



Главная | Новости | Уроки по программированию МК | Устройства и интерфейсы | Ссылки | Форум | Помощь

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., 2 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 17. Часы реального времени DS1307. Часть 2

AVR Урок 17. Часы реального времени DS1307. Часть 2

Posted on Декабрь 25, 2016 by Narod

Stream Опубликовано в I2C, Программирование AVR — 2 комментария ↓

Мета

- [Регистрация](#)
- [Войти](#)
- [RSS записей](#)
- [RSS комментариев](#)
- [WordPress.org](#)

искать здесь ...

Фильтровать

Яндекс.Директ



Жаркая аниме игра 2017 года

Эта аниме игра затягивает с первых минут, начнешь играть и забудешь про сон 18+
promo.101xp.com

Яндекс.Директ



Печатные Платы на Заказ. Звоните!

Изготовление печатных плат на заказ. От прототипов до крупных партий. Звони pcb.electropribor-penza.ru Адрес и телефон

Урок 17 Часть 2

Часы реального времени DS1307

В предыдущей части занятия мы познакомились с отличной микросхемой реального времени **DS1307**, подключили её, и начали писать исходный код для установки времени, даты и дня недели.

Продолжим писать наш код, используя тот же самый проект

Используя написанную функцию перевода из десятичного формата числа в двоично-десятичный, занесём данные в регистры микросхемы, начиная с нулевого адреса (также мы знаем что это и адрес самого первого регистра микросхемы)

```
I2C_SendByte(0); //Переходим на 0x00
I2C_SendByte(RTC_ConvertFromBinDec(0)); //секунды
```



Заходите на канал
Narod Stream

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

```
I2C_SendByte(RTC_ConvertFromBinDec(3
1)); //минуты
I2C_SendByte(RTC_ConvertFromBinDec(2
0)); //часы
I2C_SendByte(RTC_ConvertFromBinDec(5
)); //день недели
I2C_SendByte(RTC_ConvertFromBinDec(2
9)); //дата
I2C_SendByte(RTC_ConvertFromBinDec(1
)); //месяц
I2C_SendByte(RTC_ConvertFromBinDec(1
6)); //год
```

Ну, у вас конечно будут другие данные, в зависимости от того момента времени, в который вы будете запускать данную программу.

Ну, и в конце, конечно условие STOP

```
I2C_SendByte(RTC_ConvertFromBinDec(1
6)); //год
I2C_StopCondition();
while(1)
```

Соберем код, установим в скобки правильное время и пошьём контроллер.

Пока мы никак не можем проверить, как это всё записалось и ходят ли часы. Для этого нужно написать код для чтения.

Можно конечно было бы посмотреть статусы, но раз уж нам всё равно читать показания регистров, то особого смысла в этом не вижу.

Пока данный код записи в регистры полностью закомментируем. Мы будем его раскомментировать тогда, когда будем программировать новую микросхему и заносить данные регистры, каждый раз это делать нужно.

Теперь в бесконечный цикл начнем писать код считывания данных из регистров

Точно также всё начинается с условия СТАРТ.

Но для того, чтобы нам всю эту рутину каждый раз не писать, давайте напомним функцию для передачи по адресу устройства байта адреса памяти

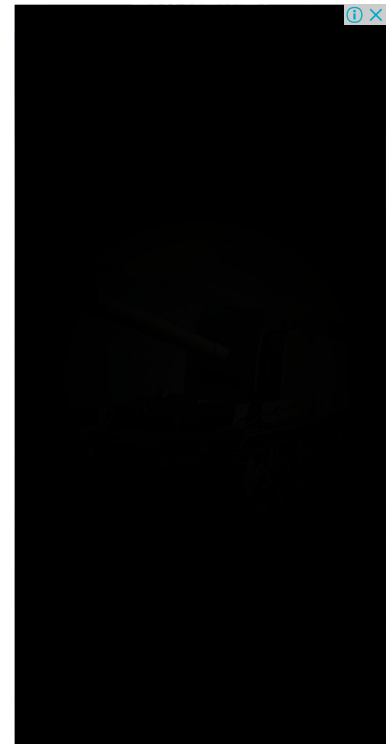
```
void I2C_SendByteByADDR(unsigned
char c,unsigned char addr)
{
    I2C_StartCondition(); // Отправим
    условие START
    I2C_SendByte(addr); // Отправим в
    шину адрес устройства + бит чтения-
    записи
    I2C_SendByte(c); // Отправим байт
    данных
    I2C_StopCondition(); // Отправим
    условие STOP
}
```

Также заодно напомним функцию чтения обычного байта из шины и чтения последнего байта из шины. У нас такие функции уже были, но они были в особом файле и были с префиксом EE_, а также там была обработка ошибки. Скопируем их себе теперь в



narod stream Просмотреть сайт: Владелец

[Главная](#) [Видео](#) [Плейлисты](#) [Каналы](#) [Обсуждение](#) [О канале](#)



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)

	7
31 ДЕНЬ	124 507
	13 098
07 ДНЕЙ	30 048
	4 366
24 ЧАСА	5 253
	1 071
СЕГОДНЯ	2 570
	580
НАПЛИШ	52
	24

Яндекс.Директ

Нужно
программир-е
контроллеров?

стандартный обычный TWI.c, убрав префиксы и всё лишнее

```
unsigned char I2C_ReadByte(void)
{
    TWCR = (1<<TWINT)|(1<<TWEN)|
    (1<<TWEA);
    while (!(TWCR & (1<<TWINT)));//
    ожидание установки бита TWIN
    return TWDR;//читаем регистр
    данных
}

unsigned char I2C_ReadLastByte(void)
{
    TWCR = (1<<TWINT)|(1<<TWEN);
    while (!(TWCR & (1<<TWINT)));//
    ожидание установки бита TWIN
    return TWDR;//читаем регистр
    данных
}
```

Напишем в TWI.h на все эти функции прототипы

```
void I2C_SendByte(unsigned char c);
//передача байта в шину
void I2C_SendByteByADDR(unsigned
char c,unsigned char addr); //
передача байта в шину на устройство
по адресу
unsigned char I2C_ReadByte(void); //
читаем байт
unsigned char
I2C_ReadLastByte(void); //читаем
последний байт
```

И теперь можно начинать писать код в бесконечный цикл функции main().

Отправим адрес 0 (адрес первого регистра) по адресу устройства

```
while(1)
{
    //Читаем время
    I2C_SendByteByADDR(0,0b11010000);
    //переходим на адрес 0
```

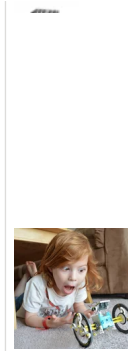
Затем вставим задержку на 300 миллисекунд. Данную задержку можно и в конце кода вставить, ну давайте попробуем здесь, поэкспериментируем, так сказать

```
I2C_SendByteByADDR(0,0b11010000); //
переходим на адрес 0
_delay_ms(300);
```

Затем посылаем в шину адрес устройства с битом чтения, отправив перед этим условие СТАРТ

```
_delay_ms(300);
I2C_StartCondition(); //Отправим
условие START
I2C_SendByte(0b11010001); //отправим
в устройство бит чтения
```

По идее мы вообще так не должны делать, так как у нас в функции, которую мы вызвали перед задержкой было в конце условие СТОП, а это не требуется. Обычно сразу СТАРТ. Но можно и так, всё работает. Если указатель установлен



Комплексное обучение. Доступные цены. Минимальные сроки. Большой опыт!
 О компании
 Услуги
 Продукция
 Преимущества
 festo.com
 Адрес и телефон
Робот-Конструктор 14 в 1. Новинка!
 14 абсолютно разных роботов! Детей не оторвать от этого конструктора!
 14v1.ru

уже туда, куда надо, то можно сразу адрес чтения и начинать читать, а не так, как было в случае с EEPROM.



Pressure Transmitters

Professional pressure transducer manufacturer in water pump
Get live quote now

Дальше читаем все регистры

```
I2C_SendByte(0b11010001); //
отправим в устройство бит
чтения
sec = I2C_ReadByte();
min = I2C_ReadByte();
hour = I2C_ReadByte();
day = I2C_ReadByte();
date = I2C_ReadByte();
month = I2C_ReadByte();
year = I2C_ReadLastByte();
```

Помним, что последний байт читается из шины без подтверждения и для этого у нас есть соответствующая функция.

В конце чтения отправим в шину условие СТОП

```
year = I2C_ReadLastByte();
I2C_StopCondition(); //Отправим
условие STOP
```

Далее, используя функцию преобразования из двоично-десятичного формата в десятичный, преобразуем считанные показания, так как мы их забрали из регистров именно в двоично-десятичном виде.

```
I2C_StopCondition(); //Отправим
условие STOP
sec = RTC_ConvertFromDec(sec); //
Преобразуем в десятичный формат
min = RTC_ConvertFromDec(min); //
Преобразуем в десятичный формат
hour = RTC_ConvertFromDec(hour); //
Преобразуем в десятичный формат
day = RTC_ConvertFromDec(day); //
Преобразуем в десятичный формат
year = RTC_ConvertFromDec(year); //
Преобразуем в десятичный формат
month = RTC_ConvertFromDec(month);
//Преобразуем в десятичный формат
date = RTC_ConvertFromDec(date); //
Преобразуем в десятичный формат
```

И в конце бесконечного цикла отправим всё это в определённом виде в USART

```
date = RTC_ConvertFromDec(date); //
Преобразуем в десятичный формат
USART_Transmit(date/10+0x30); //
Преобразуем число в код числа
USART_Transmit(date%10+0x30); //
Преобразуем число в код числа
USART_Transmit('.');
USART_Transmit(month/10+0x30); //
Преобразуем число в код числа
USART_Transmit(month%10+0x30); //
Преобразуем число в код числа
USART_Transmit('.');
USART_Transmit(year/10+0x30); //
Преобразуем число в код числа
```

```

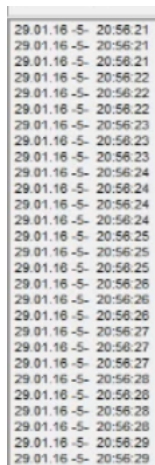
USART_Transmit(year%10+0x30);//
Преобразуем число в код числа
USART_Transmit(' ');
USART_Transmit('-');
USART_Transmit(day+0x30);//
Преобразуем число в код числа
USART_Transmit('-');
USART_Transmit(' ');
USART_Transmit(' ');
USART_Transmit(hour/10+0x30);//
Преобразуем число в код числа
USART_Transmit(hour%10+0x30);//
Преобразуем число в код числа
USART_Transmit(':');
USART_Transmit(min/10+0x30);//
Преобразуем число в код числа
USART_Transmit(min%10+0x30);//
Преобразуем число в код числа
USART_Transmit(':');
USART_Transmit(sec/10+0x30);//
Преобразуем число в код числа
USART_Transmit(sec%10+0x30);//
Преобразуем число в код числа
USART_Transmit(0x0d); //переход в
начало строки
USART_Transmit(0x0a); //перевод
каретки
}

```

Смещение на 0x30 в вычисление кода символа — это преобразование самой цифры в код цифры. Именно такая разница и есть в таблице **ascii**.

Можно конечно не париться с таким преобразованием и использовать функцию **sprintf** и она прекрасно с этим справится, но так интересно, функция **sprintf** ещё себя покажет, это я уж вам обещаю точно.

Соберём код, откроем терминальную программу, нажмём там **Connect**, Прошьём контроллер и посмотрим результат



Наши часы отлично ходят.

Предыдущая
часть

Программирование
МК AVR

Следующий
урок

Исходный код

Документация на микросхему DS1307

Программатор, модуль RTC DS1307 с микросхемой памяти и переходник USB-TTL можно приобрести здесь:

Программатор (продавец надёжный)
USBASP USBISP 2.0

Модуль RTC DS1307 с микросхемой памяти

Переходник USB-TTL лучше
купить такой (сейчас у меня именно такой и он мне больше нравится)

Смотреть ВИДЕОУРОК
(нажмите на картинку)



Post Views: 561

< AVR Урок 17.

Часы реального

2 комментария на "AVR Урок 17. Часы реального времени DS1307. Часть 2"



Vlad:

Январь 23, 2017 в 17:40

Превосходно. Особенно помогли ошибки, которые Вы в ходе урока разбираете и устраняете. Мне это помогло в поиске своих ошибок гораздо больше чем многотомники по C++. Благодарю Вас за прекрасные уроки.

[Ответить](#)



admin:

Январь 24, 2017 в 4:15 дп

Вам также спасибо за внимание к моим занятиям!

[Ответить](#)

Добавить комментарий

Ваш e-mail не будет опубликован.

Обязательные поля помечены *

Комментарий

Имя *

E-mail ***Сайт**3 - ● = ↺[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](#)

© 2018 Narod Stream