Таблица – Формат байтов команд для программирования



Примечание:

* a - адрес старших разрядов;
* b - адрес младших разрядов;
* H - 0 - мл. байт, 1 - ст. байт;
* o - вывод данных;
* i - ввод данных;
* x - произвольное значение.

### Опрос данных Flash памяти

Когда страница программируется во Flash, при чтении по адресам в пределах данной страницы возвращается $FF. Микроконтроллер готов к записи новой страницы, если запрограммированное значение считано корректно.

Это используется для определения момента, когда может быть загружена следующая страница. Важно, что запись выполняется всей страницы одновременно, и любой адрес в пределах страницы может использоваться для опроса.

Опрос данных Flash не будет работать для значения $FF, поэтому при записи этого значения необходимо подождать не менее tDD\_FLASH, прежде чем программировать следующую страницу. Поскольку микроконтроллер с очищенной памятью содержит $FF по всем адресам, можно пропустить повторную запись значения $FF.

### Опрос данных EEPROM

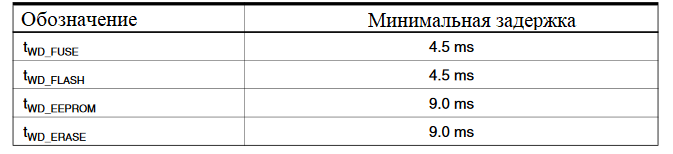
Когда новый байт был записан и в последующем программируется в EEPROM, чтение значения по этому адресу вернёт $FF. Устройство готово к новому байту, если запрограммированное значение считываться корректно.

Это используется для определения момента, когда может быть записан следующий байт.

Данное не распространяется для значения $FF, но программист должен обратить внимание на следующее: поскольку устройство с очищенной памятью содержит $FF по всем адресам, программирование ячейки значением $FF может быть пропущено.

Пропуск недопустим, если EEPROM перепрограммировано без предварительного стирания всей памяти. В этом случае опрос данных не может использоваться для значения $FF, и программист должен предусмотреть задержку не менее tWD\_EEPROM, прежде чем программировать следующий байт.

Таблица – Минимальная задержка при записи



Заключение

В результате выполнения курсового проекта было получено функциональное, структурное и принципиальное описание разработанного устройства.

Разработаны алгоритмы функционирования микроконтроллера ATmega8515. Написан код программы на языке ассемблер без использования сторонних библиотек, функций и исходных кодов.

Разработанная МК-система представляет из себя устройство управления 8 приборами жилого помещения согласно расписанию, получаемому с удаленного сервера, передаваемому по каналу UART по протоколу передачи RS-232.

Устройство обладает следующими важными при функционировании данной системы техническими характеристиками:

частота работы устройства составляет 8 МГц;

управляет до 8 приборами одновременно;

отправляет запросы по получению расписания на ПЭВМ;

отправляет запросы по получению текущего времени на ПЭВМ;

длина передачи по RS-232 на расстояние до 75 метров при 9600 бод;

обладает пультом управления оператора на 16 кнопок;

устанавливает расписанию по умолчанию в случае отсутствия связи с сервером расписания;

использует таймеры T0 и T1 для вызова прерываний;

способен хранить до 127 меток включения или выключения приборов при внутренней SRAM 512 Кбайт;

выводит текущее время МК-системы на LCD-дисплей;

работает от линии питания 12 В.

Список использованных источник

1. Хартов В.Я. Микроконтроллеры AVR. Практикум для начинающих. 2-е издание, Издательство МГТУ им. Баумана, 2012 г. – 278 с.
2. Хартов В.Я. Микропроцессорные системы: учебное пособие для студентов учреждение высшего профессионального образования, Академия, М., 2014 г. – 368 с.
3. Atmel ATmega8515 datasheet - doc2512, [Электронный ресурс] // ATmega8515 datasheet - doc2512: электронный документ ATmega8515(L) - Complete Datasheet URL: http://ww1.microchip.com/downloads/en/DeviceDoc/doc2512.pdf (дата обращения: 09.12.2019)
4. Производитель МК ATMEGA – компания Microchip [Электронный ресурс]. - URL: https://www.microchip.com/ (дата обращения 09.12.2019)

Приложение А – Исходные коды программ

**Листов 25**

.include "m8515def.inc" ;файл определений ATmega8515

.def temp = R16 ;Временный буфер

.def counter = R17 ;Счетчик для циклов

.def on\_off = R18 ;Признак того, что нужно выключить (1)

;или выключить (0) устройство

.def device\_number = R19 ;Номер устройства, которое

;необходимо включить/выключить

.def actual\_device\_statuses = R24 ;Отображает устройства, для

;которых уже выведено актуально состояние

.def temp2 = R25 ;Дополнительный временный буфер

.def a\_status = R1 ;Регистр хранения статусов устройств на порте A

.def counter2 = R2 ;Дополнительный счетчик для циклов

.def flag = R3 ;Вспомогательынй флаг для различных признаков

.def force\_devices = R4 ;Регистр устройств, запущенных в принудительном режиме

.def ascii\_numbers\_start = R5

.def byte\_to\_send = R6 ;Регистр, хранящий биты для передачи на дисплей

.def inp\_hours\_h = R7

.def inp\_hours\_l = R8

.def inp\_minutes\_h = R9

.def inp\_minutes\_l = R10

.def inp\_seconds\_h = R11

.def inp\_seconds\_l = R12

.def time\_extra = R20 ;Дополнительный регистр времени 1

.def time\_seconds = R21 ;Дополнительный регистр времени 2

.def time\_minutes = R22 ;Дополнительный регистр времени 3

.def time\_hours = R23 ;Дополнительный регистр времени 4

.equ XTALL =8000000 ;Тактовая частота в ГЕРЦАХ

.equ BAUD =9600 ;Скорость обмена данными в бит/с

.equ SPEED =(XTALL/(16\*BAUD))-1 ;Коэффициент деления для получения

;заданной скорости обмена

.dseg

.org $060

schedule\_count: .byte 1

schedule\_start: .byte 1

.cseg

.org $000

rjmp INIT

.org $006

rjmp TIME1\_OVER

rjmp TIME0\_OVER

.org $020

INIT:

;Настройка стека

ldi temp,$5F ;Установка

out SPL,temp ;указателя стека

ldi temp,$02 ;на последнюю

out SPH,temp ;ячейку ОЗУ

;Настройка UART

ldi temp, high(SPEED) ;Запись делителя

out UBRRH, temp ;для задания

ldi temp, low(SPEED) ;желаемой

out UBRRL, temp ;скорости обмена

ldi temp, (1<<UCSZ1|1<<UCSZ0) ;Выбор желаемого

out UCSRC, temp ;размера слова данных 8 бит

ldi temp, (1<<RXEN|1<<TXEN) ;Разрешение приема

out UCSRB, temp ;и передачи

;Настройка портов

ser temp ;Инициализация порта A на выход

out DDRA, temp

clr temp

out PORTA, temp

ser temp ;Инициализация порта B на выход

out DDRB, temp

clr temp

out PORTB, temp

ldi temp, 28 ;Инизиализация выводов PD2,PD3,PD4 на выход

out DDRD, temp

;Настройка дисплея

cbi PORTD, 2

cbi PORTD, 3

sbi PORTD, 4

rcall DELAY

ldi temp, 0b00000011

out PORTB, temp

cbi PORTD, 4

rcall DELAY

sbi PORTD, 4

rcall DELAY

ldi temp, 0b00000010

out PORTB, temp

cbi PORTD, 4

rcall DELAY

sbi PORTD, 4

rcall DELAY

ldi temp, 0b00000010

out PORTB, temp

cbi PORTD, 4

rcall DELAY

sbi PORTD, 4

rcall DELAY

ldi temp, 0b00001000

out PORTB, temp

cbi PORTD, 4

rcall DELAY

sbi PORTD, 4

rcall DELAY

ldi temp, 0b00000000

out PORTB, temp

cbi PORTD, 4

rcall DELAY

sbi PORTD, 4

rcall DELAY

ldi temp, 0b00001000

out PORTB, temp

cbi PORTD, 4

rcall DELAY

sbi PORTD, 4

rcall DELAY

ldi temp, 0b00000000

out PORTB, temp

cbi PORTD, 4

rcall DELAY

sbi PORTD, 4

rcall DELAY

ldi temp, 0b00000001

out PORTB, temp

cbi PORTD, 4

rcall DELAY

sbi PORTD, 4

rcall DELAY

ldi temp, 0b00000000

out PORTB, temp

cbi PORTD, 4

rcall DELAY

sbi PORTD, 4

rcall DELAY

ldi temp, 0b00000110

out PORTB, temp

cbi PORTD, 4

rcall DELAY

sbi PORTD, 4

rcall DELAY

ldi temp, 0b00000000

out PORTB, temp

cbi PORTD, 4

rcall DELAY

sbi PORTD, 4

rcall DELAY

ldi temp, 0b00001100

out PORTB, temp

cbi PORTD, 4

rcall DELAY

sbi PORTD, 4

rcall DELAY ; Закончили инициализацию дисплея

ldi temp, 0b11110000

out DDRC, temp ;Инициализируем PC0-3 на вход, PC4-7 на выход

ldi temp, 0b00001111

out PORTC, temp ;Ставим 0 на пинах PC4-7 и

;подключаем подтягивающие резисторы на порты PC0-3

;Настройка таймеров и прерываний

cli ;Запрещаем прерывания

ldi temp, (1<<TOIE0|1<<TOIE1) ;Разрешить прерывание по переполнению счетчика T0 и T1

out TIMSK, temp

ldi temp, 100 ;Установка начального

out TCNT0, temp ;значения счетчика T0 (при 8 МГц)

; Установка начального значения счетчика T1

ldi temp, 0b10000101

out TCNT1H, temp

ldi temp, 0b11101110

out TCNT1L, temp

ldi time\_seconds, 0

ldi time\_minutes, 0

ldi time\_hours, 0

ldi temp, (1<<CS02|0<<CS01|1<<CS00) ;Предделитель частоты T0 равен 1024

out TCCR0, temp ;Запуск счетчика T0

ldi temp, (1<<CS12|0<<CS11|0<<CS10) ;Предделитель частоты T1 равен 256

out TCCR1B, temp ;Запуск счетчика T1

sei ;Разрешение прерываний

MAIN:

sbis UCSRA,RXC ;Ожидание, когда бит RXC будет установлен в 1

rjmp skip\_in ;(в регистре данных есть принятый непрочитанный байт)

cli ;Временно запрещаем прерывания

in temp, UDR ;Считываем принятый байт

cpi temp, 0b00000000 ;Признак начала передачи нового расписания

rcall recieve\_schedule ;Начинаем принимать новое расписание

rcall ok\_msg ;Говорим в ответ, что всё успешно приняли

sei ;Вновь вкючаем все прерывания

rjmp main

skip\_in:

rcall out\_schedule

rcall check\_klava

rcall DELAY

rjmp main

DELAY:

; Delay 800 000 cycles (0.1 секунды задержка

; для уменьшение нагрузки на симуляцию в

; протеусе при 8.0 MHz)

ldi r19, 5

ldi r18, 15

ldi r17, 242

L1: dec r17

brne L1

dec r18

brne L1

dec r19

brne L1

ret

LOW\_DELAY:

ldi r18, 2

ldi r19, 9

L2: dec r19

brne L2

dec r18

brne L2

ret

;#### ПРОЦЕДУРА ОТПРАВКИ БАЙТА НА ДИСПЛЕЙ ####

SEND\_BYTE:

push temp ; Сохраняем данные

push temp2 ; в стеке

mov temp2, byte\_to\_send

lsr temp2

lsr temp2

lsr temp2

lsr temp2

out PORTB, temp2

rcall SEND\_HALF\_BYTE

mov temp2, byte\_to\_send

out PORTB, temp2

rcall SEND\_HALF\_BYTE

pop temp2

pop temp

ret

;#### ПРОЦЕДУРА ОТПРАВКИ ПОЛУБАЙТА НА ДИСПЛЕЙ ####

SEND\_HALF\_BYTE:

rcall LOW\_DELAY

cbi PORTD, 4

rcall LOW\_DELAY

sbi PORTD, 4

rcall LOW\_DELAY

ret

;#### ПРОЦЕДУРА ВЫСТАВЛЕНИЯ РЕЖИМА ПРИЕМА КОМАНДЫ ДЛЯ ДИПСПЛЕЯ####

SET\_COMMAND\_MODE:

rcall LOW\_DELAY

cbi PORTD, 2

rcall LOW\_DELAY

ret

;#### ПРОЦЕДУРА ВЫСТАВЛЕНИЯ РЕЖИМА ПРИЕМА ДАННЫХ ДЛЯ ДИПСПЛЕЯ####

SET\_DATA\_MODE:

rcall LOW\_DELAY

sbi PORTD, 2

rcall LOW\_DELAY

ret

;#### ПРОЦЕДУРА ОПРОСА МАТРИЧНОЙ КЛАВИАТУРЫ ####

check\_klava:

ldi temp, 0

ldi temp, 0b00011111

out PORTC, temp

sbic PINC, 3

rcall RESTART

ldi temp, (1<<5)

out PORTC, temp

sbic PINC, 3

rcall SDS

ldi temp, (1<<6)

out PORTC, temp

sbic PINC, 3

rcall STF

ldi temp, (1<<7)

out PORTC, temp

sbic PINC, 0

rcall FON

sbic PINC, 1

rcall FOFF

sbic PINC, 2

rcall GSS

sbic PINC, 3

rcall GST

ret

RESTART:

sbic PINC, 3

rjmp RESTART

ldi temp, 0

mov force\_devices, temp

ret

SDS:

sbic PINC, 3

rjmp SDS

ldi XL, low(schedule\_count)

ldi XH, high(schedule\_count)

ldi temp, 16 ;16 записей по умолчанию

st X, temp ;(по 2 для каждого устройтсва

ldi temp, 129 ;1

st Y+, temp ;заголовок, определяющий устройства

ldi temp, 12

st Y+, temp ;часы начала работы

ldi temp, 0

st Y+, temp ;минуты начала работы

ldi temp, 0

st Y+, temp ;секунды начала работы

ldi temp, 130 ;2

st Y+, temp ;заголовок, определяющий устройства

ldi temp, 12

st Y+, temp ;часы начала работы

ldi temp, 0

st Y+, temp ;минуты начала работы

ldi temp, 0

st Y+, temp ;секунды начала работы

ldi temp, 131 ;3

st Y+, temp ;заголовок, определяющий устройства

ldi temp, 12

st Y+, temp ;часы начала работы

ldi temp, 0

st Y+, temp ;минуты начала работы

ldi temp, 0

st Y+, temp ;секунды начала работы

ldi temp, 132 ;4

st Y+, temp ;заголовок, определяющий устройства

ldi temp, 12

st Y+, temp ;часы начала работы

ldi temp, 0

st Y+, temp ;минуты начала работы

ldi temp, 0

st Y+, temp ;секунды начала работы

ldi temp, 133 ;5

st Y+, temp ;заголовок, определяющий устройства

ldi temp, 12

st Y+, temp ;часы начала работы

ldi temp, 0

st Y+, temp ;минуты начала работы

ldi temp, 0

st Y+, temp ;секунды начала работы

ldi temp, 134 ;6

st Y+, temp ;заголовок, определяющий устройства

ldi temp, 12

st Y+, temp ;часы начала работы

ldi temp, 0

st Y+, temp ;минуты начала работы

ldi temp, 0

st Y+, temp ;секунды начала работы

ldi temp, 135 ;7

st Y+, temp ;заголовок, определяющий устройства

ldi temp, 12

st Y+, temp ;часы начала работы

ldi temp, 0

st Y+, temp ;минуты начала работы

ldi temp, 0

st Y+, temp ;секунды начала работы

ldi temp, 136 ;8

st Y+, temp ;заголовок, определяющий устройства

ldi temp, 12

st Y+, temp ;часы начала работы

ldi temp, 0

st Y+, temp ;минуты начала работы

ldi temp, 0

st Y+, temp ;секунды начала работы

ldi temp, 1 ;1

st Y+, temp ;заголовок, определяющий устройства

ldi temp, 18

st Y+, temp ;часы начала работы

ldi temp, 0

st Y+, temp ;минуты начала работы

ldi temp, 0

st Y+, temp ;секунды начала работы

ldi temp, 2 ;2

st Y+, temp ;заголовок, определяющий устройства

ldi temp, 18

st Y+, temp ;часы начала работы

ldi temp, 0

st Y+, temp ;минуты начала работы

ldi temp, 0

st Y+, temp ;секунды начала работы

ldi temp, 3 ;3

st Y+, temp ;заголовок, определяющий устройства

ldi temp, 18

st Y+, temp ;часы начала работы

ldi temp, 0

st Y+, temp ;минуты начала работы

ldi temp, 0

st Y+, temp ;секунды начала работы

ldi temp, 4 ;4

st Y+, temp ;заголовок, определяющий устройства

ldi temp, 18

st Y+, temp ;часы начала работы

ldi temp, 0

st Y+, temp ;минуты начала работы

ldi temp, 0

st Y+, temp ;секунды начала работы

ldi temp, 5 ;5

st Y+, temp ;заголовок, определяющий устройства

ldi temp, 18

st Y+, temp ;часы начала работы

ldi temp, 0

st Y+, temp ;минуты начала работы

ldi temp, 0

st Y+, temp ;секунды начала работы

ldi temp, 6 ;6

st Y+, temp ;заголовок, определяющий устройства

ldi temp, 18

st Y+, temp ;часы начала работы

ldi temp, 0

st Y+, temp ;минуты начала работы

ldi temp, 0

st Y+, temp ;секунды начала работы

ldi temp, 7 ;7

st Y+, temp ;заголовок, определяющий устройства

ldi temp, 18

st Y+, temp ;часы начала работы

ldi temp, 0

st Y+, temp ;минуты начала работы

ldi temp, 0

st Y+, temp ;секунды начала работы

ldi temp, 8 ;8

st Y+, temp ;заголовок, определяющий устройства

ldi temp, 18

st Y+, temp ;часы начала работы

ldi temp, 0

st Y+, temp ;минуты начала работы

ldi temp, 0

st Y+, temp ;секунды начала работы

ldi temp,'O'

rcall out\_com

ldi temp,'N'

rcall out\_com

ldi temp,' '

rcall out\_com

ldi temp,'i'

rcall out\_com

ldi temp,'n'

rcall out\_com

ldi temp,' '

rcall out\_com

ldi temp,'1'

rcall out\_com

ldi temp,'2'

rcall out\_com

ldi temp,':'

rcall out\_com

ldi temp,'0'

rcall out\_com

ldi temp,'0'

rcall out\_com

ldi temp,' '

rcall out\_com

ldi temp,'a'

rcall out\_com

ldi temp,'n'

rcall out\_com

ldi temp,'d'

rcall out\_com

ldi temp,' '

rcall out\_com

ldi temp,'o'

rcall out\_com

ldi temp,'f'

rcall out\_com

ldi temp,'f'

rcall out\_com

ldi temp,' '

rcall out\_com

ldi temp,'i'

rcall out\_com

ldi temp,'n'

rcall out\_com

ldi temp,' '

rcall out\_com

ldi temp,'1'

rcall out\_com

ldi temp,'8'

rcall out\_com

ldi temp,':'

rcall out\_com

ldi temp,'0'

rcall out\_com

ldi temp,'0'

rcall out\_com

ldi temp,0x0A ;"ПЕРЕВОД СТРОКИ" Перевод курсора на строку ниже

rcall out\_com

ldi temp,0x0D ;"ВОЗВРАТ КАРЕТКИ" Переход на начало текущей строки

rcall out\_com

ret

SDT:

sbic PINC, 3

rjmp SDT

ldi time\_seconds, 0

ldi time\_minutes, 30

ldi time\_hours, 12

ldi temp,'S'

rcall out\_com

ldi temp,'e'

rcall out\_com

ldi temp,'t'

rcall out\_com

ldi temp,'e'

rcall out\_com

ldi temp,'d'

rcall out\_com

ldi temp,' '

rcall out\_com

ldi temp,'d'

rcall out\_com

ldi temp,'e'

rcall out\_com

ldi temp,'f'

rcall out\_com

ldi temp,'a'

rcall out\_com

ldi temp,'u'

rcall out\_com

ldi temp,'t'

rcall out\_com

ldi temp,' '

rcall out\_com

ldi temp,'t'

rcall out\_com

ldi temp,'i'

rcall out\_com

ldi temp,'m'

rcall out\_com

ldi temp,'e'

rcall out\_com

ldi temp,':'

rcall out\_com

ldi temp,' '

rcall out\_com

ldi temp,'1'

rcall out\_com

ldi temp,'2'

rcall out\_com

ldi temp,':'

rcall out\_com

ldi temp,'3'

rcall out\_com

ldi temp,'0'

rcall out\_com

ldi temp,':'

rcall out\_com

ldi temp,'0'

rcall out\_com

ldi temp,'0'

rcall out\_com

ldi temp,0x0A ;"ПЕРЕВОД СТРОКИ" Перевод курсора на строку ниже

rcall out\_com

ldi temp,0x0D ;"ВОЗВРАТ КАРЕТКИ" Переход на начало текущей строки

rcall out\_com

ret

FON:

ldi temp, 0

mov flag, temp

fon\_cicle:

ldi temp, (1<<4)

out PORTC, temp

sbic PINC, 0

rcall ON\_SEVEN

sbic PINC, 1

rcall ON\_FOUR

sbic PINC, 2

rcall ON\_ONE

sbic PINC, 3

rcall CANSEL

ldi temp, (1<<5)

out PORTC, temp

sbic PINC, 0

rcall ON\_EIGHT

sbic PINC, 1

rcall ON\_FIVE

sbic PINC, 2

rcall ON\_TWO

ldi temp, (1<<6)

out PORTC, temp

sbic PINC, 0

rcall ON\_ALL

sbic PINC, 1

rcall ON\_SIX

sbic PINC, 2

rcall ON\_THREE

rcall DELAY

mov temp, flag

cpi temp, 0

breq fon\_cicle

ret

ON\_ONE:

sbic PINC, 2

rjmp ON\_ONE

ldi temp, (1<<0)

or a\_status, temp

or force\_devices, temp

out PORTA, a\_status

mov flag, temp

ret

ON\_TWO:

sbic PINC, 2

rjmp ON\_TWO

ldi temp, (1<<1)

or a\_status, temp

or force\_devices, temp

out PORTA, a\_status

mov flag, temp

ret

ON\_THREE:

sbic PINC, 2

rjmp ON\_THREE

ldi temp, (1<<2)

or a\_status, temp

or force\_devices, temp

out PORTA, a\_status

mov flag, temp

ret

ON\_FOUR:

sbic PINC, 1

rjmp ON\_FOUR

ldi temp, (1<<3)

or a\_status, temp

or force\_devices, temp

out PORTA, a\_status

mov flag, temp

ret

ON\_FIVE:

sbic PINC, 1

rjmp ON\_FIVE

ldi temp, (1<<4)

or a\_status, temp

or force\_devices, temp

out PORTA, a\_status

mov flag, temp

ret

ON\_SIX:

sbic PINC, 1

rjmp ON\_SIX

ldi temp, (1<<5)

or a\_status, temp

or force\_devices, temp

out PORTA, a\_status

mov flag, temp

ret

ON\_SEVEN:

sbic PINC, 0

rjmp ON\_SEVEN

ldi temp, (1<<6)

or a\_status, temp

or force\_devices, temp

out PORTA, a\_status

mov flag, temp

ret

ON\_EIGHT:

sbic PINC, 0

rjmp ON\_EIGHT

ldi temp, (1<<7)

or a\_status, temp

or force\_devices, temp

out PORTA, a\_status

mov flag, temp

ret

ON\_ALL:

sbic PINC, 0

rjmp ON\_ALL

ldi temp, 255

or a\_status, temp

or force\_devices, temp

out PORTA, a\_status

mov flag, temp

ret

CANSEL:

sbic PINC, 3

rjmp CANSEL

ldi temp, 255

mov flag, temp

ret

FOFF:

ldi temp, 0

mov flag, temp

foff\_cicle:

ldi temp, (1<<4)

out PORTC, temp

sbic PINC, 0

rcall OFF\_SEVEN

sbic PINC, 1

rcall OFF\_FOUR

sbic PINC, 2

rcall OFF\_ONE

sbic PINC, 3

rcall CANSEL

ldi temp, (1<<5)

out PORTC, temp

sbic PINC, 0

rcall OFF\_EIGHT

sbic PINC, 1

rcall OFF\_FIVE

sbic PINC, 2

rcall OFF\_TWO

ldi temp, (1<<6)

out PORTC, temp

sbic PINC, 0

rcall OFF\_ALL

sbic PINC, 1

rcall OFF\_SIX

sbic PINC, 2

rcall OFF\_THREE

rcall DELAY

mov temp, flag

cpi temp, 0

breq foff\_cicle

rcall DELAY

ret

OFF\_ONE:

sbic PINC, 2

rjmp OFF\_ONE

ldi temp, (1<<0)

or force\_devices, temp

com temp

and a\_status, temp

out PORTA, a\_status

mov flag, temp

ret

OFF\_TWO:

sbic PINC, 2

rjmp OFF\_TWO

ldi temp, (1<<1)

or force\_devices, temp

com temp

and a\_status, temp

out PORTA, a\_status

mov flag, temp

ret

OFF\_THREE:

sbic PINC, 2

rjmp OFF\_THREE

ldi temp, (1<<2)

or force\_devices, temp

com temp

and a\_status, temp

out PORTA, a\_status

mov flag, temp

ret

OFF\_FOUR:

sbic PINC, 1

rjmp OFF\_FOUR

ldi temp, (1<<3)

or force\_devices, temp

com temp

and a\_status, temp

out PORTA, a\_status

mov flag, temp

ret

OFF\_FIVE:

sbic PINC, 1

rjmp OFF\_FIVE

ldi temp, (1<<4)

or force\_devices, temp

com temp

and a\_status, temp

out PORTA, a\_status

mov flag, temp

ret

OFF\_SIX:

sbic PINC, 1

rjmp OFF\_SIX

ldi temp, (1<<5)

or force\_devices, temp

com temp

and a\_status, temp

out PORTA, a\_status

mov flag, temp

ret

OFF\_SEVEN:

sbic PINC, 0

rjmp OFF\_SEVEN

ldi temp, (1<<6)

or force\_devices, temp

com temp

and a\_status, temp

out PORTA, a\_status

mov flag, temp

ret

OFF\_EIGHT:

sbic PINC, 0

rjmp OFF\_EIGHT

ldi temp, (1<<7)

or force\_devices, temp

com temp

and a\_status, temp

out PORTA, a\_status

mov flag, temp

ret

OFF\_ALL:

sbic PINC, 0

rjmp OFF\_ALL

ldi temp, 255

or force\_devices, temp

com temp

and a\_status, temp

out PORTA, a\_status

ldi temp, 1

mov flag, temp

ret

GSS:

sbic PINC, 2

rjmp GSS

ldi temp,'N'

rcall out\_com

ldi temp,'e'

rcall out\_com

ldi temp,'e'

rcall out\_com

ldi temp,'d'

rcall out\_com

ldi temp,' '

rcall out\_com

ldi temp,'n'

rcall out\_com

ldi temp,'e'

rcall out\_com

ldi temp,'w'

rcall out\_com

ldi temp,' '

rcall out\_com

ldi temp,'s'

rcall out\_com

ldi temp,'c'

rcall out\_com

ldi temp,'h'

rcall out\_com

ldi temp,'e'

rcall out\_com

ldi temp,'d'

rcall out\_com

ldi temp,'u'

rcall out\_com

ldi temp,'l'

rcall out\_com

ldi temp,'e'

rcall out\_com

ldi temp,0x0A ;"ПЕРЕВОД СТРОКИ" Перевод курсора на строку ниже

rcall out\_com

ldi temp,0x0D ;"ВОЗВРАТ КАРЕТКИ" Переход на начало текущей строки

rcall out\_com

ret

GST:

sbic PINC, 3

rjmp GST

ldi temp,'N'

rcall out\_com

ldi temp,'e'

rcall out\_com

ldi temp,'e'

rcall out\_com

ldi temp,'d'

rcall out\_com

ldi temp,' '

rcall out\_com

ldi temp,'a'

rcall out\_com

ldi temp,'c'

rcall out\_com

ldi temp,'t'

rcall out\_com

ldi temp,'u'

rcall out\_com

ldi temp,'a'

rcall out\_com

ldi temp,'l'

rcall out\_com

ldi temp,' '

rcall out\_com

ldi temp,'t'

rcall out\_com

ldi temp,'i'

rcall out\_com

ldi temp,'m'

rcall out\_com

ldi temp,'e'

rcall out\_com

ldi temp,0x0A ;"ПЕРЕВОД СТРОКИ" Перевод курсора на строку ниже

rcall out\_com

ldi temp,0x0D ;"ВОЗВРАТ КАРЕТКИ" Переход на начало текущей строки

rcall out\_com

ret

STF:

sbic PINC, 3

rjmp STF

ldi temp2, 0

stf\_cicle:

ldi temp, (1<<4)

out PORTC, temp

sbic PINC, 0

rcall INP\_SEVEN

sbic PINC, 1

rcall INP\_FOUR

sbic PINC, 2

rcall INP\_ONE

sbic PINC, 3

rcall INP\_ZERO

ldi temp, (1<<5)

out PORTC, temp

sbic PINC, 0

rcall INP\_EIGHT

sbic PINC, 1

rcall INP\_FIVE

sbic PINC, 2

rcall INP\_TWO

ldi temp, (1<<6)

out PORTC, temp

sbic PINC, 0

rcall INP\_NINE

sbic PINC, 1

rcall INP\_SIX

sbic PINC, 2

rcall INP\_THREE

rcall DELAY

cpi temp2, 6

brlo stf\_cicle

rcall OUT\_INP\_TIME

ret

INP\_NINE:

sbic PINC, 0

rjmp INP\_NINE

cpi temp2, 0

breq end\_inp\_nine

cpi temp2, 2

breq end\_inp\_nine

cpi temp2, 4

breq end\_inp\_nine

ldi temp, 9

rcall INP\_PUSH

inc temp2

end\_inp\_nine:

ret

INP\_EIGHT:

sbic PINC, 0

rjmp INP\_EIGHT

cpi temp2, 0

breq end\_inp\_eight

cpi temp2, 2

breq end\_inp\_eight

cpi temp2, 4

breq end\_inp\_eight

ldi temp, 8

rcall INP\_PUSH

inc temp2

end\_inp\_eight:

ret

INP\_SEVEN:

sbic PINC, 0

rjmp INP\_SEVEN

cpi temp2, 0

breq end\_inp\_seven

cpi temp2, 2

breq end\_inp\_seven

cpi temp2, 4

breq end\_inp\_seven

ldi temp, 7

rcall INP\_PUSH

inc temp2

end\_inp\_seven:

ret

INP\_SIX:

sbic PINC, 1

rjmp INP\_SIX

cpi temp2, 0

breq end\_inp\_six

cpi temp2, 2

breq end\_inp\_six

cpi temp2, 4

breq end\_inp\_six

ldi temp, 6

rcall INP\_PUSH

inc temp2

end\_inp\_six:

ret

INP\_FIVE:

sbic PINC, 1

rjmp INP\_FIVE

cpi temp2, 0

breq end\_inp\_five

ldi temp, 5

rcall INP\_PUSH

inc temp2

end\_inp\_five:

ret

INP\_FOUR:

sbic PINC, 1

rjmp INP\_FOUR

cpi temp2, 0

breq end\_inp\_foure

ldi temp, 4

rcall INP\_PUSH

inc temp2

end\_inp\_foure:

ret

INP\_THREE:

sbic PINC, 2

rjmp INP\_THREE

cpi temp2, 0

breq end\_inp\_three

ldi temp, 3

rcall INP\_PUSH

inc temp2

end\_inp\_three:

ret

INP\_TWO:

sbic PINC, 2

rjmp INP\_TWO

ldi temp, 2

rcall INP\_PUSH

inc temp2

end\_inp\_two:

ret

INP\_ONE:

sbic PINC, 2

rjmp INP\_ONE

ldi temp, 1

rcall INP\_PUSH

inc temp2

end\_inp\_one:

ret

INP\_ZERO:

sbic PINC, 3

rjmp INP\_ZERO

ldi temp, 0

rcall INP\_PUSH

inc temp2

end\_inp\_zero:

ret

INP\_PUSH:

cpi temp2, 0

brne next\_inp1

mov inp\_hours\_h, temp

rjmp inp\_push\_end

next\_inp1:

cpi temp2, 1

brne next\_inp2

mov inp\_hours\_l, temp

rjmp inp\_push\_end

next\_inp2:

cpi temp2, 2

brne next\_inp3

mov inp\_minutes\_h, temp

rjmp inp\_push\_end

next\_inp3:

cpi temp2, 3

brne next\_inp4

mov inp\_minutes\_l, temp

rjmp inp\_push\_end

next\_inp4:

cpi temp2, 4

brne next\_inp5

mov inp\_seconds\_h, temp

rjmp inp\_push\_end

next\_inp5:

cpi temp2, 5

brne inp\_push\_end

mov inp\_seconds\_l, temp

rjmp inp\_push\_end

inp\_push\_end:

ret

OUT\_INP\_TIME:

clr time\_seconds

clr time\_minutes

clr time\_hours

ldi temp2, 10

mov temp, inp\_seconds\_l

add time\_seconds, temp

mov temp, inp\_seconds\_h

cpi temp, 0

cicle\_out\_inp\_seconds:

breq next\_out\_inp\_minutes

add time\_seconds, temp2

dec temp

rjmp cicle\_out\_inp\_seconds

next\_out\_inp\_minutes:

mov temp, inp\_minutes\_l

add time\_minutes, temp

mov temp, inp\_minutes\_h

cpi temp, 0

cicle\_out\_inp\_minutes:

breq next\_out\_inp\_hours

add time\_minutes, temp2

dec temp

rjmp cicle\_out\_inp\_minutes

next\_out\_inp\_hours:

mov temp, inp\_hours\_l

add time\_hours, temp

mov temp, inp\_hours\_h

cpi temp, 0

cicle\_out\_inp\_hours:

breq end\_out\_inp

add time\_hours, temp2

dec temp

rjmp cicle\_out\_inp\_hours

end\_out\_inp:

ret

;#### ПРОЦЕДУРА ОБНОВЛЕНИЯ СТАТУСА УСТРОЙСТВ ####

out\_schedule:

cli ;Временно запрещаем прерывания

ldi YL, low(schedule\_start)

ldi YH, high(schedule\_start)

ldi counter, 0

in temp, PORTA

mov a\_status, temp

next\_time:

ldi XL, low(schedule\_count)

ldi XH, high(schedule\_count)

ld temp, X

cp counter, temp

breq end\_out\_schedule

brsh end\_out\_schedule

inc counter

ldi on\_off, 0 ;По умолчанию устройство нужно выключить

ld temp, Y+

sbrc temp, 7 ;Если необходимо включить устройство

ldi on\_off, 1 ;то устанавливаем соотв. значение в on\_off

andi temp, 0b00001111 ;Определяем номер устройства

mov device\_number, temp ;которое включаем/выключаем

ld temp, Y+

cp temp, time\_hours ;Сравнение по часам

breq next\_minutes ;Если часы равны, то проверяем минуты

brsh skip\_MS ;Если temp больше hours то

;прпускаем минуты и секунды и не обновляем статус

inc YL ;Увеличиваем значеие Y на 2, чтобы

inc YL ;указатель стоял на следующей записи

rjmp execute\_device\_status

next\_minutes:

ld temp, Y+

cp temp, time\_minutes ;Сравнение по минутам

breq next\_seconds ;Если минуты равны, то проверяем минуты

brsh skip\_S ;Если temp больше minutes то

;прпускаем секунды и не обновляем статус

inc YL ;Увеличиваем значеие Y на 1

rjmp execute\_device\_status

next\_seconds:

ld temp, Y+

cp temp, time\_seconds ;Сравнение по секундам

breq execute\_device\_status

brsh next\_time ;Если temp больше seconds, то

;то переходим к следующей записи и не обновляем статус

rjmp execute\_device\_status ;иначе выводим статус устройства

skip\_time:

inc YL

` inc YL

inc YL

rjmp next\_time

skip\_MS:

inc YL

inc YL

rjmp next\_time

skip\_S:

inc YL

rjmp next\_time

execute\_device\_status:

mov temp, device\_number

rcall set\_bit\_temp2 ;выставляем номер бита устройства,

or actual\_device\_statuses, temp2 ;статус которого актуализируем

and temp2, force\_devices ;Если выбранное устройство находится в принудительном режиме

cpi temp2, 0 ;то пропускаем это устройство и переходим к

brne next\_time ;следующему сообщению

cpi on\_off, 1 ;Проверяем нужно ли включить устройстов

brne SET\_OFF ;Если не равно, то идем выключать

mov temp, device\_number ;Заносим в temp номер текущего устройства

rcall set\_bit\_temp2 ;Устанавливаем нужный бит в temp2

mov temp, a\_status ;Заносим в temp актуальное состояние порта A

or temp, temp2 ;Устанавливаем 1 в нужный бит

mov a\_status, temp ;Заносим значение temp обратно в порт A

rjmp next\_time

SET\_OFF:

rcall set\_bit\_temp2 ;Устанавливаем нужный бит в temp2

mov temp, a\_status ;Заносим в temp актуальное состояние порта A

com temp2 ;Инвертируем значения в регистре temp2

and temp, temp2 ;Устанавливаем 0 в нужный бит

mov a\_status, temp ;Заносим значение temp обратно в порт A

rjmp next\_time

end\_out\_schedule:

mov temp, a\_status

out PORTA, temp

sei ;Вновь разрешаем прерывания

ret

;#### ПРОЦЕДУРА УСТАНОВКИ БИТА В ПЕРЕМЕННОЙ temp2 ####

set\_bit\_temp2:

ldi temp2, 1

mov counter2, temp2

cp counter2, temp

breq set\_bit\_end

set\_bit\_cicle:

lsl temp2

inc counter2

cp counter2, temp

brne set\_bit\_cicle

set\_bit\_end:

ret

;#### ПРОЦЕДУРА ОТПРАВКИ СООБЩЕНИЯ OK ####

ok\_msg:

ldi temp,'O'

rcall out\_com

ldi temp,'K'

rcall out\_com

ldi temp,0x0A ;"ПЕРЕВОД СТРОКИ" Перевод курсора на строку ниже

rcall out\_com

ldi temp,0x0D ;"ВОЗВРАТ КАРЕТКИ" Переход на начало текущей строки

rcall out\_com

ret

;#### ПРОЦЕДУРА ПРИЕМА РАСПИСАНИЯ ####

recieve\_schedule:

ldi XL, low(schedule\_count)

ldi XH, high(schedule\_count)

ldi YL, low(schedule\_start)

ldi YH, high(schedule\_start)

ldi temp, 0 ;Устанавливаем количество записей

st X, temp ;в нуль

recieve\_cicle:

rcall in\_com ;Считываем данные

cpi temp, 0b11111111 ;Признак окончания

breq end\_recieve ;передачи расписания

st Y+, temp ;Считываем заголовок, определяющий

rcall in\_com ;номер устройства и включение/отключение его

st Y+, temp ;Считываем часы начала работы

rcall in\_com

st Y+, temp ;Считываем минуты начала работы

rcall in\_com

st Y+, temp ;Считываем секунды начала работы

ldi XL, low(schedule\_count)

ldi XH, high(schedule\_count)

ld temp, X ;Вытаскиваем количество записей на данный момент

inc temp ;Увеличиваем количество записей

st X, temp ;Записываем кол-во записей по адресу X

rjmp recieve\_cicle

end\_recieve:

ret

;#### ПРОЦЕДУРА ОТПРАВКА БАЙТА ЧЕРЕЗ UART ####

out\_com:

sbis UCSRA,UDRE ;Ожидание, когда бит UDRE

rjmp out\_com ;будет установлен в 1 (предыдущий байт отправлен)

out UDR,temp ;Отправляем байт

ret

;#### ПРОЦЕДУРА ПРИЕМ БАЙТА ЧЕРЕЗ UART ####

in\_com:

sbis UCSRA,RXC ;Ожидание, когда бит RXC будет установлен в 1

rjmp in\_com ;(в регистре данных есть принятый непрочитанный байт)

in temp,UDR ;Считываем принятый байт

ret

;#### РАЗЛИЧНЫЕ ПРЕРЫВАНИЯ ####

TIME1\_OVER:

push temp ; Сохранение регистра temp

push temp2

; Установка начального значения счетчика T1

ldi temp, 0b10000101

out TCNT1H, temp

ldi temp, 0b11101110

out TCNT1L, temp

inc time\_seconds

cpi time\_seconds, 60

brne time1\_continue

ldi time\_seconds, 0

inc time\_minutes

cpi time\_minutes, 60

brne time1\_continue

ldi time\_minutes, 0

inc time\_hours

cpi time\_hours, 24

brne time1\_continue

ldi time\_hours, 0

time1\_continue:

pop temp2

pop temp

reti

TIME0\_OVER:

push R16

push R17

push R18

push R19

push R25

; Очистка дисплея

rcall SET\_COMMAND\_MODE

ldi temp,0b00000001

mov byte\_to\_send, temp

rcall SEND\_BYTE

; Сдвигание курсора вправо на одну позицию

rcall SET\_COMMAND\_MODE

ldi temp,0b00010100

mov byte\_to\_send, temp

rcall SEND\_BYTE

; Устанавливаем режим передачи символов

rcall SET\_DATA\_MODE

rcall LOW\_DELAY

ldi temp, 'T'

mov byte\_to\_send, temp

rcall SEND\_BYTE

ldi temp, 'I'

mov byte\_to\_send, temp

rcall SEND\_BYTE

ldi temp, 'M'

mov byte\_to\_send, temp

rcall SEND\_BYTE

ldi temp, 'E'

mov byte\_to\_send, temp

rcall SEND\_BYTE

ldi temp, ':'

mov byte\_to\_send, temp

rcall SEND\_BYTE

; Сдвигание курсора вправо на одну позицию

rcall SET\_COMMAND\_MODE

ldi temp,0b00010100

mov byte\_to\_send, temp

rcall SEND\_BYTE

; Заносим код начала цифр в ascii

ldi temp, 0x30

mov ascii\_numbers\_start, temp

; Заносим в регистр temp текущее время в часах

clr temp2

mov temp, time\_hours

; Устанавливаем режим передачи символов

rcall SET\_DATA\_MODE

rcall LOW\_DELAY

hours\_cicle: ; Считаем часы

cpi temp, 10.

brlo out\_hours

inc temp2

subi temp, 10

rjmp hours\_cicle

out\_hours: ; Выводим часы

; Выводим старшую цифру

add temp2, ascii\_numbers\_start

mov byte\_to\_send, temp2

rcall SEND\_BYTE

; Выводим младшую цифру

` add temp, ascii\_numbers\_start

mov byte\_to\_send, temp

rcall SEND\_BYTE

ldi temp, ':'

mov byte\_to\_send, temp

rcall SEND\_BYTE

; Заносим в регистр temp текущее время в часах

clr temp2

mov temp, time\_minutes

minutes\_cicle:

cpi temp, 10.

brlo out\_minutes

inc temp2

subi temp, 10

rjmp minutes\_cicle

out\_minutes: ; Выводим минуты

; Выводим старшую цифру

add temp2, ascii\_numbers\_start

mov byte\_to\_send, temp2

rcall SEND\_BYTE

; Выводим младшую цифру

` add temp, ascii\_numbers\_start

mov byte\_to\_send, temp

rcall SEND\_BYTE

ldi temp, ':'

mov byte\_to\_send, temp

rcall SEND\_BYTE

; Заносим в регистр temp текущее время в часах

clr temp2

mov temp, time\_seconds

seconds\_cicle:

cpi temp, 10.

brlo out\_seconds

inc temp2

subi temp, 10

rjmp seconds\_cicle

out\_seconds:

; Выводим старшую цифру

add temp2, ascii\_numbers\_start

mov byte\_to\_send, temp2

rcall SEND\_BYTE

; Выводим младшую цифру

` add temp, ascii\_numbers\_start

mov byte\_to\_send, temp

rcall SEND\_BYTE

; Перемещаем курсор в левый нижний угол

rcall SET\_COMMAND\_MODE

ldi temp,0b11000000 ; 0xC0 - адрес левого нижнего угла дисплея

mov byte\_to\_send, temp

rcall SEND\_BYTE

; Устанавливаем режим передачи символов

rcall SET\_DATA\_MODE

ldi temp, 'D'

mov byte\_to\_send, temp

rcall SEND\_BYTE

ldi temp, 'E'

mov byte\_to\_send, temp

rcall SEND\_BYTE

ldi temp, 'V'

mov byte\_to\_send, temp

rcall SEND\_BYTE

ldi temp, 'I'

mov byte\_to\_send, temp

rcall SEND\_BYTE

ldi temp, 'C'

mov byte\_to\_send, temp

rcall SEND\_BYTE

ldi temp, 'E'

mov byte\_to\_send, temp

rcall SEND\_BYTE

ldi temp, 'S'

mov byte\_to\_send, temp

rcall SEND\_BYTE

ldi temp, ':'

mov byte\_to\_send, temp

rcall SEND\_BYTE

; Устанавливаем режим передачи символов

rcall SET\_DATA\_MODE

sbic PINA, 0

rjmp dev0\_is\_1

ldi temp, '0'

rjmp dev0\_out

dev0\_is\_1:

ldi temp, '1'

dev0\_out:

mov byte\_to\_send, temp

rcall SEND\_BYTE

sbic PINA, 1

rjmp dev1\_is\_1

ldi temp, '0'

rjmp dev1\_out

dev1\_is\_1:

ldi temp, '1'

dev1\_out:

mov byte\_to\_send, temp

rcall SEND\_BYTE

sbic PINA, 2

rjmp dev2\_is\_1

ldi temp, '0'

rjmp dev2\_out

dev2\_is\_1:

ldi temp, '1'

dev2\_out:

mov byte\_to\_send, temp

rcall SEND\_BYTE

sbic PINA, 3

rjmp dev3\_is\_1

ldi temp, '0'

rjmp dev3\_out

dev3\_is\_1:

ldi temp, '1'

dev3\_out:

mov byte\_to\_send, temp

rcall SEND\_BYTE

sbic PINA, 4

rjmp dev4\_is\_1

ldi temp, '0'

rjmp dev4\_out

dev4\_is\_1:

ldi temp, '1'

dev4\_out:

mov byte\_to\_send, temp

rcall SEND\_BYTE

sbic PINA, 5

rjmp dev5\_is\_1

ldi temp, '0'

rjmp dev5\_out

dev5\_is\_1:

ldi temp, '1'

dev5\_out:

mov byte\_to\_send, temp

rcall SEND\_BYTE

sbic PINA, 6

rjmp dev6\_is\_1

ldi temp, '0'

rjmp dev6\_out

dev6\_is\_1:

ldi temp, '1'

dev6\_out:

mov byte\_to\_send, temp

rcall SEND\_BYTE

sbic PINA, 7

rjmp dev7\_is\_1

ldi temp, '0'

rjmp dev7\_out

dev7\_is\_1:

ldi temp, '1'

dev7\_out:

mov byte\_to\_send, temp

rcall SEND\_BYTE

pop R25

pop R19

pop R18

pop R17

pop R16

reti

Приложение Б – Спецификация радиоэлементов схемы

**Листов 2**