Свежие комментарии

- Семен к записи AVR Урок 15. Внутренняя энергонезависимая память EEPROM. Часть 4
- Стас к записи STM Урок 64. HAL. LTDC. Часть 2
- Семен к записи AVR Урок 15. Внутренняя энергонезависимая память EEPROM. Часть 4
- Семен к записи AVR Урок 15. Внутренняя энергонезависимая память EEPROM. Часть 4
- Boba к записи AVR Урок 16. Интерфейс TWI (I2C). Часть 2

Форум. Последние ответы

- alexander в Программирование МК STM32
  5 дн., 19 час. назад
- Narod Stream в Программирование МК STM32
  4 нед., 1 день назад
- Zandy в Программирование МК STM32
  4 нед., 1 день назад
- Narod Stream в Программирование МК STM32
  1 месяц, 1 неделя назад
- Narod Stream в Программирование МК STM32
  1 месяц, 1 неделя назад

Февраль 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				

« Янв

Архивы

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

AVR УРОК 39. Акселерометр LSM6DS3. Часть 2

Posted on Январь 16, 2017 by Narod Stream Опубликовано в Программирование AVR — Нет комментариев ↓

Мета

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

искать здесь …

Фильтровать

Урок 39
Часть 2

Акселерометр LSM6DS3

Продолжаем работать с подключением датчика.

В **прошлой части занятия** мы подключили плату с датчиком к отладочной плате с контроллером, а также подключили переходник USART для мониторинга показаний датчика, также создали проект, инициализировали USART и написали код для проверки данного интерфейса, но пока его не проверили.

Поэтому теперь мы соберём проект, прошьём контроллер, перед этим не забыв нажать кнопку **Connect** в терминальной программе. Должно после прошивки получиться вот так

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

Если оно так, то мы на правильном пути и можно двигаться дальше.
Теперь продолжаем работать непосредственно с шиной **I2C**. Это мы тоже умеем. Чтобы воспользоваться данной шиной, из проекта, например по работе с часовой микросхемой **MyClock1307LED**, скопируем и подключим в наш проект файлы **twi.h** и **twi.c**. Также подключим данную библиотеку в файле main.h

```
#include "usart.h"
#include "twi.h"
```

Работа с шиной I2C на контроллере **Atmega328** вообще ничем не отличается от работы с данной шиной на контроллере **Atmega8**, поэтому проект у нас должен будет нормально собраться.
Вызовем инициализацию I2C в main()

```
I2C_Init();//инициализируем TWI
USART_Init (16); //115200
```

Также есть ещё одна тонкость. Мы ещё с двух ножек I2C должны подключить подтягивающие резисторы на шину питания, но я этого делать не стал и подключил их программно. Вы можете поступить по дргугому и припаять их физически. Также в ваших модулях с датчиком они могут уже быть. Вообщем, напишем функцию инициализации порта и вызовем её в main(). Заодно включим на выход ножку, к которой подключен красный светодиод, чтобы с помощью него следить за ошибками

```
//-----------------
void port_ini(void)
{
  DDRC&=~0b00110000;
  PORTC|=0b00110000;
  DDRB|=(1<<PORTB5);
}
//-----------------
int main(void)
{
  port_ini();
```

Также для нашей шины I2C мы должны расчитать скорость. Её мы оставим 100 кГц, но кварцевый резонатор у нас 16 МГц, и воспользовавшись таблицей расчета TWBR, мы вычислим его значения и внесем изменения в функции в файле twi.c

```
void I2C_Init (void)
{
  TWBR=0x48;//скорость передачи (при
16 мгц получается 100 кгц)
}
```

Создадим и подключим ещё файлы будущей библиотеки для работы с датчиком **lsm6ds3.h** и **lsm6ds3.c**, а также подключим lsm6ds3.h в main.h

```
#include "twi.h"
#include "lsm6ds3.h"
```

Также подключим его и в файле **lsm6ds3.c**

Рубрики

- 1-WIRE (3)
- ADC (6)
- DAC (4)
- GPIO (26)
- I2C (19)
- SPI (13)
- USART (8)
- Программирование AVR (131)
- Программирование PIC (8)
- Программирование STM32 (217)
- Тесты устройств и аксессуаров (1)

```
#include "lsm6ds3.h"
//————————————
```

Подключим в файле **lsm6ds3.h** все наши интерфейсы, макросы для ножки светодиода, а также из одноименного файла из проекта для контроллера stm скопируем макросы для регистров акселерометра

```
#ifndef LSM6DS3_H_
#define LSM6DS3_H_
//————————————-
#include "main.h"
#include "usart.h"
#include "twi.h"
//————————————-
#define LD_ON PORTB|=(1<<PORTB5)
#define LD_OFF PORTB&=~(1<<PORTB5)
//————————————
#define LSM6DS3_ACC_GYRO_CTRL1_XL
0X10
#define LSM6DS3_ACC_GYRO_CTRL3_C
0X12
#define LSM6DS3_ACC_GYRO_FIFO_CTRL5
0X0A
#define LSM6DS3_ACC_GYRO_CTRL9_XL
0X18
//————————————
#define
LSM6DS3_ACC_GYRO_IF_INC_DISABLED
0x00
#define
LSM6DS3_ACC_GYRO_IF_INC_ENABLED 0x04
#define LSM6DS3_ACC_GYRO_IF_INC_MASK
0x04
//————————————
#define
LSM6DS3_ACC_GYRO_BDU_CONTINUOS 0x00
#define
LSM6DS3_ACC_GYRO_BDU_BLOCK_UPDATE
0x40
#define LSM6DS3_ACC_GYRO_BDU_MASK
0x40
//————————————
#define
LSM6DS3_ACC_GYRO_FIFO_MODE_BYPASS
0x00
#define
LSM6DS3_ACC_GYRO_FIFO_MODE_FIFO 0x01
#define
LSM6DS3_ACC_GYRO_FIFO_MODE_STREAM
0x02
#define
LSM6DS3_ACC_GYRO_FIFO_MODE_STF 0x03
#define
LSM6DS3_ACC_GYRO_FIFO_MODE_BTS 0x04
#define
LSM6DS3_ACC_GYRO_FIFO_MODE_DYN_STREA
M 0x05
#define
LSM6DS3_ACC_GYRO_FIFO_MODE_DYN_STREA
M_2 0x06
#define
LSM6DS3_ACC_GYRO_FIFO_MODE_BTF 0x07
#define
LSM6DS3_ACC_GYRO_FIFO_MODE_MASK 0x07
//————————————
#define
LSM6DS3_ACC_GYRO_ODR_XL_POWER_DOWN
0x00
#define LSM6DS3_ACC_GYRO_ODR_XL_13Hz
0x10
```

```
#define LSM6DS3_ACC_GYRO_ODR_XL_26Hz
0x20
#define LSM6DS3_ACC_GYRO_ODR_XL_52Hz
0x30
#define
LSM6DS3_ACC_GYRO_ODR_XL_104Hz 0x40
#define
LSM6DS3_ACC_GYRO_ODR_XL_208Hz 0x50
#define
LSM6DS3_ACC_GYRO_ODR_XL_416Hz 0x60
#define
LSM6DS3_ACC_GYRO_ODR_XL_833Hz 0x70
#define
LSM6DS3_ACC_GYRO_ODR_XL_1660Hz 0x80
#define
LSM6DS3_ACC_GYRO_ODR_XL_3330Hz 0x90
#define
LSM6DS3_ACC_GYRO_ODR_XL_6660Hz 0xA0
#define
LSM6DS3_ACC_GYRO_ODR_XL_13330Hz 0xB0
#define LSM6DS3_ACC_GYRO_ODR_XL_MASK
0xF0
//————————————————
#define LSM6DS3_ACC_GYRO_FS_XL_2g
0x00
#define LSM6DS3_ACC_GYRO_FS_XL_16g
0x04
#define LSM6DS3_ACC_GYRO_FS_XL_4g
0x08
#define LSM6DS3_ACC_GYRO_FS_XL_8g
0x0C
#define LSM6DS3_ACC_GYRO_FS_XL_MASK
0x0C
//————————————————
#define LSM6DS3_ACC_GYRO_XEN_XL_MASK
0x08
#define LSM6DS3_ACC_GYRO_YEN_XL_MASK
0x10
#define LSM6DS3_ACC_GYRO_ZEN_XL_MASK
0x20
#define
LSM6DS3_ACC_GYRO_XEN_XL_ENABLED 0x08
#define
LSM6DS3_ACC_GYRO_YEN_XL_ENABLED 0x10
#define
LSM6DS3_ACC_GYRO_ZEN_XL_ENABLED 0x20
//————————————————
#define LSM6DS3_ACC_GYRO_OUTX_L_XL
0X28
#define LSM6DS3_ACC_GYRO_OUTX_H_XL
0X29
#define LSM6DS3_ACC_GYRO_OUTY_L_XL
0X2A
#define LSM6DS3_ACC_GYRO_OUTY_H_XL
0X2B
#define LSM6DS3_ACC_GYRO_OUTZ_L_XL
0X2C
#define LSM6DS3_ACC_GYRO_OUTZ_H_XL
0X2D
//————————————————
#endif /* LSM6DS3_H_ */
```

В файле **lsm6ds3.c** начнем писать функцию инициализации датчика

```
#include "lsm6ds3.h"
//————————————————
void Accel_Ini(void)
{
}
```

Первым делом добавим задержку для того, чтобы перед инициализацией датчик

успел нормально включиться и прошел самотестирование

```
void Accel_Ini(void)
{
  _delay_ms(500);
```

Также все мы помним, чтобы инициализировать любой датчик и убедиться, что мы работаем именно с таким датчиком, нам необходимо считать его идентификатор

Создадим функцию для чтения идентификатора, расположим её выше функции инициализации, чтобы легче было вызывать без прототипов

```
//———————————
unsigned char Accel_ReadID(void)
{
}
//———————————
void Accel_Ini(void)
```


Теперь нам нужно будет написать функцию чтения данных вообще по шине I2C неопределённой длины из датчика. Также создадим пока каркас данной функции

```
//———————————
void I2Cx_ReadData(unsigned char
Addr, unsigned char Reg, unsigned
char sz, unsigned char* value)
{
}
//———————————
unsigned char Accel_ReadID(void)
```

Во входных аргументах данной функции у нас будет адрес устройства, адрес регистра, количество байтов, которые мы будем считывать и указатель на участок памяти, в которые будут укладываться эти байты. Для этого создадим глобальную переменную в данном файле

```
#include "lsm6ds3.h"
//———————————
unsigned char read_buf[10]={0};
```

Прекрасно зная, как именно читать байты из шины I2C и помня о том. что последний считанный байт мы не подтверждаем, а генерируем в этот момент высокое состояние шины, напишем тела функции приёма байтов из шины

```
void I2Cx_ReadData(unsigned char
Addr, unsigned char Reg, unsigned
char sz, unsigned char* value)
{
  unsigned char i=sz,n=0;
  I2C_StartCondition(); //отправим
условие START
  I2C_SendByte(Addr);//передаем
адрес устройства и бит записи (0)
```

```
  I2C_SendByte(Reg);//передаем адрес
регистра
  I2C_StartCondition(); //отправим
условие START
  I2C_SendByte(Addr|0x01);//передаем
адрес устройства и бит чтения (1)
  while(1)
  {
    if(i==1) break;
    value[n]=I2C_ReadByte();
    i–; n++;
  }
  value[sz-1] = I2C_ReadLastByte();
//прочитаем последний байт
  I2C_StopCondition(); //отправим
условие STOP
}
```

Уточним адрес устройства, а также регистр и идентификатор в технической документации на датчик

Table 11. SAD+Read/Write patterns

Command	SAD[6:1]	SAD[0] = SA0	R/W	SAD+R/W
Read	110101	0	1	11010101 (D5h)
Write	110101	0	0	11010100 (D4h)
Read	110101	1	1	11010111 (D7h)
Write	110101	1	0	11010110 (D6h)

9.11 WHO_AM_I (0Fh)

Who_AM_I register (r). This register is a read-only register. Its value is fixed at 69h.

Table 44. WHO_AM_I register

0	1	1	0	1	0	0	1

Воспользовавшись функцией чтения данных из шины, напишем тело функции для считывания идентификатора

```
unsigned char Accel_ReadID(void)
{
  unsigned char ctrl=0;

I2Cx_ReadData(0xD4,0x0F,1,read_buf);
  return ctrl;
}
```

Напишем в самом верху файла функцию срабатывания красного светодиода на ошибку

```
unsigned char read_buf[10]={0};
//————————————
void Error(void)
{
  LD_ON;
}
```

Теперь вызовем функцию чтения идентификатора в функции инициализации датчика

```
void Accel_Ini(void)
{
  _delay_ms(500);
  Accel_ReadID();
  if(read_buf[0]!=0x69) Error();
}
```

Добавим в заголовочный файл прототип функции инициализации

```
#define LSM6DS3_ACC_GYRO_OUTZ_H_XL
0X2D
//————————————-
void Accel_Ini(void);
//————————————-
```

Вызовем функцию инициализации датчика в main() и удалим код отправки тестовой строки в USART

```
USART_Init (16); //115200
Accel_Ini();
while (1)
```

Соберём код, прошьём контроллер, и, если у нас светодиод не загорится, значит мы считали правильный идетификатор и мы работаем и общаемся по шине I2C именно с тем датчиком.
Это очень хорошо. Дальнейшую инициализацию мы продолжим **в следующей части** нашего урока.

Техническая документация:
Документация на датчик
Документация на оценочную плату

Приобрести плату Atmega 328p Pro Mini можно **здесь**.
Программатор (продавец надёжный) **USBASP USBISP 2.0**
Приобрести платы с датчиком LSM6DS3 можно у следующих продавцов:
Надёжный продавец **LSM6DS33 STEVAL-MKI160V1**
Здесь дешевле **LSM6DS33 STEVAL-MKI160V1**
Здесь другая плата, намного дешевле, но от другого разработчика **LSM6DS33**

Смотреть ВИДЕОУРОК (нажмите на картинку)

‹ AVR УРОК 39. Акселерометр LSM6DS3. Часть 1
AVR УРОК 39. Акселерометр LSM6DS3. Часть 3 ›

Добавить комментарий

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

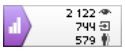
Имя *

E-mail *

Сайт

один + четыре =

Отправить комментарий



Наверх