ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 1, January 2014

Data Acquisition From Capacitance Sensor AD7746 To Central Monitoring System Using I^{2}C Protocol

Abhilash C S[1], Muralidhar N[2]

Department of Electronics and Communication, V.V.I.E.T., Mysore, India[1,2]

Abstract: Data acquisition involves gathering signals from various measurement sources and digitizing the signals for the purpose of storage, analysis and presentation at a central monitoring system. Data acquisition system comes in many different PC technology forms to offer flexibility when choosing the system. We can choose from PCI, PXI, PCI express, PXI express, PCMCIA, USB, wireless, I^{2}C and Ethernet data acquisition for test, measurement and automation applications. Data acquisition is the process of sampling signals that measure real world physical conditions and converting the resulting samples into digital numeric values that can be manipulated by a computer. Data acquisition systems typically convert analog waveforms into digital values for processing. The components of data acquisition systems include: Sensors that convert physical parameters to electrical signals and Signal conditioning circuitry to convert sensor signals into a form that can be converted to digital values. Data acquisition applications are controlled by software programs developed using various general purpose programming languages such as C, FORTRAN, Java, Lisp, Pascal and VHDL. Our project essentially deals with analysis of these samples collected by capacitance sensor from various points of the test mass. The analysis is performed using an accelerometer AD7746 which measures the capacitance and converts it to a digital data .This data is then transferred onto a register block module using the I^{2}C interface. Later it is displayed on a PC for further analysis.

Keywords: Accelerometer AD7746, I^{2}C, SDA, SCL, Decoder, Registers

I. INTRODUCTION

Data acquisition involves collection of data from accelerometer and suitably converting them into digital format, finally providing it to the central monitoring system (PC) as an output for data analysis. The project can be broadly categorized into three categories. The first is the measurement of capacitance using Accelerometer AD7746. It is followed by data acquisition section, which includes the use of I²C bus and finally the software interface module including I^{2}C interface, register blocks and decoder logic.

The initial process involves configuration of various registers of AD7746 such as configuration register, cap DAC and capacitance set-up register. Then the process of capacitance measurement followed by capacitance to digital conversion takes place. The digital capacitance data obtained is stored in 3 registers of 8-bit size each i.e., a total of 24-bit capacitance data .The I^{2}C bus is used to transfer the converted digital data from AD7746 to the register block module. The I^{2}C bus is a bi-directional bus with only two lines: SDA (Serial data) and SCL (Serial clock) for serial data and serial clocking respectively. The registers in the register block are used to store the data, address, configuration and status information. Each register is 8-bit in size. The register blocks are written or read by the PC using a decoder logic block. The 16 bit address line and 8 bit data lines are performing the operation of address and data transfer between PC and the decoder logic block.

The name I^{2}C translates into "Inter IC". Sometimes the bus is called IIC or I^{2}C bus. The original communication speed was defined with a maximum of 100 Kbps and many applications don't require faster transmissions. For those that do, there is a 400 Kbps fast and since 1998, a high speed 3.4 Mbps option is available. I^{2}C is not only used on single boards, but also to connect components which are linked via cable. Simplicity and flexibility are key characteristics that make this bus attractive to many applications.

In our approach we have designed and simulated the workflow using Xilinx and model-sim simulator. Our proposal yields a good output in data communication with low bandwidth availability. The approach is simple and fast using I^{2}C protocol.

II. PROJECT ARCHITECTURE :

The project deals with data acquisition and appropriate analysis on the data collected. The project includes developing of an algorithm for configuring the hardware components such as accelerometer AD7746 and acquiring the stored digital capacitance data from it.

The PC is connected to a decoder logic block where the incoming address is decoded and a suitable (read\write) operation is performed between the data bus and the registers present is the register block. The registers present at the register block module are read registers, write registers

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 1, January 2014

and configuration registers. This in turn is connected to an I^{2}C interface module where the data is passed on serially using the SDA line of the I^{2}C bus to the AD7746 device during configuration of the registers of AD7746. The AD7746 measures the capacitance and converts it to a digital data which is stored in three registers inside AD7746. This data is passed on to the register block of the PC interface module using the I^{2}C bus. Thus the I^{2}C bus used is a bi-direction bus. The main part of the project deals with the data transfer using I^{2}C bus which involves various processes. The communication is initiated by a start byte which is a high to low transition on the SDA line while the SCL line is high. Then the address of the register that is to be read or written is indicated in the next field where the last bit indicates a read\write operation. Following the address byte, an acknowledge needs to be received from the AD7756 such that the next data transfer takes place. The next field is the data field which is used to write in the configuration registers of the AD7746. Even this data needs to be acknowledged by AD7746.

When the capacitance data is to be read from the registers of the AD7746, the direction of data transfer is changed i.e., data transfer takes place from AD7746 to the register block of PC interface. This time the PC needs to send an acknowledgement to AD7746. The digital data obtained from AD7746 is displayed on the PC as it as or it is converted into a graphical format. The architecture diagram of the entire project is shown in Fig.1.

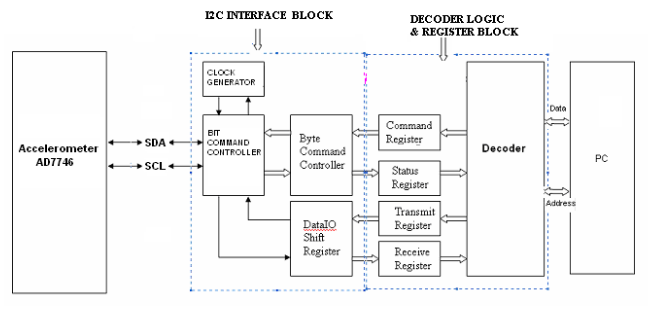

Fig.1 - Project architecture

The AD7746 is a high resolution, capacitance-to-digital converter (CDC). The capacitance to be measured is connected directly to the device inputs. The architecture features inherent high resolution, high linearity, and high accuracy (±4 fF factory calibrated). The AD7746 capacitance input range is ±4 pF (changing), while it can accept up to 17 pF common-mode capacitance (not changing), which can be balanced by a programmable on-chip, digital-to-capacitance converter (CAPDAC). The AD7746 has two capacitive channels. Each channel can be configured as single-ended or differential. The AD7746 is designed for floating capacitive sensors. The 7 registers of AD7746 which are in the scope of our project are status register, capacitance data register H, capacitance data register M, capacitance data register L, configuration register, cap DAC A and capacitance set-up register. The first 4 registers are just read-only type registers and last 3 registers are read/write registers. The configuration register, cap DAC a register and capacitance set-up register needs to be set initially for a particular value before the capacitance measurement and acquisition. After this process the AD7746 measures the capacitance of the test mass and converts it into digital data, later storing it in capacitance data register H, capacitance data register M and capacitance data register L. Using the I^{2}C bus we are acquiring these capacitance data onto our register block module, later transferring it onto central monitoring system. Fig.2 shows the schematic diagram with all modules expanded.

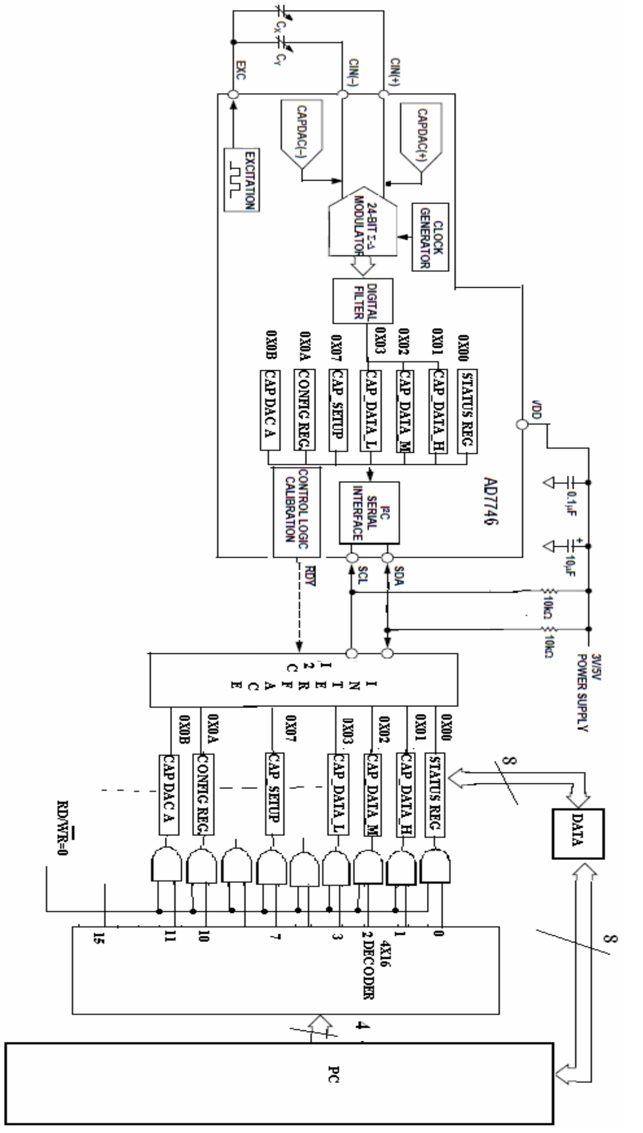

Fig.2 – Schematic diagram with all modules expanded

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 1, January 2014

III. IMPLEMENTATION

System setup for the project was implemented using the following components:

A. Capacitance sensor AD7746 – Configuring and data acquisition

To control the AD7746 device on the bus, the following protocol must be followed. First, the master (central monitoring system) initiates a data transfer by establishing a start condition, defined by a high-to-low transition on SDA while SCL remains high. This indicates that the start byte follows. The bits arrive MSB first. The peripheral that recognizes the transmitted address responds by pulling the data line low during the ninth clock pulse. This is known as the acknowledge bit.

READ OPERATION:
When a read is selected in the start byte, the content of the register that is currently specified by the address pointer is transmitted on to the SDA line by the AD7746. This is then clocked out by the master device and the AD7746 waits for an acknowledgement from the master (PC).

If an acknowledge is received from the master(PC), the address auto-incrementer function automatically increments the address pointer register and points the next addressed register content on to the SDA line for transmission to the master. If no acknowledge is received, the AD7746 return to the idle state and the address pointer is not incremented.

The address pointers' auto-incrementer allow block of data to be written or read from the starting address and subsequent incremental addresses. That means, the three data bytes should be read using one multi byte read transaction rather than three separate single byte transactions. The single byte data read transaction may result in the data bytes from two different results being mixed. The same applies for six data bytes if both the capacitive and the voltage/temperature channel are enabled. The user can also access any unique register (address) on a one-to-one basis without having to update all the registers.

WRITE OPERATION:

When a write is selected, the byte following the start byte is always the register address pointer (sub address) byte, which points to one of the internal registers on the AD7746. The address pointer byte is automatically loaded into the address pointer register and acknowledged by the AD7746. After the address pointer byte acknowledge, a stop condition, a repeated start condition, or another data byte can follow from the master (PC).

A stop condition is defined by a low-to-high transition on SDA while SCL remains high. If a stop condition is ever encountered by the AD7746, it returns to its idle condition and the address pointer is reset to Address 0x00.

If a data byte is transmitted after the register address pointer byte, the AD7746 load this byte into the register that is currently addressed by the address pointer register, send an acknowledgement, and the address pointer auto-incrementer automatically increments the address pointer register to the next internal register address. Thus, subsequent transmitted data bytes are loaded into sequentially incremented addresses.

If a repeated start condition is encountered after the address pointer byte, all peripherals connected to the bus respond exactly as outlined above for a start condition, that is, a repeated start condition is treated the same as a start condition. When a master device issues a stop condition, it relinquishes control of the bus, allowing another master device to take control of the bus. Hence, a master interested to retain control of the bus issues successive start conditions known as repeated start conditions.

B. I2C interface

The operation is started with a start byte which is a high to low transition on SDA line when SCL is high, followed by the transfer of 8-bit address. The address is transmitted bit-wise. So a variable count is initialized to 0 and is incremented until it becomes 7 indicating the entire byte of address has been transmitted. Later the master should wait for the acknowledgement from the slave. After the acknowledgement is received, the master proceeds with transmission of data byte in bit-wise pattern. Again the count variable is initialized to 0 and is incremented until it becomes 7 indicating the entire byte of data has been transmitted. Again the master should wait for the acknowledgement from the slave. After the acknowledgement is received, the master can end the communication by sending a stop byte which is a low to high transition on SDA line when SCL line is high.

The main part of the project deals with the data transfer using I^2C bus which involves various processes. The communication is initiated by a start byte which is a high to low transition on the SDA line while the SCL line is high. Then the address of the register that is to be read or written is indicated in the next field where the last bit indicates a read\write operation. Following the address byte, an acknowledge needs to be received from the AD7746 such that the next data transfer takes place. The next field is the data field which is used to write in the configuration registers of the AD7746. Even this data needs to be acknowledged by AD7746.

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 1, January 2014

When the capacitance data is to be read from the registers of the AD7746, the direction of data transfer is changed i.e., data transfer takes place from AD7746 to the register block of PC interface. This time the PC needs to send an acknowledgement to AD7746. The digital data obtained from AD7746 is displayed on the PC as it as or it is converted into a graphical format.

The flowchart of the I^{2}C controller section is shown as in Fig.3

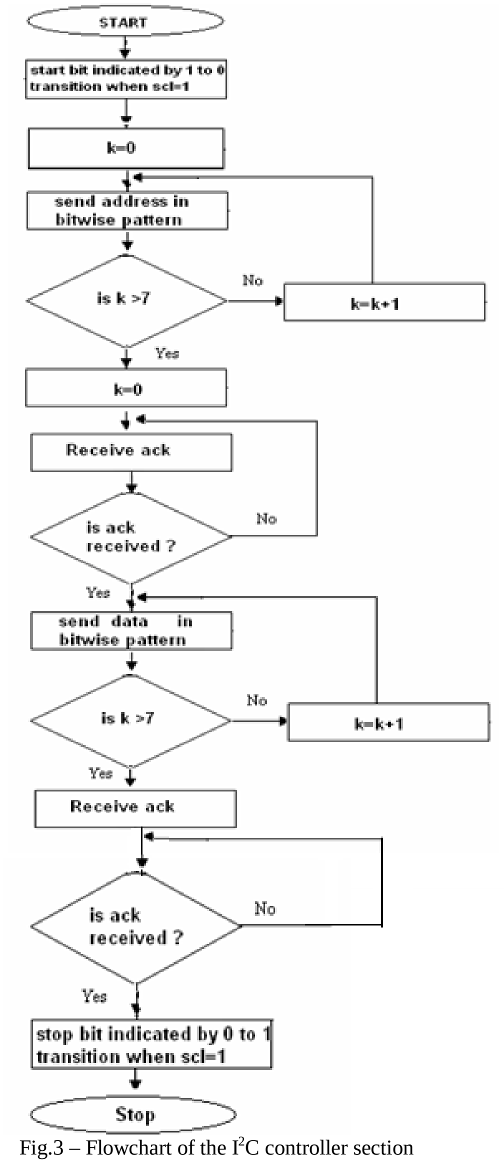
START
start bit indicated by 1 to 0 transition when scl=1
k=0
send address in bitwise pattern
is k >7
No
k=k+1
Yes
k=0
Receive ack
is ack received ?
No
Yes
send data in bitwise pattern
is k >7
No
k=k+1
Yes
Receive ack
is ack received ?
No
Yes
stop bit indicated by 0 to 1 transition when scl=1
Stop

Fig.3 – Flowchart of the I^{2}C controller section

C. Decoder logic module

Using a 4X16 decoder, the last 4bits of the address line is decoded and a suitable register is selected from which the data is either read or written depending upon the status of RD/WR bus. The enable for the decoder can be a simple '1' bit or the address lines (A15-A4) are used to enable or disable the decoder. The total number of registers available is 16, where only a few are used for our application. The read registers are those which have the capacitance data and the status register and write registers are those used to store the configuration data, capacitance set-up register.

Decoder logic algorithm :

The algorithm for the decoder logic and register block is as follows.

Step1: Define the inputs - reset, read/write, address, data, registers q1, q2, q3, q4 and outputs- registers q7,q10,q11.

Step2: Use the last 4 bits of the address lines as a input for a 4X16 decoder.

Step3: Depending upon the decoder input, a single output line is activated which is ANDed with 'RD' signal.

Step4: The output of the AND gate acts as a synchronous clock input for all the D-flip flops used as the register.

Step5: If the rd signal is '1' the data is transferred onto to one of the registers as defined by the incoming address line.

Step6: If the rd signal is '0' the data present in one of the register as defined by the incoming address line is transferred onto the data bus, which is further transferred onto the PC.

Step7: If reset 'rst' is '1' the particular register as indicated by the incoming address line is set to uninitialized state.

Algorithm for the I^{2}C module:

The coding procedure is explained in depth by the following algorithm.

Step1: Define the inputs - Serial Clock (SCL), Serial data (SDA), RESET, Outputs - Register Data out1 – Data out7, Acknowledge (ACK), startorstop and temporary variables- present state ,next state ,start, stop

Step2: Read inputs SCL, SDA, RESET and contents of input buffer.

Step3: Check the condition whether RESET is 1. If true go to step4 else go to step5.
Step4: Reset all outputs and temporary variables. Reset all input and output memory buffers to "00000000", Reset address to "0111", which denotes initial state. Go to step21

Step5: Check for Rising edge SCL. If true go to step6 else repeat step5

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 1, January 2014

Step6: Set present state to be equal to next state. Check for 1 to 0 transition of SDA when SCL is 1. If true, go to step7 Else if there is 0→1 transition, then go to step7a

Step7: Set the variables start and startorstop to 1, go to step8

Step7a: Set the variables stop and startorstop to 0, go to step4

Step8: Check whether startorstop is 1. If true go to step9 else go to step4

Step9: Check for Rising edge SCL. If true go to step11 else go to step10

Step10: Continue in the same state .i.e., present state = present state. Go to step9

Step11: IDLE state: Get the next state [SetupState_CAP]

Step12: Capacitance Setup state: Send data contained in Register Data out5 ("10000001"- to enable the capacitance channel) through SDA line to input buffer5 bitwise. Set ACK to '0' and Address to "1010"(Configuration Setup state). Get the next state [ACK State1]. Go to step15.

Step13: Configuration Setup state: Send data contained in Register Data out6 ("00000010" - single conversion) through SDA line to input buffer6 bitwise. Set ACK to '0' and Address to "1011"(Capacitance DAC Setup state). Get the next state [ACK State1]. Go to step15

Step14: Capacitance DAC Setup state: Send data contained in Register Data out7 ("11111111" - positive input, Full range 0 to 8pF) through SDA line to input buffer7 bitwise. Set ACK to '0' and Address to "0000"(Address State). Get the next state [ACK State1]. Go to step15

Step15: ACK State1: Send ACK for the received data and set SDA to '0' during this state. Check the Address and continue with the next state.
i.e., If Address is "1010", go to step13.
If Address is "1011", go to step14.
If Address is "0000", go to step16.

Step16: Address State: Send 8-bit Address stored in input buffer1 (Status) bitwise to Data out1 through SDA line. Set ACK to '0' and Address to "0001" (Higher data state). Get the next state [ACK State]. Go to step17

Step17: ACK State: Send ACK for the received data and set SDA to '0' during this state. Check the Address and continue with the next state.
i.e., If Address is "0001", go to step18.
If Address is "0010", go to step19.
If Address is "0011", go to step20.

Step18: Higher data state: Send 8-bit Converted Data stored in input buffer2 (Higher byte) bitwise to Data out2 through SDA line. Set ACK to '0' and Address to "0010" (Middle data state). Get the next state [ACK State]. Go to step17.

Step19: Middle data state: Send 8-bit Converted Data stored in input buffer3 (Middle byte) bitwise to Data out3 through SDA line. Set ACK to '0' and Address to "0011" (Lower data state). Get the next state [ACK State]. Go to step17.

Step20: Lower data state: Send 8-bit Converted Data stored in input buffer4 (Lower byte) bitwise to Data out4 through SDA line. Set ACK to '0' and Address to "0001" (Higher data state). Get the next state [ACK State]. Go to step17

Step21: Next state will always be idle i.e., channel is free.

IV. OBSERVATION AND RESULTS

The initial process of configuration of various registers of AD7746 such as configuration register, Cap DAC and capacitance set-up register is successful. The digital capacitance data is stored in 3 registers of 8-bit size each i.e., a total of 24-bit capacitance data. After reading the contents of status register from AD7746, the capacitive data is read from the three capacitance data registers. Finally it is transferred onto a PC using the decoder logic section. The screenshot as shown in Fig.4 denotes the transfer of value 10000001 to Capacitance set-up register which enables the capacitive channel for single capacitive conversion and doubles the capacitive channel conversion times and slightly improves the capacitive channel noise performance for the longest conversion times.

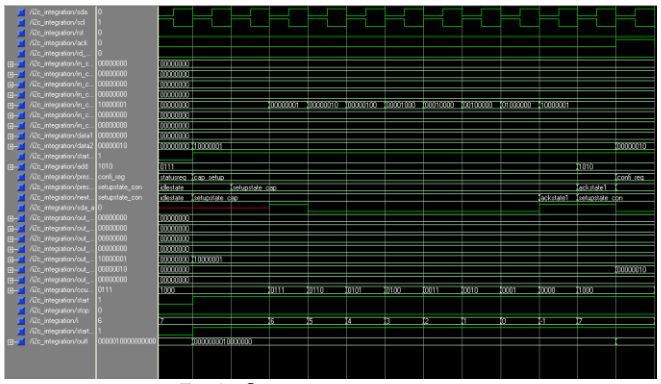

Fig.4 - Capacitance set-up register

The screenshot as shown in Fig.5 denotes the transfer of value 00000010 to Configuration set-up register which enables single conversion of capacitive data. The Configuration set-up register is also called mode set-up register.

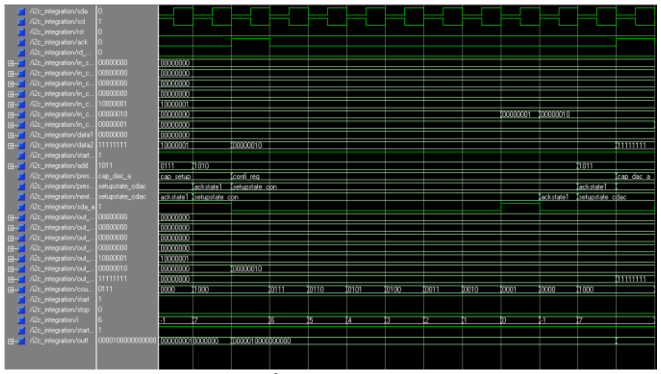
Fig.5 - Configuration set-up register

The screenshot as shown in Fig.6 denotes the transfer of value 11111111 to CAP DAC A register which connects capacitive DAC A to the positive capacitance input and enables a full scale value.

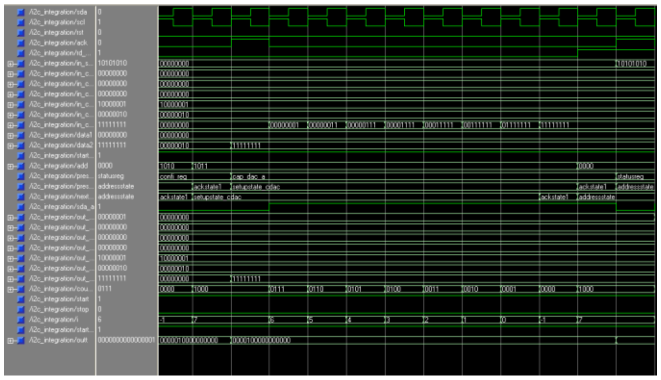
Fig.6 - CAPDAC A register

The screenshot as shown in Fig.7 denotes the contents of STATUS register which provides the status of accelerometer AD7746 whether it is ready for a new capacitance data transfer or not. This is a read-only register.

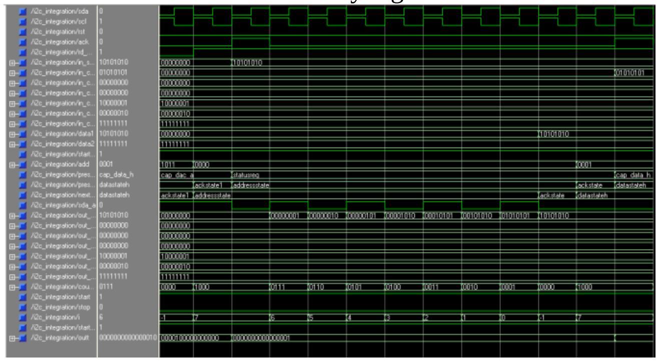
Fig.7 - STATUS register

The screenshot as shown in Fig.8 denotes the CAP DATA_H register which consists of the digital capacitive data's higher byte.

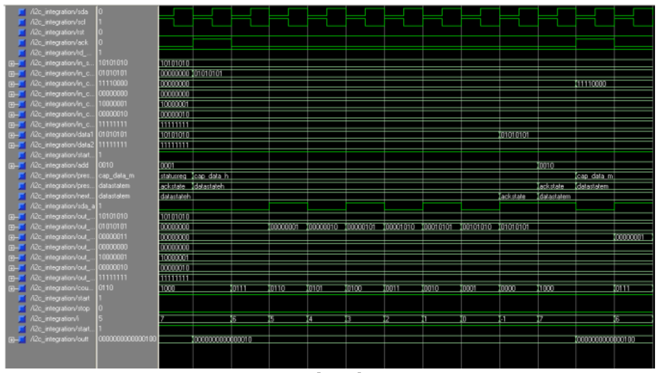
Fig.8- Cap data_h register

The screenshot as shown in Fig.9 denotes the CAP DATA_M register which displays the digital capacitive data's middle byte.

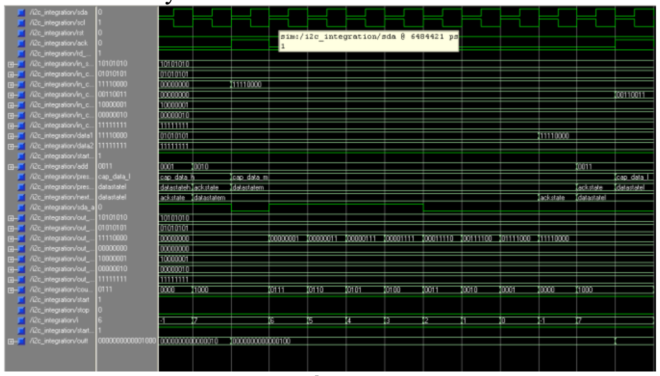
Fig.9 - Cap data_m register

The screenshot as shown in Fig.10 denotes the CAP DATA_L register which consists of the digital capacitive data's lower byte.

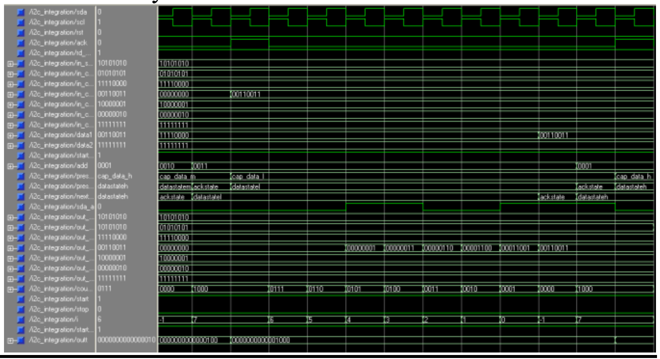
Fig.10 - Cap data_l register

V. CONCLUSION

The objectives that have been set during the start of our project have been met. The project mainly revolved around generation of the appropriate codes. The code developments and the accompanying memory requirements have been successfully developed, from scratch and the working is verified using the simulator. The applications of this project can be in fields such as position sensing, level sensing, humidity sensing and impurity detection. Applications involving position and level sensing of a particular test mass (say, Satellite) can use this accelerometer AD7746. Also a wireless I^2C protocol can be developed for this purpose if the application is used to control a test mass which is situated far-off from the controlling station, i.e., in the outer space. This application requires very low bandwidth usage.

ACKNOWLEDGMENT

We are extremely grateful to those who have helped and supported us during the project. Our deepest thanks to our advisors Anitha D.B., Prof., VVIET and Prashanth P.S., Scientist, ISRO for their continuous encouragement and suggestions throughout the course of this work. It was our pleasure to work under their guidance regarding project by explaining many scenarios where the project can be used.

REFERENCES

[1]Dale Johnson, Geotest- Marvin TestSystems, Inc., Irvine, "Implementing Serial Bus Interfaces with General Purpose Digital Instrumentation", California IEEE 2009

[2] Bruce, J.W. Gray, M.A. Follett," Personal digital assistant (PDA) based I2Cbus analysis" Dept. of Electronics. & Computer Eng., Mississippi State, IEEE Transactions on Consumer Electronics Vol. 49, No. 4, NOVEMBER 2003

[3] Mircea R. Stan, Wayne P. Burleson "Bus Invert Coding for Low Power I/O" in International conference on Very Large Scale Integration (VLSI) Systems, IEEE Transactions on Very Large Scale Integration (VLSI)Systems archive Volume 3 Issue 1, March 1995,Page 49-58

[4]Analog Devices AD7746: 24-bit, 2 Channel Capacitance to Digital Converter, in 2010.