

Свежие комментарии

- SmNikolay к записи [STM Урок 89. LAN. ENC28J60. TCP WEB Server. Подключаем карту SD](#)
- Narod Stream к записи [AVR Урок 3. Пишем код на СИ. Зажигаем светодиод](#)
- strannik2039 к записи [AVR Урок 3. Пишем код на СИ. Зажигаем светодиод](#)
- Dmitriy к записи [AVR Урок 1. Знакомство с семейством AVR](#)
- Narod Stream к записи [STM Урок 9. HAL. Шина I2C. Продолжаем работу с DS3231](#)

Форум. Последние ответы

- Narod Stream в [Программирование МК STM32](#)
1 неделя, 2 дн. назад
- Zandy в [Программирование МК STM32](#)
1 неделя, 3 дн. назад
- Narod Stream в [Программирование МК STM32](#)
3 нед. назад
- Narod Stream в [Программирование МК STM32](#)
3 нед. назад
- fireweb в [Программирование МК STM32](#)
3 нед., 3 дн. назад

Январь 2018

Пн	Вт	Ср	Чт	Пт	Сб	Вс
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				
« Дек						

Архивы

- Январь 2018
- Декабрь 2017
- Ноябрь 2017
- Октябрь 2017
- Сентябрь 2017
- Август 2017
- Июль 2017

Главная > I2C > AVR Урок 16. Интерфейс TWI (I2C). Часть 4

AVR Урок 16. Интерфейс TWI (I2C). Часть 4

Posted on Декабрь 17, 2016 by Narod Stream
Опубликовано в I2C, Программирование AVR — 7 комментариев ↓

Изготовление печатных плат партий. Звони pcb.electropribor-penza.ru Ад

Одноплатные компи .
Процессорные модули, PC-104 платы, NANO-ITX, EPIC, PCI-ITX и многие другие!
ipc2u.ru Адрес и телефон

Урок 16 Часть 4

Интерфейс TWI (I2C)

В предыдущей части занятия мы поближе познакомились со всеми практически регистрами шины TWI и уже написали код отправки в шину условия СТАРТ. Причем не только написали, а ещё и проверили отправку, считав данные из регистра статуса через порт USART. Также мы бегло ознакомились с протоколом передачи и приема. Если нам впоследствии что-то потребуется, то мы к нему ещё вернёмся.

Будем теперь думать, как нам ещё что-нибудь в шину I2C отправить. Для этого будут ещё нужны некоторые функции. Раз уж есть функция для условия СТАРТ, то, конечно же, нужна функция для передачи условия СТОП.

Поэтому зайдём в файл twi.c и напишем её, скопировав полностью предыдущую функцию и немного её подправив

```
void I2C_StopCondition(void)
{
    TWCR = (1<<TWINT)|(1<<TWSTO)|
    (1<<TWEN);
}
```

ОБРАЗОВАТЕЛЬНЫЙ ЦЕНТР ПВ

КУРС iOS РАЗРАБОТЧИКА

Высокая востребованность на рынке IT!

Заходите на канал
Narod Stream

- [Июнь 2017](#)
- [Май 2017](#)
- [Март 2017](#)
- [Февраль 2017](#)
- [Январь 2017](#)
- [Декабрь 2016](#)
- [Ноябрь 2016](#)

Здесь только изменился бит TWSTA на TWSTO, а также в этом случае мы уже никакого подтверждения от ведомого не ждём.

Также нам нужно написать функцию для передачи байта в шину I2C

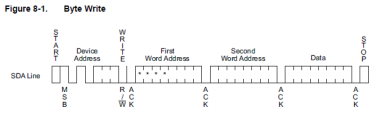
```
void I2C_SendByte(unsigned char c)
{
    TWDR = c; //запишем байт в регистр данных
    TWCR = (1<<TWINT)|(1<<TWEN); //включим передачу байта
    while (!(TWCR & (1<<TWINT))); //подождём пока установится TWIN
}
```

Здесь мы сначала записываем байт, предназначенный для отправки, в регистр TWDR, затем запустим передачу байта, установив биты TWINT и TWEN и здесь мы уже ждём подтверждения от ведомого устройства. То есть, если мы передаём адрес устройства, то если среди всех присутствующих на шине найдётся именно владелец данного адреса, то он и установит шину в низкое состояние, после чего контроллер, поняв это, установит бит TWINT в ноль, ну, или сбросит его.

Также напоминаю о необходимости добавления прототипов на данные функции в файле twi.h.

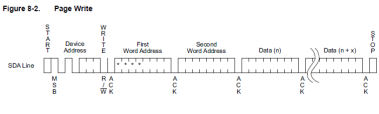
Теперь продолжим, используя вышенаписанные функции, писать код в функцию main(), чтобы нам удостовериться, что у нас шина I2C исправна и устройство, подключенное к ней, нормально откликается.

Также для достижения данной цели нам необходимо изучить, как именно нужно передавать данные нашей микросхеме EEPROM. Для этого зайдём в техническую документацию и посмотрим следующую картинку




Здесь рассказано, как передать один байт для записи в определённый адрес памяти микросхемы. Сначала мы передаём адрес и бит записи, затем подтверждение, затем старший байт адреса памяти, подтверждение, младший бит адреса памяти, подтверждение, байт данных для записи, подтверждение и СТОП.

Только нам вовсе не интересно в данную память передавать по одному байту данных, поэтому существует ещё одна картинка в даташите



Здесь мы в начале наблюдаем ту же картину. Старт, адрес устройства с битом записи, подтверждение, старший байт адреса памяти, с которого начинаем запись байтов, подтверждение, младший



Напишите нам

Рубрики

- [1-WIRE](#) (3)
- [ADC](#) (6)
- [DAC](#) (4)
- [GPIO](#) (26)
- [I2C](#) (19)
- [SPI](#) (13)
- [USART](#) (8)
- [Программирование AVR](#) (131)
- [Программирование PIC](#) (7)
- [Программирование STM32](#) (213)
- [Тесты устройств и аксессуаров](#) (1)

31 ДЕНЬ

07 ДНЕЙ

24 ЧАСА

СЕГОДНЯ

НАПИСАШ

124 507

13 098

30 048

4 365

5 253


1 071

2 568

580

58


Яндекс.Директ



Очень горячая
аниме игра

Эта аниме игра
поглощает с первых

байт адреса памяти, подтверждение, и затем идут подряд байты с подтверждениями, которые будут укладываться в ячейки памяти EEPROM, начиная с переданного адреса до тех пор, пока мы на шине не сгенерируем условие СТОП.



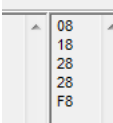
Xiaomi

24zakaz.by

Ну, давайте же что-то попытаемся записать в микросхему. Также после каждой операции будем смотреть статус её выполнения. Напишем следующий код в функцию main()

```
I2C_StartCondition(); //Отправим
условие START
USART_Transmit(TWSR); //читаем
статусный регистр
I2C_SendByte(0b10100000); //передаем
адрес и бит записи (0)
USART_Transmit(TWSR); //читаем
статусный регистр
I2C_SendByte(0); //переходим на
0x0000 – старший байт адреса памяти
USART_Transmit(TWSR); //читаем
статусный регистр
I2C_SendByte(0); // – младший байт
адреса памяти
USART_Transmit(TWSR); //читаем
статусный регистр
I2C_StopCondition(); //Отправим
условие STOP
USART_Transmit(TWSR); //читаем
статусный регистр
while(1)
```

Чем занимается данный код, прекрасно видно из комментариев. Соберём код и прошьём контроллер. Глянем в терминальную программу




И теперь с помощью таблицы будем разбираться, что же нам "рассказал" регистр TWSR


Table 66. Status codes for Master Transmitter Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the Two-wire Serial Bus and Two-wire Serial Interface Hardware	Application
0x08	A START condition has been transmitted	Load SLA+W
0x18	SLA+W has been transmitted; ACK has been received	Load data byte or
0x28	Data byte has been transmitted; ACK has been received	Load data byte or

Вот коды, которые мы получали
0x08 — условие СТАРТ было передано
0x18 — адрес и бит записи был передан и подтверждение получено
0x28 — байт данных был передан и подтверждение получено
Адрес ячейки памяти для нашего контроллера таковым не считается, это для него обычные данные.



минут, начнешь играть и забудешь про сон ¹⁸⁺
[Все об игре](#)
[Выбери свой класс](#)
[Следи за новостями](#)
[Тебя ждет подарок](#)
[promo.101xp.com](#)



Разработка мобильных приложений.

Разрабатываем все типы мобильных приложений для любых нужд бизнеса. Звоните!

[Стартапы](#)
[Коммерческие приложения](#)
[Справочные приложения](#)
[narisuemvse.by](#)
[Адрес и телефон](#)

подарки: купи с каждого он сейчас!

А **0xFF** — это типа ошибки. Ну нам уже это не важно. Скорей всего это потому, что после СТОП ведомый уже перед нами не отчитывается и флаг также не сбросится.

В **следующей части занятия** мы постараемся уже что-то в память микросхемы записать.

*Предыдущая
часть*

*Программирование
МК AVR*

*Следующая
часть*

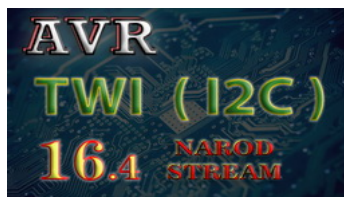
Техническая документация на микросхему AT24C32

Программатор и модуль RTC DS1307 с микросхемой памяти можно приобрести здесь:

Программатор (продавец надёжный)
USBASP USBISP 2.0

Модуль RTC DS1307 с микросхемой памяти

Смотреть **ВИДЕОУРОК** (нажмите на картинку)



👁 Post Views: 580

◀ AVR Урок 16.

Интерфейс TWI

7 комментариев на "AVR Урок 16: Интерфейс TWI (I2C). Часть 4"



Евгений: Интерфейс TWI
Июль 16, 2017 в 12:33 дп

А почему мы не используем прерывания? Ведь если у нас ведомый в процессе приёма отвалится, то мы навечно подвесим МК в этом цикле:
`while (!(TWCR & (1<<TWINT)));`
Но даже если ведомый примет посылку, получается, что всё время передачи, несмотря на наличие аппаратного TWI наш микроконтроллер будет работать с частотой 100 кГц.

Или я чего-то не понимаю?

[Ответить](#)



admin:
Июль 16, 2017 в 7:23 дп

Потому что во время выхода данного урока я ещё не знал, как их использовать.

[Ответить](#)



Сергей:

Июль 16, 2017 в 8:09 дп

Сделал все как указано, но у меня вместо 18 и 28 получаются соответственно 20 и 30, т.е. у меня отправляется не сигнал ACK, а сигнал NOT ACK. Что я делаю не так?

[Ответить](#)

admin:

Июль 17, 2017 в 5:46 дп

Флаг должен быть правильный установлен. Посмотрите урок по внешнему EEPROM, там мы используем и первый и другой вариант (и с подтверждением и без).

[Ответить](#)

Сергей:

Июль 16, 2017 в 4:22 пп

Добавление к предыдущему вопросу. Модуль спаял сам, с ардуинкой работаем без проблем — время показывает. Но сигнал возвращается именно NO ACK.

[Ответить](#)

Сергей:

Июль 16, 2017 в 5:45 пп

Нашел проблему. Если NO ACK указывает, что по данному адресу устройство не обнаружено. Запустил на ардуинке i2c_scanner, который показал адрес ds1307 есть 0x68. перевел в двоичное — 1101000, добавил 0 как флак записи. Т.е. получилось I2C_SendByte(0b11010000); После этого получил долгожданные 18 и 28.

[Ответить](#)

admin:

Июль 17, 2017 в 5:46 дп

Отлично!

[Ответить](#)

Добавить комментарий

Ваш e-mail не будет опубликован.

Обязательные поля помечены *

Комментарий



Имя *

E-mail *

Сайт

☐ × 4 = 32 ↺

Отправить комментарий

- Главная | Новости | Уроки по программированию МК
 - Программирование микроконтроллеров AVR
 - Программирование микроконтроллеров STM32
 - Программирование микроконтроллеров PIC
 - Тесты устройств и аксессуаров
- Устройства и интерфейсы | Ссылки | Форум | Помощь



Наверх