Accelerometer Example

Last updated 10/12/20

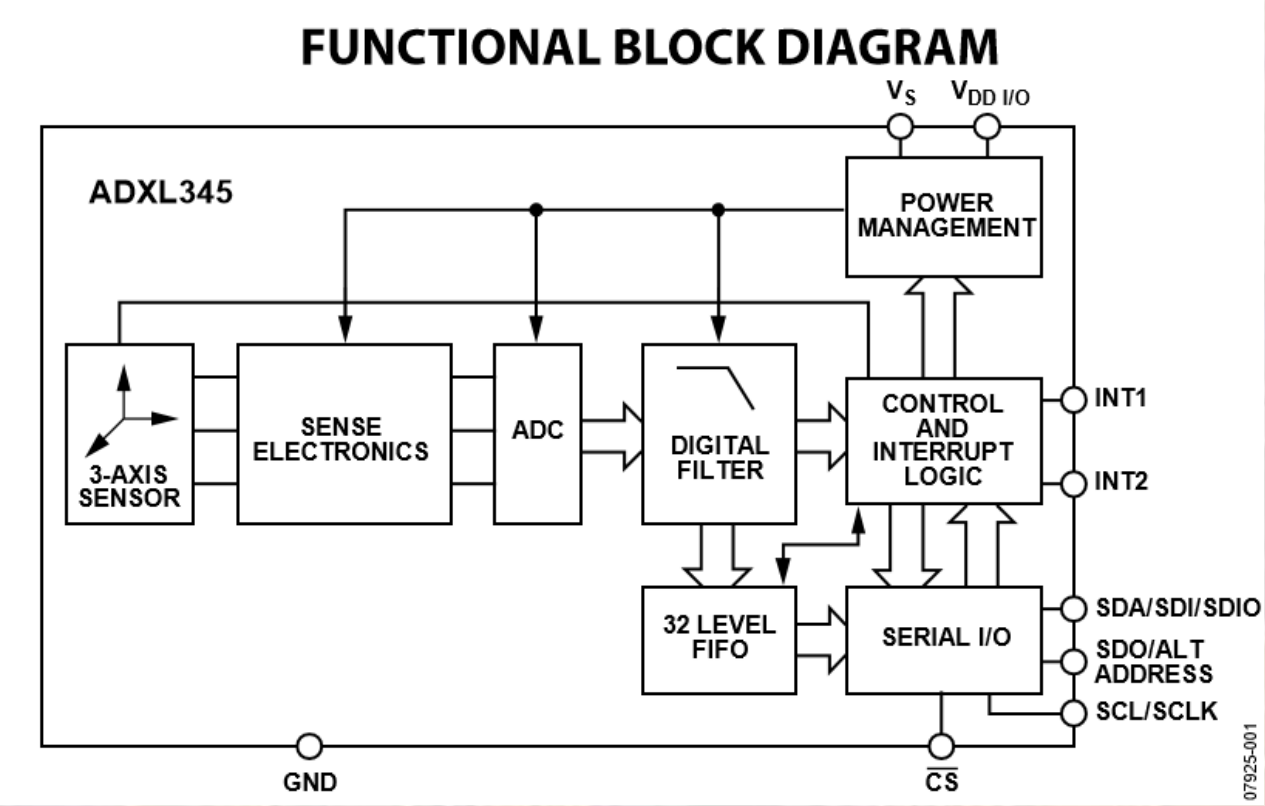
Accelerometer Intro

These slides review the implementation of an Accelerometer in the NIOS system

Upon completion: You should be able to develop your own Accelerometer system

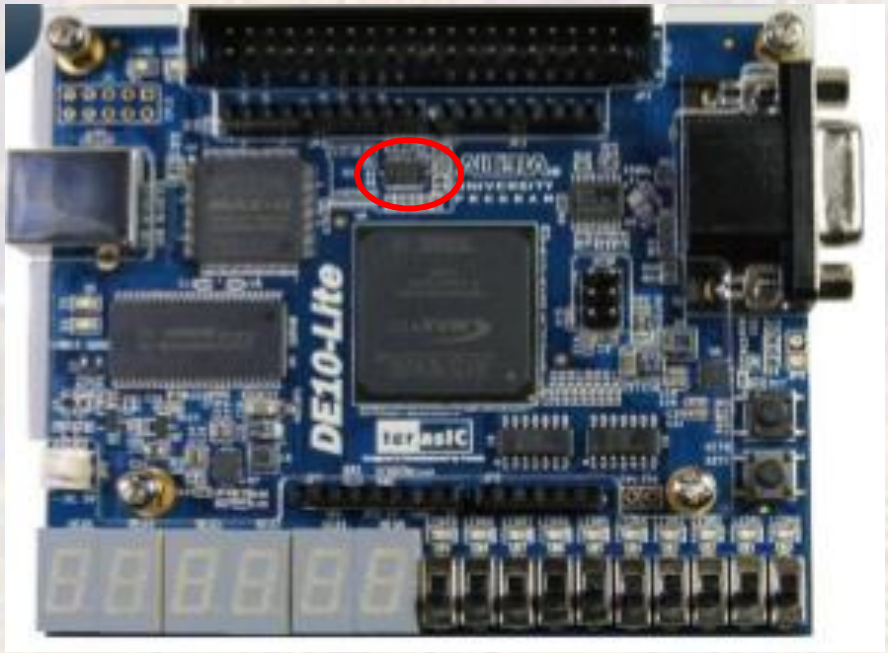
Accelerometer

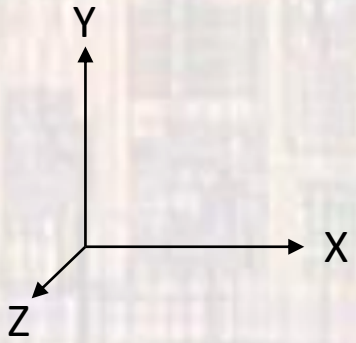
- ADXL345 – 3-axis Accelerometer
  - I2C and SPI interfaces
  - FIFO sample storage



Accelerometer

- ADXL345 – 3-axis Accelerometer
  - Selectable ±2, ±4, ±8, ±16 g measurement range
  - 10bit resolution: 4.3mg/LSB -34.5mg/LSB
  - Up to 3200Hz data rate



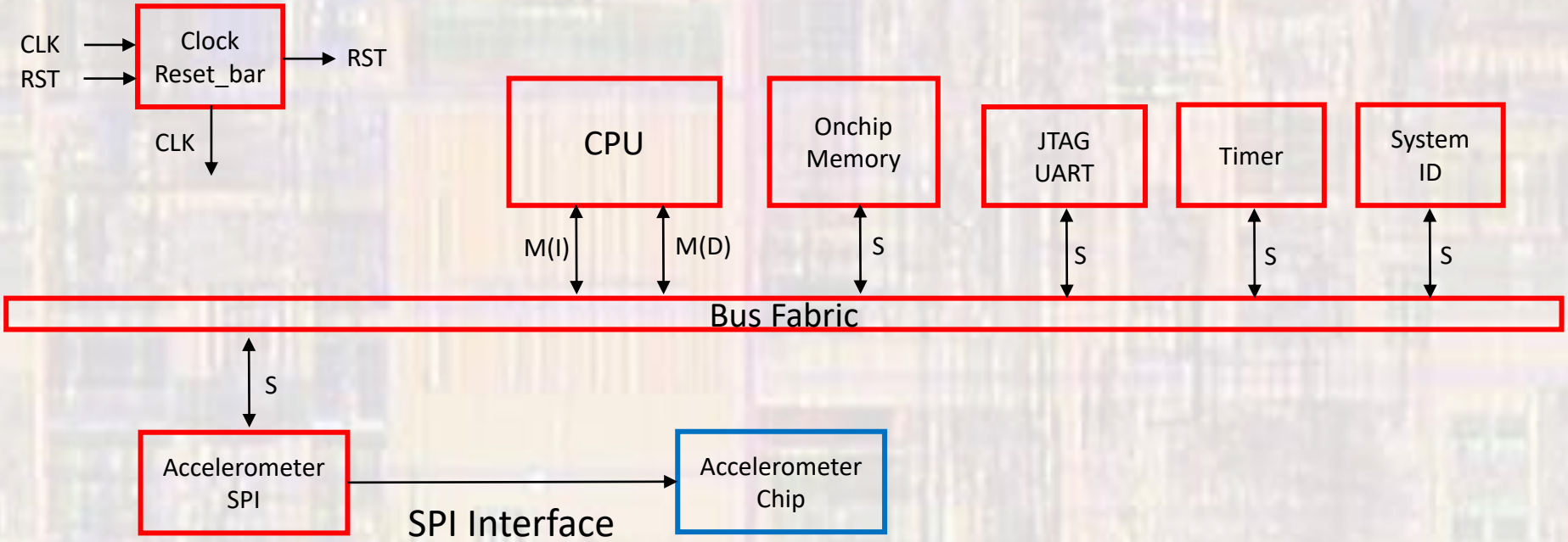


Accelerometer

- ADXL 345 Default modes
  - 4 wire SPI
  - 10bit
  - Data: right justified, sign extended
  - +/- 2g range
  - Trigger on int1

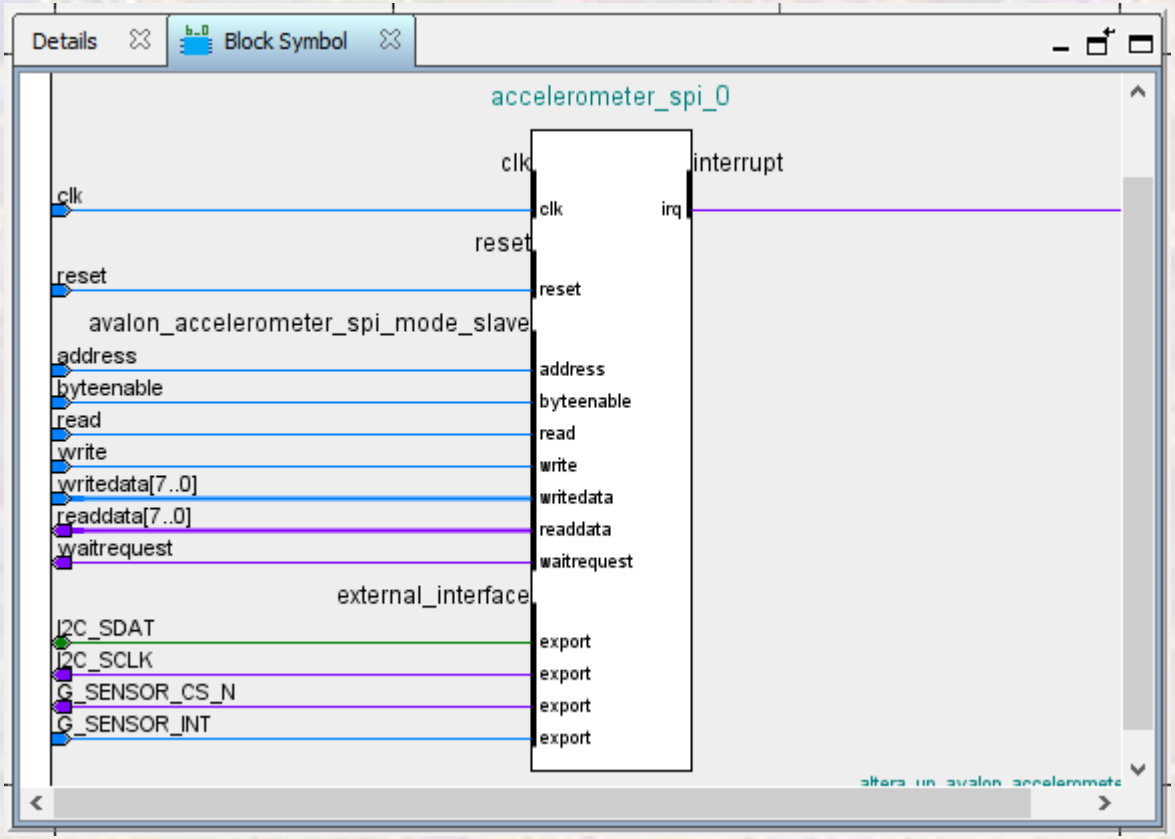
Accelerometer Example

- Accelerometer
  - Create a processor system to use the Accelerometer SPI

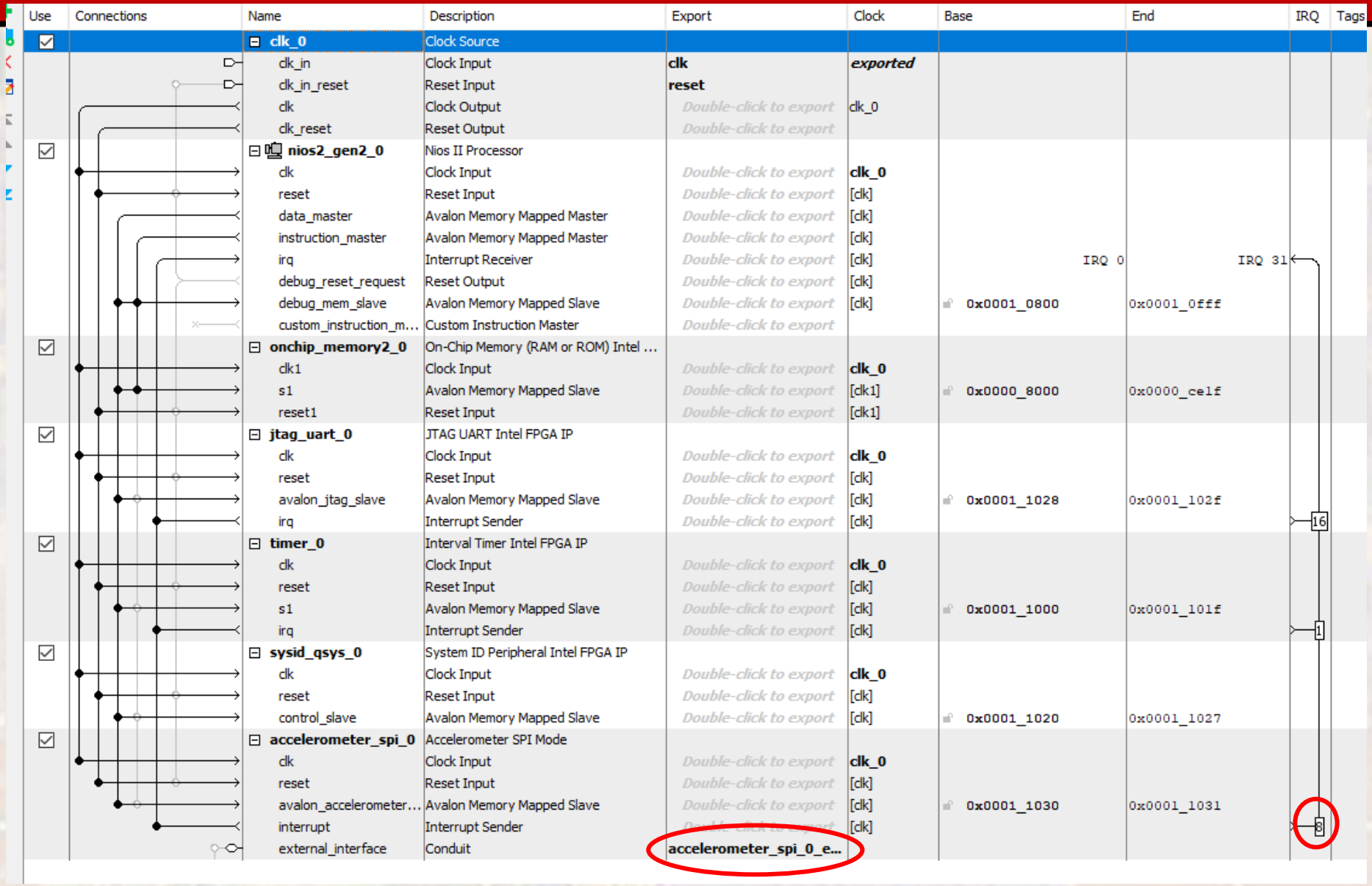


Accelerometer

- Library→ University Program → Generic IO → Altera UP Avalon Accelerometer SPI
  - No Parameters to set



Accelerometer



Accelerometer

```
--------------------------------
--
-- accelerometer_example_de10.vhdl
--
-- created 6/15/18
-- by: johnsontimoj
--
-- uses the accelerometer SPI control block to interact
-- with the on-board accelerometer
--
--------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity accelerometer_example_de10 is
   port(
      CLOCK_50 :     in std_logic;

      G_SENSOR_SCLK:    out std_logic;
      G_SENSOR_SDI:     inout std_logic;
      G_SENSOR_CS_N:    out std_logic;
      G_SENSOR_INT:     in std_logic_vector(1 downto 0)
   );
end entity;

architecture behavioral of accelerometer_example_de10 is
   --
   -- no signals

   component nios_acc is
      port (
         accelerometer_spi_0_external_interface_I2C_SDAT      : inout std_logic := 'X'; -- I2C_SDAT
         accelerometer_spi_0_external_interface_I2C_SCLK      : out   std_logic;        -- I2C_SCLK
         accelerometer_spi_0_external_interface_G_SENSOR_CS_N : out   std_logic;        -- G_SENSOR_CS_N
         accelerometer_spi_0_external_interface_G_SENSOR_INT  : in    std_logic := 'X'; -- G_SENSOR_INT
         clk_clk                                              : in    std_logic := 'X'; -- clk
         reset_reset_n                                        : in    std_logic := 'X'  -- reset_n
      );
   end component nios_acc;


begin

   u0 : component nios_acc
      port map (
         accelerometer_spi_0_external_interface_I2C_SDAT      => G_SENSOR_SDI,      -- I2C_SDAT
         accelerometer_spi_0_external_interface_I2C_SCLK      => G_SENSOR_SCLK,     -- I2C_SCLK
         accelerometer_spi_0_external_interface_G_SENSOR_CS_N => G_SENSOR_CS_N,     -- G_SENSOR_CS_N
         accelerometer_spi_0_external_interface_G_SENSOR_INT  => G_SENSOR_INT(1),   -- G_SENSOR_INT
         clk_clk                                              => CLOCK_50,          -- clk
         reset_reset_n                                        => '1'                -- reset_n
      );
end architecture;
```

Accelerometer

- altera_up_avalon_accelerometer_spi.h

altera_up_avalon_accelerometer_spi.h

```
60 /**
61  * @brief Opens the Accelerometer SPI Mode device specified by <em> name </em>.
62  *
63  * @param name -- the Accelerometer SPI Mode component name in SOPC Builder.
64  *
65  * @return The corresponding device structure, or NULL if the device is not found.
66  **/
67 alt_up_accelerometer_spi_dev* alt_up_accelerometer_spi_open_dev(const char* name);
68
69 /**
70  * @brief Reads configuration data from one of the on-board video device's registers.
71  *
72  * @param accel_spi -- the device structure
73  * @param addr -- a pointer to the location where the read address should be stored
74  *
75  * @return 0 for success
76  **/
77 int alt_up_accelerometer_spi_read_address_register(alt_up_accelerometer_spi_dev *accel_spi, alt_u8 *addr);
78
79 /**
80  * @brief Reads data from the Accelerometer's registers.
81  *
82  * @param accel_spi -- the device structure
83  * @param addr -- the device's configuration register's address
84  * @param data -- a pointer to the location where the read data should be stored
85  *
86  * @return 0 for success
87  **/
88 int alt_up_accelerometer_spi_read(alt_up_accelerometer_spi_dev *accel_spi, alt_u8 addr, alt_u8 *data);
89
```

Accelerometer

- altera_up_avalon_accelerometer_spi.h

altera_up_avalon_accelerometer_spi.h

```
91   * @brief Writes data to the Accelerometer's registers.
92   *
93   * @param accel_spi -- the device structure
94   * @param addr -- the device's configuration register's address
95   * @param data -- the data to be written.
96   *
97   * @return 0 for success
98   **/
99  int alt_up_accelerometer_spi_write(alt_up_accelerometer_spi_dev *accel_spi, alt_u8 addr, alt_u8 data);
00
01 /**
02   * @brief Reads the X Axis value from both registers from the Accelerometer and converts the value to a signed integer.
03   *
04   * @param accel_spi -- the device structure
05   * @param x_axis -- a pointer to the location where the x axis data should be stored
06   *
07   * @return 0 for success
08   **/
09  int alt_up_accelerometer_spi_read_x_axis(alt_up_accelerometer_spi_dev *accel_spi, alt_32 *x_axis);
10
11 /**
12   * @brief Reads the Y Axis value from both registers from the Accelerometer and converts the value to a signed integer.
13   *
14   * @param accel_spi -- the device structure
15   * @param y_axis -- a pointer to the location where the y axis data should be stored
16   *
17   * @return 0 for success
18   **/
19  int alt_up_accelerometer_spi_read_y_axis(alt_up_accelerometer_spi_dev *accel_spi, alt_32 *y_axis);
20
```

Accelerometer

- altera_up_avalon_accelerometer_spi.h

```
/**
 * @brief Reads the Z Axis value from both registers from the Accelerometer and converts the value to a signed integer.
 *
 * @param accel_spi -- the device structure
 * @param z_axis -- a pointer to the location where the z axis data should be stored
 *
 * @return 0 for success
 **/
int alt_up_accelerometer_spi_read_z_axis(alt_up_accelerometer_spi_dev *accel_spi, alt_32 *z_axis);
```

Accelerometer

```
*
* accel.c
*
*  Created on: Oct 7, 2017
*      Author: johnsontimoj
*
*      Basic accelerometer operation
*/
/////////////////
// Includes
/////////////////
#include "altera_up_avalon_accelerometer_spi.h"
#include "system.h"
#include <stdio.h>
#include <unistd.h>

int main(void){
   // define a pointer of type alt_up_accelerometer...
   // to use as a reference in the register functions
   //
   alt_up_accelerometer_spi_dev * acc_dev;

   // open the Accelerometer port
   //  - command is in drivers/inc/alter_up_avalon_accelerometer_spi.h
   // name reference is in system.h
   //  - "/dev/accelerometer_spi_0"
   //
   acc_dev = alt_up_accelerometer_spi_open_dev ("/dev/accelerometer_spi_0");
```

```
   // Check for error and output to the console
   //
   if ( acc_dev == NULL)
      printf ("Error: could not open acc device \n");
   else
      printf ("Opened acc device \n");

   alt_32 xAccel = 0;
   alt_32 yAccel = 0;
   alt_32 zAccel = 0;
   // read and print values
   while(1){
      alt_up_accelerometer_spi_read_x_axis(acc_dev, &xAccel);
      alt_up_accelerometer_spi_read_y_axis(acc_dev, &yAccel);
      alt_up_accelerometer_spi_read_z_axis(acc_dev, &zAccel);
      printf("%li  %li  %li\n", xAccel, yAccel, zAccel);
      usleep(100000);// 0.1sec
   } // end while

   return 0;
}
```