AVR. Учебный курс. Операционная система. Пример.

AVR. Учебный курс | 10 Июль 2008 | DI HALT | 107 Comments

Отлично, с теорией работы ОС ознакомил. Устанавливать научил, осталось научить использовать весь этот конвеерно таймерный шухер. Чем я сейчас и займусь. Сразу берем быка за рога и формулируем учебно-боевую программу.

Тестовое задание:
Пусть у нас будет **ATMega8**, с несколькими кнопками. АЦП и подключеним к компу через UART. На меге будет три светодиода.

- Девайс должен при включении начинать мигать зеленым диодом, мол работаю.
- При этом раз в секунду сканировать показания АЦП и если показания ниже порога — Моргать красным диодом.
- По сигналу с UART с целью защиты от ошибок сделать по байту ‘R’ установку флага готовности, а потом, в течении 10ms если не придет байт ‘A’ сбросить флаг готовности и игнорировать все входящие байты кроме ‘R’. Если ‘A’ придет в течении 10мс после ‘R’, то отправить в UART ответ и зажечь белый диод на 1 секунду.

Вот так вот, не сильно сложно. Но мне просто лень делать что либо сложней, а для тестовой задачи сгодится.
Итак, что у нас есть:
Две фоновые задачи, которые выполняются всегда и постоянно:

- Мигать зеленым диодом
- Проверять показания с АЦП

И цепочки:

- Прерывание АЦП — Проверка условия — зажечь диод — погасить диод.
- Прерывание UART- Проверить на R — взвести флаг (ждать ‘A’ — зажеть диод — погасить диод) — снять флаг

Описываем задачи:

- OnGreen — зажечь зеленый
- OffGreen — погасить зеленый
- OnRed — зажечь красный
- OffRed — погасить красный
- OnWhite — зажечь бело-лунный
- OffWhite — погасить бело-лунный

- Reset_R — сбросить флаг готовности
- ADC_CHK — проверить АЦП.

Собственно я это так, для ясности. Обычно я по ходу дела обвешиваю нужным функционалом. Инициализацию портов и всего прочего я опущу — не маленькие уже, DDR всякие сами выставите. Буду указывать код кусками, с пояснениями что где, а всю картину увидите в исходнике.

Итак, начинаем наращивать мясо:
Добавляем первую задачу — **Мигать зеленым диодом.** Для этого любую удобную секцию из раздела Task берем и переименовываем по своему вкусу. Будем по порядку. Под раздачу пойдет первая — Task1. Но сначала надо прописать ее номер в дефайнах. Поэтому сразу же лезем в **kerneldef.asm** и вписываем там:

```
.equ TS_Idle    = 0     ;
.equ TS_OnGreen = 1     ;  <<<<
.equ TS_Task2   = 2     ;
.equ TS_Task3   = 3     ;
. . .
```

Затем возвращаемся к таблице задач. И в поле **Task1** вписываем нашу задачу. Переименовывая метку (это не обязательно, но иначе запутаетесь в этих бесконечных **Task-n**.
Сразу же описываем саму задачу — она простая, просто зажечь диод. Одна команда процессора — SBI
Следом идет макрос **SetTimerTask**, считай это командой **API** нашей ОС. Значит что спустя 500мс надо выполнить задачу **OffGreen**

```
; Tasks
Idle:           RET
```

```
;-------------------------------------------------------------------------
OnGreen:        SBI     PORTB,1         ; Зажечь зеленый
                SetTimerTask    TS_OffGreen,500
                RET
;-------------------------------------------------------------------------
Task2:          RET
                . . .
```

Но у нас нет такой задачи! Не вопрос, добавляем!
Прописываем сначала в дефайнах:

```
.equ TS_Idle            = 0     ;
.equ TS_OnGreen         = 1     ;
.equ TS_OffGreen        = 2     ; <<<<
. . .
```

Потом добавляем ее код в область задач

```
; Tasks
Idle:           RET
;-------------------------------------------------------------------------
OnGreen:        SBI     PORTB,1         ; Зажечь зеленый
                SetTimerTask    TS_OffGreen,500
                RET
```

```
;-------------------------------------------------------------------------
OffGreen:       CBI     PORTB,1         ; Погасить зеленый
                SetTimerTask    TS_OnGreen,500
                RET
;-------------------------------------------------------------------------
Task3:          RET
                . . .
```

Видишь **первая задача ссылается на вторую, а вторая на первую**. Одна зажигает, вторая гасит. В итоге, они будут кольцом запускать друг друга, а зеленый диод, повешанный на PB1 будет мигать с интервалом 0.5с.

Теперь надо вписать их в таблицу переходов каждого на свой номер в **kerneldef.asm**:

```
TaskProcs:      .dw     Idle            ; [00]
                .dw     OnGreen         ; [01] TS_OnGreen
                .dw     OffGreen        ; [02] TS_OffGreen
                .dw     Task3           ; [03]
                . . .
```

Отлично. Фоновая задача сформирована, теперь надо только ее стартануть. Помните секцию **Background**? Вот я ее держу именно для этих случаев. Так то можно откуда угодно сделать наброс. Но удобней это делать из одного места.
Вот там и делаем:

```
Background:     RCALL   OnGreen
```

Задача все равно сформирована как процедура, так что как зайдет так и выйдет, а таймер запустится и дальше по цепи. То есть задачу можно стартануть простым RCALL. Тогда она первый раз выполнится мгновенно. А можно и через отдельный макрос **SetTask** будет выглядеть так:

```
1   Background:     SetTask TS_OnGreen
```

И выполнится в порядке общей очереди. Т.к. проц только стартовал и очеред пуста, то практически сразу же.
или по таймеру.

```
1   Background:     SetTimerTask TS_OnGreen,10000
```

Тогда мигать начнет спустя 10 секунд после запуска. Обрати внимание на то, что прямым **RCALL** мы указываем на метку, а через API мы передаем идентификатор задачи.

Да, не помешает напомнить, что из себя представляют макросы **SetTimerTask** и **SetTask**

```
1               .MACRO SetTask
2               ldi OSRG, @0            ; В OSRG номер задачи
3               rcall SendTask          ; через событийный диспетчер
4               .ENDM
```

```
5
6
7                   .MACRO SetTimerTask
8                   ldi     OSRG, @0        ; В OSRG номер задачи
9                   ldi     XL, Low(@1)     ;
10                  ldi     XH, High(@1)    ; Задержка в милисекундах
11                  rcall   SetTimer        ; поставить таймер в очередь
12                  .ENDM
```

Видишь тут используется **OSRG** (R17) и пара **X**. Для задания времени. А также функция **SetTimer**. Функция безопасна — она все значения сохраняет в стеке, а вот макрос нет — при вызове макроса **убивается X и OSRG**. Обычно это не критично, но это надо знать и помнить. Особенно когда вызваешь это дело из прерываний.
А еще я показал тебе подробно эти макросы с целью намекнуть на то, что задачи можно ставить не только вручную, а еще и программным способом.
Например, брать из памяти цепочки идентификаторов задач и передавать их через OSRG в SendTask. Получится цифровая мегашарманка :) Главное «вращать барабан» не быстрей чем они выполняются, а то очередь сорвет. А там и до виртуальной машины не далеко… Хотя нет, к черту. Java программистов надо убивать апстену! =)

А дальше в том же ключе:
Добавляем задачу проверки АЦП. Разумеется прописываем ей TS номер в дефайнах, не буду показывать.

```
1       ; Tasks
2       Idle:           RET
3       ;-----------------------------------------------------------------------------
4       OnGreen:        SBI     PORTB,1         ; Зажечь зеленый
5                       SetTimerTask    TS_OffGreen,500
6                       RET
```

```
;-------------------------------------------------------------------------------
OffGreen:       CBI     PORTB,1         ; Погасить зеленый
                SetTimerTask    TS_OnGreen,500
                RET
;-------------------------------------------------------------------------------
 ADC_CHK:       SetTimerTask    TS_ADC_CHK,1000
                OUTI    ADCSRA,1<<ADEN|1<<ADIE|1<<ADSC|3<<ADPS0
                RET
```

Как видишь, она сама себя запускает каждые 1000мс, т.е. каждую секунду. А стартует там же, из секции **Background**

```
Background:     RCALL   OnGreen
                RCALL   ADC_CHK
```

Осталось дождаться прерывания, поэтому ставим в код прерывание по выполнению АЦП:

Кладу его рядом с остальными прерываниями:

```
ADC_OK:         push    OSRG
                in      OSRG,SREG       ; Спасаем OSRG и флаги.
                push    OSRG

                IN      OSRG,ADCH       ; Взять показание АЦП
                CPI     OSRG,Treshold   ; Сравнить с порогом
```

```
7                   BRSH    EXIT_ADC        ; Если не достигнут выход
8
9                   SetTask TS_RedOn        ; Запускаем мырг красным
10
11  EXIT_ADC:       pop     OSRG            ; Восстанавливаем регистры
12                  out     SREG,OSRG
13                  pop     OSRG
14                  RETI                    ; Выходим из прерывания
```

Появилась еще одна задача — зажечь красный. Не вопрос, добавляем, прописав везде где нужно. Я же тут укажу только исполнительную часть. Сразу же впишу и задачу гашения красного. Чтобы уж в одном флаконе.

```
1   ; Tasks
2   Idle:           RET
3   ;-----------------------------------------------------------------------------
4   OnGreen:        SBI     PORTB,1         ; Зажечь зеленый
5                   SetTimerTask    TS_OffGreen,500
6                   RET
7   ;-----------------------------------------------------------------------------
8   OffGreen:       CBI     PORTB,1         ; Погасить зеленый
9                   SetTimerTask    TS_OnGreen,500
10                  RET
11  ;-----------------------------------------------------------------------------
12   ADC_CHK:       SetTimerTask    TS_ADC_CHK,1000
13                  OUTI    ADCSRA,1<<ADEN|1<<ADIE|1<<ADSC|3<<ADPS0
14                  RET
15  ;-----------------------------------------------------------------------------
```

```
16  OnRed:          SBI     PORTB,2
17                  SetTimerTask    TS_OffRed,300
18                  RET
19  ;-----------------------------------------------------------------------------
20  OffRed:         CBI     PORTB,2
21                  RET
```

Как видишь, Красный зажигается от пинка с прерывания АЦП, а гаснет по собственному пинку через 300мс. Так как АЦП проверяется раз в секунду, то если порог будет ниже, то каждую секунду будет вызов задачи RedOn и светодиод будет моргать 0.3 секундной вспышкой. Если сделать длительность вспышки больше чем частота вызова прерывания АЦП, то диод будет просто гореть, так как прерывание АЦП будет постоянно обновлять ему таймер. До тех пор пока входное напряжение на АЦП не будет выше порога, тогда диод моргнет на свою выдержку и затихнет.

Так, что там у нас следующее? **UART**? Ну не вопрос! Считаем, что **UART** у нас уже проинициализирован и готов к работе. Не верите? Глянте в секцию **init.asm** Так что добавляем прерывание на прием:

```
1   Uart_RCV:       push    OSRG
2                   in      OSRG,SREG       ; Спасаем OSRG и флаги.
3                   push    OSRG
4                   PUSH    XL              ; SetTimerTask юзает X!!!
5                   PUSH    XH              ; Поэтому прячем его!
6                   PUSH    Tmp2            ; Ну и Tmp2 нам пригодится
7
8                   IN      OSRG,UDR
9                   CPI     OSRG,'R'                ; Проверяем принятый байт
10                  BREQ    Armed           ; Если = R  - идем взводить флаг
11
```

```
12                  LDS     Tmp2,R_flag     ; Если не R и флага готовности нет
13                  CPI     Tmp2,0
14                  BREQ    U_RCV_EXIT      ; То переход на выход
15
16                  CPI     OSRG,'A'        ; Мы готовы. Это 'A'?
17                  BRNE    U_RCV_EXIT      ; Нет? Тогда выход!
18
19                  SetTask TS_OnWhite      ; Зажечь бело-лунный!
20
21  U_RCV_EXIT:     POP     Tmp2
22                  POP     XH              ; Процедура выхода из прерывания
23                  POP     XL
24                  pop     OSRG            ; Восстанавливаем регистры
25                  out     SREG,OSRG
26                  pop     OSRG
27                  RETI                    ; <<<<<< Выходим
28
29  Armed:          LDI     OSRG,1          ; Взводим флаг готовности
30                  STS     R_flag,OSRG     ; Сохраняем его в ОЗУ
31
32  ; Запускаем по таймеру задачу которая сбросит флаг через 10мс
33                  SetTimerTask    TS_ResetR,10
34
35                  RJMP    U_RCV_EXIT      ; Переход к выходу
```

Не смотрите что прерывание такое страшное, просто тут проверка по кучи условий. Также не смущайтесь того, что выход из прерывания не в конце кода, а в самой заднице середине обработчика. Какая, собственно, разница? Никуда она из JMP не выберется. Зато сэкономил на

лишнем переходе :) А в коде можно и отбивку сделать :)))))
Ну и обратите внимание на то что я сохраняю в стеке. А именно рабочий регистр OSRG, Пару X, и дополнительный Tmp регистр. Забудешь что нибудь из этого — схватишь трудно уловимый глюк который вылезти может только через несколько месяцев агрессивного тестинга.

Что там осталось? Добавить задачи в сетку? Ну тут все просто:

```
;-------------------------------------------------------------------------------
OnWhite:        SBI     PORTB,3
                SetTimerTask    TS_OffWhite,1000
                RET
;-------------------------------------------------------------------------------
OffWhite:       CBI     PORTB,3
                RET
;-------------------------------------------------------------------------------
ResetR:         CLR     OSRG
                STS     R_Flag,OSRG     ; Сбросить флаг готовности
                RET
```

Элементарно! Вроде бы ничего не забыл. Задача решена, еще куча ресурсов свободных осталось. Еще без проблем, не затрагивая уже написаное, можно повесить опрос клавиатуры, вывод на индикацию.
Скомпильнул, прошил, все с первого раза заработало как надо. А мозг включать даже не пришлось.

Готовый проект под ATMega8, чтобы прошить и посмотреть на это в действии.

Ну как? Стоило оно того?

107 thoughts on “AVR. Учебный курс. Операционная система. Пример.”

Medved

11 Апрель 2009 в 10:28

Ура я понял для чего все это надо! Пешы исчо!

---

xrayman

11 Апрель 2009 в 18:18

Напиши, плз, статейку по реализации разветвленного меню для lcd 16×2 :)

★ DI HALT

12 Апрель 2009 в 19:30

А какая связь между разветвленным меню и LCD ?

---

xrayman

12 Апрель 2009 в 19:55

связь в хранении, представлении и выводе на этот самый lcd. Ведь, как я понимаю, реализация меню для семисегментника будет немного отличаться от меню для lcd. А разветвленное, чтобы было расточительно делать его через switch..case

---

★ DI HALT

12 Апрель 2009 в 20:30

Интересна сама реализация меню, а на чем осуществлять вывод не имеет значения.

xrayman

12 Апрель 2009 в 20:40

Ну в общем-то да, реализация самого меню более интересна, но… :) Некоторые, например, предлагают хранить полностью строки, другие — словарь, из которого, при выводе, составляется текст, третьи вообще предлагают какую-то невообразимую фигню и только четвертые молча созерцают.

★ DI HALT

12 Апрель 2009 в 21:48

Я бы хранил строки. Тем более вряд ли их будет так много. А программа которая бы эти строки компоновывала из «слов» займет куда больше места. Да и отлаживать ее надо.

xrayman

12 Апрель 2009 в 22:53

Я, вот, тоже храню строки, пока 44 строки по 16 символов. Но, после того как, еще не до конца сделанное, меню стало занимать 1856 слов, меня начали терзать смутные сомнения… А так как я в программировании мк начинающий, хотелось бы посмотреть как такие вещи делают люди опытные. Чтобы укрепить или развеять мои сомнения :)

★ DI HALT

12 Апрель 2009 в 23:08

44*16=704 байта или 325 слова. Гхм, откуда у тебя там взялось 1856слов…

xrayman

12 Апрель 2009 в 23:16

ну это текст 352 слова, плюс еще собсно реализация меню, плюс еще какая-то фигня, вот и набралось… Но мне тоже кажется, что многовато.

★ DI HALT

12 Апрель 2009 в 23:18

Гхм. Чето много кода. Слишком много. На Си поди писал?

---

xrayman

12 Апрель 2009 в 23:23

ага, на codevisionavr, оптимизация по скорости
хотя пробовал адаптировать под winavr, там вообще получалось под 2100 слов

★ DI HALT

12 Апрель 2009 в 23:27

Оптимизируй по размеру. А то оптимизатор по скорости тебе для прикола все циклы в линию копипастом развернет :)))) Или еще какой прикол отмочит.

---

xrayman

12 Апрель 2009 в 23:32

пробовал, получается 1664 слова. Разница не очень большая, да и не интересно так :)

★ DI HALT

12 Апрель 2009 в 23:42

Хм. Выкини все библиотечные функции, напиши все с нуля сам. Поди какая нибудь херь вроде rprintf юзается.

---

xrayman

12 Апрель 2009 в 23:51

Тоже пробовал. Из библиотечных функций использую только delay_us и delay_ms. Соответственно задержка на число микро- и мили-секунд. Когда я писал свои, размер получался в пределах +-1 слова от нынешнего. Если их вообще убрать получается 1810 слов, т.е. разницы нету вапще.

★ DI HALT

13 Апрель 2009 в 0:17

Ну значит навертел чего то там много. С другой стороны, какой у тебя МК?

---

xrayman

13 Апрель 2009 в 0:20

мега 48

★ DI HALT

13 Апрель 2009 в 0:27

Всегда можно заменить на мега168 :))))))))))

---

xrayman

13 Апрель 2009 в 0:36

Можно, конечно, но в тестовую плату впаяна 48, а выпаивать ой как лень. Так что хотелось бы впихнуть все меню в нее :)

К чему мы, собственно, пришли — какая-то у меня кривовастенькая реализация меню. Напиши, плз, статейку по реализации сложного меню ;)

★ DI HALT

13 Апрель 2009 в 1:00

Да я то может и напишу. Только будет она на базе RTOS и голом ассемблере.

Адаптируешь? ;)

xrayman

13 Апрель 2009 в 1:07

Адаптировать не проблема, главное ж в этом деле — идея. Да и на ассемблере я писал немного, правда для тини15, но это мелочи :)

★ DI HALT

13 Апрель 2009 в 1:29

А общая идея там — хранить все дерево в памяти в виде массива. У каждой ветви будет номер по которому, как по смещению, вычисляется текстовая строка из памяти и связаный с ней номер задачи для RTOS. Самое сложное это увязать все дерево в одномерный массив. Особенно сложно (точнее затратно для памяти) это будет в случае если дерево неполное (не симметричное) Но тут можно выкрутиться за счет того, что в дырах можно хранить другие, не связанные с этим константы. =). В общем, Кури теорию бинарного дерева.

xrayman

13 Апрель 2009 в 2:46

ммм… эта… не совсем понятно…
Ну номер строки по номеру узла это понятно.
А дальше, это получается на каждый пункт будет минимум 4 задачи, реакции на события «влево»,»вправо»,»ентер» и «ескейп», причем у всех узлов, кроме листьев они будут стандартные — по смещению перейти на другой узел? Ну влево, вправо это допустим движение на одном уровне дерева между потомками узла, родительского к текущему. Ентер, ескейп — вертикальное движение по дереву. Листья — конечные функции, в которых уже можно менять какие-то параметры, а не просто шариться по дереву, с нестандартными обработчиками нажатий кнопок.
Если я правильно понял, оно больше похоже на сильноветвящееся дерево и у меня нечто отдаленно напоминающее это.

Я сейчас делаю так. Есть у меня 30 пунктов меню, я их красиво распечатал на листочке, и ручкой пронумеровал. Сделал 4 массива menu_left[30], menu_right[30], menu_enter[30], menu_esc[30], в которых хранятся номера пунктов, на которые надо переходить при соответствующем событии. Есть еще 3 массива lcdText[44][16] — двухмерный массив, в нем хранятся все строки для меню; lcdStr1[30], lcdStr2[30] — в них хранятся номера строк текста в массиве lcdText, для 1 и 2 строк экрана, для соответствующих пунктов меню. И есть переменная menu, собсно где мы сейчас.
Т.е. при нажатии на какую-то кнопку, в идеале происходит такое:
определяем событие и в зависимости от него переходим
if(menuAction==MENU_LEFT)menu=menu_left[menu];
else if(menuAction==MENU_RIGHT)menu=menu_right[menu];
…
и идет вывод на экран lcdText[lcdStr1[menu]] и lcdText[lcdStr2[menu]]

ммм, щас надо отойти, вернусь допишу :)

★ DI HALT

13 Апрель 2009 в 2:50

Нет не так.

Есть фоновая задача сканирующая клаву на предмет нажатий. Она одна и крутится постоянно. А еще есть задача выдающая на экран текущее значение указателя (в смысле строку меню).

КОгда делаешь шаг, то ты просто сдвигаешь указатель и делаешь экшн. Можно еще вычислять исходя из текущего значения свой адрес и анализировать как дальше двигаться. Еще можно запоминать в стеке (не системном, в своем) текущий путь чтобы можно было быстро вернуться обратно.

---

xrayman

13 Апрель 2009 в 14:26

ну все равно, задача выдающая на экран текущее значение указателя должна содержать кучу сравнений, если в тексте пункта содержатся динамические данные. Ну или на каждый такой пункт свой обработчик.

★ DI HALT

13 Апрель 2009 в 14:48

А зачем там куча сравнений? Динамические данные хранятся в ОЗУ и вписываются туда другим обработчиком.

---

xrayman

13 Апрель 2009 в 15:23

вопщем, сдается мне, оно все выльется в чета сопоставимое по объему. Буду ждать когда напишешь статью :)

---

arkamax

13 Апрель 2009 в 11:22

Посмотри реализацию прошивки AVR Butterfly (погугли на atmel.com). Есть менюшка, все на С, помещается в мега168 вместе с кучей всяких вкусностей.

---

xrayman

13 Апрель 2009 в 14:26

если это — http://en.wikipedia.org/wiki/AVR_Butterfly — оно, то какое-то оно стремное и на 169 меге :)

---

http://zloisop.livejournal.com/

12 Апрель 2009 в 12:28

ааа. Понятно. Мы это называли message driving engine. Но только писал я это под винду. Классно.

BOBBY

4 Май 2009 в 3:05

Классная статья, правда пришлось переписать под Algorithm Builder, но вроде работает нормально, будет время попробую поиграться с ОС на реальной железке.

---

mitiy

5 Май 2009 в 3:07

Спасибо за статью, очень интересно.
Попытался сам все попробовать и никак не выходит.
Если пишу:
Background: SetTask TS_Task1
…
Task1: SetTimerTask TS_Task1, 1000
…
RET
Запускает 1 раз и все, а если:
Background: SetTask TS_Task1
…

Task1: SetTimerTask TS_Task1, 10
…
RET
то много раз но с непонятным интервалом.
Все настройки делаю в точности как в исходниках к этой статье.
Может где-то какая-то хитрость?

★ DI HALT

5 Май 2009 в 17:47

Так сразу не скажу. Может ты что то забыл, может я что то не упомянул. Пришли мне свой проект в виде архива всей папки со всеми файлами на dihalt@dihalt.ru вечером погляжу

SATS

12 Май 2009 в 2:20

DI HALT, не знаю в каком разделе сайта написать это предложения или просьбу, но почему-то решил тут. Очень заинтересовался самопрограммированием АВРок. Может откроешь тему. Интересная ведь вешь.

★ DI HALT

12 Май 2009 в 13:29

Когда сам с ней разберусь тогда будет. Пока не вижу смысла. Полиморфный код таким образом не написать, а тратить на загрузчик память меня жаба давить :)

---

rainman

17 Сентябрь 2009 в 0:57

Начал в качестве развлечения портировать эту RTOS под Tiny13.
За час добился того, что оно стало собираться :)
Появился вот какой вопрос: что может вылезти из-за перехода на 8-битную адресацию RAM и EEPROM?
У меня такое чувство, что все должно быть норм, но в железе буду проверять позже :)

★ DI HALT

17 Сентябрь 2009 в 8:40

Ничего страшного не будет. ПРосто у тини13 очень мало ROM и очереди могут не влезть или встретиться со стеком.

★ DI HALT

17 Сентябрь 2009 в 8:41

На Тини2313 она у меня бегает со свистом, гоняя десятки цепочек. И ничего… :)

---

rainman

21 Сентябрь 2009 в 12:03

Наконец-то у меня нашелся свободный час на доделку этого примера, заливку его в железо и отладку :) Все работает, моргание диода отрабатывается со свистом :)

---

orvam

30 Сентябрь 2009 в 14:07

DI HALT можешь скинуть свой проект для тини2313, а то я все ни как не разберусь как перенести его с атмеги8.

---

aaz

10 Январь 2010 в 16:29

DI, сначала спасибо за классный сайт. :)
Есть вопрос. Пытаюсь засунуть эту няку в Tiny 2313, а там нету второго таймера. Чо-то не могу разобраться с таймерными регистрами и битами. Вот примерно чо у меня получилось:

```
RESET:
…
INIT_RTOS
OUTI TIMSK, 1<<TOIE0
…
```

В макросе INIT_RTOS получилось вот чо:

```
.MACRO INIT_RTOS

.equ MainClock = 8000000 ; CPU Clock
.equ TimerDivider = MainClock/64/1000 ; 1 mS
…
ldi OSRG, 2<<WGM00; Àâòîñáðîñ ïîñëå äîñòèæåíèÿ ðåãèñòðà ñðàâíåíèÿ
out TCCR0A, OSRG
ldi OSRG, 3<<CS00|1<<WGM02 ; Freq = CK/64
out TCCR0B,OSRG
clr OSRG
out TCNT0,OSRG
ldi OSRG,low(TimerDivider)
out OCR0A,OSRG
…
.ENDM
```

Вопрос: где я лох? :)
Просьба не пинать ногами, я в авр-ах только учусь…

★ DI HALT

11 Январь 2010 в 0:53

А в тини2313 она на другом таймере висит. Щас я тебе замылю проект на тини2313 сам увидишь как там инит таймера сделан.

aaz

11 Январь 2010 в 12:14

> А в тини2313 она на другом таймере висит.
Ну ясен пень, того-то нету. :)
Спс, ждем-с.

★ DI HALT

11 Январь 2010 в 12:20

давно уже замылил. проверяй.

aaz

11 Январь 2010 в 12:24

Гм. Нетути… :(Уведомления с сайта приходят, так что почта указана правильная.

aaz

11 Январь 2010 в 16:26

О, спасибо! Доползло наконец. :)

Alexanders

7 Сентябрь 2010 в 2:21

Интересная очень тема! А можно мне тоже замылить проект под 2313 на асме? И какого размера он получился, что в нем урезать можно, чтоб покомпактней был?
Вопрос1: размер очереди задач и размер очереди таймеров как рассчитывать и с каким запасом брать?
Вопрос2: Если заюзать дополнительные прерывания, на что обратить внимание?Какие ресурсы не трогать, какие сохранять?
Вопрос3:Я так понимаю, использование процедур внутри задач допустимо?
Заранее спасибо.

★ DI HALT

7 Сентябрь 2010 в 8:16

Диспетчер около 400 байт.

1. Таймеров — можно брать столько, сколько у нас разных задач, запускаемых по таймеру. Тогда точно хватит.
2. Очередь, ну тут думать надо, можно поопробовать циклограмму накидать и посмотреть насколько жирно получается.
3. Вполне допустимо. Никаких ограничений, главное чтобы они быстро выполнялись. Иначе все будет тупить

Alexanders

13 Сентябрь 2010 в 2:25

Попробовал-работает, интересно! Думаю для быстрого создания разовых несерьезных проектов сгодится. А вообще я полностью согласен со взглядами SWG. Если проект делается под тиражирование, то надежнее, точнее и быстрее будет сложный флаговый автомат.
А зачем в этой операционке генератор случайных чисел затесался?

★ DI HALT

13 Сентябрь 2010 в 11:49

Да просто скопирнул пример из какого то старого проекта, да вычистил от прикладной части. А его видимо забыл.

Чем надежней флаговый автомат? Да, в ряде случаев эффективней. Например когда надо точно прерывания отрабатывать по времени. Но в целом развивать и расширять (а это в тиражном проекте тоже бывает часто) гораздо проще на подобии ядра.

---

aaz

10 Январь 2010 в 16:50

Да, собственно забыл написать в чем косяк. :) Косяк в том, что не вызывается прерывание по сравнению. Похоже, что я не разобрался с тем, как инициализировать таймер, но по даташиту чо-то не могу вкурить.

Blacky

17 Март 2010 в 6:42

А задержка перед вызовом подпрограммы может быть не постоянной? Например:
SetTimerTask Task , R16

★ DI HALT

17 Март 2010 в 12:41

Может конечно, надо просто чуток макрос SetTimerTask переписать или вручную сделать вызов, без макроса.

Blacky

20 Март 2010 в 19:03

Внимание! Вся философия программирования под РТОС заключается в написании процедур, к-е выполняются за время, меньшее чем самый маленький промежуток в SetTimerTask?

★ DI HALT

21 Март 2010 в 3:20

Не совсем так. Суть в РТОС дать гарантированный ответ за время меньше определенного. Неважно какое это время (оно задается в техзадании на основании требований обьекта управления) важно чтобы оно выполнялось.

Т.е., скажем, на критические раздражители отклик должен быть не меньше чем 100мс, на менее критичные не меньше чем 1с и так далее.

Blacky

21 Март 2010 в 5:14

Т.е. можно заключить, что РТОС не пригодна для написания алгоритмов, не допускающих задержек? Например, воспроизведение полифонических мелодий.

★ DI HALT

21 Март 2010 в 5:58

Почему непригодна? Просто выстави минимальную величину задержки по твоему ТЗ и уже от нее пляши. Правда если она слшком мала, то РТОС и вправду может оказаться слишком громоздкой. Тут тогда проще сделать алгоритм решающий только твою задачу, а остальное вынести на отдельный проц. Либо взять проц помощней.

xamlo

11 Апрель 2010 в 2:49

потихоньку начинаю пробовать RTOS.
Действительно так получается очень удобно, я даже начинаю от этого тащится. а главное она действительно мало занимает, и это место не жалко.

demiurg1978

21 Апрель 2010 в 0:05

Я давно наткнулся на твой сайт. По началу заходил как гость, потом зарегистрировался. Много полезного у тебя узнал. Позаимствовал кое-какие приемы. Хороший сайт. Информация до пользователей доносится достаточно просто и доступно. Хвалю.
Программировать начал не так давно. Пишу на асме. Использую микроконтроллеры AVR. Настал у меня в программировании тяжелый момент. Потолок такой я ощутил. Программы становились сложнее, тяжелее. На сахаре.ру мне посоветовали изучить литературу по автоматному программированию и созданию операционных систем. Намучался основательно. Мозги буквально вскипели. И тут я наткнулся на твои статьи (RTOS). Это как раз и оказалось то, что мне нужно.
Внимательно прочитал материал, скачал программу (HMTR-RTOS). Скажу честно, ты молодец. Но как я тебя материл, пока разбирал на атомы твою программу!!!! Я и сам ошибаюсь, но какие ляпы у тебя попались! Пипец. Если уж пишешь для людей, тем более для новичков, то предоставляй, пожалуйста, качественную информацию.
Если уж ты говоришь, что на ассемблере нужно писать компактные и критичные по времени программы, то уж будь последователен. Я стараюсь выжать максимум из своих программ, и отслеживать ошибки.
К делу. Возьмем программу очистки SRAM. Ты ее гонял на симуляторе? Я прогнал. Последний байт памяти не записывается. Я допускаю, что тебя это устроило. Как сделал я?
Сначала я сделал так:

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

```
.EQU Data_L = r24
.EQU Data_H = r25

Initial:
outiw SPH,SPL,RAMEND

Clear_SRAM:
ldi XH,High(SRAM_START)
ldi XL,Low(SRAM_START)
ldi Data_H,High(SRAM_SIZE)
ldi Data_L,Low(SRAM_SIZE)
```

```
12
13
14
15
16
17
18
19
```

```
ldi r16,0x00
Clear_SRAM_Cycle:
st X+,r16
sbiw Data_L
brne Clear_SRAM_Cycle

Nul:
rjmp Nul
```

Потом я превратил этот кусок программы в отдельную подпрограмму. Получилось следующее:

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

```
.MACRO OUTI_SRAM
ldi XH,High(@0)
ldi XL,Low(@0)
ldi Data_H,High(@1)
ldi Data_L,Low(@1)
ldi r16,@2
rcall O_I_SRAM
.ENDMACRO

O_I_SRAM:
st X+,r16
sbiw Data_L,1
brne O_I_SRAM
ret

.MACRO Clear_SRAM
```

```
17  OUTI_SRAM SRAM_START,SRAM_SIZE-32,0 ; Минус 32 байта, для того чтобы не засрать адреса возврата. Можно и
18  меньше.
    .ENDMACRO
```

Так как мы используем подпрограмму очистки памяти в секции инициализации, то спокойно можем использовать любые регистры. А так как это отдельная подпрограмма, не нужно писать код только для очистки SRAM. То есть экономим память программ. Я использую меги, но легко подправить код для более слабых серий.
Скажу честно, по ходу разбора на атомы примера использования RTOS, я не раз назвал тебя мудаком. Ты можешь сказать, написал бы сам RTOS, раз такой умный. А я в ответ скажу еще раз. Ты для людей пишешь,тем более для новичков. Так что изволь соответствовать. Я перебрал на свой лад твой пример. Писал для мег. Найдешь ошибки, ткни носом. Даже спасибо скажу.
Когда попытался написать одну программку, я столкнулся с неприятным моментом. Моделирую ситуацию. Автомат световых эффектов. У меня STK-500. Свои программы гоняю на этой плате. Итак, кнопки я подключил к PORTA. Светодиоды к PORTC. Я не стал огород городить. 8 кнопок, 8 режимов (можно и усложнить, но я не стал, брал готовые программки). Нажимаем кнопу 0, включается первый режим (бегущий огонек вправо), кнопу 1 — второй режим (бегущий огонек влево) и т.д. При включении автомат ждет нажатия кнопы-режима. Включаю, нажимаю-работает. Нажимаю другую кнопу-режим, и тут выскочила бяка. При нажатии режим сменился, но в буфере таймерной службы осталась старая задача, которая по срабатывании таймера сработает. Уже не важно какую программу мы будем писать, эта бяка там и выскочит. Как убить старые задачи? Если тупо делать очистку очереди, убьем нужные задачи. Мож у тебя идеи есть?
Теперь, какая еще была причина тебе написать. У меня есть к тебе конретные вопросы, но не на общее обозрение. Посоветоваться хочу. Мож даже посотрудничаем. Емэйл твой я нигде не нашел. Мой: mazur1978 собака mail.ru. Черкни, если не затруднит, чтобы твой ящик отметился. С уважением, Евгений.
P.S. Хотел код на форуме выложить, да не нашел такой функции. Так что намыливайся :)

★ DI HALT

21 Апрель 2010 в 1:21

заговорщецким шепотом Хочешь открою самую страшную тайну?

На этом сайте нет готовых решений. Вообще нет! Ни одного! Готовые решения требуют уйму времени на вылизывание и доводку. Время стоит дорого и мне его никто дополнительно не оплачивает. Да и нет у меня его особо — один годный пост это от 4 до 6 часов времени просто на написание. Не веришь? Попробуй сам.

Тут идеи, концепции, методики и прочая пища для ума.

Начинающим самое то — халявщики отвалятся сами, т.к. при попытке заюзать это сырое решение хапнут багов и придется лезть и разбираться.
Те кто ковыряется сам и думает своей головой — исправят и доведут до ума.

Своего рода воспитательный момент :)

А что касается конкретно файла HMTR — это третья бета версия вторго альфа релиза этого диспетчера. Там просто прорва багов. И прикол вроде не очищающеся последней ячейки памяти это семечки не достойные даже упоминания. Там есть глюки и покруче: Например срывы стека в прерываниях, нарушения атомарности доступа, отсутствия контроля длины очереди, переполнения таймеров и еще много чего.

Программа очистки памяти у тебя бредовая. Хотя бы потому, что не очищает всю память. И зачем там делать вызов подпрограммы? Это не даст ничего, кроме две лишних команды (RCALL и RET).

Не нравится что последний байт не очищается? Ну да, не заметил. Бывает. Впрочем туда все равно запишется первый же адрес возврата — т.к. там начало стек. Но какая проблема?

```
1       RAM_Flush:      LDI     ZL,Low(SRAM_START)      ; Адрес начала ОЗУ в индекс
2                       LDI     ZH,High(SRAM_START)
3                       CLR     R16
4                                                       ; Очищаем R16
5       Flush:          ST      Z+,R16                  ; Сохраняем 0 в ячейку памяти
6                       CPI     ZH,High(RAMEND+1)       ; Достигли конца оперативки?
```

```
7                   BRNE    Flush                   ; Нет? Крутимся дальше!
8
9                   CPI     ZL,Low(RAMEND+1)        ; А младший байт достиг конца?
10                  BRNE    Flush
```

И все. Теперь очищается вся оперативка.

Как убить старые задачи? Ну очевидно написать процедуру киллер. Которая прочешет очередь по конкретному номеру задачи и вычистит все ее события. Туда же киллер таймеров. Мне пока он не требовался потому я его и не вписал в общую кучу.

З.Ы.
Сам мудак! :)

---

demiurg1978

26 Апрель 2010 в 21:51

На заметку, я сделал мигание светодиодов следующим образом.

Первый запуск так:
sbi DDRC,Led_0
Set_Timers_Queue Task_Number_0,500

Сама задача:
Blink_Led:
Task_0:
in r16,DDRC
mov r17,r16

```
com r16
andi r16,0b00000001
andi r17,0b11111110
or r16,r17
out DDRC,r16
Set_Timers_Queue Task_Number_0,100
ret
```

То есть задача потом сама себя запускает. И не нужно делать две задачи, включить светодиод, выключить светодиод.

★ DI HALT

26 Апрель 2010 в 22:22

Да разумеется, можно и конечный автоматик сделать. Это я так, для общей наглядности.

А тебе поще было ксорить.

---

demiurg1978

26 Апрель 2010 в 22:30

Хитрый китаец :)

---

dima_m

15 Июнь 2010 в 11:50

Из всех статей эти статьи для меня оказались самые сложные, но полезные. Суть уловил что к чему. Реально интересно и нужно для серьезных проектов. Для себя решил, что со временем когда молоко на губах обсохнет напишу свою ОС, тогда буду досканально ее знать и как с ней работать. Чужой труд это хорошо, но минус что не знаешь всех тонкостей проекта. Только хозяин знает. Мои проекты

постепенно усложняются и распухают. Однажды придется заняться и rtoc. Думаю что многие идеи возьму за основу отсюда. Спасибо автору сайта за труд.

Ignis

9 Ноябрь 2010 в 2:57

Сам написал аналогичную , как я называю псевдо RTOS.
использую во всех проектах. они типичные. но все равно удобно.
добавлены библиотеки работы с обменом по UART, SPI, AVRовской dataflash памятью.
в датафлэш память есть библиотека с запись в виде простейших файлов. есть библиотека сохранения и восстановления переменных из ЕЕПРОМ, указываешь только имена переменных. библиотека работы с АЦП, отвязанная от физических портов. периодичность, порядок опросов, разрядность задается для каждого порта отдельно.
индикация тоже есть. можно иметь несколько индикаторов, вывод можно задавать вспышкой определенной длинны или миганием с заданной длительностью и паузой или вывести стек сообщений в виде групп вспышек. есть возможность приема команд с любого пина в виде счетчика последовательности нажатий.
работа индикации — одна задача в стеке задач. работа со всеми входами АЦП — тоже одна задача в стеке задач.
внутренние часы.

все писано на ассемблере , с максимальным использованием имен переменных, констант, записей, макросов. по возможности оптимизировано. т.к. использовалось в проектах с особыми требованиями по времени.

Готов предоставить исходники, если интересно. Может кто захочет оформить в пакет, удобный для применения. Деньги я зарабатываю совершенно в другой сфере.

★ DI HALT

9 Ноябрь 2010 в 10:57

Пришли на clihlt@yandex.ru архивом. Было бы интересно поковырять. Спасибо!

Seaman

16 Ноябрь 2010 в 0:42

Интересно было бы взглянуть. Не сложно и мне выслать на b-a-z-a-a-r@mail.ru ? Заране благодарен.
А DI HALT'у огромное спасибо за проект! Разобравшись, понимаешь, что совершил шаг вперёд в изучении AVRов, и осознаёшь, какой труд проделал автор. Сенсей :)

Tabke

24 Февраль 2011 в 0:34

А можно и мне на thabke@mail.ru скинуть, Ignis наверное не следит за сообщениями, скинте мне кто-нибудь у кому он уже давал.

Ignis

25 Февраль 2011 в 1:52

Почему не читаю ? Читаю.

Код выслал.

и немного добавлю в развитие.

Параллельно с применением привожу код в порядок. Озаботился тем, что жалко программную память тратить под функции , которые не используешь в конкретном проекте. А лазит в код библиотеки — сугубо не правильно.

Сделал на объявлениях. Что бы при сборке, подцепляла только нужные библиотеки.
Прием завязан на выполнение правил по синтаксису объявлений. Работает.

Пример:
http://easyelectronics.ru/repository.php?act=view&id=44

P.S.
только код на asm все больше начинает походить на Си.

P.P.S
Приношу извинения хозяину темы, если влезаю в чужой монастырь.

★ DI HALT

25 Февраль 2011 в 18:15

Да ради бога, чо :)

Delfer

2 Декабрь 2010 в 1:30

Посмотрел на ОС, заинтересовало) Захотелось найти ОС, примерно с такими возможностями:
-консоль на UART
-возможность остановки/запуска процессов и изменения их приоритетов через эту консоль
-вывод процессом информации в консоль и ввод из нее
-библиотеки для работы с периферией (например представление EEPROM как файловой системы, как это у FemotoOS)
Пгуглил, нашел Contici, FreeRTOS, FemotoOS, SafeRTOS, но не увидел в ниих того, что хотелось. Не подскажите на что смотреть?

★ DI HALT

2 Декабрь 2010 в 1:36

Не знаю. Я сам с ними не работал. Но скорей всего и не найдешь таких. А работу с процессами через консоль придется допиливать самому.

---

triggery

7 Март 2011 в 22:46

DI HALT,огромное тебе спасибо тебе за твои труды! именно что выкладываешь их, поясняешь!!! очень заинтересовала статья о ОС. да разобраться наверно не судьба — не изучать же асм для этого… есть ли какие-то идеи на счет подобного на C ? ссылки какие-то… охота двигать все время вперед, а изучение асм-а — это топтание на одном месте, пока не получится, если еще и получится…

★ DI HALT

7 Март 2011 в 22:50

Дальше по курсу есть цикл «Архитектура программ» там этот же диспетчер расписан на Си.

А асм изучать все равно полезно, для понимания глубинного смысла того как работает программа.

---

elk

3 Май 2011 в 20:29

По поводу изучения асма — есть способ, который кажется мне достойным упоминания.

Берется простая программа на с, компилируется без оптимизации и генерируется смешанный с/ассемблерный листинг. Далее это дело курится со справочником по ассемблеру. Момент в том, что когда ты будешь курить ассемблерный листинг, ты уже точно будешь знать _что именно_ делает этот фрагмент когда и будешь видеть как оно отображается на ассемблерные инструкции. Таким образом вырабатывается навык чтения ассемблерного кода за вечер-два.

Важно скомпилировать с программу без оптимизиации, потому что оптимизатор может такого накрутить, что и за неделю не вкуришь.

---

triggery

7 Март 2011 в 22:47

это я не в упрек асм-у, а в упрек себе…

---

marsden

28 Май 2011 в 10:45

DI, спасибо большое за такую вещь. Переделал ее на мегу 16, навесил пару индикаторов 4-х символьных на сдвиговых регистрах, еепромку, часики 1307, обвешал своими процедурками и стало мне счастье. Но длилось оно недолго, ибо столкнулся с непоняткой — каким-то образом таймерные задачи, которые сами себя возобновляют через некоторое время перестают самовозобновляться. Например, каждую секунду считываю в ОЗУ данные с часиков, потом по запросу через УАРТ они уходят в комп. В результате получаю такой глюк — комп считывает часы и видит, что они стоят. Я когда такое увидал — впал в ступор, потому что остановить часы — это же надо битик в ДС1307 переставить, а это целая куча команд через и2ц… Только через несколько дней догнал, что сорвалась задача чтения часов, когда увидал, что на отладочной плате произошло то же самое, но!!! уже с другой задачей! Прошивка одна, платы почти один в один, почему у одной срывает одну задачу, а у другой — другую? Мозг уже плавится и кипит, ничё понять не может.
Самое интересное — сделал счетчик, который сбрасывается при вызове процедуры и инкрементируется в idle, там же проверка — если слишком много наинкрементился (пришлось двухбайтный делать) — переустанавливаем задачу в таймер опять. Казалось бы, все путем, но, опять это но… таймер отрабатывает один раз и опять срывается… Помогает только полный резет девайса. А для моей задачки это не есть гуд. Где покопаться, что просмотреть?

---

marsden

28 Май 2011 в 10:52

вернее, после перезапуска задачи она отрабатывает не один раз, а несколько, поскольку на индикаторе сегменты успевают проморгаться, но потом опять встает.

★ DI HALT

28 Май 2011 в 11:09

Ну, во первых, срывать может изза нарушения атомарности. Это главная беда сего диспетчера. И вроде бы у меня там была такая бага, не помню исправил ли я ее до заливки или после.

Суть в чем — процесс пихает в диспетчер задачу, уже начал сее действо. Т.е. взял адрес ячейки, уже готов вписать туда данные. Тут опа прерывание и в прерывании вписывается туда же. А у процесса то адрес взят и когда с прерывания на него вернется он

вобьет свою задачу на то же место. А та, что в прерывании установилась пропадет. Т.е. при добавлении в очередь не в прерываниях надо их запрещать на период Set**Task это первое.

Второе — у тебя может банально не хватать длины очереди таймеров.

---

marsden

28 Май 2011 в 11:16

Длины очереди хватает, я ее увеличил и туда не так много задач пихается, а вот с запретом прерываний может быть. Щас повтыкаю и буду гонять до посинения :)
Хотя вот при отладке один раз косяк очень интересный вылез — при постановке в таймерную очередь при выполнении
subi ZL, Low(-3) ; Выбираем следующий
sbci ZH, High(-3) ; Z+=2
вылезло такое — адрес очереди находится в ОЗУ по адресу 0x00AB, после прохода этого кода в ZH вдруг образовалась единица и следующий адрес стал 0x01AE. Сильно подозреваю, что это глюк жтага и студии.

---

marsden

28 Май 2011 в 11:50

Ура! Получилось! теперь процедура SetTimer выглядит так

```
;————————————————————————————————————
; OSRG — Timer Event
; X — Counter
SetTimer:
push ZL
push ZH
push Tmp2
push Counter
```

```
ldi ZL,low(TimersPool) ; Берем адрес очереди таймеров
ldi ZH,high(TimersPool)

ldi Counter, TimersPoolSize ; Берем число таймеров

STL01: ld Tmp2, Z ; Хватаем первый заголовок
cp Tmp2, OSRG ; Сравниваем с тем который хотим записать
breq STL02 ; Если такой уже есть, идем на апдейт

subi ZL, Low(-3) ; Выбираем следующий
sbci ZH, High(-3) ; Z+=2

dec Counter ; Уменьшаем счетчик
breq STL03 ; Если ноль переход к записи нового таймера
rjmp STL01

STL02:
IN TMP2,SREG
cli ; Если нашли такой же, то делаем ему апдейт
std Z+1, XL ; Critical Sectio Значения временем из X
std Z+2, XH ; Update Counter Оба байта
OUT SREG,TMP2 ; leave Critical Section
rjmp STL06 ; Exit
STL03: ; Если аналогичного не нашли

ldi ZL, low(TimersPool) ; То делаем добавление нового
ldi ZH, high(TimersPool) ; Заново берем адрес очереди

ldi Counter, TimersPoolSize ; И ее длинну
```

```
STL04: ld Tmp2, Z ; Хватаем первый заголовок
cpi Tmp2, $FF ; Пуст?
breq STL05 ; Переходим к записи таймера

subi ZL, Low(-3) ; Если не пуст выбираем следующий таймер
sbci ZH, High(-3) ; Z+=2

dec Counter ; Очередь кончилась?;
breq STL06 ; Да. Нет таймеров свободных. Увы. Выход
; Краша не будет, но задача не выполнится
rjmp STL04 ; Если очередь не вся, то повторяем итерацию

STL05:
IN Tmp2,SREG
cli
st Z, OSRG ; Запрет прерываний перед записью в очередь
std Z+1, XL ; Сохраняем новый таймер
std Z+2, XH ; И его время
out SREG,Tmp2 ; Разрешаем прерывания (возможно)

STL06:
pop Counter ; Выходим, достав все из стека.
pop Tmp2
pop ZH
pop ZL
ret
```

Использовал не просто пару cli — sei, а запоминал состояние SREG и потом его восстанавливал. Теперь вроде как задачу не срывает, по крайней мере пока у меня этого не получилось :) Спасибо

marsden

28 Май 2011 в 11:53

блииииннн… закон подлости…. только запости коммент и опять сорвало :((((((

★ DI HALT

28 Май 2011 в 12:16

значит где то что то забыл. Внимтельней к прерываниям. Может выходя из прерывания что то не достал или еще что забыл. Не забывай что есть еще SetTimerTask где бы тоже не помешало заатомарить обращение к ресурсам.

★ DI HALT

28 Май 2011 в 12:18

Запрещать прерывания над ДО анализа очереди на предмет свободного места. Иначе смысла нету. У тебя адрес берется сразу же и начинается поиск места. А прерывания ты запрещаешь непосредственно перед самой записью. Что ничего не меняет. Т.к. адрес то уже взят!

marsden

28 Май 2011 в 13:11

спасибо! вроде получилось, по крайней мере раньше секунд за 30 вгонял в ступор, теперь минут 10 гонял — работает :) В SetTimerTask я добавил сохранение регистров в стек. Измененный код SendTask и SetTimer привести?

wofs

14 Ноябрь 2011 в 0:06

если не сложно :)

Dan_ex

10 Июнь 2011 в 19:24

Добрый день всем!
начал изучать ассемблер (знаю C++), с нуля, с помощью данного ресурса, материал изложен оч. понятно и доходчиво.
Практическими навыками похвастаться пока к сожалению не могу, т.к. знаний пока маловато, и в связи с тем ряд вопросов про ОС, со смыслом работы я вродеб разобрался, немного не понимаю как идёт инициалицация переферии в т.ч. АЦП, так-же не могу разобраться с таймерами тоже как работают знаю только, что там есть 3 режима работы(Пробовал понимать, безуспешно пока что); прочитал статью в Хакере про МикроОС, там как-то понятнее изложить получилось.
в статью можно было-бы добавить откуда берётся Treshold; .equ Treshold = 100
когда делал по этому пример свою програмку, возник этот вопрос.
незная как инициализируется АЦП, для меня это было загадкой, и студия выдавала ошибку.
плюс опечатка в ADC_OK, там название SetTask TS_RedOn; надо SetTask TS_OnRed
и вопросы:
1) что будет если количество задач в очереди превысит максимально возможное?
2) То же и про количество таймеров.

★ DI HALT

10 Июнь 2011 в 19:35

АЦП к диспетчеру и ОС отношения не имеет. Инициализируется отдельно. Про АЦП есть своя статья, там подробно все рассказано. Равно как и про таймеры.

Инит периферии идет в самом начале, перед входом в главный цикл диспетчера. Впрочем, можно инициализировать и потом, по мере необходимости.

1) Задачи буду пропадать. Что может вызвать срыв работы логики программы
2) таймеры будут пропадать, результат тот же.

Dan_ex

10 Июнь 2011 в 20:31

Про последовательность переферии «ИНИТ» я в курсе, и про то в какой последовательности это можно делать.Не понятно как это делать(нужно это ещё понять). Там именно опечатка была просто, из-за этого у меня программа и не работала.
Я как вообще делаю: Создаю новый проект, смотрю на статью и печатаю от туда, и вставляю подробнейшие коментарии, а дальше всё прогоняю F11 попутно соображая что куда, затем в PROTEUS, после того как всё более или менее работает зашиваю в МК. так по кр. мере на СИ делал.
(К сожалению изучение МК начал с платы ARDUINO, а потом затянуло и всё свободное время убиваю на это.)
PS: статью про АЦП нашел, спасибо за быстрый ответ.

Dan_ex

18 Июнь 2011 в 16:19

Может кто помочь портровать данную ОС на ATmega328P, я вродебы на 50 процентов её допилил но вот с таймерами (таймерной службой) никак разобраться мне не дано. (:

---

alkinoy

10 Август 2011 в 15:15

В kernel.asm упущено из вида вот такое предупреждение из даташита:

Caution: An interrupt between step 5 and step 6 will make the write cycle fail, since the EEPROM Master Write Enable will time-out. If an interrupt routine accessing the EEPROM is interrupting another EEPROM access, the EEAR or EEDR Register will be modified, causing the interrupted EEPROM access to fail. It is recommended to have the global interrupt flag cleared during the four last steps to avoid these problems.

:)

welcom

17 Январь 2012 в 15:18

Все доступно и ясно, НО только для тех, кто работает или начинает работать с ассемблером. Было бы очень хорошо, если бы все статьи, касающиеся RTOS, были с поддержкой языка С. Спасибо.

★ DI HALT

17 Январь 2012 в 15:36

Есть ,но дальше в курсе. Ищи цикл про архитектуру программ. Там и про диспетчеры и про RTOS и про флаговые автоматы всякие. Примеры там на Си и псевдокоде си подобном.

Aks

25 Апрель 2012 в 15:34

marsden- 28 Май 2011:
Измененный код SendTask и SetTimer привести?
Пожалуйста и мне.

pz6tnk

8 Ноябрь 2012 в 12:56

Хэй, друзья-приятели, подскажете с моей проблемой, пожалуйста, я думаю известная беда: в проекте под Tiny26L вся эта система RTOS прекрасно работает в симуляторе, а в железе сразу же появляется затык, в зависимости от занимаемого объема флэш памяти. То есть работает работает себе, но если я добавлю в код «nop», в любую задачу, даже в неиспользуемую процедуру или в прерывание куда-нибудь, которое никогда не вызывается, все затыкается. Добавляю еще 3 nop и все опять работает. Я думал, что с прерываниями беда, но «nop» же стоит вообще не пойми где и не используется никогда. Хэу! Что за трабл с памятью?

★ DI HALT

8 Ноябрь 2012 в 13:02

Где то нарушен атомарный доступ. Либо срывает стек. Закрой в cli-sei критичные части. Особенно там где идет обращение к таймерам, цепочке задач и всем переменным которые доступны из прерывания и фона.

Biggy

2 Август 2013 в 15:08

А такой вопрос по комментариям стопицотлетней давности (см выше начиная со 2).
Ди, у тебя есть отличная статья про такое меню, но она на Сях, а Си это темный лес с зыбучими болотами на земле и гарпиями на ветках.
Нет ли что-либо подобного, но под ACM? (aka разветвеленное древовидное меню на лсд какой-нить WH)

★ DI HALT

2 Август 2013 в 15:11

Где то в проектах было. Но я сейчас уже не найду. А что тебе си то? Ты смотри алгоритм, как оно реализовано и запили по своим нуждам. Сишный код легко в асм перекладывается, чай не ООП какой нибудь.

Biggy

2 Август 2013 в 15:57

Да пару дней посидел, моск вскипел на неделю, не идут как-то дела с Си. Легко когда что-нибудь небольшое по кусочкам, а там не малые такие библиотечные функции. Ну вообщем если попадется глазу скинь плз на мыло xt_05СОБАКАmail.ru, выручишь очень.

anton82

10 Ноябрь 2013 в 2:33

А имеет право такая штука называться хотя бы псевдо RTOS?
1. Системный таймер с периодом 1мс.
2. По достижении программного таймера какой-либо задачи (или нескольких задач) нуля, системный таймер ставит флаг диспетчеру о необходимости запуска задачи. И выставляет соответствующие флаги в векторе запросов задач.
3. Диспетчер задач сбрасывает флаг необходимости его же запуска, смотрит вектор запросов задач в цикле for, находит ближайший флаг и запускает задачу. Задачи стоящие дальше имеют шанс запуститься в следующем цикле.
4. Задача запускается, сбрасывает свой флаг запроса и, по мере необходимости, подымает флаг своего таймера и устанавливает его значение.
На практике все работает.
Ограничения:
1. Задачу нужно уместить в 1мс-Т(декремент счетчиков)-Т(диспетчера). А поскольку на асме лениво пока писать, компилятор CCS; а МК — PIC16f819 с ТЧ встроенного генератора 4 МГц (ноги все заняты, соответственно выполнение одной инструкции занимает 1мкс), время для полезного кода составляет около 750 мкс. В приципе 750 инструкция это немало. Но вот работа с 1-wire почти тютелька в тютельку влезла. Зато кварц встроенный, и лапы, выделенные под него, занимаются полезным делом.
2. Если задаче с первым порядковым номером поставить программный таймер 1мс, то остальные по порядку не допросятся своего кванта. Но мы ведь этого делать не будем. Да и 2 мс не надо. Лучше, чтоб минимальный интервал был равен или больше числа задач с таймером.
3. Наверняка еще какие-то есть
На практике все работает. Крутит 4 семисегмента, 3 кнопки на одном входе АЦП, шину 1-wire(датчик температуры DS18B20), ну и всякие промежуточные вычисления и дрыганья ногами. Общая задача конечно фуфловая, но с даже такой кривой реализацией RTOS я уже не парился над общим
алгоритмом, а занимался программированием функционала.
Есть конечно желание сделать «нормальное подобие» RTOS на ASM, но времени пока нет. А раскуривать придется, чувствую, немало.

---

anton82

10 Ноябрь 2013 в 2:49

И еще, из 2 кило памяти, таймерно-диспетчерская служба заняла около 400 байт. То бишь почти 20 процентов. А вся программа — чуть больше кило. На асме, думаю, существенно ужалось бы и по размеру и по времени. Оптимизировать не стал,так как в данном случае это

просто не важно. Все работает, временные интервалы соблюдается,все везде влезает.

Celeron

16 Февраль 2014 в 0:07

Смотрите также статью: **«AVRASM: Пример использования «Диспетчера задач RTOS 2.0″ (установка и настройка)»**

Продолжение проекта: «AVRASM: Диспетчер задач RTOS 2.0 (псевдо кооперативная ОС)»…

Arcanum

6 Октябрь 2015 в 1:36

—>Как видишь, она сама себя запускает каждые 1000мс, т.е. каждую секунду. А стартует там же, из секции Background<—
вот есть ззачада как эта — сама себя стартует раз в n единиц времени.
или светодиод моргает раз в пол секунды.
вопрос: а как из других задач прекратить выполнение этих задач? убрать их из диспетчера зада и таймера?

★ DI HALT

6 Октябрь 2015 в 7:56

Механизма принудительного сброса задачи извне тут нет. Но да, можно взять и выкинуть ее из диспетчера и таймера.

Arcanum

6 Октябрь 2015 в 10:21

то есть можно глянуть в какой байт озу прописывается и написать процідурку по очистке конкретного этого байта(указателя).
как settask только наоборот. надо глянуть макрос

★ DI HALT

6 Октябрь 2015 в 13:42

У тебя же есть идентификатор задачи. Просто поиском проходишь по очередям и находишь его. А там заменяешь на переход в IDLE и все.