Опубликовано Ноябрь 23, 2016 Автор: Narod Stream — Нет комментариев ↓

Урок 45

Часть 1

Подключаем гироскоп LSM6DS0

Сегодня мы ещё раз поработаем с датчиком, который в себе объединяет сразу два функционала – акселерометр и гироскоп – **LSM6DS0**. Выполнен он с использованием технологии MEMS. Установлен на плате расширения **X-NUCLEO-IKS01A1**, предназначенной для работы с отладочной платой Nucleo. Мы будем подключать данную оценочную плату к плате Nucleo STM32F401RE.

Данный акселерометр-гироскоп также наряду с интерфейсом I2C может подключиться и с использованием интерфейса SPI. Но мы будем использовать подключение именно по I2C, так как именно такое подключение имеет место в оценочной плате X-NUCLEO-IKS01A.

Также использовать в рамках данного занятия мы данный датчик будем только как гироскоп, так как в качестве акселерометра мы его уже подключали.

Гироскоп в данном датчике имеет следующие технические характеристики:

Диапазон показаний ±245/±500/±2000 dps;

Чувствительность 8.75 – 70 mdps/LSb ;

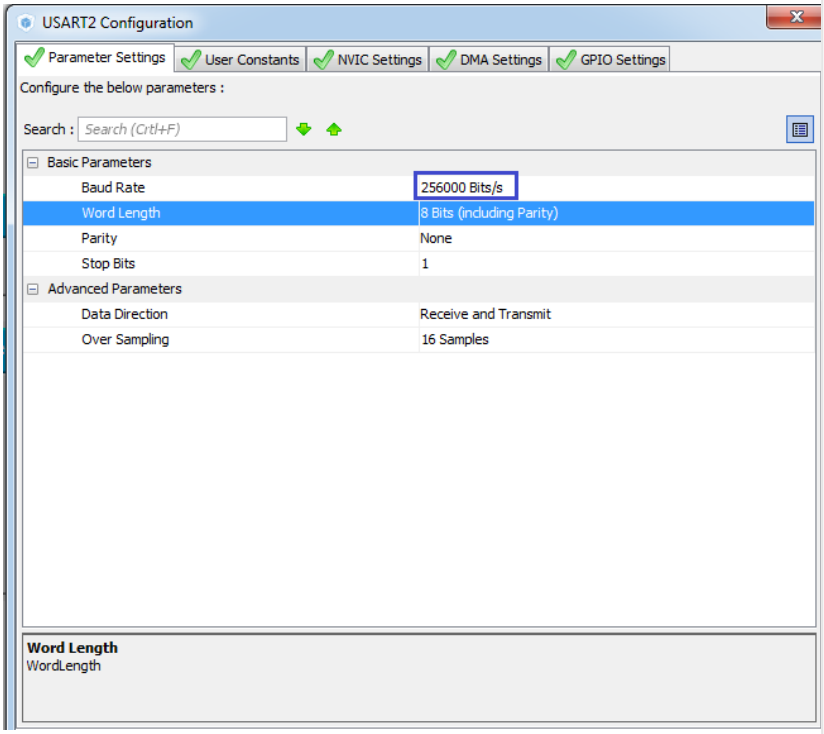
Отклонение от нуля ±30 dps при установке диапазона 2000 dps.

Частота измерений 14,9 – 952 Гц.

С некоторыми остальными показателями, регистрами, значениями и другими тонкостями гироскопа мы познакомимся в ходе его программирования.

Проект мы создадим из готового проекта, в котором мы работали с акселерометром данного датчика – из проекта Accel_LSM6DS0, только назовём мы данный проект теперь соответственно Gyro_LSM6DS0.

Запустим проект Cube MX. Изменим мы здесь только скорость USART.



Сгенерируем проект, откроем его. Настроим программатор на авторезет. Добавим файл lsm6ds0.c. Скомпилируем проект.

В бесконечном цикле пока закомментируем код вызова функции считывания данных и отправки их в USART

```
  /* USER CODE BEGIN 3 */
            //Accel_ReadAcc();
  }
```

Для универсальности проекта, так как, возможно, позже мы объединим работу с акселерометром и гироскопом в один проект, переименуем функцию Accel_Ini в файле lsm6ds0.c на Accel_Gyro_Ini. То же самое проделаем с прототипом и с вызовом данной функции в main().

Закомментируем в функции Accel_Gyro_Ini вот эту строку

```
        LD2_OFF;
//      AccInit(ctrl);
        LD2_ON;
```

Добавим функцию инициализации гироскопа по подобию функции инициализации акселерометра

```
//-----------------------------------
void GyroInit(uint16_t InitStruct)
{
	uint8_t value = 0;
}
//-----------------------------------
```

Вызовем её в функции общей инициализации

```
//	AccInit(ctrl);
	GyroInit(ctrl);
	LD2_ON;
```

В файле lsm6ds0.h заранее добавим несколько макросов, необходимых для работы с гироскопом. Код данного файла после всех изменений примет следующий вид:

```
#ifndef LIS3DSH_H_
#define LIS3DSH_H_

#include "stm32f4xx_hal.h"
#include <string.h>
//-----------------------------------
#define ABS(x)		(x < 0) ? (-x) : x
//-----------------------------------
#define LD2_Pin GPIO_PIN_5
#define LD2_GPIO_Port GPIOA
#define LD2_ON HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET) //GREEN
#define LD2_OFF HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET)
//-----------------------------------
#define LSM6DS0_ACC_GYRO_CTRL_REG5_XL	0X1F
#define LSM6DS0_ACC_GYRO_CTRL_REG6_XL	0X20
#define LSM6DS0_ACC_GYRO_CTRL_REG8	0X22
#define LSM6DS0_ACC_GYRO_CTRL_REG1_G	0X10
#define LSM6DS0_ACC_GYRO_CTRL_REG2_G	0X11
#define LSM6DS0_ACC_GYRO_CTRL_REG4	0X1E
//-----------------------------------
#define LSM6DS0_ACC_GYRO_BDU_DISABLE	0x00
#define LSM6DS0_ACC_GYRO_BDU_ENABLE	0x40
#define LSM6DS0_ACC_GYRO_BDU_MASK	0x40
//-----------------------------------
#define LSM6DS0_ACC_GYRO_ODR_XL_POWER_DOWN 0x00
#define LSM6DS0_ACC_GYRO_ODR_XL_10Hz 0x20
#define LSM6DS0_ACC_GYRO_ODR_XL_50Hz 0x40
#define LSM6DS0_ACC_GYRO_ODR_XL_119Hz 0x60
#define LSM6DS0_ACC_GYRO_ODR_XL_238Hz 0x80
#define LSM6DS0_ACC_GYRO_ODR_XL_476Hz 0xA0
#define LSM6DS0_ACC_GYRO_ODR_XL_952Hz 0xC0
#define LSM6DS0_ACC_GYRO_ODR_XL_MASK	0xE0
//-----------------------------------
#define	LSM6DS0_ACC_GYRO_ODR_G_POWER_DOWN	0x00
#define	LSM6DS0_ACC_GYRO_ODR_G_15Hz	0x20
#define	LSM6DS0_ACC_GYRO_ODR_G_60Hz	0x40
#define	LSM6DS0_ACC_GYRO_ODR_G_119Hz	0x60
#define	LSM6DS0_ACC_GYRO_ODR_G_238Hz	0x80
#define	LSM6DS0_ACC_GYRO_ODR_G_476Hz	0xA0
```

```
    #define       LSM6DS0_ACC_GYRO_ODR_G_952Hz        0x
C0
    #define       LSM6DS0_ACC_GYRO_ODR_G_MASK         0x
E0
    //————————————————————————
    #define LSM6DS0_ACC_GYRO_FS_XL_2g 0x00
    #define LSM6DS0_ACC_GYRO_FS_XL_16g 0x08
    #define LSM6DS0_ACC_GYRO_FS_XL_4g 0x10
    #define LSM6DS0_ACC_GYRO_FS_XL_8g 0x18
    #define LSM6DS0_ACC_GYRO_FS_XL_MASK   0x18
    //————————————————————————
    #define       LSM6DS0_ACC_GYRO_FS_G_245dps        0x00
    #define       LSM6DS0_ACC_GYRO_FS_G_500dps        0x08
    #define       LSM6DS0_ACC_GYRO_FS_G_1000dps        0x1
0
    #define       LSM6DS0_ACC_GYRO_FS_G_2000dps        0x1
8
    #define       LSM6DS0_ACC_GYRO_FS_G_MASK        0x18
    //————————————————————————
    #define       LSM6DS0_ACC_GYRO_OUT_SEL_BYPASS_H
PF_AND_LPF2        0x00
    #define       LSM6DS0_ACC_GYRO_OUT_SEL_BYPASS_L
PF2        0x01
    #define       LSM6DS0_ACC_GYRO_OUT_SEL_USE_HPF_
AND_LPF2        0x02
    #define       LSM6DS0_ACC_GYRO_OUT_SEL_MASK
0x03
    //————————————————————————
    #define       LSM6DS0_ACC_GYRO_BW_G_LOW        0x00
    #define       LSM6DS0_ACC_GYRO_BW_G_NORMAL
0x01
    #define       LSM6DS0_ACC_GYRO_BW_G_HIGH        0x02
    #define       LSM6DS0_ACC_GYRO_BW_G_ULTRA_HIGH
      0x03
    #define       LSM6DS0_ACC_GYRO_BW_G_MASK        0x0
3
    //————————————————————————
    #define LSM6DS0_ACC_GYRO_XEN_XL_ENABLE 0x08
    #define LSM6DS0_ACC_GYRO_YEN_XL_ENABLE 0x10
    #define LSM6DS0_ACC_GYRO_ZEN_XL_ENABLE 0x20
    #define LSM6DS0_ACC_GYRO_XEN_XL_MASK   0x08
    #define LSM6DS0_ACC_GYRO_YEN_XL_MASK   0x10
    #define LSM6DS0_ACC_GYRO_ZEN_XL_MASK   0x20
    //————————————————————————
    #define       LSM6DS0_ACC_GYRO_XEN_G_DISABLE
 0x00
    #define       LSM6DS0_ACC_GYRO_XEN_G_ENABLE
0x08
    #define       LSM6DS0_ACC_GYRO_YEN_G_DISABLE
 0x00
    #define       LSM6DS0_ACC_GYRO_YEN_G_ENABLE
0x10
    #define       LSM6DS0_ACC_GYRO_ZEN_G_DISABLE
0x00
    #define       LSM6DS0_ACC_GYRO_ZEN_G_ENABLE
0x20
    #define       LSM6DS0_ACC_GYRO_XEN_G_MASK        0x
08
    #define       LSM6DS0_ACC_GYRO_YEN_G_MASK        0x
10
    #define       LSM6DS0_ACC_GYRO_ZEN_G_MASK        0x
20
    //————————————————————————
    #define LSM6DS0_ACC_GYRO_OUT_X_L_XL   0X28
    #define LSM6DS0_ACC_GYRO_OUT_X_H_XL   0X29
    #define LSM6DS0_ACC_GYRO_OUT_Y_L_XL   0X2A
    #define LSM6DS0_ACC_GYRO_OUT_Y_H_XL   0X2B
```

```
#define LSM6DS0_ACC_GYRO_OUT_Z_L_XL   0X2C
#define LSM6DS0_ACC_GYRO_OUT_Z_H_XL   0X2D
//---------------------------------------
#define LSM6DS0_ACC_GYRO_OUT_X_L_G    0X18
#define LSM6DS0_ACC_GYRO_OUT_X_H_G    0X19
#define LSM6DS0_ACC_GYRO_OUT_Y_L_G    0X1A
#define LSM6DS0_ACC_GYRO_OUT_Y_H_G    0X1B
#define LSM6DS0_ACC_GYRO_OUT_Z_L_G    0X1C
#define LSM6DS0_ACC_GYRO_OUT_Z_H_G    0X1D
//---------------------------------------
void Accel_Gyro_Ini(void);
void AccelGyro_Read(void);
//---------------------------------------
#endif /* LIS3DSH_H_ */
```

Скопировав из другой функции, добавим в функцию GyroInit следующий код:

```
uint8_t value = 0;
 //установим бит BDU
value =
Accel_IO_Read(0xD6,LSM6DS0_ACC_GYRO_CTRL_REG8
);
value&=~LSM6DS0_ACC_GYRO_BDU_MASK;
value|=LSM6DS0_ACC_GYRO_BDU_ENABLE;
Accel_IO_Write(0xD6,LSM6DS0_ACC_GYRO_CTRL
_REG8,value);
```

Так как в коде ничего не изменилось, объяснение не требуется.

Теперь добавим следующий код сюда же:

```
Accel_IO_Write(0xD6,LSM6DS0_ACC_GYRO_CTRL_
REG8,value);
//пока выключим датчик (ODR_G = 000)
value =
Accel_IO_Read(0xD6,LSM6DS0_ACC_GYRO_CTRL_REG1
_G);
value&=~LSM6DS0_ACC_GYRO_ODR_G_MASK;
value|=LSM6DS0_ACC_GYRO_ODR_G_POWER_D
OWN;
Accel_IO_Write(0xD6,LSM6DS0_ACC_GYRO_CTRL
_REG1_G,value);
```


В данном коде мы работаем с регистром CTRL_REG1_G (адрес 0X10), с его битами, отвечающим за частоту снятия данных именно с гироскопа

7.11 CTRL_REG1_G (10h)

Angular rate sensor control register 1.

Table 40. CTRL_REG1_G register

ODR_G2	ODR_G1	ODR_G0	FS_G1	FS_G0	0(1)	BW_G1	BW_G0

1. This bit must be set to '0' for the correct operation of the device

Пока мы здесь отключаем датчик (все биты выставляем в 0).

Table 42. ODR and BW configuration setting (after LPF1)

ODR_G2	ODR_G1	ODR_G0	ODR [Hz]	Cutoff [Hz](1)
0	0	0	Power-down	n.a.

Дальше добавим следующий код

```
Accel_IO_Write(0xD6,LSM6DS0_ACC_GYRO_CTRL_
REG1_G,value);
//Full scale selection 500 dps
value =
Accel_IO_Read(0xD6,LSM6DS0_ACC_GYRO_CTRL_REG1
_G);
value&=~LSM6DS0_ACC_GYRO_FS_G_MASK;
value|=LSM6DS0_ACC_GYRO_FS_G_500dps;
Accel_IO_Write(0xD6,LSM6DS0_ACC_GYRO_CTRL
_REG1_G,value);
```

Дальше мы работаем с тем же регистром, только с другими битами, отвечающими за максимально измеряемую датчиком угловую скорость относительно оси. Думаю, нам достаточно 500 градусов в секунду, быстрее мы не разгонимся.

Table 41. CTRL_REG1_G register description

ODR_G [2:0]	Gyroscope output data rate selection. Default value: 000 (Refer to *Table 42* and *Table 43*)
FS_G [1:0]	Gyroscope full-scale selection. Default value: 00 (00: 245 dps; 01: 500 dps; 10: not available; 11: 2000 dps)
BW_G [1:0]	Gyroscope bandwidth selection. Default value: 00

Пишем код дальше

```
Accel_IO_Write(0xD6,LSM6DS0_ACC_GYRO_CTRL_
REG1_G,value);
//Включим оси
value =
Accel_IO_Read(0xD6,LSM6DS0_ACC_GYRO_CTRL_REG4
);
value&=~(LSM6DS0_ACC_GYRO_XEN_G_MASK|\
LSM6DS0_ACC_GYRO_YEN_G_
MASK|\
LSM6DS0_ACC_GYRO_ZEN_G_
MASK);
value|=(LSM6DS0_ACC_GYRO_XEN_G_ENABLE|\
LSM6DS0_ACC_GYRO_YEN_G_
ENABLE|\
LSM6DS0_ACC_GYRO_ZEN_G_E
NABLE);
Accel_IO_Write(0xD6,LSM6DS0_ACC_GYRO_CTRL
_REG4,value);
```

Как и указано в комментарии, здесь мы включаем оси гироскопа. Включим мы все 3 оси

7.21 CTRL_REG4 (1Eh)

Control register 4.

Table 58. CTRL_REG4 register

0(1)	0(1)	Zen_G	Yen_G	Xen_G	0(1)	LIR_XL1	4D_XL1

1. These bits must be set to '0' for the correct operation of the device

Table 59. CTRL_REG4 register description

Zen_G	Gyroscope's yaw axis (Z) output enable. Default value: 1 (0: Z-axis output disabled; 1: Z-axis output enabled)
Yen_G	Gyroscope's roll axis (Y) output enable. Default value: 1 (0: Y-axis output disabled; 1: Y-axis output enabled)
Xen_G	Gyroscope's pitch axis (X) output enable. Default value: 1 (0: X-axis output disabled; 1: X-axis output enabled)
LIR_XL1	Latched interrupt. Default value: 0 (0: interrupt request not latched; 1: interrupt request latched)
4D_XL1	4D option enabled on interrupt. Default value: 0 (0: interrupt generator uses 6D for position recognition; 1: interrupt generator uses 4D for position recognition)

Продолжаем писать исходный код

```
Accel_IO_Write(0xD6,LSM6DS0_ACC_GYRO_CTRL_REG4,value);
//Включим HPF и LPF2 (фильтрация)
value = Accel_IO_Read(0xD6,LSM6DS0_ACC_GYRO_CTRL_REG2_G);
value&=~LSM6DS0_ACC_GYRO_OUT_SEL_MASK;
value|=LSM6DS0_ACC_GYRO_OUT_SEL_USE_HPF_AND_LPF2;
Accel_IO_Write(0xD6,LSM6DS0_ACC_GYRO_CTRL_REG2_G,value);
```

В регистре 2 (адрес 0x11) включим бит OUT_SEL1, тем самым включив все фильтры

7.12 CTRL_REG2_G (11h)

Angular rate sensor control register 2.

Table 44. CTRL_REG2_G register

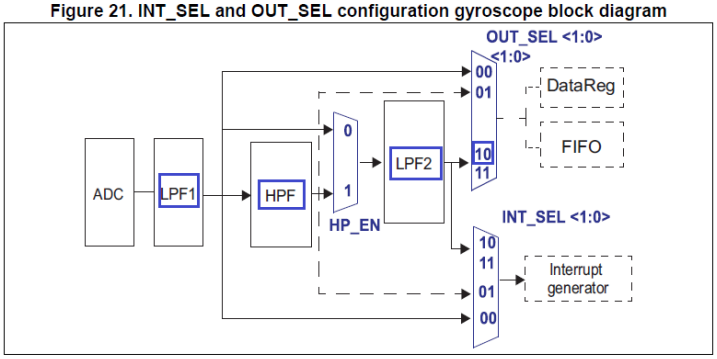
0(1)	0(1)	0(1)	0(1)	INT_SEL1	INT_SEL0	OUT_SEL1	OUT_SEL0

1. These bits must be set to '0' for the correct operation of the device

Table 45. CTRL_REG2_G register description

INT_SEL [1:0]	INT selection configuration. Default value: 00 (Refer to *Figure 21*)
OUT_SEL [1:0]	Out selection configuration. Default value: 00 (Refer to *Figure 21*)

Figure 21. INT_SEL and OUT_SEL configuration gyroscope block diagram



Дальше включим следующее:

Accel_IO_Write(0xD6,LSM6DS0_ACC_GYRO_CTRL_REG2_G,value);
//Включим BW (пропускная способность). При 952 Гц настроек будет пропускная способность 100 Гц
value = Accel_IO_Read(0xD6,LSM6DS0_ACC_GYRO_CTRL_REG1_G);
value&=~LSM6DS0_ACC_GYRO_BW_G_MASK;
value|=LSM6DS0_ACC_GYRO_BW_G_ULTRA_HIGH;
Accel_IO_Write(0xD6,LSM6DS0_ACC_GYRO_CTRL_REG1_G,value);

Здесь мы в регистре 1 включили оба бита BW_G, тем самым задав такую пропускную способность, которая позволит всем фильтрам успеть сработать.

7.11 CTRL_REG1_G (10h)

Angular rate sensor control register 1.

Table 40. CTRL_REG1_G register

ODR_G2	ODR_G1	ODR_G0	FS_G1	FS_G0	0(1)	BW_G1	BW_G0

1. This bit must be set to '0' for the correct operation of the device

Table 41. CTRL_REG1_G register description

ODR_G [2:0]	Gyroscope output data rate selection. Default value: 000 (Refer to *Table 42* and *Table 43*)
FS_G [1:0]	Gyroscope full-scale selection. Default value: 00 (00: 245 dps; 01: 500 dps; 10: not available; 11: 2000 dps)
BW_G [1:0]	Gyroscope bandwidth selection. Default value: 00

Accel_IO_Write(0xD6,LSM6DS0_ACC_GYRO_CTRL_REG4,value);
//Включим Data Rate 952 Гц
value = Accel_IO_Read(0xD6,LSM6DS0_ACC_GYRO_CTRL_REG1_G);
value&=~LSM6DS0_ACC_GYRO_ODR_G_MASK;
value|=LSM6DS0_ACC_GYRO_ODR_G_952Hz;
Accel_IO_Write(0xD6,LSM6DS0_ACC_GYRO_CTRL_REG1_G,value);

В данном коде мы включим частоту опроса данных с осей 952 Гц. Реальная частота при использовании всех возможных фильтров и пропускной способности получится 100 Гц.

Table 43. ODR and BW configuration setting (after LPF2)

ODR_G [2:0]	BW_G [1:0]	ODR [Hz]	Cutoff [Hz](1)
110	11	952	100

На этом Инициализацию можно считать завершенной.
Соберем код, прошьем контроллер и убедимся, что зеленый светодиод у нас светится.
В **следующей части** нашего занятия мы напишем функции, отвечающие за снятие данных с датчика и, отображая их в различных программах визуализации, увидим на деле работу гироскопа.

Предыдущий урок
Программирование МК STM32
Следующая часть

Техническая документация на датчик
Техническая документация на плату расширения
Программа Hyper Terminal
Программа NS Port Monitor
Программа NS Port Visual

Отладочную плату можно приобрести здесь **Nucleo STM32F401RE**

Оценочную плату можно приобрести здесь **STM32 X-NUCLEO-IKS01A1**

Смотреть ВИДЕОУРОК

‹ AVR УРОК 39. Акселерометр LSM6DS3. Часть 4

STM Урок 22. HAL. I2C. I2C to LCD2004 ›

Опубликовано в Программирование STM32

Добавить комментарий

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

Источник

Стили Формат… Шрифт Ра…

Имя *

E-mail *

Сайт

× 2 = 16

Отправить комментарий