

Avrdude - опции запуска и примеры

Прежде чем перейти к тестированию программатора вместе с микроконтроллером (МК) давайте сначала разберемся с возможностями программы avrdude, которая очень часто является основой при прошивке AVR кристаллов как в Linux, так и в других операционных системах. Приведен список всех параметров запуска программы avrdude, рассмотрим графическую оболочку, а также примеры использования avrdude.

Программа **AVRDude (AVR Downloader-Uploader)** - это очень мощный кроссплатформенный инструмент, который позволяет программировать всю линейку микроконтроллеров AVR, поддерживая при этом из коробки почти все типы доступных сейчас программаторов. Программа работает из консоли, что позволяет хорошо автоматизировать процесс прошивки микроконтроллеров но требует при этом внимательности и навыков работы с терминалом.

```

      Block Poll
Memory Type Mode Delay Size  Indx Paged  Size   Page  #Pages MinW  MaxW     Polled
-----
eeprom        4    20   128    0 no     512     4      0   9000   9000 0xff 0xff
flash        33    10    64    0 yes   8192    64     128  4500   4500 0xff 0x00
lfuse         0     0     0    0 no      1     0      0   2000   2000 0x00 0x00
hfuse         0     0     0    0 no      1     0      0   2000   2000 0x00 0x00
lock          0     0     0    0 no      1     0      0   2000   2000 0x00 0x00
calibration   0     0     0    0 no      4     0      0     0     0 0x00 0x00
signature     0     0     0    0 no      3     0      0     0     0 0x00 0x00

Programmer Type : usbasp
Description      : USBasp, http://www.fischl.de/usbasp/

avrdude: auto set sck period (because given equals null)
avrdude: warning: cannot set sck period. please check for usbasp firmware update.
avrdude: AVR device initialized and ready to accept instructions

Reading | ##### | 100% 0.00s

avrdude: Device signature = 0x1e9307
avrdude: safemode: lfuse reads as E1
avrdude: safemode: hfuse reads as D9
avrdude: reading flash memory:

Reading | ##### | 100% 4.20s

avrdude: writing output file "/tmp/1111/dump.raw"

avrdude: safemode: lfuse reads as E1
```

★ ph0en1x.net

Рис. 1. avrdude - кроссплатформенная программа для прошивки микроконтроллеров фирмы ATMEL.

Первоначальный код программы AVRdude был написан английским программистом **Брайеном Дином** (Brian S. Dean) и имел название AVRprog. Позже программа получила большой интерес со стороны пользователей и Брайен решил открыть ее код для всеобщего использования и доработки, а для того чтобы она не путалась с одноименной программой из AVRStudio - AVRProg, программа получила новое имя - AVRdude.

Программа AVRdude запускается и работает на ОС: Linux, Windows, MacOS X, FreeBSD и других.

Адрес официального сайта программы AVRdude:
<http://www.nongnu.org/avrdude/>

Параметры запуска avrdude

Запустив в консоли программу avrdude без аргументов мы сможем увидеть

список доступных опций для использования. Приведенную ниже информацию можно позже использовать как краткую справку по параметрам программы avrdude.

```
Usage: avrdude [options]
Options:
  -p <partno>          Required. Specify AVR device.
  -b <baudrate>        Override RS-232 baud rate.
  -B <bitclock>        Specify JTAG/STK500v2 bit clock period (us).
  -C <config-file>     Specify location of configuration file.
  -c <programmer>      Specify programmer type.
  -D                   Disable auto erase for flash memory
  -i <delay>           ISP Clock Delay [in microseconds]
  -P <port>            Specify connection port.
  -F                   Override invalid signature check.
  -e                   Perform a chip erase.
  -O                   Perform RC oscillator calibration (see AVR053).
  -U <memtype>:r|w|v:<filename>[:format]
                        Memory operation specification.
                        Multiple -U options are allowed, each request
                        is performed in the order specified.
  -n                   Do not write anything to the device.
  -V                   Do not verify.
  -u                   Disable safemode, default when running from a script.
  -s                   Silent safemode operation, will not ask you if
                        fuses should be changed back.
  -t                   Enter terminal mode.
  -E <exitspec>[,<exitspec>] List programmer exit specifications.
  -x <extended_param> Pass <extended_param> to programmer.
  -y                   Count # erase cycles in EEPROM.
  -Y <number>          Initialize erase cycle # in EEPROM.
  -v                   Verbose output. -v -v for more.
  -q                   Quell progress output. -q -q for less.
  -l logfile           Use logfile rather than stderr for diagnostics.
  -?                   Display this usage.

avrdude version 6.1, URL: <http://savannah.nongnu.org/projects/avrdude/>
```

Рис. 2. Список параметров программы avrdude.

Рассмотрим все опции программы по порядку:

- **-p <partno>** - является обязательной опцией, здесь мы в качестве <partno> указываем краткий псевдоним AVR микроконтроллера;
- **-b <baudrate>** - позволяет переопределить указанную для программатора в конфигурации программы скорость подключения по интерфейсу RS-232;
- **-B <bitclock>** - указываем Bit Clock Period для интерфейса отладки JTAG или ISP Clock (только для JTAG ICE). Значение <bitclock> указывается в микросекундах, для JTAG ICE по умолчанию оно установлено в 1 микросекунду и подходит для МК работающих на частотах 4МГц и выше;
- **-C <config-file>** - в качестве <config-file> указываем полный путь к файлу конфигурации с необходимыми нам настройками программы. По умолчанию используется файл /etc/avrdude.conf (Linux);
- **-c <programmer>** - в качестве <programmer> указываем псевдоним используемого программатора, например "usbasp".
- **-D** - опция запрещает автоматическое стирание Flash-памяти. Автоматическое стирание не используется в микроконтроллерах семейства ATxmega;
- **-i <delay>** - установка паузы перед каждой отправкой каждого бита для bitbang-программаторов. В качестве <delay> указывается значение в микросекундах. Это бывает необходимо если для программирования используется компьютер с очень быстрым процессором или же микроконтроллер с низкой тактовой частотой (32КГц, 128КГц), позволяет соблюдать условие: частота ISP < 1/4 частоты процессора;

- **-P <port>** - в качестве значения <port> указываем используемый программатором порт. По умолчанию используются /dev/ppi0 (параллельный порт) и /dev/cuaa0 (последовательный порт) в зависимости от программатора;
- **-F** - опция для отключения проверки сигнатуры микроконтроллера. По умолчанию перед программированием avrdude проверяет сигнатуру микроконтроллера, которая в некоторых случаях может быть повреждена, при этом микроконтроллер может продолжать нормально функционировать;
- **-e** - стирает содержимое FLASH и EEPROM памяти (заполнение значениями 0xFF), очищаются fuse-bits (биты защиты). Исключением являются микроконтроллеры семейства ATxmega в которых используется постраничная запись;
- **-O** - опция для калибровки RC-генератора в соответствии с примечанием AVR053 от Atmel. Поддерживается только на программаторах STK500v2, AVRISP mkII, и JTAG ICE mkII. Результат будет сохранен в EEPROM памяти в ячейке с адресом 0;
- **-U <memtype>:r|w|v:<filename>[:format]** - комплексная опция для указания производимой с памятью операции (чтение, запись, проверка);
- **-n** - запрет на запись в микроконтроллер, используется для отладки avrdude;
- **-V** - отключение автоматической проверки записанной информации;
- **-u** - отключить режим безопасной (safe mode) проверки и сопоставления ячеек конфигурации (fuse bits) до и по завершению программирования. Данная опция необходима если нужно изменить значения фьюзов (fuse bits), в противном случае avrdude в качестве меры безопасности изменит их значения на те которые были прочитаны перед программированием;
- **-s** - запрет вывода запросов в безопасном режиме при работе с фьюзами;
- **-t** - переводит avrdude в режим терминала (terminal mode);
- **-E <exitspec>[,<exitspec>]** - изменение состояния линий параллельного порта после программирования на указанные в аргументах опции. По умолчанию устанавливаются те состояния линий что были до начала работы;
- **-x <extended_param>** - позволяет указать дополнительный специальный параметр для используемого программатора;
- **-y** - опция что включает сохранение количества стираний МК в последних 4-х байтах памяти EEPROM;
- **-Y <number>** - указанное значение <number> будет сохранено в качестве числа циклов-стираний микроконтроллера в памяти EEPROM;
- **-v** - расширенный вывод информации о работе программы (verbose);
- **-q** - отключает отображение полосы прогресса при операциях с микроконтроллером. Для еще большего уменьшения отображаемой информации опцию следует указать дважды;
- **-l <logfile>** - перенаправление всего вывода для отладки в указанный файл, где <logfile> - полный путь к файлу для сохранения данных;
- **?** - отображение справки.

Работа с памятью (опция -U <memtype>:r|w|v:<filename>[:format])

В качестве <memtype> указываем тип памяти для работы:

- calibration - байты калибровки RC-генератора (один или несколько);
- eeprom - энергонезависимая память (EEPROM) микроконтроллера;
- efuse - дополнительный конфигурационный бит;
- flash - FLASH память микроконтроллера;
- fuse - фьюз-байт для МК только с одним fuse-байтом;
- hfuse - старший fuse-байт;
- lfuse - младший fuse-байт;
- lock - байт блокировки (ячейка защиты);
- signature - три байта что обозначают сигнатуру чипа (device ID);
- fuseN - байт с фьюзами для ATxmega чипов, N - целое число для каждого фьюза что поддерживается устройством;

- application - область приложений во Flash памяти для МК ATxmega;
- apptable - таблица приложений в области Flash памяти для устройств ATxmega;
- boot - загрузочная область Flash памяти для устройств ATxmega;
- prodsig - область с производственной сигнатурой (calibration) для устройств ATxmega;
- usersig - область с пользовательской сигнатурой для устройств ATxmega.

Дальше через двоеточие следует производимая операция с памятью МК:

- **r** - прочитать указанную область памяти и записать в указанный файл <filename>;
- **w** - прочитать данные из файла <filename> и записать в указанную память устройства;
- **v** - прочитать данные из указанного файла <filename> и из указанной области памяти (verify, проверка).

В поле <filename> указывается полный или относительный путь к файлу что используется для записи или чтения данных. Поле ":format" является не обязательным, с его помощью указывается формат используемого файла:

- i - Intel HEX;
- s - Motorola S-record;
- r - raw binary (RAW формат);
- e - ELF (Executable and Linkable Format);
- m - значения байтов для записи указываются непосредственно в командной строке в поле <filename> и разделяются пробелами или запятыми. По умолчанию байты пишутся в десятичной системе, если указать 0x - будет записано шестнадцатеричные значения, а если перед байтом стоит 0 - будет записано восьмеричное число;
- a - авто-определение формата (auto detect);
- d - десятичный формат (decimal), числа разделяются запятыми;
- h - шестнадцатеричный формат (hexadecimal), числа начинаются с 0x;
- o - восьмеричный формат (octal), перед числами ставится 0;
- b - двоичный формат (binary), перед числами ставится 0b.

По умолчанию используется авто-определение формата (auto detect).

Состояние линий параллельного порта (-E <exitspec>[,<exitspec>])

- **reset** - на линии RESET будут низкий уровень, микроконтроллер останется в состоянии сброса;
- **noreset** - на линию RESET поступит высокий уровень для запуска МК после программирования;
- **vcc** - установка высокого уровня на линии порта VCC , которая может использоваться для питания МК;
- **novcc** - подача низкого уровня на линию VCC.

Допускается использование нескольких значений через запятую.

Примеры использования avrdude

С опциями запуска разобрались, теперь давайте посмотрим как их использовать для выполнения нужных нам операций с микроконтроллером при помощи программы avrdude.

Выполним **тест** связки микроконтроллера ATtiny13 с программатором USBASP:

```
avrdude -p t13 -c usbasp
```

Произведем **чтение Flash-памяти** микроконтроллера ATmega88 в никуда (/dev/null), тест на читаемость флеш-памяти:

```
avrdude -p m88 -c usbasp -U flash:r:/dev/null:i
```

Выполним **чтение Flash-памяти** чипа ATmega8 в **файл** формата Intel HEX - /tmp/flash_dump.hex, при этом укажем что для программатора нужно использовать именно USB-порт (-P usb) и выводить больше отладочной информации (-v):

```
avrdude -p m8 -c usbasp -P usb -v -U flash:r:/tmp/flash_dump.hex:i
```

Прочитаем содержимое EEPROM-памяти микроконтроллера ATtiny85 и сохраним его в файл RAW формата (/tmp/eeprom_dump.raw), используя при этом программатор USBTiny:

```
avrdude -p t85 -c usbtiny -P usb -v -U eeprom:r:/tmp/eeprom_dump.raw:r
```

Произведем **запись данных** их HEX-файла (/tmp/program_m8.hex) **во FLASH-память** микроконтроллера ATmega8, используя программатор STK-500:

```
avrdude -c stk500 -p m8 -U flash:w:/tmp/program_m8.hex
```

Произведем **запись** данных во **FLASH и EEPROM** память одной командой, используя как источники данных для записи файлы /tmp/flash_1.hex и /tmp/eeprom_1.hex:

```
avrdude -c stk500 -p m8 -U flash:w:/tmp/flash_1.hex -U eeprom:w:/tmp/eeprom_1.hex
```

Выполним **чтение фьюзов** из микроконтроллера atmega8 и сохраним данные в файлы в шестнадцатеричном формате (hexadecimal, числа начинаются с 0x) :

```
avrdude -c usbasp -p m8 -U hfuse:r:m8_hfuse.txt:h -U lfuse:r:m8_lfuse.txt:h
```

Произведем **запись фьюзов** для микроконтроллера ATmega32, установлена частота внутреннего RC-генератора на 4MHz (Low=0xc3, High=0x99):

```
avrdude -c usbasp -p m32 -U lfuse:w:0xc3:m -U hfuse:w:0x99:m
```