Федеральное агентство железнодорожного транспорта

Дальневосточный Государственный Университет путей сообщения

ИИФО

ГОУ ВПО

Кафедра «Телекоммуникации»

**Курсовой проект**

**По дисциплине: «Микропроцессорные информационно управляющие системы»**

**По теме: «Микроконтроллеры семейства AVR фирмы Atmel»**

г. Хабаровск

2010г.

**Введение**

В классической микропроцессорной системе используются отдельная микросхема процессора, отдельные микросхемы памяти и отдельные порты ввода вывода. Стремительное развитие микропроцессорной техники требует всё большей и большей степени интеграции микросхем.

Именно поэтому были разработаны микросхемы, которые объединяют в себе сразу все элементы микропроцессорной системы. Такие микропроцессоры называются микроконтроллерами. В советское время такие микросхемы называли «Однокристальные микро ЭВМ».

Для однокристальных микроконтроллеров понятие «центральный процессор» обычно не употребляется. Так как процессор – это все-таки отдельное устройство. Функции процессора в микроконтроллере заменяет арифметико-логическое устройство (АЛУ).

Кроме АЛУ, микроконтроллер содержит в своём составе:

♦ тактовый генератор;

♦ память данных;

♦ память программ;

♦ порты ввода-вывода.

Все эти элементы соединены между собой внутренними шинами данных и адреса. С внешним миром микроконтроллер общается при помощи портов ввода-вывода. Любой микроконтроллер всегда имеет один или несколько портов. Кроме того, современные микроконтроллеры всегда имеют встроенную систему прерываний, а также встроенные программируемые таймеры, компараторы, цифроаналоговые преобразователи и многое другое.

Если речь идёт не о большом компьютере, а о портативном устройстве управления, то в нём применяются именно микроконтроллеры. Конечно, любая реальная схема редко обходится без простых логических микросхем, триггеров, счётчиков и тому подобного. Но основой всегда является микроконтроллер. Чистые микропроцессоры в настоящее время применяются только в персональных компьютерах.

**1. Микроконтроллер ATtiny2313/V фирмы Atmel**

**1.1 Структура, основные характеристики и возможности**

Микросхема ATtiny 2313 представляет собой восьмиразрядный микроконтроллер с внутренней программируемой Flash-памятью размером 2 Кбайт.

Общие сведения:

♦ использует AVRRISC архитектуру;

♦AVR–это высокое быстродействие и специальная RISC-архитектура с низким потреблением;

♦ 120 мощных инструкций, большинство из которых выполняется за один машинный цикл;

♦ 32 восьмиразрядных регистра общего назначения;

♦ полностью статическая организация (минимальная частота может быть равна 0);

♦ до 20 миллионов операций в секунду (MIPS/Sec) при тактовой частоте 20 МГц.

Сохранение программ и данных при выключенном питании:

♦ 2 Кбайт встроенной программируемой Flash-памяти, до 10000 циклов записи/стирания;

♦ 128 байт встроенной программируемой энергонезависимой памяти данных (EEPROM);

♦ до 10000 циклов записи/стирания;

♦ 128 байт внутреннего ОЗУ (SRAM);

♦ программируемые биты защиты от чтения и записи программной памяти и EEPROM.

Периферийные устройства:

♦ один 8-разрядный таймер/счётчик с программируемым определителем и режимом совпадения;

♦ один 16-разрядный таймер/счетчик с программируемым определителем, режимом совпадения и режимом захвата;

♦ четыре канала ШИМ (PWM);

♦ встроенный аналоговый компаратор;

♦ программируемый сторожевой таймер и встроенный тактовый генератор;

♦ универсальный последовательный интерфейс USI (Universal Serial Interface);

♦ полнодуплексный USART.

Особенности микроконтроллера:

♦ специальный вход debug WIRE для управления встроенной системой отладки;

♦ внутрисистемный программируемый последовательный интерфейс SPI;

♦ поддержка как внешних, так и внутренних источников прерываний;

♦ три режима низкого потребления (Idle, Power-downи Standby);

♦ встроенная система аппаратного сброса при включении питания;

♦ внутренний перестраиваемый тактовый генератор;

♦ цепи ввода-вывода и корпус;

♦ 18 программируемых линий ввода-вывода;

♦ три вида корпусов;

PDIP– 20 контактов;

SOIC – 20 контактов;

QFN/MLF – 20 контактных площадок.

Напряжения питания:

♦ 1,8 – 5,5 В (для ATtiny2313V);

♦ 2,7 – 5,5 В (для ATtiny2313).

Диапазон частот тактового генератора ATtiny2313V:

♦ 0-4 МГц при напряжении 1,8-5,5 В;

♦ 0–10 МГц при напряжении 2,7-5,5 В.

Диапазон частот тактового генератора ATtiny2313:

♦ 0-10МГц при напряжении 2,7-5,5 В;

♦ 0–20 МГц при напряжении 4,5-5,5 В.

Ток потребления в активном режиме:

♦ 1 МГц, 1,8 В: 230 мкА;

♦ 32 кГц, 1,8 В: 20 мкА (с внутренним генератором).

Ток потребления в режиме низкого потребления:

♦ не более 0,1 мкА при напряжении 1,8 В.

Блок-схема микроконтроллера

Назначение выводов микросхемы ATtiny2313 приведено на рис. 1. Блок-схема микроконтроллера ATtiny2313 приведена на рис. 2.

Ядро AVR имеет большой набор инструкции для работы с 32 регистрами общего назначения. Все 32 регистра непосредственно связаны арифметико-логическим устройством (ALU), которое позволяет выполнять команду для двух разных регистров за один такт системного генератора. Такая архитектура позволила достигнуть производительности в десять раз большей, чем у традиционных микроконтроллеров, построенных по CISC-технологии.

Рис. 1 - Назначения выводов микросхемы ATtiny2313

**1.2 Особенности микросхемы ATtiny2313**

Микросхема ATtiny2313 имеет следующие особенности:

♦ 2 Кбайт системной программируемой Flash-памяти программ;

♦ 128 байт EEPROM;

♦ 128 байт SRAM (ОЗУ);

♦ 18 линий ввода-вывода (I/O);

♦ 32 рабочих регистра;

♦ однопроводной интерфейс для внутрисхемной отладки;

♦ два многофункциональных таймера/счетчика с функцией совпадения;

♦ поддержка внешних и внутренних прерываний;

♦ последовательный программируемый USART-порт;

Рис. 2 - Блок-схема микроконтроллера ATtiny2313

♦ универсальный последовательный интерфейс с детектором начала передачи;

♦ программируемый сторожевой таймер с внутренним генератором;

♦ три программно изменяемых режима энергосбережения.

В режиме Idle происходит приостановка центрального процессора, остальные системы продолжают работать. Выход из этого режима возможен как по внешнему прерыванию, так и по внутреннему. Например, при переполнении таймера.

В режиме PowerDown сохраняется содержимое регистров, но приостанавливается работа внутреннего генератора и отключаются все остальные функции микросхемы. Выход из режима возможен по внешнему прерыванию или после системного сброса. Такое решение позволяет совмещать быстрый старт с низким энергопотреблением.

Микросхема изготовлена с использованием уникальной высокочастотной технологии фирмы Atmel. Внутренняя Flash-память программ может быть перепрограммирована при помощи ISP-интерфейса без извлечения микроконтроллера из платы. Объединение 8-разрядного RISC-процессора внутрисистемной перепрограммируемой Flash-памятью на одном кристалле делают микросхему ATtiny2313 мощным средством, которое обеспечивает очень гибкие и недорогие решения многих прикладных задач управления.

В моей курсовом проекте будет описана некоторая часть микроконтроллера.

**2. Центральное ядро процессора**

Главная функция центрального ядра процессора – управление процессом выполнения программ. Для этого центральный процессор должен иметь непосредственный доступ к памяти, должен быть способен производить все виды вычислений и выполнять запросы на прерывания.

Рассмотрим общие вопросы архитектуры AVR.

**2.1 Кратка характеристика архитектуры**

Чтобы максимально ускорить работу и сделать возможным параллельное выполнение нескольких операций, микроконтроллеры AVR используют Гавардскую архитектуру (рис. 3).

Рис. 3 - Блок-схема архитектуры AVR

Такая архитектура предусматривает отдельную память и отдельную шину адреса как для программы, так и для данных.

Каждая команда из пяти программ выполняются за один машинный цикл с использованием многоуровневой конвейерной обработки. В тот момент, когда очередная команда выполняется, следующая команда считывается из программной памяти. Такая концепция позволяет выполнять по одной команде за один такт системного генератора. Программный сегмент памяти физически представляет собой встроенную перепрограммируемую Flash-память.

Файл регистров быстрого доступа содержит 32 восьмиразрядных регистра общего назначения, доступ к которым осуществляется за один такт системного генератора. Это позволяет арифметико-логическому устройству (ОЛУ) осуществлять большинство своих операций за один такт.

Типичная операция АЛУ выполняется следующим образом: из регистрового файла читаются два операнда, выполняется операция. Результат сохранения опять же в файле регистров. Все эти три действия выполняются за один цикл тактового генератора.

Шесть из этих 32 регистраторов могут использоваться как три 16-разрядных регистра-указателя косвенной адресации. Эти сдвоенные регистры могут использоваться для адресации данных в адресном пространстве ОЗУ. Такая организация даёт возможность программного вычисления адреса.

Один из этих регистров-указателей может также использоваться в качестве указателя адреса данных, размещённых в памяти программ (Flash-памяти). Эти дополнительные составные 16-разрядные регистры именуются X, Yи Z.

АЛУ поддерживает арифметические и логические операции между двумя регистраторами или между константой и регистратором. В АЛУ также могут выполняться операции с отдельными регистраторами. После каждой арифметической операции обновляется регистр статуса для того, чтобы отразить информацию о её результате.

Последовательность выполнения программы может быть изменена командами условного и безусловного перехода, а также командой вызова подпрограммы, в которых используется непосредственная адресация.

Большинство инструкций AVRпредставляет собой одно 16-разрядное слово. Каждый адрес памяти программы содержит 16-битовую инструкцию или половину 32-разрядной инструкции.

При выполнении процедуры обработки прерывания или подпрограммы текущее значение счётчика команд (PC)сохраняется в стеке.

Стек фактически размещён в одном адресном пространстве с памятью данных SRAM (ОЗУ) и, следовательно, размер стека ограничен только размером SRAM и тем, какую часть SRAM использует остальная программа.

Программа пользователя обязательно должна инициализировать указатель стека (SP)сразу после сброса (прежде, чем будет выполнена подпрограмма или будет вызвано прерывание). Указатель стека (SP) имеет свой конкретный адрес в пространстве регистров ввода-вывода. К данным в ОЗУ (SRAM) можно получить доступ, используя память различных способов адресации, поддержанных архитектурой AVR.

Адресное пространство всех видов памяти в архитектуре AVR являются регулярным линейным. Гибкий модуль прерываний имеет ряд регистраторов управления в адресном пространстве регистров ввода-вывода и дополнительный флаг глобального разрешения прерываний в регистре статуса.

Каждый вид прерывания имеет свой отдельный вектор в таблице векторов прерываний. Прерывания имеют приоритет в соответствии с их положением в таблице векторов прерываний. Чем ниже адрес вектора прерывания, тем выше приоритет.

Пространство регистров ввода-вывода содержит 64 адреса для регистров управления периферийными устройствами, регистров управления режимами работы процессора и другими функциями ввода/вывода. К любому регистру ввода-вывода можно получить доступ непосредственно по его номеру или как к ячейке памяти данных. В адресном пространстве памяти данных регистры ввода-вывода располагаются сразу после файла регистров общего назначения (0x20 – 0x5F).

**2.2 АЛУ – арифметико-логическое устройство**

Высокоэффективное АЛУ работает в прямой связи со всеми 32 регистрами общего назначения. За один цикл тактового генератора выполняется арифметическая операция между двумя РОН либо между РОН и непосредственно указанной константой. Все операции АЛУ разделены на три главных категории:

♦ арифметические;

♦ логические;

♦ операции с разрядами.

Особенностью архитектуры является то, что при выполнении арифметических операций все операнды могут рассматриваться как числа без знака, так и числа со знаком.

Регистр статуса

Регистр статуса содержит информацию о результатах выполнения последней арифметической или логической команды. Эта информация может использоваться для того, чтобы изменить процесс выполнения программы в командах условного перехода и других условных операторах.

При вызове процедуры обработки прерывания регистр статуса автоматически не сохраняется. Программист должен обязательно предусмотреть в процедуре обработке прерываний команды, которые будут сохранять содержимое этого регистра в начале и восстанавливать его в конце. Необходимо учесть, что регистр статуса не может быть непосредственно помещён в стековую память. Сначала придётся сохранить в стеке один из регистров общего назначения, затем поместить в освободившийся регистр содержимое регистра статуса, а затем уже из этого промежуточного регистра переместить информацию в стек.

Файл регистров общего назначения

Файл регистров оптимизирован для набора AVRRISC-инструкций. Для того, чтобы достичь требуемой производительности и гибкости, файл регистров поддерживает следующие схемы ввода-вывода:

♦ вывод одного 8-разрядного операнда и ввод одного 8-разрядного результата вычислений;

♦ вывод двух 8-разрядных операндов и ввод одного 8-разрядного результата вычислений;

♦ вывод двух 8-разрядных операндов и ввод одного 16-разрядного результата вычислении;

♦вывод одного 16-разрядного операнда и ввод одного 16-разрядного результата вычислений.

На рис. 4 показана структура 32 регистров общего назначения, используемых в качестве рабочих регистров микроконтроллера.

Рис. 4 - Файл регистров общего назначения микроконтроллеров AVR

Большинство инструкций, оперирующих файлов регистров, имеет прямой доступ ко всем его регистрам, и большинство из них выполняется за один такт.

Как показано на рис. 5, каждому регистру также соответствует адрес в пространстве памяти данных, где они занимают первые 32 ячейки. Хотя физически регистры не входят в SRAM, такая организация памяти обеспечивает большую гибкость при доступе к регистрам. Указатель косвенного доступа к памяти (один из регистров X, Y или Z) может быть установлен на любой регистр из файла.

**2.3 X-регистр, Y-регистр и Z-регистр**

Регистры R26-R31, кроме своего основного назначения, имеют дополнительную функцию. Эти регистры могут служить 16-битными указателями адреса для операций, использующих косвенную адресацию. Три косвенных регистра адреса X, Y, и Zопределены так, как это показано на рис. 5.

В разных командах, использующих косвенную адресацию, эти регистры могут быть использованы как источники постоянного адреса, как адресный регистр с автоматическим приращением адреса и как регистр с автоматическим уменьшением адреса.

Рис. 5 - Сдвоенные регистры X, Y, Z

Указатель стека

Стек, главным образом, используется:

♦ для временного хранения данных;

♦ для хранения локальных переменных;

♦ для хранения адреса выхода из подпрограммы или процедуры обработки прерывания.

Регистр указателя стека всегда указывает на его вершину.

Указатель стека указывает на стековую область в памяти данных (SRAM). В стеке, кроме прочего, сохраняются;

♦ адрес выхода из подпрограммы;

♦ адрес выхода из процедуры обработки прерывания.

Поэтому в любой программе адрес начала стека необходимо определить перед тем, как будет вызвана любая подпрограмма, и перед тем, как будут разрешены прерывания. Первоначально указатель стека должен быть установлен на адрес не ниже 0x60.

Указатель стека уменьшается на единицу, когда данные записываются в стек при помощи команды PUSH, и уменьшается на два, когда в стек записывается в адрес возврата из подпрограммы или процедуры обработки прерывания.

Указатель стека увеличивается на единицу, когда данные читаются из стека при помощи команды POP, и увеличивается на два, когда данные извлекаются из стека при выходе из подпрограммы (команда RET) или завершении процедуры обработки прерывания (команда RETI).

Указатель стека во всех микросхемах AVRвыполнен в виде двух 8-разрядных регистров ввода-вывода. Число фактически используемых разрядов для каждой модели микроконтроллеров разное. В некоторых моделях, в том числе и вATtiny2313, объём памяти данных настолько мал, что для указателя стека используется только младший из регистров указателя стека (SPL). Регистр SPHу них отсутствует. Ниже показана структура регистров указателя стека для микроконтроллера ATtiny2313.

Память ATtiny2313

Теперь рассмотрим различные виды памяти микроконтроллера ATtiny2313. Архитектура AVRсодержит два основных вида памяти:

♦ память данных;

♦ память программ.

Кроме того, микросхема ATtiny2313 имеет EEPROM-память для долговременного хранения данных. Все три вида памяти представляют собой непрерывные области с линейно возрастающими адресами.

**2.4 Системная перепрограммируемая Flash-память программ**

Микросхема ATtiny2313 содержит 2 Кбайт встроенной перепрограммируемой памяти для хранения программ. Так как все команды, используемые в AVR, имеют размер 16 или 32 бита, память программ организованна как 1Rx 16. В графическом виде адресное пространство памяти программ показано на рис. 6.

Рис. 6 - Карта памяти программ

Память программ допускает не менее 10000 циклов записи/стирания. Счётчик команд (PC) микросхемы ATtiny2313 имеет 10 двоичных разрядов и поэтому может работать с адресным пространством в 1К. Микросхема поддерживает режим низкого программирования памяти программ, которое может осуществляться по последовательному SPIинтерфейсу.

В памяти программ можно размещать также любые данные, которые в процессе работы программы остаются неизменными. Для извлечения этих данных из программной памяти используется команда LPM.

**2.5 Память данных SRAM**

На рис. 6 показана организация памяти данных – ОЗУ (SRAM) микроконтроллера ATtiny 2313. Всего адресное пространство ОЗУ содержит 224 ячейки, которые заняты:

♦ файлом регистров общего назначения;

♦ дополнительными регистрами ввода-вывода;

♦ внутренней памятью данных.

Первые 32 ячейки совмещены с файлом РОН. Следующие 64 ячейки – это стандартная область, где располагаются все регистры ввода-вывода. И лишь оставшиеся 128 ячеек составляют внутреннею память данных.

Существует пять разных способов адресации для памяти данных:

♦ прямая;

♦ косвенная со смещением;

♦ косвенная;

♦ косвенная с преддекрементом;

♦ косвенная с постприращением.

Регистры R26-R31 (X, Y, Z) служат указателями адреса при всех операциях с косвенной адресацией.

При прямой адресации указывается полный адрес байта данных.

При косвенной адресации со смещением можно охватить по 63 ячеек, начиная с текущего базового адреса, на который указывает содержимое регистра Y или Z.

При использовании косвенной адресации с автоматическим преддекрементом и автоматическим постдекрементном после чтения (записи) памяти изменяется адрес, записанный в регистр X, Y или Z.

Для всех памяти способов адресации доступны все 32 регистра общего назначения, 64 регистра ввода-вывода и 128 байтов памяти данных SRAM микросхемы ATtiny2313.

Память данных EEPROM

МикросхемаATtiny2313 содержит128 байтов EEPROM-памяти. Она организована как отдельное адресное пространство для хранения данных, в котором каждый байт может быть отдельно прочитан или записан.

Обмен данными между EEPROM и центральным процессором описан ниже и происходит при помощи:

♦ регистра адреса EEPROM;

♦ регистра данных EEPROM;

♦ регистра управления EEPROM.

Процесс чтения/записи EEPROM

Регистры, используемые для доступа к EEPROM, - это обычные регистры ввода-вывода. Наличие функции автоопределения времени готовности позволяет программе пользователя самостоятельно определять тот момент, когда можно записывать следующий байт. Если программа содержит команды, которые производят запись в EEPROM, необходимо предпринять некоторые предосторожности.

В цепях питания, оснащённых хорошим фильтром, напряжение VCC ри включении и выключении будет изменяться медленно. Это заставляет устройство в течении некоторого времени работать при более низком напряжении, чем минимально допустимое напряжение для данной частоты тактового генератора.

**3. Порты ввода-вывода**

Все порты микроконтроллеров AVRв режиме цифрового ввода-вывода представляют собой полноценные двунаправленные порты, у которых каждый из выводов может работать как в режиме ввода, так и в режиме вывода. Это означает, что каждый отдельный разряд порта может быть настроен либо как вход, либо как выход, независимо от настройки всех остальных разрядов того же порта.

Настроить разряды порта можно при помощи команд сброса и установки бита SBIи CBI. То же самое касается изменения значения на выходе (если разряд сконфигурирован как выход) или включения/ отключения внутреннего резистора нагрузки (если разряд сконфигурирован как вход).

Все эти настройки выполняются отдельно для каждого вывода порта. Выходной буфер каждого из выводов порта содержит симметричный выходной каскад с высокой нагрузочной способностью. Нагрузочная способность каждого вывода любого порта достаточна для непосредственного управления светодиодным дисплеем.

Все выводы любого порта имеют индивидуально подключаемые резисторы нагрузки, которые в случае необходимости могут подключаться между этим выводом и источником питания. Входные схемы каждой линии порта имеют по два защитных диода, подключённых к цепи питания и к общему проводу, как это показано на рис. 7.

Рис. 7 - Эквивалентная схема входных цепей одного разряда порта ввода-вывода

Буква “x” в описании имён регистра означает название порта, сточная буква “n” означает номер разряда. При использовании этого имени в программе вместо этих символов нужно подставлять конкретную букву названия порта и конкретный номер разряда.

Например, PORTB3 – для бита номер 3 порта B, если в документации этот бит назывался PORTxn.

Для каждого порта ввода-вывода в микроконтроллере имеется три специальных регистра:

♦ PORTx – регистр данных;

♦ DDRx – регистр управления;

♦ PINx – регистр непосредственного чтения состояния линий порта.

Регистр непосредственного чтения состояния линий порта доступен только для чтения, в то время как регистр данных и регистр управления доступны как для чтения, та и для записи.

Однако тоже возможна запись логической единицы в любой разряд регистра PINx. Она приведёт к переключению соответствующего разряда регистра данных (PORTx). Каждый разряд регистра PORTx управляет включением и отключением резистора внутренней нагрузки, если соответствующий разряд порта находится в режиме ввода.

**3.1 Использование портов для цифрового ввода-вывода**

Каждый разряд порта представляет собой двунаправленную линию ввода-вывода с возможностью подключение внутреннего сопротивления нагрузки. На рис. 8 показана, функциональна схема одной линии порта ввода-вывода. Выходной контакт этой линии обозначен на схеме Pxn.

Рис. 8 - Упрощённая схема одной линии цифрового ввода-вывода

Конфигурация выводов

Каждый разряд порта связан с тремя разрядами трёх специальных регистров: DDxn; PORTxn; PINxn.

Как уже говорилось:

♦ бит DDxn – это разряд номер n регистра DDRx;

♦ бит PORTxn – это разряд номер n регистра PORTx;

♦ бит PINxn – это разряд номер n регистра PINx.

Бит DDxn регистра DDRx выбирает направление передачи информации соответствующего разряда. Если в DDxn записана логическая единица, разряд Pxn работает как выход. Если в DDxn записан логический ноль, разряд Pxn работает как вход.

Если разряд порта сконфигурирован как вход, установка бита PORTxn в единицу включает внутренний резистор нагрузки. Для отключения резистора нагрузки нужно в PORTxn логический ноль. Сразу после системного сброса все выводы всех портов переходят в третье (высокоимпендансное) состояние.

Если разряд порта сконфигурирован как выход (установка бита PORTxn в единицу), то эта единица появится на выходе порта. Если в разряд PORTxn записан логический ноль, то и на выходе будет ноль.

**4. Внешние прерывания**

**4.1 Назначение и режимы работы**

Для вызова внешних прерываний используются входы INT0, INT1 или любой из входов PCINT7-0. Если прерывания разрешены, то они будут вызваны, даже если выводы INT0, INT1 и PCINT7-0 сконфигурированы как выходы.

Эта особенность обеспечивает возможность генерировать прерывания программным путём. Прерывание PCI1 – это прерывание по изменению сигнала на любом из выводов PCINT7-8. Прерывание PCI0 – это прерывание по изменению любого сигнала PCINT7-0. Причём как для прерыванияPCI1, так и для прерывания PCI0 можно произвольно исключить из работы любой из закреплённых за ним входов.

Управляющие регистры PCMSK1 и PCMSK0 определяют, какие из входов будут вызывать соответствующие прерывание. Прерывание по изменению на контактах PCINT7-0 работает асинхронным образом. Поэтому данный вид прерываний может использоваться для пробуждения изо всех спящих режимов, кроме режима Idle.

Прерывания INT0 и INT1 поддерживают несколько режимов. Они могут быть вызваны по переднему фронту, по заднему фронту или по статическому сигналу низкого логического уровня. Выбор одного из этих режимов производится при помощи регистра управления внешними прерываниями – EICRA.

Когда одно из прерываний INT0 или INT1 разрешено и сконфигурировано как прерывание по низкому входному уровню, запрос на прерывание будет вырабатываться всё время, пока на входе присутствует низкий уровень.

Вызов прерываний INT0 и INT1 по низкому уровню осуществляется в асинхронном режиме. Это означает, что такое прерывание может быть использовано для пробуждения из любого спящего режима за исключением Idle.

Время запуска микроконтроллера определяется при помощи fuse-переключателей SUT и CKSEL.

**5. Восьмиразрядный таймер/счётчик с поддержкой режима ШИМ**

**5.1 Назначение и особенности**

Таймер/счётчик T0 – это универсальный восьмиразрядный счётный модуль с двумя независимыми модулями совпадения и с поддержкой ШИМ (PWM). Он позволяет формировать заданные промежутки времени (для работы в режиме реального времени). А также может служить генератором периодических сигналов.

Рассмотрим основные характеристики таймера/счётчика.

**5.2 Упрощенная блок-схема**

Упрощенная блок-схема восьмиразрядного таймера/счётчика приведена на рис. 9. Доступные для центрального ядра регистры и цепи передачи данных на схеме показаны полужирными линиями.

Рис. 9 - Блок-схема 8-разрядного таймера/счётчика

**5.3 Регистры**

Как счётный регистр таймера/счётчика (TCNT0), так и оба регистра совпадения (OCR0A, и OCR0B) представляют собой восьмиразрядные регистры. Наличие запроса на прерывание (на рис.10 он сокращённо обозначен «выз. прерыв.») всегда можно определить по состоянию соответствующее флага прерываний в регистре TIFR.

Каждое прерывание может быть индивидуально замаскировано при помощи регистра маски прерываний таймера TIMSK. Регистры TIFR и TIMSK на рис. 9 не показаны.

Таймер/счётчик может работать как от внутреннего тактового генератора через предварительный делитель, так и от внешнего тактового сигнала, поступающего на вход T0. Схема выбора источника тактового сигнала пропускает тактовые импульсы выбранного источника на вход таймера/счётчика, и каждый импульс этого сигнала увеличивает (или уменьшает) его значение.

Если не выбран ни один из источников тактового сигнала, таймер/счётчик останавливается.

Сигнал на выходе схемы выбора источника тактового сигнала (clkT0) называется сигналом синхронизации таймера.

Содержимое регистров OCR0Aи OCR0Bпостоянно сравнивается со значением таймера/счётчика. Результат сравнения может использоваться генератором для генерации сигнала ШИМ или прямоугольных импульсов переменной частоты на одном из выходов OC0Aили OC0B.

В момент совпадения в одном из каналов устанавливается соответствующий флаг OCF0Aили OCF0B, который может использоваться для генерации запроса на прерывание по совпадению.

**5.4 Используемые обозначения**

При описании счётчиков используются специальные обозначения для всех его важных состояний. Эти обозначения приведены в табл. 1.

Обозначения для основных состояний 8-разрядного счётчика

Таблица 1

|  |  |
| --- | --- |
| BOTTOM | Счётчик достигает значения BOTTOM (начало), когда его содержимое равно 0x00 |
| MAX | Счётчик достигает значения MAX (максимум), когда его содержимое равно 0xFF (десятичное 255) |
| TOP | Счётчик достигает значения TOP(вершина), когда его содержимое достигает самого высокого значению в данном режиме работы. В зависимости от режима значение TOP может быть равно либо 0xFF (MAX), либо значению, записанному в регистре OCR0A (режим сброса по совпадению) |

**5.5 Модуль счёта**

Основой восьмиразрядного таймера/счётчика является программируемый реверсивный счётный модуль. На рис. 10 показана блок-схема счётного модуля и его управляющие сигналы.

Рис. 10 - Блок-схема счётного модуля

Рассмотрим внутренние сигналы:

♦ count – увеличивает или уменьшает содержимое TCNT0 на 1;

♦ direction – выбор между уменьшением и увеличением;

♦ clear – очистка TCNT0 (установка всех битов в ноль);

♦ clkT0 – тактовая частота таймера/счётчика;

♦ top – возникает при достижении TCNT0 максимального значения;

♦ bottom – возникает при достижении TCNT0 минимального значения (нуля).

В зависимости от режима работы таймера, каждый импульс тактового сигнала (clkT0) очищает, увеличивает или уменьшает значение счётчика. Сигнал clkT0может быть получен как от внешнего, так и от внутреннего источника тактового сигнала. Это определяется битами выбора тактового сигнала (CS02:0). Когда не выбран ни один источник тактового сигнала (CS02:0 = 0), таймер останавливается.

Центральный процессор может обращаться к значению регистра TCNT0 независимо от того, присутствует ли сигнал clkT0 или нет. Команда записи, поступающая от центрального процессора, имеет приоритет над всеми другими операциями (очистки счётчика или операциями счёта).

Режимы работы таймера определяются установкой битов WGM01 и WGM00 регистра TCCR0Aи битом WGM02 регистра TCCR0B. Есть тесная связь между выбранным режимом работы счётчика и частотой сигнала на выходе OC0A.

Флаг переполнения таймера/счётчика (TOV0) устанавливается в соответствии с режимом работы, выбранным при помощи битов WGM01:0. Флаг TOV0 может использоваться для генерации прерываний центрального процессора.

**5.6 Модуль совпадения**

Основа модуля – восьмиразрядный компаратор, который непрерывно сравнивает содержимое регистра TCNT0 с содержимым каждого из двух регистров совпадения (OCR0Aили OCR0B). Каждый раз, когда содержимое TCNT0 оказывается равным содержимому OCR0Aили OCR0B, компаратор вырабатывает сигнал совпадения. Этот сигнал устанавливает соответствующий флаг совпадения (OCF0Aили OCF0B) в следующем тактовом цикле.

Если соответствующее прерывание разрешено, установка флага совпадения вызывает прерывание. Флаг совпадения автоматически сбрасывается при запуске процедуры обработки прерывания. Флаг также может быть очищен программно путём записи в него логической единицы.

В режиме генератора частот сигнал совпадения используется для генерации выводного сигнала совпадения используется для генерации выводного сигнала в соответствии с выбранным режимом работы, который определяется битами WGM02:0, а также битами выбора режима сравнения (COM0x 1:0). Сигналы max и bottom используются генератором частот в некоторых случаях для получения критических значений в отдельных режимах работы. На рис. 11 показана блок-схема модуля совпадения. На рисунке буква x – это условное обозначение. Для разных модулей совпадения x равно либо A либо B.

Рис. 11 - Блок-схема модуля совпадения

Регистры OCR0xимеют двойную буферизацию в любом режиме широтно-импульсной модуляции (ШИМ). В режиме Normal и режиме СТС (Сброс при совпадении) двойная буферизация отключается. Двойная буферизация синхронизирует момент обновления регистра OCR0xс моментом достижения таймером верхнего или нижнего пределов. Синхронизация предотвращает возникновение ассиметричных ШИМ-импульсов, то есть импульсов, длина которых равна нечётному количеству тактов. Таким образом обеспечивается высокое качество сигналов ШИМ.

Доступ к регистру OCR0xможет показаться слишком сложным. На самом деле это не так. Если двойная буферизация разрешена, центральный процессор обращается к регистрам OCR0xчерез буфер. Если буферизация отключена, центральный процессор обращается к регистрам OCR0xнепосредственно.

**5.7 Принудительное изменение состояния выхода совпадения**

Во всех не-ШИМ-режимах таймера сигнал на любом из выходов совпадения может быть изменён принудительно путём записи единицы в специальный бит FOC0x. Принудительное изменение выхода совпадения не устанавливает флаг OCF0xи не перезагружает таймер.

Сигнал на выходе OC0xбудет изменяться таким же образом, как при реальном совпадении. То есть поведение выхода OC0xбудет зависеть от установки битов COM0x1:0. В зависимости от значения этих битов сигнал на выходе будет либо установлен в единицу, либо сброшен в ноль, либо изменит своё значение на противоположное.

**5.8 Блокировка режима совпадения в момент записи регистра TCNT0**

При записи значения в регистр TCNT0 операция сравнения блокируется в течение одного такта входного сигнала таймера. Это происходит даже в том случае, если таймер остановлен. Эта особенность позволяет записывать в регистр OCR0x то же самое значение, что и в регистр TCNT0, не вызывая прерывания при поступлении на вход таймера/счётчика тактового сигнала.

**5.9 Использование модуля прерывания**

Как уже говорилось, в любом режиме работы таймера в момент записи регистра TCNT0 работа модуля сравнения приостанавливается на один период тактового сигнала. Это может привести к ошибкам при изменении содержимого регистра TCNT0 независимо от того, запущен таймер/счётчик или нет.

Если значение, записанное в TCNT0, равно значению, записанному в OCR0x, операция сравнения будет пропущена, что приведёт к неправильной работе таймера в режиме генератора частоты.

По той же причине нельзя записывать в TCNT0 значение равное BOTTOM, когда счётчик работает в режиме обратного счёта.

Настройка режимов работы выхода OC0xдолжна быть произведена перед тем, как соответствующая линия порта будет сконфигурирована как выход. Самый простой способ установить нужное значение на выходе OC0x – использовать принудительную установку (бит FOC0x) в режиме Normal. Регистры OC0x сохраняю своё значение при переключении режимов генерации сигналов.

**5.10 Модуль вывода сигнала совпадения**

Разряды COM0x1:0 выполняют две функции. Генератор частот использует биты COM0x1:0 для того, чтобы определить, как изменится сигнал на выходе модуля совпадения (OC0x) в момент обнаружения факта совпадения. В то же время биты COM0x1:0 управляют источником сигнала на выходе OC0x.

На рис. 12 показана упрощённая схема, демонстрирующая логику работы разрядов COM0x1:0.

Рис. 12 - Схема вывода сигнала совпадения

Как видно из рисунка, значение COM0x1:0 влияет на состояние порта ввода вывода микросхемы, не зависимо от главных регистров управления этим портом (DDR и PORT). Причём когда мы говорим о статусе OC0x, нужно понимать, что внутренний регистр OC0x не то же самое, что контакт микросхемы OC0x. Сразу после системного сброса в регистр OC0x записывается ноль.

Если любой из битов COM0x1:0установлен, то основная функция порта ввода-вывода отменяется, и на выход проходит сигнал совпадения (OC0x) с генератора частот. При этом, направление передачи информации контакта OC0x (вход он или выход) всё ещё зависит от соответствующего бита регистра DDR.

Значение бита, определяющего направление передачи информации для вывода OC0x в случае, если он должен работать как выход, должно быть установлено до того, как значение регистра OC0x поступит на этот выход. Альтернативные функции порта не зависят от режима работы генератора сигналов.

**5.11 Режимы работы**

Режим работы, то есть поведение таймера/счётчика и выхода сигнала совпадения, определяется как режимом работы генератора сигналов (WGM02:0), так и режимом вывода сигнала совпадения (COM0x1:0). Состояние битов, определяющих режим вывода сигнала совпадения, не влияет на последовательность подсчёта, которая определяется только состоянием битов конфигурации генератора сигналов.

Биты COM0x1:0 определяют, должен ли выходной сигнал ШИМ быть инвертирован или нет (инвертированный или не инвертированный ШИМ). Для не-ШИМ-режимов содержимое битов COM0x1:0 определяет, должен ли сигнал на выходе быть установлен в единицу, сброшен в ноль либо переключён в противоположное состояние в момент совпадения.

**5.12 Режим «Normal»**

Режим «Normal» (WGM02:0 = 0) – это самый простой из режимов работы таймера. В этом режиме направление счёта всегда вперёд (содержимое увеличивается), и принудительный сброс счётчика не выполняется. Счётчик просто переполняется, когда достигает максимально для восьми разрядов значения (TOP = 0xFF), а затем перезапускается сначала (0x00).

При нормальной работе флаг переполнения таймера/счётчика (TOV0) будет установлен в тот момент, когда TCNT0 станет равно нулю. Флаг TOV0 в этом случае ведёт себя как девятый бит, а затем исключением, что он только устанавливается, но не сбрасывается.

Используя прерывание по переполнению таймера, которое автоматически очищает флаг TOV0, можно увеличить коэффициент пересчёта программы путём. Режим Normal не имеет никаких особенностей, на которых стоило бы заострять внимание. Новое значение счётного регистра может быть записано в любой момент времени.

Модуль совпадения иногда может использоваться для вызова прерываний. Использование сигнала совпадения для генерации сигналов в режиме Normal не рекомендуется, так как это будет сильно тормозить работу процессора.

**5.13 Режим сброса при совпадении (СТС)**

В режиме сброса при совпадении или, по-другому, в режиме СТС (при WGM02:0 = 2) регистр OCR0A используется для того, чтобы управлять коэффициентом пересчёта счётчика. В режиме СТС счётчик сбрасывается в ноль при совпадении содержимого счётчика регистра (TCNT0) и регистра OCR0A.

Регистр OCR0A, таким образом, определяет максимальное значение для счётчика, а, следовательно, и его коэффициент пересчёта.

Этот режим позволяет максимально контролировать частоту сигнала на выходе модуля. Упрощается также и подсчёт внешних событий. Значение счётчика регистра (TCNT0) увеличивается до момента, пока не происходит совпадение между TCNT0 и OCR0A, и затем содержимое счётчика (TCNT0) очищается.

Прерывание может вызываться каждый раз, когда счётчик достигает значения TOP. При этом используется флагOCF0A. Если прерывание разрешено, вызывается процедура обработки прерывания, которая может использоваться для того, чтобы обновить значение TOP.

Установка TOP чересчур близко к значению BOTTOM в тот момент, когда счётчик не работает или при низком коэффициенте предварительного деления, должна выполняться осторожно, так как режим СТС не имеет двойной буферизации.

Если новое значение, записанное в OCR0A, будет ниже, чем текущее значение TCNT0, то источник пропустит момент совпадения. В результате счётчик продолжит счёт до своего максимального значения (0xFF), затем перейдёт через ноль и лишь затем произойдёт момент совпадения.

Для генерации выходного сигнала в режиме СТС выход OC0Aможет быть установлен в режим переключения выходного уровня каждый раз в момент совпадения. Для этого нужно установить в соответствующее положение биты режима вывода сигнала совпадения (COM0A1:0 = 1). Значение регистра OC0A не поступит на соответствующий внешний контакт порта, если он не сконфигурирован как выход. Сгенерированный сигнал будет иметь максимальную частоту fOC0 = fclk\_I/O/2, когда в регистр OCR0A записан ноль (0x00). Частота сигнала может быть рассчитана при помощи следующего уравнения:

где переменная N – это коэффициент предварительного деления (1, 8, 64, 256 или 1024).

Как и в режиме Normal, флаг TOV0 устанавливается каждый раз, когда счётчик досчитывает до MAX и переходит в ноль.

**5.14 Режим FastPWM (быстрый ШИМ)**

Микроконтроллер имеет несколько режимов широтно-импульсной модуляции (ШИМ). По-английски это звучит как Pulse Width Modulation (PWM).Быстрый ШИМ (fastPWM) выбирается при WGM02:0 = 3 или 7. В этом режиме формируется самый высокочастотный сигнал ШИМ. Быстрый ШИМ отличается от других режимов ШИМ тем, что для формирования сигнала счётчик формирует только возрастную последовательность. То есть изменение значение счётчика имеет вид пилообразного сигнала с односторонним наклоном.

Счёт начинается со значения BOTTOM и заканчивается значением TOP. После этого счётчик перезапускается (снова устанавливается значение BOTTOM). Значение TOP равно 0xFF при WGM2:0 = 3.

Если же WGM2:0 = 7, значение TOP определяется содержимым регистра OCR0A. В режиме неинвертирующего выхода сигнал совпадения (OC0x) сбрасывается в момент совпадения значений TCNT0 иOCR0xи перехода к BOTTOM.

В инвертирующем режиме сигнал на выходе устанавливается в момент совпадения и перехода в BOTTOM. Благодаря тому, что счётчик работает всегда только в одном направлении, частота сигнала в режиме fast PWM может быть в два раза выше, чем в режиме phasecorrect PWM, который использует пилообразный сигнал с двумя наклонами.

Благодаря высокой частоте выходного сигнала режим fast PWM хорошо подходит для создания систем регулировки мощности, для построения выпрямителей и цифро-аналоговых преобразователей. Высокая частота позволяет применять внешние компоненты (катушки, конденсаторы) небольших размеров, и тем самым, уменьшать общую стоимость системы.

В режиме fast PWM значение счётчика увеличивается до тех пор, пока не достигнет значения TOP. В следующем цикле тактового сигнала таймера счётчик очищается.

Флаг переполнения таймера/счётчика (TOV0) устанавливается каждый раз, когда счётчик достигает значения TOP. Если прерывание разрешено, то вызывается процедура обработки прерывания, которая может быть использована для обновления уровня совпадения.

В режиме fast PWM модуль совпадения используется для генерации сигнала ШИМ на выходах OC0x. Установка битов COM0x1:0 = 2 произведёт к генерации на выходе неинвертированного сигнала ШИМ. Для генерации инвертированного сигнала ШИМ необходимо установить COM0x1:0 = 3. При установке битов COM0A1:0 = 1сигнал на выходе AC0A в момент совпадения переключается в противоположное состояние при условии, что бит WGM02 = 1.

Эта опция не доступна для выхода OC0B. Фактическое значение OC0x поступит на внешний контакт микросхемы только в том случае, если он будет сконфигурирован как выход.

Сигнал ШИМ формируется путём установки (сброса) регистра OC0x в момент совпадения значений OCR0xи TCNT0, и сброса (установки) этого регистра в первом тактовом цикле, после перезагрузки счётчика (изменении его значения с TOP на BOTTOM). Частота сигнала ШИМ на выходе может быть рассчитана при помощи следующего выражения:

.

Переменная N представляет собой коэффициент предварительного деления (1, 8, 64, 256 или 1024). Отдельно нужно рассмотреть несколько случаев при генерации сигнала ШИМ, когда в регистр OCR0Aзаписывается значение, близкое к предельному.

Если в регистре OCR0Aбудет установлено значение, равное BOTTOM, то выходной сигнал будет представлять собой короткий выброс для каждого MAX + 1тактового импульса таймера.

Если в регистр OCR0Aзаписать значение MAX, то это приведёт к тому, что на выходе будет постоянно присутствовать либо высокий, либо низкий логический уровень (в зависимости от значения битов COM0A1:0).

Частота выходного сигнала в режиме fast PWM (при уровне регулирования 50%) может быть достигнута, если заставить регистр OC0x переключать свой логический уровень при каждом совпадении (COM0x1:0 = 1).

Сформированный таким образом сигнал будет иметь максимальную частоту fOC0 = fclk\_I/O/2 в том случае, когда в регистре OCR0A записан ноль. Эта особенность позволяет переключать регистр OCR0A таким же образом, как в режиме СТС, но при этом использовать все преимущества двойной буферизации, которая применяется в режиме fast PWM.

**5.15 ШИМ, корректный по фазе (PhaseCorrectPWM)**

Режим phase correct PWM (WGM02:0 = 1или 5). Формирование сигнала ШИМ происходит с большим коэффициентом пересчёта и корректного по фазе. Корректность по фазе обеспечивается благодаря работе счётчика в режиме пилообразного сигнала с двухсторонним наклоном.

Счётчик периодически изменяет направление своего счёта. Сначала он считает от BOTTOM до TOP, затем направление счёта меняется, и счётчик считает TOP до BOTTOM. Затем направление пересчёта снова меняется, и всё повторяется сначала.

Значение TOP равно 0xFF при WGM2:0 = 1 и определяется регистром OCR0Aпри WGM2:0 = 5.

В режиме неинвертирующего вывода сигнал на выходе OCR0x сбрасывается в ноль в момент совпадения содержимого регистров TCNT0 и OCR0x, если счётчик работает в прямом направлении (на увеличение). Значение устанавливается в единицу в момент совпадения, если счётчик работает на уменьшение.

В режиме инвертированного вывода картина меняется на противоположную. Режим двухстороннего наклона характеризуется более низкой максимальной частотой выходного сигнала по сравнению с предыдущим случаем, где применяется пила с односторонним наклоном. Благодаря симметричности по фазе при двустороннем наклоне такие режимы предпочитаются при создании систем управления электродвигателями.

В режиме phase correct PWM значение счётчика увеличивается, пока не достигнет значения TOP. Когда значение счётчика достигает TOP, направление счёта изменяется. Содержимое TCNT0 будет равно TOP в течение одного периода тактового сигнала таймера. Флаг переполнения таймера/счётчика (TOV0) устанавливается каждый раз, когда счётчик достигает значения BOTTOM. Флаг прерывания может использоваться для генерации запроса на прерывание. Такое прерывание будет вызвано каждый раз, когда содержимое счётчика достигнет значения BOTTOM.

В режиме phasecorrect PWM модуль совпадения используется для генерации сигнала ШИМ на выходе OC0x. При установке битов COM0x1:0 = 3. Установка битов COM0A0 = 1заставляет сигнал на выходе OC0A инвертироваться каждый раз в момент совпадения, если бит WGM02 установлен. Эта опция не доступна для вывода OC0B. Фактическое значение OC0x поступает на внешний вывод порта только в том случае, если он сконфигурирован как выход.

Сигнал ШИМ сгенерируется путём сброса (установки) OC0xв момент совпадения содержимого регистров OCR0xи TCNT0, когда счётчик работает на увеличение, и устанавливается (сбрасывается) в момент совпадения, если счётчик работает на уменьшение. Частота выходного сигнала ШИМ в режиме phase correct PWM может быть вычислена по следующей формуле:

,

Где переменная N представляет собой коэффициент предварительного деления (1, 8, 64, 256, 1024).

Крайние значения содержимого регистра OCR0A при генерации сигналов ШИМ в режиме phase correct PWM представляют собой специальные случаи. Для неинвертирующего режима при записи в регистр OCR0Aзначения BOTTOM на выходе установиться низкий логический уровень. При записи в OCR0Aзначения MAX на выходе установится логическая единица. Для инвертирующего режима сигнал на выходе будет иметь противоположные значения.

**6. Описание системы команды**

**6.1 Система команд. Общие сведения о системе команд**

В семействе AVR система команд у микроконтроллеров разных типов содержат от 89 до 130 команд. У микроконтроллеров типа 2323, 2343, 2313, 4433, 8515 и 8535 в систему команд входят 118 команд. Эту систему команд будем называть базой.

В табл. 2.1 – 2.13 описаны операции, выполняемые по командам базовой системы, и приведены мнемокоды команд, используемые при разработке программы на языке AVR Ассемблера. В таблицах дана сквозная нумерация команд, используемая далее в тексте.

Базовая система команд содержит:

▪ 33 команды регистровых операций, при выполнении которых используются только регистры общего назначения (команды №№ 1-33);

▪ 26 команд с обращением по адресу в адресном пространстве SRAM (команды №№ 34-59);

▪ 2 команды с обращением к регистрам ввода-вывода (команды №№ 60 и 61);

▪ 1 команда с обращением к FlashROM (команда № 62);

▪ 22 команды операций с битами в разрядах регистров общего назначения и регистров ввода-вывода (команды №№ 63-84);

▪ 34 команды управления ходом программы (команды №№ 85-118).

В систему команд микроконтроллеров типа t11, t12, t15, 1200, и t28, у которых нет SRAM, не входят команды с обращением по адресу в адресном пространстве SRAM за исключением команд с мнемокодами LDRd, Z (№ 40) и STZ, Rr (№ 41), по которым производится обращение к регистрам общего назначения и регистрам ввода-вывода с использованием косвенной адресации. В систему команд у этих микроконтроллеров не входят также 2 команды регистровых операций (№№ 32 и 33) и две команды управления ходом программы (№№ 86 и 88). У микроконтроллеров типа 1200 в систему команд не входит также команда с обращением к FlashROM, а у микроконтроллера типа m163, кроме того, с наличием аппаратного умножителя.

Таблица 2

|  |  |  |  |
| --- | --- | --- | --- |
| № | Операция | Мнемокод команды | Признаки результата |
| H | S | V | N | Z | C |
| 1 | $00→Rd ((Rd) (Rd) →Rd) | CKR Rd |  | 0 | 0 | 0 | 1 |  |
| 2 | (Rd) → Rd((Rd)^( Rd) → Rd) | TST Rd |  | + | 0 | + | + |  |
| 3 | (Rd) → Rd($FF-( Rd) → Rd) | COM Rd |  | + | 0 | + | + | 1 |
| 4 | - (Rd) → Rd($00-( Rd) → Rd) | NEG Rd | + | + | + | + | + | + |
| 5 | (Rd)+1→ Rd | INC Rd |  | + | + | + | + |  |
| 6 | (Rd)-1→ Rd | DEC Rd |  | + | + | + | + |  |
| 7 | 0→( Rd) →C | LSR Rd |  | + | + | 0 | + | + |
| 8 | C←( Rd) ←0 | LSL Rd | + | + | + | + | + | + |
| 9 | C→( Rd) →C | ROR Rd |  | + | + | + | + | + |
| 10 | C←( Rd) ←C | ROL Rd | + | + | + | + | + | + |
| 11 | Rd.7→( Rd.6- Rd.0) →C | ASR Rd |  | + | + | + | + | + |
| 12 | Rd.4-7↔ Rd.0-3 | SWAP Rd |  |  |  |  |  |  |
| 13 | (Rr) → Rd | MOV Rd, Rr |  |  |  |  |  |  |
| 14 | (Rd)+(Rr) → Rd | ADD Rd, Rr | + | + | + | + | + | + |
| 15 | (Rd)+(Rr)+C→ Rd | ADC Rd, Rr | + | + | + | + | + | + |
| 16 | (Rd)-(Rr) → Rd | SUB Rd, Rr | + | + | + | + | + | + |
| 17 | (Rd)-(Rr)-C→ Rd | SBC Rd, Rr | + | + | + | + | + | + |
| 18 | (Rd)-(Rr) | CP Rd, Rr | + | + | + | + | + | + |
| 19 | (Rd)-(Rr)-C | CPC Rd, Rr | + | + | + | + | + | + |
| 20 | (Rd) ^(Rr) → Rd | AND Rd, Rr |  | + | 0 | + | + | + |
| 21 | (Rd)˅(Rr) → Rd | OR Rd, Rr |  | + | 0 | + | + |  |
| 22 | (Rd) (Rr) → Rd | EOR Rd, Rr |  | + | 0 | + | + |  |
| d=0-31; r=0-31 |

Таблица 3

|  |  |  |  |
| --- | --- | --- | --- |
| № | Операция | Мнемокод команды | Признаки результата |
| H | S | V | N | Z | C |
| 23 | $FF→ Rd | SER Rd |  |  |  |  |  |  |
| 24 | K→ Rd | LDI Rd, K |  |  |  |  |  |  |
| 25 | (Rd)-K→ Rd | SUBI Rd, K | + | + | + | + | + | + |
| 26 | (Rd)-K-C→ Rd | SBCI Rd, K | + | + | + | + | + | + |
| 27 | (Rd)-K | CPI Rd, K | + | + | + | + | + | + |
| 28 | (Rd) ^K→ Rd | ANDI Rd, K |  | + | 0 | + | + |  |
| 29 | (Rd) ^K→ Rd | CBR Rd, K |  | + | 0 | + | + |  |
| 30 | (Rd) ˅K→ Rd | ORI Rd, K |  | + | 0 | + | + |  |
| 31 | (Rd) ˅K→ Rd | SBR Rd, K |  | + | 0 | + | + |  |
| d=16-31(!); K=0-255 |

Таблица 4

|  |  |  |  |
| --- | --- | --- | --- |
| № | Операция | Мнемокод команды | Признаки результата |
| H | S | V | N | Z | C |
| 32 | (Rd+1, Rd)+K→ Rd+1,Rd | ADIWRd, K |  | + | + | + | + | + |
| 33 | (Rd+1,Rd)-K→ Rd+1,Rd | SBIW Rd, K |  | + | + | + | + | + |
| d=24, 26, 28,30; K=0-63 |

**6.2 Система команд**

Таблица 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Операция | Мнемокод команды | № | Операция | Мнемокод команды |
| 34 | (Ячk) → Rd | LDS Rd, K | 35 | (Rr) →Ячk | STS k, Rr |
| d, r=0-31; k – адрес из адресного пространства SRAM |

Таблица 6

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Операция | Мнемокод команды | № | Операция | Мнемокод команды |
| 36 | (Яч(X)) → Rd | LD Rd, X | 37 | (Rr) →Яч(X) | ST X, Rr |
| 38 | (Яч(Y)) → Rd | LD Rd, Y | 39 | (Rr) →Яч(Y) | ST Y, Rr |
| 40 | (Яч(Z)) → Rd | LD Rd, Z | 41 | (Rr) →Яч(Z) | ST Z, Rr |
| 42 | 1.(Яч(X)) → Rd2. (X)+1→X | LD Rd, X+ | 43 | 1.(Rr)→Яч(X) 2.(X)+1→X | ST X+, Rr |
| 44 | 1.(Яч(Y)) → Rd2. (Y)+1→Y | LD Rd, Y+ | 45 | 1.(Rr)→Яч(Y) 2.(Y)+1→Y | ST Y+, Rr |
| 46 | 1.(Яч(Z)) → Rd2. (Z)+1→Z | LD Rd, Z+ | 47 | 1.(Rr)→Яч(Z) 2.(Z)+1→Z | ST Z+, Rr |
| 48 | 1.(X)-1→X 2. (Яч(X))→ Rd | LD Rd, -X | 49 | 1.(X)-1 →X 2.(Rr)→Яч(X) | ST -X, Rr |
| 50 | 1.(Y)-1→Y 2. (Яч(Y))→ Rd | LD Rd, -Y | 51 | 1.(Y)-1 →Y 2.(Rr)→Яч(Y) | ST -Y, Rr |
| 52 | 1.(Z)-1→Z 2. (Яч(Z))→ Rd | LD Rd, -Z | 53 | 1.(Z)-1 →Z 2.(Rr)→Яч(Z) | ST -Z, Rr |
| 54 | (Яч(Y)+q) → Rd | LDD Rd, Y+q | 55 | (Rr)→Яч(Y)+q | STDY+q, Rr |
| 56 | (Яч(Z)+q) → Rd | LDD Rd, Z+q | 57 | (Rr)→Яч(Z)+q | STDZ+q, Rr |
| 58 | 1.(SP)+1→SP 2. (СТЕК)→ Rd | POP Rd | 59 | 1.(Rr)-1→СТЕК 2.(SP)-1→SP | PUSH Rr |
| d, r = 0-31; q=0-63 |

Таблица 7

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Операция | Мнемокод команды | № | Операция | Мнемокод команды |
| 60 | (PrP) → Rd | IN Rd, P | 61 | (Pr) →PrP | OUT P,Pr |
| d, r = 0-31; P=0-63=S00-$3F |

Таблица 8

|  |  |  |  |
| --- | --- | --- | --- |
| № | Операция | Мнемокод команды | (Z.15-1) – адрес в FlashROMZ.0 = 0 – мл. байт; Z.0 =1 –ст. байт |
| 62 | (Яч(Z))→ R0 | LPM |

Таблица 9

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Операция | Мнемокод команды | № | Операция | Мнемокод команды |
| 63 | T→ Rd.b | BLD Rd, b | 64 | Rr.b →T | BSTRr, b |
| 65 | 0→ PrP.b | CBI P, b | 66 | 1→PrP.b | SBI P, b |
| 67 | 0→ SREG.b | BLSR b | 68 | 1→SREG.b | SSET b |
| d, r = 0-31; P=0-31 (!); b= 0-7 |

Таблица 10

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Операция | Мнемокод команды | № | Операция | Мнемокод команды |
| 69 | 0→I | CLI Rd, X | 70 | 1 →I | SEI |
| 71 | 0 →T | CLT | 72 | 1 →T | SET |
| 73 | 0→H | CLH | 74 | 1 →H | SEH |
| 75 | 0→S | CLS | 76 | 1→S | SES |
| 77 | 0→V | CLV | 78 | 1→V | SEV |
| 79 | 0→N | CLN | 80 | 1→N | SEN |
| 81 | 0→Z | CLZ | 82 | 1→Z | SEZ |
| 83 | 0→C | CLC | 84 | 1→C | SEC |

Таблица 11

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Операция | Мнемокод команды | № | Операция | Мнемокод команды |
| 85 | (PC)+1+k→PC | RJMP k | 86 | (Z) →PC | IJMP |
| 87 | (PC)+→СТЕК(SP)-2→SP(PC)+1+k→PC | RCALL k | 88 | (PC)+1→СТЕК(SP)-2→SP(Z) →PC | ICALL |
| 89 | (SP)+2→ SP(СТЕК) →PC | RET | 90 | (SP)+2→SP(СТЕК) →PC1→I | RETI |
| k= -2048 - +2047 |

Таблица 12

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Условие | Мнемокод команды | № | Условие | Мнемокод команды |
| 91 | I = 0 | BRID k | 92 | I = 1 | BRIE k |
| 93 | T = 0 | BRTC k | 94 | T = 1 | BRTS k |
| 95 | H = 0 | BRHC k | 96 | H = 1 | BRHS k |
| 97 | S = 02. (X)+1→X | BRGE k | 98 | S = 1 | BRLT k |
| 99 | V = 0 | BRVC | 100 | V = 1 | BRVS k |
| 101 | N = 0 | BRPL k | 102 | N = 1 | BRMI k |
| 103 | Z = 0 | BRNE k | 104 | Z = 1 | BREQ k |
| 105 | C = 0 | BRCC k | 106 | C = 1 | BRCS k |
| 107 | C = 0 | BRSH k | 108 | C = 1 | BRLO k |
| 109 | SREG.b = 0 | BRBC b, k | 110 | SREG.b = 1 | BRBS b, k |
| k= -64 - +63; b = 0 - 7 |

Таблица 13

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Условие | Мнемокод команды | № | Условие | Мнемокод команды |
| 111 | Rr.b = 0 | SBRC Rr, b | 112 | Rrb = 1 | SBRS Rr, b |
| 113 | PrP.b = 0 | SBIC P, b | 114 | PrP.b = 1 | SPIS P, b |
| 115 | (Rd) = (Rr) | CPSE Rd, Rr |  |  |  |
| d, r= 0 – 31; P = 0 – 31 (!); b = 0 - 7 |

Таблица 14

|  |  |  |
| --- | --- | --- |
| № | Операция | Мнемокод команды |
| 116 | MK → режим энергосбережения | SLEEP |
| 117 | Перезапуск WDT | WDR |
| 118 | Нет | NOP |

**6.3 Команды регистровых операций**

В группу регистровых операций входят пересылочные, арифметические и логические операции. Команды регистровых операций описаны в табл. 2, 3 и 4 Машинные коды всех команд регистровых операций имеют формат «слово», команды №№ 1 – 31 выполняются за один такт, а команды №№ 32 и 33 – за два такта.

При описании операций в табл. 2 – 4 используются следующие обозначения:

▪ Rd, Rr – регистры общего назначения с номерами d и r соответственно;

▪ (Rd), (Rr) – байты в регистрах Rd и Rr соответственно;

▪ Rd.b – разряд b (b = 0 – 7) регистра Rd, бит в разряде Rd.b;

▪ $ - указатель шестнадцатеричного кода;

▪ ^, ˅, - знаки логических операций И, ИЛИ, исключающее ИЛИ соответственно;

▪ ( ), ( ), ( ) – знаки операций НЕ, сдвиг кода вправо, сдвиг кода влево соответственно.

Арифметические операции – сложение и вычитание – могут выполняться с числами без знака в двоичном коде и с числами со знаком в дополнительном двоичном коде. В отличие от микроконтроллеров многих других семейств, в которых вычитание сводится к сложению с числом с изменённым знаком (X–Y = X + (-Y)), в микроконтроллерах семейства AVR вычитание выполняется с помощью аппаратного двоичного вычитателя.

При выполнении арифметических и логических операций кроме кода результата формируются значения признаков результата. При выполнении операций сравнения (команды №№ 18, 19 и 27) формируются только значения признаков результата.

Значения признаков результата представлены состоянием разрядов регистра состояния SREG (№$33F). Используются шесть признаков результата, которым присвоены имена C (SREG.0), Z(SREG.1) N (SREG.2), V (SREG.3, S (SREG.4) и H (SREG.5). В табл. 2.1 – 2.3 признаки, значения которых формируются при выполнении команд, отмечены знаком «+» или указано их название.

При выполнении разных операций значение признаков формируется по разным правилам.

Признак C принимает единичное значение:

▪ при появлении единицы переноса из старшего разряда при выполнении операции сложения (команды №№ 14, 15 и 32);

▪ при появлении единицы займа в старший разряд при выполнении операции сложения (команды №№ 4, 16, 17, 18, 19, 25, 26, 27 и 33);

▪ при выходе единицы за пределы разрядной сетки при выполнении операции сдвига (команды №№ 7, 8, 9, 10 и 11).

Единичное значение признака C при выполнении операций сложения и вычитания с числами без знака свидетельствует о получении неправильного результата операции вследствие переполнения разрядной сетки.

Признак Z принимает единичное значение при получении нулевого результата операции. При выполнении операции вычитания с учётом займа (команды №№ 17, 19 и 26) Z = 1, если нулевой результат получен при выполнении данной и предшествующей операций.

Признак N имеет значение, равное значению старшего разряда кода результата. При выполнении арифметических операций с числами со знаком и отсутствии переполнения разрядной сетки он представляет знак результата.

Признак V принимает единичное значение, если при выполнении арифметической операции с числами со знаком возникло переполнение разрядной сетки.

Признак S = NV представляет знак результата при выполнении арифметических операций с числами со знаком вне зависимости от переполнения разрядной сетки. При переполнении разрядной сетки (V = 1) знак результата противоположен значению в старшем разряде кода результата.

Признак H принимает единичное значение при наличии единицы переноса из разряда D3 в разряд D4 при выполнении операции вычитания и при передаче единицы из разряда D3 в разряд D4 при выполнении операции кода влево.

**6.4 Команды с обращением по адресу SRAM**

По командам с обращением по адресу в адресном пространстве SRAM выполняются операции пересылки (копирования) байта между регистром общего назначения и элементом, которому соответствует адрес в адресном пространстве SRAM. Таким элементом может быть регистр общего назначения (адреса от $00 до $1F), регистр ввода-вывода (адреса от $20 до $5F), ячейка памяти в SRAM (адреса от $60 до максимального адреса в SRAM), ячейка памяти в ERAM(адреса от максимального адреса в SRAM +1 до максимального адреса в ERAM).

Команды с обращением по адресу в адресном пространстве SRAM описаны в табл. 2.4 и 2.5.

Машинные коды команд №№ 34 и 35 имеют формат «2 слова», машинные коды остальных команд – формат «слово». Команды выполняются за 2 такта, а при обращении к ячейкам в ERAM–за 3 такта. В случае необходимости в цикл выполнения команды при обращении к ERAM может быть введён дополнительный такт ожидания.

В табл.2.4 и 2.5 используются следующие новые обозначения:

▪ ЯчА – регистр общего назначения, регистр ввода-вывода, ячейка памяти в SRAM, ячейка памяти в ERAM, к которым производится обращение с использованием адреса А из адресного пространства SRAM;

▪ (ЯчА) – байт в ЯчА;

▪X, Y, Z – пара регистров общего назначения X (R26, R27), Y (R28, R29), Z (R30, R31) соответственно;

▪ (X), (Y), (Z) – слово в паре регистров X, Y, Z соответственно;

▪ СТЕК – ЯчА, к которой производится обращение с использованием адреса, хранящегося в регистре – указателе стека SP;

▪ (СТЕК) – байт в СТЕК, е;

▪ (SP) – код числа в регистре SP.

В командах №№ 34 и 35 реализуется прямая адресация. Адрес байта (A = k) указывается в коде команды.

В командах №№ 36 – 57 реализуется косвенная адресация. Адрес байта находится в паре регистров X, Y или Z. При выполнении команд №№ 48 – 53 до выполнения пересылки адрес в паре регистров аппаратно уменьшается на единицу. При выполнении команд №№ 54 – 57 адрес, по которому производится обращение, равен адресу в паре регистров Y или Z, увеличенному на единицу q, которое указано в коде команды. Адрес в паре регистров остаётся без изменения.

В командах №№ 58 и 59 в качестве кода адреса используется код, находящейся в регистре-указателе стека. При запуске микроконтроллера в регистр-указатель стека заносится код числа 0. Для нормальной работы стека в регистр – указатель стека в начале программы необходимо занести код другого числа. Обычно в качестве такого числа используют старший адрес в SRAM. Например, для микроконтроллера типа 8515 таким адресом является число $025F.

**6.5 Команды с обращением к регистрам ввода-вывода**

По командам с обращением к регистрам ввода-вывода выполняются операции пересылки (копирования) байта между регистром общего назначения и регистром ввода-вывода с использованием для выбора регистра ввода-вывода его номера ($00 - $3F). Команды с обращением к регистрам ввода-вывода описаны в табл. 6. Машинные коды команд имеют формат «слово». Команды выполняются за один такт.

В табл. 6 используются следующие новые обозначения:

▪ PrP – регистр ввода-вывода с номером P;

▪ (PrP) – байт в регистре PrP.

При записи мнемокода команды вместо конкретного номера регистра может использоваться его символическое имя. Все регистры ввода-вывода имеют штатные имена. У микроконтроллеров разных типов регистры ввода-вывода, имеющие одинаковые имена, могут иметь разные имена. При записи в мнемокодах имён регистров вместо их номера необходимо использовать версию AVR-Ассемблера для микроконтроллера данного типа.

**6.6 Команда с обращением к FlashROM**

По команде с обращением к FlashROM выполняется пересылка (копирование) байта из половины ячейки памяти в FlashROM в регистр общего назначения R0. Команда описана в табл. 2.7. Машинный код команды имеет формат «слово». Команда выполняется за три такта.

До выполнения команды необходимо адрес, соответствующий ячейке памяти в FlashROM, в которой хранится константа, умножить на 2 и записать в пару регистров Z (№№ $30, $31).

При умножении на 2 (сдвиг кода влево на один разряд) в младшем разряде регистра ZL ($30) оказывается цифра 0. При нулевом значении младшего разряда выбирается байт из младшей половины ячейки FlashROM. Для выбора байта из старшей половины ячейки необходимо число, код которого находится в паре регистров Z, увеличить на 1 с использованием команды № 33 с мнемокодом ADIWR30, 1 или ADIWZL, 1.

Запись констант в FlashROM выполняется при программировании микроконтроллера.

**6.7 Команды операций с битами**

Команды операций с битами описаны в табл. 2.8 и 2.9. Машинные коды всех команд имеют формат «слово». Команды №№ 65 и 66 выполняются за 2 такта, остальные команды – за 1 такт.

В табл. 2.8 используется следующее новое обозначение:

▪ PrP.b – разряд b (b = 0 – 7) регистра ввода-вывода с номером P (P = $00 - $1F (!)), бит в разряде PrP.b.

По командам №№ 63 и 64 выполняется пересылка (копирование) бита между указанным разрядом регистра общего назначения и разрядом Tрегистра SREG.

По командам №№ 65 и 66 устанавливается в требуемое состояние (0 или 1 соответственно) указанный разряд регистра ввода-вывода, а по командам №№ 67 и 68 – указанный разряд регистра SREG.

При записи мнемокода команд №№ 65 и 66 вместо номера регистра (P) может быть указанно его символическое имя, а вместо номера разряды (b) –символическое имя разряда. В микроконтроллерах семейства AVR разряды многих регистров ввода-вывода имеют штатные имена. Эти имена приводятся в тексте при рассмотрении устройств, в которых эти разряды используются.

В микроконтроллерах некоторых типов разряды одноимённых регистров с одинаковыми именами имеют разные номера. При использовании штатных имён регистров ввода-вывода в штатных имён разрядов в них необходимо использовать версию AVR Ассемблера для микроконтроллера соответствующего типа.

При назначении штатного имени разряда подразумевалось, что этот разряд принадлежит определённому регистру ввода-вывода, однако в мнемокоде команд №№ 65 и 66 необходимо указывать и имя/номер регистра и имя разряда.

Требуемое значение бита (0 или 1) в разрядах регистров ввода-вывода с номерами от $20 до $3Fустанавливается с использованием команд регистровых операций с мнемокодами CBR (№ 29) и SBR (№ 31) соответственно.

В табл. 10 приведены мнемокоды команд, по которым устанавливаются в определённое состояние (0 или 1) разряды регистра SREG без указания в мнемокоде команды номера разряда.

**6.8 Команды управления ходом программы**

В группу команд управления ходом программы входят команды безусловного и условного переходов, перехода в энергосберегающий режим, перезапуска сторожевого таймера и «холостая» команда.

В табл. 11 описаны команды безусловного перехода (№№ 85 и 86), безусловного перехода с возвратом (№№ 87 и 88), возврата из подпрограммы (№89) и возврата из прерывающей программы (№90). В таблице используются следующие новые обозначения:

▪ PC – счётчик команд;

▪ (PC) – код числа в счётчике команд;

▪ k – приращение числа в счётчике команд.

При записи программы на языке ассемблера в мнемокодах команд №№85 и 87 вместо приращения k указывается метка (символический адрес), записанная перед мнемокодом команды, к выполнению которой нужно перейти. Приращение вычисляется и помещается в машинный код команды в процессе ассемблирования.

Машинные коды команд, описанные в табл. 11, имеют формат «слово». Команды безусловного перехода (№№ 85 и 86) выполняются за 2 такта, команды безусловного перехода а возвратом (№№ 87 и88) – за 3 такта, команды возврата (№№89 и 90) – за 4 такта.

Группа команд условного перехода делится на две группы. В командах первой подгруппы в качестве условия используется равенство нулю или равенство единице одного из битов в регистре SREG. Условия и мнемокоды команд первой подгруппы приведены в табл. 12. Если условие выполняется, совершается переход по адресу, который формируется с использованием приращения k, указанного в коде команды ((PC) + 1 +k→PC; -64 ≤k≤ 63).

Если условие не выполняется, происходит переход к следующей команде в программе ((PC) + 1→PC).

При записи программы на языке ассемблера в мнемокоде команд вместо приращения k указывается метка (символический адрес), записанная перед мнемокодом команды, к выполнению которой нужно перейти, если условие выполняется.

Машинные коды команд первой подгруппы имеют формат «слово». При выполнении условия команда выполняется за 2 такта, при выполнении условия – за 1 такт.

Если условие не выполняется, совершается переход к очередной команде в программе ((PC) +1→PC). Если условие выполняется, происходит переход к выполнению команды, следующей за очередной ((PC) + 1 + 1/2→PC). Приращение числа в счётчике команд (1 или 2) определяется форматом машинного кода очередной команды (1 слово или 2 слова соответственно).

Машинные коды команд условного перехода второй подгруппы имеют формат «слово». Команды выполняются за один такт, если условие не выполняется, за 2 такта, если условие выполняется и очередная команда (K1) имеет формат «слово», или за 3 такта, если условие не выполняется и очередная команда (K1) имеет формат «2 слова».

**Список литературы**

1. Белов А.В. Самоучитель разработчика устройств на микроконтроллерах AVR . – Санкт-Петербург, «Наука и техника» 2010 г.
2. http://www.atmel.ru/
3. ATtiny 2313 Data Sheet.
4. Гребнев В.В. Микроконтроллеры семейства AVRфирмы Atmel. –М. «ИП РадиоСофт» 2002 г.