**ATmega128, ATmega128L**

**8-разрядный AVR-микроконтроллер с внутрисистемно программируемой флэш-памятью емкостью 128 кбайт**

**Отличительные особенности:**

* Высокопроизводительный, маломощный 8-разрядный AVR-микроконтроллер
* Развитая RISC-архитектура
– 133 мощных инструкций, большинство из которых выполняются за один машинный цикл
– 32 8-разр. регистров общего назначения + регистры управления встроенной периферией
– Полностью статическая работа
– Производительность до 16 млн. операций в секунду при тактовой частоте 16 МГц
– Встроенное умножающее устройство выполняет умножение за 2 машинных цикла
* Энергонезависимая память программ и данных
– Износостойкость 128-ми кбайт внутрисистемно перепрограммируемой флэш-памяти: 1000 циклов запись/стирание
– Опциональный загрузочный сектор с отдельной программируемой защитой

Внутрисистемное программирование встроенной загрузочной программой
Гарантированная двухоперационность: возможность чтения во время записи

– Износостойкость 4 кбайт ЭСППЗУ: 100000 циклов запись/стирание
– Встроенное статическое ОЗУ емкостью 4 кбайт
– Опциональная возможность адресации внешней памяти размером до 64 кбайт
– Программируемая защита кода программы
– Интерфейс SPI для внутрисистемного программирования

* Интерфейс JTAG (совместимость со стандартом IEEE 1149.1)
– Граничное сканирование в соответствии со стандартом JTAG
– Обширная поддержка функций встроенной отладки
– Программирование флэш-памяти, ЭСППЗУ, бит конфигурации и защиты через интерфейс JTAG
* Отличительные особенности периферийных устройств
– Два 8-разр. таймера-счетчика с раздельными предделителями и режимами сравнения
– Два расширенных 16-разр. таймера-счетчика с отдельными предделителями, режимами сравнения и режимами захвата
– Счетчик реального времени с отдельным генератором
– Два 8-разр. каналов ШИМ
– 6 каналов ШИМ с программируемым разрешением от 2 до 16 разрядов
– Модулятор выходов сравнения
– 8 мультиплексированных каналов 10-разрядного аналогово-цифрового преобразования

8 несимметричных каналов
7 дифференциальных каналов
2 дифференциальных канала с выборочным усилением из 1x, 10x и 200x

– Двухпроводной последовательный интерфейс, ориентированный не передачу данных в байтном формате
– Два канала программируемых последовательных УСАПП
– Последовательный интерфейс SPI с поддержкой режимов ведущий/подчиненный
– Программируемый сторожевой таймер со встроенным генератором
– Встроенный аналоговый компаратор

* Специальные возможности микроконтроллера
– Сброс при подаче питания и программируемая схема сброса при снижении напряжения питания
– Встроенный калиброванный RC-генератор
– Внешние и внутренние источники прерываний
– Шесть режимов снижения энергопотребления: холостой ход (Idle), уменьшение шумов АЦП, экономичный (Power-save), выключение (Power-down), дежурный (Standby) и расширенный дежурный (Extended Standby)
– Программный выбор тактовой частоты
– Конфигурационный бит для перевода в режим совместимости с ATmega103
– Общее выключение подтягивающих резисторов на всех линиях портов ввода-вывода
* Ввод-вывод и корпуса
– 53 –программируемые линии ввода-вывода
– 64-выв. корпус TQFP
* Рабочие напряжения
– 2.7 - 5.5В для ATmega128L
– 4.5 - 5.5В для ATmega128
* Градации по быстродействию
– 0 - 8 МГц для ATmega128L
– 0 - 16 МГц для ATmega128


Расположение выводов

**Рисунок 1 – Расположение выводов у ATmega128**

**Краткий обзор**

ATmega128 – маломощный 8-разр. КМОП микроконтроллер, основанный на расширенной AVR RISC-архитектуре. За счет выполнения большинства инструкций за один машинный цикл ATmega128 достигает производительности 1 млн. операций в секунду/МГц, что позволяет проектировщикам систем оптимизировать соотношение энергопотребления и быстродействия.

**Функциональная схема**



**Рисунок 2 – Функциональная схема**

Ядро AVR сочетает богатый набор инструкций с 32 универсальными рабочими регистрами. Все 32 регистра непосредственно подключены к арифметико-логическому устройству (АЛУ), который позволяет указать два различных регистра в одной инструкции и выполнить ее за один цикл. Данная архитектура обладает большей эффективностью кода за счет достижения производительности в 10 раз выше по сравнению с обычными CISC-микроконтроллерами.

ATmega128 содержит следующие элементы: 128 кбайт внутрисистемно программируемой флэш-памяти с поддержкой чтения во время записи, 4 кбайт ЭСППЗУ, 4 кбайт статического ОЗУ, 53 линии универсального ввода-вывода, 32 универсальных рабочих регистра, счетчик реального времени (RTC), четыре гибких таймера-счетчика с режимами сравнения и ШИМ, 2 УСАПП, двухпроводной последовательный интерфейс ориентированный на передачу байт, 8-канальный 10-разр. АЦП с опциональным дифференциальным входом с программируемым коэффициентом усиления, программируемый сторожевой таймер с внутренним генератором, последовательный порт SPI, испытательный интерфейс JTAG совместимый со стандартом IEEE 1149.1, который также используется для доступа к встроенной системе отладке и для программирования, а также шесть программно выбираемых режимов уменьшения мощности. Режим холостого хода (Idle) останавливает ЦПУ, но при этом поддерживая работу статического ОЗУ, таймеров-счетчиков, SPI-порта и системы прерываний. Режим выключения (Powerdown) позволяет сохранить содержимое регистров, при остановленном генераторе и выключении встроенных функций до следующего прерывания или аппаратного сброса. В экономичном режиме (Power-save) асинхронный таймер продолжает работу, позволяя пользователю сохранить функцию счета времени в то время, когда остальная часть контроллера находится в состоянии сна. Режим снижения шумов АЦП (ADC Noise Reduction) останавливает ЦПУ и все модули ввода-вывода, кроме асинхронного таймера и АЦП для минимизации импульсных шумов в процессе преобразования АЦП. В дежурном режиме (Standby) кварцевый/резонаторный генератор продолжают работу, а остальная часть микроконтроллера находится в режиме сна. Данный режим характеризуется малой потребляемой мощностью, но при этом позволяет достичь самого быстрого возврата в рабочий режим. В расширенном дежурном режиме (Extended Standby) основной генератор и асинхронный таймер продолжают работать.

Микроконтроллер производится по технологии высокоплотной энергонезависимой памяти компании Atmel. Встроенная внутрисистемно программируемая флэш-память позволяет перепрограммировать память программ непосредственно внутри системы через последовательный интерфейс SPI с помощью простого программатора или с помощью автономной программы в загрузочном секторе. Загрузочная программа может использовать любой интерфейс для загрузки прикладной программы во флэш-память. Программа в загрузочном секторе продолжает работу в процессе обновления прикладной секции флэш-памяти, тем самым поддерживая двухоперационность: чтение во время записи. За счет сочетания 8-разр. RISC ЦПУ с внутрисистемно самопрограммируемой флэш-памятью в одной микросхеме ATmega128 является мощным микроконтроллером, позволяющим достичь высокой степени гибкости и эффективной стоимости при проектировании большинства приложений встроенного управления.

ATmega128 поддерживается полным набором программных и аппаратных средств для проектирования, в т.ч.: Си-компиляторы, макроассемблеры, программные отладчики/симуляторы, внутрисистемные эмуляторы и оценочные наборы.

**Совместимость ATmega103 и ATmega128**

ATmega128 – микроконтроллер высокой сложности, где количество местоположений ввода-вывода составляет 64 местоположения, зарезервированных в наборе инструкций. Для гарантирования обратной совместимости с ATmega103 все местоположения ввода-вывода ATmega103 совпадают с ATmega128. Некоторые позиции ввода-вывода добавлены в расширенное пространство ввода-вывода, начиная с адреса $60 до $F, (т.е. во внутреннем ОЗУ ATmega103). Доступ к данным местоположениям осуществляется только с помощью инструкций LD/LDS/LDD и ST/STS/STD, а не с помощью IN и OUT. Перераспределение внутреннего пространства ОЗУ может стать проблемой для пользователей ATmega103. Кроме того, если в коде программы используются абсолютные адреса прерываний, то это также вызовет проблему, т.к. у ATmega128 увеличено число векторов прерываний. Для решения этих проблем может использоваться режим совместимости с ATmega103, для чего необходимо запрограммировать конфигурационный бит M103C. В данном режиме не доступны функции из расширенного пространства ввода-вывода, т.о. достигнуто расположение внутреннего ОЗУ как у ATmega103. Также удалены расширенные векторы прерываний.

ATmega128 полностью совместим по расположению выводов с ATmega103 и может быть установлен на существующую плату для ATmega103. См. рекомендации по применению “Замена ATmega103 на ATmega128”, где описывается, что необходимо знать при замене ATmega103 на ATmega128.

**Режим совместимости с ATmega103**

Если запрограммировать конфигурационный бит M103C, то ATmega128 будет функционировать в режиме совместимости с ATmega103, настраивая ОЗУ, линии ввода-вывода и векторы прерываний как описано выше. Однако, некоторые новые функции ATmega128 в этом режиме станут недоступными. Данные функции перечислены ниже:

* Один УСАПП вместо двух, только асинхронный режим. Доступны только 8 младших разрядов в регистре скорости связи.
* Один 16-разр. таймер-счетчик с двумя регистрами сравнения вместо двух 16-разр. таймер-счетчиков с тремя регистрами сравнения.
* Не поддерживается двухпроводной последовательный интерфейс.
* Порт C действует только на вывод.
* Порт G выполняет только альтернативные функции (не универсальный ввод-вывод).
* Порт F действует только как цифровой ввод в дополнение к аналоговому вводу к АЦП.
* Не поддерживаются возможности автономного программирования из загрузочного сектора.
* Не возможно регулировать частоту внутреннего калиброванного RC-генератора.
* Интерфейс внешней памяти не может освободить неиспользуемые адресные сигналы для задач универсального ввода-вывода, не конфигурируются различные паузы для разных диапазонов адресов внешней памяти.

Кроме того, для большей совместимости с ATmega103 выполнены другие незначительные изменения:

* В регистре MCUCSR присутствуют только EXTRF и
* В регистре MCUCSR присутствуют только EXTRF и PORF.
* Временная последовательность не требуется для изменения периода переполнения сторожевого таймера.
* Запросы на внешние прерывания 3 - 0 генерируются только по уровню входного сигнала.
* УСАПП не имеет буфера FIFO, поэтому, переполнение при приеме данных происходит раньше.

Неиспользуемые биты ввода-вывода у ATmega103 должны быть сброшены (запись 0) для гарантирования одинакового функционирования в составе ATmega128.

**Описание выводов**

|  |  |
| --- | --- |
| **VCC** | Напряжение питания цифровых элементов |
| **GND** | Общий |
| **Порт A (PA7..PA0)** | Порт A – 8-разр. порт двунаправленного ввода-вывода с внутренними подтягивающими к плюсу резисторами (выбираются раздельно для каждого разряда). Выходные буферы порта A имеют симметричную выходную характеристику с одинаковыми втекающим и вытекающим токами. При вводе, линии порта А будут действовать как источник тока, если внешне действует низкий уровень и включены подтягивающие резисторы. Выводы порта A находятся в третьем (высокоимпедансном) состоянии при выполнении условия сброса, даже если синхронизация не запущена. Порт А также выполняет некоторые специальные функции ATmega128, описываемых далее. |
| **Порт В (PВ7..PВ0)** | Порт B – 8-разр. порт двунаправленного ввода-вывода с внутренними подтягивающими к плюсу резисторами (выбираются раздельно для каждого разряда). Выходные буферы порта В имеют симметричную выходную характеристику с одинаковыми втекающим и вытекающим токами. При вводе, линии по9рта B будут действовать как источник тока, если внешне действует низкий уровень и включены подтягивающие резисторы. Выводы порта B находятся в третьем (высокоимпедансном) состоянии при выполнении условия сброса, даже если синхронизация не запущена. Порт В также выполняет некоторые специальные функции ATmega128, описываемых далее. |
| **Порт C (PC7..PC0)** | Порт C – 8-разр. порт двунаправленного ввода-вывода с внутренними подтягивающими к плюсу резисторами (выбираются раздельно для каждого разряда). Выходные буферы порта C имеют симметричную выходную характеристику с одинаковыми втекающим и вытекающим токами. При вводе, линии порта C будут действовать как источник тока, если внешне действует низкий уровень и включены подтягивающие резисторы. Выводы порта C находятся в третьем (высокоимпедансном) состоянии при выполнении условия сброса, даже если синхронизация не запущена. Порт C также выполняет некоторые специальные функции ATmega128, описываемых далее. В режиме совместимости с ATmega103 порт C действует только на вывод, а при выполнении условия сброса линии порта C не переходят в третье состояние. |
| **Порт D (PD7..PD0)** | Порт D – 8-разр. порт двунаправленного ввода-вывода с внутренними подтягивающими к плюсу резисторами (выбираются раздельно для каждого разряда). Выходные буферы порта D имеют симметричную выходную характеристику с одинаковыми втекающим и вытекающим токами. При вводе, линии порта D будут действовать как источник тока, если внешне действует низкий уровень и включены подтягивающие резисторы. Выводы порта D находятся в третьем (высокоимпедансном) состоянии при выполнении условия сброса, даже если синхронизация не запущена. Порт D также выполняет некоторые специальные функции ATmega128, описываемых далее. |
| **Порт E (PE7..PE0)** | Порт E – 8-разр. порт двунаправленного ввода-вывода с внутренними подтягивающими к плюсу резисторами (выбираются раздельно для каждого разряда). Выходные буферы порта E имеют симметричную выходную характеристику с одинаковыми втекающим и вытекающим токами. При вводе, линии порта E будут действовать как источник тока, если внешне действует низкий уровень и включены подтягивающие резисторы. Выводы порта E находятся в третьем (высокоимпедансном) состоянии при выполнении условия сброса, даже если синхронизация не запущена. Порт E также выполняет некоторые специальные функции ATmega128, описываемых далее. |
| **Порт F (PF7..PF0)** | Порт F действует как аналоговый ввод аналогово-цифрового преобразователя. Порт F также может использоваться как 8-разр. порт двунаправленного ввода-вывода, если АЦП не используется. К каждой линии порта может быть подключен встроенный подтягивающий к плюсу резистор (выбирается раздельно для каждого бита). Выходные буферы порта F имеют симметричную выходную характеристику с одинаковыми втекающим и вытекающим токами. При вводе, линии порта F будут действовать как источник тока, если внешне действует низкий уровень и включены подтягивающие резисторы. Выводы порта F находятся в третьем (высокоимпедансном) состоянии при выполнении условия сброса, даже если синхронизация не запущена. Если активизирован интерфейс JTAG, то подтягивающие резисторы на линиях PF7(TDI), PF5(TMS) и PF4(TCK) будут подключены, даже если выполняется Сброс.Вывод TDO находится в третьем состоянии, если не введено состояние TAP, при котором сдвигаются выводимые данные.Порт F также выполняет функции интерфейса JTAG.В режиме совместимости с ATmega103 порт F действует только на ввод. |
| **Порт G (PG4..PG0)** | Порт G – 5-разр. порт двунаправленного ввода-вывода с внутренними подтягивающими к плюсу резисторами (выбираются раздельно для каждого разряда). Выходные буферы порта G имеют симметричную выходную характеристику с одинаковыми втекающим и вытекающим токами. При вводе, линии порта G будут действовать как источник тока, если внешне действует низкий уровень и включены подтягивающие резисторы. Выводы порта G находятся в третьем (высокоимпедансном) состоянии при выполнении условия сброса, даже если синхронизация не запущена.Порт G также выполняет некоторые специальные функции ATmega128.В режиме совместим9ости с ATmega103 данные выводы используются как стробирующие сигналы интерфейса внешней памяти, а также как вход генератора 32 кГц, а при действии сброса они асинхронно принимают следующие состояния: PG0 = 1, PG1 = 1 и PG2 = 0, даже если синхронизация не запущена. PG3 и PG4 – выводы генератора. |
| **RESET** | вход сброса. Если на этот вход приложить низкий уровень длительностью более минимально необходимой будет генерирован сброс независимо от работы синхронизации. Минимальная длительность внешнего импульса сброса приведена в таблице 19.Действие импульса меньшей продолжительности не гарантирует генерацию сброса. |
| **XTAL1** | вход инвертирующего усилителя генератора и вход внешней синхронизации. |
| **XTAL2** | выход инвертирующего усилителя генератора. |
| **AVCC** | вход питания порта F и аналогово-цифрового преобразователя. Он должен быть внешне связан с VCC, даже если АЦП не используется. При использовании АЦП этот вывод связан с VCC через фильтр низких частот. |
| **AREF** | вход подключения источника опорного напряжения АЦП. |
| **PEN** | вход разрешения программирования для режима последовательного программирования через интерфейс SPI. Если во время действия сброса при подаче питания на этот вход подать низкий уровень, то микроконтроллер переходит в режим последовательного программирования через SPI. В рабочем режиме PEN не выполняет никаких функций. |

**О примерах программ**

В данный документ входят примеры простых программ, которые кратко показывают как использовать различные составные части микроконтроллера. При составлении данных примеров предполагалось, что специфические файлы заголовков прописаны перед компиляцией. Следует понимать, что не все поставщики Си-компиляторов включают определения бит в файлы заголовков, а обработка прерываний в Си зависит от компилятора. Для уточнения этих особенностей см. документацию на используемый Си-компилятор.

|  |  |  |
| --- | --- | --- |
| **<-- Предыдущая страница** | [**Оглавление**](http://www.gaw.ru/html.cgi/txt/doc/micros/avr/arh128/index.htm) | [**Следующая страница -->**](http://www.gaw.ru/html.cgi/txt/doc/micros/avr/arh128/1_2.htm) |

**Ядро центрального процессорного устройства AVR**

**Введение**

В данном разделе описываются общие особенности архитектуры ядра AVR. Основная функция ядра ЦПУ заключается в гарантировании корректности выполнения программы. Помимо этого, ЦПУ должен иметь возможность адресоваться к различным видам памяти, выполнять вычисления, управлять периферийными устройствами и обрабатывать прерывания.

**Краткий обзор архитектуры**


Рисунок 3 – Функциональная схема архитектуры AVR

В целях достижения максимальной производительности и параллелелизма у AVR-микроконтроллеров используется Гарвардская архитектура с раздельными памятью и шинами программ и данных. Команды в памяти программ выполняются с одноуровневой конвейеризацией. В процессе выполнения одной инструкции следующая предварительно считывается из памяти программ. Данная концепция позволяет выполнять одну инструкцию за один машинный цикл. Память программ представляет собой внутрисистемно программируемую флэш-память.

Регистровый файл с быстрым доступом содержит 32 x 8-разр. рабочих регистров общего назначения с однотактовым циклом доступа. Благодаря этому достигнута однотактность работы арифметико-логического устройства (АЛУ). При обычной работе АЛУ сначала из регистрового файла загружается два операнда, затем выполняется операция, а после результат отправляется обратно в регистровый файл и все это происходит за один машинный цикл.

6 регистров из 32 могут использоваться как три 16-разр. регистра косвенного адреса для эффективной адресации в пределах памяти данных. Один из этих указателей адреса может также использоваться как указатель адреса для доступа к таблице преобразования во флэш-памяти программ. Данные 16-разр. регистры называются X-регистр, Y-регистр и Z-регистр и описываются далее в этом разделе.

АЛУ поддерживает арифметические и логические операции между регистрами, а также между константой и регистром. Кроме того, АЛУ поддерживает действия с одним регистром. После выполнения арифметической операции регистр статуса обновляется для отображения результата выполнения операции.

Для ветвления программы поддерживаются инструкции условных и безусловных переходов и вызовов процедур, позволяющих непосредственно адресоваться в пределах адресного пространства. Большинство инструкций представляют собой одно 16-разр. слово. Каждый адрес памяти программ содержит 16- или 32-разр. инструкцию. Флэш-память программ разделена на две секции: секция программы начальной загрузки и секция прикладной программы. Обе секции имеют раздельные биты защиты от записи и чтения/записи. Инструкция SPM (запись в секцию прикладной программы) должна использоваться только внутри секции программы начальной загрузки.

При генерации прерывания и вызове подпрограмм адрес возврата из программного счетчика записывается в стек. Стек эффективно распределен в статическом ОЗУ памяти данных и, следовательно, размер стека ограничен общим размером статического ОЗУ и используемым его объемом. В любой программе сразу после сброса должна быть выполнена инициализация указателя стека (SP) (т.е. перед выполнением процедур обработки прерываний или вызовом подпрограмм). Указатель стека – SP – доступен на чтение и запись в пространстве ввода-вывода. Доступ к статическому ОЗУ данных может быть легко осуществлен через 5 различных режимов адресации архитектуры AVR.

Гибкий модуль прерываний содержит свои управляющие регистры в пространстве ввода-вывода и имеет дополнительный бит общего разрешения работы системы прерываний в регистре статуса. У всех прерываний имеется свой вектор прерывания в соответствии с таблицей векторов прерываний. Прерывания имеют приоритет в соответствии с позицией их вектора. Прерывания с меньшим адресом прерывания имеют более высокий приоритет.

Пространство памяти ввода-вывода содержит 64 адреса с непосредственной адресацией или может адресоваться как память данных, следующая за регистрами по адресам $20 - $5F. Кроме того, ATmega128 имеет пространство расширенного ввода-вывода по адресам $60 - $FF в статическом ОЗУ, для доступа к которому могут использоваться только процедуры ST/STS/STD и LD/LDS/LDD.

**АЛУ – арифметико-логическое устройство**

Высокопроизводительное АЛУ AVR-микроконтроллеров работает в непосредственной связи со всеми 32 универсальными рабочими регистрами. АЛУ позволяет выполнить за один машинный цикл операцию между двумя регистрами или между регистром и константой. Операции АЛУ могут быть классифицированы на три группы: арифметические, логические и битовые. Кроме того, архитектурой ATmega128 поддерживаются операции умножения со знаком и без знака и дробным форматом. См. раздел "[**Набор инструкций**](http://www.gaw.ru/html.cgi/txt/doc/micros/avr/arh128/23.htm)" для подробного ознакомления.

**Регистр статуса**

Регистр статуса содержит информацию о результате только что выполненной арифметической инструкции. Данная информация может использоваться для ветвления программы по условию. Следует понимать, что регистр статуса обновляется после выполнения всех операций АЛУ в объеме предусмотренном для каждой конкретной инструкции (см. раздел Флаги в таблице инструкций). Флаги этого регистра в большинстве случаев позволяют отказаться от использования инструкций сравнения, делая код программы более компактным и быстрым.

Обратите внимание, что состояние регистра статуса автоматически не запоминается при вызове процедуры обработки прерываний и не восстанавливается при выходе из нее. Это необходимо выполнить программно.

Регистр статуса SREG AVR-микроконтроллера имеет следующую структуру:



**Разряд 7 – I: Общее разрешение прерываний**

Бит общего разрешения прерываний используется для активизации работы системы прерываний. Разрешение отдельных прерываний осуществляется в соответствующих управляющих регистрах. Если бит общего разрешения прерываний сбросить, то ни одно из прерываний не будет активным независимо от их индивидуальной конфигурации. Бит I сбрасывается в 0 аппаратно после генерации запроса на прерывание, а после выполнения инструкции возврата из прерывания RETI снова устанавливается к 1 для выполнения последующих прерываний. Бит I может также сбрасываться и устанавливаться с помощью инструкций CLI и SEI, соответственно.

**Разряд 6 – T: Хранение копируемого бита**

Специальные битовые операции BLD (копирование из Т-бита) и BST (копирование в Т-бит) используют в качестве источника и получателя данных бит T. Любой бит из регистрового файла может быть скопирован в бит T инструкцией BST, а также содержимое бита Т может быть скопировано в любой бит регистрового файла с помощью инструкции BLD.

**Разряд 5 – H: Флаг половинного переноса**

Данный флаг устанавливается при выполнении некоторых арифметических инструкций и индицирует о возникновении половинного переноса. Как правило половинный перенос широко используется в двоично-десятичной арифметике. Более подробная информация приведена в описании набора инструкций.

**Разряд 4 – S: бит знака, S = Искл. ИЛИ (N,?V)**

Бит S – результат выполнения логической операции исключающего ИЛИ между флагом отрицательного результата N и флагом переполнения двоичного дополнения V. Более подробная информация приведена в описании набора инструкций.

**Разряд 3 – V: Флаг переполнения двоичного дополнения**

Флаг переполнения двоичного дополнения V поддерживает арифметику с двоичным дополнением. Более подробная информация приведена в описании набора инструкций.

**Разряд 2 – N: Флаг отрицательного результата**

Флаг отрицательного результата N индицирует, что результатом выполнения арифметической или логической операции является отрицательное значение. Более подробная информация приведена в описании набора инструкций.

**Разряд 1 – Z: Флаг нулевого результата**

Флаг нулевого результата Z индицирует, что результатом выполнения арифметической или логической операции является ноль. Более подробная информация приведена в описании набора инструкций.

**Разряд 0 – C: Флаг переноса**

Флаг переноса C индицирует о возникновении переноса в результате выполнения арифметической или логической операции. Более подробная информация приведена в описании набора инструкций.

**Файл регистров общего назначения**

Файл регистров оптимизирован под расширенный набор инструкций AVR-микроконтроллеров. В целях достижения требуемой производительности и гибкости файлом регистров поддерживаются следующие схемы ввода-вывода:

* Один 8-разр. операнд и один 8-разр. результат
* Два 8-разр. операнда и один 8-разр. результат
* Два 8-разр. операнда и один 16-разр. результат
* Один 16-разр. операнд и один 16-разр. результат

Рисунок 4 показывает структуру 32 рабочих регистров общего назначения в ЦПУ.


Рисунок 4 – Рабочие регистры общего назначения ЦПУ AVR

Большинство инструкций работающих с файлом регистров имеют непосредственный доступ ко всем регистрам, чем достигается выполнение их за один машинный цикл.

Как показано на рисунке 4, каждый регистр имеет свой адрес в области памяти данных, для чего отведено там первые 32 позиции. Не смотря на физическую реализацию не по адресам статического ОЗУ, данная архитектура памяти обеспечивает высокую гибкость доступа к регистрам, например, регистры-указатели X, Y и Z могут быть установлены для присвоения индекса любому регистру в файле.

**X-регистр, Y-регистр и Z-регистр**

Регистры R26..R31 обладают некоторым дополнительными функциями для их общецелевого использования. Данные регистры являются 16-разр. указателями адреса для косвенной адресации в пределах памяти данных.
Три регистра косвенной адресации X, Y и Z представлены на рисунке 5.


Рисунок 5 - X, Y и Z-регистры

В различных режимах адресации данные адресные регистры выполняют функции фиксированного смещения, автоматического инкрементирования и автоматического декрементирования (см. описание набора инструкций для более подробного изучения).

Стек обычно используется для хранения временных данных, для хранения локальных переменных и для хранения адресов возврата при прерываниях и вызовах подпрограмм. Регистр указателя стека указывает на вершину стека. Обратите внимание на организацию стека, который направляется от старших в более младшие позиции статического ОЗУ. Это означает, что команда помещения в стек PUSH уменьшает значение указателя стека.

Указатель стека указывает на область стека в статическом ОЗУ данных, где размещены стеки прерываний и подпрограммы. Данная область стека в статическом ОЗУ памяти данных должна быть определена программно до вызова любой процедуры или разрешения прерываний. Устанавливаемое значение указателя стека должно быть более $60. Указатель стека однократно декрементируется при помещении данных в стек инструкцией PUSH и дважды декрементируется при помещении в стек адреса возврата при вызове подпрограмм или прерываниях. Указатель стека однократно инкрементируется при извлечении данных из стека инструкцией POP и дважды инкрементируется при извлечении адреса возврата при выполнении инструкции выхода из подпрограммы RET или выхода из процедуры обработки прерываний RETI.

Указатель стека реализован как два 8-разр. регистра в области ввода-вывода. Число фактически используемых разрядов зависит от типа микроконтроллера. Обратите внимание, что у некоторых AVR-микроконтроллеров область памяти данных настолько мала, что достаточно только регистра SPL. В этом случае регистр SPH отсутствует.

**Регистр выбора Z-страницы ОЗУ – RAMPZ**





**Разряды 7…2 – Зарезервированные разряды**

Данные зарезервированные разряды считываются как 0. При записи в данные разряды необходимо записывать нули для совместимости с последующими микроконтроллерами.

**Разряд 1 – RAMPZ0: Расширенный указатель страницы ОЗУ**

Регистр RAMPZ обычно используется для указания той страницы ОЗУ размером 64 кбайт, к которой выполняется доступ через Z-указатель. Т.к. ATmega128 не поддерживает память на статическом ОЗУ размером свыше 64 кбайт, то данный регистр используется только для выбора страницы памяти программ, доступ к которой осуществляется с помощью инструкций ELPM/SPM. Различные установки бита RAMPZ0 имеют следующий результат:

|  |  |
| --- | --- |
| RAMPZ0 = 0: | Инструкции ELPM/SPM осуществляют доступ к памяти программ в диапазоне адресов $0000 - $7FFF (младшие 64 кбайт) |
| RAMPZ0 = 1: | Инструкции ELPM/SPM выполняют доступ к памяти программ в диапазоне адресов $8000 - $FFFF (старшие 64 кбайт) |

Обратите внимание, что действие инструкции LPM не зависит от установки RAMPZ.

**Временная диаграмма выполнения инструкции**

ЦПУ AVR-микроконтроллера тактируется сигналом CLKЦПУ, который непосредственно генерируется выбранным источником синхронизации. Внутреннее деление тактовой частоты не используется.

Рисунок 6 показывает параллельность выборок и исполнения инструкций, что обеспечивается Гарвардской архитектурой и концепцией регистрового файла с быстрым доступом. Данная концепция конвейеризации обеспечивает удельную производительности 1 млн.оп в сек./МГц и предоставляет уникальное соотношение числа функций на стоимость, число функций на такт синхронизации и числа функций на Вт потребляемой мощности.


Рисунок 6 – Параллельные выборки и исполнения инструкций

Рисунок 7 иллюстрирует концепцию внутренней временной диаграммы для регистрового файла. За один такт синхронизации АЛУ выполняет действие над двухрегистровым операндом и возвращает результат обратно в регистр-получатель.


Рисунок 7 – Однотактность работы АЛУ

**Сброс и обработка прерываний**

AVR-микроконтроллеры поддерживают несколько различных источников прерываний. Все прерывания, а также сброс имеют свой индивидуальный вектор в памяти программ. Для каждого прерывания имеется собственный бит разрешения. Кроме того, имеется возможность общего разрешения работы прерываний с помощью управления соответствующим битом в статусном регистре. В зависимости от значения программного счетчика прерывания могут быть автоматически отключены, если запрограммировать биты защиты загрузочного сектора BLB02 или BLB12. Данная функция улучшает защиту программы. См. раздел “[**Программирование памяти**](http://www.gaw.ru/html.cgi/txt/doc/micros/avr/arh128/19.htm)” для уточнения деталей.

Наименьшие адреса в памяти программ по умолчанию определены как вектора сброса и прерываний. Полный перечень векторов приведен в разделе "[**Прерывания**](http://www.gaw.ru/html.cgi/txt/doc/micros/avr/arh128/6.htm)". В перечне также определяется уровень приоритетов различных прерываний. Меньшие адреса обладают более высоким уровнем приоритетом. Сброс (RESET) имеет наивысший приоритет, за ним следует INT0 – запрос на внешнее прерывание по входу INT0. Векторы прерывания могут быть перемещены в начало загрузочного сектора флэш-памяти установкой бита IVSEL в регистре управления микроконтроллером (MCUCR). См. раздел "Прерывания” для более подробного ознакомления. Вектор сброса может быть также перемещен в начало загрузочного сектора флэш-памяти путем программирования конфигурационного бита BOOTRST (см. “Самопрограммирование из сектора начальной загрузки с поддержкой чтения во время записи”).

После возникновения прерывания бит I общего разрешения прерываний сбрасывается и все прерывания запрещаются. Пользователь может программно записать лог. 1 в бит I для разрешения вложенных прерываний. В этом случае все разрешенные прерывания могут прервать текущую процедуру обработки прерываний. Бит I автоматически устанавливается после выполнения инструкции выхода из прерывания RETI.

Имеется два основных типа прерываний. Первый тип прерываний активизируется событием, которое приводит к установке флага прерываний. Для данных прерываний программный счетчик изменяется на соответствующий вектор прерывания для выполнения процедуры его обработки и затем аппаратно очищает флаг прерывания. Флаги прерывания также сбрасываются путем записи лог.1 в соответствующий разряд. Если возникает условие прерывания, но данное прерывание запрещено, то флаг устанавливается и запоминается до разрешения этого прерывания или сбрасывается программно. Аналогично, если возникает одно и более условий прерываний при сброшенном флаге общего разрешения прерываний, то соответствующий флаг устанавливается и запоминается до возобновления работы прерываний, а затем прерывания будут выполнены в соответствии с приоритетом.

Второй тип прерываний активизируется сразу после выполнения условия прерывания. Данные прерывания не обязательно имеют флаги прерываний. Если условие прерывания исчезает до его разрешения, то данный запрос игнорируется.

После выхода из прерывания AVR-микроконтроллер возвращается к выполнению основной программы и выполняет еще одну инструкцию до обслуживания любого из отложенных прерываний.

Обратите внимание, что регистр статуса автоматически не запоминается при вызове процедуры обработки прерывания и не восстанавливается при выходе из этой процедуры. Данные действия необходимо выполнить программно.

При выполнении инструкции CLI все прерывания запрещаются. Запрос на прерывание не будет отработан после выполнения инструкции CLI, даже если оно возникает одновременно с выполнением команды CLI. В следующем примере показано как избежать прерываний во время выполнения временной последовательности записи в ЭСППЗУ.

Пример кода на Ассемблере

in r16, SREG ; Запомнили состояние регистра статуса SREG

cli ; отключаем все прерывания во время отработки временной последовательности

sbi EECR, EEMWE ; Разрешаем запись в ЭСППЗУ

sbi EECR, EEWE

out SREG, r16 ; Восстанавливаем значение SREG (бит I)

Пример кода на Си

char cSREG;

cSREG = SREG; /\* Запоминаем значение SREG \*/

/\* Отключение прерываний на время задания временной последовательности \*/

\_CLI();

EECR |= (1<<EEMWE); /\* Старт записи в ЭСППЗУ EEPROM \*/

EECR |= (1<<EEWE);

SREG = cSREG; /\* Восстанавливаем значение SREG (бит I) \*/

Для разрешения прерываний используется инструкция SEI, а следующая за SEI инструкция будет выполнена перед отработкой любого отложенного прерывания, как показано в примере.

Пример кода на Ассемблере

sei ; Общее разрешение прерываний

sleep ; перевод в режим ожидания прерывания

; Прим.: Режим ожидания будет введен прежде чем запустится отработка отложенного прерывания

Пример кода на Си

\_SEI(); /\* Общее разрешение прерываний \*/

\_SLEEP(); /\* перевод в режим ожидания прерывания \*/

/\* Прим.: Режим ожидания будет введен прежде чем запустится отработка отложенного прерывания \*/

**Время реакции на прерывание**

Реакция на отработку запроса на прерывание длится минимум 4 машинных цикла. По истечении этого времени программа продолжает свое выполнение с вектора соответствующего прерывания. В течение 4 машинных циклов состояние программного счетчика помещается в стек. Как правило, по адресу вектора прерываний хранится команда перехода на процедуру обработку прерываний, а на данный переход затрачивается еще 3 машинных цикла. Если запрос на прерывание возникает в процессе исполнения инструкции, требующей более 1 машинного цикла на выполнение, то прерывание будет обработано только после выполнения этой инструкции. Если прерывание возникает во время нахождения микроконтроллера в режиме сна, то реакция на прерывание увеличится еще на 4 цикла. Данная задержка связана с временем старта из выбранного режима сна.

Выход из процедуры обработки прерывания требует 4 машинных цикла. В течение этого времени двухбайтный программный счетчик извлекается из стека, указатель стека дважды инкрементируется и устанавливается бит I в регистре статуса SREG.

|  |  |  |
| --- | --- | --- |
| [**<-- Предыдущая страница**](http://www.gaw.ru/html.cgi/txt/doc/micros/avr/arh128/1.htm) | [**Оглавление**](http://www.gaw.ru/html.cgi/txt/doc/micros/avr/arh128/index.htm) | [**Следующая страница -->**](http://www.gaw.ru/html.cgi/txt/doc/micros/avr/arh128/2.htm) |