Embedded Systems Lab
(TE3060)
Laboratory Exercise 4
Examples for HPS SoC – Controlling G-sensor

Goal: In this lab exercise we will control the G-sensor in the DE10-Nano / DE10-Standard development and education boards by using a Linux operating system on Intel's Cyclone V SoC devices. Linux runs on the ARM processor that is part of the Cyclone V SoC device. The lab shows how to control the G-sensor by accessing its registers through the built-in I2C kernel driver in Linux Board Support Package (BSP).

1. Introduction

The DE10-Nano / DE10-Standard development and education boards are equipped with the ADXL345 3-axis digital accelerometer from Analog Devices. The sensor provides I2C and SPI interfaces. The I2C interface is selected by setting the CS pin to high on both the DE10-Nano board and the DE10-Standard development board. The sensor provides user-selectable resolution up to 13-bit ± 16g. The resolution can be configured through the DATA_FORMAT(0x31) register. In this lab, the data format is configured as:

- Full resolution mode (13 bits)
- ± 16g range mode
- Left-justified mode

The X/Y/Z data values can be derived from the DATAX0(0x32), DATAX1(0x33), DATAY0(0x34), DATAY1(0x35), DATAZ0(0x36), and DATAZ1(0x37) registers. DATAX0 represents the least significant byte and DATAX1 represents the most significant byte. It is recommended to perform multiple-byte read of all registers to prevent change in data between sequential registers read. The following statement reads6 bytes of X, Y, or Z value.

A block diagram of this sensor is shown in Figure 1. For more information, refer to the datasheet of this sensor that can be found in the datasheet folder in the system CD of the development boards.

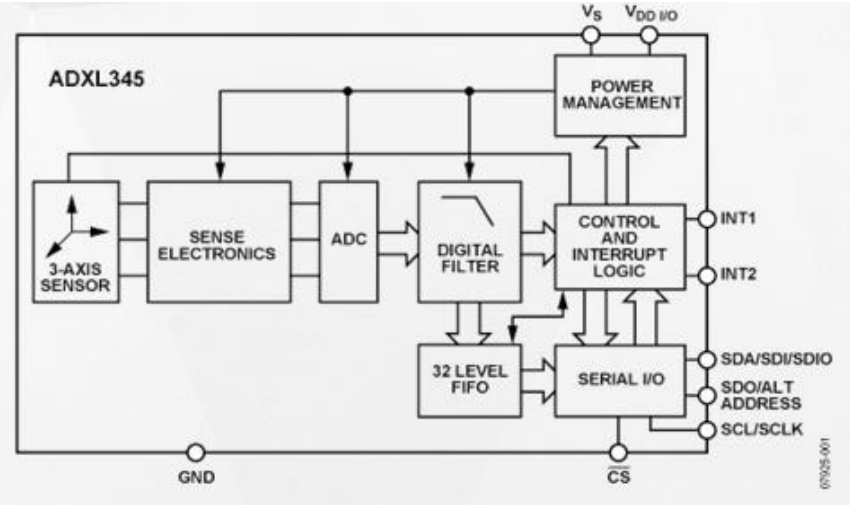


Figure 1. Analog Devices' ADXL345 accelerometer.

The block diagram of the G-sensor as how it is integrated to the SoC is shown in Figure 2. It is connected to the I2C0 controller in the HPS. The G-Sensor I2C 7-bit device address is 0x53. The system I2C bus driver is used to access the register files in the G-sensor. The G-sensor interrupt signal is connected to the PIO controller. This demonstration uses polling method to read the register data.

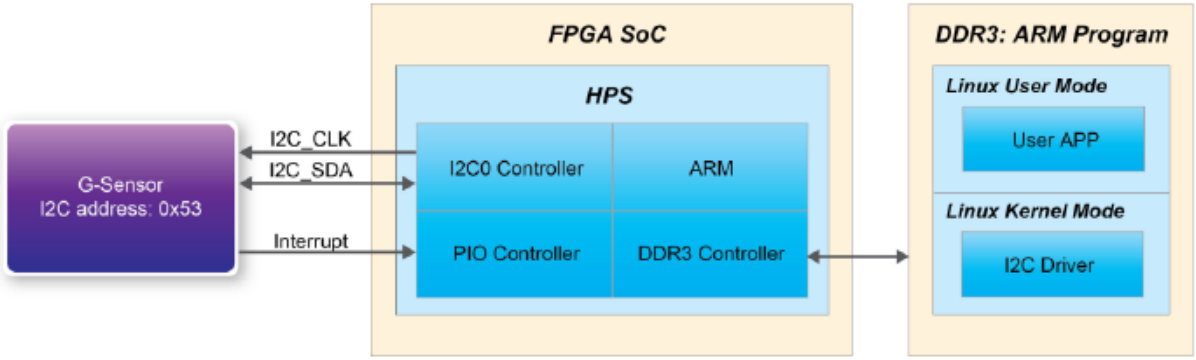


Figure 2. System block diagram of the G-Sensor controller.

The procedure to read a register value from the G-sensor register files by the existing I2C bus driver in the system is:

1. Open I2C bus driver **"/dev/i2c-0"**: file = open ("/dev/i2c-0", O_RDWR);
2. Specify G-sensor's I2C address **0x53**: ioctl(file, I2C_SLAVE, 0x53);
3. Specify desired register index in g-sensor: write(file, &Addr8, sizeof(unsigned char));
4. Read one-byte register value: read(file, &Data8, sizeof(unsigned char))

The G-sensor I2C bus is connected to the I2C0 controller, as shown in Figure 3 for the DE10-Nano development board and Figure 4 for the DE10-Standard development board. In both cases, the driver name given is '/dev/i2c-0'.

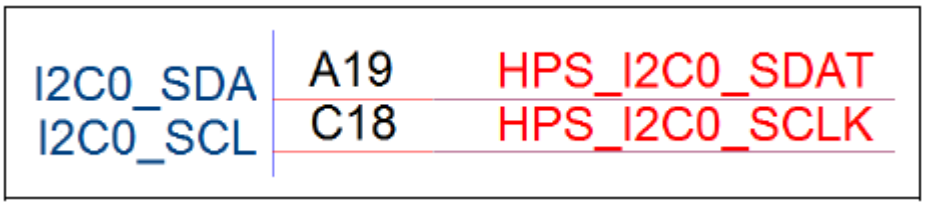


Figure 3. HPS I2C signals in DE10-Nano Development board

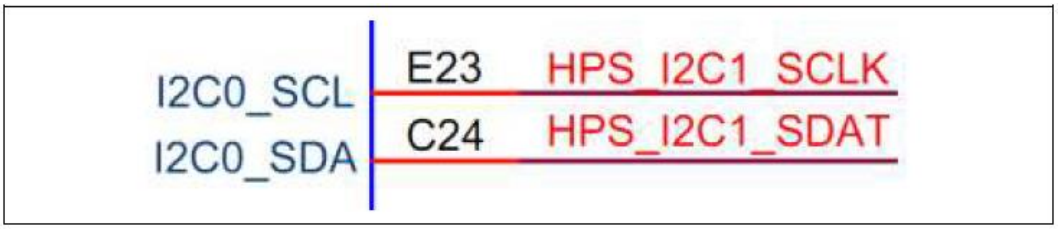


Figure 3. HPS I2C signals in DE10-Standard Development board

2. Software Implementation

To develop the application, we will review the code and files provided in the Demonstration folder in the SystemCD folder of each development board.
It is worth pointing out now that step 4 in the description of the procedure to read a register from the G-sensor register file can have the following variants:

- If we want to write a value into a register, the statement to use is
  write(file, &Data8, sizeof(unsigned char));
- If we want to read multiple byte values, the statement to use is
  read(file, &szData8, sizeof(szData8)); // where szData is an array of bytes
- If we want to write multiple byte values, the statement to use is
  write(file, &szData8, sizeof(szData8)); // where szData is an array of bytes
- It is recommended to perform multiple-byte read of all registers to prevent changes in data between sequential registers read. The following statement can be used to read 6 bytes of X, Y, or Z value
  read(file, szData8, sizeof(szData8)); // where szData is an array of six-bytes

The project consists of four files:

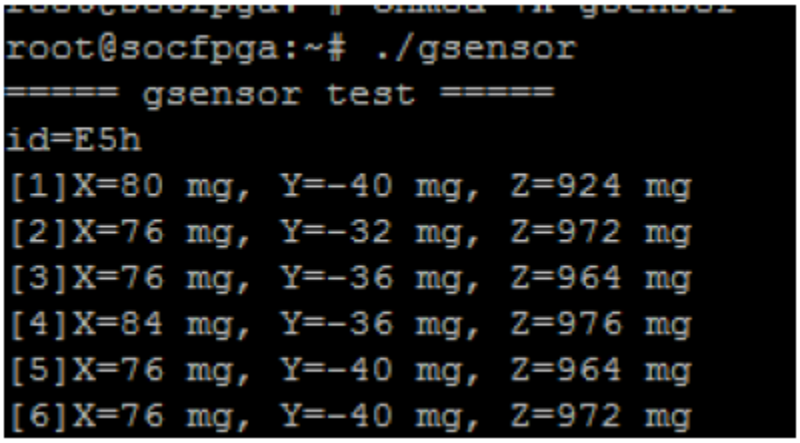
- ADXL345.h: contains a set of defines that are used to configure (write) the accelerometer registers so that it functions as per the programmer needs
- ADXL345.c: contains a set of APIs that can be used to read/write values from the accelerometer
- main.c: contains three additional APIs for reading/writing from/into the accelerometer, and contains the main function
- Makefile

We will discuss the code in class.

3. Running the exercise

Follow the next steps:

1. Compile the program with the Makefile that is included in this lab and create the executable **gsensor**
2. Copy the executable file gsensor into the microSD card under /home/root folder in Linux. Use the ethernet-based procedure we did in lab 2.
3. Connect a USB cable to the USB-to-UART connector on the DE10 Standard/DE10 Nano board and the host PC
4. Launch **PuTTYy** and establish connection to the UART port
5. Type **ls** and you should be able to see the executable file **gsensor**
6. Type **./gsensor** in the UART terminal to start the G-sensor polling
7. The demo program will show the X, Y and Z values in the PuTTY



8. If you want to finish the demo, press "**CTRL+C**" to terminate the demo.