AVR. Учебный курс. Подпрограммы и прерывания

AVR. Учебный курс | 7 Июль 2008 | DI HALT | 233 Comments

Подпрограммы

Когда один и тот же участок кода часто повторяется, то разумно как то его вынести и использовать многократно. Это дает просто колоссальный выйгрыш по обьему кода и удобству программирования.

Вот, например, кусок кода, передающий в регистр UDR байты с некоторой выдержкой, выдержка делается за счет вращения бесконечного цикла:

```
1         .CSEG
2         LDI R16,Low(RAMEND)     ; Инициализация стека
3         OUT SPL,R16             ; Обязательно!!!
4
5         LDI R16,High(RAMEND)
6         OUT SPH,R16
7
8         .equ    Byte    = 50
9         .equ    Delay   = 20
10
11        LDI     R16,Byte        ; Загрузили значение
12  Start: OUT    UDR,R16         ; Выдали его в порт
13
14        LDI     R17,Delay       ; Загрузили длительность задержки
15  M1:   DEC     R17             ; Уменьшили на 1
16        NOP                     ; Пустая операция
17        BRNE    M1              ; Длительность не равна 0? Переход если не 0
18
19        OUT     UDR,R16         ; Выдали значение в порт
20
21        LDI     R17,Delay       ; Аналогично
22  M2:   DEC     R17
23        NOP
24        BRNE    M2
25
26        OUT     UDR,R16
```

```
27
28          LDI     R17,Delay
29  M3:     DEC     R17
30          NOP
31          BRNE    M3
32
33          RJMP    Start           ; Зациклим программу
```

Сразу напрашивается повторяющийся участок кода вынести за скобки.

```
1
2           LDI     R17,Delay
3   M2:     DEC     R17
4           NOP
            BRNE    M2
```

Для этих целей есть группа команд перехода к подпрограмме CALL (ICALL, RCALL, CALL)
И команда возврата из подпрограммы RET

В результате получается такой код:

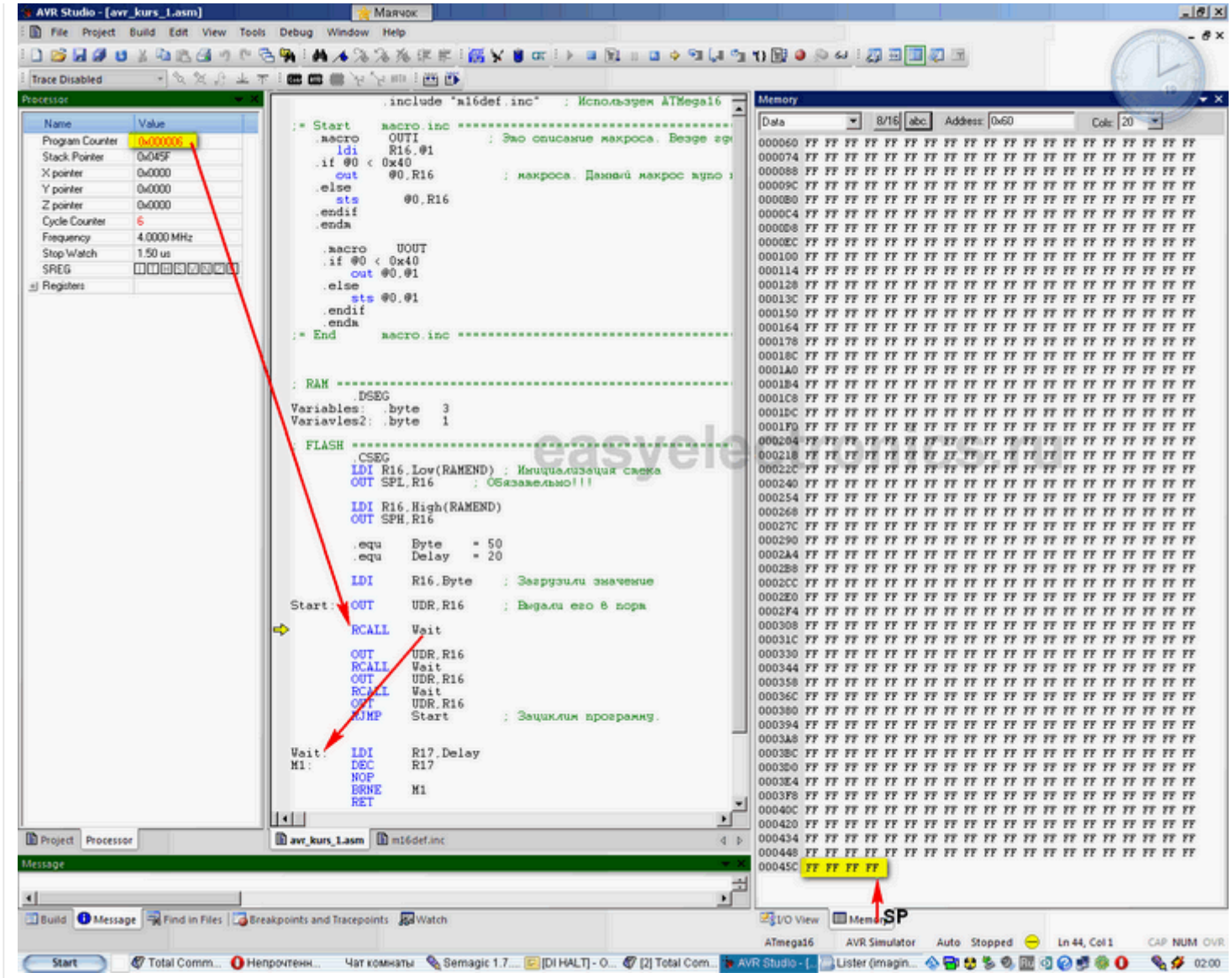
```
1           .CSEG
2           LDI R16,Low(RAMEND)     ; Инициализация стека
3           OUT SPL,R16             ; Обязательно!!!
4
```

```
5           LDI R16,High(RAMEND)
6           OUT SPH,R16
7
8           .equ    Byte    = 50
9           .equ    Delay   = 20
10
11          LDI     R16,Byte        ; Загрузили значение
12  Start:  OUT     UDR,R16         ; Выдали его в порт
13
14          RCALL   Wait
15
16          OUT     UDR,R16
17          RCALL   Wait
18          OUT     UDR,R16
19          RCALL   Wait
20          OUT     UDR,R16
21          RCALL   Wait
22          RJMP    Start           ; Зациклим программу.
23
24
25  Wait:   LDI     R17,Delay
26  M1:     DEC     R17
27          NOP
28          BRNE    M1
29          RET
```

Как видишь, программа резко сократилась в размерах. Теперь скопируй это в студию, скомпилируй и запусти на трассировку. Я хочу показать как работает команда RCALL и RET и при чем тут стек.

Вначале программа, как обычно, инициализирует стек. Потом загружает наши данные в регистры R16 и выдает первый байт в UDR… А потом по команде RCALL перейдет по адресу который мы присвоили нашей процедуре, поставив метку Wait в ее начале. Это понятно и логично, гораздо интересней то, что произойдет в этот момент со стеком.

До выполнения RCALL

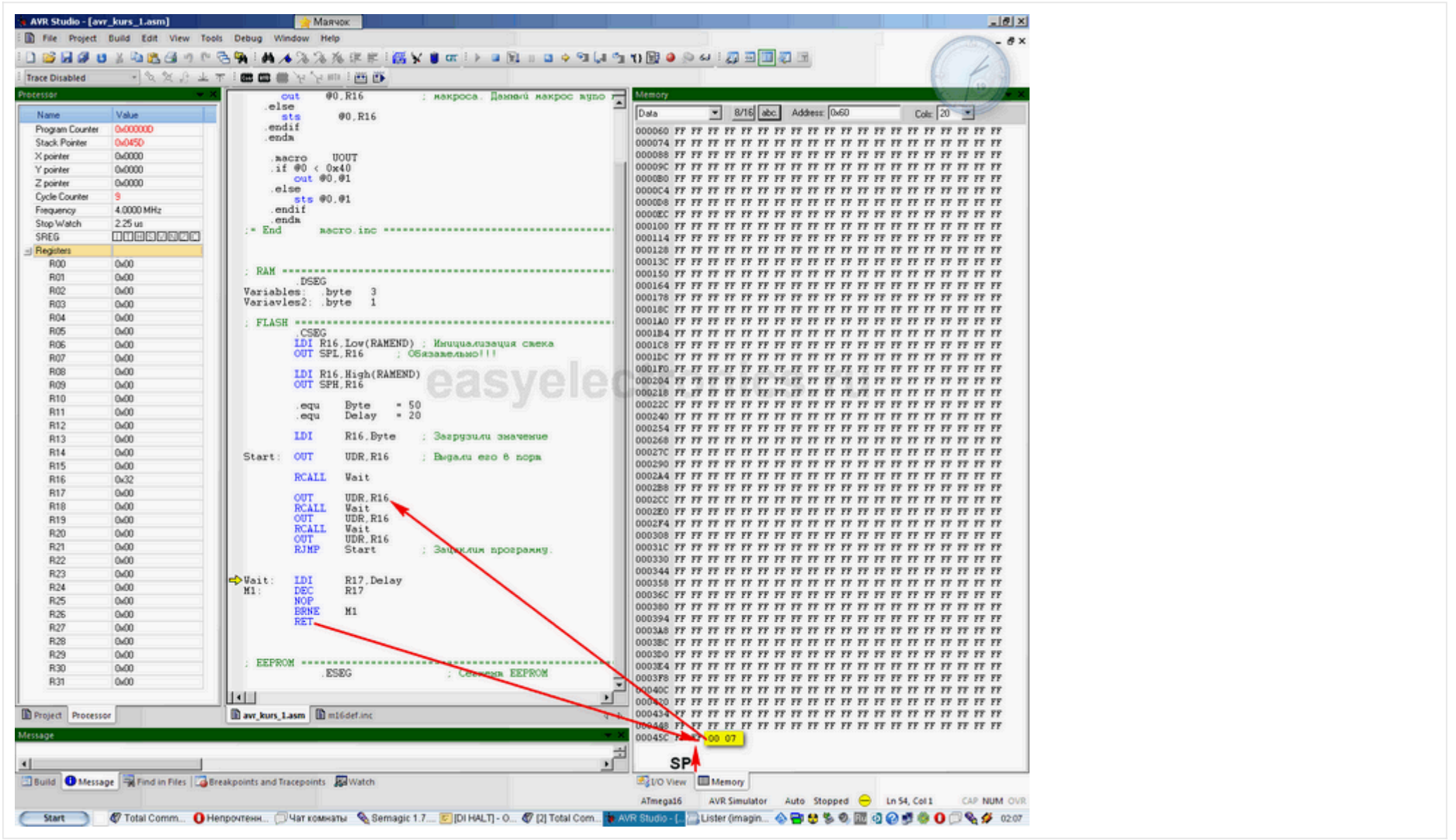


Увеличить

Адрес команды RCALL в памяти, по данным PC = 0x000006, адрес следующей команды (OUT UDR,R16), очевидно, будет 0x000007.

Указатель стека SP = 0x045F — конец памяти, где ему и положено быть в этот момент.

После RCALL



Увеличить

Смотри, в стек пихнулось число 0x000007, указатель сместился на два байта и стал 0x045D, а контроллер сделал прыжок на адрес Wait.

Наша процедура спокойно выполняется, как ей и положено, а по команде RET процессор достанет из стека наш занычeнный адрес 0x000007 и прыгнет сразу же на команду OUT UDR,R16

Таким образом, где бы мы не вызвали нашу процедуру Wait — мы всегда вернемся к тому же месту откуда вызвали, точнее на шаг вперед. Так как при переходах в стеке сохраняется адрес возврата. А если испортить стек? Взять и засунуть туда еще что нибудь? Подправь процедуру Wait и добавь туда немного бреда, например, такого

```
1
2
3
4
5
6
7
8
```

```
Wait:   LDI     R17,Delay
M1:     DEC     R17
        NOP
        BRNE    M1

        PUSH    R17             ; Ой, я не специально!

        RET
```

Перекомпиль и посмотри что будет =) Заметь, компилятор тебе даже слова не скажет. Мол все путем, дерзай :)

До команды PUSH R17 в стеке будет адрес возврата 00 07, так как в регистре R17 ,в данный момент, ноль, и этот ноль попадет в стек, то там будет уже 00 00 07.

А потом идет команда RET… Она глупая, ей все равно! RET тупо возьмет два первых верхних байта из стека и запихает их в Programm Counter.

И куда мы перейдем? Правильно — по адресу 00 00, в самое начало проги, а не туда откуда мы ушли по RCALL. А будь в R17 не 00, а что нибудь другое и попади это что-то в стек, то мы бы перешли вообще черт знает куда с непредсказуемыми последствиями. Это и называется срыв стека.

Но это не значит, что в подпрограммах нельзя пользоваться стеком в своих грязных целях. Можно!!! Но делать это надо с умом. Класть туда данные и доставать их перед выходом. Следуя железному правилу «Сколько положил в стек — столько и достань!», чтобы на выходе из процедуры для команды RET лежал адрес возврата, а не черти что.

Мозговзрывной кодинг

Да, а еще тут возможны стековые извраты. Кто сказал, что мы должны вернуться именно туда откуда были вызываны? =))) А если условия изменились и по итогам вычислений в процедуре нам ВНЕЗАПНО туда стало не надо? Никто не запрещает тебе нужным образом подправить данные в стеке, а потом сделать RET и процессор, как миленький, забросит тебя туда куда надо. Легко!

Более того, я когда учился в универе и сдавал лабы по ассемблеру, то лихо взрывал мозги нашему преподу такими конструкциями (там, правда, был 8080, но разница не велика, привожу пример для AVR):

```
        LDI     R17,low(M1)
        PUSH    R17
        LDI     R17,High(M1)
        PUSH    R17

; потом дофига дофига другого кода... для отвлечения
; внимания, а затем, в нужном месте, ВНЕЗАПНО

        RET
```

И происходил переход на метку M1, своего рода извратский аналог RJMP M1. А точнее IJMP, только вместо Z пары мы используем данные адреса загруженные в стек из любого другого регистра, иногда пригождается. Но без особой нужды таким извратом заниматься не рекомендую — запутывает программу будь здоров.

Но побалуйся обязательно, чтобы во всей красе прочувствовать стековые переходы.

Отлаженные и выверенные подпрограммы кода можно запихать в отдельный модуль и таскать их из проекта в проект, не изобретая каждый раз по велосипеду.

Иногда подпрограммы ошибочно называют функциями. Отличие подпрограммы от функции в том, что функция всегда имеет какое то значение на входе и выдает ответ на выходе, как в математике. Ассемблерная подпрограмма же не имеет таких механизмов и их приходится изобретать самому. Например, передавая в РОН или в ячейках ОЗУ.

Подпрограммы vs Макросы
Но не стоит маникально все повторяющиеся участки заворачивать в подпрограммы. Дело в том, что переход и возврат добавляют две команды, а еще у нас идет прогрузка стека на 2 байта. Что тоже не есть гуд. И если заменяется три-четыре команды, то овчинка с CALL-RET не стоит выделки и лучше запихать все в макрос.

Прерывания

Это аппаратные события. Ведь у микроконтроллера кроме ядра есть еще дофига периферии. И она работает параллельно с контроллером. Пока контроллер занимается вычислением или гоняет байтики по памяти — АЦП может яростно оцифровывать входное напряжение, USART меланхолично передавать или принимайть байтик, а EEPROMка неспеша записывать в свои тормозные ячейки очередной байт.

А когда периферийное устройство завершает свою работу оно поднимает флаг готовности. Мол, чувак, у меня все пучком, забирай результат. Процессор может проверить этот флаг в общем цикле и как то его обработать.

Но некоторые события в принципе не могут ждать, например USART — вовремя не обработаешь входящий байт, считай провафлил передачу, т.к. передающий девайс пошлет второй, ему плевать успел ты его обработать или нет. Для таких срочных дел есть прерывания.

У AVR этих прерываний с полтора десятка наберется, на каждое переферийное устройство по прерыванию, а на некотрые и не по одному. Например, у USART их целых три — Байт пришел, Байт ушел, Передача завершена.

Как это работает

Когда случается прерывание, то процессор тут же завершает текущую команду, пихает следующий адрес в стек (точно также как и при CALL) и переходит… А куда, собственно, он переходит?

А переходит он на фиксированный вектор прерывания. За каждым аппаратным прерыванием закреплен свой именной адрес. Все вместе они образуют таблицу векторов прерывания. Расположена она в самом начале памяти программ. Для Атмега16, используемой в Pinboard таблица прерываний выглядит так:

```
1  RESET           0x0000  ; Reset Vector
2  INT0addr        0x0002  ; External Interrupt Request 0
3  INT1addr        0x0004  ; External Interrupt Request 1
4  OC2addr         0x0006  ; Timer/Counter2 Compare Match
5  OVF2addr        0x0008  ; Timer/Counter2 Overflow
6  ICP1addr        0x000a  ; Timer/Counter1 Capture Event
7  OC1Aaddr        0x000c  ; Timer/Counter1 Compare Match A
8  OC1Baddr        0x000e  ; Timer/Counter1 Compare Match B
9  OVF1addr        0x0010  ; Timer/Counter1 Overflow
10 OVF0addr        0x0012  ; Timer/Counter0 Overflow
11 SPIaddr         0x0014  ; Serial Transfer Complete
12 URXCaddr        0x0016  ; USART, Rx Complete
13 UDREaddr        0x0018  ; USART Data Register Empty
14 UTXCaddr        0x001a  ; USART, Tx Complete
15 ADCCaddr        0x001c  ; ADC Conversion Complete
16 ERDYaddr        0x001e  ; EEPROM Ready
17 ACIaddr         0x0020  ; Analog Comparator
18 TWIaddr         0x0022  ; 2-wire Serial Interface
19 INT2addr        0x0024  ; External Interrupt Request 2
20 OC0addr         0x0026  ; Timer/Counter0 Compare Match
21 SPMRaddr        0x0028  ; Store Program Memory Ready
```

Как видишь, это первые адреса флеша. На каждый вектор отводится по два байта в которые мы можем записать любую команду. Но что попало туда обычно не вписывают, а ставят сразу JMP на какую нибудь метку, где уже можно спокойно сделать все что душе угодно, не стесняясь размеров кода.

Запишем эту бодягу в цивильной форме, через ORG и добавим немного кода.

```
1         .CSEG
2         .ORG $000        ; (RESET)
3         RJMP   Reset
4         .ORG $002
5         RETI             ; (INT0) External Interrupt Request 0
6         .ORG $004
7         RETI             ; (INT1) External Interrupt Request 1
8         .ORG $006
9         RETI               ; (TIMER2 COMP) Timer/Counter2 Compare Match
10        .ORG $008
11        RETI             ; (TIMER2 OVF) Timer/Counter2 Overflow
12        .ORG $00A
13        RETI               ; (TIMER1 CAPT) Timer/Counter1 Capture Event
14        .ORG $00C
15        RETI             ; (TIMER1 COMPA) Timer/Counter1 Compare Match A
16        .ORG $00E
17        RETI             ; (TIMER1 COMPB) Timer/Counter1 Compare Match B
18        .ORG $010
19        RETI             ; (TIMER1 OVF) Timer/Counter1 Overflow
20        .ORG $012
21        RETI             ; (TIMER0 OVF) Timer/Counter0 Overflow
```

```
22          .ORG $014
23          RETI            ; (SPI,STC) Serial Transfer Complete
24          .ORG $016
25          RJMP   RX_OK    ; (USART,RXC) USART, Rx Complete
26          .ORG $018
27          RETI            ; (USART,UDRE) USART Data Register Empty
28          .ORG $01A
29          RETI            ; (USART,TXC) USART, Tx Complete
30          .ORG $01C
31          RETI              ; (ADC) ADC Conversion Complete
32          .ORG $01E
33          RETI            ; (EE_RDY) EEPROM Ready
34          .ORG $020
35          RETI            ; (ANA_COMP) Analog Comparator
36          .ORG $022
37          RETI            ; (TWI) 2-wire Serial Interface
38          .ORG $024
39          RETI            ; (INT2) External Interrupt Request 2
40          .ORG $026
41          RETI            ; (TIMER0 COMP) Timer/Counter0 Compare Match
42          .ORG $028
43          RETI            ; (SPM_RDY) Store Program Memory Ready
44
45          .ORG   INT_VECTORS_SIZE        ; Конец таблицы прерываний
46
47      ;---------------------------------------------------------------
48      ; Это обработчик прерывания. Тут, на просторе, можно наворотить сколько
49      ; угодно кода.
50      RX_OK:   IN     R16,UDR         ; Тут мы делаем что то нужное и полезное
```

```
51
52          RETI                    ; Прерывание завершается командой RETI
53  ;---------------------------------------------------------------------
54
55
56  Reset:  LDI R16,Low(RAMEND)     ; Инициализация стека
57          OUT SPL,R16             ; Обязательно!!!
58
59          LDI R16,High(RAMEND)
60          OUT SPH,R16
61
62          SEI                     ; Разрешаем прерывания глобально
63          LDI   R17,(1<<RXCIE)    ; Разрешаем прерывания по приему байта
64          OUT    UCSRB,R17
65
66  M1:     NOP
67          NOP
68          NOP
69          NOP
70          RJMP M1
```

Теперь разберем эту портянку. Контроллер стартует с адреса 0000, это точка входа. Там мы сразу же делаем бросок на метку RESET. Если это не сделать, то контроллер пойдет выполнять команды из таблицы векторов, а они не для того там посажены. Да и не далеко он ускачет — без инициализации стека и наличия адреса возврата в нем первый же RETI вызовет коллапс. Ведь RETI работает почти также как и RET.

Поэтому сразу уносим оттуда ноги на RESET. Где первым делом инициализируем стек. А потом, командой SEI, разрешаем прерывания. И установкой бита в регистре периферии UCSRB включаем прерывание по приему байта.

Дальше зацикливаемся и ждем когда в приемный буффер USART извне свалится байт. Запускай это дело в эмуляцию и начинай трассировать по одной команде. Сначала, как я и говорил, проц прыгнет на метку Reset, потом выставит нужные значения и наглухо зациклится на

```
1
2
3
4
5
```

```
M1:     NOP
        NOP
        NOP
        NOP
        RJMP M1
```

До прихода байта. Но как же нам осуществить этот приход байта если весь наш эксперимент не более чем симуляция виртуального процессора в отладчике? А очень просто!

Вручную вписать этот байт в регистр и вручную же протыкать флаг, словно байт пришел. За прием байта отвечает флаг RXC в регистре периферии UCSRA, раздел USART. Найди там бит RXC и тыкни его, чтобы закрасился. Все, прерывание вроде как наступило.

Нажми F11, чтобы сделать еще один шаг по программе… Опа, стрелочка улетела в таблицу векторов, как раз на вектор

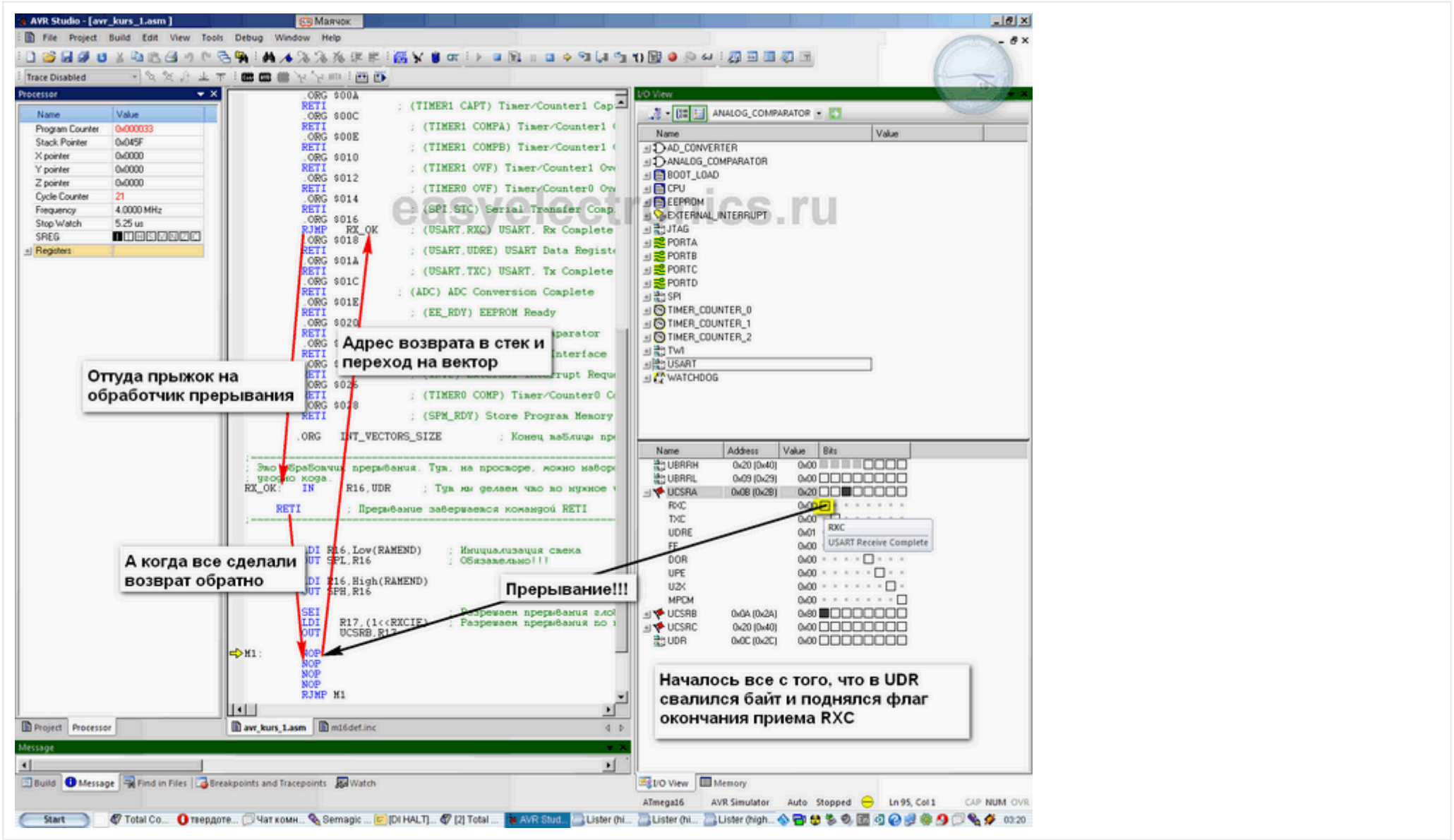
```
1
2
```

```
        .ORG $016
        RJMP   RX_OK     ; (USART,RXC) USART, Rx Complete
```

А оттуда уже прыжок сразу же на метку RX_OK, где мы забираем данные из регистра UDR в R17 и выходим из прерывания, по команде RETI.

При этом процессор вернется в наш бесконечный цикл, в то место откуда прерывался.

Вот, как это было, если по коду:



Увеличить

Вот, вроде теперь вопросов по выполнению быть не должно.

Разрешение и запрещение прерываний
Прерываний много, но по умолчанию они все запрещены. Во-первых, глобально — есть в процессоре в регистре SREG (о нем чуть позже) флаг I (interrupt) когда он равен 0, то все прерывания запрещены вообще, глобально, все без исключения.

Когда он равен 1, то прерывания глобально разрешены, но могут быть запрещены локально.

Устанавливается и сбрасывается этот флаг командами

- SEI — разрешить прерывания
- CLI — запретить прерывания (Да, по поводу моего ника, DI это, кстати, то же самое что и CLI, но для процессора Z80 ;))

Кроме того, у каждого прерывания есть еще свой собственный бит локального разрешения. В примере с UDR это бит RXCIE (Receive Complete Interrupt Enable) и расположены они в портах ввода вывода

По дефолту, после старта, все прерывания запрещены локально и глобально. И их надо разрешать вручную, не забыв прописать обработчик прерывания.

Если не прописать обработчик и по ошибке разрешить прерывания, то когда это ошибочное прерывание вдруг сработает, то процессор ускачет по вектору, а вот там все зависит от того, что ты туда прописал.

Если все неиспользуемые прерывания заглушены командой RETI то ничего не пройзойдет — как пришел так и вернется обратно. Если же там ничего нет, то процессор будет выполнять эту пустоту пока не доберется до живого кода. Это может быть как переход на обработчик другого прерывания (ниже по таблице векторов) так и начало основного кода, тело обработчика прерывания, какая либо процедура, записанная до метки start. Да что угодно, что первой попадется.

Сам понимаешь, что в таком случае ни о какой корректной работе говорить не придется. Кстати, почти все сишные компиляторы, неиспользуемые обработчики глушат на RESET т.е. случайный вызов несуществующего прерывания приводит к сбросу.

С одной стороны это бага, с другой фича — так называемая ситуация максимизации ошибки. Т.е. большой фатальный глюк, обрушивающий всю программу нахрен, куда безопасней мелкой ошибки потому, что вылазит сразу же. А ошибка может поджидать годами и спокойно пересидеть период отладки, а вылезти уже в готовом устройстве. И непременно перед тем как заказчик выпишет тебе бабло за разработку :)

Итак, повторюсь для ясности. Прерывания по дефолту запрещены локально и глобально. Разрешают их по мере надобности. Причем делают это дважды — на локальном уровне и на глобальном.

О том, что прерывание произошло извещает флаг этого прерывания в регистре ввода вывода. Пока флаг поднят считается что прерывание не обработано и процессор будет пытаться по этому прерыванию перейти (если конечно прерывания разрешены и локально, и глобально).

Флаг этого события сбрасывается либо сам, при переходе к обработчику прерывания, либо при совершении какого-либо действия. Например, чтобы сбросить флаг события прерывания RxC надо считать байт из UDR. Протрассируй программу и сам увидишь, что сброс флага RxC происходит после выполнения команды

1	`IN R16,UDR`

Либо флаг скидывают вручную — **записью в этот флаг единицы. Не нуля! Единицы!** Подробней в даташите на конкретную периферию.

Очередность прерываний
Но что будет если произошло одно прерывание и процессор ушел на обработчик, а в этот момент произошло другое прерывание?

А ничего не будет, при **уходе на прерывание происходит аппаратный глобальный запрет всех других прерываний**— просто сбрасывается флаг I, а по команде RETI, при возврате, этот флаг устанавливается в 1. В этом, кстати, отличие RET от RETI

Но! Никто не запрещает нам внутри обработчика поставить SEI и разрешить прерывания. При этом мы получим вложенные прерывания. Можно, но это опасно. Черевато переполнением стека и прочими гадостями. Так что это надо делать с твердым осознанием последствий.

Что тогда? Прерывание которое пришло во время обработки первого потеряется?

Нет, не потеряется. Флаг то его события никто сам не снимет, так что только процессор выйдет из первого обработчика (разреша при этом глобальные прерывания), как это не снятый флаг сгенерирует новое прерывание и процессор его обработает.

А теперь такая ситуация — прерывания запрещены, неважно по какой причине, и тут приходит два прерывания. Поднимает каждая свой флажок и оба ждут глобального разрешения. SEI!!! Кто пойдет первым? А первым пойдет то прерывание, чей вектор меньше по адресу, ближе к началу памяти. За ним второй.

В случае когда пришло несколько прерываний одного типа. Скажем, пока мы там ковырялись в обработчике, нам сбоку тут еще таймер три раза тикнул своим флагом, то обработается только одно событие, остальные могут потеряться.

Бег по граблям

Прерывания штука мощная, но опасная. С их помощью плодятся такие глюки, по сравнению с которыми стековые срывы так — семечки.

Вся засада багов из-за кривых прерываниях в том, что их практически невозможно отследить в отладчике.

Возникает плавающий глюк, появление которого зависит от того в каком именно месте кода произойдет вызов прерывания. Что поймать, сам понимаешь, почти невозможно.

Так что если у МК то понос, то золотуха, то программа петухом поет, а то молчит как рыба — знай, в 95% копать собаку надо в районе прерываний и их обработчиков.

Но если правильно написать прерывание, то багов оно не даст. Главное понимать в чем его опасность.

Грабли первые — спасай регистры!!!
Прерывание, когда оно разрешено, вызывается ВНЕЗАПНО, между двумя произвольными инструкциями кода. Поэтому очень важно, чтобы к моменту когда мы из прерывания вернемся все осталось как было.

Все регистры, используемые в обработчике прерываний, должны быть предварительно сохранены. Также должен быть сохранен регистр флагов SREG, в котором хранится результат логических операций. Результаты проще всего сохранять в стеке.

Приведу пример: Вот есть у нас обработчик прерывания который сравнивает байт на входе в USART и если он равен 10, выдает обратно отклик ‘t’ (ten в смысле).

```
1
2
3
4
5
6
7
8
9
10
11
```

```
RX_OK:

        IN      R16,UDR
        CPI     R16,10
        BREQ    Ten
        RJMP    Exit

Ten:    LDI     R17,'t'
        OUT     UDR,R17

Exit:
```

И у нас есть код. Код скопипащен от балды из какого то проекта, даже пояснять не буду, не в нем суть.

```
1
2
```

```
        . . .
        LPM    R18,Z
```

```
3                   CPI    R18,0
4
5                   BREQ   ExitStr
6
7                   SUBI   R16,65
8                   LSL    R16
9
10                  LDI    ZL,Low(Ltrs*2)
11                  LDI    ZH,High(Ltrs*2)
12
13                  ADD    ZL,R16
14                  ADC    ZH,R1
15
16                  . . .
```

Согласно идеи прерывания, наш обработчик может воткнуться между двумя любыми инструкциями.
Например, так:

```
1                   . . .
2                   LPM    R18,Z
3                   CPI    R18,0
4
5                   BREQ   ExitStr
6
7                   SUBI   R16,65
8   >>>>>>>>>>>Прерывание >>>>>>>>>>>
9   RX_OK:
```

```
10
11                  IN      R16,UDR
12                  CPI     R16,10
13                  BREQ    Ten
14                  RJMP    Exit
15
16          Ten:    LDI     R17,'t'
17                  OUT     UDR,R17
18
19          Exit:   RETI
20          <<<<<<<<<<< Возврат <<<<<<<<<<<<
21                   LSL    R16
22
23                   LDI    ZL,Low(Ltrs*2)
24                   LDI    ZH,High(Ltrs*2)
25
26                   ADD    ZL,R16
27                   ADC    ZH,R1
28                   . . .
```

До входа в прерывание, после команды SUBI R16,65 в R16 было какое то число из которого вычли 65.
И дальше с ним должны были провернуть операцию логического сдвига LSL R16, но тут вклинился обработчик, где в R16 записалось значение из UDR.

И мы выпали из обработчика перед командой LSL R16, но в R16 уже мусор какой то. Совершенно не те данные, что мы планировали.

Естественно вся логика работы от такого издевательства порушилась и возник глюк. Но стоит прерыванию прийти чуть раньше, на одну микросекунду — как глюк исчезнет, т.к. SUBI R16,65 будет уже после прерывания и логика не будет порушена, но может возникнуть другой глюк.

Полная лотерея, повезет не повезет. Может вылезти сразу, а может и через год идеальной работы, а потом также сгинет и сиди чеши репу что же это было — сбой по питанию, бага, или таракан по плате пробежал неудачно.

Чтобы такого не было в обязательно порядке надо сохранять регистры и SREG на входе в прерывание и доставать на выходе.

У нас тут, в обработчике, используется R17 и R16 и SREG (в него помещается результат работы команды CPI). Вот их и сохраним.

Выглдеть это будет так:

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
```

```
RX_OK:  PUSH    R16             ; Сохранили R16
        IN      R16,SREG        ; Достали SREG в R16
        PUSH    R16             ; Утопили его в стеке
        PUSH    R17             ; Туда же утопили R17

; Теперь можно со спокойной совестью работу работать.

        IN      R16,UDR
        CPI     R16,10
        BREQ    Ten
        RJMP    Exit

Ten:    LDI     R17,'t'
        OUT     UDR,R17

; А на выходе вернем все как было.
```

```
17  ; Достаем в обратном порядке
18
19  Exit:   POP     R17
20          POP     R16
21          OUT     SREG    R16
22          POP     R16
23          RETI                    ; Спокойно выходим. Регистры вернул как было.
```

Как же выносить данные из прерываний? Да по разному, можно в память сохранять, можно для этого спец регистр заиметь и знать, что он может в любой момент измениться в прерывании.

Грабли вторые — не тормози!!!
Прерывания отвлекают процессор от основных дел, более того, они блокируют другие прерывания. Поэтому в прерывании главное все сделать максимально быстро и свалить. Никаких циклов задержки, никаких долгоиграющих процедур. Никаких ожиданий аппаратного события. СКОРОСТЬ! СКОРОСТЬ! СКОРОСТЬ! Вот что должно тобой руководить при написании обработчика.

Заскочил — сделал — выскочил!

Но такая красивая схема возможна далеко не всегда. Иногда бывает надо по прерыванию делать и медленные вещи. Например, прием байтов из интерфейса и запись их в какую нибудь медленную память, вроде EEPROM. Как тогда быть?

А тут делают проще. Цель прерывания — во что бы то ни стало среагировать на событие именно в тот момент, когда оно пришло, чтобы не прозевать. А вот обрабатывать его прямо сейчас обычно и не обязательно.
Заскочил в обработчик, схватил быстро тухнущие данные, перекинул их в буффер где им ничего не угрожает, поставил где-нибудь флажок отметку «мол там байт обработать надо» и вышел. Все! Остальное пусть сделает фоновая программа в порядке общей очереди.

Либо, если иначе никак нельзя, разрешай прерывания внутри обработчика, но смотри чтобы не возникли рекурсивные прерывания, когда один и тот же обработчик разрешает прерывания сам себе, образуя множественные вложенные вызовы с зарыванием в стек.

Грабли третьи — атомарный доступ
Есть ряд операций которые должны выполняться неразрывно. Например, чтение 16ти разрядных регистров таймера.

Ядро то у нас восьми-разрядное, поэтому 16ти разрядный регистр таймера считывается в два приема сначала младший байт, а потом старший. И вот в этом месте нельзя ни в коем случае допускать, чтобы между считыванием младшего и старшего было прерывание.

Т.к. после выхода из прерывания таймер уже может много чего натикать и информация уже будет не актуальная.

Поэтому перед чтением делаем CLI, а после SEI.

Также нельзя разрешать прерывания если одна и та же многоходовая операция делается и в прерывании и в главном цикле.

Например, прерывание хватает байты из АЦП и пишет их в буффер (ОЗУ), образуя связные цепочки данных. Главный же цикл периодически из этого буффера данные читает. Так вот, на момент чтения буффера из памяти, надо запрещать прерывания, чтобы никто не посмел в этот буффер что-нибудь записать.
Иначе мы можем получить невалидные данные — половина байтов из старого замера, половина из нового.

Причем, во многих случаях не обязательно глобально блокировать все прерывания вообще, достаточно заблокировать то локальное прерывание, которое может нам нагадить.

Грабли четвертые — не блокируй!!!
Третие грабли нужно внимательно обходить, но в паранойю впадать тоже не следует. Тупо запрещать прерывания везде, где мерещится бага, не стоит. Иначе можно прозевать события, а это плохо. Нет, обработчик то выполнится, но будет это уже не актуально — хороша ложка к обеду.

Обязательно погоняй в отладчике код (да хотя бы кучу NOP) с разными прерываниями. Чтобы понять и прочувствовать как ведут себя прерывания.

Старая версия статьи. Чисто поржать :)

Assembler AVR Программирование

233 thoughts on “AVR. Учебный курс. Подпрограммы и прерывания”

axf

30 Сентябрь 2008 в 11:33

А есть в данной вариации ассемблера аналог команд PUSHAD/POPAD, сохраняющих в стек и достающих из него значения всех регистров общего назначения?

★ DI HALT

30 Сентябрь 2008 в 11:51

Нету. Тут же 32 регистра. Никакой оперативки не напасешься пихать такую массу.

Xenomorph

15 Октябрь 2008 в 2:11

Начал изучать прерывания опять же на C. И вот какое дело, обработчик прерываний у меня срабатывает когда на линии PB5 меняется уровень, в обработчеке у меня простой код если PB5 = 0 и PB6= 0 то пишу ноль, исли PB6=1 пишу еденицу. Так вот за 100 ms
у меня проходит 100 еденицы и нулей. Суть в чем если я просто посылаю по UART, то принимаю все биты. Но стоит мне записать биты в

массив, а потом перевести их в число, как на выходе я получаю какую то биллиберду. Я вот думаю успевает ли выполнится обработчик прерывания, до возникновения другого прерывания? Или может я что не так делаю???

---

sVk

20 Декабрь 2008 в 1:28

Помогите разобраться пэжл. Есть кусок программы, приведенный ниже.

В строке 3 проц уходит на обработку по метке loc_Control_VUART2UART_05 (при T=0). Там (строка 35)он
Вопрос 1: обнуляет бит VUART_RECEIVE регистра Status_VUART. Я прав?
Вопрос 2: что значит 1<<?
после чего: выходит из loc_Control_VUART2UART_05 и
Вопрос 3: продолжает «копать дальше» строку 4 и далее следующие строки?
ИЛИ
Вопрос 4: выходит из Control_VUART2UART в строке 1?
Еще общий вопрос: если рассматривать call и brxx. Обе они переходят по метке. Call переходит по метке процедуры, сохраняя в стек адрес команды, к которой надо будет вернуться после выполнения процедуры. Branch(как и jmp) тоже переходит по метке, но в стек ничего не сейвит. Я прав? Если так, то на вопрос 4 получается ответ ДА.

И еще вопрос-для закрепления знаний =):
Впорос 5: Если я , например, по строке 22 перешел в лок_3 в строке 33, то далее проц «копает» 34-ю, 35-ю, 36-ю и далее по 37-й строке выходит либо из главной процедуры либо в строку 23(если branch сейвит в stack, надеюсь что он так не делает)))

```
1 Control_VUART2UART:
2 rcall ChkUART_RTS
3 brtc loc_Control_VUART2UART_05
4 sbrs Status_VUART,VUART_RECEIVE
5 ret
6 out UDR,RX_VUART
```

```
7 #ifdef TestMode
8 mov tstUART,RX_VUART
9 sbr Status_VUART,1<<TST_UART_RX
10 #endif
11 tst NextBaudRate
12 breq loc_Control_VUART2UART_05
13 tst Count_ChkSumm
14 brne loc_Control_VUART2UART_01
15 mov ChkSumm,RX_VUART
16 cpi RX_VUART,0x82
17 brne loc_Control_VUART2UART_05
18 rjmp loc_Control_VUART2UART_04
19 loc_Control_VUART2UART_01:
20 ldi temp0,0x05
21 cp Count_ChkSumm,temp0
22 brne loc_Control_VUART2UART_03
23 cpi RX_VUART,0x54
24 brne loc_Control_VUART2UART_02
25 cp ChkSumm,RX_VUART
26 brne loc_Control_VUART2UART_02
27 mov CurrBaudRate,NextBaudRate
28 loc_Control_VUART2UART_02:
29 clr NextBaudRate
30 rjmp loc_Control_VUART2UART_05
31 loc_Control_VUART2UART_03:
32 add ChkSumm,RX_VUART
33 loc_Control_VUART2UART_04:
34 inc Count_ChkSumm
35 loc_Control_VUART2UART_05:
```

36 cbr Status_VUART,1<<VUART_RECEIVE ;
37 ret

★ DI HALT

20 Декабрь 2008 в 1:45

1<<? это макрооператор языка, не команда процессора — задвинуть 1 влево на число «?», то есть мы делаем число в котором все нули, но один бит с номером «?» установлен в 1. Ищи чему равно «?» (это должно быть макроопределение каке нибудь из серии .equ)

В строке 36 происходит обнуление бита VUART_RECEIVE в регистре Status_VUART

А в строке 37 происходит возврат к точке откуда это подпрограмму вызвали и это не строка 4, а где то явно выше ее. В строке 3 проц просто прыгает (это не CALL!!!) на метку. Выйдет по строке 37 вообще из процедуры «Control_VUART2UART:» выше, в вызвающую прогу. Не на строку 1, а куда то туда, где последний раз был «RCALL Control_VUART2UART»

Разницу между CALL и JMP/BRanch ты понял правильно. Так и есть.

Вопрос 5:
После бранча проц продолжает копать оттуда куда его послали. Возврата нет, ибо бранч не сохраняет адреса возврата.

---

sVk

20 Декабрь 2008 в 1:56

Спасибо огромное. Ты резкий до ужаса)я бы только писал столько, за сколько ты и прочел и ответил))

★ DI HALT

20 Декабрь 2008 в 2:02

А что это за хрень? Судя по меткам это гонит из виртуального уарта (VUART судя повсему это виртуальный) в реальный. Из Ethernet делаем UART? или чо?

sVk

20 Декабрь 2008 в 15:20

Вообще вся схема — это преобразователь интерфейса диагностического порта автомобиля k-line в интерфейс Bluetooth. Сам бы рад разобрался что там куда гонит. Вся прога вместе с назначениями типа .def .equ около 650 строк. Надо разобрался за 3 дня, и потом по возможности еще в протеусе смоделить(((. AVR я изучаю ровно неделю))))), контроллеры всякие — месяц))). А VUART это как я пока думаю — UART со стороны блютусника.

★ DI HALT

20 Декабрь 2008 в 15:26

650 строк эт немного :)))))) В протеусе вряд ли смоделишь. Слишком кривая среда для моделинга сложных прог. Разве что МК будет точно такой же какие там есть. В протеусе хорошо куски программ отлаживать. Какие то узлы проверять.

3 дня это сильно. Если чо стучись в аську. Оперативней будет. Номер в инфо есть.

sVk

20 Декабрь 2008 в 15:34

Оки, вообще говоря, моя задача сейчас — просто нарисовать некое подобие блок-схем логики работы устройства. Комменты есть к каждой строке, так что с этим, думаю справлюсь,понимания особого это не требует. А вот с протоколами k-line и BT, AT-командами придется разобраться))

borsh

28 Декабрь 2008 в 5:31

Все прочитал, осознал и задался вопросом: Зачем указывать в программе на конец стека ? Неужели есть задачи, где стек сдвинут от конца SRAM, а данные лежат после стека? Почему бы производителю на зашивать в указатель стека конец SRAM ???

★ DI HALT

28 Декабрь 2008 в 15:35

Да сколько угодно. Например задача в которой есть вероятность что стек зохавает важные данные, а корректность работы пофигу. Тогда делаем так, чтобы стек начинался после данных и рос до упора вперед. Дальше, если стек сорвет, процессор начнет гнать ахинею, но недолго — вачдог его ребутнет и на основании сохраненных данных он может снова восстановиться и продолжить работу.

SergeyDon

30 Октябрь 2009 в 22:55

про стек:
на своём втором компе Z80 (это проц) со стеком делал так, ставил его на конец видеопамяти в аккумулятор = 0 и циклом пробегал push A. очень быстро очищал экран, а затем стек на место и работаем дальше.

тут правда ещё не придумал как использовать!

есть вопрос мнемоники команд асма для AVR написанны везде и количество тактов есть, а машинный код можно гдето глянуть? чисто из любопытства посмотреть, дизасемблер ведь както работает?
например команда jmp XX в памяти занимает два байта или три? чет совсем уже все забыл :(

---

testicq

3 Февраль 2009 в 3:05

Легкий косячек:
«В случае ассемблерной процедуры, тебе бы пришлось вначале нужно было подсунуть гуталин вместо зубной пасты»
лишнее «нужно было»

https://me.yahoo.com/a/770uZqM_y9ul8SjKtXbXVDqaYTo-#a68f1

24 Февраль 2009 в 2:15

))

ZorG

28 Апрель 2009 в 18:55

2 DI HALT:
Ну вот и я добрался до своего первого комментария, хоть читаю твои статьи уже не первый год в Хакере :) Чувак — ты реально крут, спасибо тебе за статьи и этот ресус :)

Теперь по делу. Чего-то нигде не нашел упоминания про внешние прерывания. У меня вот имеется AT90S2313, судя по заголовочнику в AVR для него, имеют место быть два внешних прерывания.

Внимание вопрос: что должно произойти, чтобы эти прерывания случились?

И второй вопрос: есть ли возможность генерации прерываний по нажатию кнопок, иными словами при переходе уровней ног настроенных на вход в инверсное состояние?

★ DI HALT

28 Апрель 2009 в 19:20

Для сработки INT0 или INT1 должен измениться логический уровень на входах INT0 или INT1 соответственно. Как изменится — зависит от настроек этого прерывания.

ZorG

28 Апрель 2009 в 22:49

Вот черт :) слона то я и не приметил :) RTFM как говорицца. А прерывания по нажатию кнопок, видимо, возникнет только на этих ногах?

★ DI HALT

28 Апрель 2009 в 23:11

Да, но есть новые AVR, например Mega48..328 у которых есть такая милая фича как прерывание по изменению на одной из 8ми ног.

---

portvein

25 Август 2009 в 17:26

Можно глупый вопрос? =) А что делать, если есть некоторый цифровой датчик, генерирующий продолжительные по времени импульсы, эти импульсы МК считывает через INT0 (причём INT0 настроен на Any change, т.к. нужно посчитать продолжительность импульса) и выводит на сегментный индикатор динамически, т.е. косяк в том, что при возникновении прерывания индикация виснет на том что успело вылезти и продолжается только после спада.

★ DI HALT

25 Август 2009 в 17:42

а из прерывания выход делается когда?

По хорошему надо так:

Прерывание настроено на 01 пришло — мы в прерывании запустили таймер, перенастроили прерывание на 10, вышли из прерывания.

Прерывание снова пришло (конец импульса). Мы в прерывании сняли показания таймера, перенастроили прерывание снова на 01 и вышли.

В итоге, прерывание возникает два раза по началу концу импульса. а все время крутится фоновая прога которая и показвает нам индикацию и прочие плюшки. А сами прерывания длятся считанные десятки команд.

---

portvein

25 Август 2009 в 17:57

ага, данке шон))) Что-то я стормозил и считал время прямо в обработчике прерывания))))

ЗЫЖ А можно по подробнее про фичи с прерываниями по любой из восьми ног? Я так понял на ATTINY2313 такая штука есть? Только что-то не один симулятор 2313 не симулирует((((

---

tranzistor

12 Октябрь 2009 в 0:05

Цитирую: «В AVR из программы до счетчика никак не добраться». Немного поправлю, если позволите. В принципе, можно написать типа rjmp PC+2. Это конечно, не непосредственное обращение к счетчику команд, но все-таки. Хотя я такой прием юзаю редко… И кстати, если писать JMP PC+2, то он не перескочит через команду! Нужно прибавлять 4!
А теперь можно вопрос? В третьей части в файле, где определяются векторы прерываний, стоят команды rjmp. Я вот запутался, блин. Почитал Ревича мельком ту главу где про различия для этих команд (rjmp и jmp) и сделал для себя вывод:что, мол, если работаешь с Мегой16, пиши всегда jmp и call. Не ошибешься! Размер меня не интересует, проги маленькие, места всем хватит…
Прав ли я? Или даже в мегах иногда НЕОБХОДИМО использовать rjmp и rcall?
Спасибо

★ DI HALT

12 Октябрь 2009 в 2:01

Ну это ты добираешься не до счетчика, а всего лишь до текущего адреса.

Счетчик, кстати, можно изменить или получить (это вызовет переход) извратскими методами.

Например таким образом: Делаем RCALL, а возвращаемся через RJMP при этом у нас в стеке окажется PC и его можно POP нуть в регистры :)

В процедуре сразе же после RCALL перехода можно сделать:
POP R16 достать адрес возврата
POP R17
MOV R18,R16 Скопировать его
MOV R19,R17 в безопасное место
PUSH R17 положить его на место.
PUSH R16

А можно не только скопировать но и хитро изменить! Тогда по RET мы выйдем не туда откуда зашли, а туда куда хотим. Редкостный извратм но порой так прикольно :) За такие крышесносные хохмы я и люблю ассемблер :)
Этаким образом можно замутить адский конечный автомат… yyyxxxx… но мозг взорвет сразу :)

Ну и тому подобные извраты

---

mrkoin

17 Октябрь 2009 в 4:44

Так как вопрос остался неотвеченным:

Или даже в мегах иногда НЕОБХОДИМО использовать rjmp и rcall?

ИМХО, если вектор прерывания позволяет (по размеру) использовать JMP/CALL (которые длиннее RJMP/RCALL), то можно и их использовать.

Соответственно, в Меге8 (видимо — я в даташит еще не смотрел :-)) вектор прерывания имеет места всего на 2 байта, поэтому там просто приходиться использовать короткие переходы.

Jyraf

17 Ноябрь 2009 в 20:07

А подскажите пожалуйста!
Программу пишу в AVRStudio, эмуляция в Proteuse.
Устройство на ATtiny2313, программа разбита на несколько файлов:
— define.asm — определение переменных и имен регистров…
— init.asm — очистка памяти, регистров конфигурация портов…
— macro.asm — макросы…
— vectors.asm — таблица векторов прерываний…
и основной файл — ttt.asm…
Программа начинает выполнятся с метки Init:, расположенной в файле init.asm, где сбрасывает в ноль все регистры, RAM, указатель стека и инициализирует всю периферию.
В основном файле программы, при нажатии на кнопку (внешнее прерывание), программа входит в бесконечный цикл (режим СТОП) или должна начаться с начала (СТАРТ).
Для перехода в начало программы (СТАРТ) использую
rjmp Init
Так вот, в AVRStudio при пошаговом прогоне все работает правильно, а вот в Proteuse, контроллер тупо вешается и не реагирует не на какие действия (в том числе сброс кнопкой Reset). После остановки и повторного запуска эмуляции контролер снова работает нормально.
Не могу понять, это связано с глюками Proteusa, или это из за того, что метка Init: расположена не в главном файле программы.
И еще один вопрос, можно ли прерывания и подпрограммы скинуть в отдельные файлы (естественно используя .include «prer.asm»)?

★ DI HALT

17 Ноябрь 2009 в 20:09

Протеус грузит в себя хекс и ему должно быть пофигу как там на файлы разбита прога. Возможно глюк протеуса.

Вынести в один файл можно, я так сделал — у меня есть отдельный vectors.asm где все вектора и переходы. Но там пришлось поиграть с ORG метками ,чтобы компилятор не ругался. В общем. не удалось мне ORG000 оставить до инклюда, и пришлось пихать ее внутрь.

Jyraf

17 Ноябрь 2009 в 20:34

Понятно, спасибо.
А может кто-то сталкивался с такими глюками, часто встречаю коменты о том, что в Proteuse лажа, а в железе все нормально. Потому как пока в эмуляции, можно и помучатся, главное чтобы в железе все работало… Очень не хочется потом тратить время на переделку программы.

Если никто не ответит — буду первый, кто попробует это сделать, потом отпишусь. :)

Да, кстати, дайте рекомендации по составу программы в целом, что ставить в начало, что в середине, что в конце. Где расположить главный цикл программы. Я понимаю, что у каждого свои привычки, поэтому и прошу р е к о м е н д а ц и и.

★ DI HALT

17 Ноябрь 2009 в 20:47

Ну обычно такая компоновка всегда

СТарт.
Инициализация
главный цикл
подпрограммы
прерывания

Прерывания дальше потому как до них проще дострелить из вектора каким нибудь JMP.

Сама архитектура — либо диспетчер RTOS либо флаговый автомат. Я обычно на диспетчере делаю, удобней. В универах учат флаговым автоматам обычно.

Jyraf

17 Ноябрь 2009 в 21:19

Спасибо…

★ DI HALT

17 Ноябрь 2009 в 22:03

Скоро будет подробный разбор стандартных скелетов программ. Суперцикл, флаговый автомат, диспетчер, RTOS кооперативная, RTOS выстесняющая со всеми ее недостатками и достоинствами.

DIMA040891

3 Декабрь 2009 в 23:23

В AVR Studio4 неполучается инициализировать стек

.

.

.

ldi Temp,RamEnd

out SPL,Temp

.

.

.

Пишет что C:\NEW\A1\A1.asm(12): error: Operand(s) out of range in ‘ldi r16,0x45f’

Пожалуйста объясните в чём ошибка???
Контроллер ATmega8. На других МК тоже самое, только у ATiny2313 вроде как работает.

★ DI HALT

3 Декабрь 2009 в 23:32

ldi Temp,low(RamEnd)
out SPL,Temp
ldi Temp,high(RamEnd)
out SPH,Temp

так надо. У тебя тут рамэнд двухбайтный получается, т.к. у меги 8 памяти больше чем у тини.

---

sTARcRAB

11 Январь 2010 в 20:19

Вопрос по теме прерываний. Только не на асме, а на Си.
1. Есть программа: «начало программы — основной цыкл программы». Программа крутится в основном цыкле WHILE (1). После обработки преривания мы идём в «начало программы». Можно ли организовать такое выполнение прерывания?
2. Можно ли зделать так чтобы прерывание возникало при переходе 0-1 или 1-0, при этом 1 или 0 на ножке, по которой разрешено прерывание будет находится до следущего перепада 1 — 0 или 0 — 1 ?
Спасибо.

★ DI HALT

12 Январь 2010 в 1:42

1) как то можно было сделать. С помощью описания обработчика. Есть там какие то хитрые модификаторы. А можно хакнуть — по окончании обработки вызвать какое нибудь неинициализированное прервыание программно.

2) Можно, но тебе придется в обработчике события перепрограммировать это прерывание на другой фронт. Только и всего.

Maximka

19 Февраль 2010 в 15:46

Таблица прерываний это адреса куда перейдет выполнение программы после наступления какого либо события. Эти адреса жестко «прописаны» в структуре МП, и при наступлении кокого либо из них процессор автоматически перейдет на вполнение в эту ячейку. А в этой ячейке команда безусловного перехода RJMP (JMP). Если нам это прерывание не нужно но наступить оно все таки может то мы пишем RETI. Тогда получается если нам нужно использовать только первое и последнее прерывание, то чтоб заполнить адреса в памяти нужно писать

```
rjmp INT0 ; первое прерывание
reti
reti
….
reti
rjmp INTn ; последнее прерывание
```

либо

```
rjmp INT0 ; первое прерывание
.org адрес
rjmp INTn ; последнее прерывание
а остальные запрещаем
```

Я все правильно понял?))

★ DI HALT

19 Февраль 2010 в 15:53

Можно сделать с орг, но лучше забить все неиспользованные прерывания командой RETI один фиг мы это место юзать не будем, но так надежней.

Sher

25 Февраль 2010 в 0:24

Подскажите пожалуйсто,в чем проблема. Использовал прерывания в Attiny2313 — программа работала как часики. Нужно было переписать прогу на ATmega16. Вот тут и начались фокусы: после того, как было вызвано прерывание, программа перешла, как и положено на метку по вектору прерывания. Когда прерывание закончилось командой reti, курсор прыгнул не в то место, откуда было вызвано прерывание, а вообще не понятно куда, где-то в начало программы. Таблицу векторов прерывания, как и в Attiny2313 разместил в начале программы.

★ DI HALT

25 Февраль 2010 в 8:27

У тини2313 стековый указатель однобайтный, а у меги16 двухбайтный. Ты не забыл загрузить второй байт SP?

Sher

28 Февраль 2010 в 16:19

Клево, заработало))! Спасибо, DI HALT!

---

strz

1 Апрель 2010 в 15:49

DI_HALT,
симулирую UDR прерывание. Значит програма зациклена, выставляю биты в UDR и ставлю флажок RXC, жму F11, все как положенно — программа идет по вектору, затем в обработчик, а вот там я ожидал что после команды

IN R16, UDR
в окне памяти по адрессу решистра R16 я увижу тот байт что я ставил в UDR, но там девственно стоят нули.

★ DI HALT

1 Апрель 2010 в 15:51

Так и должно быть. Студия не симулирует передачу байтов из UDR. Байт надо подставлять напрямую в регистр куда ты делал IN

tyler.graip

4 Апрель 2010 в 19:37

странно, как раз сейчас это проверял,
получается после IN R16, UDR
в R16 значение которое я выставил в UDR.
использую avr simulator (1) и tiny2313

★ DI HALT

4 Апрель 2010 в 21:12

Хм, может в разных версиях симулятора по разному? У меня на мега16 (не помню, врод бы симулятор 2) не передавалось значение из регистра UDR, приходилось натыкивать уже после.

Evg333

3 Июнь 2010 в 16:56

у меня тоже 00, UDR сбрасывается первым же нажатием F11. Хорошо в комментариях написано, а то уже хотел спрашивать)

Ana-bio-z

15 Февраль 2014 в 20:43

Вот именно, что — сбрасывается при первом же нажатии F11. Поэтому пробивать биты в UDR нужно непосредственно перед выполнением команды:
IN R16,UDR. Т.е. когда стрелочка в симуляторе стоит напротив нее. Пробиваем биты, нажимаем F11, значение UDR уходит в R16, а сам UDR обнуляется.

Temp

5 Апрель 2010 в 11:12

«Бег по граблям
…
Вся засада багов из-за кривых прерыванИЯХ в том…»
Ошибочка в тексте. Простите за дотошность :))

---

dima_m

10 Май 2010 в 16:30

Выше в комментариях шла речь про внешнее прерывание int0. Как раз сейчас делаю детектор нуля через INT0, вопрос в том как правильно настроить ножку микроконтроллера.
1.Как вход с подтяжкой или вход без подтяжки?
Как понял если прерывание настраивать на низкий уровень, то с подтяжкой. А если по спадающему или нарастающему фронту сигнала, то как…? Без или с подтяжкой?

★ DI HALT

10 Май 2010 в 16:34

Как настраивать совершенно не имеет значения для работы на прерывание. Зависит от схемотехники детектора. От подтяжки только зависит уровень на висящем в воздухе выводе.

---

dima_m

12 Июнь 2010 в 22:18

Допускается ли вызывать искусственно какое либо прерывание, допустим командой RCALL или обязательно оно должно вызваться только аппаратно самим мк?

---

poljak181

19 Июль 2010 в 0:35

DI HALT, зацени, в этой строчке очепятка закралась
«Смотри, в стек пихнулось число 0×000007, указатель сместился на два байта и стал 0×035D, а контроллер сделал прыжок на адрес Wait.»
Адрес должен быть 0×035D (как на скрине)

З.Ы. Офигеннейший ресурс, учусь по нему) СПАСИБО!

---

poljak181

19 Июль 2010 в 0:37

Ааа, тараканы жрут мой мозг, 0×045D должно быть))

---

sam2sam

8 Август 2010 в 1:08

Наверно опечатка, R17 заменить на R16
«А оттуда уже прыжок сразу же на метку RX_OK, где мы забираем данные из регистра UDR в R17 и выходим из прерывания, по команде RETI.»
В листинге программы команда (прерывание):
RX_OK: IN R16,UDR

---

smallghost

24 Август 2010 в 12:44

Было

45 .ORG INT_VECTORS_SIZE ; Конец таблицы прерываний

Надо

45 .ORG INT_VECTORS_SIZE * 2 ; Конец таблицы прерываний

Т.к. константа содержит размер таблицы в словах, а не байтах.

★ DI HALT

24 Август 2010 в 13:53

Не гони, так и должно быть. Студия тоже оперирует в адресации словами, а не байтами Проверь на любом из векторов, вызывав прерывание. Они записаны в том же виде, что и инт_вектор_сайз

вот кусок для меги8

```
.equ ERDYaddr = 0x000f ; EEPROM Ready
.equ ACIaddr = 0x0010 ; Analog Comparator
.equ TWIaddr = 0x0011 ; 2-wire Serial Interface
.equ SPMRaddr = 0x0012 ; Store Program Memory Ready
```

<— вот тут должна уже быть программа, т.е. по адресу 0x0013

```
.equ INT_VECTORS_SIZE = 19 ; size in words
```

что собственно и происходит 19 = 0x0013

---

smallghost

24 Август 2010 в 16:30

Вижу тогда вот такую ошибку компилятора на первой строке кода:
_PWM.asm(83): error: Overlap in .cseg: addr=0x14 conflicts with 0x14:0x15

```
.ORG $024 ; (SPM_RDY) Store Program Memory Ready
RETI

.ORG INT_VECTORS_SIZE ; Конец таблицы прерываний
; Interrupts ============================================

; = USART, Rx Complete ========
RX_OK: PUSH R16 ;сохранили R16
IN R16,SREG ;достали SREG
PUSH R16 ;сохранили SREG
```

★ DI HALT

24 Август 2010 в 17:08

Что за камень?
Откуда взята таблица векторов для него?

---

smallghost

24 Август 2010 в 18:49

8я мега
таблица взята из твоего примера, убрал лишние прерывания которых в 8й нет на основании m8def.inc

у меня получается последняя метка адреса .ORG $024 = 36 + 2 байта на переход = 38 (INT_VECTORS_SIZE * 2).

у тебя для 16й меги последняя метка .ORG $028 = 40 + 2 байта на переход = 42 (INT_VECTORS_SIZE * 2).

Или я что-то не понимаю

---

smallghost

24 Август 2010 в 18:49

8я мега
таблица взята из твоего примера, убрал лишние прерывания которых в 8й нет на основании m8def.inc

у меня получается последняя метка адреса .ORG $024 = 36 + 2 байта на переход = 38 (INT_VECTORS_SIZE для 8й * 2).

у тебя для 16й меги последняя метка .ORG $028 = 40 + 2 байта на переход = 42 (INT_VECTORS_SIZE для 16й * 2).

Или я что-то не понимаю

★ DI HALT

24 Август 2010 в 19:38

Таблица векторов меги16

```
; ***** INTERRUPT VECTORS ************************************************
.equ   INT0addr   = 0x0002   ; External Interrupt Request 0
.equ   INT1addr   = 0x0004   ; External Interrupt Request 1
```

```
.equ   OC2addr   = 0x0006   ; Timer/Counter2 Compare Match
.equ   OVF2addr   = 0x0008   ; Timer/Counter2 Overflow
.equ   ICP1addr   = 0x000a   ; Timer/Counter1 Capture Event
.equ   OC1Aaddr   = 0x000c   ; Timer/Counter1 Compare Match A
.equ   OC1Baddr   = 0x000e   ; Timer/Counter1 Compare Match B
.equ   OVF1addr   = 0x0010   ; Timer/Counter1 Overflow
.equ   OVF0addr   = 0x0012   ; Timer/Counter0 Overflow
.equ   SPIaddr   = 0x0014   ; Serial Transfer Complete
.equ   URXCaddr   = 0x0016   ; USART, Rx Complete
.equ   UDREaddr   = 0x0018   ; USART Data Register Empty
.equ   UTXCaddr   = 0x001a   ; USART, Tx Complete
.equ   ADCCaddr   = 0x001c   ; ADC Conversion Complete
.equ   ERDYaddr   = 0x001e   ; EEPROM Ready
.equ   ACIaddr   = 0x0020   ; Analog Comparator
.equ   TWIaddr   = 0x0022   ; 2-wire Serial Interface
.equ   INT2addr   = 0x0024   ; External Interrupt Request 2
.equ   OC0addr   = 0x0026   ; Timer/Counter0 Compare Match
.equ   SPMRaddr   = 0x0028   ; Store Program Memory Ready
```

И

Таблица векторов меги8:

```
; ***** INTERRUPT VECTORS ************************************************
.equ   INT0addr   = 0x0001   ; External Interrupt Request 0
.equ   INT1addr   = 0x0002   ; External Interrupt Request 1
.equ   OC2addr   = 0x0003   ; Timer/Counter2 Compare Match
.equ   OVF2addr   = 0x0004   ; Timer/Counter2 Overflow
```

```
.equ   ICP1addr   = 0x0005   ; Timer/Counter1 Capture Event
.equ   OC1Aaddr   = 0x0006   ; Timer/Counter1 Compare Match A
.equ   OC1Baddr   = 0x0007   ; Timer/Counter1 Compare Match B
.equ   OVF1addr   = 0x0008   ; Timer/Counter1 Overflow
.equ   OVF0addr   = 0x0009   ; Timer/Counter0 Overflow
.equ   SPIaddr   = 0x000a   ; Serial Transfer Complete
.equ   URXCaddr   = 0x000b   ; USART, Rx Complete
.equ   UDREaddr   = 0x000c   ; USART Data Register Empty
.equ   UTXCaddr   = 0x000d   ; USART, Tx Complete
.equ   ADCCaddr   = 0x000e   ; ADC Conversion Complete
.equ   ERDYaddr   = 0x000f   ; EEPROM Ready
.equ   ACIaddr   = 0x0010   ; Analog Comparator
.equ   TWIaddr   = 0x0011   ; 2-wire Serial Interface
.equ   SPMRaddr   = 0x0012   ; Store Program Memory Ready
```

Как видишь у меги16 таблица другая. Адреса зовутся также, но вот на каждый вектор у меги16 отводится ДВА байта, т.к. у меги16 больше памяти и требуется более длинный адрес, чтобы покрыть все адресное пространства флеша.

Поэтому нельзя взять адреса меги16 и путем выбрасывания «лишних» адресов получить таблицу прерываний меги8, т.к. у ней вектора имеют другую длину. По крайней мере если работаешь на абсолютных адресах. Вместо абсолютных адресов лучше юзать дефайновые метки. Т.е.

.ORG INT0addr

а не

.ORG 0x0001 ; INT0 addr

Тогда компилер все сделает сам. Я так не делаю только по тому, что мне лень переписывать с нуля все это :) — у меня под все контроллеры с которыми я работал есть уже готовая таблица прерываний с абсолютными адресами которую я копирую в нужный проект. Хотя это и не очень красиво.

★ DI HALT

24 Август 2010 в 19:41

Таблицу векторов прерываний на конкретный камень надо брать из его def файла, а не от другого проца. Т.к. там и прерывания могут идти в другом порядке и чего то не хвататать и вообще смещения разные. В результате будет такой глюк который без опыта хрен найдешь.

---

smallghost

24 Август 2010 в 20:59

Спасибо, за подробный ответ!
Буду исправлять!

---

fokuz

15 Январь 2014 в 2:55

Как видишь у меги16 таблица другая. Адреса зовутся также, но вот на каждый вектор у меги16 отводится ДВА байта
А почему два байта, разве не 4?
Ведь адресация в словах
.ORG $002
RETI ; (INT0) External Interrupt Request 0
.ORG $004
RETI ; (INT1) External Interrupt Request 1

во флеше выглядит вот так:
000002 18 95
000003 FF FF
000004 18 95
000005 FF FF

Любовь Назарова

31 Август 2010 в 15:18

Спасибо огромное автору!

★ DI HALT

31 Август 2010 в 16:04

Пожалуйста :)

---

miRasH

11 Сентябрь 2010 в 20:33

Доступно и ясно. И как вам хватает нервов так всё детально описывать?)). Очень интересна,кстати,с точки зрения прерываний ATtiny28, к которой можно даже прилепить клавиатуру

Sadness

12 Сентябрь 2010 в 1:04

Помогите разобраться в коде:

это кусок кода прерывания по переполнению таймера T2 [ATmega8L]
устройство — таймер который просто отсчитывает вперёд секунды минуты и часы

при переходе с 23:59:59 на 24:00:00 должен происходить сброс переменных в ноли
чтобы было 00:00:00 но этого не происходит, таймер показывает 24:00:00 и
продолжает считать вперёд

;——Прерывание T2 [1c]——

```
tim2: sei
inc t_secl
cpi t_secl, 10
breq sh
reti

sh: ldi t_secl, 0
inc t_sech
cpi t_sech, 6
breq ml
reti

ml: ldi t_secl, 0
ldi t_sech, 0
inc t_minl
cpi t_minl, 10
breq mh
reti

mh: ldi t_secl, 0
ldi t_sech, 0
ldi t_minl, 0
inc t_minh
```

```
cpi t_minh, 6
breq hl
reti

hl: ldi t_secl, 0
ldi t_sech, 0
ldi t_minl, 0
ldi t_minh, 0
inc t_hrl
cpi t_hrl, 10
breq hh
reti

hh: ldi t_secl, 0
ldi t_sech, 0
ldi t_minl, 0
ldi t_minh, 0
ldi t_hrl, 0
inc t_hrh
cpi t_hrh, 2
breq hh2
reti

hh2: cpi t_hrl, 4
breq hh3
reti

hh3: ldi t_secl, 0
ldi t_sech, 0
```

```
ldi t_minl, 0
ldi t_minh, 0
ldi t_hrl, 0
ldi t_hrh, 0
reti
```

★ DI HALT

12 Сентябрь 2010 в 12:44

А прогнать в студии трассировку покомандно и посмотреть где код ведет себя неправильно религия не позволяет? Тут косяк в алгоритме обсчета, а не в обработке прерывания.

Sadness

12 Сентябрь 2010 в 14:00

при трассировке с начала этой подпрограммы в sh: или ml: курсор прыгал на другую подпрограмму задержки которая к этой не относится (почему я так и не понял)

★ DI HALT

12 Сентябрь 2010 в 14:05

А как ты туда попадал? Если так, то у тебя срыв стека где то.

Sadness

12 Сентябрь 2010 в 22:05

хм… стек даже не использовал

★ DI HALT

12 Сентябрь 2010 в 22:17

Т.е. даже не инициализировал? Замечательно. Значит твоя программа мертворожденная изначально. Т.е. тикать может она и тикает, но на большее будет неспособна. Т.к. по выходу из прерывания у ней сорвет стек, прога выполнить несуществующую инструкцию и перезагрузится. Можешь проверить. У тебя каждый выход из прерывания = перезагрузка. Разве что в регистрах значения не стираются как при аппаратном ресете. ПОтому то ты это и не заметил.

Перечитай еще раз статью. Внимательно. И тогда узнаешь, что стек ты юзаешь, даже если и не знаешь об этом :)

Sadness

15 Сентябрь 2010 в 12:39

Если б я его не инициировал у меня бы ни одна команда перехода не работала насколько я понимаю, а они работают, я просто в стек ничего не пихал, туда записывались только адреса для возврата

★ DI HALT

15 Сентябрь 2010 в 12:53

А покажи мне свою инициализацию стека.

Sadness

16 Сентябрь 2010 в 12:59

;——Таблица векторов прерываний——

```
rjmp main ;Reset
reti
reti
```

```
reti
rjmp tim2 ;Timer 2 Overflow
reti
reti
reti
reti
reti
reti
reti
reti
reti
reti
reti
reti
reti
reti
reti

;——Инициализация стека——

main: ldi temp, low(RAMEND)
out SPL, temp
ldi temp, high(RAMEND)
out SPH, temp
```

★ DI HALT

12 Сентябрь 2010 в 12:46

Хотя нет, налицо кривой выход из прерывания. Т.к. выходить тупо через RETI нельзя.

Где сохранение SREG и всех использованых регистров? Где их возврат «Как было» перед выходом?

★ DI HALT

12 Сентябрь 2010 в 13:07

Но и алгоритм обсчета тоже кривой. В частности hh3 никогда не выполнится.

Sadness

12 Сентябрь 2010 в 14:01

Спасибо DI HALT, разобрался это и был основной косяк

Sadness

12 Сентябрь 2010 в 14:02

Да там всего флаг I надо восстанавливать каждый раз я и подумал что этого хватит

★ DI HALT

12 Сентябрь 2010 в 14:06

Не только I но и Z, C, PE команда CPI меняет значение кучи разных флагов.

Sadness

12 Сентябрь 2010 в 22:08

правильнее было б конечно сохранять SREG, но во всей программе я SREGом не пользуюсь кроме его байта I ну и в этом прерывании cpi а так больше не используется

★ DI HALT

12 Сентябрь 2010 в 22:15

Наивный. У тебя там команда CPI которая меняет флаги Z, C, N, V, H — то есть почти все какие есть.

ФЛаги используются любыми другими командами которые определяют ветвление. Например BREQ оперирует флагом Z и если в основной программе есть какое то ветвление, то оно у тебя тоже полетит в тартар.

Sadness

15 Сентябрь 2010 в 12:49

Слегонца сменил структуру теперь там BRNE

```
tim2: in temp, SREG
push temp

inc t_secl
cpi t_secl, 10
brne tim2r

clr t_secl
inc t_sech
cpi t_sech, 6
brne tim2r

clr t_sech
inc t_minl
cpi t_minl, 10
brne tim2r
```

```
clr t_minl
inc t_minh
cpi t_minh, 6
brne tim2r

clr t_minh
inc t_hrl
cpi t_hrl, 10
brne t2c24

inc t_hrh
rjmp t2hcl

t2c24: cpi t_hrl, 4
brne tim2r
cpi t_hrh, 2
brne tim2r
clr t_hrh

t2hcl: clr t_hrl

tim2r: pop temp
out SREG, temp
reti
```

★ DI HALT

12 Сентябрь 2010 в 12:47

И зачем разрешать прерывания при входе в обработчик? Есть что то более важное?

Sadness

12 Сентябрь 2010 в 22:11

Всё исправил, всё тикает как должно, не глючит, спасибо =)

serg71277

29 Сентябрь 2010 в 16:48

У меня вопрос по прерываниях. Например: основная программа в один из моментов выполняет операцию, которая состоит из некоторого числа подпрограмм вложенных друг в друга. Можно ли в этот момент произвести внешнее прерывание(прервать выполняющуюся операцию)и по внешнему прерыванию выйти в совершенно другое место основной программы, не закончив выполнение прерванной операции.Так сказать грамотно сорвать стек?Если бы операция, которую прерывают была бы без подпрограмм, то проблем бы не было, а вот как быль в случае наличия подпрограмм?

★ DI HALT

30 Сентябрь 2010 в 2:27

В принципе это нормальная ситуация. Просто глубина стека возрастет на величину которую требует прерывание (адрес возврата, сохраняемые регистры). И если стек при этом не встретится с данными (переполнение стека), то возврат будет совершенно корректным и нормальным и программа потом спокойно выйдет из всех вложений и пойдет дальше. Т.е. гарантировано все будет Окей, главное иметь запас по глубине стека и все.

Запас вычисляется просто:
Просчитываем самое глубокое вложение — каждое вложение +2байта + затраты на самое громоздкое прерывание (2байта + все сохраняемые в стеке регистры, а если в прерываниях есть функции, то надо учитывать и это вложение) это даст максимальную глубину стека. Правда если разрешены вложенные прерывания, то надо добавить еще глубину возможного прерывания и так далее. Т.е. прикинуть вообще теоретическую вероятность того, насколько глубоко программа в принципе может зарыться в стек. И вот дальше этой границы данные не располагать ни при каких обстоятельствах, воизбежание.

★ DI HALT

30 Сентябрь 2010 в 2:30

А блин, не правильно понял вопрос.

Т.е. ты хочешь из глубокого вложения выйти в другое место по прерыванию? Тоже можно. Но тут надо знать: глубину вложения и где лежит нужный уровень стека. Тогда ты выходишь по прерыванию куда тебе надо, восставноив стек на нужную глубину. И все будет окей. Но это сложно, легко накосячить, а ошибка вылезет не сразу.

---

serg71277

30 Сентябрь 2010 в 12:41

Большое спасибо. Если можно, помогите найти иной путь решения задачи.Попробую детально: основная программа в один из моментов должна одновременно выполнять две функции(1-я динамическое свечение светодиодов с частотой 40 Гц и скважностью 1:8, и 2-я звуковое сопровождение разными тонами 400 Гц, 800 Гц с паузами между тонами).Реализовать мне это удалось без особых проблем, но получилось множество вложенных друг в друга подпрограмм. Устройство имеет кнопку внешнего прерывания. Выше приведенные функции выполняют роль свето-звуковой индикации и во время их выполнения рука тянется прервать процесс, нажав на кнопку. А теперь вопрос: можно ли реализовать эту индикацию иным способом? То есть без подпрограмм или хотя бы использовав их малое количество.

★ DI HALT

30 Сентябрь 2010 в 14:32

Нужна диспетчеризация какая нибудь. На флаговом автомате. например. И одной из ее задач-опрос кнопки. Звуки генерить таймерами через ШИМ например.

---

serg71277

30 Сентябрь 2010 в 17:17

Вопрос по поводу реализации звуков при помощи ШИМа.Я работаю ATtiny2313. Мне нужен не просто писк, а писк с регулировкой громкости. Я делаю это изменяя скважность. А в принципе требуется еще получить не просто писк, а какой то «живой» звук(все таки 21 век за окном). Кроме того писк режет ухо. Кроме того нужно динамически подсвечивать светодиоды во время звука. При этом не должно быть рывков в свечении. Подскажите, можно ли это сделать при при помощи ШИМа? Может у вас имеется какой то пример для наглядности.

Poligrafych

28 Октябрь 2010 в 2:02

А там в начале (возле пива) строка 20 OUT UDR,R16
21 RJMP Start

переходим на 12 Start: OUT UDR,R16 (получается без задержки)

★ DI HALT

28 Октябрь 2010 в 2:10

Спасибо, пофиксил.

---

sTARcRAB

14 Декабрь 2010 в 1:58

добрый вечер. ламерский вопрос про прерывания. скажем крутится в цикле код. в коде меняется переменная X. тикает также таймер. прерывание по переполнению таймера. во время прерывания берем значение переменой и в зависимости от нее меняем ШИМ на какой либо ножке. так МК будет работать?

★ DI HALT

14 Декабрь 2010 в 8:18

А чего нет то?

sTARcRAB

14 Декабрь 2010 в 10:26

ну незнаю:) с прерываниями не разбирался/не работал еще) Спасибо.

GVS

29 Декабрь 2010 в 15:21

Сталкнулся с такой штукой, что при инициализации стека указанным способо, в случае возникновения прерывания по петеполнению T0(в моем случае), мы переходим в обработчик, а вот обратно возвращаемся совсем не туда, откуда ушли.. Погоняв в студии понял, что когда мы уходим на прерывание, адрес возврата пихается в стек в ячейку 0x45F(mega8, последняя ячейка) и равен 0! Если же проинициализировать стек так:
http://easyelectronics.ru/repository.php?act=view&id=35
то адрес возврата в стеке записывается правильно, занимая два последних байта.
Ну а если до прерывания пробовать запихать в стек что-либо, то запись естественно начинается с адреса 0x45E..
Вот пример кода:
http://easyelectronics.ru/repository.php?act=view&id=36

Почему так происходит?

★ DI HALT

29 Декабрь 2010 в 17:17

Хрень какая то. У тебя много PUSH\POP в коде. Нигде не напутал с ними ничего?

Попробуй без них сделать вход в прерывание возврат обратно. Такое ощущение, что у тебя где то выполняется лишний POP и указатель стека выходит за границы памяти.

GVS

31 Декабрь 2010 в 1:13

Точно! Спасибо за наводку. Убрал из макроса «_outc» PUSH/POP и все встало на свои места! :o)

Vladimir.F

2 Февраль 2011 в 12:55

Здравствуйте. Подскажите пожалуйста, в чем может быть причина неправильного деления частоты нулевым таймером в Atmega16(вместо 40 Гц получается 39,5 Гц). Timer1 делит правильно. Других прерываний нет, только по переполнению. Пишу в AVR Studio на языке C.

★ DI HALT

2 Февраль 2011 в 17:02

1. От чего тактуется? Если от внутреннего RC то пол герца это еще очень хорошо.
2. Как делается деление? ВОзможные накладные расходы на код.

Vladimir.F

2 Февраль 2011 в 17:28

Кварц на 12 МГц.
Вот код:

```
#include
#include
#include
const unsigned char tab[]={0x05,0x01,0x03,0x02,0x06,0x04};
unsigned char StepCount=0;
unsigned char tmr1=0;
```

```
ISR(TIMER0_OVF_vect)
{
StepCount++;
if(StepCount>5)StepCount=0;
TCNT0=0x06;//отсчет 250 импульсов
tmr1++;
if(tmr1==25)/счет до 25
{
tmr1=0;
PORTB=tab[StepCount];//период делится на 6 тактов
}
}

int main (void)
{
DDRB = 0x07;
TCCR0 = 2;//предварительный делитель на 8
TIFR = 0;
TIMSK = 0b001;
GICR = 0;
sei();
while(1);
}
```

Если упростить и оставить только деление на 8 и 250 вместо 6 КГц получается примерно 5930Гц

★ DI HALT

2 Февраль 2011 в 19:34

TCNT0=0×06;//отсчет 250 импульсов

В реальности 249 т.к. на 255 будет уже переполнение.

Vladimir.F

2 Февраль 2011 в 19:46

Тогда частота выше была бы.

Vladimir.F

2 Февраль 2011 в 19:48

Я наоборот писал 0x09, тогда было ближе к 40, но все равно с погрешностью

Vladimir.F

4 Февраль 2011 в 10:04

Проблема решена, настроил таймер в режим CTC.

TU22

8 Март 2011 в 15:40

Пока контроллер занимается вычислением или гоняет байтики по памяти — АЦП может яростно оцифровывать входное напряжение, USART меланхолично передавать или принимайть байтик, а EEPROMка неспеша записывать в свои тормозные ячейки очередной байт.

5 баллов!!! Жизненное описание! Молодец!

tomba

23 Апрель 2011 в 21:46

Так должно работать(см. ниже)? Здесь подпрограмма в подпрограмме. Непонятно. В симуляторе работает, а на кристалле(мега8) нет. И ещё вопрос о команде CALL. Согласно даташиту, таковой команды нет и компилер ругается, а есть только RCALL и ICALL. Может дело в вызовах?

main:
RCALL podproga
…
…
…

podproga:
…
…
RCALL pod_podproga
…
…
RET

pod_podproga
…
…
…
RET

★ DI HALT

24 Апрель 2011 в 8:28

Стек инициализировал?

Если оно так как у тебя написано (без зацикливания) то работать не будет Суди сам:

Сначала у тебя вызывается подпрога. Потом из нее выходит и дальше уже по очереди выполнения попадаем в тело подпроги оно выполняется еще раз, вызывается под_подпрога, а потом опа внезапно RET которого не ждали и стек срывает.

А так, никаких проблем не вижу. Вызов вложенных прог может быть очень глубоким — пока стека хватит (у тини11/12 правда стек в три уровня всего)

---

tomba

24 Апрель 2011 в 19:01

Стек инициализировал. Основная программма зациклена, и подпрограммы выполняются только когда их просят. В симуляторе работает всё отлично.
На кристалле проверил — под_подпрограмма вызывается и работает, а вот с неё уже выйти не может. Может AVRка глючная?

---

tomba

24 Апрель 2011 в 19:31

```
laten:
nop
nop
nop
nop
dec r25
CPI r25, 0
BRNE laten
RET
```

Это подподпрограмма. из неё не может никак выйти.

★ DI HALT

24 Апрель 2011 в 23:17

Странно. Должно выйти по обнулению R25 может его кто в прерывании оперативно заново наполняет?

tomba

25 Апрель 2011 в 1:16

Шил-шил — теперь вообще ни из какой подпрограммы не возвращается. Скорее всего что-то с кристаллом.

nes

16 Август 2011 в 19:29

Вопросы по вот этому куску кода:

```
RCALL Wait

OUT UDR,R16
RCALL Wait
OUT UDR,R16
RCALL Wait
OUT UDR,R16
RCALL Wait
RJMP Start ; Зациклим программу.

Wait: LDI R17,Delay
M1: DEC R17
NOP
```

BRNE M1
RET

Итак, как компилятор понимает, что Wait это подпрограмма, а M1 всего лишь метка? Они объявляются одинаково ведь. Неужели только за счет того, что ранее мы сделали RCALL Wait? Получается, что если бы мы сделали RCALL M1, выполнялся бы вот этот кусок:
M1: DEC R17
NOP
BRNE M1
RET
правильно?

---

nes

16 Август 2011 в 19:32

а, ну сделал RCALL M1, получилось все, как я сказал. Тогда вы как опытный программист на асемблере поделитесь впечатлениями: не сильно ли путаются метки и подпрограммы в голове?

★ DI HALT

16 Август 2011 в 20:05

С точки зрения компилятора нет понятия метки подпрограммы и метки перехода. Есть лишь одно понятие — Адрес. Ну и метка этого адреса. А уж ссылаешься на нее переходом или вызовом подпрограммы не имеет значения. Разница лишь в том, что подпрограмма пихает данные возврата в стек и потом должен быть RET который вернет данные обратно, иначе рано или поздно будет срыв стека.

★ DI HALT

16 Август 2011 в 20:06

Да, все так и будет. Иногда я делаю подпрограммы с двумя и более входами и одним выходом. А вход делаю исходя из того как мне удобно.

X-Ray

18 Сентябрь 2011 в 22:33

Чтобы дальше мне не запутаться сразу вопрос.
LDI R17,Delay ; Загрузили длительность задержки
M1: DEC R17 ; Уменьшили на 1
NOP ; Пустая операция
BRNE M1 ; Длительность не равна 0? Переход если не 0

Почему здесь BRNE работает? Прочитал описание команды в даташите. «if (Z=0)then PC<-PC+k+1». Z подразумевает пару регистров R30 и R31? Тогда как программа понимает что нужно сравнить с 0-ём именно число из R17? Или я не правильно понимаю даташит?

★ DI HALT

18 Сентябрь 2011 в 23:16

В данном случае Z это не регистровая пара Z, а флаг Zero регистра состояния процессора SREG. Все операции сравнения и условий оперируют только регистром SREG

X-Ray

18 Сентябрь 2011 в 23:46

Насколько помню Вы еще не описывали SREG, где можно почитать?

★ DI HALT

20 Сентябрь 2011 в 14:21

Да ну конечно, внимательней надо читать курс :)
http://easyelectronics.ru/avr-uchebnyj-kurs-flagi.html

X-Ray

18 Сентябрь 2011 в 23:36

Проблема номер два. Возможно связана с новой версией студии.
При вызове подпрограммы с помощью RCALL по вашему примеру, в стеке почему то добавляется значение не 00 07, а 20 07. При этом программа работает правильно, после RET переходит именно туда куда и требовалось. А если испортить подпрограмму по вашему примеру и добавить в стек 0, при этом получится стек вида 00 20 07, программа также как и у Вас перейдет в начало. Если вставить в стек значение из R16, то стек будет выглядеть как 32 20 07, но при этом если переключить вид отображения байтов в студии с 1-Byte Integer на 2-Bite Integer, то этот стек примет вид 3200 0720. То есть здесь 07 выше 20. Что то не понятно. Как вариант можно конечно переставить на студию 4-ой версии, но хочется сразу в новом ПО разобраться.

★ DI HALT

20 Сентябрь 2011 в 14:25

Видимо где то стоит смещение в виде ORG и реальный адрес подпрограммы не 00 07 как у меня, а 20 07.

не забывайте, что при отображени памяти в виде слов они переворачиваются. Т.к. действует правило little endian т.е. младший байт по младшему адресу, тогда как в режиме 2 byte integer показывается все в человеческом представлении, словами, где старший байт должен быть на первом месте.

X-Ray

20 Сентябрь 2011 в 20:01

Спасибо за предыдущий ответ, хоть я уже сам понял что нужно было быть внимательнее))

По поводу последнего.
ORG в программе вообще не стоит. Поэтому так и не понял где копать. И появился еще один вопрос, может опять связанный с новой студией) Флаг RXC не ставится. При этом тот же RXCIE ставится щелчком и убирается. В UCSRA можно менять только два последних байта (U2X и MPCM). При этом по стандарту стоит флаг (убрать опять же нельзя) UDRE. В UCSRB можно менять все флаги кроме предпоследнего. Может что то ограничивает смену этих флагов?

★ DI HALT

20 Сентябрь 2011 в 20:23

Тут хз. Может приколы пятой студии. Флаг может и не ставится, главное чтобы переход был.

---

ek50hey

11 Ноябрь 2012 в 13:11

Поставил 6 версию (она теперь не AVR Studio а Atmel Studio называется), так там вообще бардак с этим делом. Ради эксперимента сделал около 150 строчек кода (NOPами заполнил) и попробовал добавить PUSH R16 (R16 = 01). При этом в стеке получилось 01 20 07 (вместо 01 00 07). После RETa перекинуло не на 01 00, а на 01 20. (при этом если стек не срывать то переходит на 00 07 вместо 20 07).

BigBoy

4 Ноябрь 2011 в 19:08

Вот кстати ошибка в тексте «Смотри, в стек пихнулось число 0×000007, указатель сместился на два байта и стал 0×035D, а контроллер сделал прыжок на адрес Wait.» Адрес указателя не 0×035D а 0×045D

★ DI HALT

4 Ноябрь 2011 в 21:27

ок

__bl__

23 Ноябрь 2011 в 21:33

Здравствуйте. Есть 2 вопроса по программированию. Контроллер МЕГА32А, Avrstudio 5.0.
1. Выполняется ли обработка прерывания таймера (0, 1, 2) если долго выполняется прерывание INT (0, 1, 2)?
2. Контроллер находится в пластиковой коробке на улице (постоянно). коробка не герметична. Сейчас температура упала до -2 град. Контроллер включается через 1 или 2 недели. После простоя не подаёт признаков жизни, на программатор не отзывается (программатор AVR Easy 3.2). Но прошивка заливается. После перезаливки прошивки работает нормально. При следующем запуске через неделю нужно опять заливать прошивку. Что происходит с контроллером? Контроллер служит тестовым образцом и прошивка заливалась в него раз 200.

★ DI HALT

25 Ноябрь 2011 в 0:29

1. По дефолту нет. Т.к. одно прерывание запрещает другое. Но можно в обработчике прерывания INT сделать SEI и разрешить. Правда это быдлокод. Прерывания должны быть короткими, иначе глюков не оберешься.
2. Трудно сказать. По идее срброс должен помогать.

__bl__

25 Ноябрь 2011 в 0:37

1. В предыдущей версии своего ПО вынес код из обработчика прерывания (как положено). В прерывании сбрасывал флаг и процедура запускалась в основном цикле. Такой код давал большую глючность. порадокс.

2. Уточните пожалуйста, что за «сброс».

3. Не подскажите где оперативно на русском языке можно получить ответы на вопросы о контроллерах АТМЕЛ AVR серии?

★ DI HALT

25 Ноябрь 2011 в 1:04

1. Значит глючно был написан :)
2. просто reset системы. У вас на еепром не завязано ничего? а то у аврок почему то часто епром косячит.
3. Нет таких мест, разве что форумы.

__bl__

25 Ноябрь 2011 в 1:38

В еепром ничего не записано. Программатором AVRISP 3.2 не удаётся считать прошивку, проверить фьюзы, даже Chip-ID не считывается. Но прошивка работает. После повторной прошивки контроллер восстанавливает свою работоспособность.
Контроллер запаян на макетке AVR-P40-8535. Чуть позже впаяю новый.
Как лучше впаять контроллер сразу в макетку или можно в цанговую панель?

NPLM

30 Ноябрь 2011 в 22:56

Здравствуйте, поставил AVRstudio 5 и обнаружил, что симулятор не разрешает сделать прерывания от переферии, тыкание по битам результатов не дает, при этом внешние прерывания выставляются легко. В четвертой студии все было нормально. Подскажите пожалуйста где я туплю…

__bl__

30 Ноябрь 2011 в 23:14

Добрый вечер! Если я правильно прочёл Atmelовские документы, то симулятор в AVRstudio 5 остаётся недописанным.

NPLM

30 Ноябрь 2011 в 23:29

Спасибо, значит остаемся на четвертой…

DI, про недоделки я догадался, но думал, что всё ограничивается местами жопным интерфейсом.

★ DI HALT

30 Ноябрь 2011 в 23:34

С интерфейсом там то как раз все замечательно, но интерфейс они взяли готовый от микрософта. А вот свои же шняжки убили на корню.

NPLM

30 Ноябрь 2011 в 23:58

Ну не знаю, лично мне только внешний вид понравился, с семеркой больше сочетается. :)
А вот работа с памятью стала неудобной, где разбиение на колонки? Я не нашел. Многие окна потеряли альтернативные варианты отображения. Вроде мелочи, но как-то напрягает с непривычки, может конечно я не туда смотрю и тыкаю. Есть конечно приятные фичи, но невозможность сделать прерывания убила…
Вобщем-то 4-ая студия вполне себе устраивает, 5-ую поставил только эксперимента для.

★ DI HALT

30 Ноябрь 2011 в 23:26

АВР Студия 5 недоделаное УГ.

yuzd

27 Декабрь 2011 в 19:37

Приветствую!
Подскажите по стековым переходам, есть у меня прерывание
TIM0_COMPA:
код
код
код
reti
при выходе из него я всегда хочу возвращается на M1, а не туда откуда пришел, я так понимаю нужно сначала забрать со стека предыдущий адрес перехода, потом положить свой на M0? Напишите плиз как это правильно делать.

★ DI HALT

28 Декабрь 2011 в 0:47

Месье знает толк в извращениях. Вообще это будет типичный быдлокодинг, т.к. такой возврат ни к чему хорошему не приводит, считай у тебя обрывается контекст на пол пути, а дальше прогу выносит в M1 и что там было — огнем гори. Чуть накосячишь и глюки ловить не оберешься.

Но коль желаешь потрахаться…

Раз у тебя состояние фоновой программы побоку, то сохранять регистры в стеке смысла нет никакого (им один хер капец). Потому достаточно просто pop-нуть старый адрес, куда угодно, в любой свободный регистр, он нам не нужен. И push'нуть туда новый адрес возврата. Правда надо посмотреть в каком порядке байты адреса пихаются в стек. Но тут тебе студия в помощь.

---

yuzd

28 Декабрь 2011 в 16:54

Думаю что вот так будет правильно?
POP YL
POP YH

LDI YH,low(M0)
PUSH YH
LDI YL,High(M0)
PUSH YL
reti
Попробовал, по карйней мере протеус не ругается на переполнение стека.
Программка сама маленькая, и состояние основной программы после прерывания действительно побоку, т.к. при заходе в прерывание забирается результат счета с фоновой программы, и там уже по условию с ним работается, потом надо вернутся в начало (M0) и начать все считать заново, пока не придет прерывание и не заберет результат.

★ DI HALT

28 Декабрь 2011 в 17:05

Ругаться он и не будет. Надо проверить только что порядок вбивания байт в стек верен.

---

der_poul

21 Январь 2012 в 10:08

Вот пытаюсь я проссимулировать сей процесс(приход данных по UART) Но никак не могу поставить бит RXC, не ставится он и все =(И в UDR ничего воткнуть не могу. AVR Studio 5.

---

akocur

28 Январь 2012 в 14:47

у меня несколько вопросов.
1. Что означает вот эта операция (1<<RXCIE)?
2. Как узнать в какие регистры должен прийти байт, что бы возникло прерывание? Как в даташите их называют? Что в реальности подразумевается под тем что в регистр свалился байт? К какой то ножке микросхемы подали напряжение? В даташите не нашел вывод RXC.

P.S. Может ранее об этом и спрашивали, но мне до сих пор не видны чьи либо комментарии.

Veter0k

8 Февраль 2012 в 14:49

Не могу понять почему не получается выставить ручками бит RXC? AVR Studio 5. Кто еще сталкивался с этой проблемой? какие методы ее решения?

NPLM

8 Февраль 2012 в 15:00

Потому что симулятор в 5-ой студии недоделанный, овечали уже выше.

Alex0720

28 Май 2012 в 21:31

Доброго времени суток.
При прогоне в AVR Studio 4 примера с прерываниями для Atmega16 , флаг RXC (который выставляем в рукопашную), сбрасывается самостоятельно сразу после команды RX_OK: IN R16,UDR
Но при тех же действиях для Atmega128a, RXC (только там он обзывается RXC0) почему остается активным постоянно. Имена регистров в программе, понятное дело были адаптированы под Atmega128a и весь пример проходит одинаково с Atmega16.. Кроме этого бага…

★ DI HALT

29 Май 2012 в 7:58

Вполне может глючить студия. Я там такие приколы наблюдал. Проверь в железе.

Alex0720

29 Май 2012 в 20:41

Нашел косяк..пазор на маи сидины..притупил.. %)
UDR0 не читал после вызова прерывания..теперь все ок..

xbco

2 Июнь 2012 в 18:43

Привет DI HALT спасибо за твои труды, которые пишешь. Огромнейший RESPECT! Читал книгу Д. Мортана по AVR несколько раз, но с твоими подробными статьями не сравниться. Очень понравилось как написана о организации архитектуры мк и памяти.
У меня возник табл с вектором прерывания, ты писал: «Нажми F11, чтобы сделать еще один шаг по программе… Опа, стрелочка улетела в таблицу векторов, как раз на вектор
1 .ORG $016
2 RJMP RX_OK ; (USART,RXC) USART, Rx Complete
» но у меня в авр студии v4.13 стрелка улетает на RETI между
.ORG $00C
—> RETI ; (TIMER1 COMPA) Timer/Counter1 Compare Match A
.ORG $00E
В чем может быть проблема?
Ссылка на zip с проектом авр студии: http://narod.ru/disk/51455709001.15836ea55584f16b63ca3fcbe380176c/testing2.zip.html

★ DI HALT

5 Июнь 2012 в 9:01

А тип МК Правильно выбран в симуляторе? Плюс что то ты намудрил с ORG. Вектора же жестко заданы на конкретные адреса, Потому поставь ORG000 и пропиши ВСЕ вектора вообще.

limburan

3 Сентябрь 2012 в 4:21

Приветствую! Я поставил себе Atmel Studio 6

Тестирую твой код и замечаю вот что — SEI не устанавливает флаг разрешения прерываний. Почему такое может быть?

Но и это не все. Даже если его установить вручную, при установке флага RXC в регистре UCSRA он тут же сам снимается на следующий же шаг и зацикленный NOP продолжают выполняться дальше как ни в чем ни быловало, прерывания не происходит.

Как так?

★ DI HALT

3 Сентябрь 2012 в 13:12

Поздравляю, сноси эту недоделку нах :)

Если SEI не ставится видимо косяк студии. Этот бит никто не может снять или поставить. Только ты сам.

А вот RXC снимается сам автоматом, сразу же по входу в обработчик прерывания RXC

---

limburan

3 Сентябрь 2012 в 16:47

Спасибо за быстрый ответ Я тут как раз думал снести и поставить более старую версию, но колебался, а тут прочитал твой ответ.

Еще кстати на одном форуме мне написали вот что:
«Я не знаю, что там с SEI, но что ты забыл сделать — включить приёмник битом RXEN в том же регистре UCSRB»

Что скажешь? Это действительно обязательно делать?

★ DI HALT

3 Сентябрь 2012 в 19:16

Да, обязательно. Включить RXEN для приема, TXEN для передачи. Также, если нужно, там же надо включить прерывания для прием, передачи и опустошения буфера.

limburan

3 Сентябрь 2012 в 18:34

Снес 6ую студию, поставил 5.1, все заработало как надо.

Neonorama

9 Сентябрь 2012 в 12:24

День добрый. Да с 6-ой студией явные проблемы. Прерывания не включаются, установка флагов вручную не помогает. Буду ставить 4-ку, а то целый час мучился, пока комменты не прочитал :).

Silaev

4 Октябрь 2012 в 19:51

Чтобы отладчик Atmel Studio 6 начал реагировать на прерывания надо:
В меню «Tools | Options | Debugger » поставить «Mask interrupts while stepping» на false.
Говорят глюк 6 Студии.

d.smirnov

8 Апрель 2013 в 19:12

Большое спасибо. Моск мой кипел до Вашего комментария

---

nikom

21 Декабрь 2012 в 13:58

Привет, DI!
Использую ATmega168. Таблица прерываний чуть больше, чем у ATmega16. Сначала сделал как у тебя, добавив дополнительные вектора. Получил Overlap in .cseg.
Перечитал пост, скопировал метки. Тоже не помогло.
Причём что интересно overlap только до адреса 0x28:
Error 1 Overlap in .cseg: addr=0x0 conflicts with 0x0:0x28 C:\AVR_Studio_5\MyCandle\MyCandle\MyCandle.asm 11 0 MyCandle
…
Error 20 Overlap in .cseg: addr=0x26 conflicts with 0x0:0x28 C:\AVR_Studio_5\MyCandle\MyCandle\MyCandle.asm 49 0 MyCandle

Подскажи где я накосячил.

★ DI HALT

21 Декабрь 2012 в 15:46

http://easyelectronics.ru/repository.php?act=view&id=102

У меня так. Скопировано из рабочего проекта

★ DI HALT

21 Декабрь 2012 в 15:42

Где то у тебя пересекаются адресные пространства. Покажи этот кусок кода, где ты вектора определяешь.

nikom

21 Декабрь 2012 в 21:41

```
.include «m168def.inc»
.include «macro.asm»
.include «coreinit.asm»
; RAM =================================================
.DSEG
SW_DOWN: .byte 1

; FLASH ===============================================
.include «vectors.asm»
; Interrupts ==========================================
; End Interrupts ======================================
Reset: LDI R16,Low(RAMEND)
OUT SPL,R16

LDI R16,High(RAMEND)
OUT SPH,R16
```

В файле vectors.asm прописал твой код (до этого то же самое было только в моём исполнении).

★ DI HALT

22 Декабрь 2012 в 0:13

Ну сравни построчно. Где то найдешь косяк. Может затупил и скопипастил один и тот же адрес.

---

nikom

24 Декабрь 2012 в 11:36

Спасибо! Проблема решилась всего лишь переносом включения файла coreinit.asm — поставил после vectors.asm. (подсмотрел у тебя же и потом уже понял, где накосячил)

Moonshiner

5 Январь 2013 в 2:26

Добрый день.
Помогите, пожалуйста, разбираюсь с курсом и в качестве МК взял ATmega1284P. Все дело запускал в 6-ой студии и багов замечено не было до момента симуляции прерывания от UART-а. При попытке выставить RXC0 он на следующем же шаге сбрасывается и не происходит перехода в обработчик. Код инициализации:

```
Reset: LDI R16,low(RAMEND)
OUT SPL,R16
LDI R16,high(RAMEND)
OUT SPH,R16

SEI

LDI R17,(1<<RXCIE0)
STS UCSR0B,R17
```

★ DI HALT

5 Январь 2013 в 9:23

При инициализации ты ставишь бит RXCIE, но остальные то биты (RXEN например) ты сбрасываешь в ноль. Т.е. выключаешь уарт вообще. По идее на прерывание это не должно повлиять и раз флаг есть, то должно сработать, но раз речь идет о эмуляции (тем более о студии 6 которая уг и эмулирует совсем криво), то может и не заработает.

Правильно ставить бит надо через чтение-модификацию-запись. Т.е. читаешь UCSR0B дальше по OR накладываешь на него (1<

Moonshiner

5 Январь 2013 в 22:41

Она вообще странная. Изначально стек инициализирован, что бы ты ни написал и библиотеку МК не обязательно подключать. Поэтому, когда начал осваивать курс, не понимал, к чему эти строчки.
Кстати, с ATmega16 только с установленным битом RXCIE работает, другие модели не проверял.

Demyan

5 Март 2014 в 11:18

В 6-й версии Студии была аналогичная проблема. Поставил версию 4.19 (взятую на оф. сайте Атмела). Пока всё ровненько… По крайней мере нет баги с прерываниями от UART-а. Изучаю курс дальше. :):):)

dantistus

3 Июль 2013 в 23:48

Я посмотрел по даташиту, RCALL выполняется 3 цикла, RET — 4 цикла. Я правильно понял, что вызов подпрограммы будет выполняться на 7 тактов дольше аналогичного куска кода без вызова подпрограммы?

★ DI HALT

3 Июль 2013 в 23:55

Так точно.

dantistus

7 Июль 2013 в 1:04

Огромное спасибо! Завтра буду пытаться подключить свою tiny2313 к компу. Буду сидеть на печи, посылать на МК байты, а он будет выводить их на диоды :)

RokkerRulsan

2 Август 2013 в 13:24

Добрый день! Вы не сказали, откуда мокроконтролер знает куда надо прыгать на конкретное прерывание. И почему таблица выглядит именно так? Почему каждое следующее прерывание смещено на 4 байта (2 слова)?

★ DI HALT

2 Август 2013 в 14:01

Потому что так повелели боги!

Таблица прерываний жестко зашита на этапе проектирования микроконтроллера. И изменить ее нельзя, точнее можно, но также передвинуть в фиксированное место (при бутлоадере, например). В других контроллерах, таблица прерываний бывает более гибкая и ее можно извращать по разному. Например, в каком-либо спецрегистре хранится адрес ее начала, а сама она может быть где угодно, но само смещение векторов и их порядок обычно остается всегда неизменным.

Valerii

13 Август 2013 в 12:56

А можно добавить заметку к исходнику с прерываниями, что работает только с 4-ой студией?
Я битый час парился сравнивал, перечитывал статью. В комментах оно упоминается, но будучи новичком и делая все по инструкции очень тяжело с такими ньюансами бороться.

Ilpax

15 Октябрь 2013 в 11:33

Протрассируй программу и сам увидишь, что сброс флага RxC происходит после выполнения команды

Если включить RXEN, RXC обрабатывается дважды o_O
Без RXEN флаг RXC сбрасывается сразу после перехода (или попытки перехода) даже без чтения из UDR

RokkerRulsan

10 Ноябрь 2013 в 21:43

Добрый день! Еще возник вопрос: во время вызова rcall, в памяти появляется не тот адрес, почему то старший байт равен 0x0207. Хотя вызываю с адреса 0x0007. AVR Studio 6. Возврат идет на правильное место. Но все равно как то неприятно. AVR Studio 4, такого поведения не замечено.
http://yadi.sk/d/uyukfSdOCTFey

RokkerRulsan

10 Ноябрь 2013 в 21:45

P.S. Не старший байт равен 0x0207, а старший байт не соответствует действительности.

★ DI HALT

10 Ноябрь 2013 в 21:55

Ну байты там, емнип, в обратном порядке лежат. А в студии 4 выглядит тот адрес с которого ушло?

RokkerRulsan

10 Ноябрь 2013 в 22:38

0x0007, я не понимаю откуда двойка берётся?

RokkerRulsan

10 Ноябрь 2013 в 22:46

Я немного ошибся, адрес в памяти 0x2007, Такого ядреса в пямяти вообще не существует. Максимум для mega 16 0x045f, Но вдруг я захочу поработать с этим адресом.

ulole

10 Ноябрь 2013 в 23:07

0x045F — может быть, это вершина стека для m16 после инициализации МК? Вы имеете ввиду ОЗУ? Или flash?

RokkerRulsan

10 Ноябрь 2013 в 23:11

Я про флеш. Посмотрите на картинку. Откуда в адресе возврата берется двойка? http://yadi.sk/d/uyukfSdOCTFey

★ DI HALT

10 Ноябрь 2013 в 23:18

Просто если в студии 6 симулятор глючит (а он там полный треш) то тут можно сколько угодно строить догадки. Что показывает студия 4 на этом же коде?

RokkerRulsan

10 Ноябрь 2013 в 23:27

Четвертая, показывает нормальный адрес возврата, то есть в памяти байты: 0x00 и 0x07. Откуда ушли, туда и пришли.

★ DI HALT

10 Ноябрь 2013 в 23:34

Ну значит это глюки симулятора 6й студии. Собственно то, почему я ей и не пользуюсь. Т.к. она кривая до безобразия.

ulole

10 Ноябрь 2013 в 23:48

Только что попробовал Ваш пример в 4.18 Студии,- все прекрасно. Стек работает как часы, rcall идет по нужному адресу, в окне Memory те значения, что и должны.
http://yadi.sk/d/-LhNYbu3CTnTf

RokkerRulsan

11 Ноябрь 2013 в 0:01

Да, в четвёртой студии все нормально, если есть возможность попробуйте в шестой.

---

ulole

11 Ноябрь 2013 в 1:26

Да, шестая Студия и у меня глючит.http://yadi.sk/d/bJCMlDZuCU5D7
Хотя косячное число на стеке никак не влияет на симуляцию, но все равно неприятно. Что это за число 0x20, откуда оно? Если затереть его нулями в окне Memory, возврат из подпрограммы происходит нормально. Если подправить второй байт адреса, то, соответсвенно, вернемся в другое место. Как и положено. Если не обращать внимания на эту двадцатку, то все тип-топ

RokkerRulsan

10 Ноябрь 2013 в 23:56

Еще обнаружил проблему в AVR Studio 6. В ручную не устанавливаются биты в регистрах ввода вывода. Например, следуя примеру из статьи, нужно самостоятельно выставить бит rxc для возникновения прерывания. Нажимаю щелчком мыши, бит загорается, но на следующем шагу перехода на таблицу векторов прерываний не происходит, продолжает выполняться бесконечный цикл. В AVR Studio 4 все работает.

Valerii

11 Ноябрь 2013 в 1:25

Это в комментах уже пару раз обсуждается, просто DiHalt ленится обновить статью (ну, или учит новчиков читать коменты внимательно :))

RokkerRulsan

11 Ноябрь 2013 в 2:17

Очень жалко, что шестая студия глючит. Шрифты в четвёртой плохие.

Demyan

4 Март 2014 в 12:22

Да-да-да…. Косяк студии конкретный… Пол дня убил, пока подтверждение своей догадки в коментах не нашёл… :(:(:(

Demyan

4 Март 2014 в 12:26

А из 4-ой какую лучше поставить? На сайте Атмела, последняя из четвёрок — 4.19. Как она работает? Кто юзал?

Prizrak

10 Март 2014 в 17:49

Уважаемый! То что Вы делаете — это просто супер! Статьи весьма интересны и познавательны. (Жаль что я это уже все знаю :-D)
Заметил я Ваших исходниках интересную особенность..
Вы после любых арифметических команд добавляете операцию nop. Эта дань какой-то старой традиции или же это действительная необходимость о которой просто не все знают? Допустим, СИшный компилятор пустых операций не добавляет. Поясните, пожалуйста!)

★ DI HALT

10 Март 2014 в 20:07

Где вы у меня такое увидели.

Prizrak

10 Март 2014 в 22:13

А вот прям в самом первом куске кода. В самом-самом начале. Просто мне недавно попался глючный контроллер. В нем арифметические команды начинали работать после добавления NOP после них. И тут я у Вас такое же увидел… Подумал, что не с проста это…

Prizrak

10 Март 2014 в 22:15

после операций декремента.

★ DI HALT

10 Март 2014 в 22:42

Вообще не знаю зачем я тут поставил команду. Может чтобы просто была, чтобы пример был не совсем уж коротким, никакого целенаправленного смысла нет.

Prizrak

10 Март 2014 в 23:20

А я этот сакральный смысл искал…))) Жаль разочаровываться..)) Спасибо за то, что Вы делаете!) Удачи!

Fossa

19 Май 2014 в 15:41

А вот такой интересный вопрос. Есть у меня самопальный частотомер на 16-ой меге. Крутится таймер 16-битный с полной тактовой частотой, когда переполняется — генерирует прерывание, по которому старшая часть апдейтится. Другое прерывание модулем захвата генерируется.

Что произойдёт, если оба события строго одновременно наступят (в пределах одного такта). Какое из прерываний раньше сработает — неизвестно будет, да?

★ DI HALT

19 Май 2014 в 15:53

Почему неизвестно? Известно. При событии в одном такте первым идет то ,чей адрес в таблице векторов меньше.

Valina

18 Июнь 2014 в 12:47

Доброе утро! Такой вопрос, где можно найти какой бит отвечает за вызов прерывания. Как я понимаю в даташите, только от его чтения у меня мозг закипает. Есть ли это сведённое в одно место? Ещё по поводу ошибок, есть ли описание всех ошибок?

★ DI HALT

18 Июнь 2014 в 13:07

Есть бит разрешающий прерывания вообще — он в SREG. И есть биты разрешающие прерывания по каждому источнику. Они все в регистрах соответствующей периферии. Изучайте регистры конкретной периферии (например таймера) и там найдете искомое. Там обычно два бита. Бит разрешения прерывание ***IE и флаговый бит прерывания по которому можно понять, что прерывание случилось (если, например, оно было запрещено и не обработалось само). Ну и флаговые биты прерываний надо обрабатывать и сбрасывать, чтобы прерывание не сработало еще раз. Правда в большинстве случаев это происходит автоматически, при вызове обработчика, при чтении из регистра или еще как (в периферии описано), но не всегда.

---

vlipt

2 Июль 2014 в 14:25

Микроконтроллер ATTiny 2313. Задача по прерыванию от кнопок, подключенных к PB1 и PB2 управлять светодиодом, подключенным к PB0. Часть кода

```
.org 0 ;Задание нулевого адреса старта программы
rjmp reset ;Безусловный переход к метке reset
.org 0x000B ;Адрес прерывания по изменению состояния выводов
rjmp pin_change ;Безусловный переход к метке pin_change

reset: ;Начало раздела инициализации контроллера
ldi r16,RAMEND ;Загрузка в регистр r16 адреса верхней границы ОЗУ
out SPL, r16 ;Копирование значения из r16 в регистр указателя стека SPL
sbi DDRB, 0 ;Установка 0-го бита в регистре DDRB в «1» (PB0 — выход)
ldi r16,(3<<1) ;Загрузка в r16 двух "1", смещенных на 1 разряд влево
out PORTB,r16 ;Включение подтягивающих резисторов на входах PB1 и PB2
out PCMSK,r16 ;Разрешение прерываний по изм. сост. выводов для PB1 и PB2
ldi r16,(1<<PCIE) ;Загрузка в регистр r16 единицы в разряд PCIE
out GIMSK, r16 ;Разрешение прерывания по изменению состояния выводов
```

Дальше пока не важно. По предпоследней команде в r16 загружается 0b00100000. А вот последняя команда out GIMSK, r16 в регистр GIMSK не записывает ничего. И в AVR Studio там одни нули и схема не работает. Почему не происходит запись в GIMSK?

---

vlipt

2 Июль 2014 в 14:26

PS в конце секции Reset стоит команда

```
sei ;Глобальное разрешение прерываний
```

★ DI HALT

2 Июль 2014 в 14:49

А ты в симуляторе ,часом не попутал тини2313 и ат90с2313?

---

vlipt

2 Июль 2014 в 16:39

Нет, проект создавался для ATTiny2313

---

vlipt

5 Июль 2014 в 4:46

Неужели всё так плохо? Вроде нарушений логики нет. Изначально эта программа была написана для ATTiny13. Я прогнал её в симуляторе и там в регистр GIMSK единица записывается. Для ATTiny2313 я только изменил адрес вектора прерывания согласно даташита. И единици из r16 в GIMSK перестала записываться.

---

vlipt

16 Июль 2014 в 10:52

Это явление наблюдается только в студии 5 и 6 (на 4-й не проверял, нету её). В техподдержке Atmel ответили, что проблему знают и может быть, наверное, когда-нибудь её устранят.
Эмпирическим путем удалось установить, что если в неиспользуемые разряды GIMSK записать единицы, то в значащие разряды будет записываться то, что нужно.

---

and_master

3 Август 2014 в 14:16

Привет, Di! Я так понял , что пока никто не разобрался почему в стек по твоей программе пихается 20 07 вместо 00 07. Студия 4.19. И еще вопрос, почему если после rcall следующей строкой поставить .ORG , допустим с адресом 0x0600, то в стеке по-прежнему лежат все

те же 20 07 и переход осуществляется правильно. Заранее извиняюсь, если глупость спросил)

★ DI HALT

3 Август 2014 в 14:34

Это похоже больше на глюк студии. Отладь в железе да посмотри. Возьми данные стека и отправь их по UART :)

and_master

3 Август 2014 в 15:03

Э.. Я еще не пришел к этому. Вот закажу у тебя плату и попробую). Еще два маленьких вопроса. Что будет при попытке записать в память на несуществующий адрес озу. Студия это просто игнорит, а как отнесется к этому мк? И еще. Если указатель стека будет ссылаться на область регистров, то можно ли через стек регистры изменять. Студия ошибок не выводит, но и не позволяет этого.

★ DI HALT

3 Август 2014 в 15:28

Вообще за выход из границ должен компилятор ругаться. По идее ничего не должно произойти. Хотя тут хз я не пробовал.
Стек теоретически можно навалить в регистры. Но на практике я это не проверял. Можешь опыты провести по извращенским практикам программирования :)

and_master

7 Август 2014 в 17:30

Привет Di. Ты пишешь: «Либо флаг скидывают вручную — записью в этот флаг единицы. Не нуля! Единицы! Подробней в даташите на конкретную периферию.». Это всем очевидно? В эмуляторе вроде все наоборот.

И еще. Для работы с авр какую плату из твоего ассортимента лучше выбрать. (с возможностью пошагового исполнения кода и отладки в железе) pinboard вроде нет в наличии. Сняли с производства?

★ DI HALT

8 Август 2014 в 1:20

Это от бита зависит. В некоторых сброс 1 в некоторых нулем. Надо по каждому в ДШ смотреть. В AVR обычно сброс идет 1.

Pinboard II + AVR. В комплекте будет идти плашка с JTAG отладчиком. А версии 1.1 уже нету, да. Сняли с производства. Хотя с рук наверное можно купить. Поддержка по ней еще есть :)

LynXzp

30 Сентябрь 2014 в 16:35

Вот в разделе **Мозговзрывной кодинг** нет ли случайно ошибки, в продпрограмме делается два дополнительных PUSH, но нет POP. Т.е. работать будет, но то место куда мы не хотели возвращается все равно вернемся, ведь адрес в стеке так и остался, только глубже теперь. Или же если бы вместо внезапного RET был бы IJUMP и RET было бы точно то же самое (если в вызываемой подпрограмме выход через обычный RET).

Хотя может и ошибаюсь, пишу на C, ассемблер чуть знаю только потому что без него никак. Но «такой кодинг» — хоть хлебом не корми :).

★ DI HALT

30 Сентябрь 2014 в 16:42

Ты прав, сначала два раза POP в пустоту сделать не помешает, чтобы не плодить мусор в стеке.

---

pav_s

16 Декабрь 2014 в 14:28

Подскажите пожалуйста, что может быть. Установлена последняя AtmelStudio 6.2 .
Установил SEI и RXCIE

Устанавливаю RXC, жму F11 — RXC сбрасывается PC прыгает на следующую инструкцию, как будто прерывания и небыло. По Cycle Counter и Stop Watch тоже не видно каких либо движений, просто перепрыгнуло на следующую инструкцию в основной программе и все. Текст программы один в один как в статье.

★ DI HALT

16 Декабрь 2014 в 23:04

Установлена последняя AtmelStudio 6.2 в этом вся и причина :) Последняя адекватная студия 4.19 только там симулятор нормально работал.

---

basel

24 Январь 2015 в 20:06

Я то же снес 6 студию, установил 5.1 Проблем пока не обнаружил.
Отличный материал.

DeGriz

3 Май 2015 в 0:54

Добрый день.
Столкнулся со следующим моментом, Program Counter находится на строчке RJMP M1, выставляю флаг RXC, нажимаю F11, флаг снимается, перехода на прерывания нет, PC переходит на M1 на первый NOP

M1: NOP
NOP
NOP
NOP
RJMP M1

Установлена AVR Studio 4.19

Интересно почему так происходит.

★ DI HALT

3 Май 2015 в 21:57

А прерывания все разрешены? И глобально и локально?

---

hawkone1

3 Июль 2015 в 20:55

»А оттуда уже прыжок сразу же на метку RX_OK, где мы забираем данные из регистра UDR в R17 и выходим из прерывания, по команде RETI.»
из UDR в »R16»

---

hawkone1

3 Июль 2015 в 20:45

Проблема решена?

---

kaki-malaki

11 Октябрь 2015 в 22:48

Здравствуйте товарищи!
пара нелепых вопросов от новичка:
1) в первом примере ди хальт показал, что у него в стек пихается адрес 07, у меня пихается 20 07, хотелось бы узнать откуда там 20 взялось и что в адресе является low, а что high?
2) в студии в окне I/O view крайний левый бит(все равно какого регистра) при наведении показывает, что он 7, хотелось бы уточнить,

этот 7 бит у нас является msb, так?
3) 1<<RXCIE — такие вот записи вводят малька в ступор, когда 1<<2 — значит что на 2 бита сдвинуть влево, в случае с RXCIE — это значит на 8 битов сдвинуть влево(или на 7)? в студии RXCIE был изначально пронулеван и единица оказалась в 7 бите, если бы он был не пронулеван, было бы пару едениц, то эти еденицы бывъезжали в результат в порядке очереди вместо 0?
Уже видел на сайте где-то схожее обсуждение, но найти его не удалось, так что прошу прощение за повторение

★ DI HALT

12 Октябрь 2015 в 2:16

1) Не 07, а 0007 это две большие разницы. Адрес двухбайтный и соответственно в стеке по два байта пихается. Программа у меня сверхкороткая в примере потому и адреса такие маленькие. У тебя, видимо, подальше, потому и числа больше.

Шо вы все такие ленивые то, блин. Ну не понимаешь чего то, возьми да поставь эксперимент, чтобы понять. Студия и ее отладчик для кого сделан? Поставь две метки на разные команды, друг за дружкой, у одной будет адрес N у другой N+1. И попереходи под отладчиком на первый и на второй, увидишь где у тебя low, а где high байты. То же и со сдвигами. Посдвигай под отладчиком да посмотри что происходит, какое число в итоге записывается в регистр при разных параметрах.

Методом запуска под отладчиком в режиме эмуляции решаются 99% начинающих затупов.

Ana-bio-z

18 Ноябрь 2015 в 18:58

Уважаемый Di, ты пишешь: «…во многих случаях не обязательно глобально блокировать все прерывания вообще, достаточно заблокировать то локальное прерывание, которое может нам нагадить».
У меня в основном цикле проверяется, выставленный в прерывании по переполнению таймера, флаг. При установленном флаге вызывается процедура в которой из оперативки в регистры выгружается двухбайтный результат АЦП. Чтобы получить оба байта гарантированно от одного результата преобразования хочу запретить, как ты предлагаешь, не все прерывания, а только прерывание АЦП. Но не соображу как правильно изменить один бит (ADIE) в PBB (ADCSRA) не изменяя остальные биты. Логические побитовые операции работают только с РОН. Команды CBR и SBR не дотягиваются до Memory Mapped регистров. Думал выгружать весь

ADCSRA в РОН, накладывать соответствующую битовую маску и снова загружать обратно. Но, проблема в том, что бит флага прерывания АЦП так же расположен в этом регистре. И если я таким образом сначала запрещу прерывания, а после работы с SRAM снова установлю в РОН бит ADIE и в момент переноса этого регистра обратно в ADCSRA завершится очередное преобразование, прерывание которого еще запрещено, а выставляемый в этот момент флаг прерывания я тут же затру, то прерывание потеряется!
Какие еще возможны варианты?
Конечно в данном случае проще на пару тактов запретить прерывания глобально, но все же хочется разобраться.

★ DI HALT

18 Ноябрь 2015 в 21:23

А ты внимательно почитай чем эти биты сбрасываются. Насколько я помню, флаги сбрасываются записью в них 1. Т.е. нулем ты его не изменишь вообще. Почитай ДШ внимательней на этот счет. А так чтение-модификация-запись, как обычно.

Ana-bio-z

19 Ноябрь 2015 в 3:05

Действительно, в датащите сказано, что помимо аппаратного сброса флаг можно сбросить записью логической единицы. Но тут же предостерегают, мол если вы выполняете чтение-модификацию-запись регистра статуса и управления АЦП, то наступающее прерывание может быть заблокировано.
«Alternatively, ADIF is cleared by writing a logical one to the flag. Beware that if doing a Read-Modify-Write on ADCSRA, a pending interrupt can be disabled. This also applies if the SBI and CB instructions are used».
То есть чтобы запретить прерывание АЦП нужно прочитать ADCSRA, сбросить ADIE, а так же (если вдруг единица) ADIF (чтобы не изменить его состояние), после чего записать обратно? Чтобы снова разрешить, читаем, устанавливаем ADIE, сбрасываем (если установлен) ADIF, пишем обратно? Ничего не напутал?

★ DI HALT

19 Ноябрь 2015 в 9:49

Именно. При установке ADIE тоже надо маскировать ADIF вдруг у тебя прерывание на ожидании висит. Чтобы его не потерять.

Ana-bio-z

19 Ноябрь 2015 в 9:57

Спасибо! Разобрался с еще одним пунктом.