Общая информация о программировании МК AVR на Си в WinAVR

Версия документа: 1.03

В рамках этого документа мы рассмотрим, что такое микроконтроллер и основные особенности использования языка Си (версии GCC, из поставки WinAVR) для программирования микроконтроллеров семейства AVR. Мы будем считать, что читатель уже знаком с основами использования языка Си вообще (для этого существует масса литературы, этот язык преподают в школах и высших учебных заведения и так далее).

Что такое микроконтроллер

Микроконтроллер (сокращенно называемый также МК или по-английски MCU или μC) — это программируемая микросхема. Важнейшей особенностью МК является то, что выполняемые им функции не задаются окончательно при производстве микросхемы, а в значительной степени определяются записанной в него программой. Микроконтроллеры применяются во многих современных приборах, таких, как телефоны, стиральные машины, устройствах управления промышленным оборудованием, и т. п, и конечно в роботах.

По сути, микроконтроллер - это комьютер из одной микросхемы. Типичный МК состоит из процессора, памяти программ, памяти данных и набора устройств ввода-вывода. Параметры различных МК могут очень сильно отличаться. Например, процессор может работать с тактовой частотой от 1МГц до сотен МГц, память данных может быть от нескольких десятков байт до нескольких десятков килобайт, а память программ - от сотен байт до сотен килобайт.

В среде русскоговорящих робостроителей наиболее популярны микроконтроллеры семейства AVR, выпускаемые компанией ATMEL. Более подробно устройство МК мы рассмотрим на примере микросхемы ATMega32 серии ATMega этой фирмы, используемой, наряду с другими, в модулях OpenRobotics.

Центральный процессор:

Микроконтроллеры AVR являются 8-битными. Это значит, что процессор оперирует данными размером в 1 байт. Процессор состоит из арифметико-логического устройства, 32 8-битных регистров общего назначения (РОН), регистра статуса и программного счетчика. Программный счетчик содержит адрес следующей исполняемой команды. Исходные данные для команд помещаются в регистры общего назначения, в них же помещаются результаты выполнения команд. Регистра статуса содержит дополнительную информацию о состоянии процессора и результате выполнения последней команды. Большинство команд процессор исполняет за один период тактовой частоты.

Память программ:

Память программ предназначена для хранения команд, исполняемых процессором. Она представляет собой моногкратно перепрограммируемую флэш-память, допускающую не менее 10'000 циклов перепрограммирования. Размер памяти программ в ATMega32 составляет 32Кб.

Память данных

Память данных (оперативное запоминающее устройство, ОЗУ) хранит данные, обрабатываемые процессором. Содерщимое памяти данных сохраняется только при наличии питающего напряжения. Микроконтроллер ATMega32 имеет 2Кб встроенного ОЗУ.

EEPROM

Микроконтроллеры ATMega содержат также перепрограммируемое запоминающее устройство (EEPROM). Данные в памяти EEPROM не пропадают при выключении питания. Микроконтроллер имеет специальные команды для записи в EEPROM и чтения из него. Доступ к EEPROM во много раз медленне, чем к ОЗУ. EEPROM допускает не менее 100'000 циклов перезаписи. Объем EEPROM у модели ATMega32 составляет 1 Кб.

Устройства ввода-вывода:

Микроконтроллеры ATMega имеют развитую систему устройств ввода-вывода. Ввод и вывод данных осуществляется через *порты*. Порт представляет собой группу выводов, через которые осуществляется прием и передача цифровых сигналов. Большинство портов состоят из 8 выводов (иногда меньше). Состоянием выводов порта можно управлять как одновременно, так и по отдельности. Любой вывод порта может независимо от других использоваться как вход или выход.

Многие выводы МК имеют альтернативные функции. Это значит, что вместо программного управления состоянием вывода, он подключается к одному из специальных устройств ввода-вывода. Устройства ввода-вывода МК ATmega включают в себя:

- Таймеры. Таймер является многофункциональным устройством. Он может использоваться для формирования временных задержек, подсчета количества импульсов на выводе МК, измерения длительносьти импульсов, генерации импульсов с заданной частотой и скважностью (ШИМ). Микроконтроллер ATMega32 имеет в своем составе от 3 таймера.
- Аналоговый компаратор. Сравнивает значения напряжений, присутствующих на двух выводах МК. Результат сравнения может быть прочитан из регистров.
- Аналогово-цифровой преобразователь (АЦП). Преобразует напряжение, присутствующее на выводе МК в цифровую форму. Результат преобразования может быть прочитан из регистров. Контроллер ATMega32 имеет 8 входов АЦП.
- Последовательный интерфейс SPI.
- Двухпроводный интерфейс TWI. Аббревиатурой TWI компания Atmel по юридическим причинам называет интерфейс I²C, разработанный фирмой Philips.
- Универсальный синхронно-асинхронный приемопередатчик (USART). Может использоваться, например, для связи с компьютером или КПК через последовательный порт.

Для управления устройствами ввода-вывода предназначены *регистры ввода-вывода*. Они представляют из себя специальные ячейки памяти. Записывая данные в эти ячейки и читая их содержимое, можно изменять режимы работы портов ввода-вывода, получать данные с АЦП, передавать информацию через последовательный порт, запускать таймеры, и т.д.

Прерывания:

Иногда возникает необходимость прервать исполнение программы, для того, чтобы обеспечить реакцию на внешнее событие. Например, микроконтроллер, управляющий электродвигателем, должен немедленно снять напряжение с мотора при возникновении короткого замыкания или заклинивании механизма, чтобы предотвратить выход из строя двигателя и схемы управления. Для обработки событий, требующих незамедлительной реакции, прменяются прерывания.

Прерывание - это сигнал, сообщающий процессору о возникновении какого-либо события. Примеры таких событий: изменение сигнала на входе МК (нажатие кнопки, срабатывание датчика), истечение интервала времени, поступление байта данных через последовательный порт. При этом выполнение текущей последовательности команд приостанавливается, и управление передаётся обработчику прерывания, который выполняет работу по обработке события и возвращает управление прерванной программе.

Практически все устройства ввода-вывода микроконтроллеров ATmega могут вызывать прерывания.

Основы использования Си для программирования МК AVR

Поддерживаемые типы данных

Мы приведём только основные типы данных:

Целочисленные:

1. [unsigned] char – беззнаковое или со знаком 8-битное целое;
2. [unsigned] int – беззнаковое или со знаком 16-битное целое;
3. [unsigned] long int – беззнаковое или со знаком 32-битное целое;
4. [unsigned] long long int – беззнаковое или со знаком 64-битное целое;

С плавающей запятой (дробные):

1. float, double – 32-битное с плавающей запятой;

Символьные:

1. char – один символ (1 байт), константы определяются одинарными кавычками.
2. Строки — последовательности символов с завершающим символом с кодом «0», так называемые «Null-terminating Strings». Константы определяются двойными кавычками.

ПОЛЕЗНЫЙ СОВЕТ: Если вы не хотите запоминать все эти целочисленные типы, какое из них какой длины в битах, или вам не нравится их длинное написание - к вашим услугам есть подключаемая библиотека <stdint.h>, в которой определены короткие идентификаторы целочисленных типов и явно указана их длина в битах: int8_t, uint8_t, int16_t, uint16_t, int32_t, uint32_t, int64_t, uint64_t.

Работа с портами ввода-вывода общего назначения

Все линии ввода-вывода общего назначения МК AVR сгруппированы по 8 штук (иногда меньше), такие группы называют портами и обозначают латинской буквой (A, B, C, D и так далее).

Каждому такому порту соответствует 3 регистра в МК, соответствующие биты которых отвечают за настройку режима работы порта и передачу\получение значений с этого порта (буквой «x» будет обозначаться буква соответствующего порта):

1. DDRx — регистр в котором указывается направление работы порта (вход или выход). Если соответствующий бит выставлен в 0, значит порт будет работать как вход, если в 1, значит как выход.
2. PORTx – регистр, назначение бит которого зависит от режима работы соответствующей линии порта, заданного DDRx. Если линия работает как выход, тогда бит этого регистра просто определяет состояние соответствующего выходного контакта МК. Если же линия работает как вход, тогда бит этого регистра определяет, будет ли подключен подтягивающий к «+» питания МК резистор на этом входе.
3. PINx – регистр, в битах которого можно прочитать входящие логические уровни линий порта, если они работают как входы (определяется состоянием бит регистра DDRx).

Подтягивающие резисторы удобно использовать, например, когда требуется подключить к МК обычный контактный датчик, тогда можно просто одной стороной цеплять его к земле схемы, а второй - напрямую на один из входов МК. В этом случае при незамкнутых контактах подтягивающий резистор будет поддерживать напряжение «+» питания на входе в МК, а при замыкании контактов напряжение упадёт до нулевого.

Пример использования портов ввода-вывода:

```
DDRB=0xF0;                          //Определим линии 0-3 порта B как входы, а линии 4-7 как выходы
PORTB |= 0x80;                      //Выведем в 7 линию порта B логическую единицу
PORTB |= 1 << 5;                    //Выведем в 5 линию порта B логическую единицу
PORTB |= 0x04;                      //Включим подтягивающий резистор на линии 2 порта B
uint8_t status=(PINB & 0x04) >> 2; //Сохраним в переменную status состояние 2 линии порта B
```

Работа с предопределенными регистрами

Множество функциональных блоков МК доступно для использования через специальные регистры (предопределенные ячейки памяти со специальными функциями). Например, можно записав в соответствующие регистры правильные настройки, обеспечить выдачу со специального порта МК прямоугольных импульсов нужной ширины.

При правильной настройке проектов все регистры доступны при программировании как переменные под соответствующим именем, написанном в верхнем регистре. В том числе PORTA, DDRA и PINA – это примеры таких специальных регистров доступных для использования в программе.

Подробней о всевозможных встроенных функциональных модулях и связанных с ними регистрах можно прочитать в документации к МК (в так называемых DataSheet'ах). Скачать её к любому из МК типа AVR можно по адресу: http://www.atmel.com/products/avr/default.asp, в разделе «DataSheets».

Пример использования регистров в МК ATMega32:

```
TCCR2=0x69;       //Включим таймер 2 и на нём ШИМ с множителем 1 (1 тик таймера каждый такт МК)
                  //Режим ШИМ — нормальный, при достижении заданного значения сбрасываем в 0 порт

OCR2=0x80;         //установим скважность импульса 128/255 ~= 50%
```

Определение функций, вызываемых при обработке прерываний

Для определения функций, которые будут вызываться при обработке прерываний предусмотрен специальный формат обозначения этих функций, содержащий ключевое слово SIGNAL и далее в скобках идентификатор прерывания, которое мы будем обрабатывать.

Пример определения обработчика прерывания по переполнению таймера №0:

```
#include <avr/interrupts.h>

ISR(TIMER0_OVF_vect)
{
  count++; //Увеличим свой счетчик на 1 при возникновении переполнения счетчика таймера
};
```

Вопросы размещения данных в памяти программ

Все константы по умолчанию располагаются в оперативной памяти, которой обычно в МК типа AVR не очень много. В ряде случаев это может стать серьёзной проблемой, если вам нужно иметь большой объем заранее сформированных данных. Один из самых простых выходов в данной ситуации — сохранить данные в памяти программ. Делается это следующим образом:

1. В программу добавляем ссылку на специальный заголовочный файл <avr/pgmspace.h>;
2. В определение константы добавляем модификатор PROGMEM;
3. Перед использованием данные нужно будет извлечь функцией pgm_read_byte(address).

Пример работы с данными в памяти программ:

```
#include <avr/pgmspace.h>

const char PROGMEM my_array[] = {0, ...};

void main()
{
    int my_array_index = 0;
    char my_value = pgm_read_byte(my_array + my_array_index);
    while(1);
    return 0;
};
```