

AUTOMATED MEASUREMENT OF RESISTIVITY OF THIN FILMS

A DISSERTATION

*Submitted in partial fulfillment of the
requirements for the award of the degree*
of
MASTER OF TECHNOLOGY
in
SOLID STATE ELECTRONIC MATERIALS

By

MD FAISAL KHAN



**DEPARTMENT OF PHYSICS
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE - 247 667 (INDIA)**

JUNE, 2006

CANDIDATE'S DECLARATION

I hereby declare that the work which is being presented in this dissertation report entitled **“AUTOMATED MEASUREMENT OF RESISTIVITY OF THIN FILMS”** for the partial fulfillment of the requirement for the award of degree of **“MASTER OF TECHNOLOGY”** in **“SOLID STATE ELECTRONIC MATERIALS”** and submitted in the department of Physics, Indian Institute of Technology Roorkee, Roorkee, is an authentic record of my own work carried out under the supervision and guidance of Dr. S.K.Barthwal, Associate Professor, Department of Physics, Indian Institute of Technology Roorkee, Roorkee during the period from July 2005 to June 2006.

The matter embodied in this dissertation report has not been submitted by me for the award of any other degree or diploma.

Date: 30/06/2006

Place: Roorkee



MD. FAISAL KHAN

CERTIFICATE

This is to certify that the work related to the Dissertation entitled **“AUTOMATED MEASUREMENT OF RESISTIVITY OF THIN FILMS”**, has been carried out by **MD FAISAL KHAN** of **M.Tech (Solid State Electronic Materials)** under my supervision.



Prof. S. K. Barthawal

Associate Professor

Department of physics

Indian Institute of Technology

Roorkee

ACKNOWLEDGEMENTS

This dissertation is not by far created by me alone, without encouraging support and appreciation from my guide **Dr. S. K. Barthawal**, Associate Professor, Department of Physics, Indian Institute of Technology Roorkee, Roorkee, it would not have been as enjoyable working here and writing this dissertation as it really has been. Under his guidance I have not only gained technical expertise but also learnt to apply my presence of mind and to tackle practical problems. Moreover he enhanced my self confidence whenever work seemed difficult to proceed.

I like to thanks **Dr. J. Rai**, Head of Department, Physics, Indian Institute of Technology Roorkee, and **Dr. R. Nath** Coordinator, M.Tech, Solid State Electronic Materials Indian Institute of Technology Roorkee, for giving inspiration and providing every possible support.

I wish to express my sincere thanks to Mr. **Adarsh Gupta** for providing me valuable suggestions and accurate comments on my work to direct me in the right direction when I needed it.

I have also the pleasure of having **Rajesh Jourwal**, M.Tech. Student and **Neelima Agrawal**, Research Scholar, as a very valuable colleagues by contributing with well contemplated comments on this dissertation, they have made me realize new aspects of my work.

Finally, I would like to give my appreciation to all my classmates of Solid State Electronic Materials, who cooperated all the time during the course of this work.

Date: 30/06/2006

Place: Roorkee



MD FAISAL KHAN

ABSTRACT

In the area of measurements, a big part of the technical expertise is gained from practical experience. Therefore, the learning period require to become a skilled engineer is usually very long. Nowadays, when more industries/companies go into the area of measurement by choosing hardware and software solutions instead old manual methods there is a rising need for more efficient measurement methods for engineers.

The programming ease and falling cost of PC make it very attractive to be used in automated measurements. The availability of high performance computing software can be used for data management, processing and presentation.

Computer aided measurements needs interfacing hardware which has been presented in this dissertation. First, the subject of data acquisition has been structured into adequate work areas and the required human knowledge for engineers in the measurement area is classified and described. Each part of the hardware design including, unipolar and bipolar configuration of digital to analog converter for writing digital data, selecting channels and acquiring data using multiplexing, reading digital data and controlling parameters using analog to digital converter, generating clock signal have been explained.

In a subject like measurement, which is strongly connected to practical knowledge, it is vital that presented theory is accompanied by practical related information thus finally required programs and graphs have been presented to support the proper working of interface using MATLAB and data acquisition toolbox.

To facilitate the learning of component behaviors and circuit performance in this dissertation, there is a methodology presented how to develop comprehensible hardware for automated measurement and to facilitate the understanding of programming language, structured programs for data acquisition have been explained.

This work may help engineers to find more structured ways of designing data acquisition hardware.

CONTENTS

Candidate's Declaration	i
Acknowledgements	ii
Abstract	iii
Contents	iv
List of Figures	v
List of Tables	viii
Chapter 1 Introduction	1
Chapter 2 Data Acquisition	3
2.1 The Data Acquisition System	3
2.2 The Analog Input Subsystem	13
2.3 Making Quality Measurements	24
Chapter 3 Description of Hardware	33
3.1 Parts of Hardware	33
3.2 Circuitry for Writing Digital Data	34
3.3 Circuitry for Selecting Channels	38
3.4 Circuitry for Reading Digital Data	43
3.5 Circuitry for Generating Clock	54
Chapter 4 Software for Acquiring Data	56
4.1 Introduction to MATLAB	56
4.2 Data Acquisition Tool Box	58
4.3 Digital Input/Output Object	59
4.4 Reading and Writing Operation	66
4.5 Generating Timer Events	70
4.6 Evaluating Digital I/O Object Status	72
Chapter 5 PCI 1751 Digital I/O Card	73
5.1 General Information	73
5.2 Installation	76
5.3 Operation	82
Chapter 6 Thin Films and Resistivity Measurement	90
6.1 Elementary	90
6.2 Practical Schemes	94
6.3 Low Impedance Measurement	97
Chapter 7 Results and Discussions	103
Chapter 8 Conclusion	119
References	120
	<u>Appendices</u>
Appendix A: - Programs for Data Acquisition	122

LIST OF FIGURES

Figure 2.1-Components of Data Acquisition	4
Figure 2.2-Multifunction Board	5
Figure 2.3-Relationship b/w User, Driver Software, Application Software and Hardware	12
Figure 2.4-ADC with Multiplexer	14
Figure 2.5-Plot of Channels and Time using Scanning Hardware	15
Figure 2.6-Plot of Channels and Time using SS/H Hardware	16
Figure 2.7-Sine Wave Quantized by ADC	17
Figure 2.8-Plot Quantization Error and Time	18
Figure 2.9-Unipolar and Bipolar Signal	19
Figure 2.10-Differential Input	21
Figure 2.11-Single Ended Input	21
Figure 2.12-Precision Vs Accuracy	25
Figure 2.13-Occurance of Aliasing	30
Figure 2.14-Effect of Low and High Sample Rate	31
Figure 3.1-Functional Block Diagram	33
Figure 3.2-Digital Input Analog Output	34
Figure 3.3-Pin diagram of AD 7521	34
Figure 3.4-Functional Diagram of AD7521	35
Figure 3.5-Circuit Diagram for Unipolar Binary Operation	35
Figure 3.6-Circuit Diagram for Bipolar Operation	37
Figure 3.7-Selection of Channels	38
Figure 3.8-Pin diagram of LS154	39
Figure 3.9-Pin diagram of CD4066	40
Figure 3.10-Working of CD4066	41
Figure 3.11-Pin Diagram of LS04	41
Figure 3.12-Circuit Diagram for Selecting Channels	42
Figure 3.13-Pin Diagram of ICL7135	44
Figure 3.14-Block Diagram of Analog Section of ICL7135	44
Figure 3.15-Timing Diagram for Outputs	50

Figure 3.16-Pin Diagram of CD4520	52
Figure 3.17-Circuit Diagram for Reading Data	53
Figure 3.18-Pin Diagram of 555	54
Figure 3.19-Circuitry for Generating Clock	55
Figure 4.1-Device Object as a Container	60
Figure 5.1-Location of Connectors and Jumpers	77
Figure 5.2-PCI-1751 Block Diagram	80
Figure 5.3-Pin Diagram of PCI-1751	81
Figure 5.4-Wet and Dry Contact Inputs	85
Figure 5.5-Timer and Counter Structure	86
Figure 5.6-Interrupt Sources	88
Figure 6.1-Elementary Geometry of Resistance Measurements	91
Figure 6.2-Resistances Connected in (a) Series (b) Parallel	91
Figure 6.3-Sheet Resistance	92
Figure 6.4-Two-Probe Resistance Measurements	93
Figure 6.5-Four-Probe Resistance Measurements	93
Figure 6.6-Slab-Shape Pattern of Thin Film	94
Figure 6.7-Van der Pauw Method	95
Figure 6.8-The Relationship between f and Q	96
Figure 6.9-Sheet Resistance of a Large-Area Thin Film	97
Figure 6.10-Bias Reversal Schematic	99
Figure 6.11-Frequency Spectrum of Interference Signal	100
Figure 6.12-Shielding from RFI/EMI Signals	101
Figure 6.13-Thermal Noise Voltage as a Function of Resistance and Bandwidth	102
Figure 7.1 (a)-Hardware (Picture-1)	104
Figure 7.1 (b)-Hardware (Picture-2)	105
Figure 7.2 (a)-Applied Voltage Vs Measured Voltage of ADC (ref=1.2V)	106
Figure 7.2 (b)-Applied Voltage Vs Measured Voltage of ADC (ref=1.002V)	107
Figure 7.3-Written Data (in decimal) Vs Measured Data (in Volts) of DAC1	108
Figure 7.4-Written Data (in decimal) Vs Measured Data (in Volts) of DAC2	109
Figure 7.5 (a)-Schematic Circuit Diagram for I-V Characteristic of Resistor	110

Figure 7.5 (b)-Circuit for I-V Characteristic of Resistor	111
Figure 7.6-I-V Characteristic of Resistor	112
Figure 7.7 (a)-Schematic Circuit Diagram for I-V Characteristic of Diode	113
Figure 7.7 (b)-Circuit for I-V Characteristic of Diode	114
Figure 7.8-I-V Characteristic of Diode	115
Figure 7.9 (a)-Schematic Circuit Diagram for Measuring Resistivity of Thin Film	116
Figure 7.9 (b)-Thin Film with Electrode Connections	117
Figure 7.9 (c)-Circuit for Measuring Resistivity of Thin Film	118

LIST OF TABLES

Table 2.1-Common Analog Sensors	7
Table 2.2-Relationship between Input Range, Gain, and Precision	27
Table 3.1-Code Table-Unipolar Binary Operation	36
Table 3.2-Code Table-Bipolar (Offset Binary) Operation	38
Table 3.3-Truth Table LS154	40
Table 3.4-Truth Table of LS04	41
Table 4.1-Descriptive Digital I/O Properties	60
Table 4.2-Descriptive Digital I/O Line Properties	62
Table 4.3-Digital I/O Timer Event Properties	70
Table 5.1-Summary of Jumper Settings	79
Table 5.2-Bit Map of Port Configuration Register	83
Table 5.3-Interrupt Control Register Bit Map	87
Table 5.4-Interrupt Mode Bit Values	89
Table 5.5-Triggering Edge Control Bit Values	89
Table 5.6-Interrupt Flag Bit Values	89

INTRODUCTION

In the area of measurements, a big part of the technical expertise is gained from practical experience. Therefore, the learning period require to become a skilled engineer is usually very long. Nowadays, when more industries /companies go into the area of measurement by choosing hardware and software solutions instead old manual methods there is a rising need for more efficient measurement methods for engineers. The easy availability of personal computers has revolutionized the area of automated measurement and control of experiments. In olden times the experiments were either performed manually or by dedicated instruments incorporating embedded microprocessors. These instruments could perform a specified job in a unique configuration using a program which was stored in ROM, thus prohibiting any modification in terms of data acquisition or control of the experiment.

The programming ease and falling cost of PC make it very attractive to be used in automated measurements. Further, the availability of high performance computing software can be used for data management, processing and presentation.

Computer aided measurements need interfacing hardware which has been presented in this dissertation. First, the subject of data acquisition has been structured into adequate work areas and the required human knowledge for engineers in the measurement area has been classified and described. Each part of the hardware design including, unipolar and bipolar configuration of digital to analog converter for writing digital data, selecting channel and acquiring data using decoder, reading digital data and controlling parameters using analog to digital converter, generating clock signal have been explained. In a subject like measurement, which is strongly connected to practical knowledge, it is vital that presented theory is accompanied by practical related information thus finally required programs and graphs have been presented to support the proper working of interface using MATLAB and data acquisition toolbox.

From a technical point of view, for automated measurement, knowledge in three main areas is needed: component behaviors, circuit performance and programming language. To facilitate the learning of component behaviors and circuit performance in this dissertation, there is a methodology presented how to develop comprehensible hardware for automated measurement

and to facilitate the understanding of programming language, structured programs for data acquisition have been explained.

This work may help engineers to find more structured ways of designing data acquisition hardware.

DATA ACQUISITION

2.1-The Data Acquisition System

2.1.1-Introduction

The purpose of any data acquisition system is to provide the tools and resources necessary to measure and analyze physical phenomena [16].

Data acquisition system can be thought as a collection of software and hardware that connects us to the physical world. A typical data acquisition system consists of these components:

- **Data Acquisition Hardware**

At the heart of any data acquisition system lies the data acquisition hardware. The main function of this hardware is to convert analog signals to digital signals, and to convert digital signals to analog signals.

- **Sensors and Actuators (transducers)**

Sensors and actuators can both be *transducers*. A transducer is a device that converts input energy of one form into output energy of another form. For example, a microphone is a sensor that converts sound energy (in the form of pressure) into electrical energy, while a loudspeaker is an actuator that converts electrical energy into sound energy.

- **Signal Conditioning Hardware**

Sensor signals are often incompatible with data acquisition hardware. To overcome this incompatibility, the signal must be conditioned. For example, we may need to condition an input signal by amplifying it or by removing unwanted frequency components. Output signals may need conditioning as well.

- **Computer**

The computer provides a processor, a system clock, a bus to transfer data, and memory and disk space to store data.

- **Software**

Data acquisition software allows us to exchange information between the computer and the hardware. For example, typical software allows us to configure the sampling rate of our board, and acquire a predefined amount of data.

The data acquisition components, and their relationship to each other, are shown below.

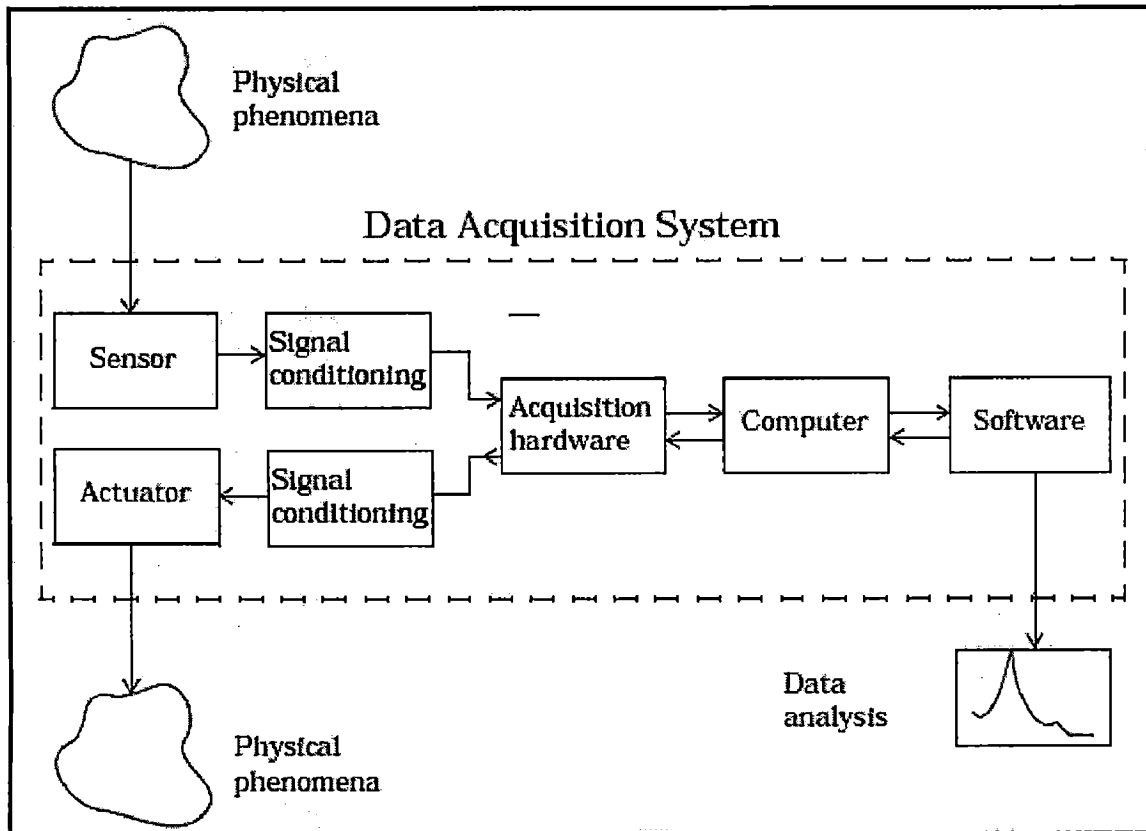


Figure 2.1-Components of Data Acquisition

The Figure 2.1 depicts the two important features of a data acquisition system:

- Signals are input to a sensor, conditioned, converted into bits that a computer can read, and analyzed to extract meaningful information. For example, sound level data is acquired from a microphone, amplified, digitized by a sound card, and stored in MATLAB for subsequent analysis of frequency content.
- Data from a computer is converted into an analog signal and output to an actuator. For example, a vector of data in MATLAB is converted to an analog signal by a sound card and output to a loudspeaker.

2.1.2-Data Acquisition Hardware

Data acquisition hardware is either internal and installed directly into an expansion slot inside our computer, or external and connected to our computer through an external cable. For example, VXI modules are installed in an external VXI chassis, and data is transferred between MATLAB and the module using an external link such as FireWire.

At the simplest level, data acquisition hardware is characterized by the *subsystems* it possesses. A subsystem is a component of our data acquisition hardware that performs a specialized task.

Common subsystems include:

- Analog input
- Analog output
- Digital input/output
- Counter/timer

Hardware devices that consist of multiple subsystems, such as the one depicted below in Figure 2.2, are called *multifunction boards*.

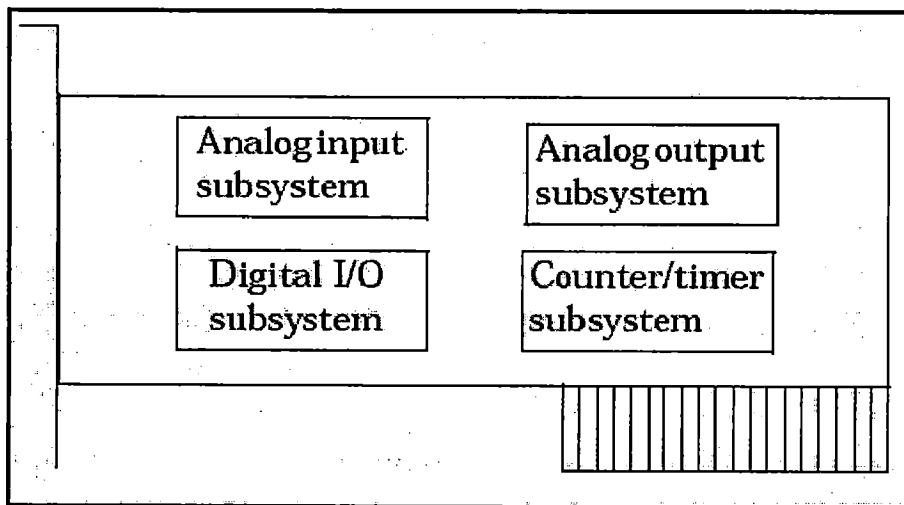


Figure 2.2-Multifunction Board

Analog Input Subsystems

Analog input subsystems convert real-world analog input signals from a sensor into bits that can be read by computer. Perhaps the most important of all the subsystems commonly available are typically multichannel devices offering 12 or 16 bits of resolution.

Analog input subsystems are also referred to as AI subsystems, A/D converters, or ADCs.

Analog Output Subsystems

Analog output subsystems convert digital data stored on computer to a real-world analog signal. These subsystems perform the inverse conversion of analog input subsystems. Typical acquisition boards offer two output channels with 12 bits of resolution, with special hardware available to support multiple channel analog output operations.

Analog output subsystems are also referred to as AO subsystems, D/A converters, or DACs.

Digital Input/Output Subsystems

Digital input/output (DIO) subsystems are designed to input and output digital values (logic levels) to and from hardware. These values are typically handled either as single bits or *lines*, or as a *port*, which typically consists of eight lines.

While most popular data acquisition cards include some digital I/O capability, it is usually limited to simple operations, and special dedicated hardware is often necessary for performing advanced digital I/O operations.

Counter/Timer Subsystems

Counter/Timer (C/T) subsystems are used for event counting, frequency and period measurement, and pulse train generation.

2.1.3-Sensors

A sensor converts the physical phenomena of interest into a signal that is input into data acquisition hardware. There are two main types of sensors based on the output they produce: digital sensors and analog sensors.

Digital sensors produce an output signal that is a digital representation of the input signal, and has discrete values of magnitude measured at discrete times. A digital sensor must output logic levels that are compatible with the digital receiver. Some standard logic levels include transistor-transistor logic (TTL) and emitter-coupled logic (ECL). Examples of digital sensors include switches and position encoders.

Analog sensors produce an output signal that is directly proportional to the input signal, and is continuous in both magnitude and in time. Most physical variables such as temperature, pressure, and acceleration are continuous in nature and are readily measured with an analog sensor. For example, the temperature of an automobile cooling system and the acceleration produced by a child on a swing all vary continuously.

The sensor to be used depends on the phenomena to be measured. Some common analog sensors and the physical variables they measure are listed below.

Sensor	Physical Variable
Accelerometer	Acceleration
Microphone	Pressure
Pressure gauge	Pressure
Resistance temperature device (RTD)	Temperature
Strain gauge	Force
Thermocouple	Temperature

Table 2.1-Common Analog Sensors

When choosing the best analog sensor to use, we must match the characteristics of the physical variable we are measuring with the characteristics of the sensor. The two most important sensor characteristics are:

- The sensor output
- The sensor bandwidth

Sensor Output

The output from a sensor can be an analog signal or a digital signal, and the output variable is usually a voltage although some sensors output current.

- **Current Signals**

Current is often used to transmit signals in noisy environments because it is much less affected by environmental noise. The full scale range of the current signal is often either 4-20 mA or 0-20 mA. A 4-20 mA signal has the advantage that even at minimum signal value, there should be a detectable current flowing. The absence of this indicates a wiring problem.

Before conversion by the analog input subsystem, the current signals are usually turned into voltage signals by a current-sensing resistor. The resistor should be of high precision, perhaps 0.03% or 0.01% depending on the resolution of hardware. Additionally, the voltage signal should

match the signal to an input range of the analog input hardware. For 4-20 mA signals, a 50 ohm resistor will give a voltage of 1 V for a 20 mA signal by Ohm's law.

- **Voltage Signals**

The most commonly interfaced signal is a voltage signal. For example, thermocouples, strain gauges, and accelerometers all produce voltage signals. There are three major aspects of a voltage signal that is needed to be considered:

1. **Amplitude:** If the signal is smaller than a few millivolts, then the signal should be amplified. If it is larger than the maximum range of analog input hardware (typically ± 10 V), it has to be divided down using a resistor network. The amplitude is related to the sensitivity (resolution) of hardware.

2. **Frequency:** Whenever data is acquired, the highest frequency to be measured should be decided. The highest frequency component of the signal determines how often the input should be sampled. If there is more than one input, and only one analog input subsystem, then the overall sampling rate goes up in proportion to the number of inputs. Higher frequencies may be present as noise, which can be removed by filtering the signal before it is digitized.

If the input signal is sampled at least twice as fast as the highest frequency component, then that signal will be uniquely characterized. However, this rate may not mimic the waveform very closely. For a rapidly varying signal, sampling rate of roughly 10 to 20 times the highest frequency is needed to get an accurate picture of the waveform. For slowly varying signals, the minimum time for a significant change in the signal is needed to be considered.

The frequency is related to the bandwidth of measurement. Bandwidth is discussed in the next section.

3. **Duration:** How long should the signal be sampled? If the data is to be stored in memory or to a disk file, then the duration determines the storage resources required. The format of the stored data also affects the amount of storage space required. For example, data stored in ASCII format takes more space than data stored in binary format.

Sensor Bandwidth

In a real-world data acquisition experiment, the physical phenomena to be measured have expected limits. For example, the temperature of automobile's cooling system varies continuously between its low limit and high limit. The temperature limits, as well as how rapidly the temperature varies between the limits, depends on several factors including habit of driving, the weather, and the condition of the cooling system. The expected limits may be readily approximated, but there are an infinite number of possible temperatures that can be measured at a given time. (As explained in section 2.2.2-Quantization on page 16, these unlimited possibilities are mapped to finite set of values by data acquisition hardware.)

The *bandwidth* is given by the range of frequencies present in the signal being measured. The bandwidth can also be thought as being related to the rate of change of the signal. A slowly varying signal has a low bandwidth, while a rapidly varying signal has a high bandwidth. To properly measure the physical phenomena of interest, the sensor bandwidth must be compatible with the measurement bandwidth.

Sensors need to be used with the widest possible bandwidth when making any physical measurement. This is the one way to ensure that the basic measurement system is capable of responding linearly over the full range of interest. However, the wider the bandwidth of the sensor, the more concern should be to eliminating sensor response to unwanted frequency components.

2.1.3-Signal Conditioning

Sensor signals are often incompatible with data acquisition hardware. To overcome this incompatibility, the sensor signal must be conditioned. The type of signal conditioning depends on the sensor being used. For example, a signal may have small amplitude and require amplification, or it may contain unwanted frequency components and require filtering. Common ways to condition signals include:

- Amplification
- Filtering
- Electrical isolation
- Multiplexing
- Excitation source

Amplification

Low-level signals – less than around 100 millivolts – usually need to be amplified. High level signals may also require amplification depending on the input range of the analog input subsystem.

For example, the output signal from a thermocouple is small and must be amplified before it is digitized. Signal amplification allows reduction in noise and use of the full range of hardware, thereby increasing the resolution of the measurement.

Filtering

Filtering removes unwanted noise from the signal of interest. A noise filter is used on slowly varying signals such as temperature to attenuate higher frequency signals that can reduce the accuracy of measurement.

Rapidly varying signals such as vibration often require a different type of filter known as an antialiasing filter. An antialiasing filter removes undesirable higher frequencies that may lead to erroneous measurements.

Electrical Isolation

If the signal of interest may contain high-voltage transients that could damage the computer, then the sensor signals should be electrically isolated from the computer for safety purposes.

Electrical isolation can also be used to ensure that the readings from the data acquisition hardware are not affected by differences in ground potentials. For example, when the hardware device and the sensor signal are each referenced to ground, problems occur if there is a potential difference between the two grounds. This difference can lead to a *ground loop*, which may lead to erroneous measurements. Using electrically isolated signal conditioning modules eliminates the ground loop and ensures that the signals are accurately represented.

Multiplexing

A common technique for measuring several signals with a single measuring device is multiplexing.

Signal conditioning devices for analog signals often provide multiplexing for use with slowly changing signals such as temperature. This is in addition to any built-in multiplexing on the DAQ board. The A/D converter samples one channel, switches to the next channel and samples it, switches to the next channel, and so on. Because the same A/D converter is sampling many

channels, the effective sampling rate of each individual channel is inversely proportional to the number of channels sampled.

Care should be taken when using multiplexers so that the switched signal has sufficient time to settle.

Excitation Source

Some sensors require an excitation source to operate. For example, strain gauges, and resistive temperature devices (RTDs) require external voltage or current excitation. Signal conditioning modules for these sensors usually provide the necessary excitation. RTD measurements are usually made with a current source that converts the variation in resistance to a measurable voltage.

2.1.4-Computer

The computer provides a processor, a system clock, a bus to transfer data, and memory and disk space to store data.

The processor controls how fast data is accepted by the converter. The system clock provides time information about the acquired data. Recording sensor reading is generally not enough. It is also needed to know when that measurement occurred.

Data is transferred from the hardware to system memory via dynamic memory access (DMA) or interrupts. DMA is hardware controlled and therefore extremely fast. Interrupts may be slow due to the latency time between when a board requests interrupt servicing and when the computer responds. The maximum acquisition rate is also determined by the computer's bus architecture.

2.1.5-Software

Regardless of the hardware being used, information must be sent to the hardware and information should be received from the hardware. Configuration information is sent to the hardware such as the sampling rate, and information is received from the hardware such as data, status messages, and error messages. The information needs to be supplied to the hardware so that it can be integrated with other hardware and with computer resources. This information exchange is accomplished with software.

There are two kinds of software:

- Driver software
- Application software

For example, suppose the Data Acquisition Toolbox is being used with an Advantech PCI-1751 48-bit Digital Input/Output Card and its associated Adv-DAQ driver.

The relationship between us, the driver software, the application software, and the hardware is shown below.

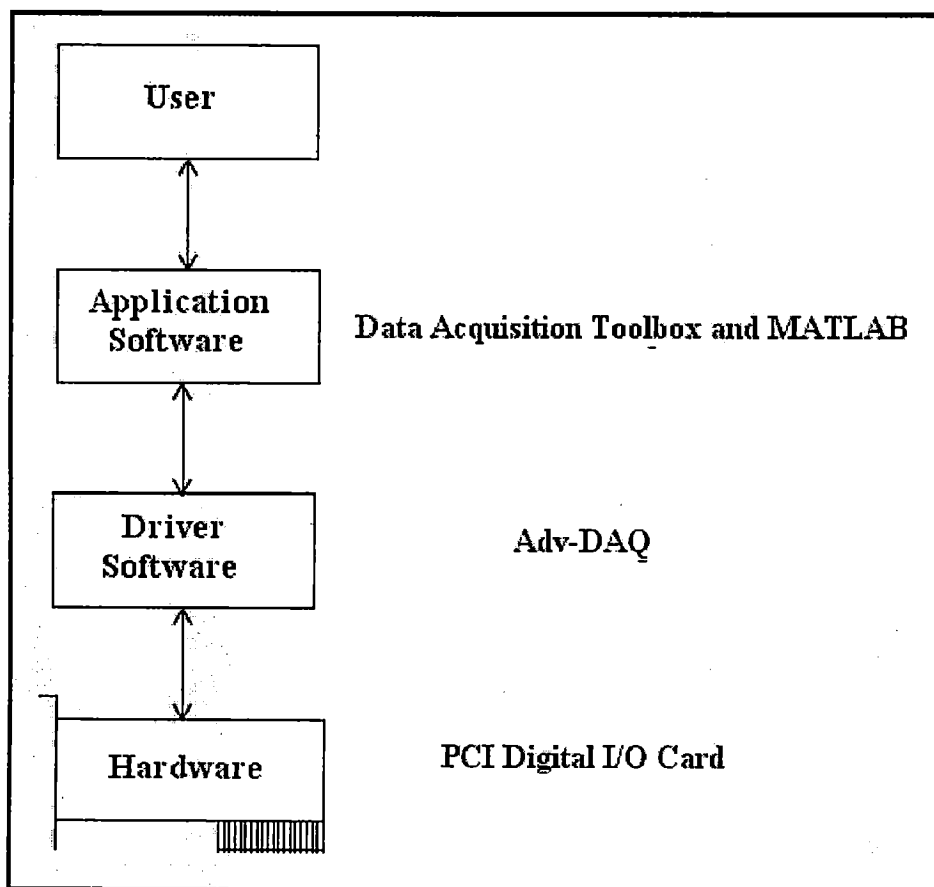


Figure 2.3-Relationship between User, Driver Software, Application Software and Hardware

The diagram illustrates that information is supplied to the hardware, and information is received from the hardware.

Driver Software

For data acquisition device, there is associated driver software that must be used. Driver software allows accessing and controlling the capabilities of hardware. Among other things, basic driver software allows to:

- Bring data on to and get data off of the board
- Control the rate at which data is acquired
- Integrate the data acquisition hardware with computer resources such as processor interrupts, DMA, and memory
- Integrate the data acquisition hardware with signal conditioning hardware
- Access multiple subsystems on a given data acquisition board
- Access multiple data acquisition boards

Application Software

Application software provides a convenient “front-end” to the driver software. Basic application software allows us to:

- Report relevant information such as the number of samples acquired
- Generate events
- Manage the data stored in computer memory
- Condition a signal
- Plot acquired data

2.2-The Analog Input Subsystem

Many data acquisition hardware devices contain one or more subsystems that convert (digitize) real-world sensor signals into numbers computer can read. Such devices are called analog input subsystems (AI subsystems, A/D converters, or ADCs). After the real world signal is digitized, it can be analyzed, it can be stored in system memory, or it can be stored to a disk file.

The function of the analog input subsystem is to *sample* and *quantize* the analog signal using one or more *channels*. The channel can be thought as a path through which the sensor signal travels. Typical analog input subsystems have eight or 16 input channels available. After data is sampled and quantized, it must be transferred to system memory.

Analog signals are continuous in time and in amplitude (within predefined limits). Sampling takes a “snapshot” of the signal at discrete times, while quantization divides the voltage (or current) value into discrete amplitudes. Sampling, quantization, channel configuration, and transferring data from hardware to system memory are discussed below.

2.2.1-Sampling

Sampling takes a “snapshot” of the sensor signal at discrete times. For most applications, the time interval between samples is kept constant (for example, sample every millisecond) unless externally clocked.

For most digital converters, sampling is performed by a sample and hold (S/H) circuit. An S/H circuit usually consists of a signal buffer followed by an electronic switch connected to a capacitor. The operation of an S/H circuit follows these steps:

1. At a given sampling instant, the switch connects the buffer and capacitor to an input.
2. The capacitor is charged to the input voltage.
3. The charge is held until the A/D converter digitizes the signal.
4. For multiple channels connected (multiplexed) to one A/D converter, the previous steps are repeated for each input channel.
5. The entire process is repeated for the next sampling instant.

Hardware can be divided into two main categories based on how signals are sampled: *scanning* hardware, which samples input signals sequentially, and *simultaneous sample and hold (SS/H)* hardware, which samples all signals at the same time. These two types of hardware are discussed below.

Scanning Hardware

Scanning hardware samples a single input signal, converts that signal to a digital value, and then repeats the process for every input channel used. In other words, each input channel is sampled sequentially. A *scan* occurs when each input in a group is sampled once. As shown below, data acquisition device should have one A/D converter that is multiplexed to multiple input channels.

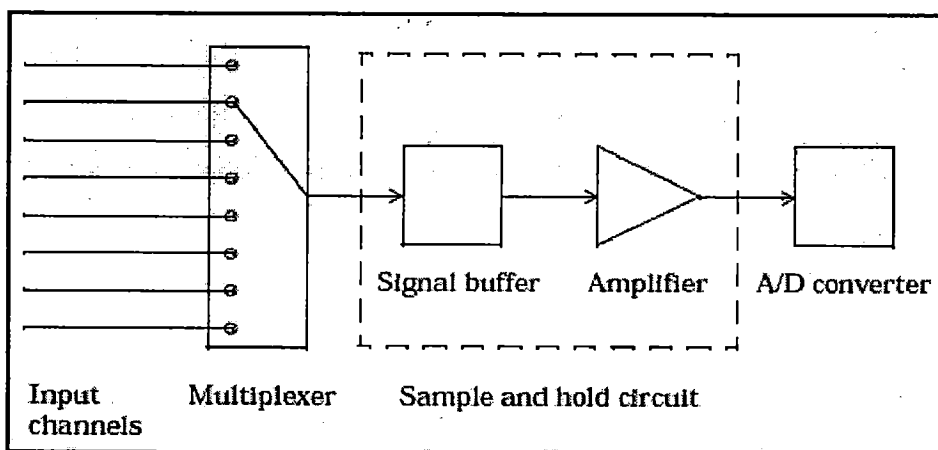


Figure 2.4-ADC with Multiplexer

Therefore, if the multiple channels are used, those channels cannot be sampled simultaneously and a time gap exists between consecutive sampled channels. This time gap is called the *channel skew*. The channel skew can be thought as the time it takes the analog input subsystem to sample a single channel.

Additionally, the maximum sampling rate hardware is rated at typically applies for one channel. Therefore, the maximum sampling rate per channel is given by the formula:

$$\text{Maximum Sampling rate per channel} = (\text{Maximum Board Rate}) / (\text{No of channels scanned})$$

Typically, this maximum rate can be achieved only under ideal conditions. In practice, the sampling rate depends on several characteristics of the analog input subsystem including the settling time and the gain, as well as the channel skew. The sample period and channel skew for a multichannel configuration using scanning hardware is shown in Figure 2.3.

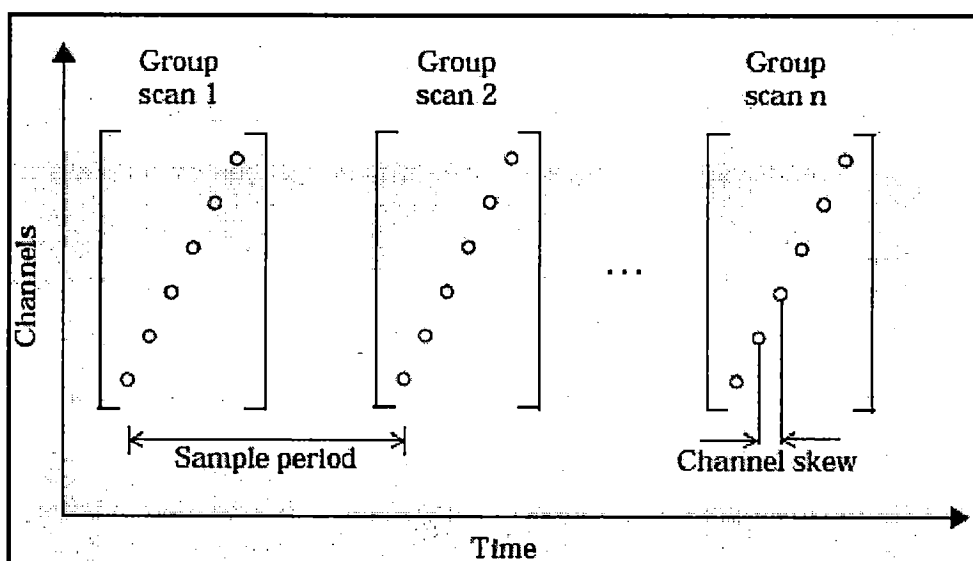


Figure 2.5-Plot of Channels and Time using Scanning Hardware

If the channel skew can not be tolerated in an application, then the hardware that allows simultaneous sampling of all channels must be used. Simultaneous sample and hold hardware is discussed in the next section.

Simultaneous Sample and Hold Hardware

Simultaneous sample and hold (SS/H) hardware samples all input signals at the same time and holds the values until the A/D converter digitizes all the signals. For high-end systems, there can be a separate A/D converter for each input channel.

For example, suppose it is needed to simultaneously measure the acceleration of multiple accelerometers to determine the vibration of some device under test. To do this, SS/H hardware must be used since it does not have a channel skew. In general, SS/H hardware is used if sensor signal changes significantly in a time that is less than the channel skew, or if transfer function is to be used or frequency domain correlation is to be performed.

The sample period for a multichannel configuration using SS/H hardware is shown in Figure 2.6.

Note that there is no channel skew.

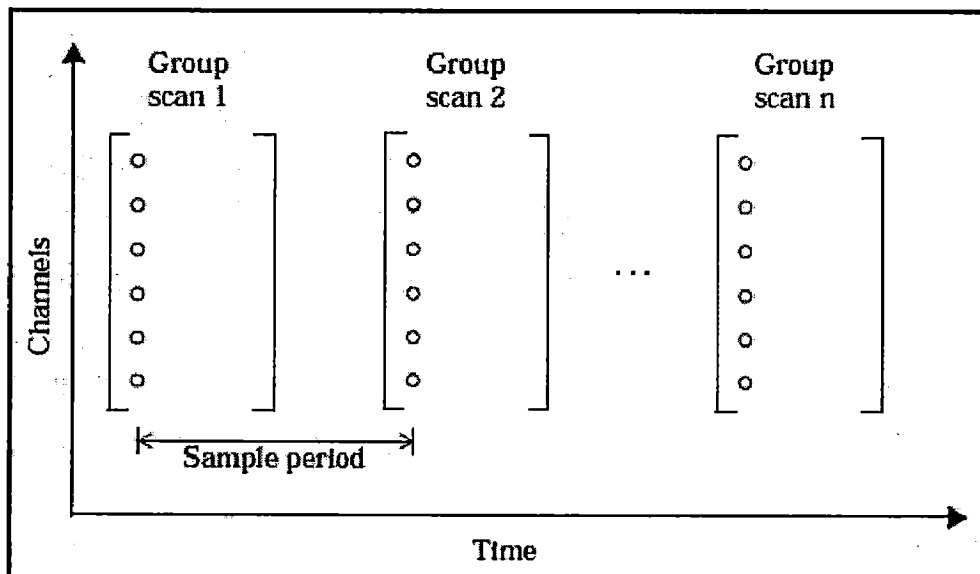


Figure 2.6-Plot of Channels and Time using SS/H Hardware

2.2.2-Quantization

As discussed in the previous section, sampling takes a snapshot of the input signal at an instant of time. When the snapshot is taken, the sampled analog signal must be converted from a voltage value to a binary number that the computer can read. The conversion from infinitely precise amplitude to a binary number is called *quantization*.

During quantization, the A/D converter uses a finite number of evenly spaced values to represent the analog signal. The number of different values is determined by the number of bits used for

the conversion. Most modern converters use 12 or 16 bits. Typically, the converter selects the digital value that is closest to the actual sampled value.

The Figure 2.7 shows a 1 Hz sine wave quantized by a 3-bit A/D converter.

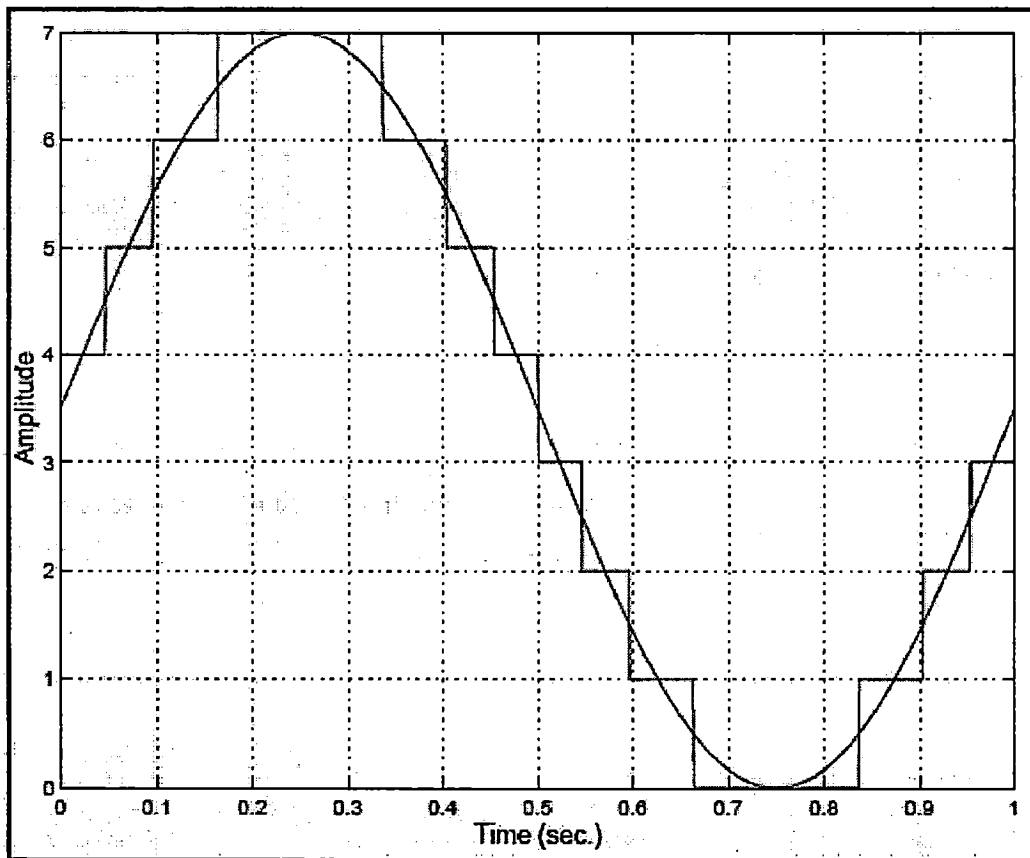


Figure 2.7-Sine Wave Quantized by ADC

The number of quantized values is given by $2^3 = 8$, the largest representable value is given by $111 = 2^2 + 2^1 + 2^0 = 7.0$, and the smallest representable value is given by $000 = 0.0$.

Quantization Error

There is always some error associated with the quantization of a continuous signal. Ideally, the maximum quantization error is ± 0.5 least significant bits (LSBs), and over the full input range, the average quantization error is zero.

As shown in Figure 2.8, the quantization error for the previous sine wave is calculated by subtracting the actual signal from the quantized signal.

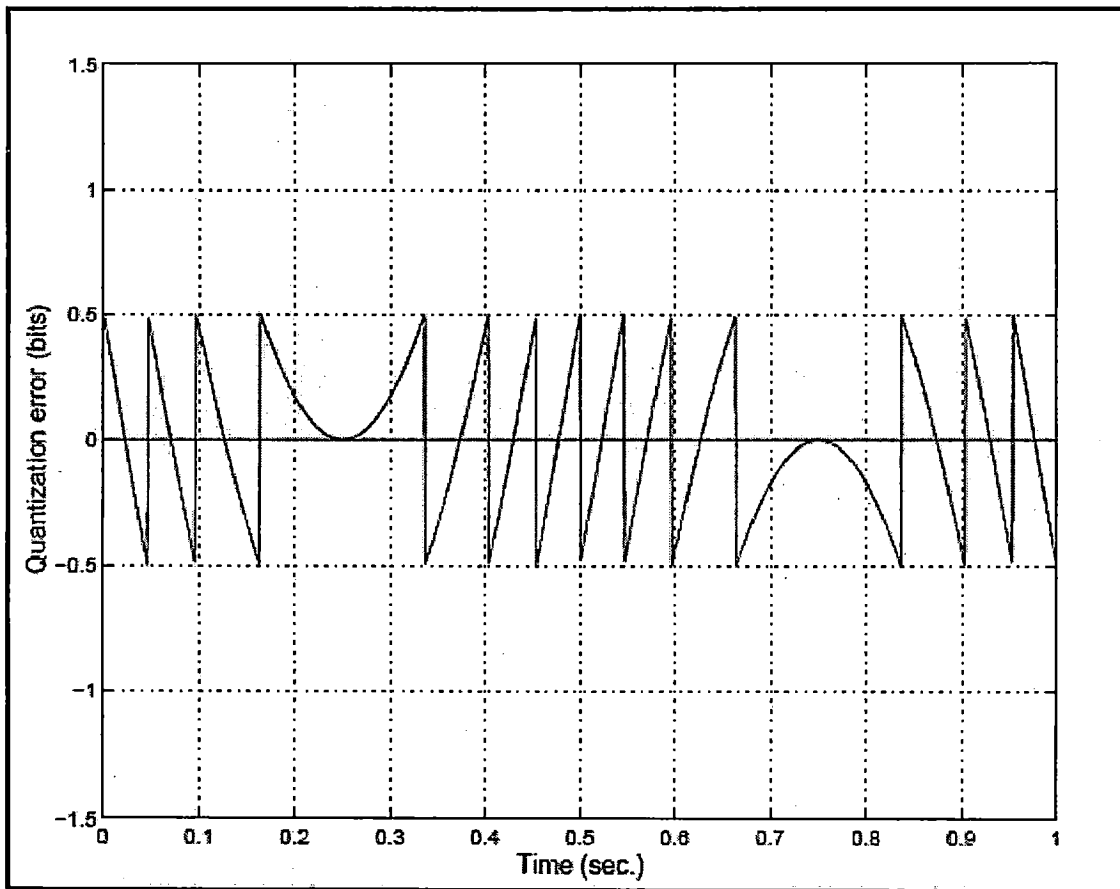


Figure 2.8-Plot Quantization Error and Time

Input Range and Polarity

The *input range* of the analog input subsystem is the span of input values for which a conversion is valid. Input range can be changed by selecting a different *gain* value. For example, National Instruments' AT-MIO-16E-1 board has eight gain values ranging from 0.5 to 100. Many boards include a programmable gain amplifier that allows changing the device gain through software.

When an input signal exceeds the valid input range of the converter, an *overrange* condition occurs. In this case, most devices saturate to the largest representable value, and the converted data is almost definitely incorrect. The gain setting affects the precision of measurement – the higher (lower) the gain value, the lower (higher) the precision.

An analog input subsystem can typically convert both *unipolar* signals and *bipolar* signals. A unipolar signal contains only positive values and zero, while a bipolar signal contains positive values, negative values, and zero. Unipolar and bipolar signals are depicted below. Refer to the Figure 2.7 on page 17 in section 2.2.2 for an example of a unipolar signal.

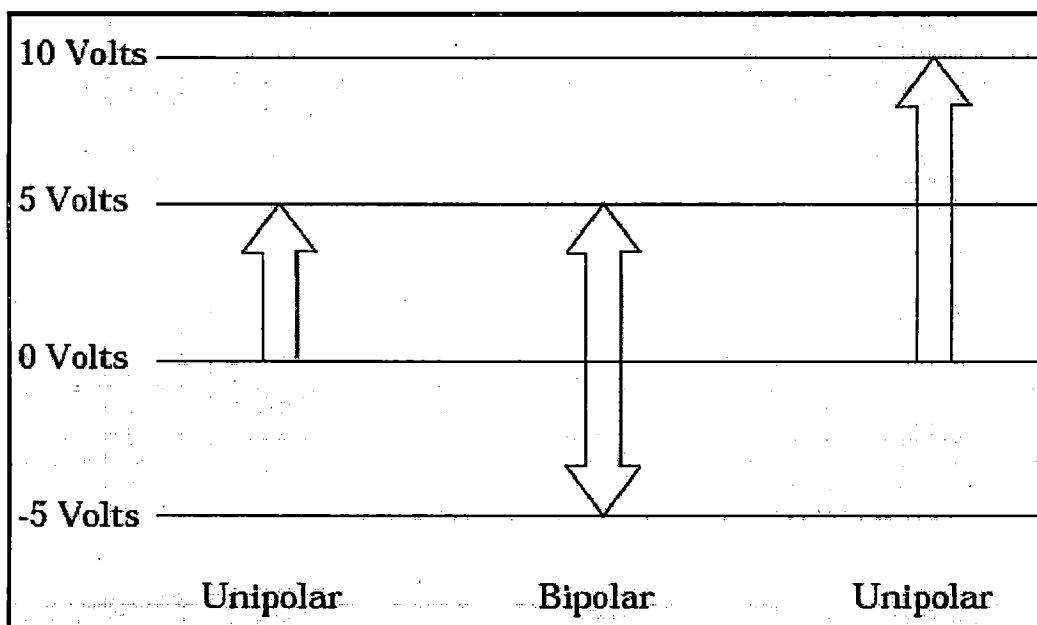


Figure 2.9-Unipolar and Bipolar Signal

In many cases, the signal polarity is a fixed characteristic of the sensor and the input range must be configured to match this polarity.

As it can be seen, it is crucial to understand the range of signals expected from sensor so that the input range of the analog input subsystem can be configured to maximize resolution and minimize the chance of an overrange condition.

How Are Acquired Samples Clocked?

Samples are acquired from an analog input subsystem at a specific rate by a clock. Like any timing system, data acquisition clocks are characterized their resolution and accuracy. Timing resolution is defined as the smallest time interval that can be measured accurately. The timing accuracy is affected by clock *jitter*. Jitter arises when a clock produces slightly different values for a given time interval.

For any data acquisition system, there are typically three clock sources that can be used: the onboard data acquisition clock, the computer clock, or an external clock. The Data Acquisition Toolbox supports all of these clock sources, depending on the requirements of the hardware.

- **The Onboard Clock:** The onboard clock is typically a timer chip on the hardware board that is programmed to generate a pulse stream at the desired rate. The onboard clock generally has high accuracy and low jitter compared to the computer clock. The onboard clock should always be used when the sampling rate is high and when a fixed time interval between samples is required. The onboard clock is referred to as the *internal clock* in this report.

- **The Computer Clock:** The computer (PC) clock is used for boards that do not possess an onboard clock. The computer clock is less accurate and has more jitter than the onboard clock, and is generally limited to sampling rates below 500 Hz. The computer clock is referred to as the *software clock* in this report.
- **External Clock:** An external clock is often used when the sampling rate is low and not constant. For example, an external clock source is often used in automotive applications where samples are acquired as a function of crank angle.

2.2.3-Channel Configuration

The input channels can be configured in any one of these two ways:

- Differential
- Single-ended

The choice of input channel configuration may depend on whether the input signal is *floating* or *grounded*.

A floating signal uses an isolated ground reference and is not connected to the building ground. As a result, the input signal and hardware device are not connected to a common reference, which can cause the input signal to exceed the valid range of the hardware device. To circumvent this problem, the signal must be connected to the onboard ground of the device. Examples of floating signal sources include ungrounded thermocouples and battery devices.

A grounded signal is connected to the building ground. As a result, the input signal and hardware device are connected to a common reference. Examples of grounded signal sources include nonisolated instrument outputs and devices that are connected to the building power system.

Note: For more information about channel configuration, refer to hardware documentation.

- **Differential Inputs**

When the hardware is configured for differential input, there are two signal wires associated with each input signal – one for the input signal and one for the reference (return) signal. The measurement is the difference in voltage between the two wires, which helps reduce noise and any voltage that is common to both wires.

As shown in Figure 2.10, the input signal is connected to the positive amplifier socket (labeled +) and the return signal is connected to the negative amplifier socket (labeled –). The amplifier has a third connector that allows these signals to be referenced to ground.

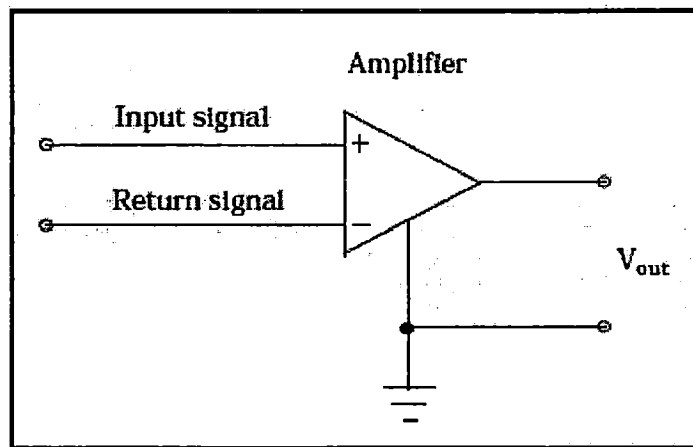


Figure 2.10-Differential Input

It is recommended that the differential inputs should be used under any of these conditions:

- ✓ The input signal is low level (less than 1 volt).
- ✓ The leads connecting the signal are greater than 10 feet.
- ✓ The input signal requires a separate ground-reference point or return signal.
- ✓ The signal leads travel through a noisy environment.

• **Single-Ended Inputs**

When the hardware is configured for single-ended input, there is one signal wire associated with each input signal, and each input signal is connected to the same ground. Single-ended measurements are more susceptible to noise than differential measurements due to differences in the signal paths.

As shown in Figure 2.11, the input signal is connected to the positive amplifier socket (labeled +) and the ground is connected to the negative amplifier socket (labeled -).

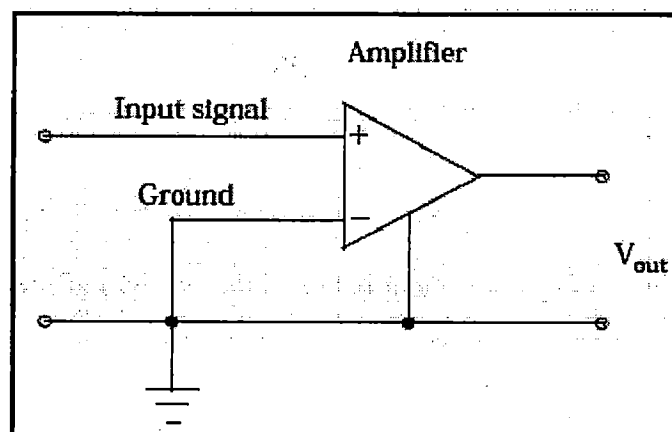


Figure 2.11-Single Ended Input

It is suggested that single-ended inputs should be used under any of these conditions:

- ✓ The input signal is high level (greater than 1 volt).
- ✓ The leads connecting the signal are less than 10 feet.
- ✓ The input signal can share a common reference point with other signals.

The differential input connectors should be used for any input signal that does not meet the preceding conditions.

2.2.4-Transferring Data from Hardware to System Memory

The transfer of acquired data from the hardware to system memory follows these steps:

1. Acquired data is stored in the hardware's first-in first-out (FIFO) buffer.
2. Data is transferred from the FIFO buffer to system memory using interrupts or DMA.

These steps happen automatically. Typically, all that's required from us is some initial configuration of the hardware device when it is installed.

The FIFO Buffer

The FIFO buffer is used to temporarily store acquired data. The data is temporarily stored until it can be transferred to system memory. The process of transferring data into and out of an analog input FIFO buffer is given below:

1. The FIFO buffer stores newly acquired samples at a constant sampling rate.
2. Before the FIFO buffer is filled, the software starts removing the samples. For example, an interrupt is generated when the FIFO is half full, and signals the software to extract the samples as quickly as possible.
3. Since servicing interrupts or programming the DMA controller can take up to a few milliseconds, additional data is stored in the FIFO for future retrieval. For a larger FIFO buffer, longer latencies can be tolerated.
4. The samples are transferred to system memory via the system bus (for example, PCI bus or AT bus). After the samples are transferred, the software is free to perform other tasks until the next interrupt occurs. For example, the data can be processed or saved to a disk file. As long as the average rates of storing and extracting data are equal, acquired data will not be missed and any application should run smoothly.

Interrupts

The slowest but most common method to move acquired data to system memory is for the board to generate an interrupt request (IRQ) signal. This signal can be generated when one sample is acquired or when multiple samples are acquired. The process of transferring data to system memory via interrupts is given below:

1. When data is ready for transfer, the CPU stops whatever it is doing and runs special interrupt handler routine that saves the current machine registers, and then sets them to access the board.
2. The data is extracted from the board and placed into system memory.
3. The saved machine registers are restored, and the CPU returns to the original interrupted process.

The actual data move is fairly quick, but there is a lot of overhead time spent saving, setting up, and restoring the register information. Therefore, depending on specific system, transferring data by interrupts may not be a good choice when the sampling rate is greater than around 5 kHz.

Direct Memory Access (DMA)

Direct memory access (DMA) is a system whereby samples are automatically stored in system memory while the processor does something else. The process of transferring data via DMA is given below:

1. When data is ready for transfer, the board directs the system DMA controller to put it into system memory as soon as possible.
2. As soon as the CPU is able (which is usually very quickly), it stops interacting with the data acquisition hardware and the DMA controller moves the data directly into memory.
3. The DMA controller gets ready for the next sample by pointing to the next open memory location.
4. The previous steps are repeated indefinitely, with data going to each open memory location in a continuously circulating buffer. No interaction between the CPU and the board is needed.

Computer supports several different DMA channels. Depending on the application, one or more of these channels can be used, For example, simultaneous input and output with a sound card requires one DMA channel for the input and another DMA channel for the output.

2.3-Making Quality Measurements

For most data acquisition applications, the signal produced by a sensor needs to be measured at a specific rate.

In many cases, the sensor signal is a voltage level that is proportional to the physical phenomena of interest (for example, temperature, pressure, or acceleration). If the slowly changing (quasi-static) phenomena like temperature is being measured a slow sampling rate usually suffices. If the rapidly changing (dynamic) phenomena like vibration or acoustic measurements are being measured, a fast sampling rate is required.

To make high-quality measurements, these rules should be followed:

- Maximize the precision and accuracy
- Minimize the noise
- Match the sensor range to the A/D range

2.3.1-Accuracy and Precision

Whenever measured data is acquired, every effort should be made to maximize its accuracy and precision. The quality of measurement depends on the accuracy and precision of the entire data acquisition system, and can be limited by such factors as board resolution or environmental noise.

In general terms, the *accuracy* of a measurement determines how close the measurement comes to the true value. Therefore, it indicates the correctness of the result. The *precision* of a measurement reflects how exactly the result is determined without reference to what the result means. The *relative precision* indicates the uncertainty in a measurement as a fraction of the result.

For example, suppose a table top is measured with a meter stick and its length is found to be 1.502 meters. This number indicates that the meter stick (and our eyes) can resolve distances down to at least a millimeter. Under most circumstances, this is considered to be a fairly precise measurement with a relative precision of around 1/1500. However, suppose the measurement is performed again and a result of 1.510 meters is obtained. After careful consideration, it is discovered that initial technique for reading the meter stick was faulty because it was not read from directly above. Therefore, the first measurement was not accurate.

Precision and accuracy are illustrated in Figure 2.12.

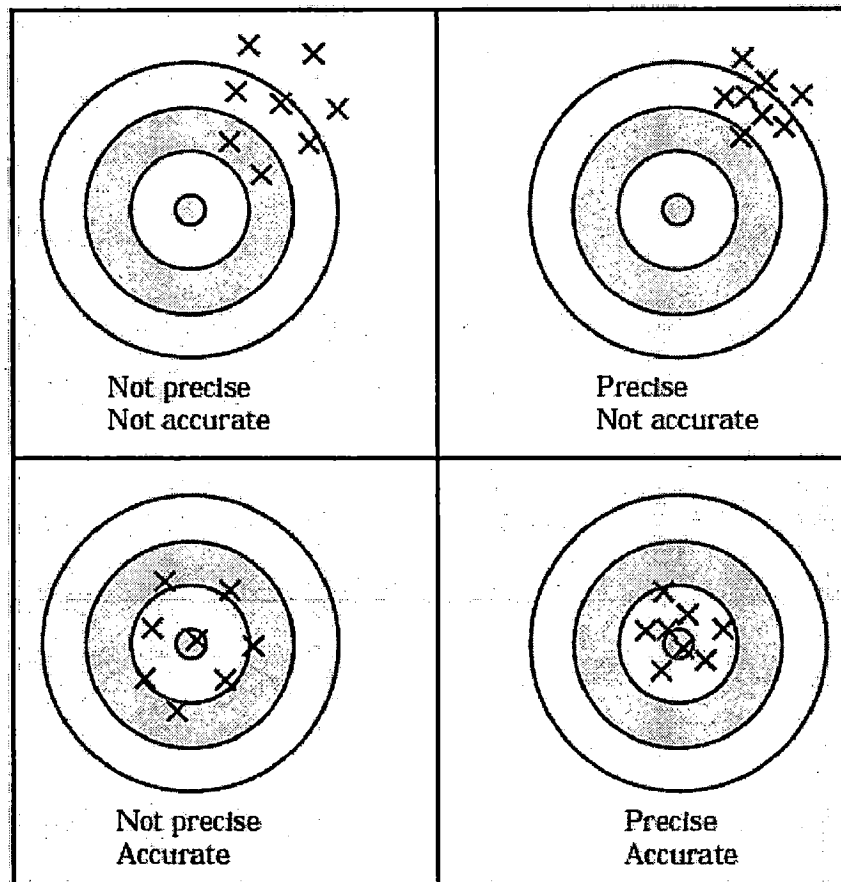


Figure 2.12-Precision Vs Accuracy

For analog input subsystems, accuracy is usually limited by calibration errors while precision is usually limited by the A/D converter. Accuracy and precision are discussed in more detail below.

Accuracy

Accuracy is defined as the agreement between a measured quantity and the true value of that quantity. Every component that appears in the analog signal path affects system accuracy and performance. The overall system accuracy is given by the component with the worst accuracy.

For data acquisition hardware, accuracy is often expressed as a percent or a fraction of the least significant bit (LSB). Under ideal circumstances, board accuracy is typically ± 0.5 LSB. Therefore, a 12-bit converter has only 11 usable bits.

Many boards include a programmable gain amplifier, which is located just before the converter input. To prevent system accuracy from being degraded, the accuracy and linearity of the gain must be better than that of the A/D converter. The specified accuracy of a board is also affected

by the sampling rate and the *settling time* of the amplifier. The settling time is defined as the time required for the instrumentation amplifier to settle to a specified accuracy. To maintain full accuracy, the amplifier output must settle to a level given by the magnitude of 0.5 LSB before the next conversion, and is on the order of several tenths of a millisecond for most boards.

Settling time is a function of sampling rate and gain value. High rate, high gain configurations require longer settling times while low rate, low gain configurations require shorter settling times.

Precision

The number of bits used to represent an analog signal determines the precision (resolution) of the device. The more bits provided by hardware board, the more precise measurement will be. A high precision, high resolution device divides the input range into more divisions thereby allowing a smaller detectable voltage value. A low precision, low resolution device divides the input range into fewer divisions thereby increasing the detectable voltage value.

The overall precision of data acquisition system is usually determined by the A/D converter, and is specified by the number of bits used to represent the analog signal. Most boards use 12 or 16 bits. The precision of measurement is given by

$$\text{Precision} = \text{one part in } 2^{(\text{number of bits})}$$

The precision in volts is given by

$$\text{Precision} = \text{voltage range} / 2^{(\text{number of bits})}$$

For example, if a 12-bit A/D converter configured for a 10 volt range are being used, then

$$\text{Precision} = 10 \text{ volts} / 2^{(12)}$$

This means that the converter can detect voltage differences at the level of 0.00244 volts (2.44 mV).

How Are Range, Gain, and Measurement Precision Related?

When the input range and gain of analog input subsystem are configured, the end result should maximize the measurement resolution and minimize the chance of an overrange condition. The actual input range is given by the formula:

$$\text{Actual input range} = \text{Input range} / \text{Gain}$$

The relationship between gain, actual input range, and precision for a Unipolar and bipolar signal having an input range of 10 V is shown in Table 2.2.

Input Range	Gain	Actual Input Range	Precision (12 Bit A/D)
0 to 10 V	1.0	0 to 10 V	2.44 mV
	2.0	0 to 5 V	1.22 mV
	5.0	0 to 2 V	0.488 mV
	10.0	0 to 1 V	0.244 mV
-5 to 5 V	0.5	-10 to 10 V	4.88 mV
	1.0	-5 to 5 V	2.44 mV
	2.0	-2.5 to 2.5 V	1.22 mV
	5.0	-1.0 to 1.0 V	0.488 mV
	10.0	-0.5 to 0.5 V	0.244 mV

Table 2.2-Relationship between Input Range, Gain, and Precision

As shown in the table, the gain affects the precision of measurement. If a gain is selected that decreases the actual input range, then the precision increases. Conversely, if a gain is selected that increases the actual input range, then the precision decreases. This is because the actual input range varies but the number of bits used by the A/D converter remains fixed.

Note With the Data Acquisition Toolbox, the range and gain need not to be specified. Instead, the actual input range desired is specified.

2.3.2-Noise

Noise is considered to be any measurement that is not part of the phenomena of interest. Noise can be generated within the electrical components of the input amplifier (internal noise), or it can be added to the signal as it travels down the input wires to the amplifier (external noise). Techniques that can be used to reduce the effects of noise are described below.

- **Removing Internal Noise**

Internal noise arises from thermal effects in the amplifier. Amplifiers typically generate a few microvolts of internal noise, which limits the resolution of the signal to this level. The amount of noise added to the signal depends on the bandwidth of the input amplifier.

To reduce internal noise, an amplifier with a bandwidth that closely matches the bandwidth of the input signal should be selected.

- **Removing External Noise**

External noise arises from many sources. For example, many data acquisition experiments are subject to 50 Hz noise generated by a.c. power circuits. This type of noise is referred to as *pick-up* or *hum*, and appears as a sinusoidal interference signal in the measurement circuit. Another common interference source is fluorescent lighting. These lights generate an arc at twice the power line frequency (100 Hz).

Noise is added to the acquisition circuit from these external sources because the signal leads act as aerials picking up environmental electrical activity.

Much of this noise is common to both signal wires. To remove most of this common-mode voltage:

- ✓ The input channels should be configured in differential mode.
- ✓ Signal wires should be used that are twisted together rather than separate.
- ✓ The signal wires should be kept as short as possible.
- ✓ Signal wires should be kept as far as possible from environmental electrical activity.

- **Filtering**

Filtering also reduces signal noise. For many data acquisition applications, a low-pass filter is beneficial. As the name suggests, a low-pass filter passes the lower frequency components but attenuates the higher frequency components. The cut-off frequency of the filter must be compatible with the frequencies present in the signal of interest and the sampling rate used for the A/D conversion.

A low-pass filter that's used to prevent higher frequencies from introducing distortion into the digitized signal is known as an antialiasing filter if the cut-off occurs at the Nyquist frequency. That is, the filter removes frequencies greater than one-half the sampling frequency. These filters generally have a sharper cut-off than the normal low-pass filter used to condition a signal. Antialiasing filters are specified according to the sampling rate of the system and there must be one filter per input signal.

2.3.3-Matching the Sensor Range and A/D Converter Range

When sensor data is digitized by an A/D converter, these two issues must be considered:

- The expected range of the data produced by sensor. This range depends on the physical phenomena being measured and the output range of the sensor.

- The range of A/D converter. For many devices, the hardware range is specified by the gain and polarity.

The sensor and hardware ranges should be selected such that the maximum precision is obtained, and the full dynamic range of the input signal is covered.

For example, suppose a microphone is being used with a dynamic range of 20 dB to 140 dB and an output sensitivity of 50 mV/Pa. If street noise is being measured in application, then it might be expected that the sound level never exceeds 80 dB, which corresponds to a sound pressure magnitude of 200 mPa and a voltage output from the microphone of 10 mV. Under these conditions, the input range of data acquisition card should be set for a maximum signal amplitude of 10 mV, or a little more.

2.3.4-How Fast Should a Signal Be Sampled?

Whenever a continuous signal is sampled, some information is lost. The key objective is to sample at a rate such that the signal of interest is well characterized and the amount of information lost is minimized.

If the signal is sampled at a rate that is too slow, then the signal is *undersampled*, and *aliasing* can occur. Aliasing can occur for both rapidly varying signals and slowly varying signals. For example, suppose the temperature is being measured once a minute. If the acquisition system is picking up a 50 Hz hum from an a.c power supply, then that hum will appear as constant noise level if it is being sampled at 30 Hz.

Aliasing occurs when the sampled signal contains frequency components greater than one-half the sampling rate. The frequency components could originate from the signal of interest in which case we are undersampling and should increase the sampling rate. The frequency components could also originate from noise in which case we may need to condition the signal using a filter.

The rule used to prevent aliasing is given by the *Nyquist theorem*, which states that:

- An analog signal can be uniquely reconstructed, without error, from samples taken at equal time intervals.
- The sampling rate must be equal to or greater than twice the highest frequency component in the analog signal. A frequency of one-half the sampling rate is called the Nyquist frequency.

However, if the input signal is corrupted by noise, then aliasing can still occur.

For example, suppose the A/D converter is configured to sample at a rate of 4 samples per second (4 S/s or 4 Hz), and the signal of interest is a 1 Hz sine wave. Since the signal frequency is one-fourth the sampling rate, then according to the Nyquist theorem, it should be completely characterized. However, if a 5 Hz sine wave is also present, then these two signals cannot be distinguished. In other words, the 1 Hz sine wave produces the same samples as the 5 Hz sine wave when the sampling rate is 4 S/s. This situation is shown in Figure 2.13.

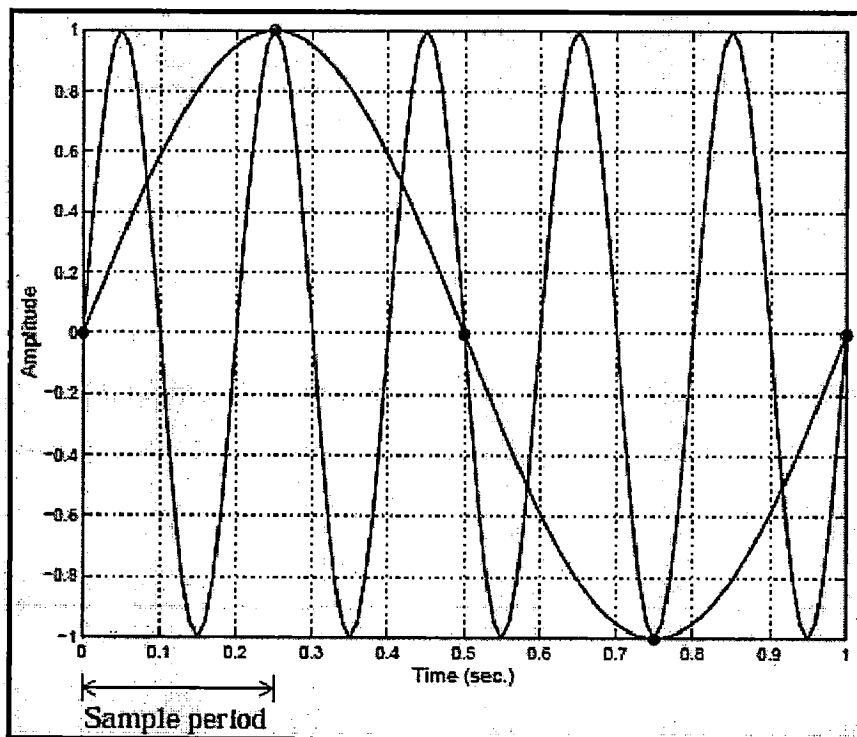


Figure 2.13-Occurance of Aliasing

In a real-world data acquisition environment, it may be needed to condition the signal by filtering out the high frequency components.

Even though the samples appear to represent a sine wave with a frequency of one-fourth the sampling rate, the actual signal could be any sine wave with a frequency of

$$(n \pm 0.25) \times \text{Sampling rate}$$

where n is zero or any positive integer. For this example, the actual signal may be at a frequency of 3 Hz, 5 Hz, 7 Hz, 9 Hz, and so on. The relationship is called the *alias* of a signal that may be at another frequency. In other words, aliasing occurs when one frequency assumes the identity of another frequency.

If the input signal is sampled at least twice as fast as the highest frequency component, then that signal may be uniquely characterized but this rate would not mimic the waveform very closely.

As shown in Figure 2.14, to get an accurate picture of the waveform, a sampling rate of roughly 10 to 20 times the highest frequency is needed.

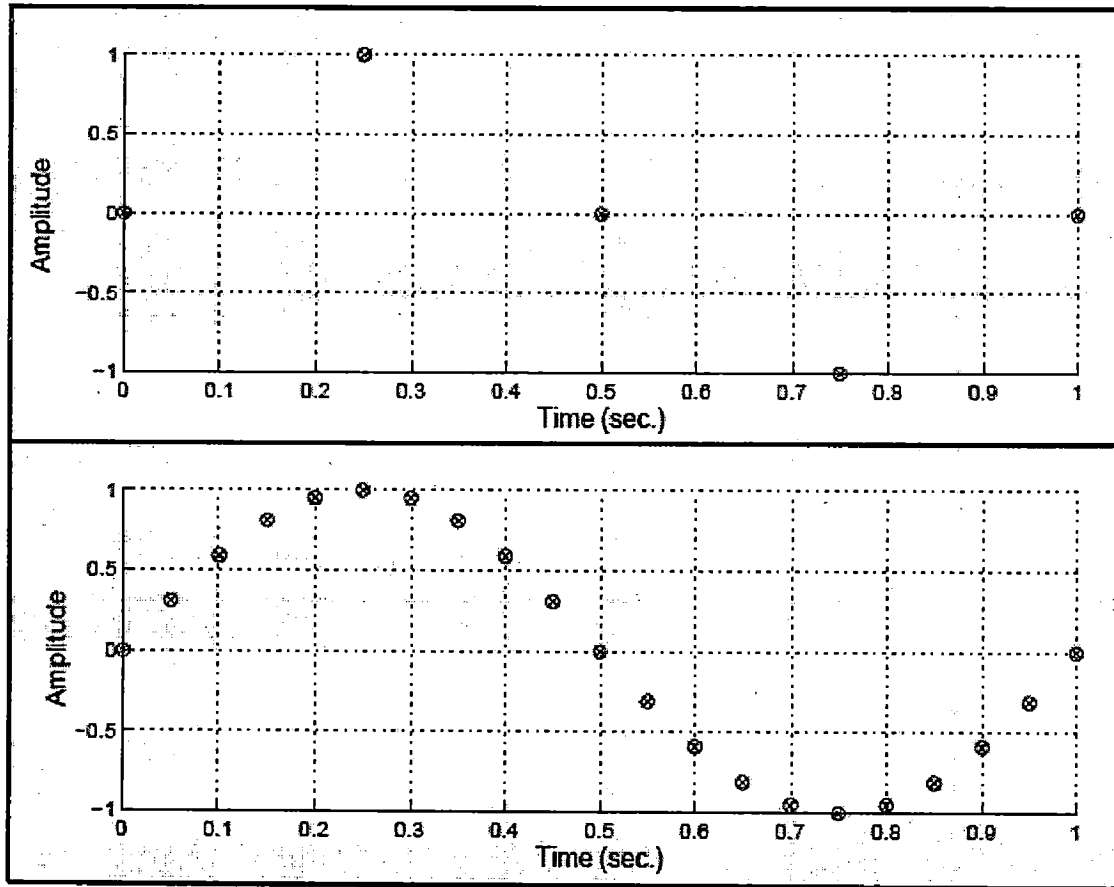


Figure 2.14-Effect of Low and High Sample Rate

As shown in the top figure, the low sampling rate produces a sampled signal that appears to be a triangular waveform. As shown in the bottom figure, a higher fidelity sampled signal is produced when the sampling rate is higher. In the latter case, the sampled signal actually looks like a sine wave.

2.3.5-How Can Aliasing be Eliminated?

The primary considerations involved in antialiasing are the sampling rate of the A/D converter and the frequencies present in the sampled data. To eliminate aliasing, following rules should be obeyed:

- Establish the useful bandwidth of the measurement.
- Select a sensor with sufficient bandwidth.
- Select a low-pass anti-aliasing analog filter that can eliminate all frequencies exceeding this bandwidth.
- Sample the data at a rate at least twice that of the filter's upper cutoff frequency.

DESCRIPTION OF HARDWARE

3.1- Parts of Hardware

After breaking the large circuits into smaller primitive building blocks large circuits can be understood by simple hand calculations, the bottle-necks in a circuit can easily be seen by circuit understanding. Thereby, circuit topology can be changed to eliminate one bottle-neck at a time, until the requirement specification is fulfilled.

There is, of course, a rather wide range of knowledge to learn, from detailed component properties up to overall circuit performance, before the circuit is design. Figure 3.1 shows the functional block diagram of the designed hardware.

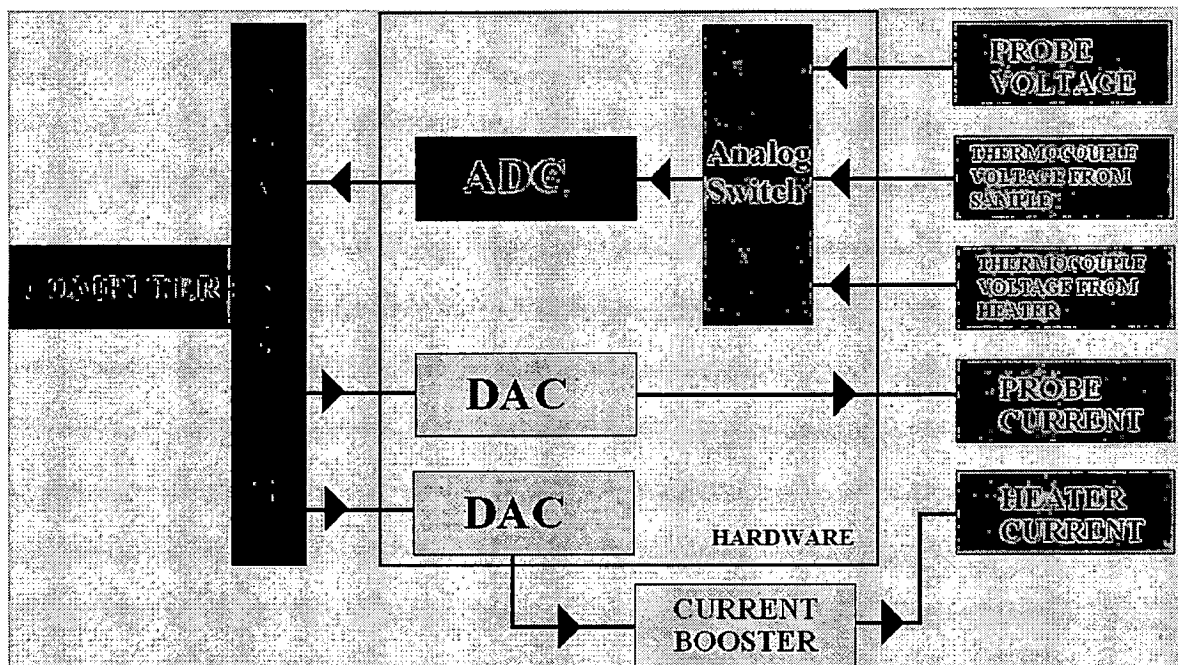


Figure 3.1-Functional Block Diagram

The hardware can be divided into following parts

- Circuitry for writing data (DAC)
- Circuitry for selecting channels
- Circuitry for reading data (ADC)
- Circuitry for generating clock

3.2-Circuitry for Writing Digital Data

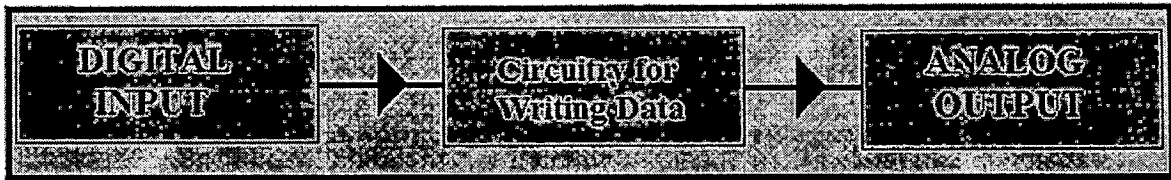


Figure 3.2-Digital Input Analog Output

This circuitry mainly consists of following components:

1. AD7521
2. LM741
3. LM336-2.5V

3.2.1-AD7521

The AD7521 is a low cost, monolithic 12-bit multiplying digital to analog converter packaged in an 18 pin DIP [1]. The devices use advanced CMOS and thin film technologies providing up to 10 bit accuracy.

The AD7521 operates from +5V to -5V supply and dissipates only 20 mW, including the ladder network.

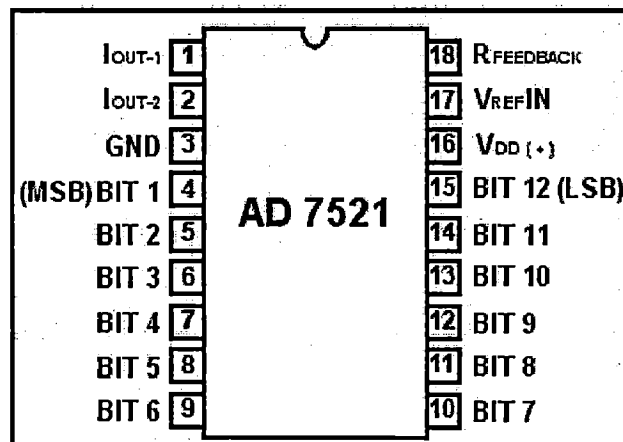


Figure 3.3-Pin diagram of AD 7521

The AD751 consists of a highly stable thin film R-2R ladder and twelve CMOS current switches on a monolithic chip. Most applications require the addition of only an output operational amplifier [11] and a voltage reference [12].

The simplified D/A circuit is shown in Figure 3.4 An inverted R-2R ladder structure is used, that is , the binary weighted currents are switched between the I_{OUT1} and I_{OUT2} bus lines, thus maintaining a constant current en each ladder leg independent of the switch state.

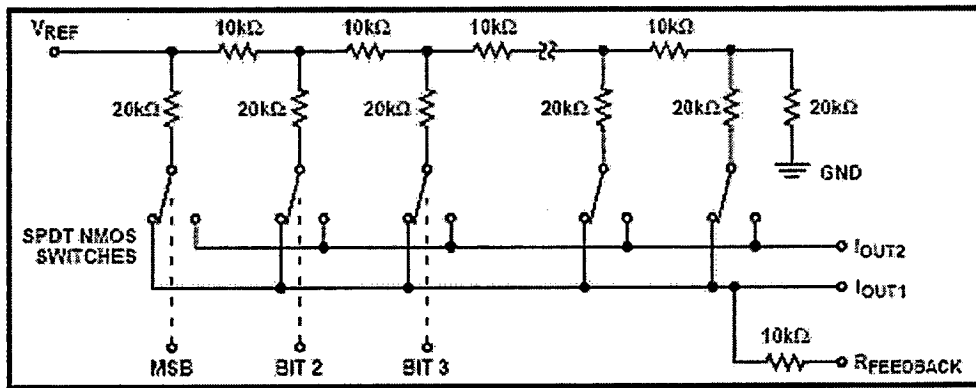


Figure 3.4-Functional Diagram of AD7521

The DAC AD7521 can be used in two ways: Unipolar Operation and Bipolar Operation. Unipolar operation means that current is unidirectional (either positive or negative) and Bipolar Operation means that current is bidirectional (both positive and negative).

- Unipolar Operation

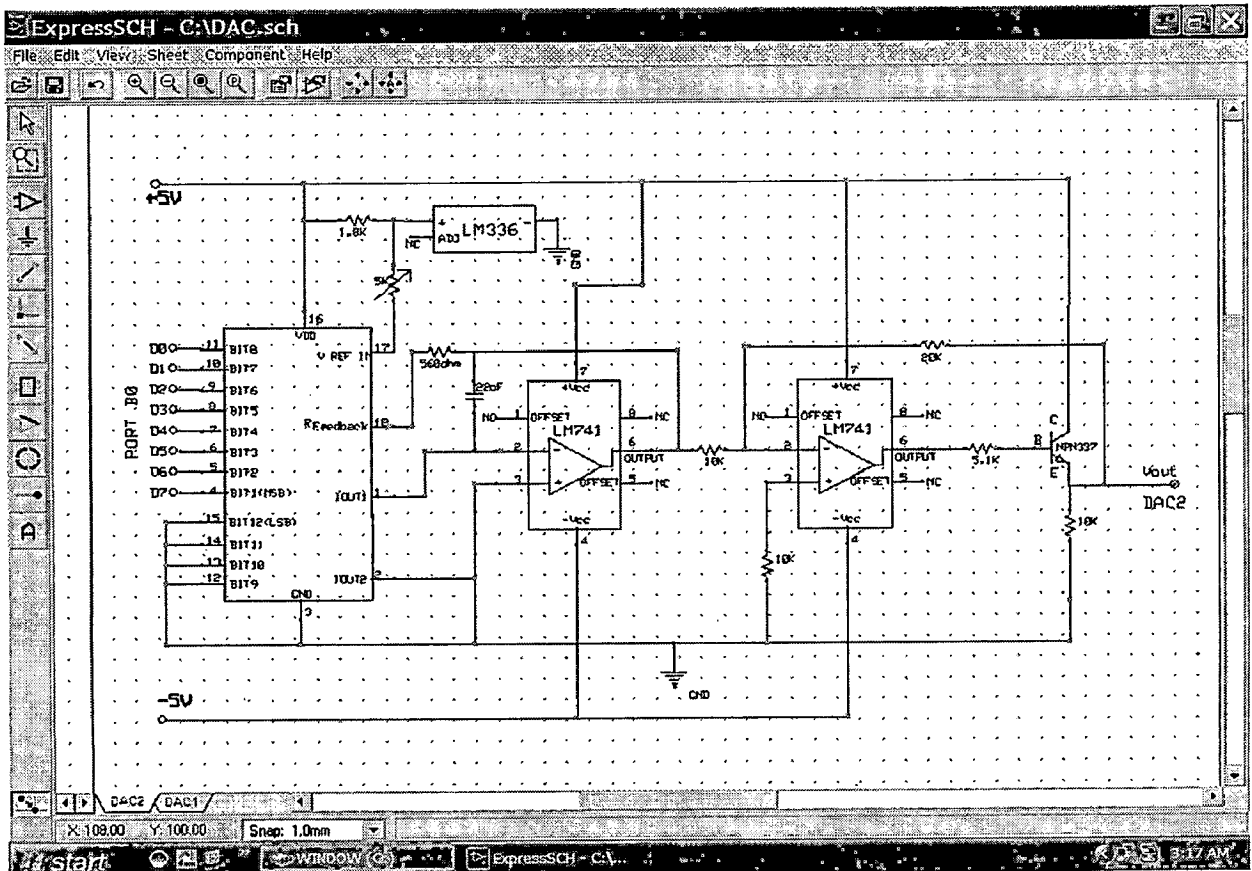


Figure 3.5-Circuit Diagram for Unipolar Binary Operation

Above figure shows the analog circuit connections required for unipolar binary operation. With a dc reference voltage (positive or negative polarity) applied at pin 17, the circuit is Unipolar D/A converter [7]. The input /output relationship is shown in Table 3.1

DIGITAL INPUT	ANALOG OUTPUT
1111111111	$V_{REF} (1-2^{-N})$
1000000001	$V_{REF} (1/2 + 2^{-N})$
1000000000	$V_{REF}/2$
0111111111	$V_{REF} (1/2-2^{-N})$
0000000001	$V_{REF} (2^{-N})$
0000000000	0

Table 3.1-Code Table-Unipolar Binary Operation

R1 at pin 17 provides full scale trim capability (i.e. load the DAC register to 11 1111 1111, adjust R1 for $V_{OUT} = V_{REF} (1 - 2^{-10})$. Alternatively, full scale can be adjusted by omitting R1 and R2 and trimming the reference voltage magnitude.

C1 Phase compensation (10 to 25 pF) may be required for stability when using high speed amplifiers.

Amplifier A1 should be selected or trimmed to provide $V_{OS} \leq 10\%$ of the voltage resolution at V_{OUT} . Additionally, the amplifier should exhibit a bias current which is low over the temperature range of interest (bias current causes output offset at V_{OUT} equal to I_B times the DAC feedback resistance, nominally 15 k Ω).

- **Bipolar Operation**

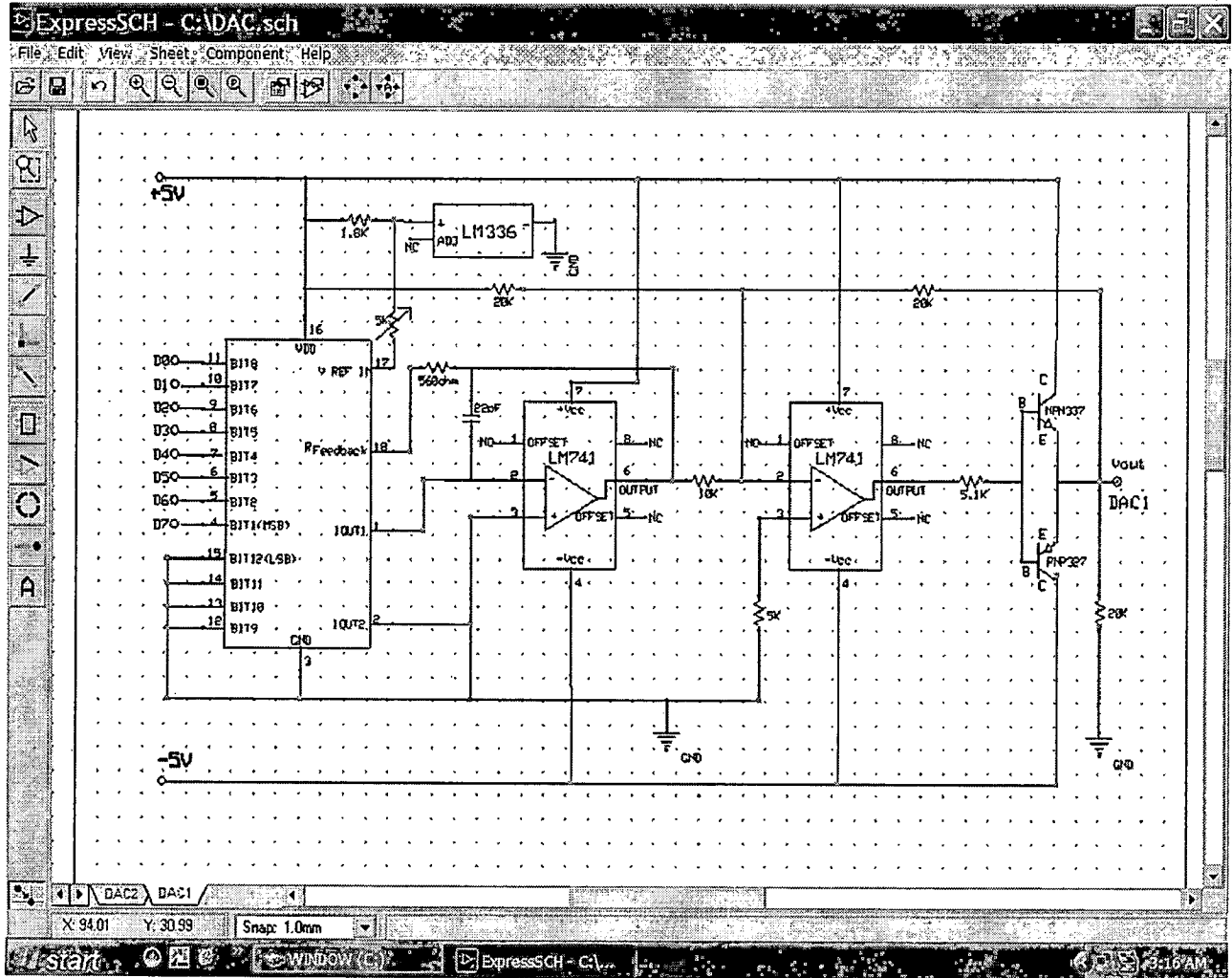


Figure 3.6-Circuit Diagram for Bipolar Operation

Figure 3.6 illustrate the circuitry for bipolar operation. With a dc reference (positive or negative polarity) the circuit provides offset binary operation [7]. The input /output relationship is shown in Table 3.2

DIGITAL INPUT	ANALOG OUTPUT
111111111	$-V_{REF} (1-2^{-(N-1)})$
100000001	$-V_{REF} (2^{-(N-1)})$
100000000	0
011111111	$V_{REF} (2^{-(N-1)})$
000000001	$V_{REF} (1-2^{-(N-1)})$
000000000	V_{REF}

Table 3.2-Code Table-Bipolar (Offset Binary) Operation

With the DAC register loaded to 10 0000 0000, adjust R1 for $V_{OUT}=0V$ (alternatively, one can omit R1 and R2 and adjust the ratio of R3 to R4 for $V_{OUT}=0V$). Full scale trimming can be accomplished by adjusting the amplitude of V_{REF} or by varying the value of R5.

As in Unipolar operation, A1 must be chosen for low V_{OS} and low I_B . R3,R4 and R5 must be selected for matching and tracking/ Mismatch of 2R3 to R4 causes both offset and Full Scale error. Mismatch of R5 to R4 or 2R3 causes full scale error. C1 phase compensation (10pF to 25pF) may be required for stability.

3.3-Circuitry for Selecting Channels

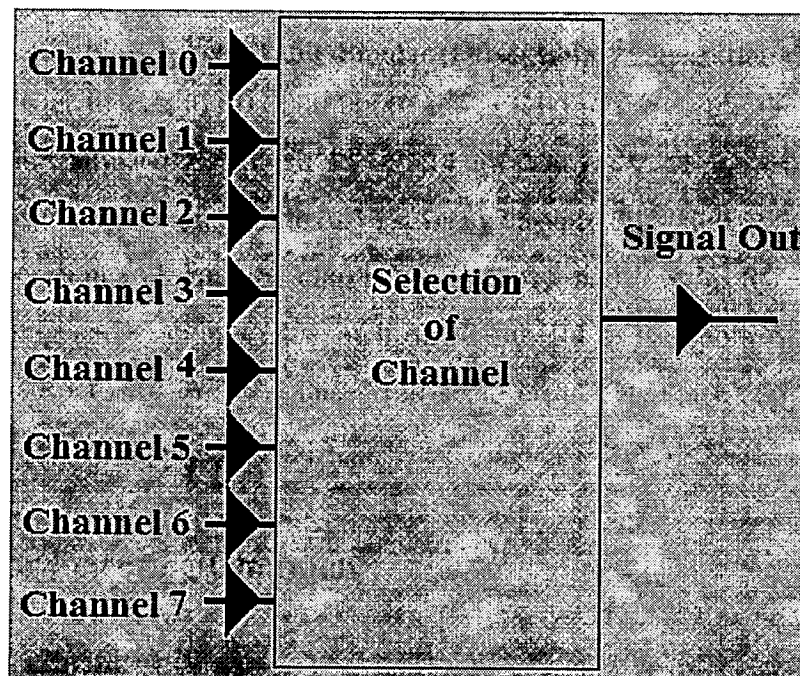


Figure 3.7-Selection of Channels

This circuitry mainly consists of following components:

1. LS154
2. CD4066
3. LS04

3.3.1-LS154

LS154 is a 4-line-to-16-line decoder; it utilizes TTL circuitry to decode four binary-coded inputs into one of sixteen mutually exclusive outputs when both the strobe inputs, G1 and G2, are low [5]. The demultiplexing function is performed by using the 4 input lines to address the output line, passing data from one of the strobe inputs with the other strobe input low. When either strobe input is high, all outputs are high. All inputs are buffered and input clamping diodes are provided to minimize transmission-line effects and thereby simplify system design.

By grounding Y8 to Y15 and D we have used it as a 3-line to 8-line decoder

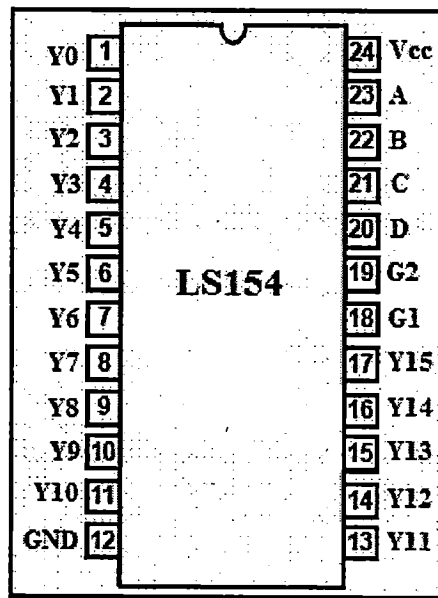


Figure 3.8-Pin diagram of LS154

Truth Table																					
Inputs					Outputs																
G1	G2	D	C	B	A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	L	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	H	L	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	L	H	L	L	L	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H
L	L	H	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H
L	L	H	L	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H
L	L	H	H	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H
L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H
L	L	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H
L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L
L	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	L	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H

H=High Level, L=Low Level, X=Don't Care

Table 3.3-Truth Table LS154

3.3.2-CD4066

The CD4066 is a quad bilateral switch intended for the transmission or multiplexing of analog or digital [3]. It has a much lower "ON" resistance, and "ON" resistance is relatively constant over the input-signal range.

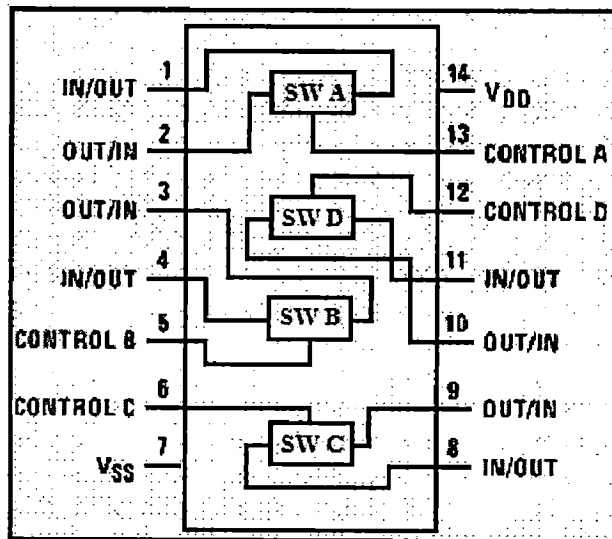


Figure 3.9-Pin diagram of CD4066

When signal at the control pin is high then input signal V_{IS} will be allowed to pass through and V_{OS} will appear at the OUT pin. Capacitor C_L is for filtering purpose, it ensures pure DC signal at the output.

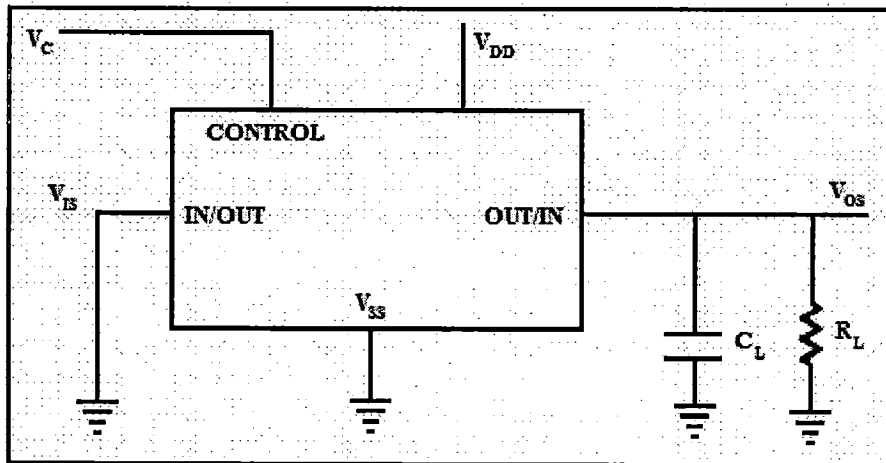


Figure 3.10-Working of CD4066

3.3.3-LS04

This device contains six independent gates each of which performs the logic INVERT function [6].

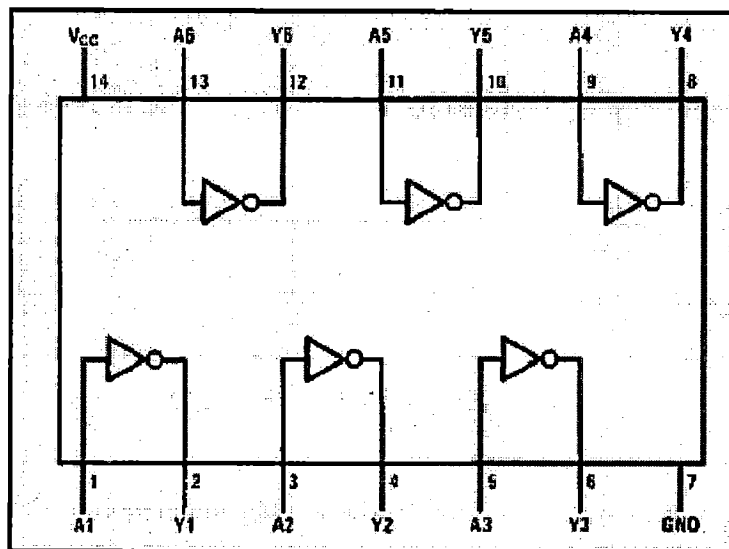


Figure 3.11-Pin Diagram of LS04

Input	Output	$Y = \bar{A}$
A	Y	
L	H	
H	L	

Table 3.4-Truth Table of LS04

3.3.4-Complete Circuit for Selecting Channels

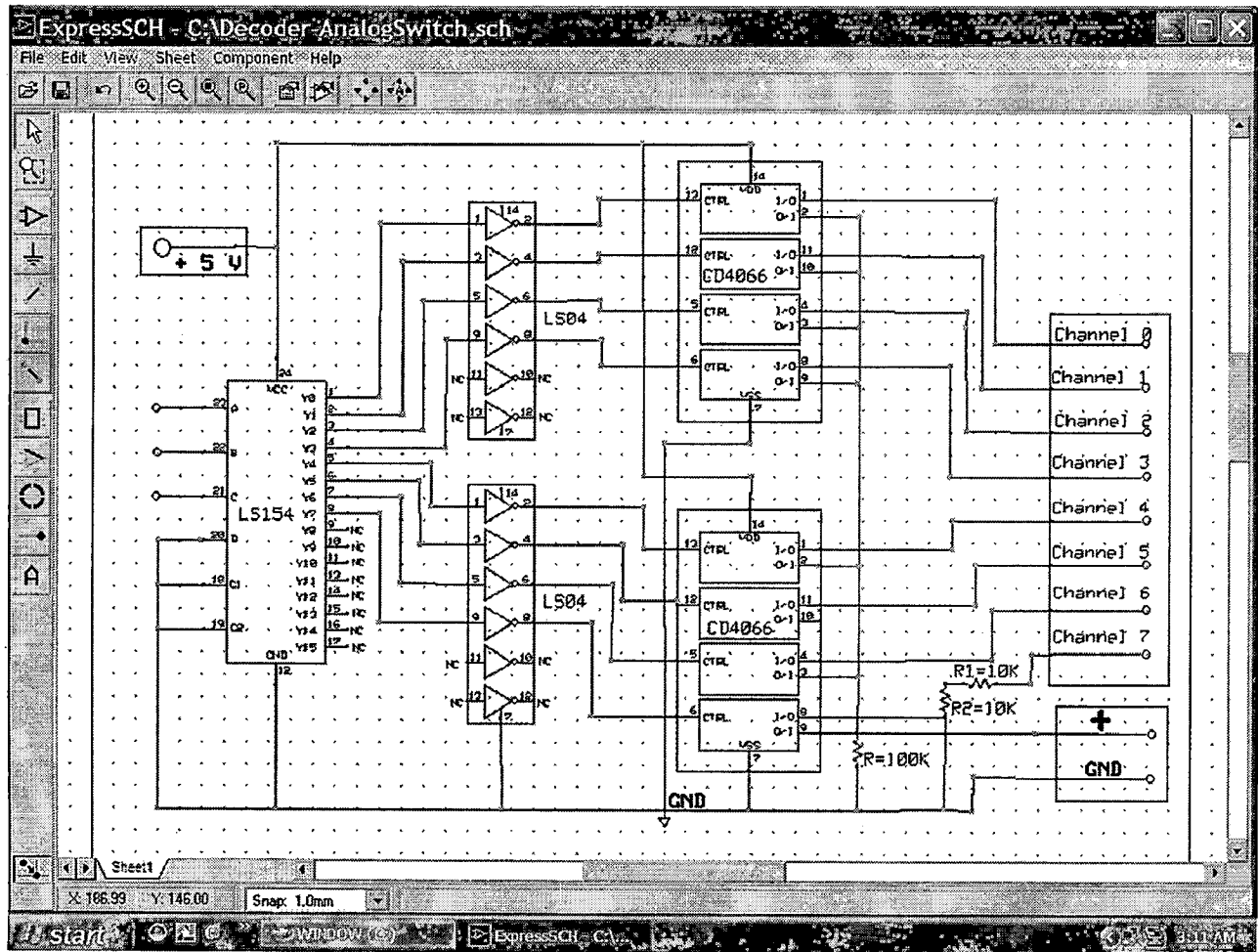


Figure 3.12-Circuit Diagram for Selecting Channels

The output pins (Y0 to Y7) of the decoder (LS154) is connected to the control pins of analog switch (CD4066) through inverter (LS04). Inverter is needed as decoder (LS154) has inverted logic (see Table 3.3).

All the output pins of the analog switch are shorted and grounded through resistor of 100 kohm. Each of the input pins of analog switch is connected to external pins and named as channel 0, channel 1,.....and so on.

The control pins A, B and C of the decoder (LS154) are used to select channels from 0 to 7. e.g. in order to select channel 0, the control signal 000 is given to A, B and C as a result only output Y0 of the decoder becomes low (all other output pins remains high), as all the output pins (Y0 to Y7) are connected to the analog switch through inverter thus the control pin of analog switch which is connected to Y0 becomes high. And the signal (at channel 0) which is associated with this control pin will be allowed to appear at the external output pin

3.4-Circuitry for Reading Digital Data

This circuitry mainly consists of following components:

1. ICL7135
2. CD4520

3.4.1-ICL7135

The ICL7135 precision A/D converter, with its multiplexed BCD output and digit drivers, combines dual slope conversion reliability with ± 1 in 20,000 count accuracy and is ideally suited for the visual display DVM/DPM market [8].

The 2.0000V full scale capability, auto-zero, and autopolarity are combined with true ratiometric operation, almost ideal differential linearity and true differential input. All necessary active devices are contained on a single CMOS IC, with the exception of display drivers, reference, and a clock.

The ICL7135 brings together an unprecedented combination of high accuracy, versatility, and true economy. It features auto-zero to less than $10\mu\text{V}$, zero drift of less than $1\mu\text{V}/^\circ\text{C}$, input bias current of 10pA (Max), and rollover error of less than one count. The versatility of multiplexed BCD outputs is increased by the addition of several pins which allow it to operate in more sophisticated systems. These include STROBE, OVERRANGE, UNDERRANGE, RUN/HOLD and BUSY lines, making it possible to interface the circuit to a microprocessor or UART.

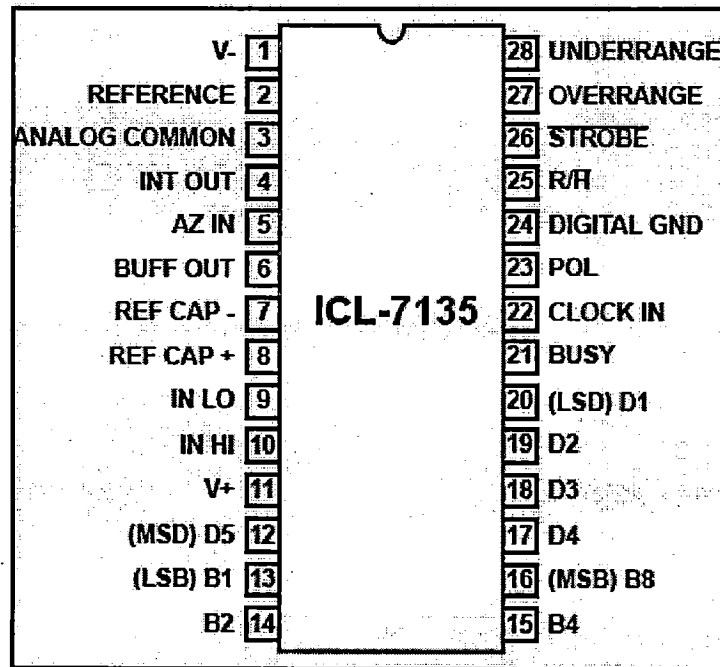


Figure 3.13-Pin Diagram of ICL7135

• Analog Section

Figure 3.14 shows the Block Diagram of the Analog Section for the ICL7135. Each measurement cycle is divided into four phases. They are

- (1) Auto-zero (AZ)
- (2) Signal-integrate (INT)
- (3) De-integrate (DE)
- (4) Zero-integrator (ZI)

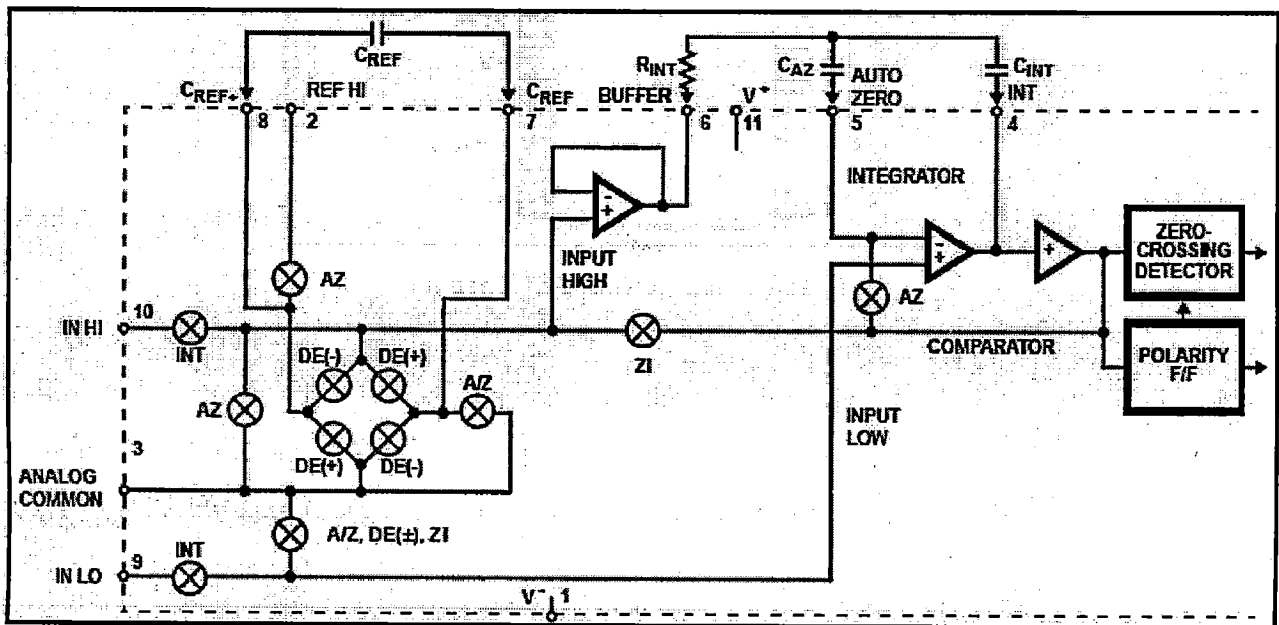


Figure 3.14-Block Diagram of Analog Section of ICL7135

Auto-Zero Phase

During auto-zero, three things happen. First, input high and low are disconnected from the pins and internally shorted to analog COMMON. Second, the reference capacitor is charged to the reference voltage. Third, a feedback loop is closed around the system to charge the auto-zero capacitor C_{AZ} to compensate for offset voltages in the buffer amplifier, integrator, and comparator. Since the comparator is included in the loop, the AZ accuracy is limited only by the noise of the system. In any case, the offset referred to the input is less than $10\mu\text{V}$.

Signal Integrate Phase

During signal integrate, the auto-zero loop is opened, the internal short is removed, and the internal input high and low are connected to the external pins. The converter then integrates the differential voltage between IN HI and IN LO for a fixed time. This differential voltage can be within a wide common mode range, within one volt of either supply. If, on the other hand, the input signal has no return with respect to the converter power supply, IN LO can be tied to analog COMMON to establish the correct common-mode voltage. At the end of this phase, the polarity of the integrated signal is latched into the polarity F/F.

De-Integrate Phase

The third phase is de-integrate or reference integrate. Input low is internally connected to analog COMMON and input high is connected across the previously charged reference capacitor. Circuitry within the chip ensures that the capacitor will be connected with the correct polarity to cause the integrator output to return to zero. The time required for the output to return to zero is proportional to the input signal. Specifically the digital reading displayed is:

$$\text{OUTPUT COUNT} = 10000 \left(\frac{V_{in}}{V_{REF}} \right)$$

Zero Integrator Phase

The final phase is zero integrator. First, input low is shorted to analog COMMON. Second, a feedback loop is closed around the system to input high to cause the integrator output to return to zero. Under normal condition, this phase lasts from 100 to 200 clock pulses, but after an overrange conversion, it is extended to 6200 clock pulses.

Differential Input

The input can accept differential voltages anywhere within the common mode range of the input amplifier, or specifically from 0.5V below the positive supply to 1V above the negative supply.

In this range the system has a CMRR of 86dB typical. However, since the integrator also swings with the common mode voltage, care must be exercised to assure the integrator output does not saturate. A worst case condition would be a large positive common-mode voltage with a near full scale negative differential input voltage. The negative input signal drives the integrator positive when most of its swing has been used up by the positive common mode voltage. For these critical applications the integrator swing can be reduced to less than the recommended 4V full scale swing with some loss of accuracy. The integrator output can swing within 0.3V of either supply without loss of linearity.

Analog COMMON

Analog COMMON is used as the input low return during autozero and de-integrate. If IN LO is different from analog COMMON, a common mode voltage exists in the system and is taken care of by the excellent CMRR of the converter. However, in most applications IN LO will be set at a fixed known voltage (power supply common for instance). In this application, analog COMMON should be tied to the same point, thus removing the common mode voltage from the converter. The reference voltage is referenced to analog COMMON.

Reference

The reference input must be generated as a positive voltage with respect to COMMON externally.

- **Digital Section**

The ICL7135 includes several pins which allow it to operate conveniently in more sophisticated systems. These include:

Run/ $\overline{\text{HOLD}}$ (Pin 25)

When high (or open) the A/D will free-run with equally spaced measurement cycles every 40,002 clock pulses. If taken low, the converter will continue the full measurement cycle that it is doing and then hold this reading as long as R/H is held low. A short positive pulse (greater than 300ns) will now initiate a new measurement cycle, beginning with between 1 and 10,001 counts of auto zero. If the pulse occurs before the full measurement cycle (40,002 counts) is completed, it will not be recognized and the converter will simply complete the measurement it is doing. An external indication that a full measurement cycle has been completed is that the first strobe pulse will occur 101 counts after the end of this cycle. Thus, if Run/HOLD is low and has

been low for at least 101 counts, the converter is holding and ready to start a new measurement when pulsed high.

STROBE (Pin 26)

This is a negative going output pulse that aids in transferring the BCD data to external latches, UARTs, or microprocessors. There are 5 negative going STROBE pulses that occur in the center of each of the digit drive pulses and occur once and only once for each measurement cycle starting 101 clock pulses after the end of the full measurement cycle. Digit 5 (MSD) goes high at the end of the measurement cycle and stays on for 201 counts. In the center of this digit pulse (to avoid race conditions between changing BCD and digit drives) the first STROBE pulse goes negative for 1/2 clock pulse width. Similarly, after digit 5, digit 4 goes high (for 200 clock pulses) and 100 pulses later the STROBE goes negative for the second time. This continues through digit 1 (LSD) when the fifth and last STROBE pulse is sent. The digit drive will continue to scan (unless the previous signal was overrange) but no additional STROBE pulses will be sent until a new measurement is available.

BUSY (Pin 21)

BUSY goes high at the beginning of signal integrate and stays high until the first clock pulse after zero crossing (or after end of measurement in the case of an overrange). The internal latches are enabled (i.e., loaded) during the first clock pulse after busy and counter are latched at the end of this clock pulse. The circuit automatically reverts to auto-zero when not BUSY, so it may also be considered a $\overline{(ZI + AZ)}$ signal. A very simple means for transmitting the data down a single wire pair from a remote location would be to AND BUSY with clock and subtract 10,001 counts from the number of pulses received - as mentioned previously there is one "NO-count" pulse in each reference integrate cycle.

OVERRANGE (Pin 27)

This pin goes positive when the input signal exceeds the range (20,000) of the converter. The output F/F is set at the end of BUSY and is reset to zero at the beginning of reference integrate in the next measurement cycle.

UNDERRANGE (Pin 28)

This pin goes positive when the reading is 9% of range or less. The output F/F is set at the end of BUSY (if the new reading is 1800 or less) and is reset at the beginning of signal integrate of the next reading.

POLARITY (Pin 23)

This pin is positive for a positive input signal. It is valid even for a zero reading. In other words, +0000 means the signal is positive but less than the least significant bit. The converter can be used as a null detector by forcing equal frequency of (+) and (-) readings. The null at this point should be less than 0.1 LSB. This output becomes valid at the beginning of reference integrate and remains correct until it is revalidated for the next measurement.

• Component Value Selection

For optimum performance of the analog section, care must be taken in the selection of values for the integrator capacitor and resistor, auto-zero capacitor, reference voltage, and conversion rate. These values must be chosen to suit the particular application.

Integrating Resistor

The integrating resistor is determined by the full scale input voltage and the output current of the buffer used to charge the integrator capacitor. Both the buffer amplifier and the integrator have a class A output stage with 100 μ A of quiescent current. They can supply 20 μ A of drive current with negligible non-linearity. Values of 5 μ A to 40 μ A give good results, with a nominal of 20 μ A, and the exact value of integrating resistor may be chosen by:

$$R_{INT} = \text{full scale voltage} / 20\mu\text{A}$$

Integrating Capacitor

The product of integrating resistor and capacitor should be selected to give the maximum voltage swing which ensures that the tolerance built-up will not saturate the integrator swing (approx. 0.3V from either supply). For ± 5 V supplies and analog COMMON tied to supply ground, a ± 3.5 V to ± 4 V full scale integrator swing is fine, and 0.47 μ F is nominal. In general, the value of C_{INT} is given by:

$$\begin{aligned} C_{INT} &= [10,000 * \text{clock period}] * I_{INT} / \text{integrator output voltage swing} \\ &= (10,000) (\text{clock period}) (20\mu\text{A}) / \text{integrator output voltage swing} \end{aligned}$$

A very important characteristic of the integrating capacitor is that it has low dielectric absorption to prevent roll-over or ratiometric errors. A good test for dielectric absorption is to use the capacitor with the input tied to the reference.

This ratiometric condition should read half scale 0.9999, and any deviation is probably due to dielectric absorption. Polypropylene capacitors give undetectable errors at reasonable cost. Polystyrene and polycarbonate capacitors may also be used in less critical applications.

Auto-Zero and Reference Capacitor

The physical size of the auto-zero capacitor has an influence on the noise of the system. A larger capacitor value reduces system noise. A larger physical size increases system noise. The reference capacitor should be large enough such that stray capacitance to ground from its nodes is negligible.

The dielectric absorption of the reference capacitor and auto-zero capacitor are only important at power-on or when the circuit is recovering from an overload. Thus, smaller or cheaper capacitors can be used here if accurate readings are not required for the first few seconds of recovery.

Reference Voltage

The analog input required to generate a full scale output is

$$V_{IN} = 2V_{REF}$$

The stability of the reference voltage is a major factor in the overall absolute accuracy of the converter [12]. For this reason, it is recommended that a high quality reference be used where high-accuracy absolute measurements are being made.

Rollover Resistor and Diode

A small rollover error occurs in the ICL7135, but this can be easily corrected by adding a diode and resistor in series between the INTegrator OUTPUT and analog COMMON or ground. The value shown in the schematics is optimum for the recommended conditions, but if integrator swing or clock frequency is modified, adjustment may be needed. The diode can be any silicon diode such as 1N914. These components can be eliminated if rollover error is not important and may be altered in value to correct other (small) sources of rollover as needed.

Max Clock Frequency

The maximum conversion rate of most dual-slope A/D converters is limited by the frequency response of the comparator. The comparator in this circuit follows the integrator ramp with a 3 μ s delay, and at a clock frequency of 160 kHz (6 μ s period) half of the first reference integrate clock period is lost in delay. This means that the meter reading will change from 0 to 1 with a 50 μ V input, 1 to 2 with a 150 μ V input, 2 to 3 with a 250 μ V input, etc. This transition at midpoint is

considered desirable by most users; however, if the clock frequency is increased appreciably above 160 kHz, the instrument will flash "1" on noise peaks even when the input is shorted. For many dedicated applications where the input signal is always of one polarity, the delay of the comparator need not be a limitation. Since the non-linearity and noise do not increase substantially with frequency, clock rates of up to ~1MHz may be used. For a fixed clock frequency, the extra count or counts caused by comparator delay will be constant and can be subtracted out digitally.

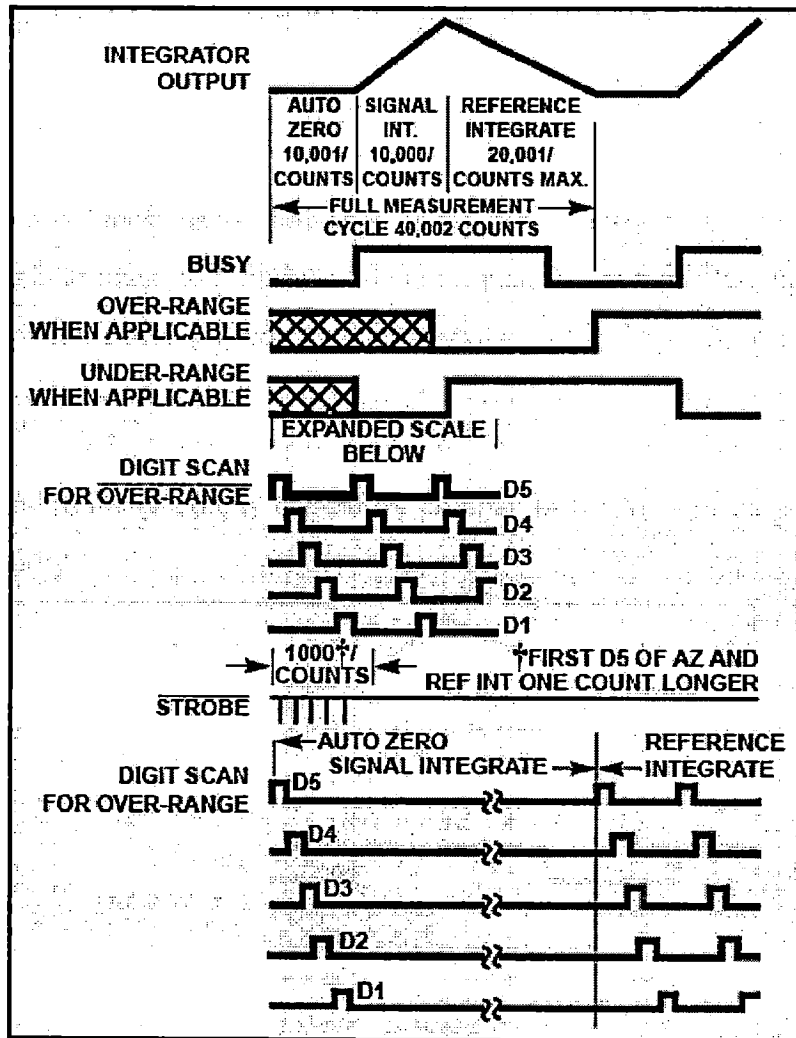
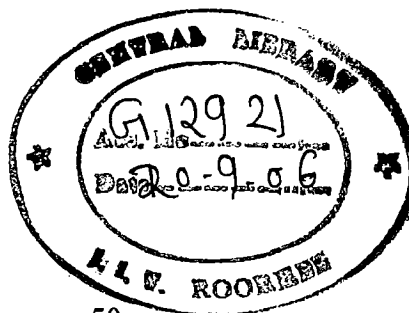


Figure 3.15-Timing Diagram for Outputs



Zero-Crossing Flip-Flop

The flip-flop interrogates the data once every clock pulse after the transients of the previous clock pulse and half-clock pulse have died down. False zero-crossings caused by clock pulses are not recognized. Of course, the flip-flop delays the true zero-crossing by up to one count in every instance, and if a correction were not made, the display would always be one count too high. Therefore, the counter is disabled for one clock pulse at the beginning of phase 3. This one-count delay compensates for the delay of the zero-crossing flip-flop, and allows the correct number to be latched into the display. Similarly, a one-count delay at the beginning of phase 1 gives an overload display of 0000 instead of 0001. No delay occurs during phase 2, so that true ratiometric readings result.

- **Evaluating the Error Sources**

1. Errors from the “ideal” cycle are caused by:
2. Capacitor droop due to leakage.
3. Capacitor voltage change due to charge “suck-out” (the reverse of charge injection) when the switches turn off.
4. Non-linearity of buffer and integrator.
5. High-frequency limitations of buffer, integrator, and comparator.
6. Integrating capacitor non-linearity (dielectric absorption).
7. Charge lost by C_{REF} in charging C_{STRAY} .
8. Charge lost by C_{AZ} and C_{INT} to charge C_{STRAY} .

- **Noise**

The peak-to-peak noise around zero is approximately $15\mu\text{V}$ (peak-to-peak value not exceeded 95% of the time). Near full scale, this value increases to approximately $30\mu\text{V}$. Much of the noise originates in the auto-zero loop, and is proportional to the ratio of the input signal to the reference.

- **Analog And Digital Grounds**

Extreme care must be taken to avoid ground loops in the layout of ICL7135 circuits, especially in high-sensitivity circuits. It is most important that return currents from digital loads are not fed into the analog ground line.

3.4.2-CD4520

The CD4520 is a dual binary counter, it consists of two identical, independent, synchronous, 4-stage counters [4]. The counter stages are toggle flip-flops which increment on either the positive edge of CLOCK or negative edge of ENABLE, simplifying cascading of multiple stages. The counter can be asynchronously cleared by a high level on the RESET line. All inputs are protected against static discharge by diode clamps to both V_{DD} and V_{SS} .

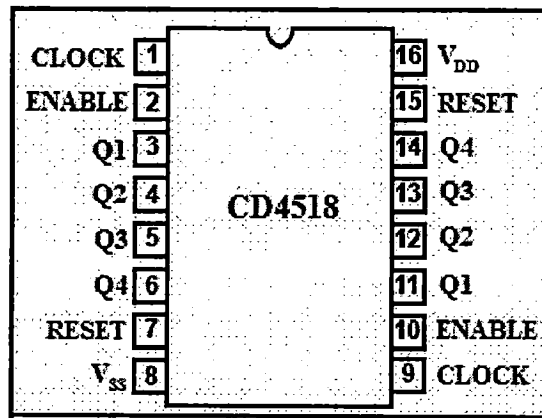


Figure 3.16-Pin Diagram of CD4520

3.4.3-Complete Circuit for Reading Data

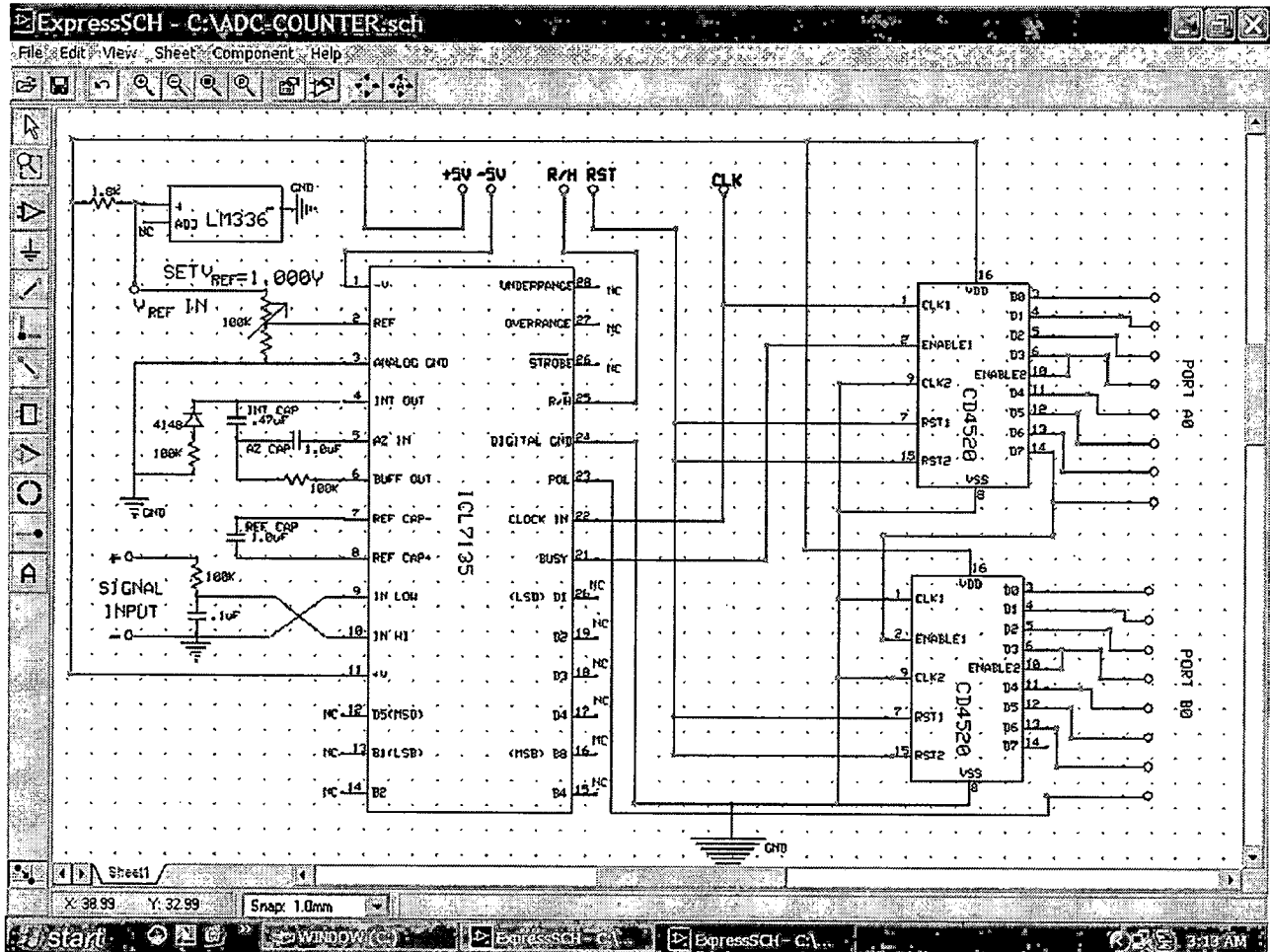


Figure 3.17-Circuit Diagram for Reading Data

The analog signal (less than 2V) is given between IN HI and IN LO pin of ICL7135 keeping R/H pin high, The BUSY goes high at the beginning of signal integrate and stays high until the first clock pulse after zero crossing. The internal latches are enabled (i.e., loaded) during the first clock pulse after busy is latched at the end of this clock pulse. The converter then integrates the differential voltage between IN HI and IN LO for a fixed time. In response to the BUSY signal the counter CD4520 produces corresponding binary number. The output pins of CD 4520 are connected to port A0 and B0 which has been configured as Input Port. Thus by using proper data acquisition commands these numbers can be read.

The R/H and RST pins are connected to the control port from where the readings can be controlled according to need. It is to be noted here that the Polarity of ICL7135 is connected as MSB; polarity is required to read negative number. The CLOCK pulse to ICL7135 and CD4520 is provided externally by separate timer circuit.

3.5-Circuitry for Generating Clock

The circuitry for generating clock pulse consists of 555 timer.

3.5.1-555 Timer

The 555 is a highly stable controller capable of producing accurate time delays, or oscillation [15]. In the time delay mode of operation, the time is precisely controlled by one external resistor and capacitor. For a stable operation as an oscillator, the free running frequency and the duty cycle are both accurately controlled with two external resistors and one capacitor. The circuit may be triggered and reset on falling waveforms, and the output structure can source or sink up to 200mA.

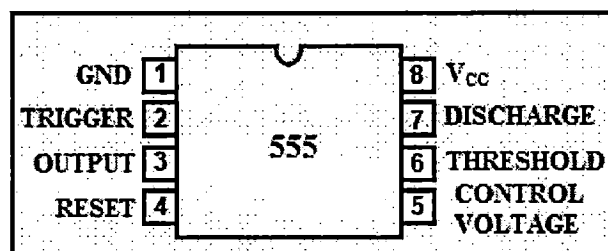


Figure 3.18-Pin Diagram of 555

3.5.2-Complete Circuit

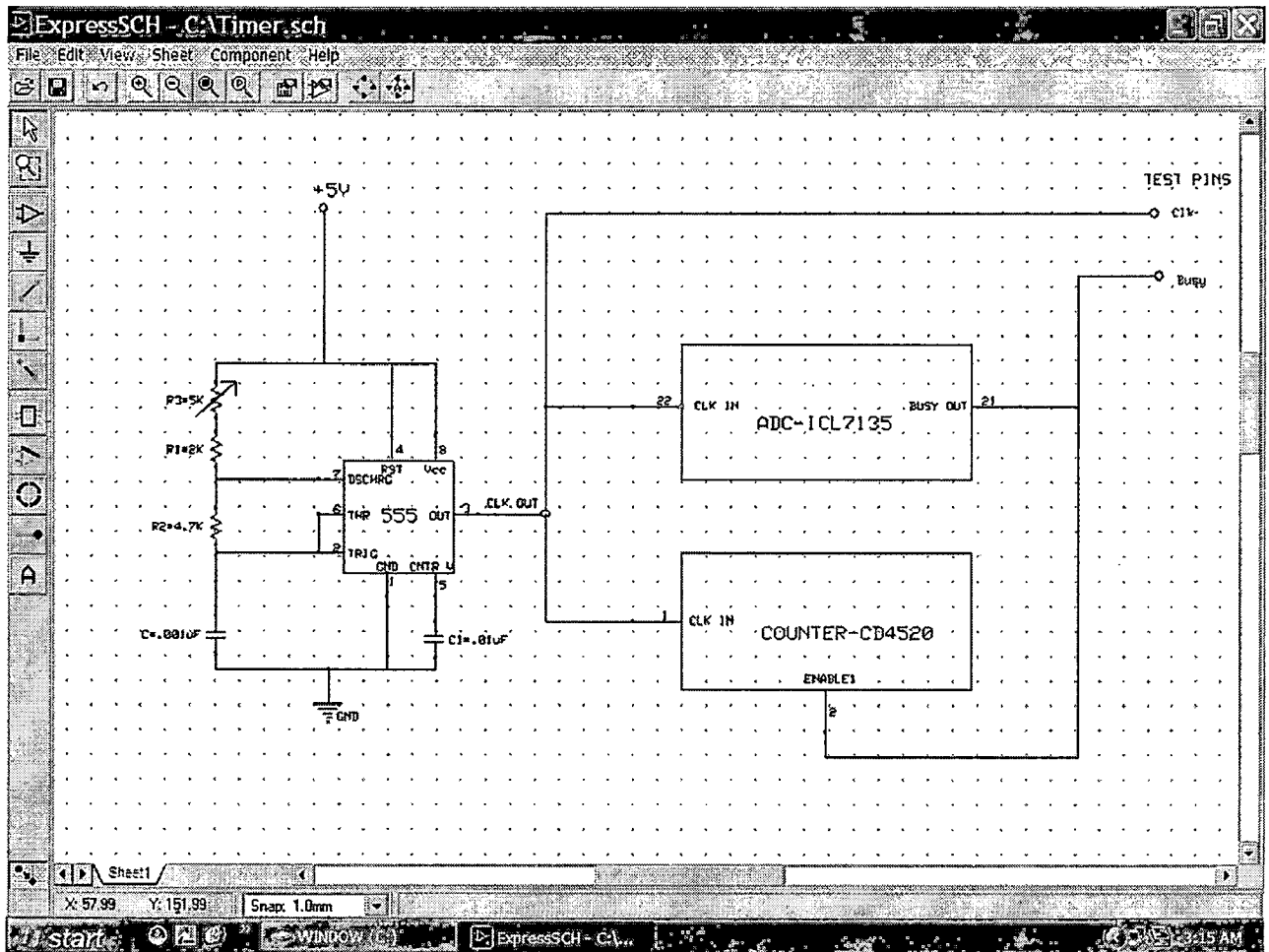


Figure 3.19-Circuitry for Generating Clock

SOFTWARE FOR ACQUIRING DATA

4.1-Introduction to MATLAB

MATLAB is a software package for high-performance numerical computation and visualization [2]. It provides an interactive environment with hundreds of built in functions for technical computation, graphics, and animation. Best of all, it also provides easy extensibility with its own high-level programming language. Typical uses include:

- Math and computation
- Algorithm development
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including graphical user interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning [13]. This allows us to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar non-interactive language such as C or FORTRAN.

The name MATLAB stands for **MATRIX LABORATORY**. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects. Today, MATLAB uses software developed by the LAPACK and ARPACK projects, which together represent the state-of-the-art in software for matrix computation.

MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis.

Toolboxes

MATLAB features a family of application-specific solutions called toolboxes [16]. Very important to most users of MATLAB, toolboxes allow learning and applying specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that

extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

The MATLAB System

The MATLAB system consists of five main parts:

1. Development Environment: This is the set of tools and facilities that help users use MATLAB functions and files. Many of these tools are graphical user interfaces. It includes the MATLAB desktop and Command Window, a command history, and browsers for viewing help, the workspace, files, and the search path.

2. The MATLAB Mathematical Function Library: This is a vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigen-values, Bessel functions, and fast Fourier transforms.

3. The MATLAB language: This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both “programming in the small” to rapidly create quick and dirty throw-away programs, and programming in the large to create complete large and complex application programs.

4. Handle Graphics: This is the MATLAB graphics system. It includes high-level commands for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level commands that allow fully customization of the appearance of graphics as well as building of complete graphical user interfaces on MATLAB applications.

5. The MATLAB Application Program Interface (API):

This is a library that allows user to write C and FORTRAN programs that interact with MATLAB. It include facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

4.2-Data Acquisition Toolbox

The Data Acquisition Toolbox is a collection of M-file functions and MEX-file dynamic link libraries (DLLs) built on the MATLAB technical computing environment [16]. The main features provided by toolbox are as follow:

- A framework for bringing live, measured data into MATLAB using PC-compatible, plug-in data acquisition hardware
- Support for these popular hardware devices:
 - ✓ National Instruments E Series boards, low cost devices such as 1200/Lab/LPM/700/500/516 boards, 445x dynamic signal acquisition (DSA) Series boards, analog output boards, and digital I/O boards (no streaming)
 - ✓ ComputerBoards (Measurement Computing Corporation) boards
 - ✓ Agilent Technologies E1432A/33A/34A VXI modules
 - ✓ Advantech automation PCI 1751 48 bits digital I/O card

Additionally, the Data Acquisition Toolbox Adaptor Kit can be used to interface unsupported hardware devices to the toolbox.

- Support for analog input (AI), analog output (AO), and digital I/O (DIO) subsystems including simultaneous AI and AO conversions
- Data buffering for background acquisitions and data logging
- Event-driven acquisitions

Exploring the Toolbox

A list of the toolbox functions available can be displayed by typing

```
>>help daq
```

The code for any function can be viewed by typing

```
>>type <function_name>
```

The help for any function can be viewed by typing

```
>>daqhelp <function_name>
```

The way any toolbox function works can be changed by copying and renaming the M-file, then modifying the copy. The toolbox can also be extended by adding new M-files, or by using it in combination with other products such as the Signal Processing Toolbox or the Instrument Control Toolbox.

4.3-Digital Input/Output Object

Digital I/O (DIO) subsystems are designed to input and output digital values to and from hardware. These values are typically handled either as single bits or *lines*, or as a *port*, which typically consists of eight lines. While most popular data acquisition boards include some digital I/O capability, it is usually limited to simple operations and special dedicated hardware is required for performing advanced digital I/O operations. The Data Acquisition Toolbox provides access to digital I/O subsystems through a digital I/O object.

Note The Data Acquisition Toolbox does not directly support buffered digital I/O or handshaking (latching). However, new M-code can be written to support this functionality. Buffered digital I/O means that the data associated with a digital I/O subsystem is stored in the engine. Handshaking allows digital I/O subsystem to input or output data after receiving a digital pulse.

4.3.1-Creating a Digital I/O Object

A digital I/O object can be created with the **digitalio** function. **digitalio** accepts the adaptor name and the hardware device ID as input arguments. The device ID refers to the number associated with board in use, when it is installed. Some vendors refer to the device ID as the device number or the board number. **daqhwinfo** function is used to determine the available adaptors and device IDs.

Each digital I/O object is associated with one board and one digital I/O subsystem. For example, to create a digital I/O object associated with an Advantech board with device ID 0

```
>>dio = digitalio ('advantech', 0);
```

The digital I/O object **dio** now exists in the MATLAB workspace. The class of **dio** can be displayed with the **whos** command.

```
>>whos dio
```

Name	Size	Bytes	Class
dio	1x1	1308	digitalio object

Grand total is 40 elements using 1308 bytes

Once the digital I/O object is created, the properties listed below are automatically assigned values. These general purpose properties provide descriptive information about the object based on its class type and adaptor.

Property Name	Description
Name	Specify a descriptive name for the device object.
Type	Indicate the device object type.

Table 4.1-Descriptive Digital I/O Properties

The values of these properties for dio can be displayed with the `get` function.

```
>>get (dio,{'Name','Type'})
```

ans =

```
'Adv-DIO'      'Digital IO'
```

4.3.2-Adding Lines to a Digital I/O Object

Line and Channels are the basic hardware device elements with which the data is acquired or outputted. After the creation of device object, channels or lines must be added to it. Channels are added to analog input and analog output objects, while lines are added to digital I/O objects.

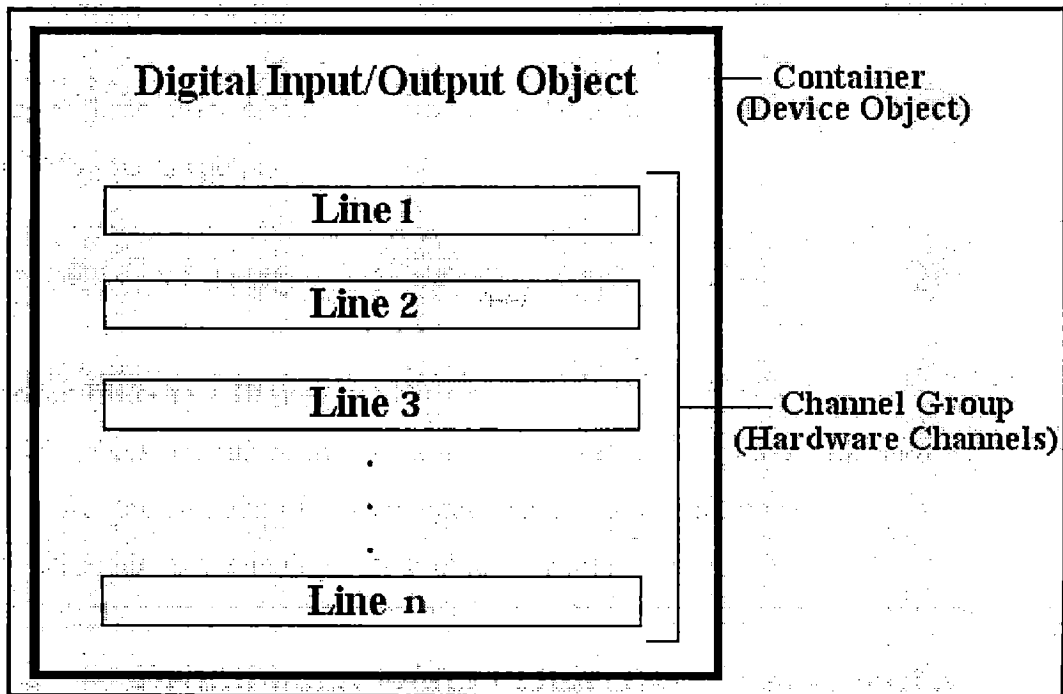


Figure 4.1-Device Object as a Container

As shown in Figure 4.1, a device object can be thought as a container for lines. The collection of lines contained by the device object is referred to as a *line group*. A line group consists of a mapping between hardware line IDs and MATLAB indices.

When adding lines to a digital I/O object, these rules must be followed:

- The lines must reside on the same hardware device. Lines can not be added from different devices, or from different subsystems on the same device.
- A line can be added only once to a given digital I/O object. However, a line can be added to as many different digital I/O objects as is desired.
- Lines can be added that reside on different ports to a given digital I/O object.

Lines are added to a digital I/O object with the **addline** function. Addline requires the device object, at least one hardware line ID, and the direction (input or output) of each added line as input arguments. The port IDs, descriptive line names, and an output argument can be specified optionally. For example, to add eight output lines from port 0 to the device object `dio` created in the preceding section

```
>>hwlines = addline (dio, 0:7, 'out');
```

The output argument `hwlines` is a *line object* that reflects the line array contained by `dio`. The class of `hwlines` can be displayed with the `whos` command.

```
>>whos hwlines
```

Name	Size	Bytes	Class
Hwlines	8x1	536	dioline object

Grand total is 13 elements using 536 bytes

The `hwlines` can be used to access lines easily. For example, one or more lines can be configured or their property values can be returned. As described in XXX, the lines can also be accessed with the `Line` property.

Once the lines are added to a digital I/O object, the properties listed below are automatically assigned values. These properties provide descriptive information about the lines based on their class type and ID.

Property Name	Description
HwLine	Specify the hardware line ID.
Index	Indicate the MATLAB index of a hardware line.
Parent	Indicate the parent (device object) of a line.
Type	Indicate a line.

Table 4.2-Descriptive Digital I/O Line Properties

The values of these properties for channels can be displayed with the get function.

```
>> get (hwlines, {'HwLine','Index','Parent','Type'})
```

```
ans =
```

```

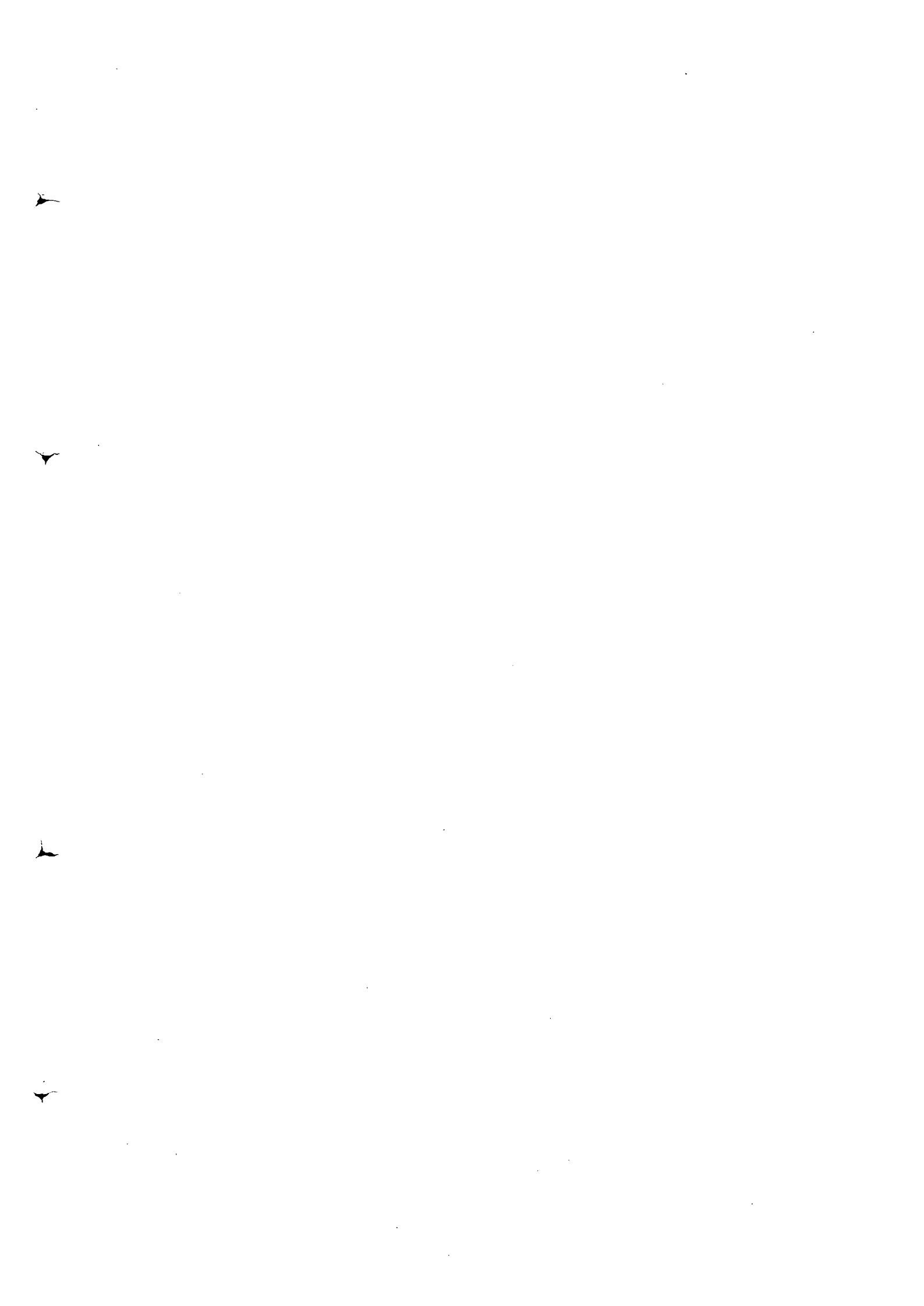
[0] [1] [1x1 digitalio] 'Line'
[1] [2] [1x1 digitalio] 'Line'
[2] [3] [1x1 digitalio] 'Line'
[3] [4] [1x1 digitalio] 'Line'
[4] [5] [1x1 digitalio] 'Line'
[5] [6] [1x1 digitalio] 'Line'
[6] [7] [1x1 digitalio] 'Line'
[7] [8] [1x1 digitalio] 'Line'
```

Line and Port Characteristics

As described in the preceding section, when the lines are added to a digital I/O object, they must be configured for either input or output. The values can be read from an input line and the values can be written to an output line. Whether a given hardware line is addressable for input or output depends on the port it resides on. Digital I/O ports can be classified into these two groups based on user's ability to address lines individually:

- **Port-configurable devices:** The lines associated with a port-configurable device can not be addressed individually. This means that all the lines must be configured for either input or output. If an attempt is made to mix the two configurations, an error is returned.

Any number of available port-configurable lines can be added to a digital I/O object. However, the engine will address all lines behind the scenes. For example, if one line is added to a DIO object, then user automatically get all lines. This means that if a digital I/O object contains lines



from a port-configurable device, and a value is written to one or more of those lines, then all the lines get written to even if they are not contained by the device object.

- **Line-configurable devices:** The lines associated with a line-configurable device can be addressed individually. This means that individual lines can be configured for either input or output. Additionally, the values can be read or written on a line-by-line basis.

The line and port characteristics can be returned with the `daqhwinfo` function. For example, Advantech card has six ports and eight digital I/O lines per port. To return the digital I/O characteristics for this card.

```
>>hwinfo = daqhwinfo(dio);
```

To display the line and port characteristics for the first port

```
>>hwinfo.Port(1)
```

```
ans =
```

```
    ID: 0
```

```
  LineIDs: [0 1 2 3 4 5 6 7]
```

```
 Direction: 'in/out'
```

```
  Config: 'line'
```

To display the line and port characteristics for the second port

```
>>hwinfo.Port(2)
```

```
ans =
```

```
    ID: 1
```

```
  LineIDs: [0 1 2 3 4 5 6 7]
```

```
 Direction: 'in/out'
```

```
  Config: 'port'
```

To display the line and port characteristics for the third port

```
>>hwinfo.Port(3)
```

```
ans =
```

```
    ID: 2
```

```
  LineIDs: [0 1 2 3 4 5 6 7]
```

```
 Direction: 'in/out'
```

```
  Config: 'port'
```

This information tells that all 48 lines can be configured for either input or output.

Referencing Individual Hardware Lines

As described in the preceding section, the lines can be accessed with the Line property or with a line object. To reference individual lines, either MATLAB indices or descriptive line names must be specified.

- **MATLAB Indices**

Every hardware line contained by a digital I/O object has an associated MATLAB index that is used to reference the line. When adding lines with the `addline` function, index assignments are made automatically. The line indices start at 1 and increase monotonically up to the number of line group members. The first line indexed in the line group represents the least significant bit (LSB), and the highest indexed line represents the most significant bit (MSB). Unlike adding channels with the `addchannel` function, the line indices can not be assigned manually with `addline`.

For example, the digital I/O object `dio` created in the preceding section had the MATLAB indices 1 through 8 automatically assigned to the hardware lines 0 through 7, respectively. To manually swap the hardware line order so that line ID 1 is the LSB, the appropriate index to `hwlines` is supplied and the `HwLine` property is used.

```
>>hwlines(1).HwLine = 1;
```

```
>>hwlines(2).HwLine = 0;
```

Alternatively, `Line` property can be used.

```
>>dio.Line(1).HwLine = 1;
```

```
>>dio.Line(2).HwLine = 0;
```

- **Descriptive Line Names**

Choosing a unique, descriptive name can be a useful way to identify and reference lines – particularly for large line groups. The descriptive names can be associated with hardware lines using the `addline` function. For example, suppose 8 lines to `dio` need to be added, and it is wanted to associate the name `TrigLine` with the first line added.

```
>>addline(dio,0,'out','TrigLine');
```

```
>>addline(dio,1:7,'out');
```

Alternatively, `LineName` property can be used.

```
>>addline(dio,0:7,'out');
```

```
>>dio.Line(1).LineName = 'TrigLine';
```

The line name can now be used to reference the line.

```
>>dio.TrigLine.Direction = 'in';
```

4.3.3-Example: Adding Lines for Advantech Hardware

This example illustrates various ways to add lines to a digital I/O object associated with an Advantech card. This board is a multiport device.

To add eight input lines to dio from port 0

```
>>addline(dio,0:7,'in');
```

To add four input lines and four output lines to dio from port 0

```
>>addline(dio,0:7,{'in','in','in','in','out','out','out','out'});
```

Suppose user want to add the first two lines from port 0 configured for input, and the first two lines from port 2 configured for output. There are four ways to do this. The first way requires only one call to addline since it uses the hardware line IDs, and not the port IDs.

```
>>addline(dio,[0 1 8 9],{'in','in','out','out'});
```

The second way requires two calls to addline, and specifies one line ID and multiple port IDs for each call.

```
>>addline(dio,0,[0 2],{'in','out'});
```

```
>>addline(dio,1,[0 2],{'in','out'});
```

The third way requires two calls to addline, and specifies multiple line IDs and one port ID for each call.

```
>>addline(dio,0:1,0,'in');
```

```
>>addline(dio,0:1,2,'out');
```

Lastly, four addline calls can be used – one for each line added.

4.4- Reading and Writing Operation

After the lines are added to a digital I/O object:

- Values to lines can be written
- Values from lines can be read

Note Unlike analog input and analog output objects, the behavior of digital I/O objects are not controlled by configuring properties. This is because buffered digital I/O is not supported, and data is not stored in the engine. Instead, either values are written directly, or values are read directly from one or more hardware lines.

4.4.1-Writing Digital Values

The values are written to one or more digital I/O lines with the **putvalue** function. **putvalue** requires the digital I/O object and the values to be written as input arguments. The values to be written can be specified as a decimal value or as a *binary vector* (**binvec**). A binary vector is a logical array that is constructed with the least significant bit (LSB) in the first column and the most significant bit (MSB) in the last column. For example, the decimal value 23 is written in **binvec** notation as $[1\ 1\ 1\ 0\ 1] = 2^0 + 2^1 + 2^2 + 2^4$. It can be judged that **binvecs** are easier to work with than decimal values since there is a clear association between a given line and the value (1 or 0) that is written to it. Decimal values can be converted into **binvec** values with the **dec2binvec** function

For example, suppose the digital I/O object **dio** is created and eight output lines are added to it from port 0.

```
>>dio = digitalio ('advantech',0);
```

```
>>addline (dio, 0:7,'out');
```

To write a value of 23 to the eight lines contained by **dio**, it can be written to the device object.

```
>>data = 23;
```

```
>>putvalue (dio, data);
```

Alternatively, it can be written to individual lines through the **Line** property.

```
>>putvalue (dio.Line (1:8), data)
```

To write a binary vector of values using the device object and the **Line** property

```
>>bvdata = dec2binvec(data,8);
```

```
>>putvalue(dio, bvdata);
```

```
>>putvalue (dio.Line(1:8), bvdata);
```

The second input argument supplied to `dec2binvec` specifies the number of bits used to represent the decimal value. Since the preceding commands write to all eight lines contained by `dio`, an eight element binary vector is required. If the number of bits is not specified, then the minimum number of bits needed to represent the decimal value is used.

Alternatively, the binary vector can be created without using `dec2binvec`.

```
>>bvdata = logical ([1 1 1 0 1 0 0 0]);
```

```
>>putvalue (dio, bvdata)
```

Rules for Writing Digital Values

Writing values to digital I/O lines follows these rules:

- If the digital I/O object contains lines from a port-configurable device, then the data acquisition engine writes to all port-configurable lines even if they are not contained by the device object.
- When writing decimal values:
 - ✓ If the value is too large to be represented by the lines contained by the device object, then an error is returned.
 - ✓ We can write to a maximum of 48 lines. To write to more than 48 lines, we must use a `binvec` value.
- When writing `binvec` values:
 - ✓ Any number of lines can be written.
 - ✓ There must be an element in the binary vector for each line to be written.
- The values can be read from a line configured for output.
- An error is returned if a negative value is written, or if value is written to a line configured for input.

4.4.2-Reading Digital Values

The values can be read from one or more digital I/O lines with the `getvalue` function. `getvalue` requires the digital I/O object as an input argument. An output argument can be specified optionally, which represents the returned values as a binary vector.

For example, suppose the digital I/O object `dio` are created & eight input lines are added from port 0.

```
>>dio = digitalio ('advantech',0);
```

```
>>addline (dio, 0:7,'in');
```

To read the current value of all the lines contained by dio and return the result to out

```
>>portval = getvalue (dio)
```

```
>>portval =
```

```
1 1 1 0 1 0 0 0
```

To read the current values of the first four lines contained by dio and return the result to out

```
>>lineval = getvalue (dio.Line (1:5))
```

```
lineval =
```

```
1 1 1 0 1
```

A binvec can be converted to a decimal value with the binvec2dec function. For example, to convert the binary vector lineval to a decimal value

```
>>out = binvec2dec (lineval)
```

```
>>out =
```

```
23
```

Rules for Reading Digital Values

Reading values from digital I/O lines follows these rules:

- If the digital I/O object contains lines from a port-configurable device, then all lines are read even if they are not contained by the device object. However, only values from the lines contained by the digital I/O object are returned.
- The value can be read from a line configured for output.
- For Advantech automation card, lines configured for input return a value of 1 by default.
- getvalue always returns a binary vector (binvec). To convert the binvec to a decimal value, use the binvec2dec function.

4.4.3-Example: Writing and Reading Digital Values

This example illustrates how to read and write digital values using a line-configurable subsystem. With line-configurable subsystems, values can be transferred on a line-by-line basis.

1. Create a device object: Create the digital I/O object dio for an Advantech automation card.

The installed adaptors and hardware IDs are found with daqhwinfo.

```
>>dio = digitalio ('advantech', 0);
```

2. Add lines: Add eight output lines from port 0 (line-configurable).

```
>>addline (dio, 0:7,'out');
```

3. Read and write values: Write a value of 13 to the first four lines as a decimal number and as a binary vector, and read back the values.

```
>>data = 13;
```

```
>>putvalue(dio.Line(1:4),data)
```

```
>>val1 = getvalue(dio);
```

```
>>bvdata = dec2binvec(data);
```

```
>>putvalue (dio.Line (1:4), bvdata)
```

```
>>val2 = getvalue (dio);
```

Write a value of 3 to the last four lines as a decimal number and as a binary vector, and read back the values.

```
>>data = 3;
```

```
>>putvalue(dio.Line(5:8),data)
```

```
>>val3 = getvalue(dio.Line(5:8));
```

```
>>bvdata = dec2binvec(data,4);
```

```
>>putvalue (dio.Line (5:8), bvdata)
```

```
>>val4 = getvalue (dio.Line (5:8));
```

Read values from the last four lines but switch the most significant bit (MSB) and the least significant bit (LSB).

```
>>val5 = getvalue (dio.Line (8:-1:5));
```

4. Clean up – When the dio is no longer needed, it should be removed from memory and from the MATLAB workspace.

```
>>>delete (dio)
```

```
>>clear dio
```

4.5-Generating Timer Events

The fact that analog input and analog output objects make use of data stored in the engine and clocked I/O leads to the concept of a “running” device object and the generation of events.

However, since the Data Acquisition Toolbox does not support buffered digital I/O operations, digital I/O objects do not store data in the engine. Additionally, reading and writing line values are not clocked at a specific rate in the way that data is sampled by an analog input or analog output subsystem. Instead, values are either written directly to digital lines with `putvalue`, or read directly from digital lines with `getvalue`.

Therefore, the concept of a running digital I/O object does not make sense in the same way that it does for analog input and analog output objects. However, a digital I/O object can be run to perform one task: generate timer events.

Timer events can be used to update and display the state of the digital I/O object.

4.5.1-Timer Events

The only event supported by digital I/O objects is a timer event. Timer events occur after a specified period of time has passed. Properties associated with generating timer events are given below.

Property Name	Description
Running	Indicates if the device object is running.
TimerFcn	Specifies the M-file callback function to execute whenever a predefined period of time passes.
TimerPeriod	Specifies the period of time between timer events.

Table 4.3-Digital I/O Timer Event Properties

A timer event is generated whenever the time specified by `TimerPeriod` passes. This event executes the callback function specified for `TimerFcn`. Time is measured relative to when the device object starts running (`Running` is ON).

Some timer events may not be processed if our system is significantly slowed or if the `TimerPeriod` value is too small. For example, a common application for timer events is to display data. However, since displaying data can be a CPU-intensive task, some of these events may be

dropped. For digital I/O objects, timer events are typically used to display the state of the device object.

4.5.2-Starting and Stopping a Digital I/O Object

The start function is used to start a digital I/O object. For example, to start the digital I/O object dio

```
>>start (dio)
```

After start is issued, the Running property is automatically set to ON, and timer events can be generated. If an attempt is made to start a digital I/O object that does not contain any lines or that is already running, then an error is returned.

A digital I/O object will stop executing under these conditions:

- The stop function is issued.
- An error occurred in the system.

When the device object stops, Running is automatically set to OFF.

4.5.3-Example: Generating Timer Events

This example illustrates how to generate timer events for a digital I/O object. The callback function daqcallback displays the event type and device object name. Note that, stop command must be issued to stop the execution of the digital I/O object.

1. Create a device object – Create the digital I/O object dio for an Advantech automation card. The installed adaptors and hardware IDs are found with daqhwinfo.

```
>>dio = digitalio ('advantech', 0);
```

2. Add lines – Add eight input lines from port 0 (line-configurable).

```
>>addline (dio, 0:7,'in');
```

3. Configure property values – Configure the timer event to call daqcallback every five seconds.

```
>>set (dio, 'TimerFcn', @daqcallback)
```

```
>>set (dio, 'TimerPeriod',5.0)
```

Start the digital I/O object. Stop command must be issued when timer events are no longer want to be generated.

```
>>start (dio)
```

The pause command ensures that two timer events are generated.

```
>>pause (11)
```

4. Clean up – When dio is no longer needed, it should be removed from memory and from the MATLAB workspace.

```
>>delete (dio)
```

```
>>clear dio
```

4.6-Evaluating the Digital I/O Object Status

At any time after a digital I/O object is created, its status can be evaluated by:

- Returning the value of the Running property
- Invoking the display summary

The Display Summary

The display summary can be invoked by typing the digital I/O object at the command line. The displayed information is designed so that the status of our data acquisition session can be evaluated quickly. The display is divided into two main sections: general summary information and line summary information.

- **General Summary Information**

The general display summary includes the device object type and the hardware device name, followed by the port parameters. The port parameters include the port ID, and whether the associated lines are configurable for reading or writing.

- **Line Summary Information**

The line display summary includes property values associated with:

- ✓ The hardware line mapping
- ✓ The line name
- ✓ The port ID
- ✓ The line direction

PCI 1751 Digital I/O Card

5.1-General Information

5.1.1-Introduction

The PCI-1751 is a 48-bit DI/O and counter/timer card with PCI bus [17]. It provides 48 bits of parallel digital input/output as well as 3 timers. It emulates mode 0 of the 8255 PPI chip, but the buffered circuits offer a higher driving capability than the 8255.

The card emulates two 8255 PPI chips to provide 48 DI/O bits. The I/O bits are divided into six 8-bit I/O ports: A0, B0, C0, A1, B1 and C1. Each port can be configured as either input or output via software. The dual interrupt handling capability provides users the flexibility to generate interrupts to a PC. A pin in the connector can output a digital signal simultaneously with the card's generating an interrupt. This card uses a high density SCSI 68-pin connector for easy and reliable connections to field devices.

Two other features give the PCI-1751 practical advantages in an industrial setting. When the system is hot reset (the power is not turned off) the PCI-1751 retains the last I/O port settings and output values if the user has set jumper JP4 to enable this feature. Otherwise, port settings and output values reset to their safe default state, or to the state determined by other jumper settings. The PCI-1751's other useful feature is it supports both wet and dry contacts, allowing it to interface with other devices more easily.

5.1.2-Numbering Convention: All numbers given in this report are in decimal format unless specifically noted otherwise. In particular, where a register address is given as (Base + 32), the decimal number "32" should be added to the base value.

5.1.3-Features

- 48 TTL level digital I/O lines.
- Emulates mode 0 of 8255 PPI
- Buffered circuits provide higher driving capability

- Interrupt handling
- Interrupt output pin for simultaneously triggering external devices with the interrupt
- High density SCSI 68-pin connector
- Output status readback
- Two 16-bit timers can be cascaded to one 32-bit timer, and can generate watchdog timer interrupts
- One 16-bit event counter can generate event interrupts
- Keeps port I/O settings and digital output states after hot system reset
- Supports dry contact and wet contact

5.1.4-Applications

- Industrial AC/DC I/O devices monitoring and control
- Relay and switch monitoring and control
- Parallel data transfer
- Sensing the signals of TTL, DTL, CMOS logic
- Driving indicator LEDs

5.1.5-Specifications

I/O channels: 48 digital I/O lines

Programming mode: 8255 PPI mode 0

5.1.6-Input Signal

- Logic high voltage: 2.0 to 5.25 V
- Logic low voltage: 0.0 to 0.80 V
- High level input current: 20 μ A
- Low level input current: -0.2 mA

5.1.7-Output Signal

- Logic high voltage: 2.4 V minimum.
- Logic low voltage: 0.4 V maximum

- High level input current: 15 mA maximum (source)
- Low level input current: 24 mA maximum (sink)
- Driving capability: 15 LS TTL

5.1.8-Interrupt Source

- PC00, PC04, PC10, PC14, Timer 1 and Counter 2.

5.1.9-Transfer Rate

(This value depends on software and speed of computer.)

- Typical: 1 MB/sec (tested under DOS, Pentium® 100 MHz CPU)
- Maximum: 1.5 MB/sec

5.1.10-Other Details

Connector: One SCSI-II 68-pin female connector

Power consumption: 5 V @ 850 mA (Typical)

5 V @ 1.0 A (Max.)

Operating temperature: 0 ~ 70° C (32° F ~ 158 °F)

Storage temperature: -20 ~ 80° C(-4° F ~ 176° F)

Humidity: 5% ~ 95% non-condensing

Dimension: 170 x 100 mm (6.9" x 3.9")

5.2-Installation

5.2.1-Initial Inspection

Before starting to install the PCI-1751, it should be made sure that there is no visible damage on the card. It should be free of marks and in perfect order on receipt.

5.2.2-Unpacking

The PCI-1751 contains components that are sensitive and vulnerable to static electricity. The static electricity should be discharged from our body to ground by touching the back of the system unit (grounded metal) before we touch the board.

Remove the PCI-1751 card from its protective packaging by grasping the card's rear panel. Card should be handled only by its edges to avoid static discharge which could damage its integrated circuits. Whenever the card is removed from the PC, it should be stored in its package for its protection.

The contact with materials that hold static electricity such as plastic, vinyl and Styrofoam should be avoided.

The product contents inside the packing should be checked. There should be one card, one CD-ROM, and it's manual. It should be made sure that nothing is missing.

5.2.3-Jumper Settings

The PCI-1751 is designed with ease-of-use in mind. It is a "plug and play" card, i.e. the system BIOS assigns the system resources such as base address and interrupt automatically. There are only two functions with 11 jumpers to be set by the user. The following section describes how to configure the card. We may want to refer to the figure below for help in identifying card components.

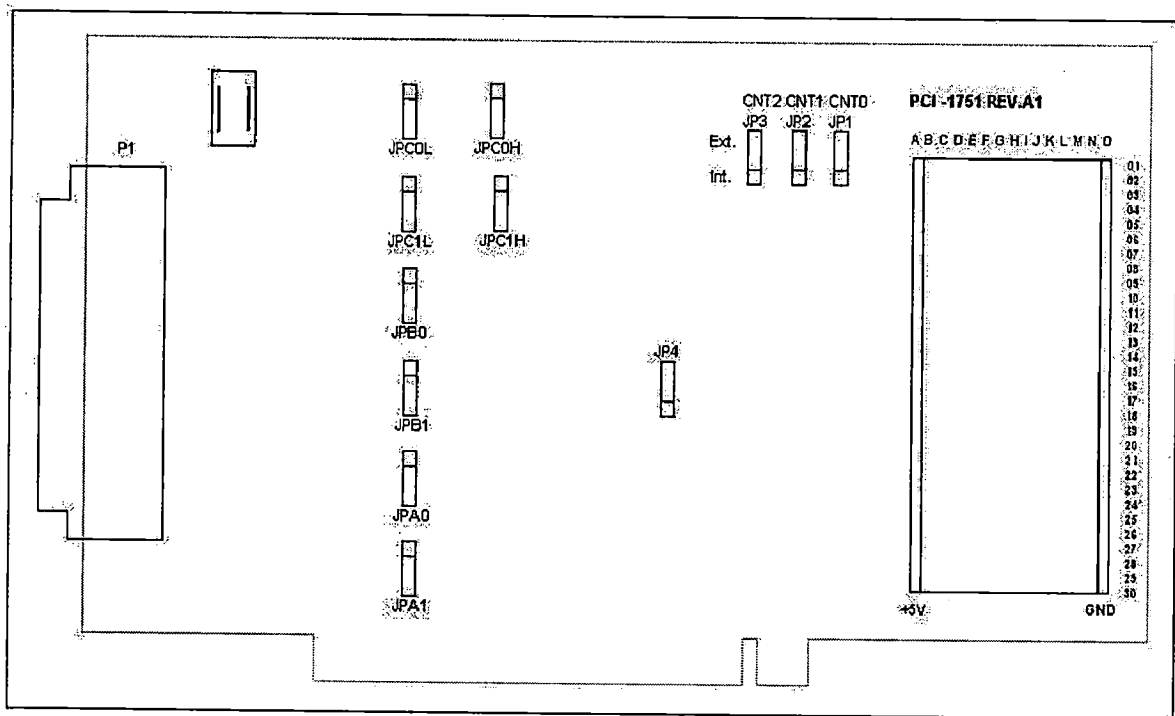


Figure 5.1-Location of Connectors and Jumpers

5.2.4-Jumper Settings to Set Ports as Input or Output by Software

By shorting the upper two pins of jumpers JPA0, JPB0, JPC0L, JPC0H, JPA1, JPB1, JPC1L or JPC1H, a user sets the corresponding ports to be configurable as input or output ports by software. (JPA0 means jumper for port A0, JPB0 means jumper for port B0, etc.) The initial state of each port after system power on or reset will be set as input logic 1 (voltage high), provided that no external signals are connected, and provided jumper JP4 does not determine otherwise (see Jumper J4 discussion below).

5.2.5-Using Jumpers to Set Ports as Output Ports

By shorting the lower two pins of the jumpers JPA0, JPB0, JPC0L, JPC0H, JPA1, JPB1, JPC1L or JPC1H, a user sets the corresponding ports to be output ports. (JPA0 means jumper for port A0, JPB0 means jumper for port B0, etc.) Shorting the lower two pins of a port's jumper pins disables the port from being software configurable as an input port. The initial state of each of these ports after system power on or reset will be logic 0 (voltage low), unless jumper JP4 determines otherwise.

5.2.6-Jumper JP4 Restores Ports to Their Condition Prior to Reset

Jumper JP4 gives the PCI-1751 a new and valuable capability. With JP4 enabled, the PCI-1751 memorizes all port I/O settings and output values, and, in the event of a hot reset, the settings and output values present at the port just prior to reset are restored to each port following reset. This feature applies to both ports set by software, and to ports configured as output ports via jumper. Depending on the application, this capability may allow a card to be reset without requiring a complete shutdown of processes controlled by the card (since port values are left unchanged and are interrupted only momentarily).

Complete loss of power to the chip clears chip memory. Thus, even if JP4 is enabled, if the power to the card is disconnected, the card's initial power-on state will be the default state (for software-set ports) or the state of an output port with voltage low output (for jumper-set ports).

When jumper JP4 is not enabled, power-off or reset results in ports returning to their default state (for software-set ports) or returning to the state of output port with voltage low output (for jumper-set ports).

5.2.7-Select Clock Source of Timers and Counter

Jumpers JP1, JP2 and JP3 are used to select the clock source of Timer 0, Timer 1 and Counter 2, respectively. The upper two pins of the jumpers must be shorted to select an external clock source, or the lower two pins should be shorted to select an internal clock source. However, the internal clock source of Timer 1 is connected to the output of Timer 0, so shorting the upper two pins of JP2 results in the cascading of Timer 0 and Timer 1 as a 32-bit timer.







Names of Jumpers	Function description	
JPA0, JPA1: Jumpers for ports A0, A1 JPB0, JPB1: Jumpers for ports B0, B1		Sets port as an output port.
JPC0L, JPC1L: Jumpers for low nibble of ports C0, C1 JPC0H, JPC1H: Jumpers for high nibble of ports C0, C1		Sets port to be software configurable as input or output (default)
JP1: Timer 0		Internal counter clock source
JP2: Timer 1		
JP3: Counter 2		External counter clock source (default)
JP4		All ports return to state held just prior to reset
		All ports return to default states (for software-set ports) or to output port output low (for jumper-set ports)

Table 5.1-Summary of Jumper Settings

5.2.8-PCI-1751 Block Diagram

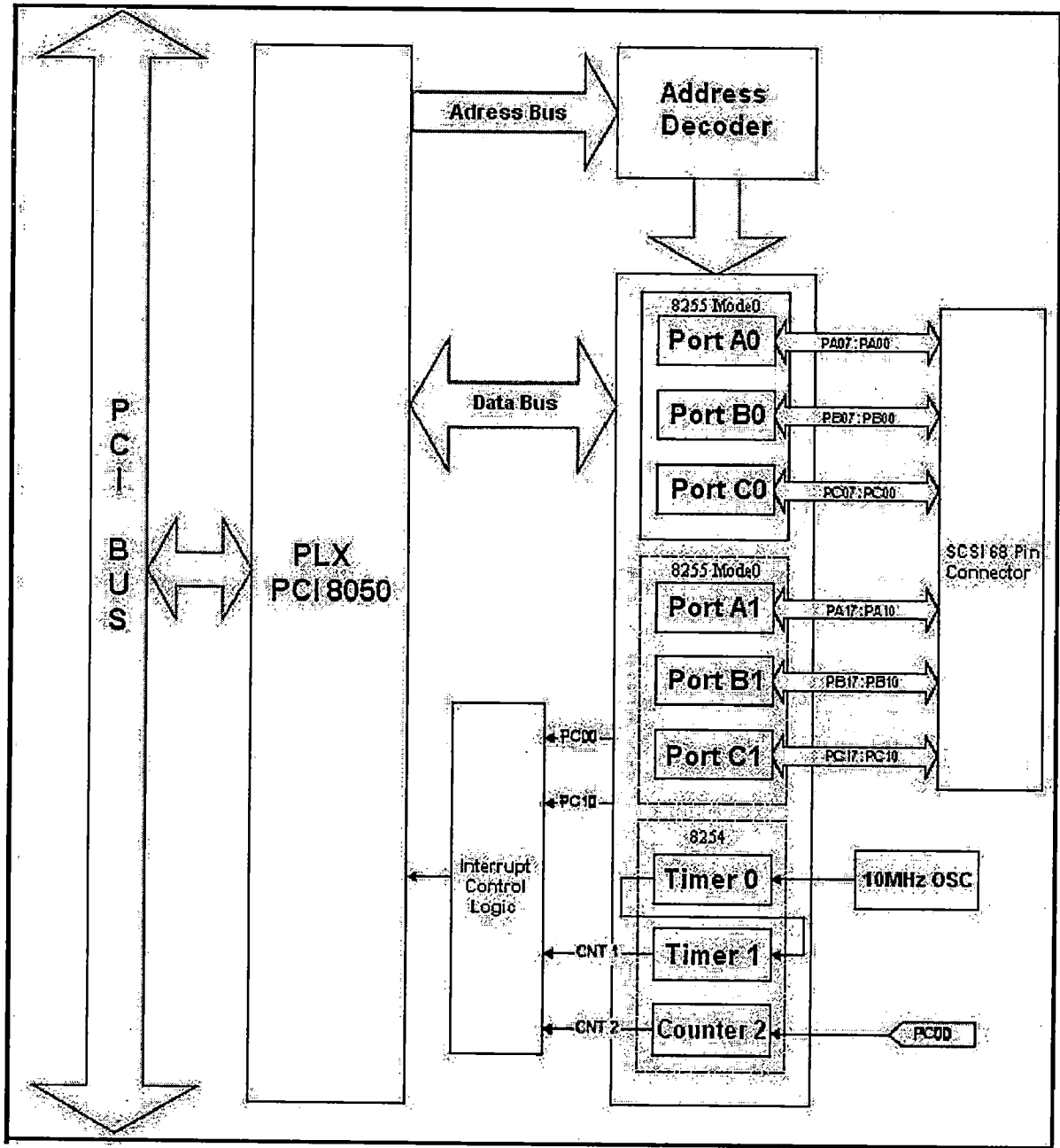


Figure 5.2-PCI-1751 Block Diagram

5.2.9-Pin Assignments

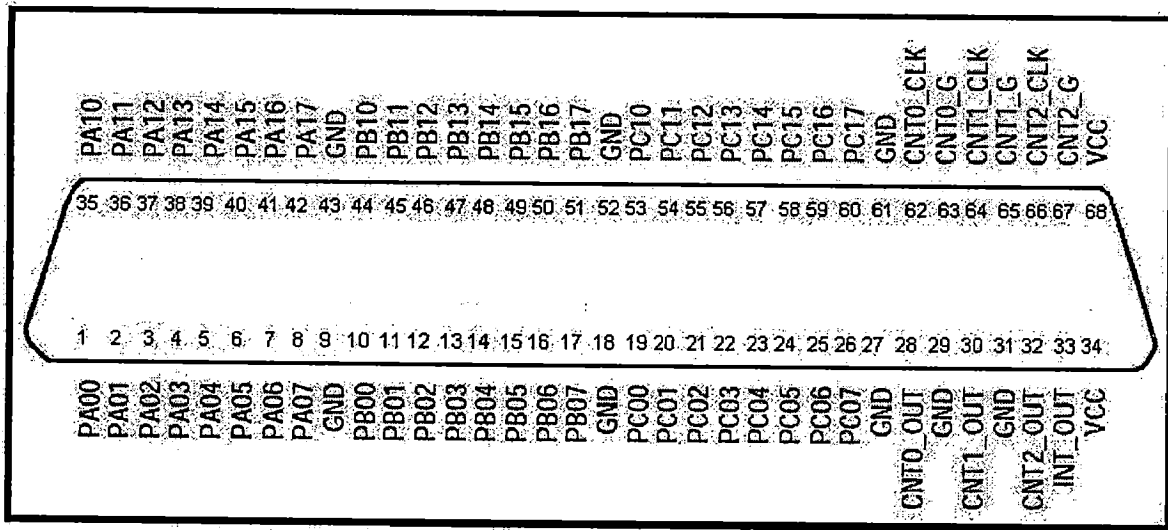


Figure 5.3-Pin Diagram of PCI-1751

Description of pin use:

PA00 ~ PA07: I/O pins of Port A0

PA10 ~ PA17: I/O pins of Port A1

PB00 ~ PB07: I/O pins of Port B0

PB10 ~ PB17: I/O pins of Port B1

PC00 ~ PC07: I/O pins of Port C0

PC10 ~ PC17: I/O pins of Port C1

CNT0_OUT, CNT1_OUT and CNT2_OUT: Output pins of Counter/Timer 0, 1 and 2

CNT0_CLK, CNT1_CLK and CNT2_CLK: External clock source of Counter / Timer 0, 1 and 2

CNT0_G, CNT1_G and CNT2_G: Gate control pins of Counter / Timer 0, 1 and 2

INT_OUT: Interrupt output. This pin changes to logic 1 whenever the PCI-1751 generates an interrupt, and returns to logic 0 when the interrupt is cleared.

GND: Ground

VCC: +5 V_{DC} voltage output

5.2.10-Installation Instructions

The PCI-1751 can be installed in any PCI slot in the computer. However, the computer user's manual should be referred to avoid any mistakes and danger before following the installation procedure below:

1. The computer and any accessories connected to the computer should be turned off.
Warning: The computer power supply should be turned off whenever any card is installed or removed, or any cables are connected or disconnected.
2. The power cord and any other cables from the back of the computer should be disconnected.
3. Then remove the cover of the computer.
4. Select an empty 5 V PCI slot. Remove the screw that secures the expansion slot cover to the system unit. Save the screw to secure the interface card retaining bracket.
5. Carefully grasp the upper edge of the PCI-1751. Align the hole in the retaining bracket with the hole on the expansion slot and align the gold striped edge connector with the expansion slot socket. Press the card into the socket gently but firmly. Make sure the card fits the slot tightly.
6. Secure the PCI-1751 by screwing the mounting bracket to the back panel of computer.
7. Attach any accessories (68-pin cable, wiring terminal, etc.) to the card.
8. Replace the cover of computer. Connect the cables which were removed in step 2.
9. Turn the computer power on.

5.3-Operation

This section describes the operating characteristics of the PCI-1751. The driver software provided allows a user to access all of the card's functions without register level programming. Please see the User's Manual for the driver bundled with this card for more information. For users who prefer to implement their own bit-level programming to drive the card's functions, information useful for making such a program is included in this chapter.

5.3.1-Digital I/O Ports

Introduction

The PCI-1751 emulates two 8255 programmable peripheral interface (PPI) chips in mode 0, but with higher driving capability than a standard 8255 chip. Each of the 8255 chips has 24 programmable I/O pins that are divided into three 8-bit ports. The total 48 DI/O pins from both chips are divided into 6 ports, designated PA0, PB0, PC0, PA1, PB1 and PC1. Each port can be programmed as an input or an output port. The I/O pins in port A0 are designated PA00, PA01, ..., PA07; the pins in port B0 are designated PB00, PB01, ..., PB07, etc. These port names are used both in this manual and in the software library. Refer to Section 5.2.9, Pin Assignments.

8255 Mode 0

The basic functions of 8255 mode 0 include:

- Two 8-bit I/O ports - port A (PA) and port B (PB)
- Port C is divided into two nibble-wide (4-bit) I/O ports:- PC upper and PC lower
- Any port can be used for either input or output.
- Output status can be read back.

Interrupt Function of the DIO Signals

Two I/O pins (PC00 and PC10) can be used to generate hardware interrupts. A user can program the interrupt control register (Base + 32) to select the interrupt sources. Refer to section 5.3.3- Interrupt Function on page 87 in this chapter for details about interrupt control.

Input/Output Control

A control word can be written to a port's configuration register (Base+3 for port 0 and Base+7 for port 1) to set the port as an input or an output port, unless the ports are set as output ports via jumpers. Table 5.2 shows the format of a control word.

D7	D6	D5	D4	D3	D2	D1	D0
Don't care	Don't care	Don't care	Port A 0: output 1: input	Port C higher bits 0: output 1: input	Don't care	Port B 0: output 1: input	Port C lower bits 0: output 1: input

Table 5.2-Bit Map of Port Configuration Register

Note: A control word has no effect if the corresponding port is set as an output port by a jumper.

Warning: Before setting any port as an output port via software, it should be ensured that a safe output value has also been set. An output voltage will appear at the pins immediately following the control word taking effect. If no output value was specified, the value will be indeterminate (either 0 or 1), which may cause a dangerous condition.

Initial Configuration

The initial configuration of each port depends on the input/output jumper setting of each port, on the setting of the jumper JP4, and on whether the power was actually disconnected or whether the system was hot reset.

If jumper JP4 is not enabled, all ports configured by software are automatically set as input ports during system start up or reset, with a default signal level of logic 1 (high). All ports are set via jumpers as output ports during system start up or reset, signal level logic 0 (0 V).

If the jumper JP4 is enabled and the initial configuration is caused by a reset, all ports will return to the states they had just prior to the reset. The reset must be a "hot" reset (power not disconnected) for enabled JP4 to return ports to their prior values. Otherwise, the card behaves as though JP4 were not enabled.

Dry Contact Support for Digital Input

Each digital input channel accepts either dry contact or 0~5 V_{DC} wet contact inputs. Dry contact capability allows the channel to respond to changes in external circuitry (e.g., the closing of a switch in the external circuitry) when no voltage is present in the external circuit. **Figure 5.4** shows external circuitry with both wet and dry contact components, connected as an input source to one of the card's digital input channels.

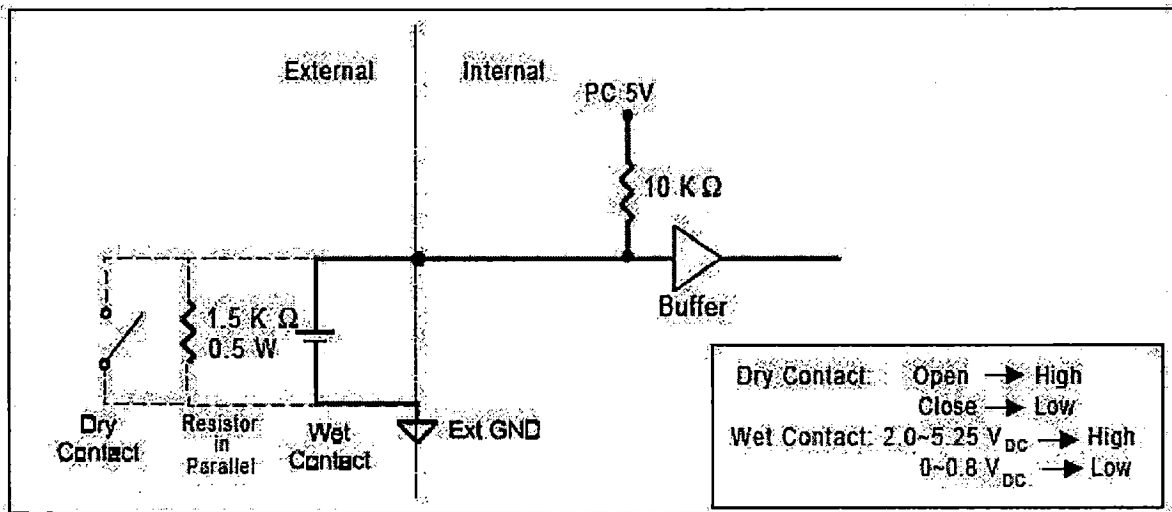


Figure 5.4-Wet and Dry Contact Inputs

Note: For wet contact configurations, a malfunction may occur if the internal resistance of the voltage source is significant ($> 1.5 \text{ k}\Omega$). It is advisable to connect a $1.5 \text{ k}\Omega$ resistor in parallel with such a voltage source to avoid a voltage rise inside the voltage source.

5.3.2 Timer/Counter Operation

Introduction

The PCI-1751 includes one 8254 compatible programmable timer/ counter chip which provides three 16-bit counters, designated as Timer 0, Timer1 and Counter 2. Each has 6 operation modes. Timer 0 and Timer 1 can be used separately or can be cascaded to create one 32-bit timer. Both Timer 1 and Counter 2 can generate interrupts to the computer. Please refer to Appendix A for more information on the operation modes of the counter chip. The block diagram of the timer/ counter system is shown in Figure 5.5.

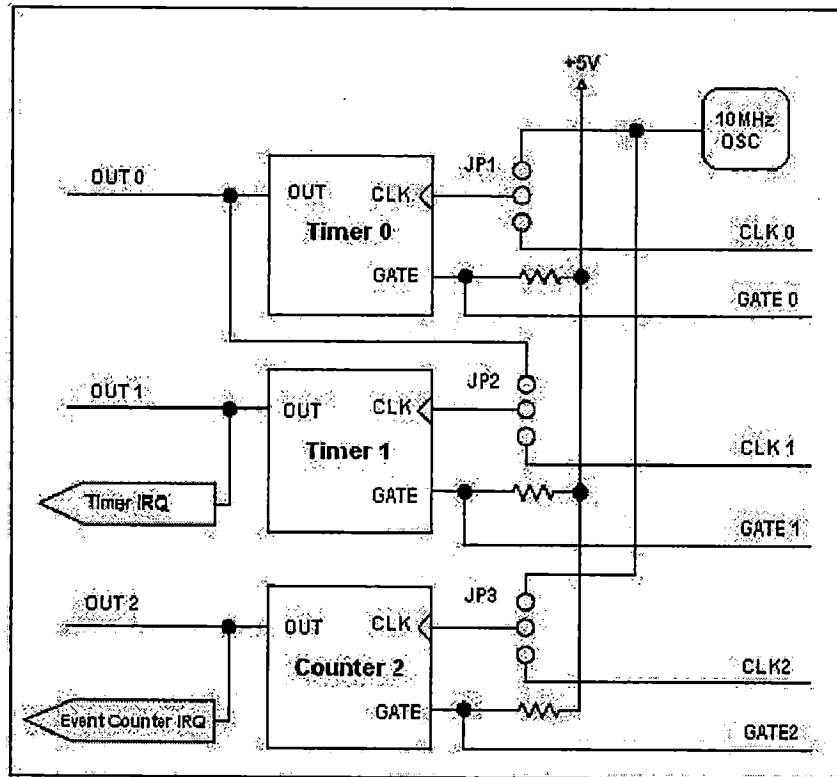


Figure 5.5-Timer and Counter Structure

Timer 0 & 1:

Two 16-bit Timers or One 32-bit Timer

Timer 0 and Timer 1 of the counter chip can be used separately or can be cascaded to create a 32-bit programmable timer by setting jumper JP2. By setting the clock source of Timer 1 to be an external source, you can use Timer 0 and Timer 1 as two separate 16 bit timers. By setting the clock source of Timer 1 to be the output of Timer 0 (internal source) these two timers are cascaded to become one 32-bit timer.

Setting jumper JP1 sets the clock source of Timer 0 to be external, and this allows Timer 0 and Timer 1 to be cascaded into a 32-bit event counter.

Counter 2

Counter 2 can be a 16-bit timer or an event counter, selectable by setting JP3. When the clock source is set for an internal source, Counter 2 is a 16-bit timer; when set as an external source, Counter 2 is an event counter. Counter 2 is set as mode 0 (interrupt on terminal count) in the driver provided by Advantech.

Timer/Counter Frequency and Interrupt

The input clock frequency of the counter/timers is 10 MHz. The output of both Timer 1 and Counter 2 can generate interrupts for the system. The maximum and minimum timer interrupt frequency is $(10 \text{ MHz}) / (2) = (5 \text{ MHz})$ and $(10 \text{ MHz}) / (65535 * 65535) = 0.002328 \text{ Hz}$, respectively.

The gates of the counter/timers are internally pulled to +5 V when gate control is enabled, but a user can also set it using the connector pins (CNT0_G, CNT1_G and CNT2_G).

5.3.3-Interrupt Function

Introduction

Two lines in each I/O port (C0 and C4) and two of the three counter outputs (Timer 1 and Counter 2) are connected to the interrupt circuitry. The Interrupt Control Register of the PCI-1751 controls how the combination of the 6 signals generates an interrupt. Two interrupt request signals can be generated at the same time, and then the software can service these two request signals by ISR. The dual interrupt sources provide the card with more capability and flexibility.

IRQ Level

The IRQ level is set automatically by the PCI plug and play BIOS and is saved in the PCI controller. There is no need for users to set the IRQ level. Only one IRQ level is used by this card, although it has two interrupt sources.

Interrupt Control Register (Base + 32)

The Interrupt Control Register (Base + 32) controls the interrupt signal source, edge and flag. Table 5.3 shows the bit map of the interrupt control register. The register is a readable/writable register. When writing to it, it is used as a control register, and when reading from it, it is used as a status register.

Port #	Port 1				Port 0			
Bit #	D7	D6	D5	D4	D3	D2	D1	D0
Abbreviation	F1	E1	M11	M10	F0	E0	M01	M00

Table 5.3-Interrupt Control Register Bit Map

M00 and M01: "mode bits" of port 0

M10 and M11: "mode bits" of port 1

E0, E1: triggering edge control bits

F0, F1: flag bits

Interrupt Source Control

The "mode bits" in the interrupt control register determine the allowable sources of signals generating an interrupt. Bit 0 and bit 1 determine the interrupt source for port 0, and bit 4 and bit 5 determine the interrupt source for port 1, as indicated in Figure 5.6. Table 5.4 shows the relationship between an interrupt source and the values in the mode bits.

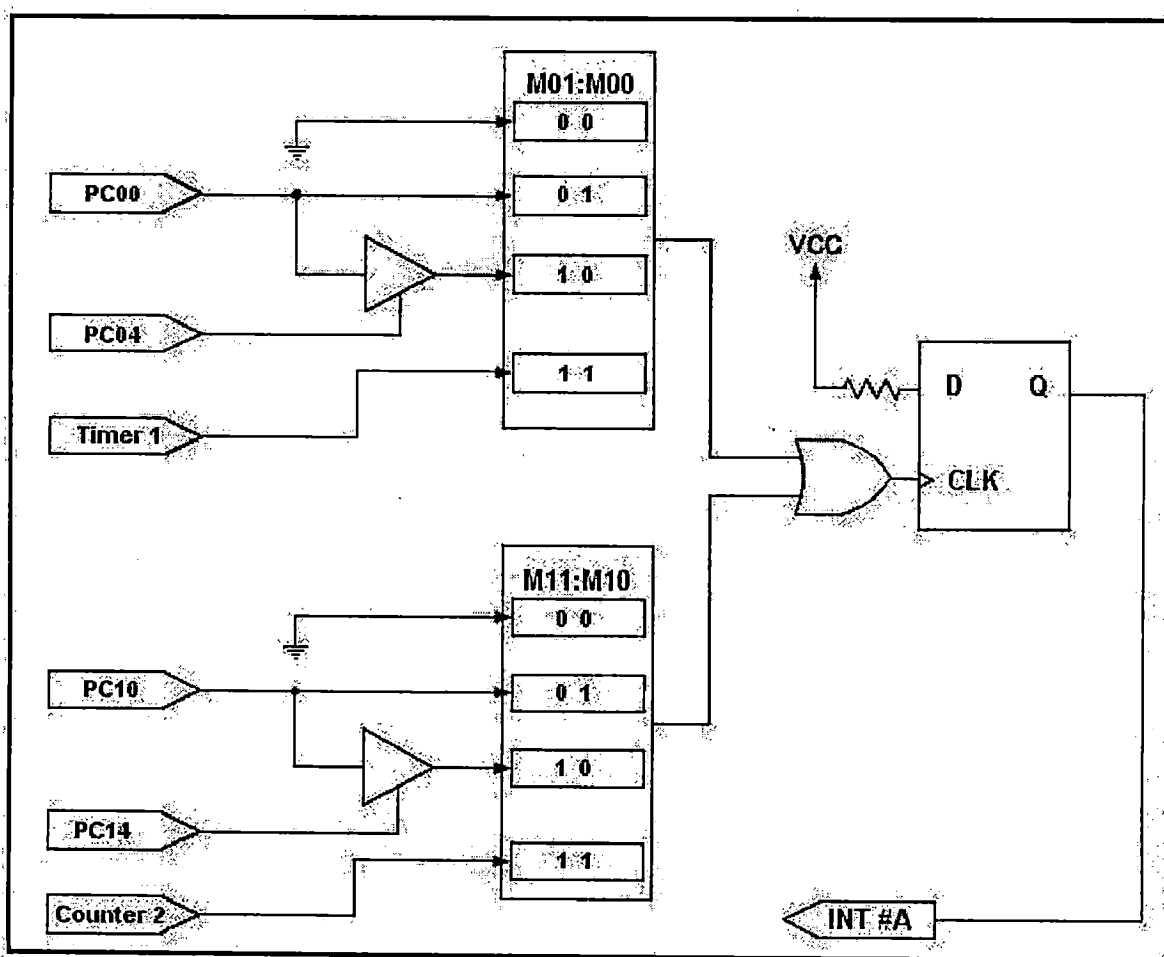


Figure 5.6-Interrupt Sources

Port 1			Port 0		
M11	M10	Description	M01	M00	Description
0	0	Disable interrupt	0	0	Disable interrupt
0	1	Source = PC10	0	1	Source = PC00
1	0	Source = PC10 & PC14	1	0	Source = PC00 & PC04
1	1	Source = Counter 2	1	1	Source = Timer 1

Table 5.4-Interrupt Mode Bit Values

Interrupt Triggering Edge Control

The interrupt can be triggered by a rising edge or a falling edge of the interrupt signal, selectable by the value written in the "triggering edge control" bit in the interrupt control register, as shown in Table 5.5.

E0 OR E1	Triggering edge of interrupt signal
1	Rising edge trigger
0	Faling edge trigger

Table 5.5-Triggering Edge Control Bit Values

Interrupt Flag Bit

The "interrupt flag" bit is a flag indicating the status of an interrupt. It is a readable and writable bit. Read the bit value to find the status of the interrupt, write "1" to this bit to clear the interrupt. This bit must be cleared in the ISR to service the next incoming interrupt.

F0 & F1		Interrupt Status
Read	1	Interrupt Exist
	0	No Interrupt
Write	1	Clear Interrupt
	0	Don't Care

Table 5.6-Interrupt Flag Bit Values

THIN FILMS AND RESISIVITY MEASUREMENT

6.1-Elementary

According to Ohm's law for circuit theory, the resistance of a material is the applied voltage V divided by the current I drawn across the material across two electrodes.

$$R = V/I$$

Where:

R = resistance (Ohms, Ω)

V = voltage (Volts, V)

I = current (Amperes, A)

This electrical resistance is proportional to the sample's length L and the resistivity ρ and inversely proportional to the sample's cross sectional area S .

$$R = \rho L/S$$

Where again:

ρ = resistivity (Ωm). The resistivity is a property of a material and does not depend on geometry

S = cross-sectional area (m^2) = width \times thickness

L = length (m)

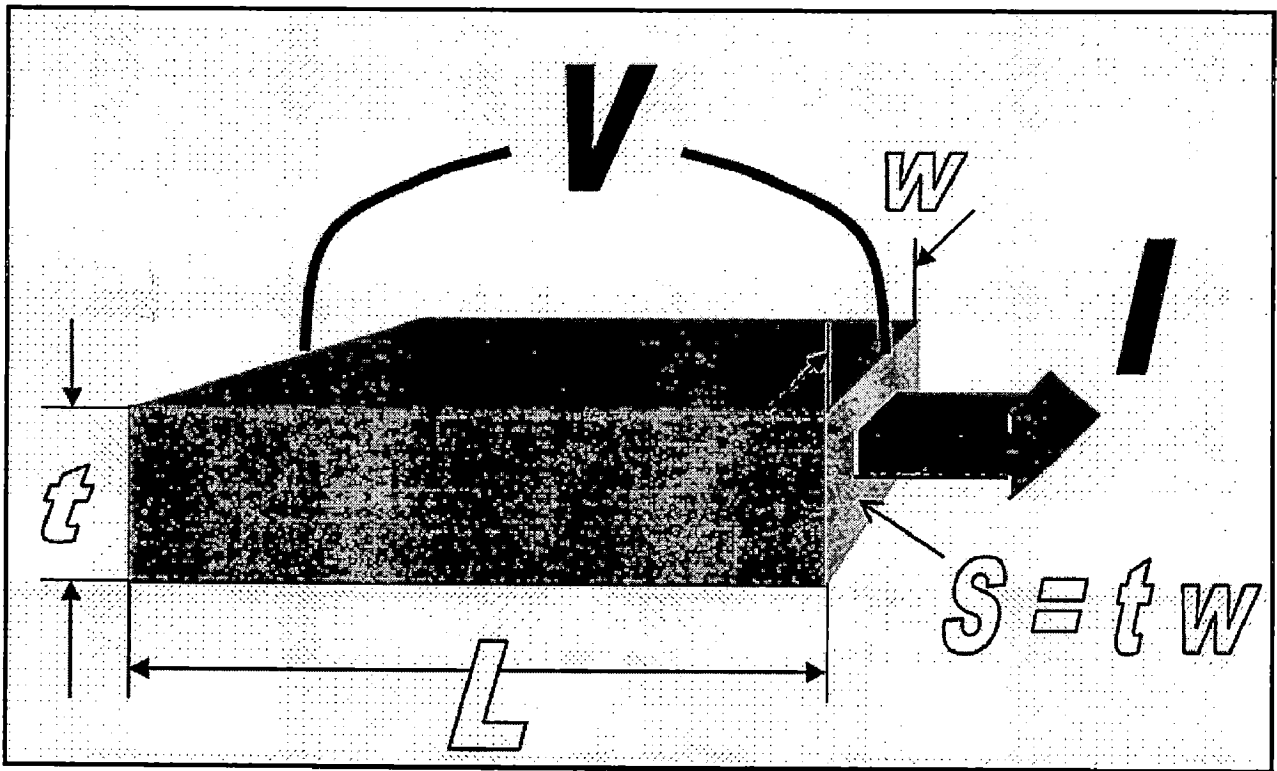


Figure 6.1-Elementary Geometry of Resistance Measurements

6.1.1-Parallel and Series Connections

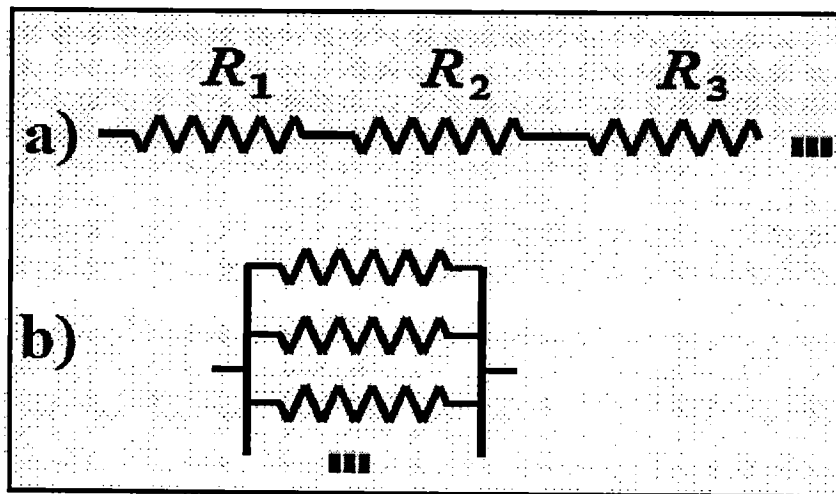


Figure 6.2-Resistances Connected in (a) Series (b) Parallel

For N serially connected resistors, $R = R_1 + R_2 + R_3 + \dots R_i \dots + R_N$. For N resistors connected in parallel, $1/R = 1/R_1 + 1/R_2 + 1/R_3 + \dots 1/R_i \dots + 1/R_N$

6.1.2-Sheet Resistance

Thin films usually have their thickness t ($\leq 1\mu m$) much smaller than all other dimensions, $t \ll w, L$. For the conducting thin films, it is convenient to introduce the so called sheet resistance (or resistance per square), R_{\square} ($\Omega/square$) "Square" means the known geometrical figure in its common sense, i.e. a parallelogram having four equal sides and four right angles (width = length, $w = L$). For a given thickness, it does not matter, whether the square is cm or μm large. Indeed, above written equation yields: $R = \rho L/wt = \rho/t = const$ which means that R_{\square} does not depend on w and L for $w = L$.

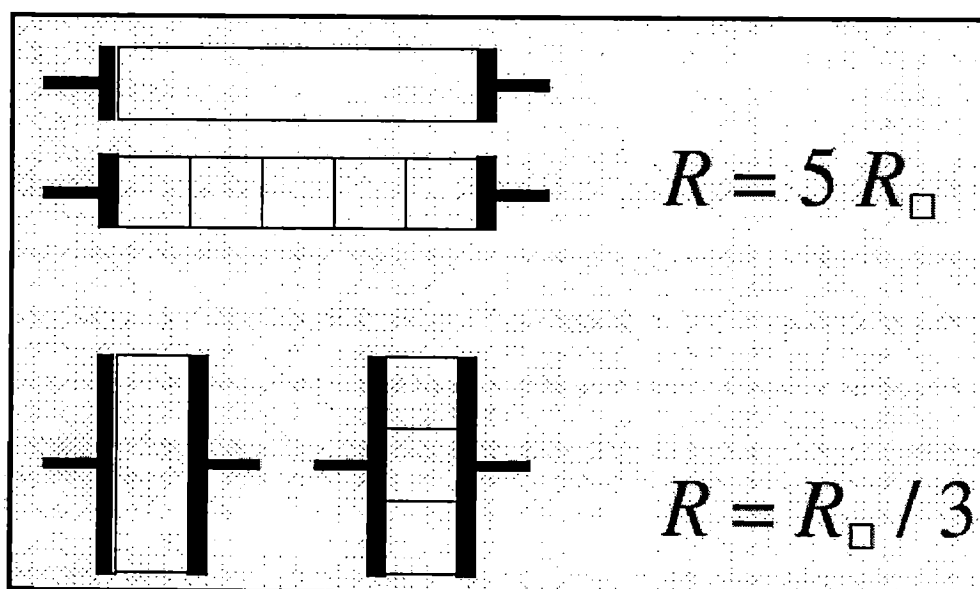


Figure 6.3-Sheet Resistance

To get the actual resistance of a strip, just count up the number of squares which fit inside the strip. Black bars represent equipotential contact pads with resistances negligibly smaller compared to the resistance of thin film.

Given R_{\square} for a thin film, it is easy to roughly estimate the resistance of the thin film of a simple shape: just count up the number of imaginary squares which fit inside the shape. For M squares in line with the bias current, the overall resistance will be $M \times R_{\square}$, and for those perpendicular to the bias current, the answer will be R_{\square}/M , (see Figure 6. 3).

6.1.3-Simple Resistance Measurements

1) Two-Probe Measurements

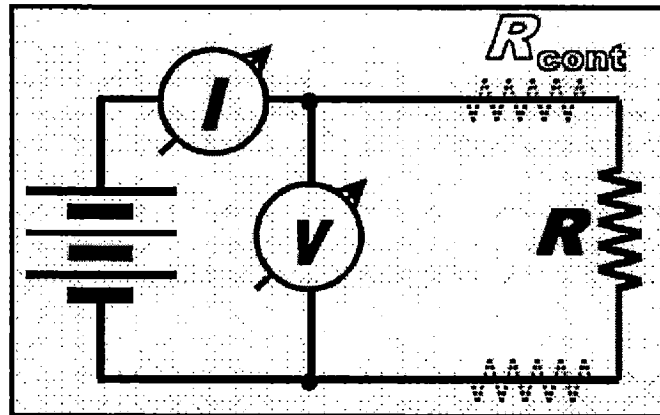


Figure 6.4-Two-Probe Resistance Measurements

Unknown contact resistance gives error.

In the two-probe measurements, an unknown resistance of connecting wires + contact resistances (together denoted as two R_{cont}) are contributing to the voltage V making the measurements erroneous, $V/I = I(2R_{cont}+R)/I = (2R_{cont}+R) \neq R$. Only acceptable if $R_{cont} \ll R$.

2) Four-Probe Measurements

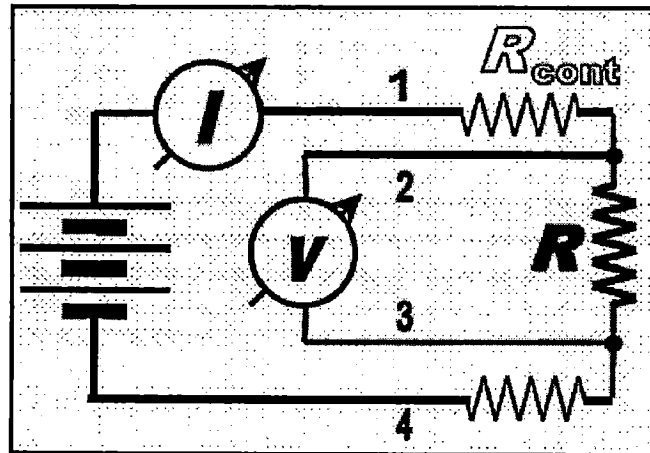


Figure 6.5-Four-Probe Resistance Measurements

Contact resistance does not contribute to the voltage V . Probes 1 and 4 are usually called “current probes”, while 2 and 3 are called as “voltage probes”.

6.2-Practical schemes

6.2.1-Patterned Thin Films: Nice Geometry

In order to measure resistivity of a thin film, as one of the characteristics of the thin-films material, one has to know the geometrical factors. One can use micro-technological thin-film patterning techniques for making slabs or strips with very well defined geometry. A typical pattern can be seen in Figure 6.6.

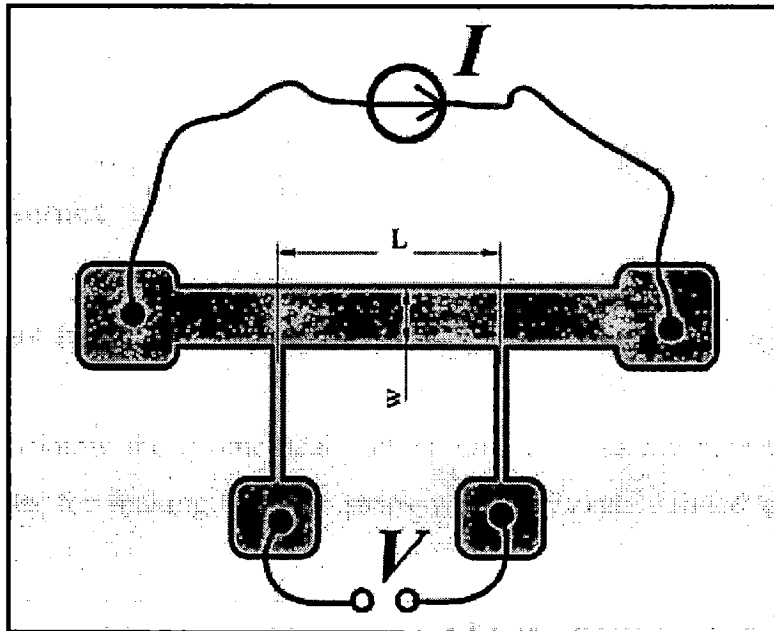


Figure 6.6-Slab-Shape Pattern of Thin Film

Thin film of thickness t patterned into a slab of width w . The distance between the voltage contacts is L . Figure 6.6 shows a typical slab-shape pattern which can be used for inferring the resistivity from measured I and V .

6.2.2-Patterned Thin Films: Arbitrary Shapes

In a case of vague geometry or complex shapes, it is difficult to calculate the resistivity from the measured voltage and current. In this case it is convenient to use the so called **van der Pauw method** [10], sketched in Figure 6.7. This method uses four isolated contacts on the boundary of an arbitrarily shaped thin film. At least two measurements should be done, according to the geometry sketched in Figure 6.7. A total of eight measurements can be done to further reduce effect of thermo-emf in contacts, with cyclic exchange of current and voltage probes [9].

The resistivity ρ can then be obtained from the following equation:

$$\exp\left(-\frac{\pi t V_{dc}^{ab}}{\rho I}\right) + \exp\left(-\frac{\pi t V_{ca}^{bd}}{\rho I}\right) = 1$$

where the upper and lower indices refer to the current and voltage contacts, respectively.

Solution of this equation can be represented in the following form:

$$\rho = \frac{\pi t}{\ln 2} \frac{(V_{dc}^{ab} + V_{ca}^{bd})}{2I} f(Q)$$

$$Q = \frac{V_{dc}^{ab}}{V_{ca}^{bd}}$$

The geometrical factors f can be obtained numerically from the following non-linear equation.

$$\frac{Q-1}{Q+1} = f \operatorname{arccosh}\left(\frac{\exp(\ln 2/f)}{2}\right)$$

If V_{dc}^{ab} and V_{ca}^{bd} are not within 10% of one another, the resistivity cannot be determined accurately.

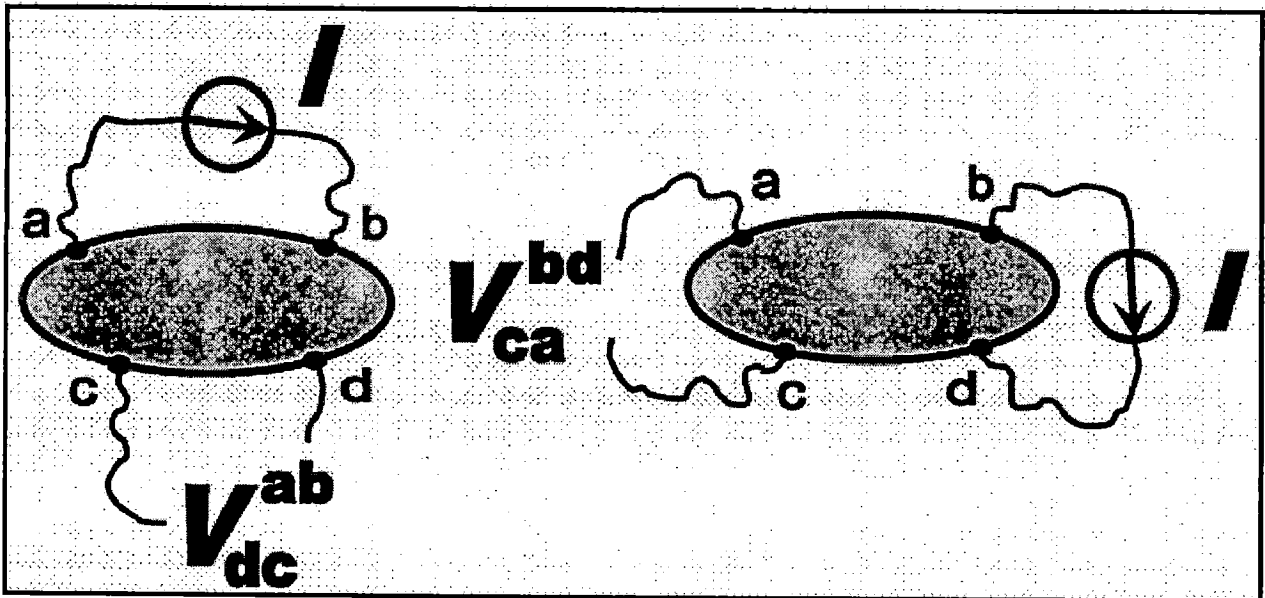


Figure 6.7-Van der Pauw Method of Resistivity Measurements of a Thin Film of Arbitrary Shape

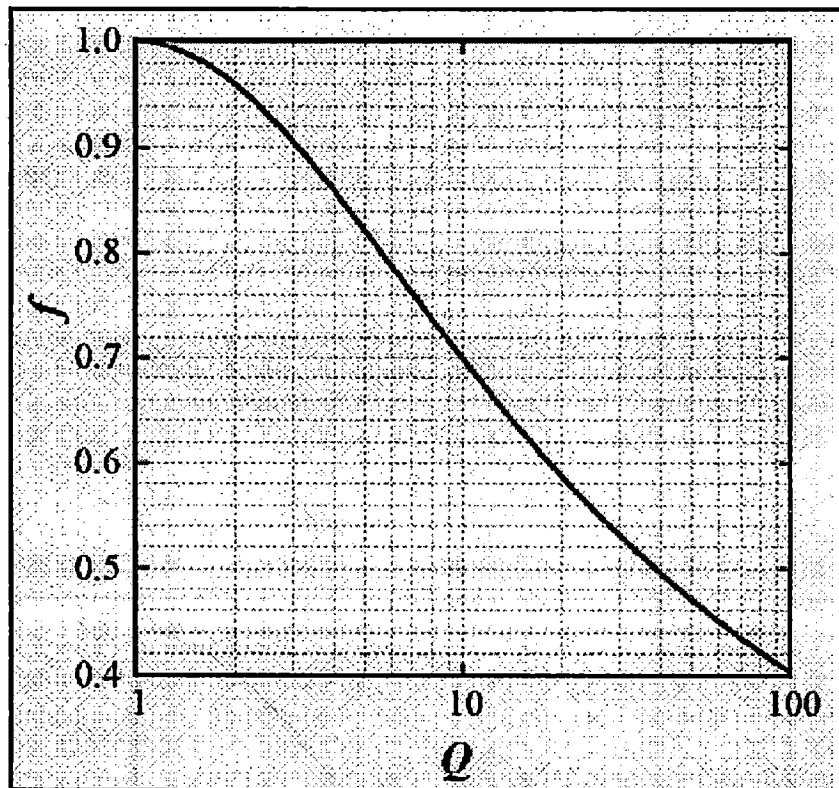


Figure 6.8-The Relationship between f and Q

6.2.3-Non-Patterned Thin Films: Large Wafers

Consider a large wafer covered by a conducting thin film of known thickness t . The goal is to infer resistivity from simple four-probe measurements. The practice is to use equally spaced probes placed in a row somewhere in the middle of the wafer.

Provided the separation s between contacts is much smaller (at least 4 times) than the radius of the wafer, and $t < s/2$.

$$R_{\square} = \frac{\pi}{\ln 2} \left(\frac{V}{I} \right) \approx 4.532 \left(\frac{V}{I} \right)$$

and

$$\rho = R_{\square} t$$

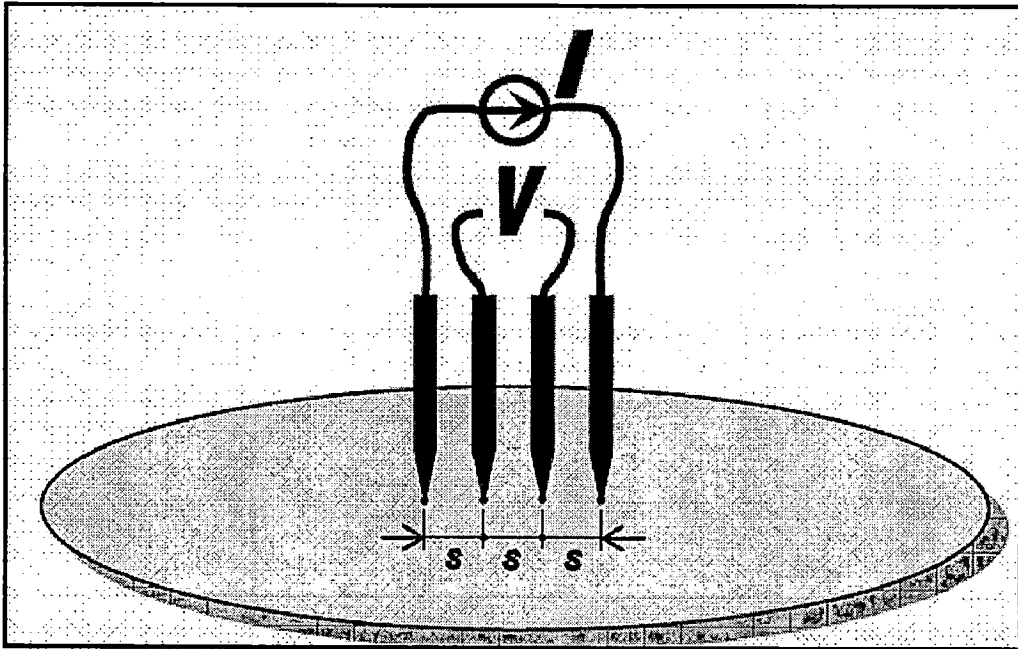


Figure 6.9-Sheet Resistance of a Large-Area Thin Film

One can also show that for the contacts placed in the corners of a square $s \times s$, the corresponding equation will be:

$$R_{\square} = \frac{\pi}{\ln \sqrt{2}} \frac{V}{I} \approx 9.065 \frac{V}{I}$$

For the cases then the wafer size is comparable with s , numerical methods should be used.

6.3-Low Impedance Measurements

Offset Voltages

- **Thermoelectric Voltage**

Low impedance measurements are associated with low-level voltage measurements. Those can be affected by various parasitic voltages among which thermoelectric voltages are the most common source of errors. These voltages are generated when different parts of a circuit made up of dissimilar metals are at different temperatures. The Seebeck coefficients S of various materials with respect to copper are summarized in the table below. To minimize thermoelectric voltages, circuits should be made using one and the same material, say, copper.

Reference Material	Material	S($\mu V / K$)
Cu	Cu	< 0.2
Cu	Ag	0.3
Cu	Au	0.3
Cu	Pb/Sn	1-3
Cu	Fe	11
Cu	CuO	1000

Connections must also be kept clean and free of oxides. As is seen from the table, clean Cu-Cu connections have $S < 0.2 \mu V/K$, while Cu-CuO connections (old oxidized contacts) may have S as high as $1000 \mu V/K$.

Minimizing temperature gradients also reduces thermoelectric voltages. All junctions should be in close proximity to each other and to have good thermal anchoring to a common, massive heat sink. Electrical insulators having high thermal conductivity must be used, such as anodized aluminum, beryllium oxide, or sapphire.

Measurements of sources at cryogenic temperatures pose special problems, since the source of signal may be near zero Kelvin while the voltmeter is at $\sim 300K$, i.e. there is a very large temperature gradient. Care must be taken as to make sure the wires coming from the room temperature parts of the measurements setup down to the cryogenic ones are uniform and are taken from one and the same batch (spoon).

To minimize the temperature gradients, equipment should reach thermal equilibrium in a constant ambient temperature. Equipment should also be kept away from direct sunlight, ventilation air-flows, and other sources of heat flow or moving air. Finally, wrapping connections in insulating foam also minimizes temperature gradients caused by air movement.

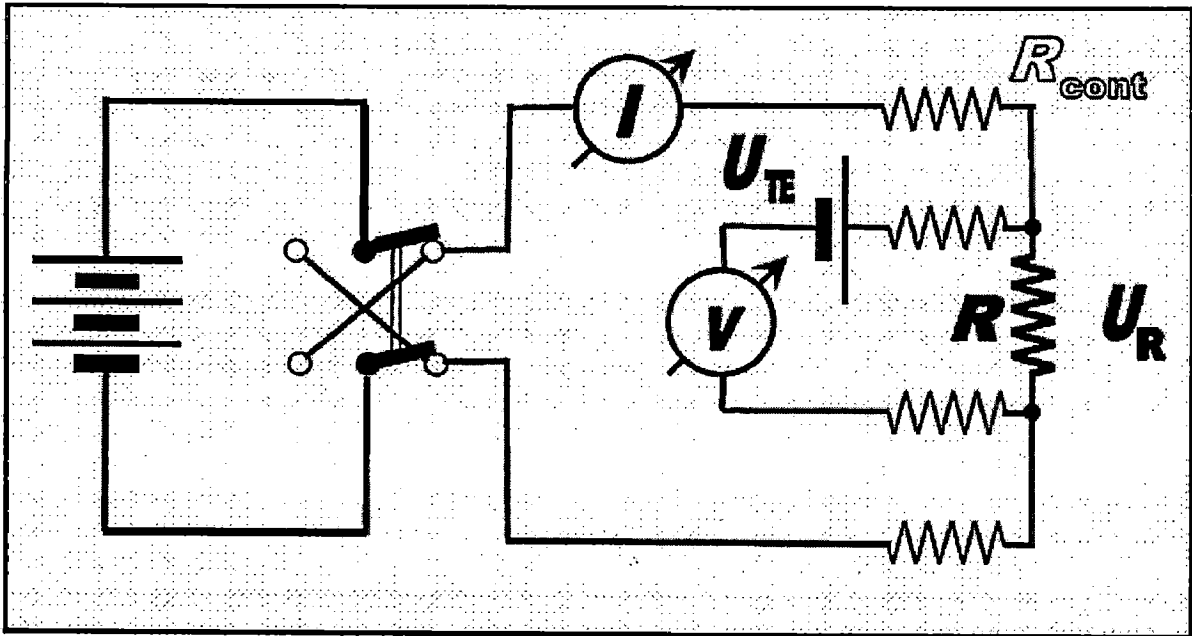


Figure 6.10-Bias Reversal Schematic

Reversing Bias

When measuring a small voltage, the error caused by thermoelectric voltages appearing in the contacts can be subtracted by taking two measurements with opposite direction of the bias current.

By subtracting one measured voltage from another, the parasitic thermoelectric voltage cancels out from the result, as is seen from the following equation:

$$\frac{V_+ - V_-}{2|I|} = \frac{(U_R + U_{TE}) - (-U_R + U_{TE})}{2|I|} = \frac{2U_R}{2|I|} = R$$

Notice that this technique cancels out the thermoelectric EMF term U_{TE} , which represents the algebraic sum of all thermoelectric EMFs in the circuit. If the measured voltage is the result of a current source flowing through the unknown resistance, then either the current-reversal method or the offset-compensated ohms method may be used to cancel the thermoelectric EMFs.

- **RFI/EMI**

RFI (Radio Frequency Interference) and EMI (Electromagnetic Interference) are terms for electromagnetic interference to measurements. These cover a wide range of frequencies. Figure 6.11 [9] shows a typical frequency (f) spectrum of these interference signals in comparison with other noise signals such as “ $1/f$ ” and thermal noise.

RFI or EMI can be caused by sources such as mobile phones, TV or radio signals, or come from computer screens or other digital equipment. It can also be caused by impulse sources, as in the

case of high-voltage arcing. In all cases, the sensitive measurement can be distorted if precautions are not properly taken. RFI/EMI signals may lead to a parasitic d.c. reading offset due to DC rectification or an input amplifier overload.

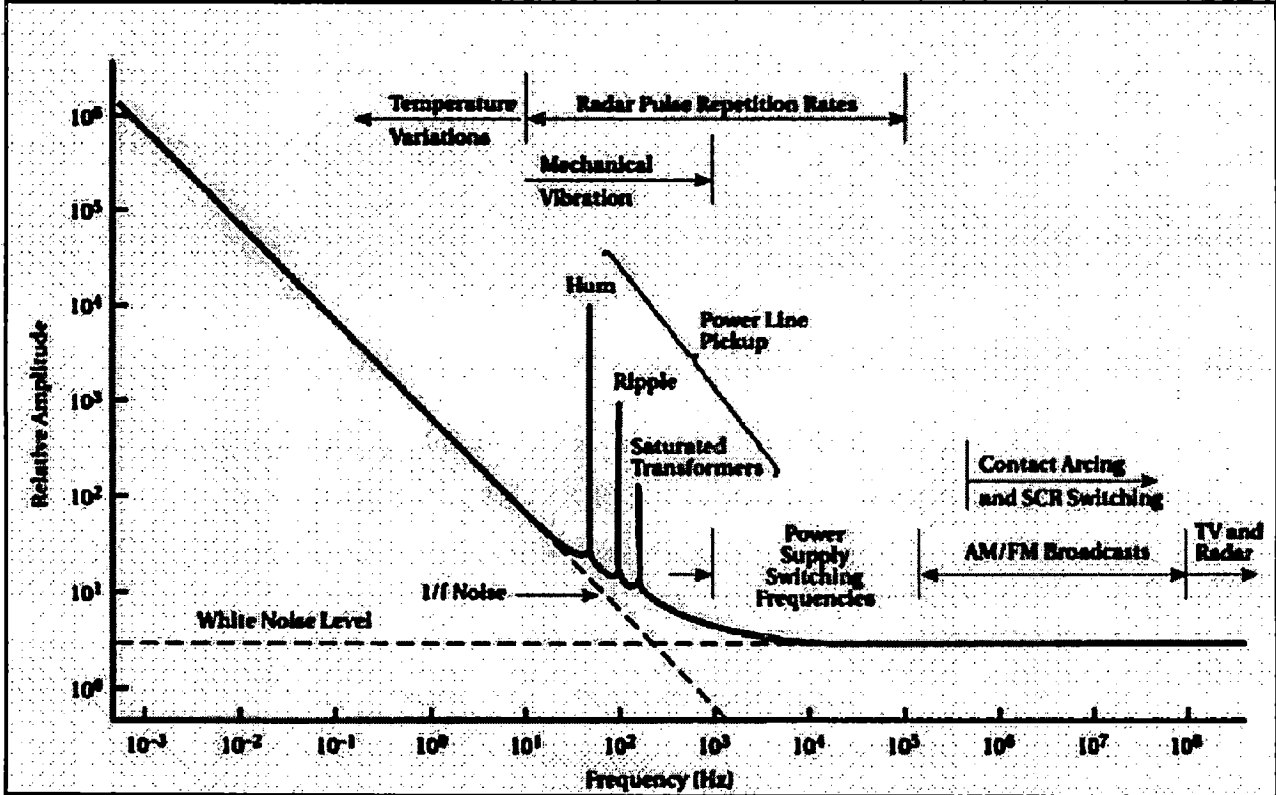


Figure 6.11-Frequency Spectrum of Interference Signal

The most obvious precaution is to keep all instruments, cables, and a sample far from the interference source. Shielding the leads and the sample (DUT) (Figure 6.12) [9] often reduces interference effects to an acceptable level. The shields should be connected to input LO, but if the RFI/EMI is also earth-ground based, connecting shields to LO may not reduce the interference. A special screening room may be necessary in some cases to attenuate the RFI signal sufficiently.

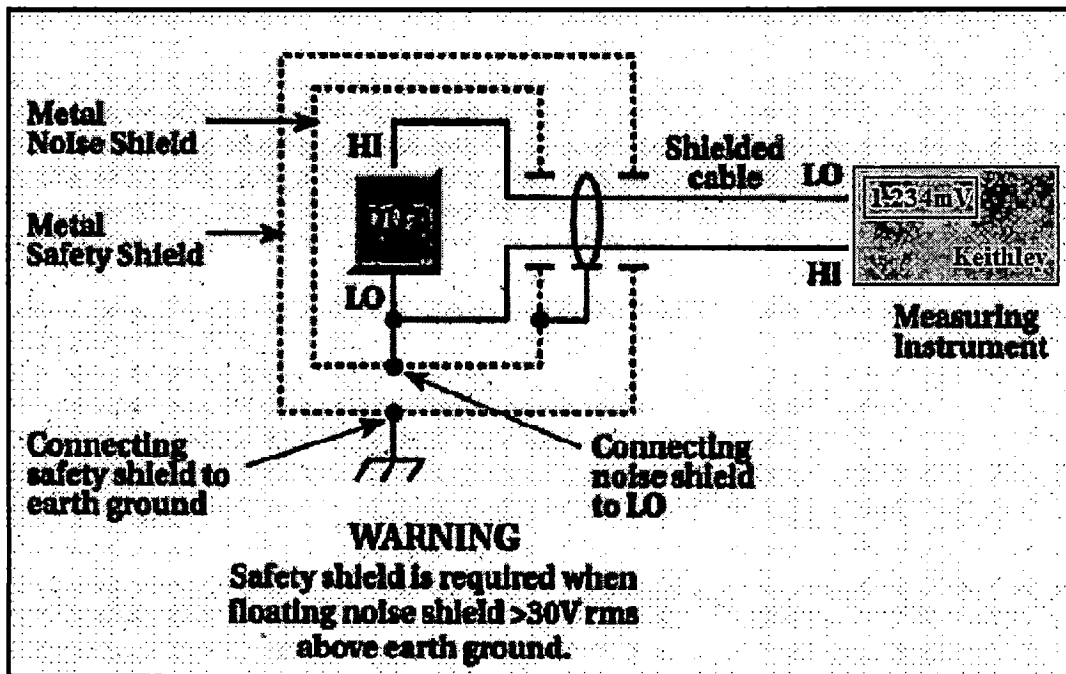


Figure 6.12-Shielding from RFI/EMI Signals

- **Johnson noise**

Johnson or thermal noise represents the ultimate resolution in an electrical measurement.

This noise is associated with the thermal energy of electrons at finite temperatures. All voltage sources have internal resistance, so all voltage sources develop Johnson noise. The noise voltage across a resistance can be calculated from the following equation:

$$V_j = \sqrt{4k_B TBR}$$

where V_j is the rms noise voltage, $k_B = 1.38 \times 10^{-23}$ J/K is the Boltzmann's constant, T is the absolute temperature of the resistor R (or an internal resistance of the signal source), B is the noise bandwidth, see Figure 6.13.

Johnson noise may be reduced by lowering the temperature of the source resistance and by decreasing the bandwidth of the measurement. Cooling the sample from room temperature (293K) to 77K decreases the voltage noise by approximately a factor of 2.

The bandwidth can be reduced by filtering and integration over multiple measurement cycles. The measurement time is then becoming longer.

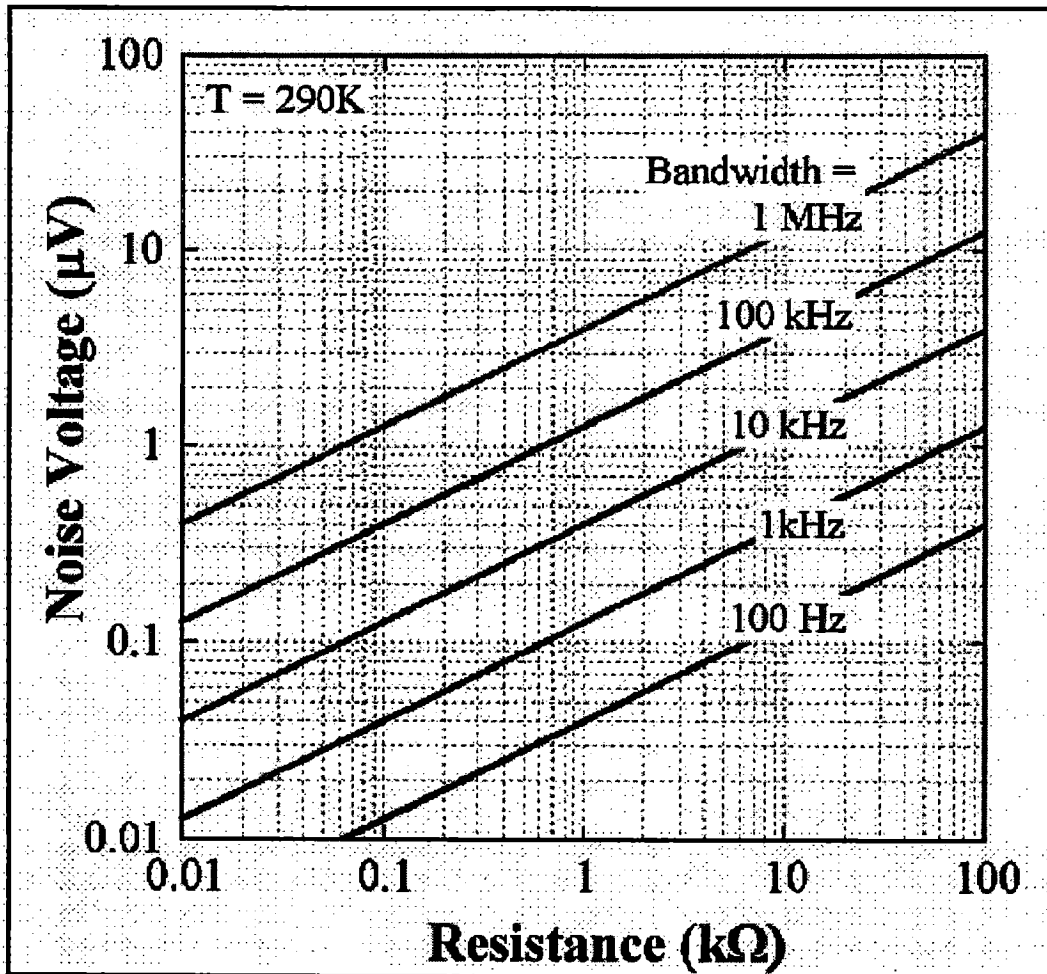


Figure 6.13-Thermal Noise Voltage as a Function of Resistance and Bandwidth

RESULTS AND DISCUSSIONS

No hardware for measurement is complete in itself till it gives satisfactory results. The photograph of hardware is shown in Figure 7.1 (a) and (b).

In order to match the read data (through hardware) with desired outcome it is needed to be calibrated. Figure 7.2 (a) shows the plot between applied voltage and measured voltage of ADC when voltage reference is 1.2V (black line shows error, increasing linearly with applied voltage). But when voltage reference is adjusted to 1.0002V (Figure 7.2 (b)) error reduces to zero. Figure 7.3 and Figure 7.4 show the plot between written data (in decimal) Vs measured data (in Volts) of DAC1 and DAC2 respectively. It is to be noted here the DAC1 is in bipolar configuration thus it has a capability of measuring negative voltage as well as positive voltage while DAC2 is in unipolar configuration it can measure positive voltage only. It can also be analyzed from the Figure 7.3 and Figure 7.4 that DAC2 can measure voltage up to +3.6V thus it can be employed for higher voltages.

For testing the proper working of the hardware and data acquisition programs, V-I characteristic of resistor and diode have been plotted. Schematic circuit diagram for I-V characteristic of resistor drawn in Express PCB software is shown in Figure 7.5 (a). Figure 7.5 (b) shows the actual circuit in bread-board. The I-V characteristic of resistor is coming almost linear (Figure 7.6). Schematic circuit diagram and the actual circuit for I-V characteristic of diode is shown in Figure 7.7 (a) and Figure 7.7 (b) respectively. Figure 7.8 shows the desirable I-V characteristic of diode with knee point equal to 0.7V

And finally the resistivity of thin film of gold has been determined. Figure 7.9 (a) shows the schematic circuit diagram for measuring resistivity of gold thin film, Figure 7.9 (b) shows the thin film with gold electrode connections. And the actual circuit for measuring resistivity of thin film in bread-board is shown in Figure 7.9 (c). The resistivity of thin film of gold is coming out to be $8.5714 \times 10^{-5} \Omega\text{cm}$

Thus by considering all these aspects it can be said that hardware is showing proper result and programs are working in accord.

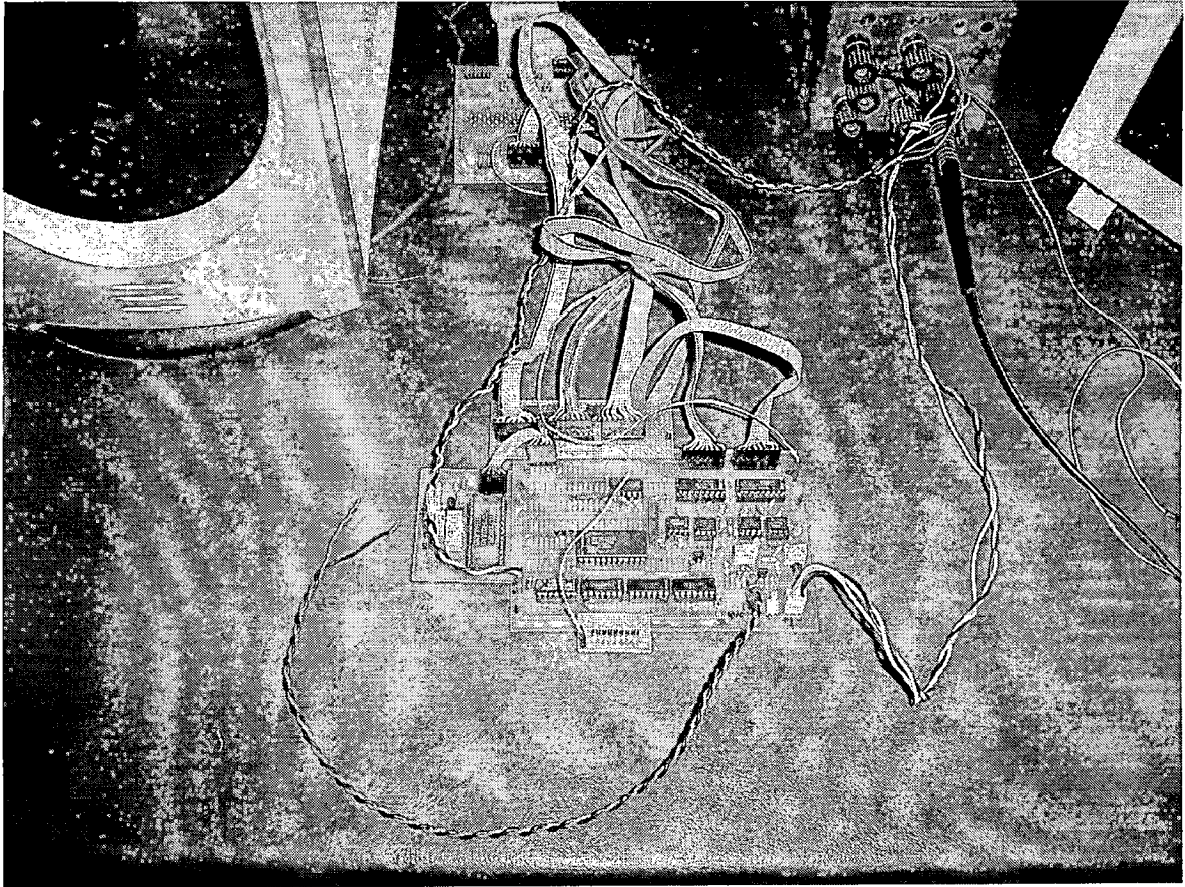


Figure 7.1 (a)-Hardware (Picture-1)

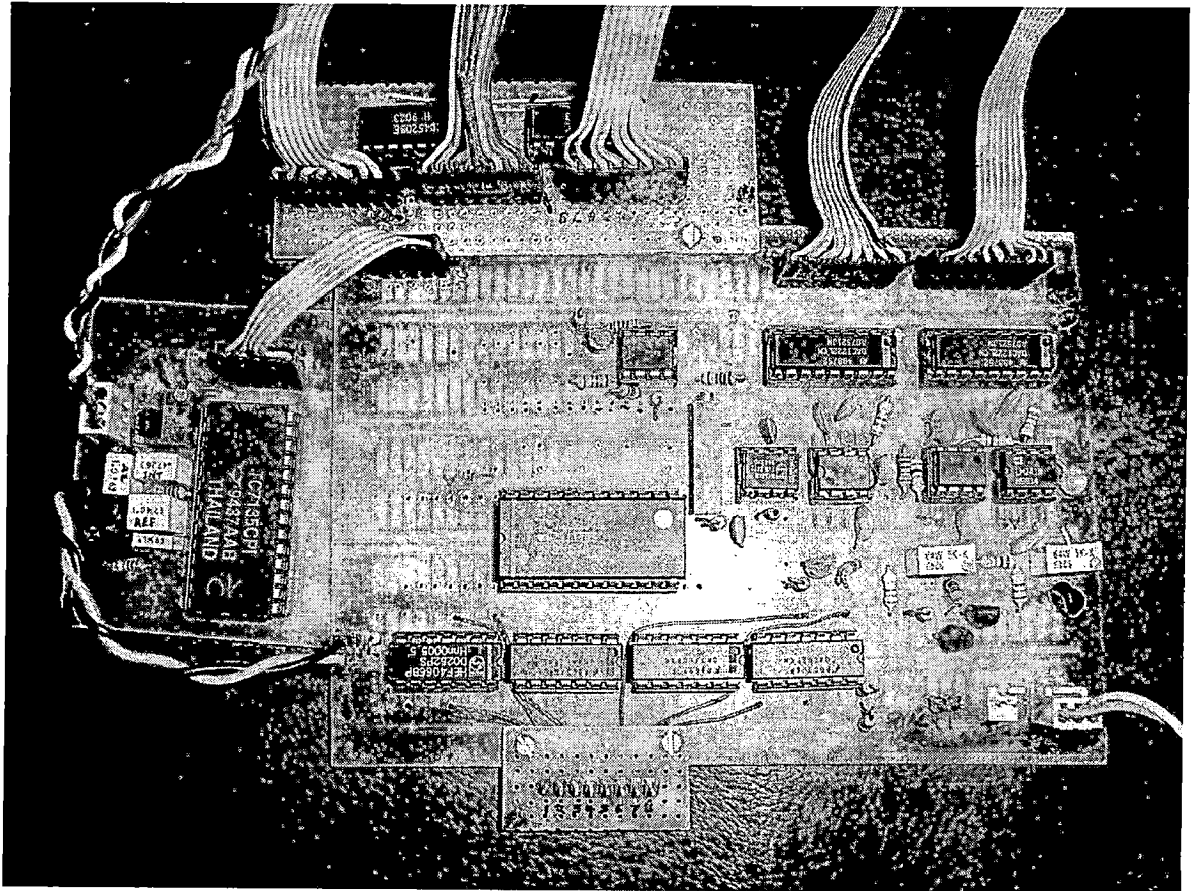


Figure 7.1 (b)-Hardware (Picture-2)

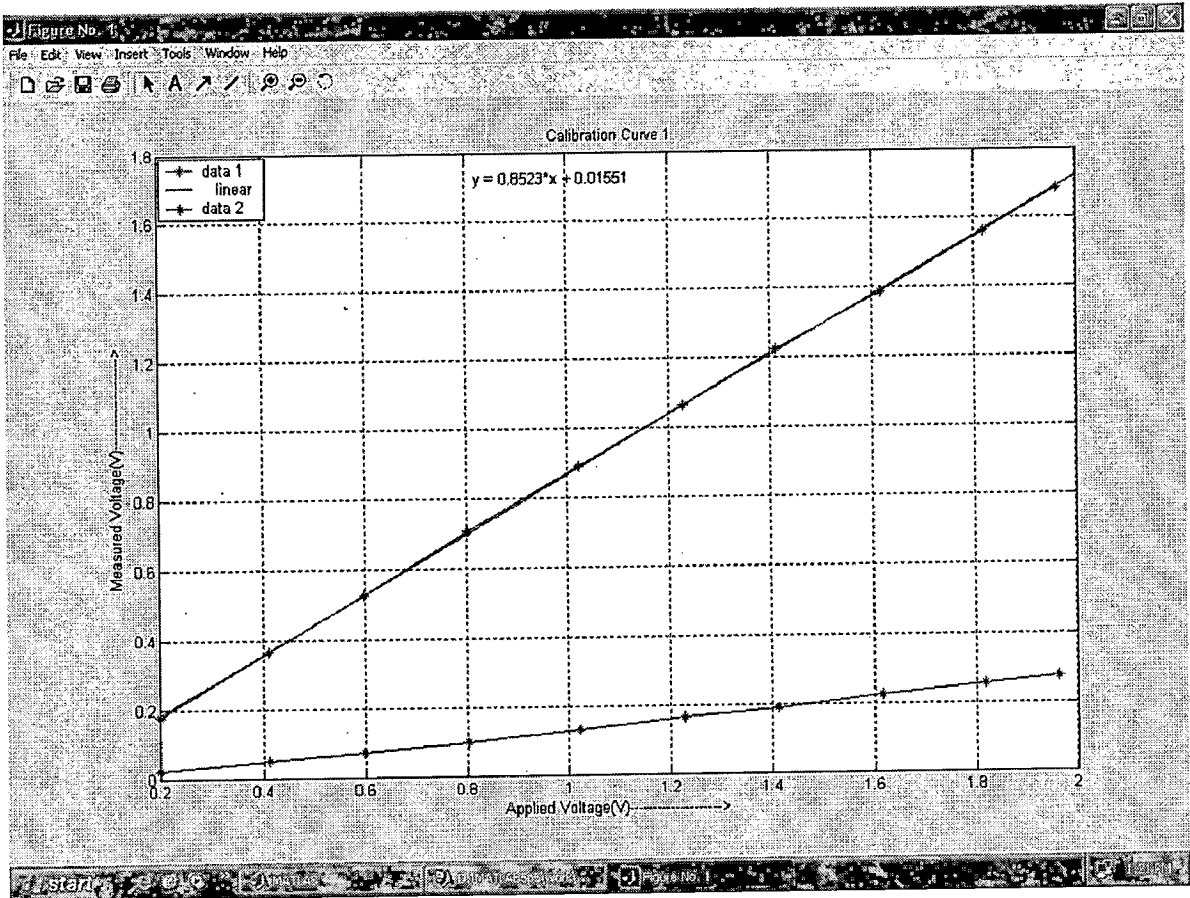


Figure 7.2 (a)-Applied Voltage Vs Measured Voltage of ADC (ref=1.2V)

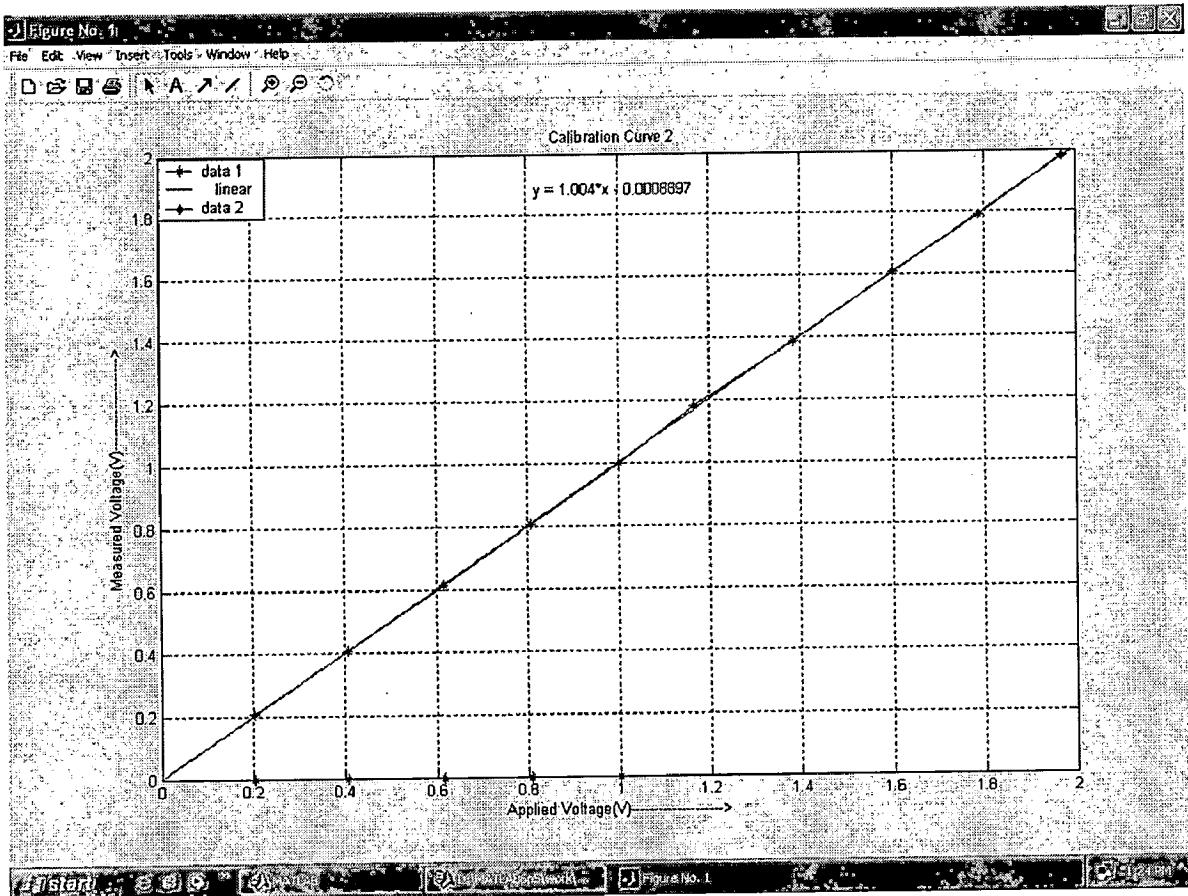


Figure 7.2 (b)-Applied Voltage Vs Measured Voltage of ADC (ref=1.002V)

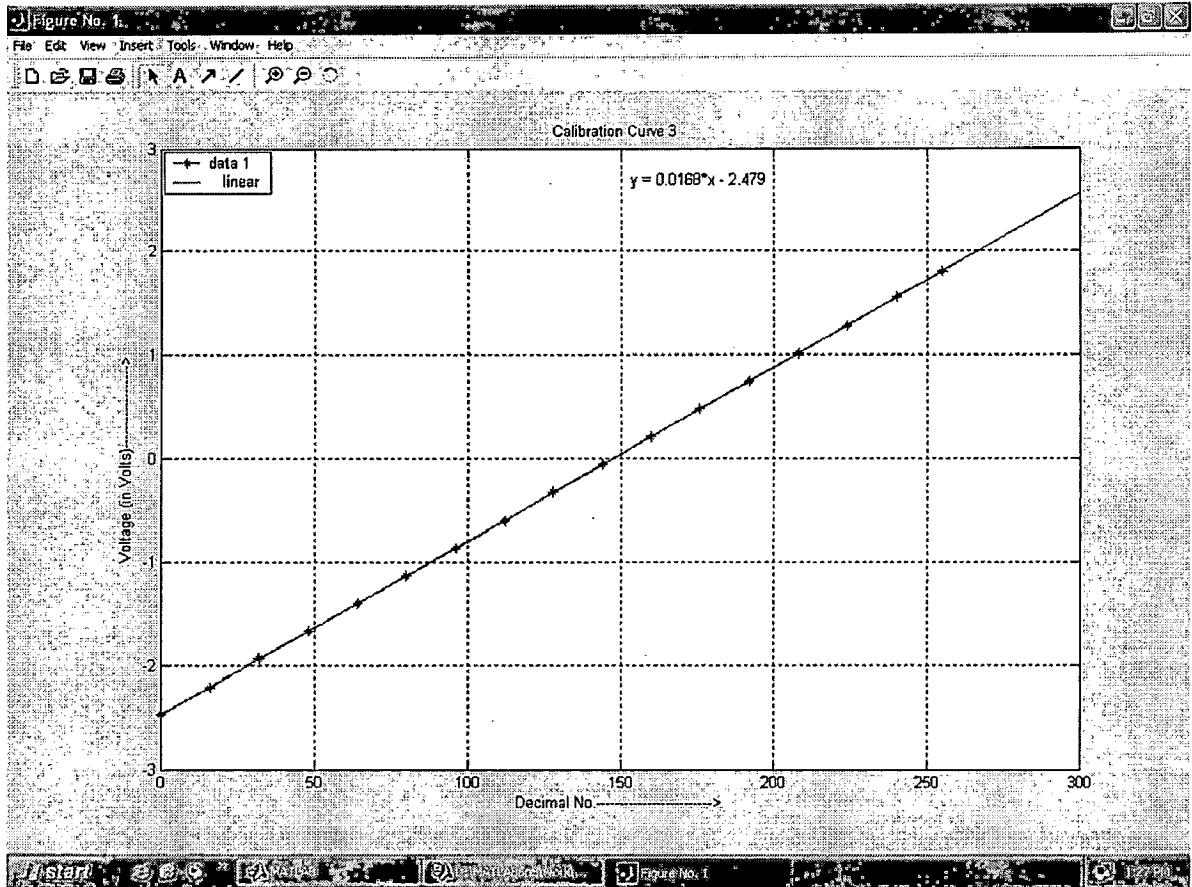


Figure 7.3-Written Data (in decimal) Vs Measured Data (in Volts) of DAC1

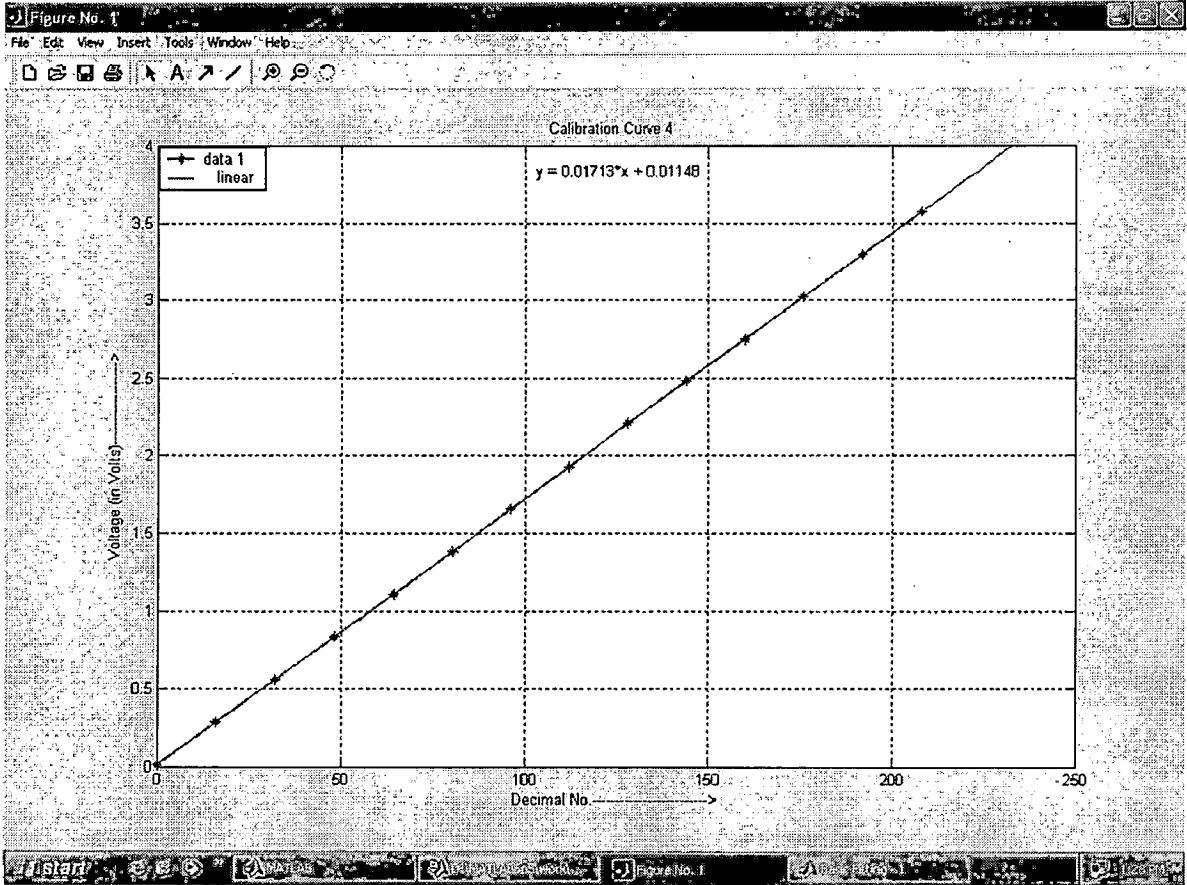


Figure 7.4-Written Data (in decimal) Vs Measured Data (in Volts) of DAC2

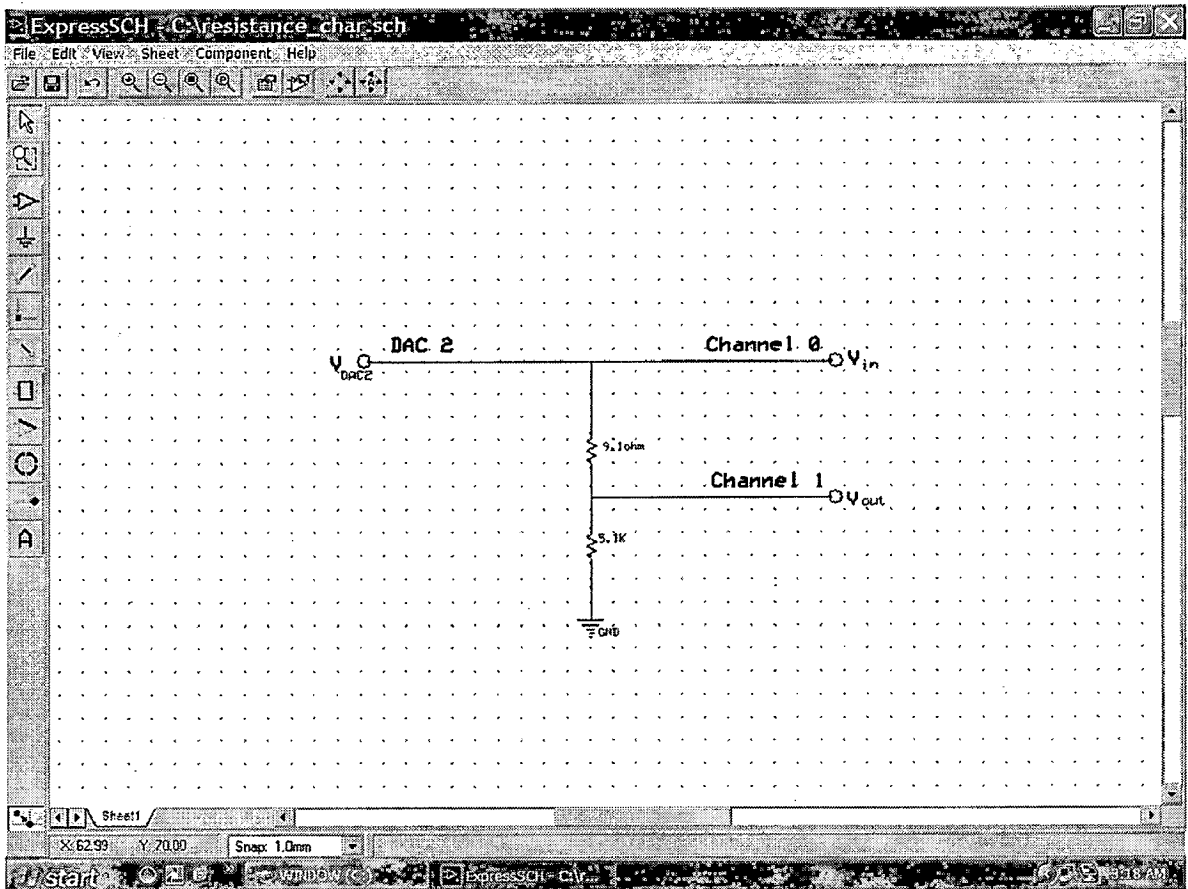


Figure 7.5 (a)-Schematic Circuit Diagram for I-V Characteristic of Resistor

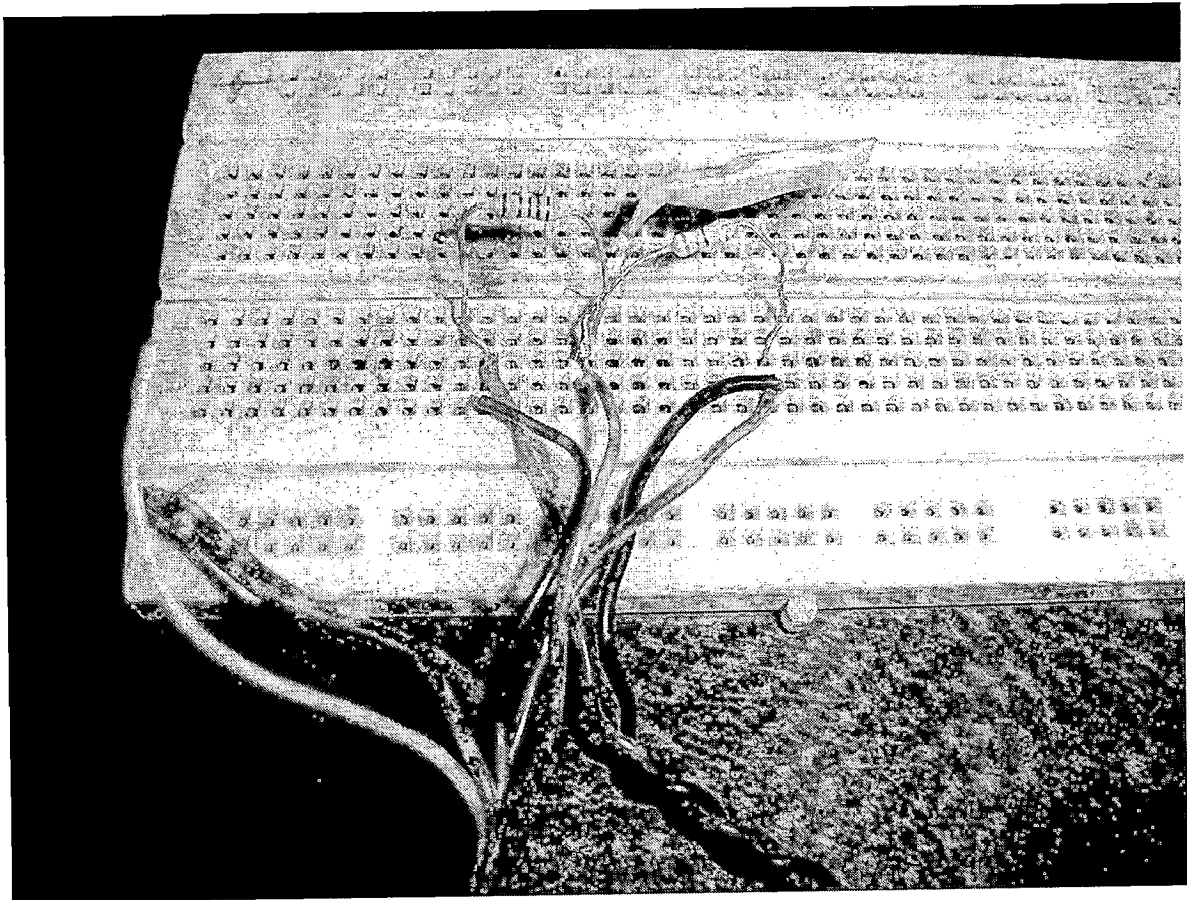


Figure 7.5 (b)-Circuit for I-V Characteristic of Resistor

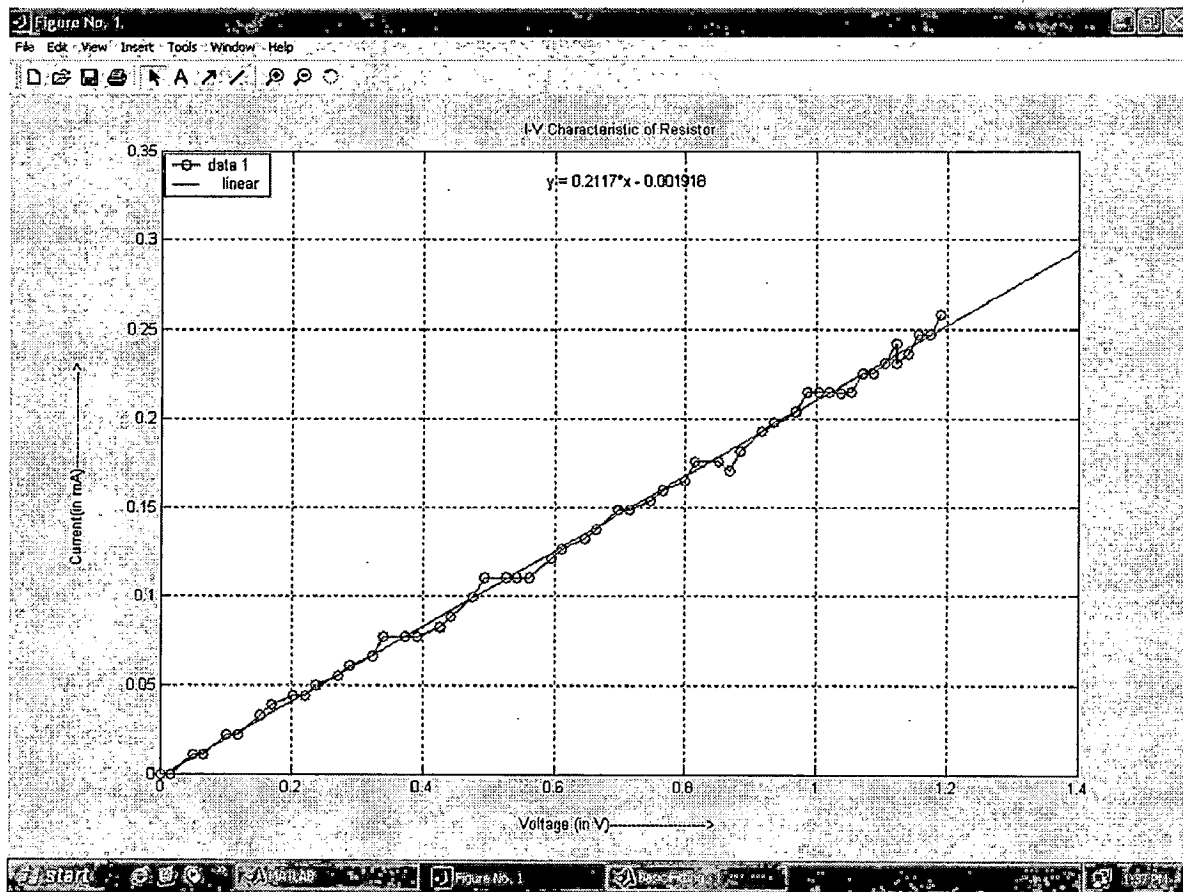


Figure 7.6- I-V Characteristic of Resistor

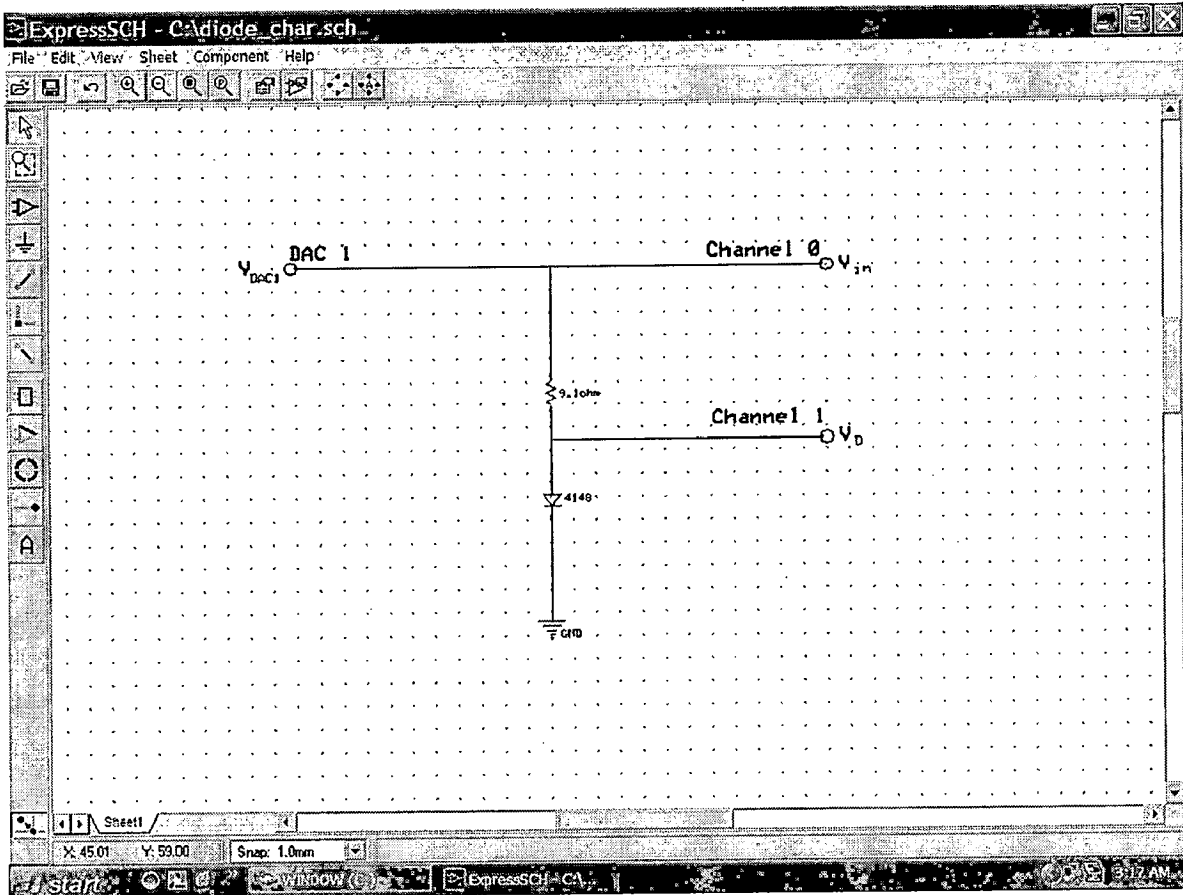


Figure 7.7 (a)-Schematic Circuit Diagram for I-V Characteristic of Diode

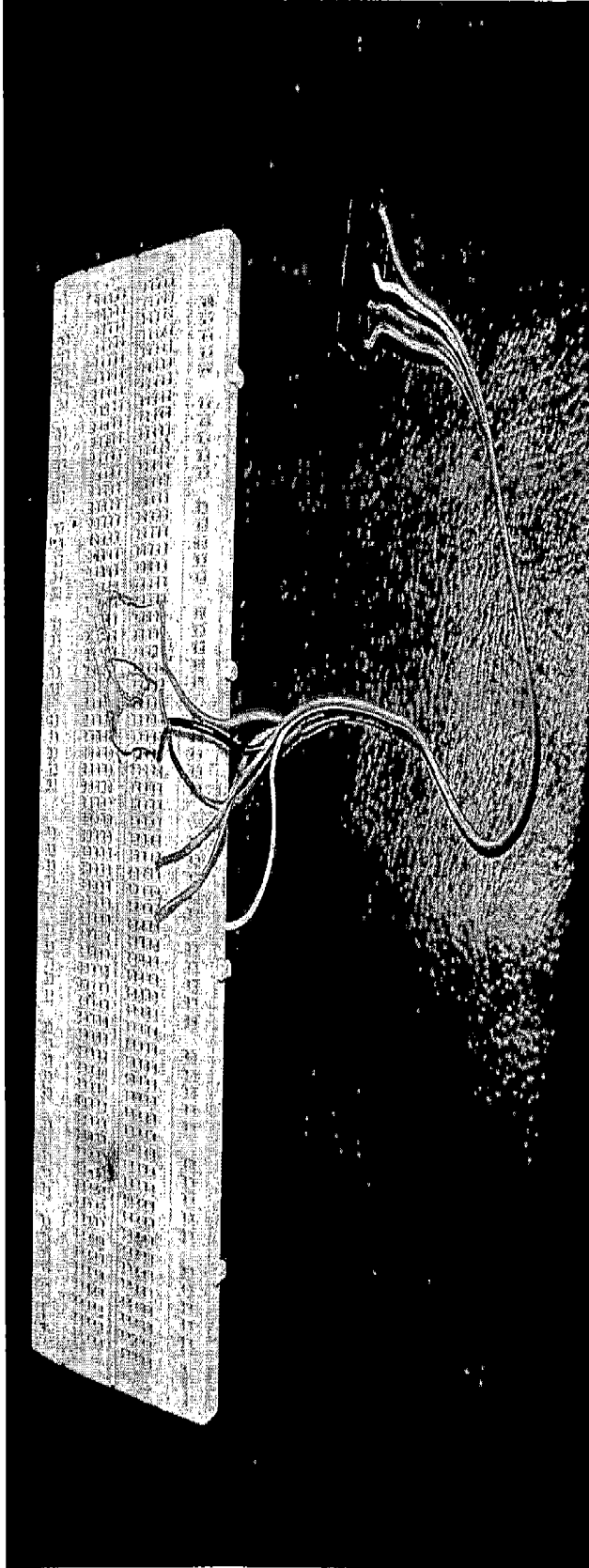


Figure 7.7 (b)—Circuit for I-V Characteristic of Diode

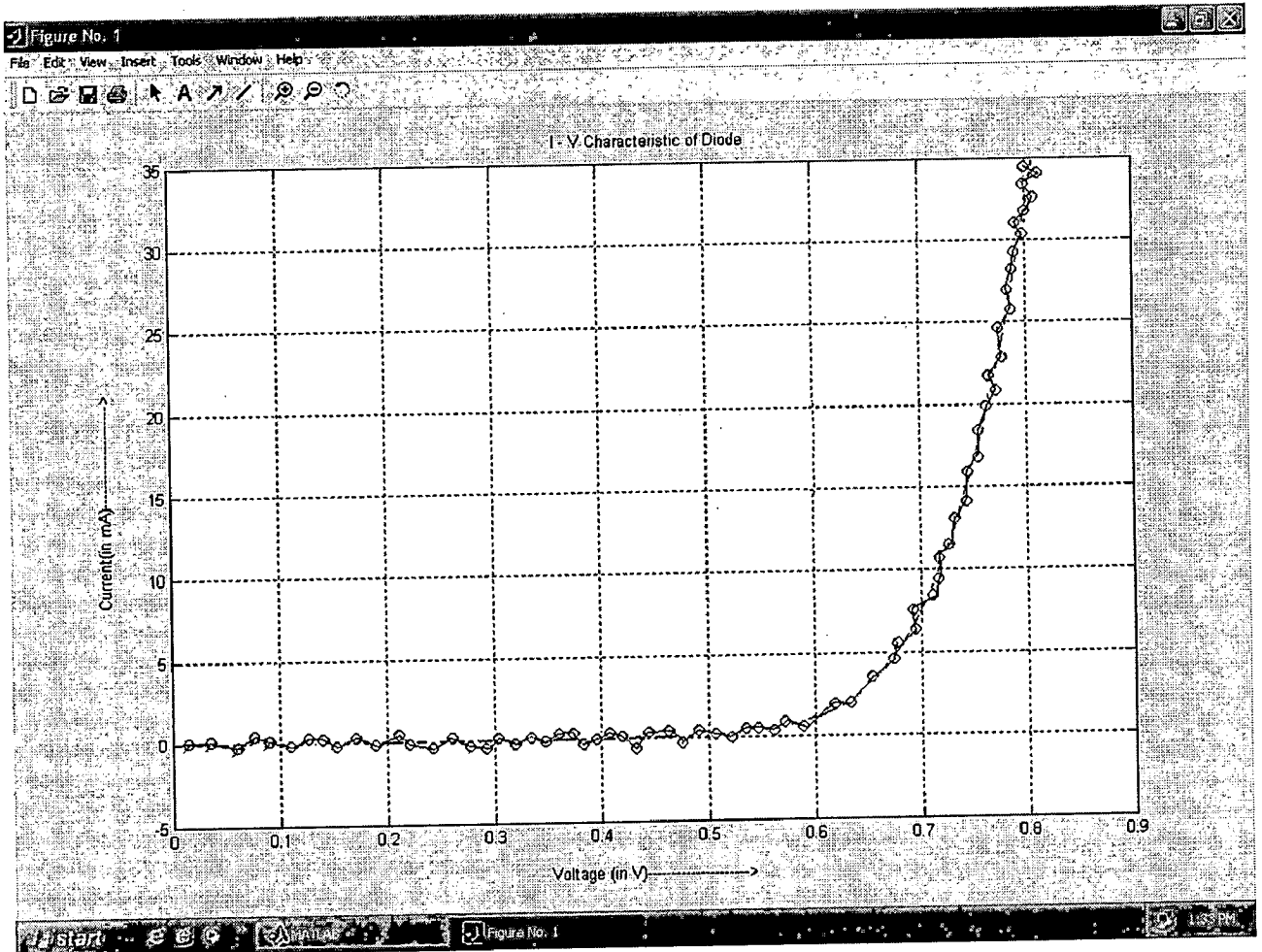


Figure 7.8- I-V Characteristic of Diode

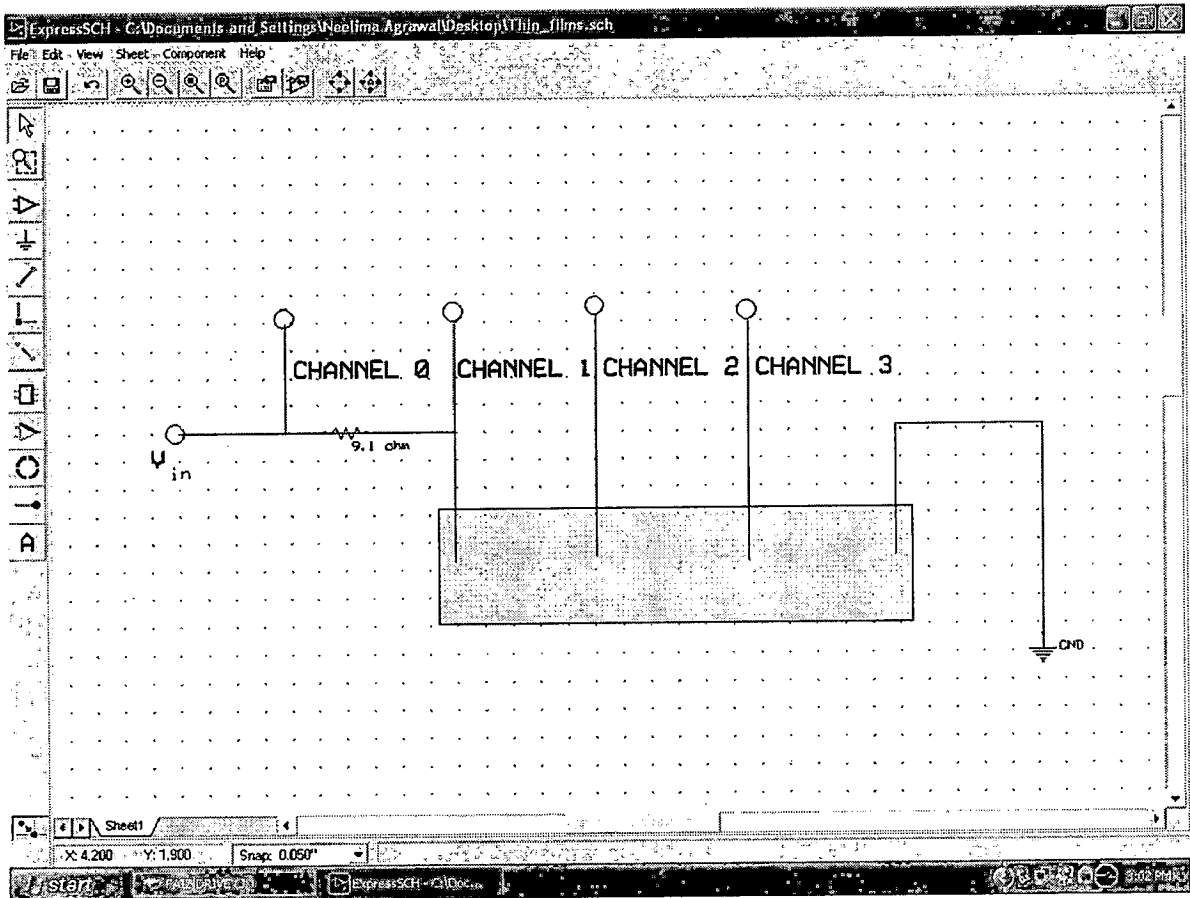


Figure 7.9 (a)-Schematic Circuit Diagram for Measuring Resistivity of Thin Film

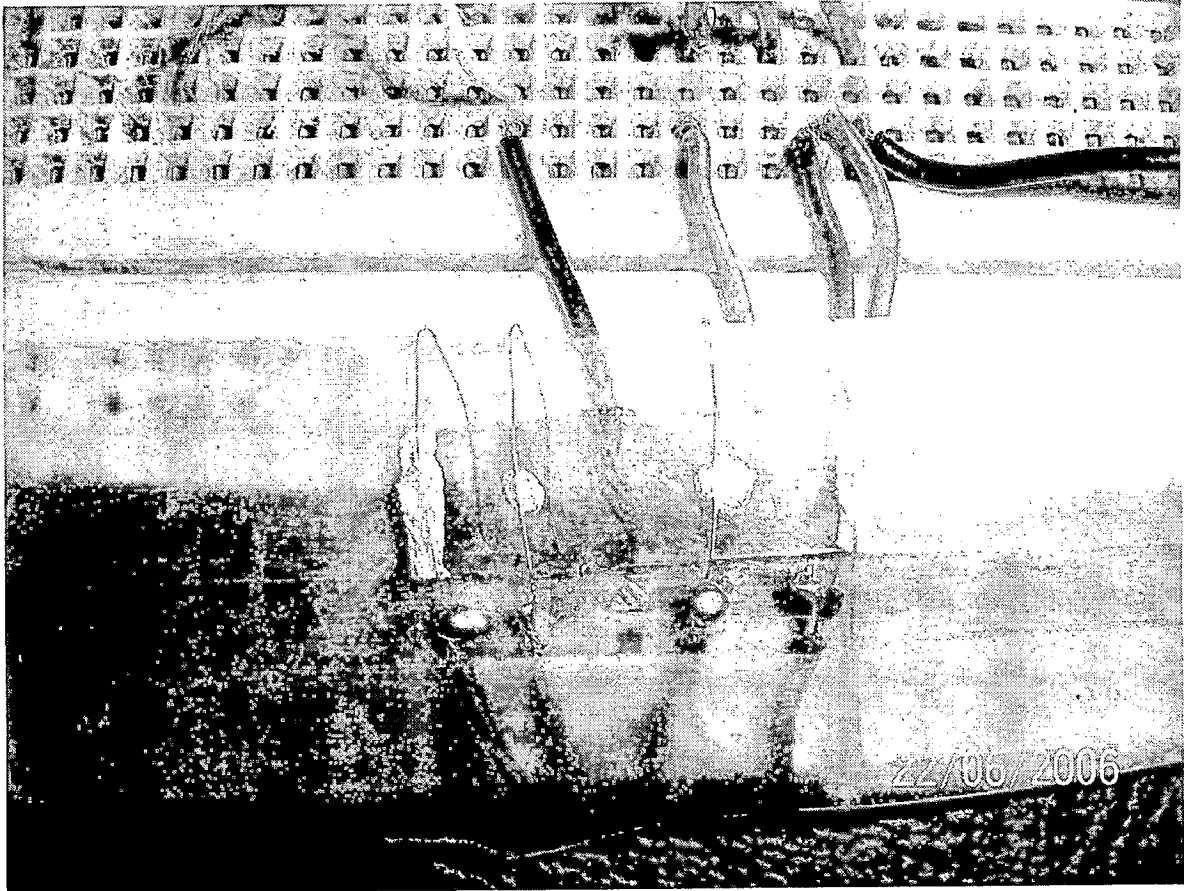


Figure 7.9 (b)-Thin Film with Electrode Connections

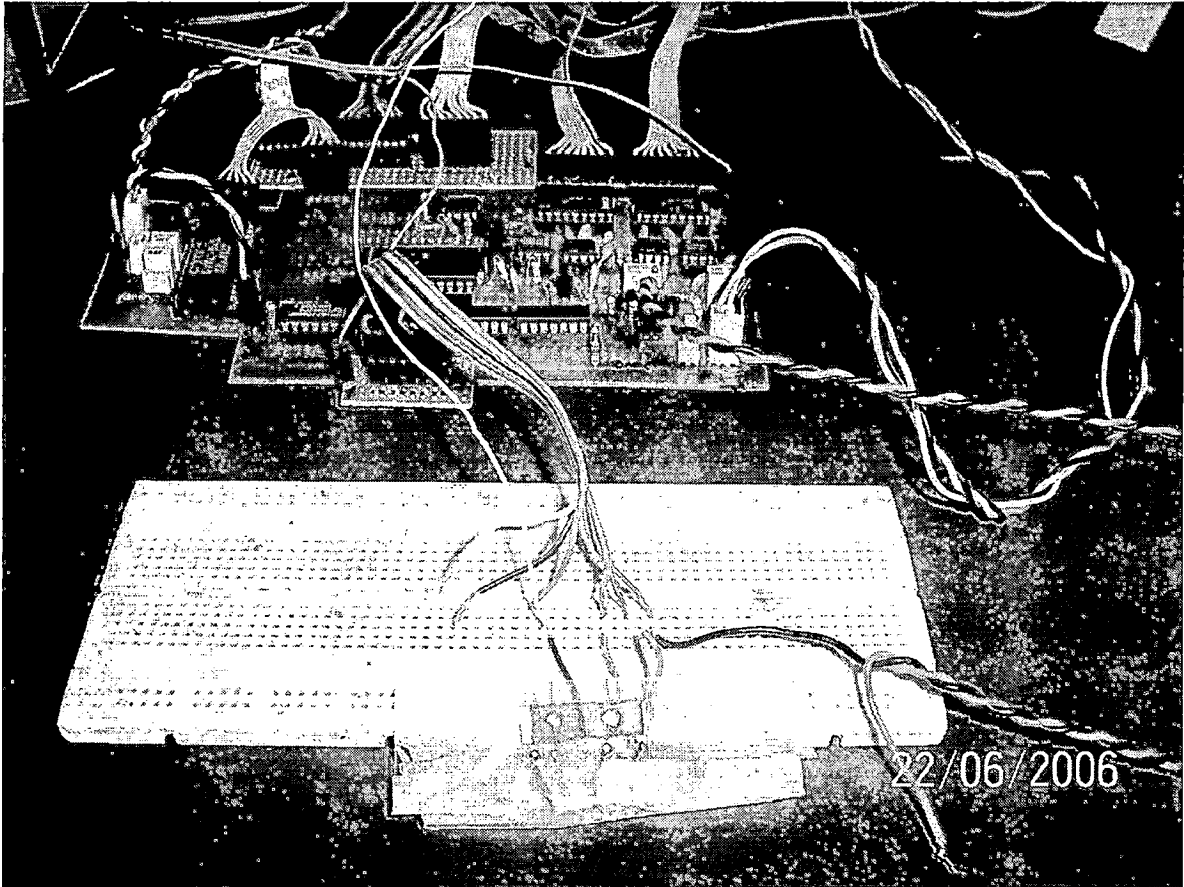


Figure 7.9 (c)-Circuit for Measuring Resistivity of Thin Film

CONCLUSION

Nowadays, when more companies go into the area of measurement by choosing hardware and software solutions instead of old manual methods, there is a rising need for more efficient measurement methods for engineers.

The easy availability of personal computers has revolutionized the area of automated measurement and control of experiments.

Computer-aided measurements need interfacing hardware which has been presented in this dissertation, and the subject of data acquisition in the measurement area has been described.

From a technical point of view, for automated measurement, knowledge in three main areas is needed: component behaviors, circuit performance, and programming language.

To facilitate the learning of component behaviors and circuit performance in this dissertation, there is a methodology presented on how to develop comprehensible hardware for automated measurement and to facilitate the understanding of programming language. Structured programs for data acquisition are explained.

Finally, the results and graphs have been presented and explained in order to evince the proper working of the hardware and utility of the data acquisition programs.

Moreover, this hardware can also be utilized for other kinds of measurement, such as Hall Effect, measurement of frequency, etc., as far as measurement is slow.

REFERENCES

- [1] AD7520/AD7530/AD7521/ AD7531 10-Bit/12-Bit Multiplying D/A Converters, Datasheet, Intersil Semiconductor, August 1997.
- [2] Chapman Stephen J., MATLAB Programming for Engineers, Third edition, Thompson Learning, Inc., 2004.
- [3] CD4066BM/CD4066BC Quad Bilateral Switch, Datasheet, National Semiconductor, June 1992.
- [4] CD4518B/CD4520B Dual Synchronous Up-Counter, Datasheet, Texas Instrument, March 2004.
- [5] DM54LS154/DM74LS154 4-Line to 16-Line Decoders/Demultiplexers, Datasheet, National Semiconductor, May 1989.
- [6] DM74LS04 Hex Inverting Gates, Datasheet, Fairchild Semiconductor, March 2000.
- [7] DAC1020/DAC1021/DAC1022 10-Bit Binary Multiplying D/A Converter DAC1220/DAC1222 12-Bit Binary Multiplying D/A Converter Datasheet, National Semiconductor, May 1996.
- [8] ICL7135, Datasheet, Intersil Semiconductor, File No. 3093.2, December 2000.
- [9] Keithley Knowledge Tutorial Book., *Low Level Measurements, 5th edition.*
<http://www.keithley.com>.
- [10] L.J. van der Pauw, *Philips Tech. Rev.*, 1959.
- [11] LM741 Operational Amplifier, Datasheet, National Semiconductor, May 1998.
- [12] LM136-2.5/LM236-2.5/LM336-2.5V Reference Diode, Datasheet, National Semiconductor, May 1998.

[13]Pratap Rudra, Getting Started with MATLAB 5, Third Edition, Oxford University Press, Inc., 2001.

[14]Reducing Resistance Measurement Uncertainty: DC Current Reversals vs. Classic Offset Compensation: (<http://www.keithley.com/servlet/Data?id=4644>).

[15]Timer 555, Datasheet, Philips Semiconductor Military Linear Products, July 1991.

[16]User's Guide Version 2, Data Acquisition Toolbox, The Math Works Inc., 3 Apple Hill Drive Natick, MA, June 2001.

[17]User's Manual, PCI-1751 48-bit Digital Input/Output Card for PCI Bus, 1st Edition, Taiwan, August 1998.

APPENDIX - A

function [] = defA()

%this function defines digital input output object and addlines to it at port a0, b0, c0, a1, b1 and

%c1

global dio a0 b0 c0 a1 b1 c1;

dio = **digitalio** ('advantech', 0); %declares and defines an digital I/O object

a0 = **addline** (dio, 0:7, 0,'in'); %adds lines to object at port 0 and declares it as input port

b0 = **addline** (dio, 0:7, 1,'in'); %adds lines to object at port 1 and declares it as input port

c0 = **addline** (dio, 0:7, 2,'in'); %adds lines to object at port 2 and declares it as input port

a1 = **addline** (dio, 0:7, 3,'out'); %adds lines to object at port 3 and declares it as output port

b1 = **addline** (dio, 0:7, 4,'out'); %adds lines to object at port 4 and declares it as output port

c1 = **addline** (dio, 0:7, 5,'out'); %adds lines to object at port 5 and declares it as output port

disp ('SUCESS');

disp ('WAIT.....While the Data is Being Measured');

function [value] = rdchan(x)

%this function reads the selected channel from the input port a0 and b0.

% Syntax: [value] = rdchan(x), where x is any decimal number between 0 and 7

global a0 b0 c0 a1 b1 c1 %declares all the ports global.

RST = 16; %defines RESET

R = 8; %defines RUN

H = 0; %defines Hold

%the order of the pin at the output port c1 is: X X X RST R/H C B A(lsb)

% 16 8 4 2 1

%c1 is a control port which decides which channel is to be selected

putvalue (c1, x); %writing value to the port c1 which has been configured as output port

for i=1:4,

putvalue (c1, RST + x); %resets all the pins of port c1

```

putvalue (c1, 0 + x);      %holds the value
putvalue (c1, R + x);    %makes only run pin high
putvalue (c1, H + x);    %holds the value
pause (.5);              %pause is necessary to get the accurate values, as ADC is slow.
D1=getvalue (a0);        %reads values from port a0
D2=getvalue (b0);        %reads values from port b0
if D2(8) ~ = 1;         %if MSB = 0, the value will be negative
    D2 (8) = 0;
    D (i) = -1 * (256 * binvec2dec (D2) + binvec2dec (D1) - 10001);
else
    D2 (8) = 0;
    D (i) = (256 * binvec2dec (D2) + binvec2dec (D1) ) - 10001;
end
end
val = mean (D) / 10000;
switch val,
case {0,1,2,3,4,5,6},
    value = val;
case 7,
    value = 2 * val;
otherwise
    disp ('WRONG CHOICE');
end

```

```

function []= wrdac1(Volts);
%this function writes the given values to the output port a1.
%at a1 we have connected DAC1
% the general Syntax of this function is: wrdac1 (Volts)
global a1;
M = ceil ((Volts+2.4791) / 0.016805);    %after calibration
putvalue (a1, M);                       %writing the value M to the output port a1

```

```

function [M]= wrdac2(Volts);
%this function writes the given values to the output port b1.
%at b1 we have connected DAC1
% the general Syntax of this function is: wrdac2 (Volts)
global b1
M=ceil((Volts-.0115)/.0171);      %after calibration
if M <=210
    putvalue (b1, M);      %writing the value M to the output port b1
else
    M = [ ];
    Disp (' "ERROR" INPUT TOO HIGH');
end

```

```

%This programs test Applied voltage Vs Measured voltage of ADC
%Calibration Curve 1 of ADC
%ref=1.2V
%y=0.852*x-0.0155;
%gain=0.852;
%offset=-0.0155;
clc;      %clears the command window.
clear all;      %clears the workspace.
inp_voltage = [0.200 0.413 0.598 0.803 1.021 1.228 1.412 1.615 1.818 1.962]; %input voltage
mes_value = [11796 13660 15269 17057 18896 20638 22224 23891 25626 26855]; %measured
z = (mes_value - 10001);
mes_voltage = z / 10000;
error = inp_voltage - mes_voltage;
fprintf ('The value on Applied Vtg and Measured Vtg and their Difference is shown');
k=[inp_voltage' mes_voltage' error']
xlswrite ('CalibrationCurve1ADC',k);%writes measured data in xl
plot (inp_voltage, mes_voltage, '*-', inp_voltage, error, 'k*-*');

```

```

grid on;
title ('Calibration Curve 1');
xlabel ('Applied Voltage(V)----->');
ylabel ('Measured Voltage(V)----->');

```

```

%This programs test Applied voltage Vs measured voltage of ADC

```

```

%Calibration Curve 2 of ADC

```

```

%ref=1.002V

```

```

%y=1.004*x-0.0008;

```

```

%gain=1.004;

```

```

%offset=-0.0008

```

```

clc; %clears the command window.

```

```

clear all; %clears the workspace.

```

```

inp_voltage = [0.204 0.406 0.615 0.808 1.001 1.168 1.383 1.602 1.786 1.972]; %Applied voltage

```

```

mes_value = [12050 14066 16154 18088 20017 21801 23870 26074 27920 29791]; %measured

```

```

z = (mes_value - 10001);

```

```

mes_voltage = z / 10000;

```

```

error = inp_voltage - mes_voltage;

```

```

fprintf ('The value on Applied Vtg and Measured Vtg and their Difference is shown');

```

```

k=[inp_voltage' mes_voltage' error']

```

```

xlswrite ('CalibrationCurve2ADC',k); %writes measured data in xl

```

```

plot (inp_voltage, mes_voltage,'*- ', inp_voltage, error, 'k*- ');

```

```

grid on;

```

```

axis ([0 2.0 0 2.0])

```

```

title ('Calibration Curve 2');

```

```

xlabel ('Applied Voltage(V)----->');

```

```

ylabel ('Measured Voltage(V)----->');

```

```

%This programs tests writing value to DAC1 and Measured Vtg
%Calibration curve 3 of DAC1
clc;                                %clears the command window.
clear all;                          %clears the workspace.
d1 = [0, 16, 32, 48, 64, 80, 96, 112, 128, 144, 160, 176, 192, 208, 224, 240, 255];
AO1 = [-2.4746, -2.2115, -1.9424, -1.6735, -1.4045, -1.1355, -0.8667, -0.598, -0.3282, -0.0589,
0.2098, 0.4787, 0.7478, 1.0168, 1.2855, 1.5542, 1.8059];
K = [d1' AO1']
xlswrite ('CalibrationCurveDAC1',k);    %writes measured data in xl
plot (d1,AO1,'*-');
grid on;
title ('Calibration Curve 3');
xlabel ('Decimal No.----->');
ylabel ('Voltage (in Volts)----->');

```

```

%This programs tests writing value to DAC2 and Measured Vtg
%Calibration curve 4 of DAC2
clc;                                %clears the command window.
clear all;                          %clears the workspace.
d2 = [0, 16, 32, 48, 64, 80, 96, 112, 128, 144, 160, 176, 192, 208];
AO2 = [0.0115, 0.2861, 0.5599, 0.8339, 1.1071, 1.3815, 1.6551, 1.9290, 2.2042, 2.4788, 2.7526,
3.0266, 3.2999, 3.5742];
k = [d2' AO2']
xlswrite ('CalibrationCurveDAC2',k);%writes measured data in xl
plot (d2,AO2,'*-');
grid on;
title ('Calibration Curve 4');
xlabel ('Decimal No.----->');
ylabel ('Voltage (in Volts)----->');

```

```

%Script File for getting I-V characteristic of Resistor
clc;                                %clears the command window.
clear all;                            %clears the workspace
defA;                                  %defines digital input output object and add lines to it
v = 0 : 0.025 : 1.2;
R = 9.1;                                %in ohm
n = length (v)
for i=1:n,
    wrdac2 (v (i));
    pause (.8)
    v0 = rdchan (0);                    %reads channel 0
    V0 (i) = v0;
    v1 = rdchan (1);                    %reads channel 1
    V1 (i) = v1;
end
I = 1000 * (V0-V1) / R;                 %I in mA
k=[I' V0' V1']
xlswrite ('data_resistance', k);        %writes measured data in xl
plot (V1-(V1(1)), I-(I(1)), 'o-', 'linewidth', 1)
grid on;
title ('I-V Characteristic of Resistor');
xlabel ('Voltage (in V)----->');
ylabel ('Current(in mA)----->');

```

```

%Script File for getting I-V characteristic of diode 4148
clc;                                %clears the command window.
clear all;                            %clears the workspace.
defA;                                  %defines digital input output object and add lines to it
V = 0 : 0.025 : 1.2;
n=length (V);

```

```

R = 9.1; %in ohm
for i=1:n,
    wrdac1 (V(i);
    pause (.5);
    v0 = rdchan (0); %reads channel 0
    V0 (i) = v0;
    v1 = rdchan (1); %reads channel 1
    V1 (i) = v1;
end
I = 1000 * (V0-V1) / R; %I in mA
k = [I', V0', V1'];
xlswrite ('data_diode', k); %writes measured data in xl
plot (V1, I)
grid on;
title ('I - V Characteristic of Diode');
xlabel ('Voltage (in V)----->');
ylabel ('Current(in mA)----->');

```

%Script file for measuring resistivity of given thin films

%using DAC1

```

clc; %clears the command window.
clear all; %clears the workspace
defA; %defines digital input output object and add lines to it
Rin = 9.1; %in ohm
w = 0.2; %width of thin films in cm
t = 5×10-6 %thickness of thin films in cm (500 Å)
l = 0.7; %length of b/w the probes in cm
A = w * t; %area of cross-section of thin films
v = 0: 0.05 : 1.2;
n = length (v);

```

```

for i=1:n,
    wrdac1 (v(i));
    pause (.4)
    v0 = rdchan (0);           %reads channel 0
    V0 (i) = v0;
    v1 = rdchan (1);         %reads channel
    V1 (i) = v1;
    v2 = rdchan (2);         %reads channel 2
    V2 (i) = v2;
    v3 = rdchan (3);         %reads channel 3
    V3 (i) = v3;
end
I = 1000 * (V0-V1) / Rin;     %I in mA
Rt = 1000 * (V2-V3) / I;     %resistance of thin films in ohms
k = [V0' V1' V2' V3' I' Rt'];
xlswrite ('Data_ThinFilm', k);
resistivity = R * A / l;     %resistivity in ohm cm
fprintf ('The resistivity of given Thin Films is %d', resistivity);

```