A REAL-TIME EMBEDDED DATA ACQUISITION SYSTEM FOR SURFACE MEASUREMENTS USING MULTIPLE LINE LASERS

by

A P VIKRAM SIMHA

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2015

Acknowledgements

First and foremost, I like to express my sincere gratitude to my adviser Dr. Roger Walker for the constant support he has provided. His vast knowledge and enthusiasm has been a motivator since the first time I met him. His guidance has helped me to get through all the phases of my thesis. I will deeply miss working alongside him, at the Transportation Instrumentation Laboratory as I move on to the next phase of my life.

I cannot express enough thanks to Dr Hao Che and Mr David Levine for their continued support and encouragement. I offer you my sincere appreciation for serving in the defense committee in such short notice, despite your busy schedule.

Special thanks to Lars-Peter Clausen of Analog Devices for his invaluable support in enabling the ADC.

My completion of this project would not have been accomplished without the support of my fellow lab-mates Ashwin, Sushruth, Ajith and Sachin.

Finally, I would like to express my deepest gratitude to my family for caring and encouraging me at all times. It was a great comfort and relief to know that you were all with me while I completed this thesis.

April 15, 2015

Abstract

A REAL-TIME EMBEDDED DATA ACQUISITION SYSTEM

FOR SURFACE MEASUREMENTS USING

MULTIPLE LINE LASERS

A P VIKRAM SIMHA, M.S.

The University of Texas at Arlington, 2015

Supervising Professor: Roger Walker

In the last few years there has been a significant increase in the number of hand held devices. These devices boast of delivering features such as high performance, low power consumption, high memory availability, serial and parallel interfacing capability, connectivity through the Ethernet, wireless, etc., at a very low cost. These embedded devices powered by open source software like Linux and Arduino have paved the way for the development of high efficiency, low cost portable products in a very short period of time.

The objective of this thesis is to enhance the portability, efficiency, and other features of the existing Roline Laser profiling system used by the Texas Department of Transportation. The focus of this research is to replace the slow and bulky processing procedure of the Roline laser profiling system with new mobile technology. This includes the use of the Intel® Galileo which features the SoC Quark 1000. The improvements include extending the capabilities for multiple laser surface measurement methods and GPS tracking information.

Table of Contents

List of Illustrations

List of Tables

Chapter 1

INTRODUCTION

The Highway Road network forms a significant part of the transportation system in the United States. Spanning over 200,000 miles [19], it is a significant contributor for the economy of the country. Usage of these facilities causes wear and tear of the highways and therefore requires a monitoring system.

Road surface profiling is a process of measuring the surface of the road by the collection of data with the help of devices like, lasers sensors, accelerometers, etc., for the purpose of calculating such pavement characteristics as the friction, ride comfort, etc., of the road surface. The US state Department of Transportation (DOT's), Road builders, and contractors use profilers to measure the various characteristics of the road.

Figure 1-1 Map of the present U.S. Highway network [17]

The Texas Department of Transportation (TxDOT) holds regular surveys to track the conditions of the State highways in Texas. Customized vehicles' equipped with different varieties of profilers, are deployed for gathering of raw data of the road surface. The collected data is further analyzed to infer information related to the quality of the road surface and will be used for maintenance and other purposes.

1.1 Need for Furtherance

From the early “Inertial Profiler” of 1960’s to present day ones, there have been many enhancements to the profiler equipment. Efficient computing units and high accuracy sensors like single point and line lasers have been introduced into the world of road profiling, which has drastically improved the accuracy of the profile data collected and its analysis.

The TxDOT in collaboration with the Transportation Instrumentation Laboratory of CSE Department at UTA have developed many profiler systems over the past decade. The Real-Time Single Point Laser Profiler and Line Laser Profiling system are some of them that are currently in use.

The existing Line Laser Profiling system has a wide foot print (100mm) laser but, in comparison to the total surface area of the road it covers only a small section of the road surface. There is a need to include more surface area of the road in the profile calculation for accuracy.

The profile data indicates the quality of road surface; it does not associate each profile value with the exact geographic location from which it has been acquired and analyzed. A procedure for binding the profile value with the accurate location to which it indicates is required.

Over decades the power of the computing units has increased, on the other front, the physical dimensions have reduced. The current profiling vehicles still use a slow and bulker computational unit. Upgrading it to a high performance compact embedded processor will provide a better and efficient solution.

1.2 Objective and Plan

This research project suggests enhancements to the shortcomings of the existing Line Laser profiling system and provides design ideas and implementation to build a better system

The Chapter 2 briefly describes the existing Line Laser profiling system and provides solution for tracking the location with the calculated profiling data.

In Chapter 3 does a comparative study on the newly available embedded devices and proposes the best suitable one for the new system.

Chapter 4, 5 discusses the implementation of the new embedded profiling system.

Chapter 6 provides the results and observations of the experiments on the new embedded profiling system.

Chapter 7 provides suggestions for future work and a brief preview to the implementation of multi laser system.

Chapter 8 provides the summary of the project.

Chapter 2

ROLINE LINE LASER SYSTEM

The Roline lasers fall under the family of wide-footprint lasers. Since its introduction in 2005, they have been used extensively in the paving industry for road profiling. This chapter provides a brief overview of the existing Roline Line Laser profiling system developed by the researchers of UTA. The TxDOT has deployed this system for road profiling on a regular basis. The detailed report on this system can be read from Technical Report 0-6610-1 [21]. Following sections provide snippets from the Technical Report 0-6610-1.

2.1 Hardware components and description

The Roline line laser profiling system consists of several key components. In the following section, each of these key components is described briefly.

2.1.1 Roline Laser

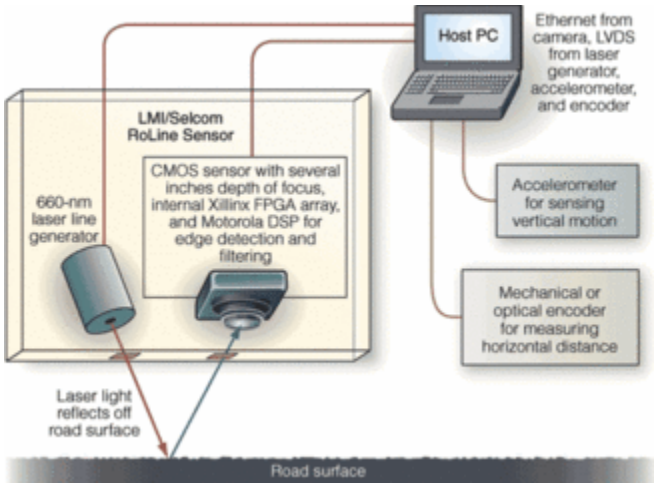

Figure 2-1 Roline Laser [20]

The figure 2-1 shows the Roline laser used for measuring the profile of the road surface. The Roline laser consists of line laser projector and CMOS sensor which are used in combination to generate the height information. The Roline laser system projects the line laser beam onto the surface of the road and it collects the reflected light beam through a connected sensor. Based on the angular deviation of the light from the normal, it determines the information related to height. This information is passed to the host machine through the RJ 45 interconnect. It also provides a sync signal of 3 KHz to indicate the start of each scan and used to synchronize the RJ 45 data. The Roline laser is capable of operating in two modes:

1. The Free mode: - gives out the raw data and host PC should process it further.
2. The Bridge mode: - gives out filtered average value of the raw data.

2.1.2 Accelerometer

It is used to sense the vertical motion of the moving vehicle. The instantaneous height of the vehicle can be calculated by double integration of the measured vertical acceleration.

2.1.3 Distance Encoder

The accurate distance travelled while performing the testing is calculated by using the distance encoder. It is connected to the rear wheel of the vehicle.

2.1.4 Start Sensor

An infrared start sensor is used to determine the starting and ending points of the test data. The starting and ending locations on the road surface is marked with the help of a reflective tape. The start sensor detects these tapes when the vehicle moves over it

during the testing. This is required to keep consistency in starting and ending points during repetitive runs.

2.1.5 ADC DT9816

Except the Ethernet packet or RJ 45 data from the Roline Laser, the remaining information from all the sensors described above is collected in real time with the help of a high resolution ADC, the DT9816. All the sensors are connected to the available analog channels of the ADC DT9816. The ADC samples the signal and passes the data to the host PC and will be used in the profile calculation.

2.2 Hardware setup

The Figure 2-1 shows the block diagram of the Single Roline Line Laser profiling system. The Roline laser is connected to the processing unit such as PC through RJ-45 interconnect. All the sensors are connected to the Analog to Digital Converter (ADC) DT9816. The sensor data is sampled by the DT9816 and it is transferred to the PC through an USB connection.

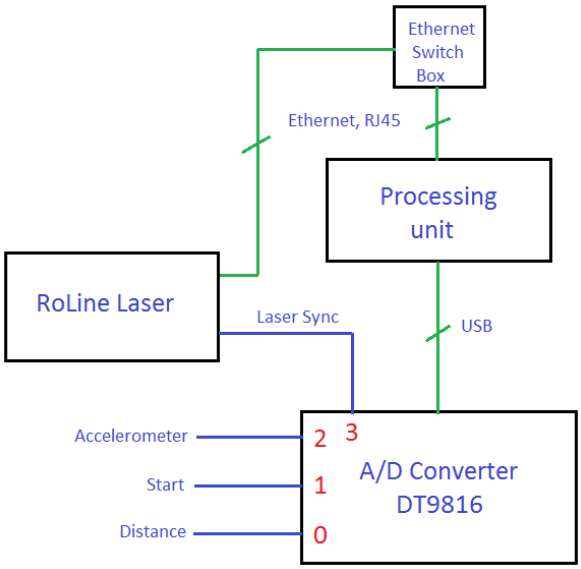

Figure 2-2 Block diagram of Roline Line Laser Profiling System

The sensors are connected to the DT9816 in the following order:

1. Distance sensor on channel 0.
2. Start sensor on channel 1
3. Accelerometer on channel 2
4. Roline Laser sync signal on channel 3.

In total, the Single Roline laser profiling system uses 4 channels on the DT9816.

The Figure 2-3 shows the connections to be made between the signal interface board and the ADC DT9816. The Power / Sync board provides the power to the whole profiler system from an external 12V DC supply.

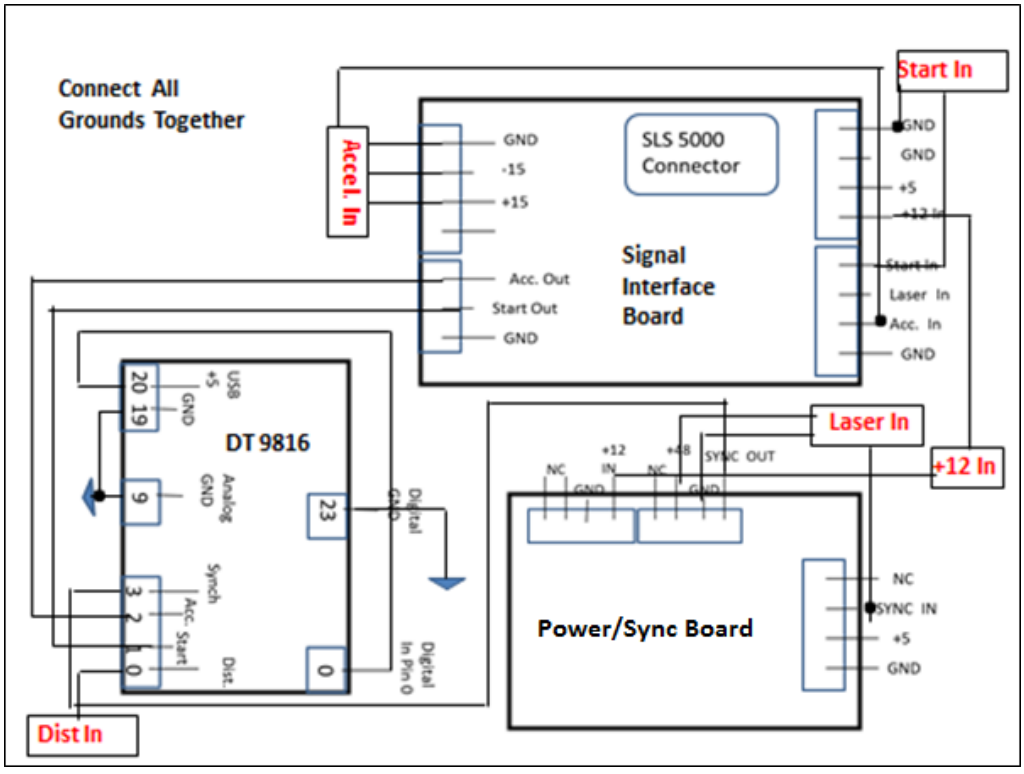

Figure 2-3 Signal Interface, DT9816, and Power module connections [21]

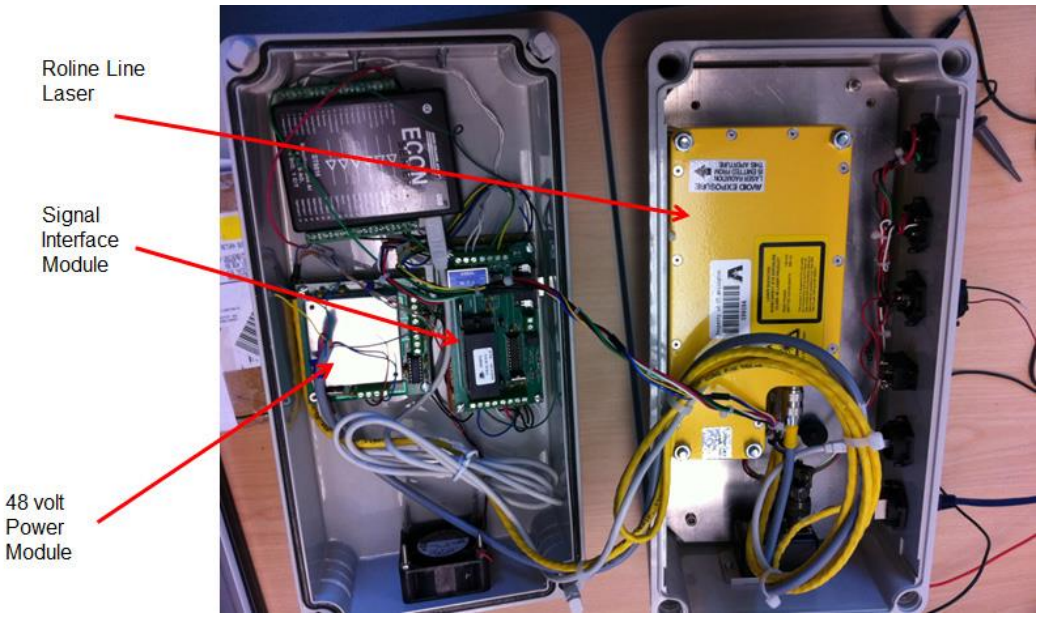

Figure 2-4 Roline Laser Profiling System [21].

The Figure 2-4 shows the Roline Laser Profiling System, all the sensors and the other components are connected to each other and placed in a container. This whole container is mounted on a test vehicle during profiling. The laser system is mounted in such a way, so as to keep track of the front left wheel path.

2.3 Software setup

The LMI Selcom provides a basic, console based C program to interact with the Roline Laser from the host machine such as a PC. This program is modified to read configuration details and spawn's another thread to initialize, control and perform data collection, using the DT9816. The Technical Report 0-6610-1 [21] provides a detailed description of these changes. Two figures describing the data flow and ADC configuration flow diagram from the Technical Report 0-6610-1[21] are shown.

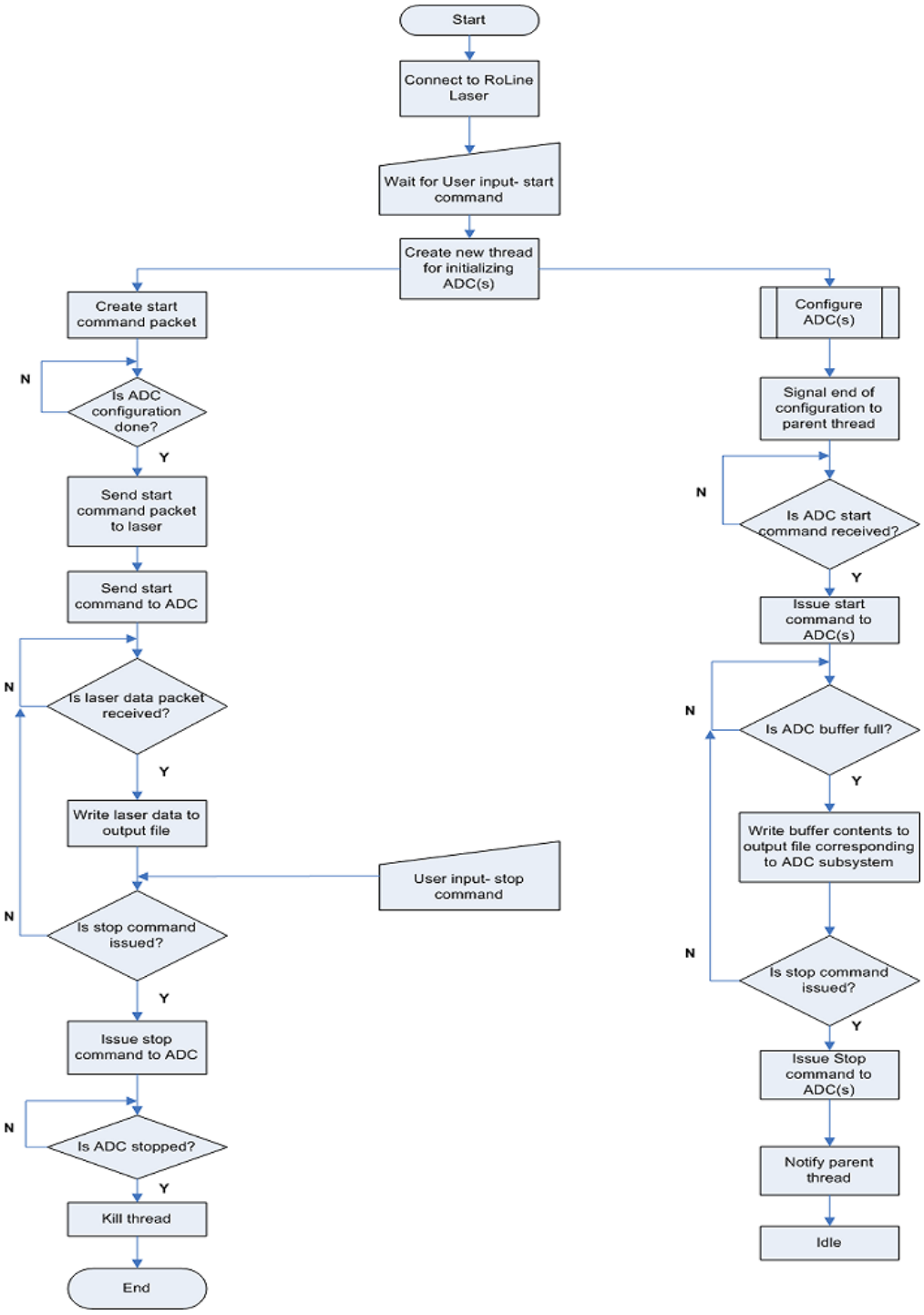

Figure 2-5 Flowchart for the Data Collection Process [21]

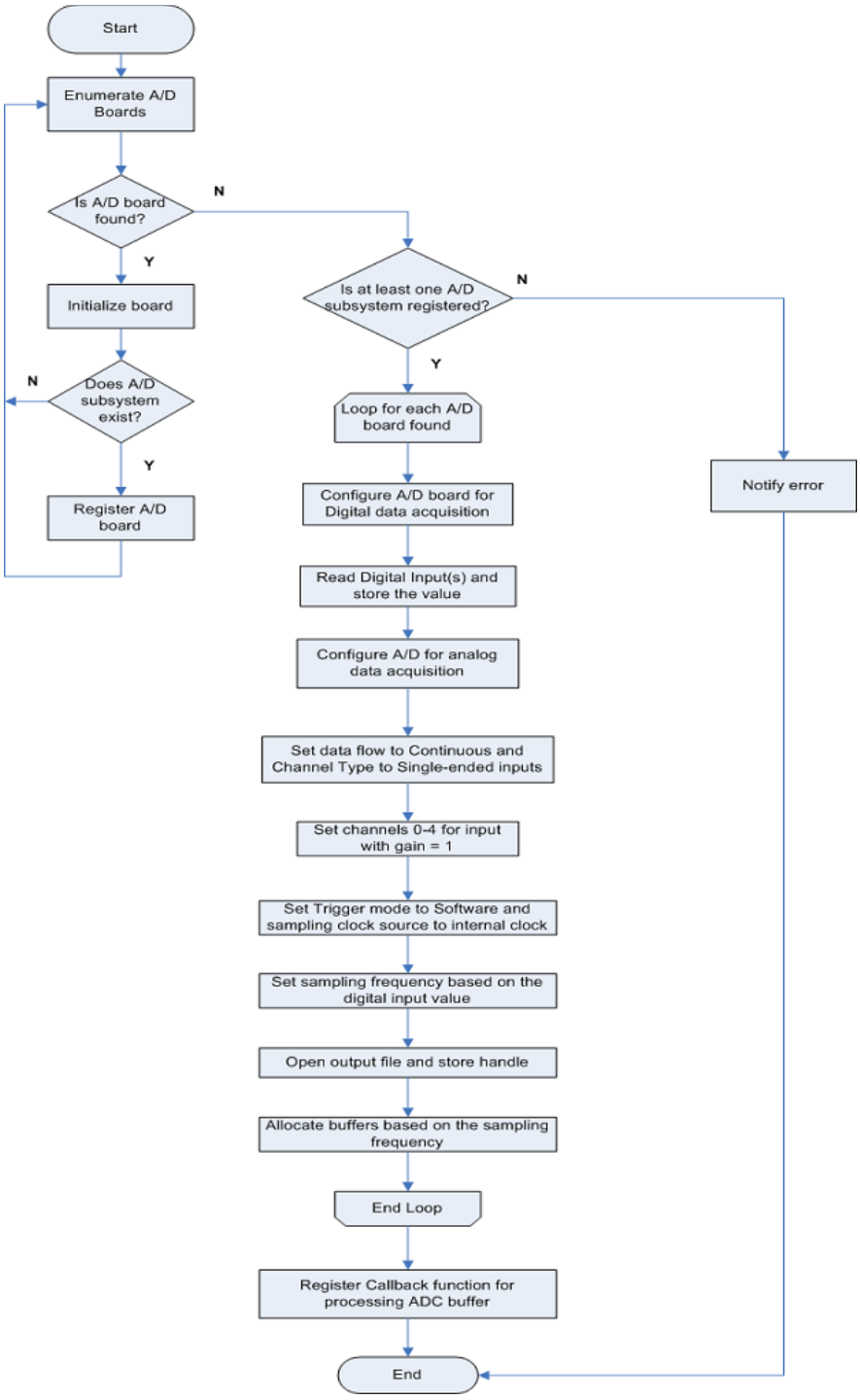

Figure 2-6 Flowchart for Configuring the ADC [21]

2.4 Tracking profile position using GPS.

Tracking the position of the profile values using the relative distance from the starting place alone is inefficient and prone to errors. The reflective adhesive tape which marks the start location can be moved, causing the profile values to point to wrong location on the surface of the road, hence associating the profile values more accurately with its geographical location is required. An easy approach to achieve this is to incorporate the GPS information along with the profile data.

2.4.1 GPS receiver

Figure 2-7 BU-353-S4 [22]

The Figure 2-7 shows the BU-353-S4, a USB GPS receiver. This compact form factor comes with a high sensitivity, low power driven chipset. It is software compatible with the Microsoft Windows OS. It follows the National Marine Electronics Association (NMEA) 0183 protocol format for delivering the GPS data [23].

The BU-353-S4 device in adherence to the NMEA format provides GPGGA and GPRMC tag information. These tags incorporate the latitude, longitude, altitude, heading, speed, timestamp and etc., information in it.

2.4.2 GPS Connection

The GPS receiver is connected to the USB port on the host PC and it provides data transfer over the serial port. The main thread initializes the serial port for reading the GPS module. Along with the ADC thread, the main thread spawns another thread called “fetch_gps_data()” to handle GPS data. It also initializes a queue for passing the GPS information to other threads.

The GPS thread, “fetch_gps_data()”, reads GPS information from the device in the NMEA format, and parses the GPGGA and GPRMC tags to acquire the latitude, longitude, altitude, heading, speed, and timestamp information. It passes this information in the form of a structure, to the ADC thread, by writing it into a queue. The ADC thread writes this GPS information along with the ADC sampled data to the text file.

Chapter 3

AN EMBEDDED APPROACH FOR ROAD PROFILING

Embedded devices in comparisons to the standard desktop computers have vastly improved in their performance. They can gather data from various sensors and compute complex operations on the gathered data in real time. The profile information is calculated by processing on gathered data. This chapter expands in detail on various features of the embedded devices available in the present market, and the selection process of a suitable one for implementing the Dual Laser Road Profiling System based on the specifications and resource availability. A new Dual Roline Laser based road profiling system has been developed by designing and implementing changes on the existing Single Roline Laser based Road Profiling System. The existing system was developed by researchers in the Transportation Instrumentation Lab.

There are three main components required for performing data acquisition in real time and processing the data to compute road profile. The components include:

1. Sensors.
2. An embedded processing unit, and
3. Analog to Digital Converter (ADC).

The existing system with Texas Department of Transportation (TxDOT) has a Windows based desktop which interfaces with the sensors and ADC to do data acquisition and profile calculation. This project is intended to replace the PC with a suitable embedded device capable enough of performing the required tasks and a compatible ADC that can be interfaced with the embedded device.

3.1 Sensors

Sensors needed for real time data acquisition are mainly dependent on the type of application it is intended to be used for. The existing road profiling system uses below listed sensors.

1. LMI Selcom's Roline laser 1130,
2. Accelerometer,
3. Start sensor,
4. Distance encoder.

The Dual Line Laser Road Profiling System includes all the sensors of its predecessor with several additions. The Roline laser and accelerometer modules has been increased to two, dedicating each for left and right wheel paths respectively. Detailed design of how all these different components in the dual laser system coexist in harmony will be discussed in detail in the future chapters.

3.2 Embedded processing units

In accordance with Moore's law, over past few decades, the number of transistors present in the integrated circuits has increased drastically over time. Keeping true with Moore's law semiconductor industry has produced lot of powerful multicore hand held embedded devices. This section considers some of the recently developed embedded devices that are compatible for use in Road Profiling system.

Several features like computational speed, compatibility of running different programs etc., are to be considered while selecting an embedded device. These features can be categorized as

1. Hardware features,
2. Software features
3. Miscellaneous features

3.2.1 Hardware features

The following are some of the features that fall under this category

1. Processor type.
2. System on Chip.(SoC)
3. Clock speed.
4. General Purpose Input Output pins (GPIO) capability.
5. Memory, size and type.
6. Peripherals supported.

3.2.2 Software features

The following features are the examples that fall under the software category

1. The under lying operating system the device supports
2. Development capability with respect to program like, support for execution of multi-threaded programs on the device etc.

3.2.3 Miscellaneous features

There are several additional features considered in the selection of the board. They are:

1. Cost of the device.
2. Manufacturer.
3. Release timeline.
4. Support from the manufacturer and online community for both hardware and software development on it.
5. Expected development time.
6. And physical features like dimensions etc.

The subsequent section lists the different embedded boards that are selected for consideration.

3.2.4 List of embedded devices to choose from.

Based on the selection features discussed earlier, six different embedded devices were selected for the feasibility study.

1. Galileo, an Intel® Quark SoC with 32 bit microcontroller.
2. BeagleBone Black, TI SoC with 32 bit ARM Cortex A8 microprocessor.
3. A10-OLinuXino-LIME, an AllWinner A1X SoC with 32 bit ARM Cortex A8 microprocessor.
4. Raspberry Pi a Broadcom SoC with 32 bit ARM 11 family microprocessor.
5. Altera DE0-Nano FPGA.
6. Edison, an Intel® Dual core Atom processor.

Figures of all these devices are shown below.

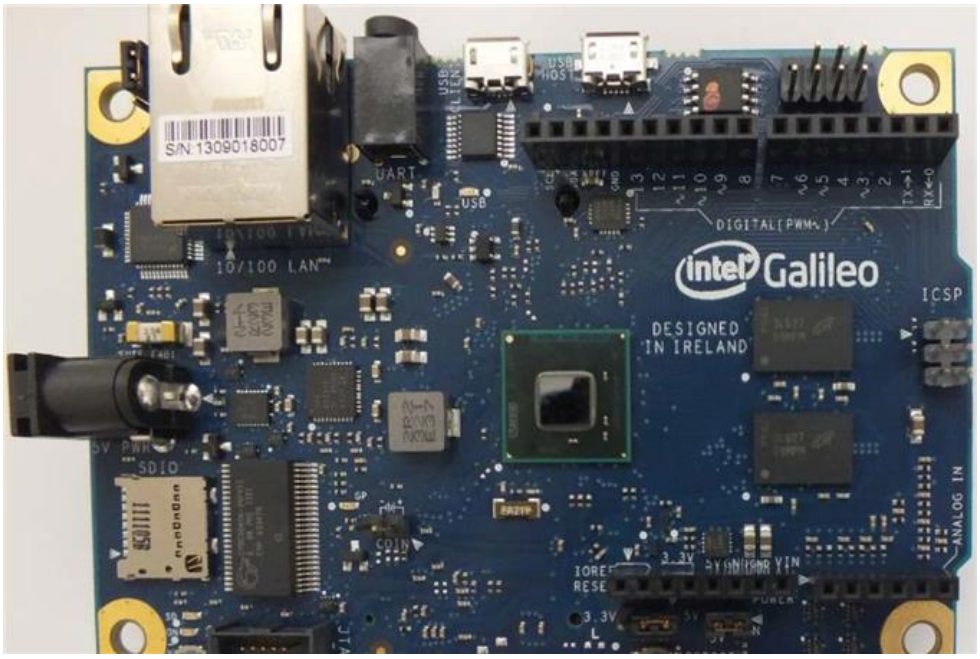

Figure 3-1 Intel® Galileo [6]

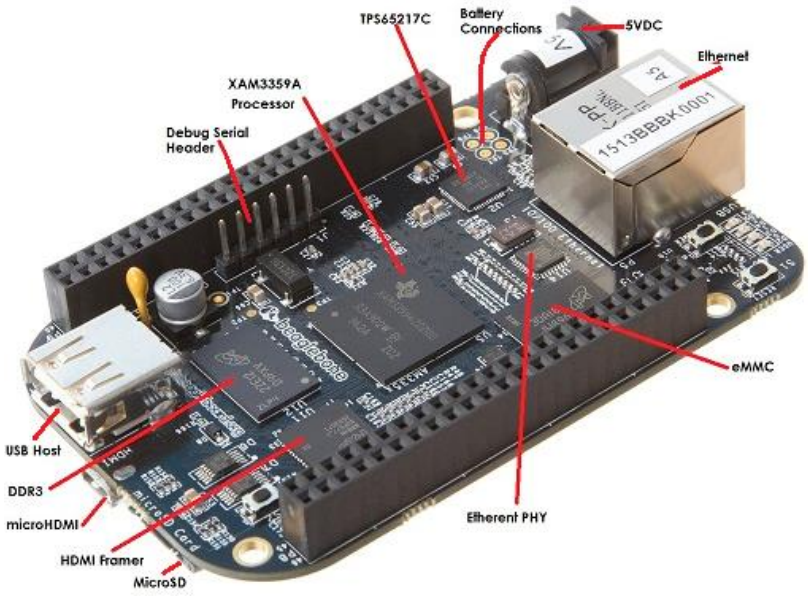

Figure 3-2 BeagleBone Black [27]

Figure 3-3 A10-OLinuXino-LIME

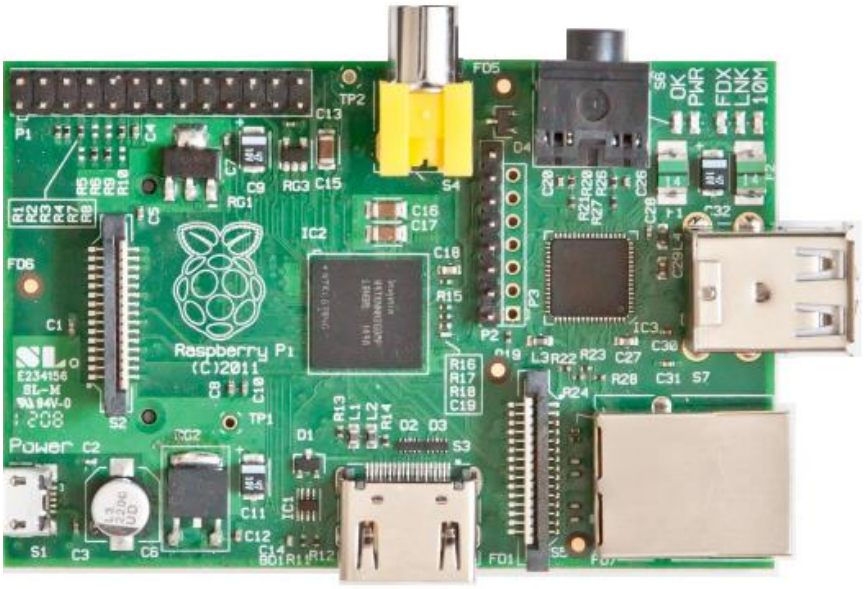

Figure 3-4 Raspberry Pi

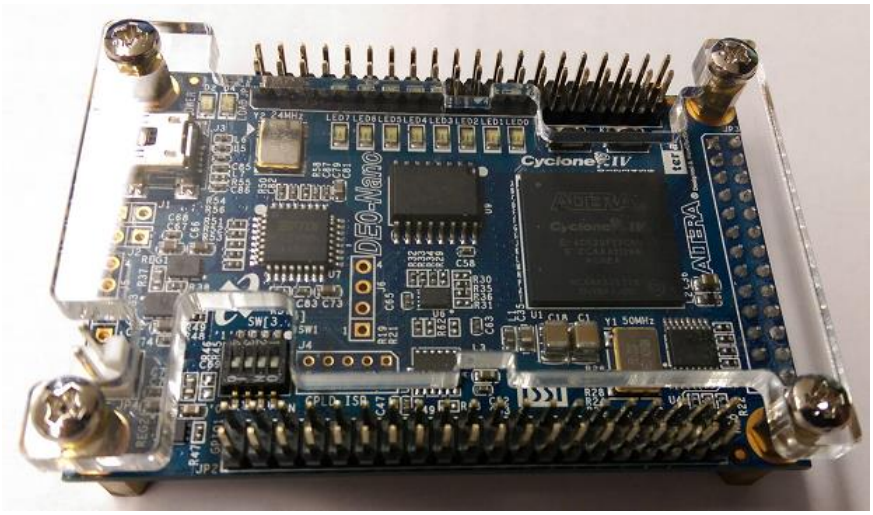

Figure 3-5 Altera DE0 NANO

The Altera DE0 NANO was not considered as the on board memory available is just 32MB of SDRAM with no SD card slot and Ethernet slot. Though it has around 40 GPIO pins, it is required to interface with a SD Card and Ethernet slot before other

development, hence the development time required to interface with the FPGA increases considerably.

The features of all the devices are compared in a comprehensible manner in the following table [27][6][3].

Table 3-1 Feature comparison among SoC's

Features	**Intel® Galileo**	**BeagleBone Black**	**A10-OLinuxXino LIME**	**Raspberry Pi**
Base Price	$70	$65	€30	$35
Released	October 2013	April 2013	December 2013	October 2012
Processor	Single Core Microcontroller	ARM Cortex A8	ARM Cortex A8	ARM1176JZFS
Clock Speed	400MHz	1GHz	1GHz	700 MHz
SoC	Intel® Quark SoC X1000	TI Sitara AM3359	AllWinner A1X	Broadcom BCM 2835
SoC Manufacturer	Intel® (USA)	TI (USA)	AllWinner Technologies (China)	Broadcom Rasberry Pi Foundation (UK)
GPIO capability	14	65	160	8
Ethernet	10/100 Mbps	10/100 Mbps	10/100 Mbps	10/100 Mbps
Size	4.2" x 2.8"	3.4" x 2.1"	3.4" x 2.1"	3.4" x 2.1"
Input Voltage	5V	5V/5V	5V/5V	5V/5V
RAM	256MB,DRAM	512MB,DDR3	512MB,DDR3	512MB,SDRAM

Table 3 1 —*Continued*

Flash	8Mb NOR, 512 KB embedded SRAM, MicroSD	2 GB on-board eMMC, MicroSD	Micro SD	SD / MMC
OS	Customized Linux 3.8.7 and Arduino	Angstrom (Default), Ubuntu, Android, ArchLinux, Gentoo, Minix, RISC OS,	Linux, Android	Raspbian (Recommended), Ubuntu, Android, ArchLinux, FreeBSD, Fedora, RISC OS
Power Rating	800mA @5V	460 mA @ 5V		350 mA @ 5V
Power Source	Supply	Supply/OTG-USB	Supply/OTG-USB	Supply/OTG-USB
Battery Support	Yes, Coin type	Optional Connectors provided	LiPo Battery connector with battery-charging capabilities	No
Peripherals	USB, UART, SPI, I2C	USB, UART, SPI, I2C, HDMI	USB, UART, SPI, I2C, HDMI	USB, UART,

Intel Edison was not available at the time of feasibility study and has been added at the end of the project into the report. The Edison includes a dual core Intel® Atom

processor based chip. It is quiet powerful and it exposes a lot of GPIO's for the sake of interfacing and can be considered for the future application.

Raspberry Pi has a less number of GPIOs for interfacing with ADC hence it is not considered in the design implementation.

Although the Intel® Galileo has less number of GPIO's and is less powerful in comparison with BeagleBone Black and A10, it out performs the two with following reasons.

1. It supports both Linux and Arduino based interfacing.
2. Interfacing using Arduino is very easy and can be used for hardware verification. This minimizes the validation time.
3. Intel® provides a Board Support Package (BSP) for development.
4. New to the market and the online community is very active.
5. Intel® is funding the devices for this project.

3.3 Analog to Digital Converters

A high resolution ADC with a large input voltage swing is important for Road Profiling systems. A desirable specification for an ADC to perform data acquisition would be to have

1. 16 bit resolution.
2. Input voltage range or swing of ±10 V peak to peak.
3. Sampling rate of at least 100 KHz.

The onboard ADC of Galileo, AD7298 has a low resolution of 12 bit and a small voltage range of 0V to 5V. These specifications are below the requirement.

The existing Roline Laser Road Profiling System uses DT9816 as the ADC to capture data from the sensors. It has high resolution and a very good voltage range, and

meets the specifications needed for the road profiling. Following are its specifications [25].

1. 16bit resolution.
2. Voltage range is from positive to negative10V peak to peak.
3. Supports 6 analog channels.
4. Sampling rate is up to 150 KHz.
5. Uses an internal clock for sampling, and it can be configured through program.

3.3.1 Need for a separate ADC.

The Data Translation DT9816 is interfaced to the PC through the USB and is supported on Windows operating systems. The Galileo, which runs a Linux kernel v3.8.7 doesn't support the DT9816 as it exposes only Windows API's for configurations. Dual Roline Laser System which intends to use Intel® Galileo for processing requires a new ADC with features similar to DT9816 but, should be supported on Linux kernel.

The Analog Device's AD7606 ADC meets the required specifications. The following are the features of AD7606: [24].

1. Supports 8 simultaneous sampled inputs.
2. Analog range of bipolar ±10 V and ±5 V peak to peak.
3. 16bit resolution
4. Sampling rate of 200 KSPS
5. Operates on a single 5V supply.
6. With a reference of 2.3V to 5V.
7. Linux device driver code is present in the main line, hence requires very minimum change to get it to work.

8. It supports serial, parallel, byte parallel operations.
9. The AD7606 can be connected in cascade with another AD7606 to extend the number of input channels from 8 to 16.
10. Has an active online community by the Analog Devices Company for support.

To conclude this chapter, in the design and implementation of a Dual Roline Laser System an embedded processing unit, the Intel® Galileo replaces the PC from the existing system. A compatible ADC like AD7606 replaces the DT9816. The next chapter discusses the hardware aspects of interfacing AD7606 with the Intel® Galileo.

Chapter 4

HARDWARE ASPECTS OF INTERFACING AD7606 WITH INTEL® GALILEO

Interfacing an ADC with an embedded processor is a two stage process. In the first stage, based on the hardware compatibility between the two, a circuit is built to support the A/D converter In the second stage a corresponding software program is designed and developed, that initializes all the components and provides a synchronous flow of control and data between the two units. The hardware details described in sections 4.1, to 4.3 discuss in detail the hardware features of Intel® Galileo and AD7606. Based on these hardware features, an optimal circuit diagram for interfacing the two has been designed and developed. The Chapter 5 deals in the software aspects of the system and discusses in detail, the operating system that is present on the Galileo, and different approaches in which the AD7606 can be configured to perform real time data acquisition.

The Roline lasers are connected through an Ethernet port from which bridge or free mode data is available. The Roline laser generates 3 KHz sync information that is collected by one of the channels of the ADC. Other than Ethernet port, through which the Galileo and Roline can exchange information, all other data from sensors like the Start, Distance Encoder and Accelerometer are collected through the remaining channels of ADC that has been interfaced with the Intel® Galileo.

4.1 Peripherals of Intel® Galileo

This section describes the hardware features of Galileo. Figure 4-1 shows the top view of the Intel® Galileo along with the peripheral it supports. Only the peripherals that are required for interfacing are being described here, details of the rest can be obtained from the user guide [1] of the board.

The following two figures shows key components and block diagram of Galileo.

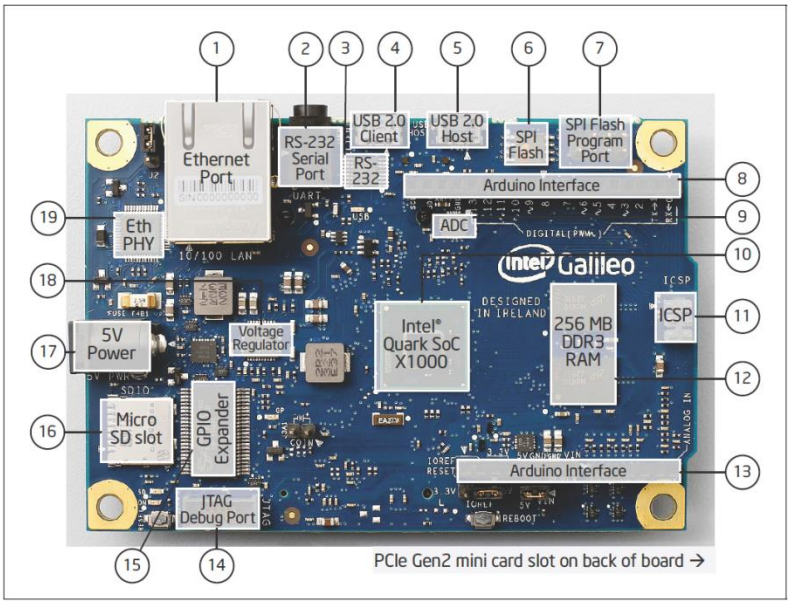

Figure 4-1 Intel® Galileo top view [1]

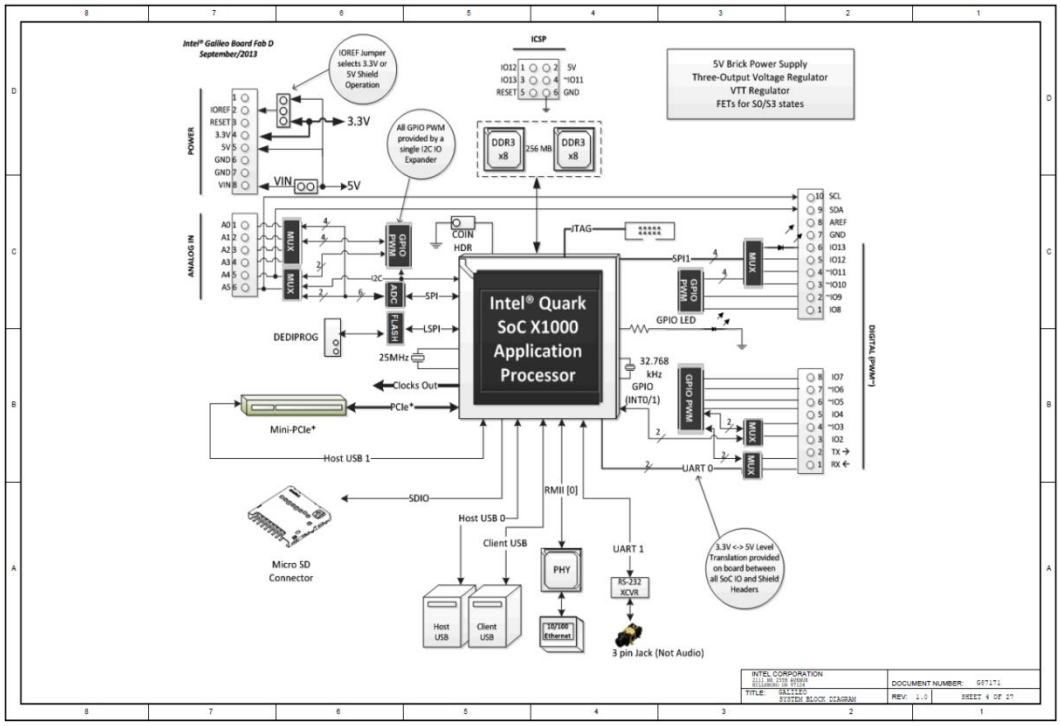

Figure 4-2 Intel® Galileo System block diagram [2]

From the figure 4-1 and figure 4-2 several key components that are being used for interfacing are described in the following sections.

4.1.1 Ethernet Port

This is indexed at (1) in the figure 4-1, It is used to interface the Roline Laser for collection of bridge or free mode data from it. This data in collaboration with the ADC data is used to calculate the profile information. In the case of Single Roline Laser System, the Roline Laser can be directly connected to the Galileo. In case of Dual Laser System, the Galileo it is connected to the Roline Laser modules through an Ethernet switch box.

4.1.2 Serial console

Either of the three, USB 2.0 Client indexed at (4) or RS-232 Serial port indexed at (2) or Ethernet port can be used for serial console access. An USB micro cable can be connected from USB 2.0 Client port to development PC, and serial console can be accessed through this. If RS232 Serial port is used, then an Audio to Serial converter in combination with USB to serial converter is needed to enable serial console. Finally serial logs can also be enabled over SSH connection. The development PC and Galileo needs to be present in the same subnet to establish SSH between the two easily. This can be done by connecting the development PC to the Galileo through the Ethernet switch box

4.1.3 Power

The Intel® Galileo needs a 5V DC power supply to operate. This is provided by connecting a 5V DC supply voltage to the Power slot indexed at (17).

4.1.4 Storage

Large amounts of data in order of gigabytes shall be collected during data acquisition. The on board flash memory of Intel® Galileo is very less, around 8

megabytes. To overcome this problem Micro SD Slot indexed at (16) in the figure 4-1 can be used to hold a micro SD up to 16 gigabytes in size. A 16 GB micro SD is used in this project to contain binaries and output the data. Binaries include, Linux kernel v3.8, the Data Collection program, and finally collected output data.

4.1.5 GPIO pins

By far, the GPIO pins are the most important part of interfacing. The Intel® Galileo exposes two sets of GPIO's. As shown in the figure 4-1, "Arduino Interface" indexed at (8) and (13) are the GPIO pins that are interfaced with the AD7606 for exchange of control and data between the AD7606.

GPIO's indexed at (13) exposes access to on board ADC, provides reference voltage used as V driver for AD7606 and common ground connection between the two.

GPIO's indexed at (8) exposes a total of 14 pins, that provides access mainly to I2C, SPI, UART, Interrupts and other general Input output. These GPIO's are all multiplexed and proper mux lines needs to be configured to access I2C, SPT, UART, or Interrupt features.

4.2 AD7606 Features and Initial setup

4.2.1 Functional Diagram

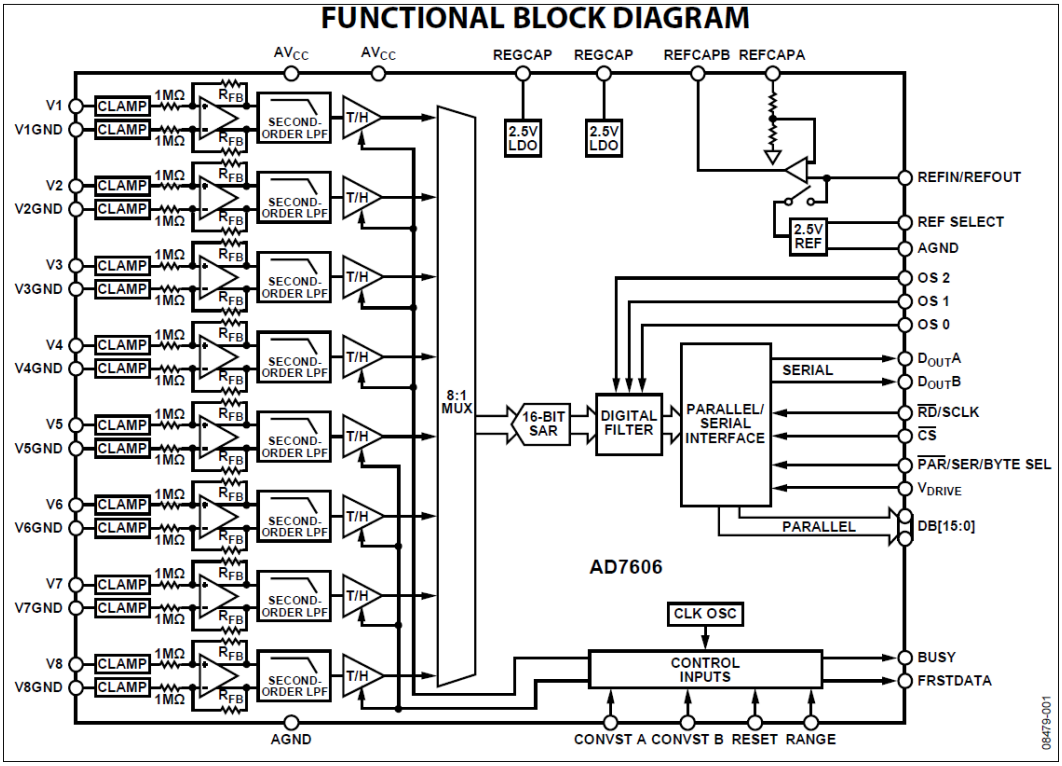

Figure 4-3 AD7606 Functional block diagram [24]

Figure 4-3 shows the functional block diagram of the AD7606. It is a 16-bit, simultaneous sampling, analog to digital data acquisition system. It supports up to eight channels. Each channel has an analog input clamp protection, a second-order antialiasing filter, a track and hold amplifier, 16-bit successive approximation ADC, a configurable digital filter, a reference voltage of 2.5V [24]. V1 to V8 are the analog inputs and upon conversion, sampled data is available either through high speed serial or parallel interfaces. Mode selectors have to be configured to choose between serial and parallel interfaces.

4.2.2 Pin diagram and functional description

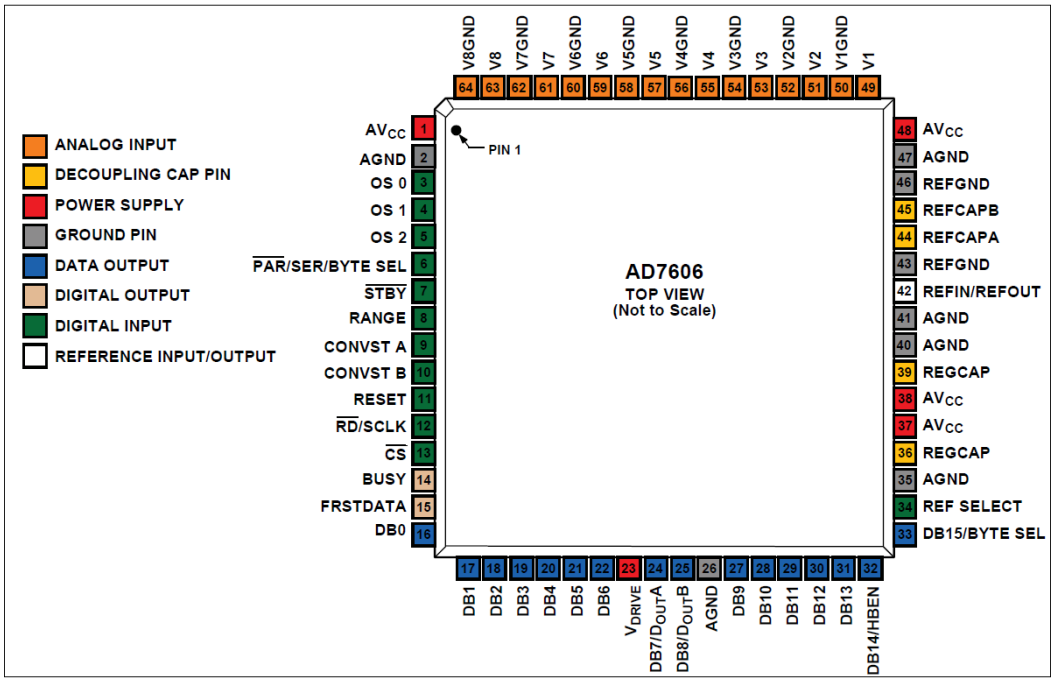

Figure 4-4 AD7606 Pin Configuration [24]

AD7606 is a 64 lead LQFP package based chip. The pin diagram of the AD7606 is shown in figure 4-4. All the pins of AD7606 can be categorized into eight different groups, based on the features, like Analog and Digital Inputs, Data and Digital Output, Power, Reference and decoupling capacitor pins. Following section provides a brief functional description of the important pins that are used. More detailed description of each pin is provided in the data sheet of AD7606, Table 6; p13 [24]

1. Analog Input: - Analog inputs are connected to Pins 49 to 64; each channel has 2 pins, source and ground respectively. In case any channel is not used, it has to be connected to ground.
2. Power: - Pins 1,37,38,48 are called AVcc, and takes 5V DC input. There are six ground pins called AGND, and all the six pins needs to be connected to ground plane of a system. The supply pins and ground pins

needs to be decoupled through capacitors as well. Pin 23 typically needs a V-Drive voltage (2.3 V to 5.25 V) from host interface. Intel ® Galileo delivers a 3.3V V-Drive for AD7606.

3. Digital Output: - Falling edge on the BUSY pin indicates the conversion process on all the channels is complete, and the sampled data should be available for reading in the output register after a certain time. A high on the BUSY indicates that a conversion is going on. FRSTDATA pin goes high when channel one is being read
4. Data Output: - Data can be read in three different modes based on which interfacing mode ADC is configured. They are parallel, serial, and byte parallel. Table 4-1 shows which pins are active in which modes.

Table 4-1 Data read pins in different modes

Interface Mode	**Data out Pins**
Parallel	DB0 to DB15
Serial	Dout A & Dout B i.e. DB7,DB8
Byte Parallel	DB0 to DB7

5. Digital Input: - These are the most important pins on the ADC; they configure the AD7606 to required mode of operation.

RESET: - Rising edge on this pin resets the ADC, upon power-on, the ADC must be reset by applying a pulse of at least 50 ns wide. Reset signal clears the data in the output register of ADC to zero and if ADC is performing any conversion, application of the reset signal will cause it to abort.

~PARALLEL/SERIAL/BYTE Interface selection Input. If this pin is tied to a logic low, parallel interface will be selected. If this pin is tied to logic high, it selects either of the

two modes; serial interface or Byte parallel mode. Selection between these two modes depends upon DB15 pin. If DB15 is tied low, AD7606 operates in serial mode; else it operates in Byte parallel mode.

Table 4-2 Serial, Parallel operations

Interface Mode	PAR/SER/BYTE select	DB15/BYTE select
Parallel	0	Parallel data output
Serial	1	0
Byte parallel	1	1

RANGE: - This pin indicates the input operating range of AD7606. A logical high on the pin will cause ADC to accept inputs between ±10 V range, and a low in this pin will accept an input between ±5 V range. If the ADC range is fixed and is not intended to vary, then this pin can be tied to either supply or to ground.

CONVST: - AD7606 has two Conversion start pins A and B, conversion on the first four channels can be started by CONVST A, and for channels five to eight by CONVST B. A transition of low to high causes the conversion to start. Both the conversion pins can be tied together and controlled by a single conversion signal from the host processor. In case of Dual Laser System, six channels on the ADC are utilized; hence both conversion start pins are be tied together.

~CS: - Chip select pin, this is an active low input pin, which is set to low whenever the host controller wants to perform sampling and read data.

~RD/SCLK: - An active low read signal is used during Parallel operation and for each pulse it will clock all the channel data out. In case of Serial communication, each period will clock one bit of data out. In the case of byte parallel, two clock periods will cause to read one channel data.

4.2.3 Typical circuit diagram

Figure 4-5 shows a typical circuit diagram of AD7606 configured in parallel mode, All the power supplies have a decoupling capacitor connected.

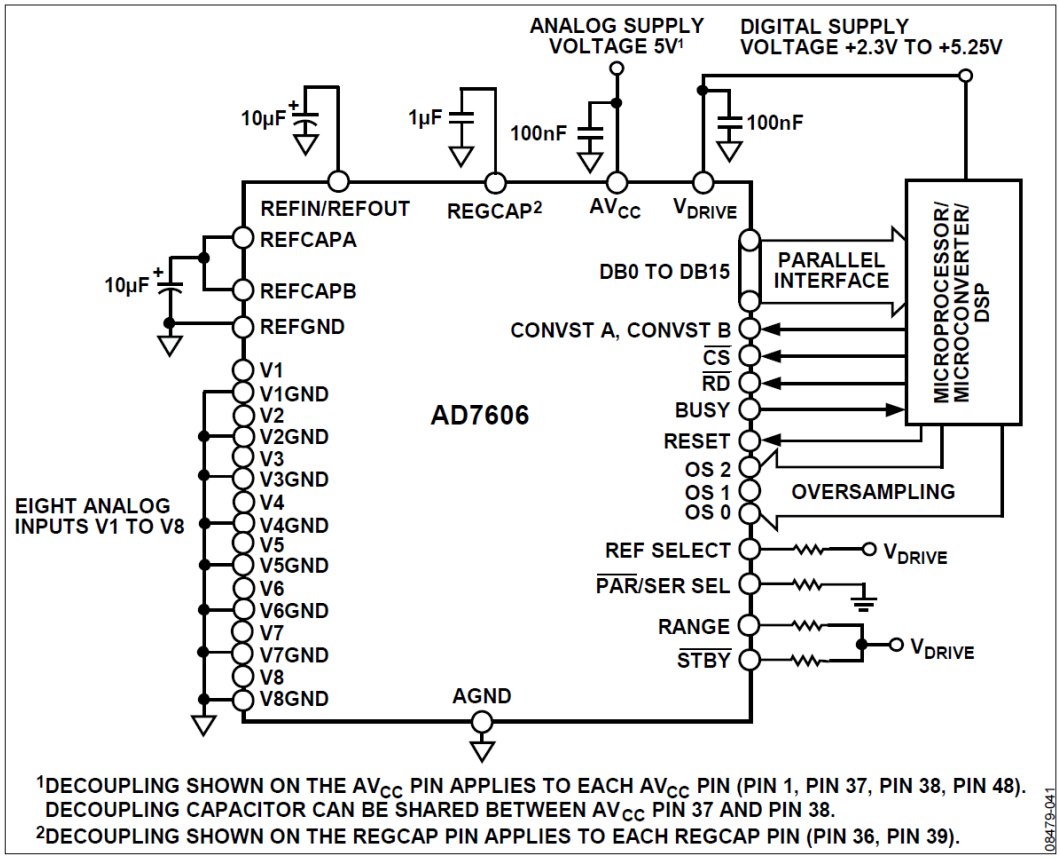

Figure 4-5 AD7606 Typical circuit diagram [24]

The figure 4-5 shows a generic microprocessor or a microcontroller being interfaced with the AD7606 in parallel operations. ADC has 8 analog inputs and any unused inputs must be grounded. A common drive voltage is provided between the ADC and host hardware. Sixteen pins are needed for reading data in parallel, Mandatory five GPIO pins, one each for start conversion, chip select, read/ clock, busy, and reset, signals are required, summing up the total number of GPIOs to twenty one. The

remaining digital input pins are tied either to logic low or to logic high, based on the required configuration. Digital filter can set to default value by hard wiring the oversampling pins. The parallel mode operation can be selected by setting the parallel/serial/byte parallel interface mode pin to ground. Range pin is set to high to select ±10 V input range.

Intel® Galileo has only fourteen GPIO pins exposed; hence parallel interface with the AD7606 would no longer be possible. To interface Galileo with the ADC in Serial mode operations, apart from the five mandatory control pins (Start conversion, chip-select, read/clock, busy, and reset) an additional pin for reading the sampled data is required. This puts the total number of GPIO pins to six, which is possible as there are several unused pins.

For interfacing ADC in byte parallel mode, the number of GPIO pins required will be thirteen. Five mandatory control pins as stated earlier and eight data pins from DB0 to DB7. DB15 pin should be tied to logic high, in order to select byte parallel mode of operations. This configuration cuts it close, as the Galileo has only fourteen GPIO's, but still the Galileo could be interfaced with AD7606 in byte parallel mode.

4.2.4 Timing diagrams

This section describes the timing diagrams of the AD7606 during parallel, byte parallel and serial mode. It also describes the timing of conversation.

1. Conversion: - Figure 4-6 shows timing waveforms of AD7606 during initial power-on sequence and during sampling. RESET pulse (t^{RESET}) is applied during power-on; this typically is at least 50ns. After the falling edge of reset signal has been detected, start Conversion (CONVST), can be applied. Raising edge of CONVST signal starts the conversion

process, the ADC is busy during conversion and it is indicated by the rising edge of the BUSY signal. Upon successful completion of sampling, falling edge of the BUSY signal can be observed. For reading the data, chip has to be selected after the falling edge of the BUSY signal; t^{CONV} in the figure 4-6 indicates the total time required for sampling and latching all eight channel values. t^{CYCLE} shown the time delay between two successive conversions. For detailed timing delays required for programming, please refer to the AD7606 data sheet [24].

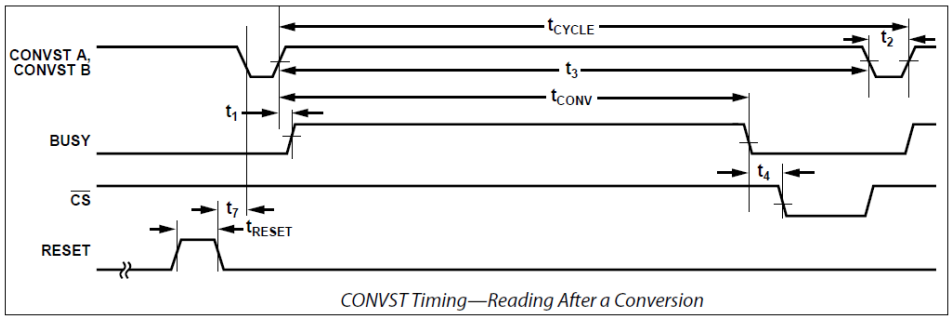

Figure 4-6 AD7606 Conversion timing diagram [24]

2. Parallel mode: - Figure 4-7 shows the timing diagram of AD7606 during parallel mode read operation. After completion of the sampling, the Chip-Select is pulled low; for each clock cycle that is applied on the READ pin, the ADC will clock out one channel data. Eight clocks are needed to read all eight channels. While reading the first channel V1, high pulse is observed on the FRSTDATA pin. This mode of operation is the fastest mode supported by the AD7606.

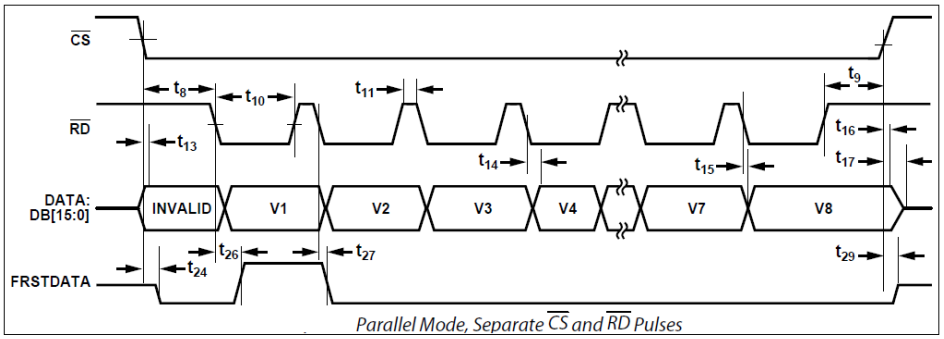

Figure 4-7 AD7606 Parallel mode timing diagram [24]

3. Byte Parallel mode: - In byte parallel mode as well, reading of data is done by driving the Chip-Select pin low after the detection of falling edge of the BUSY signal. For each clock cycle applied on the READ pin, the ADC will read one byte of information, hence sixteen bits of data per channel is read for every two clock cycles. Byte parallel mode takes sixteen clock cycles to read all the eight channel data. From the figure 4-8, high pulse on the FRSTDATA is observed for two clock cycles of the READ signal while reading channel one.

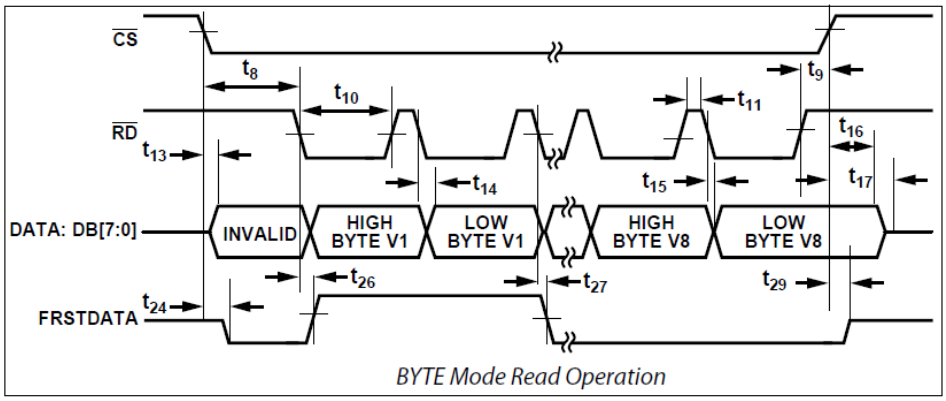

Figure 4-8 AD7606 Byte parallel mode timing diagram [24]

4. Serial mode: - Figure 4-9 shows serial mode operation of just one channel. Chip-Select is made low at the start of read operation and every clock of SCLK will clock just one bit information. Hence it takes sixteen clocks to read a channel and 128 clock cycles to read all eight channels. Like the other modes, FRSTDATA will be held high by the AD7606 till first channel data is clocked. Galileo supports Serial transfer rate of up to 25 MHz, so there should not be any problem even if AD7606 is interfaced with Galileo in serial mode.

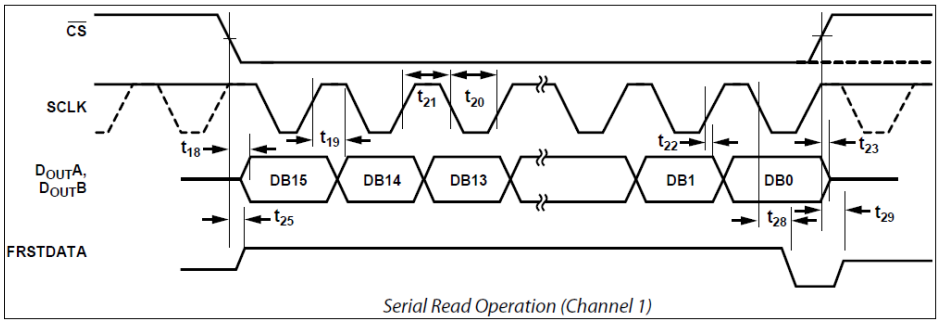

Figure 4-9 AD7606 Serial mode timing diagram [24]

4.3 EVAL-AD7606

AD7606 chip is a 64-lead Low Profile Quad Flat Package (LQFP). The leads on AD7606 are 0.5 mm pitch length. Figure 4-10 shows AD7606 and AD421 voltage regulator chip used for providing external reference voltage for AD7606. The area of the AD7606 chip is around 1 square cm and it is smaller than the one Dime coin which is shown along with it in the figure 4-10 for comparison of size.

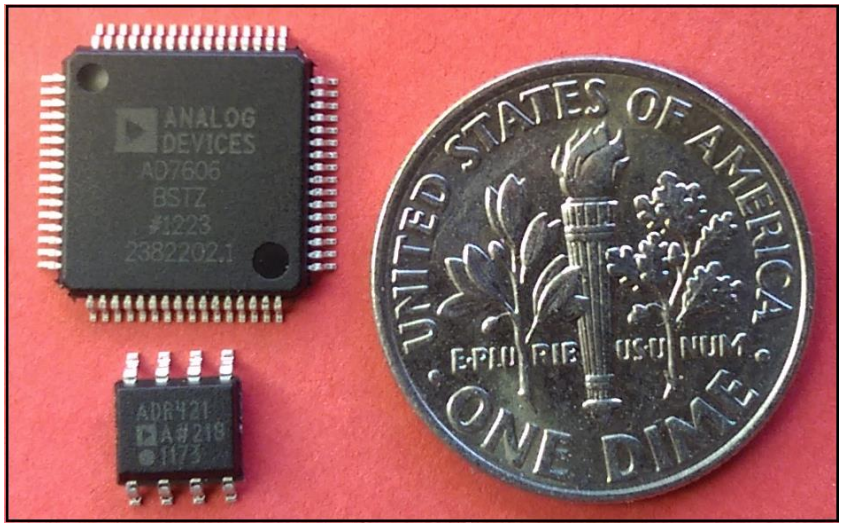

Figure 4-10 AD7606 LQFP, with 0.5 mm pitch, and an one dime for reference

Soldering these LQFP chips on a general purpose printed circuit board (PCB), which supports DIP based components, is not possible due to the difference in their pitch lengths. A custom PCB should be used to convert from LQFP to DIP interface.

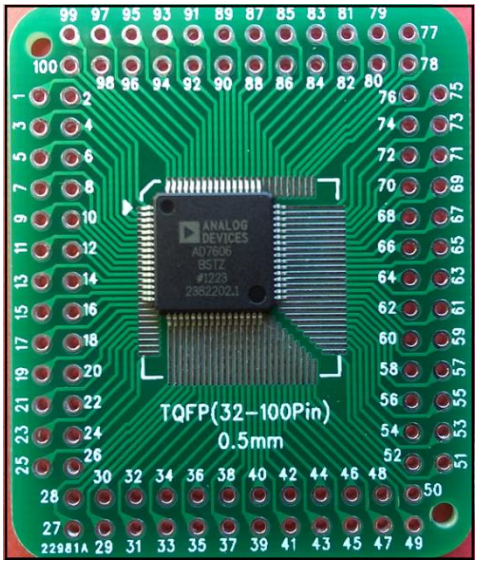

Figure 4-11 AD7606 chip placed on a custom LQFP to DIP PCB

Figure 4-11 shows the AD7606 chip placed on a customized LQFP to Dual In-line package (DIP) converter PCB. This custom PCB supports a range of LQFP packages varying from 32 leads to 100 leads of 0.5mm pitch. LQFP chips can be soldered on these converter PCB's, and DIP leads can be connected to the ends marked with pin numbers. Care must be taken to properly associate the leads of the chips to the pin numbers on the PCB.

Leads of AD7606 LQFP are quiet small, 0.5mm in length to be precise, and manually soldering these leads is a daunting task and prone to errors. Most common errors are shorting of leads. Other error that is visible while manually soldering on general purpose PCB's is introduction of noise in the circuit, which causes undesirable effects in the output.

4.3.1 Functional Block

To overcome these problems, EVAL-AD7606 has been used in this project. EVAL-AD7606 is developed by Analog Devices Inc., it is a fully featured evaluation board used for development purpose with the AD7606. The EVAL-AD7606 board is used generally in conjunction with the EVAL_CED1Z board but, it can work as a standalone unit as well. It has an on-board optional voltage reference and provides different ways to interface.

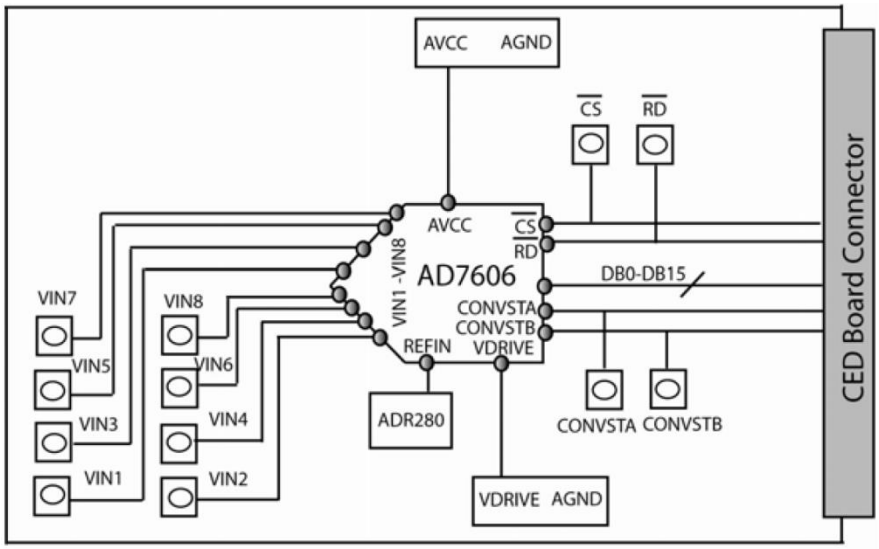

Figure 4-12 EVAL-AD7606 functional block diagram [24]

The Figure 4-12 shows the functional block diagram of EVAL-AD7606. The eight analog channel inputs are exposed through the SubMiniature version B (SMB) connectors mounted on the board. All the digital inputs and outputs and data outputs from the AD7606 chip are exposed through a 96 way connector called CED Board Connector, present at the end of the board. Power and ground lines are exposed on 96 way connector as well. This 96 way connector is used to interface the EVAL-AD7606 board with EVAL-CED1Z board available with Analog Devices Inc. Apart from the 96 way connector for interface, several test points and subminiature BNC (Bayonet Neill-Concelman) sockets are mounted on the board which expose the digital inputs and outputs. Power and ground lines can be provided through additional header connectors mounted on the EVAL board. PC software provided by Analog Devices Inc., can be used to control and perform data analysis when the EVAL-AD7606 is connected with EVAL-CED1Z board. More details can be found by going over the EVAL-AD7606 Preliminary Technical Data (PTD) sheet [24].

4.3.2 Serial mode

PCB design of the EVAL-AD7606 board consists of four layers with components present on top and bottom most layers. The default circuit connection of EVAL-AD7606 is to operate in parallel mode. For this project, the AD7606 needs to operate in either serial or byte parallel mode and in order to get the EVAL-AD7606 board to operate in either of the modes, some of the components and input settings need to be changed. Link Option Function present in Table 1 of EVAL-AD7606 data sheet [24] describes all the configuration settings the EVAL-AD7606 board can be configured with.

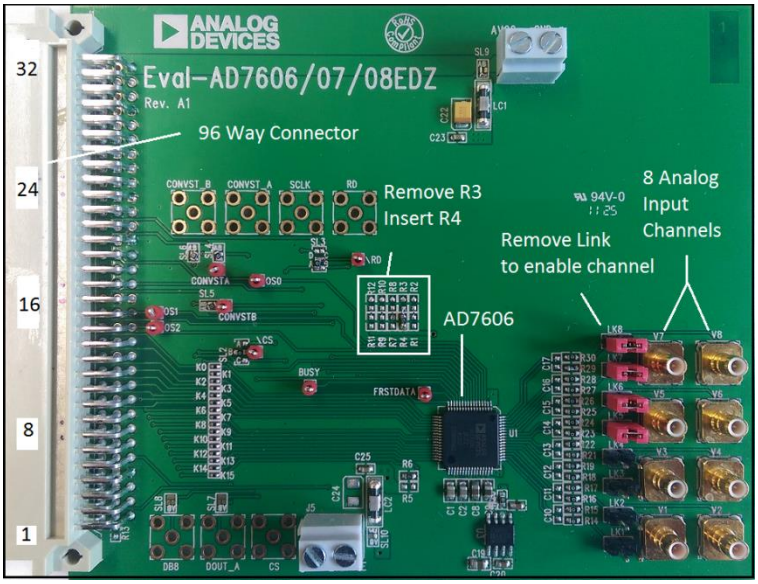

Figure 4-13 EVAL-AD7606 Board [24]

As indicated in the Table 1[24], in order to set the board in serial or byte-parallel mode, resistor R3 should be removed and resistor R4 should be inserted. This drives the ~PAR/SER select pin on AD7606 high, thereby causing AD7606 to operate in serial mode. Figure 4-13 shows the EVAL-AD7606 board and location of R3, R4, 96 way connector, and place to connect the analog inputs, and shorting red jumpers that needs

to be removed to enable the analog channels. The ~PAR/SER select is the only pin that is not connected to the 96 way connector, and hence needs a physical change on the EVAL-AD7606 board, rest of the pins are all connected to 96 way connector and can be either configured through software by driving the port to the required voltage level or hard wiring in on an external interface circuit.

4.3.3 A 96-Way connector

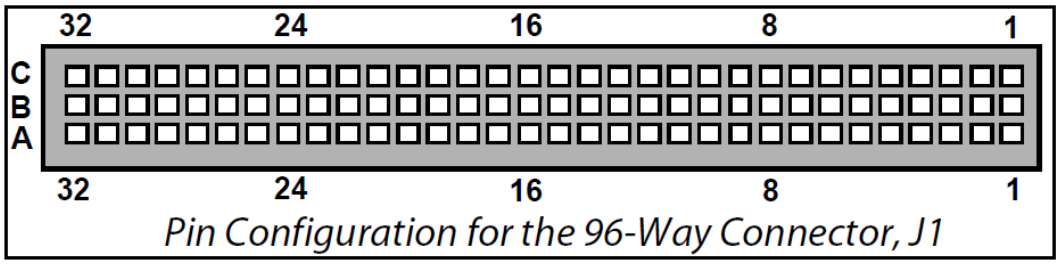

Figure 4-14 EVAL-AD7606 96-Way Connector Top View [24]

Figure 4-14 shows the pin configuration of 96-Way connector when EVAL-AD7606 board is viewed from the top. The 96 pins are arranged as 3 rows of 32 pins each. The bottom most row being row-A, following it is row-B, and finally row-C at the top. The 96-Way connector, J1 is used for the digital interface and power supply connection.[24]. Following figure 4-15 shows the association of each pin. This information has been inferred by referring the EVAL-AD7606 PTD [24]. Digital ground (DGND) and analog ground (AGND) are both tied together to form a common ground. Range pin on J1C11 needs to be driven high to enable $\pm$ 10 V. V^{drive} voltage of 3.3v is provided by Intel® Galileo. Ground of Intel® Galileo should be tied together with the ADC ground.

J1: 96-Way Connector Pin Function			
PIN	**Row A**	**Row B**	**Row C**
1		RESET	
2		DB0	
3		DB1	
4	DGND	DGND	DGND
5		DB2	
6		DB3	
7		DB4	
8	VDRIVE (+ 3.3 V)	VDRIVE (+ 3.3 V)	VDRIVE (+ 3.3 V)
9	~RD / SCLK	DB5	
10		DB6	~CS
11	AGND	DB7 / DOUT A	RANGE
12	DGND	DGND	DGND
13		DB8 / DOUT B	
14		DB9	OS2
15	~STBY	DB10	OS1
16	DGND	DGND	DGND
17	CONVST-A & B	DB11	BUSY
18	DB12	DB13	DB14
19		OS0	DB15
20	DGND	DGND	DGND
21	AGND	AGND	AGND
22	AGND	AGND	AGND
23	AGND	AGND	AGND
24	AGND	AGND	AGND
25	AGND	AGND	AGND
26	AGND	AGND	AGND
27		AGND	
28		AGND	
29	AGND	AGND	AGND
30		AGND	
31			
32	AVCC(+5 V)	AVCC(+5 V)	AVCC(+5 V)

Figure 4-15 EVAL-AD7606 96-Way Connector pin details [24]

4.4 AD7606-Interface Board

The AD7606-Interface board is used to interface the EVAL-AD7606 board with any embedded board that has at least six programmable GPIO pins. The figure 4-16 shows a block diagram of the AD7606-Interface board. It consists of four key components.

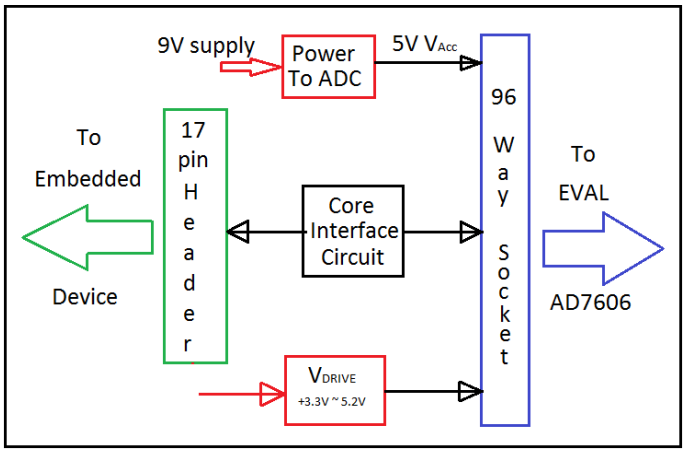

Figure 4-16 Block diagram of AD7606-Interface board.

1. 96-Way Socket.
2. Header Interface 17 pin.
3. Core Interface Circuitry.
4. Power supply.

4.4.1 A 96-Way Socket

The 96-Way socket connects with the 96-Way connector present on EVAL-AD7606 board. All the signals to and from the AD7606 are transmitted through this interconnect. The details of all these 96 pins are shown in figure 4-15.

4.4.2 Header Interface 17 pin

This consists of 17 pins that carry digital control information to and from AD7606 and, serial and byte parallel data output from the ADC. Pin 1 and Pins 11 to 17 consists of digital control signals. These are used to configure the AD7606 from Intel® Galileo or any other embedded controller. Pins 13 to 15 are mainly used to enable the optional digital present onboard the AD7606. Rest of the control pins is mandatory. Pins 2 to 9 are data out lines. In serial mode only pin 9 is used, in byte parallel all the eight data lines are used. Figure 4-17 Shows the pin functions of this header. From figure it could be made out that Pin 10 is not connected. It is a reserve pin and is intended to control the Range pin behavior from software in future.

Header Interface	
PIN	FUNCTION
1	RESET
2	DB0
3	DB1
4	DB2
5	DB3
6	DB4
7	DB5
8	DB6
9	DB7/DoutA
10	NC
11	~CS
12	SCLK/~RD
13	OS2
14	OS1
15	OS0
16	CONVST
17	BUSY

Figure 4-17 Header interface 17 pin function description

4.4.3 Core Interface Circuitry

The circuitry in this module extends from 96-Way socket to the 17 pin Header Interface. This module is mainly responsible for providing communication between the Intel® Galileo, connected to the 17 pin header interface (H1) and EVAL-AD7606 board which is connected to the 96-Way socket (J1) mounted on the AD7606 Interface board. The header interface H1 consists of a 17x2 breakout header mounting (refer to Figure 4-18). The first column that is away from the 96-Way Socket consists of ground and test points. Second column that is closer to the 96-Way Socket consists of signal from ADC. Each pin, 2 to 8 must be connected to ground by inserting jumpers during the serial mode. Pin 13 to 15 can be connected to ground through jumpers if they are not used. Column one of the remaining pins 1,9,11,12,16,17 acts as test points. These pins can be used to probe the signal during debugging. The header H3 and H4 are mainly responsible for changing from serial mode of operation to byte parallel mode of operation. The figures 4-18 and 4-19 show the location of H3 and H4. The board shown in Figure 4-18 is configured to operate in serial mode. To configure Byte Parallel mode, shift the jumper on H3 and insert a jumper on H4. Remove the jumpers from Pin 2 to 8on H1.

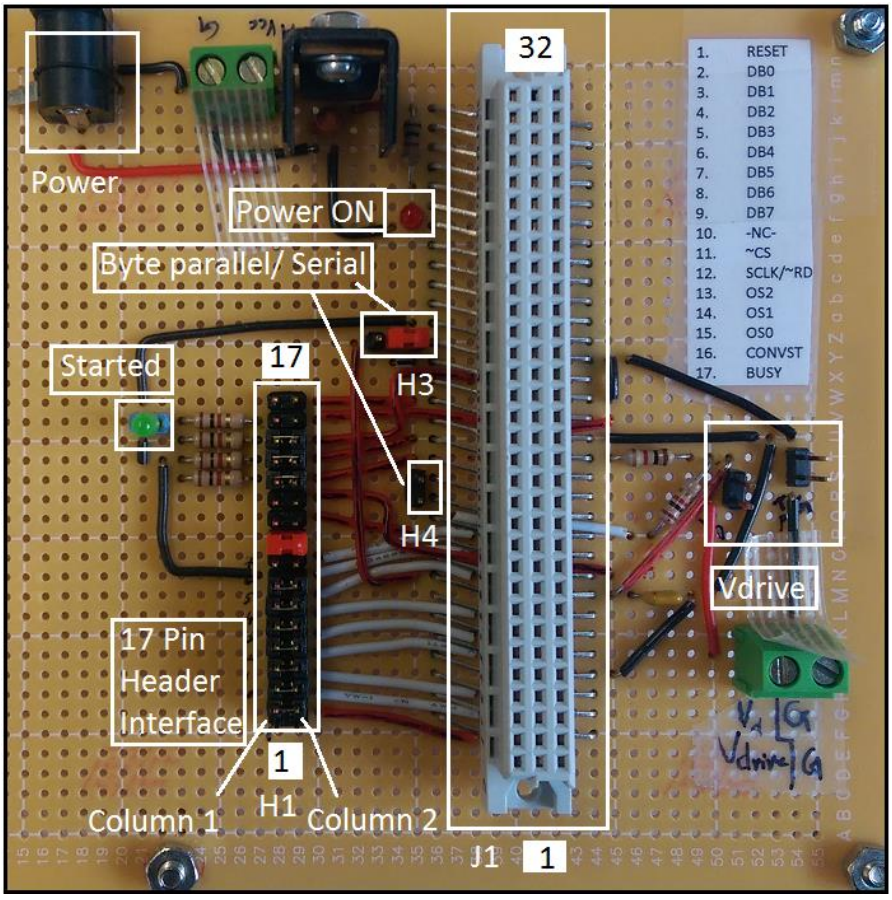

Figure 4-18 AD7606-Interface circuitry.

There are two indicators available to notify the status of the board. One is a power on red LED indicator another is a green LED to indicate the conversion process. The Figure 4-19 shows the schematic of the AD7606-Interface board.

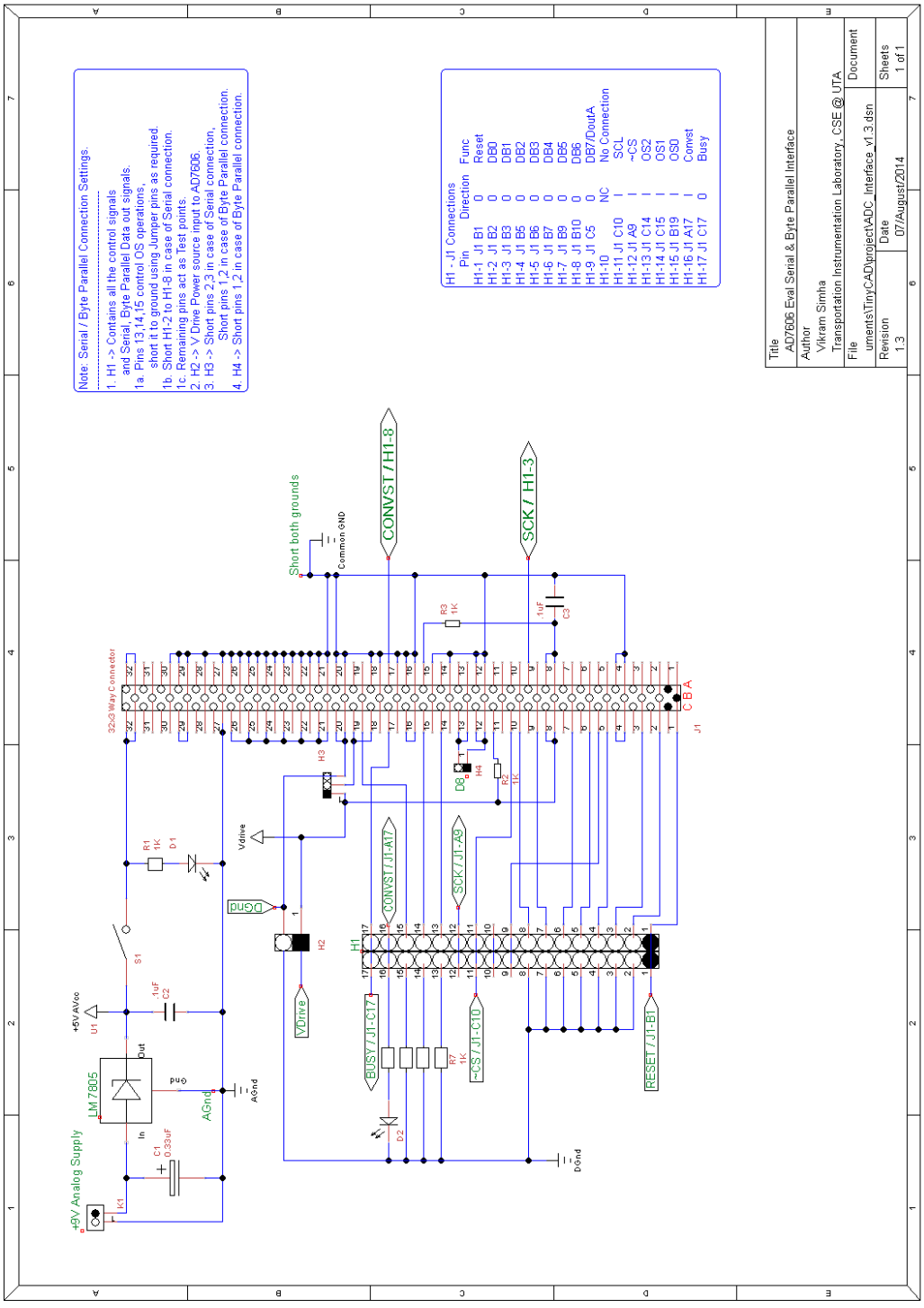

Figure 4-19 Schematic of AD7606-Interface

4.4.4 Power supply

An input voltage of at least +7V to 9V is applied to the voltage regulator (LM8705) circuit. It provides a steady voltage of +5V (V^{ACC}) to the AD7606. A red LED indicator is provided to show if the ADC is powered on or not.

The V^{DRIVE} voltage is provided by the Intel® Galileo and it's applied through the header H2. It's important to have a common ground between the ADC and Galileo.

Shows the figure of EVAL-AD7606 mounted on AD7606-Interface board.

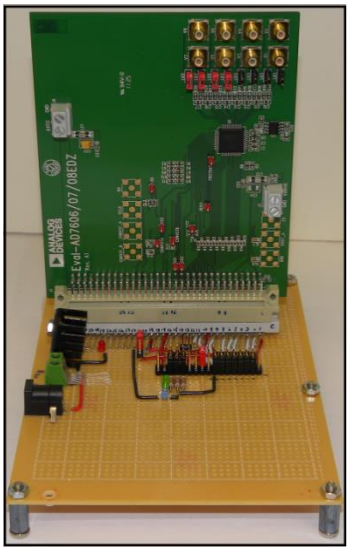

Figure 4-20 EVAL-AD7606 mounted on AD7606-Interface board.

4.5 AD7606 – Galileo Interconnect

The final phase in the hardware section is to interface the Intel® Galileo with the AD7606 Interface board. The system is configured to run in serial mode with no oversampling. For serial communication data transfer is possible through the use of Motorola's Serial Peripheral Interface protocol (SPI). Galileo supports two SPI devices, SPI-0 and SPI-1, and can act as SPI master only. SPI-0 is connected to onboard ADC (AD7298) and can't be reconfigured. SPI-1 is available for use and is exposed on GPIO

lines IO10 through IO13 on the Arduino interface. A total number of six GPIO pins are required to establish communication between the two modules. They are as follows:

1. Reset.
2. Busy.
3. Start Conversion (CONVST)
4. SPI Chip-Select (~CS).
5. SPI Serial clock (SCLK).
6. SPI Master in Slave out (MISO).

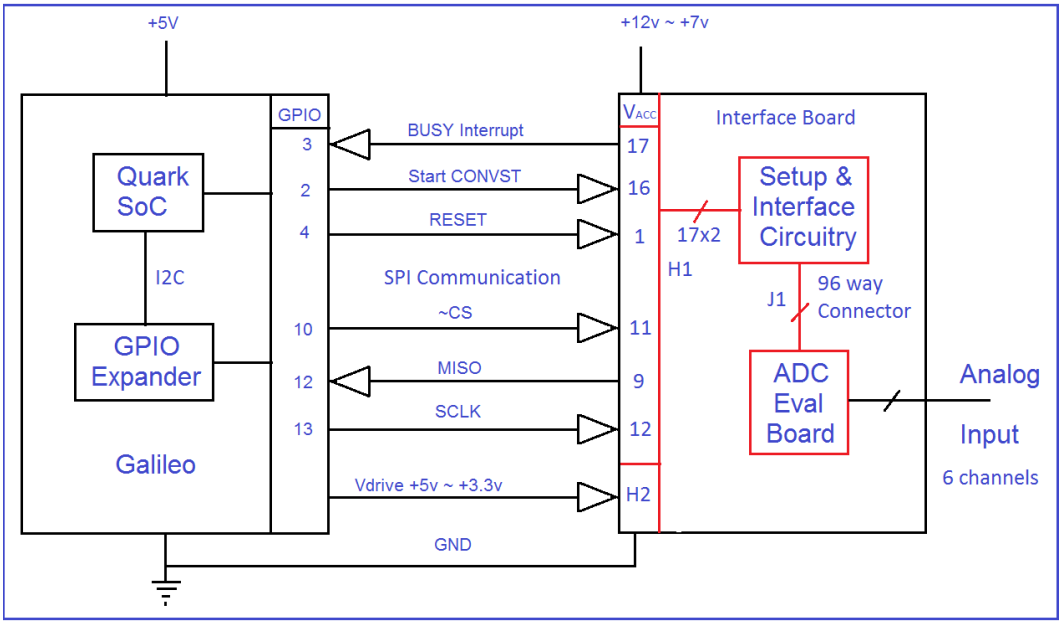

Figure 4-21 Intel® Galileo interfaced with AD7606-Interface board

The Figure 4-21 shows the pin level block diagram of Intel® Galileo interfaced with AD7606-Interface board. Reset and Start Conversion are inputs to ADC, whereas Busy is an interrupt generated by the ADC and is an input to Galileo. In the SPI section, ~Chip Select (~CS) and Serial clock (SCLK) are inputs to the ADC, and the sampled data can be collected from the ADC through Master in Slave out (MISO) line.

Table 4-3 Galileo-AD606 GPIO pin interface.

Signal	Galileo GPIO [4]	AD7606-Interface (H1)
Start Conversion	IO2	16
Busy (Interrupt)	IO3	17
Reset	IO4	1
~CS	IO10	11
MISO	IO12	9
SCLK	IO13	12

GPIOs IO2, IO3 are the only two pins that are directly connected to Quark SoC and are capable of receiving an Interrupt; hence the Busy signal must be con nected to one of the two IO lines, in this case, IO3. The Start Conversion signal must also be connected directly to Quark SoC, hence connected to IO2. The reasons for this shall be covered in later part of the next chapter (software section). The Figure 4-22 shows Intel® Galileo mounted on AD7606-Interface board with all the wirings.

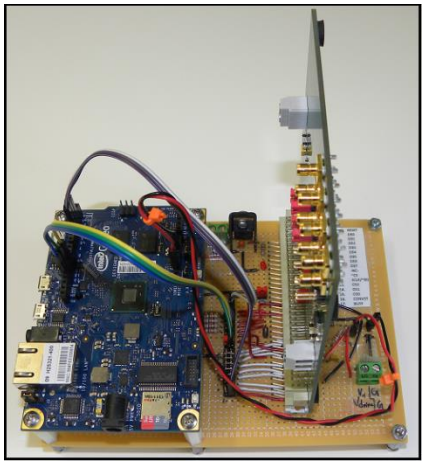

Figure 4-22 Intel® Galileo mounted on AD7606-Interface board

Chapter 5

SOFTWARE ASPECTS OF INTERFACING AD7606 WITH INTEL® GALILEO

This section designs and develops the software, required to perform data acquisition by using the hardware built from the descriptions given in chapter 4. Intel® Galileo comes pre-installed with a tiny Linux Kernel v3.8 and a customized Arduino Software Development Environment. It supports programs written in Python, C, and Arduino sketches. The user can customize and build the Linux Kernel v3.8 to enable extensive features of GNU C libraries, add custom device drivers etc. This section describes two approaches to program the system.

1. Arduino Sketch to perform data collection.
2. AD7606 Linux driver and user space Multi-threaded C program method.

5.1 The Intel® Galileo as an Arduino device.

5.1.1 Arduino

Arduino is an embedded computing unit that can interact with the physical world with ease in comparison to a PC unit like laptop. “It’s an open-source physical computing platform based on a simple microcontroller board, and an integrated development environment for writing software for the board.”[8].

Programs developed using Arduino IDE software are called “sketches”, and it can read inputs from various sensors; perform required computation on the input data, and deliver the output results. Additional hardware peripherals like Ethernet interface, SD card interface, etc., can be interfaced with predefined standardized Arduino IO pin headers, these interfaces are typically known as shields.

5.1.2 Galileo

The Intel® Galileo is the first board from Intel® that is pin compatible in both aspects, hardware and software wise, with the shields designed for Arduino UNO R3 [6]. The operating system that runs on the Galileo supports Arduino Software Development Environment. This makes writing sketches and executing them on the Galileo very easy.

The Intel® Galileo has a legacy SPI flash of 8 Mbyte that stores firmware (bootloader) and the latest sketch. Memory around 256 Kbytes to 512 Kbytes is dedicated for storing the sketch. Sketches that are larger than this can be stored on the optional micro SD card, which offers up to 32 GBytes storage space.

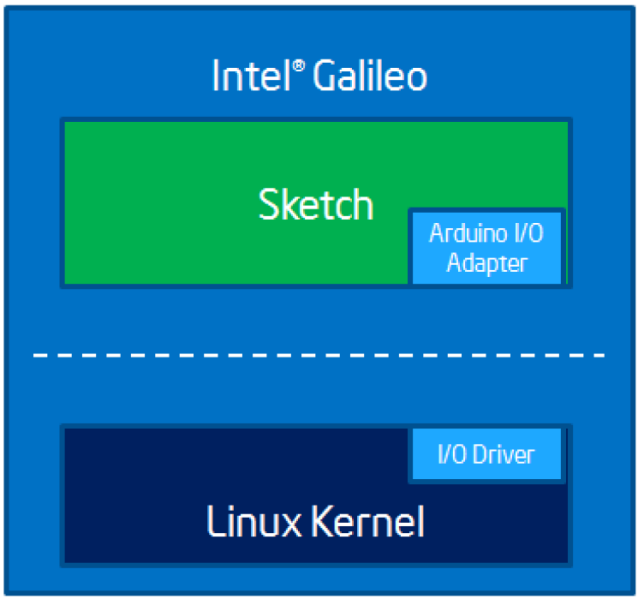

Figure 5-1 Intel® Galileo running an Arduino adapter [6]

Intel® Galileo development team has released customized version of Arduino IDE software that can generate sketches, which are compatible with the Galileo. The IDE is available on three different platforms, Windows, Linux, and MAC. Refer [9] for

downloading and installation instructions. Galileo board can be started (processing called boot strapping) by connecting the power cable to the Galileo board. Once a communication has been established between the Galileo's USB client port and development host PC by connecting a USB cable, the sketch can be uploaded to the Galileo. The Arduino IDE is used to upload the sketch. Figure 5-1 shows a sketch present in the user space section of the Linux that, uses the Arduino I/O adapter to communicate with the kernel. For more details on how to program the board, refer the Intel® Galileo Getting Started Guide [9].

5.1.3 Arduino Sketch

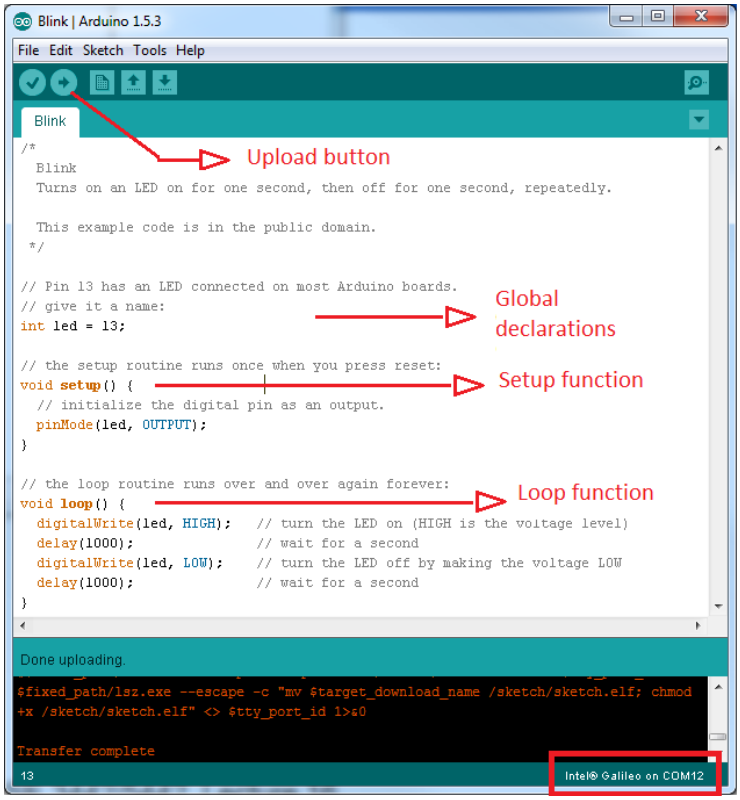

Figure 5-2 Blink example sketch on Arduino IDE

This section discusses very briefly about the layout of a sketch. The Arduino sketches consists of two parts, setup () and loop () function. Upon uploading a sketch onto the Galileo board, the setup () function is executed initially. All the hardware required initializations are generally done in this function. Setup () function runs only once. After execution of setup () method, the control is transferred to the loop () function and it shall remain here till the sketch is reset. Figure 5-2 shows an example blink program, present in the Arduino IDE. It shows the location of setup and loop functions, and the button to upload the sketch. Refer to Arduino Language Reference page [10] for more information on all the available Arduino API's

5.2 Hardware verification, the Arduino way.

The ADC hardware interfaced to the Galileo board as shown in Figure 5-21 can be verified by Arduino sketch. As explained earlier writing and uploading the sketches are made easy and less time consuming. More time can be dedicated in verifying if all the hardware connections are in place, and if AD7606 is responding in the required manner for the given input signals.

The figures 5-3 and 5-4 show the code flow for verifying the AD7606-Interface board. Based on the pin configurations shown in the figure 4-21, initialization code is written in the "setup ()" functional part of the sketch. The following points describe the activities of the "setup ()" function:

1. Based on the direction (input or output) of the connected pins they are initialized.
2. BUSY signal pin is initialized as an Interrupt and a callback function readData_SPI () is associated with it.

3. SPI-1 is initialized, clock is set, mode is set to 2, and ~ chip select is set to high.
4. Serial console is setup to enable debug messages
5. CONVST signal is set high.
6. RESET signal of one micro second is applied to AD7606.
7. Start Loop () function.

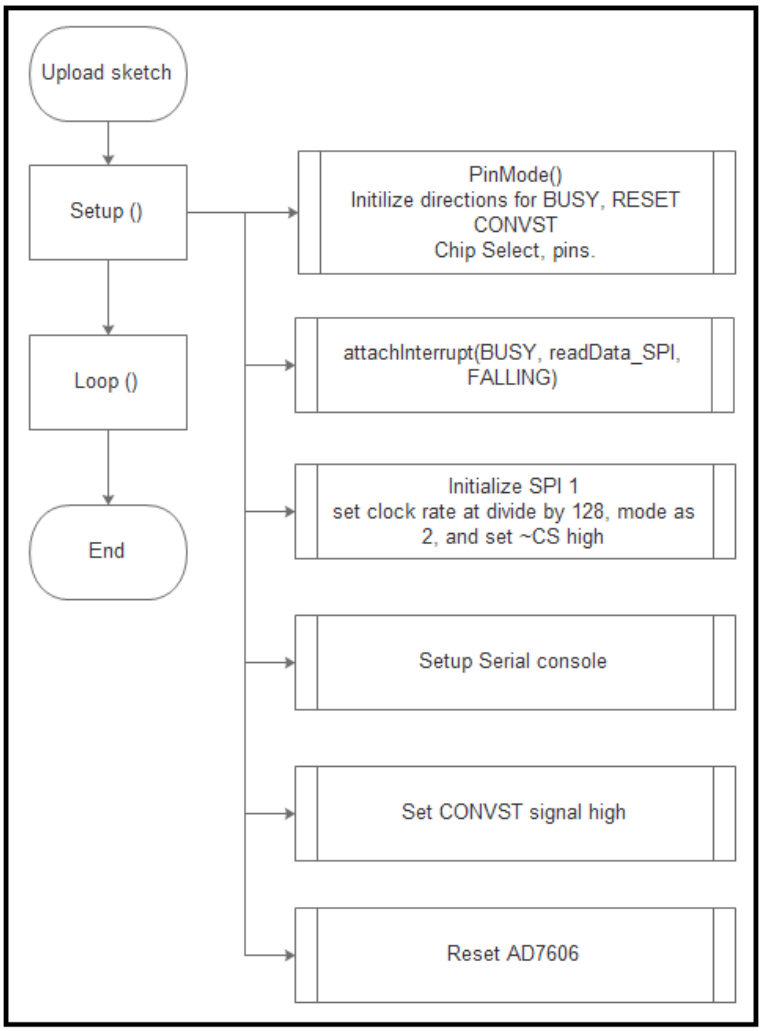

Figure 5-3 AD7606 verification using Arduino sketch, Setup ()

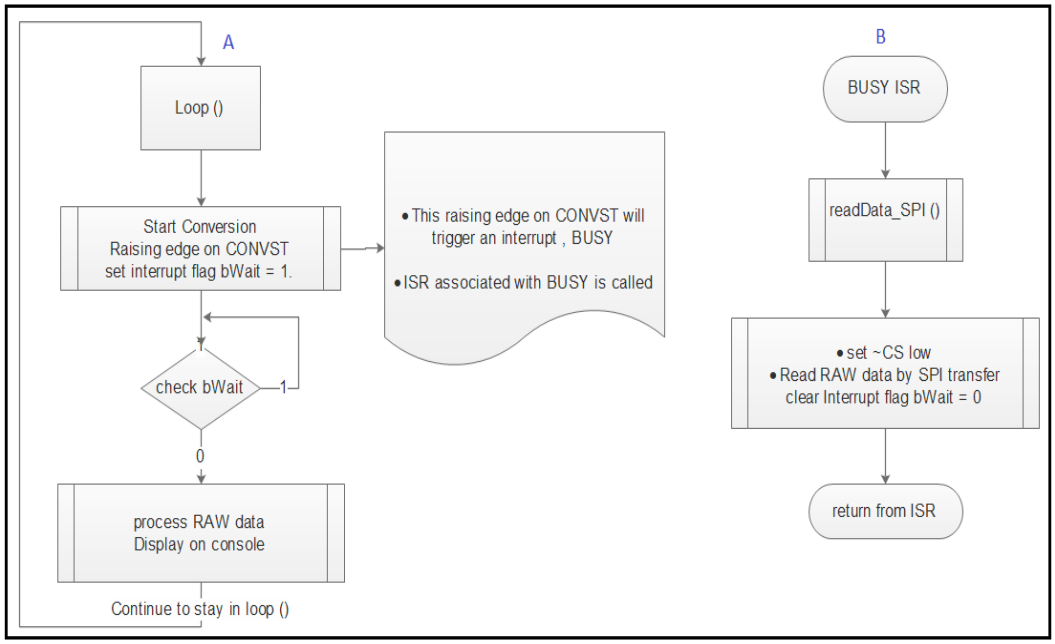

Figure 5-4 AD7606 verification using Arduino sketch, Loop ()

Figure 5-4 shows the program flow for loop () function which continuously reads AD7606 data over SPI-1. The following points summarize the tasks done by Loop () function shown in part A of the figure.

1. Starts conversion by setting CONVST pin high, the interrupt busy wait flag is set. AD7606 goes into BUSY mode and starts sampling the input signal.
2. Wait till BUSY Interrupt service routine (ISR) has been completed by monitoring interrupt busy wait flag.
3. The Arduino SPI transfers one byte of data at a time where as the ADC data length is two bytes per channel. Hence in Process the RAW data function, the data stored in a buffer during the SPI transfer is converted to suitable format.
4. Display the converted data on console for the user.

5. Continuously perform tasks 1 through 4.

The following points summarize the flow of the BUSY Interrupt service routine (ISR) that is shown in Part B of the figure 5-4.

1. ISR is triggered upon the falling edge on BUSY signal.
2. ISR calls the callback function, readData_SPI().
3. Selects the SPI-1 device by driving the corresponding chip-select low.
4. This callback function reads the data by performing SPI transfer activity.
5. Each byte read is saved in the buffer for further processing.
6. Disables the SPI-1 device by setting the “chip select” back to high.
7. Clears the Interrupt busy wait flag and sets the start conversion, CONVST signal low.
8. Returns form the ISR.

The Arduino sketches use the underlying Arduino library and hence are very small. In fact the code length of ADC verification sketch is less than one hundred lines. With the help of this sketch and oscilloscope, all the cold soldering joints on the hardware were fixed.

5.3 The Galileo Linux system

5.3.1 The Yocto Linux build setup for Intel® Galileo

“The Yocto Project is an open source collaboration project that provides templates, tools and methods to help you create custom Linux-based systems for embedded products regardless of the hardware architecture.” [26]

Started in 2010, the Yocto project intends to bring some order to the chaos present in the embedded Linux developers. It was formed by various hardware manufacturers and open-source operating system vendors.

Intel® Galileo uses Yocto project version of Linux kernel v3.8.7. This version has been customized to support all the features and configurations of the Galileo.

The Intel® Quark™ SoC X1000 Board Support Package Build Guide (BSP) [5] and Software User Guide document is provided by Intel® Galileo community [1]. This BSP provides in detail on how to download the sources that are required, procedure to setup the host machine to build, on how to compile the kernel and the file system, and finally it describes on how to boot the Intel® Galileo by placing the built images on both flash(Clanton-tiny version) and SD / USB (Clanton-full version) locations. The Clanton-full version takes more space. It also describes in detail on how to build cross compiler tool chain, which is required to compile the user space application programs.

The Galileo build system supports two types of build, depending on the packages included. The default build is the Clanton-tiny version that supports uClibc. The Clanton-full version that are built keeping SD/USB as source location for holding the binaries, can use the full or the fat version of C libraries, the eglibc or glibc. The uClibc is a scale down version of eglibc or glibc. This project runs a SD card image and uses eglibc or glibc as the C libraries. A corresponding c library has to be chosen while building the cross compiler toolchain.

5.3.2 Linux Industrial Input Output framework

The Industrial I/O (IIO) subsystem provides support for sensors mainly associated with ADCs and DACs. The IIO subsystem was developed by Jonathan Cameron in 2009 in the staging part of Linux kernel v2.6.32, the code quality has been improved and has been slowly moved out of staging (v3.5). Several other devices and sensors are falling in this category, e.g., Accelerometers, Gyros, Temperature sensors,

Pressure sensors, Proximity sensors etc. As of kernel v3.17, support exists for more than 200 IIO devices drivers in the main line.

The figure 5-5 shows the overview of IIO Subsystem, the application from the user space uses standard system call interface to communicate with the underlying kernel. Several commands like setting sampling rate, number of channels, and buffer size, etc., types of control signals are communicated to the device through the “sysfs” interface. The information from the device is available to the application through similar interface.

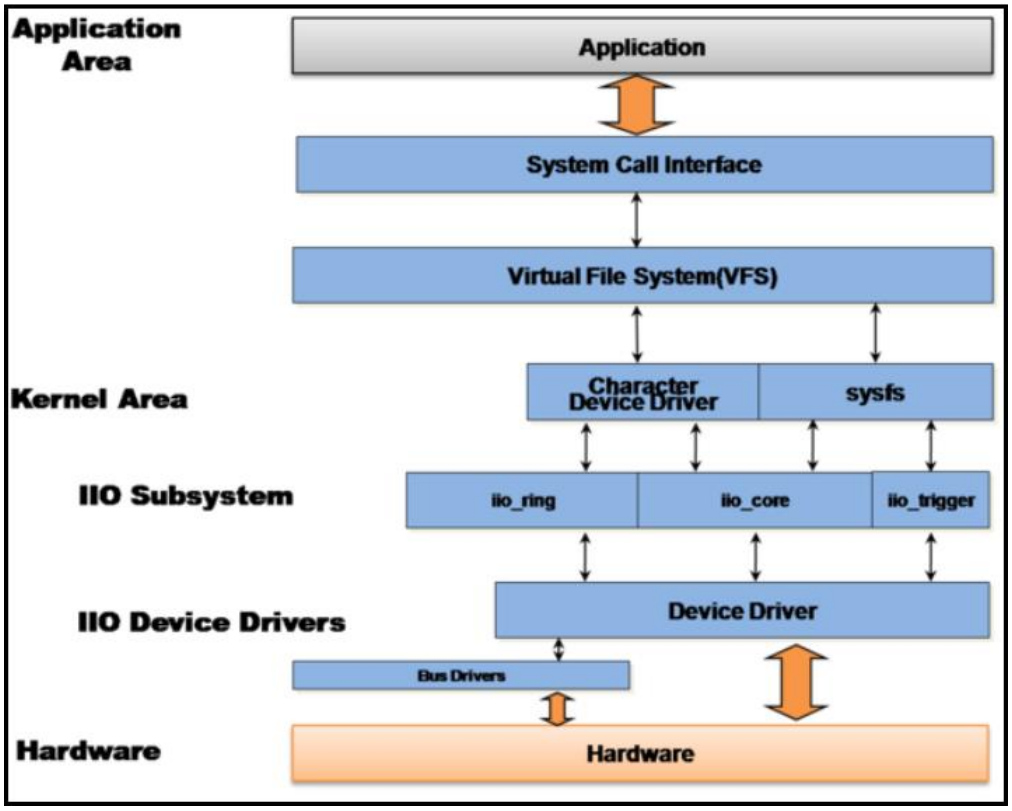

Figure 5-5 IIO Subsystem Overview [16]

The IIO subsystem consists of three modules, IIO core, IIO trigger and IIO ring. The IIO devices are registered to the IIO subsystem as IIO device drivers, which in turn are connected to the hardware through standard bus interface like SPI, I2C etc.

The input messages from the application are relayed to the device driver through the IIO trigger, and based on the type of information from the driver; it is passed to the above layers through IIO core and IIO ring layers of IIO Subsystem. The IIO core is used in transmitting single commands, and IIO ring is used for block transfer of data.

For each trigger input, the device driver gets the sampled data from the hardware; trigger handler reads and passes this to the IIO core and from there on to the application. For larger data set, this is an invitation for under performance. Each interrupt requires passing of data to the application, this induces a lot of overhead, due to context switches between the kernel space and user spaces.

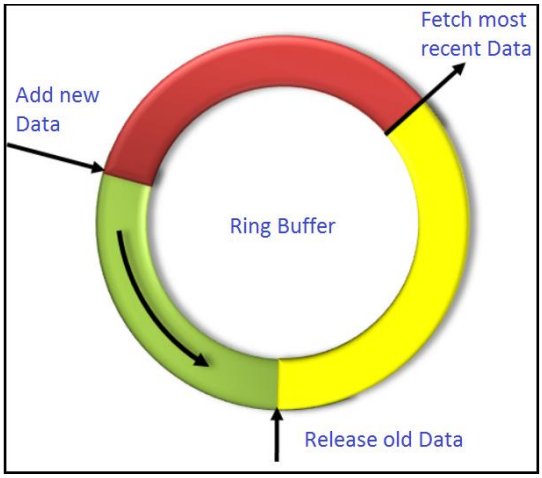

Figure 5-6 IIO Ring buffer.[16]

Figure 5-6 shows the diagram of IIO ring buffer, the data from the device is written into the single fixed size circular buffer and the user application shall read the data. This circular buffer is used to solve the fast producer (AD7606) – slow consumer (User space Application) problem.

There are different types of IIO triggers available, sysfs triggers are mostly used for single valued operations and are useful for verification, whereas continuous mode operations require the use of IIO HR timer triggers and IIO ring buffers. The data acquisition programs require continuous mode of operations.

5.4 AD7606 as a SPI platform device

Placing the AD7606 sensor interface code more closely to the hardware as a device driver in the kernel space than in the user space along with the application provides a better and a safer system.

Writing a device driver code is much harder than a user space program. Fortunately the AD7606 driver code is present in the main line Linux kernel. This reduces the development time and only needs to check how to include it in the kernel. The AD7606 driver code supports serial interface and parallel interface. The Table 5-1 shows the location of Kernel, Galileo platform and AD7606 sources files.

Table 5-1 Galileo-Source file locations

Categories	**Path**
Kernel	<Parent folder>/yocto_build/tmp/work/clanton-poky-linux-uclibc/linux-yocto-clanton/3.8-r0/linux
Galileo Platform file	<Kernel>/drivers/platform/x86/quark/intel_qrk_plat_galileo.c
AD7606 Driver files	<Kernel>/drivers/staging/iio/adc/ ad7606_core.c ad7606_spi.c ad7606_ring.c ad7606_par.c

AD7606 is connected to the Galileo via SPI and hence know as a SPI platform device and shall be registered as that with the Galileo platform file. Following sections describes the changes needed to successfully include AD7606 in the kernel.

5.4.1 Platform device initialization

Linux uses platform data to describe devices present on the board. The use of board structure which contains board specific information is a common approach in embedded and SoC based hardware, and they describe how the devices are connected to the SoC. The platform data structure states the GPIO pins used, chip variants, and several default initializations. Figure 5-7 shows the example of AD7606 platform data structure as, and Figure 5-8 shows the GPIO's associated with AD7606 in Galileo platform file.

```
/**
 * struct ad7606_platform_data - platform/board specifc information
 * @default_os:         default oversampling value {0, 2, 4, 8, 16, 32, 64}
 * @default_range:      default range +/-{5000, 10000} mVolt
 * @gpio_convst:        number of gpio connected to the CONVST pin
 * @gpio_reset:         gpio connected to the RESET pin, if not used set to -1
 * @gpio_range:         gpio connected to the RANGE pin, if not used set to -1
 * @gpio_os0:           gpio connected to the OS0 pin, if not used set to -1
 * @gpio_os1:           gpio connected to the OS1 pin, if not used set to -1
 * @gpio_os2:           gpio connected to the OS2 pin, if not used set to -1
 * @gpio_frstdata:      gpio connected to the FRSTDAT pin, if not used set to -1
 * @gpio_stby:          gpio connected to the STBY pin, if not used set to -1
 */
```

Figure 5-7 AD7606 example platform data structure [16]

```
static struct ad7606_platform_data ad7606_pdata = {
    .default_os = 0,
    .default_range = AD7606_RANGE,
    .gpio_convst = GPIO_AD7606_CONVST,
    .gpio_reset = GPIO_AD7606_RESET,
    .gpio_range = GPIO_AD7606_RANGE,
    .gpio_os0 = GPIO_AD7606_OS0,
    .gpio_os1 = GPIO_AD7606_OS1,
    .gpio_os2 = GPIO_AD7606_OS2,
    .gpio_frstdata = GPIO_AD7606_FRSTDATA,
    .gpio_stby = GPIO_AD7606_STBY,
};
```

Figure 5-8 AD7606 Platform data structure defined in Galileo

GPIO's of OSx, frstdata, stby and range have all been defined to -1 as they are hard wired. The Default range is 10, and GPIO's for busy (GPIO15), start conversion (GPIO14) and reset (GPIO28) have been assigned by referring to the Intel® Galileo IO mapping document [4].

5.4.2 Serial Interface

AD7606 is connected to the Galileo on SPI bus 1, during the boot; software must know complete SPI configurations and which devices are connected on each SPI bus. SPI devices doesn't support enumeration like the USB devices, hence the details like the bus it occupies, the chip select, etc., must be specified in detail in SPI board information structure.

Figure 5-9 shows the SPI board info structure defined to include AD7606 as a SPI device on Intel® Galileo board. AD7606 is connected on SPI bus one. Spi_onboard_dev, a SPI board structure defines the already existing SPI devices of the Galileo. AD7606 which is connected through SPI bus one conflicts with "spidev" device connection. Hence "spidev" has to be disabled. AD7606 SPI connections are defined in SPI board structure called spi_ad7606_dev.

In the SPI board structure, AD7606 describes the following

1. The device name for SPI driver registration, is “ad7606-8”
2. Maximum transfer rate is 4MHz.
3. Bus number to which it’s connected as one.
4. Chip select on this bus as zero.
5. Instance of AD7606 platform data structure.
6. Interrupt number of the GPIO as 63 (BUSY signal).
7. And finally the sampling mode as two.

```
static struct spi_board_info spi_onboard_devs[] = {
    {
        .modalias = "m25p80",
        .platform_data = &ilb_flash,
        .bus_num = LPC_SCH_SPI_BUS_ID,
        .chip_select = 0,
    },
    {
        .modalias = "ad7298",
        .max_speed_hz = 5000000,
        .platform_data = &ad7298_platform_data,
        .mode = SPI_MODE_2,
        .bus_num = 0,
        .chip_select = 0,
        .controller_data = &qrk_ffrd_spi_0_cs_0,
    },
    /*{     // SPI-1 is attached to AD7606, spidev is removed.
        .modalias = "spidev",
        .chip_select = 0,
        .controller_data = &qrk_ffrd_spi_1_cs_0,
        .max_speed_hz = 50000000,
        .bus_num = 1,
    },*/
};
static struct spi_board_info spi_ad7606_dev[] = {
    {
        /* the modalias must be the same as spi device driver name */
        .modalias = "ad7606-8",       /* Name of spi_driver for this device */
        .max_speed_hz = 4000000,      /* max spi clock (SCK) speed in HZ */
        .bus_num = 1,                 /* Framework bus number */
        .chip_select = 0,             /* Framework chip select */
        .platform_data = &ad7606_pdata,
        .controller_data = &qrk_ffrd_spi_1_cs_0, // &ad7606_chip_info, /* Blackfin only */
        .irq = 63, //-1,                         // Interrupt associated with GPIO15
        .mode = SPI_MODE_2,                      //
    },
};
```

Figure 5-9 AD7606 SPI board information

```
ad7606:
#if defined(CONFIG_AD7606) || \
        defined(CONFIG_AD7606_MODULE)

    if(ad7606_int_gpio_done)
        goto convst_mux;

    ret = intel_qrk_ad7606_int_gpio_request();
    if(ret)
        goto end;

    ad7606_int_gpio_done = 1;

convst_mux:
    if(ad7606_convst_mux_done)
        goto spi1_mux;

    printk(KERN_WARNING "AD7606: Request CONVST mux gpio");
    ret = intel_qrk_ad7606_convst_mux_request();
    if(ret)
        goto end;
    ad7606_convst_mux_done = 1;

spi1_mux:
    if(spi1_mux_done)
        goto ad7606_spi;

    printk(KERN_WARNING "AD7606: SPI1 mux select.\n");
    ret = intel_qrk_spi_gpio_req();

    if(ret){
        printk(KERN_ERR "AD7606: Error setting SPI1 mux select.\n");
        goto end;
    }

    spi1_mux_done = 1;

ad7606_spi:
    printk(KERN_WARNING "AD7606: spi registration request in restrict probe.\n");
    ret = spi_register_board_info(spi_ad7606_dev,   ARRAY_SIZE(spi_ad7606_dev))
        if (ret){
            printk(KERN_ERR "AD7606: SPI registration failed %d.\n",ret);
            goto end;
        }
#endif
```

Figure 5-10 AD7606 SPI device registration

Once the SPI board structure has been defined, it should be registered by calling spi_register_board_info() function. Figure 5-10 shows the registering activity at end of device probe function. One most important thing to note here is that all the GPIO's Busy, reset, conversion, Interrupt, and the GPIO's required by SPI need to be requested before calling the SPI registration. If the GPIO's are not available at the time of SPI registration, the registration shall be fail. The GPIO's are all reserved by the use of deferred probe. SPI registration is called only when all the GPIO's are allocated completely

5.3.4 Enabling the AD7606.driver

The kernel sources downloaded from the BSP can be customized and built. Kernel configuration is done by modifying the "menuconfig", the following command is used for that purpose.

"bitbake linux-yocto-clanton -c menuconfig "

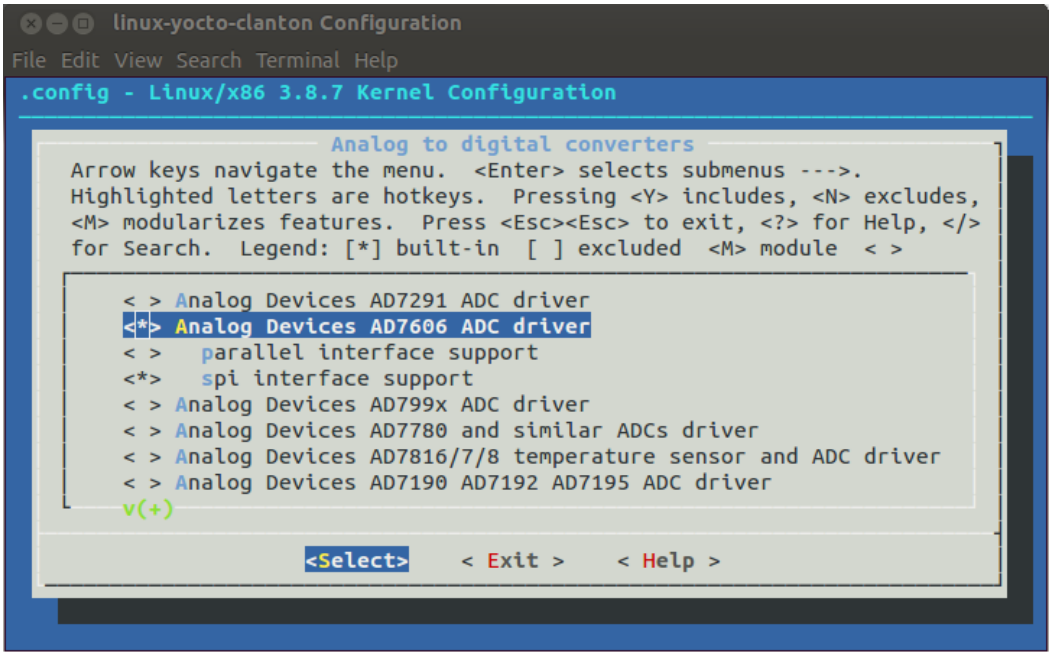

Figure 5-11 Galileo kernel menuconfig

Enable the AD7606 driver settings as shown below from the “make menuconfig”.

Device Drivers →

Staging →

IIO Staging drivers →

Analog to digital converters →

<*> Analog Devices AD7606 ADC driver

<*> spi interface support

The Figure 5-11 shows AD7606 driver being enabled in serial interface mode.

GPIO’s 0 to 15 are connected to Intel® Quark X1000 and the remaining 40 GPIO’s, from 16 to 55 are provided by the Cypress CY8C9540A I/O Expander. If any GPIO pin connected to the Cypress I/O expander is used, in that case, the code of AD7606 and the Galileo platform file which sets value to these GPIO’s, needs to be changed to support can-sleep option. If can-sleep GPIO API’s are not used, at the boot time, the setting of value to GPIOs will fail causing the driver to fail along with it.

Compile the kernel, place the image on the SD card and boot the Galileo. By enabling the boot time logs, successful registration of AD7606 as a SPI platform device can be observed. The next step involves driver testing.

5.5 Driver verification.

“Each and every IIO device, typically a hardware chip, has a device folder under /sys/bus/iio/devices/iio:deviceX. Where X is the IIO index of the device. Under every of these directory folders reside a set of files, depending on the characteristics and features of the hardware device in question.” [16]

Figure 5-12 shows the IIO device, and high resolution and sysfs IIO triggers.

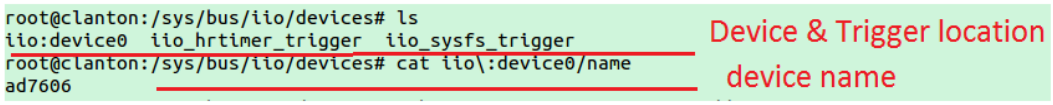

Figure 5-12 IIO Device and IIO Triggers

Driver verification can be done with the use of IIO sysfs triggers. Process is to add a sysfs trigger, associate it with the device as current trigger, and enable the channel to test. Trigger and read the RAW data. Figure 5-13 to Figure 5-17 show these steps.

```
root@clanton:/sys/bus/iio/devices# cd iio\:device0
root@clanton:/sys/bus/iio/devices/iio:device0# ls
buffer  in_voltage0_raw  in_voltage2_raw  in_voltage4_raw  in_voltage6_raw  in_voltage_scale  power         subsystem  uevent
dev     in_voltage1_raw  in_voltage3_raw  in_voltage5_raw  in_voltage7_raw  name              scan_elements  trigger
root@clanton:/sys/bus/iio/devices/iio:device0# ls scan_elements/
in_voltage0_en     in_voltage1_en     in_voltage2_en     in_voltage3_en     in_voltage4_en     in_voltage5_en     in_voltage6_en     in_voltage7_en
in_voltage0_index  in_voltage1_index  in_voltage2_index  in_voltage3_index  in_voltage4_index  in_voltage5_index  in_voltage6_index  in_voltage7_index
in_voltage0_type   in_voltage1_type   in_voltage2_type   in_voltage3_type   in_voltage4_type   in_voltage5_type   in_voltage6_type   in_voltage7_type
root@clanton:/sys/bus/iio/devices/iio:device0# ls trigger/
```

Figure 5-13 IIO Device 0 folder structure

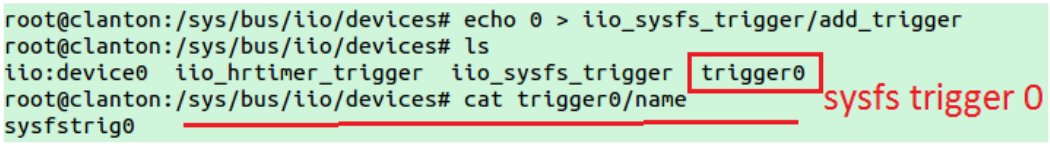

Figure 5-14 Adding a new trigger0 to IIO sysfs trigger

```
cat trigger0/name > iio\:device0/trigger/current_trigger
```

Figure 5-15 Associating trigger 0 with current trigger

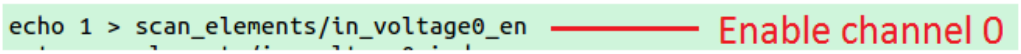

Figure 5-16 Enable the channel 0

```
root@clanton:/sys/bus/iio/devices# echo 1 > trigger0/trigger_now
root@clanton:/sys/bus/iio/devices# cat iio\:device0/in_voltage0_raw
-2    0 v
root@clanton:/sys/bus/iio/devices# echo 1 > trigger0/trigger_now ; cat iio\:device0/in_voltage0_raw
-16318    -5 V
root@clanton:/sys/bus/iio/devices# echo 1 > trigger0/trigger_now ; cat iio\:device0/in_voltage0_raw
16342    +5 V
root@clanton:/sys/bus/iio/devices# echo 1 > trigger0/trigger_now ; cat iio\:device0/in_voltage0_raw
-32512    -10 V
root@clanton:/sys/bus/iio/devices# echo 1 > trigger0/trigger_now ; cat iio\:device0/in_voltage0_raw
32722    10 V
```

Figure 5-17 Trigger and read the sampled data

The code can be converted to voltage by using the equation as shown below.

$$V_{in} = Code * (V_{max} - V_{min}) / (2^N)$$

$$V_{max} = 10,\ V_{min} = -10,\ N = 16$$

$$V_{in} = Code * 305.175\ \mu V$$

Figure 5-18 AD7606 Equation

This shows that AD7606 driver is working well for single trigger input operations. Continuous data acquisition from the AD7606 can be done by using IIO High Resolution triggers in combination with IIO ring buffers.

5.6 Data Acquisition using AD7606

This section describes in detail a user space AD7606 Data Acquisition (AD7606_DAq) program written in C for Intel® Galileo board. The Intel® Galileo compatible GCC cross compiler toolchain is necessary for development and porting of any C programs on the Galileo board. The Galileo BSP describes in detail on how to build the cross compiler toolchain. Based on the underlying C library support, two separate tool chains can be built, featuring a fully featured eglibc or a tiny uClibc. The kernel and the file system in this project has been compiled using fully featured eglibc hence, the eglibc featured toolchain is used for the application space programs as well.

The program has a custom make file which is used to compile the AD7606_DAq sources. Make files are very helpful during debugging, as it involves changes to code very often and source can be built using a simple “make” command. The compiled binary is placed on the onboard SD card and can be accessed either through a serial console or through SSH.

The AD7606 Data Acquisition program is a Linux console based real time multi-tasking data acquisition program. The program takes the required configurations through the command arguments and provides a simple menu screen for starting and stopping data acquisition, due to multi-threaded nature of the program, it is capable of listening to user request and at the same time performs data acquisition activity. The AD7606_DAq program uses AD7606 IIO custom API’s extensively to actively configure and control the AD7606 through the IIO sysfs entries it has exposed. Some of the IIO API’s are based on 2008 Jonathan Cameron’s example IIO application programing, which is distributed under GNU General Public License version 2, a free software for redistribution and modification. Jonathan Cameron is the author of the IIO framework present in Linux kernel and, present day IIO application programs uses these concepts.

5.6.1 Block diagram

The Figure 5-19 shows the block diagram of the AD7606 Data Acquisition program, from the application layer to the hardware interface layer.

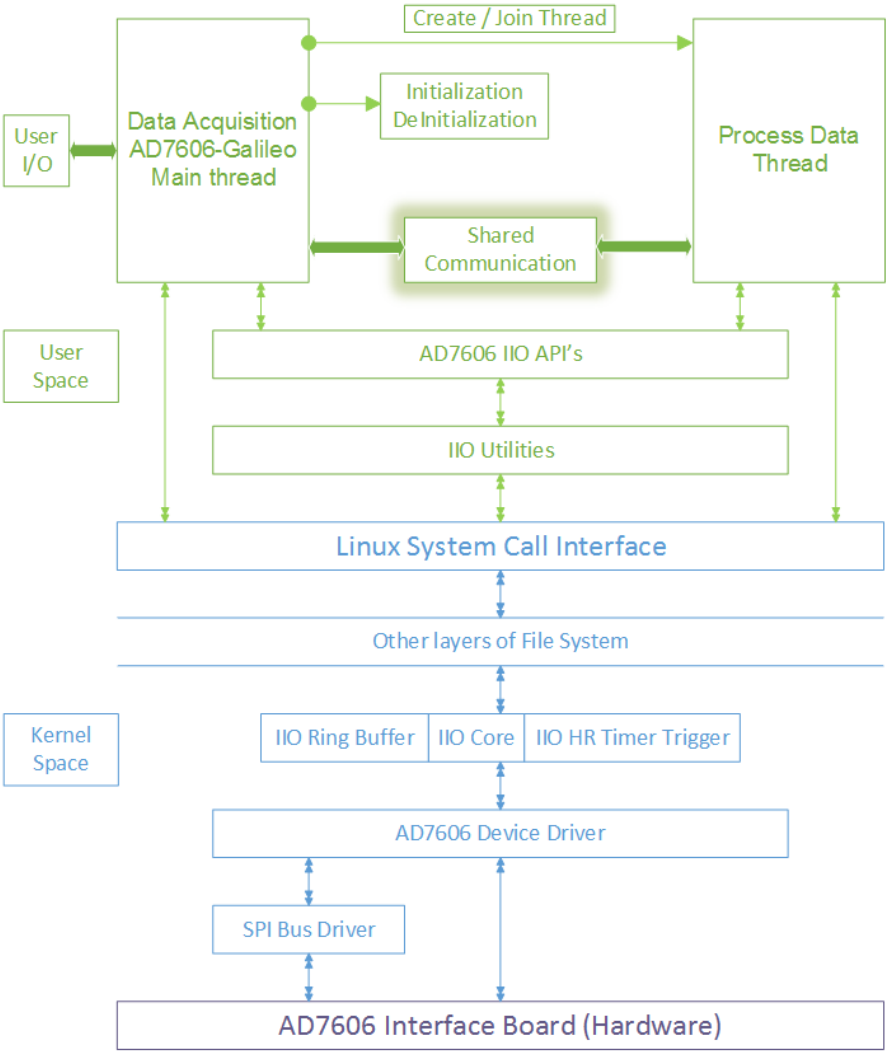

Figure 5-19 Block diagram of AD7606 Data Acquisition program

The AD7606_DAq program architecture is a state machine based and has three layers. The Linux eglibc supports pthread libraries and it has to be enabled at compile time by adding the directive "-pthread" to the CFLAG in the make file. Among the two

threads this program uses, the main thread interacts with user and does configuration and control activities, whereas the second thread is responsible for data acquisition and processing. The two threads share the state information of the system and it is made thread safe by the use of pthread mutex. The pthread condition variables are used to signal an event from one thread to another, and this provides synchronization between the two threads is achieved.

AD7606 driver with the help of IIO framework and sysfs exposes interfaces to the user space through which it can exchange information with the application layer. The driver subsequently communicates with the hardware. The sysfs file structure exposed by the IIO framework is similar among the IIO devices.

The AD7606_DAq has three layers, the activities in the first layer consists of the user interaction, calling below layer API's to configure, and the data collection and processing aspects. The bottom two layers namely AD7606 IIO API's and IIO utilities consists of transforming the user commands to IIO sysfs compatible messages. The IIO utilities deals with file names and values to write to the file and hence is generic and can be ported across all the IIO application programs. In contrast to the IIO utilities layer, the AD7606 IIO API layer is less generic, it's more specific to the AD7606 IIO device. It takes the specific inputs from the application and transforms it into generic versions that can be fed to IIO Utilities layer.

This program acquires data continuously and it is achieved by selecting the IIO High Resolution Timer Trigger in combination with the IIO Ring buffer. The support for both is enabled at the compilation of kernel. Before running the program, a HR Timer trigger needs to be added, it is similar to sysfs trigger, the one described in the earlier section. The association of trigger to the IIO device and the configuration of buffer are taken care by the AD7606_DAq program.

5.6.2 State machine

The AD7606_DAq program is driven using state machine. It consists of four states. The state change is unidirectional and both threads synchronize with each other based on the state of the system. The four states of the system are:

1. Initial
2. Executing
3. Stopping
4. Terminated

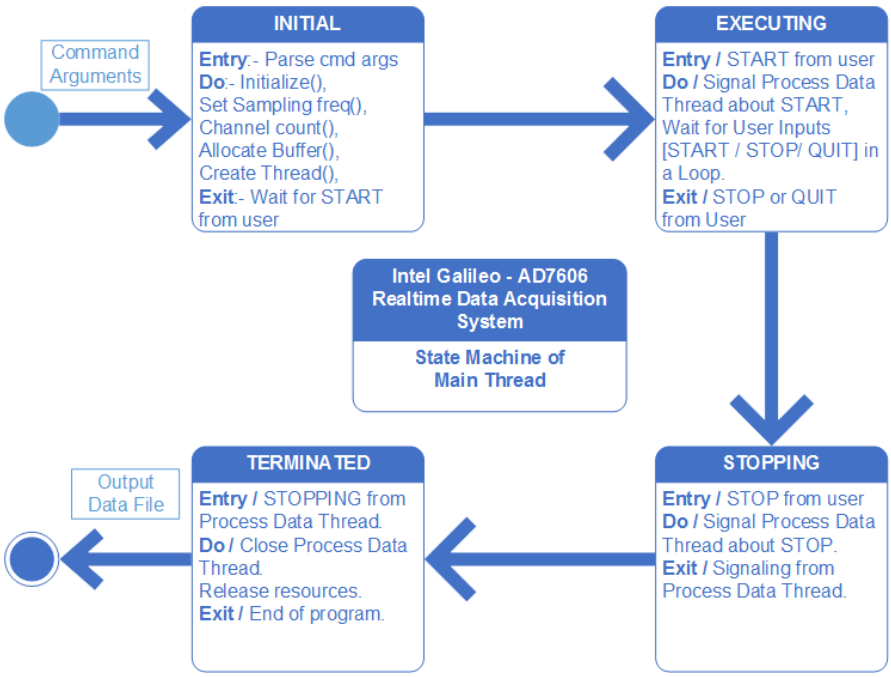

Figure 5-20 State Maching of the Main Thread

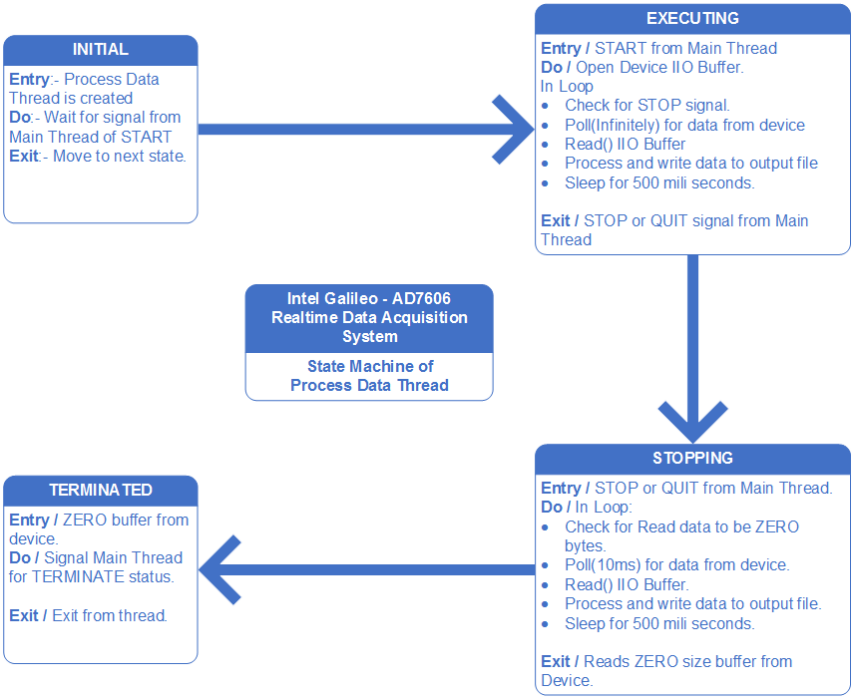

Figure 5-21 State Maching of the Process Data Thread

Figure 5-20 & 5-21 shows the behavior of Main thread and the Process Data thread in all the states.

The program is started by using the command given below.

binary.out -n <device_name> -t <trigger_name> -f <frequency> -c <channel count> -o <output file>

galileo.out –n ad7606 -t hrtimertrig0 -f 2000 -c 4 -o out.txt

In the INITIAL state, the main thread parses the arguments, verifies if the IIO Device and the IIO Trigger mentioned exists or not, Initializes by allocating memory for buffer, setting sampling rate, number of channels requested and finally creates the Process Data thread and both threads wait for start command from the user.

On START from the user, the Main thread changes the state of the system to EXECUTING and requests start sampling command to the hardware. It signals the

Process Data thread of the Start, and continues to monitor user Inputs for Stop command. The Process Data thread opens the IIO Buffer and performs these tasks in a loop.

1. Verifies if the state of the system is not STOPPING,
2. Starts polling for data, the Poll () function is a blocking call and will block the caller till any data is available on the IIO buffer file. On data availability the thread is unblocked,
3. Reads the IIO device buffer,
4. Processes the buffer and writes it to a file,
5. The thread sleeps for around 500ms before repeating the tasks from step 1.

On Main thread receiving STOP command from user, the main thread changes the system state to STOPPING and waits for event from the Process Data thread, the Process Data thread, detects the state change and initiates the stopping sequence.

1. Send stop sampling command to hardware.
2. Start polling for data with timeout of 10 milli second,
3. Read the IIO buffer and process the data read.
4. If there is no data available, quit else repeat steps 1 to 4 till no data is available.
5. Before exiting the thread, it changes state to TERMINATED and signals the Main thread.

Main thread destroys the Process Data thread, closes output file, and releases all the resources and exits the program.The final data collected can be read from the output file.

5.6.3 Sequence flow diagrams of AD7606 Data Acquisition program.

The next few figures in this section give the sequence flow of the program in detail. Figure 5-22 gives a brief overview of the whole program. Figure 5-23 to Figure 5-25 shows the Initialization phase, Figure 5-26 describes the action taken during the execution phase, followed by Figure 5-27 showing the sequence stopping phase, and finally the Terminated phase is shown in Figure 5-28,.

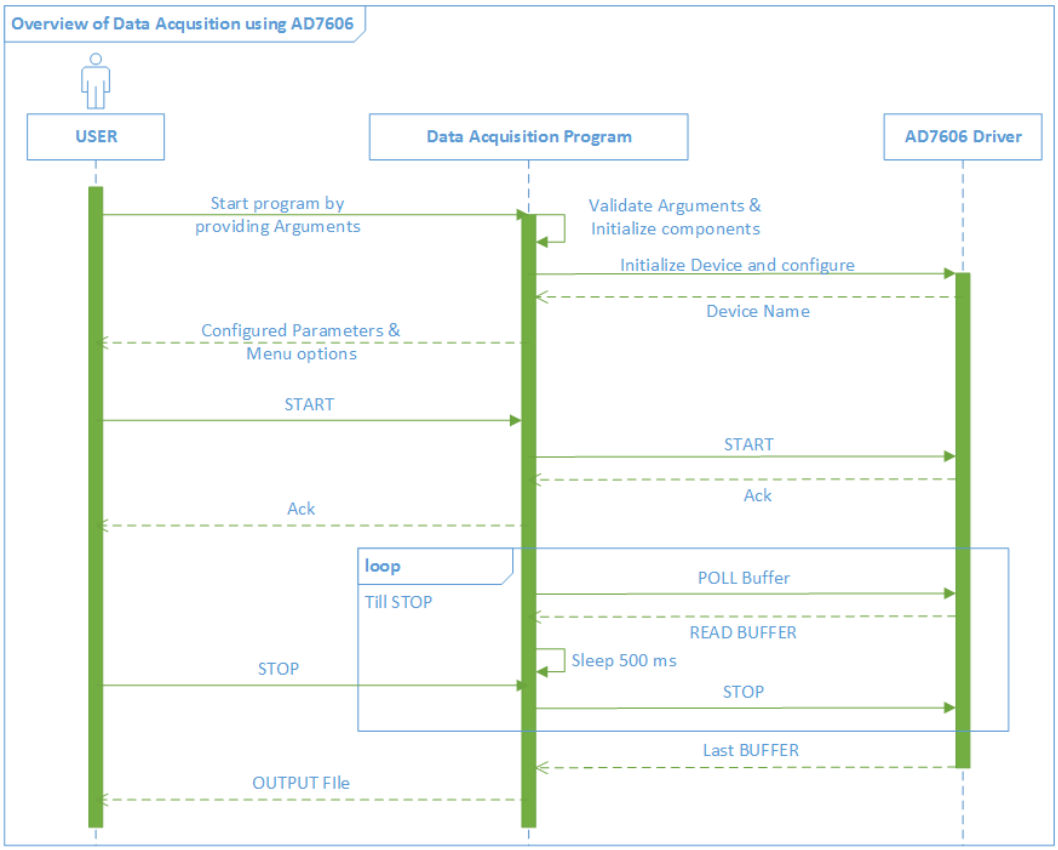

Figure 5-22 Overview of AD7606_DAq program

Next three figures shows the sequences followed during the INITIAL state.

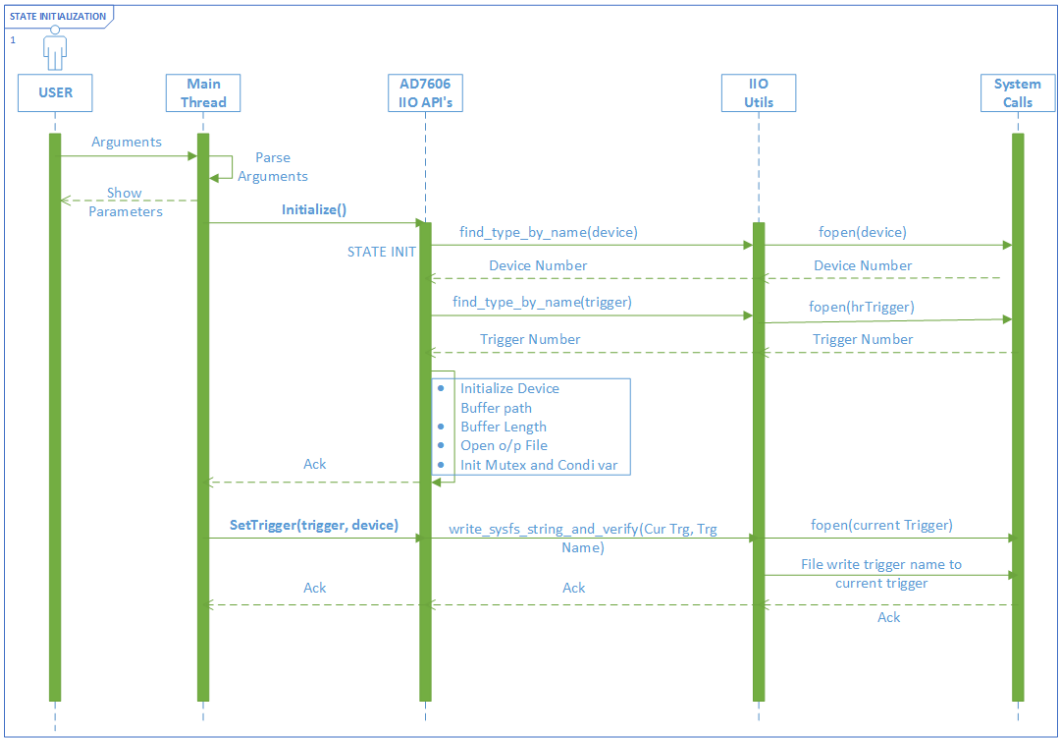

Figure 5-23 Sequence flow of Initialization phase

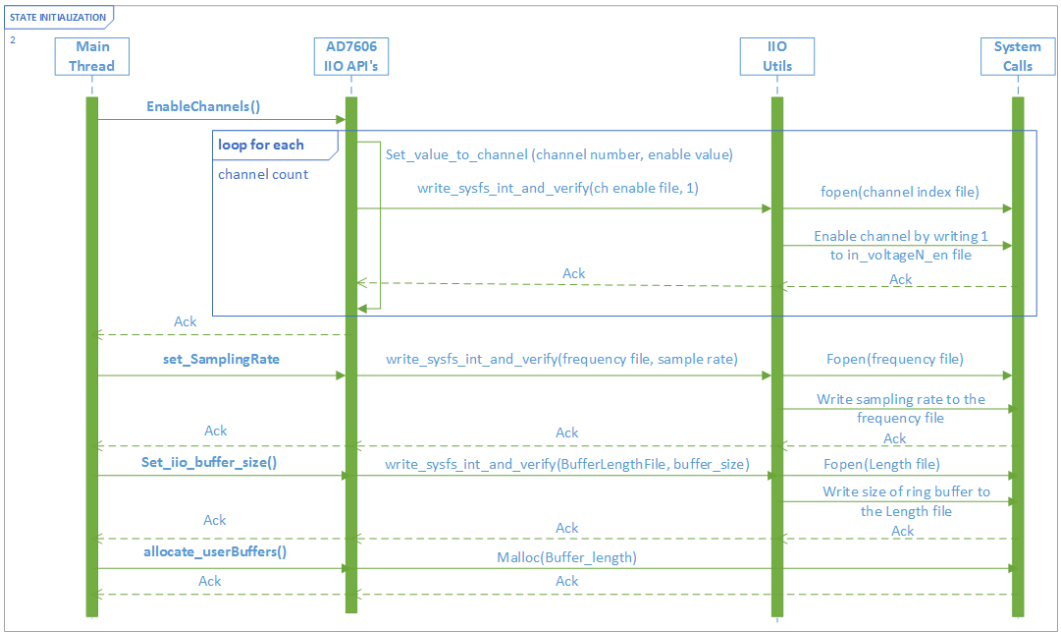

Figure 5-24 Sequence flow of Initialization phase continued

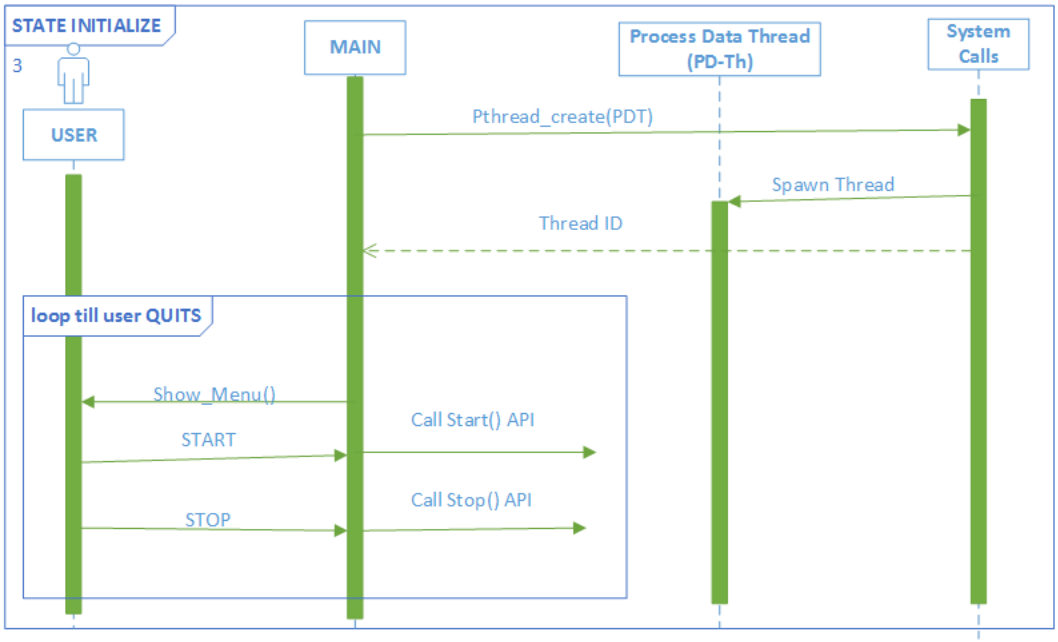

Figure 5-25 Sequence flow of Initialization phase final

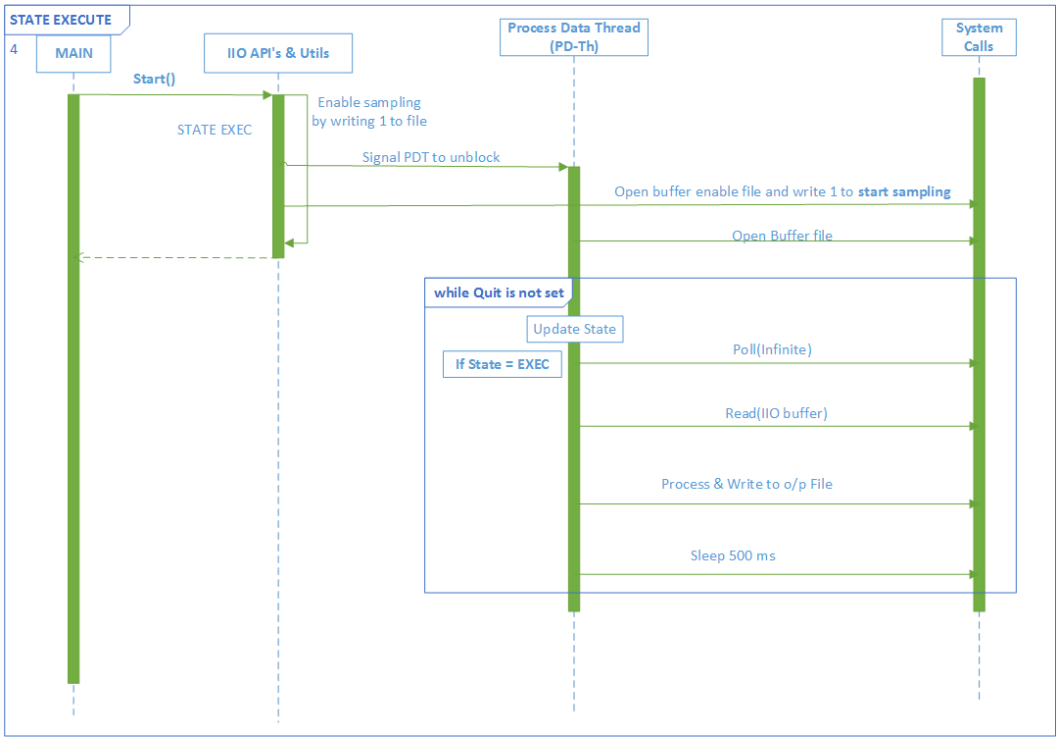

Figure 5-26 Sequence flow of Execution phase

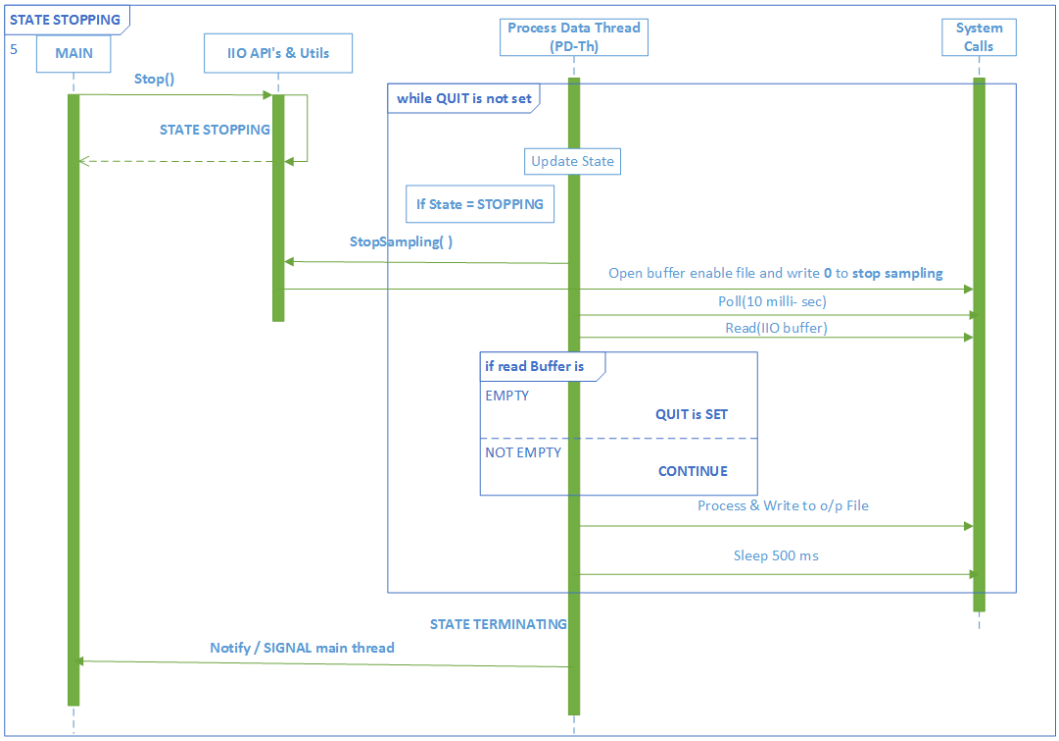

Figure 5-27 Sequence flow of Stopping phase

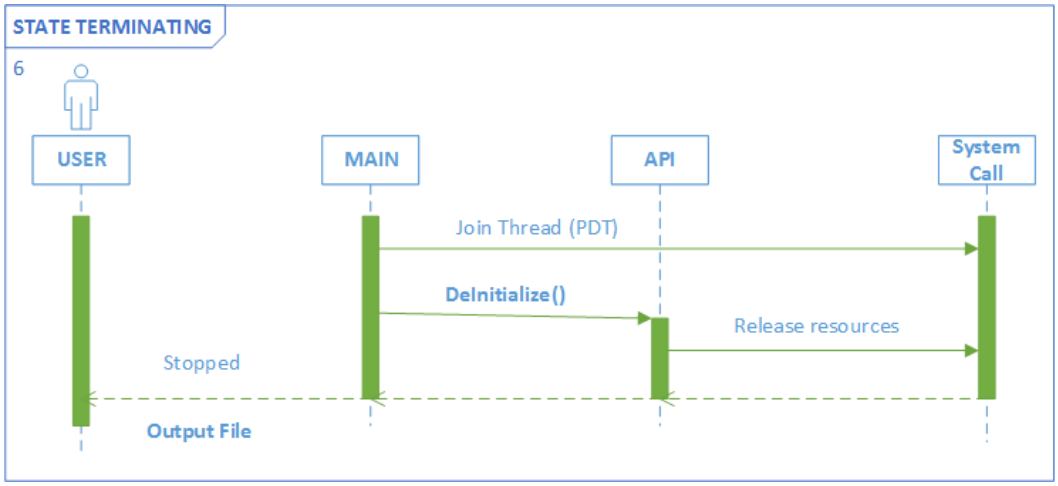

Figure 5-28 Sequence flow of Terminated phase

Chapter 6

RESULTS

This chapter briefly discusses the output of Data Acquisition program and the shortcomings of this system.

6.1 Experimental Results

The following section shows the plots of sampled data obtained using AD7606 Data Acquisition program.

This experiment used 4 analog channels as inputs, with Channel 1 taking a sine wave as input. Channel 2 to 4 took a steady voltage of 5V, +10V, -10V respectively.

Each experiment was conducted with different sampling rate and different frequency on channel 1.

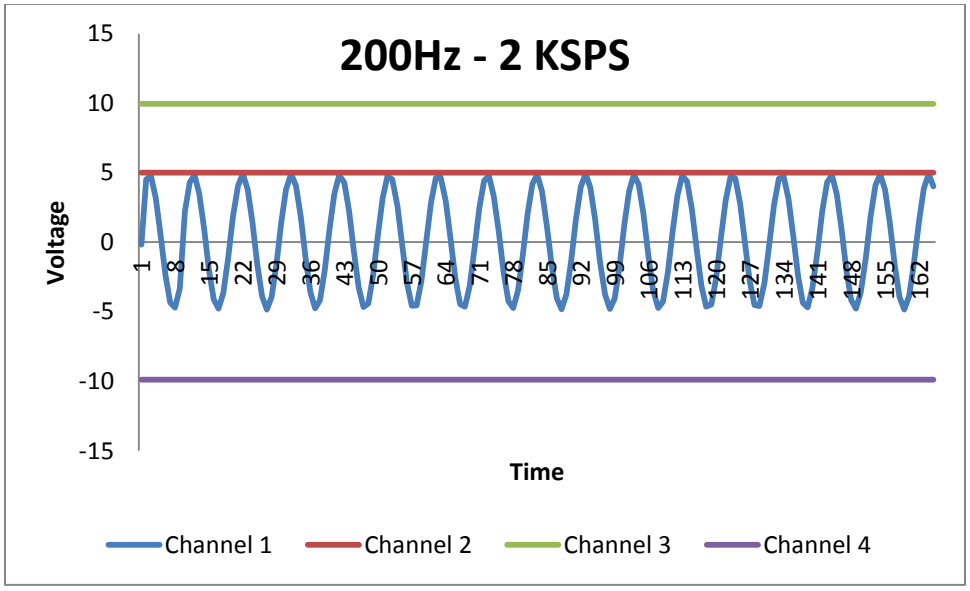

Figure 6-1 Sampling rate of 2KHz and 200Hz input

Figure 6-1 shows the plot obtained while sampling 4 analog channels at a rate of 2 KSPS. The input signal of 200 Hz sine wave is connected on channel 1 and a steady voltage of 5V, 10V, -10V is applied on each channels 2, 3 and 4 respectively.

Figure 6-2 shows the plot of 4 channels. With respect to the previous experiment, it retains the sampling rate at 2KSPS but has a different input frequency on channel 1, a 500 Hz sine wave. The remaining inputs are kept constant. From the plots it can be observed that the channel 1 is not faithfully reproduced.

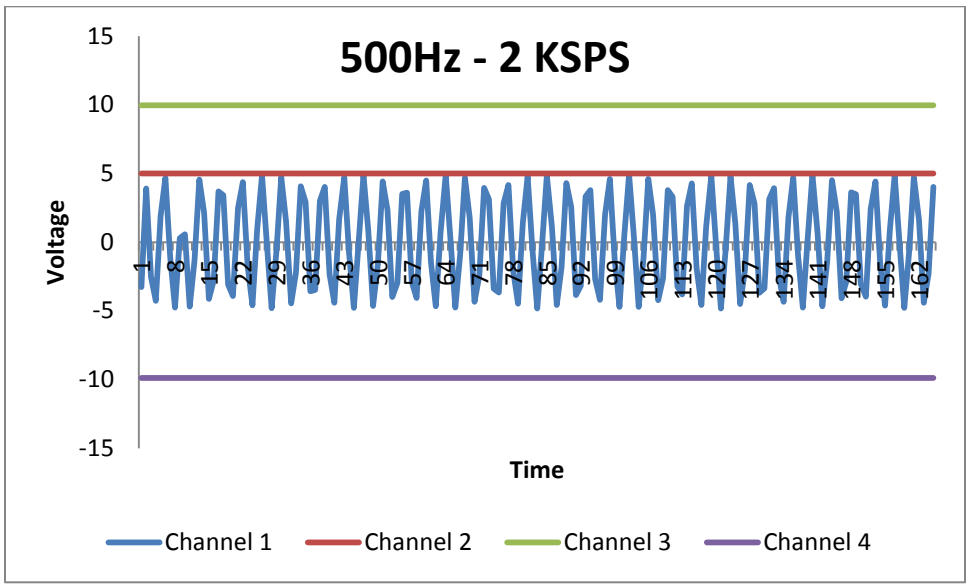

Figure 6-2 Sampling rate of 2KHz and 500Hz input

Figure 6-3 shows the plot with both sampling rate and input frequency on channel 1 being changed. The Sampling rate has been changed to 3KSPS and a 300Hz sine wave is applied to channel 1. The remaining inputs are kept constant. Even on increasing the sampling rate and reducing the input frequency, output samples were not a good reproduction of the input signal.

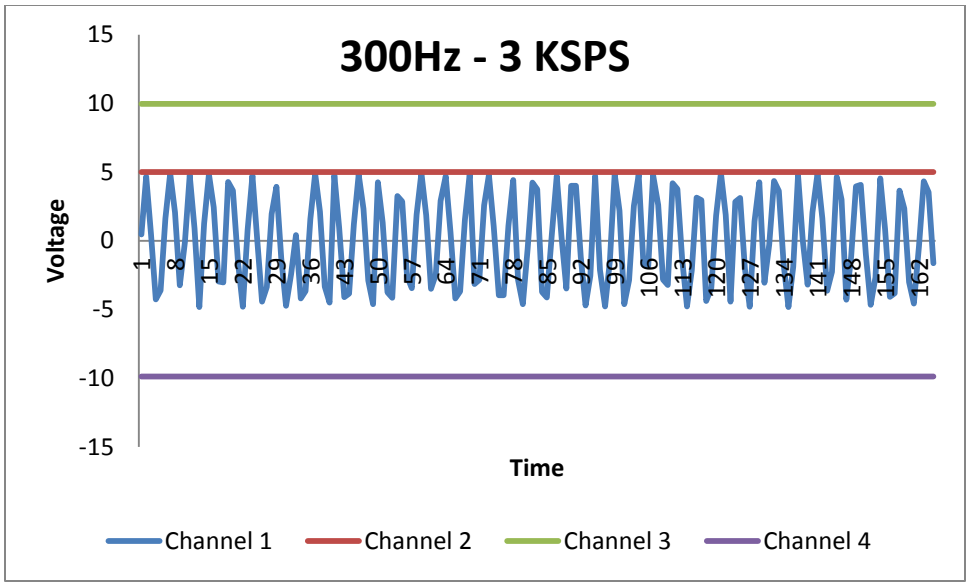

Figure 6-3 Sampling rate of 3KHz and 300Hz input

6.2 Drawbacks of Galileo AD7606 system

6.2.1 Drawbacks of Arduino sketch for real time data acquisition

The fast and easy way of development using Arduino sketch hides lot of information which could prove fatal to a real time embedded data acquisition task.

1. Arduino SPI API's are user space API's and they use the "spidev" driver module of the Linux kernel to read the device connected on SPI-1. This "spidev" module doesn't support any buffers at the kernel level hence, a lot of time is spent in transferring the control between the user space and kernel space.
2. The sketch reads one byte of data over SPI at a time, which corresponds to 16 times change of control between user space and kernel space for all the eight channels. This induces lot of delays and causes loss of data.

3. Though Intel® Galileo runs a Linux kernel, the Arduino sketches does not support threads hence, all the tasks: the data collection, the data processing and user interaction has to take place in just one thread.

6.2.2 Drawbacks of AD7606 Galileo Linux system

The maximum sampling frequency that could be generated from the pin interface of the Galileo board IO2 or GPIO 14 which is directly connected to the Quark SoC is two kilo hertz (2 KHz). The required sampling rate to work with the Roline Laser and other sensors is 15 KHz.

The Cypress CY8C9540A GPIO expander connected to the Quark SoC on the Galileo board could be used to produce the clock signal at higher rate of around 14 KHz. This still is not sufficient. There is another problem using the GPIO expander, it is connected to Quark SoC through an I2C bus, and hence the GPIO controller is implemented as a can-sleep module. This will result in generation of clocks with varying rate. This varying clock rate will not affect usage in many applications, but a data acquisition program used for road profiling system needs a stable clock. The AD7606 requires the start conversion pin to be connected to a GPIO that does not sleep. Due to these reasons the current Intel® Galileo AD7606 Data Acquisition cannot be used without changes for performing Road profiling.

Chapter 7

CONCLUSION AND FUTURE PROSPECTS

7.1 Conclusion

This research project explored the use of newly available embedded boards for use in profiling systems. It provided an implementation for such a system by using the Intel® Galileo as an embedded board and the AD7606 a compatible ADC.

The project describes in detail the features of road profiling that are required to be considered while evaluating the various embedded boards and the ADCs'. It talks about the limitations in the features supported by the embedded boards and the difficulty in designing of the interface with the ADC. This project provided different software designs and implementations, for:

1. The verification of hardware in a data acquisition system,
2. A real time data acquisition application.

The availability of the AD7606 driver sources in the main line of the Linux kernel tree along with the support provided by the online forums of Intel and Analog Devices has hastened the development time of the system.

The Arduino features of the Intel® Galileo showed fast and great results in the validation of the hardware. The Linux sysfs and IIO triggers are quick and sure way of validating the AD7606 driver.

While the AD7606 and the Intel® Galileo board are excellent products on their own, when combined together for use in a real time data acquisition system the results are not as promising. This research project also described the hardware and software constraints in the implementations of this device, which prevents its use as a real time data acquisition for road profiling.

Further research is required in discovering and including the various boundary cases in the evaluation of embedded boards.

7.2 Future Prospects for Dual Line Laser Profiler Systems

This section proposes solution for

1. Overcoming the drawbacks of the Intel® Galileo AD7606 system.
2. Implementing a Dual Line Laser Road Profiling Systems.

The sampling problems discussed in Chapter 6 can be solved by the use of an external clock generation circuit. The external clock would be used, instead of the IIO high resolution timer triggers in the Intel® Galileo AD7606 Data Acquisition system.

Dual Line Laser Road profiling systems can be implemented by enhancing the software of the existing single Roline Line Laser data acquisition program. Portable profiling systems could be built by considering the use of Intel® Next Unit of computing (NUC), in the place of the existing processing core (PC).

References

[1] Intel® Corporation, (2014 March) Intel® Galileo User Guide [Online] Available: - http://download.intel.com/support/galileo/sb/galileo_boarduserguide_330237_001.pdf

[2] Intel® Galileo Board Schematic [Online] Available: - http://download.intel.com/support/galileo/sb/galileo_schematic.pdf

[3] Intel® Galileo Board Release Notes (1.0.0) for Arduino [Online] Available: - http://download.intel.com/support/galileo/Galileo_RelNotes_329686_005.pdf

[4] Intel® Corporation, (2014 Feb 24) Intel® Galileo I/O Mappings [Online] Available: - http://download.intel.com/support/galileo/sb/galileoiomappingrev2.pdf

[5] Intel® Quark™ SoC BSP Build Guide [Online] Available: - http://download.intel.com/support/processors/quark/sb/quark_bspbuildguide_329687_001.pdf

[6] Intel® Galileo Product Brief [Online] Available: - http://download.intel.com/support/galileo/sb/galileoprodbrief_329680_003.pdf

[7] Intel® Galileo (Gen 1) Development Board Documents https://communities.intel.com/community/makers/galileo/documentation/galileodocuments

[8] Arduino Introduction http://arduino.cc/en/guide/introduction

[9] Intel® Galileo Getting started [Online] available: - http://download.intel.com/support/galileo/sb/galileo_gsg_329685008.pdf

[10] Arduino Language Reference http://arduino.cc/en/Reference/HomePage

[11] Ajit Modi, Dr Roger Walker. Galileo Lecture series, Embedded 1 class.

[12] Intel® Galileo Board community support [Online] http://www.intel.com/support/maker/galileo.htm#documents

[13] Analog Devices Wiki http://wiki.analog.com/software/linux/docs/iio/iio

[14] High speed data acquisition using the Linux Industrial IO framework. Lars-Peter Clausen, Analog Devices. http://events.linuxfoundation.org/sites/events/files/slides/iio_high_speed.pdf

[15] IIO, a new kernel subsystem. Maxime Ripard, Free Electronics. https://archive.fosdem.org/2012/schedule/event/693/127_iio-a-new-subsystem.pdf

[16] Analog Devices AD7606 Wiki http://wiki.analog.com/resources/tools-software/linux-drivers/iio-adc/ad7606

[17] United States Numbered Highways, a Wikipedia article. http://en.wikipedia.org/wiki/United_States_Numbered_Highways

[18] The Little Book of Profiling by UMTRI, Michael W. Sayers and Steven M. Karamihas.

[19] "National Highway System Length 2001" by the US Department of Transportation, Federal Highway Administration. http://www.fhwa.dot.gov/ohim/hs01/hm41.htm

[20] 3D system profiles highway surface. http://www.vision-systems.com/articles/print/volume-12/issue-2/features/spotlight-on-market-opportunities/3-d-system-profiles-highway-surfaces.html

[21] Impact of Changes in Profile Measurement Technology on QA Testing of Pavement Smoothness, Technical report 0-6610-1. http://tti.tamu.edu/documents/0-6610-1.pdf

[22] BU-353S4 GPS data sheet http://usglobalsat.com/store/download/688/bu353s4_ds.pdf

[23] NEMA format http://www.gpsinformation.org/dale/nmea.htm

[24] AD7606 Data Sheet. http://www.analog.com/media/en/technical-documentation/data-sheets/AD7606_7606-6_7606-4.pdf

[25] DT9816 Data Sheet https://datatranslation.box.com/shared/static/aabdc787ad5cbd7d1f91.pdf

[26] Yocto https://www.yoctoproject.org/

[27] BeagleBone Black http://circuitco.com/support/index.php?title=BeagleBoneBlack#Hardware_Files

Biographical Information

A P Vikram Simha has received his Bachelor of Engineering in Electronics and Communications from Visvesvaraya Technological University (VTU) in the year 2006. Having developed interest in the field of electronics during high school, he has participated in various science exhibitions and won several prizes. He did his senior design project for Hindustan Aeronautics Limited (HAL) a defense company of Government of India. This project piqued his interest in the field of Embedded Systems; he later pursued his interest in designing and developing products for mobile devices manufacturing giants like Samsung, LGE, etc. Working alongside of architects of Texas Instruments motivated him to pursue his Masters in Computer Engineering from University of Texas at Arlington. He joined UTA in the fall of 2012. While studying at UTA he has worked as Graduate Teaching assistant for courses that are related to the field of embedded systems. He has been working as a research assistant under Dr. Roger Walker in the development of state of the art Road Profiling Systems that are used by Texas Department of Transport.