

COMPUTER BASED INSTRUMENTATION SYSTEM FOR PRESSURE MEASUREMENT IN MATLAB APPLICATION

NURLIYANA BINTI MOHD JOHARI

This draft thesis is submitted as partial fulfillment of the requirements for the award of
the Bachelor of Electrical Engineering (Hons.) (Electronics)

Faculty of Electrical & Electronics Engineering
Universiti Malaysia Pahang

7 NOVEMBER 2008

DECLARATION

I declare that this thesis entitled “Computer Based Instrumentation System for Pressure Measurement in MATLAB Application” is the result of my own research except as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature :

Name : Nurliyana binti Mohd Johari

Date :

DEDICATION

In the Name of Allah, the Most Beneficent, the Most Merciful

*Special Dedication of This Grateful Feeling to My Family
(Mohd Johari bin Ismail, Norihan binti Sidek, and my brothers)
UMP FKEE Lectures, Abbas Saliimi bin Lokman,
4BEE Student
And all my beloved friends...*

Thanks for their Love, Support and Best Wishes.

“Tolong menolonglah kamu dalam perkara kebajikan dan janganlah kamu tolong menolong dalam perkara kemungkaran”

ACKNOWLEDGEMENT

Alhamdulillah, firstly I would like to thank Allah s.w.t because without His bless I will not finished my final year project successfully. I also would like to thank my supervisor and lecturer for this project; Miss Najidah binti Hambali and my co-supervisor Mr. Mohd Ashraf bin Ahmad, for their encouragement guidance, advices, critics, and helps. Without their support, my project would not have been as documented here.

My next special appreciation goes to UMP's library for supplying many relevant materials that has been used as my references for my project development. It was greatly helpful and useful.

My last appreciation goes to my entire supporter in this project which is all my colleagues especially Abbas Saliimi bin Lokman, Nor Syafiqah binti Mohamad Sharif, Norliza binti Morban, Jina anak Muli and all 4BEE students who gives their feedback, support and ideas for whole of my project. Their helps are highly appreciated. Not forget to give my sincere appreciation to my father, Mohd Johari bin Ismail and my mother, Norihan binti Sidek for their supports and motivations.

UNIVERSITI MALAYSIA PAHANG

BORANG PENGESAHAN STATUS TESIS♦

JUDUL: **COMPUTER BASED INSTRUMENTATION SYSTEM FOR
PRESSURE MEASUREMENT IN MATLAB APPLICATION**

SESI PENGAJIAN: 2008/2009

Saya NURLIYANA BINTI MOHD JOHARI (860716-33-5798)
(HURUF BESAR)

mengaku membenarkan tesis (Sarjana Muda/~~Sarjana~~ /~~Doktor Falsafah~~)* ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Tesis adalah hakmilik Universiti Malaysia Pahang (UMP).
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan tesis ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. **Sila tandakan (✓)

☐

SULIT

(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)

☐

TERHAD

(Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)

☒

TIDAK TERHAD

Disahkan oleh:

(TANDATANGAN PENULIS)

(TANDATANGAN PENYELIA)

Alamat Tetap:

**LOT 3780, KAMPUNG PAYA LUAS,
JALAN MARAN, 28000 TEMERLOH,
PAHANG**

MOHD ASHRAF BIN AHMAD
(Nama Penyelia)

Tarikh: **28 OKTOBER 2008**

Tarikh: : **28 OKTOBER 2008**

- CATATAN:
- * Potong yang tidak berkenaan.
 - ** Jika tesis ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali tempoh tesis ini perlu dikelaskan sebagai atau TERHAD.
 - ♦ Tesis dimaksudkan sebagai tesis bagi Ijazah doktor Falsafah dan Sarjana secara Penyelidikan, atau disertasi bagi pengajian secara kerja kursus dan penyelidikan, atau Laporan Projek Sarjana Muda (PSM).

“I hereby declare that I have read this thesis and in
my opinion this thesis is sufficient in terms of scope and
quality for the award of the degree of
Bachelor Electrical Engineering (Electronic)”

Signature :
Supervisor : Mr. Mohd Ashraf bin Ahmad
Date :

ABSTRACT

This project presents a mechanism of collecting data from a process control plant. Generally, this project based on process control that used the instrument to measure pressure in kPa and converted the output to the current in mA. This project used the equipments from the experiment and built the system using MATLAB, graphical user interface (GUI) to manipulate the data. From the data, students can do investigations and analysis such as plot the graph of the output and error, and uncertainty of the measurement for the calibration of the pressure transmitter. This mechanism will help students during their laboratory session. In order to support the system, data acquisition card, PCI 1710HG will be used to transfer data from pressure transmitter to the computer. The result will be displayed in the GUI including the data that have been transfer from the instrument to the computer. The graph of five-point calibration and error also shown. In addition, the results of the calculation for average and error of the data, standard deviation, combined standard uncertainty and the effective degree of freedom to get the uncertainty are included in the study analysis. Students can use this proposed mechanism to do the data analysis which is very convenience for the lab session. This system will benefit not only the students but also for the instructor.

ABSTRAK

Projek ini menerangkan mekanisma pengumpulan data daripada pelan kawalan. Secara umumnya, projek ini berdasarkan pengawalan proses yang menggunakan alat-alat untuk mengukur tekanan dalam unit kPa dan menukar hasilnya kepada unit arus dalam mA. Projek ini juga menggunakan alat-alat tersebut untuk menjalankan eksperimen dan membina sebuah sistem menggunakan MATLAB GUI untuk memanipulasi data. Pelajar-pelajar boleh menganalisa data-data yang diperolehi daripada eksperimen tersebut dengan memplot graf hasil and kesalahan, ketidakpastian ukuran bagi pengujian pemancar tekanan. Sistem ini boleh membantu pelajar-pelajar semasa menjalankan eksperimen di makmal. Selain itu, 'data acquisition card', PCI 1710HG akan digunakan untuk memindahkan data daripada pemancar tekanan kepada system di dalam computer. Keputusan analisis akan ditunjukkan di GUI termasuk data yang telah dipindahkan. Graf 5-titik pengujian dan kesalahan juga ditunjukkan. Selain itu, jawapan kepada pengiraan purata dan kesalahan data, selisihan ukuran, gabungan ketidakpastian dan efektif darjah kebebasan untuk mendapatkan ketidakpastian adalah termasuk di dalam analisis. Para pelajar boleh menggunakan sistem cadangan ini untuk membuat analisis yang akan memudahkan eksperimen di makmal. Sistem ini juga akan memberi faedah bukan sahaja kepada para pelajar malah kepada tenaga pengajar.

TABLE OF CONTENTS

	CONTENTS	PAGE
	TITLE PAGE	i
	DECLARATION	ii
	DEDICATION	iii
	ACKNOWLEDGEMENT	iv
	ABSTRACT	v
	ABSTRAK	vi
	TABLE OF CONTENTS	vii
	LIST OF TABLES	x
	LIST OF FIGURES	xi
	LIST OF APPENDICES	xiii
I	INTRODUCTION	1
	1.1 Background	1
	1.2 Objective	2
	1.3 Project Goal	3
	1.4 Project Scope	3
	1.5 Thesis Overview	4
II	LITERATURE REVIEW	6
	2.1 Introduction	6
	2.1.1 Instrumentation	7

	2.1.2	Current Signal vs. Voltage Signals	9
	2.1.3	Data Acquisition Systems and Real-Time Applications	10
	2.1.4	MATLAB	12
III		HARDWARE IMPLEMENTATION	14
	3.1	Introduction	14
	3.2	Methodology	14
	3.3	Experimental	17
	3.4	Theory of Pressure Measurement	18
	3.4.1	Absolute pressure, gauge pressure, differential pressure and vacuum	18
	3.4.2	Mechanical Transducers for Pressure Measurement	19
	3.4.3	Pressure Transmitter	20
	3.4.4	TUR for EJX110A Pressure Transmitter Calibration by MT220 Digital Manometer	22
	3.5	Data Acquisition Card	23
	3.5.1	Introduction	23
	3.5.2	DAQ Configuration	24
	3.5.2.1	Real Time Window Target Setup	24
	3.5.2.2	Installation and Configuration	26
	3.5.2.3	Installing the Kernel	27
	3.5.2.4	Testing the Installation	29
	3.5.2.5	Procedures of Creating Real-Time Application	31
	3.5.2.5.1	Creating a Simulink Model	31
	3.5.2.5.2	Entering Configuration Parameters for Simulink	39
	3.5.2.5.3	Entering Simulation Parameters for Real-Time	

	Workshop	40
	3.5.2.5.4 Creating a Real-Time Application	43
	3.5.2.5.5 Running a Real-Time Application	44
IV	SOFTWARE DEVELOPMENT	48
	4.1 Introduction	48
	4.2 Software Development	48
V	RESULTS AND DISCUSSION	54
	5.1 Introduction	54
	5.2 Results	54
	5.3 Discussions	56
	5.4 Costing and Commercialization	57
VI	CONCLUSION	58
	6.1 Conclusion	58
	6.2 Recommendations	59
	REFERENCES	60
	APPENDICES A-C	62

LIST OF TABLES

TABLE NO	TITLE	PAGE
3.1	Equipment Required for Experiment	21
4.1	Table of Formulas	53
5.1	Data of Pressure Measurement	55

LIST OF FIGURES

FIGURE NO	TITLE	PAGE
3.1	Methodology of the System	16
3.2	Connection of Equipment	17
3.3	Relationship between absolute, gauge, and vacuum measurements	19
3.4	PCI-1710HG	23
3.5	Required Products for Real Time Window Target	25
3.6	Simulink Model rtvdp.mdl	29
3.7	Output Signals of rtvdp.mdl	30
3.8	Create a New Model	31
3.9	Empty Simulink Model	32
3.10	Block Parameters of to Workspace	33
3.11	Board Test OK	34
3.12	Block Parameters of Analog Input	35
3.13	Scope Parameters Dialog Box	36
3.14	Axes Properties in Scope Window	37
3.15	Scope Properties: Axis 1	37
3.16	Completed Simulink Block Diagram	38
3.17	Configuration Parameter – Solver	40
3.18	Configuration Parameters – Hardware Implementation	41

3.19	System Target File Browser	42
3.20	Configuration Parameters – Real-Time Workshop	43
3.21	Connect to Target from the Simulation Menu	45
3.22	Start Real-Time Code from Simulation Menu	46
4.1	Flowchart of the System	49
4.2	GUI	52
5.1	Results and Analysis	56

LIST OF APPENDICES

APPENDIX	TITLE	PAGE
A	Degree of Freedom	62
B	Data Sheet PCI-1710HG	63
C	Coding of m-file (MATLAB)	65

CHAPTER 1

INTRODUCTION

1.1 Background

Process Control is the automated control of a process. Process control is used extensively in oil refining, chemical processing, electrical generation and the food and beverage industries where the creation of a product is based on a continuous series of processes being applied to raw materials [1]. The Faculty of Electric & Electronics Engineering has a laboratory complete with instruments which is used in control system. This project is using those equipments to do the experiment and build a system to manipulate the data that obtained from the experiment so that students can do the study analysis such as plot the graph of the output and error, uncertainty of the measurement for the calibration of the pressure transmitter.

In current laboratory Industrial Instrumentation subject session, students do the five-point calibration of pressure transmitter to get the current reading. They also plot

the graph for five-point calibration and graph for error of the pressure transmitter. The uncertainty evaluation also calculated manually. The idea to develop this system using MATLAB is to help students to do study analysis during laboratory session. Beside that, data acquisition card, PCI 1710HG will be used to transfer data online from pressure transmitter to the system in computer.

1.2 Objective

This project has three major objectives which are understand the basic measurement principles of pressure transmitter, recognize the hardware that can be used to transfer data from instrument which is pressure transmitter to the computer, and develop software that can be implemented in laboratory session for Industrial Instrumentation course, BEE4523, that helps students in study analysis.

This project is based on instrumentation for pressure measurement that needs to know how to use pressure transmitter and other devices used in the experiment. Understand the basic principles will lead to the successful of the experiment. To connect between instrument and computer, hardware will be needed to transfer the data. It must be suitable so that the transferring process do not facing any problem.

The main objective of this project is to develop system that can be implemented in laboratory session for Industrial Instrumentation course. This system can be used to do study analysis so that it will be easier to the students to conduct their laboratory session.

1.3 Project Goal

The hardware used, PCI 1710HG will transfer the data measured from the pressure transmitter to the system in the computer in real-time application and the system does the study analysis. In addition, this project will benefit the students while conducting a laboratory experiment in Industrial Instrumentation subject, BEE4523. Students can use the proposed system for measuring pressure, calculating the output error, getting the graph for five-point calibration, graph error and evaluate the uncertainty.

1.4 Project Scope

To achieve the objective of this project, several work scopes are determined. The scope that has been proposed to the project includes study the basic measurement principles of pressure transmitter, search about data acquisition that will be needed to use as hardware to transfer the data and explore how to use MATLAB and studying the other programs using MATLAB to develop a system.

To understand the basic measurement, studying is the most important in order to use the instrument correctly without parallax error while taking the data. To determine the DAQ card that is suitable to transfer data from pressure transmitter to the computer, some research about data acquisition is needed to confirm the type of the DAQ card.

The system that will be designed is using MATLAB GUI that never been used before. Mastering this software is important in order to develop the system. Studying the function of coding and practicing in build the program using MATLAB will help the most to develop the system.

1.5 Thesis Overview

This thesis consists of 6 chapters that clarify about the system entitled computer based instrumentation system for pressure measurement in MATLAB application. Each chapter described about important part of the project such as the hardware and software development.

The first chapter divided into 5 parts which are the background of the project, objectives, project goal, project scope and the overview of the thesis.

Literature review is included in chapter II. A literature review is a body of text that aims to review the critical points of current knowledge on a particular topic and most often associated with science-oriented literature. [2] Literature review is also known as the current practice that has been used now.

The hardware includes the instruments used in the experiment, theory of pressure measurement and explanation of data acquisition card is explained in chapter III which is hardware part. The methodology is also included in this chapter. Beside that, the

interfaces between the hardware and the system and the real-time application also explained.

In chapter IV, the contents are about the software that has been developed. The functions of the GUI that have designed are described in this chapter.

Chapter V explained the results of this project and includes the discussion while chapter VI is the conclusion and the future development and recommendations.

CHAPTER II

LITERATURE REVIEW

2.1 Introduction

This chapter discussed about literature review that is mostly taken from journals and books. Internet sources also use to add more information about this project. The journals taken after considered the content which are either related or not to this project. The purpose of doing literature review is to find, read, and analyze the body of literature published on this project. This information can be used to improve this project as well.

2.2 Instrumentation

Many techniques have been developed for the measurement of pressure such as manometer that is limited to measuring pressures near to atmospheric, bourdon tube gauge that uses a coiled tube which as it expands due to pressure increase [3]. This article is about techniques to measure the pressure and it gives some examples of instrument that can be used to measure the pressure. The selection of instrument is very important to adapt it to the current situation or to the project conducted.

A pressure transducer is fundamentally any device that converts an applied pressure into an electrical signal. There have been many different types of pressure transducer developed over the years such as bonded foil, thick film, thin film & semiconductor strain gauge. All of these sensing technologies are pressure transducers and they provide an electrical signal typically a millivolt output signal which varies with changes in pressure when connected to an appropriate power supply. A pressure transmitter is simply a pressure transducer with some extra electronics to transmit a 4 to 20 mA output signal. At first pressure transmitters would only be found in large process plants and the sensors were bulky and relatively expensive. In recent years other industries have adopted the 4 - 20mA output signal pressure transmitter. An amplified voltage output is much more robust than a millivolt output signal and can be used over mediums distances without any noticeable signal losses. Typically only a few resistor components are required to condition the millivolt output from a strain gauge pressure sensor. Current output pressure transmitters for gauge, absolute & differential pressure sensing where a robust 2 wire 4 - 20mA current output loop is required which can work over long distances without signal degradation [4].

This article is about difference between pressure transducer and pressure transmitter. They produce an electrical signal such as in milivolt or in miliampere. There

are many advantages on pressure transmitter over pressure transducer such as they are robust, unnoticeable signal losses and can work over long distances without signal degradation.

A transducer is a device which converts one form of energy into another form of energy. In the field of electrical instrumentation, “a transducer is defined as a device which converts a physical quantity, a physical condition, or mechanical output into an electrical signal”. Most of the methods of converting mechanical output into an electrical signal work equally well for the bellows, the diaphragm and the Bourdon tube. In this conversion, a mechanical motion is first converted into a change in electrical resistance and then the change in resistance is converted into a change in electrical current or voltage. Electrical pressure transducer consists of three elements:

1. Pressure sensing element such as bellow, a diaphragm or a Bourdon tube
2. Primary conversion element, e.g. resistance or a voltage.
3. Secondary conversion element

Pressure instrument calibration is the process of adjusting the instruments output signal to match a known range of pressures. All instruments tend to drift from their last setting. This is because spring stretch, electricals components undergo slight changes on the atomic level, and other working parts sag, bend, or lose their elasticity. Basic calibration procedure includes zero, span, and linearity adjustments. Proper calibration provides the desired beginning and ending pressures, and produces an output signal that is proportional to the process pressure. Calibration of the instrument is carried out by applying to it an air or liquid pressure whose value is accurately known. The method used depends upon the range of the instrument [5].

2.3 Current Signal and Voltage Signals

There are two types of signals that can be resulted from the experimentation of pressure which are voltage signal and current signal.

Voltage signals are normally standardized in the voltage ranges 0 to 5 V, 0 to 10 V, or 0 to 12 V, with 0 to 5 V being the most common. The requirements of the transmitter are a low output impedance to enable the amplifier to drive a wide variety of loads without a change in the output voltage, low temperature drift, low offset drift, and low noise. Its low output impedance enables the driver to charge up the line capacitance, achieving a quick settling time. Current signals are standardized into two ranges; these are 4 to 20 mA and 10 to 50 mA, where 0 mA is a fault condition. The latter range was the preferred standard but has now been dropped, and the 4 to 20 mA range is accepted standard. The requirements of the transmitter are high output impedance, so that the output current does not vary with load, low temperature, offset drift, and low noise. The main disadvantage of the current signal is its longer settling time due to the high –output impedance of the driver which limits the available current to charge up the line capacitance. After the line capacitor is charged, the signal current at the controller is the same as the signal current from the transmitter and not affected by the normal changes in lead resistance. The internal resistance of the controller is low for current signals i.e., a few hundred ohms. Again a differential signal connection eliminates noise and ground problems [6].

For the highest possible pressure sensor accuracy it will need an output signal which is not easily corrupted and has a very high resolution. Digital output signals do not suffer from signal losses or interference like analogue ones do, either the complete signal gets through as originally transmitted from the pressure sensor or none at all [4]. This article is about the advantages using digital output signal than using analog output

signal. To get a high resolution on the output signal, an analogue to digital converter is needed. There is no need to convert a digital signal to analogue because it can be easily interfaced to a computer or data acquisition card via a digital connection. Another benefit is the microprocessor inside the DAQ is also digital so that can eliminate linearity errors.

2.4 Data Acquisition Systems and Real-Time Applications

Data acquisition systems (DAS) interface between the real world of physical parameters, which are analog, and the artificial world of digital computation and control. With current emphasis on digital systems, the interfacing function has become an important one; digital systems are used widely because complex circuits are low cost, accurate and relatively simple to implement. In addition, there is rapid growth in the use of microcomputers to perform difficult digital control and measurement functions. Computerized feedback control systems are used in many different industries today in order to achieve greater productivity in our modern industrial societies. The device that perform the interfacing function between analog and digital worlds are analog-to-digital (A/D) and digital-to-analog (D/A) converters, which together are known as data converters [7].

Some data-acquisition boards are designed to interface with several different sensors. The sensor support includes sensor excitation, linearization, cold reference compensation, and conversion of the output to engineering units. Multisensors plug-in boards typically contain four sections. The first section performs the signal conditioning for the sensors, multiplexes to the appropriate sensor and amplifies the signal with a programmable gain. The second section performs A/D conversion. The third section

incorporates a microcomputer with on-board memory used in data processing to perform tasks such as linearization, reference junction compensation, and engineering unit conversion. The microprocessor also provides the logic to scan the sensors, adjust the amplifier gain, and transfer the data to the standard bus registers. The standard bus interface incorporates drivers and receivers to facilitate communication with a PC host computer. The input/output ports vary from card to card, but a typical configuration that utilizes serial communication has three ports. One port is used to transfer data and instructions to the card, and the other two ports are used to transfer data and indicate status to the host computer. The card is programmed from the PC host computer and the digitized data are transferred from the card to the memory of the PC using standard bus in the host computer. External bus structure is not required for data transmission, because the entire data-acquisition system is contained within the PC. All further processing and preparation of graphics are performed on the host computer using commercially available software [8].

Besides A/D and D/A converters, data acquisition and distribution systems may employ one or more of the following: transducers, amplifiers, filters, nonlinear analog functions, analog multiplexers, and sample-holds. The interconnection of these components makes a portion of a computerized feedback control system. The input of the system is a physical parameter such as temperature, pressure, flow, acceleration, and position, which are analog quantities. The parameters are first converted to electrical signal by means of transducer; once in electrical form, all further processing is done by electronic circuits.

Real-time operating system is needed to do this project. Operating system arbitrates and controls the resources of a computer system and resolves conflicts, optimizes the performance, helps the user to easily implement device-oriented application programs. The software in the computer used this application to collect the data from instruments. There are specific ways to set the configuration of the real-time

operation depends on the software used. RTOS is used for the process control computer application. It is capable of managing real-time resource scheduling and control problems in computer based industrial process control systems. Any real-time computer system must be able to respond interrupts from external devices [5].

A project based on an array of metal oxide sensors (MOS) devices has been develop referring to the paper by Hsin-Ti Chueh and John V. Hatfield. A dedicated real-time data acquisition system with programmable voltage generator, self-calibration techniques and custom visual software programs for a desktop PC and a hand-held computer have been designed and constructed for this application. This paper also describes an example of software calibration to correct source errors for this data acquisition system. Experimental results show that this data acquisition system can accurately measure a large range of resistive sensor responses and is suitable for measuring both conducting polymer and MOS sensors [9].

The data acquisition system designed is explained in details. Sensor arrays, where each sensor has low specificity, need appropriate data acquisition system and pattern recognition software to recognize simple and complex odors. The data acquisition systems interact continuously with the environment and embedded software synchronously reads the sensor data and controls the circuits [9].

2.5 MATLAB

“Using MATLAB with the Instrument Control Toolbox and the Data Acquisition Toolbox helped us to meet our project’s deadline and make our project successful” Reed

Farrar, Newport Corporation [10]. This is actually user's story about Newport Corporation that develops a variety of products for the semiconductor research industry that requires quick and accurate analysis of test data. They have found out that using MATLAB is the fastest way to analyze results from the instruments. They used to spend days on just one test but with MATLAB they can run the same test and analyze the results in a few hours. Newport increases the efficiency and quality of test data by using MATLAB, the Instrument Control Box and the Data Acquisition Toolbox to acquire and analyze measurement results from within a single environment.

A MATLAB-based graphical user interface (GUI) program has been develop and implemented for tomographic viscometer data processing. The tomographic viscometer is based on a velocity profile measurement using magnetic resonance imaging (MRI) or ultrasonic Doppler velocimetry (UDV). This program enables users to process image data, to calculate rheological properties and to visualize results [11]. This paper presents a MATLAB-based software program for analysis of fluid rheological properties. The program structure is about allowing users to process tomographic data from either MRI or UDV and to calculate the rheological properties immediately after data acquisition.

CHAPTER III

HARDWARE IMPLEMENTATION

3.1 Introduction

This chapter discussed about methodology used to conduct this project and the equipment used for the experimental purpose. The experiment includes the measurement of the pressure. The important part of hardware is DAQ configuration is also included in this part.

3.2 Methodology

This project is based on instrumentation so that some instruments are needed to complete this project. Below is the block diagram of overall project is shown. The instrument will be used to measure pressure before continue to do the study analysis from the data measured using system designed.

In this project, PCI 1710HG is used to transfer data from pressure transmitter to the computer. The hardware is interfaced to the system. Once this is done, the data will be stored in the system that is built using MATLAB. The data that have been transferred and collected in the system will be analyzed to get average, output error, standard deviation and combined standard uncertainty. The confidence limits also can be calculated. The graph for the five point calibrations and error for pressure transmitter will be displayed using this system. The methodology of the system is shown in the Figure 3.1 below.

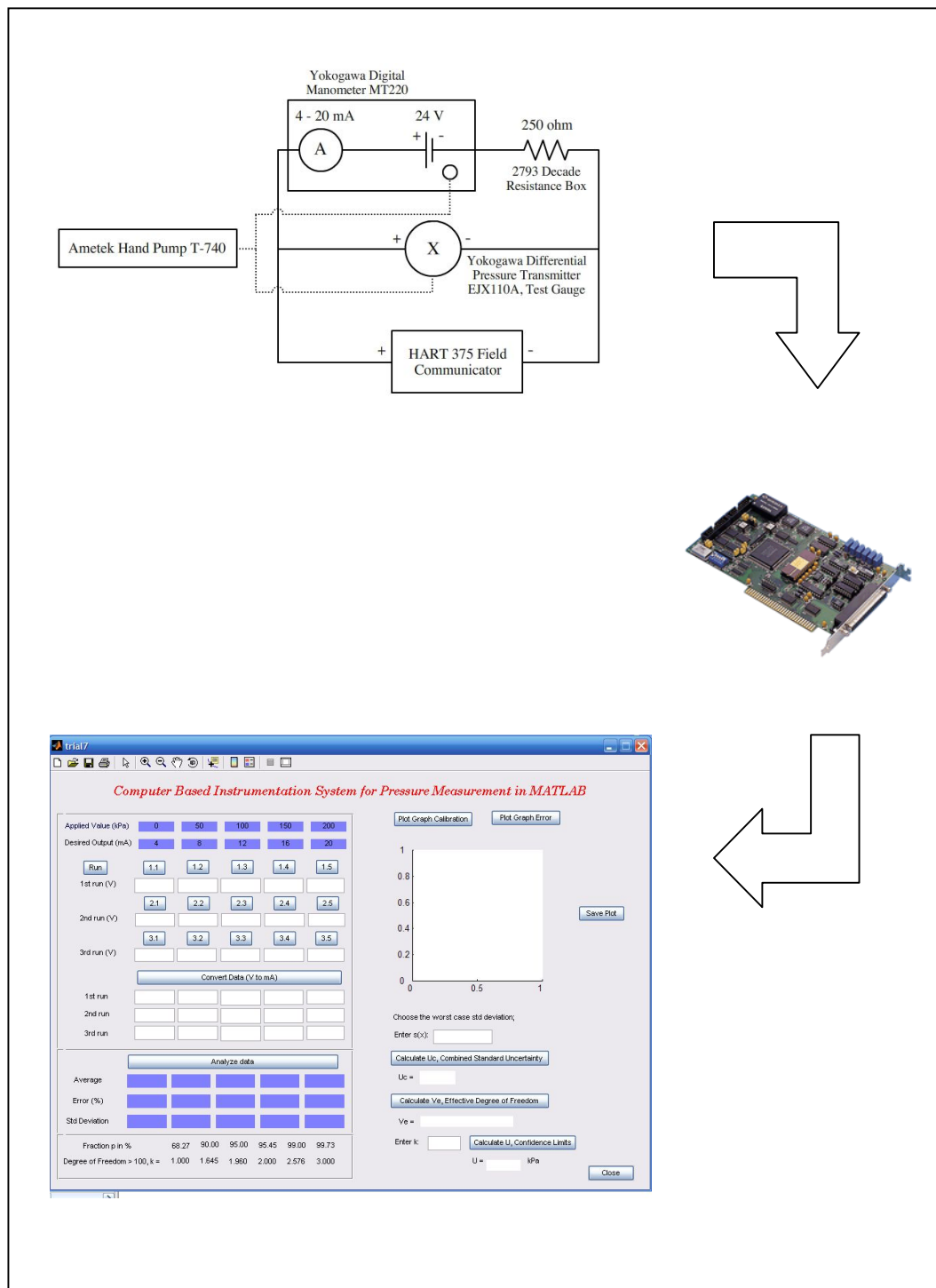


Figure 3.1: Methodology of the System

3.3 Experiment Investigations

This project is based on instrumentation that specific on pressure instrumentation. The equipments needs to do the experiment are Yokogawa Digital Manometer, MT220, 2739 Decade Resistance box, 250 Ω , Ametek Hand pump, T-740, Yokogawa Differential Pressure Transmitter EJX110A, Test Gauge, and HART 375 Field Communicator.

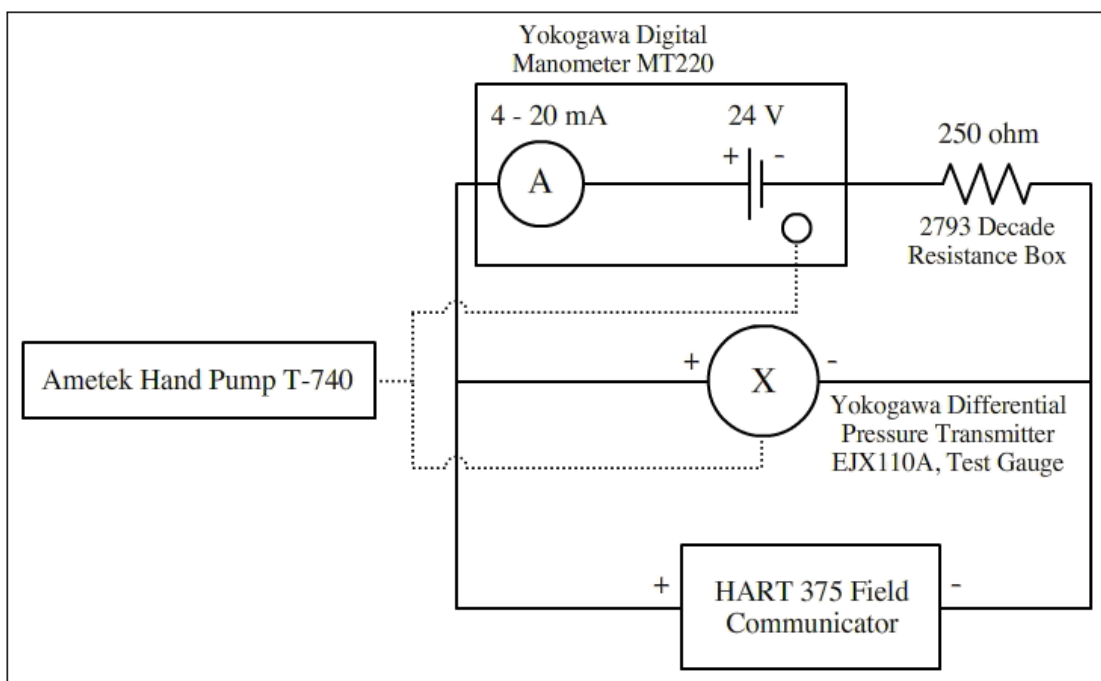


Figure 3.2: Equipment Connection

The equipments are connected as shown in figure above. The scale of the MT220 is change to kPa with maximum reading 200 kPa and range 700 kPa. The Master Standard Unit (MSU) is adjusted such that Unit Under Test (UUT) reads the first point. The first point on the output is 4 mA which is corresponds to an input of 0.0 kPa. The

value of MSU is increased by 50 kPa until the maximum values applied which is 200 kPa. This step is run for three times so that the average of the output can be calculated.

3.4 Theory of Pressure Measurement

3.4.1 Absolute pressure, gauge pressure, differential pressure and vacuum

Pressure is the force exerted by a gas or liquid on a surface. The SI unit of pressure measurement is the Pascal (Pa). Other common units are N/m^2 , Torr, psi and bar. It is critically to specify the reference point of the pressure. When measured a pressure in a system with perfect vacuum or absolute zero as the basis then we call the value of the pressure as the absolute pressure. When the pressure is measured with reference to the atmospheric pressure as the basis then the measurement is called the gauge pressure. The relationship between absolute pressure and gauge pressure is expressed as,

$$P_a = P_g + 101.3 \quad \text{----- (1)}$$

Where,

P_a and P_g = Absolute and gauge pressure respectively, kPa

The 101.3 in the equation is the standard atmospheric pressure at the earth's surface in kPa.

Vacuum gauges are used when the pressure being measured has a value less atmospheric pressure. Vacuum pressure may be expressed as absolute pressure or vacuum units. For example, 10 kPa vacuum signifies a pressure of 10 kPa below atmospheric pressure, that is, an absolute pressure of 91.3 kPa ($101.3 \text{ kPa} - 10 \text{ kPa}$).

Measurement above atmospheric pressure

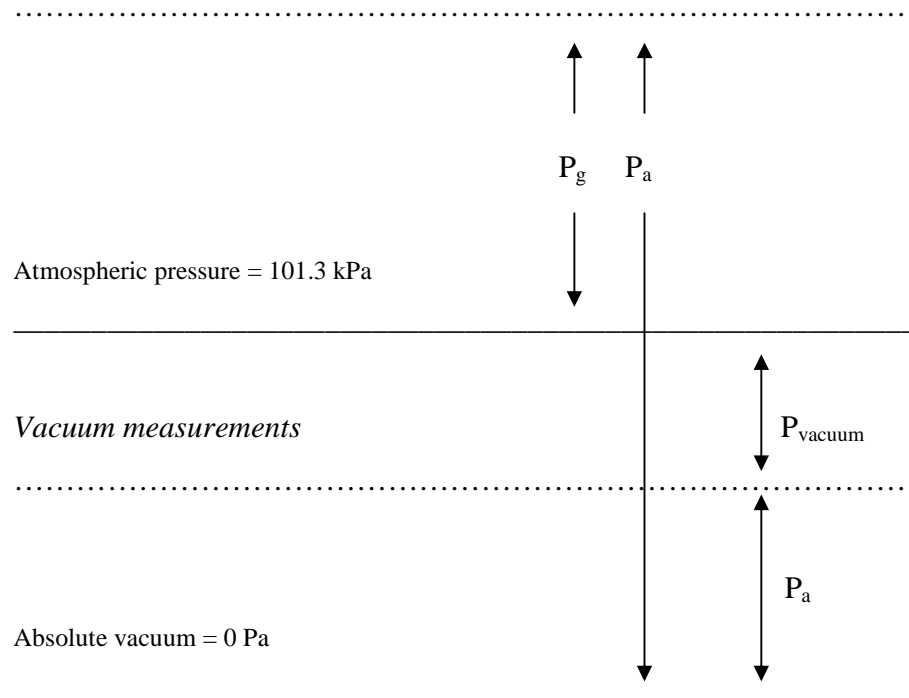


Figure 3.3: Relationship between absolute, gauge and vacuum measurements

Differential pressure signifies the difference in pressure between two points. Differential pressure measurement is useful for measuring flow or level.

3.4.2 Mechanical Transducers for Pressure Measurement

Pressure is measured by the force it exerts on a mechanical element and the corresponding deflection of the mechanical element. The common types of mechanical devices used are the Bourdon gauge, Bellows and Diaphragm.

The Bourdon gauge consists of a tube with elliptical cross section bent in the form of the arc of a circle. One end of the tube is fixed and the other is free to rotate. The free end is closed while the fixed end is connected to the pressure to be measured. When a pressure is applied inside the tube the elliptical cross section tends to become more circular. This produces a torque which tries to move the free end in a direction such that the arc becomes straighter. This small movement is converted to an electrical signal by using a suitable technique.

In a case of a bellows the pressure is applied to a bellows made of metal like stainless steel or phosphor bronze. One side is fixed and the other side is free to move. The application of pressure causes a small deflection to the free end which is converted to an electrical signal using a suitable technique.

The diaphragm transducers are made of thin metallic diaphragms. They are fixed along the periphery and enclosed in a diaphragm box. When pressure is applied to one side of the diaphragm it causes a deflection in the middle of the diaphragm which is converted to an electrical signal by appropriate methods.

3.4.3 Pressure Transmitter

The deflection of the mechanical transducers due to the action of pressure is converted to an electrical signal by a variety of methods. Linear Variable Differential Transducers (LVDT), Potentiometric Transducers, Capacitance Transducers, Piezoresistive and Silicon Resonant Bridges are some of the more important methods used for this purpose.

Of these methods are Silicon Resonant Sensor Method is extremely accurate and stable. The Yokogawa transmitters are based on the Silicon Resonant Sensor Technique that measures a strain-induced frequency created by the silicon resonators.

The entire sensor is made from a homogenous silicon crystal. On the silicon sensors two 'H' shaped resonators are provided. These resonators are patterned on the silicon crystal itself. One resonator is at the center of the crystal while the other is at the outer edge of the diaphragm. When no pressure is acting on the diaphragm both of the bridge oscillate at a frequency of 90 kHz.

When the pressure is applied to the diaphragm the center bridge goes into tension and the outer bridge into compression. As a result the frequencies of one resonator increase and the other decreases. This difference in frequencies is measured and converted to a frequency output. The measurement can be directly converted to digital output by pulse counters. The advantages of this technique are the high accuracy, long-term stability and the facility to directly get digital signals without having to use A/D conversion.

The equipment required for performing the experiments are shown in the table below:

Table 3.1: Equipment Required for Experiment

No	Unit Name	Model Number
1	Differential pressure transmitter	Yokogawa Model No. EJX110A
2	Pressure Gauge	0-4 bar g
3	Pressure calibrator-Digital manometer	Yokogawa Model No. MT220
4	Hand-Pump	Amatek T-740

3.4.4 TUR for EJX110A Pressure Transmitter Calibration by MT220 Digital Manometer

Range of EJX110A Transmitter	= 0 to 200 kPa
Accuracy	= $\pm 0.04\%$ of span
	= $200 \times 0.04/100$
	= 0.08 kPa
Accuracy of MT220 Digital Manometer	= $[200 \times 0.01/100] + [700 \times 0.005/100]$
	= 0.02 + 0.035
	= 0.055
TUR	= 0.08/0.055
	= 1.45

We can use the MT220 digital manometer as the MSU for the EJA430A pressure transmitter calibration.

By using the Amatek handheld pump, we also can calibrate the EJA430A pressure transmitter.

$$\begin{aligned} \text{Accuracy of T-740} &= 0.5/100 \times 200 \text{ kPa} \\ &= 1 \text{ kPa} \end{aligned}$$

$$\text{Accuracy of the EJA430A transmitter} = 0.08 \text{ kPa}$$

$$\begin{aligned} \text{TUR} &= 0.08/1 \\ &= 0.08 \end{aligned}$$

Therefore the MT220, Digital Manometer is used as the reference master standard, and the hand-pump is the pressure generator. Generally, the MT220 has an

accuracy close the pressure transmitter, EJX110. Hence, it is used as a comparison tool only.

3.5 Data Acquisition Card (DAQ Card)

3.5.1 Introduction

The output of each value from pressure transmitter is recorded and sent to the system in the computer using hardware that interfaced with the system. The hardware that has been choose is data acquisition PCI-1710HG.



Figure 3.4: PCI-1710HG

The PCI-1710 is a multi-function data acquisition card for the PCI bus. This card provides multiple measurement and control functions. Model PCI-1710 offers 16 12-bit single ended channels of A/D input, 16 channels of digital inputs, 16 channels of digital outputs, two 12-bit channels of analog output, and one 16-bit timer/counter with a time base of 10 MHz. Model PCI-1710 also allows the user to configure the A/D inputs as differential analog inputs. This card provides an on-board FIFO (First In First Out)

memory buffer that can store up to 4K A/D samples. The card provides a programmable counter for generating a pacer trigger for the A/D conversion. The counter chip is an 82C54 or equivalent, which includes three 16-bit counters on a 10 MHz clock. One counter is used as an event counter for counting events coming from the input channels. The other two are cascaded together to make a 32-bit timer for a pacer trigger [12].

3.5.2 DAQ Configuration

3.5.2.1 Real Time Window Target Setup

Real time window Target enables to run Simulink and Stateflow models in real time on desktop or laptop PC for rapid prototyping or hardware-in-the-loop simulation of control system and signal processing algorithms. A real-time execution can be created and controlled entirely through Simulink. Using Real-Time Workshop, C code can be generated, compiled, and started real-time execution on Windows PC while interfacing to real hardware using PC I/O board (PCI-1710HG). I/O device drivers are included to support an extensive selection of I/O boards, enabling to interface to sensors, actuators, and other devices for experimentation, development, and testing real-time systems. Simulink block diagram can be edited and Real-Time Workshop can be used to create a new real-time binary file. This integrated environment would implement any designs quickly without lengthy hand coding and debugging. Figure 3.4 shows the required product of Real Time Window Target.

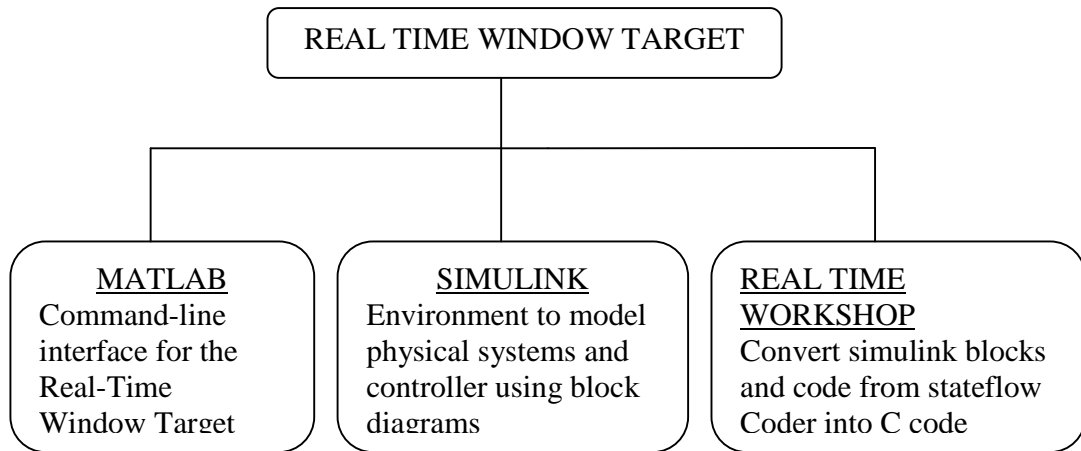


Figure 3.5: Required Products of Real Time Window Target

Real –Time Window Target includes a set of I/O blocks that provide connection between the physical I/O board and real –time model. Hardware –in-the-loop simulations can be ran quickly observed how Simulink model responds to real-world behavior. I/O signals can be connected using the block library for operation with numerous I/O boards.

The following types of blocks are included:

- Digital input blocks: connect digital input signal to simulink block diagram to provide logical inputs.
- Digital output blocks: connect logical signals from Simulink block diagram to control external hardware.
- Analog input blocks: enable to use A/D converters that digitalize analog signals for us as inputs to simulinks block diagrams.
- Analog output blocks: enable simulinks block diagram to use D/A converters to output analog signals from simulink model using I/O board(s).

- Counter Input Blocks: enable to count pulses or measure frequency using hardware counters on I/O board(s).
- Encoder Input blocks. Enable to include feedback from optical encoders.

3.5.2.2 Installation and Configuration

The Real-Time Windows Target is a self-targeting system where the host and the targeting computer are the same computer. It can be installed on a PC-compatible computer running Windows NT 4.0, Windows 2000, or Windows XP. The Real-Time Windows Target requires one of the following C compilers which are not included in with the Real-Time Windows Target:

- Microsoft Visual C/C++ compiler- Version 5.0, 6.0, 7.0.
- Watcom C/C++ compiler- version 10.6, and 11.0. During installation of watcom C/C++ compiler, a DOS target is specified in addition to windows target to have the necessary libraries available for linking.

After installation, the MEX utility is run to select compiler as the default compiler for building real-time applications.

Real-Time Workshop uses the default C compiler to generate executable code, and the MEX utility uses this compiler to create MEX-files.

This procedure is executed in order to select either a Microsoft Visual C/C++ compiler before build an application. Note, the LCC compiler is not supported:

1. mex- setup is typed in the MATLAB window

MATLAB will display the following message:

```
Please choose your compiler for building external
interface
(MEX) files. Would you like mex to locate installed
compilers? ([y]/n):
```

Then a letter "y" is typed

MATLAB will display the following message:

```
Select compiler:
[1]: WATCOM compiler in c:\watcom
[2]: Microsoft compiler in c:\visual
[0]: None
Compiler:
```

Next, a number is typed. For example, number 2 is typed to select the Microsoft compiler.

MATLAB will display the following message:

```
Please verify your choices:
Compiler: Microsoft 5.0
Location: c:\visual
Are these correct? ([y]/n)
```

Finally, a letter "y" is typed

MATLAB will reset the default compiler and display the message:

```
The default options file:
"C:\WINNT\Profiles\username\Application
Data\MathWorks\MATLAB\mexopts.bat" is being updated.
```

3.5.2.3 Installing the kernel

During installation, all software for the Real-Time Window target is copied onto hard drive. The kernel is not automatically installed. Installing the kernel sets up the kernel to start running in the background each time when computer is started. The kernel can be installed just after the Real-Time Windows Target has been installed. The installation of the kernel is necessary before a Real-Time Windows Target can be executed:

1. `rtwintgt-install` is typed in the MATLAB window.

MATLAB will display the following message

```
You are going to install the Real-Time Windows Target
kernel.
Do you want to proceed? [y]:
```

2. The kernel installation is continued by typing a letter “y”.

MATLAB will install the kernel and display the following message

```
The Real-Time Windows Target kernel has been
successfully installed.
```

The computer has to be restart if a “restart” message being displayed.

3. The kernel should be checked whether it was correctly installed. Then, `rtwho` is typed.

MATLAB would display a message similar to

```
Real-Time Windows Target version 2.5.0 (c) The Mathworks,
inc.1994-2003
MATLAB performance = 100.0%
Kernel timeslice period = 1ms
```

After the kernel being installed, it remains idle, which allows windows to control the execution of any standard Windows application. Standard Windows applications include internet browsers, word processors, MATLAB, and so on. It is only during real-time execution of model that the kernel intervenes to ensure that the model is given priority to use the CPU to execute each model updating at the prescribed sample intervals. Once the model update at a particular sample interval completed, the kernel releases the CPU to run any other Windows application might need servicing.

3.5.2.4 Testing the installation

The installation can be tested by running the model `rtvdp.mdl`. This model does not have any I/O blocks, so that this model can be run regardless of the I/O boards in computer. Running this model would test the installation by executing Real-Time Workshop, the Real-Time Window Target, and the Real-Time Windows Target kernel. After the Real-Time Windows Target kernel being installed, the entire installation can be tested by building and running a Real-Time application. The Real-Time Window Target includes the model `rtvdp.mdl`, which already has the correct Real-Time Workshop options selected for users:

1. `rtvdp` is typed in the MATLAB window

The simulink model `rtvdp.mdl` window will be opened as shown in Figure 3.6

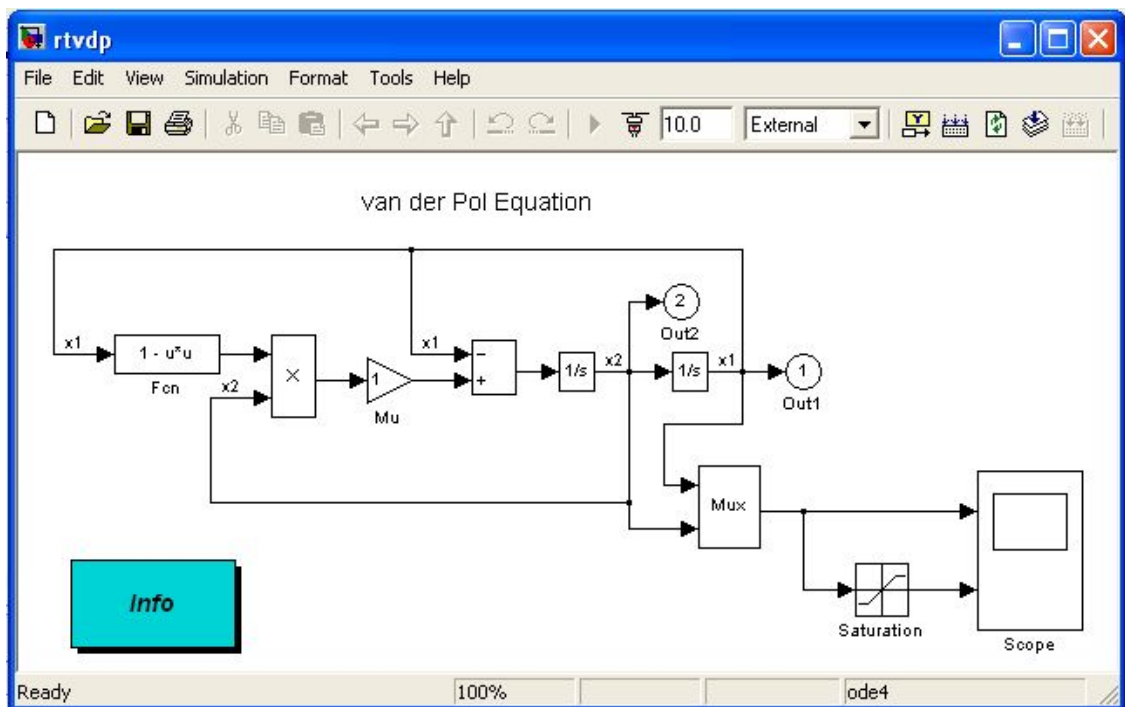


Figure 3.6: Simulink Model `rtvdp.mdl`

2. From the Tools menu, it should be pointed to Real-Time Workshop and then clicked Build Model. The MATLAB window display the following messages:

```

# # # starting Real-Time Workshop build for model: rtvdp
# # # Invoking Target Language Compiler on rtvdp.rtw
. . .
# # # Compiling rtvdp.c
. . .
# # # Create Real-Time Windows Target module rtvdp.rwd.
# # # Successful completion of Real-Time Workshop build
procedure for model:rtvdp

```

3. From the Simulation menu, External should be clicked and followed by clicking Connect to target.

The MATLAB window displayed the following message:

Model rtvdp loaded

4. Start Real-Time Code is clicked from Simulation menu.

The scope window will display the output signals. After the Real-Time application has been run, Scope Window should indicate such a figure as shown in Figure 3.7.

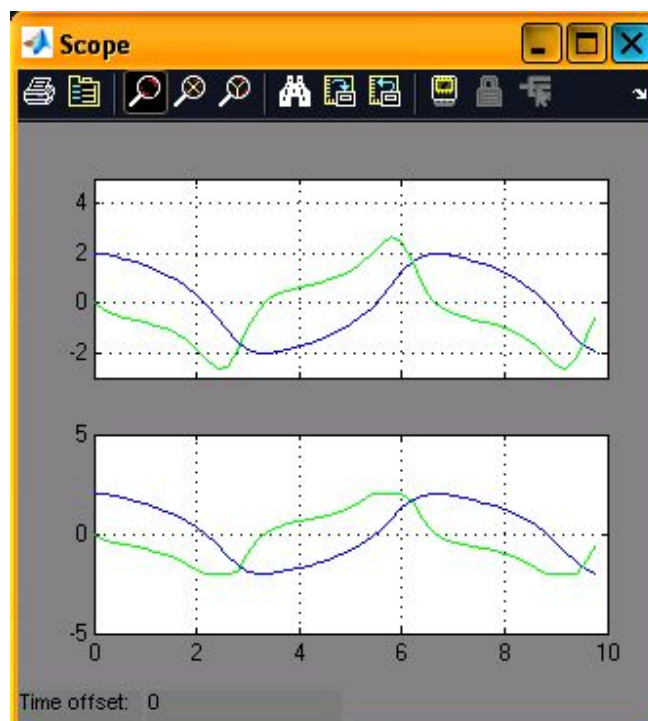


Figure 3.7: Output Signals of rtvdp.mdl

5. From the Simulation menu, after the Stop Real time Code is clicked, the real-time application will stop running and then the Scope window will stop displaying the output signals.

3.5.2.5 Procedures of Creating Real-Time Application

3.5.2.5.1 Creating a Simulink Model

This procedure explains how to create a simple Simulink model. This model is used as an example to learn other procedures in the Real-Time windows Target. A Simulink model has to be created before it can be run as simulation or create a real-time application:

1. Simulink is typed in the MATLAB Command Window.
The simulink Library browser window is opened as shown in Figure 3.8.
2. From the toolbar, the Create a new model button is clicked.

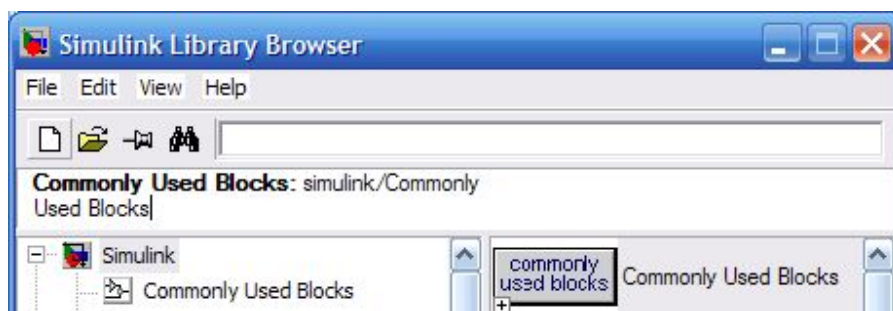


Figure 3.8: Create a new model

An empty Simulink window is opened. With the toolbar and status bar disabled, the window looks like the following figure:

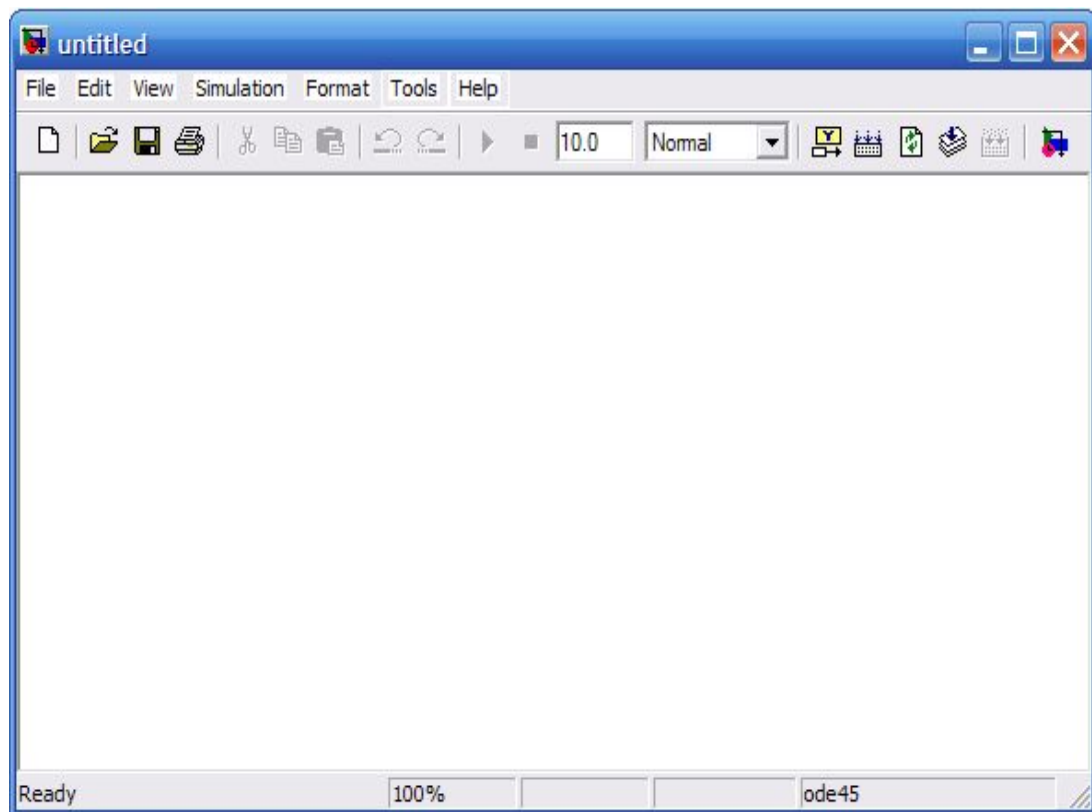


Figure 3.9: Empty Simulink Model

3. In the Simulink Library Browser window, Simulink is double-clicked, and then Sinks is also double clicked. Next, Simout is clicked and dragged to the Simulink window. Scope is clicked and dragged to the Simulink window too. Real-Time window Target is clicked. Analog input is clicked and dragged to the Simulink window.
4. The Analog input is connected to the scope input by clicking-and-dragging a line between the blocks. Likewise, the scope is connected to the connection between analog input and simout.
5. The simout block is double clicked. The block parameters dialog box opens.

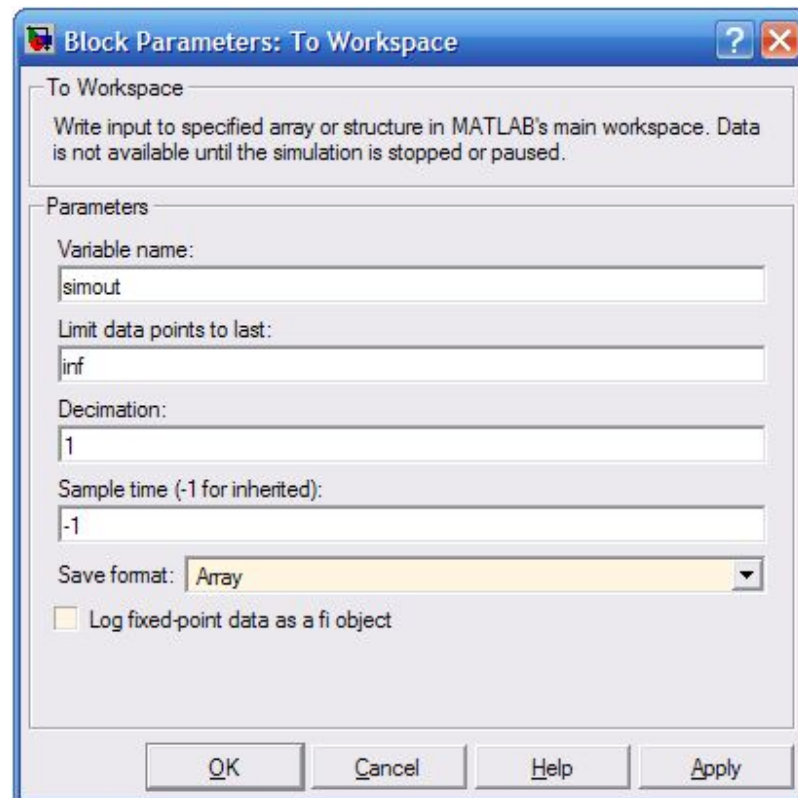


Figure 3.10: Block Parameters of To Workspace

6. OK is clicked.
7. The Analog Input block is double clicked.
The Block Parameters dialog box will open.
8. The Install new board button is clicked a board name. For example, it should be pointed to Advantech and then clicked PCI-1710HG.
9. One of the following is selected:
 - For an ISA bus board, a base address is entered. This value must match the base address switches or jumpers set on the physical board. For example, to enter a base address of 0x300 in the address box, 300 is typed. The base address also could be selected by selecting check boxes A9 through A3.
 - For a PCI bus board, the PCI slot is entered or the Auto-detect check box is selected.
10. The Test button is clicked.

The Real-Time window Target tries to connect to the selected board and the following message would display if successful. It can be seen in Figure 3.11.

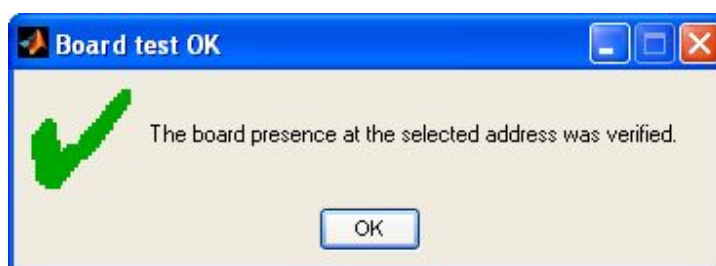


Figure 3.11: Board test OK

11. On the message box, OK is clicked.
12. The same value as entered in the Fixed step size box from the Configuration Parameters dialog box is entered in the Sample time box. For example, 0.001 is entered.
13. A channel vector that select the analog input channels that are using on this board is entered in the output Channels box. The vector can be any valid MATLAB vector form. For example, to select analog output channel on PCI-1710HG board.
14. The input range for all the analog input channels that have been entered in the input channels box is chosen from the Input range list. For example, with the PCI-1710HG, 0 to 5 V is chosen.
15. From the block input signal list, the following options is chosen:
 - Volt - expects a value equal to the analog output voltage.
 - Normalized unipolar – Expects a value between 0 to +1 that is converted to the full range of the output voltage regardless of the output range. For example, an analog output range of 0 to +5 volts and -5 volt would both be converted from values between 0 to +1.
 - Normalized bipolar - expects a value between -1 and +1 that is converted to the full range of the output voltage regardless of the output voltage range.

- Raw – Expects a value of 0 to $2^n - 1$. For example, a bit A/D converter would expect a value between 0 and $2^{12} - 1$ (0 to 4095).

The advantage of this method is expected value is always an integer with no round errors.

- The initial value is entered for each analog output channel that has been entered in the Output channels box. For example, if 1 is entered in the Output Channels Box, and an initial value of 0 volts is needed, 0 is entered.
- The final value is entered for each analog channel that has been entered in the Output channels box. For example, if 1 is entered in the Output Channels box, and a final value is needed, 0 is entered.

The dialog box would look similar to the Figure 3.12 if Volts is chosen.



Figure 3.12: Block Parameters of Analog Input

- One of the following is executed:
 - Apply is clicked to apply the changes to the model and the dialog box is left open

- b. OK is clicked to apply the changes to the model and the Block Parameters: Analog Output dialog box will close.
19. Parameters dialog box is closed, and the parameter values are saved with the Simulink model.
20. In the Simulink window, the Scope block is double clicked.
A Scope window will open
21. The Parameters button is clicked.
A Scope parameters dialog box will open
22. The General tab is clicked. The number of graphs that is needed in one Scope window is entered in the Number of axes box. For example, 1 is entered for a single graph. Do not select the floating scope check box. In the Time range box, the upper value for the time range is entered. For example, 1 second is entered. From the Tick labels list, bottom axis only is chosen.
From the Sampling list, decimation is chosen and 1 is entered in the text box.
The Scope parameters dialog box would look like such a Figure 3.13 as shown.

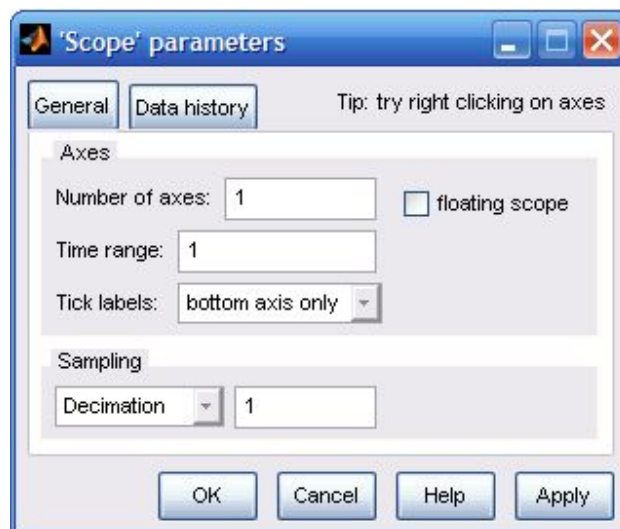


Figure 3.13: Scope parameters Dialog Box

23. One of the following done:
 - a. Apply is clicked to apply the changes to the model and the dialog box is left open.

- b. OK is clicked to apply the changes to the model and the Scope parameters: dialog box is close.

24. In the Scope window, it should be pointed to the y-axis as shown in the Figure 3.14, and then right-clicked.

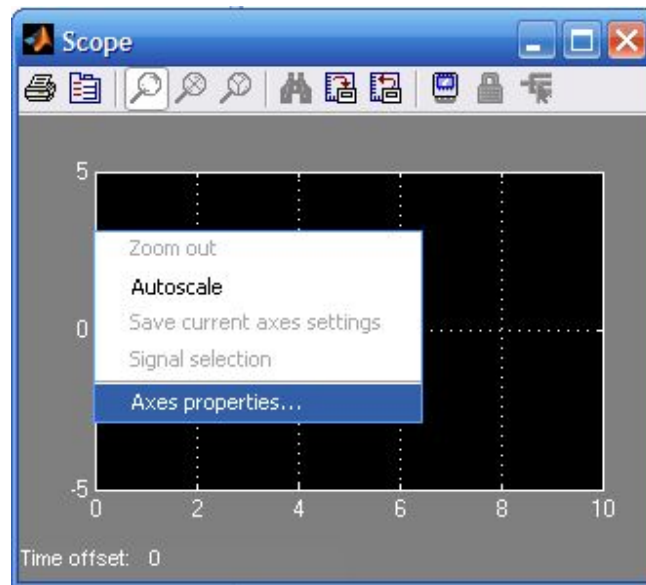


Figure 3.14: Axes properties in scope window

- 25. Axes Properties is clicked from the pop-up menu.
- 26. The Scope Properties: axis 1 dialog box is opened. In the Y-min and Y-max text boxes, the range for the y-axis is entered in the Scope window. For example, -2 and 2 are entered as shown in the Figure 3.15.

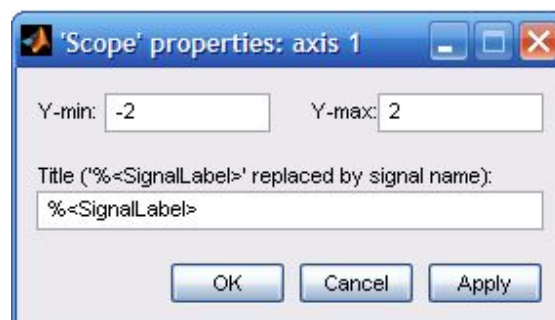


Figure 3.15: Scope Properties: axis 1

- 27. One of the following is done:

- a. Apply is clicked to apply the changes to the model and the dialog box is left open.
- b. OK is clicked to apply the changes to the model and the Axes Parameters: dialog box is closed.

The completed Simulink block diagram is shown in figure 3.16.

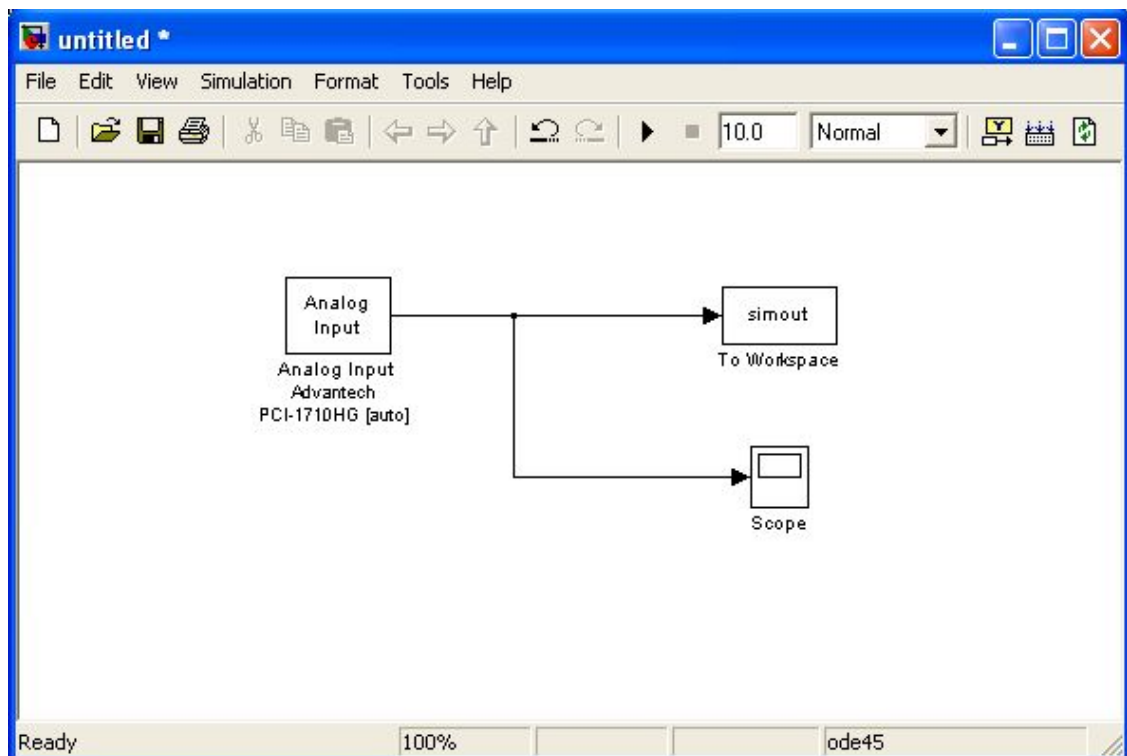


Figure 3.16: Completed Simulink Block Diagram

Save As is clicked from the File menu. The Save As dialog box is opened. In the file name text box, a filename for the Simulink model is entered and Save is clicked. For example, `rtwin_model` is typed
 Simulink saved the model in the file , `rtwin_model.mdl`.

3.5.2.5.2 Entering Configuration Parameters for Simulink

The configuration parameters give information to Simulink for running a simulation. After created a Simulink model, the configuration parameters could be entered for Simulink.

1. In the Simulink window, Configuration Parameters is clicked from the Simulation menu. In the Configuration Parameters dialog box, the Solver tab is clicked.
The Solver pane will open.
2. In the Start Time box, 0 . 0 is entered. In the Stop time box, the amount of time that model needs to run is entered. For example, 10 seconds is entered.
3. From the Type list, Fixed-step is chosen. Real-Time Workshop does not support variable step solvers.
4. From the Solver list, a solver is chosen. For example, the general purpose solver ode5 (Dormand-Prince) is chosen.
5. In the Fixed step size box, a sample time is entered. For example, 0 . 001 seconds is entered for a sample rate of 1000 samples/second.
6. From the Tasking Mode list, singleTasking is chosen. Multitasking is chosen for models with blocks that have different sample times.

The Solver pane would look similar to the Figure 3.17.

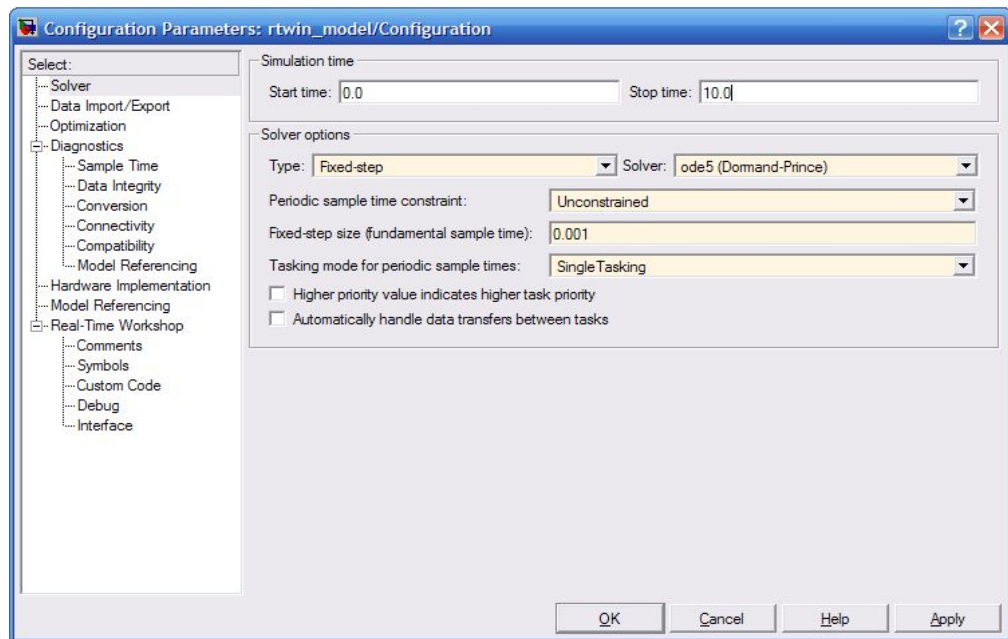


Figure 3.17: Configuration Parameter – Solver

7. One of the following is done:

- Apply is clicked to apply the changes to the model and the dialog box is left open.
- OK is clicked to apply the changes to the model and the Configuration Parameters dialog box is closed.

3.5.2.5.3 Entering Simulation Parameters for Real-Time Workshop

The simulation parameters are used by Real-Time Workshop for generating C code and building a real-time application.

1. In the Simulink window, Configuration Parameters is clicked from the Simulation menu as shown in Figure 3.18.
2. The Hardware Implementation node is clicked.
3. From the Device type list, 32-bit Real-Time Windows Target is chosen.

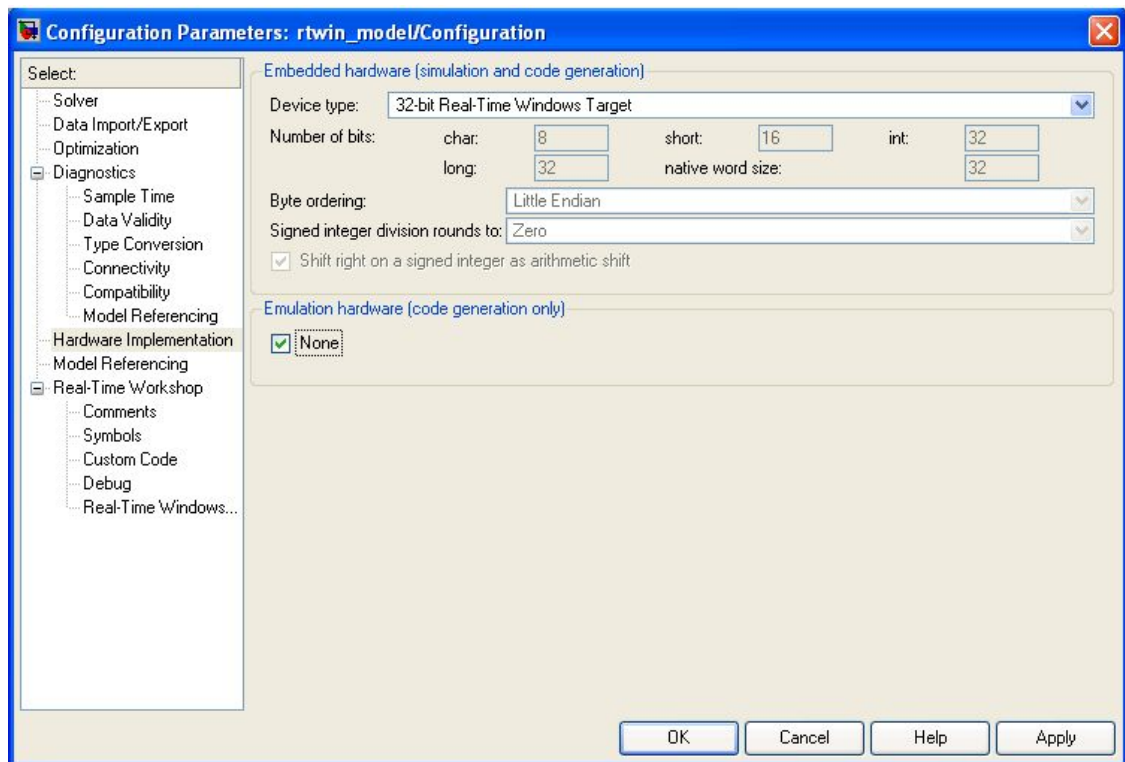


Figure 3.18: Configuration Parameters – Hardware Implementation

4. The Real-Time Workshop node is clicked.
The Real-Time Workshop pane will open.
5. In the Target selection section, the Browse button is clicked at the RTW system target file list. The System Target File Browser will open as shown in Figure 3.19.
6. The system target file is selected for the Real-Time Window Target and OK is clicked.

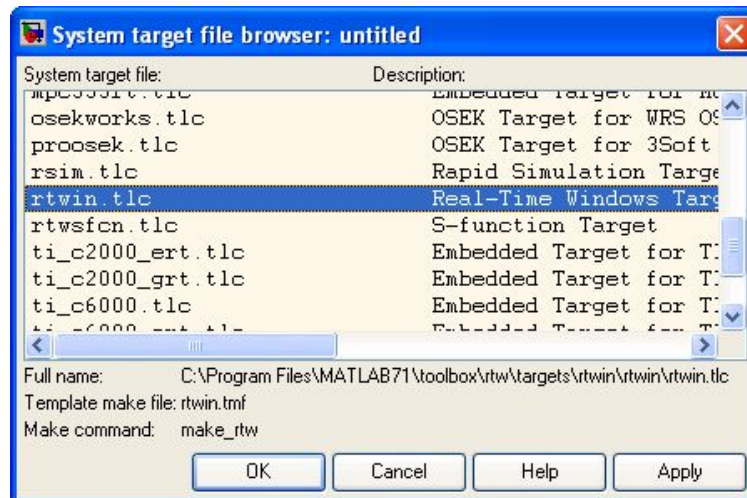


Figure 3.19: System Target File Browser

The system target file `rtwin.tlc`, the template makefile `rtwin.tmf`, and the make command `make_rtw` are automatically entered into the Real-Time Workshop pane. Although not visible in the Real-Time Workshop pane, the external target interface MEX file `rtwinext` is also configured after OK is clicked. This allows external mode to pass new parameters to the real-time application and to return signal data from the real-time application. The data is displayed in Scope blocks or saved with signal logging.

The Real-Time Workshop pane would look similar to the Figure 3.20.

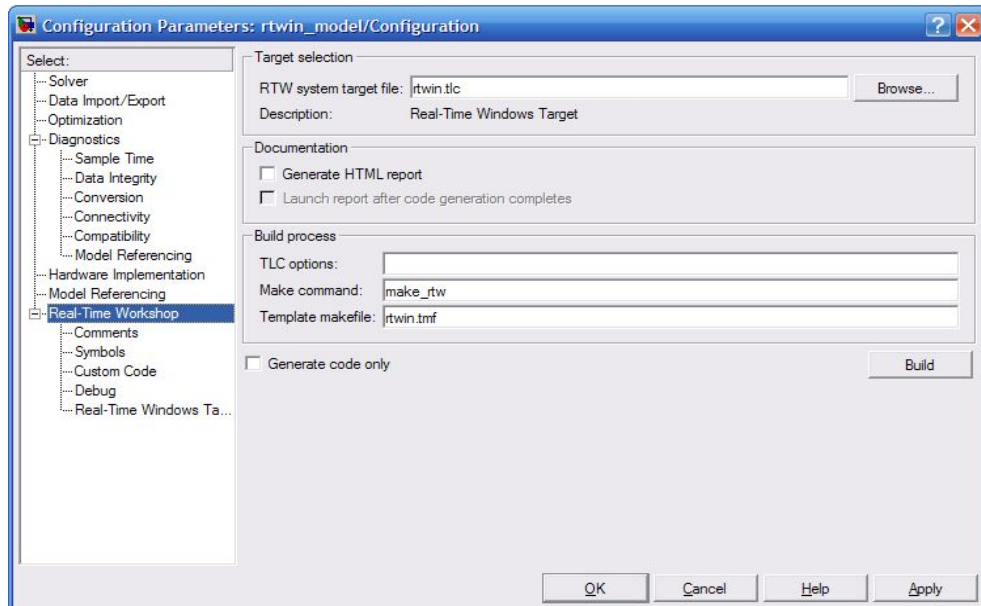


Figure 3.20: Configuration Parameters – Real-Time Workshop

7. After one of the following is done:

- Apply is clicked to apply the changes to the model and the dialog box is left open.
- OK is clicked to apply the changes to the model and the Configuration Parameters dialog box is closed.

3.5.2.5.4 Creating a Real-Time Application

Real-Time Workshop generates C code from the Simulink model, and then the Microsoft Visual Basic C++ compiler and links that C code into a real-time application. After parameters are entered into the Configuration Parameters dialog box for Real-Time Workshop, a real-time application could be built.

1. In the Simulink window, and from the Tools menu, it should be pointed to Real-Time Workshop, and then clicked Build Model. The build process does the following:
 - Real-Time Workshop creates the C code files `rtwin_model.c` and `rtwin_model.h`
 - The make utility `make_rtw.exe` creates the makefile `rtwin_model.mk` from the template makefile `rtwin.tmf`.
 - The make utility `make_rtw.exe` builds the real-time application `rtwin_model.rwd` using the makefile `rtwin_model.mk` created above. The file `rtwin_model.rwd` is binary files that refer to as the real-time application. The real-time application could be run with the Real-Time Windows Target kernel.
2. The Simulink model is connected to the real-time application
 After the real-time application is created, MATLAB could be closed and started again later, and then executable is connected and run without having to rebuild.


3.5.2.5.5 Running a Real-Time Application

The real-time application is run to observe the behavior of the model in real time with the generated code.

The process of connecting consists of

- Establishing a connection between your Simulink model and the kernel to allow exchange of commands, parameters, and logged data.
- Running the application in real time.

After the real-time application is built, the model could be run in real time.

1. From the Simulation menu, External is clicked, and then Connect To Target is connected from the Simulation menu. Also, it could be connected to the target from the toolbar by clicking . It can be seen in figure 3.21.

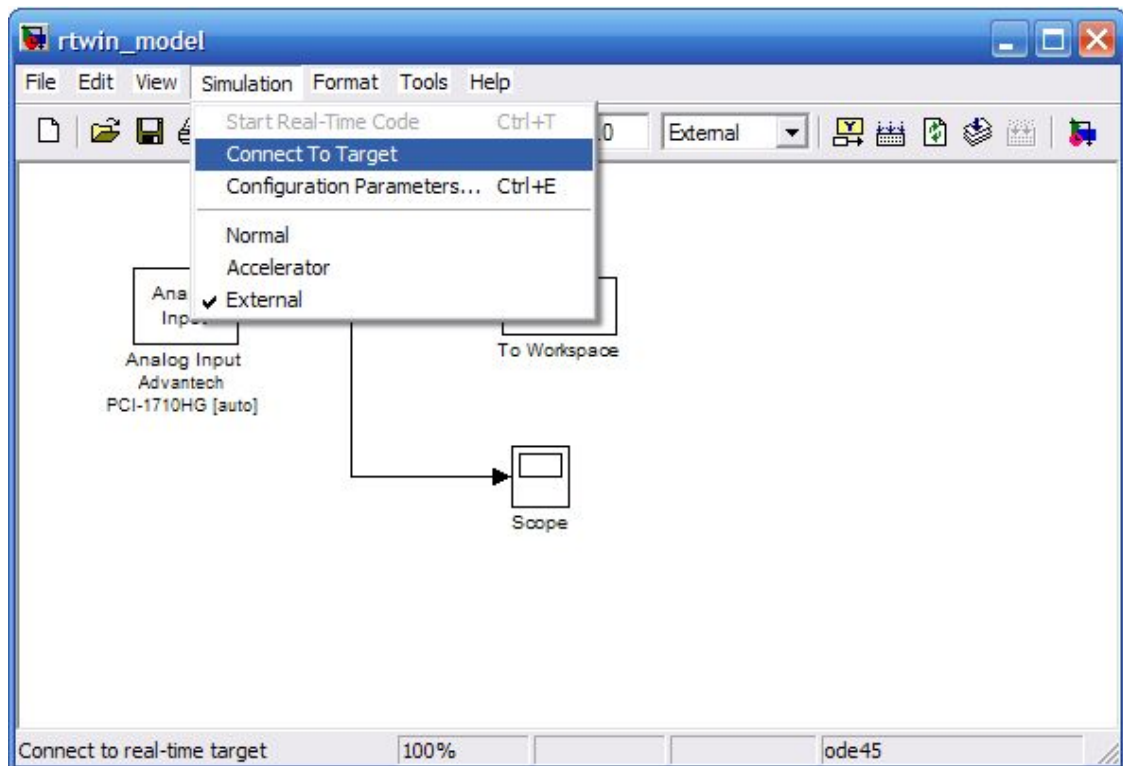


Figure 3.21: Connect to Target from the Simulation menu

MATLAB will display the message

Model rtwin_model loaded

2. In the Simulation window, and from the Simulation menu, Start Real-Time Code is clicked. The execution also could be started from the toolbar. It can be seen in Figure 3.22.

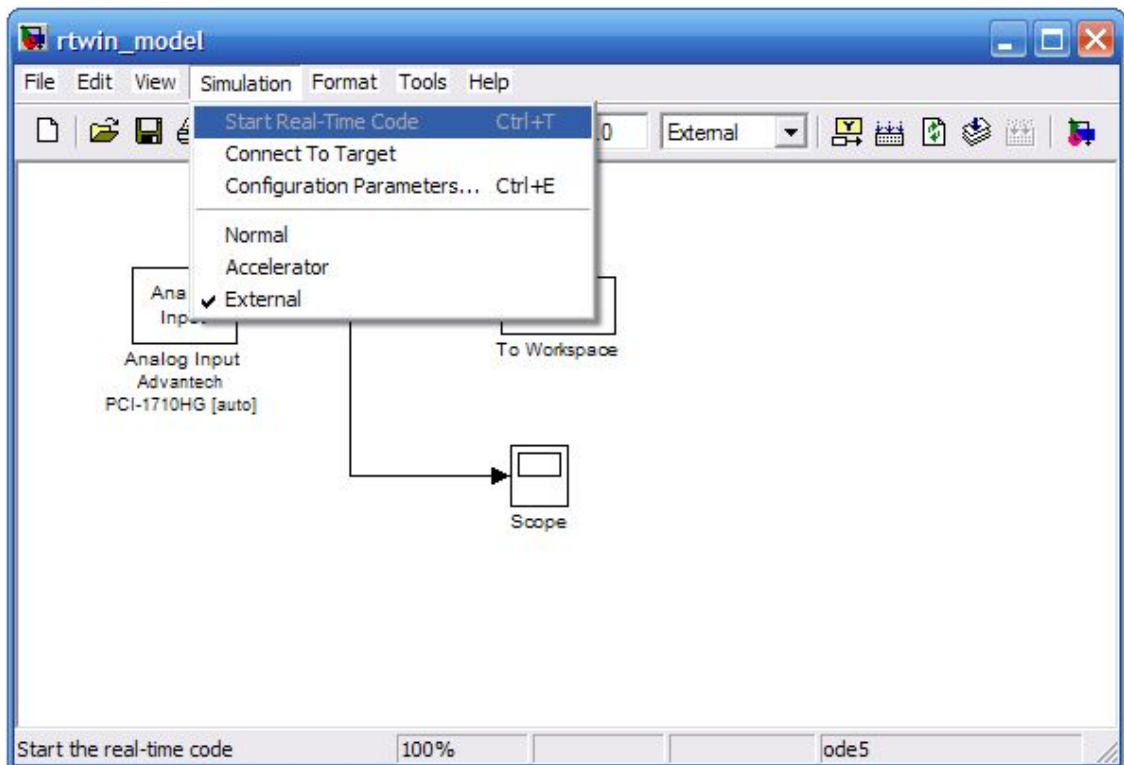


Figure 3.22: Start Real-Time Code from Simulation menu

Simulink runs the execution and plots the signal data in the Scope window.

In the model, the Scope window displays 1000 samples in 1 second, increases the time offset, and then displays the samples for the next 1 second.

Note:

Transfer of data is less critical than calculating the signal outputs at the selected sample interval. Therefore, data transfer runs at lower priority in the remaining CPU time after real-time application computations are performed while waiting for another interrupt to trigger the next real-time application update. The result may be a loss of data points displayed in the Scope window.

3. One of the following is done:

- The executable is let to be run until it reaches the stop time.
- Stop Real-time Code is clicked from the Simulation menu.

The real-time application is stopped.

4. In the Simulation window, Disconnect From Target is clicked from the Simulation menu.
5. From the Simulation menu, external is clicked
MATLAB will display the message
`Model rtwin_model unloaded`

CHAPTER IV

SOFTWARE DEVELOPMENT

4.1 Introduction

This chapter will discuss about the software that has been develop using MATLAB. The system designed will be proposed to be used by students that taking the subject Industrial Instrumentations, BEE4523.

4.2 Software Development

Using hardware that has been introduced earlier, this system can be used to do the study analysis. The system is build using MATLAB GUI (graphical user interface) and can fulfill students' need while conducting laboratory experimentation. The flowchart of the system is shown below in Figure 4.1.

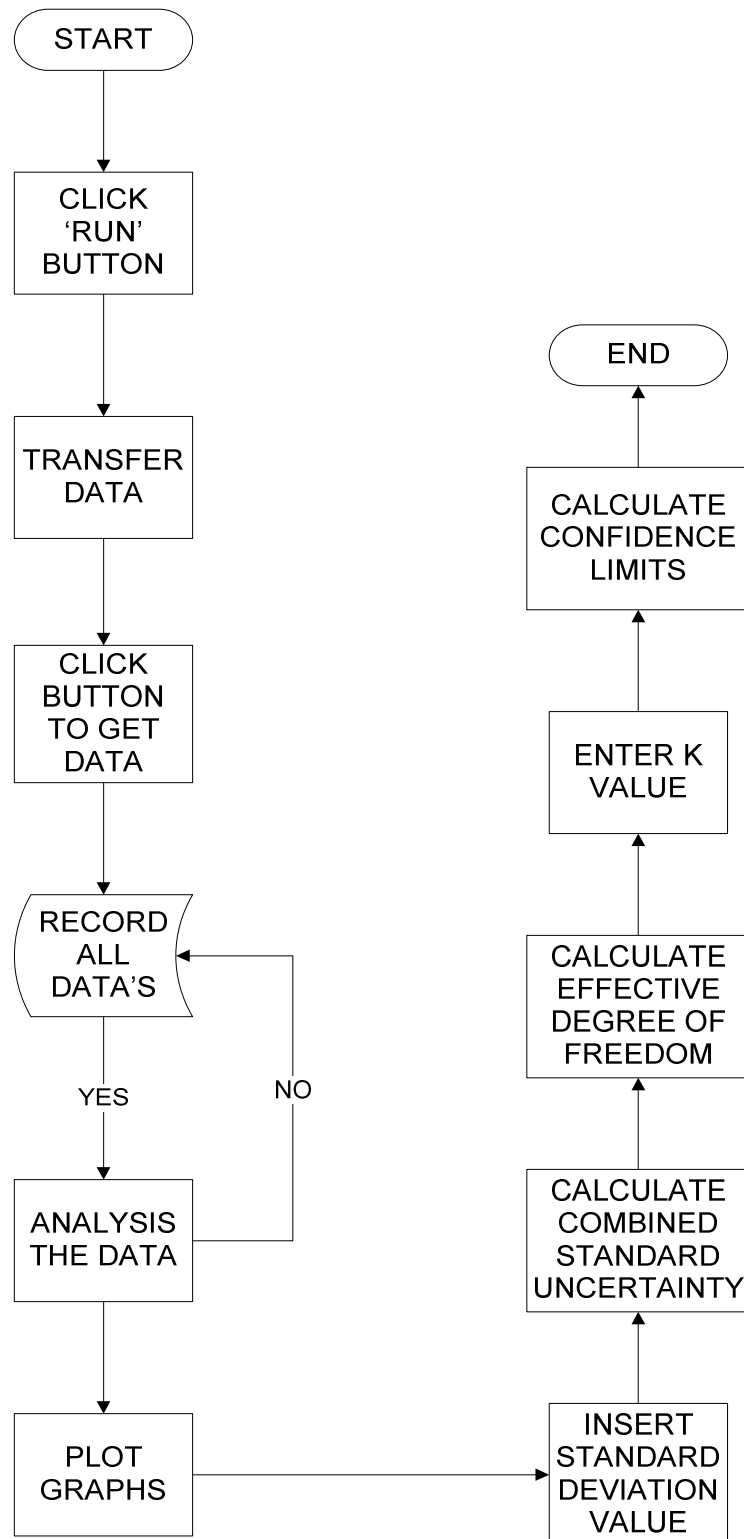


Figure 4.1: Flowchart of the System

From the experiment, the data will be transferred using the hardware to the system by online. After run the system, button '1.1' until button '3.5' must be clicked so that the system will capture the data from the DAQ card. When the system received the data, it will be recorded and saved. After all the data have been collected, the study analysis of the data can be done.

In this system, user must click the 'convert data' first to convert the reading from the voltage to mili Ampere. After that, they can click 'analyze data' button to calculate the average of the data that have been recorded, calculate the output error that compared the desired output and the actual output from the experiment and the standard deviation of the data collected. The answer will be shown in the space provided. The average and output error are needed to plot the graph for five-point calibration and graph for output error. The graph will be shown after the 'plot graph calibration' or 'plot graph error' is clicked. The graph can be saved by user if they click the save plot button.

The standard deviation is calculated to find the uncertainty due to the repeatability of the experiment, U_1 . The calculation is shown because the user must choose the worst case standard deviation. After user key in the worst case standard deviation, the system can calculate u_c , combined standard uncertainty. Beside the uncertainty due to the repeatability of the experiment, there are three other factors that also considered. They are uncertainty contribution due to MSU error, U_2 , uncertainty due to UUT resolution/MSU resolution, U_3 , and uncertainty due to previous calibration, U_4 . The answer of combined standard uncertainty will be shown in the space below of the button.

To complete the study analysis of the data, the effective degree of freedom, V_e also calculated. The formula involved the calculation of combined standard uncertainty, are U_1 , U_2 , U_3 , and U_4 . Degree of freedom for U_1 is 2 while for the others are infinity.

For the first time users, they will get the answer more than 100 because there is no uncertainty due to previous calibration.

Lastly, the confidence limits are calculated. This will need a table that is provided in the system. User must choose the fraction p in percent (68.27, 90.00, 95.00, 95.45, 99.00, and 99.73) that will determine the confidence interval, k value. The unit of confidence limits is kPa. This system is proposed to the student that taking the BEE4523, Industrial Instrumentation subject, so that they will do the experiment once which means the table of degree of freedom needed is only for 100 and above.

There is a close button on a right sight, bottom of the GUI window that will ask a confirmation when user want to close the window. The GUI window is shown in the Figure 4.2.

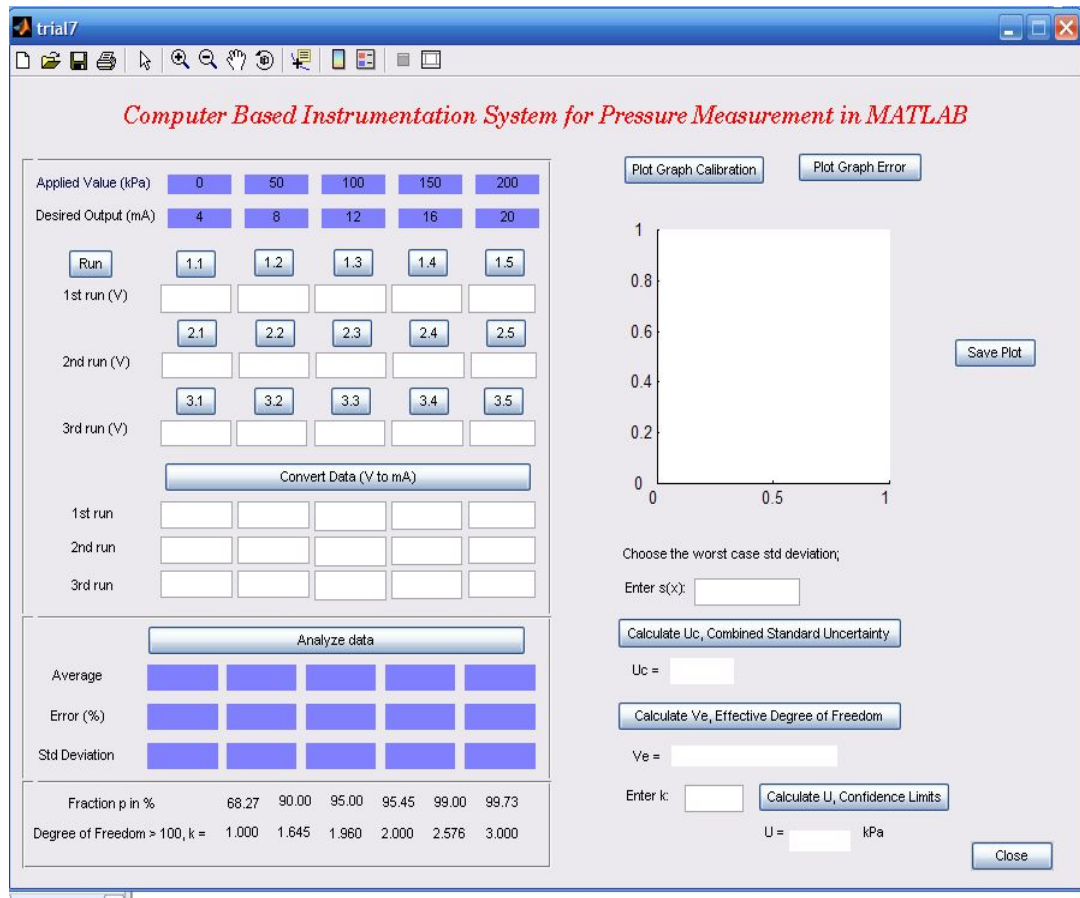


Figure 4.2: GUI

Table below show the formulas that have been use in the system to do the study analysis of the data.

Table 4.1: Table of formulas

No	Name	Formula
1	Standard Deviation	$s(x_k) = \sqrt{\frac{1}{(n-1)} \sum_{j=1}^k (x_k - \bar{x})^2}$
2	Combined Standard Uncertainty	$u_c = \sqrt{u_1^2 + u_2^2 + u_3^2}$
3	Effective Degree of Freedom	$V_e = \frac{u_c^4}{\frac{u_1^4}{v_1} + \frac{u_2^4}{v_2} + \frac{u_3^4}{v_3} + \frac{u_4^4}{v_4}}$
4	Confidence Limits	$u = u_c k$

CHAPTER V

RESULTS AND DISCUSSION

5.1 Introduction

This chapter is explained about the results that have been achieved from this project. The result from the experiment and the analysis of the data is shown in this part.

5.2 Results

An experiment has been conducted to get the measured or actual output of the pressure transmitter. The data collected are recorded in the table below. The average is calculated in order to plot the graph of the calibrations. The output error also calculated to get the graph error.

Table 5.1: Data of Pressure Measurement

No (%)	MSU applied value (kPa)	Desired UUT output (mA)	Actual UUT output (mA)				Output error (%)
			1 st run (mA)	2 nd run (mA)	3 rd run (mA)	Average (mA)	
0	0	4.00	4.006	4.007	4.005	4.006	0.150
25	50	8.00	8.038	8.038	8.035	8.037	0.463
50	100	12.00	12.060	12.063	12.006	12.043	0.358
75	150	16.00	16.092	16.094	16.094	16.093	0.581
100	200	20.00	20.120	20.121	20.120	20.120	0.600

The results of study analysis are shown in Figure 5.1. The first button need to be clicked is 'Run' button. The button '1.1' until '3.5' provided to get the data from the DAQ card. After column for the first run, second run and third run are filled, they are converted to the current, mA. The 'analyze data' button is clicked to get average, output error and standard deviation. The answer will appear in the space provided. From these data, the graph is plotted. The graphs are also shown in the GUI window. After user entered the worst case of standard deviation which is the largest value, combined standard uncertainty is calculated and then followed by effective degree of freedom. The value of confidence interval, k is determined by user. Lastly, the confidence limit of the data is calculated.

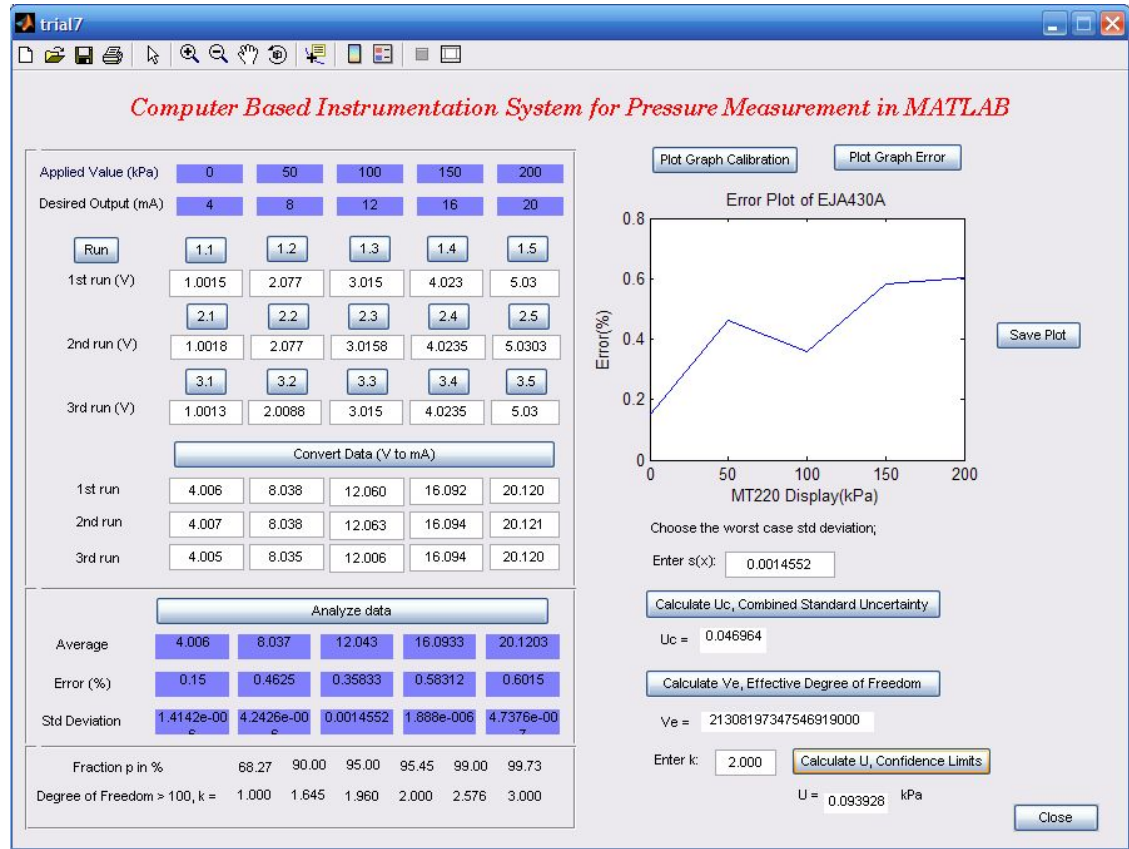


Figure 5.1: Results and analysis

5.3 Discussions

Generally, this project is succeeded and the objectives are achieved. However, there are some problems while doing this project such as conducting the experiment, problems with instruments and the hardware.

The equipment must be complete to conduct the experiment but sometimes facing problems when the instruments can not be used. Beside that, the hardware PCI-1710HG provided by the laboratory is limited. Sharing the DAQ cards give some difficulty because needs to take turns to use it. The computer at the laboratory that has been installed the DAQ also limited so that the project progressed is slow when need to use the computer.

For the software designed, there are problems earlier but has been overcome after find the solution to capture the data from MATLAB workspace to the GUI. The buttons in the GUI also too much that make it seems not user friendly. User must click the button one by one to get the data from the DAQ. The way to reduce the buttons is using the instrument that can memorize the reading of each pressure measured.

5.4 Costing and Commercialization

This project does not involve any cost. The instruments are provided in the laboratory and the data acquisition card is bough by the lab.

One of the objectives of this project is to develop software that can be implemented in laboratory session for Industrial Instrumentation course, BEE4523, which helps students in study analysis. Therefore, this project will be used as a tool for analysis in students' laboratory session. This project can not be used for another commercialization purpose.

CHAPTER VI

CONCLUSION

6.1 Conclusion

This project is including hardware and software design. The hardware part consists of the instruments that provided in the laboratory and PCI-1710HG. This PCI is used to interface between the instruments and the system in the computer. The system is developing using MATLAB graphical user interface. This system will do the study analysis of the data. The results will display on the GUI itself. The results include average, output error, graphs and uncertainty.

This project that is creating to help students in their laboratory session will be succeeded. They can use the proposed system to do the study analysis that easier and faster to finish their lab. This system will benefit not only the students but also for the instructor.

6.2 Recommendations

There is no boundary to explore new things. In future, this system can be improved so that students can do their work easier. By using this system the accuracy of the output is more convinced. The experimentation can be done only in few minutes and the analysis will also taking less than they can do now.

APPENDIX A

Value of $t_p(\nu)$ from the t-distribution for degree of freedom ν that defines an Interval $-t_p(\nu)$ to $+t_p(\nu)$ that encompasses the fraction p of the distribution.

Degree of Freedom ν	Fraction p in percent					
	68.72*	90.00	95.00	95.45	99.00	99.73*
1	1.84	6.31	12.71	13.97	63.66	235.8
2	1.32	2.92	4.30	4.53	9.92	19.21
3	1.20	2.35	3.18	3.31	5.84	9.22
4	1.14	2.13	2.78	2.87	4.60	6.62
5	1.11	2.02	2.57	2.65	4.03	5.51
6	1.09	1.94	2.45	2.52	3.71	4.90
7	1.08	1.89	2.36	2.43	3.50	4.53
8	1.07	1.86	2.31	2.37	3.36	4.28
9	1.06	1.83	2.26	2.32	3.25	4.09
10	1.05	1.81	2.23	2.28	3.17	3.96
11	1.05	1.80	2.20	2.25	3.11	3.85
12	1.04	1.78	2.18	2.23	3.05	3.76
13	1.04	1.77	2.16	2.21	3.01	3.69
14	1.04	1.76	2.14	2.20	2.98	3.64
15	1.03	1.75	2.13	2.18	2.95	3.59
16	1.03	1.75	2.12	2.17	2.92	3.54
17	1.03	1.74	2.11	2.16	2.90	3.51
18	1.03	1.73	2.10	2.15	2.88	3.48
19	1.03	1.73	2.09	2.14	2.86	3.45
20	1.03	1.72	2.09	2.13	2.85	3.42
25	1.02	1.71	2.06	2.11	2.79	3.33
30	1.02	1.70	2.04	2.09	2.75	3.27
35	1.01	1.70	2.03	2.07	2.72	3.23
40	1.01	1.68	2.02	2.06	2.70	3.20
45	1.01	1.68	2.01	2.06	2.69	3.18
50	1.01	1.68	2.01	2.05	2.68	3.16
100	1.005	1.660	1.984	2.025	2.626	3.007
	1.000	1.645	1.960	2.000	2.567	3.000

*For a quantity Z described by a normal distribution with expectation μ_z and standard deviation σ , the interval $\mu_z \pm k\sigma$ encompasses $p = 68.27, 95.45$ AND 99.73 percent of the distribution for $k = 1, 2$ and 3 respectively.

APPENDIX B

Coding of the MATLAB programming

```

function varargout = trial7(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',    mfilename, ...
    'gui_Singleton', gui_Singleton, ...
    'gui_OpeningFcn', @trial7_OpeningFcn, ...
    'gui_OutputFcn', @trial7_OutputFcn, ...
    'gui_LayoutFcn', [] , ...
    'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

handles.output = hObject;
set(hObject,'toolbar','figure');
guidata(hObject, handles);
function varargout = trial7_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function edit1_Callback(hObject, eventdata, handles)
user_entry = str2double(get(hObject,'string'));
if isnan(user_entry)
    errordlg('You must enter a numeric value','Wrong Input','modal')
end

if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function edit2_Callback(hObject, eventdata, handles)
user_entry = str2double(get(hObject,'string'));
if isnan(user_entry)
    errordlg('You must enter a numeric value','Wrong Input','modal')
end

```

```
function edit2_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function edit3_Callback(hObject, eventdata, handles)
user_entry = str2double(get(hObject,'string'));
if isnan(user_entry)
    errordlg('You must enter a numeric value','Wrong Input','modal')
end
```

```
function edit3_CreateFcn(hObject, eventdata, handles)

if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function edit4_Callback(hObject, eventdata, handles)
user_entry = str2double(get(hObject,'string'));
if isnan(user_entry)
    errordlg('You must enter a numeric value','Wrong Input','modal')
end
```

```
function edit4_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function edit5_Callback(hObject, eventdata, handles)
user_entry = str2double(get(hObject,'string'));
if isnan(user_entry)
    errordlg('You must enter a numeric value','Wrong Input','modal')
end
```

```
function edit5_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function edit6_Callback(hObject, eventdata, handles)
user_entry = str2double(get(hObject,'string'));
if isnan(user_entry)
    errordlg('You must enter a numeric value','Wrong Input','modal')
end
```

```
function edit6_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function edit8_Callback(hObject, eventdata, handles)
user_entry = str2double(get(hObject,'string'));
if isnan(user_entry)
    errordlg('You must enter a numeric value','Wrong Input','modal')
end
```

```
function edit8_CreateFcn(hObject, eventdata, handles)

if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function edit9_Callback(hObject, eventdata, handles)
user_entry = str2double(get(hObject,'string'));
if isnan(user_entry)
    errordlg('You must enter a numeric value','Wrong Input','modal')
end
```

```
function edit9_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function edit10_Callback(hObject, eventdata, handles)
user_entry = str2double(get(hObject,'string'));
if isnan(user_entry)
    errordlg('You must enter a numeric value','Wrong Input','modal')
end
```

```
function edit10_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function edit11_Callback(hObject, eventdata, handles)
user_entry = str2double(get(hObject,'string'));
if isnan(user_entry)
    errordlg('You must enter a numeric value','Wrong Input','modal')
end
```

```
function edit11_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function edit15_Callback(hObject, eventdata, handles)
user_entry = str2double(get(hObject,'string'));
if isnan(user_entry)
    errordlg('You must enter a numeric value','Wrong Input','modal')
end
```

```
function edit15_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function edit12_Callback(hObject, eventdata, handles)
user_entry = str2double(get(hObject,'string'));
if isnan(user_entry)
    errordlg('You must enter a numeric value','Wrong Input','modal')
end
```

```
function edit12_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```

function edit14_Callback(hObject, eventdata, handles)
user_entry = str2double(get(hObject,'string'));
if isnan(user_entry)
    errordlg('You must enter a numeric value','Wrong Input','modal')
end

function edit14_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function analysis_Callback(hObject, eventdata, handles)

a = get(handles.edit1,'String');
b = get(handles.edit2,'String');
c = get(handles.edit3,'String');
d = get(handles.edit4,'String');
e = get(handles.edit5,'String');
f = get(handles.edit6,'String');
g = get(handles.edit7,'String');
h = get(handles.edit8,'String');
i = get(handles.edit9,'String');
j = get(handles.edit10,'String');
k = get(handles.edit11,'String');

ave1 = (str2num(a)+str2num(f)+str2num(k))/3;
p = num2str(ave1);

ave2 = (str2num(b)+str2num(g)+str2num(l))/3;
q = num2str(ave2);

ave3 = (str2num(c)+str2num(h)+str2num(m))/3;
r = num2str(ave3);

ave4 = (str2num(d)+str2num(i)+str2num(n))/3;
s = num2str(ave4);

set(handles.ave1,'String',p);
set(handles.ave2,'String',q);
set(handles.ave3,'String',r);
set(handles.ave4,'String',s);
set(handles.ave5,'String',t);
guidata(hObject,handles);

```



```

u = get(handles.ave1,'String');
v = get(handles.ave2,'String');
w = get(handles.ave3,'String');

er1 = ((str2num(u)-4.000)/4.000)*100;
ea = num2str(er1);

er2 = ((str2num(v)-8.000)/8.000)*100;
eb = num2str(er2);

er3 = ((str2num(w)-12.000)/12.000)*100;
ec = num2str(er3);

er4 = ((str2num(aa)-16.000)/16.000)*100;
ed = num2str(er4);

set(handles.er1,'String',ea);
set(handles.er2,'String',eb);
set(handles.er3,'String',ec);
set(handles.er4,'String',ed);
set(handles.er5,'String',ee);
guidata(hObject,handles);

io1 = get(handles.ave1,'String');
io2 = get(handles.ave2,'String');
io3 = get(handles.ave3,'String');
io4 = get(handles.ave4,'String');
io5 = get(handles.ave5,'String');

sd1=(sqrt(0.5))*((((str2num(a))-(str2num(io1))).^2)+(((str2num(f))-
(str2num(io1))).^2)+(((str2num(k))-(str2num(io1))).^2));
sd1a = num2str(sd1);

sd2=(sqrt(0.5))*((((str2num(b))-(str2num(io2))).^2)+(((str2num(g))-
(str2num(io2))).^2)+(((str2num(l))-(str2num(io2))).^2));
sd2a = num2str(sd2);

sd3=(sqrt(0.5))*((((str2num(c))-(str2num(io3))).^2)+(((str2num(h))-
(str2num(io3))).^2)+(((str2num(m))-(str2num(io3))).^2));
sd3a = num2str(sd3);

sd4=(sqrt(0.5))*((((str2num(d))-(str2num(io4))).^2)+(((str2num(i))-
(str2num(io4))).^2)+(((str2num(n))-(str2num(io4))).^2));
sd4a = num2str(sd4);

set(handles.sd1,'String',sd1a);

```

```

set(handles.sd2,'String',sd2a);
set(handles.sd3,'String',sd3a);
set(handles.sd4,'String',sd4a);
set(handles.sd5,'String',sd5a);
guidata(hObject,handles);

sdb = get(handles.edit16,'String'); %uncertainty due to the repeatability of the
experiment
sdc = str2num(sdb)/(sqrt(3)); %u1 (kPa)

%error in MSU = (0.0001*200)+(0.00005*700)
%maxer = 0.055*(10^3)
%u2 = maxer/(sqrt 3)

u2 = 0.03175; %kPa

a = get(handles.ave1,'String');
b = get(handles.ave2,'String');
c = get(handles.ave3,'String');
d = get(handles.ave4,'String');
e = get(handles.ave5,'String');

res1 = (str2num(a)-4);
res2 = (str2num(b)-8);
res3 = (str2num(c)-12);
res4 = (str2num(d)-16);
res5 = (str2num(e)-20);
resolution = ((res1+res2+res3+res4+res5)/5);

u3 = resolution/sqrt(3)

uc = sqrt ((sdc.^2)+(u2.^2)+(u3.^2));

uc1 = num2str(uc);

set(handles.uc,'String',uc1);

function calcVeCallback(hObject, eventdata, handles)

z = get(handles.uc,'String'); %uc

a = get(handles.edit16,'String'); b = (str2num(a)/(sqrt(3)));
v1 = 2;
ve = ((z.^4)/((b.^4)/v1));
ve1 = num2str(ve);
set(handles.ve,'String',ve1);

```

```
function edit16Callback(hObject, eventdata, handles)
user_entry = str2double(get(hObject,'string'));
if isnan(user_entry)
    errordlg('You must enter a numeric value','Wrong Input','modal')
end
```

```
function edit16_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function calcConf_Callback(hObject, eventdata, handles)
```

```
a = get(handles.'String');
b = get(handles.edit17,'String');

c = (str2num(a))*(str2num(b));
u = num2str(c);
```

```
set(handles.U,'String',u);
guidata(hObject,handles);
```

```
function edit17_Callback(hObject, eventdata, handles)
```

```
user_entry = str2double(get(hObject,'string'));
if isnan(user_entry)
    errordlg('You must enter a numeric value','Wrong Input','modal')
end
```

```
function edit17_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function gError_Callback(hObject, eventdata, handles)
```

```
f1 = str2double(get(handles.er1,'String'));
f2 = str2double(get(handles.er2,'String'));
f3 = str2double(get(handles.er3,'String'));
f4 = str2double(get(handles.er4,'String'));
f5 = str2double(get(handles.er5,'String'));
```

```

axes(handles.axes1)
x = [0 50 100 150 200];
y = [f1 f2 f3 f4 f5];

plot(x,y);
title('Error Plot of EJA430A');
xlabel('MT220 Display(kPa)');
ylabel('Error(%)');
guidata(hObject, handles); %updates the handles

function gCalib_Callback(hObject, eventdata, handles)

h1 = str2double(get(handles.ave1,'String'));
h2 = str2double(get(handles.ave2,'String'));
h3 = str2double(get(handles.ave3,'String'));
h4 = str2double(get(handles.ave4,'String'));
h5 = str2double(get(handles.ave5,'String'));

axes(handles.axes1)
x = [0 50 100 150 200];
y = [h1 h2 h3 h4 h5];
plot(x,y);
title('5-Point Calibration of EJA430A');
xlabel('MT220 Display(kPa)');
ylabel('EJX110A Output(mA)');
guidata(hObject, handles); %updates the handles

function close_Callback(hObject, eventdata, handles)
selection = quest('Do you want to close?',...
    'Close Request',...
    'Yes','No','Yes');
switch selection,
    case 'Yes',
        delete(gcf)
    case 'No'
        return
end

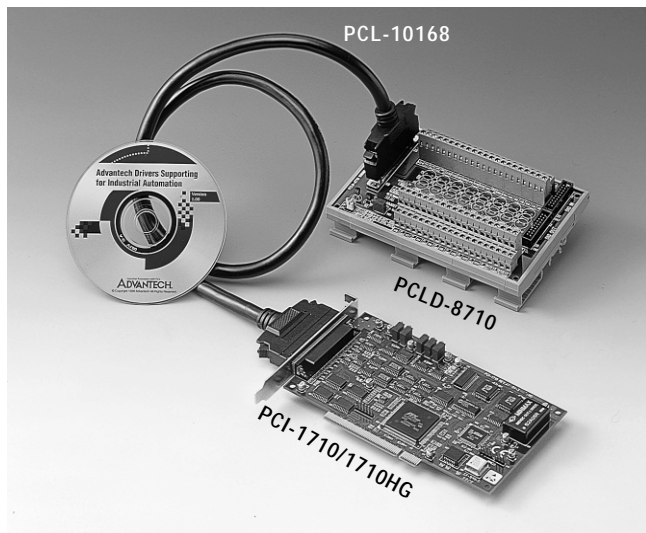
function saveplot_Callback(hObject, eventdata, handles)

savePlotWithinGUI(handles.axes1)

```

PCI-1710 PCI-1710HG

**12-bit, 100 kHz, (High-gain),
PCI-bus Multi-function DAS Card**



Introduction

The PCI-1710/1710HG is a multifunction DAS card for the PCI bus. Its advanced circuit design provides higher quality and more functions, including the five most desired measurement and control functions: 12-bit A/D conversion, D/A conversion, digital input, digital output, and counter/timer.

Mixed Single-ended or Differential Analog Inputs

The PCI-1710/1710HG features an automatic channel/gain scanning circuit. The circuit, rather than your software, controls multiplexer switching during sampling. The on-board SRAM stores different gain values and configurations for each channel. This design lets you perform multi-channel high-speed sampling (up to 100 kHz) with different gains for each channel and with free combination of single-ended and differential inputs.

On-board FIFO (First In First Out) Memory

The PCI-1710/1710HG has an on-board FIFO buffer which can store up to 4K A/D samples. The PCI-1710/1710HG generates an interrupt when the FIFO is half full. This feature provides continuous high-speed data transfer and more predictable performance on Windows systems.

On-board Programmable Counter

The PCI-1710/1710HG provides a programmable counter for generating a pacer trigger for the A/D conversion. The counter chip is an 82C54 or equivalent, which includes three 16-bit counters on a 10 MHz clock. One counter is used as an event counter for counting events coming from the input channels. The other two are cascaded together to make a 32-bit timer for a pacer trigger.

Features

- 16 single-ended or 8 differential analog inputs, or a combination
- 12-bit A/D converter, with up to 100 kHz sampling rate
- Programmable gain for each input channel
- Free combination of single-ended and differential inputs
- On-board 4 K samples FIFO buffer
- Two 12-bit analog output channels
- 16 digital inputs and 16 digital outputs
- Programmable pacer/counter

Special Shielded Cable for Noise Reduction

The PCL-10168 shielded cable is specially designed for the PCI-1710/1710HG for reducing noise in the analog signal lines. Its wires are all twisted pairs, and the analog lines and digital lines are separately shielded, providing minimal cross talk between signals and the best protection against EMI/EMC problems.

Specifications

Analog Input:

- **Channels:** 16 single-ended or 8 differential (software programmable)
- **Resolution:** 12-bit
- **On-board FIFO:** 4 K samples
- **Conversion time:** 8 ms
- **Input range:**(V, software programmable)

	PCI-1710	PCI-1710HG
Bipolar	$\pm 10, \pm 5, \pm 2.5, \pm 1.25, \pm 0.625$	$\pm 10, \pm 5, \pm 1, \pm 0.5, \pm 0.1, \pm 0.05, \pm 0.01, \pm 0.005$
Unipolar	$0 \sim 10, 0 \sim 5, 0 \sim 2.5, 0 \sim 1.25$	$0 \sim 10, 0 \sim 1, 0 \sim 0.1, 0 \sim 0.01$

- **Maximum Input Overvoltage:** ± 30 V
- **Common Mode Rejection Ratio (CMRR):**

PCI-1710		PCI-1710HG	
Gain	CMRR	Gain	CMRR
0.5, 1	75dB	0.5, 1	75dB
2	80dB	10	90dB
4	84dB	100	106dB
8	84dB	1000	106dB

12-bit, 100 kHz, (High-gain), PCI-bus Multi-function DAS Card

- Maximum A/D data throughput: (Hz, depending on PGIA settling time)

PCI-1710: 100 k

PCI-1710HG:

Gain	Speed
0.5, 1	100k
5, 10	35k
50, 100	7k
500, 1000	770

- Accuracy: (Depends on gain)

PCI-1710		PCI-1710HG		Remark
Gain	Accuracy	Gain	Accuracy	
0.5, 1	0.01% of FSR ± 1 LSB	0.5, 1	0.01% of FSR ± 1 LSB	S.E./D
2	0.02% of FSR ± 1 LSB	5, 10	0.02% of FSR ± 1 LSB	S.E./D
4	0.02% of FSR ± 1 LSB	50, 100	0.04% of FSR ± 1 LSB	D
8	0.04% of FSR ± 1 LSB	500, 1000	0.08% of FSR ± 1 LSB	D

* S.E.: Single-ended D: Differential

- Linearity error: ± 1 LSB
- Input impedance: 1 GW
- Trigger mode: Software, on-board programmable pacer or external

Analog Output:

- Channels: 2
- Resolution: 12-bit
- Relative accuracy: $\pm 1/2$ LSB
- Gain error: ± 1 LSB
- Maximum update rate: 100 K samples / sec (polling)
- Slew rate: 10 V / μ s
- Output range (software programmable):
Internal reference: 0 ~ +5 V @ -5 V,
0 ~ +10 V @ -10 V
External reference: 0 ~ +x V @ -x V ($-10 \leq x \leq 10$)

Digital Input:

- Channels: 16
- Input voltage:
Low: 0.4 V max.
High: 2.4 V min.
- Input load:
Low: -0.2 mA @ 0.4 V
High: 20 mA @ 2.7 V

Digital Output:

- Channels: 16
- Output voltage:
Low: 0.4 V max. @ 8.0 mA (sink)
High: 2.4 V min. @ -0.4 mA (source)

Programmable Timer/Counter

- Counter chip: 82C54 or equivalent
- Counters: 3 channels, 16 bits, 2 channels are permanently configured as a 32-bit programmable pacer; 1 channel is free for user applications
- Input, gate: TTL/CMOS compatible
- Time base:
Channel 1: 10 MHz
Channel 2: Takes input from output of channel 1
Channel 0: Internal 1 MHz or external clock (10 MHz max.) selected by software.

General:

- CE certified to CISPR 22 class B
- I/O Connector: 68-pin SCSI-II female connector
- Power consumption: +5 V @ 850 mA (Typical),
+5 V @ 1.0 A (Max.)
- Operating temperature: 0 ~ +60° C (32 ~ 140° F) (refer to IEC 68-2-1, 2)
- Storage temperature: -20 ~ +70° C (-4 ~ 158° F)
- Operating humidity: 5 ~ 95% RH non-condensing (refer to IEC 68-2-3)
- Dimensions: 175 mm (L) x 100 mm (H) (6.9" x 3.9")
- MTBF: over 64,770 hrs @ 25° C, grounded-fix environment

Pin Assignments

A10	68	34	A11
A12	67	33	A13
A14	66	32	A15
A16	65	31	A17
A18	64	30	A19
A110	63	29	A111
A112	62	28	A113
A114	61	27	A115
AIEND	60	26	AIEND
DA0_REF	59	25	DA1_REF
DA0_OUT	58	24	DA1_OUT
AGND	57	23	AGND
DI0	56	22	DI1
DI2	55	21	DI3
DI4	54	20	DI5
DI6	53	19	DI7
DI8	52	18	DI9
DI10	51	17	DI11
DI12	50	16	DI13
DI14	49	15	DI15
DOEND	48	14	DOEND
DO0	47	13	DO1
DO2	46	12	DO3
DO4	45	11	DO5
DO6	44	10	DO7
DO8	43	9	DO9
DO10	42	8	DO11
DO12	41	7	DO13
DO14	40	6	DO15
DOEND	39	5	DOEND
CNT0_CLK	38	4	PACER_OUT
CNT0_OUT	37	3	TRG_GATE
CNT0_GATE	36	2	EXT_TRG
+12V	35	1	+5V

Ordering information

- PCI-1710: 12-bit, 100 kHz, PCI-bus Multifunction DAS Card, user's manual and driver CD-ROM. (cable not included)
- PCI-1710HG: 12-bit, 100 kHz, High-gain, PCI-bus Multifunction DAS Card, user's manual and driver CD-ROM. (cable not included)
- PCLD-8710: Wiring Terminal Board with CJC circuit (cable not included)
- PCL-10168: 68-pin SCSI-II cable with male connectors on both ends and special shielding for noise reduction, 1m.
- ADAM-3968: 68-pin SCSI-II Wiring Terminal Board for DIN-Rail Mounting

REFERENCE

- [1] <http://www.pcmag.com/>
- [2] http://en.wikipedia.org/wiki/Literature_review
- [3] http://en.wikipedia.org/wiki/Pressure_measurement
- [4] <http://www.sensorone.co.uk/news.html>
- [5] S K Singh, Industrial Instrumentation and Control, Second Edition, McGraw-Hill
- [6] William Dunn, Chapter Thirteen, Fundamental of Industrial Instrumentation and Process Control, McGraw-Hill, 2005
- [7] Data Acquisition Systems (DAS) in General, Reproduced from the Handbook of Measuring System Design, 2005
- [8] James W. Dally, William F. Riley, Kenneth G. McConnell Instrumentation for Engineering Measurements, Second Edition
- [9] Hsin-Ti Chueh, John V. Hatfield, A Real-Time Data Acquisition System for a Hand-held Electronic Nose, 2002

- [10] Farrar, R. (2004). Newport Corporation Reduces Data Acquisition and Analysis Time by Hundred of Hours.
<https://tagteamdbserver.mathworks.com/ttservererroot/>
- [11] Young Jin Choi, Kathryn L. McCarty, Micheal J. McCarty, A MATLAB Graphical User Interface Program for Tomographic Viscometer Data Processing, 2004
- [12] <http://www.bb-elec.com/product.asp?sku=PCI-1710>