**Lesson 43**

**Part 1**

**We connect the accelerometer LSM6DS3**

        Today we will look at the sensor, which also combines two functionalities - an accelerometer and a gyroscope. This accelerometer is also an accelerometer made using MEMS technology - **LSM6DS3**. It is installed on the expansion board **STEVAL-MKI160V1**, which in turn is inserted into the expansion board**X-NUCLEO-IKS01A1**, designed to work with the debug card Nucleo. We will connect this evaluation board to the Nucleo STM32F401RE board.

        This accelerometer-gyroscope also can, along with the I2C interface, connect using the SPI interface. But we will use the connection by I2C.

Also, we will use this sensor as an accelerometer in this lesson. As a gyroscope, we will connect it in later studies. In comparison with the accelerometer **LSM6DS0**considered by us in the past , this sensor has better characteristics:

        The reading range is ± 2g / ± 4g / ± 8g / ± 16g;

The sensitivity is 0.061 - 0.49 mg / digit;

Deviation from zero ± 40 mg.

The reading speed of data is 12.5 Hz - 6.66 kHz.

With some other indicators, registers, values ​​and other details of the accelerometer, we will get acquainted in the course of its programming.

To program this sensor, we will create the Accel\_LSM6DS3 project from the previous project Accel\_LSM6DS0. In addition, we will also rename the files lsm6ds0.h and lsm6ds0.c respectively in lsm6ds3.h and lsm6ds3.c.

Run the project in the Cube MX. We will not touch anything there, even the USART speed will be 256000 bps. Generate the project for Keil, open it, turn on the programmer for auto-cutting, connect the file lsm6ds0.c.

Correct in the main.h file in the string zero to three

#include "stm32f4xx\_hal.h"

#include "lsm6ds **3**.h"

The same is done in the file lsm6ds3.c

#include "lsm6ds **3**.h"

// ---------------

We will collect the project.

We will not delete anything yet. The file lsm6ds3.h will be changed to the next state by copying the prepared code into it together with all the macros:

#ifndef LSM6DS3\_H\_

#define LSM6DS3\_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 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\_STREAM 0x05

#define LSM6DS3\_ACC\_GYRO\_FIFO\_MODE\_DYN\_STREAM\_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

// ----------------

void Accel\_Ini (void);

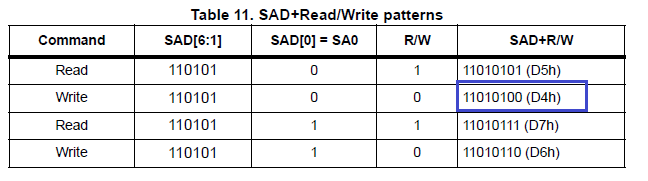
void Accel\_ReadAcc (void);

// ----------------

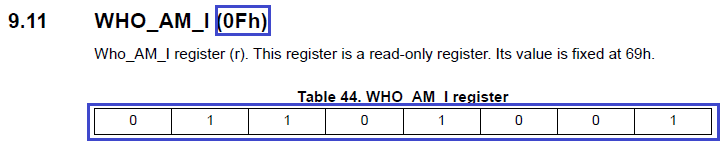
#endif / \* LSM6DS3\_H\_ \* /

The purpose of the macros will be discussed in the process of writing the code. We will collect the project. We will comment on the erroneous code.

We will find in the technical documentation the address on which we will address by I2C (page 35). In this expansion board, the SA0 foot has been connected to the common wire so that we can use both sensors and that there are no identical addresses. Therefore we will use 0xD4.



Find the address of the ID storage register and the identifier itself



Let's correct the code in reading the identifier

        uint8\_t ctrl = 0x00;

**ctrl = Accel\_IO\_Read (0xD4,0x0F);**

        return ctrl;

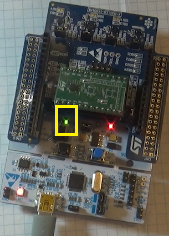
Also, fix the code in the initialization function

        HAL\_Delay (1000);

**if (Accel\_ReadID () == 0x69) LD2\_ON;**

        else Error ();

We will connect the controller, collect the code and check whether the ID was considered



The LED is green, then the identifier is read correctly.

Now start writing the sensor setting before using

void AccInit (uint16\_t InitStruct)

{

        uint8\_t value = 0;

**// auto-increment the address of the register**

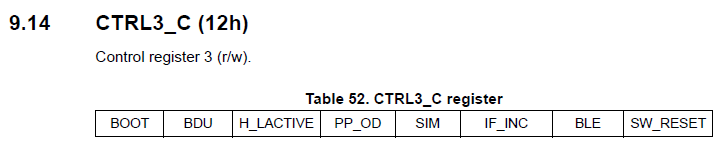
**value = Accel\_IO\_Read (0xD4, LSM6DS3\_ACC\_GYRO\_CTRL3\_C);**

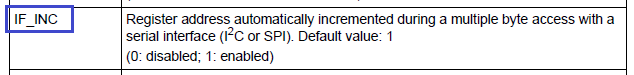
**value & = ~ LSM6DS3\_ACC\_GYRO\_IF\_INC\_MASK;**

**value | = LSM6DS3\_ACC\_GYRO\_IF\_INC\_ENABLED;**

**Accel\_IO\_Write (0xD4, LSM6DS3\_ACC\_GYRO\_CTRL3\_C, value);**

With this code, we refer to the control register 3 and include the auto-increment bit of the register address





While for today we will finish with initialization.

In the [**next part of the**](http://narodstream.ru/stm-urok-43-podklyuchaem-akselerometr-lsm6ds3-chast-2/) lesson, we will completely finish our work with this sensor.

**Lesson 43**

**Part 2**

# ****We connect the accelerometer LSM6DS3****

In the [**previous part of the**](http://narodstream.ru/stm-urok-43-podklyuchaem-akselerometr-lsm6ds3-chast-1/) lesson we got acquainted with the documentation for the sensor, created a project for it, added some macros and set up library files, also wrote the function of reading the accelerometer identifier and started writing the sensor initialization function.

Continue to work with registers of management

        Accel\_IO\_Write (0xD4, LSM6DS3\_ACC\_GYRO\_CTRL3\_C, value);

**// set the BDU bit**

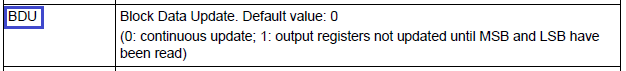
**value = Accel\_IO\_Read (0xD4, LSM6DS3\_ACC\_GYRO\_CTRL3\_C);**

**value & = ~ LSM6DS3\_ACC\_GYRO\_BDU\_MASK;**

**value | = LSM6DS3\_ACC\_GYRO\_BDU\_BLOCK\_UPDATE;**

**Accel\_IO\_Write (0xD4, LSM6DS3\_ACC\_GYRO\_CTRL3\_C, value);**

Here we include the BDU bit, which we already know well from the previous session. The register is the same, the bit includes the sixth



We go further on registers

        Accel\_IO\_Write (0xD4, LSM6DS3\_ACC\_GYRO\_CTRL3\_C, value);

        // select FIFO mode

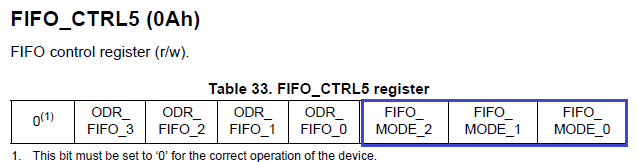
**value = Accel\_IO\_Read (0xD4, LSM6DS3\_ACC\_GYRO\_FIFO\_CTRL5);**

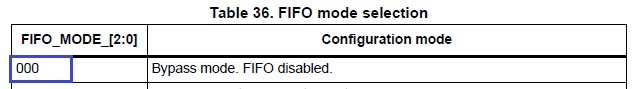
**value & = ~ LSM6DS3\_ACC\_GYRO\_FIFO\_MODE\_MASK;**

**value | = LSM6DS3\_ACC\_GYRO\_FIFO\_MODE\_BYPASS;**

**Accel\_IO\_Write (0xD4, LSM6DS3\_ACC\_GYRO\_FIFO\_CTRL5, value);**

In this code, we select the input-output mode, working already with the fifth FIFO control register (there are separate registers for FIFO). We set the mode without changing (bypass)





We continue to write the code for setting the control registers

        Accel\_IO\_Write (0xD4, LSM6DS3\_ACC\_GYRO\_FIFO\_CTRL5, value);

**// while we turn off the sensor (ODR\_XL = 0000)**

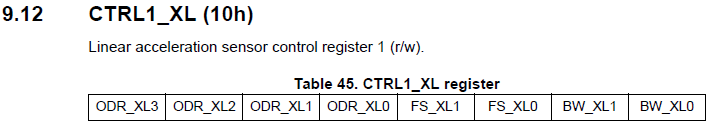
**value = Accel\_IO\_Read (0xD4, LSM6DS3\_ACC\_GYRO\_CTRL1\_XL);**

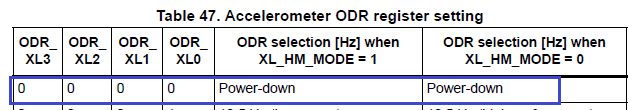
**value & = ~ LSM6DS3\_ACC\_GYRO\_ODR\_XL\_MASK;**

**value | = LSM6DS3\_ACC\_GYRO\_ODR\_XL\_POWER\_DOWN;**

**Accel\_IO\_Write (0xD4, LSM6DS3\_ACC\_GYRO\_CTRL1\_XL, value);**

Well, here everything is the same as in the last lesson. While we disconnect the sensor. We use the first control register





The next code will be:

        // Full scale selection 2G

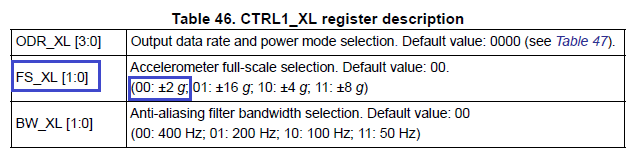
**value = Accel\_IO\_Read (0xD4, LSM6DS3\_ACC\_GYRO\_CTRL1\_XL);**

**value & = ~ LSM6DS3\_ACC\_GYRO\_FS\_XL\_MASK;**

**value | = LSM6DS3\_ACC\_GYRO\_FS\_XL\_2g;**

**Accel\_IO\_Write (0xD4, LSM6DS3\_ACC\_GYRO\_CTRL1\_XL, value);**

It's understandable. We include the maximum possible measurement (the amplitude of the measurement is 2G). The register is the same.



We write further:

        // Turn on the axes

        value = Accel\_IO\_Read (0xD4, LSM6DS3\_ACC\_GYRO\_CTRL9\_XL);

        value & = ~ (LSM6DS3\_ACC\_GYRO\_XEN\_XL\_MASK | \

                                         LSM6DS3\_ACC\_GYRO\_YEN\_XL\_MASK | \

                                         LSM6DS3\_ACC\_GYRO\_ZEN\_XL\_MASK);

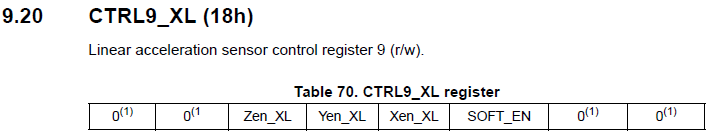
        value | = (LSM6DS3\_ACC\_GYRO\_XEN\_XL\_ENABLED | \

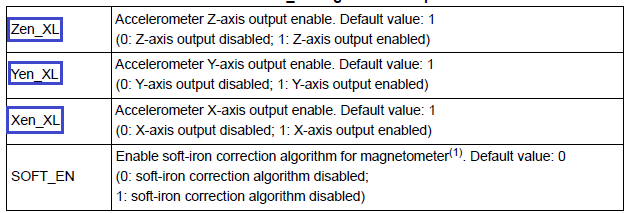
                                        LSM6DS3\_ACC\_GYRO\_YEN\_XL\_ENABLED | \

                                        LSM6DS3\_ACC\_GYRO\_ZEN\_XL\_ENABLED);

        Accel\_IO\_Write (0xD4, LSM6DS3\_ACC\_GYRO\_CTRL9\_XL, value);

Here we already include the axes (from which axes the data will be taken). We use, as usual, all three axes - X, Y and Z. The register uses the ninth





We write further our code:

        Accel\_IO\_Write (0xD4, LSM6DS3\_ACC\_GYRO\_CTRL9\_XL, value);

**// Enable Data Rate 104 Hz**

**value = Accel\_IO\_Read (0xD4, LSM6DS3\_ACC\_GYRO\_CTRL1\_XL);**

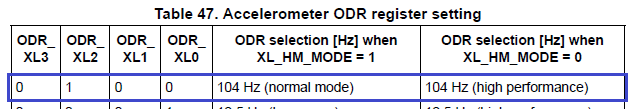
**value & = ~ LSM6DS3\_ACC\_GYRO\_ODR\_XL\_MASK;**

**value | = LSM6DS3\_ACC\_GYRO\_ODR\_XL\_104Hz;**

**Accel\_IO\_Write (0xD4, LSM6DS3\_ACC\_GYRO\_CTRL1\_XL, value);**

}

This is the end of our initialization. Here, of course, we must include our sensor. The frequency of data extraction will be 104 kHz. The first register. We have already used it above



Also, we will correct the code of data collection from the sensor by simply changing the sensor address and registers:

        uint8\_t i = 0;

        buffer [0] = Accel\_IO\_Read ( **0xD4**, **LSM6DS3\_ACC\_GYRO\_OUTX\_L\_XL**);

        buffer [1] = Accel\_IO\_Read ( **0xD4**, **LSM6DS3\_ACC\_GYRO\_OUTX\_H\_XL**);

        buffer [2] = Accel\_IO\_Read ( **0xD4**, **LSM6DS3\_ACC\_GYRO\_OUTY\_L\_XL**);

        buffer [3] = Accel\_IO\_Read ( **0xD4**, **LSM6DS3\_ACC\_GYRO\_OUTY\_H\_XL**);

        buffer [4] = Accel\_IO\_Read ( **0xD4**, **LSM6DS3\_ACC\_GYRO\_OUTZ\_L\_XL**);

        buffer [5] = Accel\_IO\_Read ( **0xD4**, **LSM6DS3\_ACC\_GYRO\_OUTZ\_H\_XL**);

In the function Accel\_ReadAcc while the text display of data from the sensor through the terminal

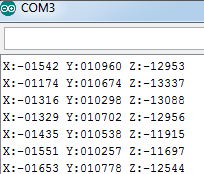
        zval = buffer [2];

**sprintf (str1, "X:% 06d Y:% 06d Z:% 06d \ r \ n", xval, yval, zval);**

**HAL\_UART\_Transmit\_DMA (& huart2, (uint8\_t \*) str1, strlen (str1));**

// buf2 [0] = 0x11;

We will collect the code, we will sew the controller and check in the monitor of the Arduino port the work of our sensor



Comment out the output code on the terminal. Uncomment the output code in the visualization program.The program will also use NS Port Monitor, so the code will be compiled in accordance with the requirements of the protocol of this program (first bytes 0x11 and 0x55).

// sprintf (str1, "X:% 06d Y:% 06d Z:% 06d \ r \ n", xval, yval, zval);

// HAL\_UART\_Transmit\_DMA (& huart2, (uint8\_t \*) str1, strlen (str1));

**buf2 [0] = 0x11;**

**buf2 [1] = 0x55;**

**buf2 [2] = (uint8\_t) (xval >> 8);**

**buf2 [3] = (uint8\_t) xval;**

**buf2 [4] = (uint8\_t) (yval >> 8);**

**buf2 [5] = (uint8\_t) yval;**

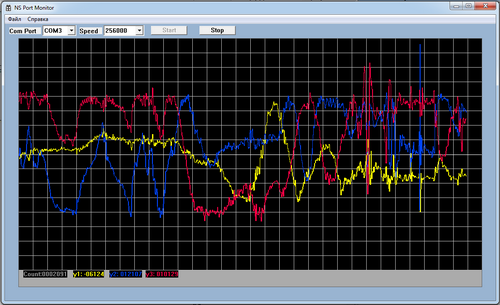
**buf2 [6] = (uint8\_t) (zval >> 8);**

**buf2 [7] = (uint8\_t) zval;**

**HAL\_UART\_Transmit\_DMA (& huart2, buf2.8);**

        if (xval> 1500)

We will collect the project, we will sew the controller, run the program and check the result in practice (click on the image to enlarge the image)

[](http://narodstream.ru/wp-content/uploads/2016/11/image14_1104.png)