Я уже писал про фьюз биты / байты (fuse bits / bytes) микроконтроллеров AVR много статей назад. Но, судя по большому количеству вопросов от читателей, тема не раскрыта полностью. В чем же проблема с установкой фьюз бит? Вроде бы есть картинка, на которой нарисовано какие галочки ставить, какие снимать – должно быть все просто. Но разработчики различных программ для программирования микроконтроллеров в своих программах используют настолько разнообразные варианты установки фьюз бит, что нетрудно запутаться. Чтобы как то прояснить вопрос установки фьюз бит (по крайней мере, касательно моих проектов в этом блоге) я взялся обобщить информацию по различным программам и свести все в одном месте.

для начала – **1 ОБЩАЯ ИНФОРМАЦИЯ.**

Fuse bits называют область (**4 байта**) в AVR микроконтроллерах отвечающую за начальную (глобальную) конфигурацию. Этими битами мы указываем микроконтроллеру, с каким задающим генератором ему работать (внешним / внутренним), делить частоту генератора на коэффициент или не нужно, использовать ножку сброса как сброс или как дополнительный порт ввода-вывода, количество памяти для загрузчика и многое, многое другое. У каждого контроллера свой набор фьюзов. Все фьюзы прописаны в даташите на микроконтроллер. С завода, по умолчанию, фьюзы выставлены для работы микроконтроллера от внутреннего задающего генератора. Ничего довешивать не нужно подал питание, и он работает. Если нужно как-то изменить работу микроконтроллера, например, заставить его работать от внешнего задающего генератора, нужно изменить соответствующие фьюзы.

Физически фьюз биты расположены в четырех специальных байтах:
- Lock Bit Byte – лок биты для защиты программы от копирования;
- Fuse Extended Byte – дополнительный байт – особые функции;
- Fuse High Byte – старший байт;
- Fuse Low Byte – младший байт.

Вот как распределены фьюз биты по байтам для ATtiny2313 (взято из даташита):

Table 64. Lock Bit Byte[1]

Lock Bit Byte	Bit No	Description	Default Value
	7	–	1 (unprogrammed)
	6	–	1 (unprogrammed)
	5	–	1 (unprogrammed)
	4	–	1 (unprogrammed)
	3	–	1 (unprogrammed)
	2	–	1 (unprogrammed)
LB2	1	Lock bit	1 (unprogrammed)
LB1	0	Lock bit	1 (unprogrammed)

Note: 1. "1" means unprogrammed, "0" means programmed

Table 66. Fuse Extended Byte

Fuse Extended Byte	Bit No	Description	Default Value
	7	–	1 (unprogrammed)
	6	–	1 (unprogrammed)
	5	–	1 (unprogrammed)
	4	–	1 (unprogrammed)
	3	–	1 (unprogrammed)
	2	–	1 (unprogrammed)
	1	–	1 (unprogrammed)
SELFPRGEN	0	Self Programming Enable	1 (unprogrammed)

Table 67. Fuse High Byte

Fuse High Byte	Bit No	Description	Default Value
DWEN[3]	7	debugWIRE Enable	1 (unprogrammed)
EESAVE	6	EEPROM memory is preserved through the Chip Erase	1 (unprogrammed, EEPROM not preserved)
SPIEN[1]	5	Enable Serial Program and Data Downloading	0 (programmed, SPI prog. enabled)
WDTON[2]	4	Watchdog Timer always on	1 (unprogrammed)
BODLEVEL2[4]	3	Brown-out Detector trigger level	1 (unprogrammed)
BODLEVEL1[4]	2	Brown-out Detector trigger level	1 (unprogrammed)
BODLEVEL0[4]	1	Brown-out Detector trigger level	1 (unprogrammed)
RSTDISBL[5]	0	External Reset disable	1 (unprogrammed)

Table 68. Fuse Low Byte

Fuse Low Byte	Bit No	Description	Default Value
CKDIV8	7	Divide clock by 8	0 (programmed)
CKOUT	6	Output Clock on CKOUT pin	1 (unprogrammed)
SUT1	5	Select start-up time	1 (unprogrammed)[1]
SUT0	4	Select start-up time	0 (programmed)[1]
CKSEL3	3	Select Clock source	0 (programmed)[2]
CKSEL2	2	Select Clock source	1 (unprogrammed)[2]
CKSEL1	1	Select Clock source	0 (programmed)[2]
CKSEL0	0	Select Clock source	0 (programmed)[2]

Важно знать!

Исторически так сложилось, что если фьюз равен:
0 – значит, запрограммирован / прошит / активен
1 – значит, НЕ запрограммирован / НЕ прошит / Не активен

Это нужно запомнить!
Почему так? Объясню. Сейчас конфигурационные байты записываются во флеш памяти и поменять их можно сколько угодно раз. Раньше, когда флеш памяти еще не было, для конфигурации какого-либо чипа в его архитектуре имелись специальные перемычки (fuse) которые разово физически сжигались. Вот поэтому, по старинке, если перемычка цела – «1» значит эта функция не задействована и наоборот – перемычку сожгли – «0» значит функция задействована.

Вот такая логика и является источником проблем с установкой фьюз бит.

2 НЕБОЛЬШОЙ ЛИКБЕЗ ПО НАЗНАЧЕНИЮ ФЬЮЗОВ.

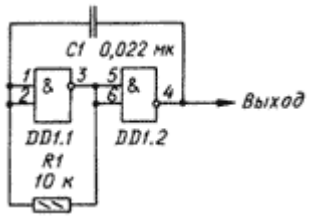
Здесь описаны не все фьюзы – только основные. Подробнее (и правильнее) о фьюзах нужно смотреть в даташитах на каждый конкретный микроконтроллер.

CKSEL – выбор тактового генератора для микроконтроллера.
Для работы микроконтроллера (как и для любого процессора) нужны тактовые импульсы. Источником тактового сигнала может быть:
– внутренний RC генератор. Никаких дополнительных элементов не нужно. Удобно, но RC генератор имеет небольшую точность работы (вплоть до 10% погрешности) и, кроме того, «плывет» от температуры. Для некритичных по времени приложений вполне годиться.
– внешний кварцевый (или керамический) резонатор. Нужен сам резонатор, плюс два конденсатора на 15-30пФ. Соответственно, будут заняты две ножки микроконтроллера – XLAT1 и XLAT2. Применяется там, где нужны точные замеры времени или частота работы микроконтроллера выше, чем может дать внутренний RC генератор.
– еще можно тактировать микроконтроллер от внешнего источника тактового сигнала. Это может быть другой микроконтроллер (для синхронизации работы) или внешняя схема, дающая нужный сигнал. Тактовый сигнал подается на ножку XLAT1.

Источник тактового сигнала для микроконтроллера задается комбинацией битов CKSEL3…0.
Это может быть (для ATTiny2313, выборочно):
CKSEL3…0 = 0000 – Внешний тактовый сигнал;
CKSEL3…0 = 0010 – Внутренний тактовый генератор – частота 4 МГц;
CKSEL3…0 = 0100 – Внутренний тактовый генератор – частота 8 МГц;
CKSEL3…0 = 1101 – Внешний тактовый генератор – кварц частотой от 3 до 8 МГц;
CKSEL3…0 = 1111 – Внешний тактовый генератор – кварц частотой больше 8 МГц.

Как оживить микроконтроллер, если неправильно установлены CKSEL?
Если Вы выставили фьюз биты на внешний генератор, а его нет, то микроконтроллер «пропадет» для программатора. В этом случае придется припаять кварц к соответствующим ножкам или подать тактовые импульсы на ножку XLAT1 микроконтроллера.

В «боевых» условиях получить тактовый сигнал можно несколькими способами:
– собрать несложный генератор на логике – паять можно прямо на ножках логики;



– если рядом имеется осциллограф, то у него есть источник образцового сигнала. Частота его, обычно, не большая, но фьюзы исправить хватит;
– если есть еще один микроконтроллер – делаем выход тактового сигнала на ножку микроконтроллера (нужно запрограммировать фьюз CKOUT) и подаем этот сигнал на XLAT1;
– есть еще «метод пальца» – крайне не рекомендую…

CKOUT – разрешает вывод тактовой частоты на ножку CLKO микроконтроллера (для тактирования других устройств).
CKOUT = 1 – ножка микроконтроллера работает как обычный порт ввода-вывода;
CKOUT = 0 – на ножку микроконтроллера выдается сигнал тактового генератора.

CKOPT – задает размах тактового сигнала на внешнем генераторе.
CKOPT = 1 – размах небольшой – генератор работает в экономном режиме. Нормально генератор может работать лишь при небольших частотах и в условиях близким к идеальным. При значительных помехах, большой тактовой частоте, перепадах (скачках) напряжения питания, микроконтроллер может работать нестабильно;
CKOPT = 0 – задающий генератор работает на полную мощность, устойчив к помехам и может работать во всем диапазоне частот. Если нет особых требований к энергосбережению – советую всегда программировать этот бит.

SCKDIV8 – деление тактовой частоты на 8.
Тут все просто:
SCKDIV8= 1 – микроконтроллер работает на частоте задающего генератора;
SCKDIV8= 0 – микроконтроллер работает на частоте в 8 раз меньше частоты задающего генератора;

SUT – задает скорость запуска микроконтроллера.
После снятия «сброса» (или подачи питания) программа, записанная в микроконтроллер, начинает работать не мгновенно. Микроконтроллер выжидает некоторое время, для того, чтобы нормально запустился тактовый генератор, установилось напряжение питания и т.д. Время ожидания до запуска программы и задают биты SUT1…0. Чаще всего нам не критична скорость запуска, поэтому советую ставить на максимум.
SUT1..0 = 11 – максимальное время запуска (чуть больше 65 mS).
На время запуска еще влияет CKSEL0, но это уже детали …

RSTDISBL –разрешает использовать ножку Reset как еще один порт ввода-вывода.
Иногда нужная вещь, но нужно знать -
после программирования RSTDISBL микроконтроллер уже нельзя будет прошить последовательным программатором! Поэтому без особой надобности не трогайте его.
RSTDISBL = 1 – ножка сброса работает как сброс;
RSTDISBL = 0 – ножка сброса работает как еще один порт ввода-вывода, последовательное программирование отключено.

SPIEN – разрешение на последовательное программирование.
По умолчанию запрограммирован (0) – разрешено последовательное программирование.
SPIEN = 0 – разрешено последовательное программирование;
SPIEN = 1 – запрещено последовательное программирование.

WDTON – включает Watch Dog Timer.
Для ответственных приложений, там, где недопустимо зависание программы (будь то ошибка программы или злостная помеха), применяют Watch Dog Timer. Это внутренний таймер микроконтроллера, работающий от своего независимого генератора. При переполнении этого таймера микроконтроллер сбрасывается и начинает выполнять программу с начала. Программист должен в тесте программы (обычно в главном цикле) вставить специальную команду обнуления этого таймера (WDR). Команда периодически выполняется и обнуляет таймер, не давая ему переполнится. Если микроконтроллер «повис» перестают выполняться команды обнуления, таймер переполняется и сбрасывает микроконтроллер.
WDTON = 1 – Watch Dog Timer – отключен (можно включить программно);
WDTON = 0 – Watch Dog Timer – включен (программно выключить нельзя).
В обычных приложениях не нужен.

BODLEVEL и BODEN – контроль напряжения питания микроконтроллера (Brown-out Detector).
Если питание микроконтроллера опуститься к минимально допустимому или чуть ниже, то работа микроконтроллерабудет нестабильной. Возможны ошибочные действия, потеря данных, случайное стирание EEPROM. Микроконтроллер умеет следить за уровнем своего питания (**BODEN=0**) и когда оно достигает уровня, который задается битами **BODLEVEL**, сбрасывается и держится в ресете пока уровень не поднимется до рабочего уровня. В некритических приложениях можно не использовать.

JTAGEN – разрешает интерфейс JTAG (внутрисхемный отладчик).
При активации некоторые линии микроконтроллера отдаются под интерфейс. Но зато можно подключать JTAG отладчик и с его помощью легко отладить любую программу прямо в схеме – удобно.
JTAGEN = 1 – запрещен JTAG;
JTAGEN = 0 – разрешен JTAG.

DWEN – бит, разрешающий работу DebugWire
– еще одного отладочного интерфейса. DebugWire однопроводный отладочный интерфейс работающий через ножку сброса, поэтому «не отнимает» у микроконтроллера ножки портов ввода-вывода.
DWEN= 1 – запрещен DebugWire ;
DWEN= 0 – разрешен DebugWire .

AVR микроконтроллеры могут во время своей работы изменять содержимое области программ (программировать сам себя).
SELFPRGEN – бит, разрешающей программе производить запись в память программ.
SELFPRGEN = 1 – изменение области программ запрещено;
SELFPRGEN = 0 – разрешено изменение области программ.

EESAVE – защита EEPROM от стирания.
При подаче команды полного стирания микроконтроллера (обычно осуществляется при каждом программировании кристалла) стирается и EEPROM. Если Вы хотите чтобы

EEPROM оставалось нетронутой – активируйте этот фьюз. Это актуально если в EEPROM хранятся важные данные.
EESAVE = 1 – стирать EEPROM вместе с Flash;
EESAVE = 0 – оставлять EEPROM при очистке нетронутым.

AVR микроконтроллеры могут иметь бутлоадер – это область в конце памяти, в которой можно разместить загрузчик, который предназначен для загрузки и запуска основной программы.
BOOTRST – как раз и заставляет микроконтроллер запускаться с области бутлоадера.
BOOTRST = 1 – микроконтроллер запускает программу с нулевого адреса;
BOOTRST = 0 – микроконтроллер запускает программу с бутлоадера.

BOOTSZ0..1 – задает размер бут сектора (области памяти программ для бутлоадера).

Lock Bits – Это отдельный фьюз байт который предназначен для защиты области программ и/или EEPROM от копирования. Полное стирание восстанавливает эти биты в исходное состояние.

Еще раз повторюсь, это не полный перечень фьюз бит, для каждого конкретного микроконтроллера смотрите даташит.

3 ЧАСТО ИСПОЛЬЗУЕМЫЕ КОНФИГУРАЦИИ ФЬЮЗ БИТ.

Для примера приведу некоторое количество конфигураций для микроконтроллеров. Картинки фьюзов сняты с Algorithm Builder'а.

Во всех картинках фьюзы как по даташиту:
- снятая галочка– fuse bit = 0, фьюз запрограммирован / активный;
- установленная галочка– fuse bit = 1, фьюз НЕ запрограммирован / НЕ активный.

Для **UniProf** - ставить как на картинке;
Для **PonyProg, CVAVR, AVR Studio** – ставить инверсно.

3.1 ATtiny13

ATTiny13.pdf - Даташит для ATTiny13/13V
ATtiny13_default_internal_RC_1.2 - Фьюзы ATtiny13 заводские настройки внутренний RC генератор на 1.2МГц
ATtiny13_internal_RC_4.8 - Фьюзы ATtiny13 внутренний RC генератор на 4.8МГц
ATtiny13_internal_RC_9.6 - Фьюзы ATtiny13 внутренний RC генератор на 9.6МГц
ATtiny13_internal_RC_0.128 - Фьюзы ATtiny13 внутренний RC генератор на 128кГц

3.2 ATtiny2313

ATTiny2313.pdf - Даташит ATTiny2313
ATtiny2313_default_internal_RC_1.0 - Фьюзы ATtiny2313 заводские настройки внутренний RC генератор на 1.0МГц
ATtiny2313_internal_RC_4.0 - Фьюзы ATtiny2313 внутренний RC генератор на 4.0МГц
ATtiny2313_internal_RC_8.0 - Фьюзы ATtiny2313 внутренний RC генератор на 8.0МГц
ATtiny2313_external_0.9_3.0 - Фьюзы ATtiny2313 внешний генератор на 0.9-3.0МГц

ATtiny2313_external_3.0_8.0 - Фьюзы ATtiny2313 внешний генератор на 3.0-8.0МГц
ATtiny2313_external_8.0_20.0 - Фьюзы ATtiny2313 внешний генератор на 8.0-20.0МГц

3.3 ATmega8

ATmega8.pdf - Даташит на ATmega8
ATmega8_default_internal_RC_1.0 - Фьюзы ATmega8 заводские настройки внутренний RC генератор на 1.0МГц
ATmega8_internal_RC_2.0 - Фьюзы ATmega8 внутренний RC генератор на 2.0МГц
ATmega8_internal_RC_4.0 - Фьюзы ATmega8 внутренний RC генератор на 4.0МГц
ATmega8_internal_RC_8.0 - Фьюзы ATmega8 внутренний RC генератор на 8.0МГц
ATmega8_external_1.0_16.0 - Фьюзы ATmega8 внешний генератор на 1.0-16.0МГц

3.4 ATmega48/88/168

ATMegaX8.pdf - Даташит ATMega48/88/168/V
ATmega48_88_168_default_internal_RC_1.0 - Фьюзы ATmega48/88/168 заводские настройки внутренний RC генератор на 1.0МГц
ATmega48_88_168_internal_RC_8.0 - Фьюзы ATmega48/88/168 внутренний RC генератор на 8.0МГц
ATmega48_88_168_internal_RC_0.128 - Фьюзы ATmega48/88/168 внутренний RC генератор на 128кГц
ATmega48_88_168_external_0.4_25.0 - Фьюзы ATmega48/88/168 внешний генератор на 0.4_25.0МГц

А теперь то, для чего писалась эта статья –
4 УСТАНОВКА ФЬЮЗ БИТ В РАЗЛИЧНЫХ ПРОГРАММАХ.

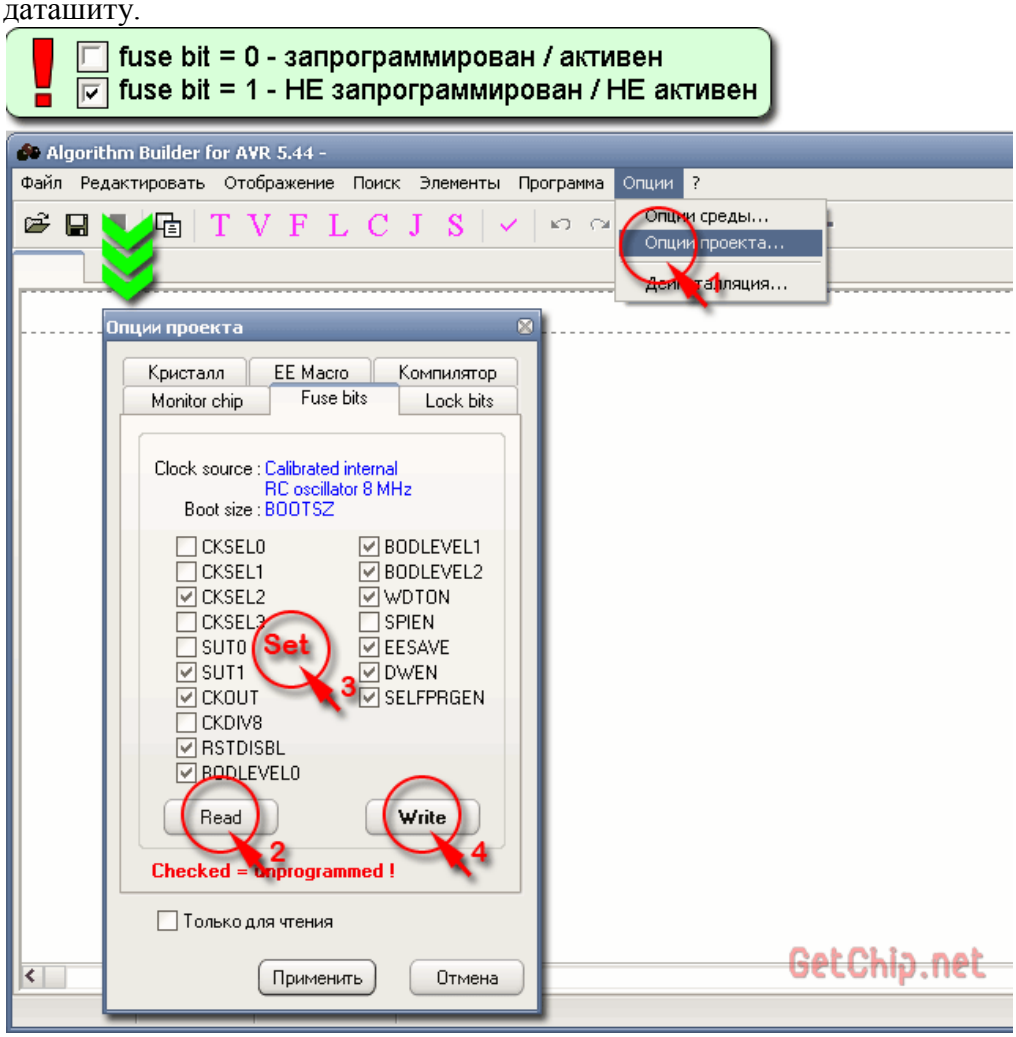
Общий алгоритм установки фьюз бит должен быть следующим:

- прошиваем Flash и, если нужно, EERROM;
– открываем окно прошивки фьюзов, считываем текущие фьюзы микроконтроллера;
– модифицируем только те фьюзы которые нам нужны;
– обращаем внимание на критичные для последовательного программирования фьюзы RSTDISBL, SPIEN, др.

4.1 Начнем, пожалуй, с Algorithm Builder’а.

http://algrom.net/russian.html
Раз я выкладываю картинки именно с него, нужно знать как устанавливаются в нем фьюзы.
Логика установки фьюзов в Algorithm Builder’а, я считаю, самая правильная – строго по

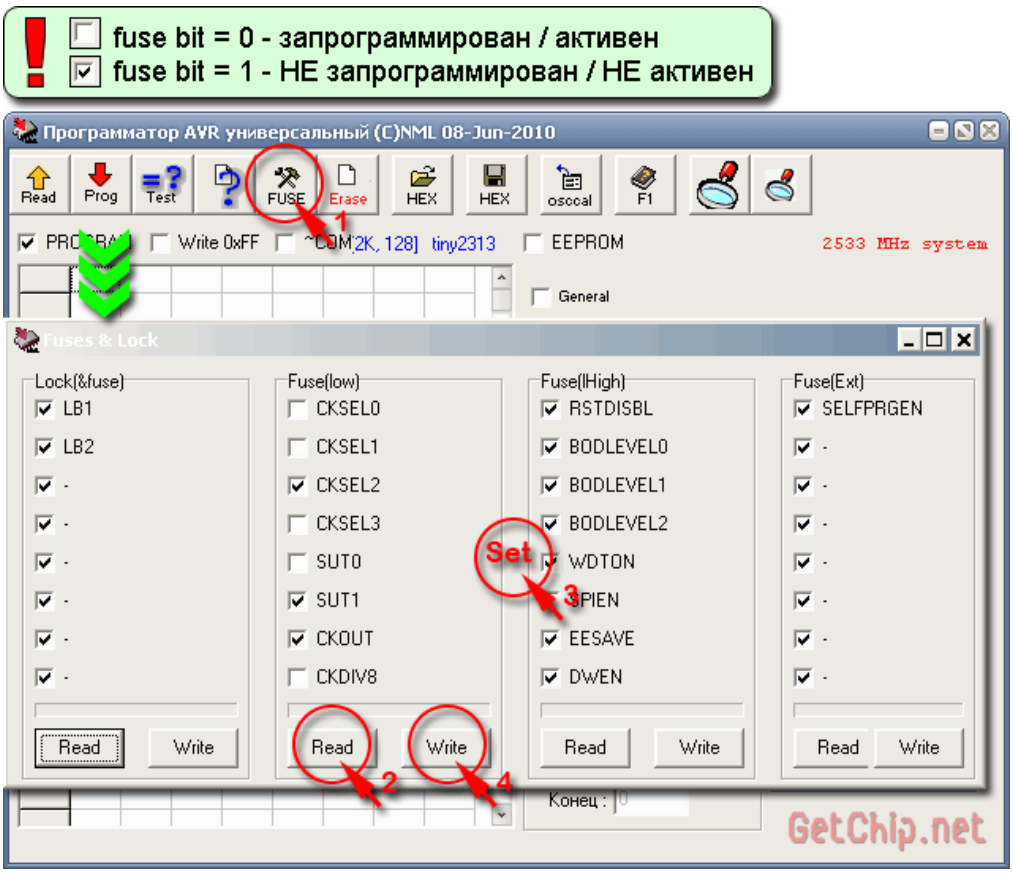
даташиту.



4.2 UniProf.

http://avr.nikolaew.org/progr

Логика установки фьюз аналогична Algorithm Builder.

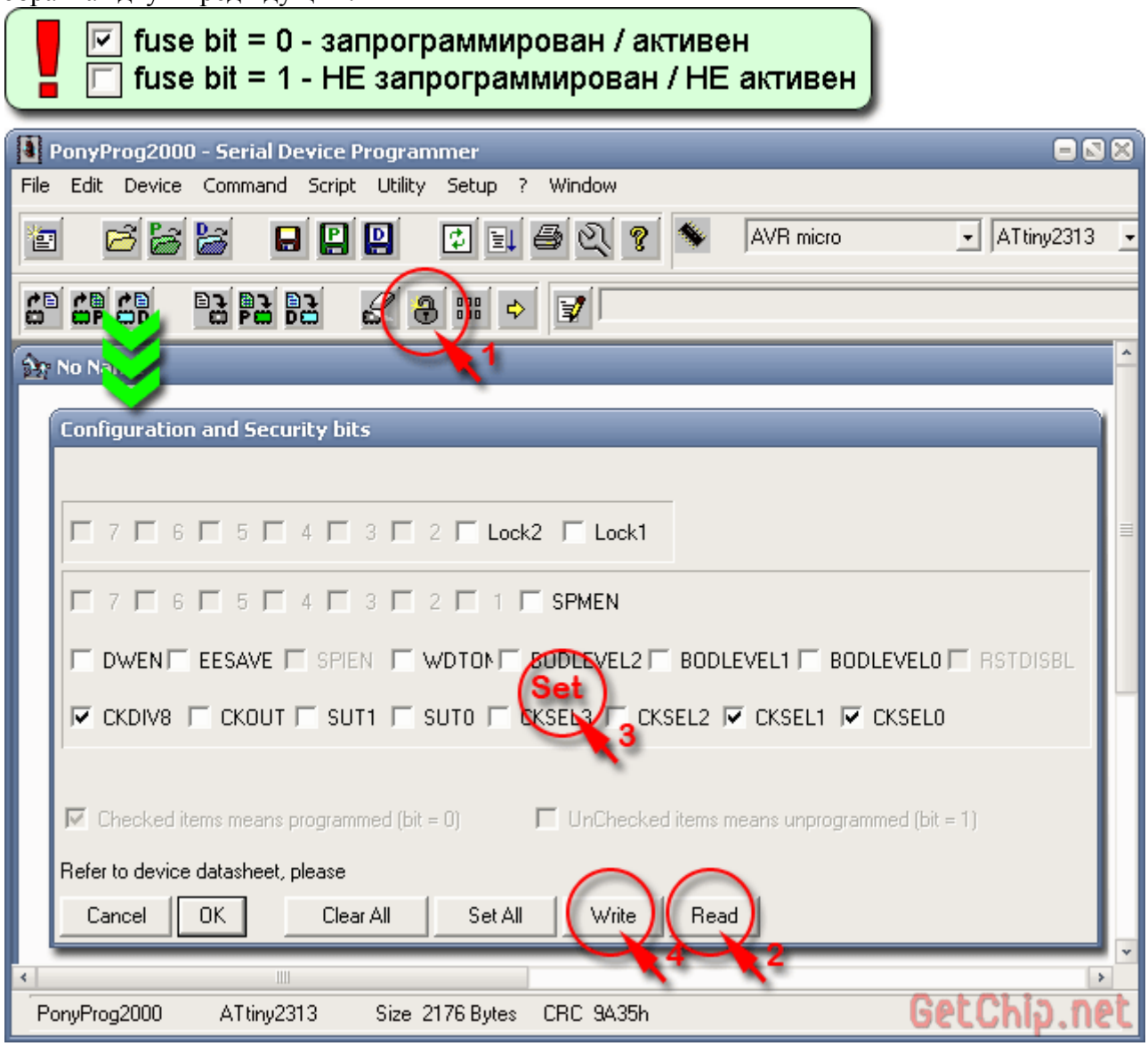


4.3 PonyProg.

http://www.lancos.com/prog.html

Еще одна хорошая программа для программирования микроконтроллеров. Логика

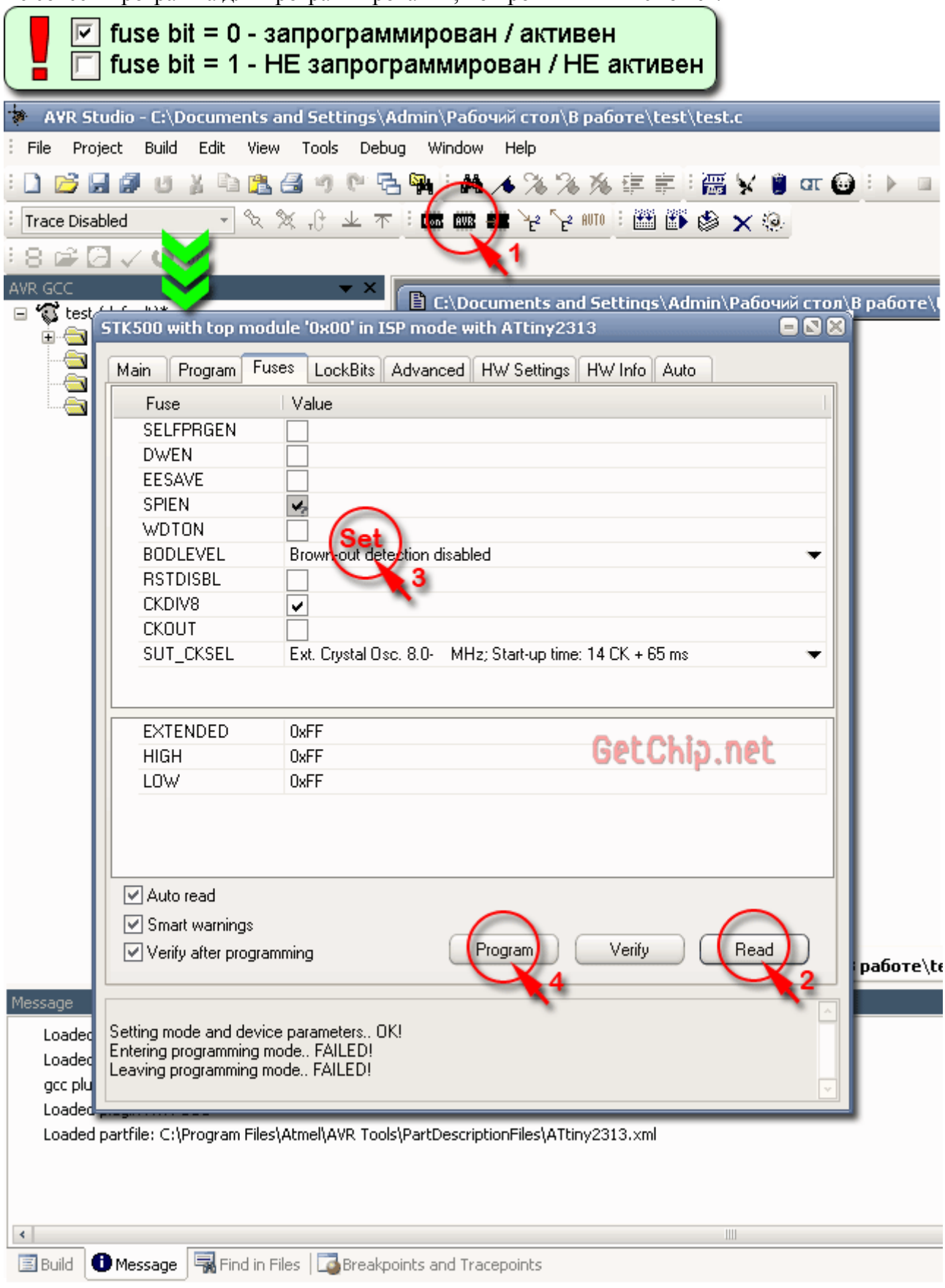
обратная двум пред идущим.



4.4 AVR Studio.

http://www2.atmel.com/

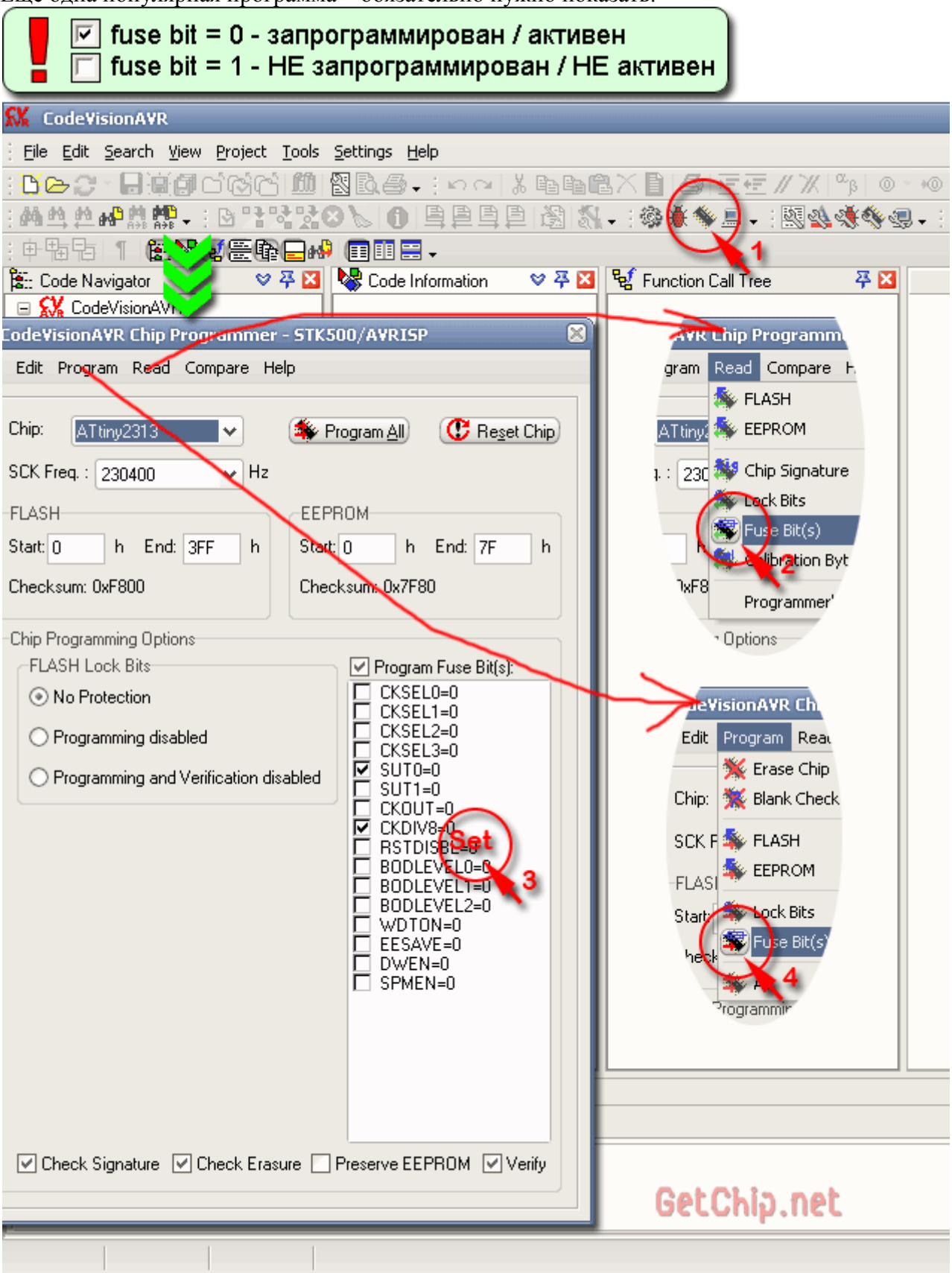
Не совсем программа для программирования, но прошить HEX сможет.



4.5 Code VisionAVR.

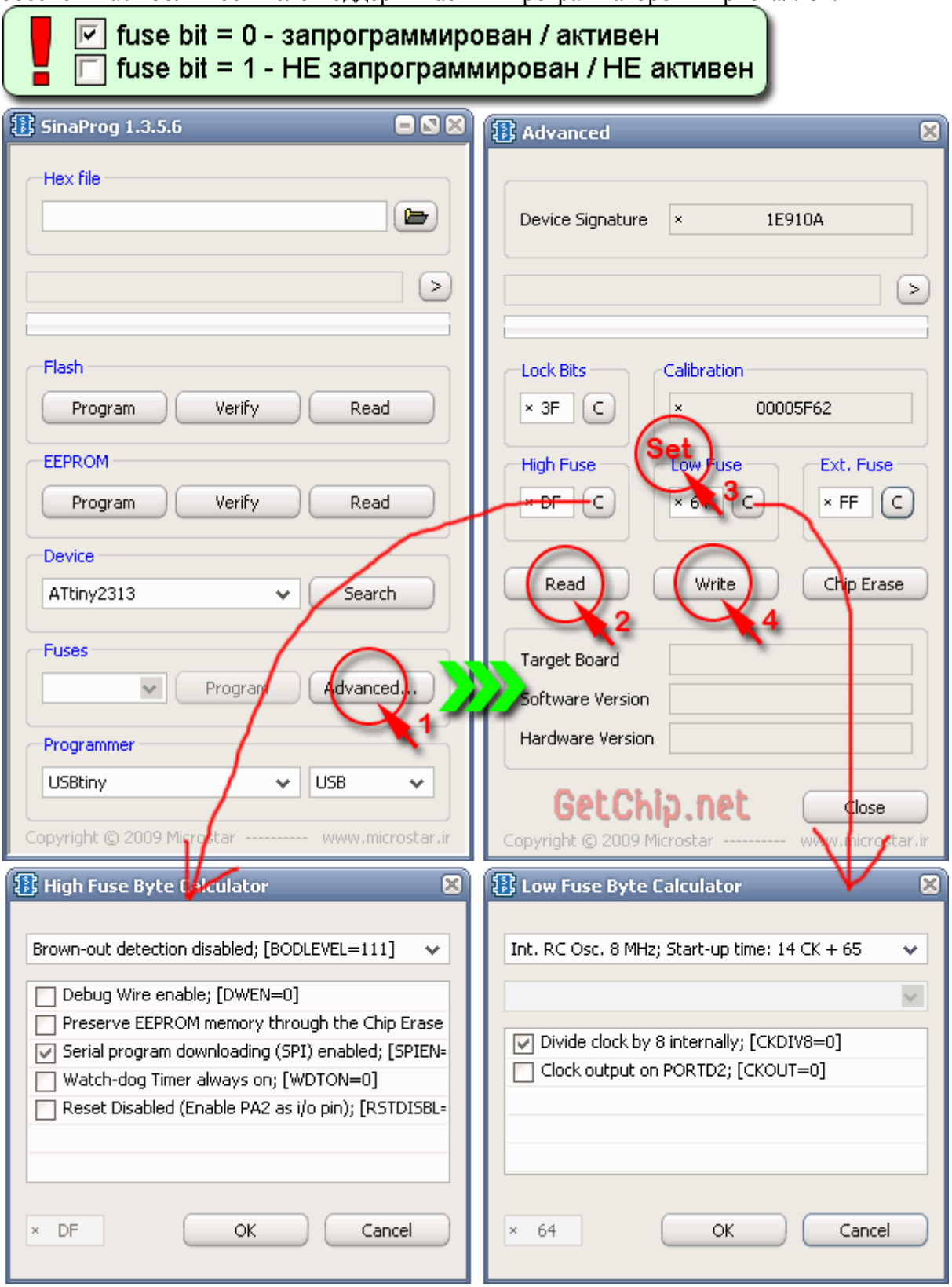
http://www.hpinfotech.ro/html/cvavr.htm

Еще одна популярная программа – обязательно нужно показать.



4.6 SinaProg.

Оболочка для AVRDUDE. Удобная и приятная в управлении программа. AVRDUDE обеспечивает большое число поддерживаемых программаторов и кристаллов.



Я выбирал программы с которыми удобно работать и они доступны и популярны.

5 ФЬЮЗ КАЛЬКУЛЯТОР ДЛЯ AVR.

Если Вам нужна определенная конфигурация микроконтроллера, а изучение даташита ни к чему не приводят (и не удивительно, информация по фьюзам, там старательно размазана по всему документу), есть выход – **Fuse Calculator!**
Фьюз калькулятор – это специальная программа (или on line сервис) призванная помочь в конфигурации микроконтроллера. Как правило они просты и доступны в использовании. По большому счету, каждая среда программирования уже содержит в себе фьюз калькулятор, но есть универсальные с большими возможностями и более удобные. Хотелось бы рассказать об одном из популярных on line калькуляторе – Engbedded Atmel AVR® Fuse Calculator.
http://www.engbedded.com/fusecalc/

Все очень просто – небольшие комментарии на картинке помогут.

Engbedded Atmel AVR® Fuse Calculator

Device selection

Выбираем микроконтроллер

Select the AVR device type you want to configure. When changing this setting, default fuse settings will automatically be applied. Presets (hexadecimal representation of the fuse settings) can be reviewed and even be set in the last form at the bottom of this page.

AVR part name: ATtiny2313 Select (141 parts currently listed)

Feature configuration

Можно выбрать нужные функции ...

This allows easy configuration of your AVR device. All changes will be applied instantly.

Features

Int. RC Osc. 8 MHz; Start-up time: 14 CK + 65 ms; [CKSEL=0100 SUT=10]; default value

- ☐ Clock output on PORTD2; [CKOUT=0]
- ☑ Divide clock by 8 internally; [CKDIV8=0]
- ☐ Reset Disabled (Enable PA2 as i/o pin); [RSTDISBL=0]
- Brown-out detection disabled; [BODLEVEL=111]
- ☐ Watch-dog Timer always on; [WDTON=0]
- ☑ Serial program downloading (SPI) enabled; [SPIEN=0]
- ☐ Preserve EEPROM memory through the Chip Erase cycle; [EESAVE=0]
- ☐ Debug Wire enable; [DWEN=0]
- ☐ Self programming enable; [SELFPRGEN=0]

Apply feature settings

Manual fuse bits configuration

This table allows reviewing and direct editing of the AVR fuse bits. All changes will be applied instantly.

Note: ☐ means unprogrammed (1); ☑ means programmed (0).

... или сразу установить фьюзы

Фьюзы как для PonyProg, CVAVR, AVR Studio.
Для UniProf - инвертировать!

Bit	*Low*	*High*	*Extended*
7	☑ **CKDIV8** Divide clock by 8	☐ **DWEN** debugWIRE Enable	
6	☐ **CKOUT** Clock output	☐ **EESAVE** EEPROM memory is preserved through chip erase	
5	☐ **SUT1** Select start-up time	☑ **SPIEN** Enable Serial programming and Data Downloading	
4	☑ **SUT0** Select start-up time	☐ **WDTON** Watchdog Timer Always On	
3	☑ **CKSEL3** Select Clock Source	☐ **BODLEVEL2** Brown-out Detector trigger level	
2	☐ **CKSEL2** Select Clock Source	☐ **BODLEVEL1** Brown-out Detector trigger level	
1	☑ **CKSEL1** Select Clock Source	☐ **BODLEVEL0** Brown-out Detector trigger level	
0	☑ **CKSEL0** Select Clock Source	☐ **RSTDISBL** External reset disable	☐ **SELFPRGEN** Self Programming Enable

Apply manual fuse bit settings

Current settings

These fields show the actual hexadecimal representation of the fuse settings from above. These are the values you have to program into your AVR device. Optionally, you may fill in the numerical values yourself to preset the configuration to these values. Changes in the value fields are applied instantly (taking away the focus)!

Вот такие инструменты есть для работы с фьюз битами. Выбирайте!

В прошлых статьях я советовал тебе не лезть к этим битам. И на это были свои основания, так как неправильно выставив эти биты ты можешь наглухо заблокировать контроллер для дальнейшей перепрошивки или вообще какого либо использования.

Но без знания этой особенности контроллера далеко не уедешь. Так что распишу все по порядку. У разных версий контроллеров число FUSES разное, какие то могут отсутствовать, но основные есть всегда. Вот по ним и пройдемся.

Конфигурационные биты находятся в особой области памяти и могут быть изменены только с помощью программатора при записи контроллера. Есть старший байт и младший байт. Младший байт обычно отвечает за частоту, а старший за всякие фенечки.

Итак, главное: В Atmel AVR принята следующая нотация: сброшенный в ноль fuse bit считается активным, т.е. включенным.

Пример Бит RSTDSBL, как можно догадаться из названия, это RESET DISABLE. Включил эту опцию и у тебя нога RESET превращается в порт ввода-вывода, но за это ты теряешь возможность перешить контроллер через ISP.

Так вот, чтобы выключить RESET (и получить большое западло с прошивкой в обмен на мелкую подачку в виде дополнительной ножки) в этот бит надо записать 0.

С одной стороны нелогично и криво. Как бы во всем мире принята нотация, что ноль это выключено, а тут, понимаешь, наоборот. С другой стороны, это их контроллер, что хотят то и делают. Один раз запомнить и все. Да и вообще, в электронике часто за сигнал берут ноль.

Однако контроллеры делают электронщики, а прошивающие программы — программисты. Как бы логично. И вот эти программисты взяли и заварили адскую путаницу с галочками. Нет бы им раз и навсегда принять за стандарт, что галочка это 1, а не ВКЛЮЧЕНО (что, напомню, является нулем). И поэтому в одних прошивающих программах галочка означает, что опция включена (в FUSE бит записывается 0),в других, обычно написанных электронщиками, галочка означает единицу. Т.е. с точностью до наоборот.

А что будет если перепутать? А будет ОЧЕНЬ плохо. Контроллер войдет в неправильный режим и может заблокируется наглухо. Т.е. раз прошил и все. Приехал.

Нет, спасти его можно, но для этого тебе потребуются дополнительные ухищрения в виде высоковольтного программатора, JTAG адаптера или генератора тактов. Все зависит от того в какой режим ты загонишь контроллер своими неправильными настройками.

Новичку, обычно, бывает проще сходить и купить новый МК, чем оживить заблокированный. Но не спеши отправлять его в помойку. Пометь и отложи на будущее, разберешься оживишь.

Конфигурация тактового сигнала
По умолчанию все контроллеры AVR (кроме старых серий AT90S2313, AT90S8535 итд)

сконфигурированы так, чтобы работать от внутреннего источника тактов. Т.е. стоить подать на них питание и они начинают работать. Ничего больше и не нужно.

За источник тактов отвечают биты **CKSEL**
Выставив их правильным образом можно выбрать частоту работы контроллера, а также источник тактового сигнала.

- CKSEL3…0 = 0000 — Внешний источник сигнала.

Т.е. на вход XTAL1 подаются прямоугольные импульсы. Такое иногда делают в синхронных системах, когда несколько контроллеров работают от одного генератора.

Техническое отступление
В этот режим часто попадают, когда пытаются выставить контроллер на работу от внешнего кварца (CKSEL=1111), но либо путают нотацию, либо из-за прикола с обратной нотацией битов во всяких извратских прошивающих программах. Раз и кристалл заблокировался. Но, на самом деле, наглухо, с помощью CKSEL, заблокировать кристалл нельзя. Обычно все решается напайкой кварца и запуском от этого кварца. Худшее же что может случиться — потребуется внешний генератор тактов. Который бы оживил кристалл. Делается он за пять минут из любой микросхемы ТТЛ логики, например из К155ЛА3 — схем в инете навалом. Или на таймере 555, либо можно взять второй МК и на нем написать простую программку, дрыгающую ножкой. А если есть осциллограф, то с него можно поиметь сигнал контрольного генератора — его клемма должна быть на любом осциле. Землю осцила на землю контроллера, а выход генератора на XTAL1.

Но что делать если зуд нестерпимый, контроллер залочен, а никакой микросхемы для реанимации под рукой нету? Тут иногда прокатывает метод пальца. Прикол в том, что на тело человека наводится весьма нефиговая наводка частотой примерно 50Гц. Всякий кто хватался за щупы осциллографа руками помнит какие шняги тут же возникают на экране — вот это оно! А почему бы эту наводку не заюзать как тактовый сигнал? Так что припаиваешь к выводу XTAL1 провод, хватаешься за него рукой, и жмешь на чтение или запись контроллера :) Предупреждаю сразу, метод работает через жопу, далеко не с первого раза, читает долго и порой с ошибками, но на перезапись FUSE битов в нужную сторону должно хватить. Пару раз у меня такой фокус получался.

CKSEL3…0 = 0100 – 8 MHz от внутреннего генератора(обычно по умолчанию стоят такие)
Для большинства AVR такая конфигурация CKSEL означает тактовку от внутреннего генератора на 8Мгц, но тут могут быть варианты. Так что в этом случае втыкай внимательно в даташит. В табличку Internal Calibrated RC Oscillator Operating Modes

Иногда нужно иметь внешний тактовый генератор, например, чтобы его можно было подстраивать без вмешательства в прошивку. Для этого можно подключить RC цепочку, как показано на схеме и подсчитать частоту по формуле **f = 1/3RC**, где f будет частотой в герцах, а R и C соответственно сопротивлением резистора и емкостью конденсатора, в омах и фарадах.

- CKSEL3…0 = 0101 – для частот ниже 0.9 MHz
- CKSEL3…0 = 0110 – от 0.9 до 3 MHz
- CKSEL3…0 = 0111 – от 3 до 8 MHz
- CKSEL3…0 = 1000 – от 8 до 12 MHz

Данная табличка справедлива только для ATmega16 у других МК может отличаться. Уточняй в даташите!

Проблема у внутреннего генератора и внешних RC цепочек обычно в нестабильности частоты, а значит если сделать на ней часы, то они будут врать, не сильно, но будут. Поэтому иногда полезно запустить контроллер на кварце, кроме того, только на кварце можно выдать максимум частоты, а значит и производительности проца.

- CKSEL3…0 = 1001 — низкочастотный «часовой» кварц.
  На несколько десятков килогерц.
  Используется в низкоскоростных устройствах, особенно когда требуется точная работа и низкое потребление энергии.

Для обычных кварцев ситуация несколько иная. Тут максимальная частота кварца зависит также и от бита **CKOPT** когда CKOPT = 1 то:

- CKSEL3…0 = 1010 или 1011 — от 0,4 до 0.9 MHz
- CKSEL3…0 = 1100 или 1101 — от 0,9 до 3 MHz
- CKSEL3…0 = 1110 или 1111 – от 3 до 8 MHz (либо от 1 до 16Мгц при CKOPT=0)

А если **CKOPT** равен 0 то при тех же значения CKSEL можно поставить кварц от 1 до 16MHz.

Разумеется, кварц на 16MHz можно поставить только на Мегу без индекса ”L”. (Хотя, как показывает практика, Lку тоже можно неслабо разогнать. У меня ATMega8535L заработала на 16Мгц, но были странные эффекты в работе. Поэтому я не стал так извращаться и разгон снял). Опять же, все выше сказанное в точности соответствует только Меге 16, у других может незначительно отличаться.

Бит **CKOPT** задает размах тактового сигнала. Т.е. амплитуду колебаний на выходе с кварца. Когда CKOPT = 1 то размах маленький, за счет этого достигается меньшее энергопотребление, но снижается устройчивость к помехам, особенно на высоких скоростях (а предельной, судя по таблице выше, вообще достичь нельзя. Точнее запуститься то он может запустится, но вот надежность никто не гарантирует). А вот если CKOPT активизировать, записать в него 0, то размах сигнала сразу же станет от 0 до питания. Что увеличит энергопотребление, но повысит стойкость к помехам, а значит и предельную скорость. При оверклокинге МК тем более надо устанавливать CKOPT в 0.

Также стоит упомянуть бит SCKDIV8 которого нет в Atmega16, но который часто встречается в других контроллерах AVR. Это делитель тактовой частоты. Когда он установлен, т.е. в нуле, то частота выставленная в битах CKSEL0…3 делится на 8, на чем в свое время прилично застрял Длинный, долго пытаясь понять чего это у него западло не работает. Вся прелесть в том, что этот делитель можно отключить программно, записав в регистр CLKPR нужный коэффициент деления, например один. Весь прикол в том, что SCKDIV8 активен по дефолту! Так что внимательней!

Биты SUT задают скорость старта МК после снятия RESET или подачи питания. Величина там меняется от 4ms до 65ms. Мне, за всю практику, пока не довелось эту опцию использовать — незачем. Так что ставлю на максимум 65ms — надежней будет.

Бит **RSTDISBL** способен превратить линию **Reset** в одну из ножек порта, что порой очень нужно когда на какой-нибудь крошечной Tiny не хватает ножек на все задачи, но надо помнить, что если отрубить Reset то автоматически отваливается возможность прошивать контроллер по пяти проводкам. И для перешивки потребуется высоковольтный параллельный программатор, который стоит несколько тысяч и на коленке сделать его проблематично, хотя и возможно.

Второй заподлянский бит это **SPIEN** если его поставить в 1, то у тебя тоже мгновенно отваливается возможность прошивать по простому пути и опять будет нужен параллельный программатор. Впрочем, успокаивает то, что сбросить его через SPI невозможно, по крайней мере в новых AVR (в старых, в AT90S*** было можно)

WDTON отвечает за Собачий таймер, он же Watch Dog. Этот таймер перезагружает процессор если его периодически не сбрасывать – профилактика зависаний. Если WDTON поставить в 0, то собаку нельзя будет выключить вообще.

BODLEVEL и **BODEN** — это режим контроля за напряжением. Дело в том, что при определенном пороге напряжения, ниже критического уровня, контроллер может начать сильно глючить. Самопроизвольно может запортачить, например, EEPROM или еще что откосить. Ну, а ты как думал, не покорми тебя с пару недель — тоже глючить начнешь :)

Так вот, для решения этой проблемы есть у AVR встроенный супервизор питания. Он следит, чтобы напруга была не ниже адекватного уровня. И если напруги не хватает, то просто прижимает RESET и не дает контроллеру стартовать. Вот эти два фуза и рулят этой фичей. **BODEN** включает, а **BODLEVEL** позволяет выбрать критический уровень, один из двух. Какие? Не буду раскрывать, посмотри в даташите (раздел System Control and Reset).

JTAGEN — Включить JTAG. По умолчанию активна. Т.е. JTAG включен. Из-за этого у MEGA16 (а также 32 и прочих, где есть JTAG) нельзя использовать вывода порта C, отвечающие за JTAG. Но зато можно подключать JTAG отладчик и с его помощью лезть контроллеру в мозги.

EESAVE — Защита EEPROM от стирания. Если эту штуку включить, то при полном сбросе МК не будет стерта зона EEPROM. Полезно, например, если в EEPROM записываются какие-либо ценные данные по ходу работы.

BOOTRST — перенос стартового вектора в область бутлоадера. Если эта галочка включена, то МК стартует не с адреса 00000, а с адреса бутсектора и вначале выполняет бутлоадер. Подробней про это было написано в статье про прошивку через лоадер.

BOOTSZ0..1 — группа битов определяющая размер бут сектора. Подробней смотри в даташите. От контроллера к контроллеру они отличаются.

Lock Bits
Это, собственно, и к фузам то отношения не имеет. Это биты защиты. Установка этих битов запрещает чтение из кристалла. Либо флеша, либо ЕЕПРОМА, либо и того и другого сразу. Нужно, только если ты продаешь свои устройства. Чтобы злые конкуренты не слили прошивку и не заказали в китае более 9000 клонов твоего девайса, оставив тебя без штанов. Опасности не представляют. Если ты заблокируешь ими кристалл, то выполни полное стирание и нет проблемы.

Характерной особенностью установленных лок битов является считываемая прошивка — в ней байты идут по порядку. Т.е. 00,01, 02, 03, 04… FF, 00… Видел такую срань? Значит не судьба тебе спереть прошивку — защищена =)

Техника безопасности
И главное правило при работе с FUSE битами — ВНИМАНИЕ, ВНИМАНИЕ и ЕЩЕ РАЗ ВНИМАНИЕ! Не выставляйте никогда FUSE не сверившись с даташитом, даже если срисовываете их из проверенного источника.

Мало ли в какой нотации указал их автор, в прямой или инверсной. Так что если повторяете какую-либо конструкцию, то перед тем как ставить фузы, проверьте то ли вы вообще ставите!

Обязательно разберитесь что означает галочка в прошивающей программе. Ноль или единицу. Включено или выключено! Стандарта нет!!!

Если фуз биты задаются двумя числами — старший и младший биты, то выставляются они как в даташите. Где 0 это включено.