Главная › I2C › AVR Урок 17. Часы реального времени DS1307. Часть 1

AVR Урок 17. Часы реального времени DS1307. Часть 1

Posted on Декабрь 25, 2016 by Narod Stream Опубликовано в I2C, Программирование AVR — 1 комментарий ↓

Урок 17
Часть 1

Часы реального времени DS1307

Продолжаем занятия по программированию **МК AVR**.

И сегодня мы познакомимся с очень хорошей микросхемой **DS1307**. Данная микросхема представляет собой **часы реального времени** (**real time clock** или **RTC**).

Также, благодаря тому, что общение микроконтроллера с данной микросхемой будет происходить с применением интерфейса **I2C**, мы ещё лишний раз на деле закрепим тему программирования данной шины.

Данная микросхема представлена компанией **Dallas**, вот её распиновка и основные технические характеристики

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

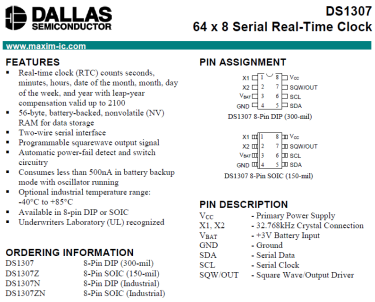
- Январь 2018
- Декабрь 2017
- Ноябрь 2017
- Октябрь 2017
- Сентябрь 2017
- Август 2017
- Июль 2017

Мета

- Регистрация
- Войти
- RSS записей
- RSS комментариев
- WordPress.org

искать здесь …
Фильтровать

- Июнь 2017
- Май 2017
- Март 2017
- Февраль 2017
- Январь 2017
- Декабрь 2016
- Ноябрь 2016



DALLAS SEMICONDUCTOR

DS1307
64 x 8 Serial Real-Time Clock

www.maxim-ic.com

FEATURES
- Real-time clock (RTC) counts seconds, minutes, hours, date of the month, day of the week, and year with leap-year compensation valid up to 2100
- 56-byte, battery-backed, nonvolatile (NV) RAM for data storage
- Two-wire serial interface
- Programmable squarewave output signal
- Automatic power-fail detect and switch circuitry
- Consumes less than 500nA in battery backup mode with oscillator running
- Optional industrial temperature range: -40°C to +85°C
- Available in 8-pin DIP or SOIC
- Underwriters Laboratory (UL) recognized

PIN ASSIGNMENT

X1 1 8 Vcc
X2 2 7 SQW/OUT
VBAT 3 6 SCL
GND 4 5 SDA
DS1307 8-Pin DIP (300-mil)

X1 1 8 Vcc
X2 2 7 SQW/OUT
VBAT 3 6 SCL
GND 4 5 SDA
DS1307 8-Pin SOIC (150-mil)

ORDERING INFORMATION

DS1307	8-Pin DIP (300-mil)
DS1307Z	8-Pin SOIC (150-mil)
DS1307N	8-Pin DIP (Industrial)
DS1307ZN	8-Pin SOIC (Industrial)

PIN DESCRIPTION

Vcc	- Primary Power Supply
X1, X2	- 32.768kHz Crystal Connection
VBAT	- +3V Battery Input
GND	- Ground
SDA	- Serial Data
SCL	- Serial Clock
SQW/OUT	- Square Wave/Output Driver

Здесь мы видим, что есть у нас ножки SDA и SCL, назначение которых мы очень прекрасно знаем из **предыдущего занятия**. Также есть ножки X1 и X2 для подключения кварцевого резонатора на 32768 Гц, ножки питания — VCC и GND, выход для импульсов продолжительностью 1 секунда либо другой частоты в зависимости от настроек определенных регистров, а также плюсовой контак для батарейки, которая подключается для поддержания хода часов в момент отключения основного питания. Отрицательный контакт данной батарейки мы подключаем к общему проводу питания.

Также мы видим, что данная микросхема исполняется в планарных и DIP-корпусах.

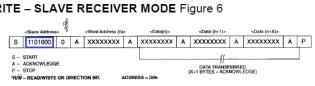
Питаться данная микросхема может как и от 3 вольт, так и от 5 вольт.

Обращение к данной микросхеме по интерфейсу I2C происходит, в принципе, также. как и к микросхеме памяти, которую мы использовали на прошлом уроке. Конечно, будут свои нюансы, но об этом позже.

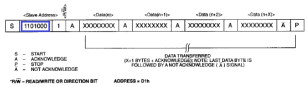
Так как данная микросхема у меня установлена в том же модуле, в котором установлена и микросхема EEPROM, а шина обмена у нас одна, то "узнавать" микросхема DS1307 о том, что обращаются именно к ней, будет, конечно, по адресу, который у неё другой, нежели у микросхемы EEPROM.

Вот диаграммы приёма и передачи данных микросхемы

DATA WRITE – SLAVE RECEIVER MODE Figure 6



DATA READ – SLAVE TRANSMITTER MODE Figure 7



Адрес, по которому мы будем обращаться к данной микросхеме, выделен синим.

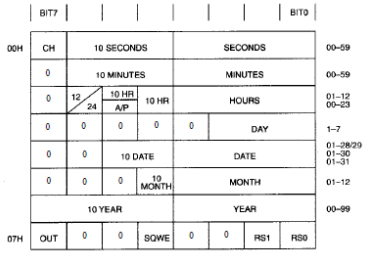
В принципе. особой разницы с диаграммами микросхемы EEPROM мы на видим.

Ещё отличие в обращении будет в том, что адресация памяти будет уже однобайтная, так как ячеек памяти или **регистров** у данной микросхемы очень мало.

Вот что из себя представляют данные регистры

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)



	BIT7						BIT0	
00H	CH	10 SECONDS			SECONDS			00–59
	0	10 MINUTES			MINUTES			00–59
	0	12/24	10 HR A/P	10 HR	HOURS			01–12 00–23
	0	0	0	0	0	DAY		1–7
	0	0	10 DATE		DATE			01–28/29 01–30 01–31
	0	0	0	10 MONTH	MONTH			01–12
	10 YEAR				YEAR			00–99
07H	OUT	0	0	SQWE	0	0	RS1	RS0

Назначение данных регистров:

00h — секунды. Секунды хранятся в двоично-десятичном виде. То есть в младших 4 битах хранятся единицы секунд, а в более старших трёх — десятки. Также есть бит SH — это бит запуска микросхемы.

01h — минуты. Хранятся аналогично.

02h — более универсальный регистр. Здесь хранятся часы. В четырех младших битах — единицы чаов, в следующих более старших двух — десятки, в следующем 6 бите — флаг того, после полудня сейчас время или до полудня, в 7 бите — режим хранения — 12- часовой или 24-часовой.

03h — день недели. Хранится в младших 3 битах, остальные биты не используются.

04h — здесь хранится день месяца, также в двоично-десятичном формате. В четыреё малдших битах — единицы, в двух следующих постарше — десятки, остальные биты не используются.

05h — номер месяца в году — хранится в двоично-десятичном формате точно также, как и часы.

06h — номер года, причём не полный четырёхзначный, а только двузначный. В младших четырех битах — единицы, в старших — десятки.

Вот этими семью регистрами мы и будем пользоваться. Последний регистр предназначен для конфигурирования частоты импульсов на импульсном выходе микросхемы, это делается в младших двух битах регистра. по умолчанию он будет 1 гц частотой, нам этого достаточно, чтобы помигать двоеточием, поэтому мы не будем пользоваться данными битами. Биты SOWE и OUT также применяются для настройки и включения формирователя даннх квадратных импульсов.

Проект для работы с данной микросхемой был создан обычным образом с именем **MyClock1307**, файлы, связанные с EEPROM оттуда убраны, а добавлены файлы **RTC.c** и **RTC.h**.

Содержание файла main.h у нас теперь вот такое

```
#ifndef MAIN_H_
#define MAIN_H_
#define F_CPU 8000000UL
```

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdio.h>
#include <stdlib.h>
#include "usart.h"
#include "twi.h"
#include "RTC.h"
#endif /* MAIN_H_ */
```

В главном файле **MyClock1307.c** создадим глобальные переменные для хранения показаний времени, даты и дня недели и после этого полное содержание после удаления всего лишнего в нём будет вот таким

```
#include "main.h"
unsigned char
sec,min,hour,day,date,month,year;
int main(void)
{
  I2C_Init();
  USART_Init (8);
  while(1)
  {
  }
}
```

От прошлого кода останется лишь инициализация I2C и USART.

Теперь нам надо как-то вообще запустить микросхему. Если микросхема новая, либо никогда не использовалась, либо кто-то специально для каких-то целей изменил значение бита CH, то она ещё не "ходит".

Ну, вообще, как только мы установим все значения в регистрах микросхемы, так она и запустится и наши часы пойдут.

Подключение или схема использована также вся из прошлого занятия, то есть время смотреть мы будем посредством шины USART в терминальной программе.

Поэтому, собственно, используя наши знания предыдущего занятия, напишем писать функцию установки времени.

Первым делом мы, само собой, передадим условие СТАРТ

```
//Устанавливаем время
I2C_StartCondition();
```

Затем передаём адрес с битом записи 0

```
I2C_StartCondition();
I2C_SendByte(0b11010000);
```

Перейдём на адрес 0, а значит к той части памяти, где расположен самый первый регистр

```
I2C_SendByte(0b11010000);
I2C_SendByte(0);//Переходим на 0x00
```

Прежде чем писать какие-то значения в регистры микросхемы, мы вспомним, что числа мы сначала должны преобразовать в двоично-десятичный формат, который будет удобен для регистров. Для этого мы зайдём в файл

RTC.c и такую функцию и напишем. Она будет очень лёгкой и в объяснении не нуждается

```
unsigned char
RTC_ConvertFromBinDec(unsigned char
c)
{
  unsigned char ch = ((c/10)<<4)|
(c%10);
  return ch;
}
```

Ну и также давайте напишем и функцию обратного типа, переводящую число из двоично-десятичного формата в десятичный. С помощью неё мы, наоборот, будем считанные показания времени преобразовывать в вид, удобный нашему восприятию (ЧПИ — человеко-понятный интерфейс)

```
unsigned char
RTC_ConvertFromDec(unsigned char c)
{
  unsigned char ch = ((c>>4)*10+
(0b00001111&c));
  return ch;
}
```

Здесь также всё придельно ясно, мы сдвигаем вправо старшую тетраду байта, умножаем её на десять и прибавляем младшую тетраду (старшую отмаскировываем нулями)

Напишем прототипы данных функций в файле **RTC.c**

```
  #include "main.h"
  unsigned                    char
RTC_ConvertFromDec(unsigned char c);
//перевод двоично-десятичного числа
в десятичное
unsigned char
RTC_ConvertFromBinDec(unsigned char
c); //перевод десятичного числа в
двоично-десятичное
```

Соберём код, а прошивать контроллер пока не будем. Нам нужно ещё дописать код записи в регистры и написать в бесконечный цикл процедуру чтения времени и даты и отправку всего этого в USART, а затем уж прошьём полностью весь код, прописав правильные значения времени и даты в установку времени.

Сделаем мы всё это в **следующей части** занятия.

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)

‹ STM Урок 40. Знакомство с модулем NUCLEO STM32F767ZI

AVR Урок 17. Часы реального времени DS1307. Часть 2 ›

Один комментарий на “AVR Урок 17. Часы реального времени DS1307. Часть 1”

Melamed:
Октябрь 4, 2017 в 11:39 дп

Проанализировав ваши функции **RTC_ConvertFromBinDec** и **RTC_ConvertFromDec**, пришел к выводу, что время и дата в микросхеме DS1307 хранятся в следующим формате: в младших четырех битах хранится десятичные единицы, а старших 4 битах — десятичные десятки. Поэтому достаточно для выделения десятичных единиц выполнить операцию побитного и с числом 0x0F, а десятков с помошью побитного сдвига на 4 бита
ed = min & 0x0f;
dec = min>>4;
И у меня это работает

Ответить

Добавить комментарий

Ваш e-mail не будет опубликован.
Обязательные поля помечены *
Комментарий
Имя *
E-mail *
Сайт

6 − 5 =

Отправить комментарий