Министерство образования и науки Российской Федерации

Государственное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»

(МГТУ им. Н.Э. Баумана)

Факультет “Информатика и системы управления”

Кафедра “Компьютерные системы и сети”

УТВЕРЖДАЮ

Зав. кафедрой ИУ6,

д.т.н., проф. Сюзев В.В.

“\_\_\_” 2011 г.

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Курсового проекта

УПРАВЛЯЮЩИЙ КОНТРОЛЛЕР

Микропроцессорные системы

Руководитель,

к.т.н., доц. В.Я. Хартов

Исполнитель,

студ.гр. ИУ6-83 К.С. Сафронова

2011

# Реферат

РПЗ 62 с., 10 табл., 27 рис., 7 источников, 1 прил.

ПРОЕКТ, МИКРОКОНТРОЛЛЕР, ТАЙМЕР, НАГРУЗКИ, УМНЫЙ ДОМ.

Объектом проектирования является управляющий контроллер.

Цель работы – эскизный проект цифрового устройства ограниченной сложности.

Устройство выполнено в виде одноплатного модуля, предназначенного для использования в составе других устройств.

Материалы по курсовому проекту представлены в виде графической части и расчетно-пояснительной записки.

# Содержание

[Перечень сокращений 4](#_Toc218362564)

[Введение 5](#_Toc218362565)

[1. Описание и синтез структурно-функциональной схемы устройства 6](#_Toc218362566)

[1.1. Анализ требований 6](#_Toc218362567)

[1.2. Синтез структурной схемы 7](#_Toc218362568)

[1.3. Назначение узлов устройства 8](#_Toc218362569)

[2. Описание и синтез принципиальной схемы устройства 9](#_Toc218362571)

2.[1. Выбор элементной базы 9](#_Toc218362572)

[2.2. Описание архитектуры и технические характеристики МК Atmel AT89C51 13](#_Toc218362574)

[2.2.1. Технические характеристики МК Atmel AT89C51 13](#_Toc218362574)

[2.2.2. Структурная схема МК Atmel AT89C51 14](#_Toc218362574)

[2.2.3. Логическая организация памяти 15](#_Toc218362574)

[2.2.4. Параллельные порты ввода/вывода данных 18](#_Toc218362574)

[2.2.5. Таймеры МК Atmel AT89C51 20](#_Toc218362574)

[2.2.6. Обработка прерываний МК Atmel AT89C51 22](#_Toc218362574)

[2.2.7. Канал последовательной связи с интерфейсом RS-232C 24](#_Toc218362574)

[2.2.8. Последовательный порт USART МК Atmel AT89C51 26](#_Toc218362574)

2.[3. Описание принципиальной схемы устройства 28](#_Toc218362572)

[2.4. Устранение помех в цепи питания 30](#_Toc218362574)

[3. Расчет потребляемой мощности 31](#_Toc218362577)

[4. Описание алгоритмов програмы. 33](#_Toc218362578)

5. [Распределение адресного пространства МК Atmel AT89C51 36](#_Toc218362572)

6. [Тестирование и отладка программы для МК Atmel AT89C51 40](#_Toc218362572)

6.[1. Тестирование и отладка программы в симуляторе Proteus 40](#_Toc218362572)

[6.2. Оценка времени выполнения программы. 45](#_Toc218362578)

[7. Программирование памяти МК . 46](#_Toc218362579)

[8. Заключение 51](#_Toc218362584)

[9. Список использованных источников информации 52](#_Toc218362585)

[Приложение 1. Листинг программы 53](#_Toc218362586)

# Перечень сокращений

МК – микроконтроллер;

УК – управляющий контроллер;

ТЗ – техническое задание;

ПЭВМ - персональная электронно-вычислительная машина;

ПЗУ – постоянное запоминающее устройство;

ЭУ – элемент управления;

ОЗУ – оперативное запоминающее устройство;

ИМС – интегральная микросхема;

ЭРИ – электро-радио элемент;

**Введение**

Курсовой проект выполняется на основании учебного плана кафедры ИУ6. Темой данного проекта был выбран УК, предназначенный для управления работой нагрузок. В наше время, человечество все больше и больше пытается автоматизировать производство товаров и услуг, приспособить оборудование к выполнению определенных задач и просто сделать свою жизнь легче и комфортнее. В любой сфере жизни возникает ситуация, когда необходимо включить/выключить нагрузку. Заниматься этим вручную сегодня уже не актуально. К тому же управлять нагрузкой иногда нужно в отсутствии человека. Выполнить такую задачу поможет предлагаемый УК.

Использование данного УК может быть разнообразным. В домашнем хозяйстве это управление бытовыми электроприборами, освещением, устройствами для аквариума и др. С помощью УК мы можем создать так называемый «умный дом», где управление бытовыми приборами осуществляется автоматически. В производстве это управление промышленными электроустановками, такими как конвейеры, транспортеры, электродвигатели, мешалки. К тому же предлагаемый УК предназначен для управления несколькими нагрузками одновременно, что увеличивает наши возможности и желания.

1. **Описание и синтез структурно-функциональной схемы устройства**
   1. **Анализ требований**

Основаниями для выполнения работы являются: учебный план кафедры ИУ6 и техническое задание на курсовой проект. Согласно ТЗ целью данной работы является разработка МК, а именно таймера для включения/выключения подключенных к нему электроприборов согласно заданному расписанию на текущую неделю для «умного дома». Количество электроприборов – 8. Источник питания электроприборов – промышленная сеть 220 В 50 Гц. Расписание в формате «день недели – часы – минуты» загружается в МК из ПЭВМ по последовательному каналу RS-232. Необходимо предусмотреть установку начального времени, а также работу МК – системы с автономным источником питания. Данный УК предлагается разработать на основе МК Atmel AT89C51.

В качестве задач, которые необходимо решить в процессе работы, можно перечислить следующие: разработка программного обеспечения для микроконтроллера, моделирование работы устройства, синтез структурной и принципиальной схемы устройства.

Разрабатываемую микроконтроллерную систему можно разделить на две составные части:

* аппаратная часть – представляет собой коммутацию микроконтроллера, цепи блока питания и интерфейса приёма данных от внешнего устройства, обеспечивающего задание текущего времени, а также режимов работы каждого из подключенных электроприборов.
* программная часть – представляет собой программу, находящуюся в ПЗУ микроконтроллера и непосредственно выполняющую как счёт времени, так и запуск или отключение соответствующего электроприбора. Программа находится во FLASH-памяти микроконтроллера и функция изменения её пользователем не предусмотрена.

Анализ требований, предъявляемый к функционированию, системы показывает, что разрабатываемая система представляет собой аппаратно-программный модуль, являющийся печатной платой, на которой находится микроконтроллер, соединенный с элементами управления нагрузками и автономный источник питания. В зависимости от режима работы, МК либо отсчитывает импульсы, равные текущему времени, и каждый раз сравнивает с введенным расписанием включения/выключения нагрузок, либо находится в режиме задания начального времени, либо в режиме задания/изменения расписания. Определение режима работы, ввод расписания и начального времени производится с ПЭВМ через COM-порт по последовательному каналу связи RS-232. Для этого порт, по которому вводится информация извне, всегда стоит в режиме готовности принять информацию.

МК Atmel AT89C51 осуществляет счёт времени, сверку с ним загруженных расписаний и выдачу управляющих сигналов на элементы управления включением/выключением соответствующего прибора. Программа микроконтроллера осуществляет вычисление текущего времени в формате пригодном для сравнения с загруженным пользователем расписанием. При совпадении одного из загруженных расписаний с текущим временем осуществляется выдача соответствующего управляющего сигнала на один из выводов микроконтроллерной системы. Прерывание от порта UART происходит при выборе пользователем режима задания начального времени или режима задания расписания и подаче соответствующего сигнала через последовательный интерфейс.

**1.2. Синтез структурной схемы**

На рисунке 1 приведена общая структурная схема функционирования разрабатываемого УК. По результатам предыдущего пункта можно выделить следующие структурные блоки устройства:

* микроконтроллер (Atmel AT89C51);
* блок питания;
* элемент управления нагрузкой;
* согласующее устройство;

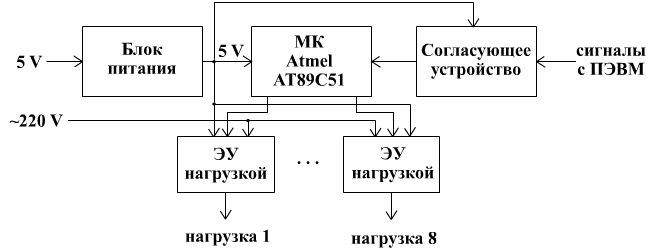


Рис. 1. Структурная схема

С помощью согласующего устройства пользователь с ПЭВМ задает расписание включения/выключения нагрузок, задает начальное время. МК осуществляет сравнение текущего времени с заданным расписанием и при совпадении подает сигнал на соответствующий элемент управления нагрузкой, включая/выключая данную нагрузку. Работа МК, согласующего устройства, ЭУ нагрузкой осуществляется с помощью подачи напряжения в 5 В, а нагрузки (электроприборы) работают от промышленной сети в 220 В. В блоке питания предусмотрен автономный источник питания – батарея.

**1.3. Назначение узлов устройства**

1. Микроконтроллер Atmel AT89C51 выполняет следующие функции:

* Отсчет таймером времени (1 неделя);
* Сравнение текущего времени с заданным расписанием;
* Вырабатывает активные управляющие сигналы для включения/выключения нагрузок на соответствующих параллельных портах ввода/вывода;
* Прием данный через последовательный канал ввода/вывода от ПЭВМ;
* Хранение данных, полученных от ПЭВМ.

1. Блок питания выполняет ряд функций:

* Подача напряжения в 5 В на основные узлы устройства (МК, согласующего устройства, ЭУ нагрузкой);
* Содержит элементы устранения помех в цепях питания;
* Содержит источник резервного автономного питания – гальваническая батарея в случае отключения основного питания для поддержания работы таймера. Но включение приборов заработает только при подаче основного напряжения;
* Присутствует стабилизатор напряжения для сглаживания импульсов.

1. Элемент управления нагрузкой представляет собой переключатель, который под воздействием управляющего сигнала от соответствующего параллельного порта ввода/вывода МК (высокое напряжение – включение, низкое напряжение – выключение) подает на нагрузку переменное напряжение 220 В и нагрузка (электроприбор) включается или выключается. В качестве ЭУ нагрузкой может выступать реле, симисторы или тиристоры.
2. Согласующее устройство необходимо при передачи данных от ПЭВМ по последовательному каналу связи RS-232 через COM-порт на UART порт МК. Данный блок согласовывает напряжение между COM-портом и МК.

2. Описание и синтез принципиальной схемы устройства

**2.1 Выбор элементной базы**

В соответствии с ТЗ данный УК разрабатывается на основе МК из семейства MCS-51 Atmel AT89C51. Семейство MCS-51 в настоящее время является наиболее известным семейством. Первый микроконтроллер этого семейства 8051, выпущенный фирмой Intel в 1980 г., снискал большую популярность в мире благодаря удачной архитектуре ядра, возможности гибкого выбора соотношения внутренней и внешней памяти, невысокой стоимости. Со временем популярность МК значительно возросла благодаря увеличению степени интеграции и развитию их архитектуры, повышению производительности, включению дополнительных ресурсов ввода/вывода. Преимущество использования микроконтроллеров MCS-51 в том, что они имеют широкое распространение и поэтому дешево стоят. Их легко программировать, так как они имеют гибкую систему команд и имеется огромное количество разнообразной литературы по программированию. Не требуют каких-то особенных технических условий и занимают мало места на печатной плате.

Базовая архитектура MCS-51 предусматривает:

* Разделенную память программ и память данных (гарвардская архитектура). Память программ имеет объем 4 Кбайта, память данных – 128 байт;
* АЛУ аккумуляторного типа. При этом один из операндов поступает в АЛУ из регистра-аккумулятора, а результат операции помещается в аккумулятор;
* Наличие блока регистров специальных функций, включающего в себя четыре восьмиразрядных порта ввода-вывода с альтернатиными функциями, один последовательный дуплексный канал ввода-вывода, два 16-разрядных таймера, двухуровневую систему прерываний.

Первую линию микроконтроллеров MCS-51 представляли микроконтроллеры 8051/8751/8031, разработанные по технологии n-МОП, имевшими постоянную или перезаписываемую с ультрафиолетовым стиранием память программ либо только внешнюю память. Работы по совершенствованию технологии перезаписываемой памяти программ привели к созданию микроконтроллеров с электрически стираемой Flash-памятью, вытеснивших в настоящее время микроконтроллеры с ультрафиолетовым стиранием памяти. МК подобного типа выпускают фирмы Atmel, Philips и другие с маркой 89С51.

1. Основным узлом данного устройства является микроконтроллер AT89C51 фирмы Atmel, который представляет собой однокристальную микроЭВМ с системой команд и архитектурой MCS-51. Графическое обозначение микроконтроллера приведено на рисунке 2.

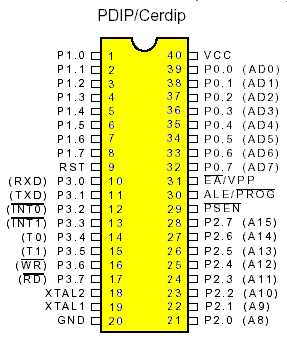


Рис. 2. Пример цоколевки корпуса AT89C51

1. Для согласования уровней напряжений между COM-портом и микроконтроллером будем использовать микросхему MAXIM MAX232, предназначенная специально для такого согласования и имеющая встроенные средства для такого преобразования уровней. Работая от стандартного напряжения 5 В, она преобразует его в +14/-14 В, необходимые интерфейсу RS-232. Графическое обозначение MAX232 приведено на рисунке 3.
2. В блоке питания присутствует стабилизатор напряжения L7805 для сглаживания импульсов напряжения, выбор которого обуславливается высокой точностью и качеством стабилизации (сглаживания), а также низкой потребляемой мощностью и отсутствием необходимости в каких-либо дополнительных электро-радио элементах, кроме сглаживающих конденсаторов. Последние две цифры обозначения показывают выходное напряжение прибора (5 В). Графическое обозначение микросхемы L7805 показано на рисунке 4.

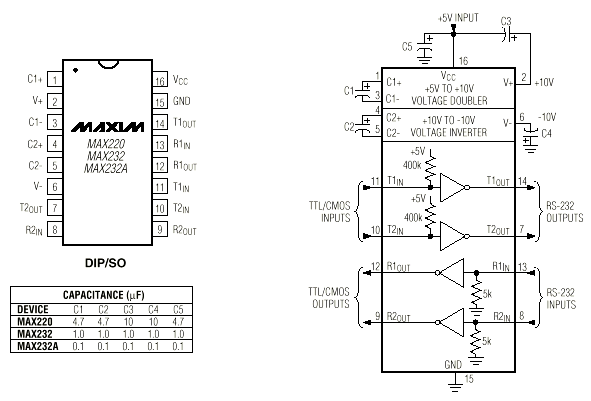


Рис. 3. Графическое обозначение микросхемы MAXIM

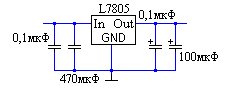


Рис. 4. Графическое обозначение микросхемы L7805

1. В ЭУ нагрузкой коммутируем мы силовую высоковольтную цепь, 220 вольт, а МК у нас низковольтный, работает на пять вольт. Поэтому во избежание эксцессов нужно произвести потенциальную развязку. То есть сделать так, чтобы между высоковольтной и низковольтной частью не было прямого электрического соединения. Например, сделать оптическое разделение. Для этого существует специальная сборка — симисторный оптодрайвер MOC3051. Его графическое обозначение представлено на рисунке 5.
2. В ЭУ нагрузкой в качестве элемента переключения будем использовать симистор ВТ139 - устройство для коммутации нагрузки на переменном токе. Графическое обозначение симистора смотри на рисунке 6.

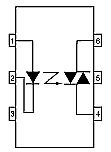
[](http://www.promelec.ru/UPLOAD/fck/image/lines_pict/mic_14_6_1.gif)

Рис. 5. Графическое обозначение микросхемы MOC3051

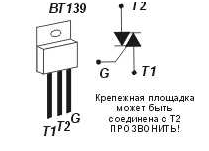


Рис. 6. Графическое обозначение симистора ВТ139

**2.2 Описание архитектуры и технические характеристики МК Atmel AT89C51**

**2.2.1. Технические характеристики МК Atmel AT89C51**

Технические характеристики:

* совместимость с MCS-51;
* 4 Кбайт встроенной перепрограммируемой Flash памяти;
* 1000 циклов Запись/Удаление;
* трехуровневая программная защита памяти;
* 128 байт встроенного ОЗУ;
* 32 программируемых портa ввода\вывода;
* два 16 битных счетчика\таймера;
* шесть источников прерываний;
* программируемый последовательный порт;
* низкое потребление в режиме ожидания и энергосберегающий режим "Power Down";

МК Atmel AT89C51 - низкопотребляющий, быстродействующий 8-ми битный CMOS микроконтроллер с 4 Кбайтами Flash памяти. При производстве микроконтроллеров были использованы Хай-Тэк технологии фирмы Atmel. В частности, МК изготовлены согласно промышленной рекомендации MCS-51, которая распространяется и на цоколевку. Однокристальную Flash память можно программировать как изнутри, так и извне (используя программатор). Сама память расположена на одном кристалле с процессором, что позволило добиться максимальной производительности.

MK Atmel AT89C51 обеспечивает следующие стандартные характеристики: 4 Кбайта Flash памяти, 128 байт RAM, 32 линии ввода/вывода, два 16-битных таймера/счетчика, пятивекторная двухуровневая архитектура прерываний, полный дуплексный последовательный порт, встроенный в кристалл генератор и часы. Кроме того, МК AT89C51 - разработка "static logic", а следовательно, работоспособен вплоть до нулевой частоты. В режиме ожидания процессор остановлен, но ОЗУ, таймер/счетчики, последовательный порт и система прерываний продолжают функционировать. Режим "Power Down" сохраняет содержимое ОЗУ, но замораживает генератор, блокирует все другие функции МК, пока не будет осуществлён аппаратный сброс ("Reset").

**2.2.2. Структурная схема МК Atmel AT89C51**

МК серии AT89 фирмы Atmel представляют собой восьмиразрядные однокристальные микроЭВМ с системой команд и архитектурой MCS-51. На рисунке 7 приведена структурная схема МК Atmel AT89C51.

В состав МК входят:

* блок АЛУ;
* перепрограммируемая Flash-память;
* оперативная память данных RAM;
* блок синхронизации и управления БСУ;
* четыре восьмиразрядных порта ввода/вывода P0, P1, P2. P3 с альтернативными функциями;
* регистры специальных функций SFR.

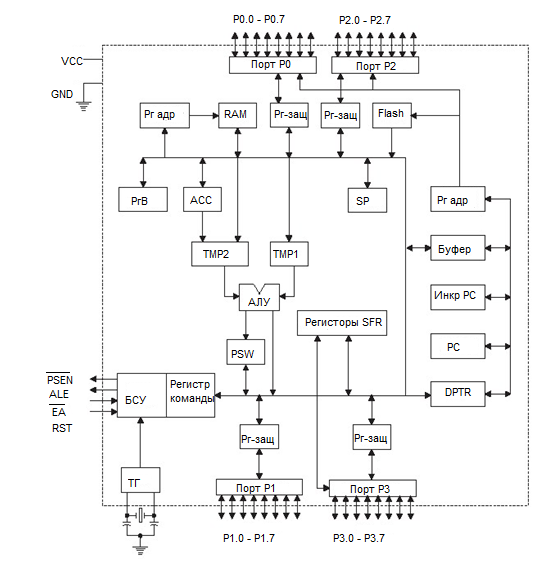
****

Рис.7. Структурная схема MK Atmel AT89C51

Все устройства объединяет внутренняя восьмиразрядная шина данных. Внешний интерфейс представляют 32 линии портов ввода/вывода, выводы тактового генератора (XTAL1, XTAL2), вход сброса RST, выходные сигналы стробирования адреса ALE при работе с внешней памятью и обращения к памяти программ не PSEN, сингал отключения резидентной памяти программ не EA. Дополнительная группа внешних сигналов управления содержит сигналы, передаваемые через порт P3. В данной работе будет задействован вывод порта P3.0 как приемник RxD по последовательному каналу ввода/вывода и выводы порта P2.0 – P2.7 для управления работой 8 нагрузок.

Flash-память, применяемую в микроконтроллерах, можно неоднократно перепрограммировать, число допустимых циклов перепрограммирования достигает 10000. Адресация ячеек памяти осуществляется через программный счетчик PC и 16-разрядный регистр-указатель DPTR. При превышении максимального адреса внутренней памяти адрес ячейки направляется в порты P0, P2 для поиска ячейки во внешней памяти. Ячейки внутренней памяти данных адресуются через регистр адреса RAM или указатель стека SP.

**2.2.3. Логическая организация памяти**

В микроконтроллерах с архитектурой MCS-51 используется память двух типов:

* постоянная (ROM) - память программ предназначена для хранения команд (программ), управляющих функционированием микроконтроллера, хранения различных констант, не меняющихся во время работы программы. Память программ представляет собой электрически стираемое ППЗУ (Flash-ПЗУ). В МК Atmel AT89C51 резидентная память программ имеет объем 4 Кбайта, но может быть расширена до 64 Кбайт.
* оперативная (RAM), применяемая для хранения данных. В МК Atmel AT89C51 резидентная память данных имеет объем 128 байт.

Начальная область адресов постоянной памяти программ ROM, начиная с адреса 0003h, зарезервирована для таблицы прерываний. В связи с этим при использовании прерываний основную программу, вызываемые подпрограммы и обработчики прерываний следует размещать выше таблицы прерываний. А учитывая, что при сбросе микроконтроллера программный счетчик принимает значение 0000h, по указанному адресу следует разметить команду перехода к основной программе. Переход из резидентной памяти программ к внешней, выше 4 кбайт, происходит автоматически. Используя управляющий сигнал интерфейса не EA = 0 можно отключит резидентную память программ и работать только с внешней памятью, размером до 64 Кбайт. Организация постоянной памяти программ представлена на рисунке 8.

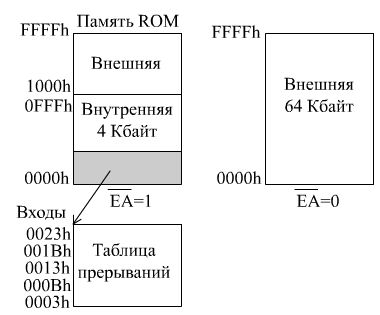


Рис. 8. Организация постоянной памяти

Адресное пространство оперативной памяти данных RAM охватывает регистровый сегмент (00h…1Fh), битовый сегмент (20h…2Fh), ячейки внутренней памяти данных и регистры специальных функций. Регистровый сегмент представлен 32 регистрами, организованными в четыре банка, по восемь регистров в каждом банке. Выбор банка регистров (0, 1, 2, 3) осуществляется с помощью 2 бит регистра состояния микроконтроллера, выбор регистра в каждом банке – по именам R0 – R7. При работе с регистрами выбранного банка доступ к остальным регистрам можно произвести только как к ячейкам памяти адресного пространства памяти данных. Организация оперативной памяти данных представлена на рисунке 9.

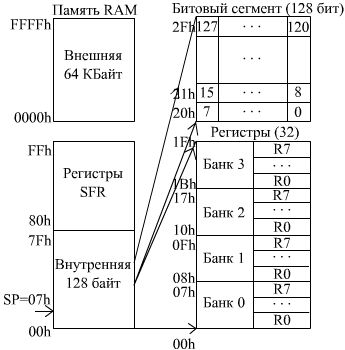


Рис. 9. Организация оперативной памяти

Выше регистрового сегмента по адресам 20h…2Fh расположен битовый сегмент. Занимающий 16 байт (128 бит). Доступ к элементам этого сегмента можно осуществить двояко: либо побитно с помощью команд битовых операций и восьмиразрядных адресов битов, либо побайтно. Выбирая с помощью одной команды одновременно 8 бит и применяя в дальнейшем маску для выделения нужных битов.

Стек располагается во внутренней памяти данных. При сбросе микроконтроллера указатель стека SP принимает значение 0007h.

В старшей половине внутреннего адресного пространства (80h…FFh) располагаются регистры специальных функций. Обращение к ним обычно осуществляется по именам. Также по именам можно обращаться к отдельным битам этих регистров. Выполняя битовые операции. Все это необходимо учитывать при работе с ячейками для хранения данных при объеме памяти 256 байт.

Память данных микроконтроллера является расширяемой за счет дополнительной внешней оперативной памяти объемом до 64 Кбайт.

Для адресации ячеек внешней и внутренней памяти использую косвенно-регистровую адресацию через регистры DPTR, R0 или R1.

**2.2.4. Параллельные порты ввода/вывода данных**

Микроконтроллеры имеют четыре встроенных восьмиразрядных порта P0, P1, P2, P3. Каждый из портов помимо выполнения основных функций ввода-вывода наделен дополнительными функциями.

* Порт P0 при подключении внешней памяти работает в мультиплексном режиме, осуществляя сначала передачу младшего байта адреса памяти, затем принимая или передавая байт данных. Старший байт адреса в этом случае выводится через порт P2. при программировании памяти программ и верификации через порт P0 пересылаются данные.
* Порт P1 при программировании и чтении внутренней памяти программ служит для передачи младшего байта адреса, порт P2 - для передачи старшего байта адреса.
* Порт P3 используется для приема внешних запросов прерываний, выдачи сигналов чтения и записи внешней памяти данных, канала последовательного ввода-вывода и таймеров.

Каждый разряд порта имеет в своем составе триггер-защелку (D-триггер), выходной драйвер и входные буферы.

1. Порт 1. Имеет наиболее простое решение имеет. При выводе данные (0 или 1) записываются в триггер-защелку. В состоянии «0» сигнал с инверсного выхода триггера открывает выходной транзистор, устанавливая на выходной линии порта низкий уровень логической нуля. В противном случае транзистор закрыт и на выходной линии порта присутствует высокий уровень логической единицы. При вводе данных выходной транзистор должен быть закрыт. Для этого необходимо предварительно установить триггер-защелку в единицу. Если от источника данных поступит сигнал «0», то потенциал выходной линии станет низким и при вводе на внутреннюю шину данных микроконтроллера поступит сигнал «0».
2. Порт P0. Является мультиплексируемым и двунаправленным. При обращениях к внешней памяти сигнал «Управление» переключает ключ мультиплексора MUX в верхнее положение. При этом адресный код (1 или 0) открывает один из выходных транзисторов, устанавливая высокий или низкий уровень сигнала на выходе. Аналогично вслед за этим происходит вывод данных. Когда порт неактивен, его выход находится в состоянии высокого импеданса. При работе порт P0 в режиме ввода-вывода необходимо его выход соединить внешним резистором с источником питания, «подтянув» выход к уровню логической единицы.
3. Порт P2 . Имеет сходное решение, за исключением выходной цепи, что позволяет сохранить записываемый старший байт адреса при обращении к внешней памяти и протяжении всего командного цикла. Схема порта на рисунке 10.

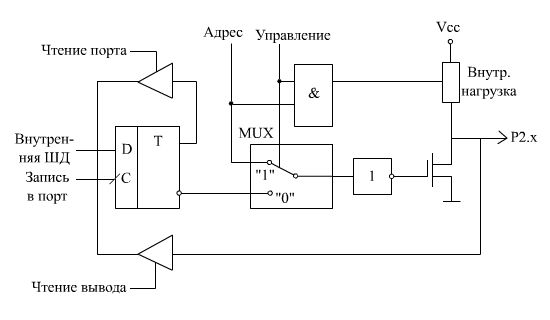


Рис. 10. Схема порта P2

1. Порта P3. Схема по сравнению с портом P1 имеет небольшое отличие. Триггер-защелка выполняет ту же функцию при выводе и вводе данных. Однако схема согласования с выходным транзистором изменена для осуществления альтернативного управления выходом. В режиме ввода-вывода порта сигнал альтернативной функции принимает значение «1», что не препятствует выполнению функций ввода-вывода. В случае вывода альтернативных сигналов, например чтения или записи, в выходную цепь поступает сигнал «0». Что ведет к открыванию транзистора и появлению выходного сигнала низкого уровня. Прием альтернативных сигналов, например запроса прерывания, происходит при автоматической установке триггера в состояние «1». Схема порта на рисунке 11.

* P3.0 - RXD (Принимаемые данные последовательного порта)
* P3.1 - TXD (Передаваемые данные последовательного порта)
* P3.2 - INT0 (Внешнее прерывание 0)
* P3.3 - INT1 (Внешнее прерывание 1)
* P3.4 - T0 (внешний вход таймера 0)
* P3.5 - T1 (внешний вход таймера 1)
* P3.6 - WR (Внешний строб записи)
* P3.7 - RD (Внешний строб чтения)

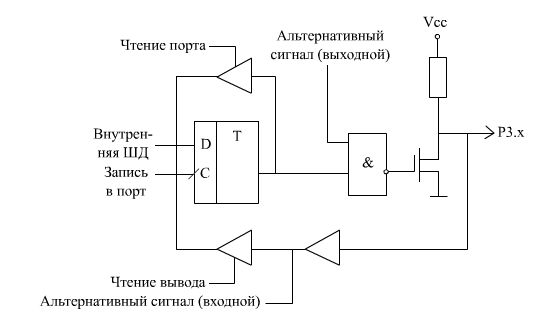


Рис. 11. Схема порта P3

В данной работе будет задействован вывод порта P3.0 как приемник RxD по последовательному каналу ввода/вывода и выводы порта P2.0 – P2.7 для управления работой 8 нагрузок.

**2.2.5. Таймеры МК Atmel AT89C51**

В базовых моделях микроконтроллеров семейства MCS-51 имеются два программируемых 16-разрядных таймера (счетчика) T/Cx (x = 0, 1), которые могут быть использованы в качестве таймеров и как счетчика внешних событий. В первом случае содержимое таймер (счетчика) инкрементируется в каждом машинном цикле, т.е. через каждые 12 периодов колебаний кварцевого резонатора, во втором - под воздействием перехода из 1 в 0 внешнего входного сигнала, подаваемого на соответствующий вывод Tx микроконтроллера.

Для управления режимами работы таймеров T/C0, T/C1 используют два регистра специальных функций TMOD и TCON.

Таблица 1. Структура регистра TMOD

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер разряда | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Имя | GATE1 | C/неT1 | M1 | M0 | GATE0 | C/неT0 | M1 | M0 |

* GATEx – управление блокировкой. Если бит установлен в единицу, то таймер (счетчик) T/Cx разрешен для счета до тех пор, пока на входе неINTx присутствует высокий уровень сигнала и бит управления TRx установлен в состояние «1». Если бит сброшен, то работа T/Cx разрешается после установки бита управления TRx в единицу.
* C/неTx – бит выбора режима работы таймера. Если бит равен 0, таймер работает от внутреннего источника сигналов синхронизации. Если бит равен 1, то работает счетчик от внешних сигналов на входе Tx.
* M1, M0 – биты режима работы таймера (всего четыре режима).

00 (режим0) – младшая половина таймера TLx работает как пятиразрядный предделитель, выходные сигналы которого поступают в старшую половину THx;

01 (режим1) – 16-разрядный режим работы таймера (счетчика). TLx и THx включены последовательно;

10 (режим2) – восьмиразрядный автоматически перезагружаемый таймер (счетчик). THx хранит значение, которое должно быть загружено в TLx каждый раз при переполнении TLx;

11 (режим3) – таймер (счетчик) T/C1 останавливается. TL0 работает как восьмиразрядный таймер (счетчик), и его режим определяется управляющим битом таймера T/C0. TH0 работает только как восьмибитный таймер, и его работа определяется управляющим битом таймера T/С1.

Таблица 2. Структура регистра TCON

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер разряда | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Имя | TF1 | TR1 | TF0 | TR0 | - | - | - | - |

* TF1 – флаг переполнения таймера T/C1. Устанавливается при переполнении таймера (счетчика), сбрасывается при обслуживании прерывания аппаратно;
* TR1 – бит управления таймера T/C1. Устанавливается (сбрасывается) программой для пуска (останова) таймера;
* TF0 – флаг переполнения таймера T/C0. Устанавливается при переполнении таймера (счетчика), сбрасывается при обслуживании прерывания аппаратно;
* TR0 – бит управления таймера T/C0. Устанавливается (сбрасывается) программой для пуска (останова) таймера;

В данной работе использовался первый режим (01) работы таймера. Структура таймера, работающего в первом режиме, приведена на рисунке 12.

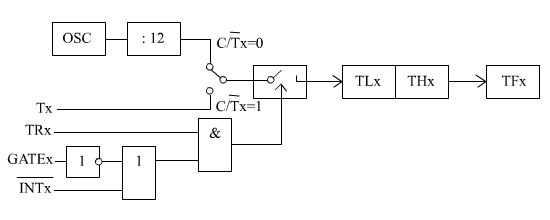


Рис. 12. Структура таймера в режиме 1

В этом режиме таймер имеет 16-разрядный формат. При переходе из состояния «все единицы» в состояние «все нули» (переполнение таймера) устанавливается в единицу флаг прерывания от таймера TFx, используемый для формирования запроса прерывания INTR. Входные синхросигналы OSC поступают на вход таймера, когда управляющий бит TRx установлен в состояние «1», и либо управляющий бит GATEx равен 0, либо на внешний вывод запроса прерывания неINTx поступает сигнал 1. В таймерном режиме, при значении C/неT = 0, можно отсчитать интервал времени до 65536 машинных циклов (длительность периода машинного цикла составляет 12 периодов тактовой частоты генератора). Именно поэтому был выбран кварцевый резонатор с частотой 12 МГц. При данной частоте таймер отсчитывает 65536 мкс.

**2.2.6. Обработка прерываний МК Atmel AT89C51**

Систему прерываний в МК с архитектурой MCS-51 по способу организации и принципу действия можно отнести к системам радиального типа. Все запросы, внешние и внутренние, обрабатываются путем аппаратного вызова прерывающих подпрограмм через таблицу прерываний по фиксированным адресам. Всего обрабатываемых запросов прерываний – 5. Для внешних прерываний зарезервированы адреса 0003h и 0013h, для внутренних прерываний от двух таймеров – адреса 000Вh и 001Вh, от последовательного канала ввода-вывода – 0023h. Для каждого прерывания в таблице прерываний отведено 8 байт.

Каждый из запросов прерываний может быть разрешен или запрещен с помощью регистра маски прерываний IE, формат которой приведен в таблице 3.

Таблица 3. Структура регистра IE

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер разряда | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Имя | EA | - | - | ES | ET1 | EX0 | ET0 | EX0 |

Здесь назначение битов: EX0, EX1 – маски внешних прерываний; ET0, ET1 – маски прерываний от таймеров T0, T1; ES – маска запроса от последовательного канала; EA – общая маска прерываний. При значении 1 соответствующего бита регистра прерывание разрешено, при 0 – запрещено. Бит EA при значении 1 разрешает работу системы прерываний, при 0 все прерывания запрещены.

Система прерываний MCS-51 является двухуровневой. Запросы верхнего уровня имеют более высокий приоритет по сравнению с запросами нижнего уровня. Особенностью системы прерываний МК MCS-51 является возможность «прикрепления» запроса к верхнему или нижнему уровню прерываний. Достигается это с помощью регистра приоритетов IP, имеющего следующий формат, таблица 4.

Таблица 4. Структура регистра IP

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер разряда | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Имя | - | - | - | PS | PT1 | PX0 | PT0 | PX0 |

Здесь назначение битов: PX0, PX1 – приоритеты внешних прерываний; PT0, PT1 – приоритеты прерываний от таймеров T0, T1; PS – приоритет запроса от последовательного канала. При значении 1 соответствующего бита регистра запрос относится к верхнему уровню прерываний, при значении 0 – к нижнему.

Структурная схема системы прерываний приведена на рисунке 13.

Запросы внешних прерываний неINT0, неINT1 могут быть статическими с низким уровнем сигнала либо динамическими с переключением сигнала из 1 в 0. Выбор способа обработки зависит от значений битов IT0, IT1 в регистре TCON. Прерывания от таймеров возникают при переполнении таймеров, т.е. при установке в единицу флагов переполнения TF0, TF1. Сброс флагов происходит автоматически при переходе к прерывающей программе. Прерывания от последовательного канала происходят при установке в состояние «1» флага готовности передатчика TI или приемника RI, поэтому в начале работы прерывающей программы путем проверки выясняется источник прерываний. Сброс флагов в обоих случаях выполняется программными средствами.

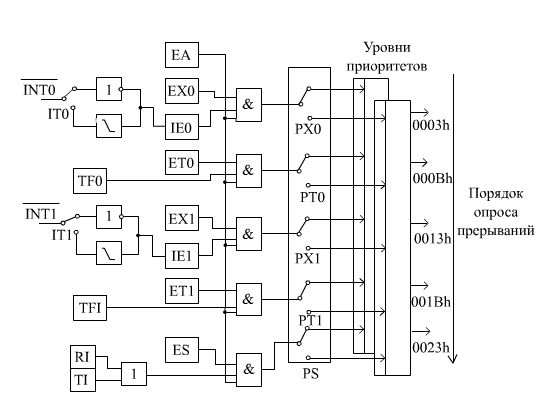


Рис.13. Структурная схема системы прерываний

В данной работе используется прерывание от последовательного канала ввода/вывода. Прерывание от порта UART происходит при выборе пользователем режима задания начального времени или режима задания расписания и подаче соответствующего сигнала через последовательный интерфейс.

**2.2.7. Канал последовательной связи с интерфейсом RS-232C**

Широкое распространение для связи с компьютером получил простейший стандарт RS-232C, реализуемый COM-портами. RS-232C – интерфейс синхронной и асинхронной передачи данных, определяемый стандартом EIA RS-232C. Стандарт описывает управляющие сигналы интерфейса, пересылку данных, электрический интерфейс и типы разъемов. Стандартом предусмотрены асинхронный и синхронный режим обмена, но COM-порты поддерживают только асинхронный режим.

Стандарт RS-232C использует несимметричные передатчики и приемники: сигнал передается относительно общего провода – схемной земли. Все сигналы RS-232C имеют специально выбранные уровни, обеспечивающие высокую помехоустойчивость связи. Отметим, что данные передаются в отрицательной логике. Уровни сигналов на выходе передатчика должны быть в диапазонах от -15 до -5 В и от +5 до +15 В для представления единицы и нуля соответственно.

Основными характеристиками интерфейса являются:

* максимальная удаленность объекта от компьютера – 15 м;
* максимальная скорость передачи – 115 Кбит/с;
* характер сигнала – несимметричный по напряжению;
* схема соединения – полный дуплекс, от точки к точке.

Данные по интерфейсу RS-232C перемещаются в последовательном коде побайтно. В режиме асинхронного обмена каждый байт обрамляется стартовым и стоповым битом.Данные могут передаваться как в одну так и в другую сторону (дуплексный режим). Получив стартовый бит, приемник выбирает из линии биты данных через определенные интервалы времени.

Компьютер имеет 25-контактный (DB25P) или 9-контактный (DB9P) разъем для подключения RS-232C. Назначение контактов разъема DB9P приведено в таблице 5.

Таблица 5. Сигналы по интерфейсу RS-232C

|  |  |  |
| --- | --- | --- |
| Контакт DB9P | Имя сигнала | Направление |
| 1 | DCD | Вход |
| 2 | RxD | Вход |
| 3 | TxD | Выход |
| 4 | DTR | Выход |
| 5 | GND | - |
| 6 | DSR | Вход |
| 7 | RTS | Выход |
| 8 | CTS | Вход |
| 9 | RI | Вход |

Сигналы интерфейса RS-232C имеют следующие назначения:

* FG- защитное заземление (экран);
* TxD – данные, передаваемые компьютером в последовательном коде;
* RxD – данные, принимаемые компьютером в последовательном коде;
* RTS – сигнал запроса передачи, используемый для уведомления приемника модема о наличии данных для передачи;
* CTS – сигнал уведомления передатчика о готовности модема принять данные;
* DSR – включение модема и готовность к работе;
* SG – сигнальное заземление, нулевой провод;
* DCD – обнаружение несущей данных;
* DTR – включение терминала и готовность к работе;
* RI – индикатор вызова.

Для подключения произвольного устройства сопряжения (МС) к компьютеру через RS-232C обычно используют трех- или четырехпроводную линию связи, но можно задействовать и другие сигналы интерфейса. Для четырехпроводной линии связи используют сигналы TxD, RxD, SG, FG. Для двухпроводной линии связи в случае только передачи из компьютера во внешнее устройство используют сигналы SG и TxD. Все 10 сигналов интерфейса задействуют только при соединении компьютера с модемом.

В качестве преобразователей уровней напряжения в настоящее время широко используются специальные интерфейсные микросхемы, выпускаемые различными фирмами, среди них микросхемы фирмы MAXIM. Они содержат преобразователь напряжения +5 В в напряжение +10 В, инвертор, преобразующий напряжение +10 В в напряжение -10 В, и преобразователь уровней сигналов последовательного интерфейса.

В данной работе необходимо осуществить передачу данные (начальное время, расписание работы нагрузок) с компьютера на МК через COM-порт по каналу последовательной связи с интерфейсом RS-232C. Это двухуровневая линия связи, на COM-порте будет задействован разъем TxD (3) и GND. Канал связи будет соединяться с МК Atmel AT89C51 через порт UART (P3.0 – RxD). К качестве преобразователя уровней напряжения между COM-портом и МК будет стоять микросхема MAXIM MAX232.

**2.2.8. Последовательный порт USART МК Atmel AT89C51**

Уникальный синхронный (асинхронный) приемопередатчик USATR используется для приема и передачи информации, представленной последовательным кодом, поэтому его часто называют также последовательным портом.

Поток данных, передаваемых по каналу USART, представляет собой совокупность посылок или кадров. Каждый кадр содержит стартовый бит, 8 или 9 бит данных и стоповый бит. Стартовый бит имеет уровень логического нуля, стоповый – уровень логической единицы.

Режимы работы:

1. Режим 0 (синхронные прием-передача). Информация передается и принимается через вывод приемника RxD (линия порта P3.0). Через вывод передатчика TxD (P3.1) передаются сигналы синхронизации, сопровождающие каждый передаваемый или принимаемый бит. Число пересылаемых битов – 8. Частота передачи равна 1/12 частоты кварцевого резонатора.
2. Режим 1. В этом режиме передаются через вывод TxD или принимаются через RxD 10 бит информации: стартовый бит, 8 бит данных и стоповый бит. При приеме информации стоповый бит поступает в разряд RB8 регистра SCON. Скорость приема-передачи – величина переменная и задается таймером.
3. Режим 2. В этом режиме через вывод TxD передаются или через RxD принимаются 11 бит информации: стартовый бит, 8 бит данных (D0 - D7), программируемый девятый бит и стоповый бит. При передачи девятый бит данных (TB8 из регистра SCON) может быть использован по усмотрению программиста. Например, в целях повышения достоверности передачи путем контроля по четности девятый бит используют для передачи признака паритета из слова состояния программы. При приеме девятый бит данных помещается в разряд RB8 регистра SCON, при этом стоповый бит теряется.
4. Режим 3. В этом режиме порт работает так же, как и в режиме 2, за исключением частоты приема-передачи, которая является величиной переменной и задается таймером.

Режим работы приемопередатчика устанавливается в регистре управления (состояния) SCON. Его формат приведен в таблице 6.

Таблица 6. Структура регистра SCON

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер разряда | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Имя | SM0 | SM1 | SM2 | REN | TB8 | RB8 | TI | TI |

Разряды регистра управления приемопередатчик SCON имеют следующее функциональное назначение:

* SM0, SM1 – биты режима работы приемопередатчика (00 – режим 0, 01 – режим 1, 10 – режим 2, 11 – режим 3) устанавливаются (сбрасываются) программно;
* SM2 – бит управления режимом приемопередатчика, устанавливается программно для запрета приема байтов данных, в которых девятый бит равен 0;
* REN – бит разрешения приема, устанавливается (сбрасывается) программно для разрешения (запрета) приема последовательных данных;
* TB8 – передаваемый девятый разряд данных, устанавливается (сбрасывается) программно;
* RB8 – принятый девятый разряд данных, устанавливается (сбрасывается) аппаратно;
* TI – флаг прерывания передатчика, устанавливается аппаратно по окончанию передачи байт. Сбрасывается программно после обслуживания прерывания;
* RI – флаг прерывания приемника, устанавливается аппаратно при приеме байта. Сбрасывается программно после обслуживания прерывания.

Режим работы приемопередатчика устанавливается путем загрузки в старшие разряды регистра SCON двухразрядного кода.

В данной работе при передачи данных с ПЭВМ (начальное время, расписание работы нагрузок) используется режим 2 работы последовательного порта USART.

**2.3. Описание принципиальной схемы устройства**

Центральным узлом схемы устройства является МК Atmel AT89C51. Микроконтроллер Atmel AT89C51 выполняет следующие функции:

* Отсчет таймером времени (1 неделя);
* Сравнение текущего времени с заданным расписанием;
* Вырабатывает активные управляющие сигналы для включения/выключения нагрузок на соответствующих параллельных портах ввода/вывода;
* Прием данный через последовательный канал ввода/вывода от ПЭВМ;
* Хранение данных, полученных от ПЭВМ.

В данном курсовом проекте был использован кварцевый резонатор X1 для задания тактовой частоты микроконтроллера AT89C51. Этот способ позволяет очень точно задать тактовую частоту микроконтроллера (разброс частот обычно составляет не превышает 0.01%). Такой уровень точности требуется для обеспечения вычисления характеристик внешнего импульсного сигнала. Номиналы емкостей конденсаторов С5 и С6 в данной схеме определяются производителем микроконтроллера для конкретной резонансной частоты кварца. Для проектируемого устройства был использован кварц – 12МГц, и емкость конденсаторов – 0,1 мкФ, что является приемлемым для используемого типа микроконтроллера. Кварцевый резонатор подсоединяется в выводам тактового генератора МК AT89C51 к XTAL1 (19) и XTAL2(18).

На вход сброса RST (9) МК AT89C51 подается RC – цепочка (С12, R1) для обеспечения сброса таймера при запуске МК.

Блок питания выполняет ряд функций:

* Подача напряжения в 5 В на основные узлы устройства (МК, согласующего устройства, ЭУ нагрузкой);
* Содержит элементы устранения помех в цепях питания;
* Содержит источник резервного автономного питания – гальваническая батарея в случае отключения основного питания для поддержания работы таймера. Но включение приборов заработает только при подаче основного напряжения;
* Присутствует стабилизатор напряжения для сглаживания импульсов.

С помощью ключа К1 в схеме регулируется источник питания: либо напряжение 5 В, либо автономный резервный источник питания – гальваническая батарея. Также в блоке питания присутствует стабилизатор напряжения L7805 для сглаживания импульсов напряжения, выбор которого обуславливается высокой точностью и качеством стабилизации (сглаживания), а также низкой потребляемой мощностью и отсутствием необходимости в каких-либо дополнительных электро-радио элементах, кроме сглаживающих конденсаторов (С1, С2, С3, С4). Последние две цифры обозначения показывают выходное напряжение прибора (5 В).

Для согласования уровней напряжений между COM-портом и микроконтроллером использовалась микросхема MAXIM MAX233, предназначенная специально для такого согласования и имеющая встроенные средства для такого преобразования уровней. Работая от стандартного напряжения 5 В, она преобразует его в +14/-14 В, необходимые интерфейсу RS-232.

MAX233 – универсальный приемопередатчик, предназначенный для сопряжения микропроцессорных элементов, для работы которых требуется напряжение 5 В и устройств с протоколом, имеющим опорное напряжение +14/-14 В. Для питания этой микросхемы необходимы те же 5 В, от которых питается микроконтроллер Atmel AT89C51, что во многом обусловило наш выбор именно этого прибора. Его мощность составляет 890мВт. Подстройка работы MAX233 осуществляется с помощью дополнительных внешних конденсаторов C7, С8, С9, С10.

В данной работе необходимо осуществить передачу данные (начальное время, расписание работы нагрузок) с компьютера на МК через COM-порт по каналу последовательной связи с интерфейсом RS-232C. Это двухуровневая линия связи, на COM-порте будет задействован разъем TxD (3) и GND (5). Канал связи будет соединяться с МК Atmel AT89C51 через порт UART (P3.0 – RxD). В канале связи от МAX232 до COM-порта используются дополнительные конденсатор и резистор C11 и R2.

На вход RxD (P3.0) МК Atmel AT89C51 поступают внешние управляющие сигналы от ПЭВМ по последовательному каналу связи с интерфейсом RS-232C, посредством которых задаётся режим работы микроконтроллера и устанавливается время и расписания. Также на микроконтроллер поступает питающее напряжение от блока питания.

При совпадении текущего времени и одного из времен в одном из заданных расписаний микроконтроллер выдаёт сигнал высокого (для включения электроприбора, при совпадении текущего времени с временем включения данного прибора в расписании) или низкого (для выключения электроприбора, при совпадении текущего времени с временем выключения данного прибора в расписании) уровня с соответствующего бита порта (P2.0..P2.7 для приборов с 1-го по 8-й соответственно).

После подачи пользователем сигнала выбора режима работы микроконтроллера на последовательный интерфейс, микроконтроллер переходит в режим ожидания ввода остальных данных соответствующих данному режиму (для режимов установки времени и установки расписания).

В элементе управления нагрузкой в качестве элемента переключения будем использовать симистор ВТ139 - устройство для коммутации нагрузки на переменном токе, встречно параллельно соединенные два тиристора. На положительной полуволне синусоиды пропускает ток один тиристор, на отрицательной - другой тиристор. Причем пропускают только при наличии управляющего сигнала. Если сигнал управления снять, то на следующем же периоде оба тиристора замкнутся и цепь оборвется. Часто используют для управления бытовой нагрузкой. Но в ЭУ нагрузкой коммутируем мы силовую высоковольтную цепь, 220 вольт, а МК у нас низковольтный, работает на пять вольт. Поэтому во избежание эксцессов нужно произвести потенциальную развязку. То есть сделать так, чтобы между высоковольтной и низковольтной частью не было прямого электрического соединения. Например, сделать оптическое разделение. Для этого существует специальная сборка — симисторный оптодрайвер MOC3051.

**2.4. Устранение помех в цепях питания**

Для уменьшения наводок и падений напряжений, связанных с переходными процессами в шинах питания и земли применены 1 сглаживающий конденсатор емкостью 0,1 мкФ на 3-4 корпуса МС. Конденсаторы подключаются между шинами питания и земли и устанавливаются в непосредственной близости от обслуживаемых корпусов.

Кроме того, для компенсации бросков тока в системе питания непосредственно около разъема подачи питания установлен электролитический конденсатор С1 ёмкостью 100мкФ.

1. **Расчет потребляемой мощности**

Суммарная мощность, которая потребляет устройство есть суперпозиция мощностей элементов, из которых это устройство состоит. Разрабатываемое устройство – печатная плата, на которой располагаются ИМС и ЭРИ. Следовательно, суммарная максимальная мощность, потребляемая устройством РΣmax, может быть вычислена по следующей формуле:

РΣМАХ = РΣИС + РΣЭРЭ ,

где РΣИС – суммарная мощность, потребляемая интегральными микросхемами;

РΣЭРЭ – суммарная мощность, рассеиваемая электрорадиоэлементами.

Мощность ИМС состоит из двух составляющих: мощность в статическом режиме и мощность в динамическом режиме. Первая составляющая есть произведение напряжения питания и тока, который протекает через микросхему. В документации эти параметры обычно обозначаются, как Vcc и Icc. Вторая составляющая имеет место, когда схема имеет частые переключения и прямо пропорциональна частоте переключений. Так как частота переключений мала, этой составляющей можно пренебречь. Мощность, потребляемая интегральными микросхемами:

РΣИС = РΣИСстат + РΣИСдин ,

где РΣИСстат – статическая мощность;

РΣИСдин – динамическая мощность.

Таблица 7. Значения потребляемый мощностей

|  |  |
| --- | --- |
| Устройство | Потребляемая мощность, мВт |
| AT89С51 | 75 |
| MAX232 | 890 |
| MOC3051 | 300 |
| ВТ139 | 40 |
| L7805 | 30 |
| Резисторы | 125 |

Для проектируемого устройства составляющие суммарной потребляемой мощности будут считаться следующим образом:

Мощность, рассеиваемая ЭРЭ (резисторы имеют фиксированную рассеиваемую мощность): РΣR = 26шт \* 0,125Вт = 3,25 Вт

Р∑имс = РL7805 + РMAX233 + РАТ90S8535

В статическом режиме мощность ИМС: Р∑имс = 75 + 890 + 300\*8 + 40\*8 + 30 = 3,71 Вт;

Данный расчёт является приблизительным, поскольку работа схемы осуществляется в асинхронном режиме, и в каждый момент времени существует большое количество неактивных цепей.

Суммарная максимальная мощность, потребляемая микроконтроллерной системой:

Рмах = Р∑имс + РR = 3,71 + 3,25 = 6,96 Вт

1. **Описание алгоритмов программы**

Рассмотрим алгоритм основной программы. Всю программу логически можно разделить на 3 части:

* отсчет времени (1 неделя);
* проверка включения/выключения прибора (каждую секунду);
* прием данных по последовательному каналу.

Алгоритм блока отсчета времени приведен на рисунке 14.

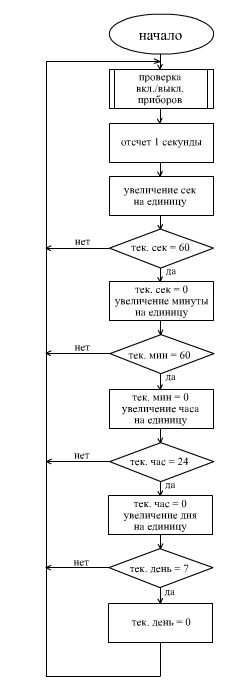


Рис. 14. Алгоритм блока отсчета времени

Каждую секунду в цикле мы вначале переходим к блоку проверки вкл./выкл. приборов. Далее начинаем сам отсчет времени. В соответствии с ТЗ нам необходимо разработать УК с вводом расписания на неделю. Для этого в программе предусмотрен отсчет времени сроком в неделю. После отсчета 1 секунды мы увеличиваем значение текущей секунды на единицу и сравниваем это значение с 60. Если текущих секунд не 60, то мы возвращаемся в начало цикла: проверяем вкл./выкл. приборов и отсчитываем новую секунду. Если текущая секунда равна 60, то обнуляем это значение, увеличиваем значение текущих минут на единицу и сравниваем значение текущих минут с 60. Алгоритм подсчета и проверки минут, часов, дней аналогичен описанному алгоритму, но значение текущих часов сравнивается с 24, а значение текущих дней сравнивается с 7. После того, как было отсчитано время, протяженностью в неделю, все текущие значения обнуляются и отсчет начинается заново.

Алгоритм блока проверки включения/выключения приборов приведен на рисунке 15.

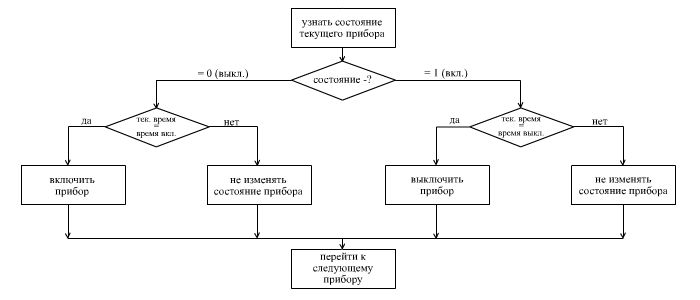


Рис. 15. Алгоритм проверки вкл./выкл. приборов

В начале данного блока мы должны узнать состояние текущего прибора, т.е. находится ли он во включенном или выключенном состоянии. Напомним, что приборы подключены к порту P2, сигналы на котором и показывают данное состояние. Если прибор включен, т.е. на соответственном выводе порта 1, мы должны узнать, надо ли его выключить или оставить включенном. Для этого надо сравнить текущее время с временем выключения данного прибора. Если время совпадает, то прибор выключаем. Если время не совпадает, то состояние прибора не меняем. Аналогично с выключенным прибором. Мы должны узнать, надо ли его включить или оставить в прежнем выключенном состоянии. Для этого также надо сравнить текущее время с временем включения данного прибора. Если время совпадает, то прибор необходимо включить, если же не совпадает – состояние прибора оставляем неизменным. После этого переходим к следующему прибору и повторяем все описанные действия. Как все приборы будут опрошены, мы возвращаемся в цикл блока отсчета времени.

Алгоритм блока приема данных по последовательному каналу приведен на рисунке 16.

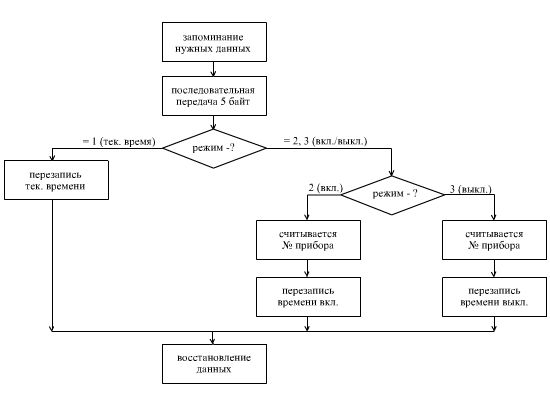


Рис. 16. Алгоритм приема данных по последовательному каналу

Перед обработкой прерывания необходимо запомнить ряд данных, которые понадобятся после выполнения обработки прерывания. Далее осуществляется последовательный прием 5 байт от ПЭВМ по последовательному каналу ввода/вывода. После необходимо определить, какой режим использовался. При режиме 01 происходит установка начального времени, т.е. перезапись текущего. При режиме 02 происходит перезапись времени включения выбранного прибора, а при режиме 03 происходит перезапись времени выключения выбранного прибора. После перезаписи данных необходимо восстановить ряд данных и осуществить возврат к основной программе.

1. **Распределение адресного пространства** **МК Atmel AT89C51**

Память программ и память данных, размещенные на кристалле MCS-51 физически и логически разделены, имеют различные механизмы адресации, работают под управлением различных сигналов и выполняют разные функции.

В микроконтроллерах с архитектурой MCS-51 используется память двух типов:

* постоянная (ROM) - память программ предназначена для хранения команд (программ), управляющих функционированием микроконтроллера, хранения различных констант, не меняющихся во время работы программы. Память программ представляет собой электрически стираемое ППЗУ (Flash-ПЗУ). В МК Atmel AT89C51 резидентная память программ имеет объем 4 Кбайта, но может быть расширена до 64 Кбайт.
* оперативная (RAM), применяемая для хранения данных. В МК Atmel AT89C51 резидентная память данных имеет объем 128 байт. Память данных (ОЗУ) предназначена для хранения переменных в процессе выполнения прикладной программы, адресуется одним байтом и имеет емкость 128 байт. Кроме того, к адресному пространству РПД примыкают адреса регистров специальных функций (SFR), которые перечислены в таблице 8.

Память программ, так же как и память данных, может быть расширена до 64 Кбайт.

Таблица 8. Адреса регистров специальных функций

|  |  |  |
| --- | --- | --- |
| Символ | Наименование | Адрес |
| **\*** ACC | Аккумулятор | 0E0H |
| **\*** B | Регистр-расширитель аккумулятора | 0F0H |
| **\*** PSW | Слово состояния программы | 0D0H |
| SP | Регистр-указатель стека | 81H |
| DPTR | Регистр-указатель данных (DPH) | 83H |
| **\*** P0 | Порт 0 | 82H |
| **\*** P1 | Порт 1 | 80H |
| **\*** P2 | Порт 2 | 90H |
| **\*** P3 | Порт 3 | 0A0H |
| **\*** IP | Регистр приоритетов | 0B0H |
| **\*** IE | Регистр маски прерываний | 0B8H |
| TMOD | Регистр режима таймера/счетчика | 0A8H |
| **\*** TCON | Регистр управления/статус таймера | 89H |
| TH0 | Таймер 0 (старший байт) | 88H |
| TL0 | Таймер 0 (младший байт) | 8CH |
| TH1 | Таймер 1 (старший байт) | 8AH |
| TL1 | Таймер 1 (младший байт) | 8DH |
| **\*** SCON | Регистр управления приемопередатчиком | 8BH |
| SBUF | Буфер приемопередатчика | 98H |
| PCON | Регистр управления мощностью | 99H |

Память данных с диапазоном адресов 20h – 7fh может быть использована для хранения переменных и данных. В программе для МК используются переменные для хранения дня, часа, минуты включения и выключения восьми нагрузок, а также переменные для хранения текущего дня, часа, минуты, секунды.

Переменные для текущего времени:

CURR\_DAY equ 20h

CURR\_HOUR equ 21h

CURR\_MIN equ 22h

CURR\_SEC equ 23h

Переменные для расписания работы 1 нагрузки:

D0\_DAY equ 24h

D0\_HOUR equ 25h

D0\_MIN equ 26h

D0\_DAY\_OFF equ 27h

D0\_HOUR\_OFF equ 28h

D0\_MIN\_OFF equ 29h

Переменные для расписания работы 2 нагрузки:

D1\_DAY equ 2Ah

D1\_HOUR equ 2Bh

D1\_MIN equ 2Ch

D1\_DAY\_OFF equ 2Dh

D1\_HOUR\_OFF equ 2Eh

D1\_MIN\_OFF equ 2Fh

Переменные для расписания работы 3 нагрузки:

D2\_DAY equ 30h

D2\_HOUR equ 31h

D2\_MIN equ 32h

D2\_DAY\_OFF equ 33h

D2\_HOUR\_OFF equ 34h

D2\_MIN\_OFF equ 35h

Переменные для расписания работы 4 нагрузки:

D3\_DAY equ 36h

D3\_HOUR equ 37h

D3\_MIN equ 38h

D3\_DAY\_OFF equ 39h

D3\_HOUR\_OFF equ 3Ah

D3\_MIN\_OFF equ 3Bh

Переменные для расписания работы 5 нагрузки:

D4\_DAY equ 3Ch

D4\_HOUR equ 3Dh

D4\_MIN equ 3Eh

D4\_DAY\_OFF equ 3Fh

D4\_HOUR\_OFF equ 40h

D4\_MIN\_OFF equ 41h

Переменные для расписания работы 6 нагрузки:

D5\_DAY equ 42h

D5\_HOUR equ 43h

D5\_MIN equ 44h

D5\_DAY\_OFF equ 45h

D5\_HOUR\_OFF equ 46h

D5\_MIN\_OFF equ 47h

Переменные для расписания работы 7 нагрузки:

D6\_DAY equ 48h

D6\_HOUR equ 49h

D6\_MIN equ 4Ah

D6\_DAY\_OFF equ 4Bh

D6\_HOUR\_OFF equ 4Ch

D6\_MIN\_OFF equ 4Dh

Переменные для расписания работы 8 нагрузки:

D7\_DAY equ 4Eh

D7\_HOUR equ 4Fh

D7\_MIN equ 50h

D7\_DAY\_OFF equ 51h

D7\_HOUR\_OFF equ 52h

D7\_MIN\_OFF equ 53h

Переменные для запоминания данных во время передачи по посл. каналу

MEM\_R0 equ56h

MEM\_BYTE1 equ 57h

MEM\_BYTE2 equ 58h

MEM\_BYTE3 equ 59h

MEM\_BYTE4 equ 5Ah

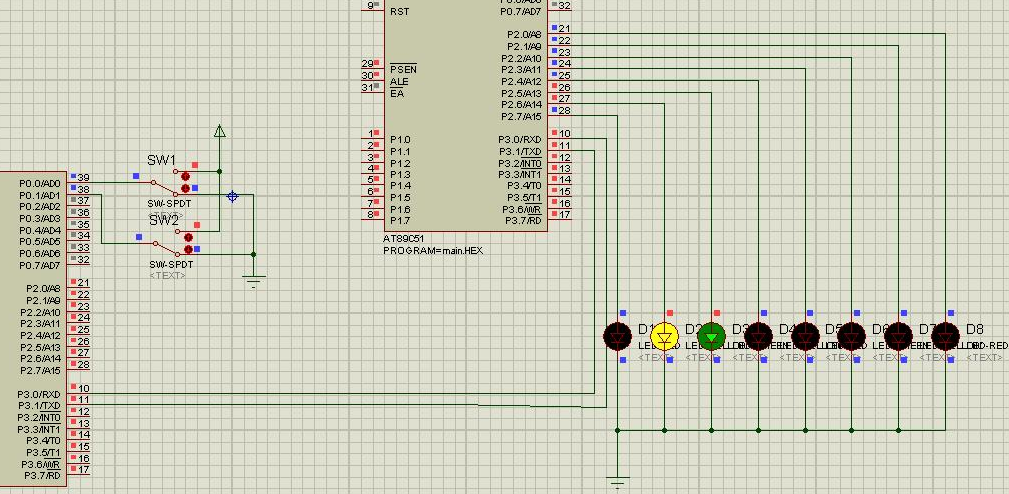
MEM\_BYTE5 equ 5Bh

1. **Тестирование и отладка программы для МК Atmel AT89C51**
   1. **Тестирование и отладка программы в симуляторе Proteus**

Текст программы нельзя полностью отладить в программаторе Keil, так как там нет внешнего источника сигналов. В нем можно увидеть лишь запись констант в регистр специальных функций и отследить работу необходимых разрядов портов. Для отладки используется более удобная программа. Программа Proteus позволяет тестировать работу программы микроконтроллера и периферийных устройств в режиме реального времени. С помощью этой системы отладки легко находятся ошибки в программах и эмулируются все участки алгоритма программы.

В собранную схему в программе Proteus необходимо подгрузить для каждого МК свой файл с расширением .asm и сделать компиляцию. Результат можно просмотреть, нажав кнопку play или с помощью пошагового отладчика.

Для удобства в коде программы записано расписание, при котором каждый прибор включается по очереди и горит по 2 секунды. Результат моделирования приведен на рисунке 17.



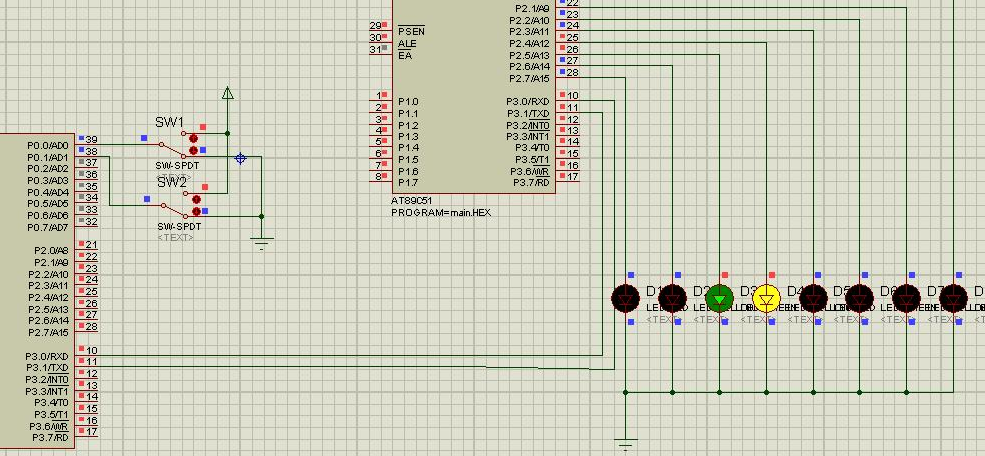


Рис. 17. Результат моделирования

Так же можно отладить программу, использую пошаговую отладку. С помощью нажатия клавиши f10 мы можем проследить запись данных в память и состояние регистров. На рисунке 18 мы видим, что после прохождения кода до строчки 017В в память записались данные по соответствующим адресам, начиная с 24h и заканчивая 53h. Эти адреса отведены для хранения расписания включения/выключения нагрузок и текущего времени.

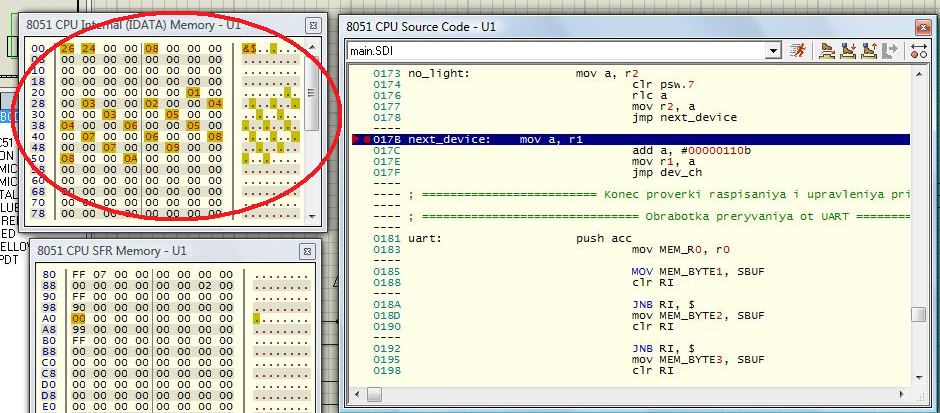


Рис. 18. Пошаговая отладка

Так, с помощью пошаговой отладки дошли до места в коде программы, когда горит 1 и 2 прибор (рисунок 19).

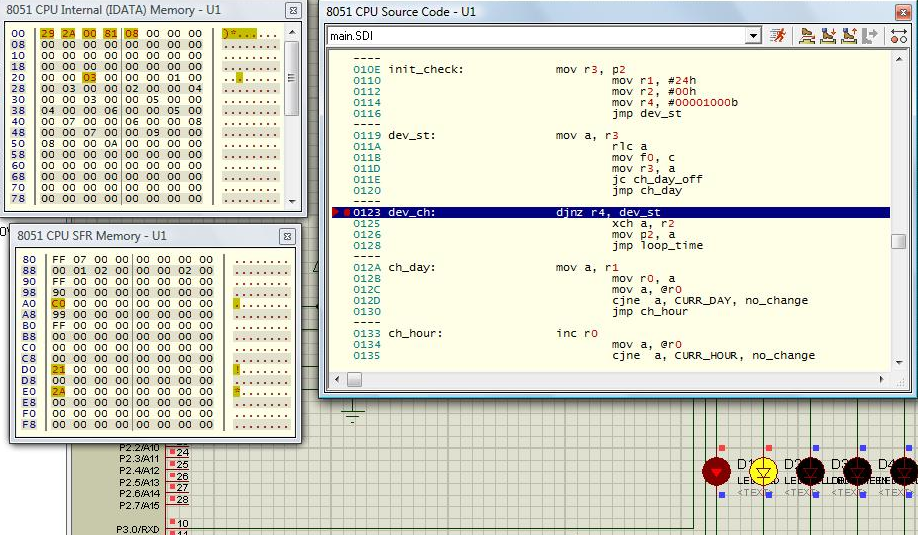


Рис. 19. Пошаговая отладка – продолжение

Так же можно проследить значение регистров в каждый момент времени. Для прошлой ситуации, когда 1 и 2 прибор горят, видно, что в регистре r1 хранится адрес 2 прибора – 2Ah, в r4 хранится количество приборов – 8, состояние порта 2 – С0, что равно 11000000, т.е. горят 1 и 2 прибор и т.д. Значения регистров представлены на рисунке 20.

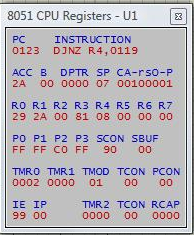


Рис. 20. Состояние регистров

В данной программе мы так же можем проверить работу передачи расписания через последовательный канал ввода/вывода. Для примера, пользователь после того, как все лампочки прогорели, хочет обнулить время и начать включение/выключение нагрузок заново. Для этого он переключает ключ 1 в состояние единицы. С помощью пошаговой отладки мы можем просмотреть передачу байтов по каналу другому МК. На рисунке 21 видно, как в адресе 57h находится значение режима 01 и в адресе 5B находится значение 0F.

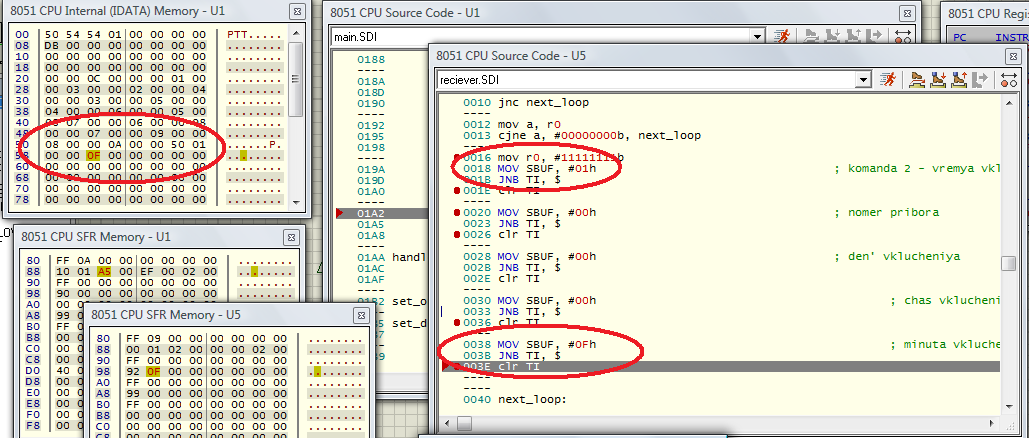


Рис. 21. Отладка передачи данных

В качестве еще одного примера рассмотрим ситуацию, когда включение/выключение нагрузок осталось прежним, а изменилось начальное время. Пусть изначально в коде программы забито время: 7 день (06h), 23 часа (17h), 50 минут (32h) и 0 секунд. Напомним, что для простоты тестирования в кода 1 минута равно 1 секунде. Тогда получается, что при запуске МК система отсчитает 10 мин (10 секунд в нашем случае) и обнулит дни, часы, минуты и секунды. И тогда в 1 минуту (1 секунда в нашем случае) загорится 1 прибор, во 2 минуту – 2 прибор и т.д.

В памяти переменные текущего времени занимают адреса:

CURR\_DAY equ 20h

CURR\_HOUR equ 21h

CURR\_MIN equ 22h

CURR\_SEC equ 23h

Запишем в код программы начальное время (изменится только приведенная часть кода):

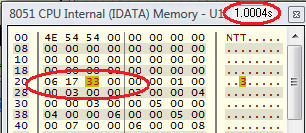
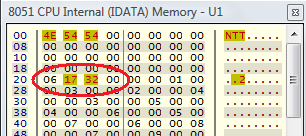
mov CURR\_DAY, #06h

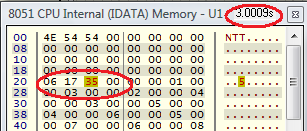
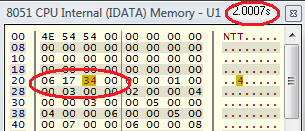
mov CURR\_HOUR, #17h

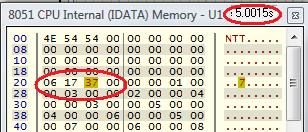
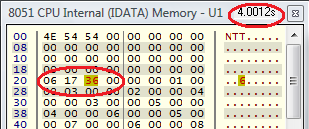
mov CURR\_MIN, #32h

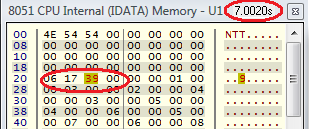
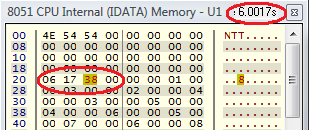
mov CURR\_SEC, #00h

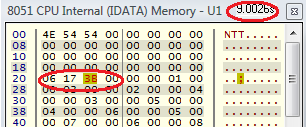
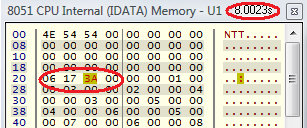
С помощью пошагового отладчика проследим увеличение времени (перезапись данных в памяти каждую секунду), обнуление переменных по истечению 10 секунд и включение 1 нагрузки в 1ую секунду. Результаты на рисунках 22 и 23.

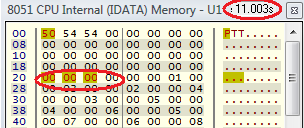
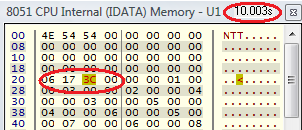












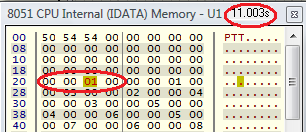


Рис. 22. Состояние памяти при отсчете времени

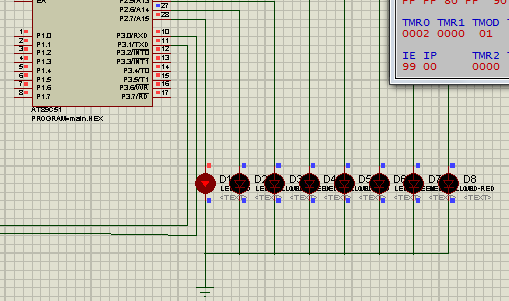


Рис. 23. Включение 1 нагрузки

* 1. **Оценка времени выполнения программы**

В данной работе можно проследить задержку при переключении приборов:

* Время включения 1 прибора - 
* Время включения 2 прибора - 
* Время выключения 1 прибора - 
* Время включения 3 прибора - 
* Время выключения 2 прибора – 
* Время включения 4 прибора - 

Таким образом, мы видим, что задержка включения/выключения приборов совсем незначительная. За неделю задержка набегает в размере 1 секунды. Это меньше 1%, поэтому данной задержкой можно пренебречь.

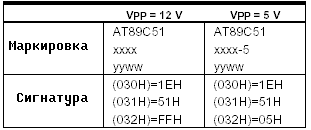
1. **Программирование памяти МК**

При программировании микроконтроллера полученный в результате компиляции программы машинный код загружается в память программ. МК AT89C51 поставляется со встроенным Flash. В первоначальном ("чистом") состоянии по всем адресам во Flash записаны FFh. Микроконтроллер AT89С51 может поддерживать один из двух режимов программирования:

* Режим программирования c высоким напряжением сигнала разрешения(12 В);
* Режим программирования c низким напряжением сигнала разрешения (VCC = 5 В).

Первый режим удобен при использовании специальных программаторов, а второй — при программировании микроконтроллера внутри пользовательской системы. Существует специальная система маркировок для определения напряжения программирования AT89C51. Соответствующая маркировка (она маркируется на лицевой стороне) и коды сигнатуры устройства указаны в таблице 9.

Таблица 9. Маркировка и сигнатуры МК AT89С51



Память в МК AT89C51 программируется побайтно, в любом режиме программирования. Чтобы перепрограммировать любой нечистый байт во Flash, необходимо стереть всю Flash память. Для этого существует Режим "Чип-очистки" (Chip Erase Mode).

Алгоритм программирования:

1. Установить требуемый адрес памяти на линиях адреса.
2. Ввести соответствующую команду или данные по линиям адреса.
3. Активировать правильную комбинацию управляющих сигналов.
4. В режиме высокого напряжения поднять напряжение на входе  до 12 В.
5. Подать импульс на вход  для записи байта в массив Flash‑памяти или Lock-битов. Цикл записи, как правило, занимает не более 1,5 мс.
6. Повторить пункты 1-5, изменяя адрес и данные, пока не запишется вся необходимая информация.

* /*Polling* (опрос данных) - Сигнал в МК, сигнализирующий конец цикла записи. По окончании цикла записи истинные данные доступны на всех выводах, и можно начинать следующий цикл.
* *RDY*/ (готов/занят) - После того, как на ALE подан высокий уровень, прогресс программирования байта можно проверить сигналом *RDY*/ на выводе P3.4, который имеет низкий логический уровень. Во время программирования это сигнализирует ЗАНЯТО. По окончании программирования на P3.4 снова высокий уровень, что сигнализирует о ГОТОВНОСТИ.

Проверка программы:

Если биты блокировки LB1 и LB2 не запрограммированы, программируемые кодовые данные могут быть прочитаны для верификации через адрес и строковые данные. Биты блокировки не могут проверяться непосредственно. Проверка блокирующих битов достигается с соблюдением допустимых особенностей.

Чип-очистка:

Весь Flash-массив электрически стирается, используя соответствующую комбинацию управляющих сигналов и удерживая *ALE/* на низком логическом уровне в течение 10 мс. Это записывает логическую единицу в весь Flash массив. Операция Чип-очистка должна быть выполнена до записи новой программы во Flash.

Чтение байтов сигнатуры:

Если не принимать во внимание то, что P3.6 и P3.7 должны быть переведены в низкий логический уровень, чтение байтов сигнатуры производится так же, как нормальная проверка адресов 030H, 031H, и 032H.

Возвращаются следующие значения:

(030H) = 1EH - показывает, что изготовлено в Atmel

(031H) = 51H - идентифицирует 89C51

(032H) = FFH - 12 V программирование

(032H) = 05H - 5 V программирование

Интерфейс программирования:

Все крупные производители программаторов обычно поддерживают МК фирмы Atmel. Параметры режимов программирования и интерфейс программирования показаны на рисунках 24, 25 и в таблице 9.

Таблица 9. Режимы программирования Flash

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| РЕЖИМ | | RST | no-psen<> | ale-no-prog<> | no-ea/VPP | P 2.6 | P 2.7 | P 3.6 | P 3.7 |
| ЗАПИСЬ ДАННЫХ | | H | L | strob | H/12V(1) | L | H | H | H |
| ЧТЕНИЕ ДАННЫХ | | H | L | H | H | L | L | H | H |
| БЛОКИРОВКА ЗАПИСИ | Бит 1 | H | L | strob | H/12V | H | H | H | H |
| Бит 2 | H | L | strob(2) | H/12V | H | H | L | L |
| Бит 3 | H | L | strob | H/12V | H | L | H | L |
| ЧИП-ОЧИСТКА | | H | L | strob | H/12V | H | L | L | L |
| ЧТЕНИЕ БАЙТОВ СИГНАТУРЫ | | H | L | H/12V | H | L | L | L | L |
| *Примечание:* 1. Байт сигнатуры в местоположении 032H определяет какое должно быть напряжение программирования - VPP = 12 V или VPP = 5 V | | | | | *Примечание:* 2. Чип-Очистка, требует 10 ms импульса no-prog<> | | | | |

|  |  |
| --- | --- |
| fig3 | fig4 |
| Рис. 24. Программирование Flash | Рис. 25. Проверка Flash |

Примечание:

На выводы P1.x, подаются младшие сигналы адреса (A0 – A7), а на выводы P2.0 – P2.3 – старшие разряды(A8 – A11) при программировании и проверке Flash-памяти.

При программирование Flash-памяти на порт P0 поступают данные, а при проверке – данные считываются с него с использованием подтягивающих резистров с номиналом 10 кОм.

Временные диаграммы при программировании Flash:

Таблица 10. Параметры

**Таблица 5.** Режимы программирования Flash.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TA = 21°C дo 27°C, VCC = 5.0 ± 10% | | | | | |
| СИМВОЛ | ПАРАМЕТР | MIN | MAX | ЕДИНИЦЫ |  |
| VPP(1) | Напряжение программирования | 11.5 | 12.5 | V |  |
| IPP(1) | Потребляемый ток при программировании | · | 1.0 | mA |  |
| 1/tCLCL | Частота генератора | 4 | 24 | MHz |  |
| tAVGL | Установка адреса к no-prog | 48tCLCL | · | · |  |
| tGHAX | Задержки адреса после no-prog | 48tCLCL | · | · |  |
| tDVGL | Установка данных к no-prog | 48tCLCL | · | · |  |
| tDHDX | Задержки данных после no-prog | 48tCLCL | · | · |  |
| tEHSH | Время переключения P2.7 (no-enable) к VPP | 48tCLCL | · | · |  |
| tSHGL | Установки VPP к no-prog | 10 | · | micro-s |  |
| tGHSL(1) | Задержки VPP после установки no-prog | 10 | · | micro-s |  |
| tGLGH | Ширина импульса no-prog | 1 | 110 | micro-s |  |
| tAVQV | Адресации принятых данных | · | 48tCLCL | · |  |
| tELQV | Время установкиno-enableдля принятых данных | · | 48tCLCL | · |  |
| tGHBL | Переключение от высокого уровняno-prog к низкому уровню no-busy | · | 1.0 | micro-s |  |
| tWC | Время цикла записи байта | · | 2.0 | ms |  |
| *Примечание:* 1. Только при программировании напряжением 12V | | | | |  |

|  |
| --- |
| waveforms-h  Рис. 26 Осциллограммы при программировании и проверке Flash |
|  |
| waveforms-l |
| Рис. 27. Осциллограммы при программировании и проверке Flash |

1. **Заключение**

В результате проектирования был разработан управляющий контроллер на основе семейства MCS-51. Разработанное устройство, благодаря использованию современной элементной базы, отличается повышенной точностью и надежностью. Высокая степень интеграции используемых радиоэлементных средств обеспечивает уменьшение массогабаритных параметров и потребляемой мощности.

Из реализации видно, что современные микроконтроллеры, к числу которых относится использованный контроллер AT89С51 в сочетании с современными языками программирования, средствами разработки, отладки и моделирования позволяют решать достаточно сложные задачи при минимуме затрат.

1. **Список использованных источников информации**
2. Хартов В.Я. Микропроцессорные системы. Учебное пособие. Издательский центр «Академия», 2010 г.
3. Голубцов М.С. Микроконтроллеры: от простого к сложному. М.:САЛОН-ПРЕСС, 2003г.
4. Хартов В.Я. Курс лекций «Микропроцессорные системы», 2010 г.
5. L7800.pdf [http://www.st.](http://www.s.)com Стабилизаторы L78хх, техническое описание
6. MAX233xPP.pdf <http://www.maxim-ic.com> MAX233xPP, техническое описание
7. Предко М. Руководство по микроконтроллерам. Том 1. Москва: Постмаркет, 2001. – 416с.
8. Хартов В.Я. Микроконтроллеры AVR. Практикум для начинающих. Москва. Издательство МГТУ им. Н.Э. Баумана, 2010 г.

**Приложение 1. Листинг программы**

; Переменные для текущего времени

CURR\_DAY equ 20h

CURR\_HOUR equ 21h

CURR\_MIN equ 22h

CURR\_SEC equ 23h

; Переменные для задания работы нагрузки 1

D0\_DAY equ 24h

D0\_HOUR equ 25h

D0\_MIN equ 26h

D0\_DAY\_OFF equ 27h

D0\_HOUR\_OFF equ 28h

D0\_MIN\_OFF equ 29h

; Переменные для задания работы нагрузки 2

D1\_DAY equ 2Ah

D1\_HOUR equ 2Bh

D1\_MIN equ 2Ch

D1\_DAY\_OFF equ 2Dh

D1\_HOUR\_OFF equ 2Eh

D1\_MIN\_OFF equ 2Fh

; Переменные для задания работы нагрузки 3

D2\_DAY equ 30h

D2\_HOUR equ 31h

D2\_MIN equ 32h

D2\_DAY\_OFF equ 33h

D2\_HOUR\_OFF equ 34h

D2\_MIN\_OFF equ 35h

; Переменные для задания работы нагрузки 4

D3\_DAY equ 36h

D3\_HOUR equ 37h

D3\_MIN equ 38h

D3\_DAY\_OFF equ 39h

D3\_HOUR\_OFF equ 3Ah

D3\_MIN\_OFF equ 3Bh

; Переменные для задания работы нагрузки 5

D4\_DAY equ 3Ch

D4\_HOUR equ 3Dh

D4\_MIN equ 3Eh

D4\_DAY\_OFF equ 3Fh

D4\_HOUR\_OFF equ 40h

D4\_MIN\_OFF equ 41h

; Переменные для задания работы нагрузки 6

D5\_DAY equ 42h

D5\_HOUR equ 43h

D5\_MIN equ 44h

D5\_DAY\_OFF equ 45h

D5\_HOUR\_OFF equ 46h

D5\_MIN\_OFF equ 47h

; Переменные для задания работы нагрузки 7

D6\_DAY equ 48h

D6\_HOUR equ 49h

D6\_MIN equ 4Ah

D6\_DAY\_OFF equ 4Bh

D6\_HOUR\_OFF equ 4Ch

D6\_MIN\_OFF equ 4Dh

; Переменные для задания работы нагрузки 8

D7\_DAY equ 4Eh

D7\_HOUR equ 4Fh

D7\_MIN equ 50h

D7\_DAY\_OFF equ 51h

D7\_HOUR\_OFF equ 52h

D7\_MIN\_OFF equ 53h

; Переменные для запоминания данных во время передачи по посл. каналу

MEM\_R0 equ56h

MEM\_BYTE1 equ 57h

MEM\_BYTE2 equ 58h

MEM\_BYTE3 equ 59h

MEM\_BYTE4 equ 5Ah

MEM\_BYTE5 equ 5Bh

; При сбросе МК программный счетчик принимает значение 00, надо осуществить переход к основной программе

org 0h

jmp start

org 23h

jmp uart ; адрес таблицы прерываний для прерывания от UART

; Основная программа

start:

; Разрешаем прерывания от последовательного канала, используя регистр маски прерывания IE

mov ie, #10011001b

mov SCON,#10010000b ; задаем режим2 работы порта UART через регистр управления SCON

; Вводим для примера расписание работы нагрузок: 1мин – вкл. 1ая нагр., 2 мин – вкл. 2ая нагр., 3 мин – 1ая нагр. выкл., а 3 нагр. вкл., 4 мин – 2ая нагр. выкл, а 4 нагр. вкл. и т.д.

mov D0\_DAY, #00h

mov D0\_HOUR, #00h

mov D0\_MIN, #01h

mov D0\_DAY\_OFF, #00h

mov D0\_HOUR\_OFF, #00h

mov D0\_MIN\_OFF, #03h

mov D1\_DAY, #00h

mov D1\_HOUR, #00h

mov D1\_MIN, #02h

mov D1\_DAY\_OFF, #00h

mov D1\_HOUR\_OFF, #00h

mov D1\_MIN\_OFF, #04h

mov D2\_DAY, #00h

mov D2\_HOUR, #00h

mov D2\_MIN, #03h

mov D2\_DAY\_OFF, #00h

mov D2\_HOUR\_OFF, #00h

mov D2\_MIN\_OFF, #05h

mov D3\_DAY, #00h

mov D3\_HOUR, #00h

mov D3\_MIN, #04h

mov D3\_DAY\_OFF, #00h

mov D3\_HOUR\_OFF, #00h

mov D3\_MIN\_OFF, #06h

mov D4\_DAY, #00h

mov D4\_HOUR, #00h

mov D4\_MIN, #05h

mov D4\_DAY\_OFF, #00h

mov D4\_HOUR\_OFF, #00h

mov D4\_MIN\_OFF, #07h

mov D5\_DAY, #00h

mov D5\_HOUR, #00h

mov D5\_MIN, #06h

mov D5\_DAY\_OFF, #00h

mov D5\_HOUR\_OFF, #00h

mov D5\_MIN\_OFF, #08h

mov D6\_DAY, #00h

mov D6\_HOUR, #00h

mov D6\_MIN, #07h

mov D6\_DAY\_OFF, #00h

mov D6\_HOUR\_OFF, #00h

mov D6\_MIN\_OFF, #09h

mov D7\_DAY, #00h

mov D7\_HOUR, #00h

mov D7\_MIN, #08h

mov D7\_DAY\_OFF, #00h

mov D7\_HOUR\_OFF, #00h

mov D7\_MIN\_OFF, #0Ah

mov CURR\_DAY, #00h

mov CURR\_HOUR, #00h

mov CURR\_MIN, #00h

mov CURR\_SEC, #00h

mov p2, #00h ; выключение всех нагрузок: 0 – выкл., 1 – вкл.

**; основной цикл**

loop:

jmp init\_check ; проверка надо ли вкл./выкл. Какую-нибудь нагрузку

; отсчет секунды

; 1 сек = 20\*50000мкс, таймер маx = 65535мкс (65535-15535=50000, т.е. надо задать 15535=3СAF, но с учетом задержки на выполнение операций задаем чуть больше 3СBB (15547)

**;отсчет времени**

loop\_time:

mov r3, #00010100b ; в r3 записываем 20

timer:

mov tl0, #0BBh ; запись числа 3CBB в младшую половину таймера

mov th0, #03Ch ; запись числа 3CBB в старшую половину таймера

mov tmod, #1 ; запись в регистр TMOD 00000001 – 1 режим

setb tcon.4 ; запись в бит управления TR0 1 регистра TCON – запуск таймера 0

l1: jnb tcon.5, l1 ; цикл крутится пока бит переполнения TF0 регистра TCON не равен 1 (отсчет 50000), как станет 1 , переход на след. строчку

clr tcon.4 ; запись в бит управления TR0 0 регистра TCON – остановка таймера 0

clr tcon.5 ;бит переполнения TF0 регистра TCON обнулен

djnz r3, timer ; уменьшаем r3 и сравниваем с 0, если не 0 – считаем дальше timer, если 0 – таймер отсчитал сек

inc CURR\_SEC ; увеличиваем текущую сек

mov a, CURR\_SEC

cjne a, #0000001b, loop ; сравниваем аккумулятор с 00000001 (по идее вместо 00000001 надо 00111100=60 сек в минуте) если не равно переходим на loop и считаем след. секунду

mov CURR\_SEC, #00h ; обнуляем секунды

inc CURR\_MIN ; увеличиваем минуту

mov a, CURR\_MIN

cjne a, #000111100b, loop ; сравниваем аккумулятор с 00111100 (00111100=60 мин в часе) если не равно переходим на loop и считаем след. секунду

mov CURR\_MIN, #00h ; обнуляем минуту

inc CURR\_HOUR ; увеличиваем час

mov a, CURR\_HOUR

cjne a, #00011000b, loop ; сравниваем аккумулятор с 00011000 (00011000=24 часа в дне) если не равно переходим на loop и считаем след. секунду

mov CURR\_HOUR, #00h ; обнуляем час

inc CURR\_DAY ; увеличиваем день

mov a, CURR\_DAY

cjne a, #00000111b, loop ; сравниваем аккумулятор с 00000111 (00000111=7 дней в неделе) если не равно переходим на loop и считаем след. секунду

mov CURR\_DAY, #00h

jmp loop

**; проверка вкл./выкл.**

; r3 – для текущего состояния порта2

; r1 – для адреса прибора

; r2 – для нового состояния прибора

; r4 – кол-во приборов 8

init\_check:

mov r3, p2 ; состояния порта2

mov r1, #24h ; адрес 8ого прибора (день вкл.) ;

mov r2, #00h ; новое состояние прибора

mov r4, #00001000b ; 8 приборов

jmp dev\_st

dev\_st:

mov a, r3

rlc a ; сдвиг а влево через перенос в бит С регистра PSW

mov f0, c ; в f0 состояние прибора текущего

mov r3, a ; перезапись r3 со сдвигом

jc ch\_day\_off ; если вкл., надо ли выкл. или оставить? ( переход при =1)

jmp ch\_day ; если выкл., надо ли вкл. или оставить? (переход при =0)

dev\_ch:

djnz r4, dev\_st ; уменьшаем r4 и сравниваем с 0, если не 0, то не все приборы опрошены и переходим по метке, если 0, то вывод новых состояний проборов на порт2

xch a, r2

mov p2, a

jmp loop\_time

**; прибор выкл. – надо вкл. или оставить?**

ch\_day:

mov a, r1

mov r0, a

mov a, @r0 ; в а значение то, что находится по адресу (день вкл. текущего прибора)

cjne a, CURR\_DAY, no\_change ; если не равно, то не менять состояние прибора

jmp ch\_hour ; если равно, то проверка совпадения часа

ch\_hour:

inc r0 ; переход на адрес часа вкл. текущего прибора

mov a, @r0

cjne a, CURR\_HOUR, no\_change ;если не равно, то не менять состояние прибора

jmp ch\_min ;если равно, то проверка совпадения минуты

ch\_min:

inc r0; переход на адрес мин вкл. текущего прибора

mov a, @r0

cjne a, CURR\_MIN, no\_change ;если не равно, то не менять состояние прибора

jmp light ;если равно, то меняем значение состояния прибора - зажигаем

**; прибор вкл. – надо выл. или оставить?**

ch\_day\_off:

mov a, r1

mov r0, a

inc r0

inc r0

inc r0; переход на адрес дня выкл. текущего прибора

mov a, @r0 в а значение то, что находится по адресу (день выкл. текущего прибора)

cjne a, CURR\_DAY, no\_change change ;если не равно, то не менять состояние прибора

jmp ch\_hour\_off ;если равно, то проверка совпадения часа

ch\_hour\_off:

inc r0; переход на адрес часа выкл. текущего прибора

mov a, @r0

cjne a, CURR\_HOUR, no\_change ;если не равно, то не менять состояние прибора

jmp ch\_min\_off ;если равно, то проверка совпадения минуты

ch\_min\_off:

inc r0; переход на адрес минуты выкл. текущего прибора

mov a, @r0

cjne a, CURR\_MIN, no\_change;если не равно, то не менять состояние прибора

jmp no\_light ;если равно, то меняем значение состояния прибора – выключаем

**; без изменений**

no\_change:

mov a, r2

clr psw.7 ; обнуляем бит с регистра PSW

rlc a ; сдвиг влево а

mov c, f0 ; в с текущее состояние прибора

addc a, #00h; а = а + 00h + с

mov r2, a ; запись 0 в новое состояние текущего прибора

jmp next\_device

**; зажигаем**

light:

mov a, r2

clr psw.7

rlc a

mov r2, a

inc r2 ; запись 1 в состояние прибора

jmp next\_device

no\_light:

mov a, r2

clr psw.7

rlc a

mov r2, a

jmp next\_device

next\_device:

mov a, r1

add a, #00000110b ; а = а + 6 – адрес следующего прибора

mov r1, a

jmp dev\_ch

**; Передача по UART**

uart: push acc ; записываем в стек значение а

mov MEM\_R0, r0 ; запоминаем значение регистра r0

MOV MEM\_BYTE1, SBUF ; перенос из буфера в переменную 1ого полученного байта

clr RI ; чистим флаг прерывания приемника

JNB RI, $ ;ожидаем приема №прибора или тек. дня (ждем установки RI в 1)

mov MEM\_BYTE2, SBUF ; из буфера переносим №прибора или тек. день

clr RI ; чистим флаг прерывания приемника JNB RI, $ ;ожидаем приема дня вкл./выкл. или тек. часа

mov MEM\_BYTE3, SBUF ; перенос из буфера полученного значения

clr RI ; чистим флаг прерывания приемника

JNB RI, $ ;ожидаем приема часа вкл./выкл. или тек. мин

mov MEM\_BYTE4, SBUF ; перенос из буфера полученного значения

clr RI ; чистим флаг прерывания приемника JNB RI, $ ;ожидаем приема мин вкл./выкл. или не используется

mov MEM\_BYTE5, SBUF; перенос из буфера полученного значения clr RI ; чистим флаг прерывания приемника

**; Обработка поступивших данных**

; 01 – режим установки тек. Времени

; 02 – режим задания времени вкл. прибора

; 03 – режим задания времени выкл. прибора

handler: mov a, MEM\_BYTE1 ; в а записываем режим работы (код команды)

cjne a, #00000001b, set\_on\_off ; если не 1, то управление прибором

jmp set\_curr\_time ; если 1, то режим установки тек. Времени

**; управление приборами**

set\_on\_off: cjne a, #00000010b, set\_dev\_off ; если не 2, то выкл. прибора

**; вкл. прибора**

set\_dev\_on: mov r0, MEM\_BYTE2 ; запись №прибора в r0

mov a, #1Eh

l2: add a, #06h ; изначально -6, далее прибавляем по 6 столько раз, каков номер прибора, чтобы попасть на нужный адрес

djnz r0, l2

mov r0, a ; запоминаем адрес в r0

mov @r0, MEM\_BYTE3 ; перезапись дня вкл.

inc r0 ; переход на адрес часа вкл.

mov @r0, MEM\_BYTE4 ; перезапись часа вкл.

inc r0 ; переход на адрес мин вкл.

mov @r0, MEM\_BYTE5 ; перезапись мин вкл.

jmp set\_end ; переход на восстановление системы

**; выкл. прибора**

set\_dev\_off: mov r0, MEM\_BYTE2 ; запись №прибора в r0

mov a, #21h

l3: add a, #06h; изначально -6, далее прибавляем по 6 столько раз, каков номер прибора, чтобы попасть на нужный адрес

djnz r0, l3 mov r0, a ; запоминаем адрес в r0

mov @r0, MEM\_BYTE3 ; перезапись дня выкл.

inc r0 ; переход на адрес часа выкл.

mov @r0, MEM\_BYTE4 ; перезапись часа выкл.

inc r0 ; переход на адрес мин вкл.

mov @r0, MEM\_BYTE5 ; перезапись мин вкл.

jmp set\_end ; переход на восстановление системы

**; режим установки тек. времени**

set\_curr\_time: mov CURR\_DAY, MEM\_BYTE2; перезаписываем день

mov CURR\_HOUR, MEM\_BYTE3 ; перезаписываем час

mov CURR\_MIN, MEM\_BYTE4 ; перезаписываем мин

**; восстановление системы**

set\_end: mov r0, MEM\_R0 ; восстанавливаем r0

pop acc ; из стека забираем а

reti ; возврат к программе

end

**Код для 2ого МК (передатчика)**

; При сбросе МК программный счетчик принимает значение 00, надо осуществить переход к основной программе

org 0h

jmp start

**; Начало программы**

start: mov SCON,#10010000b ; задаем режим2 работы порта UART через регистр управления SCON

mov ie, #10011001b; Разрешаем прерывания от последовательного канала, используя регистр маски прерывания IE

mov r0, #00000000b ; r0 – флаг, когда 0 – надо передавать, 1 – передача завершена

**; основной цикл (1ый ключ)**

loop: clr c

mov a, p0 ; в а состояние порта 0

rrc a ; сдвиг а вправо

mov r1, a ; перезапись а в r1 с изменениями

jnc next\_loop

mov a, r0

cjne a, #00000000b, next\_loop ; если с=0, то переход по метке (см. 2ой ключ)

mov r0, #11111111b ; идет передача, записываем во флаг 1

MOV SBUF, #01h ; режим 1 – установка тек. времени

JNB TI, $ ; успешная передача

clr TI ; обнуляем флаг переполнения передатчика

MOV SBUF, #00h ; передача дня

JNB TI, $

clr TI

MOV SBUF, #00h ; передача часа

JNB TI, $

clr TI

MOV SBUF, #00h ; передача мин

JNB TI, $

clr TI

MOV SBUF, #0Fh ; не используется

JNB TI, $

clr TI

**; проверка 2ого ключа**

next\_loop:

mov a, r1

rrc a

jnc no\_reload ; если с=0, то нет передачи

mov r0, #00000000b ; если с=1, то будем передавать заново

**; нет передачи**

no\_reload:

call delay ; задержка 1 секунда

jmp loop

**; задержка 1 секунда**

delay:

mov r3, #00010100b

timer:

mov th0, #00111100b ;

mov tl0, #10111011b ;

mov tmod, #1

setb tcon.4

l1: jnb tcon.5, l1

clr tcon.4

clr tcon.5

djnz r3, timer

ret

end