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)

Москва, 2019

Оглавление

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 пульт 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.

1 Конструкторская часть

1.1 Проектирование МК-системы

1.1.1 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:

1) микроконтроллер;

2) блок обмена информацией с ПЭВМ;

3) пуль оператора;

4) блок реле для управления питанием приборов

5) LCD-дисплей

Обобщенная структура проектируемого устройства представлена на рисунке 1.

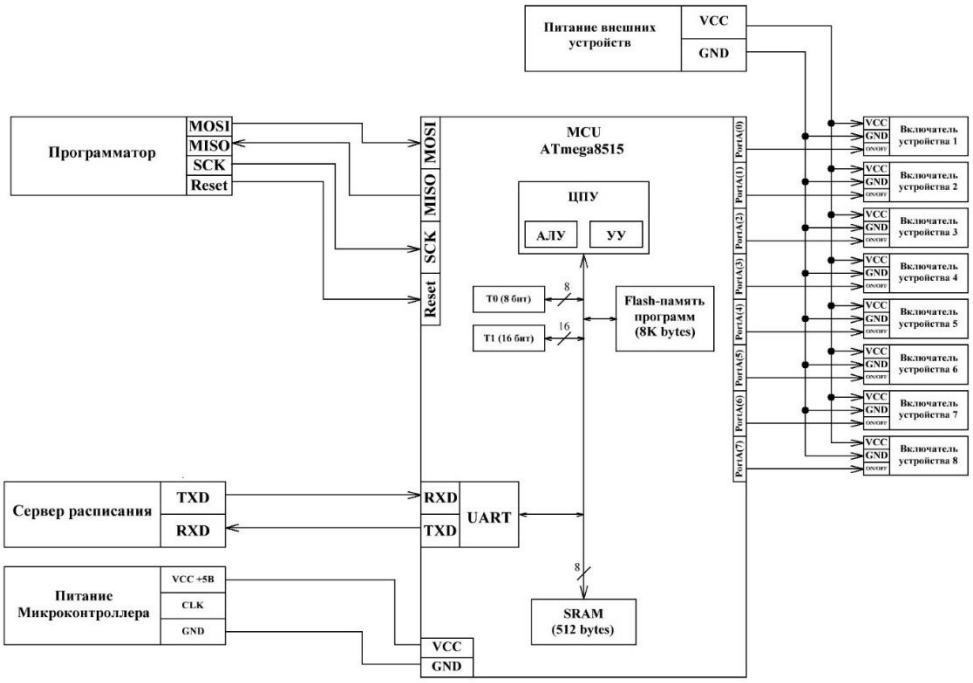

Рисунок 1 – Структурная схема устройства управления приборами жилого помещения

1.1.2 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 1 – 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.

1.1.3 Описание архитектуры и технических характеристик микроконтроллера

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

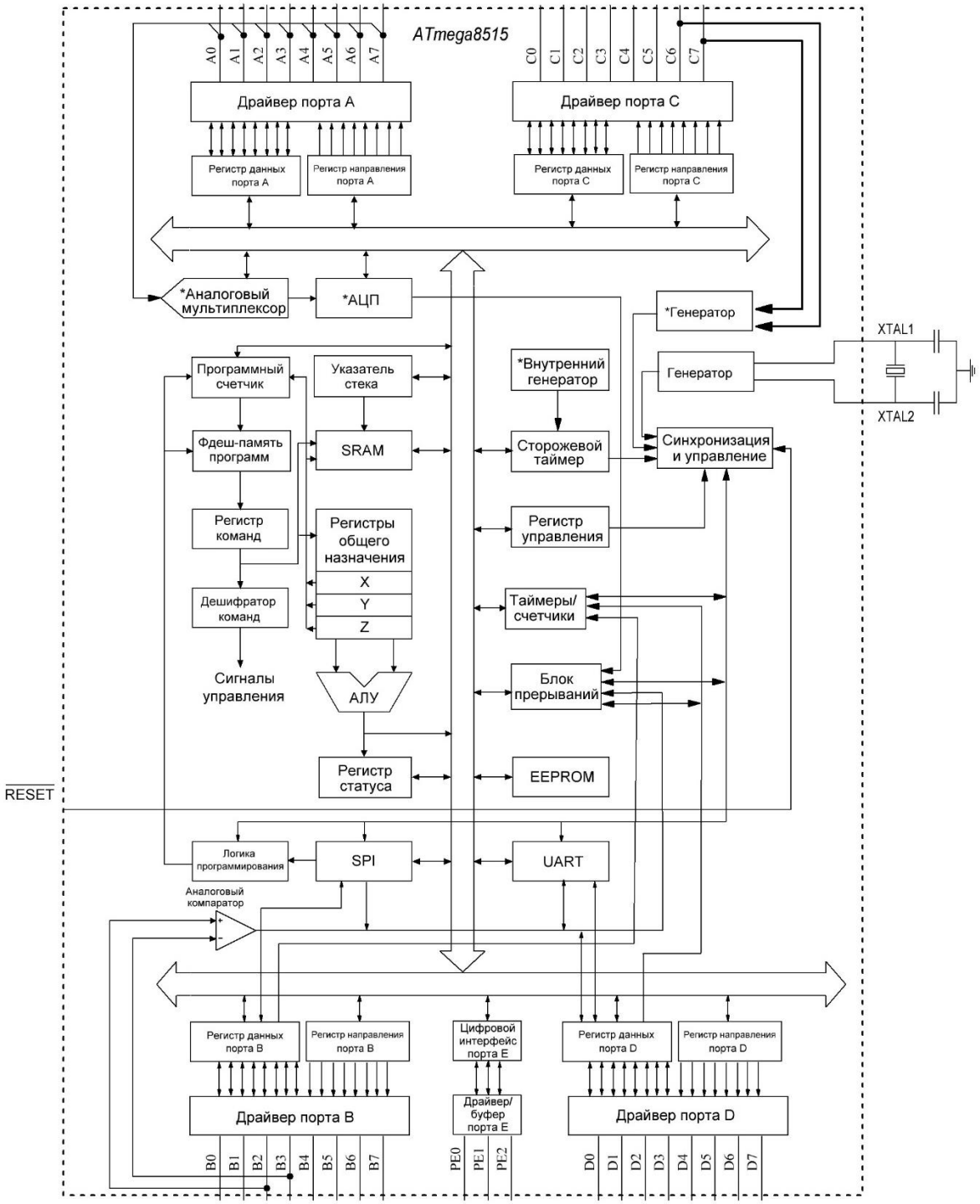

Рисунок 2 – Функциональная схема микроконтроллера ATmega8515

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.

1.1.4 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.

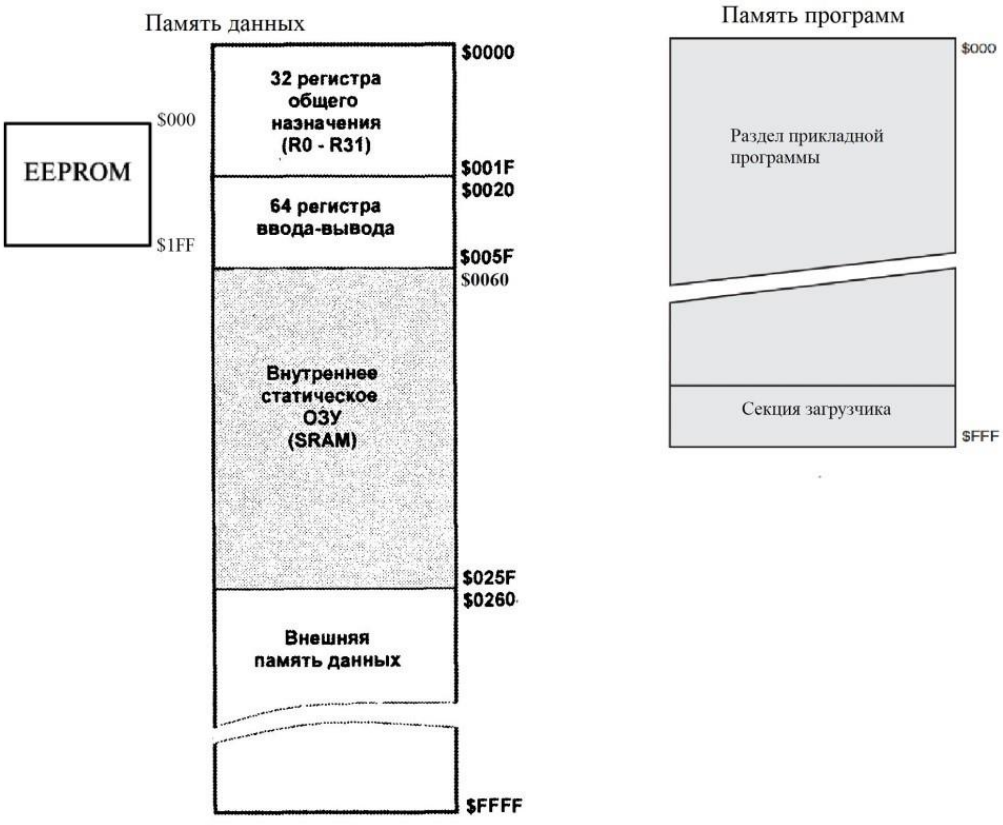

Рисунок 3 – Адресные пространства МК ATmega8515

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.

1.1.5 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.

1.2 Разработка функциональной схемы

1.2.1 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 A 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.

1.2.2 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 4 – 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 2 – 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:

1) возникла необходимость получения более актуального расписания;

2) оператор нажимает кнопку «ЗСР»;

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

4) оператор нажимает клавишу «ВКЛП»;

5) оператор выбирает первое устройство нажатием клавиши «1»;

6) устройство выключается;

7) оператор снова запрашивает расписание клавишей «ЗСР»;

8) от сервера приходит новое расписание;

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

10) приборы продолжают работать по новому расписанию.

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

1.2.3 Блок передачи данных к ПЭВМ

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

Сигнал TxD с 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.

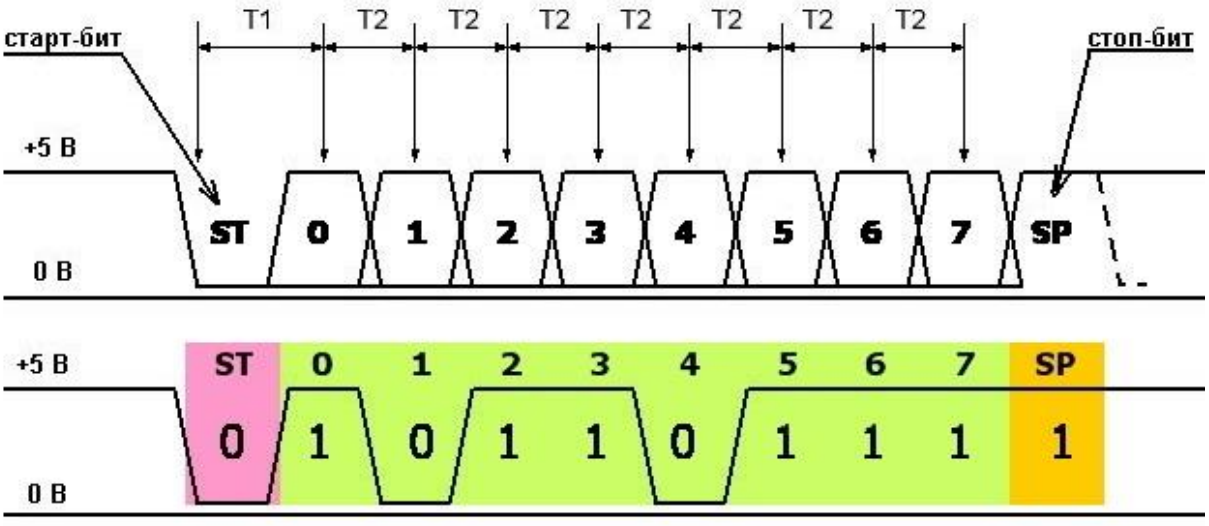

Рисунок 5 – Кадр UART

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.

1.2.4 Блок реле

Для управление блоком реле используется порт А. Каждый вывод порта А 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.

1.2.5 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.

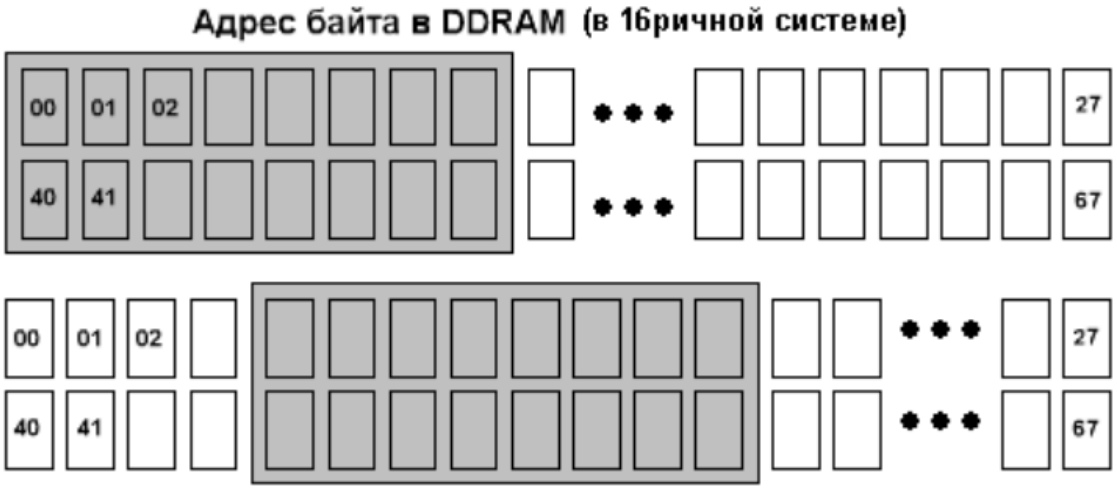

Рисунок 6 – Окно отображения дисплея LM016L

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 3 – Команды управления дисплеем LM016L

DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Значение
0	**0**	**0**	**0**	**0**	**0**	**0**	**1**	Очистка экрана. Счетчик адреса на 0 позицию DDRAM
0	**0**	**0**	**0**	**0**	**0**	**1**	—	Адресация на DDRAM сброс сдвигов, Счетчик адреса на 0
0	**0**	**0**	**0**	**0**	**1**	I/D	S	Настройка сдвига экрана и курсора
0	**0**	**0**	**0**	**1**	D	C	B	Настройка режима отображения
0	**0**	**0**	**1**	S/C	R/L	—	—	Сдвиг курсора или экрана, в зависимости от битов
0	**0**	**1**	DL	N	F	—	—	Выбор числа линий, ширины шины и размера символа
0	**1**	AG	AG	AG	AG	AG	AG	Переключить адресацию на SGRAM и задать адрес в SGRAM
1	AD	AD	AD	AD	AD	AD	AD	Переключить адресацию на DDRAM и задать адрес в DDRAM

Значения каждых битов следующие:

1) 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;

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

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

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

5) B — сделать курсор в виде мигающего черного квадрата;

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

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

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

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

10) F — размер символа 0 — 5x8 точек. 1 — 5x10 точек;

11) AG — адрес в памяти CGRAM;

12) AD — адрес в памяти DDRAM.

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

1.3 Разработка принципиальной схемы

1.3.1 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:

1) Микроконтроллер ATmega8515;

2) Блок реле для управления питанием приборов;

3) Матричная клавиатура 4x4;

4) LCD-дисплей для индикации текущего времени МК-системы;

5) Преобразователь входного напряжения.

1.3.2 Конфигурация выводов микроконтроллера

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

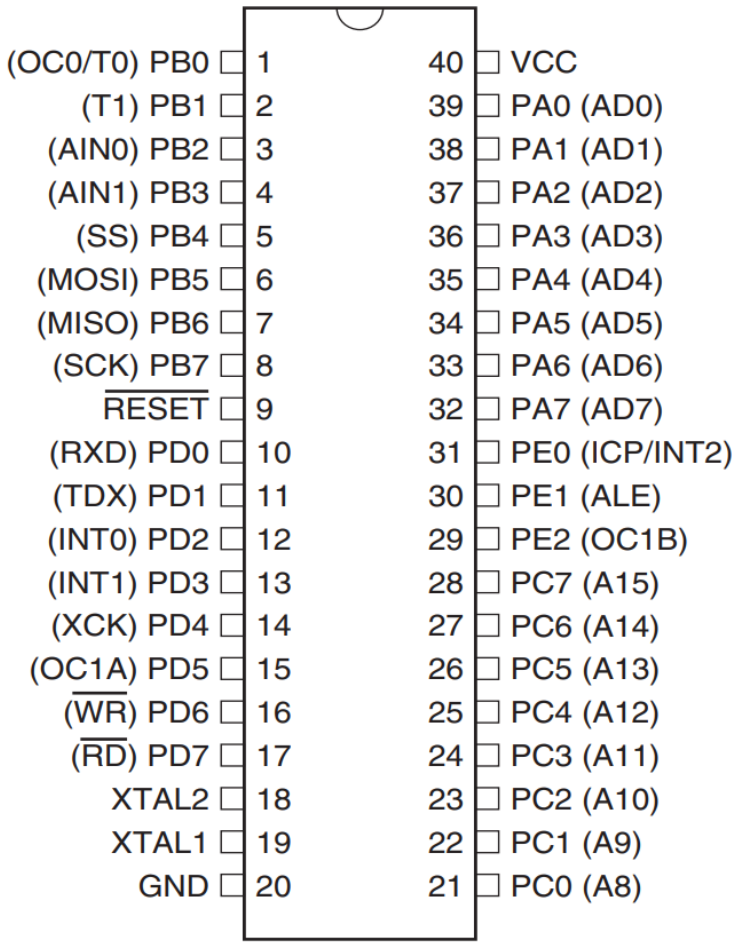

Рисунок 7 – Микроконтроллер ATmega8515 в корпусе PDIP

1.3.3 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 КР142ЕН5А. Для нормальной работы данный стабилизатор шунтируется конденсаторами на входе и выходе. Схема подключения представлена на рисунке 8.

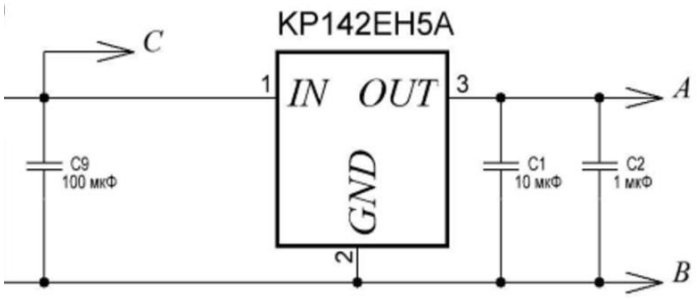

Рисунок 8 – Схема подключения стабилизатора напряжения КР142ЕН5А

В схеме выше подключение конденсатор С9 используется для сглаживания скачков напряжения питания на входе, а конденсаторы С1 и С2 используются в качестве фильтров, сглаживая подаваемое напряжения на устройства.

1.3.4 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.

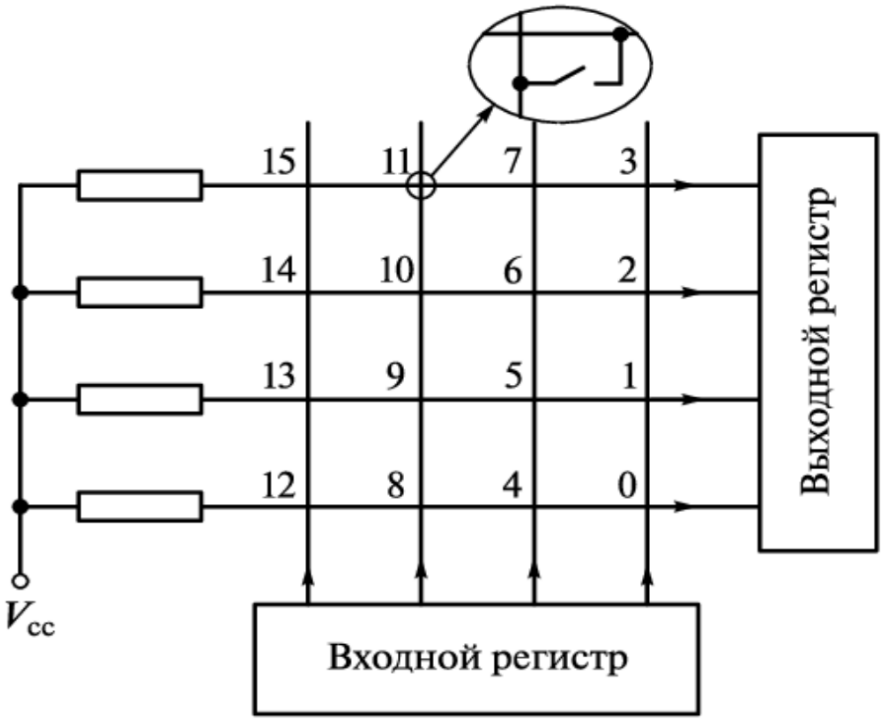

Рисунок 9 – Матричная клавиатура

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

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

Таким образом, все 16 кнопок обрабатываются с помощью 8 пинов порта С 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.

1.3.5 Подключение LCD-дисплея

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

1) 8-битный режим. В этом режиме данные передаются сразу одним байтом за один такт. При этом используются сразу 8 контактов;

2) 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.

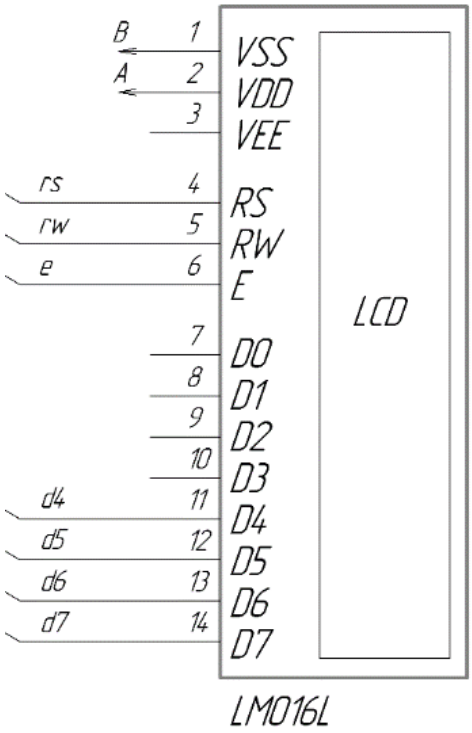

Рисунок 10 – Подключение LCD-дисплея LM016L

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.

1.4 Расчёт параметров

1.4.1 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 $\frac{8\,000\,000}{50} = 160\,000$ тактов. Число 160 000 можно разбить на 2 множителя, например $1024 \times 156 \approx 160\,000$, где 1024 – это делитель частоты СК, 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 $256 \times 31\,250 = 8\,000\,000$, где 256 – делитель частоты СК, 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).

1.4.2 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.

1.4.3 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.

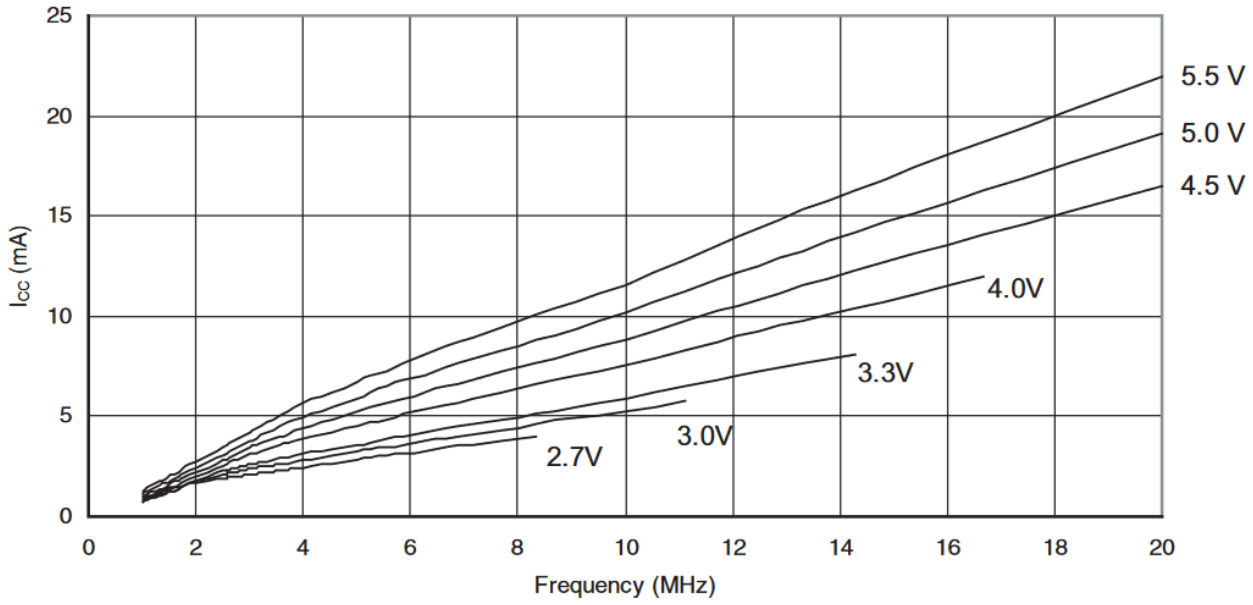

Рисунок 11 – График зависимости потребляемого тока для ATmega8515

По графику определяем ток $I_{cc}(8MHz, 5V) \approx 8$ 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 КР142ЕН5А 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: $P = I_{cc} * U_{пит}$.

Потребляемая мощность устройств представлена в таблице 4.

Таблица 4 – Потребляемая мощности компонентов МК-системы

Микросхема	P_{one}, мВт	Количество	P_{summ}, мВт
ATmega8515	40	1	40
КР142ЕН5А	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.

1.5 Описание алгоритмов функционирования устройства

1.5.1 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.

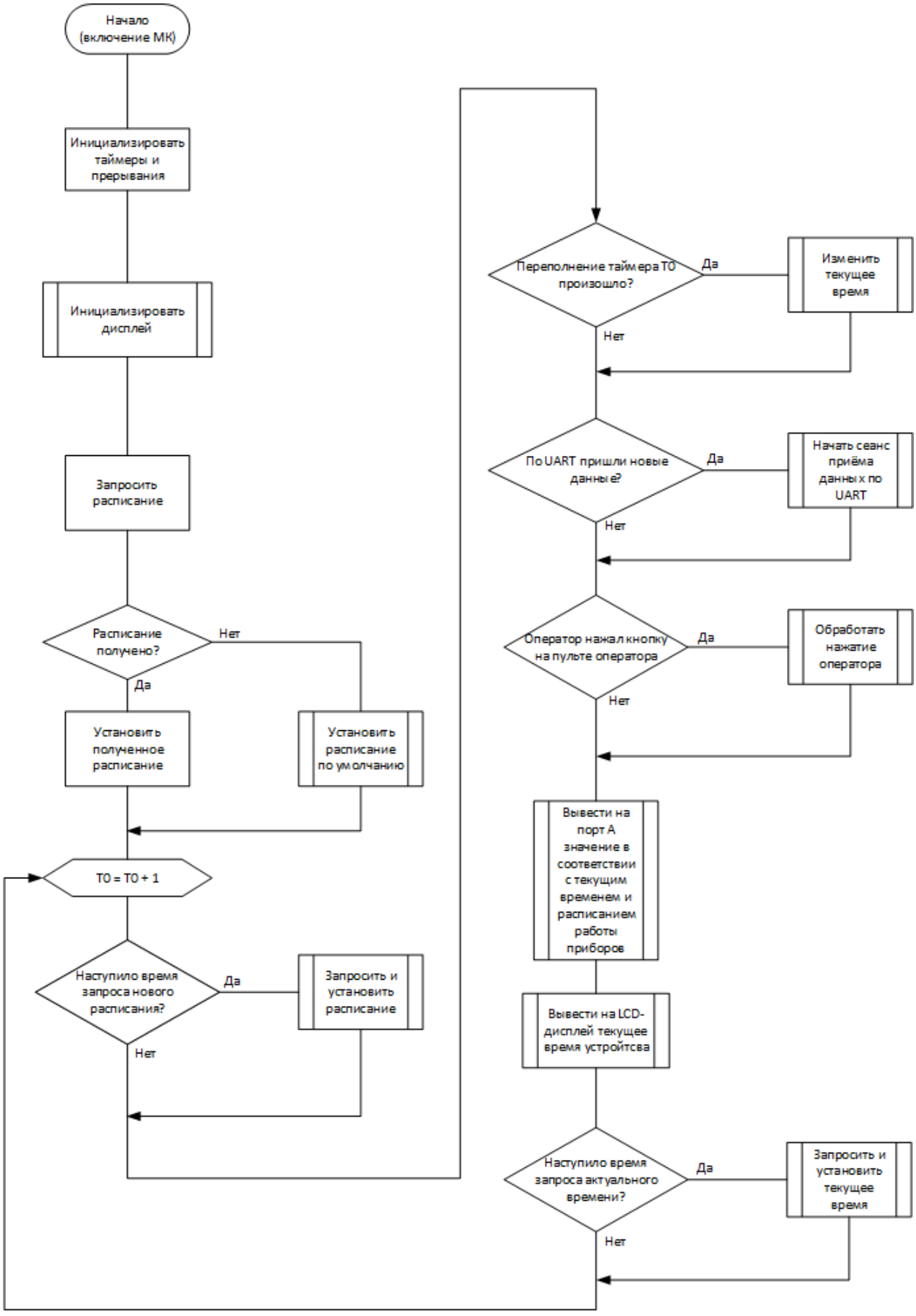

Рисунок 12 – Обобщенная схема-алгоритма работы программы

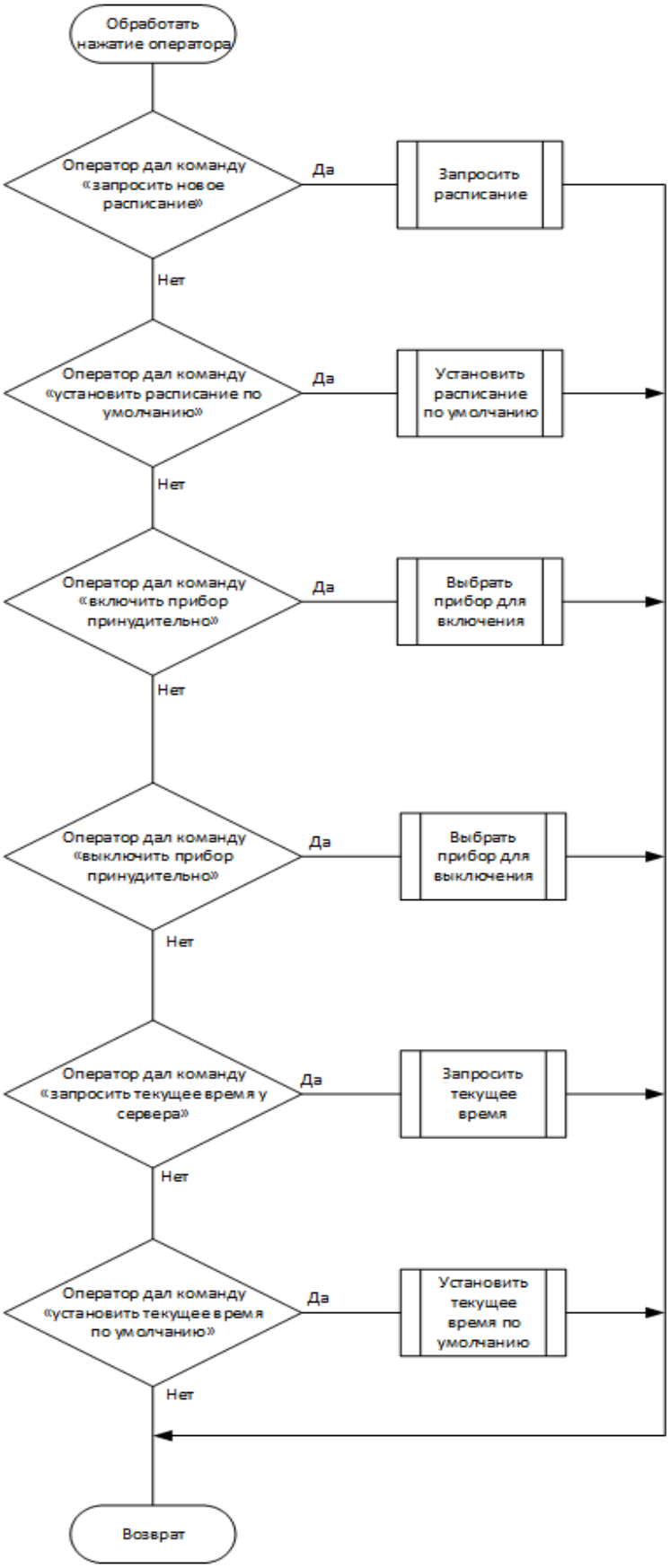

Рисунок 13 – Схема-алгоритм обработки нажатия клавиши на пульте оператора

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.

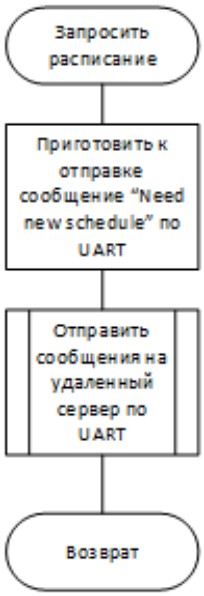

Рисунок 14 – Схема-алгоритм отправки запроса нового расписания

Рисунок 15 – 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 16 – 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.

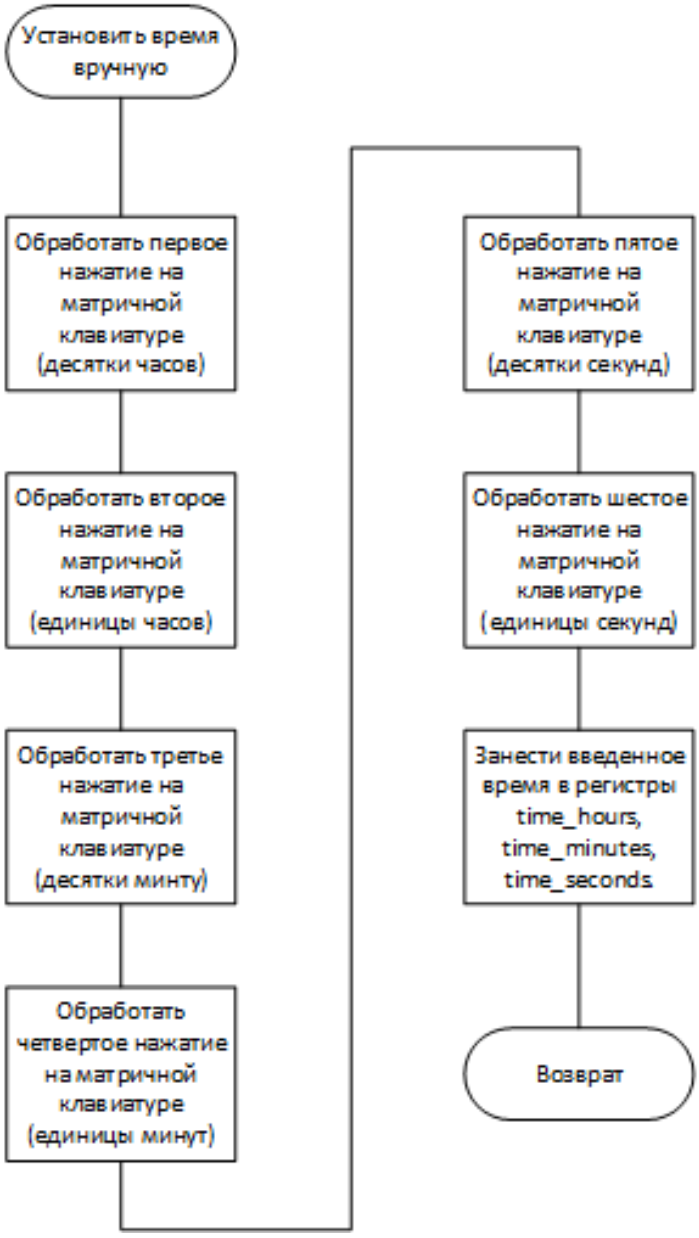

Рисунок 17 – Схема-алгоритм установки времени вручную

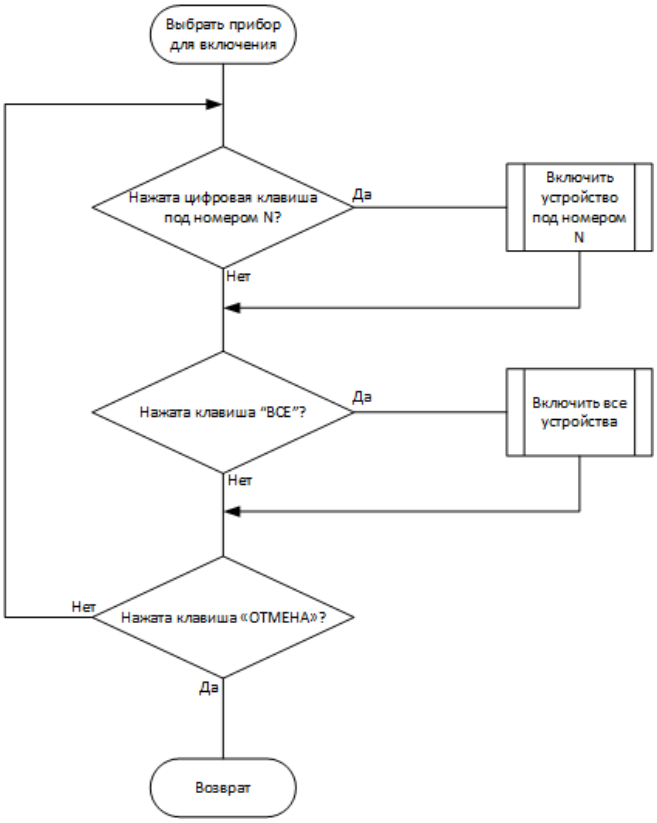

Рисунок 18 – Схема-алгоритм выбора прибора для включения

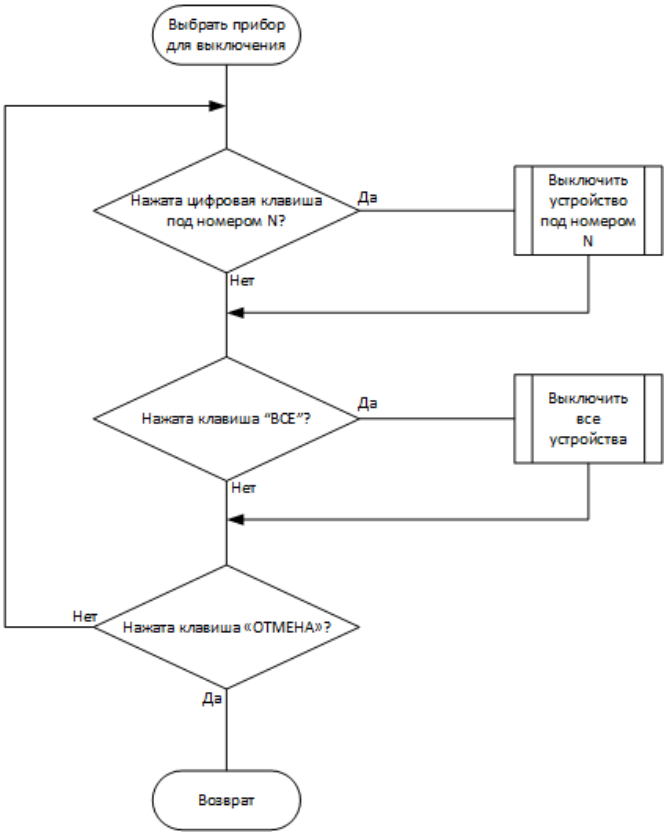

Рисунок 19 – Схема-алгоритм выбора прибора для выключения

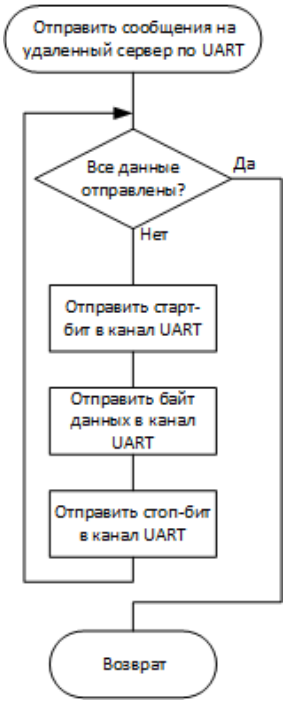

Рисунок 20 – Отправка сообщения по UART

Рисунок 21 – Прием сообщения по UART

1.5.1 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 22 – Формат передачи расписания

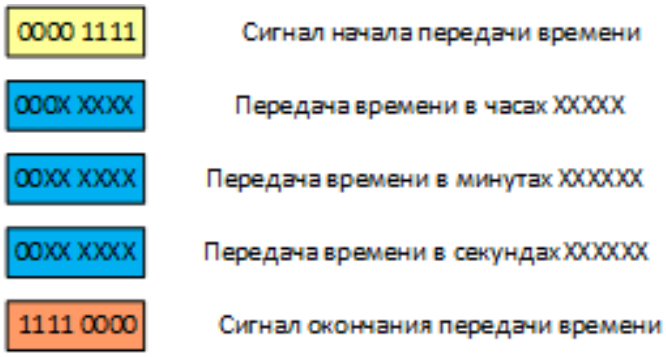

Рисунок 23 – Формат передачи времени

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.

2 Технологическая часть

2.1 Характеристика использованных систем разработки

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

1) AVR Studio 4 – для отладки программного кода на ассемблере;

2) 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.

2.2 Оценка количества задействованной памяти микроконтроллера ATmega8515

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

```
ATmega8515 memory use summary [bytes]:
Segment   Begin    End      Code   Data   Used    Size   Use%
----------------------------------------------------------------
[.cseg] 0x000000 0x000a38   2558      0   2558    8192  31.2%
[.dseg] 0x000060 0x000062      0      2      2     512   0.4%
[.eseg] 0x000000 0x000000      0      0      0     512   0.0%

Assembly complete, 0 errors. 3 warnings
```

Рисунок 24 – Количество занимаемой памяти программой

2.3 Симуляция в Proteus 8

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

Для симуляции работы МК 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.

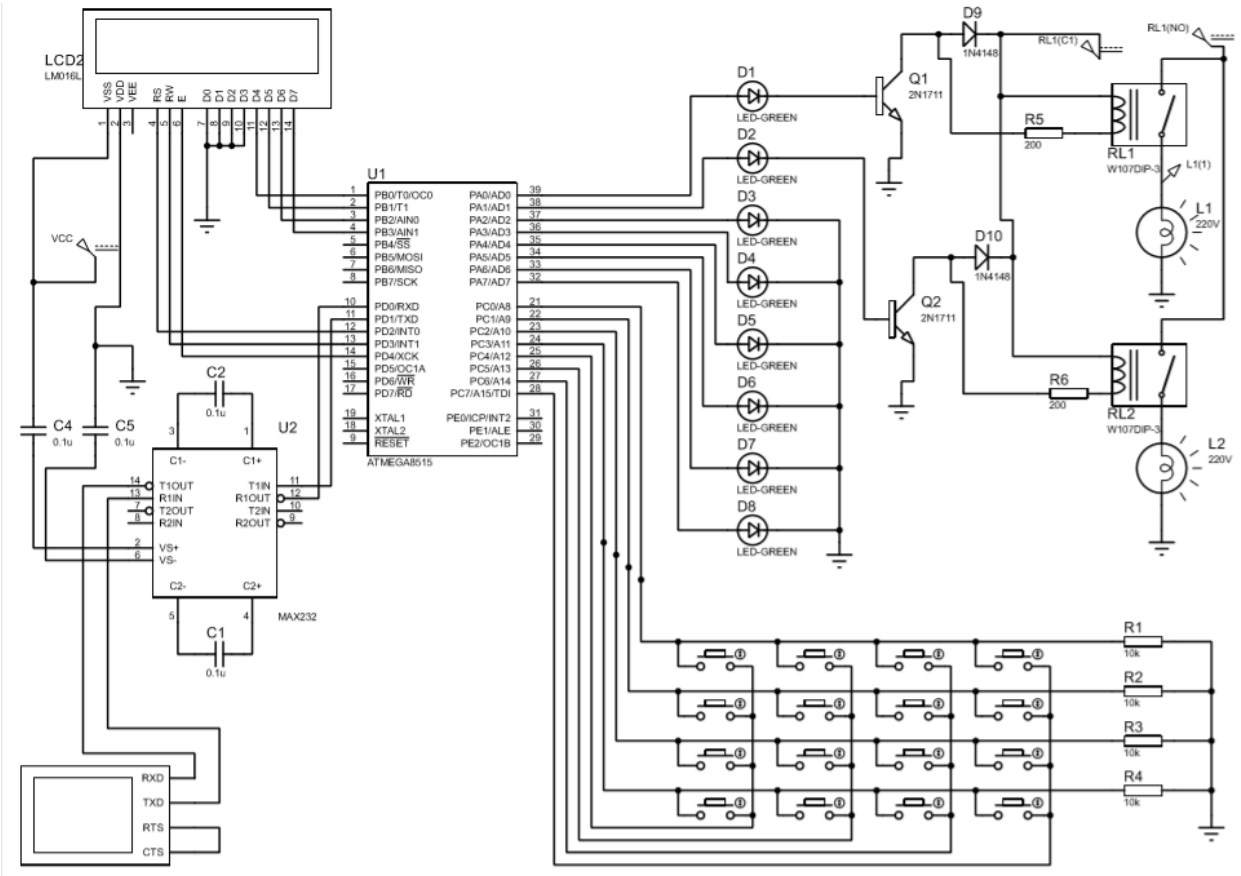

Рисунок 25 – Упрощенная схема разрабатываемой МК-системы

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.

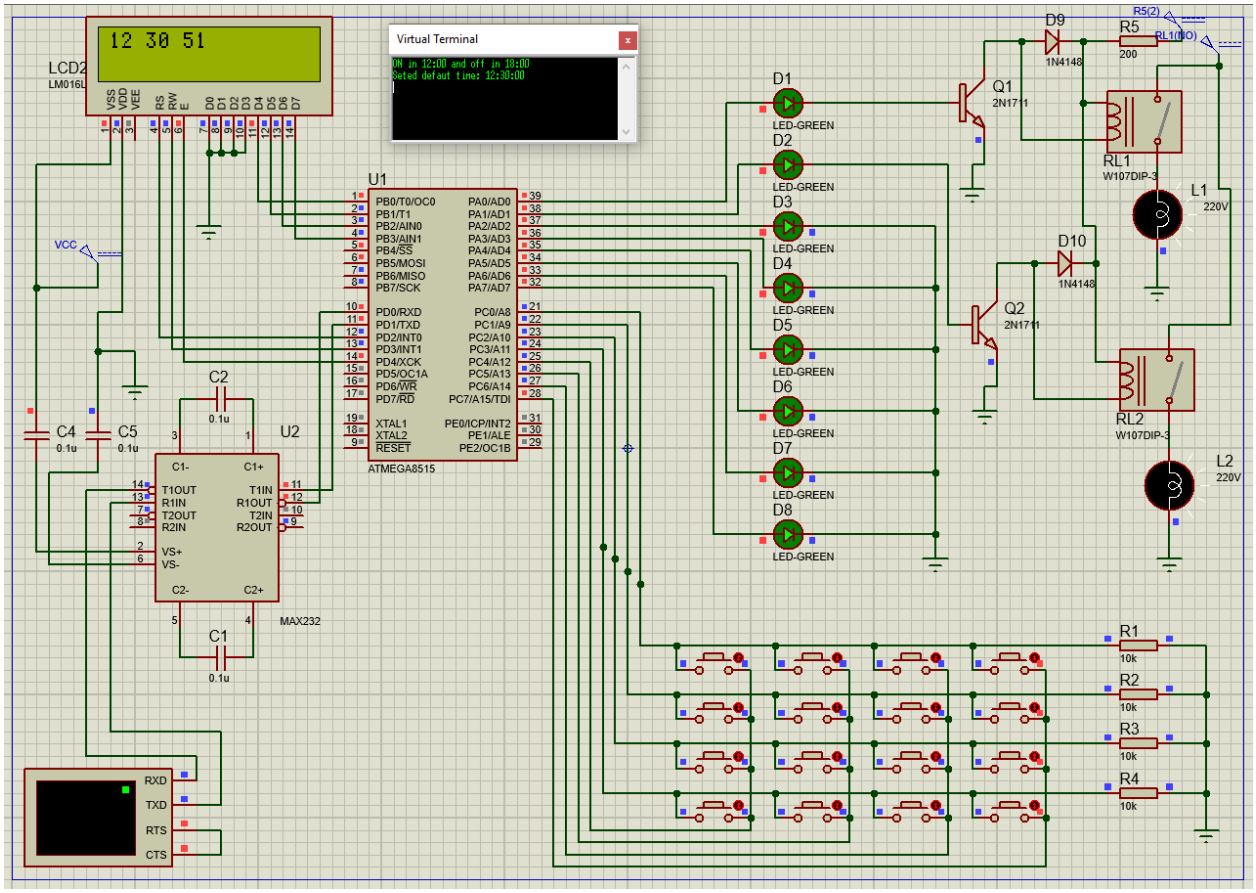

Рисунок 26 – Симуляция в Proteus

Матричная клавиатура представляет из себя систему из 16 кнопок, соединенных с портами порта С 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 А 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.

2.4 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):

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

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

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

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

5) GND — земля.

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

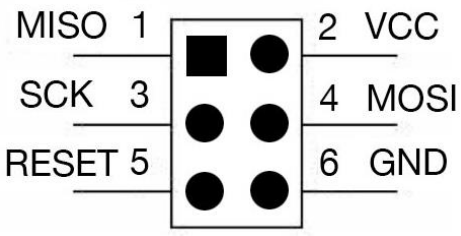

Рисунок 27 – Разъем Atmel 6-pin ISP

2.4.2 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.

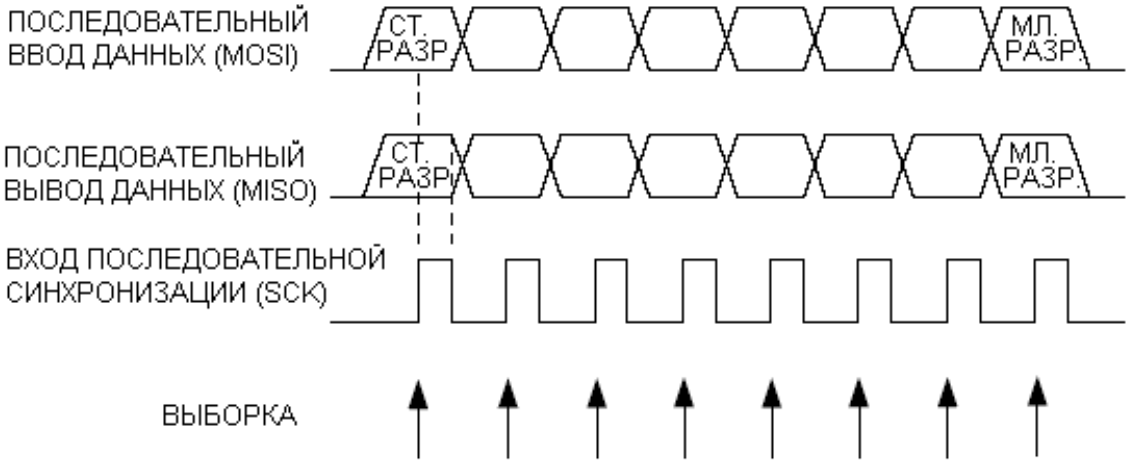

Рисунок 28 – Последовательное программирование по интерфейсу SPI

1) 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».

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

3) 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.

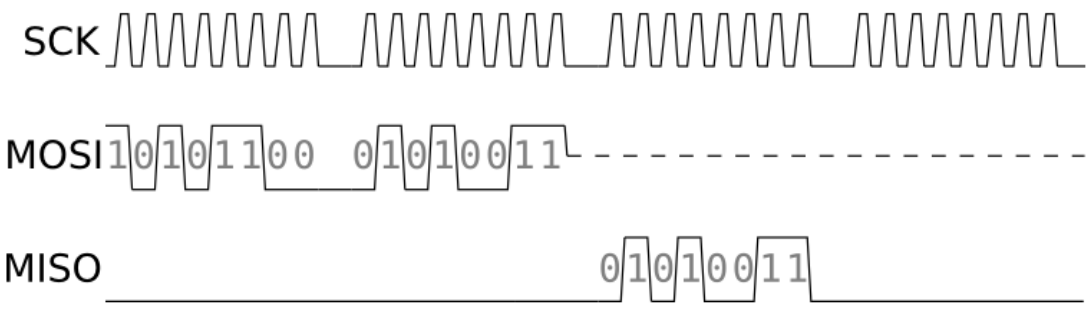

Рисунок 29 – Команда «Program Enable»

4) 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.

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

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

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

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

Таблица 5 – Формат байтов команд для программирования

Instruction	Instruction Format				Operation
	Byte 1	Byte 2	Byte 3	Byte4	
Programming Enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx	Enable Serial Programming after RESET goes low.
Chip Erase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Chip Erase EEPROM and Flash.
Read Program memory	0010 **H**000	0000 **aaaa**	**bbbb bbbb**	**oooo oooo**	Read **H** (high or low) data **o** from Program memory at word address **a**:**b**.
Load Program memory Page	0100 **H**000	0000 xxxx	xxx**b bbbb**	**iiii iiii**	Write **H** (high or low) data **i** to Program memory page at word address **b**. Data low byte must be loaded before Data high byte is applied within the same address.
Write Program memory Page	0100 1100	0000 **aaaa**	**bbb**x xxxx	xxxx xxxx	Write Program memory Page at address **a**:**b**.
Read EEPROM Memory	1010 0000	00xx xxx**a**	**bbbb bbbb**	**oooo oooo**	Read data **o** from EEPROM memory at address **a**:**b**.
Write EEPROM Memory	1100 0000	00xx xxx**a**	**bbbb bbbb**	**iiii iiii**	Write data **i** to EEPROM memory at address **a**:**b**.
Read Lock bits	0101 1000	0000 0000	xxxx xxxx	xx**oo oooo**	Read Lock bits. “0” = programmed, “1” = unprogrammed. See Table 81 on page 179 for details.
Write Lock bits	1010 1100	111x xxxx	xxxx xxxx	11**ii iiii**	Write Lock bits. Set bits = “0” to program Lock bits. See Table 81 on page 179 for details.
Read Signature Byte	0011 0000	00xx xxxx	xxxx xx**bb**	**oooo oooo**	Read Signature Byte **o** at address **b**.
Write Fuse bits	1010 1100	1010 0000	xxxx xxxx	**iiii iiii**	Set bits = “0” to program, “1” to unprogram. See Table 84 on page 181 for details.
Write Fuse High Bits	1010 1100	1010 1000	xxxx xxxx	**iiii iiii**	Set bits = “0” to program, “1” to unprogram. See Table 83 on page 180 for details.
Read Fuse bits	0101 0000	0000 0000	xxxx xxxx	**oooo oooo**	Read Fuse bits. “0” = programmed, “1” = unprogrammed. See Table 84 on page 181 for details.
Read Fuse High Bits	0101 1000	0000 1000	xxxx xxxx	**oooo oooo**	Read Fuse high bits. “0” = programmed, “1” = unprogrammed. See Table 83 on page 180 for details.
Read Calibration Byte	0011 1000	00xx xxxx	0000 00**bb**	**oooo oooo**	Read Calibration Byte

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.

2.4.3 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 t_{DD_FLASH}, прежде чем программировать следующую страницу. Поскольку микроконтроллер с очищенной памятью содержит $FF по всем адресам, можно пропустить повторную запись значения $FF.

2.4.4 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 t_{WD_EEPROM}, прежде чем программировать следующий байт.

Таблица 6 – Минимальная задержка при записи

Обозначение	Минимальная задержка
t_{WD_FUSE}	4.5 ms
t_{WD_FLASH}	4.5 ms
t_{WD_EEPROM}	9.0 ms
t_{WD_ERASE}	9.0 ms

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:

1) частота работы устройства составляет 8 МГц;

2) управляет до 8 приборами одновременно;

3) отправляет запросы по получению расписания на ПЭВМ;

4) отправляет запросы по получению текущего времени на ПЭВМ;

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

6) обладает пультом управления оператора на 16 кнопок;

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

8) использует таймеры T0 и T1 для вызова прерываний;

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

10) выводит текущее время МК-системы на LCD-дисплей;

11) 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 включаем 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