« Урок 19. Формирование сигнала, при помощи ЦАП (R2R).
Урок 18.2. Вывод изображения на графический дисплей WG12864 »

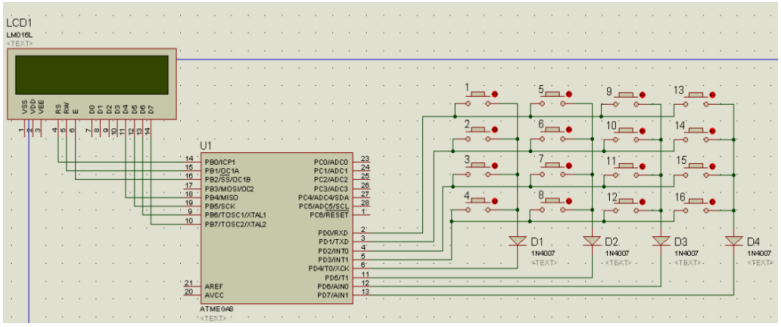
Урок 20. Опрос матричной клавиатуры.

admin | 24.08.2013 | Статьи\ AVR Учебный курс | Комментарии (40)

 В очередном уроке, я поведаю читателю о том, как пишутся статьи на данном сайте, чтобы можно было не только скопировать готовый код, но и проследить за ходом мыслей. Поэтому урок будет в другом формате.

Первым делом выбирается тема изучения. Эта статья — про матричную клавиатуру. Далее нужно разобраться, что это такое, основной источник — википедия. Если там нет информации — гугл. После прочтения 3-5 ссылок, главное понять что оно из себя представляет и зачем вообще нужно. Ответ на первый вопрос должен быть примерно таким — это набор кнопок, включенных и опрашиваемых особым образом. Так почему не использовать обычное включение кнопок? Это ответ на второй вопрос — подобное включение, при использовании 16 и более кнопок, позволяет сэкономить ножки микроконтроллера. Все очень просто, на первом этапе этой информации более чем достаточно.

Теперь следующий шаг, изучение схемы. Во время изучения схемы микроконтроллер представляем черным ящиком, который может либо дрыгать ножками, либо читать информацию с ножек. Смотрим на особенности схемы, на одну ножку приходится 4 кнопки по горизонтали (строки), аналогично 4 кнопки приходится на одну ножку по вертикали (столбцы). Кнопки в пределах одной строки/столбца объединены общим проводом. Получается матрица 4x4.



Теперь нужно понять принцип работы более детально. Ножки, подключенные к общему проводу строк, настраиваются как вход, те что подключены к столбцам настроены как выход. Когда активен только первый столбец, то мы однозначно знаем, что нажаты могут быть только кнопки с 1 по 4. Далее, переключаемся на второй столбец, сканируем кнопки с 5 по 8 и т.д. Остается лишь читать состояния входов. Диоды нужны, чтобы защитить входы микроконтроллера, если нажато несколько кнопок одновременно. Этой информации должно быть достаточно для того, чтобы написать собственную прошивку. Написать свою прошивку это интересно и главное, ты знаешь как она работает. Поэтому я стараюсь все прошивки писать с нуля. Перейдем к программной части.

Начнем с того, что опрос кнопок, сам по себе, не является целью. Возможно в основном цикле будет программа, использующая клавиатуру, поэтому логично производить опрос независимо от основного цикла, через равные промежутки времени. Поможет нам в этом прерывание таймера по совпадению. Его суть в том, что когда таймер сделает заданное количество тиков основная программа встанет на «паузу» и выполнится код записанный отдельно от основного цикла. Этот код и будет нашим опросом.

Можно использовать Таймер1, но он имеет множество полезных функций и может пригодиться для других вещей, поэтому задействуем таймер2. С какой периодичностью производить опрос? С такой, чтобы не пропустить самые быстрые нажатия. Для надежности, решил взять 20 раз в секунду, т.е. прерывание должно происходить раз в 50мс. Таймер2 не очень годится для длинных отсчетов, потому что максимальное число, которое можно записать в регистр сравнения 0xFF = 255. При минимальной частоте таймера 7813Гц, один тик будет длиться 1/7813=0,000128сек, т.е. максимальный промежуток между прерываниями 0,000128*255=32,6мс. В данном случае не принципиально 50 или 32, поэтому этот результат меня устроил.

За одно прерывание нужно опросить 4 столбца, для этого сделаем цикл for, переменная i хранит номер опрашиваемого столбца.

```
int i=0;
// Timer 2 output compare interrupt service routine
interrupt [TIM2_COMP] void timer2_comp_isr(void)
{
    for(i=0; i<4; i++)
    {

    }
}
```

Прерывание работает, цикл крутится, нужно добавить переключения столбцов. Для этого, проще всего, создать массив, в котором будут заранее сконфигурированы состояния PORTD, эти состояния будут по очереди записываться в порт.

```
int i=0;
char portState[4]= {0xEF,0xDF,0xBF,0x7F};

// Timer 2 output compare interrupt service routine
interrupt [TIM2_COMP] void timer2_comp_isr(void)
{
    for(i=0; i<4; i++)
    {
       PORTD=portState[i];
    }
}
```

С переключениями порта разобрались, теперь нужно в течение каждого переключения, проверить каждую строку — не замкнута ли кнопка, т.е. произвести опрос PIND.0-PIND.3. Для этого сделаем в одном цикле еще один цикл, используем for, переменная j хранит номер строки.

```
int i=0,j=0;
char portState[4]= {0xEF,0xDF,0xBF,0x7F};

interrupt [TIM2_COMP] void timer2_comp_isr(void)
{
    for(i=0; i<4; i++)
    {
       PORTD=portState[i];
       for(j=0; j<4; j++)
       {
       }
    }
}
```

Теперь нужно добавить сам опрос входа, но если писать if(PIND.0==0), то получится много условий, код будет не компактным, поэтому, вспоминаем про «логическое и»: если число 1 и число 2 равны единице, то результат будет равен единице, во всех остальных случаях результат 0. Добавляем массив, в котором записаны номера пинов от 0 до 3. Выделяем из всего PIND, при помощи «логического и», нужный пин и проверяем равен ли он нулю.

```
int i=0,j=0;
char portState[4]= {0xEF,0xDF,0xBF,0x7F};
char inputState[4]={0x01,0x02,0x04,0x08};

interrupt [TIM2_COMP] void timer2_comp_isr(void)
{
    for(i=0; i<4; i++)
    {
       PORTD=portState[i];
       for(j=0; j<4; j++)
       {
          if(((PIND&inputState[j])==0))
          {

          }
       }
    }
}
```

Неплохо еще вывести на экран какой нибудь символ. Для этого можно создать двухмерный массив, выбор символа, будет зависеть от номера столбца и строки. Символы до 9 соответствуют номерам кнопок, на остальные задействованы буквы. Символ выводится в момент сканирования строки, что очень удобно, так как мы знаем номер столбца и строки.

```
int i=0,j=0;
char portState[4]= {0xEF,0xDF,0xBF,0x7F};
char inputState[4]={0x01,0x02,0x04,0x08};
char symbol[4][4]={{'1','2','3','4'},
                   {'5','6','7','8'},
                   {'9','A','B','C'},
                   {'D','E','F','D'}};

// Timer 2 output compare interrupt service routine
interrupt [TIM2_COMP] void timer2_comp_isr(void)
```

```
{
    for(i=0; i<4; i++)
    {
       PORTD=portState[i];
       for(j=0; j<4; j++)
       {
          if(((PIND&inputState[j])==0))
          {
             lcd_putchar(symbol[i][j]);
          }
       }
    }
}
```

Все супер, все работает, но… когда нажимаешь на кнопку один раз выводится сразу несколько символов. Поэтому введем проверку. После того, как кнопка была нажата, программа крутится в пустом бесконечном цикле, до тех пор, пока кнопка не будет отжата. Это позволит выводить символы по одному. В итоговую программу добавлена кнопка для очистки экрана.

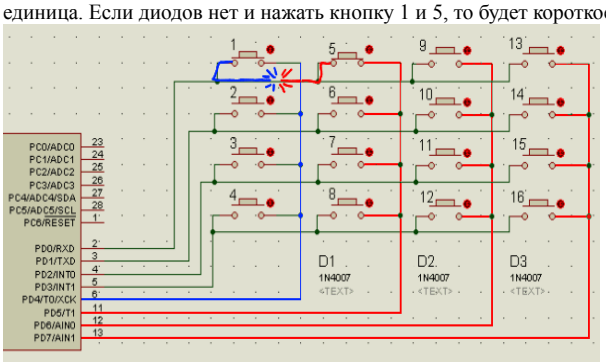
```
#include <mega8.h>
// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x18 ;PORTB
#endasm
#include <lcd.h>

int i=0,j=0;
char portState[4]= {0xEF,0xDF,0xBF,0x7F};
char inputState[4]={0x01,0x02,0x04,0x08};
char mass2[4][4]={{'1','2','3','4'},
                  {'5','6','7','8'},
                  {'9','A','B','C'},
                  {'D','E','F','D'}};

// Timer 2 output compare interrupt service routine
interrupt [TIM2_COMP] void timer2_comp_isr(void)
{
    for(i=0; i<4; i++)
    {
       PORTD=portState[i];
       for(j=0; j<4; j++)
       {
          if(((PIND&inputState[j])==0))
          {
             while((PIND&inputState[j])!=inputState[j]){};
             lcd_putchar(mass2[i][j]);
          }
       }
    }
}

void main(void)
{

// Port C initialization
// Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State6=T State5=T State4=T State3=T State2=T State1=P State0=P
PORTC=0x03;
DDRC=0x00;

// Port D initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=In Func2=In Func1=In Func0=In
// State7=1 State6=1 State5=1 State4=1 State3=P State2=P State1=P State0=P
PORTD=0xFF;
DDRD=0xF0;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: 7,813 kHz
// Mode: CTC top=OCR2
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x0F;
TCNT2=0x00;
OCR2=0xC3;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x80;

lcd_init(8);

// Global enable interrupts
#asm("sei")
while (1)
      {
          if(PINC.0==0)
          {
          lcd_clear();
          }
      };
}
```

Примерно в такой последовательности я рассуждал, когда писал данную прошивку. Дальше должна следовать еще отладка на железе. Но в пределах данной статьи ограничусь симулятором.

Матричная клавиатура 4x4

Все что я хотел сказать данной статьей, что собирать из маленьких готовых кирпичиков гораздо проще, чем пытаться сразу написать все целиком. Разбейте большую задачу на множество маленьких задач и постепенно их решайте. Главное, разобраться с принципом работы устройства и заставить прошивку работать, оптимизировать ее можно после. Прошивка и схема доступны тут.

Update: Зачем нужны диоды? Допустим некий момент времени, когда опрашиваем первый столбец, соответственно на остальных столбцах логическая единица. Если диодов нет и нажать кнопку 1 и 5, то будет короткое замыкание.



40 комментариев: Урок 20. Опрос матричной клавиатуры.

- Dk on 09.11.2014 в 22:51

  Делаю нечто подобное, даже проще, но только с двумя портами(один на вход другой на выход) и вот что-то ничего не выходит — диод не горит. Вот код

  ```
  #include

  void main(void)
  {
  DDRA=0xFF;
  DDRB=0x00;

  PORTB=0xFF;
  PORTA=0xFF;

  DDRD=0xFF;

  while (1)
  {
  if (PINB.0==0)
  {
  PORTD.0=1;
  }
  else
  {
  PORTD.0=0;
  }

  }
  }
  ```

- Dk on 09.11.2014 в 23:07

  суть такая, я хочу чтобы при нажатии кнопки которая соединяет PINA.0 и PINB.0 загорался диод который стоит на PIND.0

- 

admin on 09.11.2014 в 23:27

По вашему коду на PINB.0 будет всегда + питания

- Денис on 29.11.2014 в 16:20

А можно ли как-нибудь записывать введенные с клавиатуры данные в переменную? Ну например, чтобы потом исходя из введенных данных запустить ШИМ с определенным заполнением?

Суть в том, как я понял, 1 опрос (1 прокрутка прерывания) это 1 символ. Т.е. каждый последующий опрос затрет инфу с предыдущего.

В общем вопрос как записывать данные с клавы в переменную. При том, что нужны числа от 0,0 до 10,0 и с одним знаком после запятой.
PS^ С ШИМом проблем нету, разобрался.

- admin on 29.11.2014 в 17:24

нажимаем кнопку — выводится символ, сдвигается позиция курсора, в следующее нажатие символ 2 + запятая, затем дробная часть, попробуйте, не проверял, но начал бы с такого

```
float result, result_3;
char result_1, result_2;

if(((PIND&inputState[j])==0))
{
    while((PIND&inputState[j])!=inputState[j]){};
    lcd_gotoxy(pos, 0);
    lcd_putchar(mass2[i][j]);
    if(pos == 0) //в первый проход записываем десятки
    {result_1=mass2[i][j]-0x30; pos++;}
    if(pos == 1)  //в следующий раз единицы
    {result_2=mass2[i][j]-0x30;  pos++;}
    if(pos == 2)  //теперь дробную часть
    {
       result_3=mass2[i][j]-0x30;
       result = (float)result_1*10+(float)result_2+(float)+(float)(result_3/10);
       pos=0;
    }
}
```

- Денис on 29.11.2014 в 17:39

Попробую, спасибо.

- Роман on 22.02.2015 в 16:50

пожалуста объясни зачем диоды! не совсем ясно что они дают в схеме если нажато сразу несколько кнопок

- admin on 22.02.2015 в 17:48

Добавил в конце поясняющую картинку

- Иван on 27.03.2015 в 10:24

Делаю вывод на четырехзначный семисегмент вывод цифр соответствующих нажатым кнопкам клавиатуры, но программа не хочет работать. Я несколько модифицировал ваш обработчик прерывания для 2 таймера:
interrupt [TIM2_COMP] void timer2_comp_isr(void)
{
PORTC=0x00;

for(i=0; i<4; i++)
{ //записываю состояния порта
PORTD=portState[i];
for(j=0; j<4; j++)
{
if(((PIND&inputState[j])==0))
{
while((PIND&inputState[j])!=inputState[j]){};
if(pos == 0) //в первый проход записываем десятки
{

```
n=mass2[i][j]-0x30; pos++;
PORTC.0 = 1;
}
if(pos == 1) //в следующий раз единицы
{
n=mass2[i][j]-0x30; pos++;
PORTC.1 = 1;
}
if(pos == 2) //теперь дробную часть
{
n=mass2[i][j]-0x30; pos++;
PORTC.2 = 1 ;
}
if(pos == 3) //теперь дробную часть
{
n=mass2[i][j]-0x30; pos++;
PORTC.3 = 1;
}

switch(n)
{
case 0:{PORTB=0b10001000;break;}
case 1:{PORTB=0b11101011;break;}
case 2:{PORTB=0b01001100;break;}
case 3:{PORTB=0b01001001;break;}
case 4:{PORTB=0b00101011;break;}
case 5:{PORTB=0b00011001;break;}
case 6:{PORTB=0b00011000;break;}
case 7:{PORTB=0b11001011;break;}
case 8:{PORTB=0b00001000;break;}
case 9:{PORTB=0b00001001;break;}
}
}
}
}

}
```

PORTB — на семисегмент, PORTD — на клавиатуру
Правильна ли логика разработчика прерывания?

- 
admin on 27.03.2015 в 11:36

  С ходу не заметил ничего необычного, но практика показывает, что лучше опрос клавиатуры вытащить в основной цикл (в Вашем случае это 100% нужно сделать), а индикацию оставить внутри прерывания.

- 
Иван on 27.03.2015 в 20:45

  Большое спасибо.

- 
vv4 on 28.03.2015 в 03:26

  А если у меня матрица 5x5, и я подключаю не уместившиеся столбцы/колонки к PORTC.0 и PINC.4, то придется делать проверку после того, как были опрошены ножки от PD, или это все можно как-то впихнуть в обычную проверку?

- 
admin on 28.03.2015 в 17:32

  микроконтроллер все действия выполняет последовательно, добавьте в существующую проверку еще одно условие, нечто вроде такого
  if(((PIND&inputStateD[j])==0) && (PINC&inputStateC[j]))

- 
vv4 on 29.03.2015 в 00:47

  Спасибо. Кстати, гораздо проще использовать для матричной клавиатуры два порта: один, для строк, как выход, а другой, для столбцов, как вход. Лично мне так удобнее.

- 
Den on 07.02.2016 в 22:26

  Как сделать опрос матричной клавиатуры 3x4 ?

- 
admin on 08.02.2016 в 21:54

точно также, принцип не меняется

- 
Иван on 12.03.2016 в 20:22

Технический момент, как цеплять конденсаторы на матричную клавиатуру? Хочется одновременно и от дребезга избавиться и код прошивки подсократить

- 
admin on 13.03.2016 в 01:54

логично предположить, что на каждую кнопку нужно 4 кондера, поэтому идея плохая, от дребезга итак спасает то, что опрос происходит через промежутки времени, если добавить примитивную проверку старого значения с новым, то этого будет более чем достаточно.

- 
Дмитрий on 08.04.2016 в 20:29

Как изменить программу, если некоторые функции конструируемого устройства требуют обработки длительного нажатия на клавишу? Например: светодиод горит, пока кнопка нажата.

- 
admin on 08.04.2016 в 23:44

http://avr-start.ru/?p=4500

- 
qw123 on 10.04.2016 в 01:47

А всё таки диоды не обязательны. Можно оптимизировать прошивку. А именно если опрашиваем первый столбец, то зачем на остальных логическая 1? Можно на все столбцы подать логический 0.

- 
Дмитрий on 20.04.2016 в 22:02

Как при таком алгоритме осуществить проверку, что ни одна клавиша не нажата?

- 
admin on 20.04.2016 в 22:14

пройти все столбцы, если никуда не зашли то значит все отжаты

- 
Михаил on 27.06.2016 в 16:06

Все хорошо. но делать цикл, выполнение которого не детерминировано в обработчике прерывания — верх безумства. Не проще ли сделать обработку каждый тик таймера с запоминанием состояния, тогда и прерывание будет обрабатываться за предсказуемое время и состояние всегда можно отследить, а кроме прочего можно и сделать обработку «длительного» удержания кнопки, а особых целей

- 
admin on 27.06.2016 в 21:21

Все верно пишите.

- 
Михаил on 28.06.2016 в 19:26

если кому интересно, то вот проект, правда на ассемблере, для демонстрации матричной клавиатуры по прерываниям с состояниями подключение в архиве

Дмитрий on 23.07.2016 в 04:01

Я может ошибаюсь. Но по-моему диоды надо развернуть в обратную сторону. И второе. В некоторых устройствах делают вход в меню путём зажатия одной клавиши и нажиманием другой клавиши. Ну, что бы случайно не войти. Так вот. При таком алгоритме опроса, как реализовать что бы отслеживалось нажатие 2 клавиш одновременно. В коде я посмотрю, там переменная с кодом кнопки всё время перезаписывается…

Viktor on 31.10.2016 в 02:09

Уважаемый Админ Вот беда уже 4 -тые сутки не могу разобраться с клавиатурой , Дело в том что у меня
строки (PORTC.4 по PORTC.7), а
СТОЛБЦЫ (PORTD.2 PORTD.3 и PORTC.2,PORTC.3) не могу использовать один порт ну схема такая помогите пожалуйста

Viktor on 01.11.2016 в 23:29

Дарагой админ может чемто подскажите что делать по вашему примеру сканируется весь порт ,а это не допустима что делать???

admin on 02.11.2016 в 15:24

не очень понимаю ваш вопрос, поэтому не знаю чем помочь. проверяйте каждый пин отдельно, если не можете использовать порт целиком.

Ярослав on 28.11.2016 в 02:38

Здравствуй, админ.
Я до компании с глупыми вопросами))
Переделую по новой программу, и столкнулся с проблемой, клавиатура в протеусе работает, на железе нет…
На двух разных чипах (atmega 16a) пробовал…
Опрашиваю в прерыванию по таймеру1 с периодом 0.05 сек.
К любым двум ножкам которые включают столбцы прислосяю светодиодик в любой полярности, он немножко засвечивается, логически что ножками дрыгает…
Собрано всё на «минимальной системной плате»

Ярослав on 28.11.2016 в 02:45

Раньше эта клава была на другом порту и с другой начинкой, была обработка внутри опроса, сейчас сделал флаги…
Раньше проблем не было, сейчас для проверки выводу флаг на дисплей, а в реальном железе эффекта нет…
Может быть из за помех каких-то?

admin on 28.11.2016 в 13:47

Может из за не правильной настройки порта, или банальной невнимательности, тут бесполезно гадать, тестер в руки и проверять каждый пин по отдельности, затем уже смотреть на логику самой программы.

Ярослав on 28.11.2016 в 15:30

ну ножки которые включают столбцы (PB4-PB7) дрыгаются, ножки которые опрашивают строки подтянуты к плюсу и при нажатии клавиши появляется логический ноль в такт пульсации столбцов…

в опросе на одной из клавиш написал чтобы на ножке другого порта сделала лог. единицу когда идет опрос, то этого не происхотит, будто клавиша не нажата, так торт работает, правда на вход еще не проверял, думаю сегодня вечером напишу програмку для проверки…

вот код :
порт:
```
DDRB=(1<<DDB7) | (1<<DDB6) | (1<<DDB5) | (1<<DDB4) | (0<<DDB3) | (0<<DDB2) | (0<<DDB1) | (0<<DDB0);
PORTB=(1<<PORTB7) | (1<<PORTB6) | (1<<PORTB5) | (1<<PORTB4) | (1<<PORTB3) | (1<<PORTB2) | (1<<PORTB1) | (1<<PORTB0);
```

функции:
```
void key_single (int d){

static int pb_old;
```

```
pb_new = PORTB;

contr=1;

if(pb_new==pb_old){
if (k= 4)
dig=d;
}

void key_long (int sh,int lo){

static int pb_old;

pb_new = PORTB;

contr=1;

if(pb_new==pb_old){
if (k= 4 & k=10)
dig=lo;
}
```

сам опрос (в прерывании по таймеру, период 0.05сек.):

```
contr=0;
for (i=0; i=4){
key=dig;
pb_new=0;}

k=0;
}
```

- 
Ярослав on 28.11.2016 в 15:32

```
contr=0;
for (i=0; i=4){
key=dig;
pb_new=0;}

k=0;
}
```

- 
Ярослав on 28.11.2016 в 15:33

ой…
что-то не получается отправить опрос…

- 
Ярослав on 28.11.2016 в 16:15

Сделал на одном из столбцов задержку в 2 милисекунды после как включил столбец и перед тем как проверять состояния и этот столбец заработал…

- 
Ярослав on 28.11.2016 в 16:24

опыты показали что и одной микро секунды хватает…

- 
Serhio on 04.12.2016 в 08:03

Извиняюсь за повтор.
Здравствуйте. Скажите строчка:
if(((PIND&inputState[j])==0))
точнее ее часть:
PIND&inputState[j]
равносильно этой ?:
PIND = PIND&inputState[j]
Результат сохраняется в PIND ???
Я понимаю мы по очереди перебираем?!!
PIND.0 == 0
PIND.1 == 0
PIND.2 == 0
PIND.3 == 0

- 

admin on 04.12.2016 в 15:47

да по очереди перебираем, но нет не равносильна. смотрите есть порт D, точнее PIND это состояние всего порта, т.е. если нам нужно проверить первую ногу мы бы могли это сделать так:

```
if(PIND.0 == 0)...
```

либо

```
if(PIND & 0b00000001 == 0)
```

или так

```
if(PIND & 0x01 == 0)
```

. Оба варианта дадут одинаковый результат, вы проверите состояние ноги, просто вторая запись более удобна для перебора массивом.

Добавить комментарий

Ваш e-mail не будет опубликован. Обязательные поля помечены *

Имя *

E-mail *

Сайт

Комментарий

Можно использовать следующие HTML-теги и атрибуты: `<a href="" title=""> <abbr title=""> <acronym title=""> <b> <blockquote cite=""> <cite> <code> <del datetime=""> <em> <i> <q cite=""> <strike> <strong>`

Отправить комментарий

3 комментария

Ваш комментарий...

поделиться Отправить

Сергей Барковский
Здравствуйте. Скажите строчка:
if(((PIND&inputState[j])==0))
точнее ее часть:
PIND&inputState[j]
равносильно этой ?:
Показать полностью...
3 дек 2016 Комментировать 3

Дмитрий Верещагин
Спасибо за помощь, прямо толчек моего долгостроя. Твой метод последовательного обьяснения гениален!
13 сен 2013 Комментировать 1

Иван Герасимов
для работы в железе, скорее всего, после переключения PORTD=portstate необходимо будет сделать паузу #asm ("nop"), потому что иначе реальный порт мк не будет успевать переключаться - я как раз напоролся на такие грабли
26 авг 2013

Артур Тележкин
спасибо за инфу, может кому пригодится
27 авг 2013 Ответить

Евгений Негробов
Я где там только задержки не пихал, все до одного места.
Ну, они избавляют от дребезга, но проблему одной кнопки не решают.
Так что "скорее всего" не катит, надо конкретно найти косяк и проверить в железе работоспособность.
24 янв 2014 Ответить

Написать комментарий...

Свежие записи

- Консоль на STM32 11.10.2017

Последние комментарии

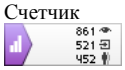
- Зёзя on Урок 8. Передача данных через UART в AVR микроконтроллерах
  Спасибо за статью, с...

- Alexsk88 on STM32 работа с Flash.
  Привет автор. Усердн...

Рубрики

- AVR Учебный курс (42)
- Linux (16)
- STM32 (21)
- ПЛИС (15)
- Программирование (22)
- Разное (4)
- Схемы (6)
- Электроника для начинающих (19)

Счетчик



AVR-START.RU