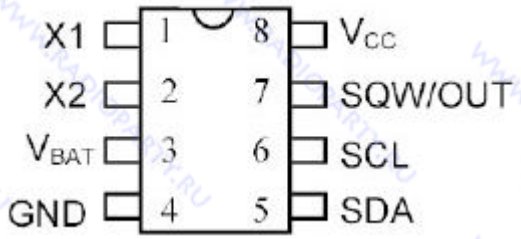
Подключение DS1307 к микроконтроллерам AVR

Цель: Изучить работу микросхемы часов реального времени с интерфейсом I2C (TWI) и сопряжение с микроконтроллером ATmega8.

DS1307 - микросхема часов реального времени с интерфейсом **I2C(TWI)**. Часы/календарь хранят следующую информацию: секунды, минуты, часы, день, дату, месяц и год. Конец месяца автоматически подстраивается для месяцев, в которых менее 31 дня, включая поправку для високосного года. Часы работают в 24-часовом или 12-часовом формате с индикатором AM/PM. DS1307 имеет встроенную схему контроля питания, которая обнаруживает пропадание питания и автоматически переключает схему на питание от батареи.



Vbat - вход батареи для любого стандартного 3 В литиевого элемента или другого источника энергии. Для нормальной работы напряжение батареи должно поддерживаться между 2.5 и 3.5 В. Уровень, при котором запрещён доступ к часам реального времени и пользовательскому ОЗУ, установлен внутренней схемой равным 1.25 x Vbat. Литиевая батарея ёмкостью 35 mAh или больше достаточна для питания DS1307 в течение более чем 10 лет при отсутствии питания.

SCL (Последовательный Тактовый Вход) - SCL используется, чтобы синхронизировать передачу данных через последовательный интерфейс.

SDA (Вход/Выход Последовательных Данных) - SDA - вход / выход данных для 2-проводного последовательного интерфейса. Это выход с открытым стоком, который требует внешнего притягивающего резистора.

SQW/OUT (Меандр / Выходной Драйвер) - Когда бит SQWE установлен в 1, на выходе SQW/OUT вырабатываются импульсы в форме меандра одной из четырех частот: 1 Гц, 4 кГц, 8 кГц, 32 кГц. Вывод SQW/OUT - с открытым стоком, требует внешнего притягивающего резистора.

X1, X2 - выводы для подключения стандартного кристалла кварца 32.768 кГц. Внутренняя схема генератора рассчитана на работу с кристаллом, имеющим номинальную емкость (CL) 12.5 пФ.

GND – Земля.

VCC – питание 5 вольт.

DS1307 работает как ведомое устройство на последовательной шине. Для доступа к нему надо установить состояние **START** и передать код идентификации устройства, сопровождаемый адресом регистра. К последующим регистрам можно обращаться последовательно, пока не установлено состояние **STOP**. Когда **VCC** падает ниже 1.25 x Vbat, устройство прекращает связь и сбрасывает адресный счетчик. В это время оно не будет реагировать на входные сигналы, чтобы предотвратить запись ошибочной информации. Когда **VCC** падает ниже Vbat, устройство переключается в режим хранения с низким потреблением. При включении питания устройство переключает питание с батареи на **VCC**, когда напряжение питания превысит Vbat + 0.2V, и реагирует на входные сигналы, когда **VCC** станет более 1.25 x Vbat. Когда питание находится в пределах нормы, устройство полностью доступно, и данные могут быть записаны и считаны. Когда к устройству подключена трёхвольтовая батарея и VCC ниже 1.25 x Vbat, чтение и запись запрещены. Однако отсчёт времени при этом работает. Когда **VCC** падает ниже Vbat, питание ОЗУ и отсчёта времени переключается на внешнюю батарею 3 В.

Информацию о времени и дате получают, считывая соответствующие регистры. Регистры часов показаны в таблице ниже. Время и календарь устанавливаются или инициализируются путём записи байтов в соответствующие регистры. Содержание регистров времени и календаря хранится в двоично-десятичном **(BCD)** формате, поэтому перед выводом информации на LCD дисплей или семисегментный индикатор необходимо преобразовать двоично-десятичный код в двоичный или ANSII - код.

Бит 7 регистра 0 - это бит остановки хода часов (Clock Halt). Когда этот бит установлен в 1, генератор остановлен. Когда сброшен в ноль, генератор работает, а часы считают время.

АДРЕСА	БИТЫ							
	7	6	5	4	3	2	1	0
0x00	CLOCK HALT	ДЕСЯТКИ СЕКУНД			СЕКУНДЫ			
0x01	—	ДЕСЯТКИ МИНУТ			МИНУТЫ			
0x02	—	24/12	AM/PM ДЕСЯТКИ ЧАСОВ	ДЕСЯТКИ ЧАСОВ	ЧАСЫ			
0x03	—	—	—	—	—	ДЕНЬ НЕДЕЛИ		
0x04	—	—	ДЕСЯТКИ ДАТЫ		ЕДИНИЦЫ ДАТЫ			
0x05	—	—	—	Десяток месяца	ЕДИНИЦЫ МЕСЯЦА			
0x06	—	ДЕСЯТКИ ЛЕТ			ГОД			
0x07	Output	—	—	SQWE	—	—	RS1	RS0

DS1307 может работать в 12-часовом или 24-часовом режиме. Бит 6 регистра часов задаёт один из этих режимов. Когда он равен 1, установлен 12-часовой режим. В 12-часовом режиме высокий уровень бита 5 сообщает о послеполуденном времени. В 24-часовом режиме бит 5 - второй бит 10 часов (20-23 часа).

Регистр управления DS1307 предназначен для управления работой вывода **SQW/OUT**. Бит **OUT** - управление выходом. Этот бит управляет выходным уровнем на выводе **SQW/OUT**, когда генерация меандра запрещена. Если SQWE = 0, логический уровень на выводе **SQW/OUT** равен 1, если **OUT** = 1, и 0 - если **OUT** = 0. **SQWE** - Разрешение меандра. Когда этот бит установлен в 1, разрешается генерация меандра. Частота меандра зависит от значений битов RS0 и RS1. Эти биты управляют частотой меандра, когда его генерация разрешена. В таблице ниже показаны частоты, которые могут быть заданы RS битами.

RS1	RS0	Частота
0	0	1 Гц
0	1	4.096 кГц
1	0	8.192 кГц
1	1	32.768 кГц

DS1307 поддерживает двунаправленные 2-проводную шину и протокол передачи данных. Устройство, которое посылает данные на шину, называется передатчиком, а устройство, получающее данные - приемником. Устройство, которое управляет передачей, называется ведущим. Устройства, которые управляются ведущим - ведомые. Шина должна управляться ведущим устройством, которое вырабатывает последовательные такты (SCL), управляет доступом к шине, и генерирует состояния СТАРТ и СТОП. DS1307 работает как ведомое на 2-х проводной шине.

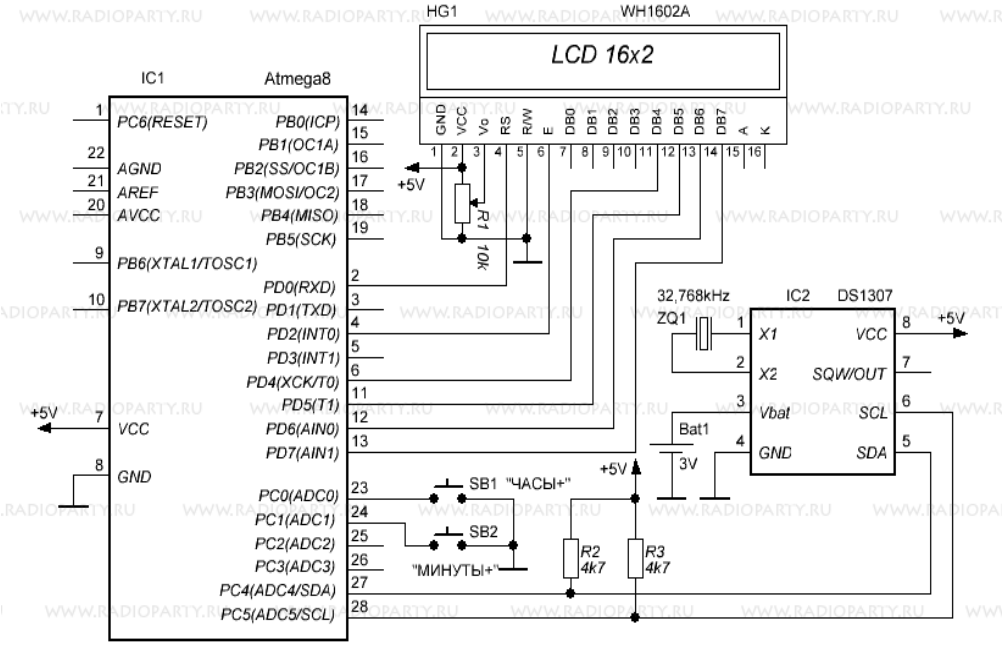
Для работы с DS1307 необходимо организовать функцию чтения из микросхемы и функцию записи.

1. **Режим записи в DS1307**. Последовательные данные и такты получены через SDA и SCL. После передачи каждого байта передаётся подтверждающий бит **ASK.** Состояния **START** и **STOP** опознаются как начало и конец последовательной передачи. Распознавание адреса выполняется аппаратно после приема адреса ведомого и бита направления. Байт адреса содержит семибитный адрес DS1307, равный 1101000, сопровождаемым битом направления (R/W), который при записи равен 0. После получения и расшифровки байта адреса DS1307 выдаёт подтверждение **ASK** на линии SDA. После того, как DS1307 подтверждает адрес ведомого и бит записи, ведущий передает адрес регистра DS1307. Тем самым будет установлен указатель регистра в

DS1307. Тогда ведущий начнет передавать байты данных в DS1307, который будет подтверждать каждый полученный байт. По окончании записи ведущий сформирует состояние **STOP.**

2. **Режим чтения из DS1307**. Первый байт принимается и обрабатывается как в режиме ведомого приёмника. Однако в этом режиме бит направления укажет, что направление передачи изменено. Последовательные данные передаются по SDA от DS1307, в то время как последовательные такты - по SCL в DS1307. Состояния **START** и **STOP** опознаются как начало и конец последовательной передачи. Байт адреса - первый байт, полученный после того, как ведущим сформировано состояние **START**. Байт адреса содержит семибитный адрес DS1307, равный 1101000, сопровождаемым битом направления (R/W), который при чтении равен 1. После получения и расшифровки байта адреса DS1307 выдаёт подтверждение **ASK** на линии SDA. Тогда DS1307 начинает передавать данные, начинающиеся с адреса регистра, на которые указывает указатель регистра. Если указатель регистра не записан перед инициированием режима чтения, то первый адрес, который читается - это последний адрес, оставшийся в указателе регистра. DS1307 должен получить неподтверждение **NOASK**, чтобы закончить чтение.

Рассмотрим особенности работы с DS1307 на примере простых часов, которые будут показывать часы, минуты и секунды. Данные будут выводиться на LCD дисплей 16x2. Две кнопки "Часы+" и "Минуты+" позволят подвести нужное время. Микроконтроллер Atmega 8 тактируется от внутреннего генератора частотой 1 MHz, поэтому не забудьте поменять фьюзы. Ниже представлена схема подключения.



Управляющая программа включает в себя наборы функций работы с шиной TWI, часами DS1307, LCD дисплеем:

I2CInit - инициализация шины;

I2CStart - передача условия START;

I2CStop - передача условия STOP;

I2CWriteByte - запись данных;

I2CReadByte - чтение данных;

DS1307Read - функция чтения данных из DS1307;

DS1307Write - Функция записи данных в DS1307;

lcd_com - передача команды в LCD;

lcd_data - передача данных в LCD;

lcd_string - функция вывода строки в LCD;

lcd_num_to_str - функция вывода символа типа int;

lcd_init - инициализация LCD.

Код программы:

```
#include <avr/io.h>
#include <util/delay.h>

// Функция инициализация шины TWI
void I2CInit(void)
{
TWBR = 2; // Настройка частоты шины
TWSR = (1 << TWPS1)|(1 << TWPS0); // Предделитель на 64
TWCR |= (1 << TWEN); // Включение модуля TWI
}

// Функция СТАРТ
void I2CStart(void)
{
TWCR = (1 << TWINT)|(1 << TWEN)|(1 << TWSTA); // Передача условия СТАРТ
while(!(TWCR & (1 << TWINT))); // Ожидание установки флага TWINT
}

// Функция СТОП
```

```
void I2CStop(void)
{
TWCR = (1 << TWINT)|(1 << TWEN)|(1 << TWSTO); // Передача условия СТОП
while(TWCR & (1 << TWSTO)); // Ожидание завершения передачи условия СТОП
}

// Функция записи данных по шине
uint8_t I2CWriteByte(uint8_t data)
{
TWDR = data; // Загрузка данных в TWDR
TWCR = (1 << TWEN)|(1 << TWINT); // Сброс флага TWINT для начала передачи данных
while(!(TWCR & (1 << TWINT))); // Ожидание установки флага TWINT
// Проверка статуса
// Если адрес DS1307+R и принято "подтверждение"(0x18)
// или адрес DS1307+W и принято "подтверждение"(0x40)
// или передается байт данных и принято "подтверждение"(0x28)
if((TWSR & 0xF8) == 0x18 || (TWSR & 0xF8) == 0x40 || (TWSR & 0xF8) == 0x28) return 1; // OK
else return 0; // ОШИБКА
}

// Функция чтения данных по шине
uint8_t I2CReadByte(uint8_t *data,uint8_t ack)
{
// Возвращаем "подтверждение" после приема
if(ack) TWCR |= (1 << TWEA);
// Возвращаем "неподтверждение" после приема
// Ведомое устройство не получает больше данных
// обычно используется для распознования последнего байта
else TWCR &= ~(1 << TWEA);
// Разрешение приема данных после сброса TWINT
TWCR |= (1 << TWINT);
while(!(TWCR & (1 << TWINT))); // Ожидание установки флага TWINT
// Проверка статуса
// Если принят байт данных и возвращается "подтверждение"(0x50)
// или принят байт данных и возвращается "ненеподтверждение"(0x58)
if((TWSR & 0xF8) == 0x50 || (TWSR & 0xF8) == 0x58)
{
*data = TWDR; // Читаем данные из TWDR
return 1; // OK
}
else return 0; // ОШИБКА
```

```
}

// Функция чтения данных из DS1307
uint8_t DS1307Read(uint8_t address,uint8_t *data)
{
uint8_t res;
I2CStart(); // СТАРТ
res = I2CWriteByte(0b11010000); // адрес DS1307+W
if(!res)    return 0; // ОШИБКА
// Передача адреса необходимого регистра
res = I2CWriteByte(address);
if(!res)    return 0; // ОШИБКА
I2CStart(); // Повторный СТАРТ
res = I2CWriteByte(0b11010001); // адрес DS1307+R
if(!res)    return 0; // ОШИБКА
// Чтение данных с "неподтверждением"
res = I2CReadByte(data,0);
if(!res)    return 0; // ОШИБКА
I2CStop(); // СТОП
return 1; // OK
}

// Функция записи данных в DS1307
uint8_t DS1307Write(uint8_t address,uint8_t data)
{
uint8_t res;
I2CStart(); // СТАРТ
res = I2CWriteByte(0b11010000); // адрес DS1307+W
if(!res)    return 0; // ОШИБКА
// Передача адреса необходимого регистра
res = I2CWriteByte(address);
if(!res)    return 0; // ОШИБКА
res = I2CWriteByte(data); // Запись данных
if(!res)    return 0; // ОШИБКА
I2CStop(); // СТОП
return 1; // OK
}

// Функции работы с LCD
#define RS PD0
#define EN PD2
// Функция передачи команды
void lcd_com(unsigned char p)
{
PORTD &= ~(1 << RS); // RS = 0 (запись команд)
```

```
PORTD |= (1 << EN); // EN = 1 (начало записи команды в LCD)
PORTD &= 0x0F; PORTD |= (p & 0xF0); // старший нибл
_delay_us(100);
PORTD &= ~(1 << EN); // EN = 0 (конец записи команды в LCD)
_delay_us(100);
PORTD |= (1 << EN); // EN = 1 (начало записи команды в LCD)
PORTD &= 0x0F; PORTD |= (p << 4); // младший нибл
_delay_us(100);
PORTD &= ~(1 << EN); // EN = 0 (конец записи команды в LCD)
_delay_us(100);
}
// Функция передачи данных
void lcd_data(unsigned char p)
{
PORTD |= (1 << RS)|(1 << EN); // RS = 1 (запись данных), EN - 1 (начало записи команды в LCD)
PORTD &= 0x0F; PORTD |= (p & 0xF0); // старший нибл
_delay_us(100);
PORTD &= ~(1 << EN); // EN = 0 (конец записи команды в LCD)
_delay_us(100);
PORTD |= (1 << EN); // EN = 1 (начало записи команды в LCD)
PORTD &= 0x0F; PORTD |= (p << 4); // младший нибл
_delay_us(100);
PORTD &= ~(1 << EN); // EN = 0 (конец записи команды в LCD)
_delay_us(100);
}

// Функция вывода строки на LCD
void lcd_string(unsigned char command, char *string)
{
lcd_com(0x0C);
lcd_com(command);
while(*string != '\0')
{  lcd_data(*string);
string++;
}
}

// Функция вывода переменной
void lcd_num_to_str(unsigned int value, unsigned char nDigit)
{
 switch(nDigit)
 {
  case 4: lcd_data((value/1000)+'0');
  case 3: lcd_data(((value/100)%10)+'0');
```

```
  case 2: lcd_data(((value/10)%10)+'0');
  case 1: lcd_data((value%10)+'0');
 }
}

// Функция инициализации LCD
void lcd_init(void)
{
PORTD = 0x00;
DDRD = 0xFF;

_delay_ms(50); // Ожидание готовности ЖК-модуля

// Конфигурирование четырехразрядного режима
PORTD |= (1 << PD5);
PORTD &= ~(1 << PD4);

// Активизация четырехразрядного режима
PORTD |= (1 << EN);
PORTD &= ~(1 << EN);
_delay_ms(5);

lcd_com(0x28); // шина 4 бит, LCD - 2 строки
lcd_com(0x08); // полное выключение дисплея
lcd_com(0x01); // очистка дисплея
_delay_us(100);
lcd_com(0x06); // сдвиг курсора вправо
lcd_com(0x0C); // включение дисплея, курсор не видим
}

int main(void)
{
_delay_ms(100);

DDRC = 0x00;
PORTC = 0xFF;

lcd_init(); // Инициализация LCD
I2CInit(); // Инициализация шины I2C

lcd_string(0x81, "«асГ Sa DS1307"); // Часы на DS1307
lcd_string(0xC4, "  :  :  ");

// Запускаем ход часов
uint8_t temp;
```

```
DS1307Read(0x00,&temp);
temp &= ~(1 << 7); // обнуляем 7 бит
DS1307Write(0x00,temp);

while(1)
{
unsigned char hour, minute, second, temp;
// Читаем данные и преобразуем из BCD в двоичную систему
DS1307Read(0x00,&temp); // Чтение регистра секунд
second = (((temp & 0xF0) >> 4)*10)+(temp & 0x0F);
DS1307Read(0x01,&temp); // Чтение регистра минут
minute = (((temp & 0xF0) >> 4)*10)+(temp & 0x0F);
DS1307Read(0x02,&temp); // Чтение регистра часов
hour = (((temp & 0xF0) >> 4)*10)+(temp & 0x0F);

lcd_com(0xC4);
lcd_num_to_str(hour, 2); // Выводим на экран часы
lcd_com(0xC7);
lcd_num_to_str(minute, 2); // Выводим на экран минуты
lcd_com(0xCA);
lcd_num_to_str(second, 2); // Выводим на экран секунды

if((PINC & (1 << PC0)) == 0) // Если нажата кнопка
{
while((PINC & (1 << PC0)) == 0){} // Ждем отпускания кнопки
hour++; // Увеличиваем часы на 1
if(hour > 23) hour = 0;
// Преобразуем из двоичной системы в BCD и записываем в DS1307
uint8_t temp;
temp = ((hour/10) << 4)|(hour%10);
DS1307Write(0x02, temp);
_delay_ms(100);
}

if((PINC & (1 << PC1)) == 0) // Если нажата кнопка
{
while((PINC & (1 << PC1)) == 0){} // Ждем отпускания кнопки
minute++; // Увеличиваем минуты на 1
if(minute > 59) minute = 0;
// Преобразуем из двоичной системы в BCD и записываем в DS1307
uint8_t temp;
temp = ((minute/10) << 4)|(minute%10);
DS1307Write(0x01, temp);
_delay_ms(100);
}
```

```
}
}
```

Установка fuse-битов микроконтроллера

This table allows reviewing and direct editing of the AVR fuse bits. All changes will be applied instantly.
Note: ☐ means unprogrammed (1); ☑ means programmed (0).

Bit	Low	High
7	☐ **BODLEVEL** Brown out detector trigger level	☐ **RSTDISBL** Disable reset
6	☐ **BODEN** Brown out detector enable	☐ **WTDON** Enable watchdog
5	☐ **SUT1** Select start-up time	☑ **SPIEN** Enable Serial programming and Data Downloading
4	☑ **SUT0** Select start-up time	☐ **CKOPT** Oscillator Options
3	☑ **CKSEL3** Select Clock Source	☐ **EESAVE** EEPROM memory is preserved through chip erase
2	☑ **CKSEL2** Select Clock Source	☑ **BOOTSZ1** Select Boot Size
1	☑ **CKSEL1** Select Clock Source	☑ **BOOTSZ0** Select Boot Size
0	☐ **CKSEL0** Select Clock Source	☐ **BOOTRST** Select Reset Vector

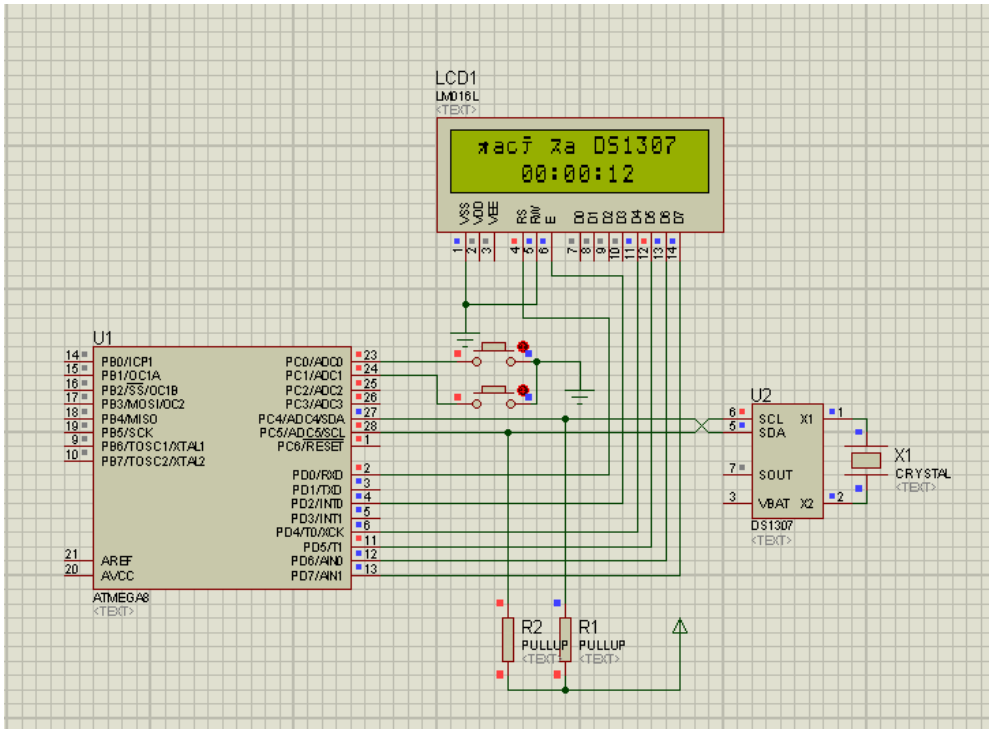
Apply manual fuse bit settings

Current settings

These fields show the actual hexadecimal representation of the fuse settings from above. These are the values you have to program into your AVR device. Optionally, you may fill in the numerical values yourself to preset the configuration to these values. Changes in the value fields are applied instantly (taking away the focus)!

Low	High	Action	*AVRDUDE arguments*
0x E1	0x D9	Apply values Defaults Apply manual changes to the values on the left side, or load factory default values for the selected device.	`-U lfuse:w:0xe1:m -U hfuse:w:0xd9:m` Select (try triple-click) and copy-and-paste this option string into your avrdude command line. You may specify multiple -U arguments within one call of avrdude.

Пример реализации в среде Proteus.



Задания для выполнения:

1. Изучить работу микросхемы часов реального времени.
2. Составить конспект по микросхеме DS1307.
3. Изучить программный код для управления данной микросхемой.
4. Реализовать в среде Proteus представленный код.

Результаты работы отправить на e-mail: rasov@rambler.ru с темой DS1307_ФИО