# AVR. Учебный курс. Стартовая инициализация

**AVR. Учебный курс** | 9 Июль 2008 | DI HALT | 54 Comments

**Инициализация памяти**

Мало кто подозревает о том, что при включении в оперативке далеко не всегда все байты равны 0xFF. Они могут, но не обязаны. Равно как и регистры РОН не всегда равны нулю при запуске. Обычно да, все обнулено, но я несколько раз сталкивался со случаями когда после

перезапуска и/или включения-выключения питания, микроконтроллер начинал творить не пойми что. Особнно часто возникает когда питание выключаешь, а потом, спустя некоторое время, пара минут, не больше, включаешь. А всему виной остаточные значения в регистрах.

Итак, возьмите себе за правило после каждого включения, в разделе инициализации, еще даже до инициализации стека, делать зануление памяти и очистку всех регистров. Разумеется делается это все в цикле. Вот примерный вариант кода:

```
1   RAM_Flush:      LDI     ZL,Low(SRAM_START)      ; Адрес начала ОЗУ в индекс
2                   LDI     ZH,High(SRAM_START)
3                   CLR     R16                     ; Очищаем R16
4   Flush:          ST      Z+,R16                  ; Сохраняем 0 в ячейку памяти
5                   CPI     ZH,High(RAMEND+1)       ; Достигли конца оперативки?
6                   BRNE    Flush                   ; Нет? Крутимся дальше!
7
8                   CPI     ZL,Low(RAMEND+1)        ; А младший байт достиг конца?
9                   BRNE    Flush
10
11                  CLR     ZL                      ; Очищаем индекс
12                  CLR     ZH
```

Поскольку адрес оперативки у нас двубайтный, то мы вначале смотрим, чтобы старший байт совпал с концом, а потом добиваем оставшиеся 255 байт в младшем байте адреса.
Далее убиваем все регистры от первого до последнего. Все, контроллер готов к работе.

```
1                   LDI     ZL, 30          ; Адрес самого старшего регистра
2                   CLR     ZH              ; А тут у нас будет ноль
3                   DEC     ZL              ; Уменьшая адрес
```

```
4                   ST      Z, ZH           ; Записываем в регистр 0
5                   BRNE    PC-2            ; Пока не перебрали все не успокоились
```

За процедурку зануления регистров спасибо Testicq

Либо значения сразу же инициализируются нужными величинами. Но, обычно, я от нуля всегда пляшу. Поэтому зануляю все.

**З.Ы.**
Кстати, о оперативке. Нашел я недавно планку оперативной памяти на 1килобайт, древнюю как говно мамонта, еще на **ферромагнитных кольцах.**

Assembler Мелочи Программирование

## 54 thoughts on “AVR. Учебный курс. Стартовая инициализация”

**nwanomaly**
30 Сентябрь 2008 в 21:17
я ещё помню, когда смотрел по линку на программатор на сайт Николаева, то тоже видел там раздел про «тонкости». про использование символических имён, работу с флагами и компаратор в режиме пониженно потребления ) мелочи, а знать надо

---

**Ivan A-R**
30 Сентябрь 2008 в 21:23
Такой метод инициализации катит только если у тебя переменные инициируются нулями. Но такое не всегда бывает.

Как вариант, хранить во флеше слепок начальных значений переменных и копировать их. Или просто не забывать инициировать их по мере надобности.

А чистить регистры и место для стека считаю пустой тратой тактов =) Регистры ты так или иначе будешь инициировать по ходу дела, а стек у тебя изначально нулевой и при росте перекроет любые левые значения.

---

**★ DI HALT**

30 Сентябрь 2008 в 21:32

Ну так а что мешает потом забить переменные нужынми значениями. Все проще чем слепок памяти держать.

По поводу регистров. Да вот вчера трахался с тем, что у меня на Tiny2323 после вырубания питания значение в регистре торчало по несколько минут!!! Врубаю МК — на тебе — мигать начинать самопроизвольно. Мигание у меня зависило от состояния регистра R20.

---

**Ivan A-R**

30 Сентябрь 2008 в 22:43

Сам подход, что ты полагаешь, что у тебя в регистрах есть значения по умолчанию — порочен. Сейчас ты чистишь всё что можно после сброса и это работает. Но когда тебе потребуется в ходе работы переинициализировать какой-то модуль, ты опять столкнёшься с тем, что в регистрах у тебя мусор. Каждый модуль должен сам инициализировать то что ему нужно.
Я у себя для каждого модуля пишу процедуру void xxxInit(void), которая полностью готовит модуль к работе. И это при том, что ‘C’ умеет инициировать переменные при старте.

---

**★ DI HALT**

30 Сентябрь 2008 в 22:52

Не забывай, что я пишу на асме у меня часто регистры юзаются как статичные переменные. Т.е. очень часто какой либо регистр юзается только для одной конкретной цели.

---

**Ivan A-R**

30 Сентябрь 2008 в 23:07

Знаю. Я тоже пишу на асме. Но это тебя не освобождает от инициализации переменных =)

Тут фишка такая, когда ты используешь умолчальные инициализации, ты тереяшь в понятности кода.

Сделай инициализацию явно, и уже потом, на этапе оптимизации сможешь эту строчку закомментить с комментарием «// а оно и так уже такое». В дальнейшем избавишь себя от головной боли.

---

**SWG**

1 Октябрь 2008 в 0:47

Задание начальных значений переменным должно сидеть в подсознании, на чем бы не писал. В этом отношении хороши компиляторы семейства Паскалей. Например, в Дельфи компилятор постоянно предупреждает, если какая-то переменная может перед использованием иметь неопределенное значение, (например, значение задается в куске кода или цикле, который может в некоторых случаях не выполняться. Си в этом отношении не так строг, много чего пропускает, что потом сносит башку при отладке.
Вчера я начал писать программку уже для контроллера бамперов своего робота, и в самом начале, в процедуре инициализации, сразу поставил кусок (В ходовом контроллере тоже):

```
if PCON.1 = 0 then // Если был сброс по питанию, Очищаем память!
begin
FSR := $20; //Начало ОЗУ
while FSR < $40 do // Первые 32 байта.
begin
INDF := 0; // Очищаем память!
Inc(FSR);
end;
PCON.1 := 1;
end;
```

Т.Е. по холодному старту очищаю область памяти, используемую пока в программе, далее в программе тем переменным, которые должны быть отличны от нуля, присваиваются конкретные значения. Сначала хотел сделать этот кусок на асме, но потом посмотрел, как его расписал компилятор, и решил оставить на Паскале. Вот этот кусок после компиляции:

```
;ROBO_SWG_2.ppas,81 :: begin
;ROBO_SWG_2.ppas,82 :: if PCON.1 = 0 then // Если был сброс по питанию, Очищаем память!
$00FA $1303 BCF STATUS, RP1
$00FB $1683 BSF STATUS, RP0
$00FC $080E MOVF PCON, 0
$00FD $00F1 MOVWF STACK_1
$00FE $3000 MOVLW 0
$00FF $18F1 BTFSC STACK_1, 1
$0100 $3001 MOVLW 1
$0101 $00F1 MOVWF STACK_1
$0102 $0871 MOVF STACK_1, 0
$0103 $3A00 XORLW 0
$0104 $1D03 BTFSS STATUS, Z
$0105 $2910 GOTO ROBO_SWG_2_L_9
$0106 $ ROBO_SWG_2_L_8:
;ROBO_SWG_2.ppas,84 :: FSR := $20; //Начало ОЗУ
$0106 $3020 MOVLW 32
$0107 $0084 MOVWF FSR
;ROBO_SWG_2.ppas,85 :: while FSR < $40 do // Первые 32 байта.
$0108 $ ROBO_SWG_2_L_12:
$0108 $3040 MOVLW 64
$0109 $0204 SUBWF FSR, 0
$010A $1803 BTFSC STATUS, C
$010B $290F GOTO ROBO_SWG_2_L_13
;ROBO_SWG_2.ppas,87 :: INDF := 0; // Очищаем память!
```

```
$010C $0180 CLRF INDF, 1
;ROBO_SWG_2.ppas,88 :: Inc(FSR);
$010D $0A84 INCF FSR, 1
;ROBO_SWG_2.ppas,89 :: end;
$010E $2908 GOTO ROBO_SWG_2_L_12
$010F $ ROBO_SWG_2_L_13:
;ROBO_SWG_2.ppas,90 :: PCON.1 := 1;
$010F $ ROBO_SWG_2_L_16:
$010F $148E BSF PCON, 1
$0110 $ ROBO_SWG_2_L_17:
;ROBO_SWG_2.ppas,91 :: end;
```

Всего 23 байта. Можно сделать меньше, но зачем?
При других видах сброса (например, по сторожевому таймеру) память пока не очищаю, (вдруг какие-то данные понадобятся), только задаю значения основным переменным.

---

**nwanomaly**

1 Октябрь 2008 в 1:12

я больше программлю под обычные писишки. на сях в основном.
про инициализацию в общем могу сказать так. вначале надо изучать компилятор. как и что он устанавливает при определении переменной. изучить проц — обнуляется ли в нём всё при старте?

потом все глобальные переменные определить — как надо. все те, что в функциях — смотреть внимательно.

я конечно понимаю, что тут не бывает — когда в массив 1000 на 1000 забивают мемкопи инициализирующий )

вообщем, инициализация тоже имеет в себе тонкости.

---

**SWG**

1 Октябрь 2008 в 1:54

В микроконтроллерах при сбросе или холодном старте обычно задается наиболее безопасная конфигурация периферии в спецрегистрах, регистры же общего назначения и оперативка никогда не очищаются, по крайней мере в массовых типах микроконтроллеров. (Среди тех PIC, ATMEL и INTEL, про которые читал или использовал сам, таких не попадалось). Так что «Спасение утопающих — дело рук самих утопающих». Просто должно войти в привычку не пускать ничего на самотек, все определять заранее. Сразу отпадет куча проблем, а высвободившееся при отладке время можно будет направить на повышение «интелекта» самой программы.

---

**Cluster**

1 Октябрь 2008 в 11:19

А разьве прерывание таймера не выполнится после выхода из прерывания INT0?

---

**★ DI HALT**

1 Октябрь 2008 в 12:34

Выполнится, но мне надо было прерывание таймера внутри обработчика INT0

---

**Cluster**

1 Октябрь 2008 в 12:44

Слабо представляю зачем это нужно. Сколько по времени у тебя выполнялся обработчик INT0?

---

**★ DI HALT**

1 Октябрь 2008 в 13:04

На обработчик INT0 у меня повешен альтернативный режим работы девайса. Так что он перехватывает на себя управление и держит его до тех пор пока не отработает всю логику.

**SWG**

1 Октябрь 2008 в 16:42

Вот тут как раз бы и пригодились многоуровневые приоритетные прерывания. Хорошая была штука! При работе в реальном времени с информацией, которую специально для контроллера никто повторять не будет, они многое очень сильно упрощали. Например, в одной из разработок, контролировавшей процент искажений одновременно в 4 телеграфных каналах, в диалоговом режиме, с одновременным накоплением и выдачей результатов, и обслуживанием еще управляющего канала — аппарата «диспетчера», чтобы уложиться в требуемую точность в +-1%, требовалось для худшего случая (если во время обработки прерывания от одного канала поступали прерывания от всех остальных 4), уложиться до 200мксек. Мне удалось обрабатывать самые худшие случаи максимум за 160! (на 580 процессоре с контроллером прерываний 580ВН59). Правда, пришлось поддержать аппаратно простенькой схемкой (на 555ЛП5), сравнивавшей текущее состояние с предыдущим, которое обработчик прерывания обновлял в специальном регистре, и также формировавшей запрос прерывания при каждом изменении полярности в любом из каналов. Мерялось смещение каждого перехода в каждом знаке, с определением максимального отдельно по + и по — для каждого из каналов, с пересчетом в проценты. Тот контроллер до сих пор в работе, правда, прошивку в 573РФ5 пришлось один раз обновить (заряд стек). Сделал я его году в 86, а программа была около 3600 байт. Всего на плате могло быть установлено до 8кб(4шт РФ5 по 2к). ОЗУ было 4 кб.

---

**graskittism**

18 Октябрь 2008 в 21:57

Полностью согласен с мнением автора.

---

**testicq**

17 Февраль 2009 в 0:28

Может кому пригодиться…

Вот мои 10 байт кода, обнуляющие все регистры R00-R31:

```
        LDI     ZL, 30    ; +----------------------+
        CLR     ZH        ; |                      |
        DEC     ZL        ; | Очистка РОН (R00-R31) |
        ST      Z, ZH     ; |    [10 байт кода]     |
        BRNE    PC-2      ; +----------------------+
```

P.S. Работает на любых! МК AVR.
P.P.S. Интересно — можно ли сделать еще короче?

---

**★ DI HALT**

21 Апрель 2010 в 2:32

Классно. Но я почему-то был свято убежден, что данный метод не работает. То ли у меня студия глюкнула и не показывала изменения регистров то ли еще что. Щас и не вспомню, но почему то тогда у меня с регистрами цикл не прокатил. Щас проверил — все ок.

---

**dima_m**

13 Июнь 2010 в 11:19

Мне пригодилось, уже применяю, проверил, работает. Круто спасибо.

---

**phantom lord**

29 Март 2011 в 9:39

Чё-то я не понимаю, как такое работает. Тут в одной строке две команды: BRNE и вычитание PC-2. Да и вычитание как-то не по-ассемблерному выглядит.
Да и вообще, команда BRNE k выполняет запись PC <- PC + k + 1. То есть для перехода на две строчки выше нужно записать BRNE -3.

Буду благодарен за пояснения :-)

**AlexandrBuryakov**

19 Август 2012 в 21:54

Может «pc-2» мы для компилятора пишем, а он сам уже подставляет так как нужно?

**★ DI HALT**

19 Август 2012 в 22:32

Именно так. PC это Programm Counter Адрес текущей инструкции в памяти программ.

**Andriy**

9 Апрель 2010 в 0:47

Гонял код чистки SRAM, и заметил что она чистится не до конца, память по адресу RAMEND остается неизменной.
По моему это связано этим:
ST Z+,R16 // в Z=RAMEND-1, ячейку по адресу Z зачистили и сделали инкримент Z,
…
CPI ZL,Low(RAMEND) // Z=RAMEND, проверили и опа вышли из цикла, а ячейку с адресом
// RAMEND не почистили

правильно будет так
CPI ZL,Low(RAMEND+1)

**★ DI HALT**

9 Апрель 2010 в 1:24

Хм, действительно…

**http://obstinatus.myopenid.com/**

17 Декабрь 2010 в 13:47

Несущественное замечание.
DI_HALT, у тебя в примере «плюсединица» лишняя, и в примере товарища Andriy её там нет.

```
CPI ZH,High(RAMEND+1) ; а правильнее High(RAMEND)
BRNE Flush ; хотя результат всеравно один и тот же
CPI ZL,Low(RAMEND+1) ;)
BRNE Flush
```

а то я, не читая комментов, не понял сначала зачем она там взялась, пока в VMLABe не поковырял.

**Denzell**

11 Сентябрь 2010 в 15:25

а есть ли смысл заюзать этот код как подпрограмму? или чистка — обнуление всего нужны только при старте?

**★ DI HALT**

11 Сентябрь 2010 в 17:18

Только при старте.

**ProhodivshiyMimo**

25 Октябрь 2010 в 6:28

>Структура же программы при этом следующая:
>…
>Инициализация памяти

>Инициализация стека
>…

>Инициализация ядра. Память, стек, регистры:
[code]
Reset: LDI R16,Low(RAMEND) ; Инициализация стека
OUT SPL,R16 ; Обязательно!!!

LDI R16,High(RAMEND)
OUT SPH,R16

; Start coreinit.inc
RAM_Flush: LDI ZL,Low(SRAM_START) ; Адрес начала ОЗУ в индекс
LDI ZH,High(SRAM_START)
CLR R16 ; Очищаем R16
Flush: ST Z+,R16 ; Сохраняем 0 в ячейку памяти
CPI ZH,High(RAMEND) ; Достигли конца оперативки?
BRNE Flush ; Нет? Крутимся дальше!
…
[/code]

В итоге что же инициализировать в первую очередь: память или стек?

---

**★ DI HALT**

25 Октябрь 2010 в 13:09

В первую очередь память, а потом уже стек. Иначе flush цикл сотрет стековые данные. С другой стороны, если никаких вызовов и переходов не было (прога то только запустилась) То и стек пуст и стирать там нечего.

---

**XanderEVG**

16 Март 2013 в 16:53

некоторые неопытные новички вызывают RAM_Flush как функцию. т е через RCALL.
только выглядит функция почему то вот так у меня:

```
RAM_Flush: LDI ZL,Low(SRAM_START) ; Адрес начала ОЗУ в индекс
LDI ZH,High(SRAM_START)
CLR R16 ; Очищаем R16
Flush: ST Z+,R16 ; Сохраняем 0 в ячейку памяти
CPI ZH,High(RAMEND) ; Достигли конца оперативки?
BRNE Flush ; Нет? Крутимся дальше!

CPI ZL,Low(RAMEND) ; А младший байт достиг конца?
BRNE Flush

CLR ZL ; Очищаем индекс
CLR ZH
```

и работает эта конструкция, пока перед вызовом функции не накопится кода столько, чтобы указатель в стеке стал двубайтным(много инклюдов-обработчиков прерываний сверху). тогда трабл, да такой что хрен найдешь)))

думаю безопаснее все же сделать вот так:

```
CPI ZH,High(RAMEND-1)
BRNE Flush
```

и пусть последние два байта не очищаются

**v.m.s.**

17 Февраль 2011 в 5:36

Еще не понял в чем прикол, но в коде для инициализации оперативки строки:
CPI ZL,Low(RAMEND+1) ; А младший байт достиг конца?
BRNE Flush

походу — лишние. Закавычил их, и оперативка всеравно очищается (заполняется 00 или FF) от первого до последнего байта.
А вы поняли в чем прикол?

**v.m.s.**

17 Февраль 2011 в 5:41

прогонял код для ATмеги32M1.

**v.m.s.**

17 Февраль 2011 в 6:02

Так, пойдука я спать. А то с закавыченной строкой

LDI ZL,Low(SRAM_START) ; Адрес начала ОЗУ в индекс

код всеравно работает. ОЗУ инициализируется. :)

**alkinoy**

16 Март 2011 в 20:49

Обратил внимание на закругленные углы дорожек на плате памяти. Боялись помех? К тому же они, похоже, посеребренные…

**Ivan A-R**

16 Март 2011 в 21:24

Просто от руки тяжело прямые углы рисовать =)

---

**buffer2008**

23 Декабрь 2011 в 23:54

Подскажите пожалуйста зачем в строке ST Z+,R16 возле Z стоит + что происходит при этом?
CPI ZH,High(RAMEND+1)
CPI ZL,Low(RAMEND+1) почему RAMEND+1, что дает такая комбинация?
Хотелось бы знать эти нюансы.
И еще, при симуляции в AVRStudio этого участка программы:
1 Flush:
2 ST Z+,R16 ; Сохраняем 0 в ячейку памяти
3 CPI ZH,High(RAMEND+1) ; Достигли конца оперативки?
4 BRNE Flush ; Нет? Крутимся дальше!
5 CPI ZL,Low(RAMEND+1) ; А младший байт достиг конца?
6 BRNE Flush
она очень надолго зависает на участке 2-4 строки, как раз на 30-м РОН.
Извините за глупые, на первый взгляд, вопросы — я только начал вникать в AVR контроллеры. Для меня не все еще очевидно.

---

**★ DI HALT**

24 Декабрь 2011 в 0:05

1. плюс это увеличение Z на единицу после выполнения команды. Погляди в даташите список всех команд.

2. RAMEND это константа указывающая на конец памяти конкретного контроллера. Прописана в его *def.inc файле.

3. Ну так в цикле же надо очистить всю память. Вот студия и тупит пока все это перелопатит. Во время отладки можно закомментить, чтобы не мешала.

---

**buffer2008**

24 Декабрь 2011 в 2:43

Благодарю за указанное направление! Начало прояснятся. Я то думал чо оно на этом куске кода зациклилось…
Пощел подчитаю внимательно Даташит и аппноты :-))

---

**Andrey41**

19 Февраль 2012 в 22:35

Я так думаю, что участок кода
Flush: ST Z+,R16 ; Сохраняем 0 в ячейку памяти
CPI ZH,High(RAMEND+1) ; Достигли конца оперативки?
BRNE Flush ; Нет? Крутимся дальше!

CPI ZL,Low(RAMEND+1) ; А младший байт достиг конца?
BRNE Flush

CLR ZL ; Очищаем индекс
CLR ZH
будет работать только для тех МК, у которых RAMEND выражается однобайтовым числом, т.к.
в этом случае старший байт будет равен 0. В таком случае в ZH загрузится 0 и программа будет сравнивать только младший байт. А всё это потому, что к двухбайтовому числу 1 так не прибавляется. А такой код будет работать

CPI ZH,High(RAMEND) ; Достигли конца оперативки?
BRNE Flush: ; Нет? Крутимся дальше!
CPI ZL,Low(RAMEND) ; А младший байт достиг конца?

BRNE Flush
RJMP PC+3
Flush: ST Z+,R16
RJMP PC-6
CLR ZL ; Очищаем индекс
CLR ZH

---

**★ DI HALT**

19 Февраль 2012 в 22:40

Z+ инкрементирует сразу регистровую пару, а не только ZL

---

**Andrey41**

19 Февраль 2012 в 22:53

Вы меня не поняли. На пример у мега 8535 RAMEND=25F
RAMEND+1=260. Старший байт будет=$2, а мл.=$60
А у Вас 2+1=3 и 5F+1=60 в итоге RAMEND=360

---

**Andrey41**

19 Февраль 2012 в 23:00

Извините , был не прав.

---

**AlexandrBuryakov**

19 Август 2012 в 21:27

этот кусок кода тоже ОЗУ чистит?

```
;очистка стековой памяти
ldi r16,low(RAMEND)
out SPL,r16
.if (RAMEND)>=0x0100
ldi r16,high(RAMEND)
out SPH,r16
clr r16
.endif

…
```

И что такое листинг?

---

**★ DI HALT**

19 Август 2012 в 21:32

Это загрузка указателя стека.

Листинг — это код программы со всеми адресами.

---

**AlexandrBuryakov**

19 Август 2012 в 21:40

Понятно, благодарю.
При отключении листинга в AVR Studio я просто отключаю просчёт участка кода на ошибки, но в компиляции он принимает участие, так?
И второй вопрос, я в директивах почти нигде не видел такого:
.if
.else

.endif
На что они влияют?

---

**★ DI HALT**

19 Август 2012 в 21:52

Да, листинг это просто для того, чтобы потом поглядеть на результат. Например, когда используются сложные макросы с кучей параметров. А в листниге все макросы развернуты. Можно найти ошибки в макросах. Иногда в нем удобно адреса смотреть.

Это директивы условной компиляции. Если участо заключенный в IF выполняет условие указанное в скобках, то он компилируется, если нет — пропускается.
В AVR существует два ассемблера avrassembler и avrassembler2 второй помощней немного. Описание всех директив, ключей и прочая дока на компилятор доступна в справке которая вызывается по F1 в avrstudio 4 (как в 5й ил 6й хз, может тоже есть)

---

**AlexandrBuryakov**

19 Август 2012 в 21:43

И какие ещё существуют директивы?

---

**AlexandrBuryakov**

19 Август 2012 в 21:48

И плюс ко всему, только сейчас предположил, этот кусок кода сам определяет нужно ли использовать старший байт в инициализации или нет и решает это компилятор?

---

**★ DI HALT**

19 Август 2012 в 21:53

Да

**Alexm**

28 Октябрь 2014 в 18:27

Ребята, подскажите, где определена символическая метка SRAM_START, которую мы используем в программах этого урока? Я не нашёл её определения в подключаемом файле m16def.inc, в отличие от метки RAMEND.

**★ DI HALT**

29 Октябрь 2014 в 1:21

А в даташите посмотри. Там сразу же после пачки РОН регистров оно идет. И для всех AVR одинаковая.

**★ DI HALT**

29 Октябрь 2014 в 1:30

А вообще это стандартная метка, она определена была в inc файле. Ну по крайней мере на тот момент когда писалась эта статья.

**★ DI HALT**

29 Октябрь 2014 в 1:46

А, понял в чем твоя проблема. Ты юзаешь AvrAssembler первой ревизии, а я AvrAssembler2

c:\Program Files (x86)\Atmel\AVR Tools\AvrAssembler2\Appnotes\
вот тут в файлах def.inc все это есть.

**Alexm**

1 Ноябрь 2014 в 4:07

Да, действительно, спасибо!
Выходит, что когда мы пишем в начале файла строчку

include «m16def.inc»,
то мы подключаем .inc файл из папки AvrAssembler2\Appnotes, а не из папки AvrAssembler\Appnotes?

---

**★ DI HALT**

1 Ноябрь 2014 в 10:58

Я не уверен, но возможно это зависит от выбора симулятора (1 или 2). Хотя может атмел выкинул на свалку свой первый компилер и оставил только второй.