МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

Технологический институт
Федерального государственного образовательного учреждения высшего профессионального образования
«Южный федеральный университет»

ПРИОРИТЕТНЫЙ НАЦИОНАЛЬНЫЙ ПРОЕКТ «ОБРАЗОВАНИЕ»

КАФЕДРА КОНСТРУИРОВАНИЯ ЭЛЕКТРОННЫХ СРЕДСТВ

КОНСПЕКТ ЛЕКЦИЙ

по курсам

Проектирование центральных и периферийных устройств ЭВС,
Микропроцессоры и ЭВМ в микросистемах

Для студентов специальностей 210202, 210108

ФЭП

Таганрог 2009

Лукьяненко Е.Б.
Конспект лекций по курсам «Проектирование центральных и периферийных устройств ЭВС, Микропроцессоры и ЭВМ в микросистемах». 2009. – 105 с.: ил.

В 1-й части конспекта рассматриваются запоминающие устройства, различные виды памяти и программирование памяти. Во 2-й части конспекта рассматриваются микроконтроллеры AVR. Изложены основные принципы функционирования и особенности архитектуры микроконтроллеров Atmel AVR. Подробно описаны приемы программирования микросхем этого семейства и их отдельных узлов. Вы найдете готовые рецепты для программирования большинства основных функций современной микроэлектронной аппаратуры: от реакции на нажатие кнопки или построения динамической индикации до сложных протоколов записи данных во внешнюю память или особенностей подключения часов реального времени. В книге учтены особенности современных моделей AVR и сопутствующих микросхем последних лет выпуска. Особое внимание уделяется обмену данными микроэлектронных устройств с персональным компьютером, основные параметры микроконтроллеров AVR, перечень команд и тексты программ для них, а также список используемых терминов и аббревиатур. Дополнением к конспекту является сборник лабораторных работ по этому же курсу: Таганрог: Изд-во ТТИ ЮФУ, 2008. – 41с., №4224.

Терминология

1. Процессор, ЦПУ (центральное процессорное устройство) - состоит из АЛУ (арифметико-логического устройства) и устройства управления.
2. Микропроцессор (МП) – БИС, состоящая из процессора, запоминающего устройства (ЗУ) и устройства ввода-вывода.
3. МикроЭВМ – имеет в своем составе один или несколько процессоров, БИС ПЗУ и ОЗУ, БИС управления вводом-выводом.
4. Контроллер это микроЭВМ с небольшими вычислительными ресурсами и упрщенной системой команд, ориентированный на выполнение логического управления различным оборудованием.

Области применения контроллеров

1. Изделия и услуги массового потребления: бытовые электроприборы (кондиционер, швейная машина и т.п.), досуг и развлечения (игровые автоматы, аудио- и видеотехника и т.п.).
2. Промышленные установки и приборы: станки и оборудование с цифровым управлением, роботы, управление технологическими процессами.
3. Измерительная техника, испытательное оборудование: измерительные приборы, испытательные стенды, экспериментальные установки.
4. Автомобильная промышленность: системы управления автомобильными двигателями, автоматические бензоколонки.
5. Бизнес и торговля: машинки для отсчета купюр, кассовые аппараты.

Области применения универсальных ЭВМ

1. Научно-технические расчеты: математические расчеты, статистика, математическое моделирование, расчет орбит искусственных спутников Земли.
2. Финансы, коммерция: расчет зарплаты, банковское дело, электронная коммерция.
3. Производство и управление: автоматизация производства, системы компьютерного проектирования.
4. Информационная техника и телекоммуникации: компьютерные сети, спутниковая связь, оплата телефонных разговоров.
5. Транспорт: бронирование мест, спутниковые навигационные системы, диспетчерские службы, оперативный анализ, управление городским транспортом.
6. Образование и подготовка специалистов: тренажеры, дистанционное обучение, банки информации, электронные учебные пособия.
7. Медицина: компьютерные методы обследования, анализ истории болезни.

1. ЗАПОМИНАЮЩИЕ УСТРОЙСТВА

Микросхемы памяти в общем объеме выпуска ИС занимают около 40% и играют важную роль.

1 Классификаци запоминающих устройств

Важнейшим признаком ЗУ является способ доступа к данным. По этому признаку различаются 3 вида ЗУ:

1. Адресные ЗУ: код на адресном входе указывает ячейку, с которой ведется обмен.

Адресные ЗУ: делятся на RAM (Random ACCESS Memory) (ОЗУ-оперативные запоминающие устройства) и ROM (Read-Only Memory) (ПЗУ-постоянные запоминающие устройства).

RAM делятся на статические - SRAM (Static RAM) и динамические -DRAM (Dynamic RAM).

В статических ОЗУ запоминающими элементами являются триггеры. В динамических ОЗУ данные хранят в виде зарядов конденсаторов, образуемых элементами МОП-структур. Запоминающие конденсаторы разряжаются, поэтому каждые несколько миллисекунд данные должны регенерироваться

Плотность упаковки динамических элементов памяти в несколько раз выше, чем статических.

Динамические ОЗУ характеризуются наибольшей информационной емкостью и невысокой стоимостью, но имеют большее энергопотребление и меньшее быстродействие.

Постоянная память типа ROM имеет следующие разновидности:

1. Программируемые при изготовлении ИМС с помощью одной из масок. Эта память типа ПЗУМ (ПЗУ масочные). [ROM(M)]
2. Память, программируемая пользователем (ППЗУ – программируемые ПЗУ):
   - PROM – содержимое записывается однократно в память.
   - EPROM и EEPROM – содержимое может быть заменено путем стирания информации и записи новой.

В EPROM – стирание путем облучения кристалла ультрафиолетовыми лучами (РПЗУ-УФ – репрограммируемые ПЗУ с УФ стиранием).

В EEPROM – стирание происходит электрическими сигналами (РПЗУ-ЭС – репрограммируемые ПЗУ с электрическим стиранием).

Запись данных для EPROM и E^2PROM производится элетрическими сигналами.

2. Последовательные ЗУ:
   - FIFO;

- Стековые (LIFO);
- Файловые;
- Циклические.

В FIFO запись в буфер становится сразу доступной для чтения, т.е. поступает в конец цепочки (First In - First Out) – «первый пришел – первый вышел».

В файловых – данные поступают в начало цепочки.

В циклических ЗУ – слова доступны одно за другим с постоянным периодом, определяемым емкостью памяти. К такому типу относится видеопамять (VRAM).

В стековых ЗУ считывание происходит в обратном порядке (последний принят – первый вышел) – LIFO (Last In – First Out).

3. Ассоциативные ЗУ.

Поиск информации происходит по некоторому признаку, а не по ее расположению в памяти.

1.2. Структуры ЗУ

Для ПЗУ и статических ОЗУ характерны структуры 2D, 3D и 2DM

1.2.1. Структура 2D

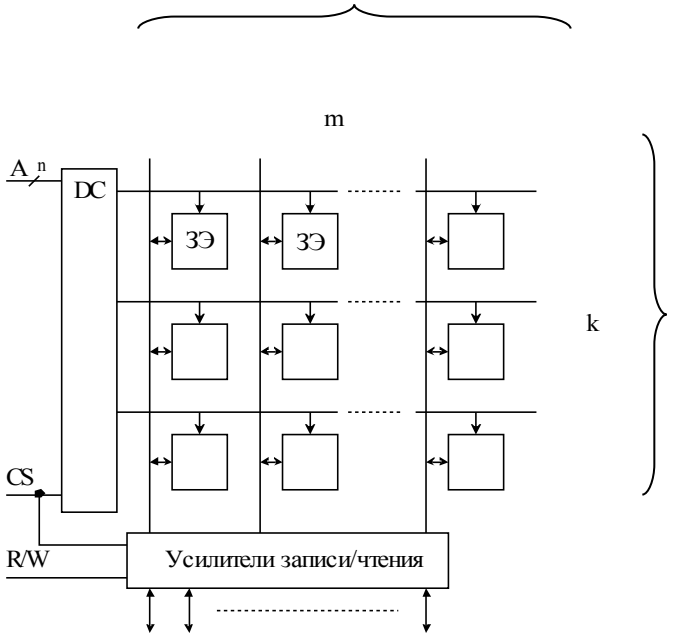


Рис.1

k – число хранимых слов;

m – разрядность слов;

M=k×m – информационная емкость памяти (в битах).

Дешифратор адресного кода при наличии разрешающего сигнала CS активизирует одну из выходных линий, разрешая доступ ко всем элементам выбранной строки. Элементы одного столбца соединены вертикальной линией -внутрен-

ней линии данных (разрядной линией, линией записи/считывания). Направление обмена определяется усилителями записи/чтения под воздействием сигнала R/W (Read – чтение, Write - запись).

Структура 2D применяется только в ЗУ малой емкости из-за чрезмерного усложнения дешифратора при росте числа хранимых слов.

1.2.2. Структура 3D

Позволяет упростить дешифраторы адреса с помощью двухкоординатной выборки запоминающих элементов.

Пример ЗУ типа ROM (только чтение данных, одноразрядная организация):

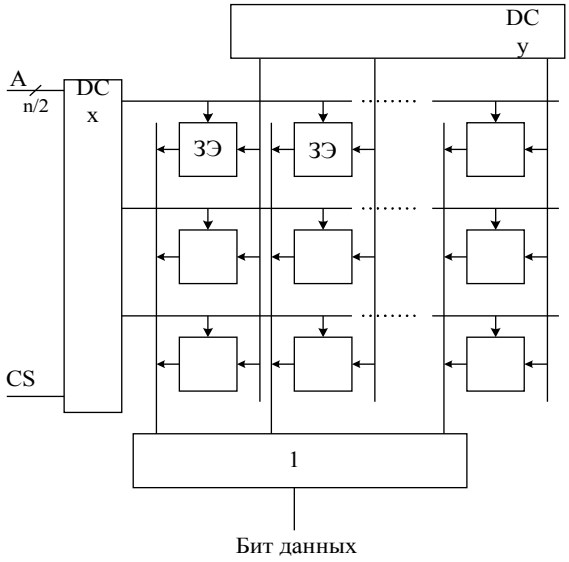


Рис. 2

Выбирается ЗЭ, находящийся на пересечении линий выходов дешифраторов.

Например для ЗУ емкостью 1 К слов потребуется для 2D – дешифратор с 1024 выходами, а для 3D – 2 дешифратора с 32 выходами.

1.2.3. Структура 2DM (модифицированная)

Сочетаются достоинства обеих структур: упрощается дешифрация адреса и не требуются запоминающие элементы с двухкоординатной выборкой.

Рассмотрим ЗУ типа ROM.

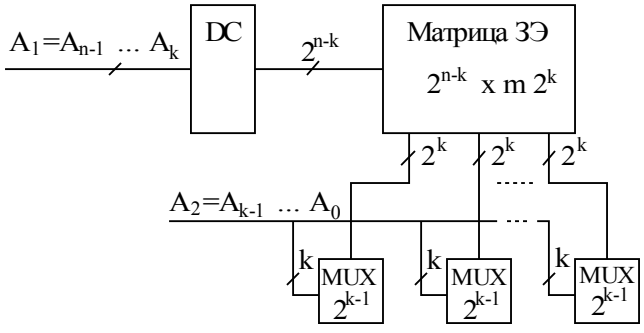


Рис. 3

Дешифратор выбирает в матрице ЗЭ целую строку (как в 2D). Однако длина строки многократно превышает разрядность хранимых слов. При этом увеличивается число строк матрицы и уменьшается число выходов дешифратора.

1.3. Типы ЗУ

1.3.1. Масочные ЗУ [ROM(M)]

Элементами связи могут быть диоды, биполярные транзисторы, МОП-транзисторы. Программируются с помощью одной из масок при изготовлении ЗУ.

Диодные ЗЭ

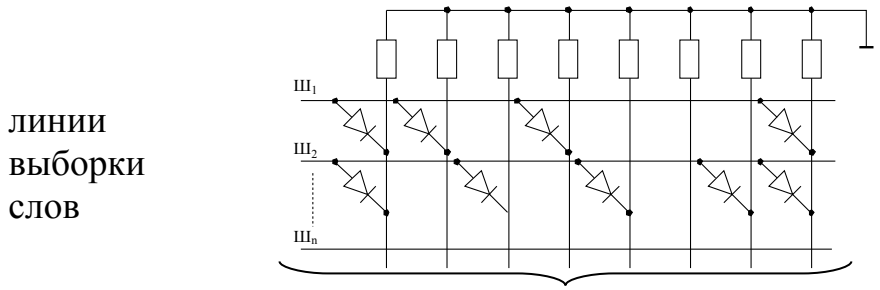


Рис. 4

При наличии диода высокий потенциал выбранной горизонтальной линии передается на соответствующую вертикальную линию и в данном разряде появляется «1».

При возбуждении (высокий потенциал) линии $Ш_1$ считывается слово 11010001. При возбуждении $Ш_2$ считывается слово 10101011. Линии выборки ($Ш_1$-$Ш_n$) являются выходами дешифратора адреса.

Матрица МОП-транзисторных элементов

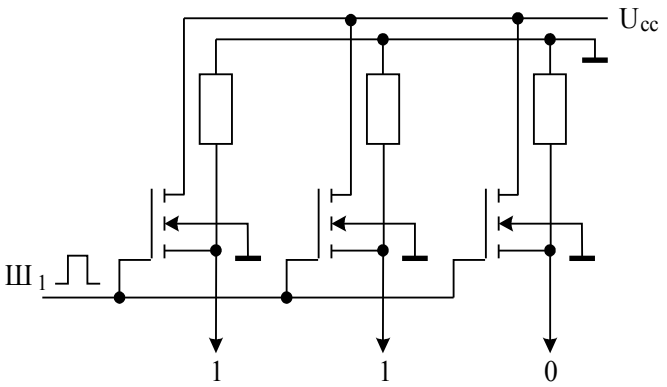


Рис. 5

В МОП-транзисторах, соответствующих хранению нуля, увеличивают толщину подзатворного окисла, что ведет к увеличению порогового напряжения транзистора. В этом случае рабочие напряжения не могут открыть транзистор, что соответствует его отсутствию.

Масочные ЗУ отличаются высоким уровнем интеграции.

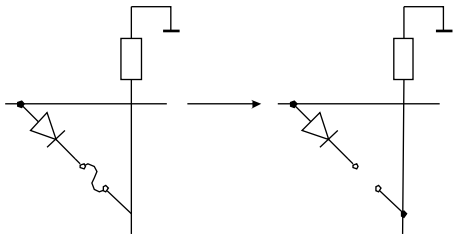
Область применения: хранение стандартной информации, имеющей широкий круг потребителей. Это прошивка кодов букв русского и латинского алфавита, таблицы типовых функций (sin, квадратичной функции и др.), стандартное программное обеспечение и т.п.

1.3.2. ЗУ типа PROM

Такие ЗУ программируются пользователем устранением или созданием перемычек.

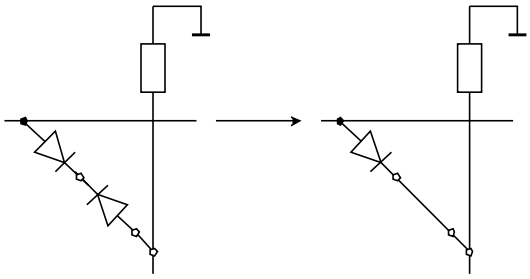
Устранение части перемычек свойственно ЗУ с

1) плавкими перемычками (типа *fuse* – предохранитель). В исходном состоянии ЗУ имеет все перемычки, а при программировании часть их ликвидируется путем расплавления импульсами тока (большой амплитуды и длительности). Эти перемычки включаются в электроды диодов или транзисторов. Изготавливаются металлическими (нихром) и поликристаллическими (кремниевыми).



Исходное состояние перемычка удалена

Рис. 6



Перемычка образована

Рис. 7

2) другой тип перемычки: два встречно включенных диода. В исходном состоянии цепь можно считать разомкнутой. Для записи «1» к диодам приклады-

вается высокое напряжение, пробивающее диод, смещенный в обратном направлении. Диод пробивается с образованием в нем короткого замыкания.

3) схемы с тонкими пробиваемыми диэлектрическими перемычками (типа *antifuse*) более компактны и совершенны. Применяются в ПЛИС.

Представителем ЗУ с плавкими перемычками является м/сх К155РЕ3 (ТТЛ).

Плавкие перемычки занимают довольно много места, поэтому уровень (степень) интеграции ниже, чем у масочных ЗУ. Однако имеют невысокую стоимость, т.к. изготовитель выпускает микросхему без учета конкретного содержимого ЗУ. Программирует ЗУ пользователь.

Среди отечественных PROM ведущее место занимают микросхемы серии К556. Емкость 1-64 Кбит и $\tau_{доступа}$=70-90 нс.

1.3.3. ЗУ типов EPROM и E²PROM

Это репрограммируемые ЗУ.

В EPROM (РПЗУ-УФ) – информация стирается ультрафиолетовыми лучами, а в E²PROM (РПЗУ-ЭС) – электрическими сигналами.

Запоминающими элементами (современных) РПЗУ являются транзисторы типов МНОП (метал-нитрид-окисел-полупроводник) и ЛИЗМОП (лавинная инжекция заряда).

МНОП-транзистор

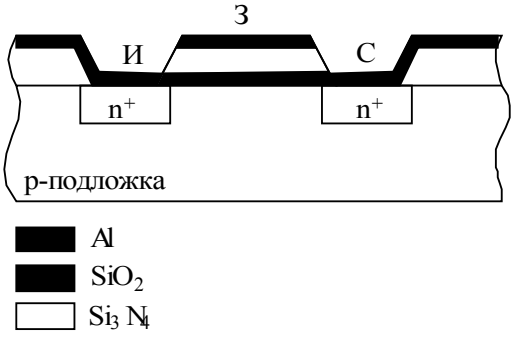
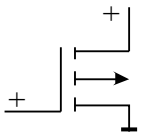


Рис. 8

Над каналом расположен тонкий слой оксида кремния SiO_2 (≤5 нм), далее идет толстый слой нитрида кремния Si_3N_4 и Al затвор. Благодаря туннельному эффекту носители заряда могут проходить через тонкую пленку SiO_2. Они скапливаются на границе раздела SiO_2-Si_3N_4, где возникают центры захвата заряда. Этот заряд и является носителем информации, хранимой МНОП-транзистором. Заряд записывают созданием под затвором напряженности электрического поля, достаточной для возникновения туннельного эффекта. Наличие заряда влияет на пороговое напряжение транзистора. Для него отрицательный заряд увеличивает пороговое напряжение (транзистор закрыт), а положительный заряд уменьшает пороговое напряжение (транзистор открыт). Заряды создаются при приложении напряжения на затвор ($\pm U_з$) ($+U_з$ создает отрицательные заряды, а $-U_з$ –

положительные). Одно из состояний МНОП-транзистора принимается за «0», другое – за «1».



n-канальный

При программировании используется напряжение около 20В. После 10^4… 10^6 перезаписей МНОП-транзистор перестает устойчиво хранить заряд.

РПЗУ на МНОП-транзисторах энергонезависимы и могут хранить информацию десятками лет. Старая информация стирается записью нулей во все ЛЭ. Тип ЗУ - РПЗУ-ЭС.

ЛИЗМОП-транзистор

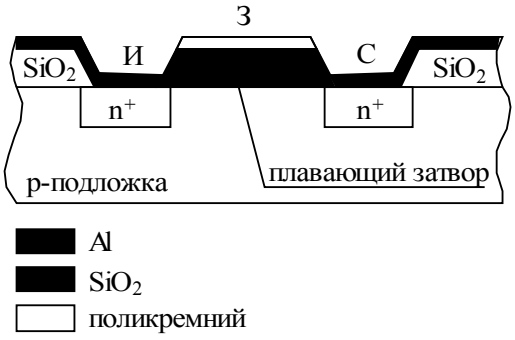


Рис. 9

Транзисторы имеют плавающий затвор из поликремния. На рис. он является вторым, дополнительным к управляющему затвору.

Такие транзисторы используются в РПЗУ-УФ и в РПЗУ-ЭС.

Принцип работы: в плавающий затвор вводится заряд, влияющий на величину порогового напряжения. Он сохраняется там в течении длительного времени.

При подачи напряжения на управляющий затвор, сток и исток импульса положительного напряжения 20…25 В в p-n-переходах возникает лавинный пробой, область которого насыщается электронами. Часть электронов с высокой энергией проникает через потенциальный барьер в плавающий затвор, где и сохраняется многие годы.

Отрицательный заряд плавающего затвора увеличивает пороговое напряжение настолько, что транзистор всегда закрыт.

При отсутствии заряда транзистор работает в обычном ключевом режиме.

Для стирания информации УФ лучами в корпусе делают окошко. УФ лучи вызывают фототоки и тепловые токи и заряды покидают плавающий затвор. Время стирания – десятки минут. Число циклов – 10…100.

При электронном стирании на затвор подается ноль Вольт, а на сток и исток – высокое напряжение. Число циклов 10^4…10^6.

ЭС стирание вытесняет УФ стирание.

Среди отечественных РПЗУ-УФ известна серия К573, а среди РПЗУ-ЭС – серии КР558 (n-МОП) и К1609, К1624, К1626 на ЛИЗМОП.

1.3.4. *Статические ОЗУ (SRAM)*

Довольно дорогостоящие, но имеющие высокое быстродействие. Широко используются в КЭШ-памяти. Имеют обычно структуру 2DM. При небольшой емкости – 2D.

Запоминающий элемент – триггер с цепями установки и сброса. Применеяются схемотехнологии: ТТЛ(Ш), $И^2Л$, ЭСЛ, n-МОП, КМОП, AsGa и др. Это микросхемы серии К537 (КМОП) и К132 (n-МОП).

Запоминающие элементы

на n-МОП транзисторах

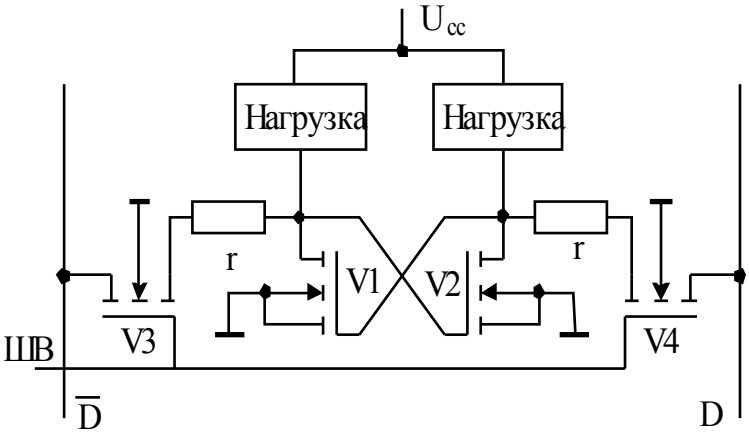


Рис. 10

RS-триггер выполнен на транзисторах V1, V2. Транзисторы V3 и V4 – ключи выборки.

При обращении и ЗЭ появляется высокий потенциал на шине выборки ШВ. Этот потенциал открывает ключи выборки (транзисторы Т3, Т4) Через D и $\overline{D}$ считываются данные. Через D и $\overline{D}$ можно записывать данные в триггер, подовая низкий потенциал на шину. Тогда при подаче «0» на $\overline{D}$ снижается стоковое напряжение транзистора V1, что запирает транзистор V2. Триггер установлен в состоянии «1».

1.3.5. Статические ЗУ типа БиКМОП.

БиКМОП – биполярная и КМОП-технология.

Применительно к SRAM это реализация триггеров на КМОП, а цепей выдачи данных, имеющих значительную емкостную нагрузку – по биполярной схемотехники (ЭСЛ или ТТЛШ).

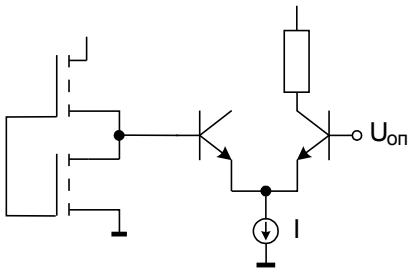


Рис. 11

1.3.6. Динамические ЗУ (DRAM)

Данные хранятся в виде зарядов емкостей МОП-структур. Такой ЗЭ проще триггерного, что позволяет размещать на кристалле в 4-5 раз больше ЗЭ.

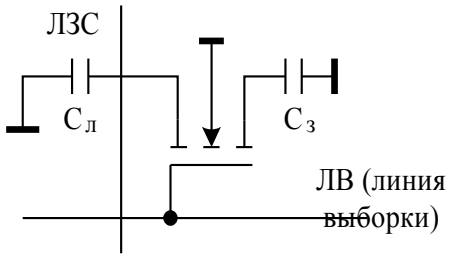


Рис. 12

Ключевой транзистор отключает запоминающий конденсатор от линии записи-считывания или подключает его к ней. Сток транзистора не имеет внешнего вывода и образует одну из обкладок конденсатора (поликремний). Между обкладками расположен тонкий слой оксида кремния.

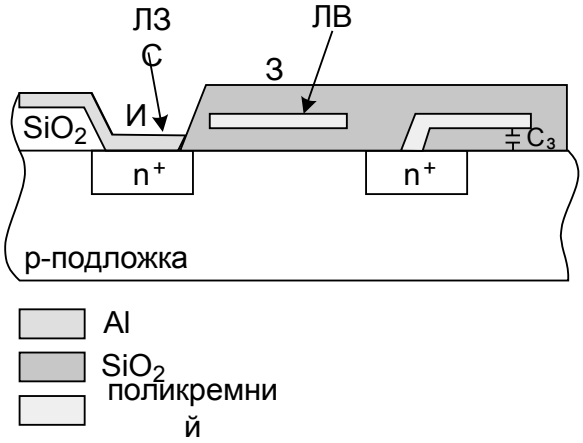


Рис. 13

В режиме хранения транзистор заперт. При выборке данного ЗЭ на затвор подается напряжение, открывающее транзистор. Емкость $C_з$ подключается к линии записи-считывания. И в зависимости от того, заряжая емкость или разряжена, по разному влияет на потенциал ЛЗС.

Процесс чтения состояния ЗЭ

Перед считыванием производится предзаряд ЛЗС до уровня половины $E_П$ ($U_{сс}/2$).

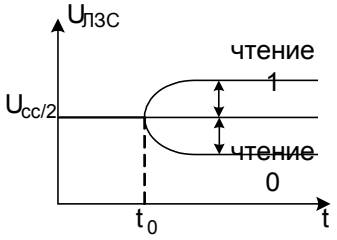


Рис. 14

Для считывания нуля справедливы следующие рассуждения. До выборки ЗЭ емкость ЛЗС имела заряд:

$$Q = C_Л U_{сс} / 2$$

После выборки этот же разряд имеет суммарную емкость

$$C_Л \frac{U_{СС}}{2} = (C_Л + C_З)(\frac{U_{СС}}{2} - \Delta U)$$

Приравнивая правые части, получим:

$$C_Л + C_З: \quad Q = (C_Л + C_З)(\frac{U_{СС}}{2} - \Delta U)$$

Откуда:

$$\Delta U = \frac{U_{СС} C_З}{2(C_Л + C_З)} \approx \frac{U_{СС}}{2} \frac{C_З}{C_Л}$$

($C_л >> C_з$)

Считывание является разрушающим – подключение $C_з$ к ЛЗС изменяет ее заряд.

Стараются увеличивать $C_з$. Для этого применяют диэлектрик двуокись титана. Он имеет ε в двадцать раз больше, чем оксид кремния.

Т.к. емкость $C_З$ имеет саморазряд, необходимо периодически (с периодом 1-15 мс) восстанавливать напряжение на $C_З$, или говорят регенерировать.

2. МИКРОКОНТРОЛЛЕРЫ AVR

После выхода на рынок первого микропроцессора (МП) – 8080 компании Intel развитие МП пошло по двум главным направлениям. С одной стороны это высокопроизводительные процессоры типа Pentium, применяемые в персональных компьютерах (ПК), рабочих станциях. С другой стороны – более простые и дешевые микроконтроллеры, ценою от одного до 25 долларов. В составе микроконтроллеров (МК) имеются модули того же функционального назначения, что и в универсальных МП. Обширная область применения МК обозначила и большие объемы продаж. Наиболее известны МК семейств 8051 (Intel), Z80, Z86, (фирма Zilog), MC68 (Motorola), PIC 16/17 (Microchip), AVR (Atmel).

МК AVR состоят из трех семейств:

1. Classic AVR (AT90S1200 и др.). Имеют память программ 1-8 кбайт х 16 разрядов.
2. Tiny AVR (ATTiny 11, 12, 13, 15, 26, 28, ATTiny2313). Память программ 1-2 кбайта х 16 разрядов.
3. Mega AVR (ATmega 8, 16, 32, 64 и др.) Память программ 8-64 кбайта х 16 разрядов.

Семейство ATtiny обозначается следующим образом:

AT	tiny11	-	2	P	W

AT - фирма Atmel

tiny11 - семейство Tiny

2 - максимальная тактовая частота 2 МГц

P - тип корпуса PDIP

W - диапазон рабочих температур

2.1. Системы счисления

Позиционные системы счисления основываются на отображении чисел с помощью основания B и экспоненоры i. Число можно записать в виде

$$Z=\pm(a_{n-1}B^{n-1}+\ldots+a_1B^1+a_0B^0+a_{-1}B^{-1}+a_{-2}B^{-2}+\ldots)=\pm\sum_{i=-n}^{n-1}a_iB^i,$$

Где B – *основание* в виде целого числа, a – значение разрядов цифр.
Применяются следующие системы счисления:

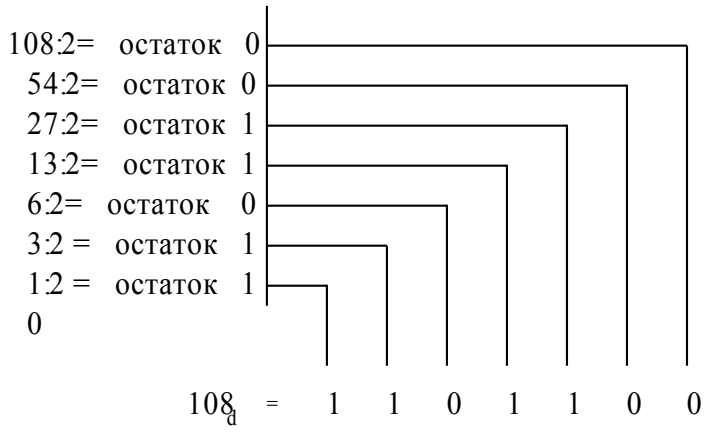
- Двоичная система: B=2, a={0,1}
  Например: 0b10100101 или 10100101_b
- Десятичная система: B=10, a={0, 1, 2, 3,…9}
  Например: 15, 37, 15_d, 37_d.
- Восьмеричная система: B=8, a={0, 1, 2, 3,…7}
- Шестнадцатеричная система: B=16, a={0, 1, 2, 3,…9, A, B, C, D, E, F}, где A=10, B=11, C=12, D=13, E=14, F=15.
  Например: $1A, 0x1A, $1A_h$.

Двоичный код имеет ряд преимуществ: схемы с двумя состояниями являются самыми помехоустойчивыми, а, следовательно, и надежными. Кроме того, двоичная система подходит для вычисления функций логических переменных (истинно и ложно).

Переведем двоичное число 0b00011001 в десятеричный код:

$$Z=1\times2^0+0\times2^1+0\times2^2+1\times2^3+1\times2^4+0\times2^5+0\times2^6+0\times2^7=25_d$$

Преобразование десятичного числа в двоичное:



Преобразование десятичного числа после запятой в двоичный код.

2^{-1}	2^{-2}	2^{-3}	2^{-4}
0,5	0,25	0,125	0,0625

Пусть дано число $0{,}3125_{10}$. Вычитаем из него последовательно приведенные в таблице числа. Если вычитается, то ставим 1, если не вычитается – 0 .

0,3125-0,5= 0 - старший
0,3125-0,25= 1
0,0625-0,125= 0
0,0625-0,0625= 1 – младший

Отрицательные двоичные числа. Представляются в дополнительном коде и называются двоичным числом со знаком.

Двоичное число 00101101

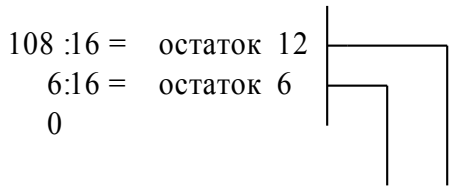
В дополнительном коде 11010010. Прибавляем единицу и получаем отрицательное число в дополнительном коде: 11010011.

Сравнивая положительные и отрицательные числа, можно сделать следующий вывод: старший разряд восьмиразрядного положительного числа равен нулю, а старший разряд отрицательного числа равен единице.

Шестнадцатеричная система счисления.

При всех достоинствах двоичные числа имеют один недостаток: необходимость использования большого количества разрядов. Преодолеть это неудобство позволяет использование шестнадцатеричных чисел. Здесь каждый разряд может быть представлен шестнадцатью цифрами. Для представления восьмиразрядных двоичных чисел достаточно всего лишь двух шестнадцатеричных разрядов, что значительно сокращает запись числа.

Преобразование десятичного числа в шестнадцатеричное производится по следующей схеме:



108_d = 6 C

Преобразование двоичного кода в шестнадцатеричный производится следующим образом:

0b1000 0100 0010 1101 = 0x842D

Двоично-десятичный код

Чтобы упростить вычисления ввели двоично-десятичный код (BCD – Binary Coder Decimal). Двоичное число, каждые четыре разряда которого выражают одну цифру десятичного числа, и есть код BCD.

Например: 69_d = 0b0110 0b1001

2.2. Базовая структура микроконтроллера

Структура микроконтроллера (МК) представлена на рис. 15.

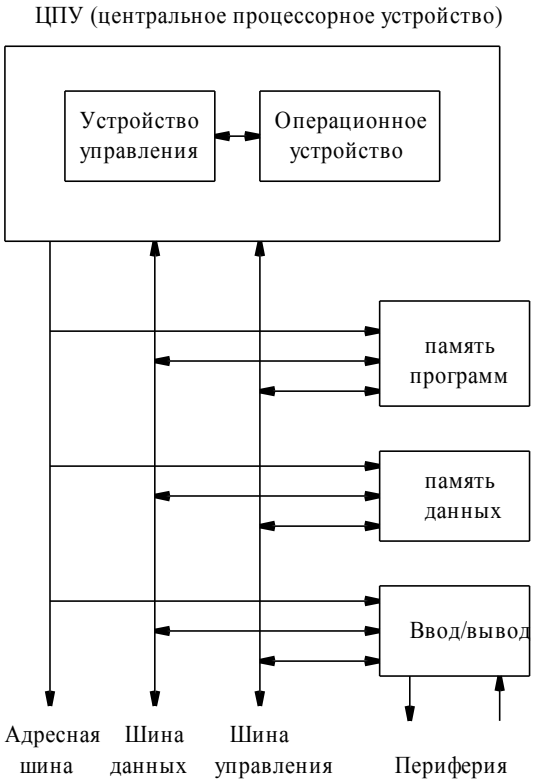


Рис. 15

Операционное устройство состоит из арифметико-логического устройства (АЛУ), накапливающего сумматора и вспомогательных регистров. В классических МК почти половина всех команд служит для передачи данных от вспомогательных регистров к накапливающему сумматоры (аккумулятору) и обратно. В МК AVR все 32 вспомогательных регистра связаны с АЛУ и с ними можно выполнять необходимые операции в течение одного такта.

В дальнейшем будет осуществляться привязка к микроконтроллеру AVR AT-Tiny2313. Он имеет следующие характеристики:

- Flash память программ 2 к
- Память данных EEPROM 128 байт
- ОЗУ данных SRAM 128 байт
- Тактовая частота 0-16 МГц
- Число линий ввода/вывода 18
- 16-и разрядный таймер 1
- 10-и разрядный АЦП -
- 8-и разрядный таймер 1
- Сторожевой таймер Y
- ШИМ каналов 4
- Аналоговыйы компаратор Y
- Детектор снижения напряжения Y
- UART 1

2.3. Архитектура памяти

По классической концепции фон Неймана данные и команды хранятся в одной памяти. В AVR принята Горвардская архитектура. В ней разделены память программ и память данных.

а). *Память программ* реализована на основе программируемой и электрически стираемой ФЛЭШ-технологии (память 16-ти разрядная, двухбайтовая).

б). *Внутренняя память* для энергозависимых данных - статическая память SRAM. Поэтому AVR может работать с тактами вплоть до нуля герц (пошаговая работа).

в). Для данных, которые должны сохраняться после отключения питания, служит память EEPROM (электрически стираемое РПЗУ).

2.4. Структура памяти МК AVR

Структура памяти МК AVR приведена на рис. 16.

Карта памяти

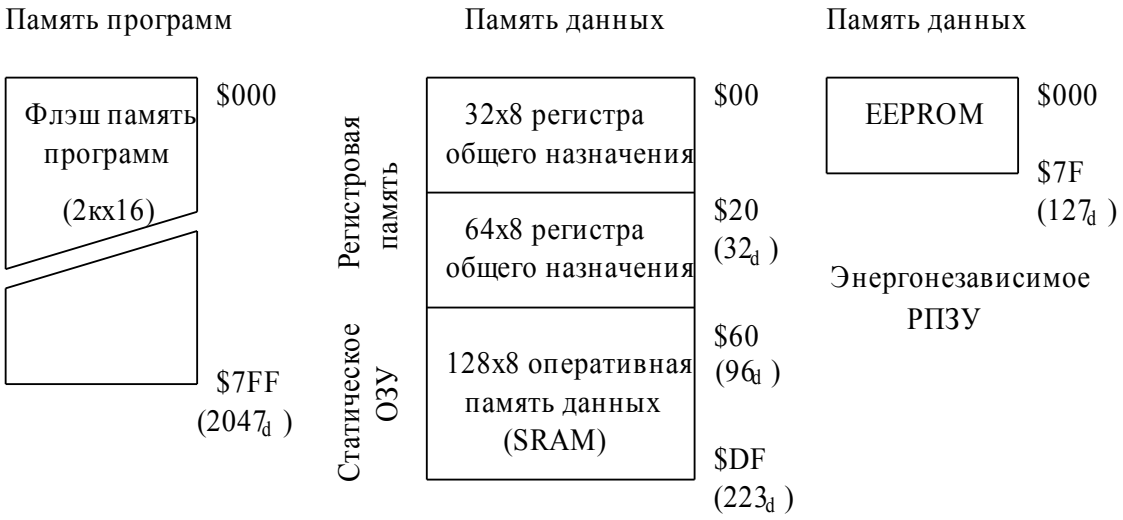


Рис. 16

Память разделена на три части: флэш память программ, регистровая память и статическое ОЗУ, энергонезависимая память EEPRO(M). Все команды занимают в памяти программ по 16 бит. (1 кбайт х16 – 1024 16-битных слова).

2.4. Регистры общего назначения

Все регистры общего назначения (РОН) объединены в регистровый файл быстрого доступа. В МК AVR имеется 32 РОН - (R0-R31). Все они доступны АЛУ. Поэтому любой РОН можно использовать во всех командах и как операнд-источник, и как операнд-приемник. Такое решение позволяет выполнять операцию за один машинный цикл. Исключение составляют пять арифметических и логических команд, выполняющх действия между константой и регистром (SBCI, SUBI, CPI, ANDI, ORI), и команда загрузки константы в регистр (LDI). Эти команды могут обращаться только к регистрам R16 – R31. Регистровый файл имеет структуру, показанную на рис. 17.

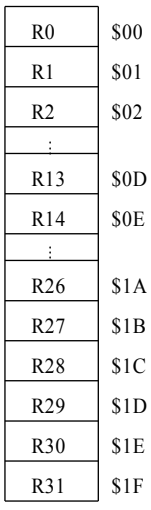


Рис. 17

Последние шесть регистров (R26-R31) могут объединяться в три 16-разрядных регистра X, Y, Z. Они используются в качестве указателей при косвенной адресации памяти данных. Регистры определены следующим образом (рис. 18):

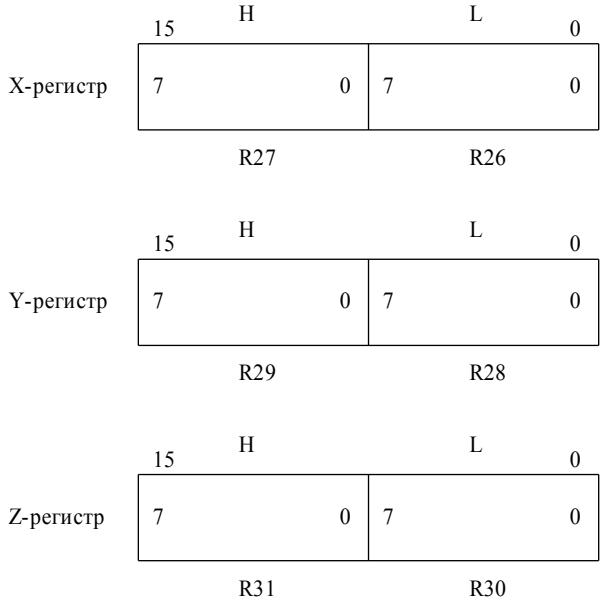


Рис. 18

К регистрам (например, Z) можно обращаться по отдельности, как к регистрам ZL и ZH, но можно и объединить таким образом, что ZL (Lower Z – младший) будет содержать биты 0…7 16-битного числа, а ZH (higher Z – старший) биты 8…15.

2.5. Регистры ввода/вывода

Регистры ввода/вывода расположены в пространстве ввода/вывода размером 64 байта. Размер каждого регистра 8 бит. Регистры ввода/вывода можно разделить на две группы.

1. *Служебные регистры* микроконтроллера: регистр состояния, регистры разрешения прерываний, регистры таймера, аналогового компаратора, сторожевого таймера, обращения к памяти EEPROM и др.

2. *Регистры портов ввода/вывода.* Они служат для связи микроконтроллера с внешним миром.

К любому регистру ввода/вывода можно обратиться с помощью команд IN и OUT. Они выполняют пересылку данных между одним из 32-разрядных РОН и пространством ввода/вывода.

2.5.1. Конфигурирование портов ввода/вывода

Каждый вывод порта может быть индивидуально сконфигурирован как вход или выход. При функционировании в качестве входа, к нему может быть подключено подтягивающее сопротивление. Схема вывода порта приведена на рис.19.

Схема вывода порта

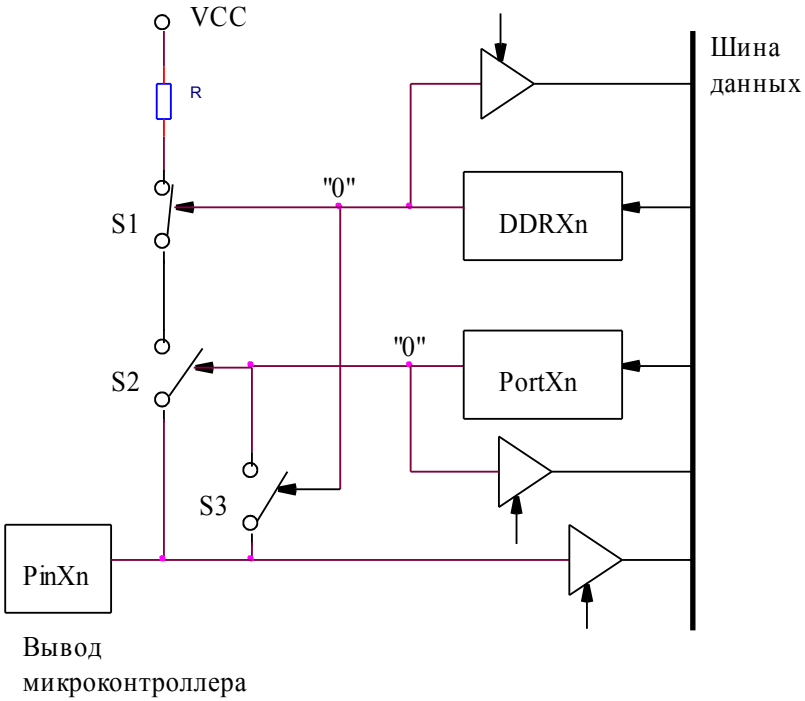


Рис. 20

Буква X соответствует любому порту ввода/вывода: A, B, C, D. Номера разрядов 0…7 внутри регистра представлены буквой n. Например: DDRA1.

Положение ключей S1…S3 на рис. 20 соответствуют сигналам низкого уровня на выходах регистров DDRXn и PortXn.

Для обращения к портам служат три регистра: DDRX, PortX и PinX.

Регистр направления передачи данных – DDRX.

Определяет назначение вывода порта: *вход* или *выход*.

а). Если разряд”n” в регистре DDRXn имеет низкий уровень (логический “0”), то вывод сконфигурирован как *вход*. При этом ключ S3 разомкнут и отделяет регистр PortXn от вывода PinXn. Подключением подтягивающего резистора R управляет регистр PortXn. Если на выходе PortXn логический “0”, то ключ S2 разомкнут и подтягивающий резистор отключен. Если на выходе PortXn логическая “1”, то ключ S2 замкнут и к выводу PinXn подключен подтягивающий резистор. Входной сигнал передается через буфер на шину данных.

б). Если разряд”n” в регистре DDRXn имеет высокий уровень (логическая “1”), то вывод сконфигурирован как *выход*. При этом ключ S3 замкнут и подключает регистр PortXn к выводу PinXn. Ключ S1 разомкнут и отключает подтягивающий резистор от вывода PinXn. Приэтом в регистр PortXn записывается значение, подлежащее выводу (0 или 1).

в). С помощью управляемых буферов (треугольники на схеме) при подаче на них управляющего сигнала “логическая 1” можно считать в шину данных состояние регистров DDRXn, PortXn, PinXn.

Все конфигурации управления портом показаны в таблице1.

Табл.1

Регистр DDRXn	Регистр PortXn	Ввод/ вывод	Подтягивающее сопротивление	Описание
0	0	вход	отключено	Высокоомный вход
0	1	вход	подключено	Вход с подтягивающим резистором
1	0	выход	отключено	Выход: лог. “0”
1	1	выход	отключено	Выход: лог. “1”

Порты микроконтроллеров AVR при низком уровне сигнала на выходе принимают втекающие токи до20 мА и вытекающие токи (при логической 1) до 6-7 мА.

Большинство портов альтернативно применяются для выполнения дополнительных функций: входы таймеров, прерываний, аналоговых компараторов и др.

Микроконтроллеры понимают только двоичный код (машинный язык). Чтобы упростить запись, используют шестнадцатеричную запись. Но понимать (и запоминать) такие команды практически невозможно. Поэтому применяют специальный язык, Ассемблер, в котором ставится в соответствие шестнадцатеричный код команды определенной мнемонике.

Например:

Ассемблер	Команда в 16-ом коде	Описание
ldi r16,$A5	16A5	Константа A5 загружается в регистр r16

Для инициализации портов применяются команды:

команда	описание
Ldi reg,number	Загружает число number (0…255) в регистр reg (16-31)
Out ioreg,reg	Пересылает содержимое регистра reg в регистр ввода/выода ioreg
Ser reg	Устанавливает все биты регистра reg в “1” (регистры r16-r31)
Clr reg	Сбрасывает в “0” все биты регистра reg (регистры r0-r31)

Корпус микроконтроллера ATTiny2313 и назначение выводов приведены на рис. 21.

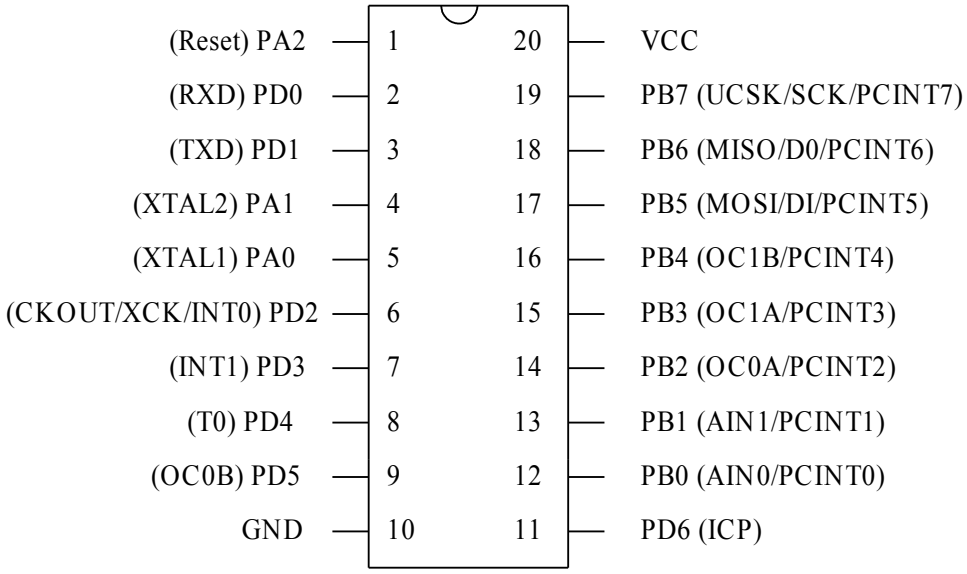


Рис. 21

Пример инициализации портов.

Пусть выводы D0, D4, и D6 – входы и подключены к кнопкам. Мы хотим, чтобы подтяжка была только на входах D4 и D6. Выводы B0…B7 – выходы. Остальные выводы – неподключенные. При включении питания все выходы должны находиться в состоянии лог. 0.

При инициализации портов надо учесть, что мы не можем заносить двоичный код непосредственно в регистры ввода/вывода. Мы должны сначала записать двоичный код в рабочий регистр, а затем переслать содержимое рабочего регистра в регистр ввода/вывода.

Схем устройства приведена на рис. 22.

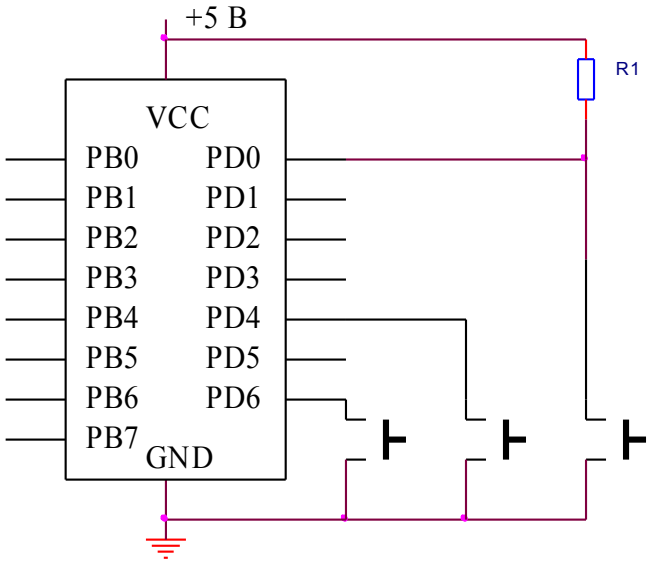


Рис. 22

В схеме неиспользуемые входы PD1-PD3, PD5 установим как выходы. Фрагмент программы инициализации портов будет выглядеть следующим образом:

```
Ldi     r16,0b0101110 ;D0, D4, D6 – входы, остальные выходы
Out     DDRD,r16 ;число из рег. r16 пересылается в рег. DDRD
Ldi     r16,0b1010000 ;PD4,PD6 – подтяжка, D0-высокоомный вход
Out     PortD,r16
Ser     r16           ;установка всех битов рег. R16 в 1
Out     DDRB,r16      ;все разряды порта B - выходы
Clr     r16
Out     PortB,r16     ;на выходе порта B – лог. Нули
```

2.6. Ассемблер

Для отладки программ и создания файла прошивки МК используются программы AVR-ассемблер (WAVRASM.EXE), и AVR-STUDIO. Пользователь создает текстовую программу (расширение *.asm) с помощью любого редактора или в AVR-STUDIO.

После трансляции программы на машинный язык создается файл *.obj, который используется для отладки (симуляции) программы. После отладки программы создается загрузочный файл (для прошивки МК), имеющий расширении е *.hex. В нем находятся машинные команды для записи в память программ в виде ASCII – кода в одном из форматов: Generic (Atmel), S-record (Motorolla) или Intellec 8/MDS (Intel). Большинство программаторов обрабатывают эти форматы.

Программа на ассемблере представляет текстовый файл, который состоит из символьных обозначений *команд, меток и директив*.

Метки используются для обозначения текущей строки некоторым именем (меткой) для использования в командах переходов. После метки ставится двоеточие.

Директивы транслятора ассемблера не транслируются в машинные коды, а используются в процессе ассемблирования. Директива начинается с точки.

Синтаксис формата Intel Intellec 8/MDS показан на рис. 22:

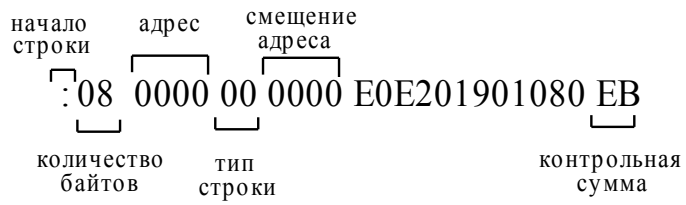


Рис. 22

Типы строк:

00 – данные

01 – окончание записи. Указывает на окончание файла данных. После типа строки следует только контрольная сумма.

02 – адреса сегментов для расширения адресного пространства.

2.6.1. Директивы Ассемблера

.CSEG – сегмент кода. Директива указывает компилятору, что код программы должен быть помещен в *память программ.*

.DEF – назначить регистру символическое имя, по которому программа может обращаться к этому регистру.

Например.

```
.DEF    tmp=r16; регистру r16 назначено имя tmp. В дальнейшем
                 можно писать вместо r16  tmp.
```

.DEVICE – сообщает ассемблеру о типе микроконтроллера.

Например.

```
.Device ATTiny2313 ;Назначен тип микроконтроллера
                    ATTiny2313
```

Если директива отсутствует, то ассемблер допускает, что разрешены все команды, а границы памяти отсутствуют.

.ORG – определяет абсолютный адрес.

Например.

```
.org   $10 ;Устанавливается счетчик адреса текущего сегмента на
           абсолютный адрес $10
```

Если директива .org не указана, то счетчик адреса команд изначально устанавливается на ноль.

.DSEG – определяет сегмент данных в SRAM.

.EQU – сопоставляет символьному имени некоторое арифметическое или логическое выражение.

Например.

```
.equ   B=0x4  ;Буква B находится по адресу 0x4.
.equ   tab_size=20 ;Размеры таблицы tab_size равны 20 байт.
```

.Exit – завершает ассемблирование файла.

.ESEG – определяет начало сегмента EEPROM.

.INCLUDE – вставляет в исходный файл другой файл.

.LIST – активизирует создание файла листинга, содержащий команды из исходного файла вместе с соответствующими кодами операций и адресами.

.NOLIST – отключает создание листинга.

.SET – назначает выражению символическое имя.

.DB – определяет однобайтную константу в памяти программ или EEPROM.

.DW – определяет двухбайтную константу (от -32768 до 65533) в памяти программ или EEPROM. Если в EEPROM (восемь разрядов) записываются двухбайтные константы, слова, то вначале записывается младший байт, а старший байт размещается по адресу, следующему за адресом младшего байта.

С учетом директив программа инициализация портов запишется в виде:

```
.device ATTiny2313 ; тип микроконтроллера
.nolist ; запрещает создание листинга
```

```
.include “C:\AVR\def\tn2313def.inc” ; вставляемый файл tn2313def
.list     ; разрешает создание листинга
.def   tmp=r16 ; назначает регистру r16 имя tmp
.CSEG ;код программы будет помещен в память программ
.ORG $000 ; устанавливает счетчик адреса на ноль
rjmp  Init ; команда перехода на метку Init
init:
ldi   tmp,0b0101110
out   ddrD, tmp
ldi   tmp,0b1010000
out   PortD,tmp
ser   tmp
out ddrB,tmp
clr   tmp
out   PortB,tmp
```

2.7. Регистр состояния SREG

Регистр доступен для записи и чтения и после подачи сигнала сброса инициализируется нулями. 8 битов регистра служат для обозначения результатов выполнения операций. Все биты могут быть проверены с помощью команд.

Регистр SREG

7	6	5	4	3	2	1	0
I	T	H	S	V	N	Z	C

C – (carry), флаг переноса. Указывает на переполнение (перенос) после выполнения арифметической или логической операции. Устанавливаетсяы в 1, если в результате операции произошел выход за границы байта.

Z – (Zero), флаг нуля. Устанавливается в 1, если результат арифметической или логической операции равен 0.

N –(Negative), флаг отрицательного результата. Указывает на отрицательный результат после выполнения арифметической или логической операции.

V – флаг переполнения при вычислении в дополнительном коде.

S – (Sign), $S = N \oplus V$. Флаг знака. Применяется для определения фактического результата арифметической операции. Устанавливается в 1, если результат арифметической операции меньше нуля.

H – (Halt Carry), флаг половинного переноса. Указывает на переполнения в младшем полубайте (0…4 разряды). Устанавливается в 1, когда происходит перенос из младшего полубайта в старший.

T – (Transfer or Copy), флаг копирования. Предназначен для свободного применения программистом.

I – (Global Interrupt). Общее разрешение прерываний. Для разрешения прерываний флаг I устанавливается в 1.

2.7. Счетчик команд и выполнение программы

Счетчик команд представляет собой регистр, в котором содержится адрес следующей выполняемой команды. Напрямую из программы он недоступен. Размер счетчика зависит от объема памяти программ (12-16 разрядов).

При нормальном выполнении программы содержимое счетчика команд автоматически увеличивается на единицу (или на два в зависимости от выполняемой команды) в каждом машинном цикле. После включения питания и после сброса МК в счетчик команд автоматически загружается стартовый адрес $0000. Как правило, по этому адресу располагается команда безусловного перехода к инициализационной части программы.

Этот порядок нарушается при выполнении команд перехода и возврата из подпрограммы, при возникновении прерываний. При возникновении прерывания в счетчик команд загружается адрес соответствующего вектора прерывания. По адресам векторов прерываний размещаются команды перехода к подпрограммам обработки прерываний.

2.7.1. Выполнение команды

Порядок выполнения команды микроконтроллером приведен на рис. 23.

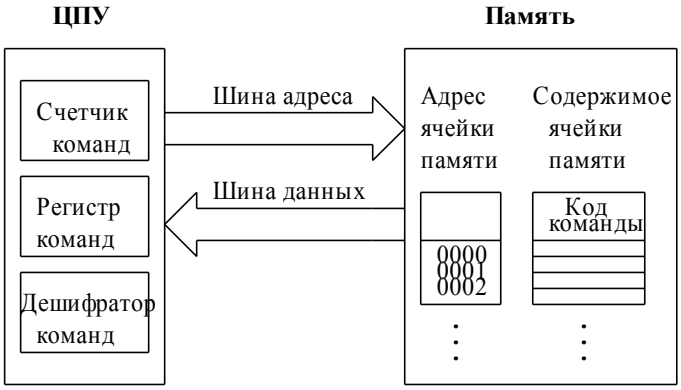


Рис. 23

Значение, указанное счетчиком команд, «выставляется» на шину адреса и в память посылается информация об адресе команды. Из памяти считывается код команды и выставляется на шину данных. ЦПУ считывает код команды из памяти по указанному адресу и помещает его в регистр команд. Дешифратор команд расшифровывает команду и посылает указание в АЛУ для ее выполнения.

В микроконтроллерах AVR используется конвейерная обработка команд. Во время первого машинного цикла происходит выборка команды из памяти программ и ее декодирование. Во время второго цикла эта команда выполняется, а параллельно происходит выборка и декодирование второй команды. Таким образом выполнение команды происходит за один машинный цикл.

В результате выполнения команды, изменяющей содержимое счетчика команд (команды перехода, вызова и возврата из подпрограммы), происходит разрыв в работе конвейера и следствие - задержка выполнения программ мы на 2-4 машинных цикла.

2.8. Команды микропроцессора

2.8.1. Команды типа «проверка/пропуск»

В командах этого типа производится проверка условия, результат которого влияет на выполнение следующей команды. Если условие истинно (лог. 1), следующая команда игнорируется. Используется следующие основные команды:

- Sbis ioreg,bit (**S**kip if **B**it in **I**/O-Register in **S**et) – пропуск следующей команды, если установлен разряд в регистре ввода/вывода
- Sbic ioreg,bit (**S**kip if **B**it in **I**/O-Register is **C**leared) - пропуск следующей команды, если очищен разряд в регистре ввода/вывода
- Sbrs reg,bit (**S**kip if **B**it in **r**egister in **S**et) – пропуск следующей команды, если установлен разряд в регистре reg.
- Sbrc reg,bit (**S**kip if **B**it in **r**egister is **C**leared) – пропуск следующей команды, если очищен разряд в регистре.

2.8.2. Команды условного перехода

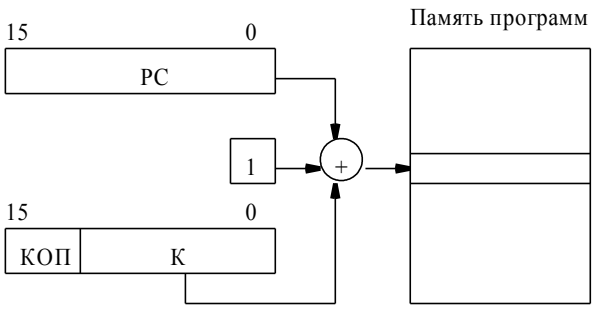
В этих командах производится проверка условия, результат которой влияет на состояние счетчика команд. Если условие истинно, происходит переход по заданному адресу. Если же условие ложно, выполняется следующая команда.

Команды условного перехода имеют ограничение по области действия. Максимальная величина перехода равна -63…+63 команды.

- **Brne** label (**Br**anch if **N**ot **E**qual) – переход, если не равно. Проверяется флаг нуля и переходит на метку label, если флаг сброшен, то есть не ноль. Перед этой командой должна быть команда сравнения или команда вычитания.
- **Breq** label (**Br**anch if **Eq**ual) – переход, если равно. Проверяет флаг нуля и переходит на метку, если флаг установлен.
- **Brbs** bit,label (**Br**anch if **B**it in SREG is **s**et) – переход, если установлен разряд bit в регистре SREG/
- **Brbc** bit,label (**Br**anch if **B**it in SREG is **c**leared) – переход, если сброшен бит в регистре SREG.
- **Brcc** label (**Br**anch if **C**arry is **c**leared) – переход, если сброшен флаг переноса.
- **Brcs** label (**Br**anch if **C**arry is **s**et) – переход, если установлен флаг переноса.

2.8.3. Команды относительного перехода

1. **RJMP** – команда относительного перехода. Относительный переход по адресу, который должен находиться в диапазоне PC±2K слов. Адрес возврата (команда, следующая после команды rjmp), помещается в стек. При выполнении команды содержимое счетчика команд изменяется прибавлением к нему (или вычитанием) некоторого значения K, являющегося операндом команды (рис. 24). В качестве операндов используются метки.



КОП - команда относительного перехода

Рис. 24

2.8.4. Команды косвенного перехода

По команде **IJMP** осуществляется переход по адресу, который находится в регистре Z. Процесс выполнения команды сводится к загрузке содержимого регистра Z в счетчик команд (рис.25).

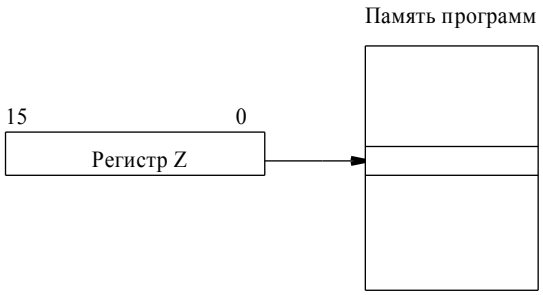


Рис. 25.

Команда не имеет ограничений по области действия.

2.8.5. Команды арифметических операций

ADD – сложение без учета переноса.
Синтаксис: ADD Rd,Rr

Операция: Rd←Rd+Rr

Операнды: r0…r31

Описание: Содержимое регистра Rr прибавляется к содержимому регистра Rd. Результат сохраняется в регистре Rd.

ADC – сложение с учетом переноса.

Синтаксис: ADC Rd,Rr

Описание: содержимое регистра Rr и флаг переноса из регистра SREG прибавляются к содержимому регистра Rd.

ADIW – сложение непосредственного значения со словом, находящимся в регистрах X, Y или Z.

Синтаксис: ADIW R_dl,K ; К – константа. Находится в диапазо–не 0…63.

Операция: R_dh:R_dl← R_dh:R_dl+K

Описание: константа К прибавляется к содержимому регистровой пары R_dh:R_dl. Команда работает с регистрами X, Y или Z.

Например:

```
Adiw   ZL,1   ; прибавить 1 к указателю Z
```

SUB – вычитание без учета переноса.

Синтаксис: SUB Rd,Rr

Операция: Rd←Rd – Rr

Операнды: r0…r31

Описание: содержимое регистра Rr вычитается из содержимого регистра Rd.

SUBI – вычитание непосредственного значения без учета переноса.

Синтаксис: SUBI Rd,const

Операция: Rd←Rd – const

Операнды: r16…r31, константа 0…255

Описание: Восьмибитная константа вычитается из содержимого регистра Rd.

Например:

```
Subi   XL,low($4F23) ; вычитаем младший байт числа $4F23 из XL
Subi  XH,high($4F23) ; вычитаем старший байт числа $4F23 из XH
```

SBC вычитание с учетом переноса.

Синтаксис: SBC Rd,Rr

Операция: Rd←Rd – Rr - C

Операнды: r0…r31

Описание: содержимое регистра Rr и флаг переноса вычитается из содержимого регистра Rd

SBCI – вычитание непосредственного значения с учетом переноса.

Синтаксис: SBCI Rd,const

Операция: Rd←Rd – const – C

Операнды: r16…r31, const=0…255.

Описание: восьмибитная константа и флаг переноса вычитаются из содержимого регистра Rd.

SBIW – вычитание непосредственного значения из слова.

Синтаксис: SBIW R_dl,const

Операция: $R_dh:R_dl \leftarrow R_dh:R_dl - const$

Операнды: XL, YL, ZL, const=0…63

Описание: шестибитная константа вычитается из содержимого регистровой пары X, Y, Z.

Пример:

```
Sbiw   XL,1  ; содержимое регистра X уменьшается на единицу.
```

2.8.6. Команды сравнения

CP – сравнение.

Синтаксис: CP Rd,Rr

Операция: Rd-Rr

Операнды: r0…r31

Описание: сравнение регистров Rd и Rr. При этом регистр Rr вычитается из регистра Rd без сохранения результата (содержимое не изменяется). Влияет на флаги регистра SREG.

Например:

```
Cp   r4,r16 ; сравниваем содержимое регистров r4 и r16/
```

CPC – сравнение с переносом.

Синтаксис: CPC Rd,Rr

Операция: Rd – Rr – C

Операнды: r0…r31

Описание: содержимое регистров Rd и Rr сравнивается путем вычитания. При этом учитывается перенос, полученный в результате выполнения предыдущей операции. Содержимое регистров Rd и Rr не изменяется.

CPI – сравнение с непосредственным значением.

Синтаксис: CPI Rd,const

Операция: Rd – const

Операнды: r16…r31, const=0…255.

Описание: содержимое регистра не изменяется. Изменяются только флаги в регистре SREG.

Пример:

```
Cpi   r16,3  ; сравнение содержимого регистра r16 с числом 3
Breq   gleich ; переход, если равно
```

CPSE – сравнение с пропуском следующей команды, если равно.

Синтаксис: cpse Rd,Rr

Операция: Если Rd=Rr, то PC←PC+2, иначе PC←PC+1

Операнды: r0…r31

Описание: Содержимое регистров сравнивается путем вычитания. И если Rd=Rr, то следующая команда пропускается. Сами регистры и флаги остаются неизменными.

2.8.7. Команды логических операций

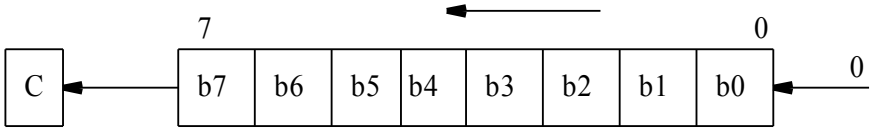
AND – логическое «И».
Синтаксис: AND Rd,Rr
Операция: Rd←Rd×Rr
Операнды: r0…r31
Описание: результат записывается в регистр Rd.
ANDI – логическое «И» с непосредственным значением.
Синтаксис: ANDI Rd,const
Операция: Rd←Rd×const
OR – логическое «ИЛИ».
ORI – логическое «ИЛИ» с непосредственным значением.
EOR – исключающее «ИЛИ».
Операция: Rd ← Rd ⊕ Rr
Логические операции выполняются *поразрядно.*
Например, логическое умножение (маска разряда):

```
  1 0 1 1 0 1 1 0
  0 0 0 0 1 1 0 0
  ---------------
  0 0 0 0 0 1 0 0
```

Команда «ИЛИ» применяется для установки определенных разрядов в 1:

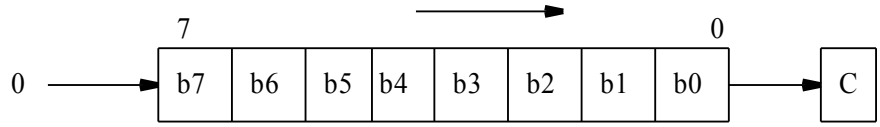
```
          3 2 1 0
  1 0 1 1 0 1 0 1
+
  0 0 0 0 1 0 1 0
  ---------------
  1 0 1 1 1 1 1 1
```

В приведенном выражении 1 и 3 разряды установлены в 1.
LSL – логический сдвиг влево.
Синтаксис: lsl Rd
Операция:



Операнды: r0…r31
Описание: все разряды регистра Rd сдвигаются на одну позицию влево; флагу переноса присваивается значение разряда 7, а в разряд 0 записывается логический 0.
Пример:
Lsl r0 ;сдвиг влево содержимого регистра r0, что эквивалентно умножению r0 на два.

LSR – логический сдвиг вправо
Синтаксис: lsr Rd
Операция:



Операнды: r0…r31
Пример:
Lsr Rd ; сдвиг вправо содержимого регистра r0, что эквивалентно делению содержимого r0 на два.

2.8.8. Команды загрузки памяти

LPM – загрузка данных из памяти программ.
Синтаксис: lpm Rd,Z
Операция: Rd←Z
Операнды: r0…r31
Описание: байт памяти программ, адресуемый указателем Z, загружается в регистр Rd

В памяти программ располагаются двухбайтные числа (16 разрядов), а команда LPM считывает однобайтные числа. Команда функционирует следующим образом (рис. 26):

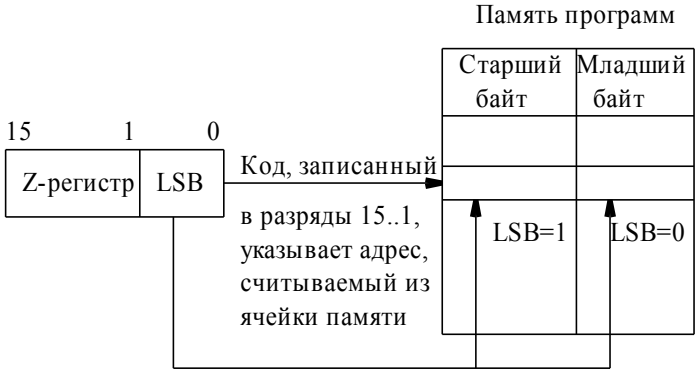


Рис. 26

Поэтому, если мы хотим считать данные, расположенные в памяти программ по адресу 144_d, то мы должны сдвинуть число 144 влево. В разряде LSB будет записан 0 и мы считываем младший байт. Если затем прибавить 1 к содержимому указателя, то мы считаем старший байт.
Пример:
Ldi ZH,high(144<<1) ; загрузка адреса в регистр Z со сдвигом влево на один разряд
Ldi ZL,low(144<<1)
Lpm r16,Z ; считывание младшего байта данных, на-

ходящихся по адресу, указанному в
регистре Z, в регистр r16
adiw ZL,1 ; к Z прибавляем 1
lpm r17,Z ; считываем старший байт данных

LPM Rd,Z+

То же. Но после пересылки байта в регистр Rd содержимое регистра Z увеличивается на 1.

ST – запись содержимого регистра РОН в SRAM с помощью указателя X,Y,Z.

Синтаксис: st X,Rr

Операция: X←Rr

Операнды: r0…r31

Описание: содержимое регистра Rr сохраняется в ячейке SRAM по адресу, находящемуся в регистре X. Содержимое X остается неизменным.

Синтаксис: st X+,Rr

Операция: X←Rr, X←X+1

Операнды: r0…r31

Описание: указатель X увеличивается на 1

STS – прямая запись в SRAM.

Синтаксис: sts k,Rr

Операция: k←Rr

Операнды: r0…r31

Описание: загрузка в ячейку SRAM содержимого регистра Rr по абсолютному 16-битному адресу k. k=0…65535

LD – загрузка регистра РОН из SRAM с помощью регистров X,Y,Z.

Синтаксис: а) ld Rd,X
б) ld Rd,X+

Операция: а) Rd←X
б) Rd←X, X←X+1

Операнды: r0…r31

Описание: байт из ячейки SRAM, адресуемый регистром X, загружается в регистр Rd. В случае (а) содержимое X остается неизменным, в случае (б) – увеличивается на 1.

MOV – копирование регистра.

Синтаксис: mov Rd,Rr

Операция: Rd←Rr

Операнды: r0…r31

Описание: содержимое регистра Rr копируется в регистр Rd. Содержимое регистра Rr остается неизменным.

IN – загрузка из порта ввода/вывода в регистр РОН.

Синтаксис: in Rd,P ; P – порт

Операция: Rd←P

Операнды: r0…r31

Описание: загрузка в регистр Rd содержимого порта ввода/вывода P

Например:

In r16,PortB ; считываем состояние порта B в регистр r16

OUT – передача содержимого регистра в порт ввода/вывода
Синтаксис: out P,Rr
Операция: P←Rr
Операнды: r0…r31
Например:
Out PortB,r16 ; передаем в портB содержимое регистра r16
INC – инкремент
Синтаксис: inc Rd
Операция: Rd←Rd+1
Операнды: r0…r31
Описание: к содержимому регистра Rd прибавляется 1. Флаг переноса не изменяется.
DEC – декремент
Синтаксис: dec Rd
Операция: Rd←Rd – 1
Операнды: r0…r31
Описание: из содержимого регистра Rd вычитается 1.

2.9. Пример включения светодиодов

На рис. 27 приведена схема включения светодиодов, подключенных к выводом B0…B2 и управляемых кнопками K1…K3.

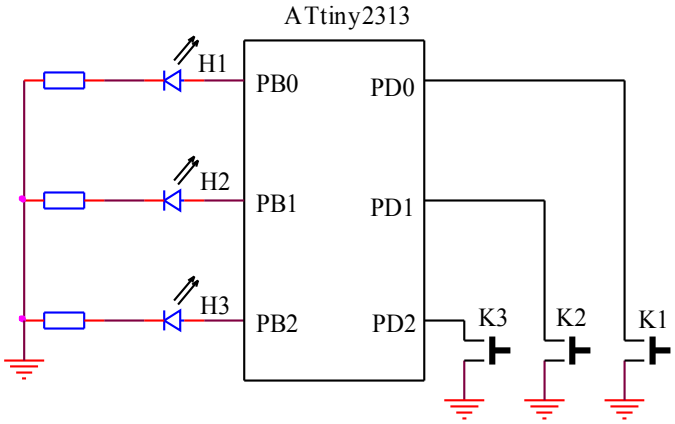


Рис. 27

При нажатии одной из кнопок загорается соответствующий светодиод.
На рис. 28 приведена схема алгоритма, описывающая поведение заданного устройства.

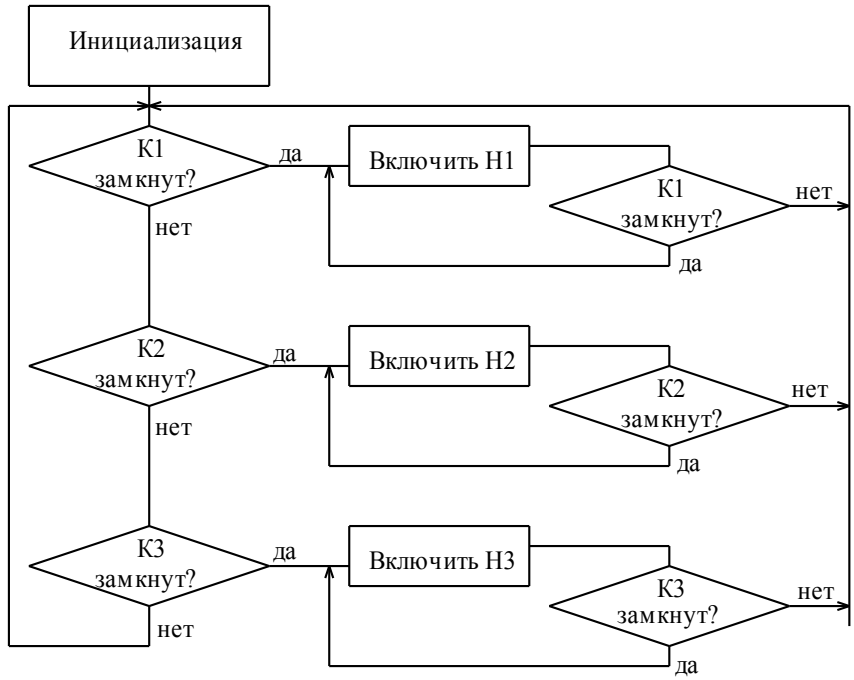


Рис. 28

```
    .device  ATtiny2313
    .nolist
    .include  "…\tn2313def.inc"
    .list
    .def  tmp=r16
    .cseg
    .org  0
    rjmp  Reset
Reset:
    ser   tmp
    out   ddrB,tmp  ;порт B – выходы
    ldi   tmp,0b11111000
    out   ddrD,tmp  ;D0 … D2 – входы;
    ldi   tmp,0b00000111
    out   portD,tmp ;D0 … D2 – подтяжка
Start:
    clr   tmp
    out   portB,tmp ;все СИД выключены
    sbis   PinD,0 ;
    rjmp   on1 ;если К1 замкнут, то переходим на on1
    sbis    PinD,1
    rjmp   on2
    sbis   PinD,2
    rjmp   on3
    rjmp   Start
```

```
On1:
    ldi   tmp,0b00000001 ;включение 1-го светодиода
    out   PortB,tmp
    sbis   PinD,0       ;если замкнуто, то переходим на On1
    rjmp   on1
    rjmp   Start

On2:
    ldi   tmp,0b00000010   ;включение 2-го светодиода
    out   PortB,tmp
    sbis   PinD,1           ;если замкнуто, то переходим на On2
    rjmp   on2
    rjmp   Start

On3:
    ldi   tmp,0b00000100   ;включение 3-го светодиода
    out   PortB,tmp
    sbis  PinD,2           ;если замкнуто, то переходим на On3
    rjmp   on3
    rjmp   Start
```

2.10. Тактовый генератор

Микроконтроллеры могут работать с встроенным генератором с внутренней или внешней RC-цепочкой, с внешним керамическим или кварцевым резонатором, от сигнала внешней синхронизации. Выбор режима осуществляется программированием конфигурационных ячеек Fuse Bit (**Fu**nction **Se**lect Bits). Это биты CKSEL3…0.

Схема организации тактового генератора приведена на рис. 29.

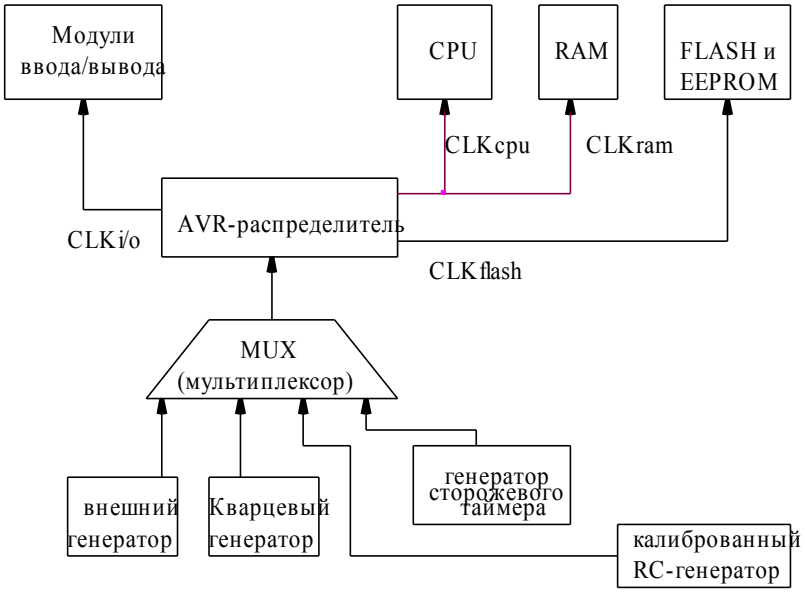


Рис. 29

2.10.1. Выбор источника тактовых импульсов

Выбор источника тактовых импульсов производится с помощью конфигурационных ячеек CKSEL3…0. Зависимость режима работы МК от конфигурационных ячеек приведена в таблице 2.

Табл. 2

Режим работы	CKSEL3…0
Внешний сигнал синхронизации	0 0 0 0
Внешний RC-генератор, 4 МГц	0 0 1 0
Внутренний RC-генератор, 8МГц	0 1 0 0
Генератор сторожевого таймера, 128 кГц	0 1 1 0
Внешний кварцевый или керамический резонатор	1 0 0 0 – 1 1 1 1

Внешний генератор подключается к выводу XTAL1 (рис. 30):

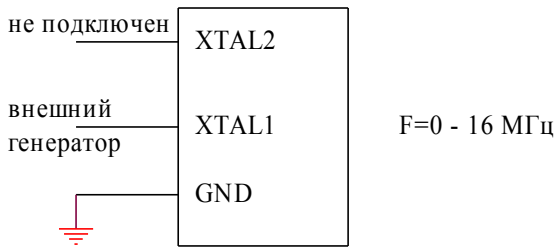


Рис. 30

Кварцевый резонатор подключается к выводам XTAL1, XTAL2 (рис. 31).

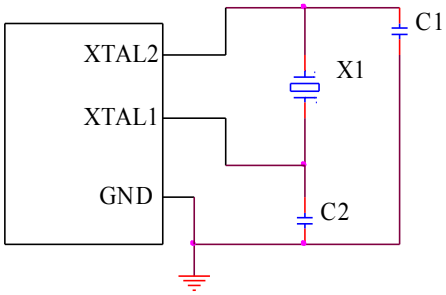


Рис. 31

При подключении кварцевого резонатора ячейки CKSEL3…1 устанавливаются в зависимости от частоты кварца (таблица 3).

Табл. 3

CKSEL3…1	Частота резонатора, МГц	C1, C2 пФ
1 0 0	0,4 – 0,9	-
1 0 1	0,9 – 3,0	12 – 22
1 1 0	3,0 – 8,0	12 – 22
1 1 1	8,0 – 16,0	12 – 22

Конфигурационные ячейки CKSEL0 и SUT1…0 управляют задержкой включения микроконтроллера. Зависимости приведены в таблице 4.

Табл. 4

CKSEL0	SUT 1…0	t_s, такты	t_r, мс	Режим работы
0	00	258	14СК+4 мс	Керамический резонатор, быстрое нарастание питания
0	01	258	14СК+65мс	Керамический резонатор, медленное нарастание питания
0	10	1к	14СК	Керамический резонатор. Схема BOD включена
0	11	1к	14СК+65мс	Керамический резонатор, быстрое нарастание питания
1	00	1к	14СК+65мс	Керамический резонатор, медленное изменение питания
1	01	16к	14СК	Кварцевый резонатор. Схема BOD

				включена
1	10	16к	14СК+4 мс	Кварцевый резонатор, быстрое нарастание питания
1	11	16к	14СК+6 5мс	Кварцевый резонатор, медленное нарастание питания

В таблице 4 обозначено:

- СК – период тактовой частоты
- (0) ☑ CKSEL0 (запрограммировано)
- (1) ☐ CKSEL1 (не запрограммировано)

2.10.2. Управление тактовой частотой

Микроконтроллеры имеют возможность программного уменьшения частоты сигнала, поступающего от тактового генератора. При этом замедляется работа всех устройств микроконтроллера.

Для управления предделителем тактового сигнала предназначен регистр ввода/вывода CLKPR.

Регистр CLKPR

7	6	5	4	3	2	1	0
CLKPSE	-	-	-	CLKPS3	CLKPS2	CLKPS1	CLKPS0

Разряд CLKPSE служит для разрешения изменения частоты тактового сигнала (логическая 1), а разряды CLKPS3…0 задают коэффициент деления предделителя (таблица 5).

Табл. 5

CLKPS3	CLKPS2	CLKPS1	CLKPS0	Коэффициент деления
0	0	0	0	1
0	0	0	1	2
0	0	1	0	4
0	0	1	1	8
0	1	0	0	16
0	1	0	1	32
0	1	1	0	64
0	1	1	1	128

1	0	0	0	256

Для изменения содержимого разрядов CLKPS3…0 надо записать в разряд CLKPSE логическую 1, а в разряды CLKPS3…0 логический 0. Затем в течение следующих четырех машинных циклов занести требуемое значение в разряды CLKPS3…0, при этом разряд CLKPSE сбрасывается в 0.

Начальное состояние разрядов CLKPS3…0 определяется конфигурационной ячейкой CKDIV8. Если она не запрограммирована (логическая 1), при запуске микроконтроллера в разрядах CLKPS3…0 содержится значение 0000. То есть деление частоты отсутствует. Если же ячейка CKDIV8 запрограммирована (логический 0), стартовым значением разрядов CLKPS3…0 является значение 0011 (коэффициент деления 8).

2.11. Сброс микроконтроллера

Реинициализация, или «сброс», переводит микроконтроллер в определенное устойчивое состояние. Сброс может быть вызван следующими событиями:

- Включение напряжения питания;
- Подача сигнала низкого уровня (аппаратный сброс).
- Тайм-аут сторожевого таймера.
- Падение напряжения питания ниже заданной величины.
- Сброс по интерфейсу JTAG.

При наступлении любого из указанных событий во все регистры ввода/вывода заносятся их начальные значения, а в счетчик команд загружается адрес вектора сброса. По этому адресу должна находиться команда безусловного перехода RJMP на начало программы.

Разрешение использования вывода микроконтроллера для внешнего сброса задается конфигурационной ячейкой RSTDISBL. Если она не запрограммирована (1), контакт PA2 – вывод сброса. Если запрограммирована (0), контакт PA2 – порт ввода/вывода.

Логика работы сброса следующая. При наступлении события, приводящего к сбросу микроконтроллера, формируется внутренний сигнал сброса. Одновременно запускается таймер формирования задержки сброса. По истечении определенного времени внутренний сигнал сброса снимается и начинается выполнение программы.

2.11.1. Сброс по включению питания

Схема сброса по включению питания удерживает микроконтроллер в состоянии сброса до тех пор, пока напряжение питания не превысит некоторого порогового значения. По его достижению схема запускает таймер задержки сброса. После окончания задержки запускается микроконтроллер (рис. 32).

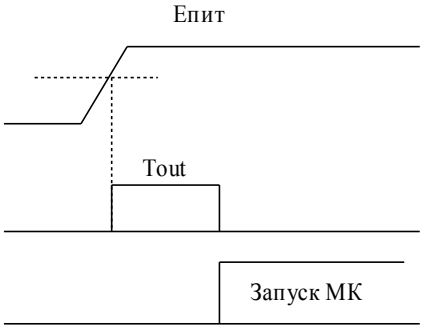


Рис. 32.

2.11.2. Аппаратный сброс

Аппаратный сброс производится подачей на вывод «Сброс» сигнала низкого уровня. При достижении на выводе «Сброс» порогового напряжения Vrst запускается таймер задержки сброса, формирующий задержку Tout. Затем микроконтроллер запускается (рис. 33).

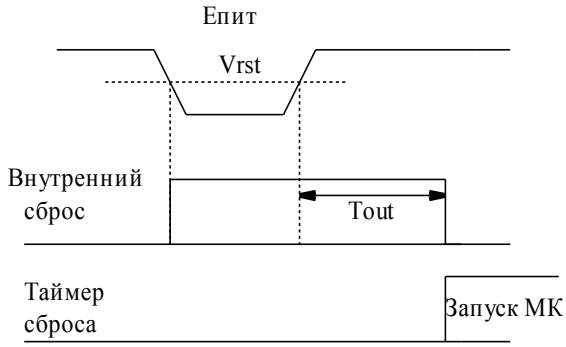


Рис. 33

2.11.3. Сброс при снижении напряжения питания

Микроконтроллеры имеют схему BOD (**B**rown – **O**ut **D**etection), которая отслеживает величину напряжения источника питания. При снижении напряжения питания ниже некоторого значения схема BOD переводит микроконтроллер в состояние сброса. Когда напряжение питания вновь увеличится до порогового значения, запускается таймер задержки и через время Tout запускается микроконтроллер (рис. 34).

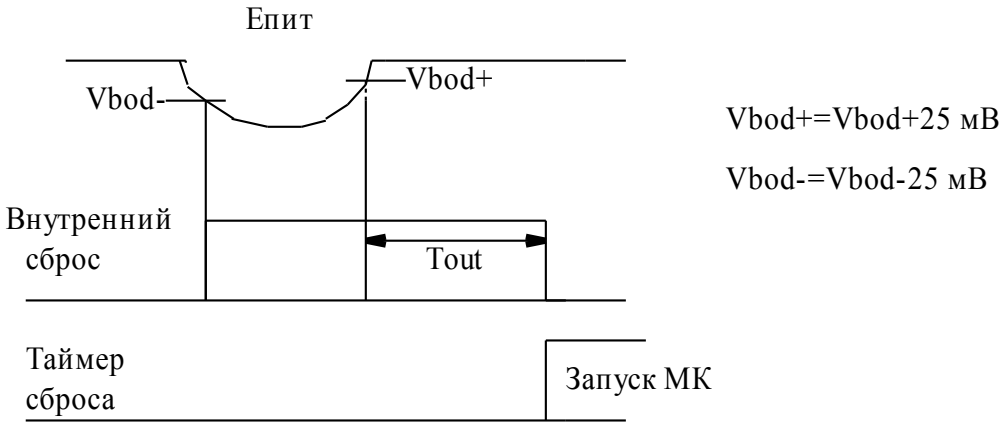


Рис. 34

Уровень срабатывания схемы BOD задается конфигурационными ячейками BODLEVEL2…0 (таблица 6).

Табл. 6

BODLEVEL2…0	Порог срабатывания, В
1 1 1	Схема BOD выключена
1 1 0	1,8
1 0 1	2,7
1 0 0	4,0

2.11.4. Схема сброса

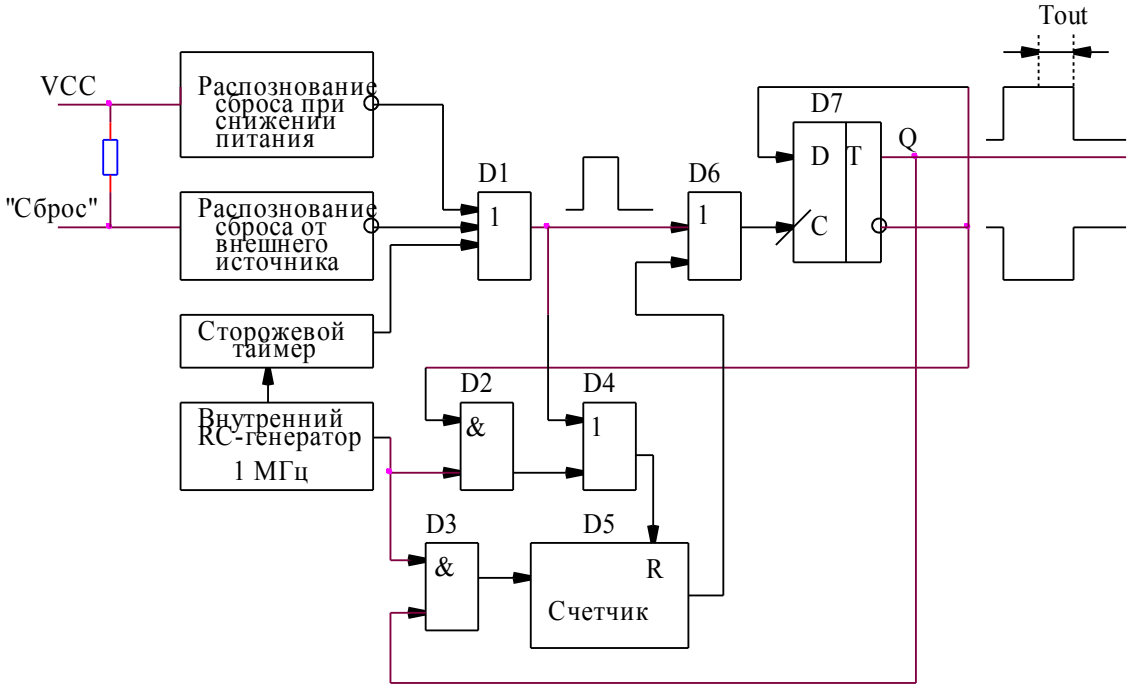


Рис. 35

Источник сброса определяется вентилем D1. Он устанавливаетD-триггер D7 в состояние логической 1 и сбрасывает двоичный счетчик D5. Счетчик удерживается в нулевом состоянии, пока активен источник сброса. Как только выходной сигнал с D1 изменяется на логический 0, начинает работать двоичный счетчик, так как на выходе Q микросхемы D7 установлена логическая 1. Когда счетчик достигнет заранее заданного состояния, его выходной сигнал переключает D-триггер в состояние логического 0 и внутренний сигнал сброса будет снят. Задержка Tout необходима, потому что внутреннему генератору нужно несколько миллисекунд, чтобы колебания стали стабильны.

2.12. Конфигурационные ячейки

Конфигурационные ячейки доступны при параллельном программировании. Они определяют некоторые параметры конфигурации микроконтроллера (Таблица 7).

Табл. 7

Состояние	Имя	Описание

по умолчанию	параметра	
(1) ☐	EESAVE	Определяет влияние команды «Стирание кристалла» на EEPROM-память. 0 – не стирает, 1 – стирает
(1) ☐	WDTON	Определяет режим работа сторожевого таймера. 0 – всегда включен, 1 – может быть выключен программно
(1) ☐	BODLEVEL2…0	Определяет порог срабатывания схемы (по умолчанию 111 - выключен)
(1) ☐	RSTDISB	1 – внешний сброс разрешен 2 – внешний сброс запрещен
(0) ☑	CKDIV8	Определяет начальное состояние делителя тактового сигнала: 0 – делит часоту на 8, 1 – не делит
(1) ☐	CKOUT	Определяет состояние выходного буфера тактового сигнала: 0 – тактовый сигнал подключен к выводу МК, 1 – отключен
(0) ☑	SUT0	Определяет длительность задержки сброса Tout
(1) ☐	SUT1	
(0) ☑	CKSEL0	Определяет режим работы тактового генератора и длительность задержки сброса Tout (по умолчанию включен внутренний RC-генератор на 4 МГц)
(1) ☐	CKSEL1	
(0) ☑	CKSEL2	
(0) ☑	CKSEL3	

2.13. Формирование временных задержек

В основе формирования задержки лежит задание количества тактов и постепенное уменьшение этого значения до нуля. Если число больше 255, то мы должны распределить это число по нескольким регистрам и последовательно их обрабатывать. Младший байт при этом уменьшается на 1 до тех пор, пока его значение не изменится с 00 до FF. При этом устанавливается флаг переноса. Затем уменьшается на 1 следующий по порядку старший байт и т.д. Пример уменьшения приведен в таблице 8.

Табл. 8

Старший байт	Младший байт	Перенос
0x1A	0x02	Нет
0x1A	0x01	Нет
0x1A	0x00	Нет
0x1A	0xFF	Да
0x19	0xFF	Нет
0x19	0xFE	да

Пусть мы хотим получить секундную задержку при частоте тактового генератора 4 МГц. Период тактовой частоты равен

$$T=1/Ft=1/4\ Мгц=0{,}25\ мкс.$$

Для создания задержки 1с нам потребуется $1с*10^6$мкс/0,25мкс=$4*10^6$ импульсов тактовой частоты.

Если в цикле содержится столько команд, что они выполняются за 5 тактов, то количество подсчитываемых импульсов равно:

$$4*10^6:5=800\ 000.$$

В шестнадцатеричном коде это число запишется 0x0C3500. Делим это число на группы по два разряда:

0x00; 0x35; 0x0C.

Загружаем эти числа в рабочие регистры:

```
Ldi   delay1,0x00
Ldi   delay2,0x35
Ldi   delay3,0x0C
```

Для организации цикла воспользуемся командами:

Sbci reg,number (**Su**b**t**ract **I**mmediate with **C**arry) – вычитание константы number (0…255) и флага переноса из регистра reg, сохраняя результат в этом же регистре.

Subi reg,number ; вычитание константы number без учета переноса.

Например, команда cbci delay2,0 вычитает 1 из регистра delay2, если флаг переноса установлен, и 0 – в противном случае.

Цикл задержки в 1 с запишется следующим образом:

```
Loop:
1 такт   subi   delay1,1 ; вычитаем 1 из delay1
1 такт   sbci   delay2,0 ; вычитаем 1 из delay2, если флаг C=1
1 такт   sbci   delay3,0 ; вычитаем 1 из delay3, если флаг C=1
```

2 такта brcc loop ; возвращаемся к началу (loop), если флаг C сброшен.

2.14. Бегущий огонек

В схеме последовательно зажигается один из восьми светодиодов. Каждый светодиод находится во включенном состоянии 1 с. Схема устройства приведена на рис. 36.

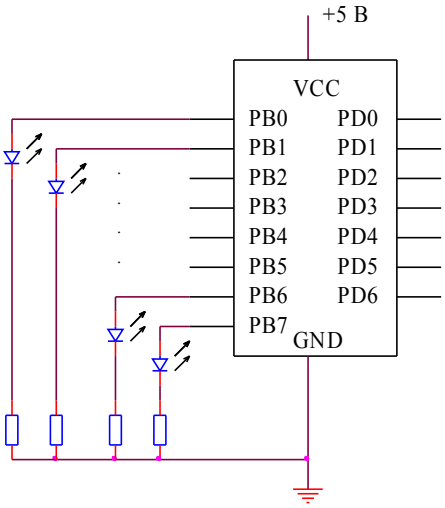


Рис. 36

На рис. 37 показан алгоритм работы устройства.

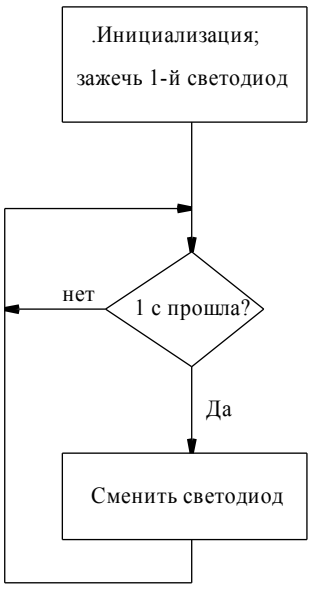


Рис. 37

Программа, написанная на языке Ассемблер приведена ниже.

```
.device  ATtiny2313
```

```
    .nolist
    .include  "…\tn2313def.inc"
    .list
    .def  tmp=r16
    def  Delay1=r17
    def  Delay2=r18
    def  Delay3=r19
    .cseg
    .org  0
    rjmp  Reset
Reset:
    ser   tmp
    out   DDRB,tmp ; B0…B7 – выходы
    out   DDRD,tmp ; D0…D7 – выходы
    ldi   tmp,0b00000001 ; включаем 1-й светодиод
    Out   PortB,tmp
    ; обработка длительности свечения
    Start:
    Ldi   delay1,0x00
    Ldi   delay2,0x35
    Ldi   delay3,0x0C
    Loop:
    subi   delay1,1 ; вычитаем 1 из delay1
    sbci   delay2,0 ; вычитаем 1 из delay2, если флаг C=1
    sbci   delay3,0 ; вычитаем 1 из delay3, если флаг C=1
    brcc   loop
    ; меняем светодиод
    In   tmp,PortB  ; считываем число из порта B (00000001)
    Lsl   tmp         ; сдвиг влево     (00000010)
    Brcc   PC+2     ; переход, если сброшен флаг переноса
    Ldi   tmp,0b00000001
    Out   PortB,tmp
    Rjmp   Start
```

2.15. Подпрограммы

Встречаются случаи, когда из разных мест программы необходимо выполнить одно и то же действие. Тогда это действие удобно записать как *подпрограмму* с соответствующей меткой. Пусть требуется в программе периодически запускать двигатель. Тогда в главной программе вставляем команду **rcall** с соответствующей меткой, которая и будет запускать каждый раз двигатель (рис. 37).

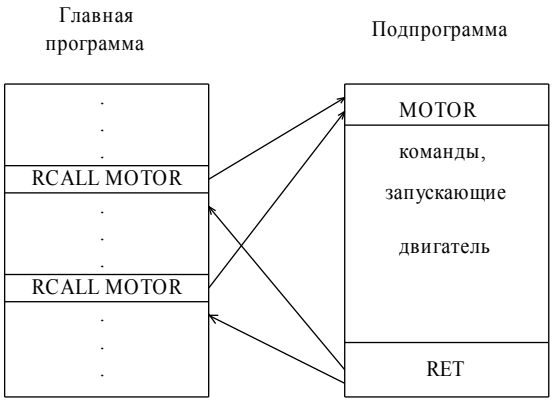


Рис. 37

Выполнение подпрограммы заключается в следующем. В счетчике команд (Program Counter – PC) хранится адресм следующей выполняемой команды. При выполнении последовательных команд содержимое счетчика увеличивается на 1 и он указывает на следующую команду прграммы. При вызове подпрограммы содержимое счетчика команд помещается на вершину стека (область памяти, в которой хранится адрес возврата из подпрограммы). Загрузка в стек происходит с конца области стека в порядке уменьшения адресов.

Когда выполняется команда возврата из подпрограммы, число, находящееся на вершине стека, загружается обратно в счетчик команд, и микроконтроллер возвращается к выполнению программы, следующей за командой вызова подпрограммы.

В основной программе необходимо указать, в каком месте ОЗУ следует поместить стек. В качестве вершины стека указывается адрес верхней границы ОЗУ.

Для работы со стеком предусмотрены два регистра ввода/вывода SPL и SPH, которые являются регистрами указателя стека (**s**tack **p**ointer). SPL – младший байт (lou), SPH – старший байт (hight).

Адрес верхней границы ОЗУ для заданного типа микроконтроллера хранится в константе RAMEND, определенной во включаемом файле. Для модели AT-tiny2313 размер константы равен 1 байт, поэтому используется только две команды:

```
Ldi   tmp,LOW(RAMEND)
Out   SPL,tmp
```

Для микроконтроллеров, имеющий больший объем памяти, необходимо использовать еще две строчки:

```
Ldi   tmp,high(RAMEND)
Out   SPH,tmp
```

Эти команды записываются в секции Reset до первого вызова подпрограммы.

2.16. Формирование временных интервалов с помощью таймера/счетчика

В микроконтроллере имеется счетный регистр ввода/вывода TCNT0. Он автоматически считает в прямом направлении, сбрасывается в ноль после достижения значения 255 ($FF). Этот таймер используется для выполнения операций, связанных со временем.

Для конфигурирования таймера используется регистр TCCR0B. С помощью битов 0…2 этого регистра задается режим работы таймера.

Регистр TCCR0B

2	1	0
CS02	CS01	CS00

При различных значениях в ячеках CS00-CS02 устанавливаются различные коэффициенты деления тактовой частоты (таблица 9).

CS02	CS01	CS00	Описание
0	0	0	Таймер остановлен
0	0	1	Частота таймера равна тактовой частоте СК
0	1	0	СК/8
0	1	1	СК\64
1	0	0	СК/256
1	0	1	СК/1024
1	1	0	Считает внешние импульсы по отрицательному фронту на выводе T0
1	1	1	Считает внешние импульсы по положительному фронту на выводе T0

Конфигурирование таймера производится командами, расположенными в области Reset. Например:

```
Ldi   tmp,0b00000101
Out   TCCR0B,tmp   ; установлен режим СК/1024
```

По этим командам частота таймера задается в 1024 раза меньше тактовой частоты. Если Ft= 4 МГц, то $F_{таймера}$=4000 кГц:1024=3,9 кГц.

При использовании таймера программа «Бегущий огонек» запишется следующим образом.

```
.device  ATtiny2313
.nolist
.include  "…\tn2313def.inc"
.list
```

```
    .def  tmp=r16
    def  mark=r17
    def  counter=r18
    .cseg
    .org  0
    rjmp  Reset
Reset:
    Ldi   tmp,low(RAMEND) ; загрузка стека
    Out   SPL,tmp
    ser   tmp
    out   DDRB,tmp ; B0…B7 – выходы
    out   DDRD,tmp ; D0…D7 – выходы
    ldi   tmp,0b00000001 ; включаем 1-й светодиод
    Out   PortB,tmp
    Ldi   tmp,0b00000101 ; конфигурирование таймера
    Out   TCCR0B,tmp     ; F таймера= f такт/1024
    Ldi   mark,200       ; загрузка в регистр mark числа 200
    Ldi   counter,4      ; загрузка в регистр counter числа 4
      ; обработка длительности свечения
    Timer:
    Clr   tmp
    In   tmp,TCNT0       ;считываем состояние таймера в tmp
    Cp   tmp,mark        ; сравниваем числа в tmp и в mark
    Brbs   4,Timer       ; если результат отрицательный, то есть число в TCNT0
меньше числа в mark, возвращаемся к Timer
    Clr   tmp
    Out   TCNT0,tmp    ; сброс TCNT0, если результат положительный, то есть чис-
ло в TCNT0 больше числа в mark
    Dec   counter        ; уменьшаем число в регистре counter на 1
    Brne   Timer         ; переходим на метку, если counter не равен 0.
    ; меняем светодиод
    In   tmp,PortB  ; считываем число из порта B (00000001)
    Lsl   tmp       ; сдвиг влево     (00000010)
    Brcc   PC+2     ; переход, если сброшен флаг переноса
    Ldi   tmp,0b00000001
    Out   PortB,tmp
    Rjmp   Timer
```

Подсчитаем время задержки:

$T_{таймера}$=mark*(1/3,9 кГц) =200*0,25 мс=50 мс

Время цикла с учетом того, что цикл подпрограммы Timer равен 5 тактов, получим:

50 мс*5=0,25 с

С учетом регистра counter общее время равно

0,25 с*5=1,25 с.

2.17. Сторожевой таймер

Сторожевой таймер обозначается WDT – **W**atch**d**og **t**imer. Это встроенный таймер, работающий от независимого генератора с частотой 1 МГц. Он служит для повышения надежности программ. Если программа зависла или зациклилась, то сторожевой таймер через некоторое время сбросит ее и начнет выполнять сначала. Чтобы избежать непреднамеренного сброса микроконтроллера, сторожевой таймер надо периодически программно сбрасывать, причем раньше, чем он сбросит микроконтроллер.

Для сброса сторожевого таймера служит команда Wdr – **r**eset the **w**atch**d**og timer.

Управление таймером осуществляется регистром ввода/вывода WDTCR.

Регистр WDTCR

7	6	5	4	3	2	1	0
WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0

Биты WDP3…0 устанавливают время сброса сторожевого таймера (таблица 10).

Табл. 10

WDP3…0	Timer
0000	16 мс
0001	32 мс
0010	64 мс
0011	0,125 с
0100	0,25 с
0101	0,5 с
0110	1,0 с
0111	2,0 с
1000	4,0 с
1001	8,0 с

WDE (3-й бит) – разрешение включения таймера. 1 – таймер включен, 0 – таймер выключен.

WDCE (4-й бит) – защита от выключения таймера или изменения времени сброса. Выключение таймера или установление тайм-аута можно осуществить только при установленном разряде WDCE, который аппаратно сбрасывается через 4 машинных цикла.

Конфигурационной ячейкой WDTON сторожевой таймер может быть установлен в одно из состояний: WDTON=1 – сторожевой таймер может быть выключен

программно и WDTON=0 – сторожевой таймер включен постоянно и не может быть выключен. Разряд WDE всегда читается как 1.

Уровень 1. WDTON=1.

Выключение. Для выключения таймера записать одной командой логическую 1 в разряды WDE и WDCE и в течение следующих четырех машинных тактов записать логический 0 в разряды WDE и WDCE

Установка. Для установления периода тайм-аута надо одной командой записать логическую 1 в разряды WDE и WDCE и в течение следующих четырех тактов записать в WDE логическую 1, в WDCE - логический 0, в WDP3…0 – требуемые значения.

Уровень 2. WDTON=0.

Для изменения тайм-аута надо одной командой записать логическую 1 в разряды WDE и WDCE и в течение следующих четырех тактов записать требуемые значения в WDP3…0, одновременно сбрасывая разряд WDCE.

Разряды WDIF, WDIE определяют реакцию на прерывания. Перед началом установки тайм-аута необходимо сбросить сторожевой таймер командой WDR.

Пример фрагмента программы включения сторожевого таймера на 4 с. (Уровень 1).

```
Wdr
Ldi   tmp,0b00011000; WDCE=1, WDE=1.
Out   wdtcr,tmp
Ldi   tmp,0b00101000 ; WDE=1, WDCE=0, WDP3…0=1000
Out   wdtcr,tmp
```

Пример выключения сторожевого таймера (Уровень 1).

```
Wdr
Ldi   tmp,0b00011000 ; WDE=1, WDCE=1
Clr   tmp
Out   wdtcr,tmp ; WDE=0, WDCE=0
```

2.18. Создание двух задержек 0,5 мс и 500 мс с помощью регистров X, Y

Запишем задержки в две подпрограммы. Причем вторая подпрограмма будет использовать первую.

```
Rcall   d05ms   ; вызов подпрограммы задержки 0,5 мс
.
.
.
Rcall   d500ms ; вызов подпрограммы задержки 500 мс
.
.
.
    ; подпрограмма задержки на 0,5 мс
D05ms:
Wdr
```

```
Ldi   YL,low(497) ; загрузка в регистр YH:YL константы 497
Ldi   YH,high(497)  ; загрузка в YH:YL константы 497
D05_1:
Sbiw   YL,1        ; вычитание  из содержимого YH:YL единицы
Brne   d05_1    ; если сброшен флаг нуля Z (результат предыдущей операции не равен 0), то переход на d05_1, если 0 – переход на следующую команду
Ret              ; возврат из подпрограммы
; подпрограмма задержки на 500 мс
D500ms:
Ldi   YL,low(1000) ; загрузка в регистр YH:YL константы 1000
Ldi   YH,high(1000)  ; загрузка в YH:YL константы 1000
D500_1:
Rcall d05ms      ; вызов подпрограммы задержки на 0,5 мс
Sbiw   XL,1
Brne   d500_1
Ret
```

2.19. Регистр управления микроконтроллером MCUCR

Регистр управления MCUCR (**M**icro**c**ontroller **U**nit **C**ontrol **R**egister) содержит разряды управления общими функциями микроконтроллера. Он управляет «спящим» режимом, характером срабатывания внешних прерываний, подтягивающими резисторами. Регистр находится в области ввода/вывода и доступен для чтения и записи.

Регистр MCUCR

7	6	5	4	3	2	1	0
PUD	SM1	SE	SM0	ISC11	ISC10	ISC01	ISC00

PUD (7-й разряд) – осуществляет общее управление (для всех выводов портов) подтягивающими резисторами. Если PUD=0, состояние подтягивающих резисторов определяется разрядом PortXn для каждого входа порта. Если же PUD=1, подтягивающие резисторы отключаются от всех выводов микроконтроллера.

SE (**S**leep **E**nable, 5-й разряд) – разрешение спящего режима. Разряд должен быть установлен в 1 для того, чтобы сделать эффективной команду sleep. Установка в 1 делается непосредственно перед командой sleep.

С помощью разрядов SM0…1 (Sleep Mode – спящий режим) осуществляется выбор режима микроконтроллера, в который он перейдет после команды sleep (таблица 11).

Табл. 11

SM1	SM0	Режимы
0	0	Idle
0	1	ADC

		Noise Reduction
1	0	Power Down
1	1	Power Save

2.19.1. «Спящие режимы» центрального процессора

Если микроконтроллер питается от аккумуляторов или батарей, то важно уменьшить энергопотребление. Для этого можно снизить напряжение питания или выбрать низкую тактовую частоту, так как ток потребления КМОП-схем пропорционален тактовой частоте. Кроме того, микроконтроллер можно перевести в «спящий режим» (sleep mode). Типичное значение тока потребления при отключенном сторожевом таймере приведено в таблице 12.

Табл. 12

Uпит, В	Fт, МГц	Активный, мА	Ждущий мА	Пониженное энергопотребление, мкА
5	12	9	2,4	0,75
5	2	4	1	0,75
3,3	6	2,5	0,7	0,1
3,3	2	1,8	0,4	0,1

Для перевода микроконтроллера в один из режимов пониженного энергопотребления надо разряд SE регистра MCUCR установить в логическую 1, а затем выполнить команду sleep.

Когда во время режима sleep происходит прерывание, центральный процессор выходит из спящего режима, выполняет подпрограмму обработки прерывания и продолжает выполнение программы с команды, следующей после команды sleep. Если режим sleep должен быть отключен в результате прерывания по уровню сигнала, низкий уровень должен сохраняться более 16 мс для перехода генератора в рабочий режим.

В микроконтроллере может быть выбран один из режимов:

1. *ждущий режим.* (Idle Mode). Работа центрального процессора приостанавливается, но таймер/счетчик, сторожевой таймер, система прерываний и тактирования остаются активными. Благодаря этому центральный процессор может быть возвращен в активный режим работы сторожевым таймером, внешним прерыванием.

2. *режим пониженного энергопотребления.* (Power Down Mode). Системный генератор, а значит и весь микроконтроллер находится в отключенном состоянии. В таком режиме с помощью внутреннего RC-генератора может включаться только сторожевой таймер со своим собственным обеспечением тактовой частоты.

Если сторожевой таймер отключен, то в нормальное состояние микроконтроллер может перевести сигнал сброса на выводе RESET или внешнее прерывание.

3. *Режим снижения шумов АЦП.* (Power Noise Reduction).

4. *Экономичный режим.* (Power Save). Выход из режима производится от сторожевого таймера, внешним прерыванием, прерыванием от таймера.

2.20. Прерывания

Прерывания служат для выполнения задачи, определяемой внутренним или внешним событием микроконтроллера. При возникновении прерывания микроконтроллер сохраняет в стеке содержимое счетчика команд PC и загружает в него адрес вектора прерывания. По этому адресу находится команда безусловного перехода к подпрограмме обработки прерывания.

Последней командой подпрограммы должна быть команда RETI, которая обеспечивает возврат в основную программу, восстановление предварительно сохраненного счетчика команд и установление разряда I регистра SREG в логическую 1.

Младшие адреса памяти программ отведены под таблицу векторов прерывания. Каждому прерыванию соответствует адрес в этой таблице, который загружается в счетчик команд при возникновении прерывания.

Таблица векторов прерываний приведена в таблице 13.

Табл. 13

Адрес	Источник прерывания	Описание прерывания
$0000	rjmp RESET	Включение питания, вывод RESET, сторожевой таймер
$0001	Rjmp EXT_INT0	Внешнее прерывание 0
$0002	Rjmp EXT_INT1	Внешнее прерывание 1
$0003	Rjmp TIM1_CAPT	Захват таймера T1
$0004	Rjmp TIM1_COMPA	Совпадение A таймера T1
$0005	Rjmp TIM1_OVF	Переполнение таймера T1
$0006	Rjmp TIM0_OVF	Переполнение таймера

		T0
$0007	Rjmp USART0_RXC	Прием завершен
$0008	Rjmp USART0_DRE	Регистр данных пуст
$0009	Rjmp USART0_TXC	Передача завершена
$000A	Rjmp ANA_COMP	Прерывание от аналогового компаратора
$000B	Rjmp PCINT	Прерывание по изменению состояния вывода
$000C	Rjmp TIMER1_COMPB	Совпадение B таймера T1
$000D	Rjmp TIMER0_COMPA	Совпадение A таймера T0
$000E	Rjmp TIMER)_COMPB	Совпадение B таймера T0
$000F	Rjmp USI_START	Старт USI
$0010	Rjmp USI_OVER-LOW	Переполнение USI
$0011	Rjmp EE_READY	EEPROM, готово
$0012	Rjmp WDT_OVERLOW	Переполнение сторожевого таймера

Если прерывания в работе микроконтроллера не предусматриваются, то на месте таблицы векторов прерываний может быть размещена часть основной программы.

Пример записи прерываний.

```
.CSEG        ; запись в память программ
.Org   0
Rjmp   Reset
Rjmp   EXT_INT0  ; команда перехода к подпрограмме обработки прерывания по INT0
Rjmp   EXT_INT1  ;команда перехода к подпрограмме обработки прерывания по INT1
Reset:
Программа
EXT_INT0:
.
.
EXT_INT1:
.
.
```

2.21. Обработка внешних прерываний

В микроконтроллере ATtiny 2313 имеется два внешних прерывания: INT0 (порт PD2) и INT1 (порт PD3). Порты должны быть сконфигурированы как входы (с подтяжкой или без нее).

За обработку внешних прерываний отвечает регистр **GIMSK** (**G**eneral **I**nterrupt **M**a**sk** Register). Он разрешает или запрещает внешние прерывания INT0 и INT1.

О состоянии прерывания сигнализирует его флаг. Флаги для внешних прерываний находятся в регистре флагов **EIFR** (**E**xternal **I**nterrupt **F**lag **R**egister).

Если возникает прерывание, то в регистре флагов прерываний устанавливается в 1 соответствующий флаг.

Прерывания могут стать активными только тогда, когда в регистре состояния SREG установлен в 1 разряд общего разрешения прерываний **I** (7-й разряд). Теперь, если наступает прерывание, то выполнение программы ответвляется по соответствующему адресу, а разряд общего разрешения прерываний **I** сбрасывается в 0, чем дальнейшие прерывания блокируются. Если требуется прервать подпрограмму другим прерыванием, то после входа в подпрограмму обработки прерывания надо установить программно флаг **I** в логическую 1 (команда **SEI**).

Вместе с входом в подпрограмму обработки прерывания аппаратно сбрасывается также и соответствующий флаг, вызвавший прерывание.

2.21.1. Регистр GIMSK

Разрешения на внешние прерывания вырабатываются через регистр GIMSK.

Регистр GIMSK

7	6	5	4	3	2	1	0
INT1	INT0	PCIE	-	-	-	-	-

Если разряд INT1 установлен в логическую 1, то внешнее прерывание 1 будет разрешено, пока будет установлен разряд **I** в регистре SREG. Будет ли прерывание вызываться по фронту сигнала или по уровню логического 0 определяют разряды ISC11, ISC10 в регистре MCUCR. Действие этих разрядов приведено в таблице 14.

Табл. 14

ISC11	ISC10	Пояснение
0	0	Прерывание INT1 по уровню логического 0
0	1	-
1	0	Прерывание INT1 по нарастающему фронту
1	1	Прерывание INT1 по спадающему фронту

Если разряд INT0 установлен в логическую 1, то будет разрешено внешнее прерывание 0 при условии, что разряд **I** в регистре состояния равен 1. Разряды ISC01, ISC00 в регистре MCUCR определяют условие вызова прерывания (таблица 15).

Табл. 15

ISC01	ISC00	Пояснение
0	0	Прерывание INT0 по уровню логического 0
0	1	-
1	0	Прерывание INT0 по спадающему фронту
1	1	Прерывание INT0 по нарастающему фронту

Если в разряде PCIE записана логическая 1 и флаг **I** регистра состояния так же установлен в 1, то разрешается внешнее прерывание по изменению состояния выводов PCINT7…0 микроконтроллера (выводы PB0 – PB7). К возникновению прерывания приводит любое изменение сигнала на любом выводе.

2.21.2. Регистр EIFR (External Interrupt Flag Register)

Состояние внешнего прерывания определяются регистром EIFR – общий регистр флагов прерывания.

Регистр EIFR

7	6	5	4	3	2	1	0
INTF1	INTF0	PCIF	-	-	-	-	-

INTF1 – флаг внешнего прерывания INT1. Если в результате события на выводе INT1 сформировался запрос на внешнее прерывание, то этот разряд устанавливается в 1. Флаг сбрасывается аппаратно при запуске подпрограммы обработки прерывания.

INTF0 – то же, но по INT0.

PCIF – если в результате события на любом из выводов PCINT7…0 сформировался запрос на прерывание, этот разряд устанавливается в 1. Флаг сбрасывается аппаратно при запуске подпрограммы обработки прерывания.

2.21.3. Возврат из прерывания

Возврат из прерывания осуществляется по команде RETI. При этом флаг I аппаратно устанавливается в 1.

ЦПУ автоматически восстанавливает содержимое счетчика команд, и основная программа продолжает свое выполнение с того места, где она была прервана.

2.22. Дребезг контактов

Если в устройстве используется механический переключатель, то переход из состояния «включено» в состояние «выключено» и наоборот сопровождается явлением, получившим название «дребезг». При этом контакты вибрируют, многократно касаясь друг друга в течение единиц миллисекунд (рис. 38).

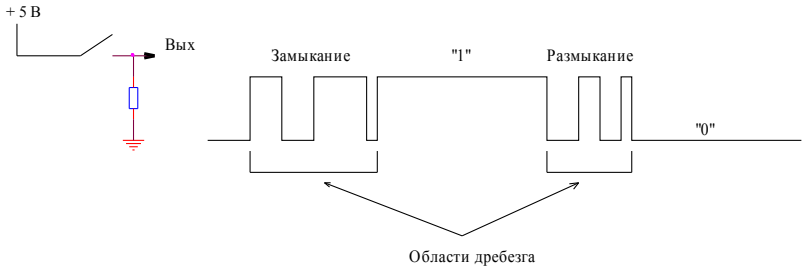


Рис. 38

Микроконтроллер может ошибочно принимать многократные замыкания контактов как ввод новых данных и последовательность сигналов будет записана неправильно.

Для борьбы с дребезгом контактов разработано много способов. Один из них – применение RS-триггеров (рис. 39).

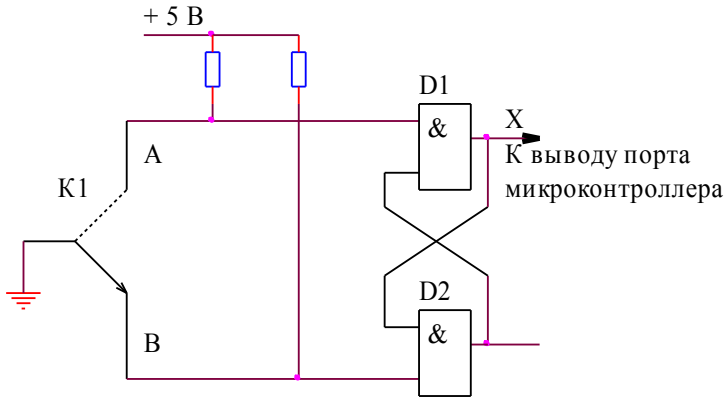


Рис. 39

Ключ К1 подключает цепи А или В к земляной шине, перебрасывая тем самым триггер, выполненный на микросхемах D1 и D2, из одного устойчивого состояния в другое. Временные диаграммы, поясняющие работу схемы, показаны на рис 40.

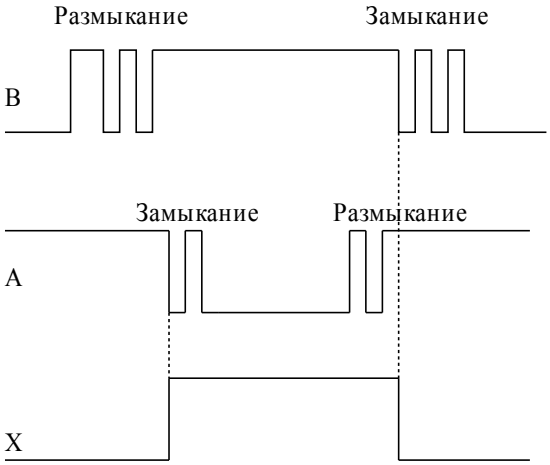


Рис. 40

2.22.1. Программное устранение дребезга

Устранить дребезг контактов можно программно. Возьмем переключающий контакт (рис. 41).

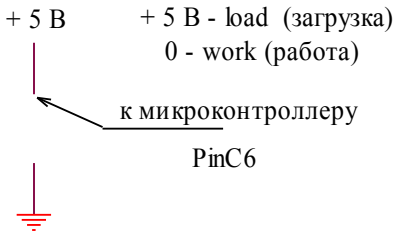


Рис. 41

Пусть необходимо подавать на вход микроконтроллера PinC6 с помощью переключателя 0 или 1. Проверку состояния переключателя произведем по алгоритму, приведенному на рис. 42.

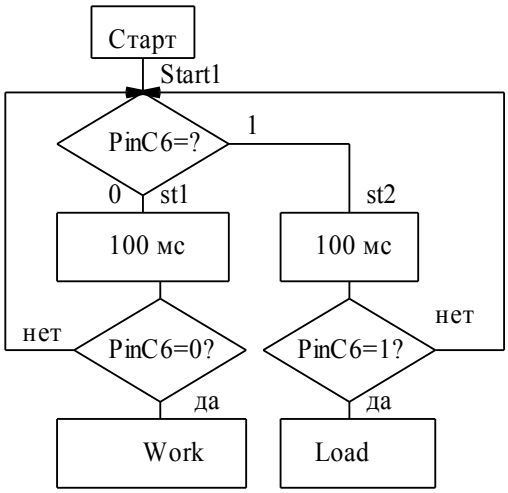


Рис. 42

Сущность алгоритма заключается в следующем. Проверяем состояние на входе PinC6 микроконтроллера. Команда выполняется за время такта. Если, например, получен 0, то делается повторная проверка контакта PinC6, но с задержкой 100 мс, когда дребезг контакта прекратится. И если состояние 0 подтверждается, то переходим к дальнейшему выполнению программы (Work). Если же не подтверждается, то повторно проверяется контакт.

Программа запишется следующим образом.

```
In   tmp,PinC
Andi   tmp,0b01000000 ; маскируем 6-й разряд
Subi   tmp,0b01000000 ; вычитаем единицу из шестого разряда
Brim   st1; если было 0, то переход на st1
Rjmp   st2 ; если 1 – то на st2
St1:
Rcall   tim01s
Sbic   PinC,6  ; если 6-й разряд 0 то переход в режим Work
Rjmp   Start ; иначе на Start
Rjmp   Work
St2:
Rcall   tim01s
Sbis   PinC,6   ; Если 6-й разряд 1, то переход в режим Load
Rjmp   Start   ; если иначе, то на Start
Rjmp   Load
```

2.23. Обработка прерываний от таймеров

За обработку прерываний от таймеров (восьмиразрядный таймер T/C0 и шестнадцатиразрядный – T/C1) отвечает регистр TIMSK (**T**imer/Counter **I**nterrupt **M**a**sk** Register) – регистр маски прерываний от таймеров/счетчиков. Он управляет прерываниями во взаимосвязи с таймерами T/C0 и T/C1.

Формат регистра TIMSK имеет вид:

Регистр TIMSK

7	6	5	4	3	2	1	0
TOIE1	OCIE1A	OCIE1B	-	ICIE1	OCIE0B	TOIE0	OCIE0A

1 и 7 р. TOIE0 или 1 - разрешение прерывания по переполнению таймера Т0 или Т1;

5 и 6 р. OCIE1A или В – разрешение прерывания по событию «Совпадение А или И» таймера Т1;

3 р. ICIE1 – флаг разрешения прерывания по событию «Захват» таймера Т1;

0 и 2 р. OCIE0A и В – разрешение прерывания по событию «совпадение А или В» таймера Т0.

Для разрешения какого-либо прерывания от таймера надо установить в «1» соответствующий разряд регистра TIMSK и флаг I регистра SREG.

Для индикации наступления прерываний от таймеров Т0, Т1 предназначен регистр TIFR (Timer/Counter Interrupt Flag Register- регистр флагов прерываний от таймеров).

Регистр TIFR

7	6	5	4	3	2	1	0
TOV1	OCF1A	OCF1B	-	ICF1	OCF0B	TOV0	OCF0A

7-й и 1-й разряды: TOV1, TOV0 – флаги прерывания по переполнению таймеров Т1 и Т0;

6-й и 5-й разряды: OCV1A, OCF1B – флаги прерывания по событию «Совпадение А» или «Совпадение В» таймера Т1;

3-й разряд: ICF1 – флаг прерывания по событию «Захват» таймера Т1;

2-й и 0-й разряды: OCF0B, OCF0A – флаги прерывания по событию «Совпадение А» или «Совпадение В» таймера Т0.

При наступлении какого-либо события соответствующий флаг регистра TIFR устанавливается в 1. При запуске подпрограммы обработки прерывания он аппаратно сбрасывается в 0.

2.24. Восьмиразрядный таймер/счетчик Т/С0

Таймер использует три вывода микроконтроллера. Эти выводы – линии портов ввода/вывода общего назначения. Функции, используемые этими выводам при работе совместно с таймерами/счетчиками, являются их альтернативными функциями. Эти выводы надо самостоятельно сконфигурировать в соответствии с их функциональным назначением. Назначение портов приведено в таблице 16.

Табл. 16

Выводы таймера	Линии портов ввода/вывода	Описание
T0	PD4	Вход внешнего сигнала таймера T0
OC0B	PD5	Выход схемы сравнения A таймера T0
OC0A	PB2	Выход схемы сравнения B таймера T0

Восьмиразрядный таймер T0 может получать импульсы тактовой частоты СК, импульсы с предварительного делителя, импульсы с внешнего вывода или может быть остановлен соответствующими установками регистра TCCR0A(B). Флаг переполнения таймера находится в регистре TIFR. Биты управления таймером расположены в регистре TCCR0A(B). Разрешение и запрещение прерываний от таймера управляется регистром TIMSK. Структурная схема таймера/счетчика T/C0 приведена на рис. 43.

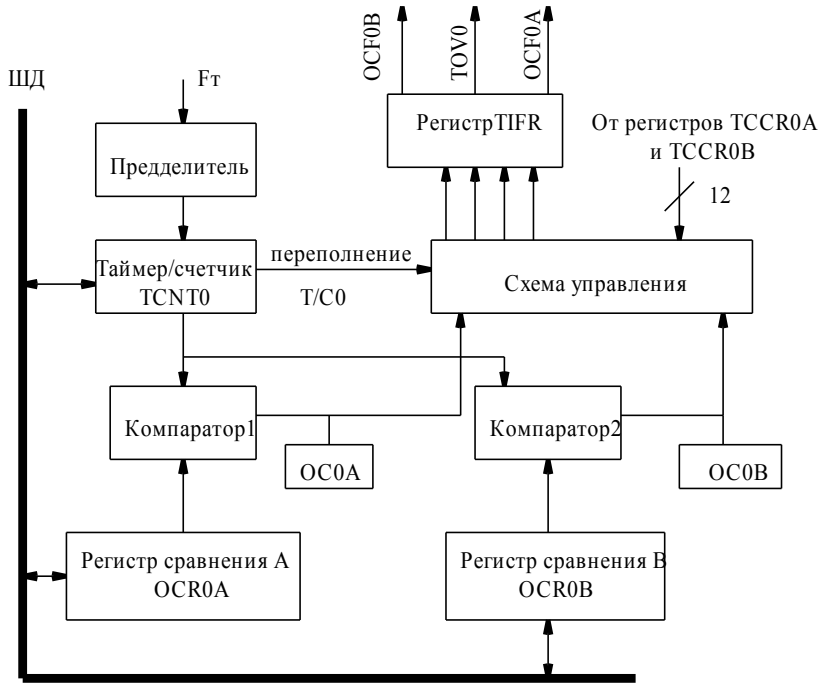


Рис. 43

Таймер T0 генерирует прерывание при переполнении счетного регистра TCNT0, а так же при равенстве счетного регистра и регистра сравнения. Разрешение прерываний находятся в регистре TIMSK, а флаги прерываний – в регистре TIFR.

Регистры сравнения OCR0(A,B) входят в состав блока сравнения. Во время работы таймера T/C0 производится непрерывное сравнение этого регистра с регистром TCNT0. В случае равенства содержимого этих регистров в следующем машинном цикле устанавливается флаг OCF0 регистра TIFR и генерируется прерывание (если разряд I регистра SREG равен 1). Кроме того, при наступлении этого события может изменяться состояние выводов OC0A и OC0B. Для этого эти выводы должны быть сконфигурированы *как выходы.*

Регистры TCCR0A и TCCR0B предназначены для управления модулем таймера/счетчика.

Регистр TTCR0A

7	6	5	4	3	2	1	0
COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00

Регистр TTCR0B

7	6	5	4	3	2	1	0
FOC0A	FOC0B	WGM02	-	-	CS02	CS01	CS00

Режим работы таймера T/C0 определяется состоянием разрядов WGM00 и WGM01 регистра TCCR0A (таблица 17).

Табл. 17

Номер режима	WGM01	WGM00	Режим работы
0	0	0	Normal
1	0	1	Phase Correct PWM - ШИМ с точной фазой
2	1	0	CTC (Clear Timer on Compare) - сброс при совпадении
3	1	1	Fast PWM - быстродействующий ШИМ

2.24.1. Номер режима 0 – режим Normal

В режиме Normal производятся следующие действия:

- Регистр Normal считает от 0 до 255 и затем снова с нуля. При этом устанавливается флаг переполнения TOV0.

- При равенстве счетного регистра и регистра сравнения устанавливается флаг OCF0A (или B) и если разряд OCIE0A (или B) регистра TIMSK установлен в 1, генерируется прерывание.
- Может изменяться состояние вывода OC0A (или B) микроконтроллера. Как оно будет изменяться, определяется разрядами COM0A1(A0) и COM0B1 (B0) (таблица 18).

Табл. 18

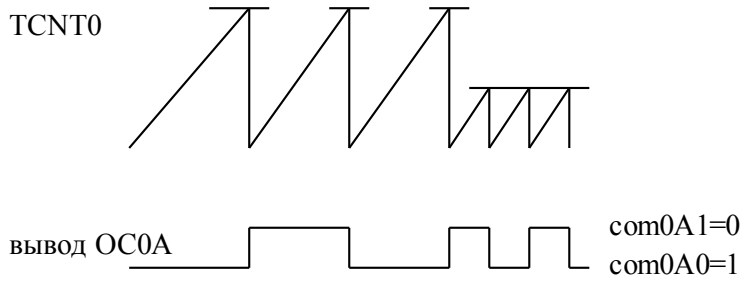
COM0N1	COM0N0	Описание
0	0	Таймер T/C0 отключен от вывода OC0A(B)
0	1	Состояние вывода OC0A(B) меняется на противоположное
1	0	Вывод OC0A сбрасывается в 0
1	1	Вывод OC0A устанавливается в 1

N=A или B

При необходимостисостояние вывода OC0A(B) может быть изменено принудительно записью логической 1 в разряд FOC0n регистра TCCR0B (в соответствии с разрядами COM0N1 и COM0N0). Прерывание при этом не генерируется. (FOC – Force Output Compare).

2.24.2. Номер режима 2 – режим CTC (сброс при совпадении)

В этом режиме максимально возможное значение счетного регистра TCNT0 определяется регистром сравнения OCR0A. Счет начинается со значения $00. Когда обнуляется счетчик, устанавливается флаг прерывания TOV0 и флаг OCF0 регистра TIFR. С установкой флага может изменяться состояние вывода OC0A(B) микроконтроллера. Поведение вывода определяется состояниями COM0N1 и COM0N0.



Частота генерируемого сигнала определяется по формуле:

$$f_{OC0A} = \frac{f_{clk}}{2N(1+OCR0A)},$$

где N – коэффициент деления предделителя (1 – 1024).

При необходимости состояние вывода OC0A может быть изменено принудительно, записью логической 1 в разряд FOC0A регистра TCCR0B. Прерывание при этом не генерируется и сброс счетного регистра не производится.

2.24.3. Пример формирования частоты звучания ноты

В качестве примера выберем формирование частоты ноты в музыкальном автомате. В нем имеются константы частоты нот и время звучания ноты (константы длительности). Кроме того, в подпрограммах формируются длительности задержки 20 мс и 120 мс.

Фрагмент программы формирования звучания ноты приведен ниже:

```
;загрузка константы частоты в регистр сравнения
Out   OCR1AH,XH
Out   OCR1AL,XL
;формирование длительности ноты и воспроизведение ее
Ldi   tmp,0b01000000   ;включение режима переключения вывода OC1A
Time1:
Rcall   Time_120ms   ;вызов подпрограммы задержки 120 мс
Dec     Ntim              ; в NTIM загружена константа длительности ноты
Brne    Time1   ; когда NTIM станет равным 0, то выполняем программу дальше
Ldi     tmp,0b10000000   ; по этому коду OC1A переключается в 0, то есть пре-
                         кращается звучание

; подпрограмма формирования задержки 20 мс

Time_20ms:
Ldi     YL,low(3000)
 Ldi     YH,(3000)
Ms_20
Sbiw     YL,1   ; вычитание из YL единицы
Brne      ms_20  ; когда результат станет равным нулю, возвращаемся из под-
                   программы
Ret

; подпрограмма задержки 120 мс

Time_120ms:
Rcall     Time_20ms
Rcall     Time_20ms
Rcall     Time_20ms
Rcall     Time_20ms
```

```
Rcall    Time_20ms
Rcall    Time_20ms
Ret
```

2.24.4. Номер режима 3 – быстродействующий ШИМ (Fast PWM)

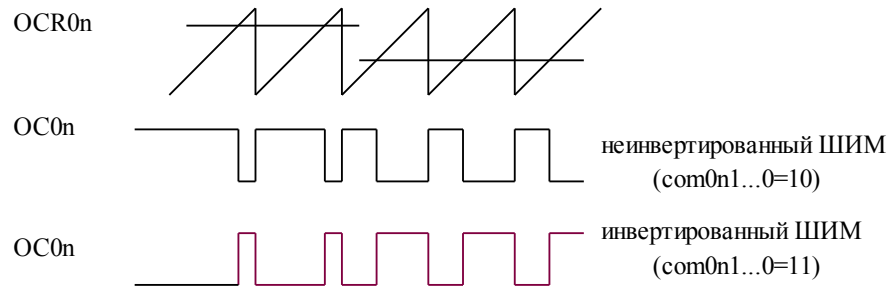
Этот режим позволяет генерировать высокочастотный сигнал с широтно-импульсной модуляцией. В связи с высокой частотой генерируемого сигнала данный режим может использоваться при регулировании мощности, выпрямлении, цифроаналоговом преобразовании и др.

Счетный регистр функционирует как суммирующий счетчик по каждому импульсу f_t. Состояние счетчика меняется от $00 до $FF, после чего счетный регистр сбрасывается и цикл повторяется. При достижении счетчиком максимального значения устанавливается флаг TOV0 регистра TIFR (и возникает прерывание по переполнению). При равенстве содержимого счетного регистра и регистра сравнения OCR0 устанавливается флаг OCF0 регистра TIFR (и возникает прерывание по совпадению)

Поведение вывода OC0 микроконтроллера в этом режиме определяется разрядами com0n1 и com0n0 (таблица 19).

Табл. 19

com0n1	com0n0	Поведение вывода OCn
0	0	Таймер отключен от вывода OCn
0	1	-
1	0	Сбрасывается в 0 при равенстве регистра TCNT0 и OCR0n. Устанавливается в 1 при TCNT0=$00 (неинвертированный ШИМ)
1	1	Устанавливается в 1 при равенстве TCNT0 и OCR0n. Сбрасывается в 0 при TCNT0=$00. (инвертированный ШИМ).



Частота генерируемого сигнала равна

$$f_{OCon} = \frac{f_{clk}}{256N},$$

где N – коэффициент деления предделителя.

2.24.4. Номер режима 4 – ШИМ с точной фазой (Phase Correct PWM)

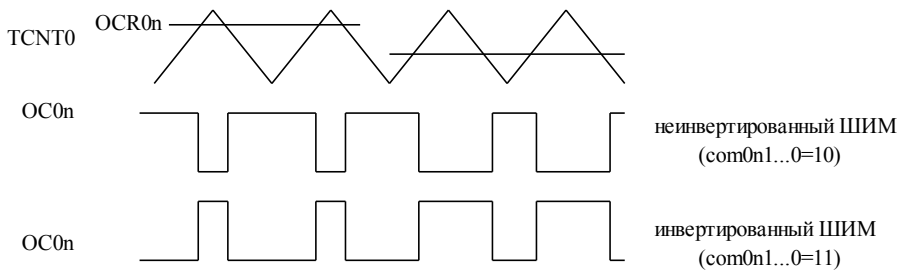
Предназначен, как и режим 2, для генерации сигналов с ШИМ. Однако в этом режиме счетный регистр функционирует как реверсивный счетчик, изменение состояния которого происходит по каждому импульсу тактового сигнала. Состояние счетчика сначала изменяется от $00 до $FF, а затем обратно до $00. Максимальная частота сигнала в этом режиме в два раза меньше максимальной частоты сигнала в режиме Fast PWM.

При достижении счетчиком максимального значения ($FF) происходит смена направления счета. При достижении минимального значения ($00) так же происходит смена направления счета и устанавливается флаг прерывания TOV0 регистра TIFR.

При равенстве содержимого счетного регистра и регистра сравнения OCR0 устанавливается флаг OCF0 регистра TIFR и изменяется состояние вывода OC0 в соответствии с разрядами com01 и com00 (таблица 20).

Табл. 20

com0n1	com0n0	Поведение вывода OC0n
0	0	Таймер TCNT0 отключен от вывода OC0n
0	1	-
1	0	Сбрасывается в 0 при прямом счете и устанавливается в 1 при обратном счете (неинвертированный ШИМ)
1	1	Устанавливается в 1 при прямом счете и сбрасывается в 0 при обратном счете (инвертированный ШИМ).



Частота генерируемого сигнала равна

$$f_{OC0n} = \frac{f_{clk}}{512N},$$

где N – коэффициент деления предделителя.

2.24.5. Пример

При несанкционированном открывании двери контакт размыкается и включается сирена с частотой 500 Гц.

Схема устройства приведена на рис. 44.

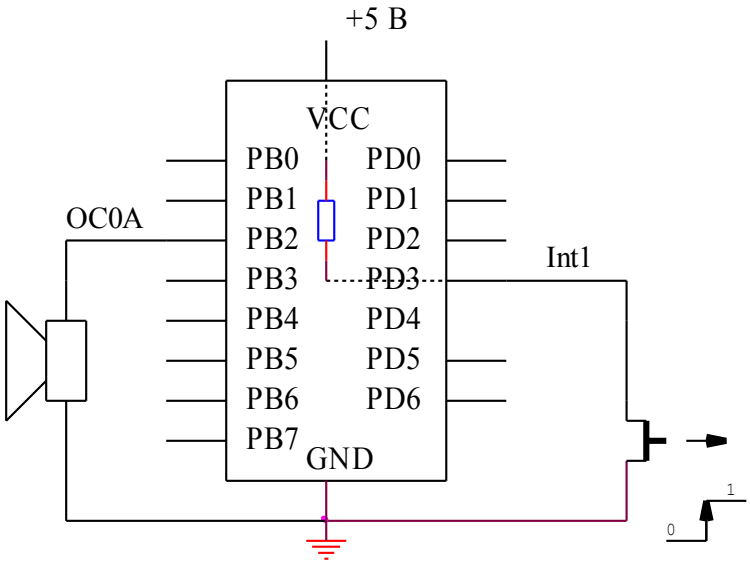


Рис.44

Требования к портам PB2 и PD3 следующие:

PD3:

- Вход с подтяжкой
- Прерывание по нарастающему фронту (в регистр MCUCR записывается число 0b00001000; 3 разряд ICS11=1, 2 разряд ICS10=0)
- Разрешить прерывание по INT1 (в регистр GIMSK записывается число 0b10000000; 7 разряд Int1=1)

- Разрешить общее прерывание (команда sei)

PB2:

- Выход
- OC0A установить в 0
- Запустить таймер TCNT0 (в регистр TCCR0B записывается число 0b00000101 (CS02…CS000=101) (деление на 1024. f=4 кГц)
- Установить режим CTC (сброс при совпадении) (в регистр TCCR0A записывается число 0b00000010; 1 разряд wgm01=1, 0 разряд wgm00=0)
- Состояние вывода OC0A меняется на противоположное (в регистр TCCR0A записывается число 0b01000000; 7-й разряд com0A1=0, 6-й разряд com0a0=1). В результате в регистр OC0A записывается, с учетом предыдущей строчки, число 0b01000010
- Установить в регистре OCR0A число 16 (4 кГц:8=500 Гц).

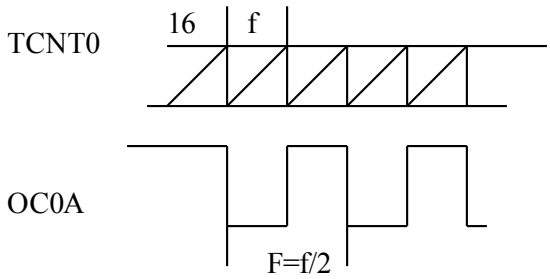
С учетом изложенного программа будет иметь вид:

```
.device   ATtiny2313
.nolist
.include   "…\tn2313def.inc"
.list
.def   tmp=r16
.CSEG
.org  0
rjmp   reset
nop    ; Int0
rjmp   Ext_Int1

reset:
ldi   tmp,low(RAMEND)   ; загрузка стека
out   SPL,tmp
clr   tmp                ; D-входы
out   ddrD,tmp
ser    tmp                    ; B – выходы
out   ddrB,tmp
out   PortD,tmp                ; подтяжка на входы порта D
clr   tmp                ; на всех выходах порта B – 0
out   PortB,TMP
ldi   tmp,0b10000000 ; разрешить прерывания по Int1
out   GIMSK,tmp
ldi   tmp,0b00000101 ; конфигурирование таймера
out   TCCR0B,tmp            ; f=Ft/1024; 4 кГц
ldi   tmp,0b01101000 ; прерывание по нарастающему фронту (3 и 2 р = 1…0); установка спящего режима Power Down (6 и 4 р. = 1…0); разрешение спящего режима (5р=1)
```

```
out   MCUCR,tmp
sleep                    ; спящий режим

; обработка прерывания
Ext_Int1:
ldi   tmp,16             ; установка в регистр OCR0A числа 16
out   OCR0A,tmp
ldi   tmp,0b01000010 ; сброс при совпадении и OC0A меняется на противоположное
out   TCCR0A,tmp
```



2.25. Шестнадцатиразрядный таймер/счетчик T/C1

Организован как два 8-ми разрядных регистра. В него добавлен режим захвата (вход ICR). Назначение его – сохранение в определенный момент времени состояния таймера в регистре захвата (ICR1H, ICR1L). Это действие производится по фронту сигнала на выводе ICP (PD6), либо по сигналу от аналогового компаратора. Одновременно устанавливается флаг регистра TIFR и генерируется запрос на прерывание.

16-разрядный регистр физически размещается в двух 8-разрядных регистрах. Соответственно, для обращения к ним надо выполнить по две операции чтения или записи.

Для того, чтобы запись/чтение обоих байт содержимого 16-разрядного регистра происходило одновременно, в составе таймера имеется специальный 8-разрядный регистр TEMP, предназначенный для хранения старшего байта значения. Этот регистр используется процессором и программно недоступен.

Для выполнения цикла записи 16-разрядного регистра первым загружается старший байт, который помещается в регистр TEMP. При последующей записи младшего байта он объединяется с содержимым регистра TEMP и оба байта одновременно (в одном машинном цикле) записываются в 16-разрядный регистр.

Регистр управления модулем T/C1 TCCR1A полностью аналогичен регистру TCCR0A таймера T/C0.

Регистр TTCR1A

7	6	5	4	3	2	1	0
COM1A1	COM1A0	COM1B1	COM1B0	-	-	WGM11	WGM10

Установка режима выводов OC1A(B) с помощью битов com1A1…0 и com1B1…0 совпадает с T/C0.

Регистр управления TCCR1B немного отличается от аналогичного в T/C0 и имеет вид:

Регистр TTCR1B

7	6	5	4	3	2	1	0
ICNS1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10

Биты WGM13, WGM12 совместно с WGM11 и WGM10 определяют режим работы таймера (таблица 21).

Табл. 21

WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Режим (операция)
0	0	0	0	Normal
0	0	0	1	PWM, Phase Correct, 8-bit
0	0	1	0	PWM, Phase Correct, 9-bit
0	0	1	1	PWM, Phase Correct, 10-bit
0	1	0	0	CTC
0	1	0	1	Fast PWM, 8-bit
0	1	1	0	Fast PWM, 9-bit
0	1	1	1	Fast PWM, 10-bit
1	0	0	0	PWM, Phase and Frequency Correct
1	0	0	1	PWM, Phase and Frequency Correct
1	0	1	0	PWM, Phase Correct
1	0	1	1	PWM, Phase Correct
1	1	0	0	CTC
1	1	0	1	-
1	1	1	0	Fast PWM
1	1	1	1	Fast PWM

Биты CS12, CS11, CS10 определяют частоту таймера

Бит ICNC1 – управляют схемой подавления помех блока захвата. Если 0 – схема подавления выключена, если 1 – схема включена и захват происходит в случае четырех одинаковых выборок.

Бит ICES1 – выбор активного фронта сигнала захвата. Если 0 – то сохранение счетного регистра в регистре захвата по спадающему фронту, если 1 – по нарастающему.

Регистр TCCR1C имеет вид:

Регистр TTCR1C

7	6	5	4	3	2	1	0
FOC1A	FOC1B	-	-	-	-	-	-

FOC1A(B) – принудительное изменение состояния вывода OC1A(B). При записи логической 1 состояние вывода OC1 изменяется в соответствии с установками разрядов COM1A1…0 и COM1B1…0. Прерывание при этом не генерируется и сброс таймера (в режиме CTC) не производится. Функция доступна, когда не используется ШИМ.

В состав T/C1 входят:

16-ти разрядный таймер/счетчик – TCNT1H, TCNT1L;

16-ти разрядные регистры сравнения A и B – OCR1A(B)H, OCR1A(B)L.

Регистры управления прерываниями от таймеров T/C1 TIMSK и TIFR рассмотрены ранее.

2.26. Память данных EEPROM

Это энергонезависимая память (128 байт). В отличие от ОЗУ EEPROM сохраняет свое содержимое при отсутствии питания. Содержимое EEPROM можно считывать и изменять во время работы программы. Но можно заносить данные при программировании. Для этого используется директива **.ESEG**. Она определяет начало EEPROM-сегмента. Для обращения к EEPROM используются 3 регистра ввода/вывода: регистр адреса EEAR, регистр данных EEDR и регистр управления EECR.

2.26.1. *Регистр адреса EEAR*

Имеет длину 1 байт. Байты доступны для чтения и записи.

Регистр EEAR

7	6	5	4	3	2	1	0
EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0

Для программирования или чтения байта данных памяти EEPROM в регистр адреса EEAR должен быть записан соответствующий адрес.

2.26.2. *Регистр данных EEDR*

При записи в этот регистр загружаются данные, которые должны быть помещены в EEPROM. При чтении в этот регистр помещаются данные, считанные из EEPROM.

2.26.3. *Регистр управления EECR*

Используется для управления доступом к EEPROM-памяти. Формат регистра:

Регистр EECR

7	6	5	4	3	2	1	0
-	-	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE

EEPM1, EEPM0 – биты режима программирования EEPROM.

EEPM1	EEPM0	Время программирования, мс	Операция
0	0	3,4	Стирание и запись в одной операции
0	1	1,8	Только стирание
1	0	1,8	Только запись
1	1	-	резерв

EERIE – разрешение прерывания от EEPROM. Этот разряд управляет генерацией прерывания, возникающего при завершении цикла записи в EEPROM/ Если 1, то прерывания разрешены.

EEMPE – управление разрешением записи в EEPROM. Сбрасывается аппаратно через 4 машинных цикла.

EEPE – разрешение записи в EEPROM. При установке разряда в 1 происходит запись данных в EEPROM. (если EEMPE=1).

EERE – разрешение чтения из EEPROM. При установке разряда в 1 выполняетсся чтение данных из EEPROM. По окончании чтения разряд сбрасывается аппаратно.

Запись одного байта в EEPROM-память состоит из этапов:

1. Дождаться готовности памяти к записи данных (флаг EEPE=0).
2. Загрузить байт данных в регистр EEDR, а требуемый адрес – в регистр EEAR.
3. Установить в 1 флаг EEMPE.
4. В течении 4-х машинных циклов после установки флага EEMPE записать 1 в разряд EEPE.

Длительность записи составляет 3-7 мс. По окончании записи разряд EEPE аппаратно сбрасывается, после чего программа может начать запись следующего байта.

При выполнении этой последовательности команд рекомендуется запретить все прерывания (сброс бита **I** регистра SREG).

Фрагмент программы, осуществляющей запись в EEPROM:

```
EEWrite:
   Sbic  EECR,EEPE ; Sbic-пропуск следующей команды, если разряд EEPE регистра EECR очищен.
   Rjmp  EEWrite   ; ждать, пока флаг EEPE не будет сброшен
Cli                ; сброс флага общего разрешения прерываний
Out    EEAR,AddrReg  ; загрузка адреса
Out    EEDR,DataReg  ; загрузка данных
Sbi    EECR,EEMPE    ; установка разряда EEMPE в регистре EECR
Sbi    EECR,EEPE     ; установка разряда EEPE в регистре EECR (выдать
       строб записи в EEPROM)
Sei    ; разрешить прерывания.
```

Чтение EEPROM

После загрузки требуемого адреса в регистр EEAR, программа должна установить разряд EERE регистра EECR в 1. Когда запрошенные данные будут находиться в регистре данных EEDR, произойдет аппаратный сброс этого разряда.

Фрагмент программы, осуществляющий чтение EEPROM.

```
EERead:
Sbic      EECR,EEPE   ; ждать окончания текущей записи, пока EEPE не ста
    нет равным 0
Out       EEAR,AddrREg  ; загрузка адреса
Sbi       EECR,EERE     ; выдать строб чтения из EEPROM
 Sbi          EECR,EERE     ; выдать строб чтения из EEPROM
In        DataReg,EEdr       ; прочитанный байт отправить в РОН.
```

Меры предосторожности

У памяти EEPROM есть один недостаток: при снижении питания хранящиеся в памяти данные могут быть повреждены. Чтобы избежать повреждения данных можно воспользоваться одним из решений:

1. Удерживать микроконтроллер в состоянии сброса все время, пока напряжения питания ниже нормы. Для этого использовать конфигурационные ячейки BOD.

2. Удерживать микроконтроллер в спящем режиме (Power Down), пока напряжение питания находится ниже нормы. В этом режиме микроконтроллер не может выполнять никаких команд и поэтому защищен от непреднамеренной записи в EEPROM.

3. Хранить константы во Flach памяти, если они не должны меняться во время работы программы.

2.27. Микропроцессорный тренажер на микроконтроллере

Тренажер показан на рис.

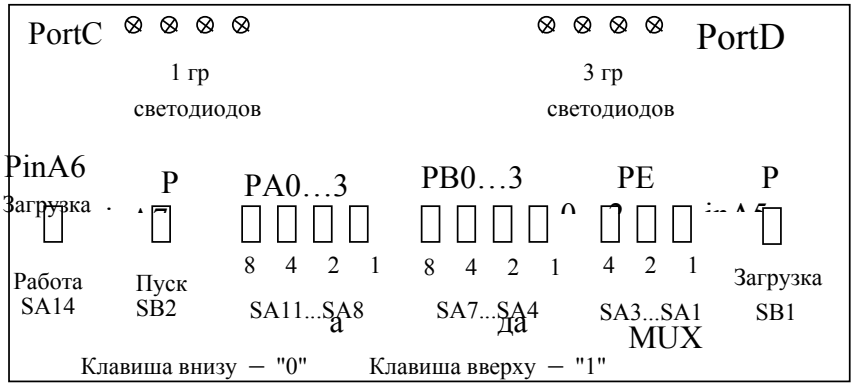


Рис. 45

В настоящее время программа имеет ряд недостатков. Для чтения данных необходимо писать команду ЧТЕНИЕ. Нельзя прочитать данные в регистре Q и содержимое регистра B. Для выхода из режима РАБОТА надо нажать клавишу ПУСК. Поэтому была поставлена задача: исключить нажатие клавиши ПУСК при переходе в режим ЗАГРУЗКА; исключить команду ЧТЕНИЕ; иметь возможность вывода данных из АЛУ, регистра Q , и по адресу регистра B. Для этого составлены алгоритмы и по ним написана программа. Программа выполнения микроопераций переходит на метку wor3. С нее и начнем описание алгоритмов и программы.

В этой части программы опрашиваем состояние клавиш a7 – ПУСК и a6 - переключатель ЗАГРУЗКА/РАБОТА. Они могут иметь три состояния:

- 00 (a7=0, a6=0), то есть переключатель a6 находится в положении РАБОТА, а клавиша a7 – нажата (ПУСК).
- 10 (a7=1, a6=0); a6 находится в положении РАБОТА, а a7 – отжата, то есть исходное состояние в режиме РАБОТА.
- 11 (a7=1 a6=1); в этом положении переключатель ЗАГРУЗКА/РАБОТА находится в положении ЗАГРУЗКА.

Мы должны дешифровать эти три состояния и перейти на выполнение соответствующей подпрограммы. Алгоритм дешифрации:

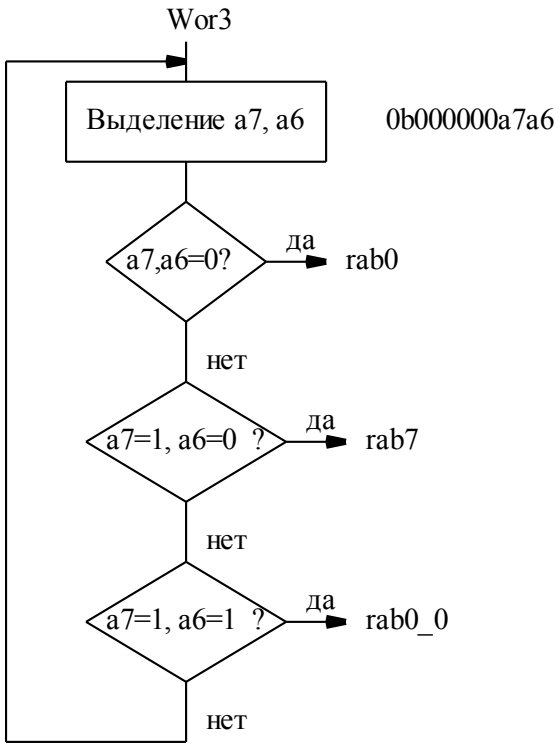


Рис. 46

Эта часть программы записывается в виде:

```
wor3:                   ; начало
in      tmp,PinA    ; считывание состояния контактов порта  A в рег. tmp
andi        tmp,0b11000000   ; маскирование разрядов a7,a6; 0ba7a6000000
swap        tmp         ; перестановка полубайтов a3 a2 a1 a0 a7 a6 a5 a4
lsr         tmp         ; сдвиг tmp вправо 0 a3 a2 a1 a0 a7 a6 a5
lsr         tmp         ; сдвиг tmp вправо 0 0 a3 a2 a1 a0 a7 a6
cpi         tmp,1       ; Из tmp вычитаем 1. Результат нигде не сохраняется. Но
                        ;если результат отрицательный, в регистре SREG
                        ;устанавливается в 1 флаг отрицательного результата
brmi        rab0        ; переход на rab0, если a7 a6 = 0 (00)
cpi         tmp,3
brmi        rab7        ;переход на rab7, если a7 a6 = 2 (10)
cpi         tmp,4
brmi        rab0_0      ; переход на rab0_0, если a7 a6 = 3 (11)
rjmp        wor3        ; возврат на начало wor3
```

Алгоритм и программа для rab0:

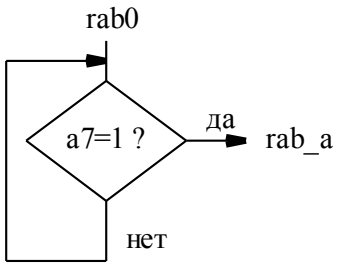


Рис. 47

```
Rab0:
In          tmp,PortA   ; считываем порт A
Andi        tmp,0b10000000 ; маскируем 7-й разряд
Swap        tmp          ;  перестановка полубайт
Lsr         tmp       ; сдвиг вправо регистра tmp
Lsr         tmp       ; сдвиг вправо регистра tmp
Lsr         tmp       ; сдвиг вправо регистра tmp остается число 0b0000000a7
Cpi         tmp,1     ; вычитаем из состояния a7 1-цу
Brmi        rab0      ; если a7=0, ПУСК замкнут, возврат на rab0/ Т.е. ждем, когда
                          отпустим кнопку ПУСК
rjmp        rab_a     ; если a7=1, ПУСК разомкнут и переход на rab_a
rab_a:
out         PortD,nol    ; обнуляем PortD (в начале программы записана строчка:
                           ; .def nol=r0, а в r0 заносится 0 в секции Инициализация)
rjmp        work1        ; переходим на начало режима РАБОТА, но с другим
                           ;адресом
```

Алгоритм и программа для rab7 :

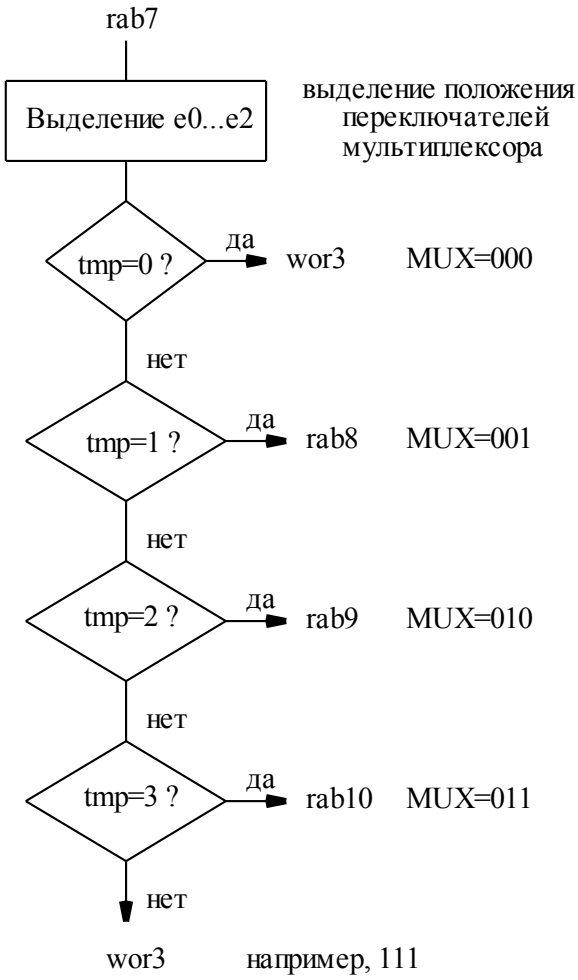


Рис. 48

Ниже приведена программа для указанного алгоритма.

```
rab7:
in      tmp,PinE    ; загрузка в tmp состояния контактов мультиплексора
cpi     tmp,1       ; порт E имеет только контакты e0, e1, e2, поэтому
                    маскировать не надо
brmi    wor3        ; то есть если mux=000, то переходим на wor3
cpi     tmp,2
brmi    rab8        ; если mux=001, то переход на rab8 и считываем АЛУ
cpi     tmp,3
brmi    rab9        ; если mux=010, то переход на rab9 и считываем регистр Q
cpi     tmp,4
brmi    rab10       ;  если mux=011, то переход на rab10 и считываем данные
                    ; по адресу регистра B
rjmp    wor3        ; возврат к wor3
```

rab8, rab9 и rab10 имеют вид:

<u>rab8:</u>

```
out     PortD,r8   ; выставляем результат АЛУ в порт D
rjmp    wor3

rab9:
out     PortD,r6   ; выставляем содержимое регистра Q в  порт D
rjmp    wor3

Rab10:
clr     ZH
ldi     ZL,0xE0 ; загрузка нач. адреса ОЗУ
add     ZL,r21   ; сложение нач. адреса ОЗУ с адресом рег. B
                 ; (в 1-й тетраде)
ld      tmp,Z    ; загрузка содержимого рег. B по адресу
                 ; Z в рег. Tmp
```

И наконец сочетание a7=1 и a6=1 приводит к переходу к rab0_0 и далее к началу всей программы. То есть начинается ЗАГРУЗКА.

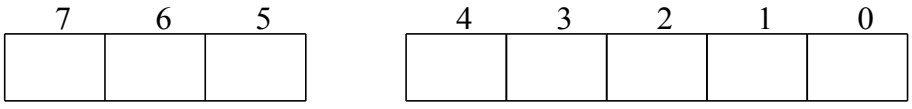
```
rab0_0:
out    PortD,nol   ;  загрузка в порт D нулевых значений
rjmp   START       ; переход на режим ЗАГРУЗКА и начало всей программы
```

2.27. Музыкальный автомат

- EEPROM
- Функция сравнения
- Генерация звука
- Косвенная адресация

Мелодия записывается в память программ. Константы, определяющие частоту ноты и ее длительность, записываются в EEPROM.

Формат кодирования ноты:



Адреса констант длительности нот

Адреса констант частоты нот

Таким образом, можно задать 2^3=8 значений длительности ноты и 2^5=32 значения частоты ноты. В одной октаве содержится 12 нот, то есть можем закодировать почти 3 октавы.

Характеристики длительности ноты приведены в таблице.

Адрес константы	Нота	Имя длительности	Продолжительность ноты, с	Константы длительности
000		t16	0,1	1
001		t8	0,2	2
010		t8h	0,3	3
011		t4	0,4	4
100		t4h	0,6	6
101		t2	0,8	8
110		t2h	1,2	12
111		t1	1,6	16

За первую ноту берем ноту соль первой октавы.

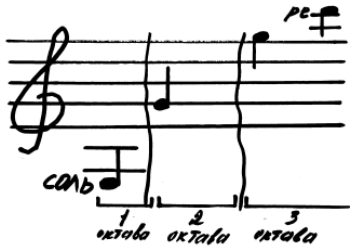


Рис. 45

Выбираем частоту кварцевого резонатора равной 4 Мгц. В октаве имеется 12 нот. Отношение частот в октаве равно 2. Частота ноты «ля» 2-й октавы равна 440 Гц. Поэтому нота ля 1-й октавы имеет частоту 220 Гц, а 3-й октавы-880 Гц.

Отношение соседних частот равно: $\sqrt[12]{2}$. Для формирования нот определенной частоты рассчитываем коэффициент деления частоты кварца. Для ноты «ля» второй октавы:

$$\frac{4000000\,Гц}{440\,Гц} = 9091.$$

С учетом того, что выбранный способ воспроизведения звукового сигнала уменьшает частоту в 2 раза, необходимо константу уменьшить в 2 раза и записать в память число 9091:2=4546. И так определяются константы для каждой ноты.

Записываем константы в EEPROM. Константы равны от 10204 (соль первой октавы) до 1700 (ре 3 октавы). Ноты записываются кодами от 0000 (10204) до 11111 (1700). [Код 11111 (pause) служит для перехода к подпрограмме Pause1].

Музыкальный автомат состоит из микроконтроллера, динамика, подключенного через усилитель модности к выводу OC1A. Схема автомата показана на рис. 46.

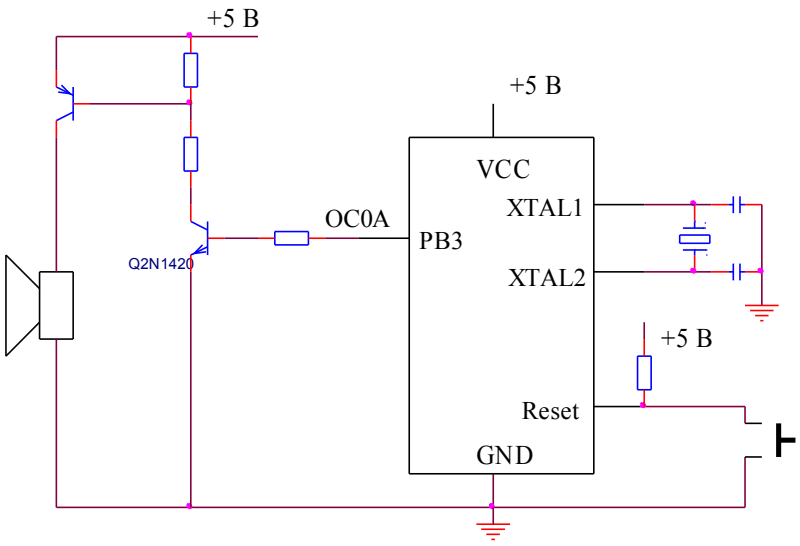


Рис. 46

1. Запись мелодии

Мелодия записывается в память программ. Код ноты занимает 1 байт. Поэтому используем директиву .db.

```
.CSEG
.org    128 ; адрес начала расположения кодов нот
.db     t4+la_1, t1+la_2, t4+sol_2, t4+pha_1, t4+sol_2, t2h+mi_1 и  так далее
```

2. Запись констант частоты и длительности нот

Константы частоты – двухбайтные числа. Для их записи используем директиву .dw. Константы длительности – однобайтные числа, поэтому применяем директиву .db.

```
.ESEG
.org    0
.dw     10204, 9629, 9091, 8584, 8097, 7648, 7220, 6802  и так далее

.org    0x40
.db     1, 2, 3, 4, 6, 8, 12, 16
```

3. Инициализация

- назначение имен длительностям и частотам;

```
.equ t16 =0b00000000
.equ t8  =0b00100000
.equ t8h =0b01000000
.equ t4  =0b01100000
.equ t4h =0b10000000
.equ t2  =0b10100000
.equ t2h =0b11000000
.equ t1  =0b11100000

.equ sol_1  =0b00000000
.equ soldi_1=0b00000001
.equ labe_1 =0b00000001
.equ la_1   =0b00000010
.equ ladi_1 =0b00000011
.equ sibe_1 =0b00000011
.equ si_1   =0b00000100
```

- назначение входов и выходов;
- включение режима CTC таймера T/C1 (сброс при совпадении с константой).
- установка частоты $f=f_t$
- установка прерываний по INT0.

4. Установка адреса первой ноты.

Используется команда LPM. Первая нота записана по адресу \$90 или 144_{10}.

```
Ldi    ZH,high(144<<1) ; адрес первой ноты сдвигаем влево на один разряд и
                         записываем в Z.
Ldi    ZL,low(144<<1)
```

5. Чтение нот

```
Start1:
       Lpm   tmp,Z    ; считываем константу по адресу, указанному в регистре Z
                        (144) в регистр tmp
Mov   NFreq,tmp   ; скопируем ноту в регистр Nfreq и в регистр r22
Mov   r22,tmp
```

6. а. Выделение кода длительности ноты

```
andi   tmp,0b11100000  ; маскируем код длительности
swap  tmp                ; перестановка полубайтов
lsr     tmp              ;  логический сдвиг вправо (например, (00000101)
```

6. б. Чтение константы длительности

Константа расположена в EEPROM по адресу, начинающемуся с 0x40 (64_{10}).

```
Clr     tmp
```

```
Subi   tmp,-64       ; прибавляем к tmp начальный адрес 64 и по этому адресу чита-
ем константу из EEPROM:
Out    eear,TMP     ;  загрузка адреса в адресный регистр
Sbi    EECR,EERE  ; установить разряд EERE в регистре EECR (для чтения)
Sbi    EECR,EERE  ; так как чтение производится за четыре такта.
In     NTim,EEDR   ; константу запишем в регистр NTIM
Mov    r19,NTim       ; скопируем константу в R19
```

7. Выделение кода частоты ноты

```
Andi   NFreq,0b00011111   ; маскируем частоту
```

8. Сравнение кода частоты с числом 31

```
Cpi    NFreq,31
Breq   pause   ;   если NFreq=31, то переходим на подпрограмму Pause
```

9а.Чтение из EEPROM константы частоты

```
Здесь мы должны адрес умножить на 2, так как константы состоят из 2-х байт
Lsl    NFreq          ; сдвиг влево (умножение на 2)
Out    EEAR,NFreq   ;  загрузка адреса в адресный регистр
Sbi    EECR,EERE          ;  установить разряд EERE в регистре EECR
Sbi    EECR,EERE
In     XL,EEDR            ; младший байт записываем в регистр XL
Inc    Nfreq          ; увеличиваем на 1 адрес NFreq
Out    EEAR,NFreq
Sbi    EECR,EERE
Sbi    EECR,EERE
In     XH,EEDR,EEDR  ; старший байт записываем в регистр XH
```

9б. Загрузка константы частоты в регистр сравнения

```
Out    OCR1AH,XH
Out    OCR1AL,XL
```

1. Формирование длительности ноты и воспроизведение ее

```
Ldi    tmp,0b01000000   ;   включение режима переключения вывода OC1A
Out    TCCR1A,tmp
Time1:
Rcall  Time_125ms   ; вызов подпрограммы задержки 125 мс
Dec    NTim                ; в NTim загружена константа длительности
Brne   Time1               ; когда NTim станет равным нулю, то выполняем програм-
```

му далее

Во время задержки выводятся колебания на вывод OC1A

```
Ldi    tmp,0b10000000
Out    TCCR1A,tmp      ; по этому коду OC1A переключается в 0 и прекращается звучание.
```

3.1. Аналоговый компаратор

Аналоговый компаратор сравнивает значения напряжений, присутствующих на двух выводах микроконтроллера. Результатом сравнения является логическое значение, которое может быть считано из программы. По результату сравнения может быть сгенерировано прерывание. В качестве неинвертирующего входа (AIN0) используется вывод PB0? A в качестве инвертирующего входа (AIN1) – вывод PB1. Выводы портов должны быть сконфигурированы как входы. Внутренние подтягивающие резисторы необходимо отключить.

Структурная схема компаратора приведена на рис. 47.

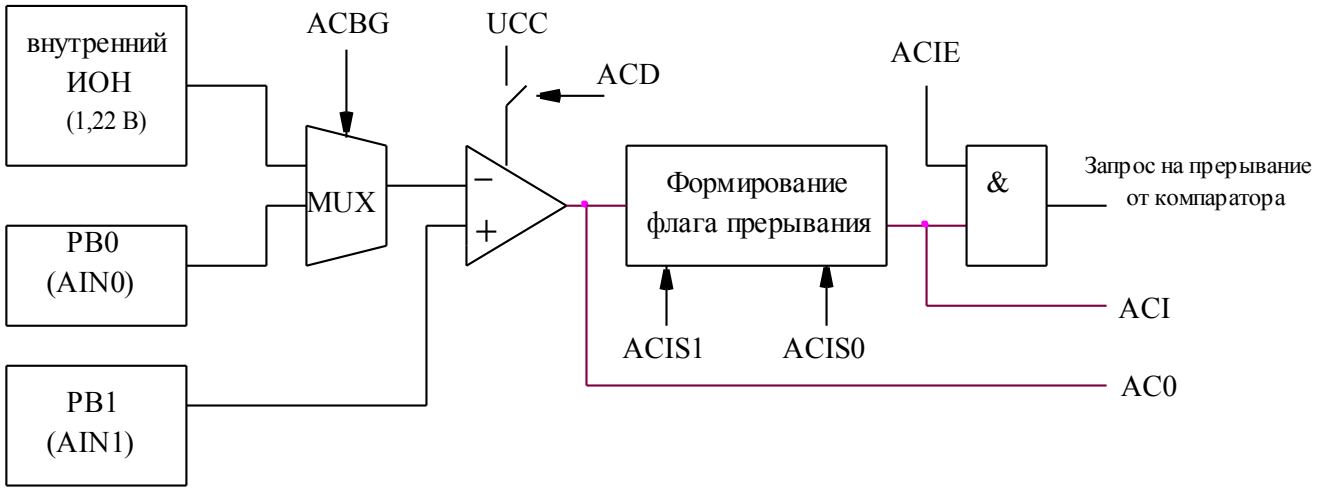


Рис. 48

Управление компаратором и контроль за его состоянием осуществляется регистром ACSR (**A**nalog **C**omporator **S**tatus **R**egister).

Регистр ACSR

7	6	5	4	3	2	1	0
ACD	ACBG	AC0	ACI	ACIE	ACIC	ACIS1	ACIS0

ACD – включение компаратора (0-включено, 1-выключено)
ACBG – подключение к неинвертирующему входу компаратора внутреннего ИОН (0 – не подключен, 1 – подключен)
AC0 – выход компаратора
ACI – флаг прерывания от компаратора
ACIE – разрешение прерывания от компаратора (1 – разрешено)
ACIC – подключение компаратора к схеме захвата таймера T1 (1 – подключен)

ACIS1…ACI0 – условие возникновения прерывания от компаратора.

ACIS1	ACIS0	Условие
0	0	Любое изменение состояния выхода компаратора
0	1	-
1	0	Изменение состояния из 1 в 0
1	1	Изменение состояния из 0 в 1

По своему действию этот узел является обычным компаратором. Если напряжение на выводе AIN0 больше напряжения на выводе AIN1, то результат сравнения равен 0. В противном случае – 1. Результат сохраняется в разряде AC0. регистра ACSR.

При включении питания все разряды регистра ACSR сбрасываюся в 0, поэтому компаратор автоматически включается. Чтобы его выключить, разряд ACD следует установить в 1.

Если состояние выхода компаратора (разряд AC0) изменилось заданным образом, устанавливается флаг прерывания ACI и генерируется запрос на прерывание. Для разрешения прерывания необходимо записать 1 в разряд ACIE и установить флаг I регистра SREG.

Регистр DIDR (**D**igital **I**nput **D**isable **R**egister).

Регистр DIDR

7	6	5	4	3	2	1	0
-	-	-	-	-	-	ANT1D	ANT0D

Если записать 1 в AIN1D и AIN0D, то выводы цифровых буферов разрешены. При этом снижается потребление тока.

Параметры цифрового компаратора:

$U_{см0} = 40 мВ$

$I_{вх0} = 50 нА$

$Время\_отклика - 500 нс$

Пример применения компаратора для определения длительности импульса с выбором предела измерения

Мы можем преобразовать какой либо электрический параметр (сопротивление, напряжение и др.) в длительность импульса. Затем с помощью МК, используя компаратор, можно определить длительность этого импульса, то есть измерить параметр схемы.

Что бы можно было измерять, не используя переключатели пределов измерения, делают автоматический выбор пределов. Далее мы рассмотрим программу, обеспечивающую определение длительности импульсов с автоматическим выбором пределов измерения.

Предел измерения может быть выбран следующим образом. Сначала заполняем измеряемый импульс импульсами высокой частоты (F_T). Количество импульсов равно 10 000. Если импульсов более 10 000, то уменьшаем частоту в 10 раз и 10 000 импульсов займут промежуток времени, в 10 раз больший. И так далее.

Компаратор реагирует на значение амплитуды входного сигнала и выдает значение 0 или 1 в 5-й разряд (AC0) регистра ACSR. На рисунке приведено схематическое изображение процесса измерения.

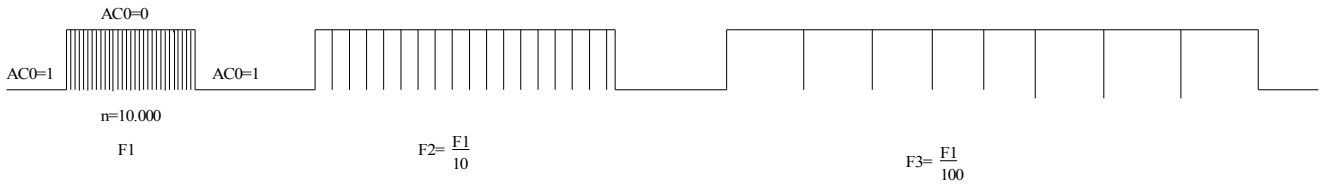


Рис. 49

Алгоритм программы будет выглядеть следующим образом.

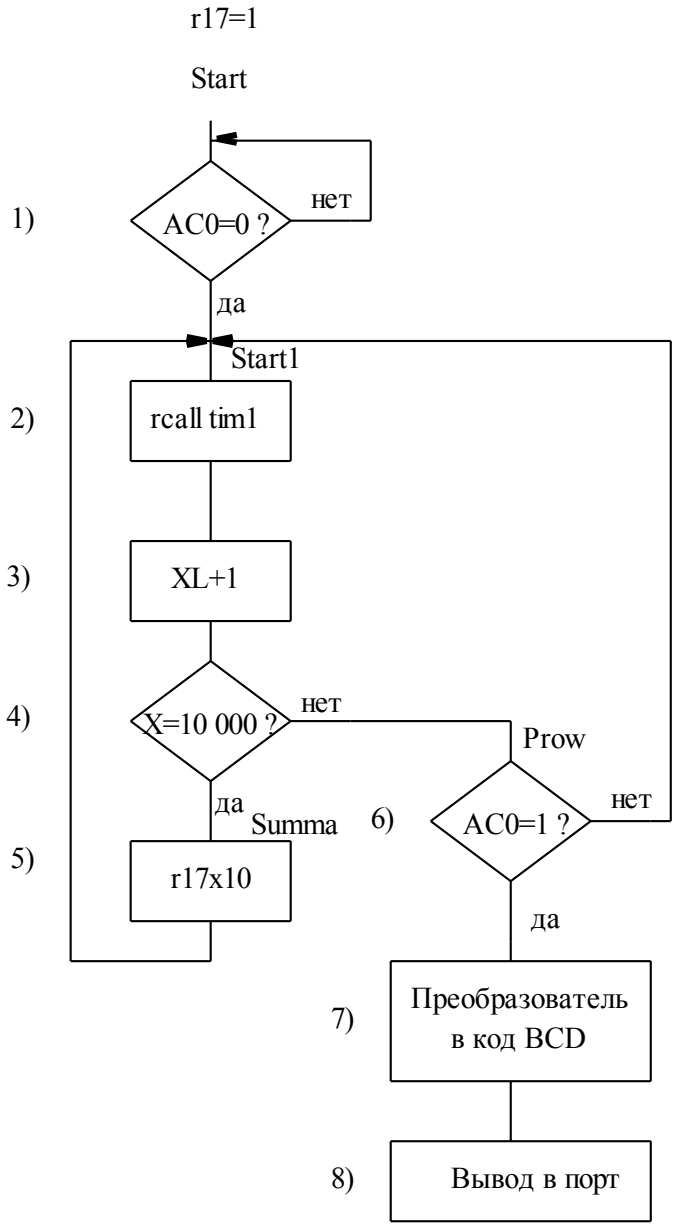


Рис. 50

1). Проверяем состояние разряда AC0 компаратора. Если AC0 = 1, то ждем, пока AC0 станет равным 0. (1 – если напряжение на AIN0<AIN1, и 0 – если AIN0>AIN1).
2). Переход в подпрограмму tim1, которая уменьшает частоты в 10 и 100 раз.
3). Прибавляем 1 к содержимому регистра X.
4). Проверяем, в регистре X число равно 10 000?, если «да», то есть длительность импульса большая, и мы должны уменьшить частоту в 10 раз.
5). Умножаем число в регистре r17 (1) на десять, если число в X > 10 000
6). Если число в регистре X не достигло 10 000, то проверяем AC0 равно 1 ?. Если нет, то возвращаемся к START1? Если «да», то преобразуем двоичное число в код BCD.
7). Выводим полученные значения в порт на индикаторы.
Программа.

```
Sbic   ACSR,5       ; проверяем 5-й разряд (ACO) компаратора
Rjmp start

Start1:
Rcall  tim1
Adiw XL,1   ; прибавить к X единицу
Movw   Y,X  ;   содержимое X копируем в регистр Y
A1:
Ldi       r20,200   ; 10 000:50 = 200
A2:
Sbiw     Y,50
Brmi      Prow
Dec       r20      ; 200-1
Brne     Summa
Rjmp     A2

Summa:
Mov      tmp,r17
Add      tmp,r17
Add      tmp,r17
Add      tmp,r17
.
.
.
Add      tmp,r17       ; повторяем 10 раз

Prow:
Sbis     ACSR,5
Rjmp     Start1
Rjmp     BCD

Tim1:
Mov      r18,r17
Tim2:
Subi     r18, 1     ;вычесть из r18 единицу
Breq     tim2      ; если не ноль, то возврат к tim2
Ret                    ; если 0, то выход из подпрограммы Tim1
```

3.2. Аналого-цифровой преобразователь

Входит в состав моделей ATtiny15, Atmega8, 32, 64, 128 и др.

Это 10-ти разрядный АЦП последовательного приближения. Его параметры:

- Абсолютная погрешность – 2 МЗР
- Нелинейность 0,5 МЗР
- Быстродействие до 15000 выборок/сек
- Входное сопротивление – 100 Мом

На входе АЦП располагается четырехканальный аналоговый мультиплексор, предоставляющий пользователю 4 канала с несимметричными входами, либо 1 канал с дифференциальным входом с возможностью 20-ти кратного усиления.

Для нессиметричных входов диапазон напряжении 0…Vcc. Для дифференциальных – 0…2,6 В. В качестве ИОН для АЦП может использоваться внутренний или внешний источник или напряжение питание микроконтроллера.

АЦП может функционировать в двух режимах:

1 Режим одиночного преобразования – запуск каждого преобразования инициируется пользователем.

2. Режим непрерывного преобразования – запуск преобразований выполняется непрерывно через определенные интервалы времени.

Работа АЦП поразрядного уравновешивания

АЦП состоит из ЦАП, компаратора и регистра преобразования:

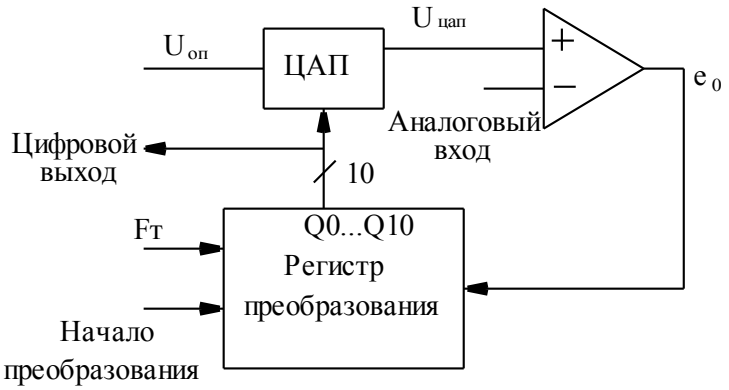


Рис. 51

После подачи импульса «начало преобразования» запускается АЦП.

В первом такте на выходе регистра преобразования в старшем разряде (9) выставляется 1: 0b1000000000. На выходе ЦАП устанавливается $U_{цап} = \frac{1}{2}U_{оп}$. и это напряжение сравнивается компаратором с входным аналоговым напряжением. Здесь возможны два случая:

А). Если $U_{цап} > U_{аналог}$ то e_0=1 и этой единицей сбрасывается 9-й разряд регистра преобразования. Становится 0b0000000000.

Б). Если $U_{цап} < U_{аналог}$ то e_0=0 и 1 в 9-ом разряде остается, то есть остается число 0b1000000000

Во втором такте на выходе регистра преобразования в 8-ом разряде выставляется 1. (А в 9-ом разряде будет 1 или 0 в зависимости от результатов а) или б).

Повторяются пункты а) или б).

В 3-ем такте на выходе регистра преобразования в 7-ом разряде выставляется 1. Повторяются пункты а) или б) и так далее.

Весь процесс займет количество тактов, равное количеству разрядов (10).

Кроме АЦП имеется 4-е мультиплексора:
MUX1 – выбирает опорное напряжение для ЦАП: Vcc – источник питания; AREF – к выводу PB0 (AREF) можно подключить внешний ИОН и фильтрующий конденсатор для повышения помехозащищенности; внутренний ИОН – внутренний источник опорного напряжения 2,56 В.
MUX2, MUX3 – управляет выбором входного сигнала – несимметричный вход или дифференциальный. При дифференциальном входе можно установить коэффициент усиления усилителя 1 или 20.
MUX4 – выбирает тип выходного канала (одиночный или дифференциальный).

Выбор входных мультиплексоров, выбор ИОН осуществляется регистром ADMUX. Формат регистра:

Регистр ADMUX

7	6	5	4	3	2	1	0
REFS1	REFS0	ADLAR	-	-	MUX2	MUX1	MUX0

REFS1…0 – выбор источника опорного напряжения
ADLAR – выравнивание резу3льтата преобразования
MUX2…0 – выбор входного канала

MUX2…0	Несимметричный вход	Диф. вх. Положит.	Диф. вх. Отрицат.	Усиление
000	ADC0 (PB5)			
001	ADC1 (PB2)			
010	ADC2 (PB3)			
011	ADC3 (PB4)			
100			ADC2 (PB3)	1
101		ADC2 (PB3)		20
110			ADC3 (PB4)	1
111		ADC2 (PB3)		20

(100, 101 – калибровка смещения)

Выбор источника опорного напряжения

REFS1	REFS0	ИОН

0	0	Напряжение питания микроконтроллера. Только для каналов с нессиметричным выходом
0	1	Внешний ИОН, подключенный к выводу PB0 (AREF). Внутренний ИОН – отключен
1	0	Внутренний ИОН 2,56 В, отключенный от вывода PB0 (AREF)
1	1	Внутренний ИОН 2,56 В, подключенный к выводу PB0 (AREF)

Выравниваем результата управляет разряд ADLAR. Если этот разряд установлен в 1, то результат преобразования выравнивается по левой границе 16-разрядного слова, если сброшен в 0 – по правой границе.

Регистр ADCSR управляет модулем АЦП и осуществляет контроль за его состоянием.

Регистр ADCSR

7	6	5	4	3	2	1	0
ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0

Тактовым сигналом АЦП является сигнал с предделителя. Коэффициент деления определяется разрядами ADPS2…0.

ADEN – разрешение работы АЦП (1 – вкл., 0 – выкл.)

ADSC – запуск преобразования (1 – начать преобразование)

ADFR – выбор режима АЦП (0 – одиночное преобразование)

ADIF – флаг прерывания от компаратора

ADIE – разрешение прерывания от компаратора

ADPS2…0 – выбор частоты преобразования

Цикл преобразования начинается по первому нарастающему фронту тактового сигнала после установки в 1 разряда ADSC. Длительность цикла составляет 13 тактов. Выборка и запоминание входного сигнала занимает 1,5 тактов. Через 13 тактов преобразование завершается, разряд ADSC аппаратно сбрасывается в 0 (в режиме одиночного преобразования) и результат сохраняется в регистре данных АЦП.

Тактовым сигналом АЦП является сигнал с предделителя. Коэффициент деления определяется разрядами ADPS2…0

ADPS2	ADPS1	ADPS0	Коэффициент деления
0	0	0	2

0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Наибольшая точность преобразования достигается, если тактовая частота модуля АЦП находится в диапазоне 50…200 кГц. Отсюда и выбирается коэффициент деления.

Пример.

Тактовая частота микроконтроллера равна 4 Мгц. Ориентируемся на тактовую частоту АЦП 100кГц. Тогда

$$K = \frac{4000\_кГц}{100\_кГц} = 40$$

Выбираем К=42. Уточняем тактовую частоту АЦП:

$$F_{Тацп} = \frac{4000}{32} = 125\_кГц$$

3.2.1. *Повышение точности преобразования*

1. Надо правильно выбрать тактовую частоту модуля АЦП.
2. Надо уменьшить шумы

   а) конструктивно: земля располагается под аналоговыми компонентами: соединения в одной точке аналоговой и цифровой земли.

   б) свести к минимуму помехи, наводимые ядром процессора: выбрать «спящий режим» - ADC Noise Reduction. В этом режиме функционирует только АЦП и сторожевой таймер.

   Надо переключить АЦП в режим одиночного преобразования и разрешить прерывание от АЦП, после чего перевести микроконтроллер в режим ADC Noise.

   После остановки процессора начнется цикл преобразования. При завершении преобразования будет сгенерировано прерывание от АЦП, которое переведет микроконтроллер в рабочий режим и начнется выполнение подпрограммы обработки этого прерывания.

3.3. Передача данных

Передача данных может осуществляться асинхронно и синхронно.

При синхронной последовательной передаче данных синхронизируется передача отдельных битов данных с помощью одновременно передаваемого передатчиком тактового сигнала. Применяется для обмена данными между микроконтроллером и периферийными схемами.

При асинхронной передаче данных тактовый сигнал не передается. Поэтому требуется строго распределять интервалы времени в каналах передачи и приема. Временная развертка обычно имеет кварцевую стабилизацию. Применение, например, связь между ПК и принтером, модемом и др.

3.3.1. *Стандарты асинхронной передачи данных*

1. Токовый интерфейс.

Сила тока – 20 мА. Устойчив к помехам. Скорость передачи данных 9600 бод (бит/сек). Интерфейс не нормирован, но нашел широкое применение.

Пример передающего контура для токового интерфейса.

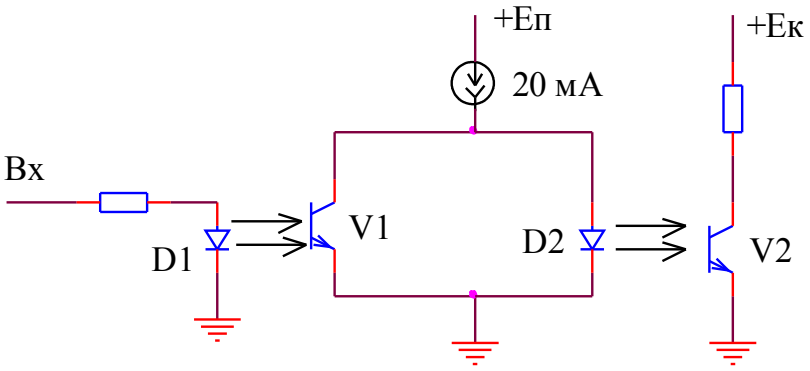


Рис. 52

Соединение двух устройств осуществляется через замкнутый приемо-передаточный контур. В контуре протекает постоянный ток 20 мА.

При передаче лог. «0» транзистор V1 – закрыт и ток протекает через светодиод D2. Транзистор V2 – открыт и на выходе лог «0»

При передаче лог «1» транзистор V1 – открыт и насыщен. Ток 20 мА протекает через транзистор V1. Транзистор V2 – закрыт и на выходе – лог. «1».

Гальваническая развязка между передающей и принимающей стороной происходит с помощью оптронов.

Такие устройства пригодны для передачи сигналов на расстояние до 1 км и находят широкое применение в промышленности.

2. Интерфейс RS232C (протокол V.24)/ Передает сигналы 0 и 1 как значения напряжения. Интерфейс устарел, но находит широкое применение.

На стороне приемника высокому уровню сигнала выделен диапазон -3…-15 В («отметка»), а низкому уровню выделен диапазон +3…+15 В («пробел»).

Для преобразования уровней выпускаются микросхемы, например, MAX232.

Для коммутации надо иметь по крайней мере три провода: скрещенные передающий и принимающий и провод заземления.

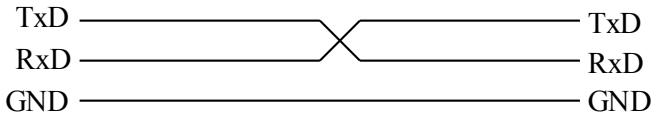


Рис. 53

Максимальная скорость передачи 19000 бод. Расстояние – 20 м.

3. Интерфейс RS423A (протокол V.11).

Скорость передачи до 100 000 бод и дальность до 1200 м. Передатчик и приемник соединены коаксиальным кабелем.

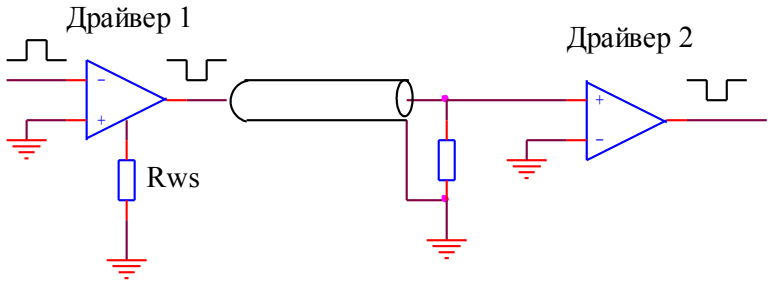


Рис. 54

Rws – настраивает время нарастания сигнала.
Драйверы преобразуют уровень TTL в стандарт V.11.

4. Интерфейс RS422A (протокол V.10).

Передача данных происходит симметрично через витую пару.

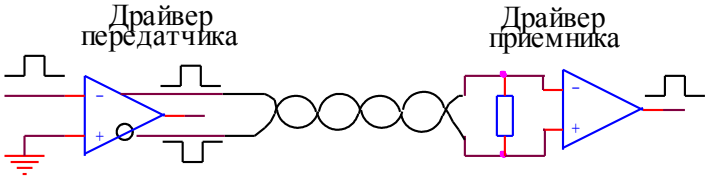


Рис. 55

Имеет высокую помехоустойчивость. Скорость передачи – до 10 Мбод при дальности 1200 м.

График скорости передачи интерфейсов приведен на рис. 56.

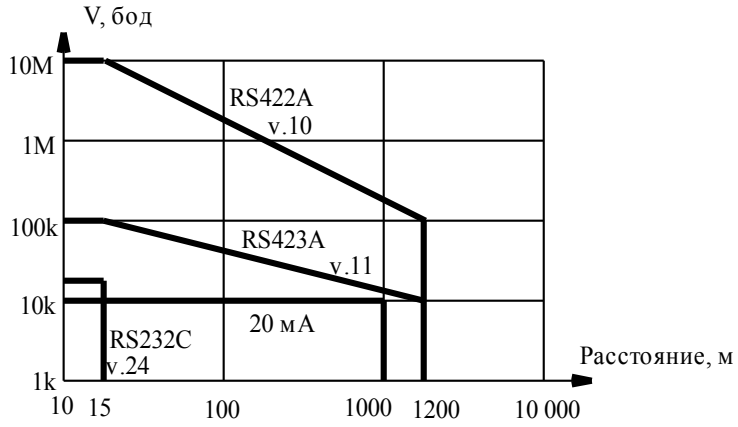


Рис. 56

3.3.2. *Формат передачи по асинхронному интерфейсу*

Согласно стандарту V.24 в лини передачи данных в состоянии ожидания установлена лог 1 («отметка»). Чтобы передать приемнику сообщение о начале передачи, посылается стартовый бит с уровнем лог. 0 («пробел»). После этого следуют разряды данных (7 или 8 разрядов). Сначала передается младший бит.

Для надежности передачи данных добавляют бит четности. Он дополняет разряды данных так, что достигается прямая четность (сумма всех битов, включая и бит четности, четная) или непрямая четность (сумма нечетная). Передача заканчивается стоп-битом с уровнем лог. 1. (продолжительность бита 1; 1,5 или 2 длины бита).

После отправления стоп-бита линия данных переходит в состояние ожидания и готова к следующей передаче.

Пример. Передача байта \$E5 (11100101_b) с непрямой четностью и стоп-битом.

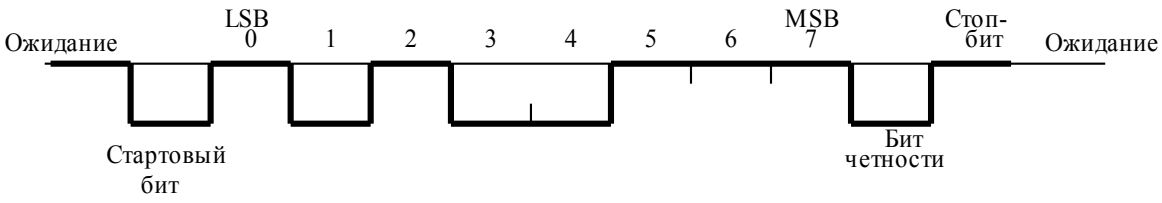


Рис. 53

Различие в длительности тактов на передающей и принимающей стороне не должно превышать 3 %.

3.4. Приемо-передатчик UART

(**U**niversal **A**sinchronous **R**eceiver (прием) and **T**ransmitter (передача)

Все микроконтроллеры семейства AVR оснащены аппаратным приемопередатчиком UART. Он может работать в дуплексном режиме, то есть одновременно передавать и принимать данные.

Для работы UART выделены 4 регистра ввода/вывода:

- регистр управления UCR (**U**art **C**ontrol **R**egister)
- регистр состояния USR (**U**art **S**tatus **R**egister)
- регистр данных UDR (**U**art **D**ata **R**egister)
- регистр настройки скорости передачи данных UBRR (**U**art **B**aund **R**ate **R**egister).

3.4.1. *Передающий элемент UART (трансмиттер)*

Схема вывода данных

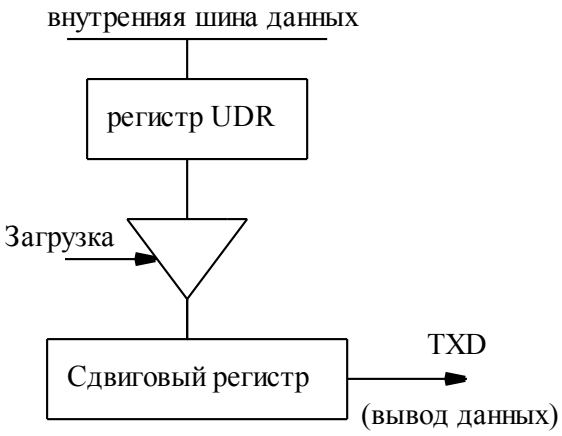


Рис. 54

Например, посылаемый байт 0b00101101.
Сдвиговый регистр:

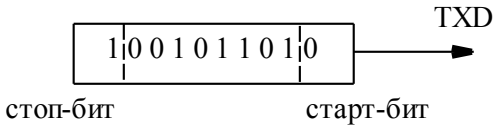


Рис. 55

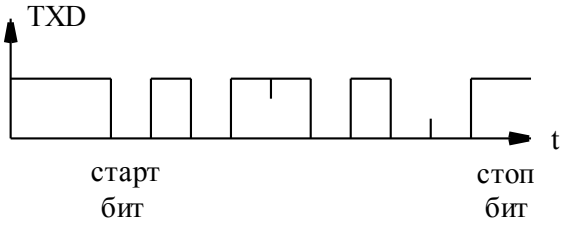


Рис. 56

Скорость выдачи битов на выход передатчика (вывод TXD) определяется параметром **baud rate** (скорость передачи, бод), которым можно управлять.

3.4.2. *Принимающий элемент UART (ресивер)*

Приемник модуля UART непрерывно проверяет состояние входа RXD: на каждый бит приходится 16 отсчетов. При обнаружении на выводе RXD низкого уровня (то есть старт-бита) микроконтроллер пропускает 6 отсчетов, а затем делает 3 выборки (8, 9 и 10 отсчеты). Также происходит выборка и в других битах. Поэтому фронты сигнала не имеют большого значения. Если микроконтроллер обнаруживает в 3-хвыборках лог. 0, то он определяет, что пришел старт-бит и модуль UART начинает считывать приходящий байт.

Если же значения трех отсчетов бита не совпадают, то значение бита принимается равным значению двух одинаковых отсчетов.

Скорость передачи данных управляется регистром UBRR. Она должна быть одинаковой для принимающего и передающего устройств. Скорость передачи определяется по формуле:

$$\text{Baud Rate} = \frac{ck}{16 \times (UBRR + 1)},$$

Где Ск – тактовая частота. [UBRR состоит из двух регистров UBRRH : UBRRL, в которые заносятся числа].

Например. $F_{рез}$=4 МГц. В регистр UBRR записано число 25. Тогда

$$\text{boud Rate} = \frac{4000\,000}{16 \times 26} = 9615 бит/с(бод)$$

Имеется набор стандартных скоростей передачи: 2400, 4800, 9600 и т.д., которой надо придерживаться для совместимости устройств. Поэтому используют резонаторы с частотами от 1,8432 до 9,216 МГц.

Если взять 3,6864 МГц, то при UBRR=23 скорость передачи равна 9600 бод.

Пример. Передать число из рабочего регистра Iden другому устройству UART.

```
Ldi     tmp,0b00001000 ; разрешение передачи
Out     UCR,tmp
Out     UDR,Iden  ; посылаем число
```

UART можно использовать для связи с последовательным портом компьютера (RS-232). Чтобы послать данные через последовательный порт можно воспользоваться программой Hyper Terminal. (Пуск/Программы/Стандартные/Связь). Здесь можно создать соединение с конкретным портом (например, COM1), выбрать скорость передачи, количество передаваемых битов, установить контроль четности и т.д.

Выводы разъема порта подключаются к выводам RXD и TXD через преобразователь уровней: MAX232, AD232 и др.

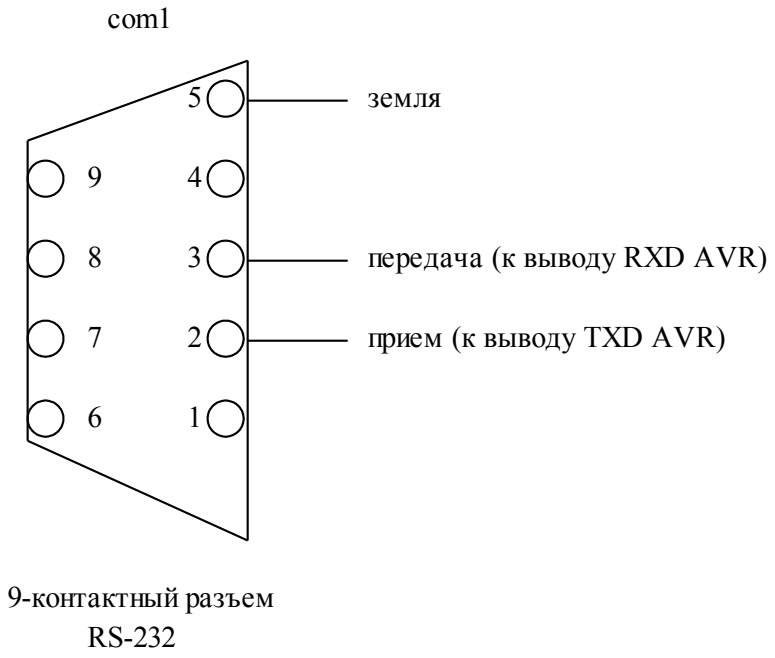


Рис. 54

Регистр управления UCR имеет формат:

Регистр UCR

7	6	5	4	3	2	1	0
RXCIE	TXCIE	UDRIE	RXEN	TXEN	CHR9	RXB8	TXB8

7р. RXCIE – разрешение прерывания по завершению приема (0 – запрещено, 1 – разрешено).
6р. TXCIE – разрешение прерывания по завершению передачи (0 – запрещено, 1 – разрешено).
5р. UDRIE – разрешение прерывания при очистке регистра данных UART:
0 – перывание «регистр данных UART пуст» запрещено,
1 – перывание «регистр данных UART пуст» разрешено.
4р. RXEN – разрешение приема (0 – запрещено, 1 – разрешено).
3р. TXEN – разрешение передачи (0 – запрещено, 1 – разрешено).
2р. CHR9 – формат посылка
0 – 8-битные данные + старт-бит и стоп-бит)
1 – 9-битные данные + старт-бит и стоп-бит).
1р. RXB8 – 8-й бит принимаемых данных (при обмене 9-битными данными).
0р. TXB8 – 8-й бит передаваемых данных (при обмене 9-битными данными).

Регистр состояния USR имеет формат:

Регистр USR

7	6	5	4	3	2	1	0
RXC	TXC	UDRE	FE	OR	-	-	-

7р. RXC – прием завершен (устанавливается в 1 после приема символа и сохранению его в UDR).
6р. TXC – передача завершена (устанавливается в 1 после передачи символа и отсутствия других данных в UDR).
5р. UDRE – регистр данных UART пуст:
0 – байт, содержащийся в регистре UDR, еще не перегружен в сдвиговый регистр
1 – содержимое регистра UDR перегружено в сдвиговый регистр.
4р. FE - ошибка кадрирования
0 – ошибка отсутствует (корректный стоп-бит).
1 – обнаружена ошибка кадрирования (некорректный стоп-бит).
3р. OR – переполнение
0 – содержимое UDR успешно перегружено в сдвиговый регистр
1 – содержимое UDR было перезаписано до того, как байт был скопирован в сдвиговый регистр.

3.4.3. Передача данных

Работа передатчика разрешается установкой в 1 разряда TXEN регистра UCR. При установке разряда вывод TXD подключается к передатчику UART и начинает функционировать как выход независимо от установок регистров управления портом.

Передача инициируется записью передаваемых данных в регистр данных UDR. После этого данные пересылаются из регистра UDR в сдвиговый регистр передатчика.

После пересылки слова данных в сдвиговый регистр флаг UDRE регистра UCS устанавливается в 1, что означает готовность передатчика к получению нового слова данных. В этом состоянии флаг остается до следующей записи в буфер. Одновременно с пересылкой в регистре формируется служебная информация – старт-бит, возможный бит четности, а так же один или два стоп-бита.

После загрузки сдвигового регистра его содержимое начинает сдвигаться вправо и поступать на вывод TXD. Скорость сдвига определяется настройками контроллера тактовых сигналов.

Если во время передачи в регистр UDR было записано новое слово данных, то после передачи последнего стоп-бита оно пересылается в сдвиговый регистр. Если к моменту окончания передачи кадра такой записи выполнено не было, устанавливается флаг прерывания «передача завершена» TXC регистра UCR. Сброс флага осуществляется аппаратно при входе в подпрограмму обработки соответствующего прерывания или программно, записью в этот разряд лог. 1.

Выключение передатчика осуществляется сбросом разряда TXEN регистра UCR. Если в момент выполнения этой команды осуществлялась передача, сброс разряда произойдет только после завершения текущей и отложенной передач, то

есть после очистки сдвигового и буферного регистров передатчика. При выключенном передатчике вывод TXD может использоваться как контакт ввода/вывода общего назначения.

Ниже приведен пример подпрограммы передачи по интерфейсу UART. Эта подпрограмма ждет очистки буфера передатчика, а затем загружает в него новое значение.

```
UART_Transmit:
Sbis    USR,UDRE    ;  ждем очистки буфера передатчика (регистр данных пуст)
Rjmp    UART_Transmit
;  скопировать 9-й разряд данных из r17 в TXB8
Cbi     UCR,TXB8    ;   очистить TXB8 регистра UCR
Sbrc    r17,0               ; проверяет 0-й бит рег. R17 и пропускает следующую команду, если бит установлен
Sbi     UCR,TXB8    ;  установить в 1 бит TXB8
;  загрузить младший байт данных в буфер, начать передачу
Out     UDR,r16
Ret
```

3.4.4. Прием данных

Работа приемника разрешается установкой разряда RXEN регистра UCR. При установке разряда вывод RXD подключается к приемнику UART и начинает функционировать как вход независимо от установок регистров управления портом.

Прием данных начинается сразу же после обнаружения приемником корректного старт-бита. Каждый разряд содержимого кадра затем считывается с частотой, определяемой установками контроллера скорости передачи. Считанные разряды данных последовательно помещаются в сдвиговый регистр приемника до обнаружения первого стоп-бита. После этого содержимое сдвигового регистра пересылается в буфер приемника, из которого принятое значение может быть получено путем чтения регистра данных модуля. Блок приемника имеет два флага, показывающий состояние обмена: флаг ошибки кадрирования FE и флаг переполнения OR. Флаг FE устанавливается в 1, если значение первого стоп-бита принятого кадра не соответствует требуемому, т.е. равно 0.

Флаг OR индицирует потерю данных из-за переполнения буфера приемника. Установленный флаг OR означает, что между прошлым байтом, считанным из регистра UDR, и байтом, считанным в данных момент, произошла потеря одного или нескольких кадров.

Для индикации состояния приемника в модуле UART используется флаг прерывания «прием завершен» RXC регистра USR. Этот флаг устанавливается в 1 при наличии в буфере приемника непрочитанных данных. После прочтения флаг сбрасывается в 0.

Выключение приемника осуществляется сбросом разряда RXEN регистра UCR. Приемник выключается сразу же после сброса разряда, т.е. кадр, принимаемый в этот момент, теряется.

Пример.

```
UART_Receive:
; ждать загрузки данных в буфер приемника
Sbis    USR,RXC
Rimp    UART_Receive
; прочитать младший байт данных
In      r18,UDR
```

В микроконтроллерах семейства Mega имеется универсальный синхронно/асинхронный приемопередатчик USART. Модули USART при работе в асинхронном режиме совместимы с модулями UART как по расположению разрядов управляющих регистров, так и по функционированию. При работе модуля в синхронном режиме изменение состояния вывода TXD происходит по одному из фронтов сигнала XCK . Если разряд UCPOL равен 0, изменение состояния вывода происходит по нарастающему фронту, если же – 1 , то по спадающему фронту.

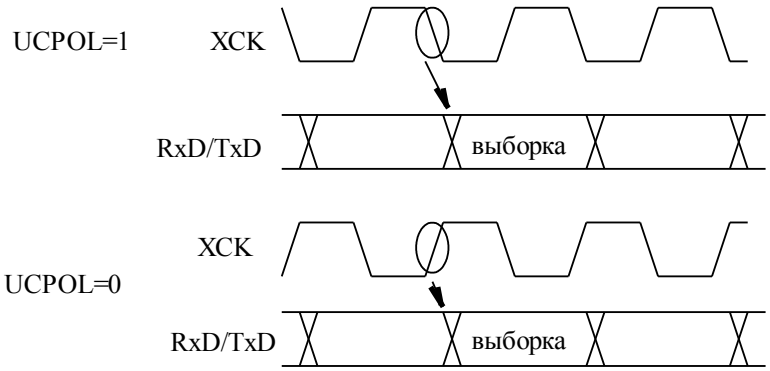


Рис. 55

3.5. Синхронная передача данных через последовательный интерфейс SPI (**S**erial **P**eripheral **I**nterface)

Через интерфейс SPI можно обмениваться данными между ведущим микроконтроллером (Master) и одним или несколькими ведомыми (Slave).

Каждый микроконтроллер, имеющий интерфейс SPI? Может быть сконфигурирован на режимы как Master, так и Slave.

Интерфейс SPI обеспечивает полнодуплексный обмен – посылку данных в обоих направлениях одновременно. Интерфейс реализует синхронный режим передачи.

Ведущий формирует тактовый сигнал, синхронизирующий обмен, и определяет, кто должен передавать данные (он или ведомый).

Схема соединения микроконтроллеров приведена на рис. 56.

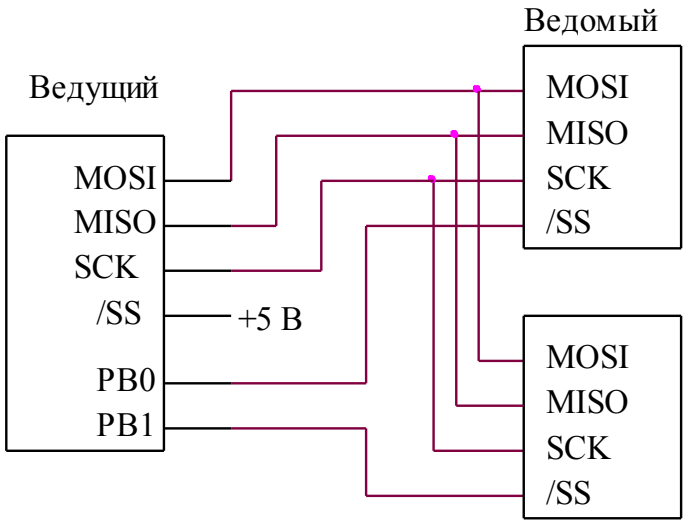


Рис. 56

На рисунке обозначено:
SCK (**S**PI **C**lo**k**) – тактовый сигнал.
MOSI (**M**aster **O**ut, **S**lave **I**n) – выход ведущего, вход ведомого.
MISO (**M**aster **I**n, **S**lave **O**ut) – вход ведущего, выход ведомого.
/PS (**S**lave **s**elect) – выбор ведомого.

3.5.1. *Регистры ввода/вывода модуля SPI*

Имеется три регистра:
.SPDR (**SP**I **D**ata **R**egister) – регистр данных; содержит посылаемый или принятый байт данных.
.SPCR (**SP**I **C**ontrol **R**egister) – регистр управления, определяет функционирование модуля SPI.
.SPSR (**SP**I **S**tatus **R**egister) – регистр состояния, отображает состояние модуля SPI.

Регистр SPDR – восьмиразрядный регистр, в который помещается байт для передачи его другому устройству, или помещается принятый байт.
Регистр управления SPCR. Формат регистра:

Регистр SPCR

7	6	5	4	3	2	1	0
SPIE	SPE	DORD	MSTR	CPOL	CPHA	CPR1	CPR0

7 р. SPIE – разрешение прерывания от SPI: 0 – запрещено; 1 – разрешено.
6 р. SPE - разрешение SPI: 0 – модуль SPI выключен; 1 – модуль SPI включен.
5 р. DORD – порядок передачи данных: 0 – первым передается старший разряд;

1 – первым передается младший разряд.

4 р. MSTR – выбор режима работы: 0 – режим ведомого; 1 – режим ведущего.

3 р. CPOL – полярность тактового сигнала: 0 – во время ожидания на SCK при сутствует низкий уровень; 1 - во время ожидания на SCK присутствует высокий уровень.

2 р. CPHA – фаза тактового сигнала: 0 – данные считываются по нарастающему фронту SCK; 1 – по спадающему фронту SCK.

1р. CPR1, CPR0 – скорость передачи

0	0	SCK=CK/4
0	1	SCK=CK/16
1	0	SCK=CK/64
1	1	SCK=CK/128

Регистр SPSR имеет формат (регистр состояния):

Регистр SPSR

7	6	5	4	3	2	1	0
SPIF	WCOL	-	-	-	-	-	-

7 р. SPIF (**SP**I **I**nterrupt **F**lag) – флаг прерывания от интерфейса SPI. Этот флаг указывает на завершение передачи и вызывает запрос на прерывание, как только в регистре SPCR будет установлен разряд SPIE, а в регистре SREG – разряд общего разрешения прерывания I.

6 р. WCOL (**W**rite **COL**lision) – конфликт записи. Устанавливается в случае, если во время передачи данных делается попытка записи в регистр данных SPI, что приводит к разрушению переданного байта данных. При этом текущая передача доводится до завершения, а новый байт не записывается в сдвиговый регистр. Флаг WCOL должен быть сброшен пользователем вручную.

Перед выполнением обмена необходимо разрешить работу модуля SPI. Для этого устанавливается в 1 разряд SPE регистра SPCR. Режим работы определяется состоянием разряда MSTR. Передача данных осуществляется следующим образом. При записи в регистр данных SPI ведущего микроконтроллера запускается генератор тактового сигнала модуля SPI и данные начинают поразрядно выдаваться на вывод MOSI и соответственно поступать на вывод MOSI ведомого микроконтроллера. После выдачи последнего разряда текущего байта генератор тактового сигнала останавливается с одновременной установкой в 1 флага «конец передачи» (SPIF). Если прерывания от модуля SPI разрешены (флаг SPIE регистра SPCR установлен в 1) , генерируется запрос на прерывание . После этого ведущий микроконтроллер может начать передачу следующего байта, либо, подав на вход / SS ведомого микроконтроллера напряжение высокого уровня, перевести последний в режим ожидания.

3.5.2. *Протокол передачи данных*

Формат передачи выбирается с помощью разрядов CPHA и CPOL регистра управления SPCR (фаза и полярность тактового сигнала).

CPHA=0 – данные считываются по нарастающему фронту SCK.

CPOL=0 – во время ожидания на SCK присутствует низкий уровень.

Соответствующие этим режимам форматы обмена данными через SPI приведены на рис. 57.

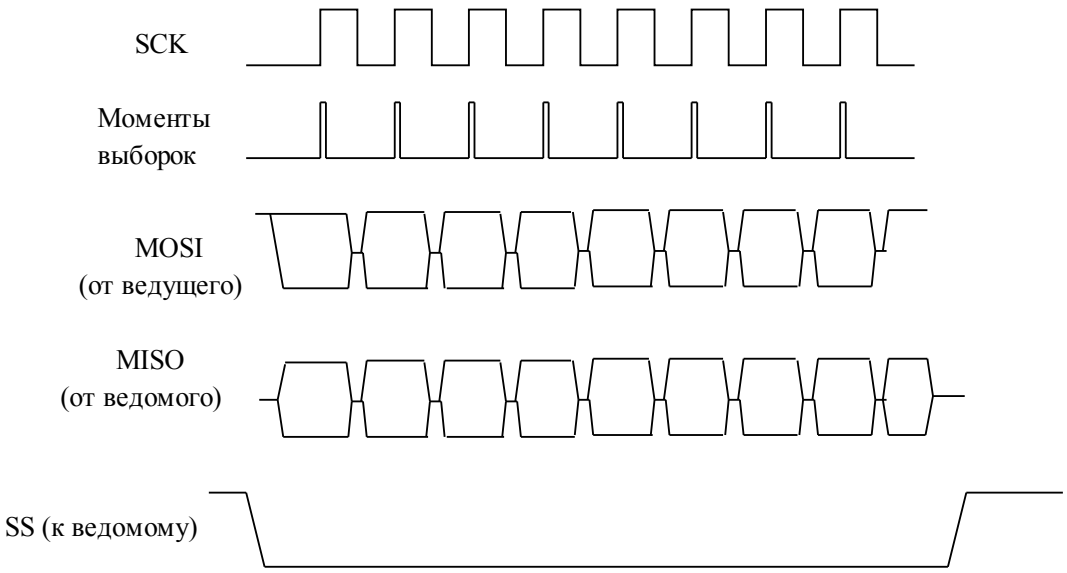


Рис. 57

Сначала записывается передаваемый байт в регистр данных SPDR. Чтобы начать передачу данных, Master переводит линию /SS в состояние логического 0. По нарастающему фронту тактового импульса принимаются биты, расположенные на входах Master и Slave. По ниспадающему фронту тактового импульса следующий бит сдвигается. По окончании передачи данных в регистре состояния SPSR устанавливается флаг прерывания SPIF и может генерироваться прерывание.

3.6. **Последовательный двухпроводный интерфейс TWI**

Интерфейс TWI (**T**wo-**w**ire Serial **I**nterface) является аналогом интерфейса I^2C фирмы Philips (I^2C – Inter-IC - шина для передачи данных между интегральными схемами). Применяется в микроконтроллерах семейства Atmega. Зона действия шины – до 3 м.Можно объединять до 128 устройств с помощью двух линий.

.SCL (**S**erial **C**lock **L**ine) – линия последовательной передачи синхроимпульсов.

.SDA (**S**erial **DA**ta Line) – линия последовательной передачи данных.

Дополнением к линии являются два подтягивающих резистора.

3.6.1. *Соединение устройств шиной TWI*

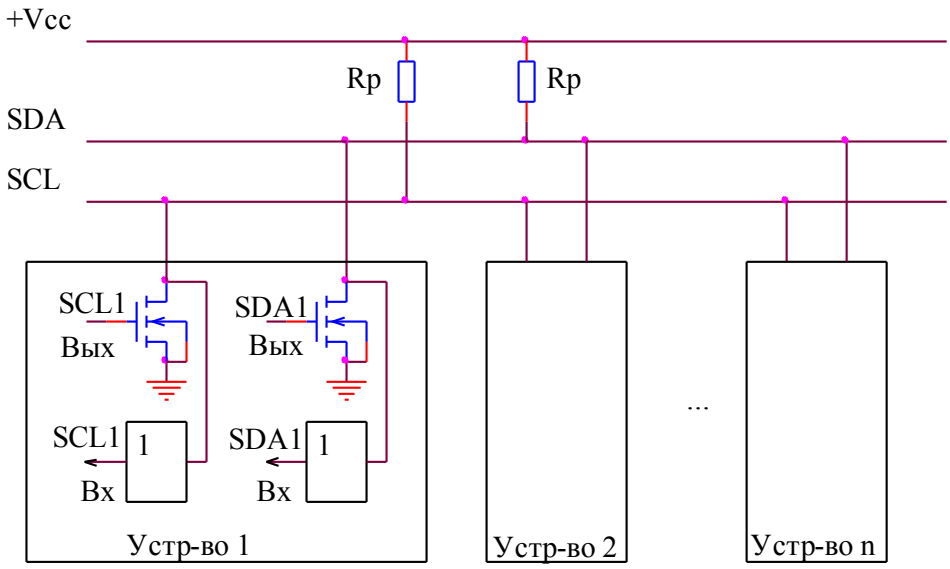


Рис. 58

Линия SDA. Пока через шину не передаются данные, линия SDA имеет высокий потенциал. Данные по линии SDA передаются последовательно. Они действительны в фазе высокого уровня такта и могут изменять состояние только в фазе низкого уровня. Каждый блок, подсоединенный к шине, во время передачи данных может быть или приемником (ресивером) или передатчиком (трансмиттером).

По шине TWI вместе с данными передается и служебная информация. Их совокупность называется пакетом.

Скорость передачи данных определяется частотой тактового сигнала в линии SCL. Верхняя граница – 100 кГц. Нижняя – не ограничена. Требования к фронтам: фронт сигналов должен быть не более 300 нс. Отсюда емкостная нагрузка линий не должна превышать 400 пФ.

3.6.2. *Протокол шины*

Описывает процесс передачи данных по шине и не допускает ошибочной интерпретации состояния шины.

Протокол передачи байта данных по TWI приведен на рис. 59.

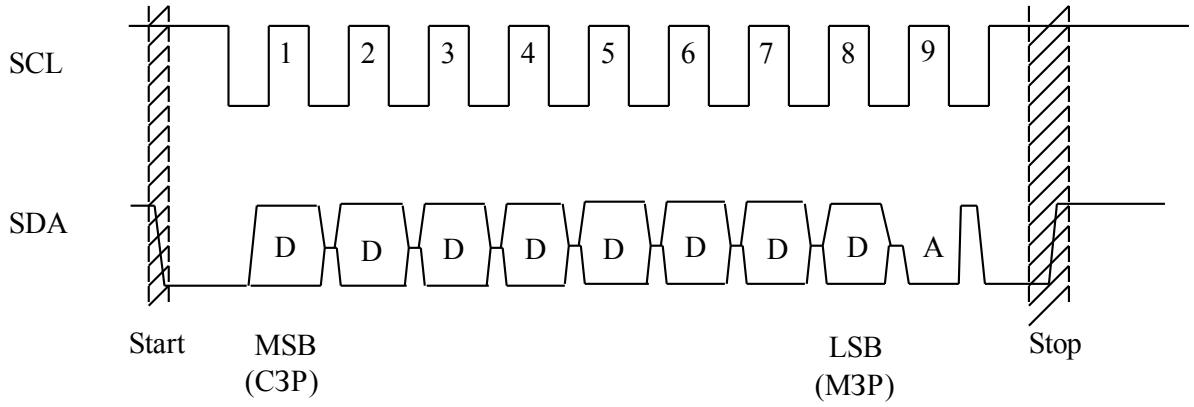


Рис. 59

Состояние START формируется при смене уровня SDA с 1 на 0 при высоком уровне SCL. Состояние STOP – при смене уровня SDA с 0 на 1 при высоком уровне SCL.

B – бит данных. Бит не может быть изменен при высоком уровне SCL (изменяется только при низком уровне SCL).

A – бит квитирования. Вырабатывается приемником. Подтверждает получение байта.

3.6.3. *Условие начала передачи*

Лини SCL и SDA находятся в состоянии высокого уровня. Ведущее устройство, которое хочет начать передачу данных, изменяет состояние шины SDA к низкому уровню.

Линии SCL и SDA находятся в состоянии высокого уровня. Ведущее устройство, которое хочет начать передачу данных, изменяет состояние линии SDA к низкому уровню (рис. 60).

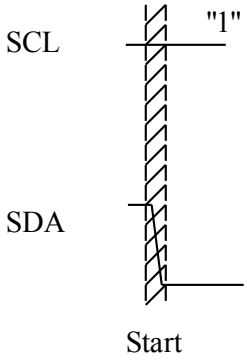


Рис. 60

После наступления условия начала передачи шина будет занята ведущим устройством, создавшим это условие. Другие ведущие устройства будут заблокированы.

Все устройства, подключенные к шине, должны распознать условие начала передачи и переключиться на прием.

Устройство, создавшее условие начала передачи, является для шины ведущим, а остальные – ведомыми. Ведущее устройство отвечает за тактовый сигнал и становится передатчиком.

3.6.4. *Бит подтверждения*

Этот бит (acknowledge) – это реакция приемника на принятый байт данных. Он является для передатчика признаком того, что приемник физически присутствует и «прослушивает» линию. Вместе с тем, бит подтверждения можно рассматривать как сигнал синхронизации.

Бит подтверждения генерируется приемником после каждого байта, кроме последнего. Такое отрицательное квитирование сообщает, что передача данных завершена.

На рис. 61 показана выработка бита квитирования приемником.

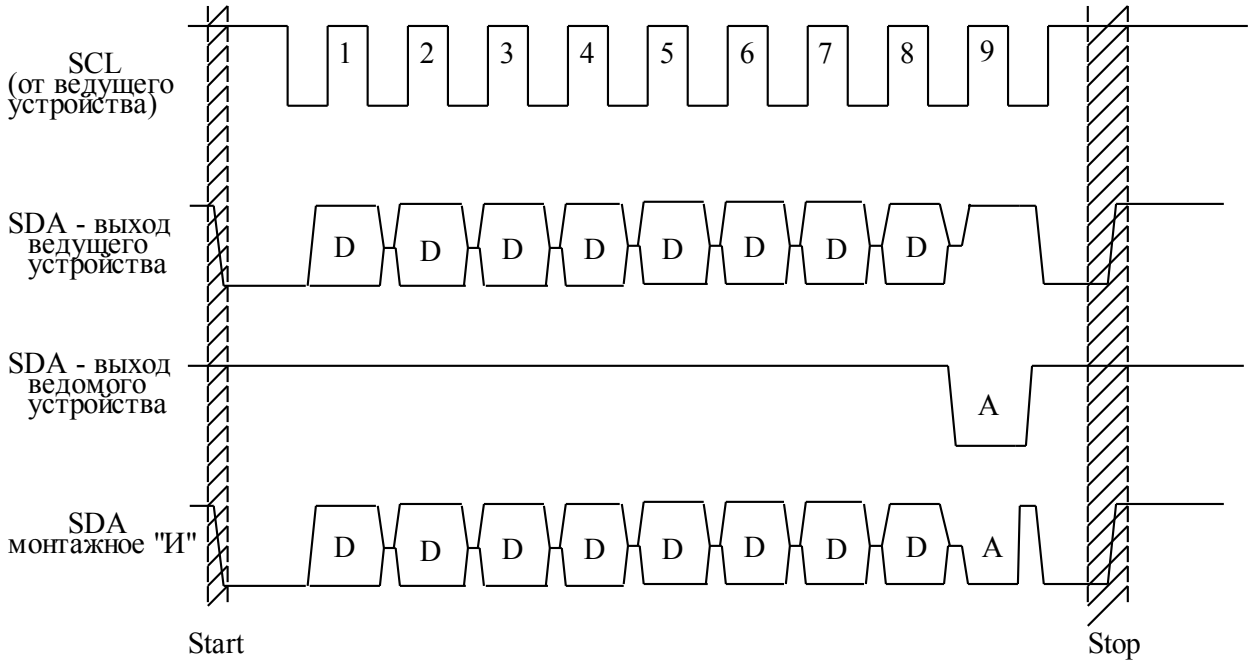


Рис. 61

3.6.5. *Условие завершение передачи*

Изменение в линии SDA уровня сигнала с 0 на 1, когда на SCL – высокий уровень, оценивается как условие завершения передачи (рис. 62).

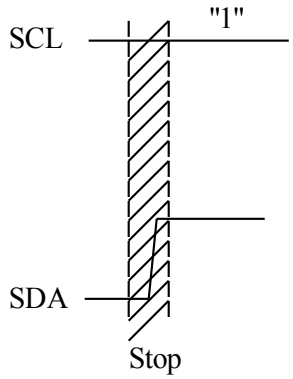


Рис. 62

3.6.6. *Адресация ведомого устройства*

А) Адрес ведомого устройства.

Первый байт данных представляет собой адрес ведомого устройства. Он сопоставлен определенному устройству и имеет длину 7 бит. Теоретически можно адресовать до 128 ведомых устройств (2^7). В действительности система состоит не более, чем из 20 ведомых блоков. Адрес ведомого устройства состоит из двух частей: постоянной и переменной (рис. 63).

Постоянная часть адреса ведомого устройства				Переменная часть адреса ведомого устройства			Направление передачи данных
7	6	5	4	3	2	1	0 R/W

Рис. 63

б) Постоянная часть адреса.

Описывает требования к определенным группам устройства и определяется изготовителем кристалла. Постоянная часть адреса жестко прошита в интегральной схеме и не может быть изменена пользователем.

в) Переменная часть адреса.

Служит для выбора устройства из группы однотипных кристаллов, среди которых все имеют постоянную часть адреса ведомого устройства. Переменная часть определяется пользователем.

г) Разряд направления передачи данных.

Разряды 7…1 идентифицируют ведомый блок. 0-й разряд задает направление передачи данных, то есть данные должны быть приняты или переданы.

Если разряд 0 равен 1, то ведущее устройство находится в режиме приемника, а ведомое – передатчика.

Если разряд 0 равен 0, то ведущее устройство работает как передатчик, а ведомое – как приемник. При этом ведомые устройства используют автоинкремент

при внутренней адресации и данные сохраняются в адресах. Следующих один за другим.

Адрес ведомого устройства подтверждается этим устройством битом квитирования.

3.6.7. *Модуль TWI микроконтроллера*

Структурная схема модуля показана на рис. 64.

Рис. 64

Взаимодействие программы с модулем TWI осуществляется с помощью пяти регистров ввода/вывода:

TWBR – регистр скорости передачи
TWSR – регистр состояния (Status)
TWAR – регистр адреса (Adress)
TWDR – регистр данных (Data)
TWCR – регистр управления (Control).

Для подключения модуля к шине используются выводы SCL и SDA, которые являются линиями портов ввода/вывода. К ним можно подключить внутренние подтягивающие резисторы.

Контроллер скорости передачи (**B**it **R**ate Generator)

Блок задает период следования импульсов на линию SCL с помощью регистра TWBR.

$$f_{scl} = \frac{f_{clk}}{16 + 2TWBR}.$$

Здесь f_{clk} – тактовая частота процессора, TWBR – значение, записанное в регистр TWBR.

Блок шинного интерфейса (регистр данных)

В блок входят:

1. Контроллер состояний Старт/Стоп.

2. Контроллер арбитража. Определяет конфликты на шине.
3. Регистр данных TWDR. Содержит данные передаваемого или принимаемого пакета. Это сдвиговый регистр. Одновременно с выдвиганием содержимого регистра на шину, данные с нее вдвигаются в этот регистр.

Регистр адреса проверяет принятый адрес на соответствие значению, находящемуся в регистре TWAR. Формат регистра TWAR:

Регистр TWAR

7	6	5	4	3	2	1	0
TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE

TWA6…0 - в разрядах содержится адрес, на который устройство будет отзываться при работе в режиме ведомого.
TWGCE – разрешение распознавания общих вызовов. .если 1 – отзывается на общие вызовы, 0 – распознавание общих вызовов запрещено.

Блок управления (регистр состояния TWCR и регистр управления TWCR.
При наступлении определенных событий блок управления формирует в регистре состояния TWSR код статуса и устанавливает флаг запроса на прерывание. До момента сброса этого флага на линии SCL удерживается низкий уровень, приостанавливая передачу данных по шине.

Формат регистра TWCR:

Регистр TWCR

7	6	5	4	3	2	1	0
TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE

7 р. TWINT. Флаг прерывания от модуля TWI. Сброс флага осуществляется записью в 7 р. 1.

6 р. TWEA. Разрешение бита подтверждения. Если 1 – то устройство формирует сигнал подтверждения, когда это необходимо. 0 – не формирует.

5 р. TWSTA. Флаг состояния Старт. При записи в 5 р. 1 модуль проверяет состояние шины TWI и если шина свободна, формирует сигнал СТАРТ. Если шина занята, модуль ожидает состояния СТОП и затем формирует СТАРТ.

4 р. TWSTO. Флаг состояния СТОП. В режиме ведущего установка флага TWSTO в 1 формирует состояние СТОП.

3 р. TWWC. Флаг конфликта записи. Флаг устанавливается в 1 при попытке записи в регистр TWDR, когда флаг прерывания TWINT – сброшен.

2 р. TWEN. Разрешение работы модуля TWI. При записи 1 модуль TWI включается и берет на себя управление выводами SCL и SDA. При сбросе разряда TWEN в 0 модуль TWI отключается.

0 р. TWIE. Разрешение прерывания от модуля TWI. Если записана 1 и флаг I регистра SREG установлен в 1, то прерывание от модуля TWI разрешено.

Формат регистра состояния TWSR:

Регистр TWSR

7	6	5	4	3	2	1	0
TWS7	TWS6	TWS5	TWS4	TWS3	-	-	-

Регистр находится в состоянии «только чтение». Разряды TWS7…3 отражают состояние модуля TWI.

3.6.8. *Взаимодействие прикладной программы с модулем TWI*

Взаимодействие основывается на использовании прерывания от модуля, которое возникает после каждого события на шине (прием байта, формирование СТОП, СТАРТ и т.п.).

Если прерывание от модуля TWI запрещено, то необходимо постоянно следить за флагом TWINT для реагирования на события, происходящие на шине.

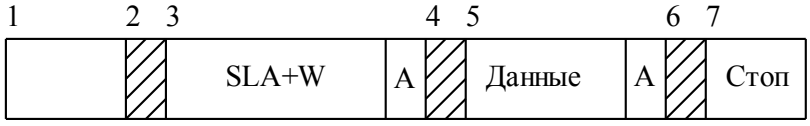
Установка флага TWINT регистра TWCR означает, что модуль TWI закончил выполнение очередной операции и ожидает реакции программы.

В разрядах TWS7…3 регистра TWSR формируются значения, характеризующие текущее состояние шины TWI. Программа должна проанализировать это значение и задать дальнейшее поведение модуля TWI (с помощью регистров TWCR и TWDR).

Рассмотрим взаимодействие прикладной программы с модулем TWI.

Действие: передача одного байта данных от ведущего к ведомому.

Состояние шины TWI:



1...7 - шаги программы

- флаг TWINT установлен

1 – выполнить запись в TWCR для формирования состояния СТАРТ.

2 – флаг TWINT устанавливается в 1. Код статуса в TWSR подтверждает успешное формирование состояния СТАРТ.

3 – убедиться в успешном формировании состояния СТАРТ. Загрузить SLA+W (адресный пакет со сброшенным битом Ц – передача данных) и в регистр TWDR и сформировать требуемые флаги в TWCR, сбрасывая флаг TWINT.

4 – флаг TWINT устанавливается в 1. Код статуса в TWSR подтверждает передачу байта SLA+W и прием A.

5 – убедиться в успешной передаче байта SLA+W и получении A. Загрузить данные в TWDR и сформировать флаги в TWCR? Сбрасывая флаг TWINT.

6 – флаг TWINT устанавливается в 1. Код статуса в TWSR подтверждает передачу данных и прием A

7 – Убедиться в успешной передаче данных и получении A. Сформировать флаги в TWSR, сбрасывая TWINT.

Любой этап взаимодействия прикладной программы с модулем TWI состоит из трех частей:

1 ч. После завершения модулем выполнения какой-либо операции, он устанавливает флаг TWINT регистра TWCR в 1 и ожидает реакции программы. Пока флаг равен 1, на линии SCL удерживается низкий уровень.

2 ч. После установки флага TWINT пользователь должен занести в регистры модуля значения, соответствующие следующему этапу обмена.

2 ч. После обновления содержимого регистров модуля, пользователь должен сформировать в регистре TWCR команду на выполнение следующего этапа обмена. При загрузке в регистр нового значения надо сбросить флаг TWINT. После сброса флага модуль начнет выполнение операции, определяемой содержимым регистра TWCR.

3.6.9. *Режимы работы модуля TWI*

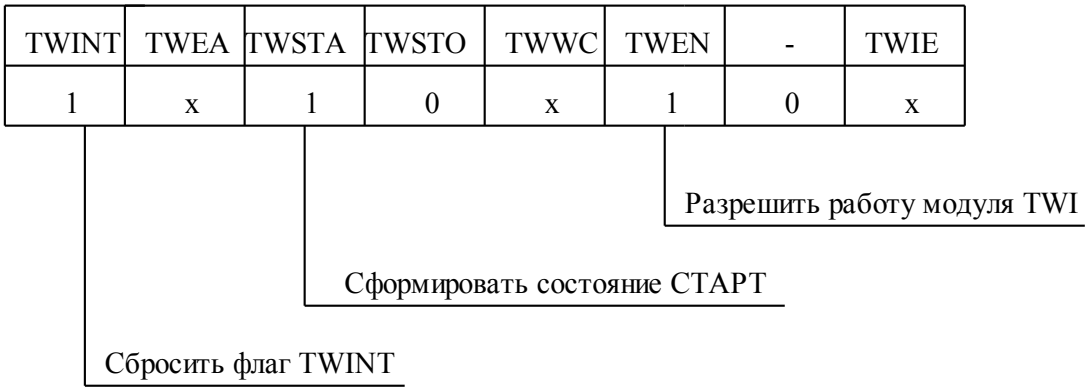
1. Ведущий передатчик (Master Transmitter)
2. Ведущий приемник (Master Receiver)
3. Ведомый передатчик (Slave Transmitter)
4. Ведомый приемник (Slave Receiver).

3.6.10. *Режим работы «ведущий передатчик»*

В этом режиме осуществляется передача данных от ведущего устройства к ведомому. Для переключения устройства в режим ведущего модуль TWI должен сформировать на шине состояние СТАРТ. Формат адресного пакета, передаваемого затем, определяет, в каком из режимов будет работать ведущий.

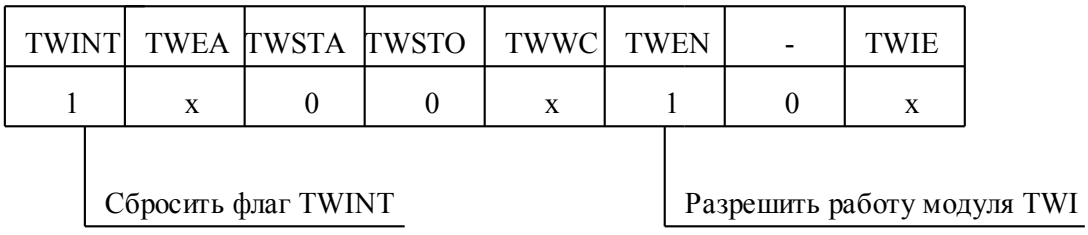
При передаче пакета SLA+W TWI переходит в режим «ведущий передатчик».

Формирование состояния СТАРТ начнется после записи в регистр TWCR значения:



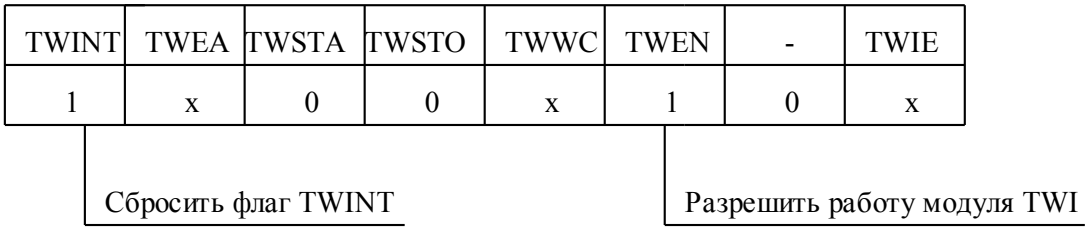
После записи этого значения модуль TWI начнет контролировать состояние шины и сформирует состояние СТАРТ сразу же, как только шина станет свободной. По окончании формирования состояния СТАРТ устанавливается флаг TWINT.

Для переключения модуля в режим «ведущий передатчик» надо передать по шине пакет SLA+W. Для этого содержимое пакета загружается в регистр TWDR, а в регистр TWCR заносится следующее значение:



После передачи адресного пакета и приема бита подтверждения флаг TWINT установится в 1.

После передачи адресного пакета передаются пакеты данных. Данные загружаются в регистр TWDR. Передача начинается после записи в регистр ЕЦСК значения:



После передачи последнего байта данных, ведущий должен сформировать на шине состояние СТОП:

TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
1	x	0	1	x	1	0	x

Разрешить работу модуля TWI

Сформировать состояние СТОП

Сбросить флаг TWINT

3.6.11. *Режим «Ведущий приемник»*

Ведущий принимает данные от ведомого устройства. Формат адресного пакета определяет, в каком из режимов будет работать ведущий.

При передачи пакета SLA+W модуль переходит в режим «ведущий передатчик», а при передаче пакета SLA+К – переходит в режим «Ведущий приемник».

Сначала формируется состояние СТАРТ. Затем передается адресный пакет и байт информации. Полученный от ведомого байт находится в регистре TWDR.

3.6.12. *Режимы «ведомый приемник», «ведомый передатчик»*

В режиме «Ведомый приемник» ведомое устройство принимает данные от ведущего. Перед тем, как переключить модуль в режим «Ведомый приемник», надо занести в регистр TWAR адрес устройства и затем записать в регистр TWCR следующие значения:

TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
0	1	0	0	0	1	0	x

Сбросить флаг TWINT

Разрешить работу модуля TWI

После инициализации модуль будет ожидать адресного пакета с адресом. Значение бита, расположенного после адреса, определяет режим, в который переключится модуль. Если этот бит равен 0, модуль переключится в режим «ведомый приемник», если R/W=1 – в режим «ведомый передатчик».

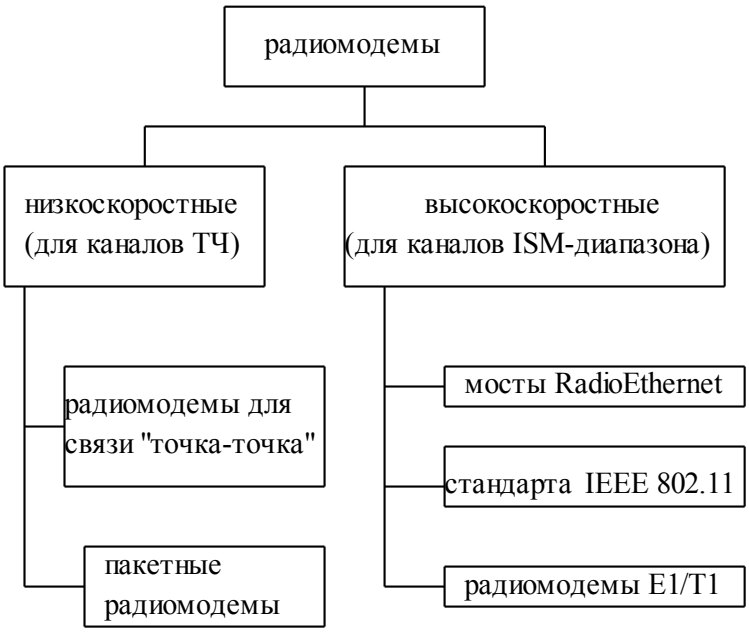
Чтобы прервать поток данных, надо сбросить в 0 разряд TWEA.

4. РАДИОМОДЕМЫ

Передача данных по радиоканалу во многих случаях надежнее и дешевле, чем передача по арендованным каналам или каналам сотовых сетей связи.

Под понятием «радиомодем» можно подразумевать любое устройство, осуществляющее **мо**дуляцию/**дем**одуляцию данных для их передачи по радиоканалу.

Классификация радиомодемов:



Низкоскоростные модемы используются в телефонных каналах тональной частоты. Работают в КВ- и УКВ-диапазонах.

Пакетные радиомодемы ориентированы на работу в едином радиоканале со многими пользователями.

Высокоскоростные радиомодемы ISM-диапазонов работают в нелицензируемых диапазонах, выделенных для промышленного, научного и медицинского оборудования (ISM – **I**ndustrial, **S**cientific and **M**edical bands): 908-928 МГц, 4835 Ггц и 5,725-5,85 Ггц. Возможность свободного использования диапазонов ISM определяет широкую популярность этих модемов.

4.1. Физическая реализация радиомодемов

Типичная станция пакетной связи включает в себя персональный компьютер (ноутбук), собственно радиомодем, приемопередатчик (радиостанцию) УКВ или КВ диапазона (рис. 65).

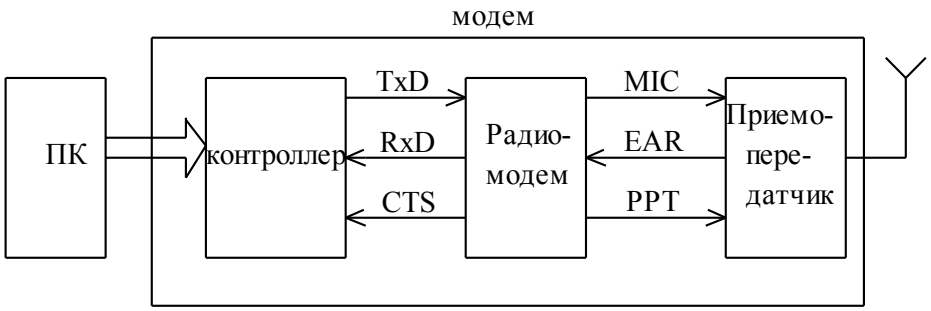


Рис. 65

Компьютер взаимодействует с радиомодемом через последовательный интерфейс RS-232. Передаваемые из ПК в радиомодем данные могут быть или командой , либо информацией, предназначенной для передачи по радиоканалу.

В первом случае команда декодируется и исполняется. Во втором случае – формируется кадр в соответствии с протоколом AX.25. Перед передачей кадра последовательность его битов кодируется линейным кодом без возврата к нулю NRZ_I (**N**on **R**eturn in **Z**ero-**I**nverted). Согласно правилам кодирования NRZ-I перепад физического уровня сигнала происходит в случае, когда в исходной последовательности данных встречается нуль (рис. 66).

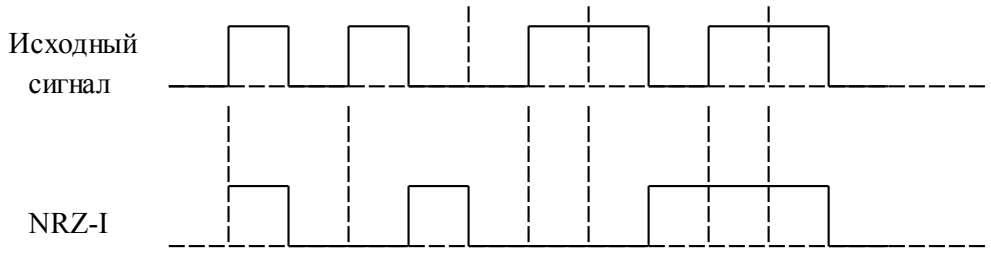


Рис. 66

Блоки модема связаны между собой линиями: TxD – для передачи кадров в коде NRZ-I, RxD – для приема кадров от модема в коде NRZ-I, PPT – для подачи сигнала включения модулятора. CIS – для подачи сигнала занятости канала.

Перед передачей кадра контроллер включает модем сигналом по линии PPT, а по линии TxD посылает кадр в коде NRZ-I. .модем модулирует получаемую последовательность (в соответствии с принятым способом модуляции). Промодулированный сигнал с выхода модулятора поступает на вход MIC передатчика.

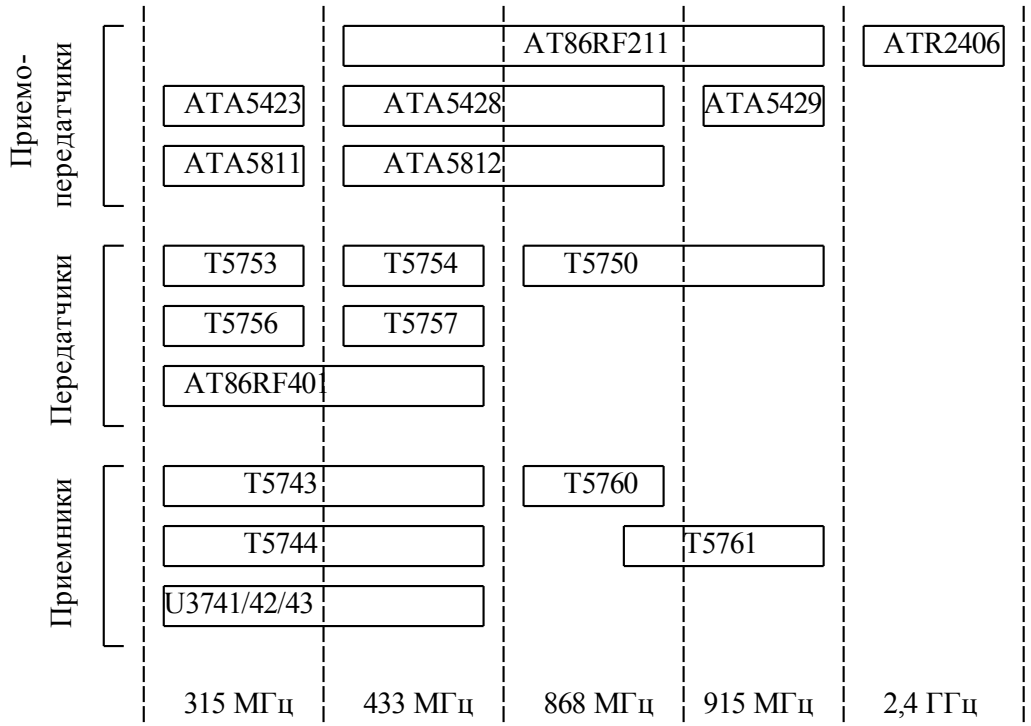
При приеме кадров модулированная последовательностью импульсов несущая частота поступает с выхода EAR приемника радиостанции на вход демодулятора. С демодулятора принятый кадр поступает по линии RxD в контроллер.

Сигнал PPT помимо включения выполняет функцию переключения мощности передачи.

4.2. Беспроводные решения от Atmel

Atmel выпускает микросхемы передатчиков, приемников и приемопередатчиков для диапазонов 315, 345, 433, 868, 915 Мгц и 2,4 ГГц.

Состав семейства:



Эти приемники и передатчики применяются в приложениях, требующих низкой скорости передачи (до 32 кБод): измерения температуры, давления, в датчиках движения, телеметрии, разнообразных счетчиках, бесконтактных ключах, системах сигнализации и безопасности, радиомодемах и т.д. Дальность связи – до 100 м.

Передатчики фирмы ATMEL

Наименование	Встроенный микроконтроллер	Частота передачи, МГц	Ток потребления/sleep, мА/мкА	Скорость передачи, кБод	Напряжение питания, В	Выходная мощность, дБм
1	2	3	4	5	6	7
U2741B	-	300-450	10/0,35	20	2,2-5,5	3
U2745B	-	310-440	10/2	20	2-4	3
T5750	-	868-928	8,5/0,35	32	2-4	5,5
T5753	-	310-330	9/0,35	32	2-4	8
T5754	-	429-439	9/0,35	32	2-4	7,5
AT86RF401	AVR 8 бит	264-456	23/0,1	10	2-5	6
ATAx862-3	Marc 4 бит	310-330	9,5/1	32	2-4	10
ATAx862-4	Marc 4 бит	429-439	9,5/1	32	2-4	10
ATAx862-8	Marc 4 бит	868-928	9,5/1	32	2-4	10

4.2.1. *Передатчик T5750*

Передатчик T5750 работает в паре с приемником T5760/61.

Схема включения передатчика приведена на рис. 67.

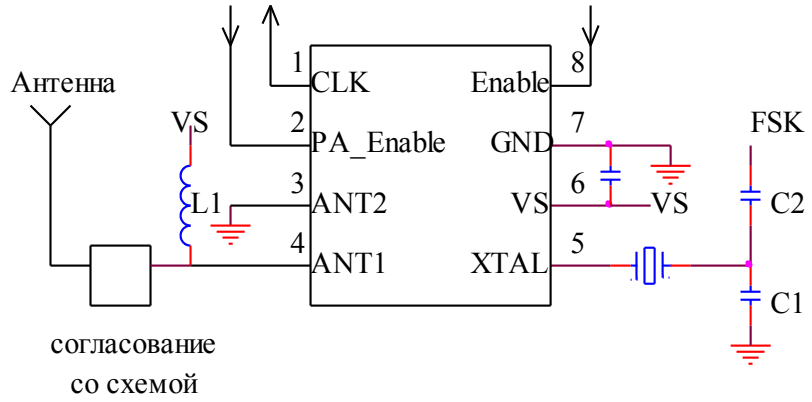


Рис. 67

При отсутствии передачи подачей логического 0 на выводы Enable (вход разрешен) и PA_Enable (вход разрешения усилителя мощности) микросхема переключается в режим sleep, ток потребления снижается до 0,35 мкА.

Передатчик имеет выход CLK для синхронизации внешнего микроконтроллера, который используется для кодирования сигнала. T5750 поддерживает амплитудное и частотное манипулирование, имеет одиночный вывод с открытым коллектором для подсоединения внешней (петлевой или ¼) антенны.

4.2.2. *Амплитудная и частотная манипуляция*

Пусть на вывод Enable передатчика подана логическая 1. Сигнал с вывода CLK подается на внешний микроконтроллер. Сигнал на выводе PA_Enable управляет включением усилителя мощности (0 – выключен, 1 – включен). При этом реализуется амплитудная манипуляция. Временная диаграмма передатчика T5750 в режиме амплитудной манипуляции приведена на рис. 68.

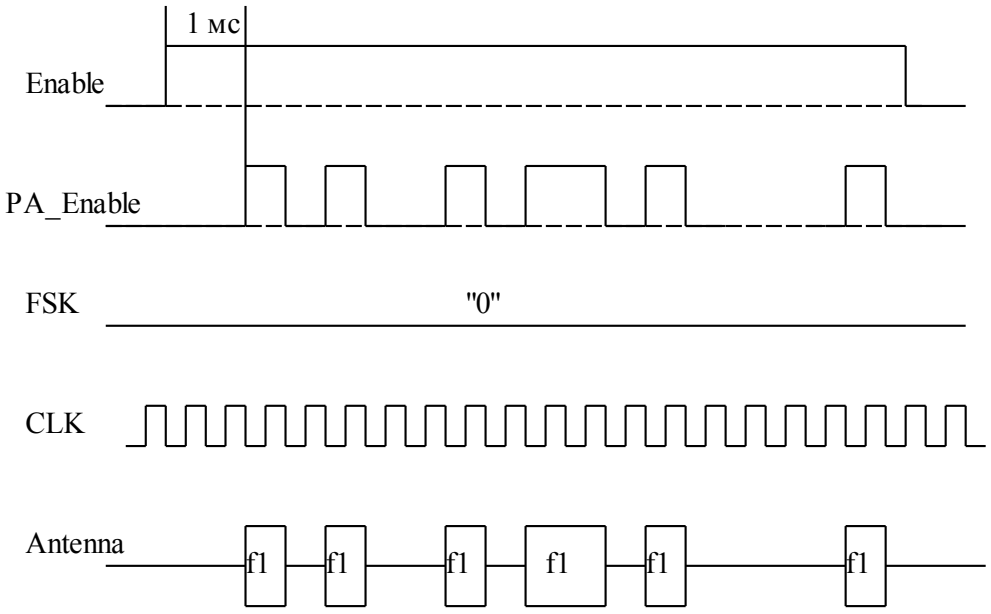


Рис. 68

Для передачи на частоте 868,3 Мгц нужен кварцевый резонатор 13,567187 МГц

Для передачи на частоте 915 Мгц нужен кварцевый резонатор 14,296875 МГц.

Для стабилизации кварцевого генератора и ФАПЧ требуется около 1 мс после подачи «1» на вывод ENABLE. Поэтому вывод PA_ENABLE следует удерживать на «0» более 1 мс.

Для амплитудной манипуляции конденсатор С2 не требуется.

Для частотной манипуляции требуются оба конденсатора С1 и С2. При этом на Enable подается «1». На вывод PA_Enable – «1» через 1 мс (включение усилителя мощности).

Изменение частоты достигается подключением и отключением конденсатора С2 параллельно нагрузочному конденсатору С1 путем подачи на вывод FSK «0» или высокоимпедансного состояния. Частота изменяется от f1 к f2.

Ниже (рис. 69) приводится временная диаграмма передатчика Т5750 в режиме частотной манипуляции.

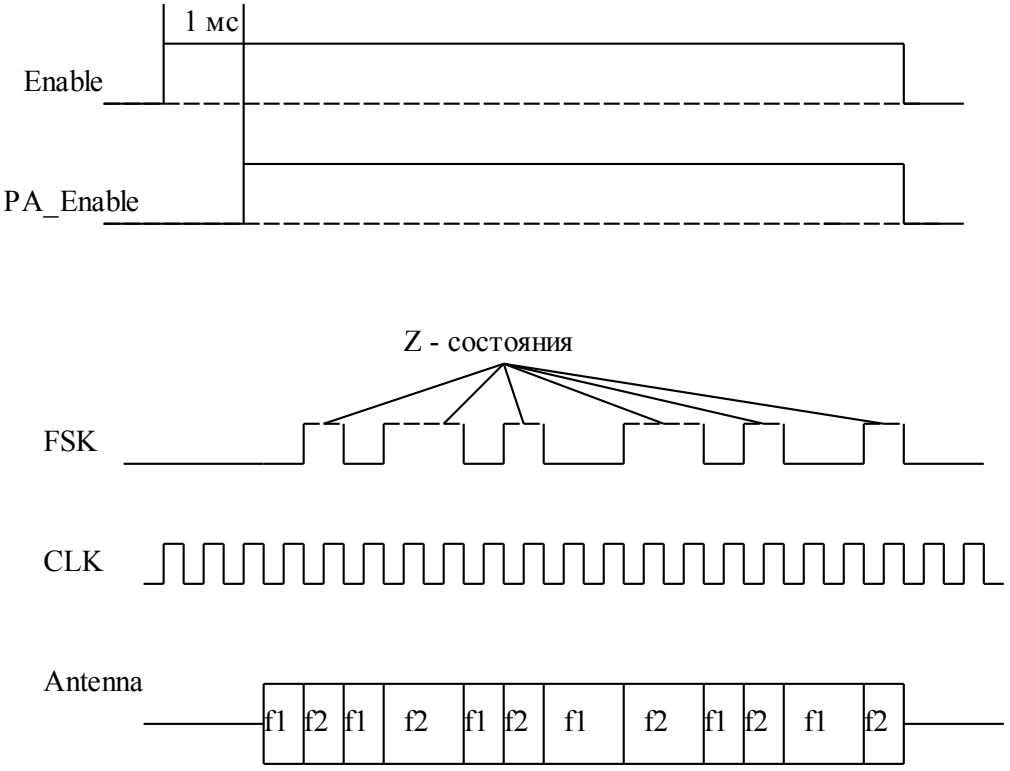


Рис. 69

Усилитель мощности имеет выход с открытым коллектором и обеспечивает ток, не зависящий от нагрузки. Таким образом, выходная мощность определяется подключенной нагрузкой.

Для приложений с небольшой дальностью связи рекомендуется использовать печатную петлевую антенну.

Выходная мощность определяется сопротивлением излучения антенны

$$R_{rad} = 31 кОм \left(\frac{A}{l^2} \right)^2,$$

Где А – внутренняя площадь петлевой антенны, l – длина волны передаваемого сигнала.

Мощность излучения антенны равна

$$P_{rad} = R_{rad} I_{loop}^{\ 2}.$$

4.2.3. *Передатчик AT86RF401*

Это интеллектуальный миниатюрный передатчик на базе 8-битного микроконтроллера AVR. Для организации законченной радиосистемы необходимы: кварцевый резонатор, литиевый элемент питания, три конденсатора, дроссель и петлевая антенна.

.выходная мощность передатчика может изменяться программно в пределах 36 дБ с шагом 1 дБ путем конфигурирования усилителя мощности.

Отработка этого передатчика может производиться в AVR-Studio. По сложности – как ATTiny2313. Добавлены 20 регистров ввода/вывода.

4.2.4. *Передатчики ATAxR862*

Новые передатчики выполнены на базе четырехбитных микроконтроллеров MARC4: T44C862. Они подходят для приема сигналов и передачи. ATAx862 содержат RAM, ROM, 16 линий ввода/вывода, два 8-разрядных таймера/счетчика с функцией модуляции и демодуляции, супервизор напряжения питания, сторожевой таймер, генератор тактовых импульсов с внешним выводом (RC-генератор, кварцевый генератор 32 кГц и 4 МГц).

4.2.5. *Приемники*

Приемники фирмы Atmel поддерживают скорость передачи в режиме АМ до 10 кБод и в режиме ЧМ до 3,2 кБод.

Принципы построения всех приемников одинаков. Приемник находится в режиме sleep. Затем включается на короткий промежуток времени и проверяет присутствие полезного сигнала на входе. Если сигнал обнаружен, приемник переходит в активный режим и передает данные во внешний микроконтроллер.

Приемник конфигурируется с помощью двух регистров разрядностью 12 бит, определяющих скорость передачи данных, интервалы и параметры различных режимов.

4.2.6. *Приемопередатчики*

Отдельно в ряду радиочастотной продукции Atmel стоит приемопередатчик AT86RF-211, предназначенный для работы с внешним микроконтроллером AVR.

Это недорогое решение для беспроводных применений (телеметрия, датчики) для скоростей до 64 кбит/с и полудуплексной передачей. Расстояние передачи составляет до 30 м в закрытом помещении и до 300 м в открытом пространстве.

Наименование	Частота передачи, МГц	Скорость передачи данных, Кбит/с	Выходная мощность, дБм	Ток потребления, мкА/мА	Напряжение питания, В
AT86RF211	915/868/433	64	16/12/14	0,5/35	2,4-3,75

Структура приемника супергетеродинная, двойного преобразования. AT86RF211 включает в себя переключатель режимов прием/передача (Rx/Tx). Имеет 8 уровней выходной мощности. Он поддерживает частотную манипуляцию, манчестерское кодирование не требуется.

Последовательное соединение AT86RF211 с микроконтроллером в режиме передачи приведено на рис. 70.

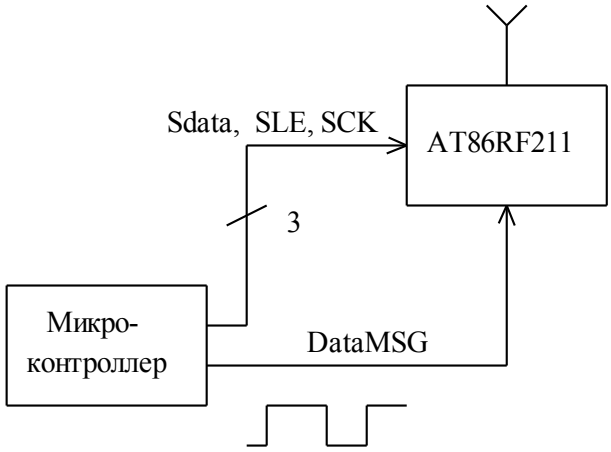


Рис. 70

Микросхема конфигурируется через микроконтроллер. Устанавливаются частота и режим (прием или передача). Далее данные, поступающие на вывод DataMSG немедленно передаются в эфир (Tx); либо сигнал, поступающий с антенны, демодулируется и передается побитно в микроконтроллер по тому же выводу DataMSG (Rx). Данные в кристалле не хранятся и не обрабатываются.

Обмен данными между AVR микроконтроллером и радиочастотным модулем AT86RF211 осуществляется через три регистра ввода/вывода общего назначенияы микроконтроллера, которые используются для чтения и записи данных в 16 внутренних регистров AT86RF211. 8 регистров AT86RF211 используются для конфигурирования нормального режима обмена данными. Размеры регистров варьируются от 32 бит до 1 бита. Режимы чтения и записи позволяют осуществлять доступ к отдельным битам регистра.

Сигналы SLE, SCK и SDATA микроконтроллера формируют последовательный цифровой интерфейс для управления работой приемопередатчика и могут быть назначены на любые выводы общего назначения AVR. Сигнал SLE (en-

able/chip select) и SCK (clock) работают на выход. Сигнал DataMSG используется для организации полудуплексной передачи между микроконтроллером и радиочастотным модулем.

При организации асинхронного интерфейса UART оба вывода RxD и TxD микроконтроллера соединяются с двунаправленным вводом/выводом DataMSG AT86RF211. Таким образом осуществляется полудуплексная передача, при отсутствии передачи данных вывод TxD должен находиться в высокоимпедансном состоянии.

Для организации синхронной передачи можно использовать узел SPI. Обмен данными между AT86RF211 и микроконтроллером по SPI интерфейсу (рис. 71).

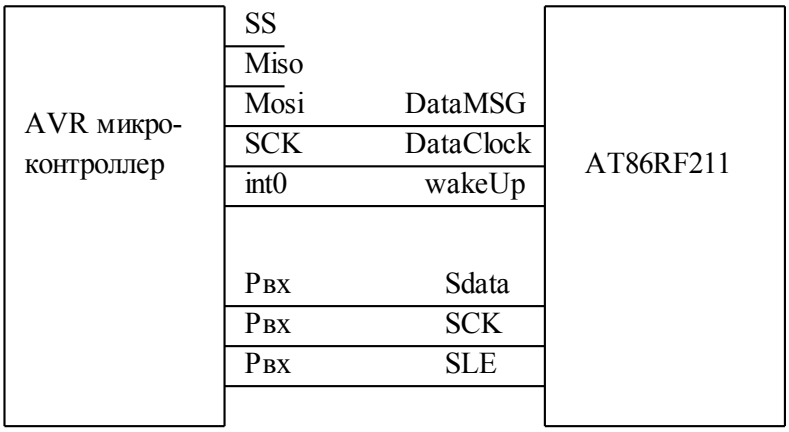


Рис. 71

Если тактовая частота микроконтроллера велика, может оказаться, что скорость SPI окажется слишком большой для AT86RF211 в режиме передачи и AT86RF211 не будет успевать передавать данные в эфир. Скорость порта SPI в режиме приема будет всегда совместима с AT86RF211.