Palestine Polytechnic University

College of Engineering & Technology
Electrical & Computer Engineering Department

Graduation Project

Control Mouse Using an Electronic Glove

Project Team
Khalid A.K Daud
Mu'tasem Alheeh
Yasser Abu Zneid

Project Supervisor
Eng. Amal Al-Dweik

Hebron – Palestine

June, 2008

جامعة بوليتكنك فلسطين
الخليل – فلسطين
كلية الهندسة و التكنولوجيا
دائرة الهندسة الكهربائية و الحاسوب

اسم المشروع:

Control Mouse Using an Electronic Glove

أسماء الطلبة:

خالد داود معتصم الحيح ياسر أبو زنيد

بناء على نظام كلية الهندسة و التكنولوجيا و إشراف و متابعة المشرف المباشر على المشروع وموافقة أعضاء اللجنة الممتحنة تم تقديم هذا المشروع إلى دائرة الهندسة الكهربائية و الحاسوب وذلك للوفاء بمتطلبات درجة البكالوريوس في الهندسة تخصص هندسة أنظمة الحاسوب

توقيع المشرف

.....

توقيع اللجنة الممتحنة

....

توقيع رئيس الدائرة

.....

DEDICATION

To Our Parents

To Our Friends

ACKNOWLEDGMENT

Our appreciation to:

Palestine Polytechnic University

College of Engineering & Technology

Electrical & Computer Engineering Department

Our supervisor: Eng. Amal Dweik for her help and support

Any one who helped us.

ABSTRACT

The aim of this project is to design a wearable mouse, which user can wear by his hand and perform as a mouse wirelessly. User can move mouse, click, drag, scroll, and do other operations that can be done by mouse. By using this system user can perform mouse operations without restricting cables, and without needing a platform to move mouse on it.

By a wearable glove that user can wear; on which sensors are attached, sensors' readings are read by a PIC; then transmitted wirelessly using a Bluetooth to the PC. When data reaches the PC it will be analyzed to control mouse.

The project works successfully; we have built the glove circuit by which one can control mouse, moving it up, down, left or right, in addition to other operations that can be done by mouse cursor.

الملخص

النظام عبارة عن قفاز الكتروني قادر على التحكم بجميع العمليات الخاصة بفأرة الحاسوب (mouse)، بحيث يستطيع المستخدم التحكم بمؤشر الماوس بيده من خلال هذا القفاز؛ والاستغناء عن الماوس التقليدي وبالتالي لا يحتاج إلى سطح معين ليحرك عليه الماوس بل يستطيع التحكم بيده في الهواء.

باستخدام هذا النظام يستطيع الشخص تحريك الماوس إلى اليمين أو اليسار أو الأعلى أو الأسفل أو بشكل قطري، كما يستطيع القيام بعملية النقر للزر الأيمن والأيسر للماوس والنقر المزدوج، إضافة إلي ذلك تحريك الكائنات الموجودة على الشاشة والقيام بعملية الاستعراض في صفحات الانترنت وغيرها كما هو موجود بالنسبة للزر الأوسط في الماوس.

وبعد إتمام العمل في المشروع، تم تنفيذه بنجاح. حيث يتم من خلاله التحكم بالماوس فيستطيع المستخدم تحريك المؤشر و قيام بجميع عمليات الماوس الاخرى .

INDEX

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE 1

Introduction

1.1 General description of the project

1.2 Project Objectives

1.3 Literature Review

1.4 Time Plan

1.5 Finance Study

1.6 System Requirements

1.7 Risk Management

1.8 Report Contents

1.1 General description of the project

The aim of this project is to design a glove that user can wear by his hand and operates as a mouse. User can move mouse, click, scroll, and drag objects on the screen.

In this project we'll use a microcontroller that reads the measurements from sensors fixed on hand and send these data to a transmitter that will transmit it wirelessly to a receiver which is cabled to computer, computer then reads received data and interprets it to produce mouse movements.

This project combines many theoretical concepts that will be applied practically such like: reading from sensors; using Microcontroller, sending data wirelessly, and interpretation of raw binary data by a PC. Finally, we will use the Bluetooth technology in wireless transmission which is a new approach in wireless communication.

This chapter demonstrates a general idea about the project, its objectives, literature review for related projects, time plan, financial study, describes functional and non-functional system's and user's requirements and talks about expected risks and how to manage them.

1.2 Project Objectives

The main objective of our project is to design a mouse that is controlled wirelessly by human hand that wears a glove at which electronic parts are attached.

To achieve this object many sub objectives are to be done:

- Read sensors' data by PIC microcontroller.
- Apply Bluetooth technology to transfer data wirelessly.
- Build software at the PC side to control the mouse cursor operations.

1.3 Literature Review

There are several projects and studies related to our project:

1.3.1 Nontraditional Cursor Control [2]

The aim of this project was to design a motion-sensing glove that enables one to control the cursor on computer screen without the use of a mouse. It was built around an ATmega32 microcontroller, the system functions as a two-button mouse with vertical scrolling.

Regarding to this project we have applied many concepts in using accelerometer to control mouse, such like how they used accelerometer tilting to move cursor right, left, up and down.

However, they connected their system to PC serially using Microsoft Serial Protocol, but in our project we have used Bluetooth to connect the system wirelessly to PC, also we have used a different accelerometer and using digital outputs instead of analog ones.

1.3.2 Accelerometer Mouse [3]

This project is an accelerometer based tilt mouse, with two buttons and auto scroll ability. It consists of four accelerometers: two for position, and two for each of the buttons. The mouse communicates to a computer via the PS/2 protocol, or an optional PS/2 to USB converter and the mouse can be connected through a USB port. Furthermore, this mouse is an OS independent and has been tested for compatibility on a Macintosh and on an IBM compatible PC.

In this project four accelerometers were used, in our system one was enough, and also reduce system budget. One was enough to detect x and y tilting. Also they used PS/2 protocol to communicate with PC; in our system we used Bluetooth.

1.4 Time Plan

The following time plan shows the time scheduling in the first semester.

Week / Tasks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Data gathering																
Study Sensors																
Study Bluetooth																
Study PIC																
Documentation																

Figure 1-1: First Semester Time Plan

The following time plan shows the time scheduling in the second semester.

Week / Tasks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
PIC testing																
Implement serial connection between PIC & PC																
Accelerometer design & work																
Software programming																
Bluetooth circuit design																
Integrating System																
Testing																
Documentation																

Figure 1-2: Second Semester Time Plan

1.5 Finance Study

The following table shows the parts needed for the project and their cost.

Component	Required Number	Price NIS
Microcontroller PIC 18F4550	1	55
Bluetooth module Parani-ESD-200	1	700
Accelerometer ADXL202	1	140
Bluetooth adapter	1	45
Regulator LM 317	2	6
Wires and push buttons		30
Resistors and capacitors		20
Total		**996 NIS**

Table 1-1: Hardware cost

From the previous table, the total price is about 1000 NIS; it seems too much to spend such cost to control mouse. But as known, the research cost is always too high. When developing project in industry the real cost will become too much smaller.

1.6 System Requirements

We have two types of requirements; user requirements and system requirements.

User Requirements

User requirements contain the following:

- **Functional Requirements:**

- User can wear a glove to control the mouse.
- User can move mouse cursor.
- User can perform left click operation.
- User can perform right click operation.
- User can perform scroll operation.
- User can drag objects on screen.

- **Non-Functional Requirements:**

Non-Functional Requirements contain the following:

Flexibility

The glove should provide flexible movements for user hand, that isn't restricted to narrow distance.

Correctness

The transmitter circuit must send the appropriate code that represents the right code of the action performed. The receiver circuit must receive and send the code to the computer to take specific action.

Comfort

Glove sensors and circuit should be comfort to user; not heavy, not large and not annoying.

System Requirements

System requirements contain the following:

- **Functional Requirements**

- System needs a wireless transmitter/receiver for data transfer.
- System needs to read data from sensors.
- System needs to send measurements from sensors to a transmitter module.
- System needs a program at the computer to read data from serial port and use it to control mouse cursor.

- **Non Functional Requirements**

Reliability

The system should be reliable, meaning that it must send the code represent the action which generated not other code.

Speed

The system must have a fast response to the user's actions.

Low Power Consumption

Since this system is potable it will operate using a battery, so in order to make system work for a reasonable time, it must consume low power, and so we should reduce the number of components as much as possible.

1.7 Risk Management

There are some possible risks that may occur in our project in both hardware and software. In this section we will briefly describe those risks and show their management.

1.7.1 Technology Risks

Such risks may occur because of the software or hardware used in the system.

Hardware Risks:

The risks that may occur are:

- Device failure: the PIC may crash because of high voltage supply or other problems.
- The device operates differently from what its expected.
- Bluetooth connection Malfunctions.
- The receiver circuit does not respond.
- The needed time for response is so long that makes the user nervous.

Software Risks:

- Program of the PIC.
- The software on the computer side.

The PIC software has two main tasks:

1. It listens to the data coming from the sensors and adjusts it in an appropriate format.

2. It sends data coming from the sensors out to the transmitter through the USART.

The PC software has two main tasks:

1. It listens for incoming data on the USART, analyses them, and determines the format of data.
2. Uses incoming data to control mouse.

Any error in the above tasks mean that the system will not operate correctly and this will considered as a risk that must be avoided.

1.7.2 People Risks

The risks that may occur are:

- Illness of one or more of group members.
- Group meeting difficulties

1.7.3 Tools Risks

Lose of any support software or hardware used to develop the system, like microcontroller programming tools and case tools in software is considered as a risk.

1.7.4 Requirements Risks

Risks may occur if new changes are required in the system requirements that need major changes in the system design.

1.7.5 Estimation Risks

Risks may occur from the wrong estimation in the system design, implementation, resources and management.

There risks such as: read data by PIC from sensors at the inappropriate times will be generate incorrect code.

1.7.6 Risk Avoidance:

The following strategies will be taken to avoid risks mentioned above:

- Taking care when using hardware components and using them according to their specifications.
- Taking care of the team's member's health during the project development.
- Good estimation and usage of the projects budget and resources.
- Good estimation of system requirements.

1.7.7 Risk Management:

Risk management will be as follows:

- Software development environment risks will be handled by the backup of software.
- Including an extra amount of the hardware components we already have, so when any problem occurs we can find an alternative one.
- People risks are handled by using work load balancing on students especially when a member can't perform some of his tasks.

1.8 Report Contents

This documentation is categorized into chapters. Below it's a brief description about each one:

- **Chapter One: Introduction**

This chapter demonstrates a general idea about the project, its importance, literature review, time plan, finance study, system requirements and risk management.

- **Chapter Two: Theoretical Background**

This chapter provides a theoretical background about parts and technologies we used in the project.

- **Chapter Three: Project Conceptual Design**

This chapter describes the system in its abstract formula. It describes the project objectives, design options, general block diagram, system modeling and how the system works.

- **Chapter Four: Detailed Technical Project Design**

This chapter discusses a detailed description about project phases, subsystem design, overall system design and user system interface.

- **Chapter Five: Software System Design**

This chapter handles the software related to our system, depicts flow charts about system operation and illustrates different algorithms and techniques considered in writing the system software.

- **Chapter Six: System Implementation and Testing**

This chapter manifests the implementation procedures to be acted so as to integrate the project. Then, a sequence of procedural testing will be listed. The testing comprises both software and hardware testing.

- **Chapter Seven: Problems, Conclusions and Future work**

This chapter lists the problems faced us in accomplishing the system and how they were solved. Conclusions and future work are also proposed.

CHAPTER TWO 2

Theoretical Background

2.1 Overview

2.2 Project Components

2.3 Project Integrity

2.4 Theoretical Background

2.5 Summary

2.1 Overview

The aim of this project is to design a glove, that user can wear by his right hand and operates as a mouse; user can move mouse, click, scroll, drag objects on the screens and other operations.

In this project we'll use a microcontroller that reads the measurements from sensors fixed on hand fingers and send these data to a transmitter that will transmit it wirelessly to a receiver which is cabled to computer, computer then reads received data and interprets it to produce the cursor movements.

This chapter will talk briefly about project's components and provides theoretical background about technologies that could be used in our project design which involves:

Wireless communication:

- RF technology.
- Infrared technology.
- Bluetooth technology.

Besides, the chapter provides theoretical background about microcontrollers and accelerometers.

2.2 Project Components

The hardware components used are:

- **Microcontroller:** PIC microcontroller accepts data through digital inputs and sends it out on USART link.

- **Bluetooth IC**: convert USART data from microcontroller to wireless signal which received at PC side by another Bluetooth IC.

- **Accelerometer**: sensor that measures the tilt of hand according to static acceleration which will be converted to displacement that will move the cursor.

The software components consists two programs: first, a program at PC side that receives data from serial port, interprets it, and performs the operation associated with it, second, a program at PIC that reads sensors' data and send them.

More details about project components are introduced in chapter three.

2.3 Project Integrity

The only ethical issue that we could come up with involving our device is safety. Because there will be powered electrical components in close proximity with the body, we want to make sure that the user will not have to worry about issues of heat dissipation and electrostatic discharge. In our final testing of our device, we will ensure that the temperature of the device's surface in contact with skin does not exceed a temperature that is comfortable to the user.

2.4 Theoretical Background

2.4.1 Wireless Communication

Wireless is a term that refers to any type of electrical connection that is achieved without using hard wired connection. Wireless communication is the transfer of data or information between electrical or electronic components over a distance without using electrical wires. Distances can vary from very short distances (few meters like TV remote control) to very long distances (thousands of kilometers for radio waves).

Wireless communication is considered to be a branch of telecommunications; it is used in large number of applications including: cellular telephones, networks, satellite television, GPS units, wireless computer keyboard and mice, and many other applications.

Wireless communication can take place via many technologies, such like Radio frequency, infrared, and Bluetooth. However, a brief description about Radio frequency and infrared technologies will be provided. Then detailed information about Bluetooth technology will be provided.

2.4.1.2 Radio Frequency Communication

Radio is the transmission of signals, by modulating electromagnetic waves with frequencies below those of visible light. Information is carried by modulating (changing) of some properties of waves (such as amplitude or frequency). When radio waves pass an electrical conductor, it produces a current which can be detected and can be transformed to other signals that carry information.

"Electromagnetic spectrum is a limited natural resource, the use of which is governed by physical laws as well as national legislation. It has been estimated that as much as 75% of usable radio spectrum is reserved for use by various national governments and military applications. The amount of bandwidth available for commercial, private and public use is severely constrained and use of particular frequency bands is limited to individual countries or groups of countries."[1]

2.4.1.3 Infrared Technology

Visible light is energy at wavelengths between 380 and 780 nm. Ultraviolet (UV) has a wavelength shorter than visible light and Infrared (IR) wavelengths are longer than 780 nm. The approximate frequencies for the different forms of light are: [1]

- IR - 3×10^{11} Hz to 4×10^{14} Hz
- Visible light - 4×10^{14} Hz to 7.5×10^{14} Hz
- UV - 7.5×10^{14} Hz to 3×10^{17} Hz

Infrared technology introduced in many applications such like: target acquisition and tracking, remote control for radio and television, security systems, short range wireless communication, and many other applications.

Infrared has many advantages that applications can benefit from; the major advantages can be listed as follows:

- Low power consumption: so it is good for laptops and telephones.
- Low circuitry costs: low cost for coding and decoding circuitry.
- Security: infrared beam has a straight direction so its not possible for nearby devices to detected.
- High noise immunity: it's not likely to interfere with other signals.

Although IR has many advantages, it has problems that prevent many applications to use this technology such like:

- Line of sight: transmitters and receivers must able to see each other.
- Blocking: it's blocked by materials; people, walls ...etc.
- Short range: this is not enough for many applications.
- Light and weather sensitivity: rain, fog, pollution can affect transmission.

2.4.2 Bluetooth Technology

Bluetooth is a standard for wireless communication between devices with short range radio frequency (from 10 to 100 m). So any two devices that have this standard can communicate with each other without the need of making a connection between them.

Bluetooth is considered to be a low cost, low power technology, originally developed to connect devices such like laptops and mobile phones without cables. It makes wireless communication between devices in a small area as easy as possible; devices can talk together even there is no line of sight between them since Bluetooth is based on radio link. The link between devices which Bluetooth provides is direct, secure, and reliable to transmit both data and voice.

The Bluetooth specifications are open; can be downloaded from the internet, and no license is required to use it.

Bluetooth strengths

- **Low Cost :**
  Bluetooth chips are relatively cheap.

- **Tiny:** it's small.

- **Low-power consumption :**
  - 1 mW for class 3 (Range of 10 meters).
  - 10 mW Extended Class 1(Range of 100 meters).

- **It works all over the world :**
  It Operates on ISM (Industrial, Scientific & Medical) radio band. Which is an:
  - Unlicensed band.
  - Globally available (2.4 GHz).

- **Supports point-to-point &Point-to-Multi point communication.**

- **High Security:** It allows authentication & encryption.

- **Protection against interference:**
  It uses frequency-hopping spread spectrum Technology at 1600 hop/s.

- **Support of Time critical Connections:**
  By also supporting a Synchronous link (in addition to The Asynchronous link)

How Bluetooth Operates

Bluetooth networking transmits data via low-power radio waves. It communicates on a frequency of 2.45 gigahertz (actually between 2.402 GHz and 2.480 GHz, to be exact). This frequency band has been set aside by international agreement for the use of industrial, scientific and medical devices (ISM). [4]

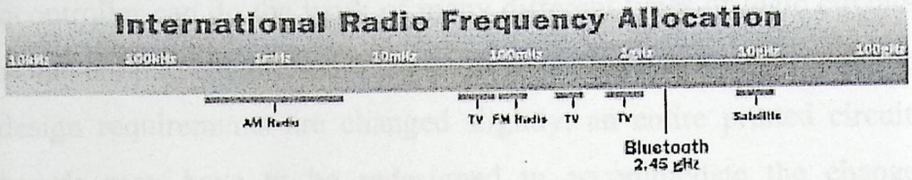


Figure 2-1: International Radio Frequency Allocation

Bluetooth can connect many devices simultaneously, with all devices in the same range (around 10 m). It uses the SSFH (Spread Spectrum Frequency Hopping) technique to avoid interference between connections. In this technique each Bluetooth transmitter changes its frequency within the range (2.402 GHz and 2.480 GHz) 1600 times per second automatically [4], and so no two transmitters will have the same frequency at the same time. Even if it is happened that two transmitters have the same frequency at the same time it likely that this interference will last only a tiny fraction of a second.

2.4.2 Microcontrollers

Microcontroller is a highly integrated single-chip microcomputer. Microcontrollers include a CPU to process information, program memory to store instructions, data memory to store information, system timing, and input/output sections to communicate with the outdoor world.

A microcontroller can do the work of many different types of logic circuits. Discrete logic circuits are permanently wired to perform the function they were designed to do. If the design requirements are changed slightly, an entire printed circuit board or many boards may have to be redesigned to accommodate the change. With a microcontroller performing the logic functions, most changes can be made simply by reprogramming the microcontroller. That is, the software (program) is changed rather than the hardware (logic circuits). This makes the microcontroller a very attractive building block in any digital system. With a microcontroller-based design, the designer can simply add a feature set to the product with minimal software/hardware changes. Microcontrollers can also be used to replace analog circuitry. Special interface circuits can be used to enable a microcontroller to input and output analog signals.

Microcontroller Architecture

Figure 2-2 shows the general block diagram of a microcontroller.

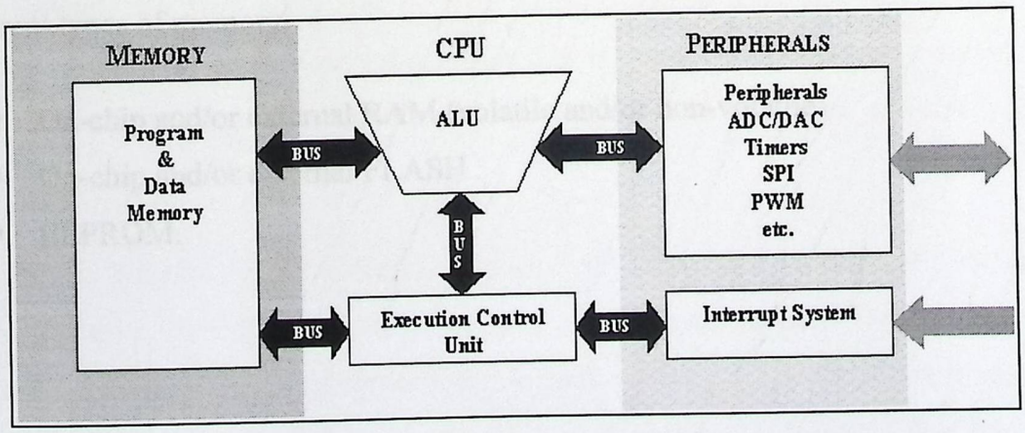


Figure 2-2: Microcontroller General Block Diagram

In general any microcontroller architecture cantinas the following main components:

Central Processing Unit CPU

Central Processing Unit (CPU) is the heart of a microcontroller where all of the arithmetic and logical operations are performed. This is the calculator part of the microcontroller. The CPU gets program instructions from the program memory.

Program and Data Memory

Program Memory contains a set of CPU instructions organized into a particular sequence to do a particular task. Program Memory is referred to as Read Only Memory (ROM) or OTP/EPROM. OTP or "One-Time Programmable" can be programmed only once and the program is stored permanently, even when the microcontroller power is turned off.

Program memory enables the microcontroller to immediately begin running its program as soon as it is turned on.

Data Memory can be both read and written and is required for the program stack, data storage, and program variables. This type of memory is commonly referred to as Random Access Memory (RAM). Each memory location has a unique address which the CPU uses to find the information it needs. A typical microcontroller contains both ROM and RAM type memory. For the storage of data one microcontroller may use different types of memory:

- On-chip and/or external RAM (volatile and/or non-volatile)
- On-chip and/or external FLASH
- EEPROM.

Inputs/Outputs and Peripherals

Microcontrollers require interface sections to communicate with external circuitry. Input ports allow data and status conditions to be read into the microcontroller while the output ports allow the microcontroller to affect external logic systems. The interfaces between the microcontroller and the outdoor world vary with the application, and may include display units, keypads, switches, sensors, relays, motors, and so on. Peripherals are used for the input output operations with external sensors and actuators and for the communication with external networks over standard buses.

Analog to Digital and Digital to Analog Converters

An analog-to-digital converter (abbreviated ADC, A/D or A to D) is an electronic circuit that converts continuous signals to discrete digital numbers. The reverse operation is performed by a digital to analog converter (DAC).

The resolution of the converter indicates the number of discrete values it can produce over the range of voltage values. It is usually expressed in bits.

Some types of microcontrollers may contain ADC input ports to interface with devices that outputs analog voltages. These ports eliminate the need of an external ADC chip.

Universal Asynchronous Receiver/Transmitter (UART)

A universal asynchronous receiver/transmitter is a type of "asynchronous receiver/transmitter", a computer hardware that translates data between parallel and serial forms. UARTs are commonly used in conjunction with other communication standards such as EIA RS-232.

A UART is usually an individual (or part of an) integrated circuit used for serial communications over a computer or peripheral device serial port. UARTs are now commonly included in microcontrollers. A dual UART or **DUART** combines two UARTs into a single chip. Many modern ICs now come with a UART that can also communicate synchronously; these devices are called USARTs.

Microcontroller Applications

Microcontroller's applications are more or less limited only by the user imagination. Microcontrollers now reside in our televisions, keyboards, modems, printers, telephones, cars, household appliances, and every other part of home and work life. The market for microcontrollers continues to expand rapidly, encompassing a wide range of consumer, industrial, automotive, and telecommunication applications. The emergence of new low cost microcontrollers offers a wealth of benefits for today's consumer applications and represents an entirely new profit source for manufacturers. In the past, the high cost of electronics limited the use of microcontrollers to "high tech" applications such as video recorders, stereo systems, and high-end durable goods such as washing machines. Today, the application base has broadened to include systems such as coffee machines, irons, shavers, and cleaners, where the introduction of electronics helps to provide product differentiation and allows the inclusion safety features.

PIC Microcontroller

PIC is a family of RISC microcontrollers made by Microchip Technology , regarded that PIC stands for Peripheral Interface Controller, the 8-bit PIC was developed in 1975 to improve performance of the overall system by offloading I/O tasks from the CPU. The PIC used simple microcode stored in ROM to perform its tasks.

PICs use a RISC instruction set, which varies in length from about 35 instructions for the low-end PICs to about 70 instructions for the high-end PICs. The instruction set includes instructions to perform a variety of operations on the accumulator and a constant or the accumulator and a memory location, as well as for conditionally executing code and jumping/calling other parts of the program and returning from them, and specific hardware features like interrupts and one low-power mode called sleep. Microchip provides a freeware IDE package called MPLAB that also includes a software simulator as well as an assembler.

2.4.3 Accelerometers

What is an accelerometer?

An accelerometer is an electromechanical device that measures acceleration forces. These forces may be static; like the constant force of gravity, or dynamic; which caused by moving or vibrating the accelerometer.

What are accelerometers useful for?

When you measure the amount of static acceleration due to gravity, you can find out the angle the device is tilted with respect to the earth. By sensing the amount of dynamic acceleration, you can analyze the way the device is moving.

An accelerometer can help to understand its surrounding better. Using acceleration measurements provided by an accelerometer, a good programmer can write codes that are useful in many applications; such like analyze problems in a car engine using vibration testing.

In the computing world, accelerometers are used in laptops to protect hard drives from damage. If you accidentally drop the laptop, the accelerometer detects the sudden freefall, and switches the hard drive off so the heads don't crash on the platters. In a similar fashion, accelerometers are the industry standard way of detecting car crashes and deploying airbags at just the right time.

How do accelerometers work?

There are many different ways to make an accelerometer! Some accelerometers use the piezoelectric effect - they contain microscopic crystal structures that get stressed by accelerative forces, which cause a voltage to be generated. Another way to do it is by sensing changes in capacitance. If you have two microstructures next to each other, they have a certain capacitance between them. If an accelerative force moves one of the structures, then the capacitance will change. Add some circuitry to convert from

capacitance to voltage, and you will get an accelerometer. There are even more methods, including use of the piezoresistive effect, hot air bubbles, and light. [5]

Issues related to accelerometers

Analog vs. digital

The output of the accelerometer can be analog or digital output. Analog style accelerometers output a continuous voltage that is proportional to acceleration. For example: 2.5V for 0g, 2.6V for 0.5g, 2.7V for 1g. Digital accelerometers usually use pulse width modulation (PWM) for their output. This means there will be a square wave of a certain frequency, and the amount of time the voltage is high will be proportional to the amount of acceleration.

Analog outputs can be connected to A/D converters (may be embedded in microcontrollers) and achieve the value of acceleration in a few microseconds. Digital outputs can be connected with purely digital inputs of microcontroller and measuring acceleration using timing resources.

Number of axis

For most projects, two is enough. However, if you want to attempt 3d positioning, you will need a 3 axis accelerometer, or two 2 axis ones mounted at right angles.

Sensitivity

Generally speaking, the more sensitivity the better. This means that for a given change in acceleration, there will be a larger change in signal. Since larger signal changes are easier to measure, you will get more accurate readings.

Bandwidth

This means the amount of times per second you can take a reliable acceleration reading. For slow moving tilt sensing applications, a bandwidth of 50Hz will probably suffice. If you intend to do vibration measurement, or control a fast moving machine, you will want a bandwidth of several hundred Hz.

Impedance/buffering

This is by far the single most common source of problems in projects involving analog accelerometers; for A-D conversion to work properly, the connected device must have output impedance under 10kΩ. Unfortunately, Analog Devices' analog accelerometers have an output impedance of 32kΩ. The solution to this is to use a low input offset rail to rail op amp as a buffer to lower the output impedance.

2.5 Summary

In this chapter we have talked about theoretical information about technologies and components related to our project. We talked about wireless communication technologies such as: Radio frequency, Infra red, and Bluetooth. Also we have talked about microcontrollers' architecture. Finally we have talked about accelerometer sensors.

CHAPTER THREE 3

Project Conceptual Design

3.1 Overview

In this chapter we are going to provide a detailed description for system objectives, system components, list of design options along with their strengths and weakness, graphical explanations such like: general block diagrams and system modeling diagrams, and finally a description of how the system works.

3.2 Detailed Project Objectives

The main objective of our project is to design a mouse that is controlled wirelessly by human hand that wears a glove at which electronic parts are attached.

The following objectives are also to be achieved:

- Read sensors' data by PIC microcontroller, interpret these measurements and transfer it to Bluetooth module.
- Apply Bluetooth technology to transfer data wirelessly from user hand to computer (PC).
- Build software at the PC side which accepts data from serial port and perform required action (click, drag…)

3.3 Design Options

In this system there are many design options, below are these options:

3.3.1 Processor Options

Option 1: Using a microprocessor such 8085, 8086, etc, with needed components such memory, A/D converters, etc.

This option have the advantage of using only what is required of components but increase the complexity of connecting components and cause a larger size.

Option 2: Using a microcontroller that contain all needed components; memory, A/D converters, etc.

This option seems more appropriate, because:

- Using CMOS (Complementary Metal Oxide Semiconductor) fabrication technique which requires much less power than older fabrication techniques, which permits battery operation.

- Using advanced memory options such as EEPROM (Electrically Erasable Programmable ROM) which is economic option in case of holding small amount of parameters that need to be changed over time (slow option), Flash when a requirement for large amounts of non-volatile program memory (faster than EEPROM), and Battery backed-up static RAM which is useful when a large non-volatile program and DATA space is required and it is much faster than other types of non-volatile memory so it is well suited for high performance application.

- Power management and low voltage: microcontroller contains low voltage parts that require at most 5V to operate and that microcontroller's clock can be slowed up or even stop putting the microcontroller in sleep mode which mean low power consumption.

- Having many components in a single chip (computer on a single chip) reduce the need for external components which mean lower cost components and less size. components such as CPU core, memory, I/O, UART, USART, ADC, Watchdog Timer, …

3.3.2 Wireless technology options

There are many options of wireless technologies that we can use such:

Option 1: RF technology

RF technology is useful for many reasons such as communication between two parties, it does not require a line of sight, not blocked by common materials; it can penetrate most solids and pass through walls, cover larger range than Bluetooth and IR, and not sensitive to weather/environmental conditions.

In the other hand it has some drawbacks where it has higher cost than infrared, operates at lower speed since its data rate transmission is lower than wired and infrared transmission, and required Federal Communications Commission (FCC) licenses for some products.

Option 2: Infrared technology

Infrared technology has many advantages that make it suitable in many applications since it need low power requirements, simple circuitry which enables it to be incorporated into integrated circuit of a product, improve security because directionality of the beam helps ensure that data isn't leaked or spilled to nearby devices, noise immunity not to have interference from signals of other devices, portable and low cost.

Infrared is not suitable for our project since it considered to be line of sight which mean that it will decrease the flexibility of the user and can be blocked by common materials which make it less reliable.

Option 3: Bluetooth: we will use this technology.

Bluetooth improves Low-power consumption; it requires 1mW to 10mW in order to operate, small which allow using it in many applications without the restriction of the size, operates in ISM radio band which is globally available. The important points in Bluetooth which make us choose it are that Bluetooth chips are small and don't require line of sight which increase flexibility. On the other hand, Bluetooth ensures reliability since no interference will occur to the signals being transferred by using frequency-hopping spread spectrum technology at 1600 hop/s. Finally Bluetooth is a new technology that wasn't used before in our university projects.

3.3.3 Bluetooth options

Option 1: Using a PIC that includes internally a Bluetooth unit, so no need to add an externally Bluetooth chip, using this option has the advantages of less size chip and reducing the complexity of connecting external Bluetooth chip, an example of such chips is the ToothPIC which combines a PIC18LF6720 microcontroller and a BlueMatik Bluetooth radio.

Option 2: Using a separate Bluetooth chip; which we must connect to a PIC; using this option has the advantage of choosing an appropriate chip without restrictions of what is available of Bluetooth modules in option 1, there are many commercial Bluetooth IC's; e.g. Blue-KC21 and Parani-ESD200. (We will use a separate Bluetooth chip)

3.3.4 Motion detection options

In order to detect hand motion to control cursor many options are to be considered:

Option 1: Track position using accelerometer

An accelerometer can be used to track hand position in air, by taking the output of the accelerometer which represent the acceleration according to force applied on it, and then using mathematical equations by which position can be determined.

But according to our literature review the accelerometer had to be held exactly perpendicular to earth surfaced, that is it's not flexible, and error will be large.

Option 2: Tilting measurement

In this option titling of accelerometer is measured, we will use this option, it is easier to implement and it is more reliable.

3.4 Detailed Project Components

3.4.1 PIC 18F4550

Microcontroller is needed in order to receive signals from sensors, and then send them out to Bluetooth module (via USART) which transmits signals wirelessly to the computer.

The microcontroller we used is PIC 18F4550 which has the following features:

- **High-Performance RISC CPU**
  - Only 75 single word instructions to learn.
  - All instructions are single cycle (1μs) except for program branches.
  - Operating speed: DC - 20MHz clock input.
  - 8 k Bytes Flash Program Memory.
  - 368 Byte RAM Data Memory.
  - 256 Byte EEPROM Data Memory.
  - In-circuit serial programming.

- **Peripheral Features**
  - High current sink/source 25 mA/25 mA
  - Three external interrupt
  - Enhanced Capture/ Compare/ PWM (ECCP) module
  - Compatible 10-bit, up to 13-channels Analog-to-Digital Converter module (A/D)
  - Dual analog comparators

- **Special Microcontroller Features.**
  - C compiler optimized architecture with optional extended instruction set
  - 100,000 erase/write cycle Enhanced Flash program memory typical
  - 1,000,000 erase/write cycle Data EEPROM memory typical
  - Flash/Data EEPROM Retention: > 40 years
  - Self-programmable under software control
  - Priority levels for interrupts

- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
- Programmable period from 41 ms to 131s
- Programmable Code Protection
- Single-Supply 5V In-Circuit Serial
- Programming™ (ICSP™) via two pins
- In-Circuit Debug (ICD) via two pins
- Optional dedicated ICD/ICSP port (44-pin devices only)
- Wide operating voltage range (2.0V to 5.5V)

- **Oscillators**
  - Four Crystal modes
  - Two external Clock modes, up to 48 MHz
  - 8 user-selectable frequencies
  - Secondary oscillator using Timer1 @ 32 kHz
  - Fail-Safe Clock Monitor

- **I/O and Packages**
  - 34 I/O pins with individual direction control.
  - 40-pin DIP.

Figure 3-1: PIC18F4550

3.4.2 Bluetooth module: Parani-ESD 200

Parani-ESD is a module device for wireless serial communication using Bluetooth technology that is international a standard for short range wireless communications.

Parani-ESD can communicate with other Bluetooth devices that support the Serial Port Profile.

Parani-ESD lineup has several models with different communication ranges from 30m (Parani- ESD200/210) up to 100m (Parani-ESD100/110) for use with various applications. The Parani-ESD delivers better quality of communication than a standard RS232 cables.

Parani-ESD has a compact design and can be placed conveniently into devices or equipment. Its detachable antenna optimizes the quality and distance for wireless communications.

Parani-ESD supports FHSS (Frequency Hopping Spread Spectrum), which is a technique, native to Bluetooth that allows the Parani-ESD minimize radio interference while decreasing the likelihood of over-air hijacking. Parani-ESD also supports authentication and Bluetooth data encryption.

Parani-ESD can be configured and controlled by typical AT commands. Users can easily configure Parani-ESD by using a terminal program such as HyperTerminal and can use Bluetooth wireless communication without modifying user's existing serial communication program. In addition to the basic AT commands, Parani-ESD provides some expanded AT commands for various functions. User friendly ParaniWizard and ParaniWIN are also provided for easy setup on Microsoft Windows.

Operation Modes

A Bluetooth device can play a role as a master or slave. Master tries to connect itself to other Bluetooth devices, and slave is waiting to be connected from other Bluetooth devices. A Bluetooth connection is always made by a pair of master and slave devices. A slave can be in two modes, Inquiry Scan or Page Scan mode. Inquiry Scan mode is waiting for a packet of inquiry from other Bluetooth device and Page Scan mode is waiting for a packet of connection from other Bluetooth device. Every

Bluetooth device has its unique address, called BD (Bluetooth Device) address, which is composed of 12 hexa-decimal numbers.

Parani-ESD has 4 operation modes as follows:

Mode0

In this mode, there is no response when power on or software reset, and Parani-ESD is just waiting for AT command input. Neither master nor slave is assigned to Parani-ESD in mode0. User can change the configuration parameters of Parani-ESD in this mode. Parani -ESD must be in Mode0, when it is directly controlled by AT commands. The factory default is set to Mode0.

Mode1

Parani -ESD tries to connect the last connected Bluetooth device. Parani -ESD in Mode1 is to be a master and tries to connect the last connected Bluetooth device. Parani-ESD always stores the BD address of the Bluetooth device to which Parani-ESD has connected last.When Parani-ESD is initially used or after hardware reset, there is no BD address stored in Parani-ESD. In this case, Mode1 will not be able to work properly. The mode change to Mode1 can be made after Parani-ESD succeeds to connect to one other Bluetooth device. Once changed to Mode1, Parani-ESD will try to connect automatically the last connected Bluetooth device whenever the unit is powered on or software reset. Parani -ESD in Mode1 cannot be discovered or connected by other Bluetooth devices.

Mode2

Parani -ESD is waits for a connection from the last connected Bluetooth device. Parani -ESD in Mode2 is to be a slave and waiting for the connection only from the last connected Bluetooth device. Just like Mode1, if there is no BD address stored in Parani-ESD, the mode change from other operation modes to Mode2 is not work properly. Once changed to Mode2, Parani-ESD will wait for the connection from the last connected Bluetooth device whenever the unit is powered on or software reset. Parani -ESD in Mode2 cannot be discovered or connected to Bluetooth devices other than the last connected device.

Mode3

Parani -ESD is waiting for the connection from any other Bluetooth devices. In Mode 3 the Parani -ESD is discoverable and can be connected to by other Bluetooth devices.

Bluetooth Interface:

Parani-ESD 200 :

- Bluetooth v1.2
- Class 2
- Level - 4 dBm
- Protocols - RFCOMM, L2CAP, SDP
- Profiles - General Access Profile, Serial Port Profile
- Working distance

  Default Antenna - Default Antenna 30 meters

Physical properties:

Parani-ESD200 : 18 x 20 x 11.7(mm)

Power: 3V~3.3V, ESD 200: minimum 300mA

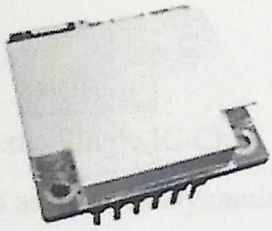


Figure 3-2: Parani-ESD 200

3.4.3 Accelerometer: ADXL202

The ADXL202 are high precision, low power, complete 2-axis accelerometers with signal conditioned voltage outputs, all on a single, monolithic IC.

The ADXL202 is complete dual axis acceleration measurement systems on a single monolithic IC. They contain a polysilicon surface-micromachined sensor and signal conditioning circuitry to implement open loop acceleration measurement architecture. For each axis, an output circuit converts the analog signal to a duty cycle modulated (DCM) digital signal that can be decoded with a counter/timer port on a microprocessor.

The ADXL202 are capable of measuring both positive and negative accelerations to a maximum level of ±2 g or ±10 g. The accelerometer measures static acceleration forces such as gravity, allowing it to be used as a tilt sensor.

The ADXL202 are available in 5 mm × 5 mm × 2 mm, 8-pad hermetic LCC packages.

Features

- 2-Axis Acceleration Sensor on a Single IC Chip
- Measures Static Acceleration as Well as Dynamic Acceleration
- Duty Cycle Output with User Adjustable Period
- Low Power <0.6 mA
- Tilt Sensors
- Bandwidth Adjustment with a Single Capacitor Per Axis
- mg Resolution at 60 Hz Bandwidth
- +3 V to +5.25 V Single Supply Operation
- 1000 g Shock Survival

More details and figures about the above previous component and chips are in the appendix of project.

3.5 Project Design Block Diagram

The following figure (Figure 3-3) shows the general block diagram of the system in which we can see that sensors readings are transmitted by a Bluetooth transmitter to a receiver that sends these readings to PC through a PIC.

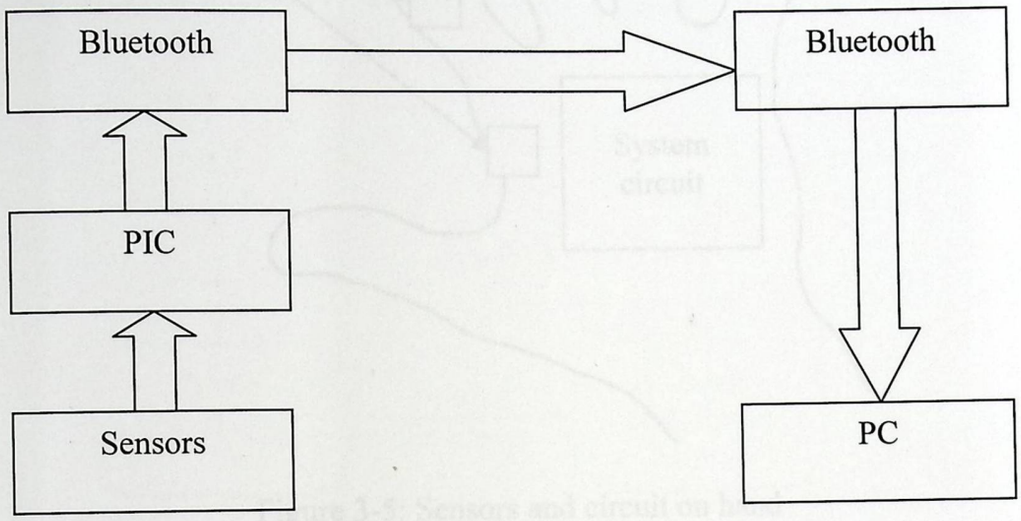


Figure 3-3: general block diagram

Sensors are connected to PIC, the following figure shows how they are connected.

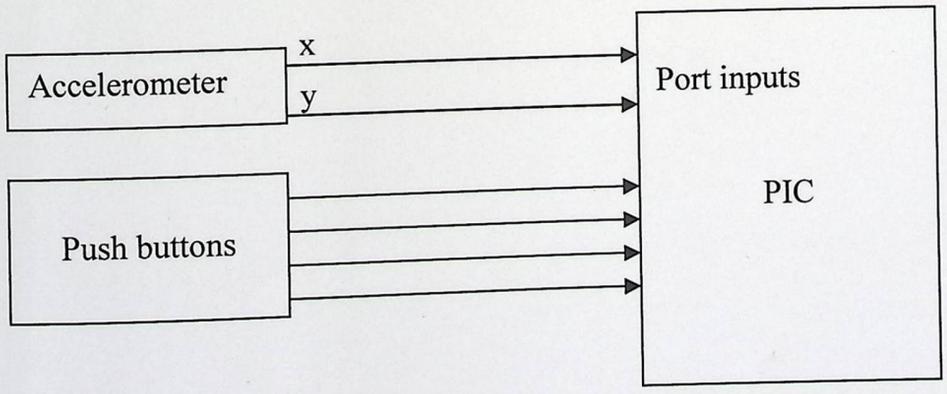


Figure 3-4: Sensors connected to PIC

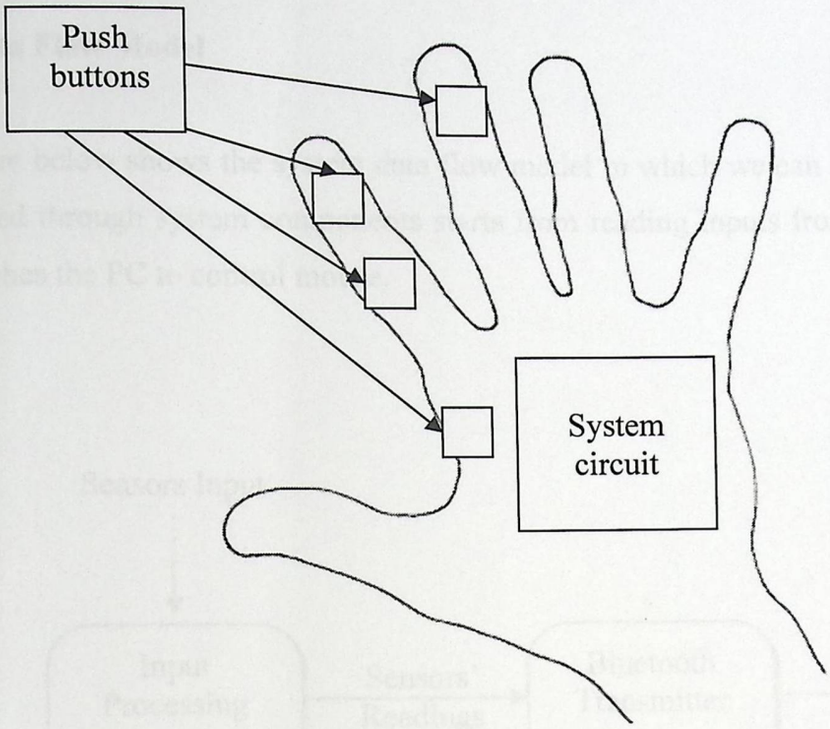


Figure 3-5: Sensors and circuit on hand

3.6 System Modeling

3.6.1 Data Flow Model

The figure below shows the system data flow model in which we can see how data is transferred through system components starts from reading inputs from sensors until data reaches the PC to control mouse.

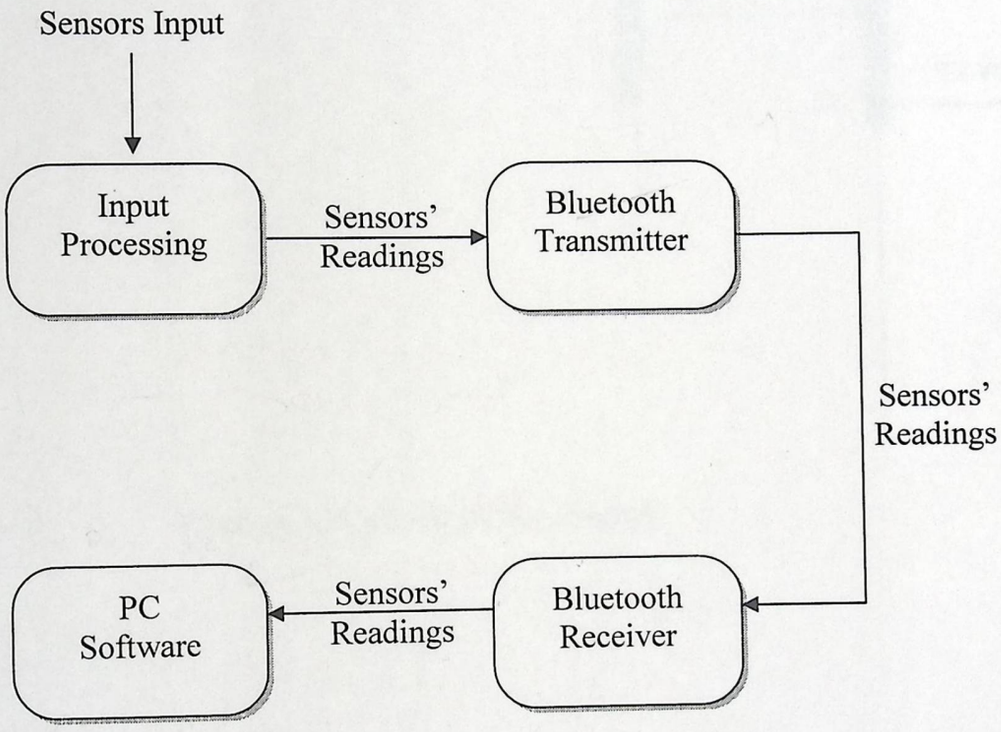


Figure 3-6: System data flow model

3.6.2 Sequence Diagram

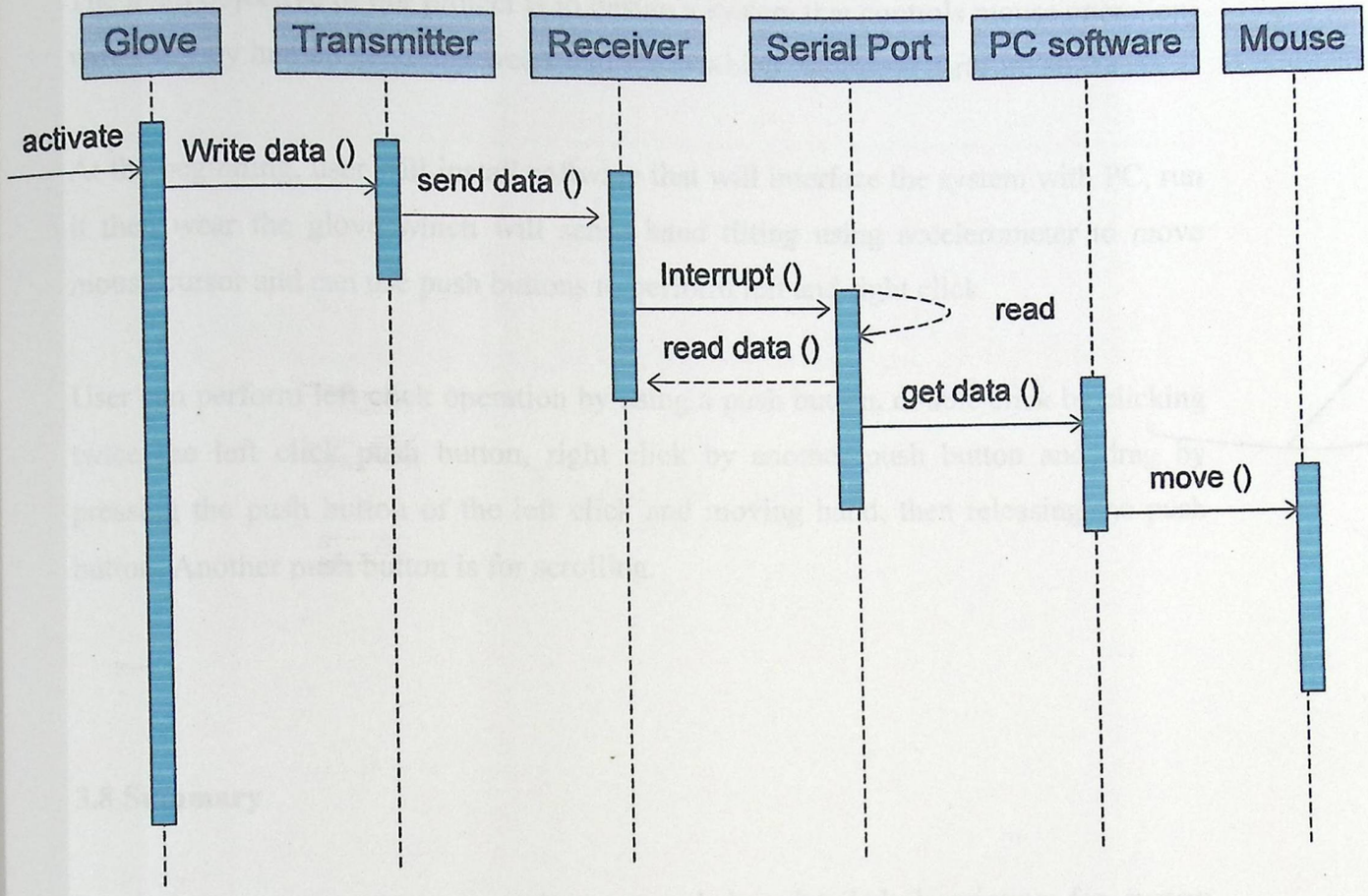


Figure 3-7: System Sequence Diagram

3.6.2 Sequence Diagram

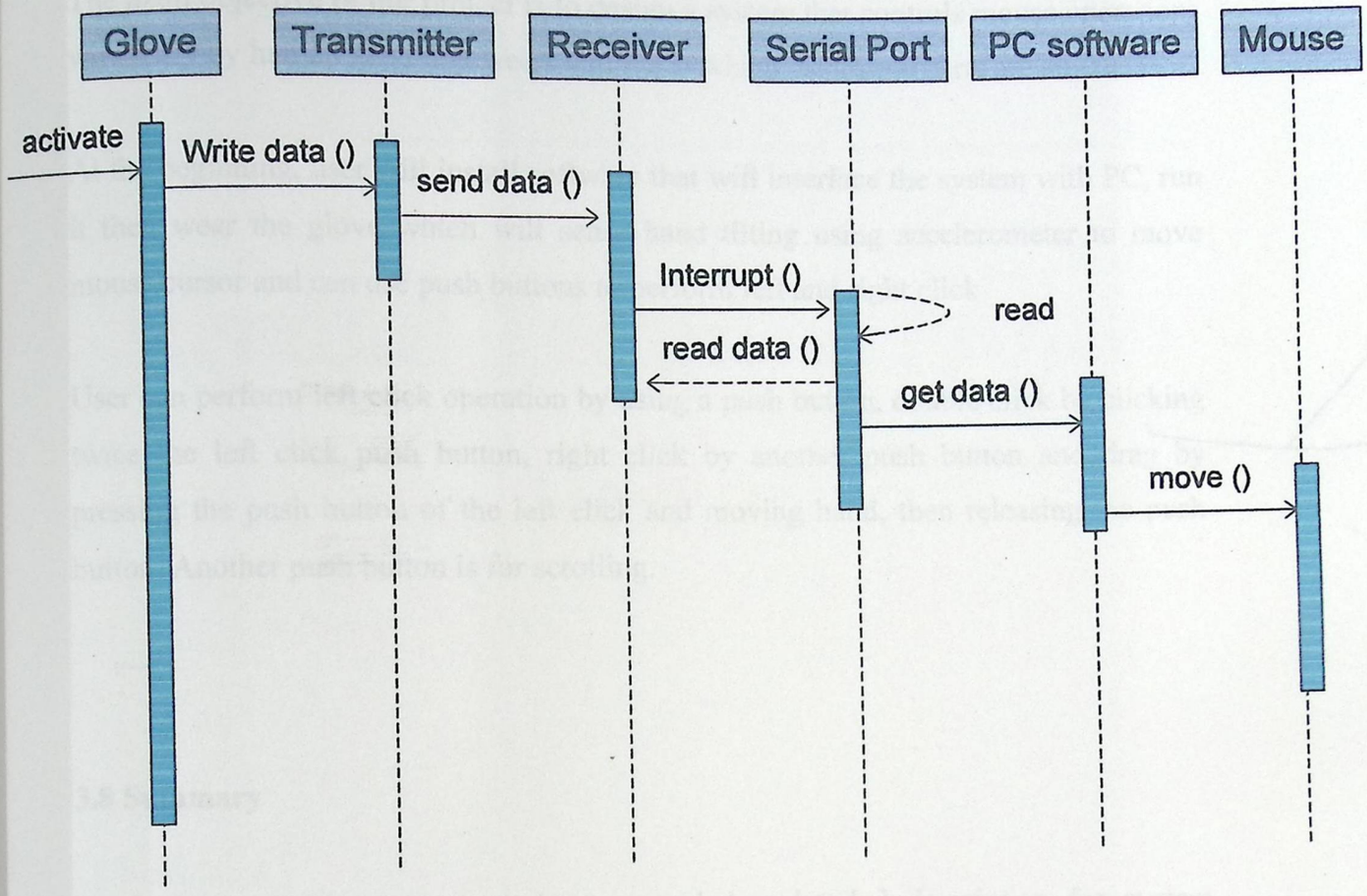


Figure 3-7: System Sequence Diagram

3.7 How System Works

The main objective of our project is to design a system that controls mouse operations wirelessly by human hand that wears a glove at which electronic parts are attached.

At the beginning, user will install software that will interface the system with PC, run it then wear the glove which will sense hand tilting using accelerometer to move mouse cursor and can use push buttons to perform left and right click

User can perform left click operation by using a push button, double click by clicking twice the left click push button, right click by another push button and drag by pressing the push button of the left click and moving hand, then releasing the push button. Another push button is for scrolling.

3.8 Summary

At the end of this chapter we have provided a detailed description for system objectives, system components, list of design options along with their strengths and weakness, graphical explanations such like: general block diagrams and system modeling diagrams, and finally a description of how the system works.

CHAPTER FOUR 4

Detailed Technical Project Design

4.1 Overview

In this chapter we will provide a detailed description about project phases, subsystems schematics and design, and schematic for overall system design. Then we will talk about User-System Interface that is how the user can interact with the project.

4.2 Detailed Description of the Project Phases

Phase 1: Microcontroller (PIC18F4550) Studying

First of all we have studied how to program and use the microcontroller, so we wrote a simple C program that takes a reading from a switch and light a LED according to it. The Aim of this phase was to practice on using MPLAP program that used to assemble the C code to Hexadecimal code and then using the Superpro program to write .hex code into microcontroller memory.

Phase 2: Accelerometer Design and Testing

In this phase we have studied how to implement the accelerometer circuit and what additional parts needed to be connected to the accelerometer in order to work properly (the circuit of the accelerometer and additional parts are to be discussed in the following section), and after implementing the circuit we started testing operation in which each of the accelerometer outputs (Xout & Yout) were connected to one of the oscilloscope channels and the result was a square wave (0V-low, 5V-high) for both Xout and Yout; that represents the digital output.

Phase 3: Implementing Serial Connection between Microcontroller and PC

Since our project is Human Interface System (HIS) with the computer we need to make connection between project circuit and PC which we assumed it to be wirelessly using Bluetooth module, but unfortunately this module wasn't available at early stages of our work, so we tried an alternative solution which is implementing this connection serially.

Serial connection includes using an IC called MAX232 which is needed to make voltage levels conversion from TTL in PIC to RS-232 in PC serial port and vice versa. In this connection PIC used USART pins for transmitting and receiving, and software was built at the computer side which requires to accept data from serial port and show them in an order to test our work.

Phase 4: Acquiring Accelerometer Readings and Understand How to Deal With it

After the previous phase we start to determine how we could make use of the accelerometer readings which are (as mentioned before) square waves in which the duty cycle [1] for each axis determines the amount acceleration in that axis.

A program was needed in both microcontroller and PC; in microcontroller to detect this duty cycle and in PC to show the results. In case of microcontroller the program was implemented to set the timer to be 0 at the rising edge of the square wave (the ON part of the cycle) and take the value of the timer at the end of ON, then send this value through USART serially to PC which in turn shows this result through the implemented program and repeats for all cycles.

After analyzing the results we were able to understand how the readings are changed by tilting the accelerometer in all directions – more details about these readings will be presented in chapter 6.

Phase 5: Calibration of the Accelerometer

By the previous results we recognized that a calibration procedure is required in which two offset values are determined each for one axis then any value that is larger than that offset value is considered as increasing in the direction (to the RIGHT in case of x-axis, and DOWN in case of y-axis) and any value less than that offset is considered as decreasing in the direction (to the LEFT in case of x-axis, and UP in case of y-axis).

Phase 6: Implementing Programs at Microcontroller and PC that Accepts Accelerometer Readings and Control Mouse Operations

After we understood the operation of the accelerometer and how to consider its reading in controlling mouse cursor, it's the time to implement all that to control mouse cursor, so we adjust the microcontroller program so that the readings from accelerometer are used to build what we called packets which are sent to the computer. These packets contains information required to determine what operation to take – as a brief description of these packets, each operation that mouse may take is represented by a bit (e.g. Left Click: pressed if the bit is 1 and released if the bit is 0). The whole explanation of the packet format will be in the following chapter.

The same is done at PC side where the software also is adjusted so that it accepts each packet from the microcontroller and analyzes it then performs the desired operation, and implements the interface that will be used by the user.

Phase 7: Integrate Bluetooth Module and Bluetooth USB Adapter into the Project

At this phase we replaced serial connection by wireless connection using Bluetooth technology in which Bluetooth Module chip is connected to the microcontroller through USART pins instead of MAX232 and Bluetooth USB adapter at PC side and make the communication between microcontroller and PC by transmitting data wirelessly between these Bluetooth devices.

Phase 8: Integrate Project Components at the Glove

As a final step all components which are Microcontroller, Accelerometer, Bluetooth module and Push Buttons with their additional parts are integrated at one board by wire wrapping them along with the required power supply (DC Battery), then integrating this board to a glove that user can wear.

4.3 Subsystem Detailed Design

4.3.1 Microcontroller Circuit

Microcontroller is needed to receive signals from sensors (Accelerometer & Push Buttons), and then interprets these signals using software that resides in its memory in order to formulate packets which are sent out serially to the Bluetooth module.

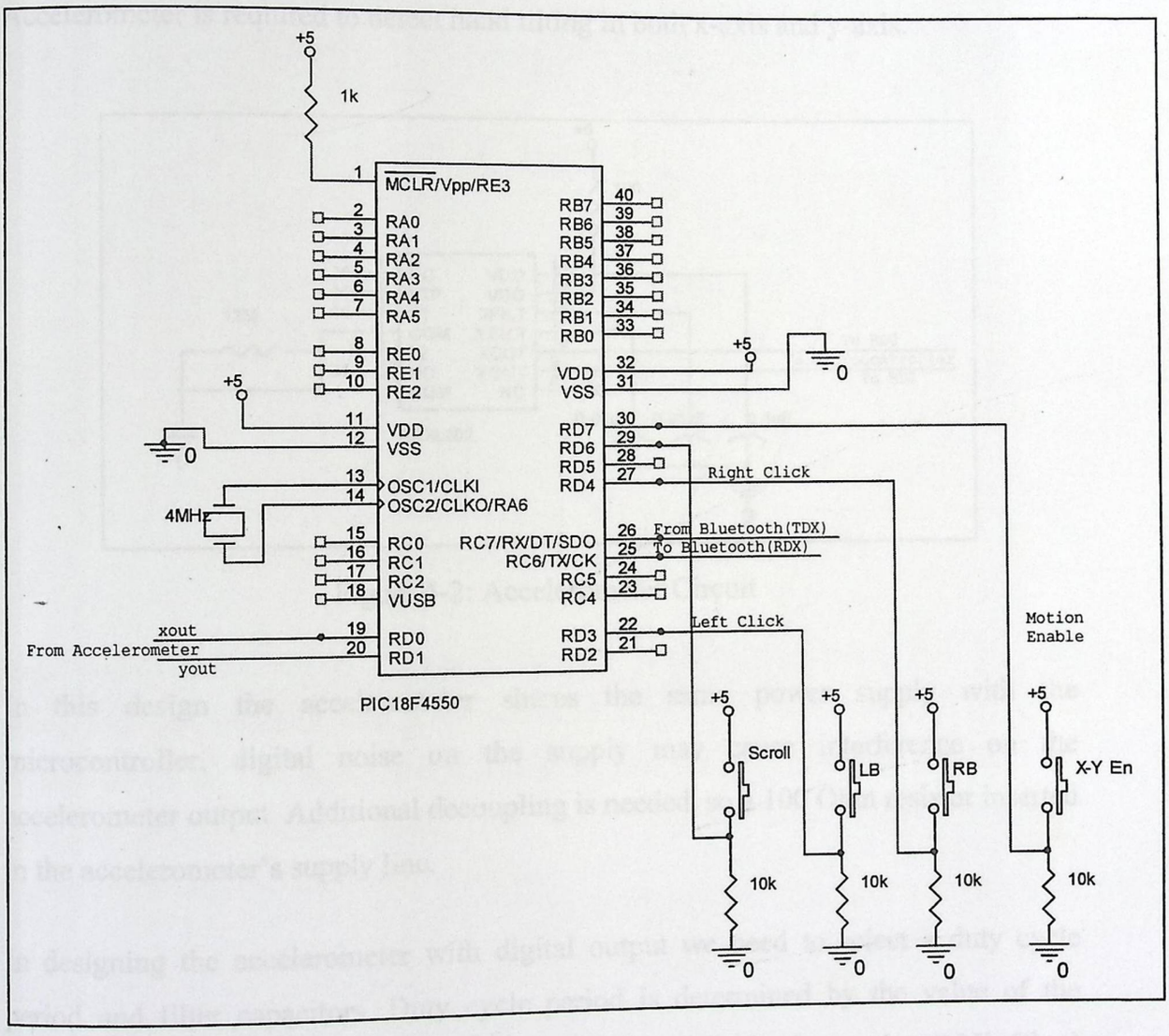


Figure 4-1: Microcontroller Circuit

Here the microcontroller is connected with both accelerometer outputs by two pins from port D (RD0 & RD1) in order to detect their signals. And since we need to transmit data to the Bluetooth module which is done through USART output (Tx) which is connected to Bluetooth input RDX. Also to detect signals from other three

sensors (Left Click, Right Click and Scroll) we simply connected Push Buttons to another three pins from port D (RD3 for Left Click, RD4 for Right Click and RD7 for Drag). Finally there is an Enable which is required to be pressed to when the user start to move the cursor and released when to stop the cursor or perform another action like Left Click, Right Click or Scroll.

4.3.2 Accelerometer Circuit

Accelerometer is required to detect hand tilting in both x-axis and y-axis.

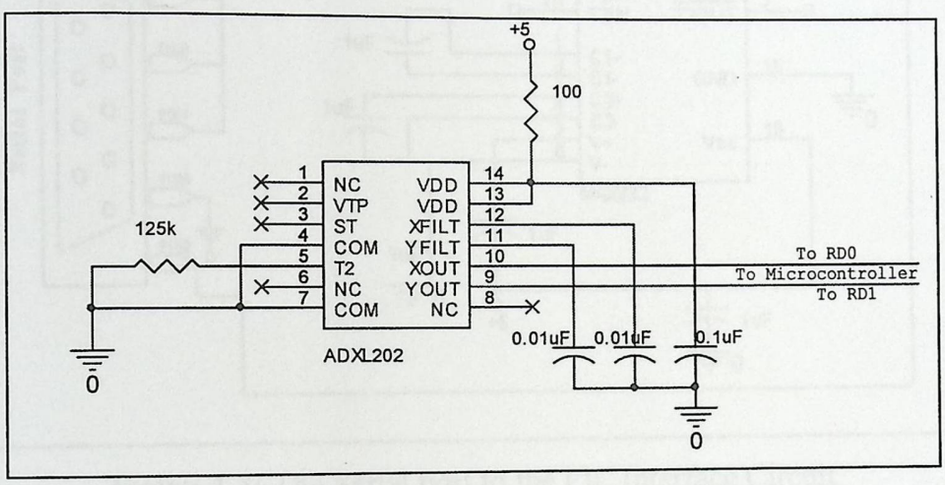


Figure 4-2: Accelerometer Circuit

In this design the accelerometer shares the same power supply with the microcontroller, digital noise on the supply may cause interference on the accelerometer output. Additional decoupling is needed, so a 100 Ohm resistor inserted in the accelerometer's supply line.

In designing the accelerometer with digital output we need to select a duty cycle period and filter capacitors. Duty cycle period is determined by the value of the resistor connected to the T2 pin (pin 5), here we chose the resistor value (125k Ohm) to obtain the minimum duty cycle period available (1ms) which provided the highest frequency; since speed is one of the functional requirements, and choose filter capacitors (0.01uF) so that the accelerometer is sampled at 500Hz.

4.3.3 The Serial port to the PIC Interface Circuit

This circuit was designed to make serial communication with the PC serially through serial port to perform all needed communications with the PC since Bluetooth module wasn't available early. Bluetooth is to be implemented as final step in the project instead of the serial port to the PIC Interface Circuit.

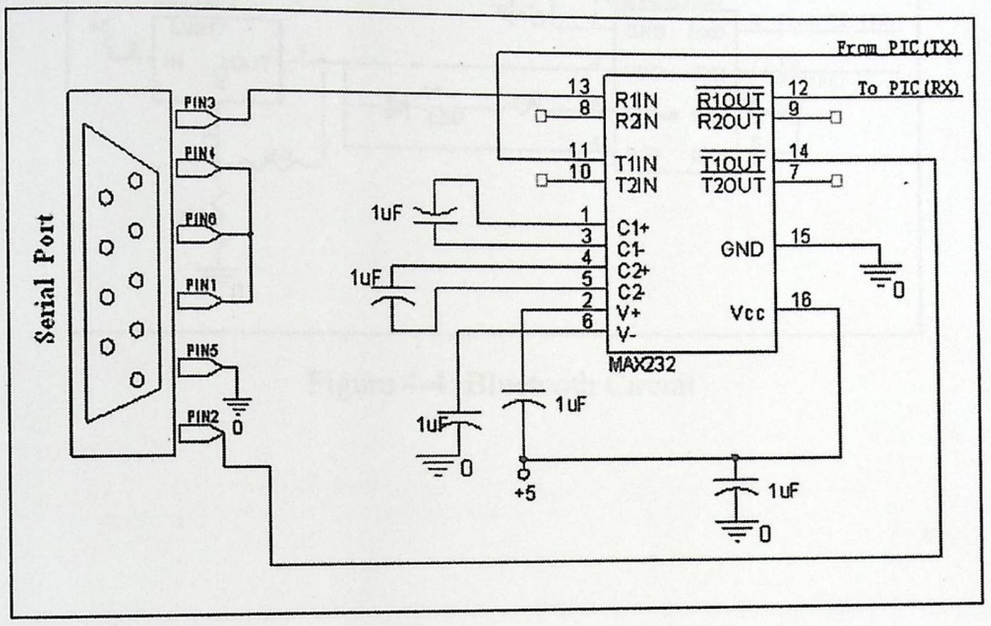


Figure 4-3: The Serial port to the PIC Interface Circuit

Since serial port at the PC works at RS-232 signal voltage levels and that PIC microcontroller works at TTL levels, we need MAX232 to adapt the RS-232 signal voltage levels to TTL logic. And it just needs one voltage (+5V) and generates the necessary RS-232 voltage levels (-10V and +10V) internally. It requires only 5 capacitors as shown in the semantic above.

4.3.4 Bluetooth Module Circuit

Bluetooth module uses the Bluetooth wireless connection to communicate with the Bluetooth USB Adapter at the PC, and the Universal Asynchronous Receiver/Transmitter (UART) to communicate with the host microcontroller.

The RXD and TXD pins of the module are connected directly to TX and RX pins of the microcontroller respectively. The Status pin goes low when the Bluetooth connection exists with the Bluetooth USB Adapter at the PC; a LED is used to tell the user about the status of the connection. A 3.3 voltage regulation circuit is required to power the Bluetooth module.

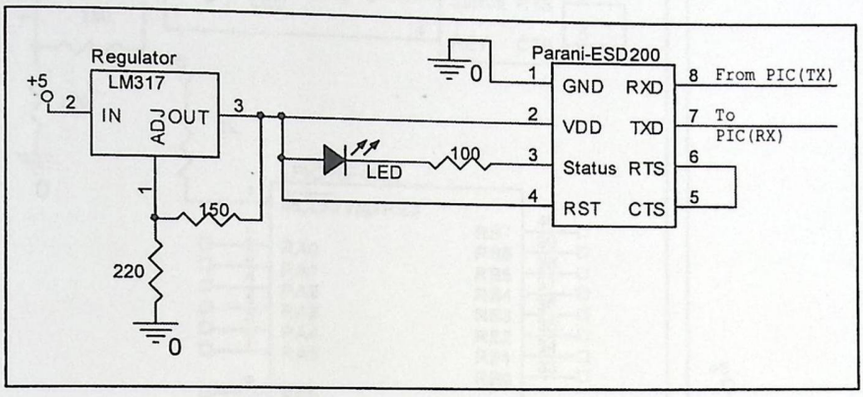


Figure 4-4: Bluetooth Circuit

4.4 Overall System Design

The complete schematic for the project is shown in figure 4-5.

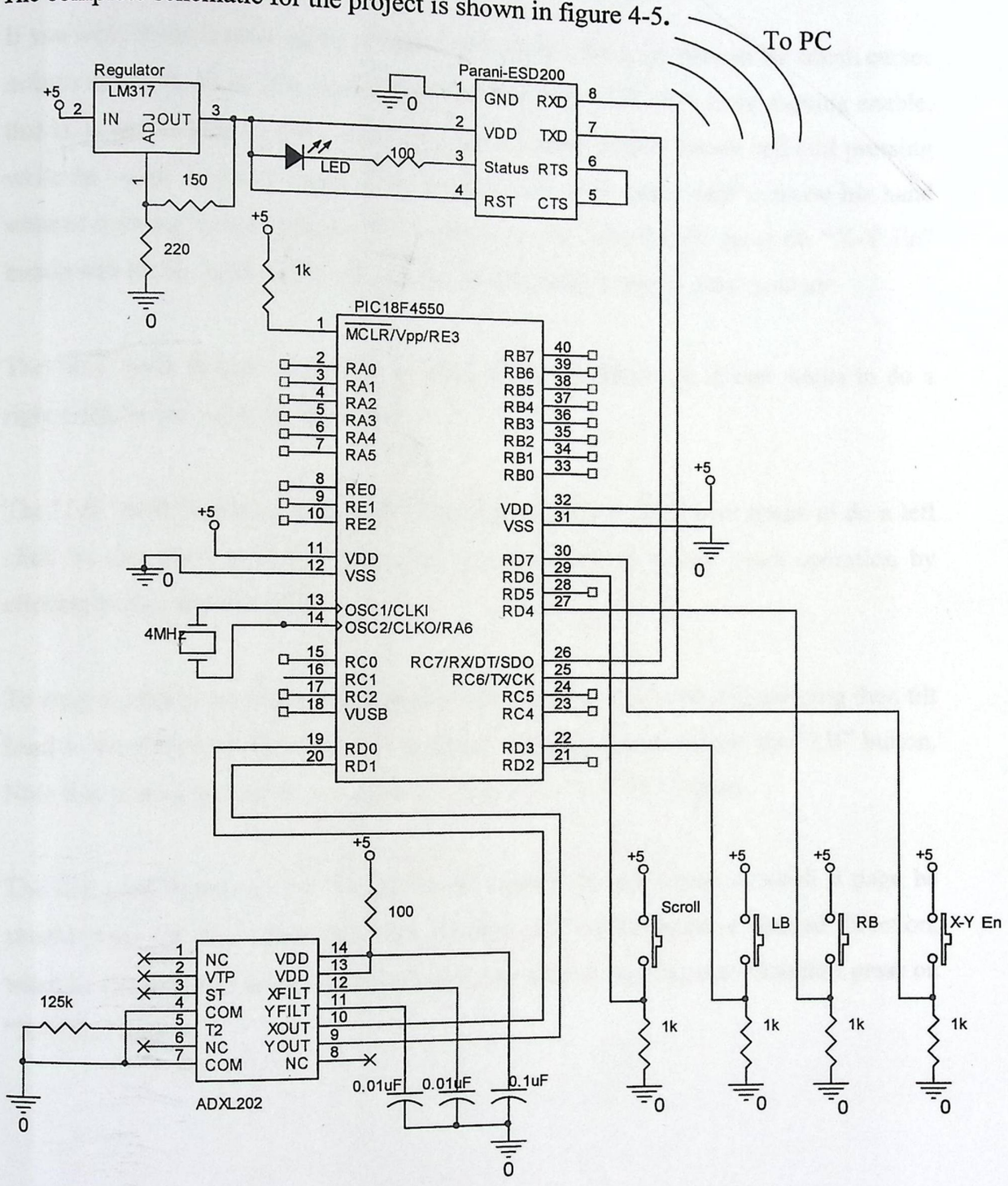


Figure 4-5: Overall System Schematic

The previous figure shows the overall system schematic, this circuit will be put on hand by which user can control mouse cursor.

If you look at the bottom right of figure you can see the push buttons by which cursor actions are controlled. The push button labeled with "X-Y En" is for moving enable, that is, if user wants to move cursor he should press on this button and still pressing while he wants to move cursor, this is important when user want to move his hand without moving mouse cursor. So to move cursor, one should press on "X-Y En" button and tilt his hand right, left, up or down in order to move mouse cursor.

The "RB" push button is used to do right click operation. So if user wants to do a right click he can click on this button.

The "LB" push button is used to do left click operation. So if user wants to do a left click he can click on this button, he can also perform double click operation by clicking twice on this button.

To drag an object on screen, user should press on the "LB" and still pressing then tilt hand to the directions he wants, when finish dragging he can release the "LB" button. Note that in drag operation one shouldn't press on "X-Y En" button.

The last push button is the "Scroll" push button. If user wants to scroll a page he should press on this button and still pressing and tilt his hand to desired direction, when he finishes he can release this button. Note here also that one shouldn't press on "X-Y En" button.

4.5 User-System Interface

4.5.1 Hardware Interface

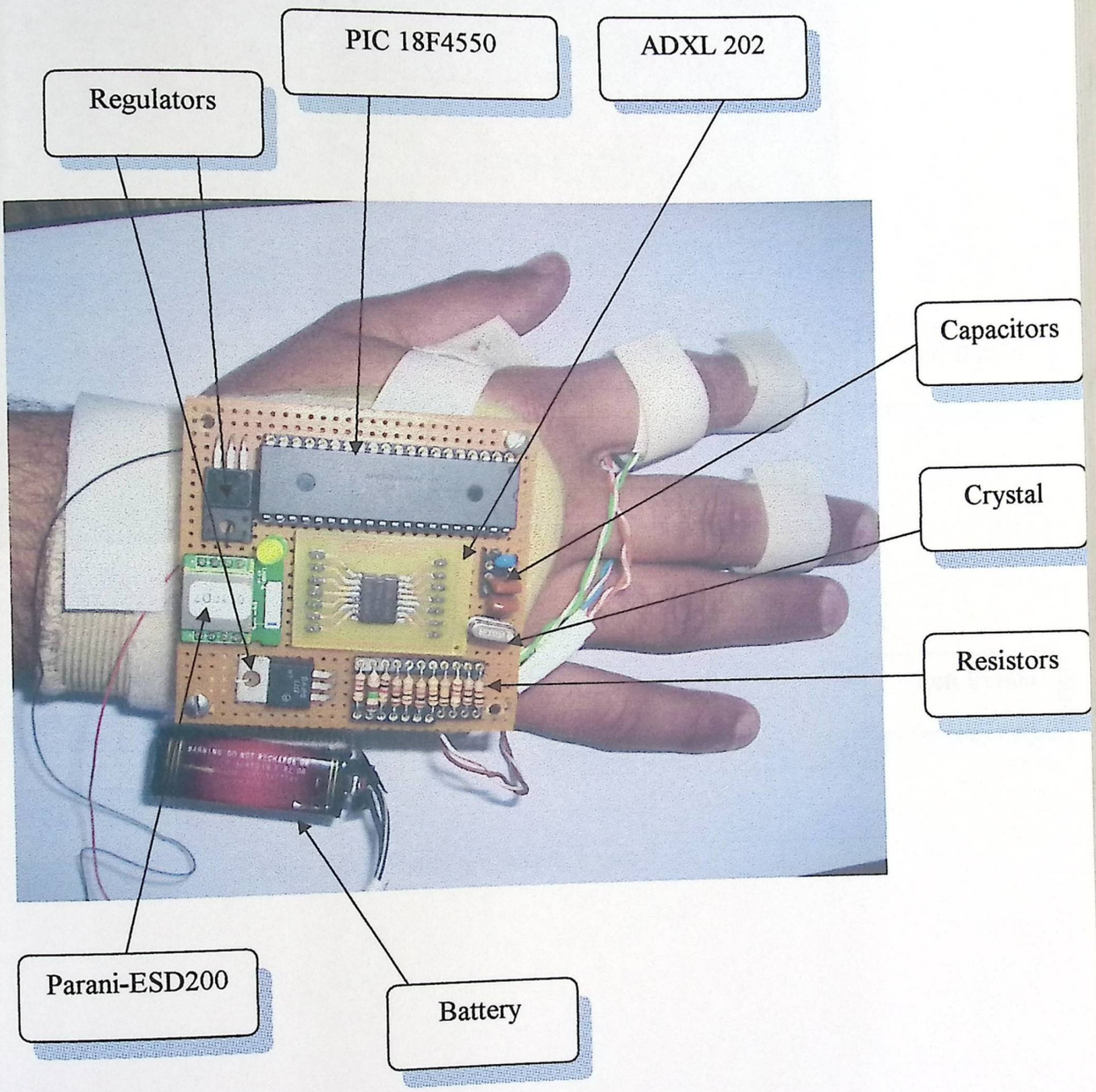


Figure 4-6: Hardware interface part 1

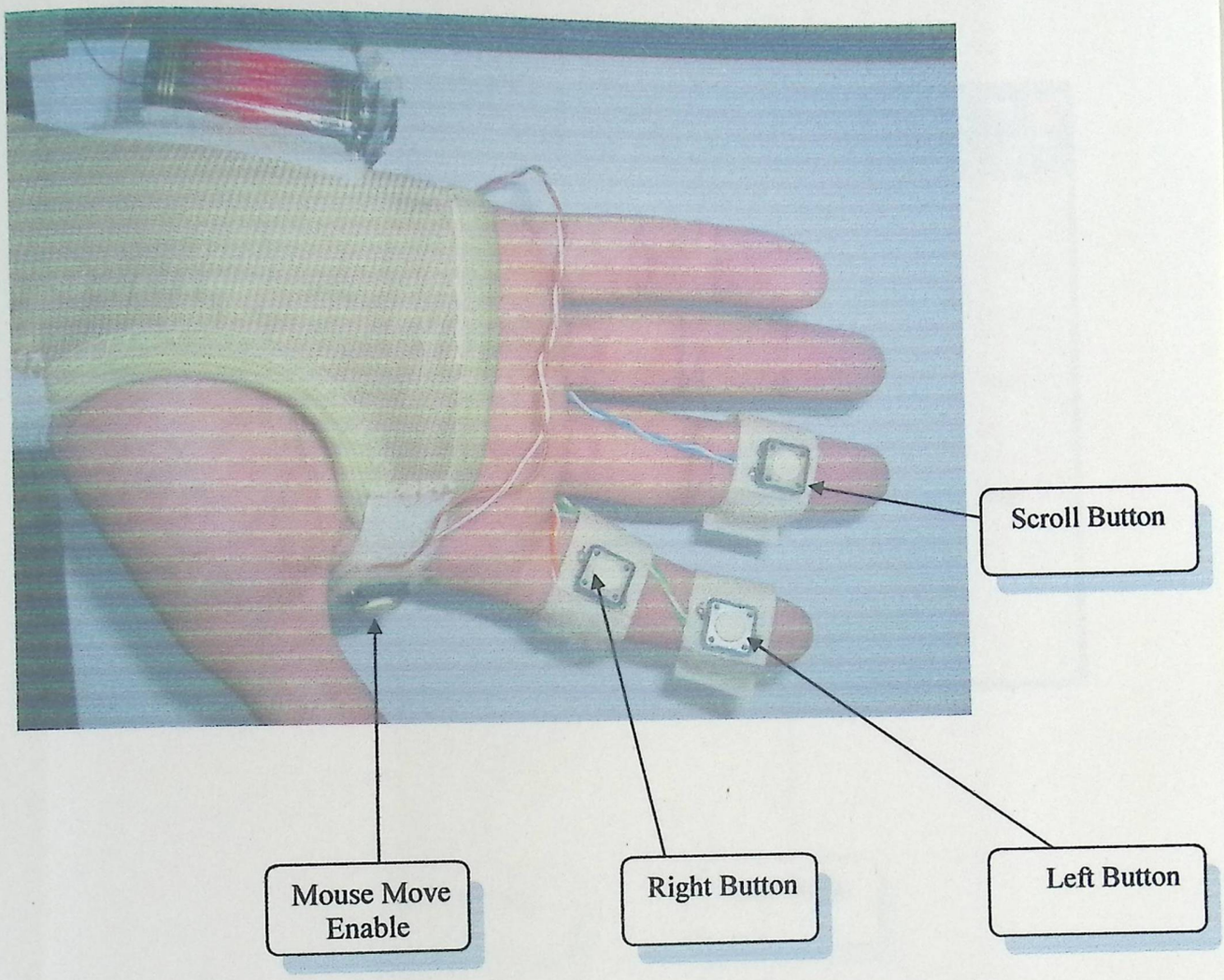


Figure 4-7: Hardware interface part 2

4.5.2 Software Interface

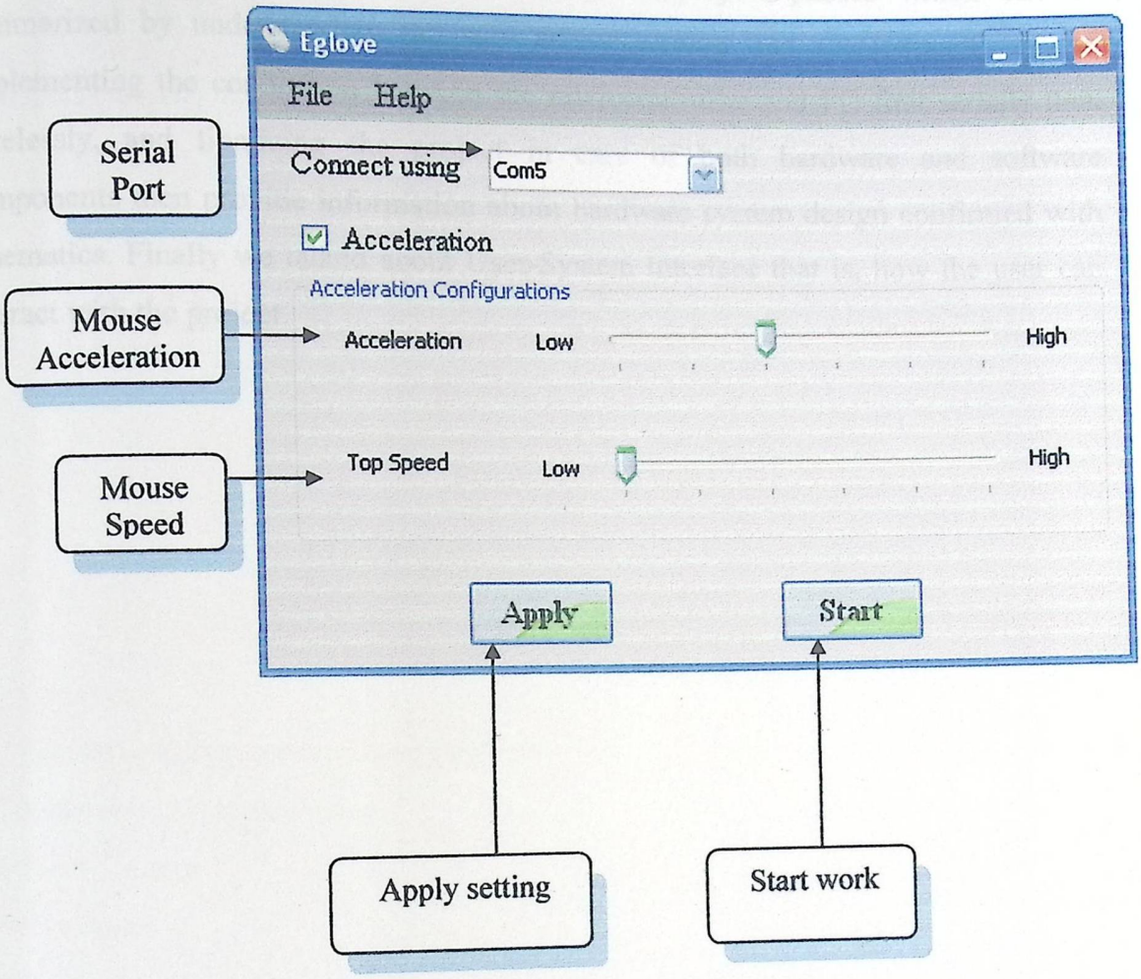


Figure 4-8: Software interface

4.6 Summary

This chapter has provided a description about project phases which can be summarized by understanding the way microcontroller and accelerometer works, implementing the connection between the microcontroller and PC first serially then wirelessly, and finalizing the project in case of both hardware and software components then provide information about hardware system design confirmed with schematics. Finally we talked about User-System Interface that is, how the user can interact with the project.

CHAPTER FIVE 5

Software System Design

5.1 Overview

In this chapter we will provide a description about our system software requirement specifications including a detailed description for the PIC program and PC program. Also we will present many algorithms and flowcharts that describe our system.

5.2 Software Requirement Specifications

Since our project interacts to PC to control cursor operations, it's needed to have two programs one resides in PIC and the other in PC. In this Section we will provide the software requirements needed to program PIC and PC.

5.2.1 Microcontroller software

For the microcontroller software, there are needs; the following paragraphs describe them.

- Win 95/98/ME/NT/2000/XP

The PIC microcontroller needs to work on one of these Windows versions in order to write the code in a text editor also used an appropriate program to transfer the code to microcontroller.

- Assembler or C Compiler

The PIC microcontroller programming deals with hexadecimal code, so there is a need for a C compiler or an assembler to convert the code from language code to machine code "hexadecimal".
The PIC code was written in C programming language according to PIC programming specifications which can be found in MPLAB C18 C Compiler Libraries provided by *MICROCHIP*.
The compiler we used is the MPLAB 7.13 version C compiler in which one can write the C code for a specific PIC, then after compiling a hex code file will be generated.

- SuperPro program

SuperPro is a program used to load hex files from PC to a PIC specified by user; it requires its specific programmer which is connected to PC through USB. When user put PIC in the programmer he can load the hex file to the SuperPro program then he can program PIC, after programming, the hex code will be loaded completely to PIC.

5.2.1 PC software

In the PC side we need a program that receives packets from glove, analyzes it, and controls cursor according to these packets. The PC program was written in VB.NET programming language.

We need also a program that deals with Bluetooth dongle that is connected to PC which receives packets from the Parani Bluetooth. This program is BlueSoleil.

BlueSoleil is a Windows 2000/XP/Windows Vista Bluetooth stack that allows communication with Bluetooth enabled devices. It is often bundled with Bluetooth dongles.

The program is also available as a standalone purchase from the vendor's website. Regardless of whether the bundled or the standalone version is purchased, the software enforces licensing restrictions which tie it to the address of a specific Bluetooth dongle.

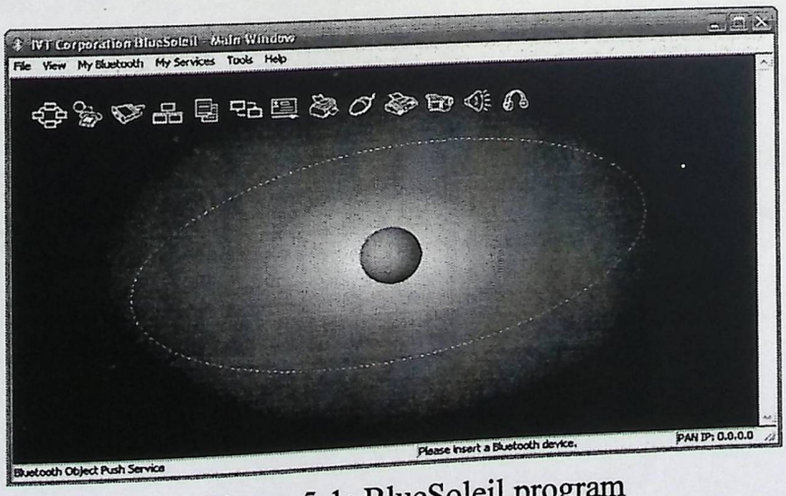


Figure 5-1: BlueSoleil program

5.3 Software Implementation

Recall that our project is to control mouse using an electronic glove on which sensors are attached. To achieve that, two programs should be running; one in the PIC and the other in PC on which cursor will be controlled.

The program that resides in the microcontroller (PIC) will accept accelerometer and push buttons readings and then form packets that will be sent to PC through the Bluetooth module. These packets describe the sensors' status, and according to those packets cursor will be controlled.

The system works as the following:

- If user want to move cursor then he should push on "Movement Push Button" and tilt his hand according to the following conditions:
    - If user wants to move cursor right then he should tilt his hand right.
    - If user wants to move cursor left then he should tilt his hand left.
    - If user wants to move cursor up then he should tilt his hand up.
    - If user wants to move cursor left then he should tilt his hand left.
- If user wants to click mouse left he should click on "Left Click Push Button"
- If user wants to click mouse right he should click on "Right Click Push Button"
- If user wants to scroll mouse he should click on "Scroll Push Button" and tilt his hand according to above description.
- If user wants to drag an object he should press on "Left Click Push Button" and still press for a short period, then tilt his hand according to above moving conditions, then release the push button.

For simplicity and speed each packet the PIC sends to PC contains one byte of a specified format. All needed information for PC program are to be placed in this packet.

	Scroll	Y Enable	X Enable	Up / Down	Right / Left	Right Click	Left Click
X	0/1	0/1	0/1	0/1	0/1	0/1	0/1

X: don't care

Figure 5-2: PIC Packet Format

Figure 5-2 shows the format for each packet which PIC sends. The byte packet is initially zeros '00000000' (it is initialized after each sending). After each checking operation PIC performs for movement, left, right, and scroll buttons packet is modified according to each checking by doing an ORing operation between specific binary value with packet. The following table shows the binary values specified by each case.

Modifying Packet								**Case**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
0	1	0	0	0	0	0	0	Scroll push button pressed
0	0	1	0	1	0	0	0	Moving Up
0	0	1	0	0	0	0	0	Moving down
0	0	0	1	0	1	0	0	Moving right
0	0	0	1	0	0	0	0	Moving left
0	0	0	0	0	0	1	0	Right push button pressed
0	0	0	0	0	0	0	1	Left push button pressed

Table 5-1: Modification packets

The PIC program acts as the following:

Step 0: Initialize Bluetooth connection

Step 1: Initialize send packet

Step 2: Calculate DutX and DutyY

Step 3: Check for hand tilting (for mouse movement)

Step 4: Check for left, right and scroll

Step 5: Go to step 1

On the other hand the PC program acts according to the following algorithm:

Step 1: Listen to serial port

Step 2: Receive packet and convert it to array of bits

Step 3: Check for cursor movement

Step 4: Check for cursor buttons' actions

Step 5: Go to step 1

5.4 Flowcharts

In this section we will provide flowchart schematics that describe the system operation; a general flow chart that describes the overall system will be provided, then detailed flow charts that describe both programs in PIC and PC will be also presented.

The following flowchart describes the overall system work.

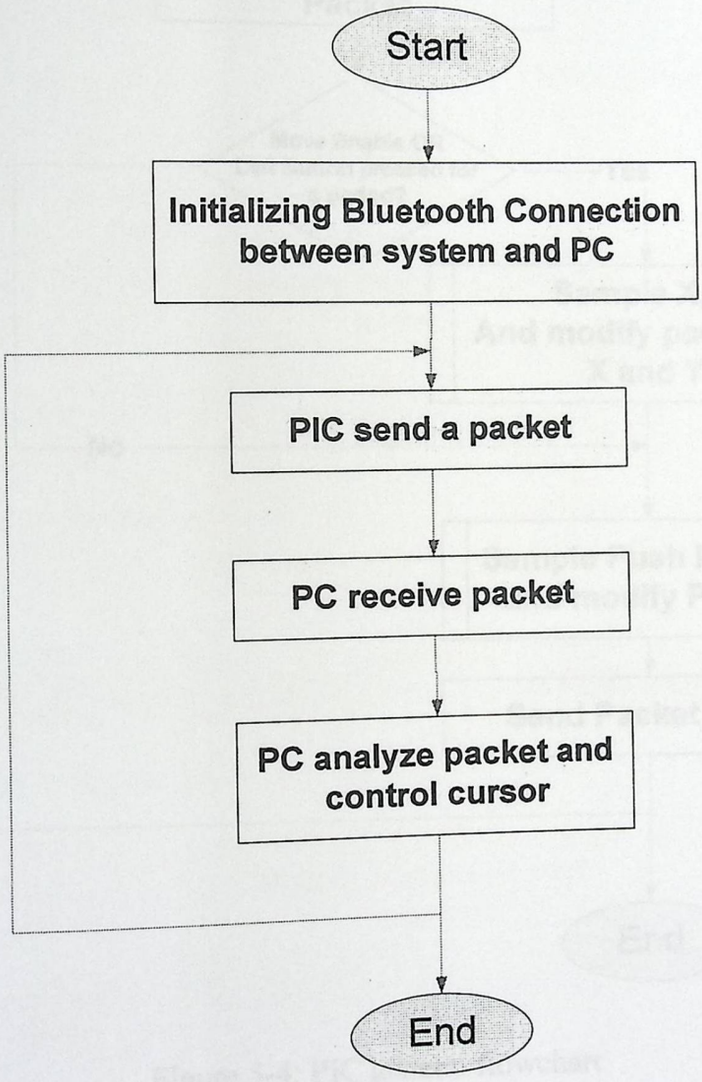


Figure 5-3: General System Flowchart

The following figure shows the general flowchart of PIC program.

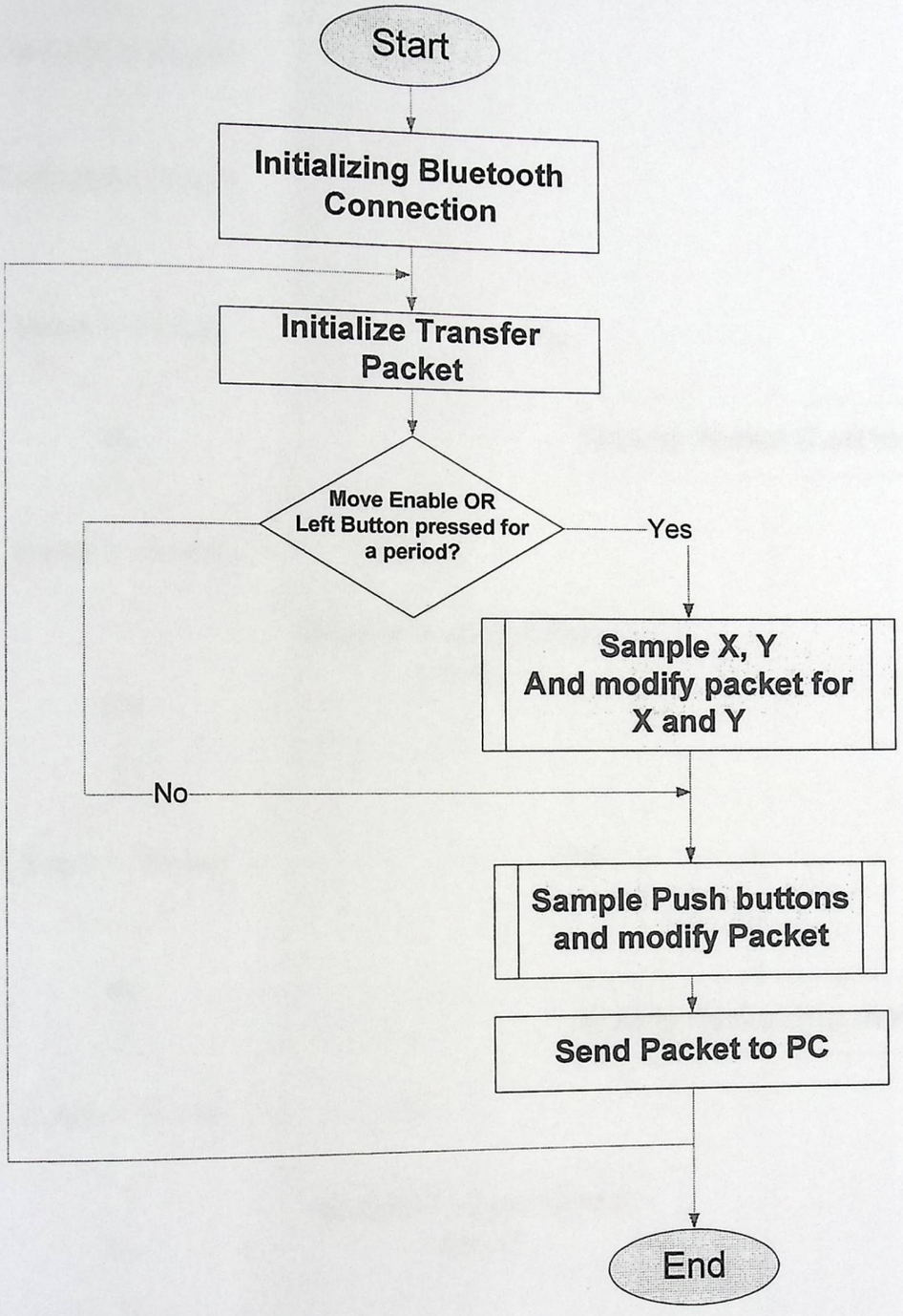


Figure 5-4: PIC general flowchart

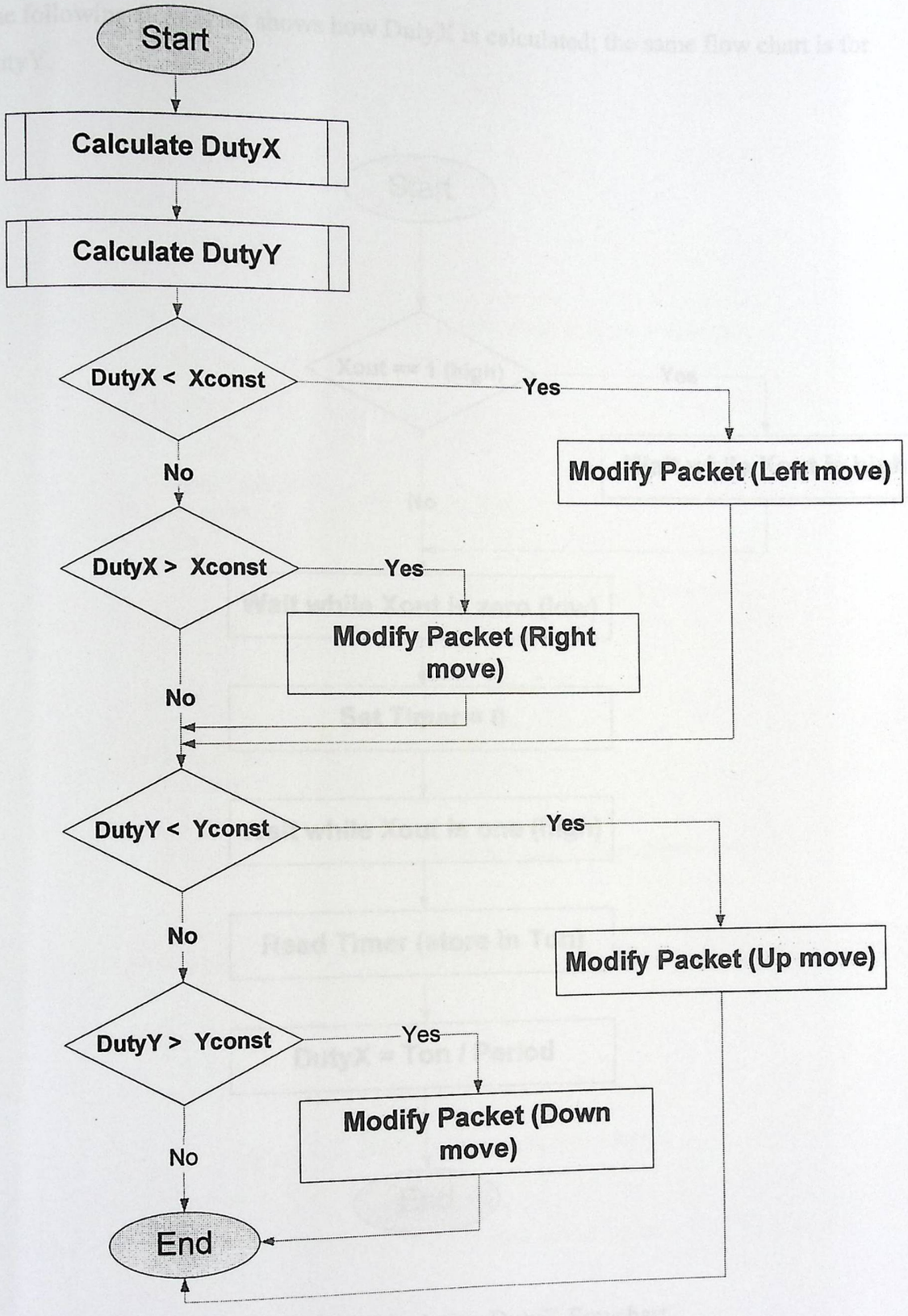


Figure 5-5: Sample X, Y flowchart in PIC

The following flow chart shows how DutyX is calculated; the same flow chart is for DutyY.

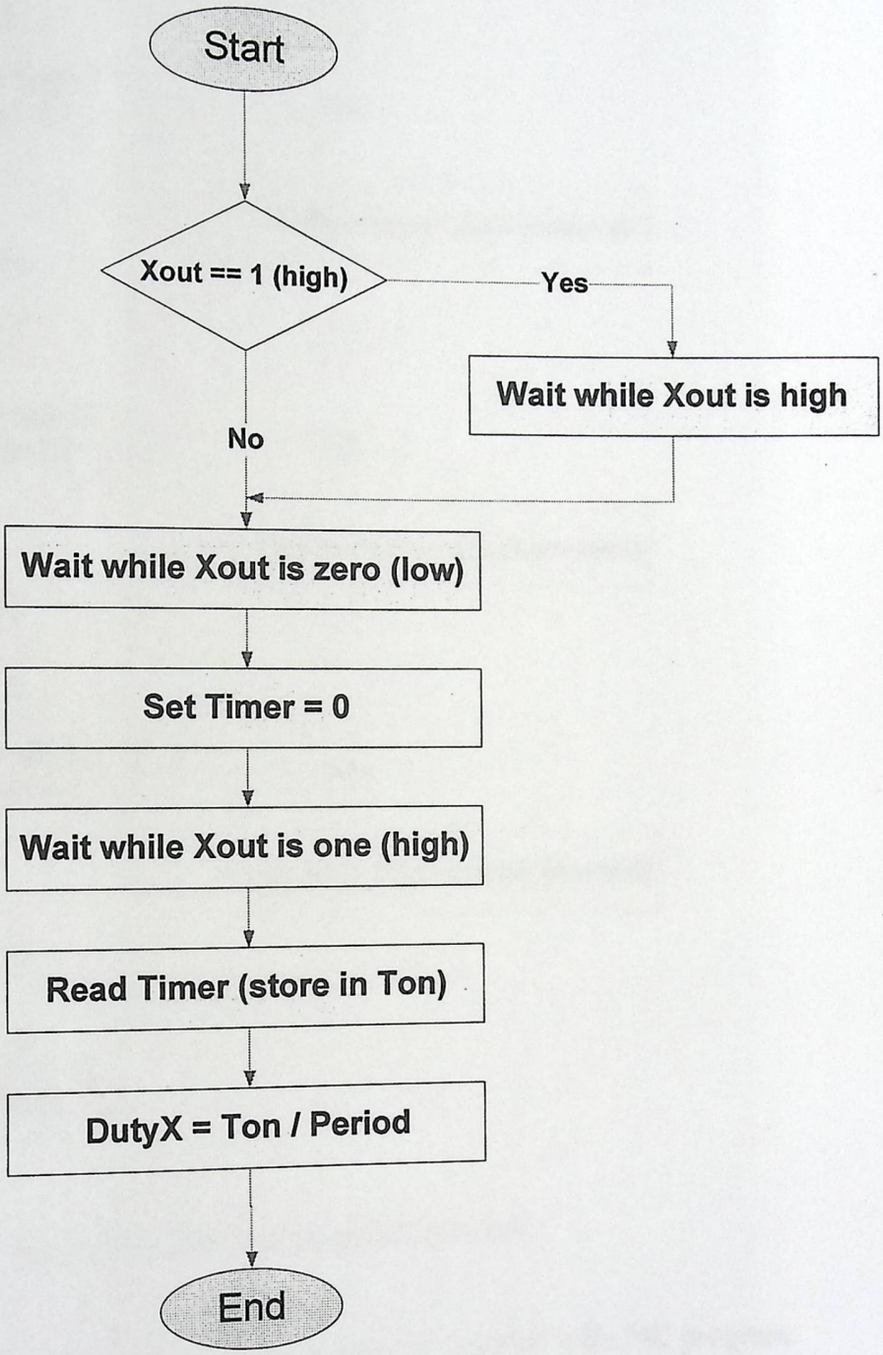


Figure 5-6: Calculating DutyX flowchart

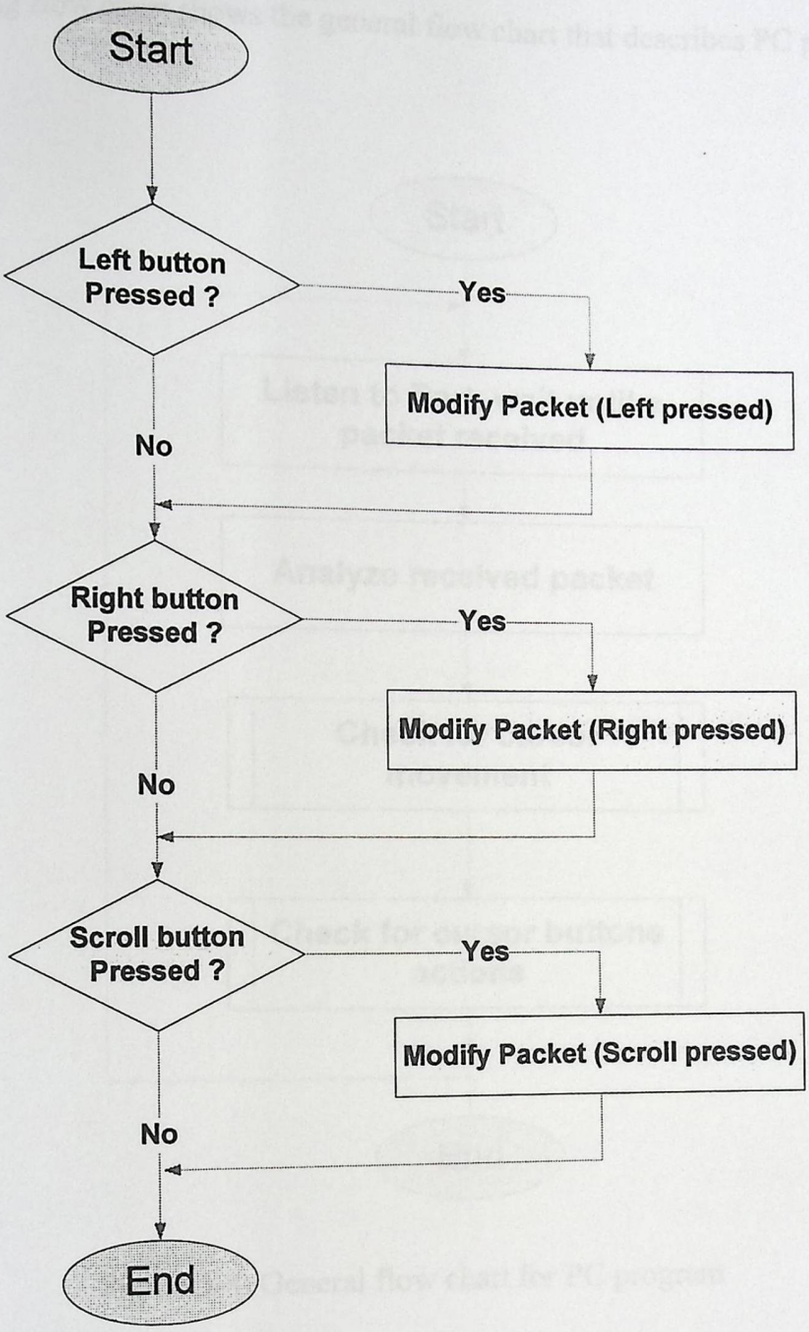


Figure 5-7: Buttons sampling flowchart

The above flow chart shows how push buttons data are sampled with PIC program.

The following flow chart shows the general flow chart that describes PC program.

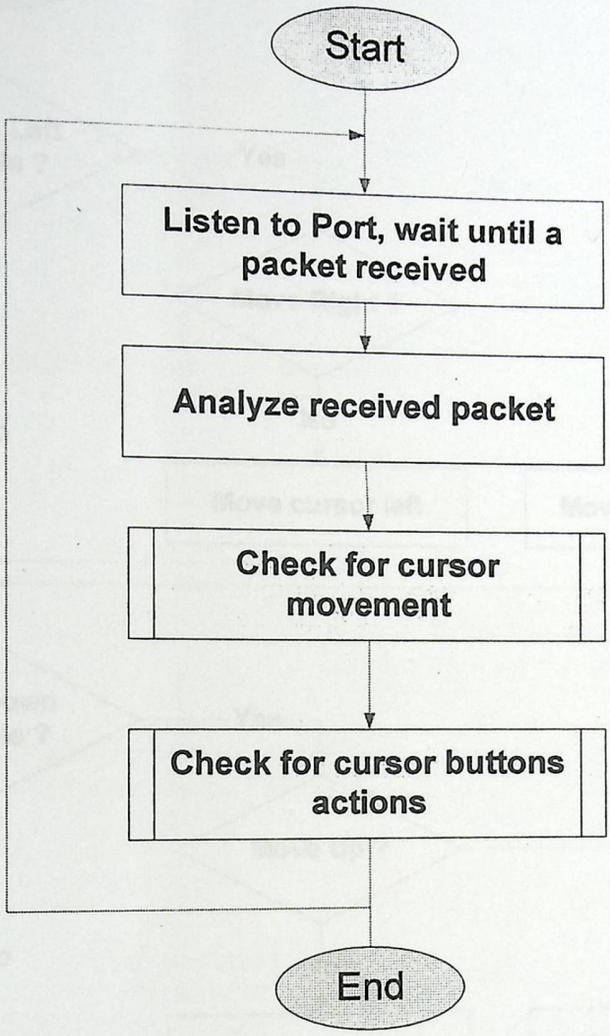


Figure 5-8: General flow chart for PC program

The following flow chart shows how PC program checks for mouse movement.

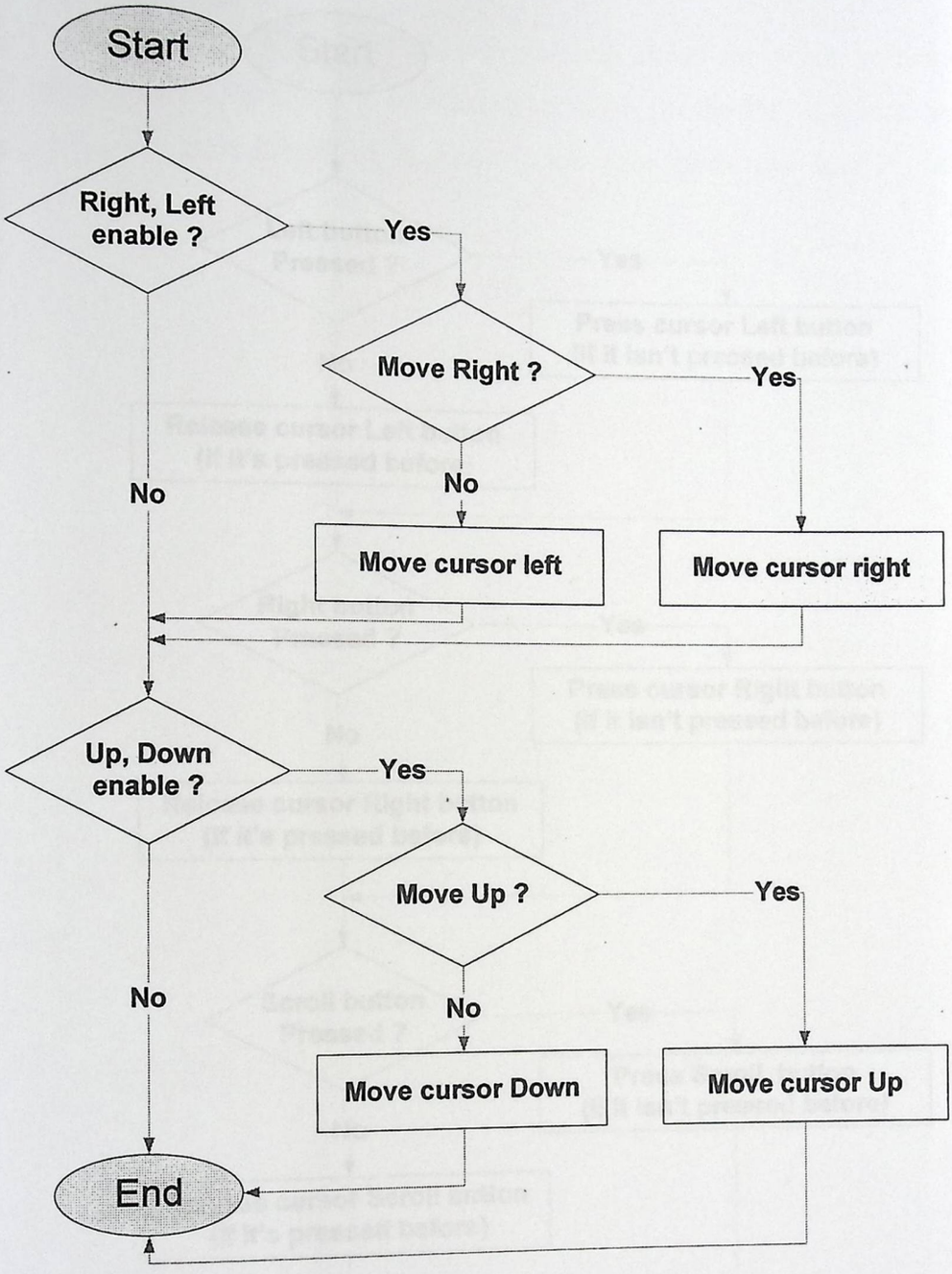


Figure 5-9: Checking mouse movement flowchart

The following flow chart shows how PC program checks for push buttons' actions.

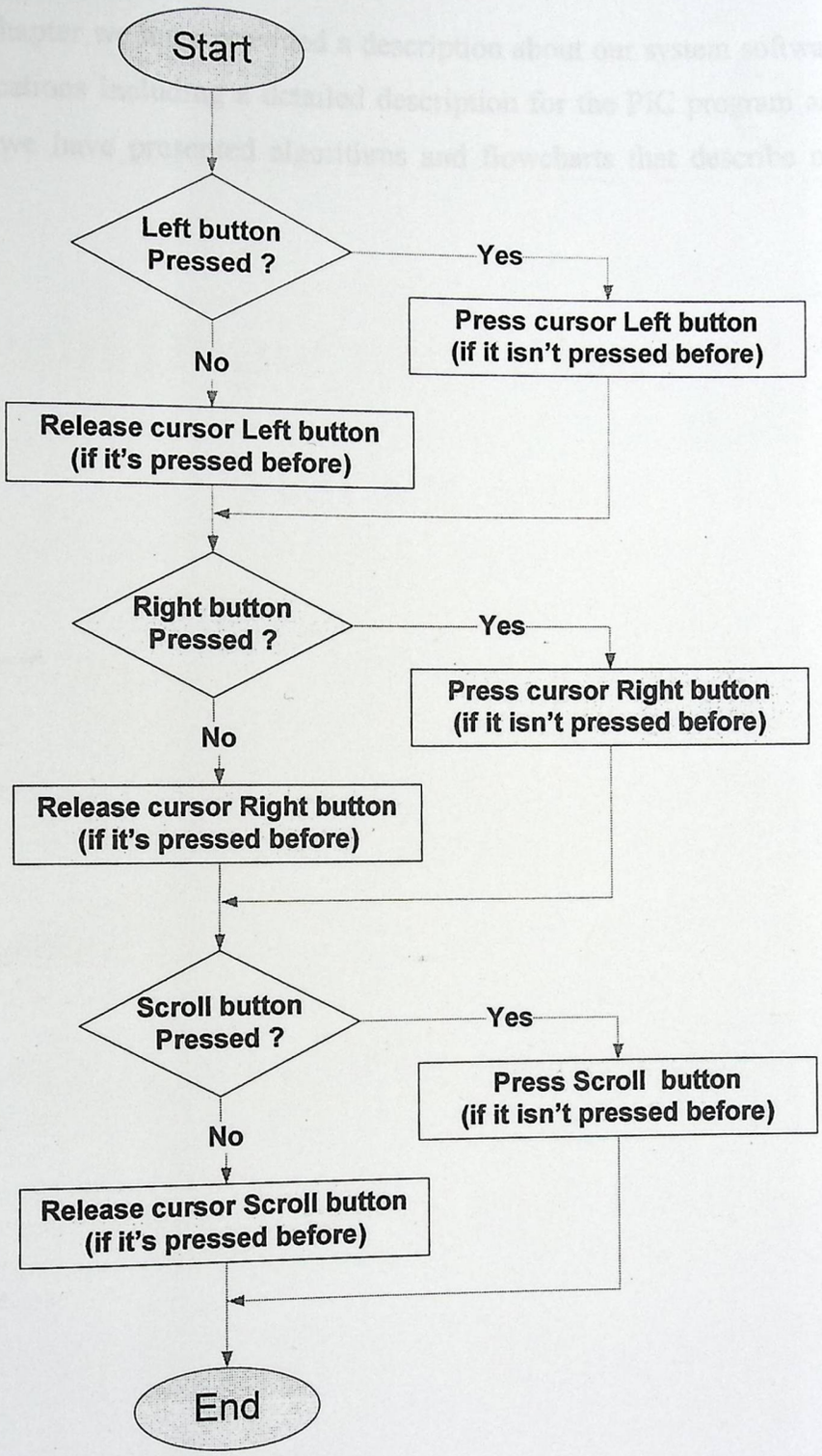


Figure 5-10: Checking mouse buttons flowchart

5.5 Summary

By the end of this chapter we have provided a description about our system software requirement specifications including a detailed description for the PIC program and PC program. Also we have presented algorithms and flowcharts that describe our system.

CHAPTER SIX 6

System Implementation and Testing

6.1 Overview

This chapter demonstrates the methods and procedures used to test and examine the system operation and behavior. System testing is an important and critical step in implementing a system. It senses the effectiveness of that system just before introducing it to its users.

This system has more than one issue to be tested. Some testing parts reflect a software, or hardware case. Also, testing procedures concentrate on a single device independent from the overall system.

After finishing the design of the system and drawing the system schematic, the next step was to test each part individually, and then implement the system using the wire wrapping.

6.2 Unit Testing

At this stage each unit was tested independently to ensure that the component realizes its specified function. This way of testing simplifies trouble shooting detection.

6.2.1 Testing the Accelerometer

Testing the accelerometer involves connecting its circuit; which includes a resistor and two capacitors and then taking the output from Xout and Yout from the accelerometer to the oscilloscope which shows us the output at the form of duty cycles.

The following figure shows the outputs of accelerometer on the oscilloscope. The upper output is for Xout and the lower is for Yout.

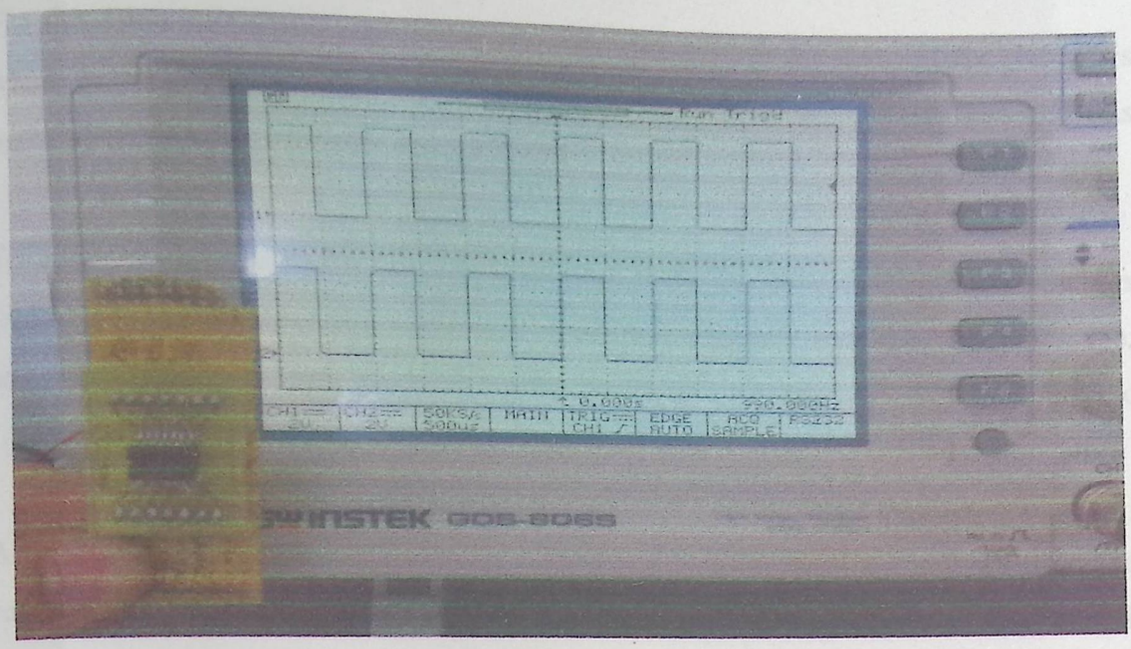


Figure 6-1: Accelerometer output testing

6.2.2 Testing the Bluetooth module

The Bluetooth module that we used is the Parani-ESD200. The factory default mode of module is mode 0 (more information about other modes in the appendix). In this mode there is no response when power on or software reset, and Parani-ESD is just waiting for AT command input. Neither master nor slave is assigned to Parani-ESD in mode0. User can change the configuration parameters of Parani-ESD in this mode.

The factory default serial port configuration of this device is presented in the following table. In our system we used the default settings of Bluetooth.

Serial Port Setting	Values
Baud Rate	9600 bits/second
Data Bits	8
Parity	No parity
Stop Bit	1
Hardware Flow Control	Use

Table 6-1: Parani-ESD default serial port configurations

For this Bluetooth when resets it sends "OK" message in the form of six characters which is "\r\nok\r\n" where "\r" is Carriage return and "\n" is Line feed and this message must be received by system (PIC).

To establish a connection, an AT command must be sent to Bluetooth which is "AT+BTSCAN\r", after this command, Bluetooth responds with the "OK" message; however, Bluetooth may responds with "ERROR" message, at this moment the "AT+BTSCAN\r" command must be resent to Bluetooth until it responds with "OK".

After it accepts the "AT+BTSCAN\r" command it waits for a connection, once the connection is established it sends a message of 24 characters that contains CONNECT and the address of connected device of 12 characters, this message will be in the form "\r\nCONNECT 112233445566\r\n". After that we can deal with Bluetooth as a serial port; write to it characters or received characters from it.

In order to test Bluetooth it was connected to the serial port (COM1) of PC according to the following circuit.

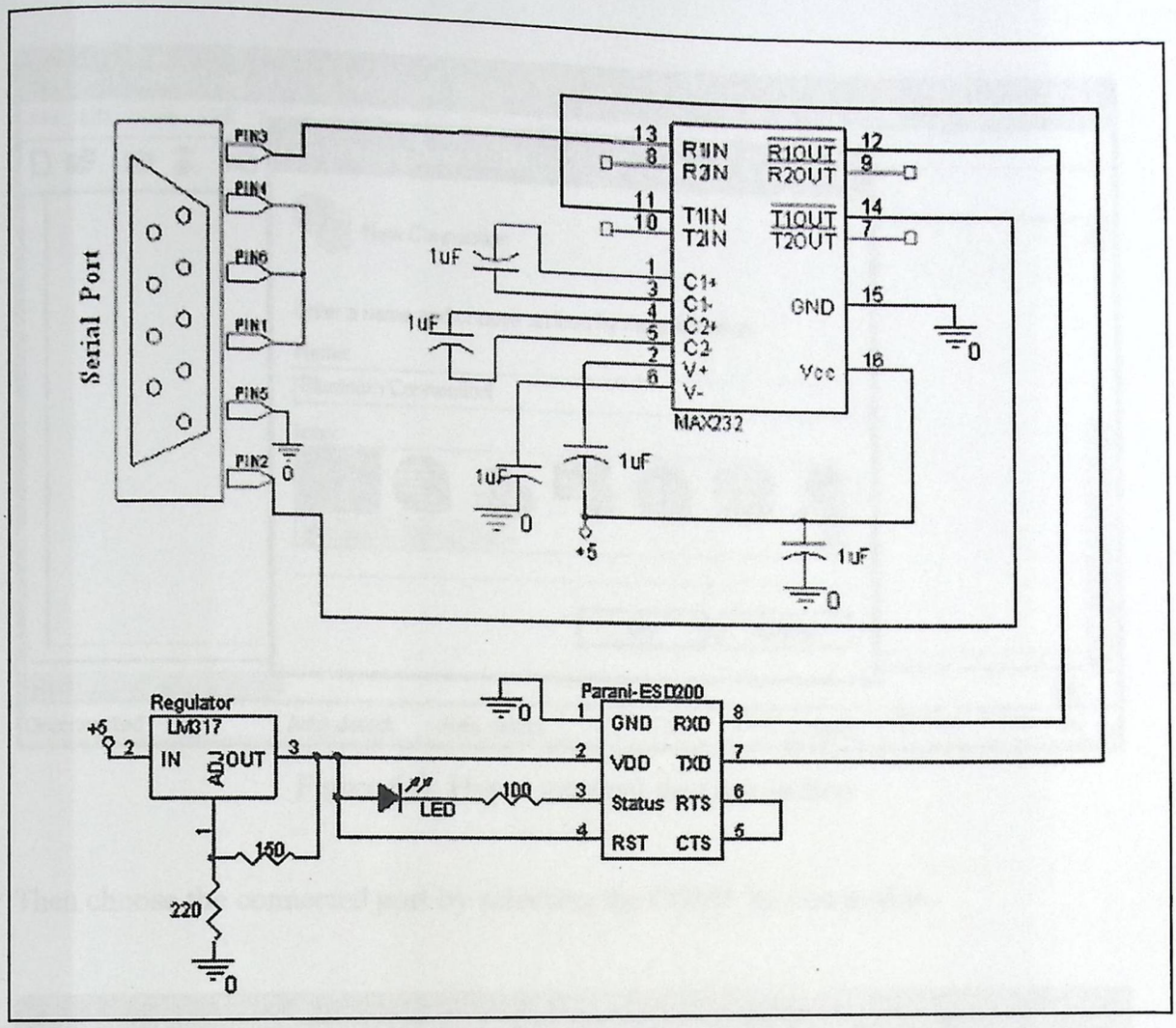


Figure 6-2: Bluetooth testing circuit

To test Bluetooth we used the HyperTerminal program in order to verify its operation. The following steps demonstrate how Bluetooth was tested.

1. Establish a hyper terminal connection with COM1

In this step a hyper terminal connection with COM1 should be established, with the same default serial port configurations of Bluetooth summarized in Table 6-1. However, we ignored the use of flow control.

First, start the hyper terminal and make a new connection as shown in figure 6-3 and enter the name of connection.

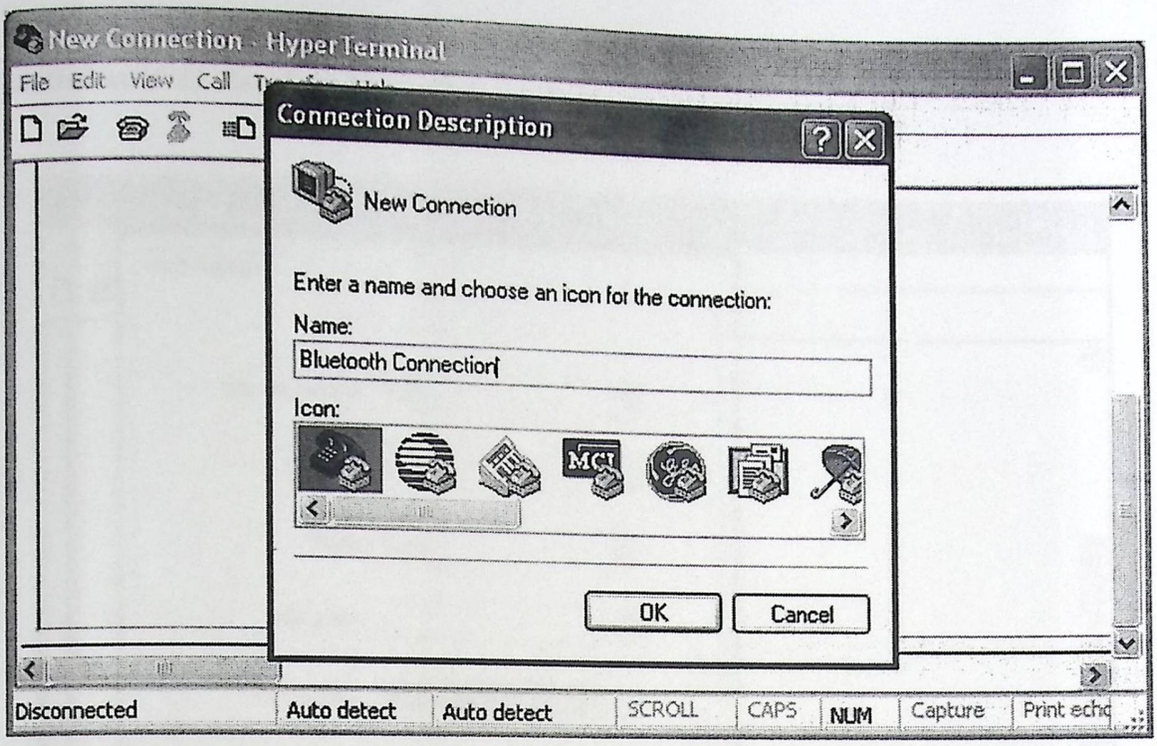


Figure 6-3: Hyper terminal new connection

Then choose the connected port by selecting the COM1 for connection.

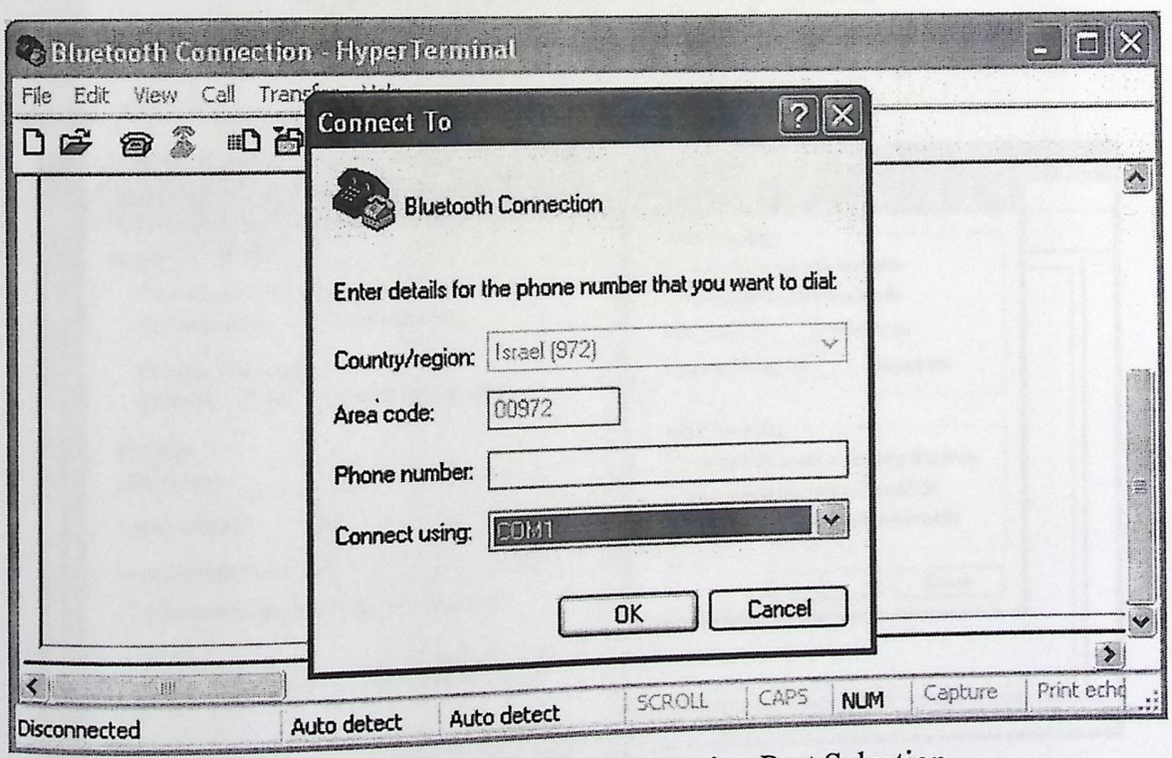


Figure 6-4: Hyper terminal connection Port Selection

Edit the COM1 properties by choosing the Baud rate 9600 bit per second and flow control none as shown in figure 6-5

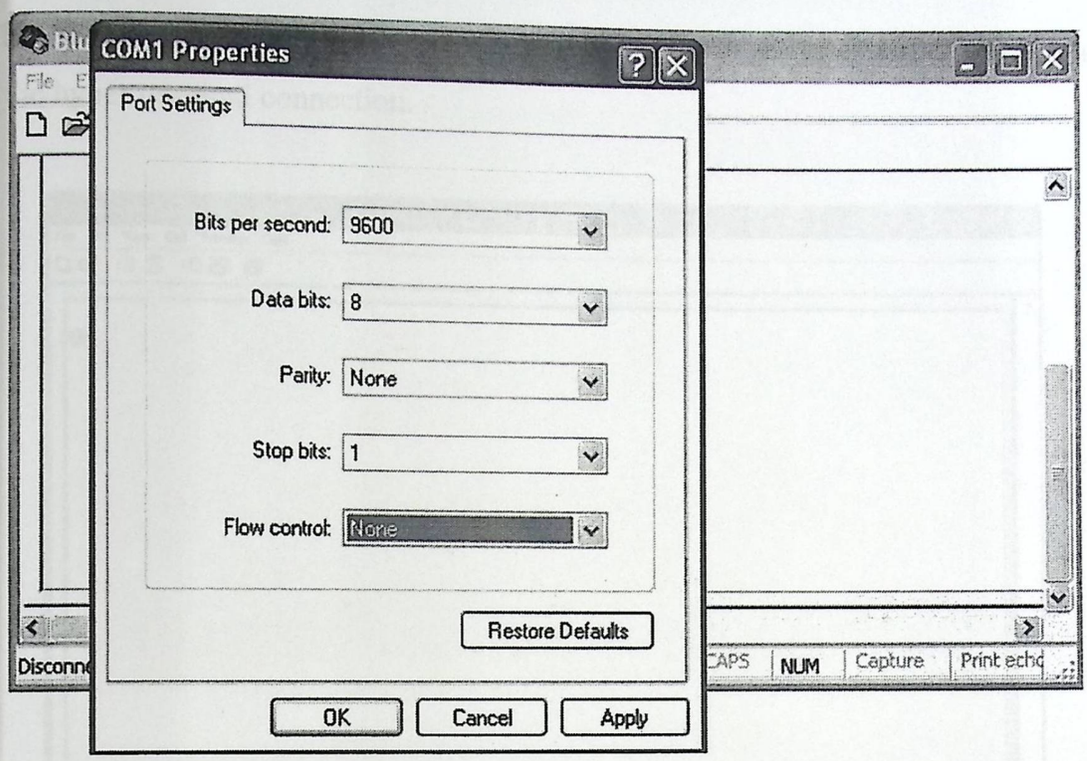


Figure 6-5: Hyper terminal COM properties

Edit the ASCII properties to show characters as normal text

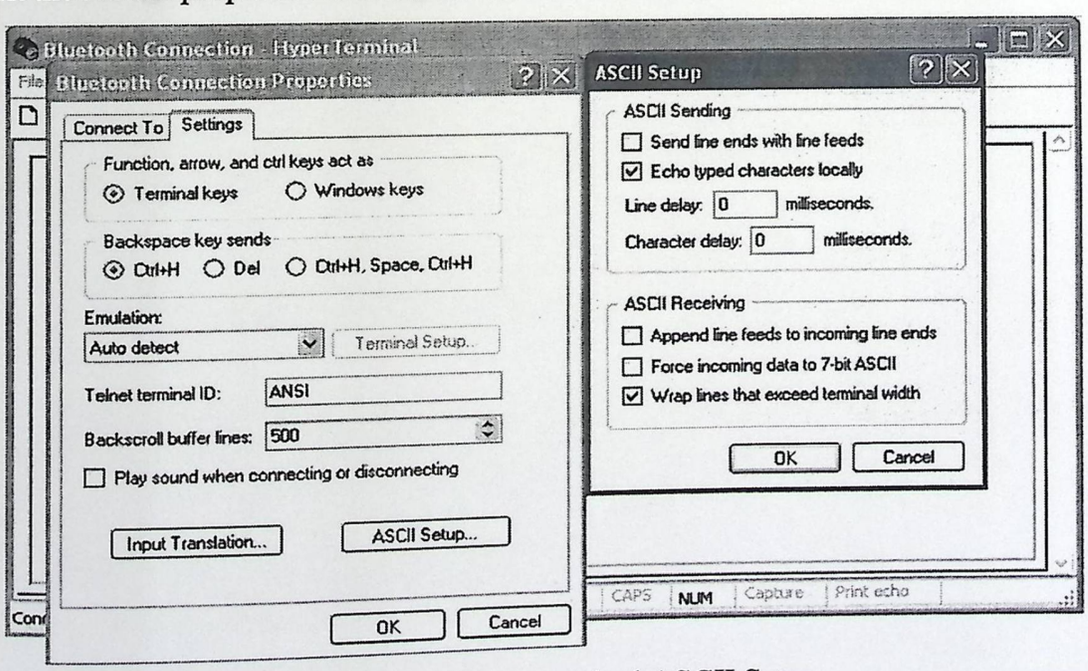


Figure 6-6: Hyper terminal ASCII Setup

2. Power on the Bluetooth

When Parani Bluetooth is powered on, it sends the "OK" message which can be seen in the hyper terminal connection.

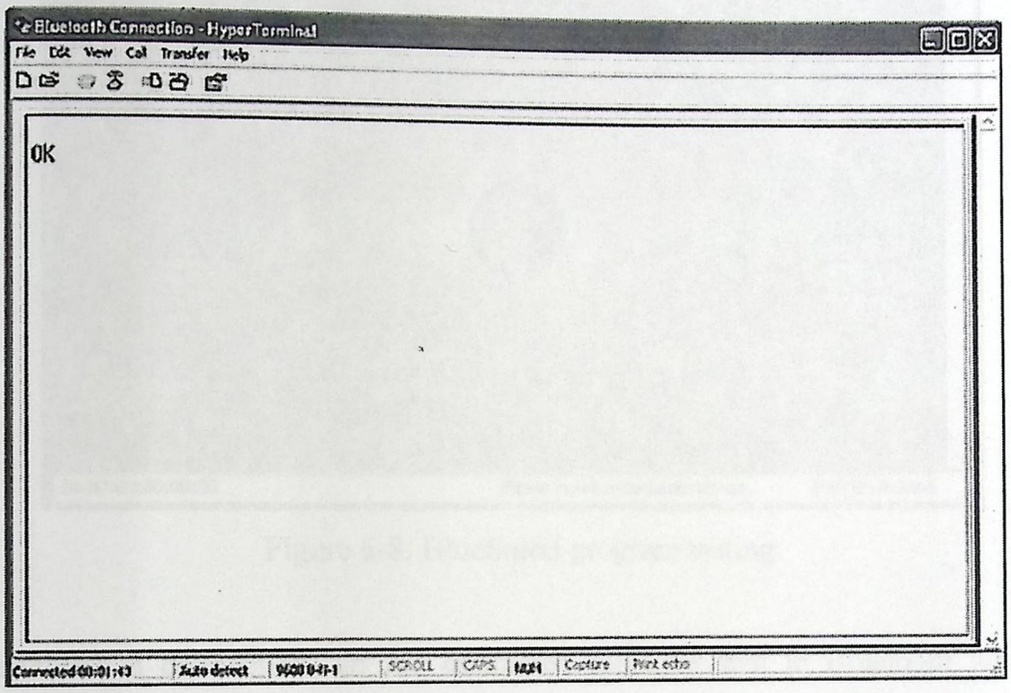


Figure 6-7: OK message from Parani-ESD 200 to hyper terminal

3. Run BlueSoleil program to connect with Bluetooth

At this step use the BlueSoleil program to search for Bluetooth device.

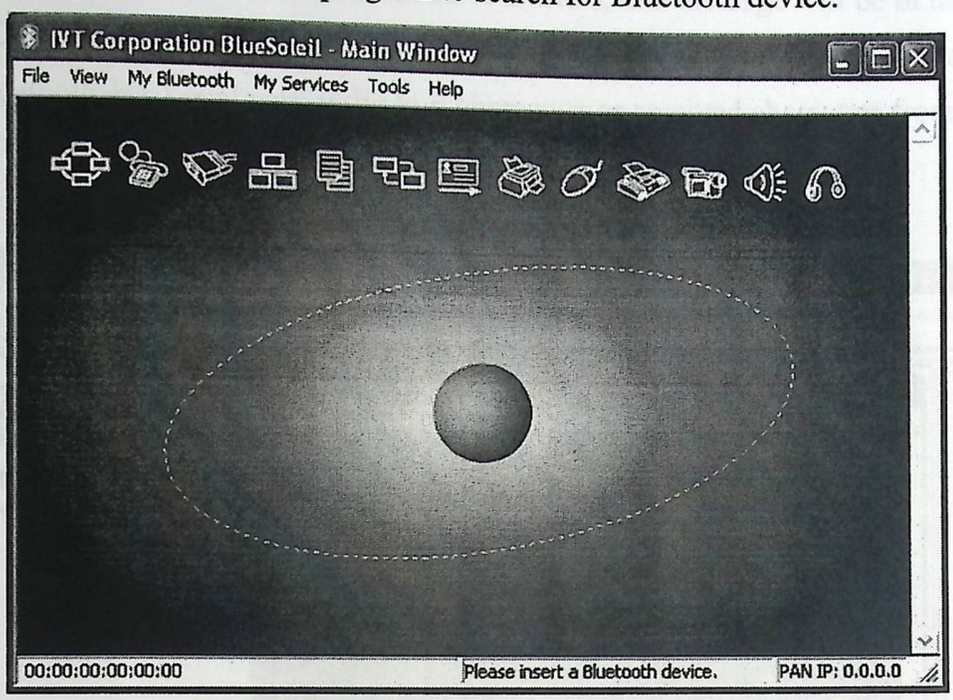


Figure 6-8: BlueSoleil program testing

To establish a connection, an AT command must be sent to Bluetooth which is "AT+BTSCAN\r", after this command Bluetooth responds with the "OK" message.

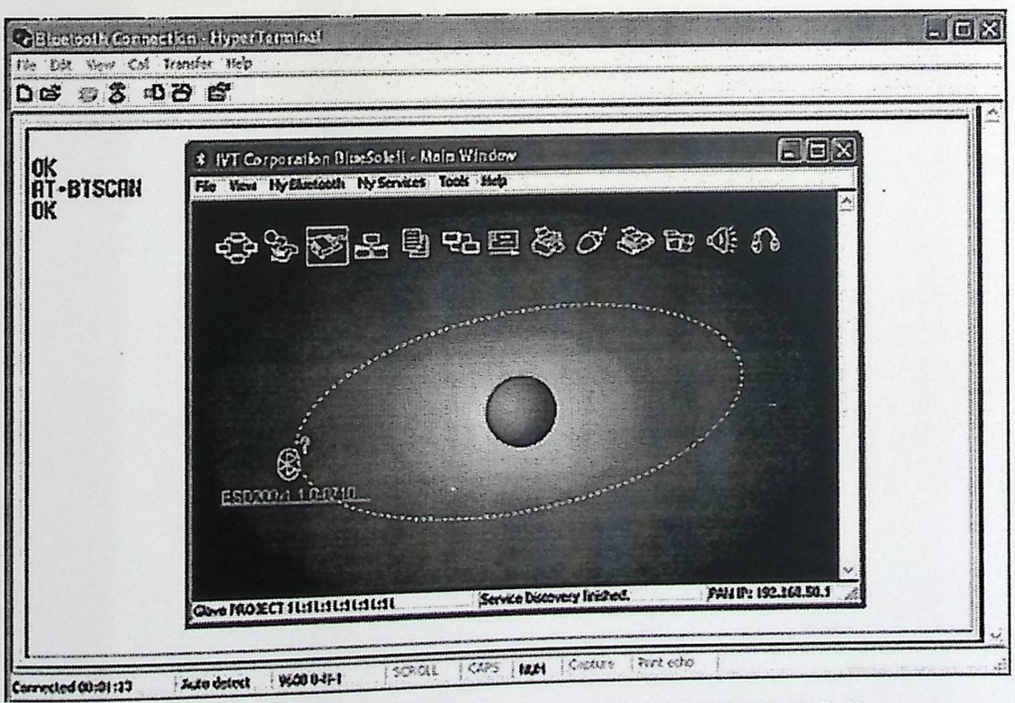


Figure 6-9: Bluetooth response to "AT+BTSCAN\r"

After it accepts the "AT+BTSCAN" command it waits for a connection, once the connection is established it sends a message of 24 characters that contains CONNECT and the address of connected device of 12 characters, this message will be in the form "\r\n CONNECT 112233445566\r\n". As shown in figure 6-10 after that we can deal with Bluetooth as a serial port; write to it characters or received characters from it.

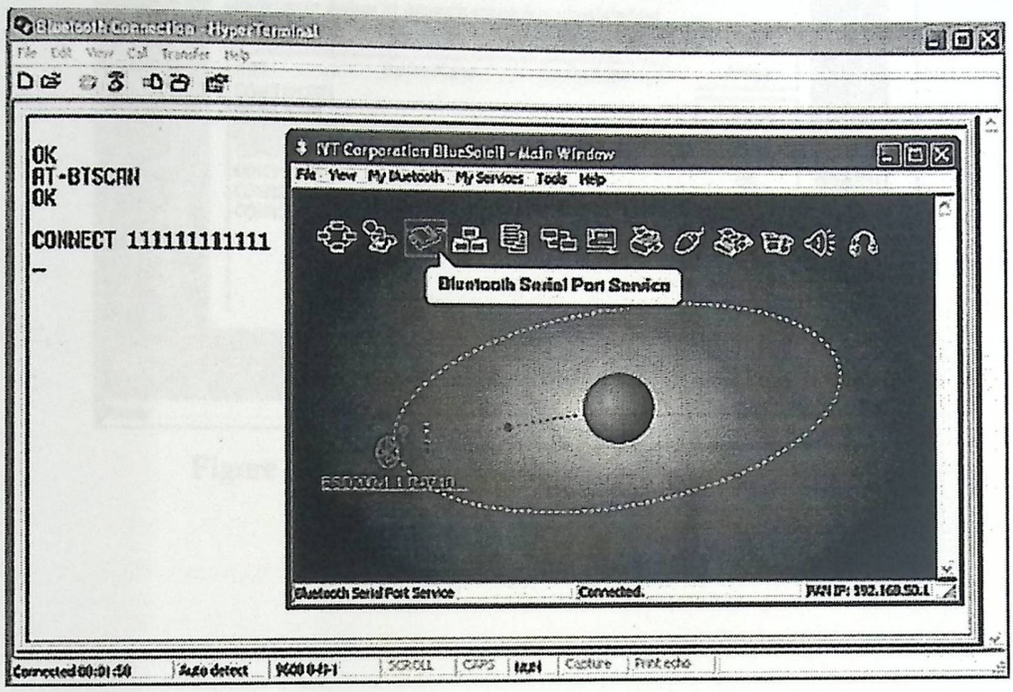


Figure 6-10: Bluetooth is connected to BlueSoleil

User can change the port for receiving by choosing tool in BlueSoleil program menu and select Configuration -> Quick connect and assign the port name for Bluetooth device as shown in figure 6-11.

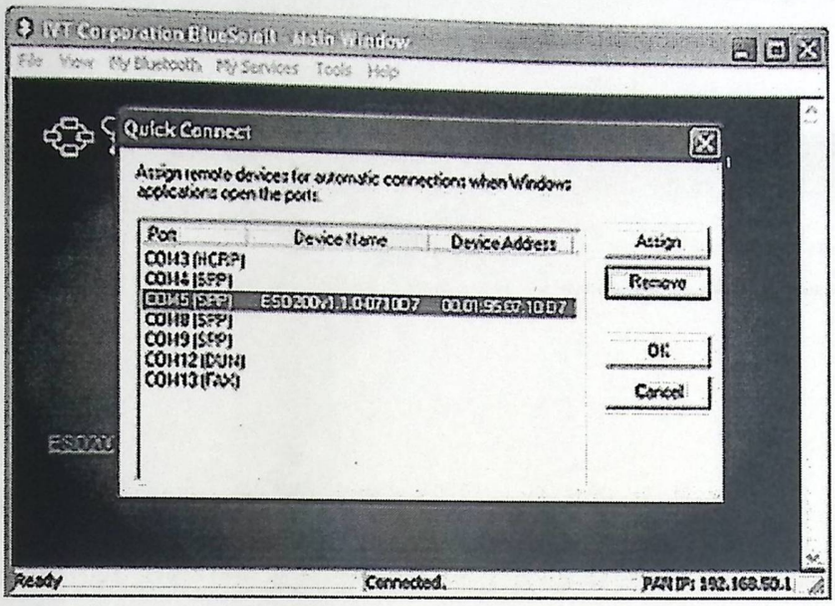


Figure 6-11: Port change in BlueSoleil program

4. Send characters to an existing program

We used a Visual Basic .NET program; the user sends hello massage by hyper terminal to Bluetooth device, the Bluetooth device sends this massage serially. The massage is shown in a VB.net form.

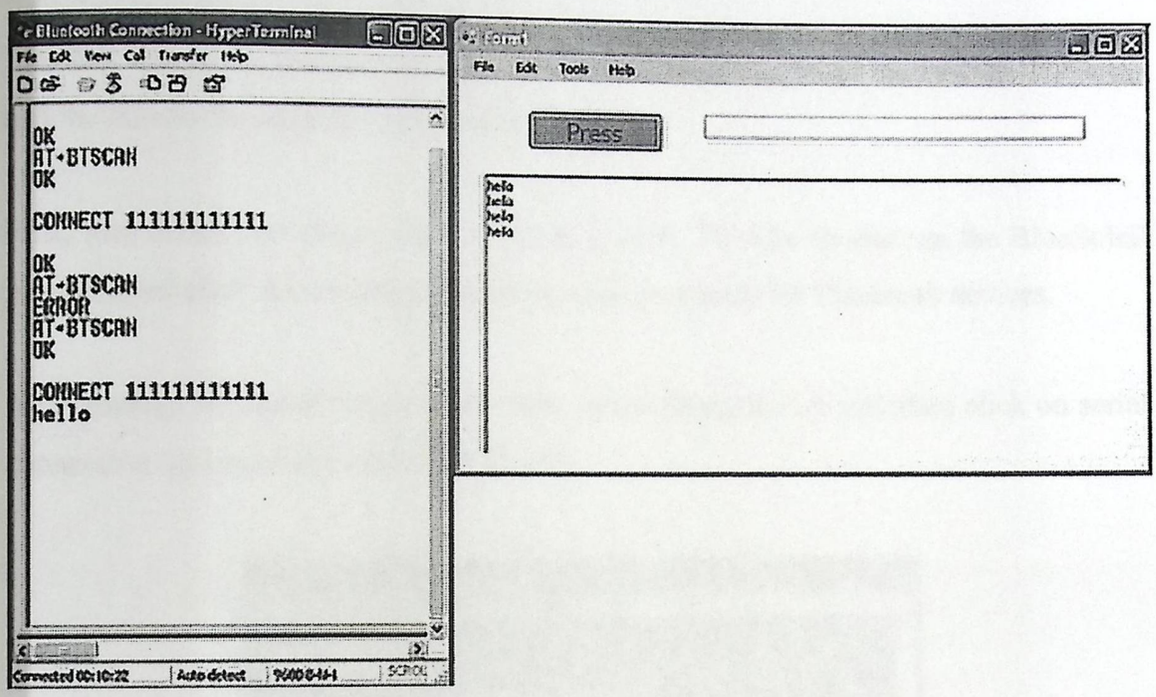


Figure 6-12: Sending through Bluetooth

6.3 Integrated testing

In this section we will test the overall system including circuit on hand and the software implemented in PC and PIC.

6.3.1 Connection Testing

The first step to operate system is to establish a connection between system Bluetooth and the Bluetooth adapter connected to PC.

First, user should set the power on for the system. Then he should run the BlueSoleil program and click on the orange circle in order to search for Bluetooth devices.

After finding Bluetooth user should click on the found device and then click on serial connection option as shown in figure 6-13.

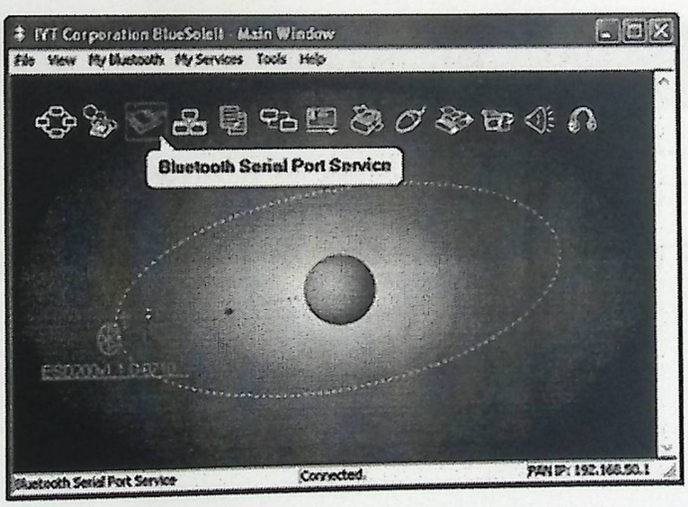


Figure 6-13: Bluetooth is connected

After establishing the connection which takes a few seconds the Green LED associated with Bluetooth will turn on. This indicates that the Bluetooth is connected; if it is not then Bluetooth is not connected.

To test the output of PIC program and the operation of PC program a small function was added that shows the received byte packets, and also shows the operation of cursor by writing it on a text box. The actions of mouse cursor also show the output of PC program.

6.3.2 Mouse cursor testing

In order to move mouse cursor the "Movement Enable push button" must be pressed. This was done in order that user when he wants; he can move his hand without moving mouse cursor.

To move mouse cursor right one should tilt his hand right. The output can be shown in figure 6-14; we can see that the byte was sent is "40" which is equal to "00101000".

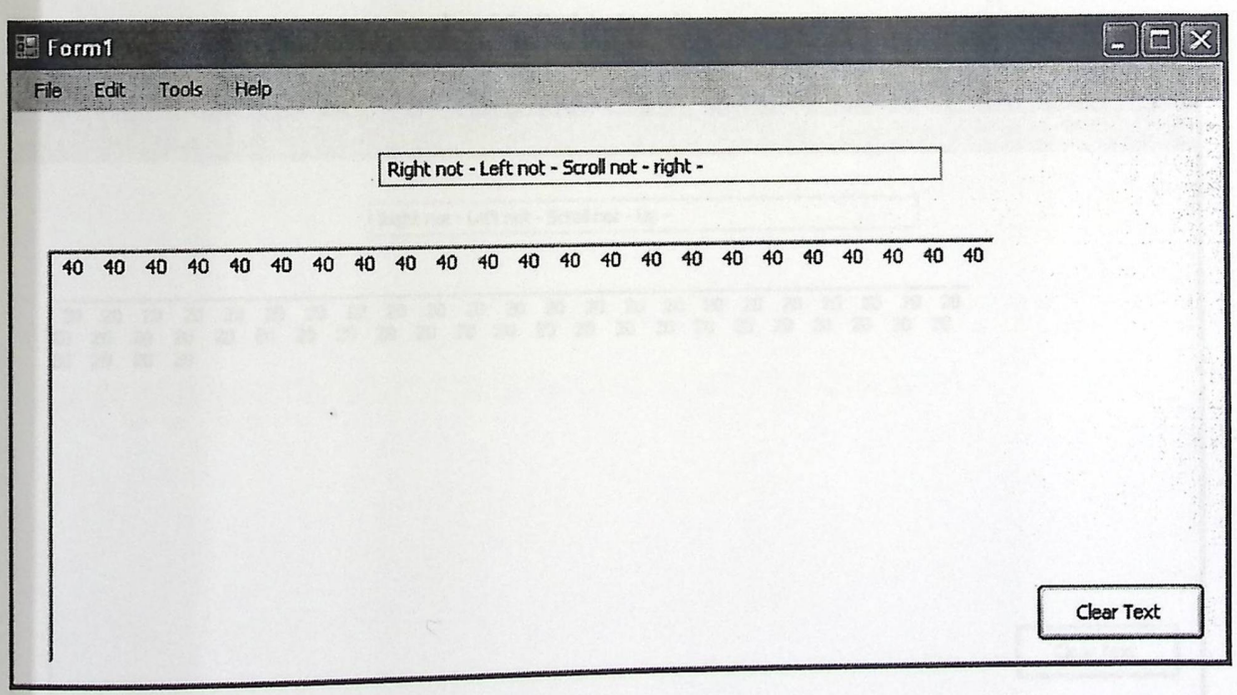


Figure 6-14: Mouse cursor moving right testing

To move mouse cursor left, one should tilt his hand left.

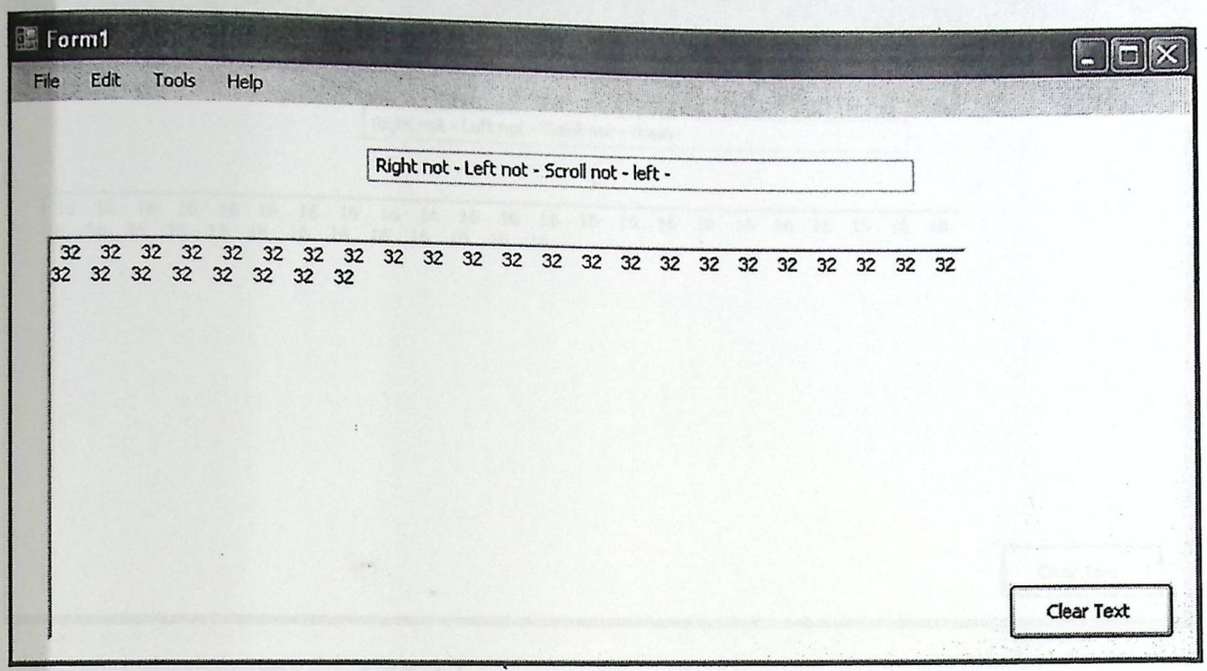


Figure 6-15: Mouse cursor moving left testing

To move mouse cursor up, one should tilt his hand up.

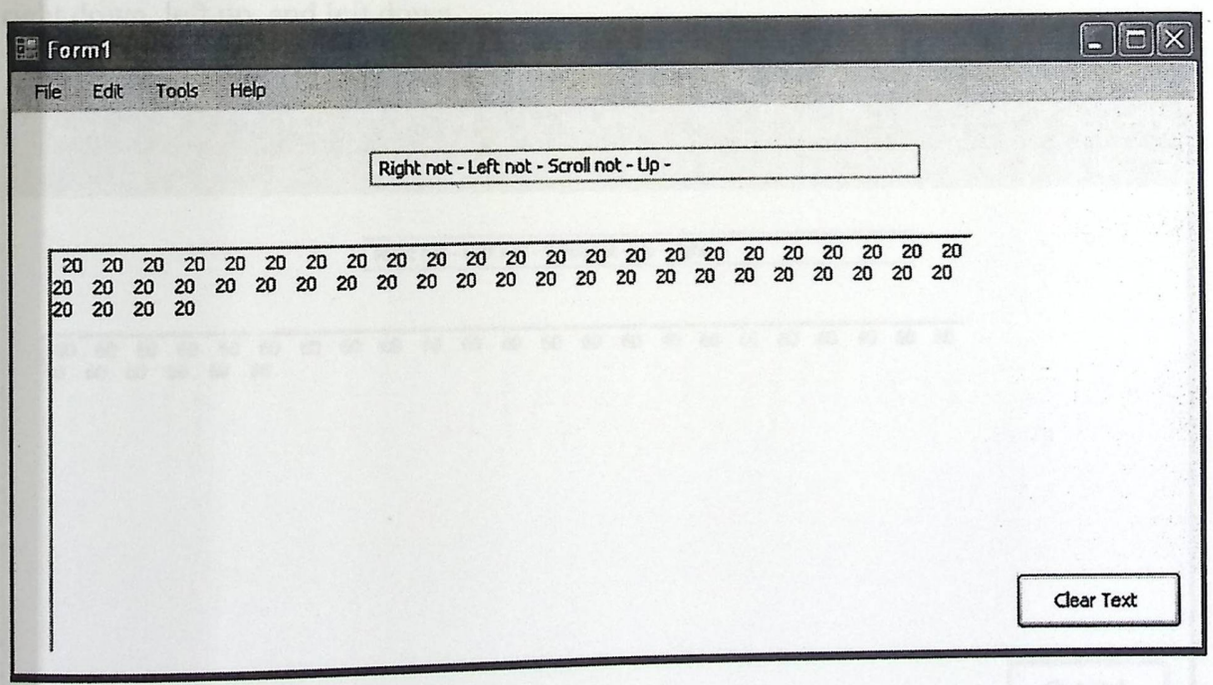


Figure 6-16: Mouse cursor moving up testing

To move mouse cursor down, one should tilt his hand down.

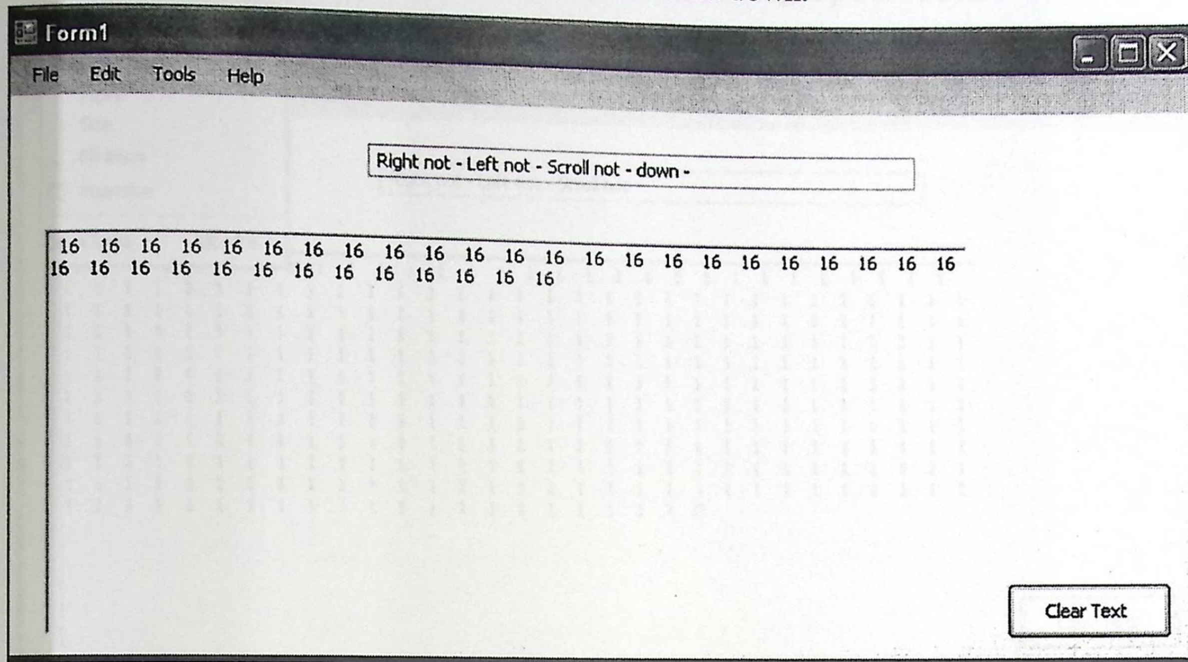


Figure 6-17: Mouse cursor moving down testing

By combining x with y movement one can move mouse cursor diagonal; right up, right down, left up, and left down.

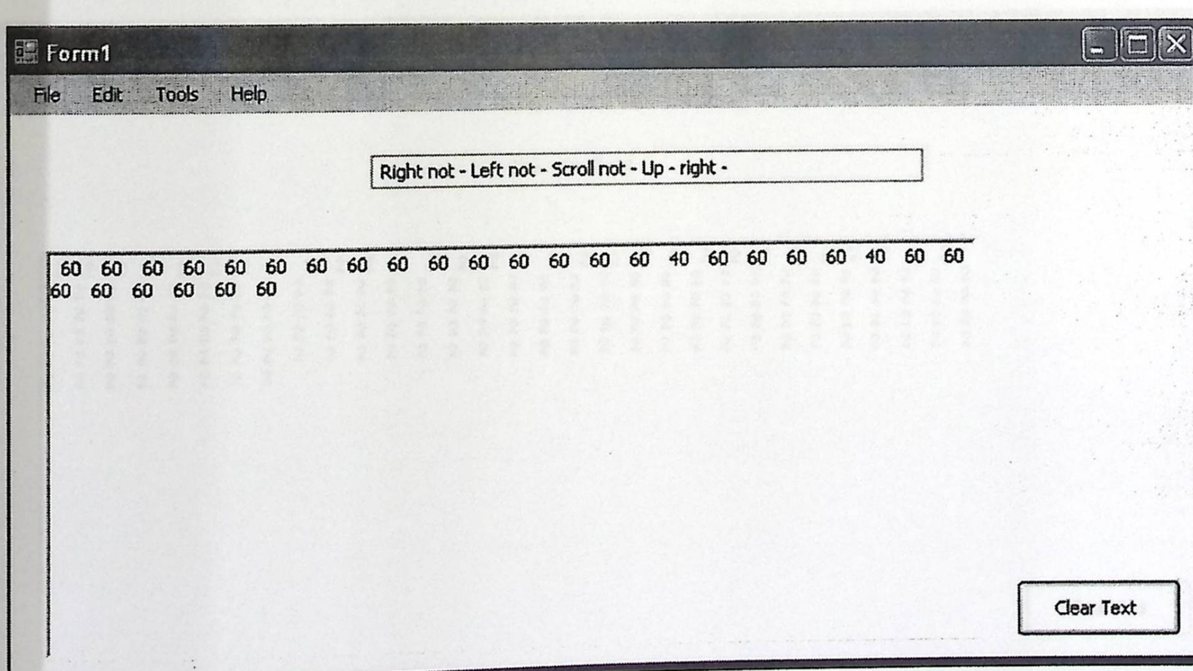


Figure 6-18: Mouse cursor moving up – right testing

To perform right click operation, user should click on "Right push button".

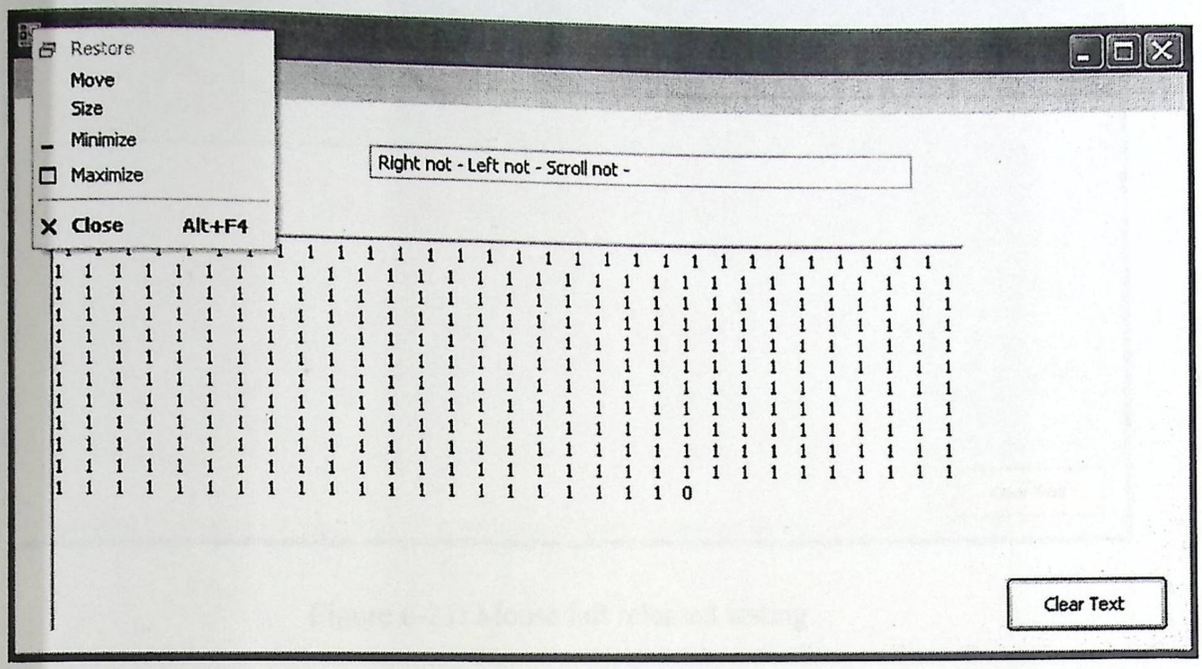


Figure 6-19: Mouse right click testing

To perform left click operation, user should click on "Left push button". This can be demonstrated by the following two figures (press and release).

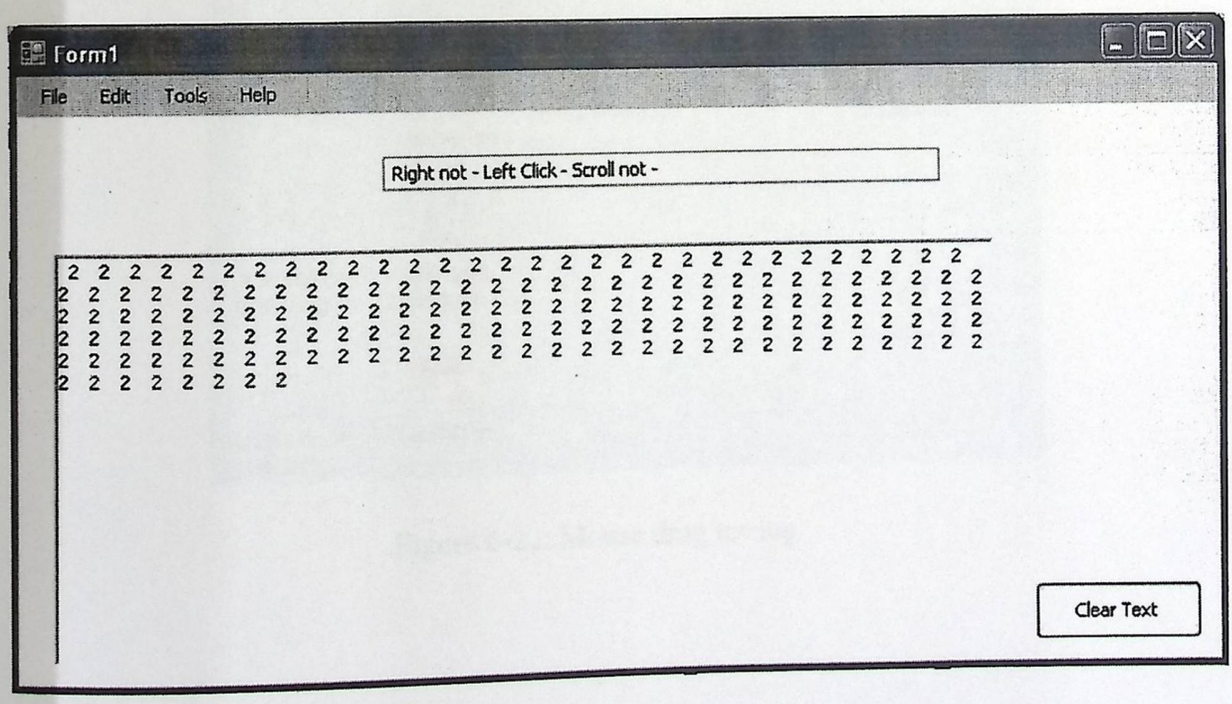


Figure 6-20: Mouse left pressed testing

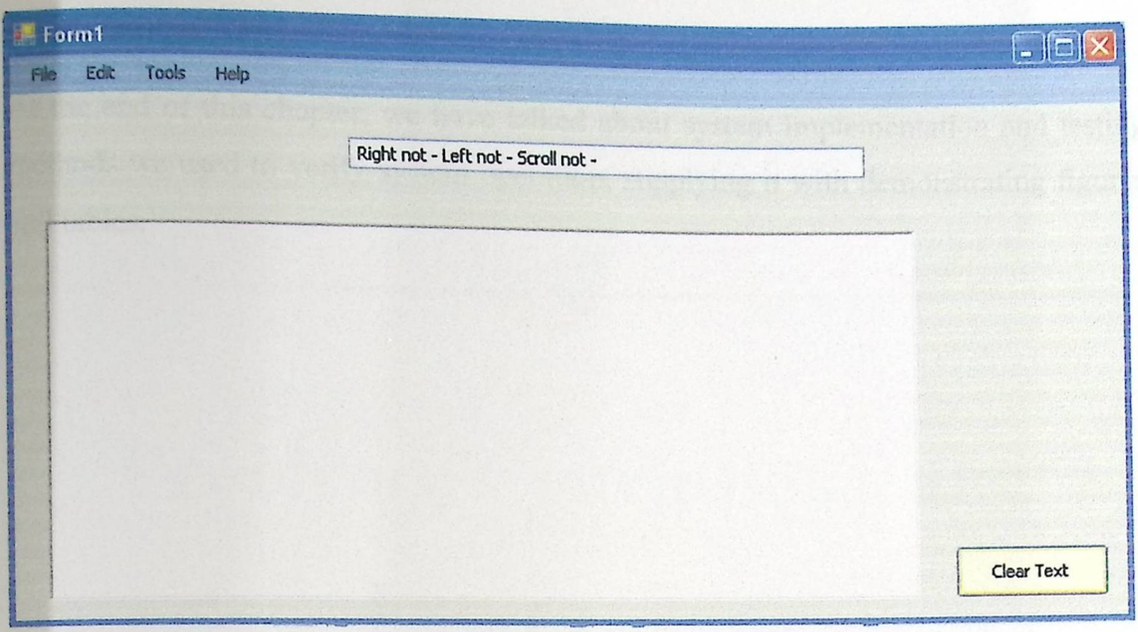


Figure 6-21: Mouse left released testing

To drag an object user should press on left push button and still for a short period then move object by tilting his hand. When finishing he can release left push button.

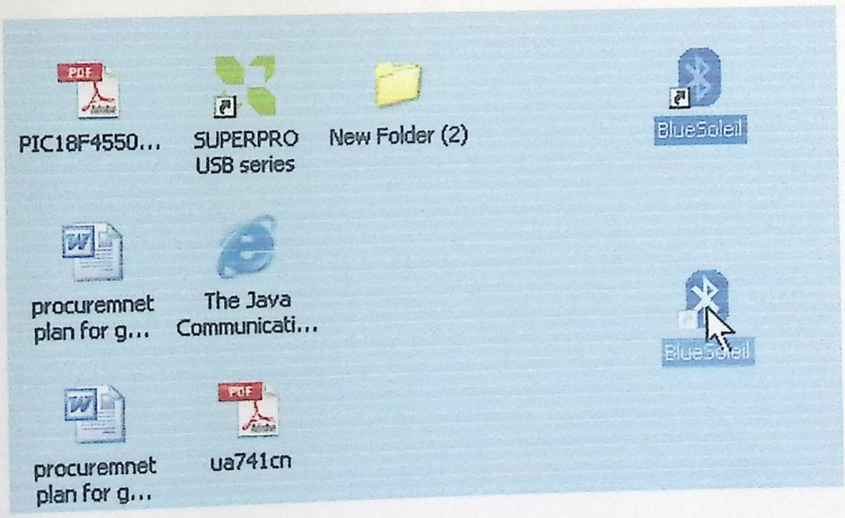


Figure 6-22: Mouse drag testing

6.4 Summary

At the end of this chapter, we have talked about system implementation and testing methods we used to verify system operation, supplying it with demonstrating figures and tables.

CHAPTER SEVEN 7

Problems, Conclusions, and Future Work

7.1 Overview

This chapter will address the learning outcomes of this project. It will also list the problems faced us in accomplishing the system, in addition, future work will be proposed.

7.2 Conclusions

In this project we have learned different approaches and experiences in developing projects and in the way of thinking to solve problems. Many conclusions and outcomes can be stated here, the most important ones are described:

- By the end of this project, we have finished designing and implementing an electronic glove that can be used to control mouse cursor.

- We learned how to apply theoretical information on practical applications that human can use.

- We learned how to use and how to program PIC, which is very useful in many applications.

- We used a new technology –Bluetooth technology, and it was a good chance for us to learn this new technology and apply it in our project.

- Team working is very useful; working is balanced on team members and so time and efforts will be saved.

- We have concluded that group thinking is the best way in solving problems that faces projects, in which many points of views can be considered.

7.3 Problems

Problems are natural things, especially when one develops his first project, but solving those problems is a success. Here are the most important problems that we faced during system development.

- Project components weren't available at the start of semester, so we had to wait until they were available so some time was wasted.

- Damaging in some devices because of wrong connections, our PIC and accelerometer were damaged so we were forced to get new ones.

- We faced a problem in dealing with the accelerometer outputs; firstly, we decided to use the analog outputs, but when connecting to PIC the signal drops, we spent many days to solve this problem but unfortunately we didn't succeed. After many days of searching we decided to take digital outputs instead of analog. We found that this method is easier and consumes no additional components, which is important because circuit will be placed on hand and it must be small.

- When testing system at the first time, data was sent at a very high rate, this caused the response of PC to be late; we solved this by adding a delay before sending each packet. It succeeded and the PC program worked well.

7.4 Future Work

Many ideas and applications can be developed upon our project. Here are some suggested ideas:

- Using the electronic glove to control a specific machine, by which user can control his machine using his hand and fingers movements.

- Design a glove that can be using for graphics simulations; such like animation.

- Integrate system to an electronic game.

- Upgrade system to use it in virtual reality applications.

REFERENCES

Books:

[1] IBM Corp, "*An Introduction to Wireless Technology*", October 1995

Websites:

[2] Andrew Sawchuk, Joseph Tanen, "*Nontraditional Cursor Control*", Circuit Cellar, 2006
URL: http://www.circuitcellar.com/library/print/0606/Sawchuk191/index.htm

[3] Aseem Kohli, "*Accelerometer Mouse*", Cornell University, Spring 2005
URL:http://instruct1.cit.cornell.edu/Courses/ee476/FinalProjects/s2005/mouse%20webpage%20KM249_AK288/INDEX.HTM

[4] Julia Layton and Curt Franklin, "*How Bluetooth Works*",
URL: https://www.bluetooth.org/apps/content

[5] A beginner's guide to accelerometers,
URL: http://electronics.howstuffworks.com

APPENDICES

Appendix A: Project Units' Data Sheets

Appendix B: Source code of PIC program

Appendix C: Source code of PC program

Appendix A: Project Units' Data Sheets

PIC 18F4550 Data Sheet
(Selected Papers)

PIC18F2455/2550/4455/4550
Data Sheet

28/40/44-Pin High-Performance, Enhanced Flash USB Microcontrollers with nanoWatt Technology

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Trademarks

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

Microchip received ISO/TS-16949:2002 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona and Mountain View, California in October 2003. The Company's quality system processes and procedures are for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

MICROCHIP PIC18F2455/2550/4455/4550

28/40/44-Pin High-Performance, Enhanced Flash USB Microcontrollers with nanoWatt Technology

Universal Serial Bus Features:

- USB V2.0 Compliant
- Low Speed (1.5 Mb/s) and Full Speed (12 Mb/s)
- Supports Control, Interrupt, Isochronous and Bulk Transfers
- Supports up to 32 endpoints (16 bidirectional)
- 1-Kbyte dual access RAM for USB
- On-chip USB transceiver with on-chip voltage regulator
- Interface for off-chip USB transceiver
- Streaming Parallel Port (SPP) for USB streaming transfers (40/44-pin devices only)

Power-Managed Modes:

- Run: CPU on, peripherals on
- Idle: CPU off, peripherals on
- Sleep: CPU off, peripherals off
- Idle mode currents down to 5.8 µA typical
- Sleep mode currents down to 0.1 µA typical
- Timer1 oscillator: 1.1 µA typical, 32 kHz, 2V
- Watchdog Timer: 2.1 µA typical
- Two-Speed Oscillator Start-up

Flexible Oscillator Structure:

- Four Crystal modes including High Precision PLL for USB
- Two External Clock modes, up to 48 MHz
- Internal oscillator block:
  - 8 user-selectable frequencies, from 31 kHz to 8 MHz
  - User-tunable to compensate for frequency drift
- Secondary oscillator using Timer1 @ 32 kHz
- Dual oscillator options allow microcontroller and USB module to run at different clock speeds
- Fail-Safe Clock Monitor
  - Allows for safe shutdown if any clock stops

Peripheral Highlights:

- High-current sink/source 25 mA/25 mA
- Three external interrupts
- Four Timer modules (Timer0 to Timer3)
- Up to 2 Capture/Compare/PWM (CCP) modules:
  - Capture is 16-bit, max. resolution 6.25 ns (TCY/16)
  - Compare is 16-bit, max. resolution 100 ns (TCY)
  - PWM output: PWM resolution is 1 to 10-bit
- Enhanced Capture/Compare/PWM (ECCP) module:
  - Multiple output modes
  - Selectable polarity
  - Programmable dead time
  - Auto-Shutdown and Auto-Restart
- Enhanced USART module:
  - LIN bus support
- Master Synchronous Serial Port (MSSP) module supporting 3-wire SPI™ (all 4 modes) and I^2C™ Master and Slave modes
- 10-bit, up to 13-channels Analog-to-Digital Converter module (A/D) with programmable acquisition time
- Dual analog comparators with input multiplexing

Special Microcontroller Features:

- C compiler optimized architecture with optional extended instruction set
- 100,000 erase/write cycle Enhanced Flash program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory typical
- Flash/Data EEPROM Retention: > 40 years
- Self-programmable under software control
- Priority levels for interrupts
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 41 ms to 131s
- Programmable Code Protection
- Single-Supply 5V In-Circuit Serial Programming™ (ICSP™) via two pins
- In-Circuit Debug (ICD) via two pins
- Optional dedicated ICD/ICSP port (44-pin devices only)
- Wide operating voltage range (2.0V to 5.5V)

Device	Program Memory		Data Memory		I/O	10-bit A/D (ch)	CCP/ECCP (PWM)	SPP	MSSP		EAUSART	Comparators	Timers 8/16-bit
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)					SPI™	Master I^2C™			
PIC18F2455	24K	12288	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F2550	32K	16384	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F4455	24K	12288	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3
PIC18F4550	32K	16384	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3

MICROCHIP PIC18F2455/2550/4455/4550

28/40/44-Pin High-Performance, Enhanced Flash USB Microcontrollers with nanoWatt Technology

Universal Serial Bus Features:

- USB V2.0 Compliant
- Low Speed (1.5 Mb/s) and Full Speed (12 Mb/s)
- Supports Control, Interrupt, Isochronous and Bulk Transfers
- Supports up to 32 endpoints (16 bidirectional)
- 1-Kbyte dual access RAM for USB
- On-chip USB transceiver with on-chip voltage regulator
- Interface for off-chip USB transceiver
- Streaming Parallel Port (SPP) for USB streaming transfers (40/44-pin devices only)

Power-Managed Modes:

- Run: CPU on, peripherals on
- Idle: CPU off, peripherals on
- Sleep: CPU off, peripherals off
- Idle mode currents down to 5.8 µA typical
- Sleep mode currents down to 0.1 µA typical
- Timer1 oscillator: 1.1 µA typical, 32 kHz, 2V
- Watchdog Timer: 2.1 µA typical
- Two-Speed Oscillator Start-up

Flexible Oscillator Structure:

- Four Crystal modes including High Precision PLL for USB
- Two External Clock modes, up to 48 MHz
- Internal oscillator block:
  - 8 user-selectable frequencies, from 31 kHz to 8 MHz
  - User-tunable to compensate for frequency drift
- Secondary oscillator using Timer1 @ 32 kHz
- Dual oscillator options allow microcontroller and USB module to run at different clock speeds
- Fail-Safe Clock Monitor
  - Allows for safe shutdown if any clock stops

Peripheral Highlights:

- High-current sink/source 25 mA/25 mA
- Three external interrupts
- Four Timer modules (Timer0 to Timer3)
- Up to 2 Capture/Compare/PWM (CCP) modules:
  - Capture is 16-bit, max. resolution 6.25 ns (TCY/16)
  - Compare is 16-bit, max. resolution 100 ns (TCY)
  - PWM output: PWM resolution is 1 to 10-bit
- Enhanced Capture/Compare/PWM (ECCP) module:
  - Multiple output modes
  - Selectable polarity
  - Programmable dead time
  - Auto-Shutdown and Auto-Restart
- Enhanced USART module:
  - LIN bus support
- Master Synchronous Serial Port (MSSP) module supporting 3-wire SPI™ (all 4 modes) and I^2C™ Master and Slave modes
- 10-bit, up to 13-channels Analog-to-Digital Converter module (A/D) with programmable acquisition time
- Dual analog comparators with input multiplexing

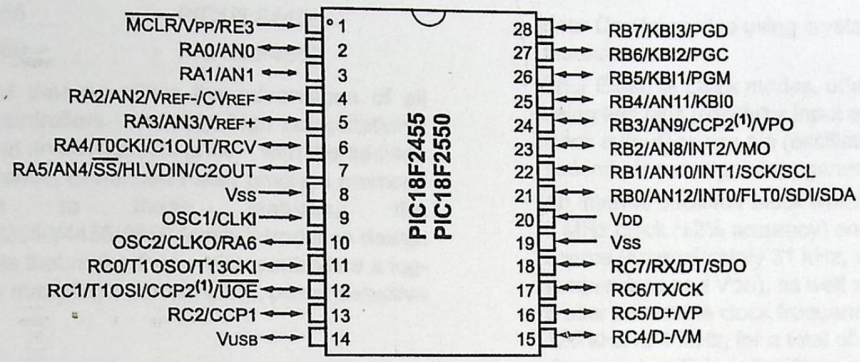
Special Microcontroller Features:

- C compiler optimized architecture with optional extended instruction set
- 100,000 erase/write cycle Enhanced Flash program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory typical
- Flash/Data EEPROM Retention: > 40 years
- Self-programmable under software control
- Priority levels for interrupts
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 41 ms to 131s
- Programmable Code Protection
- Single-Supply 5V In-Circuit Serial Programming™ (ICSP™) via two pins
- In-Circuit Debug (ICD) via two pins
- Optional dedicated ICD/ICSP port (44-pin devices only)
- Wide operating voltage range (2.0V to 5.5V)

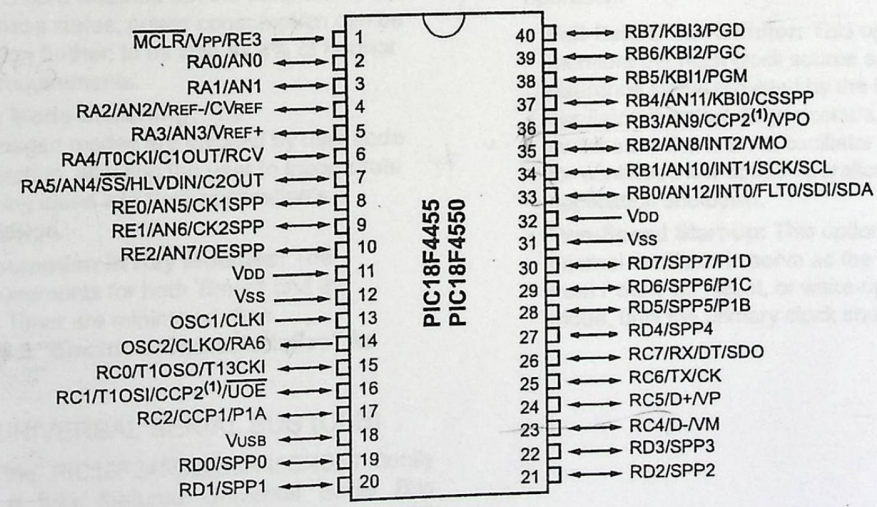
Device	Program Memory		Data Memory		I/O	10-bit A/D (ch)	CCP/ECCP (PWM)	SPP	MSSP		EAUSART	Comparators	Timers 8/16-bit
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)					SPI™	Master I^2C™			
PIC18F2455	24K	12288	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F2550	32K	16384	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F4455	24K	12288	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3
PIC18F4550	32K	16384	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3

Pin Diagrams

28-Pin PDIP, SOIC



40-Pin PDIP



Note 1: RB3 is the alternate pin for CCP2 multiplexing.

1.0 DEVICE OVERVIEW

This document contains device specific information for the following devices:

- PIC18F2455
- PIC18F2550
- PIC18F4455
- PIC18F4550
- PIC18LF2455
- PIC18LF2550
- PIC18LF4455
- PIC18LF4550

This family of devices offers the advantages of all PIC18 microcontrollers – namely, high computational performance at an economical price – with the addition of high endurance, Enhanced Flash program memory. In addition to these features, the PIC18F2455/2550/4455/4550 family introduces design enhancements that make these microcontrollers a logical choice for many high-performance, power sensitive applications.

1.1 New Core Features

1.1.1 nanoWatt TECHNOLOGY

All of the devices in the PIC18F2455/2550/4455/4550 family incorporate a range of features that can significantly reduce power consumption during operation. Key items include:

- **Alternate Run Modes:** By clocking the controller from the Timer1 source or the internal oscillator block, power consumption during code execution can be reduced by as much as 90%.
- **Multiple Idle Modes:** The controller can also run with its CPU core disabled but the peripherals still active. In these states, power consumption can be reduced even further, to as little as 4% of normal operation requirements.
- **On-the-fly Mode Switching:** The power-managed modes are invoked by user code during operation, allowing the user to incorporate power-saving ideas into their application's software design.
- **Low Consumption in Key Modules:** The power requirements for both Timer1 and the Watchdog Timer are minimized. See **Section 28.0 "Electrical Characteristics"** for values.

1.1.2 UNIVERSAL SERIAL BUS (USB)

Devices in the PIC18F2455/2550/4455/4550 family incorporate a fully featured Universal Serial Bus communications module that is compliant with the USB Specification Revision 2.0. The module supports both low-speed and full speed communication for all supported data transfer types. It also incorporates its own on-chip transceiver and 3.3V regulator and supports the use of external transceivers and voltage regulators.

1.1.3 MULTIPLE OSCILLATOR OPTIONS AND FEATURES

All of the devices in the PIC18F2455/2550/4455/4550 family offer twelve different oscillator options, allowing users a wide range of choices in developing application hardware. These include:

- Four Crystal modes using crystals or ceramic resonators.
- Four External Clock modes, offering the option of using two pins (oscillator input and a divide-by-4 clock output) or one pin (oscillator input, with the second pin reassigned as general I/O).
- An internal oscillator block which provides an 8 MHz clock (±2% accuracy) and an INTRC source (approximately 31 kHz, stable over temperature and V_{DD}), as well as a range of 6 user selectable clock frequencies, between 125 kHz to 4 MHz, for a total of 8 clock frequencies. This option frees an oscillator pin for use as an additional general purpose I/O.
- A Phase Lock Loop (PLL) frequency multiplier, available to both the high-speed crystal and external oscillator modes, which allows a wide range of clock speeds from 4 MHz to 48 MHz.
- Asynchronous dual clock operation, allowing the USB module to run from a high-frequency oscillator while the rest of the microcontroller is clocked from an internal low-power oscillator.

Besides its availability as a clock source, the internal oscillator block provides a stable reference source that gives the family additional features for robust operation:

- **Fail-Safe Clock Monitor:** This option constantly monitors the main clock source against a reference signal provided by the internal oscillator. If a clock failure occurs, the controller is switched to the internal oscillator block, allowing for continued low-speed operation or a safe application shutdown.
- **Two-Speed Start-up:** This option allows the internal oscillator to serve as the clock source from Power-on Reset, or wake-up from Sleep mode, until the primary clock source is available.

1.2 Other Special Features

- **Memory Endurance:** The Enhanced Flash cells for both program memory and data EEPROM are rated to last for many thousands of erase/write cycles – up to 100,000 for program memory and 1,000,000 for EEPROM. Data retention without refresh is conservatively estimated to be greater than 40 years.
- **Self-Programmability:** These devices can write to their own program memory spaces under internal software control. By using a bootloader routine, located in the protected Boot Block at the top of program memory, it becomes possible to create an application that can update itself in the field.
- **Extended Instruction Set:** The PIC18F2455/2550/4455/4550 family introduces an optional extension to the PIC18 instruction set, which adds 8 new instructions and an Indexed Literal Offset Addressing mode. This extension, enabled as a device configuration option, has been specifically designed to optimize re-entrant application code originally developed in high-level languages such as C.
- **Enhanced CCP Module:** In PWM mode, this module provides 1, 2 or 4 modulated outputs for controlling half-bridge and full-bridge drivers. Other features include auto-shutdown for disabling PWM outputs on interrupt or other select conditions and auto-restart to reactivate outputs once the condition has cleared.
- **Enhanced Addressable USART:** This serial communication module is capable of standard RS-232 operation and provides support for the LIN bus protocol. Other enhancements include Automatic Baud Rate Detection and a 16-bit Baud Rate Generator for improved resolution. When the microcontroller is using the internal oscillator block, the EUSART provides stable operation for applications that talk to the outside world without using an external crystal (or its accompanying power requirement).
- **10-bit A/D Converter:** This module incorporates programmable acquisition time, allowing for a channel to be selected and a conversion to be initiated, without waiting for a sampling period and thus, reducing code overhead.
- **Dedicated ICD/ICSP Port:** These devices introduce the use of debugger and programming pins that are not multiplexed with other micro-controller features. Offered as an option in select packages, this feature allows users to develop I/O intensive applications while retaining the ability to program and debug in the circuit.

1.3 Details on Individual Family Members

Devices in the PIC18F2455/2550/4455/4550 family are available in 28-pin and 40/44-pin packages. Block diagrams for the two groups are shown in Figure 1-1 and Figure 1-2.

The devices are differentiated from each other in six ways:

1. Flash program memory (24 Kbytes for PIC18FX455 devices, 32 Kbytes for PIC18FX550).
2. A/D channels (10 for 28-pin devices, 13 for 40/44-pin devices).
3. I/O ports (3 bidirectional ports and 1 input only port on 28-pin devices, 5 bidirectional ports on 40/44-pin devices).
4. CCP and Enhanced CCP implementation (28-pin devices have 2 standard CCP modules, 40/44-pin devices have one standard CCP module and one ECCP module).
5. Streaming Parallel Port (present only on 40/44-pin devices).

All other features for devices in this family are identical. These are summarized in Table 1-1.

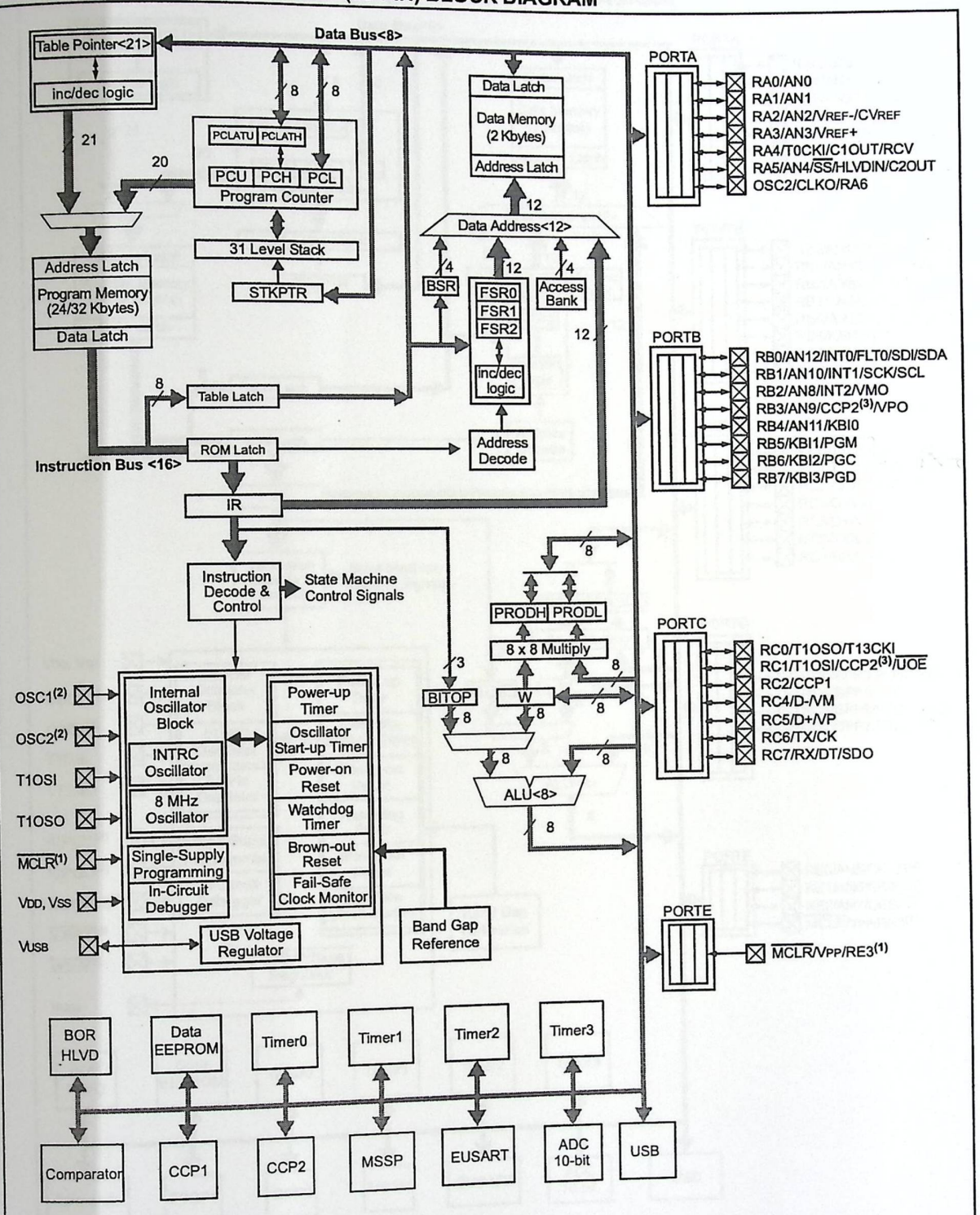
The pinouts for all devices are listed in Table 1-2 and Table 1-3.

Like all Microchip PIC18 devices, members of the PIC18F2455/2550/4455/4550 family are available as both standard and low-voltage devices. Standard devices with Enhanced Flash memory, designated with an "F" in the part number (such as PIC18**F**2550), accommodate an operating V_{DD} range of 4.2V to 5.5V. Low-voltage parts, designated by "LF" (such as PIC18**LF**2550), function over an extended V_{DD} range of 2.0V to 5.5V.

TABLE 1-1: DEVICE FEATURES

Features	PIC18F2455	PIC18F2550	PIC18F4455	PIC18F4550
Operating Frequency	DC – 48 MHz	DC – 48 MHz	DC – 48 MHz	DC – 48 MHz
Program Memory (Bytes)	24576	32768	24576	32768
Program Memory (Instructions)	12288	16384	12288	16384
Data Memory (Bytes)	2048	2048	2048	2048
Data EEPROM Memory (Bytes)	256	256	256	256
Interrupt Sources	19	19	20	20
I/O Ports	Ports A, B, C, (E)	Ports A, B, C, (E)	Ports A, B, C, D, E	Ports A, B, C, D, E
Timers	4	4	4	4
Capture/Compare/PWM Modules	2	2	1	1
Enhanced Capture/ Compare/PWM Modules	0	0	1	1
Serial Communications	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART
Universal Serial Bus (USB) Module	1	1	1	1
Streaming Parallel Port (SPP)	No	No	Yes	Yes
10-bit Analog-to-Digital Module	10 Input Channels	10 Input Channels	13 Input Channels	13 Input Channels
Comparators	2	2	2	2
Resets (and Delays)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), $\overline{MCLR}$ (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), $\overline{MCLR}$ (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), $\overline{MCLR}$ (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), $\overline{MCLR}$ (optional), WDT
Programmable Low-Voltage Detect	Yes	Yes	Yes	Yes
Programmable Brown-out Reset	Yes	Yes	Yes	Yes
Instruction Set	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled
Packages	28-pin PDIP 28-pin SOIC	28-pin PDIP 28-pin SOIC	40-pin PDIP 44-pin QFN 44-pin TQFP	40-pin PDIP 44-pin QFN 44-pin TQFP

FIGURE 1-1: PIC18F2455/2550 (28-PIN) BLOCK DIAGRAM

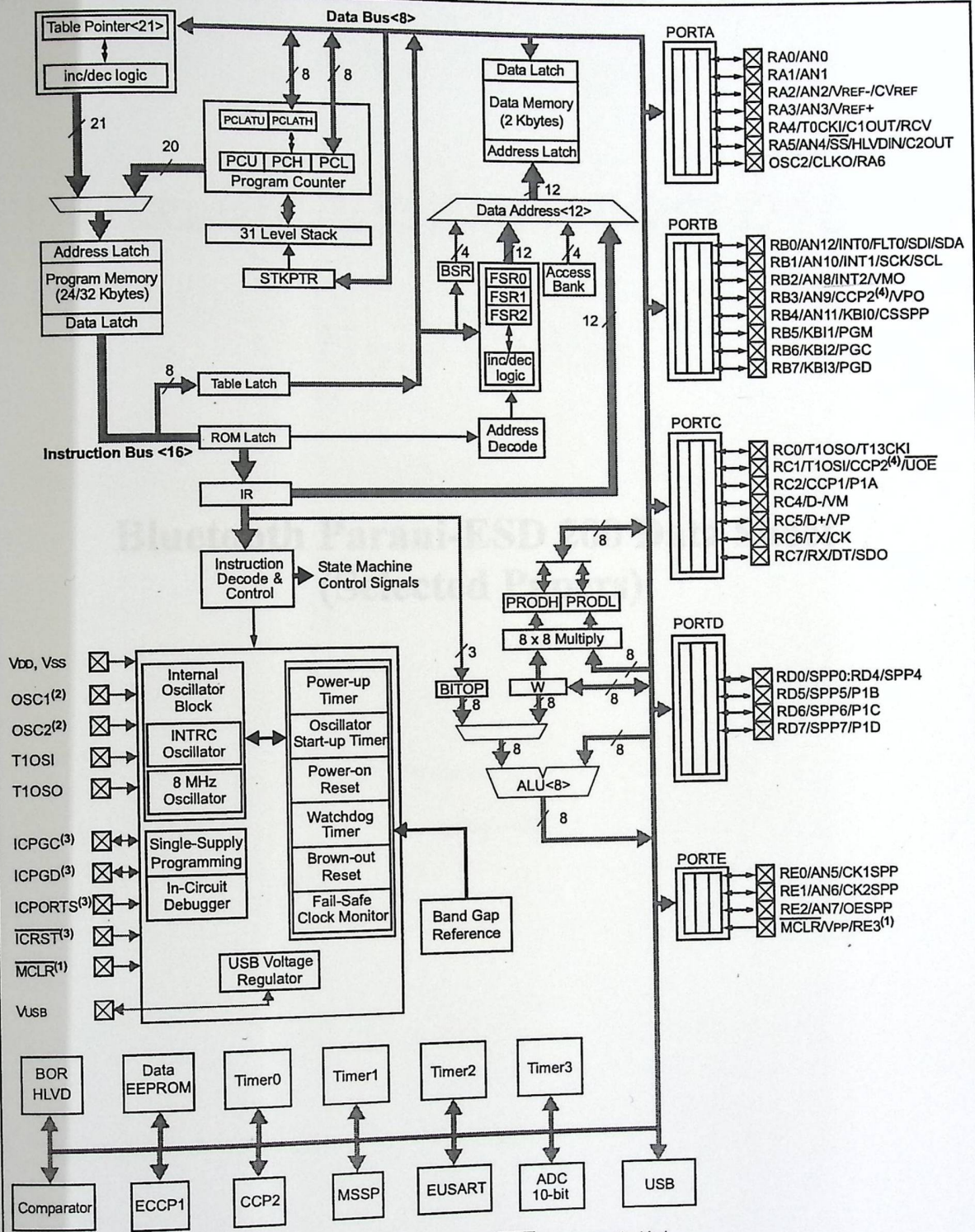


Note 1: RE3 is multiplexed with $\overline{MCLR}$ and is only available when the $\overline{MCLR}$ Resets are disabled.

2: OSC1/CLKI and OSC2/CLKO are only available in select oscillator modes and when these pins are not being used as digital I/O. Refer to **Section 2.0 "Oscillator Configurations"** for additional information.

3: RB3 is the alternate pin for CCP2 multiplexing.

FIGURE 1-2: PIC18F4455/4550 (40/44-PIN) BLOCK DIAGRAM



Note 1: RE3 is multiplexed with $\overline{MCLR}$ and is only available when the $\overline{MCLR}$ Resets are disabled.
2: OSC1/CLKI and OSC2/CLKO are only available in select oscillator modes and when these pins are not being used as digital I/O. Refer to **Section 2.0 "Oscillator Configurations"** for additional information.
3: These pins are only available on 44-pin TQFP under certain conditions. Refer to **Section 25.9 "Special ICPORT Features (Designated Packages Only)"** for additional information.
4: RB3 is the alternate pin for CCP2 multiplexing.

Bluetooth Parani-ESD 200 Data Sheet
(Selected Papers)

Parani-ESD100/110/200/210

User Guide

Version 1.1.4

2008-01-07

User Guide for the Parani-ESD100/110/200/210

Version 1.1.4
Firmware version 1.1.X
Last revised on January 7, 2008
Printed in Korea

Trademark

Parani™ is a trademark of Sena Technologies, Inc.
Windows® is a registered trademark of Microsoft Corporation.
Ethernet® is a registered trademark of XEROX Corporation.

Notice to Users

When a system failure may cause serious consequences, protecting life and property against such consequences with a backup system or safety device is essential. The user agrees that protection against consequences resulting from system failure is the user's responsibility.
This device is not approved for life-support or medical systems.
Changes or modifications to this device not explicitly approved by Sena Technologies will void the user's authority to operate this device.

Precautions and Safety

Electricity

Use only the supplied AC adapter. Use of unauthorized power adapter is not recommended. Electrical shock may result.
Do not kink or crease the power cable or place heavy objects on the power cable. Fire can result from damaged power cables.
Do not handle power plug and adapter with wet hands. Electrical shock may result.
Immediately power off the product and unplug the AC adapter if smoke or odors emit from the product and adapter. Fire can result from improper use.
Immediately power off the product and unplug the AC adapter if water or other liquids are present. Fire can result from improper use.

Product

Parani-ESD meets the RS-232 standards. Do not wire with non-standard products. Damage to your products may result from improper use.
Do not drop or subject the device to impact. Damage to your products may result from improper use.
Keep away from harsh environments including humid, dusty, and smoky areas. Damage to your products may result from improper use.
Do not use excessive force on the buttons or attempt to disassemble the device. Damage to your products may result from improper use.
Do not place heavy objects on the product. Damage to your products may result from improper use.

Technical Support

Sena Technologies, Inc.
210 Yangjae-dong, Seocho-gu
Seoul 137-130, Korea
Tel: (+82-2) 573-5422
Fax: (+82-2) 573-7710
E-Mail: support@sena.com
Website: http://www.sena.com

Revision History

Revision	Date	Name	Description
V1.1.2	2007-05-26	Yh Moon	Initial Revision History
V1.1.3	2007-06-25	Yh Moon	Update Approval, Command Validity, 3.4 data bit, B3.6, remove at+dfu, S22, B3.4
V1.1.4	2007-01-07	Yh Moon	Firmware v1.1.3 update reflected

1. Introduction

1.1. Overview

Parani-ESD is a module device for wireless serial communication using Bluetooth technology that is international a standard for short range wireless communications. Parani-ESD can communicate with other Bluetooth devices that support the Serial Port Profile.

Parani-ESD lineup has several models with different communication ranges from 30m (Parani-ESD200/210) up to 100m (Parani-ESD100/110) for use with various applications. The Parani-ESD delivers better quality of communication than a standard RS232 cables.

Parani-ESD has a compact design and can be placed conveniently into devices or equipment. Its detachable antenna optimizes the quality and distance for wireless communications.

Parani-ESD supports FHSS (Frequency Hopping Spread Spectrum), which is a technique, native to Bluetooth that allows the Parani-ESD minimize radio interference while decreasing the likelihood of over-air hijacking. Parani-ESD also supports authentication and Bluetooth data encryption.

Parani-ESD can be configured and controlled by typical AT commands. Users can easily configure Parani-ESD by using a terminal program such as HyperTerminal and can use Bluetooth wireless communication without modifying user's existing serial communication program. In addition to the basic AT commands, Parani-ESD provides some expanded AT commands for various functions. User friendly ParaniWizard and ParaniWIN are also provided for easy setup on Microsoft Windows.

1.2. Package Check List

1.2.1. Single/Bulk Unit Package

- Parani-ESD100/200
  - Parani-ESD100/200 module
  - on-board chip antenna

- Parani-ESD110/210
  - Parani-ESD110/210 module
  - Stub Antenna
  - Antenna extension cable

1.2.2. Starter Kit

- Jig board
- Serial data cable
- DC Power Adapter
- A hardcopy of Quick Start Guide
- CD-ROM including the Configuration S/W and User Guide

1.3. Product Specification

<table>
<tr><th></th><th>Parani-ESD100/110</th><th>Parani-ESD200/210</th></tr>
<tr><td rowspan="2">Serial Interface</td><td colspan="2">Serial speeds 1200bps to 230400bps
Flow Control: None, Hardware RTS/CTS</td></tr>
<tr><td>2.54mm Header 2X6</td><td>2.54mm Header 1X4X2</td></tr>
<tr><td rowspan="6">Bluetooth Interface</td><td colspan="2">Bluetooth v1.2 *</td></tr>
<tr><td colspan="2">Protocol: RFCOMM, L2CAP, SDP</td></tr>
<tr><td colspan="2">Profile: Serial Port Profile</td></tr>
<tr><td>Class 1</td><td>Class 2</td></tr>
<tr><td>Level: 18dBm</td><td>Level: Max. 4dBm</td></tr>
<tr><td>Parani-ESD100-Working distance:
Nominal 100m
Parani-ESD110-Working distance:
Default–Default Antenna 100m
Default–Dipole Antenna 150m
Dipole–Dipole Antenna 200m
Patch–Dipole Antenna 400m
Patch–Patch Antenna 1000m</td><td>Parani-ESD200-Working distance:
Nominal 30m
Parani-ESD210-Working distance:
Default–Default Antenna 30m
Default–Dipole Antenna 50m
Dipole–Dipole Antenna 80m
Patch–Dipole Antenna 150m
Patch–Patch Antenna 300m</td></tr>
<tr><td>Configuration</td><td colspan="2">ParaniWIN, ParaniWizard, Modem AT command set</td></tr>
<tr><td>Firmware Update</td><td colspan="2">ParaniUpdater</td></tr>
<tr><td>Power</td><td colspan="2">Supply voltage: 3.3V DC
Supply current:
- Parani-ESD100/110 : minimum 300mA
- Parani-ESD200/210 : minimum 150mA
Nominal power consumption
Parani-ESD100 : 43mA@9600bps, 61mA@115Kbps
Parani-ESD110 : 44mA@9600bps, 57mA@115Kbps
Parani-ESD200 : 33mA@9600bps, 40mA@115Kbps
Parani-ESD210 : 31mA@9600bps, 39mA@115Kbps</td></tr>
<tr><td>Environmental</td><td colspan="2">Operating temperature: -10 ~ 55 °C
Storage temperature: -20 ~ 70 °C
Humidity : 90% (Non-condensing)</td></tr>
<tr><td rowspan="3">Physical properties</td><td>Parani-ESD100 Dimension
27.5 mm L (1.08 in.)
30 mm W (1.18 in.)
14 mm H (0.55 in.)</td><td rowspan="2">Dimension
18 mm L (0.7 in.)
20 mm W (0.78 in.)
12 mm H (0.47 in.)</td></tr>
<tr><td>Parani-ESD110 Dimension
27.5 mm L (1.08 in.)
27.7 mm W (1.09 in.)
14 mm H (0.55 in.)</td></tr>
<tr><td>Weight
5 g</td><td>Weight
2 g</td></tr>
<tr><td>Approvals</td><td colspan="2">FCC(A), MIC, CE, SIG</td></tr>
<tr><td>Warranty</td><td colspan="2">3-year limited warranty</td></tr>
</table>

Note *:

Bluetooth v1.2 supports improved AFH function. AFH function is to mitigate the interference between WiFi and Bluetooth radios by automatically avoiding the active WiFi channel from Bluetooth link. However, AFH does not provide a complete solution making WiFi and Bluetooth work together in harmony. It is highly recommended for users to test their wireless system enough before deployment since the overall system performance is affected by various environmental factors such as distance between them.

3. Configuration

3.1. Operation Modes

In addition to the serial port configurations, the Parani-ESD requires also includes some settings for Bluetooth. For getting the most out of Parani-ESD, user should understand the following Bluetooth connection schemes.

A Bluetooth device can play a role as a master or slave. Master tries to connect itself to other Bluetooth devices, and slave is waiting to be connected from other Bluetooth devices. A Bluetooth connection is always made by a pair of master and slave devices. A slave can be in two modes, Inquiry Scan or Page Scan mode. Inquiry Scan mode is waiting for a packet of inquiry from other Bluetooth device and Page Scan mode is waiting for a packet of connection from other Bluetooth device. Every Bluetooth device has its unique address, called BD (Bluetooth Device) address, which is composed of 12 hexa-decimal numbers.

Parani-ESD has 4 operation modes as follows:

Table 3-1 The Parani-ESD Operation Modes

Mode	Description
Mode0	In this mode, there is no response when power on or software reset, and Parani-ESD is just waiting for AT command input. Neither master nor slave is assigned to Parani-ESD in mode0. User can change the configuration parameters of Parani-ESD in this mode. Parani-ESD must be in Mode0, when it is directly controlled by AT commands. The factory default is set to Mode0.
Mode1	Parani-ESD tries to connect the last connected Bluetooth device. Parani-ESD in Mode1 is to be a master and tries to connect the last connected Bluetooth device. Parani-ESD always stores the BD address of the Bluetooth device to which Parani-ESD has connected last. When Parani-ESD is initially used or after hardware reset, there is no BD address stored in Parani-ESD. In this case, Mode1 will not be able to work properly. The mode change to Mode1 can be made after Parani-ESD succeeds to connect to one other Bluetooth device. Once changed to Mode1, Parani-ESD will try to connect automatically the last connected Bluetooth device whenever the unit is powered on or software reset. Parani-ESD in Mode1 cannot be discovered or connected by other Bluetooth devices.
Mode2	Parani-ESD is waits for a connection from the last connected Bluetooth device. Parani-ESD in Mode2 is to be a slave and waiting for the connection only from the last connected Bluetooth device. Just like Mode1, if there is no BD address stored in Parani-ESD, the mode change from other operation modes to Mode2 is not work properly. Once changed to Mode2, Parani-ESD will wait for the connection from the last connected Bluetooth device whenever the unit is powered on or software reset. Parani-ESD in Mode2 cannot be discovered or connected to Bluetooth devices other than the last connected device.
Mode3	Parani-ESD is waiting for the connection from any other Bluetooth devices. In Mode 3 the Parani-ESD is discoverable and can be connected to by other Bluetooth devices.

3.2. Serial Ports

The applicable settings for serial ports are as follows.

Table 3-2 The Parani-ESD Serial Port Settings

Serial Port Settings	Values
Baud rate	1200, 2400, 4800, [9600], 19200, 38200, 57600, 115200, 230400
Data bite	[8]
Parity	[No parity], Even parity, Odd parity
Stop bit	[1], 2
Hardware Flow Control	[Use], No Use

The values in box are the factory default settings.

3.3 Data Bit

Parani-ESD supports only 8 data bit. In the case of 7 data bit and even/odd parity, use Parani-ESD 8 data bit and none parity. At this time, master and slave are Parani-SD, Parani-ESD or Parani-MSP series. But 7 data bit and none parity is not support.

3.4 Hardware Flow Control

Parani-ESD plugged into its host system transmits data from host to the other side Bluetooth device. This data is saved temporarily in the internal buffer of Parani-ESD and sent repeatedly until the transmission is completed packet by packet. When the radio transmission condition is not good enough to send data promptly, it can cause a transmission delay. If the host sends more data when the buffer is full, buffer overflow will make Parani-ESD malfunction consequently. In order to prevent this buffer overflow, Parani-ESD works as follows.

When using hardware flow control, Parani-ESD disables RTS so that it stops receiving any further data from the host when the buffer becomes full. RTS will be re-enabled again to begin receiving data from the host when the buffer has created more room for more data.

When hardware flow control is not being used, the Parani-ESD clears the buffer to secure room for the next data when the buffer becomes full. This can mean a loss of data may occur. As the transmission data becomes large, the possibility of data loss becomes greater.

For large data transmissions, the use of hardware flow control is highly recommended.

3.5 Software and Utility

This configuration software and utility for firmware update is included with the product, which also can be downloaded from http://www.sena.com

Table 3-3 Configuration Software

Software	Purpose	Operating System
ParaniWIN	Configuration	MS Windows 98SE or Higher
ParaniWizard	Pairing Configuration	MS Windows 98SE or Higher
ParaniUpdater	Firmware Update	MS Windows 98SE or Higher

A.1.2. Parani-ESD200/210

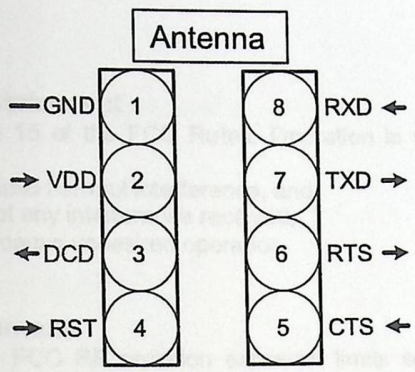


Figure A-2 Pin Assignment of Parani-ESD200/210

Table A-2 Pin Assignment of Parani-ESD200/210

Pin #	Signal	Direction	Description	Signal Level
1	GND	-	Power Ground	Ground
2	VDD	Input	DC Input (3.0~3.3V)	Power
3	DCD	Output	Bluetooth Connect Detect (Active Low)	TTL
4	RST	Input	Reset (Active Low)	TTL
5	CTS	Input	UART Clear to Send	TTL
6	RTS	Output	UART Ready to Send	TTL
7	TxD	Output	UART Data Output	TTL
8	RxD	Input	UART Data Input	TTL

A.1.3. DCD Signal (Status: Bluetooth Connect Detect)

Status of Bluetooth connection will be delivered to Host PC via DCD line. Connection Module → low signal

A.1.4. RST Signal

RST signal will be used for setting the Parani-ESD to factory defaults. RST should be on 0V status for at least 1 second for the reset to occur.

4. Approval Information

4.1. FCC

4.1.1. FCC Compliance Statement

This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions:

(1) This device may not cause harmful interference, and
(2) This device must accept any interference received,

Including interference that may cause undesired operation

4.1.2. RF Exposure Statement

The equipment complies with FCC RF radiation exposure limits set forth for an uncontrolled environment. This device and its antenna must not be co-located or operation in conjunction with any other antenna or transmitter.

4.1.3. Do not

Any changes or modifications to the equipment not expressly approved by the party responsible for compliance could void user's authority to operate the equipment.

4.2. MIC

4.3. CE

4.4. SIG

5. RF Information

5.1. Radio Frequency Range

2.402~2.480GHz

5.2. Number of Frequency Channel

79 channels

5.3. Transmission Method

FHSS(Frequency Hopping Spread Spectrum)

5.4. Modulation Method

GFSK(Gaussian-filtered Frequency Shift Keying)

5.5. Radio Output Power

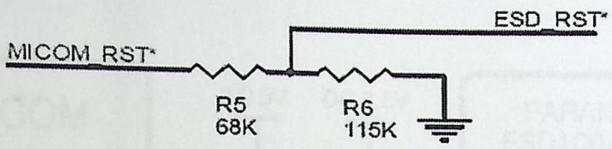
Products	Radio Output Power
Parani-ESD100	+18dBm
Parani-ESD110	+18dBm
Parani-ESD200	+4dBm
Parani-ESD210	+4dBm

5.6. Receiving Sensitivity

Products	Receiving Sensitivity
Parani-ESD100	-88dBm
Parani-ESD110	-88dBm
Parani-ESD200	-80dBm
Parani-ESD210	-80dBm

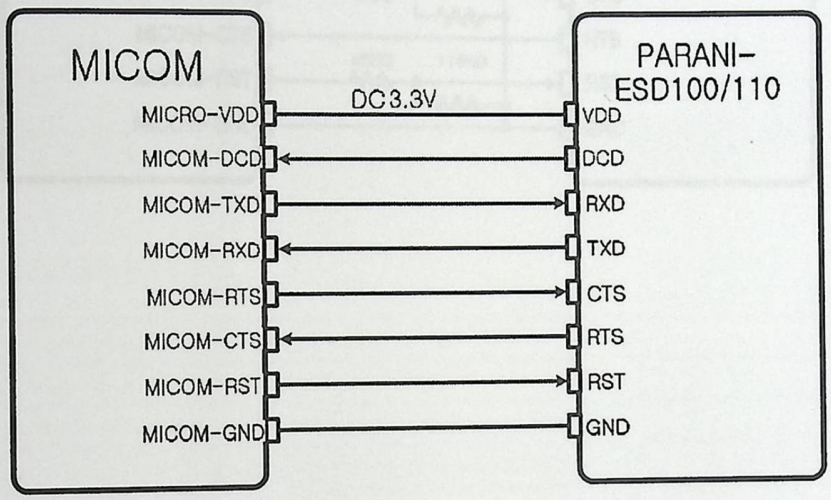
5.7. Power Supply

Products	Power Supply
Parani-ESD100	DC3.3V
Parani-ESD110	DC3.3V
Parani-ESD200	DC3.3V
Parani-ESD210	DC3.3V

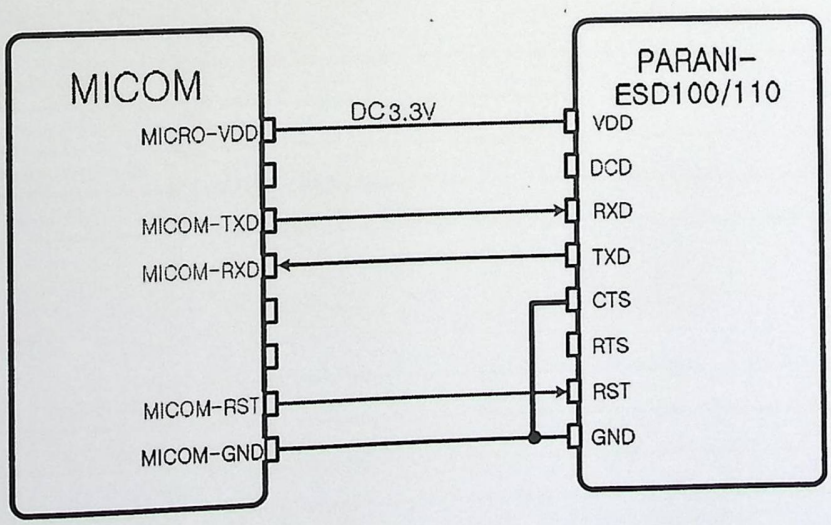


A.2.2. Parani-ESD200/210

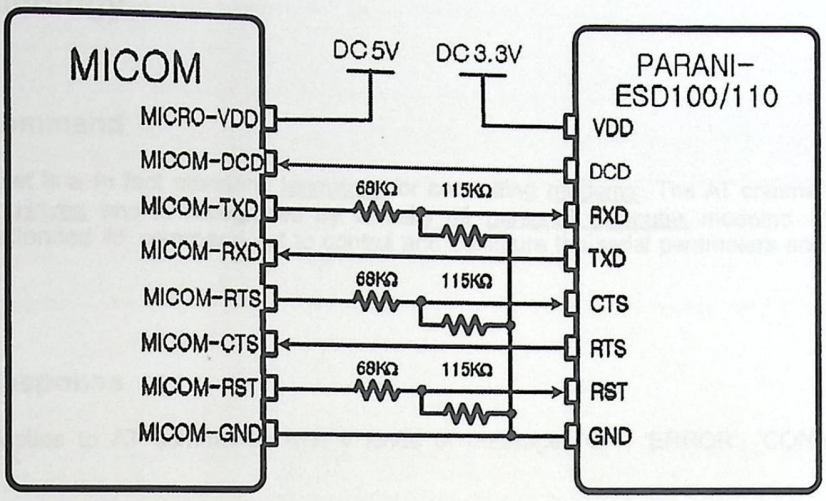
A.2.2.1. When TTL level of MICOM is 3.3V



A.2.2.2. When TTL level of MICOM is 3.3V and Hardware Flow Control is not used



A.2.2.3. When TTL level of MICOM is 5V



Appendix B: AT Commands

B.1. Terminology

B.1.1. AT Command

AT command set is a in fact standard language for controlling modems. The AT command set was developed by Hayes and is recognized by virtually all personal computer modems. Parani-ESD provides the extended AT command set to control and configure the serial parameters and Bluetooth connection.

B.1.2. AT Response

Parani-ESD replies to AT commands with 4 kinds of message, 'OK', 'ERROR', 'CONNECT' and 'DISCONNECT'.

B.1.3. Operation Mode

Mode	Description
Mode0	Waiting for AT commands
Mode1	Attempting to connect to the last connected Bluetooth device
Mode2	Waiting for a connection from the last connected Bluetooth device
Mode3	Waiting for the connection from another Bluetooth device

B.1.4. Operation Status

Status	Description
Standby	Waiting for AT commands
Pending	Executing tasks
Connect	Transmitting data

B.1.5. Security

Security	Description
Authentication	Pin Code (or Pass key)
Encryption	Data encryption

B.1.6. Symbols

The symbols are used for the description of command syntax as follows:

Symbols	Meaning	ASCII Code
↵	Carriage return	0x0D
↙	Line feed	0x0A
↯	Carriage return + Line feed	
00112233445566	Bluetooth device address	
N or m	One digit decimal number	
To	Timeout in seconds	

B.2. Command Category

Command Category		Index	AT Commands
RESET		1 2	ATZ AT&F
SERIAL PORT		3 4	AT AT+UARTCONFIG,b,p,s,h
BLUETOOTH	Information	5 6 7 8 9	AT+BTINFO? AT+BTINQ? AT+BTLAST? AT+BTVER? AT+BTRSSI,n
	Mode	10	AT+BTMODEn
	Status	11 12 13 14 15 16 17	+++ AT+SETESC,nn ATO AT+BTCANCEL AT+BTSCAN AT+BTSCAN,n,to AT+BTSCAN112233445566,to
	Connection	18 19 20 21 22	ATD ATD112233445566 ATA ATA112233445566 ATH
	Security	23 24 25 26 27	AT+BTKEY=$string AT+BTSD? AT+BTCSD AT+BTFP,n AT+BTSEC,a,e
	Miscellaneous	28 29	AT+BTNAME=$string AT+BTLPM,n
S-REGISTER		30 31 32	AT&V ATSnn? ATSnn=mm

B.3. Command Description

B.3.1. ATZ↵

Response	↯OK↯
Purpose	Software Reset
Description	This has the same effects as Powercycling the unit. This command disconnects any connected Bluetooth device, and stops ongoing tasks. After rebooting, the status will be decided by the preset operation mode. Some AT commands require the ATZ command be run so that the commands can take effect.
Reference	AT&F, AT+BTCSD, AT+UARTCONFIG

B.3.2. AT&F↵

Response	↯OK↯
Purpose	Hardware reset
Description	This has the same effect as initialization by pressing the factory reset button. All parameters are initialized to factory defaults
Reference	ATZ

B.3.3. AT↵

Response	↯OK↯
Purpose	Check the connection status with host equipment
Description	Check if the connection to host equipment is operating normally. The serial parameters of Parani-ESD must be same as those of host equipment. If not, the Parani-ESD will not respond or 'ERROR' message will appear or an abnormal sequence of strings will appear.
Reference	AT+UARTCONFIG, ATZ, AT&F

B.3.4. AT+UARTCONFIG,Baudrate,Parity,Stopbit,Hwfc↵

Response	↯OK↯
Purpose	Set Serial parameters
Parameters	Baudrate=1200/2400/9600/14400/19200/38400/57600/115200/230400 (Default=9600) Parity=N/E/O (Default=N) Stopbit=1/2 (Default=1) Hwfc(Hardware Flow Control)=0/1 (Default=1)
Description	The Serial parameters can be set or changed. The factory default is 9600, N, 1. To take effect the ATZ command must be used or Powercycle the unit.
Reference	AT, ATZ, AT&F, ATS
Example	AT+UARTCONFIF,9600,N,1

B.3.5. AT+BTINFO?↵

Response	↯112233445566,DeviceName,Mode,Status,Auth,Encryp,FlowControl↯ ↯OK↯
Purpose	Display Bluetooth settings
Description	The current Bluetooth settings are displayed including BD address, Device name, Operation mode, Operation status, Authentication, Data Encryption, and Hardware Flow Control. The initial value of Device name is 'ESD100v1.1.3-445566'. ESD stands for Parani-ESD, v1.1.3 for the version of firmware, and 445566 for the last 6 digits of BD address. Mode=MODE0/MODE1/MODE2/MODE3 Status=STANDBY/PENDING/CONNECT Auth=0/1 (Authentication is not activated when 0) Encrypt=0/1 (Encryption is not activated when 0) FlowControl=HWFC/NoFC
Reference	AT+BTNAME, AT+BTMODE, AT+BTSEC, ATS14?
Example	↯000B530011FF,SENA,MODE0,PENDING,1,1,HWFC↯

B.3.6. AT+BTINQ?↵

Response	↯112233445566,FriendlyName,CoD↯ ↯112233445566,FriendlyName,CoD↯ ↯112233445566,FriendlyName,CoD↯ ↯OK↯
Purpose	Search Bluetooth devices nearby
Description	The Bluetooth devices in Inquiry scan mode nearby are displayed with their BD addresses, Device names, and Class of device. Maximum 15 devices are scanned for 30 seconds. (Default 10 value in S-register 6)
Reference	AT+BTSCAN, ATD, AT+BTINFO?

B.3.7. AT+BTLAST?↵

Response	↯112233445566↯
Purpose	Display the BD address of the last connected device
Description	The Bluetooth device last connected to this Parani-ESD is displayed with its BD address.
Reference	AT+BTSCAN, ATD, AT+BTINFO?, AT+BTINQ?

B.3.8. AT+BTVER?↵

Response	↯ESD100v1.1.3↯ ↯OK↯
Purpose	Display device firmware version
Description	Display device firmware version
Reference	AT+BTINFO?

B.3.9. AT+BTRSSI,n↵

Response	↯OK↯ ↯0,255,0,0↯(repeatedly)

Purpose	Test signal strength
Parameters	n=0: Stop signal strength test n=1: Start signal strength test
Description	When Bluetooth connection is established, you can use this command in Standby status. The signal strength will be displayed repeatedly in order of Status, LinkQuality, Status, RSSI. If the LinkQuality is close to 255 and RSSI is close to 0, the signal strength is in good standing.
Example	+++ AT+BTRSSI,1 ↯OK↯ 0,255,0,0

B.3.10. AT+BTMODE,n↵

Response	↯OK↯
EPurpose	Set operation mode
Parameters	n=0: MODE0 (Default) n=1: MODE1 n=2: MODE2 n=3: MODE3
Description	When the operation status is 'Pending' currently, change the status to 'Standby' with AT+BTCANCEL prior to this command. To take effect the ATZ must be executed or Powercycle the unit
Reference	AT+BTINFO?
Example	AT+BTMODE,2 ↯OK↯ ATZ

B.3.11. +++↵

Response	↯OK↯
Purpose	Convert the operation status of 'Connect' to 'Standby'
Description	In 'Connect' status, data from host is transmitted to the other side Bluetooth device, and any AT command is not accepted but this command, which is not echoed on the screen. When Parani-ESD encounters a character '+' from host, it stops the data transmission and waits for next 2 characters. If the next 2 characters aren't both '+', it restart to transmit data including the first '+' as well. If not, it converts the operation status to 'Standby'. If the data from host includes '+++', it will convert the operation status to 'Standby'. Notice that Parani-ESD holds data transmission when it encounters '+', until receiving next character. '+' is an escape sequence character by default, which is changeable by AT+SETESC.
Reference	AT+SETESC, ATO, AT+BTCANCEL

B.3.12. AT+SETESC,nn↵

Response	↯OK↯
Purpose	Change the escape sequence character
Description	Escape sequence character set to '+' by default is changeable. The parameter nn must be a printable character.
Reference	+++, ATO

Example	AT+SETESC,42

B.3.13. ATO↵

Response	None
Purpose	Convert the operation status of 'Standby' to 'Connect'
Description	You can convert the operation status of 'Standby' to 'Connect' ready to transmit data.
Reference	+++, AT+SETESC

B.3.14. AT+BTCANCEL↵

Response	↯OK↯
Purpose	Terminate the current executing task
Description	This terminates a current executing task, such as Inquiry scan and Page scan, then converts the operation status to 'Standby'
Reference	AT+BTSCAN, ATD, AT+BTINQ?

B.3.15. AT+BTSCAN↵

Response	↯OK↯ ↯CONNECT 112233445566↯
Purpose	Wait for inquiry and connection from other Bluetooth devices
Description	This allows the inquiry and connection from the other Bluetooth devices. The operation status will be in 'Pending' after this command. When connection is made and released, the operation status is back to 'Pending'. To convert the operation status to 'Standby' AT+BTCANCEL must be used. This has the same effect as AT+BTSCAN,3,0. When connection is made with other Bluetooth device, response will be 'CONNECT' with its BD address.
Reference	ATD, AT+BTINQ?, AT+BTCANCEL

B.3.16. AT+BTSCAN,n,to↵

Response	↯OK↯ ↯CONNECT 112233445566↯ or ↯OK↯ ↯ERROR↯
Purpose	Wait for inquiry and connection from other Bluetooth devices for a given duration
Parameters	n=1: Allows Inquiry scan n=2: Allows Page scan n=3: Allows both of Inquiry scan and Page scan to= Time duration in seconds
Description	For the given to, Parani-ESD is waiting for the inquiry and connection from other Bluetooth devices. If the parameter of to is 0, it will wait forever. When connection is made with other Bluetooth device, response will be 'CONNECT' with its BD address. If there is no connection made within this time duration, response is 'ERROR' and the operation status becomes to 'Standby'.
Reference	ATD, AT+BTINQ?, AT+BTCANCEL
Example	AT+BTSCAN,2,30

B.3.17. AT+BTSCAN112233445566,to↵

Response	↯OK↯ ↯CONNECT 112233445566↯ or ↯OK↯ ↯ERROR↯
Purpose	Wait for connection by the Bluetooth device with given BD address
Parameters	112233445566=BD address to= time duration in seconds
Description	Parani-ESD will wait to be connected to by the Bluetooth device with the given BD address. If the parameter of to is 0, it will wait forever. When connection is made with the Bluetooth device, response will be 'CONNECT' with its BD address. If there is no connection made within this time duration, response is 'ERROR' and the operation status becomes to 'Standby'.
Reference	ATD, AT+BTINQ?, AT+BTCANCEL
Example	AT+BTSCAN000B530011FF,30

B.3.18. ATD↵

Response	↯OK↯ ↯CONNECT 112233445566↯ or ↯OK↯ ↯ERROR↯
Purpose	Connect to the last connected Bluetooth device
Description	Parani-ESD saves the BD address of the Bluetooth device most recently connected to. If it fails to make a connection, response will display an 'ERROR'.
Reference	AT+BTINQ?, AT+BTSCAN

B.3.19. ATD112233445566↵

Response	↯OK↯ ↯CONNECT 112233445566↯ or ↯OK↯ ↯ERROR↯
Purpose	Connect to a specific Bluetooth device with a given BD address
Parameters	112233445566=BD address
Description	Parani-ESD attempts to connect to the Bluetooth device with the given BD address. To make successful connection, the Bluetooth device must be in Page scan mode. This attempt continues for 5 minutes. If it fails to make connection, response is 'ERROR'.
Reference	AT+BTINQ?, AT+BTSCAN
Example	ATD000B530011FF

B.3.20. ATA↵

Response	↯OK↯ ↯Start ACL Open↯ ↯ACL Connect Success↯ or ↯ACL Connect Fail↯
Purpose	ACL connect to the last connected Bluetooth device
Description	If it make connection, response will display an 'ACL Connect Success', and if fail to connection, display 'ACL Connection Fail'. Must have reboot for new ACL connection.

B.3.21. ATA112233445566↵

Response	↯OK↯ ↯Start ACL Open↯ ↯ACL Connect Success↯ or ↯ACL Connect Fail↯
Purpose	ACL connect to a specific Bluetooth device with a given BD address
Parameters	112233445566 = BD address
Description	Parani-ESD attempts to ACL connect to the Bluetooth device with the given BD address. To make successful ACL connection, the Bluetooth device must be in Page scan mode. If it make connection, response will display an 'ACL Connect Success', and if fail to connection, display 'ACL Connection Fail'. Must have reboot for new ACL connection.
Example	ATA000B530011FF

B.3.22. ATH↵

Response	↯OK↯ ↯DISCONNECT↯
Purpose	Release the current connection
Description	The current Bluetooth connection will be disconnected. It takes about 30 seconds to detect an abnormal disconnection such as power off and moving out of service range.
Reference	ATD, AT+BTSCAN

B.3.23. AT+BTKEY=$string↵

Response	↯OK↯
Purpose	Change pin code
Parameters	$*string*= New pin code (Default="1234")
Description	Pin code is a string, which allows up to 16 alpha-numeric characters. Based on this pin code, Parani-ESD generates a link key which is used in actual authentication process
Reference	AT+BTCSD, AT+BTFP, AT+BTSD?, AT+BTSEC, ATZ, AT&F
Example	AT+BTKEY="apple"

B.3.24. AT+BTSD? ↵

Response	↯112233445566↯ ↯OK↯

Purpose	Display a list of Bluetooth devices sharing the same pin code
Description	Once a connection is made with a pin code, Parani-ESD saves the Bluetooth device with its link key, generated by the pin code. The connection to a device listed in Parani-ESD can be made automatically without the authentication process. The maximum number kept on the list is 5.
Reference	AT+BTCSD, AT+BTFP, AT+BTKEY, AT+BTSEC, ATZ, AT&F

B.3.25. AT+BTCSD↵

Response	↯OK↯
Purpose	Clear the list of Bluetooth devices sharing the same pin code
Description	This clears the list of Bluetooth devices linked with the same key in flash memory. To take effect the ATZ command must be used or Powercycle the unit.
Reference	AT+BTFP, AT+BTKEY, AT+BTSD?, AT+BTSEC, ATZ, AT&F

B.3.26. AT+BTFP,n↵

Response	↯OK↯
Purpose	Set generation of link key every time of connection
Parameters	n=0: Inactivate (Default) n=1: Activate
Description	If n is set to 1, Parani-ESD asks for the pin code every time a connection is made. This can be used to increase security.
Reference	AT+BTCSD, AT+BTKEY, AT+BTSD?, AT+BTSEC, ATD, ATZ, AT&F

B.3.27. AT+BTSEC,Authentication,Encryption↵

Response	↯OK↯
Purpose	Set authentication and data encryption
Parameters	*Authentication*=0: Inactivate (Default) *Authentication*=1: Activate *Encryption*=0: Inactivate (Default) *Encryption*=1: Activate
Description	If the authentication is activated, the pin code must be set by AT+BTKEY command. Data encryption cannot be used when authentication is not enabled, i.e. *Authentication*=0 and *Encryption*=1 will not work properly.
Reference	AT+BTCSD, AT+BTFP, AT+BTSD?, AT+BTSD?, ATZ, AT&F

B.3.28. AT+BTNAME=$*string*↵

Response	↯OK↯
Purpose	Change device name
Parameters	*$string*= New device name (Default="ESDv1.1.3-445566")
Description	Parani-ESD can have a user friendly name for easy identification. The name allows up to 30 alpha-numeric characters.
Reference	AT+BTINFO?, AT+BTINQ?
Example	AT+BTNAME="My-Parani-ESD"

B.3.29. AT+BTLPM,n↵

Response	↯OK↯
Purpose	Set low power mode
Parameters	n=0: Inactivate (Default) n=1: Activate
Description	During no data transmission, Parani-ESD can be in low power mode to save the power. It takes a few seconds to wake the Parani-ESD out of low power mode.

B.3.30. AT&V↵

Response	↯S0:m0;S1:m1; ···Sn:mn↯ ↯OK↯
Purpose	Display all the S-register
Description	All parameters are stored at S-register in flash memory. These values are sustained until hardware reset.
Reference	ATS

B.3.31. ATSnn? ↵

Response	↯value↯ ↯OK↯
Purpose	Display a given S-register
Parameters	nn= Address of S-register
Description	A specific S-register will be displayed.
Reference	AT&V

B.3.32. ATSnn=mm↵

Response	↯OK↯
Purpose	Change S-register value
Parameters	nn= Address of S-register mm= New value of S-register
Description	Some S-registers are optimized for the overall performance and protected and cannot be changed. When users try to change these S-registers, response is 'ERROR'. For details of S-register, refer Appendix. B.
Reference	AT&V
Example	ATS10=0

B.4. Command Validity

AT Command	Operation Status		
	Standby	**Pending**	**Connect**
AT	○	○	

ATZ	○	○	
AT&F	○	○	
AT+BINQ?	◎		
ATD112233445566	◎		
ATD	◎		
ATA112233445566	◎		
ATA	◎		
AT+BTSCAN	◎		
AT+BTSCAN,n,to	◎		
AT+BTSCAN112233445566,to	◎		
AT+BTCANCEL		○	
+++			○
AT+SETESC	◎		
ATO	●		
ATH	●		
AT+BTSEC,Auth,Encr	◎		
AT+BTLAST?	○	○	
AT+BTMODEn	◎		
AT+BTNAME="Name"	◎		
AT+BTKEY="nnnn"	◎		
AT+BTINFO?	○		
AT+BTLPM,n	◎		
AT+BTSD?	○	○	
AT+BTCSD	◎		
AT+BTFP,n	◎		
AT+UARTCONFIG,b,p,s,h	◎		
AT+USEDIP?	○	○	
AT+BTVER?	○	○	
AT+BTRSSI,n	●		

◎ Valid only when Parani-ESD is not connected to other Bluetooth device.
● Valid only when Parani-ESD is connected to other Bluetooth device.

Appendix C: S-Register

S-registers contains 52 parameters for the Parani-ESD. These are stored in flash memory and the values will be saved unless hardware reset is executed. The value of S-register can be accessed and changed with ATS command. Some S-registers not shown below are set to maximize the performance of Parani-ESD. Thus it is not recommended to change these S-registers.
Change the value of S-register only in Standby status. Turn Parani-ESD off and on.

C.1. S1: Force to Reconnect (default 1)

S1=0, Parani-ESD in Mode1 does not try to reconnect when disconnected.
S1=1, Parani-ESD in Mode1 keeps trying to reconnect when disconnected.

C.2. S3: Stream UART Policy (default 0)

S3=0, the priority of UART streaming is throughput.
S3=1, the priority is latency, which minimizes the delay of data transmission. This is useful in case of transmitting very small data quickly.
When this value is 1, in order to minimize latency, Parani-ESD sends the received data immediately. When this value is 0, the Parani-ESD maximizes throughput, the Parani-ESD stores received data for a short time and sends a large data packet. If the packet length is less than 100 bytes, having latency being the priority is recommended. If the packet length is more than 100 bytes, having throughput as the priority is recommended. Also, if you want to use high baudrate, throughput priority will be more effective. Just for reference, the buffer length for receiving data is 2 Kbytes.

C.3. S4: Enable Remote Name Query (default 1)

S4=0, Parani-ESD will query only the BD address. This speeds up the inquiry process.
S4=1, Parani-ESD will query the BD address, device name and class of device.
When this value is 1, Parani-ESD finds not only BD address but also friendly name. When this value is 0, Parani-ESD finds only BD address. When set to 0 this will make queries much faster. When using the pairing button, finding friendly name will be omitted automatically.

C.4. S6: Enable Low Power Mode (default 0)

S6=0, deactivate Low Power Mode.
S6=1, activate Low Power Mode.
This value decides whether Parani-ESD works in Low Power Mode or not. When this value is 0, Parani-ESD works only in active power mode. When Parani-ESD works in Low Power mode, delay in transferring data may occur.

C.5. S10: Enable Response Message (default 1)

S10=0, Parani-ESD does not send response messages to the host system.
S10=1, Parani-ESD sends response messages to host system.
This value decides whether Parani-ESD sends response messages such as OK, ERROR, CONNECT, DISCONNECT or not. When this value is 0, Parani-ESD will not send any response messages. If the response messages conflicts with your host programs or devices that is connected to Parani-ESD, change this value to 0.

C.6. S11: Enable Escape (default 1)

S11=0, Parani-ESD does not allow escape sequence characters. The operation status of Connect cannot be changed to Standby. Since the Parani-ESD skips the process of detecting escape sequence characters, more efficient data transmission can be had.
S11=1, Parani-ESD allows for the escape sequence charactesr. Whenever it is needed, the Connect

status can be changed to Standby.

C.7. S12: Clear Data Buffer When Disconnected (default 0)

S12=0, Parani-ESD does not clear the data buffer received from host system when disconnected.
S12=1, Parani-ESD clears the data buffer when disconnected.

C.8. S13: Enable DCD Signal (default 1)

S13=0, DCD signal off
S13=1, DCD signal on

C.9. S14: Enable DTR Transfer (Only ESD100/110, default 1)

S14=0, DTR/DSR signal is transferred in a loop-back fashion..
S14=1, DTR signal is transferred to DSR of remote device.

C.10. S15: Enable Disconnect by DTR (Only ESD100/110, default 0)

S15=0, DTR signal cannot release the connection.
S15=1, The Bluetooth connection can be released when DTR signal is off.
This value decides whether Bluetooth connection is released when DTR signal drops or not. If this value is 1, you can use DTR signal in order to disconnect Bluetooth connection.

C.11. S22: Faster Connection (default 0)

S22=0, none
S22=1, page scan
S22=2, inquiry scan
S22=3, page/inquiry scan

C.12. S23: Intercharacter Timeout Setting (default 0)

S23=0 : Not used
S23=1 : 1 x S26
S23=2 : 10 x S26
S23=3 : 100 x S26
Connecting time is average 1.5sec faster than normal mode.

C.13. S24: Maximum Number of Inquiry Result (default 10)

The maximum number of inquiry list can be controlled. This value is up to 15.

C.14. S26: Intercharacter Timeout (default 0)

S23=1 x S26=50 : Timeout-> 50msec
S23=2 x S26=50 : Timeout-> 500msec
S23=3 x S26=3 : Timeout-> 300msec

Inter Character Time Out	* Optimal Value(S23 x S26)
50ms	180
100ms	235
200ms	340

* When 10 bytes data are sent every intercharacter timeout, they are sent separately by 10 bytes at

the optimal value. If the intercharater timeout is set below the optimal value, the data will be put together and sent by 20, 30, 40 bytes or more.

C.15. S28: Escape Sequence Character (default 43)

The decimal number of the ASCII code of escape sequence character can be controlled. The initial value is 43, the ASCII code of '+'.

C.16. S31: Page Timeout (default 300)

This is the timeout in seconds to attempt connection with the ATD command. After this timeout expires, the Parani-ESD will restart automatically. If this value is 0, Parani-ESD will attempt to connect without restarting

C.17. S33: Inquiry Timeout (default 30)

This is the timeout in seconds to execute inquiry scan.

C.18. S37: Supervision Timeout (default 16000)

This is the timeout in 625μsec to presume disconnection, which is set to 16000 initially. (16000×625μsec=10sec)
The smaller the value becomes, the more quickly Parani-ESD can detect an abnormal disconnection. But when the communication is suspended, it may be regarded as disconnection.

C.19. S45: IAC(Inquiry access code)

The reserved IAC addresses are 0x9E8B00 ~ 0x9E8B3F. The general inquiry IAC is 0x9E8B33. So default value is 0x9E8B33.

C.20. S46: BD Address of Last Connected Device

This saves the BD address of the Bluetooth device connected most recently.

C.21. S47: Select Low Power Mode (default 0)

S47=0, Select Park Mode
S47=1, Select Sniff Mode
When this value is 0, Parani-ESD works only in park mode. When Parani-ESD works in park mode, delay in transferring data may occur. And if the value is 1, Parani-ESD works in sniff mode. Parani-ESD works always low power mode but the data will be put together.

C.22. S48: Low Power Max Interval (default 2048)

This is the max interval value to use low power mode, which is set to 2048 initially. (2048 x 625μsec = 1280msec)

C.23. S49: Low Power Min Interval (default 800)

This is the min interval value to use low power mode, which is set to 800 initially. (800 x 625μsec = 500msec)
When sniff mode, a small sniff interval increases power consumption, a large sniff interval increases latency. For low power modes, typical values of the sniff interval are from about 0.5 second to 1.28 seconds

C.24. S50: Low Power Sniff Attempt (default 8)

This is the sniff attempt value, which is set to 8 initially. (8 x 0.625μsec = 5msec)

C.25. S51: Low Power Sniff Timeout (default 8)

This is the sniff timeout value, which is set to 8 initially. (8 x 0.625μsec = 5msec)
The sniffing slave listens for traffic during the sniff slots determined by the sniff attempt parameter. If no message addressed to the sniffing slave is received, the sniffing slave ceases listening for packets. If a message with the sniffing slave's active member address is received, it continues listening for further sniff timeout slots after the sniff slot.

C.26. S52: Low Power Park Timeout (default 5)

This is the park timeout value, which is set to 5 initially. (5sec)
During no data transmission in park timeout, Parani-ESD can be in low power mode to save the power. It takes a few seconds to wake the Parani-ESD out of low power mode.

Appendix D: Trouble Shooting

D.1. No Data Transmission

D.1.1. Device Settings

Check whether the Baud rate of Parani-ESD matches that of its host equipment.
Check whether the host equipment of Parani-ESD uses Hardware Flow Control. Parani-ESD is initially set to Use of Hardware Flow Control. If your host equipment does not use Hardware Flow Control, please disable the Hardware flow control option by ParaniWIN or AT command.

D.2. Data Loss or Malfunctioning

D.2.1. Hardware Flow Control

When transmitting large amounts of data with No Hardware Flow Control, Parani-ESD may clear the data buffer unexpectedly. The possibility becomes greater as the RF transmission environment becomes worse.

D.2.2. Response Message

AT response messages from the Parani-ESD may affect the host system unexpectedly. Do not use Parani-ESD If your applications cannot allow for this wireless time delay.

D.3. Transmission Delay

D.3.1. RF Processing Delay

It takes 30msec approximately for a Parani-ESD to complete a data transmission to the other Bluetooth device. This time delay cannot be reduced and may enlarge as the RF transmission environment becomes worse. Do not use Parani-ESD If your applications cannot allow for this time delay.

D.3.2. RF Transmission Environment

If there are many Bluetooth devices working in a small area and/or the RF communication distance is too great and/or there are some obstacles affecting RF performance, the Parani-ESD repeats the transmission packet by packet due to interferences and/or low RF performance. This may lead to increased data transmission time delays.

ADXL 202 Data Sheet

Low Cost ±2 *g*/±10 *g* Dual Axis *i*MEMS® Accelerometers with Digital Output

ADXL202/ADXL210

FEATURES

2-Axis Acceleration Sensor on a Single IC Chip
Measures Static Acceleration as Well as Dynamic Acceleration
Duty Cycle Output with User Adjustable Period
Low Power <0.6 mA
Faster Response than Electrolytic, Mercury or Thermal Tilt Sensors
Bandwidth Adjustment with a Single Capacitor Per Axis
5 m*g* Resolution at 60 Hz Bandwidth
+3 V to +5.25 V Single Supply Operation
1000 *g* Shock Survival

APPLICATIONS

2-Axis Tilt Sensing
Computer Peripherals
Inertial Navigation
Seismic Monitoring
Vehicle Security Systems
Battery Powered Motion Sensing

GENERAL DESCRIPTION

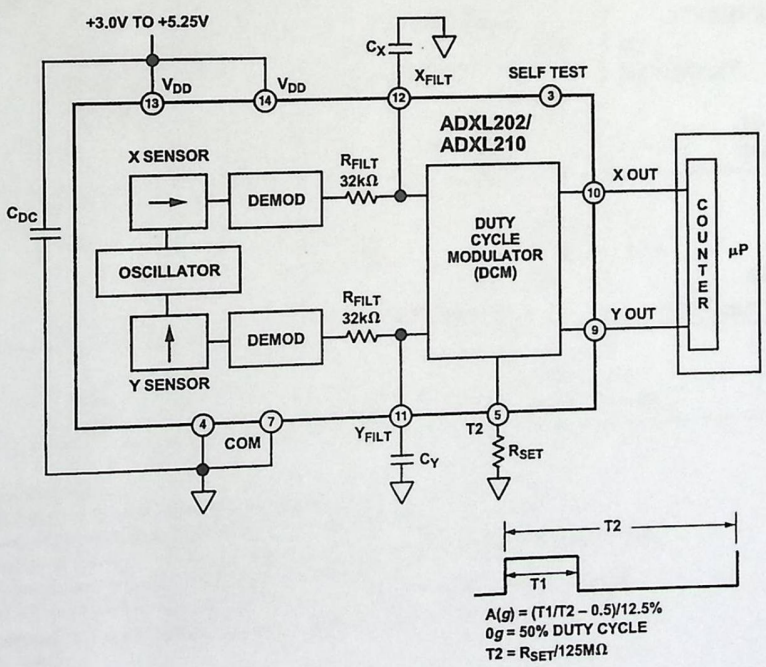
The ADXL202/ADXL210 are low cost, low power, complete 2-axis accelerometers with a measurement range of either ±2 *g*/±10 *g*. The ADXL202/ADXL210 can measure both dynamic acceleration (e.g., vibration) and static acceleration (e.g., gravity).

The outputs are digital signals whose duty cycles (ratio of pulsewidth to period) are proportional to the acceleration in each of the 2 sensitive axes. These outputs may be measured directly with a microprocessor counter, requiring no A/D converter or glue logic. The output period is adjustable from 0.5 ms to 10 ms via a single resistor (R_{SET}). If a voltage output is desired, a voltage output proportional to acceleration is available from the X_{FILT} and Y_{FILT} pins, or may be reconstructed by filtering the duty cycle outputs.

The bandwidth of the ADXL202/ADXL210 may be set from 0.01 Hz to 5 kHz via capacitors C_X and C_Y. The typical noise floor is 500 $\mu g/\sqrt{Hz}$ allowing signals below 5 m*g* to be resolved for bandwidths below 60 Hz.

The ADXL202/ADXL210 is available in a hermetic 14-lead Surface Mount CERPAK, specified over the 0°C to +70°C commercial or –40°C to +85°C industrial temperature range.

FUNCTIONAL BLOCK DIAGRAM



*i*MEMS is a registered trademark of Analog Devices, Inc.

REV. B

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781/329-4700 World Wide Web Site: http://www.analog.com
Fax: 781/326-8703

ADXL202/ADXL210–SPECIFICATIONS

($T_A = T_{MIN}$ to T_{MAX}, T_A = +25°C for J Grade only, V_{DD} = +5 V, R_{SET} = 125 kΩ, Acceleration = 0 *g*, unless otherwise noted)

Parameter	Conditions	ADXL202/JQC/AQC Min	Typ	Max	ADXL210/JQC/AQC Min	Typ	Max	Units
SENSOR INPUT	Each Axis							
Measurement Range[1]		±1.5	±2		±8	±10		*g*
Nonlinearity	Best Fit Straight Line		0.2			0.2		% of FS
Alignment Error[2]			±1			±1		Degrees
Alignment Error	X Sensor to Y Sensor		±0.01			±0.01		Degrees
Transverse Sensitivity[3]			±2			±2		%
SENSITIVITY	Each Axis							
Duty Cycle per *g*	T1/T2 @ +25°C	10	12.5	15	3.2	4.0	4.8	%/*g*
Sensitivity, Analog Output	At Pins X_{FILT}, Y_{FILT}		312			100		mV/*g*
Temperature Drift[4]	Δ from +25°C		±0.5			±0.5		% Rdg
ZERO *g* BIAS LEVEL	Each Axis							
0 *g* Duty Cycle	T1/T2	25	50	75	42	50	58	%
Initial Offset			±2			±2		*g*
0 *g* Duty Cycle vs. Supply			1.0	4.0		1.0	4.0	%/V
0 *g* Offset vs. Temperature[4]	Δ from +25°C		2.0			2.0		m*g*/°C
NOISE PERFORMANCE								
Noise Density[5]	@ +25°C		500	1000		500	1000	$\mu g/\sqrt{Hz}$
FREQUENCY RESPONSE								
3 dB Bandwidth	Duty Cycle Output		500			500		Hz
3 dB Bandwidth	At Pins X_{FILT}, Y_{FILT}		5			5		kHz
Sensor Resonant Frequency			10			14		kHz
FILTER								
R_{FILT} Tolerance	32 kΩ Nominal		±15			±15		%
Minimum Capacitance	At X_{FILT}, Y_{FILT}	1000			1000			pF
SELF TEST								
Duty Cycle Change	Self-Test "0" to "1"		10			10		%
DUTY CYCLE OUTPUT STAGE								
F_{SET}			125 MΩ/R_{SET}			125 MΩ/R_{SET}		
F_{SET} Tolerance	R_{SET} = 125 kΩ	0.7		1.3	0.7		1.3	kHz
Output High Voltage	I = 25 μA	V_S – 200 mV			V_S – 200 mV			mV
Output Low Voltage	I = 25 μA			200			200	mV
T2 Drift vs. Temperature			35			35		ppm/°C
Rise/Fall Time			200			200		ns
POWER SUPPLY								
Operating Voltage Range		3.0		5.25	2.7		5.25	V
Specified Performance		4.75		5.25	4.75		5.25	V
Quiescent Supply Current			0.6	1.0		0.6	1.0	mA
Turn-On Time[6]	To 99%		160 C_{FILT} + 0.3			160 C_{FILT} + 0.3		ms
TEMPERATURE RANGE								
Operating Range	JQC	0		+70	0		+70	°C
Specified Performance	AQC	–40		+85	–40		+85	°C

NOTES

[1]For all combinations of offset and sensitivity variation.

[2]Alignment error is specified as the angle between the true and indicated axis of sensitivity.

[3]Transverse sensitivity is the algebraic sum of the alignment and the inherent sensitivity errors.

[4]Specification refers to the maximum change in parameter from its initial at +25°C to its worst case value at T_{MIN} to T_{MAX}.

[5]Noise density ($\mu g/\sqrt{Hz}$) is the average noise at any frequency in the bandwidth of the part.

[6]C_{FILT} in μF. Addition of filter capacitor will increase turn on time. Please see the Application section on power cycling.

All min and max specifications are guaranteed. Typical specifications are not tested or guaranteed.

Specifications subject to change without notice.

ABSOLUTE MAXIMUM RATINGS*

Acceleration (Any Axis, Unpowered for 0.5 ms) ..... 1000 *g*
Acceleration (Any Axis, Powered for 0.5 ms) ..... 500 *g*
+V_S ..... –0.3 V to +7.0 V
Output Short Circuit Duration
(Any Pin to Common) ..... Indefinite
Operating Temperature ..... –55°C to +125°C
Storage Temperature ..... –65°C to +150°C

*Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; the functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Drops onto hard surfaces can cause shocks of greater than 1000 *g* and exceed the absolute maximum rating of the device. Care should be exercised in handling to avoid damage.

PIN FUNCTION DESCRIPTIONS

Pin	Name	Description
1	NC	No Connect
2	V_{TP}	Test Point, Do Not Connect
3	ST	Self Test
4	COM	Common
5	T2	Connect R_{SET} to Set T2 Period
6	NC	No Connect
7	COM	Common
8	NC	No Connect
9	Y_{OUT}	Y Axis Duty Cycle Output
10	X_{OUT}	X Axis Duty Cycle Output
11	Y_{FILT}	Connect Capacitor for Y Filter
12	X_{FILT}	Connect Capacitor for X Filter
13	V_{DD}	+3 V to +5.25 V, Connect to 14
14	V_{DD}	+3 V to +5.25 V, Connect to 13

PACKAGE CHARACTERISTICS

Package	θ_{JA}	θ_{JC}	Device Weight
14-Lead CERPAK	110°C/W	30°C/W	5 Grams

PIN CONFIGURATION

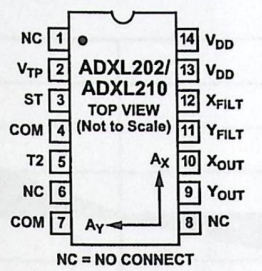


Figure 1 shows the response of the ADXL202 to the Earth's gravitational field. The output values shown are nominal. They are presented to show the user what type of response to expect from each of the output pins due to changes in orientation with respect to the Earth. The ADXL210 reacts similarly with output changes appropriate to its scale.

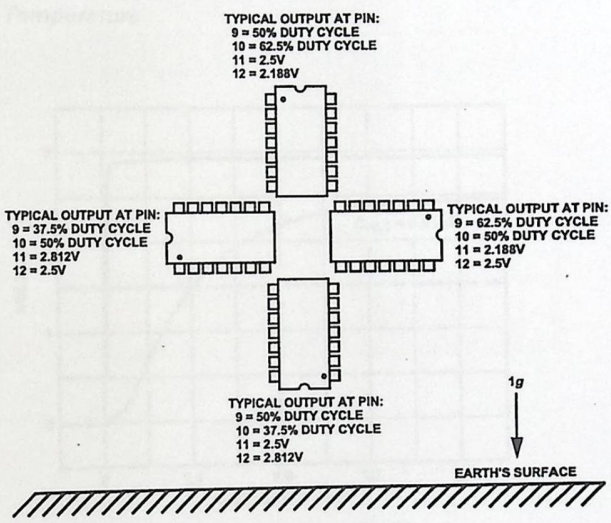


Figure 1. ADXL202/ADXL210 Nominal Response Due to Gravity

ORDERING GUIDE

Model	*g* Range	Temperature Range	Package Description	Package Option
ADXL202JQC	±2	0°C to +70°C	14-Lead CERPAK	QC-14
ADXL202AQC	±2	–40°C to +85°C	14-Lead CERPAK	QC-14
ADXL210JQC	±10	0°C to +70°C	14-Lead CERPAK	QC-14
ADXL210AQC	±10	–40°C to +85°C	14-Lead CERPAK	QC-14

CAUTION

ESD (electrostatic discharge) sensitive device. Electrostatic charges as high as 4000 V readily accumulate on the human body and test equipment and can discharge without detection. Although the ADXL202/ADXL210 features proprietary ESD protection circuitry, permanent damage may occur on devices subjected to high energy electrostatic discharges. Therefore, proper ESD precautions are recommended to avoid performance degradation or loss of functionality.

TYPICAL CHARACTERISTICS (@ +25°C R_{SET} = 125 kΩ, V_{DD} = +5 V, unless otherwise noted)

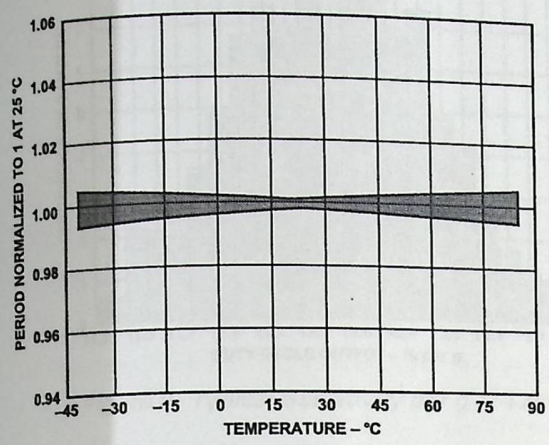


Figure 2. Normalized DCM Period (T2) vs. Temperature

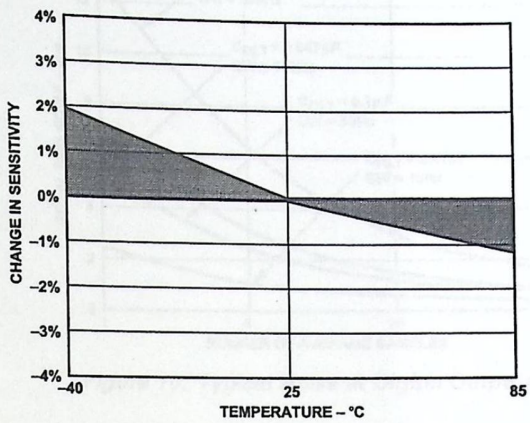


Figure 5. Typical X Axis Sensitivity Drift Due to Temperature

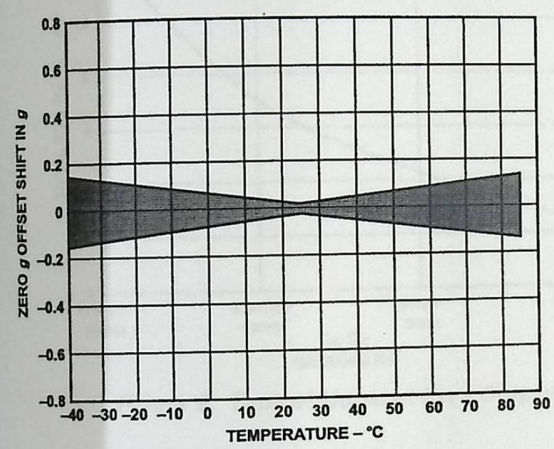


Figure 3. Typical Zero g Offset vs. Temperature

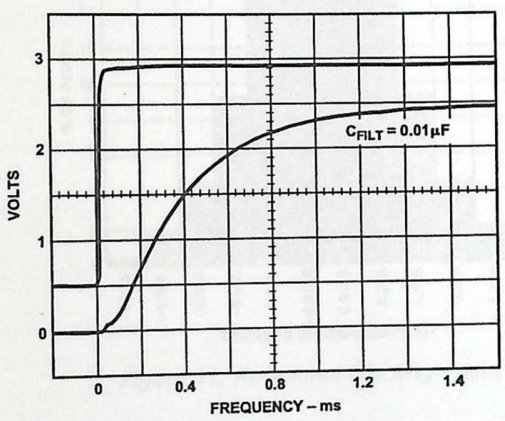


Figure 6. Typical Turn-On Time

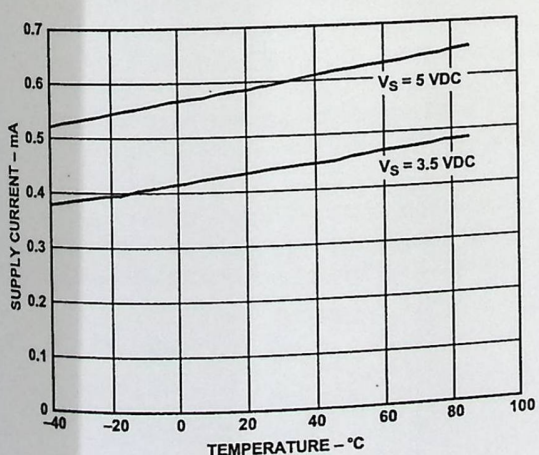


Figure 4. Typical Supply Current vs. Temperature

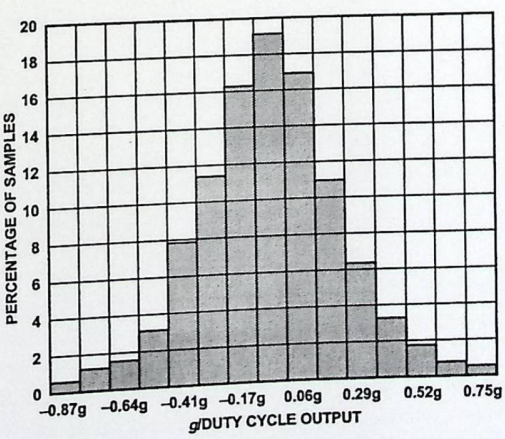


Figure 7. Typical Zero g Distribution at +25°C

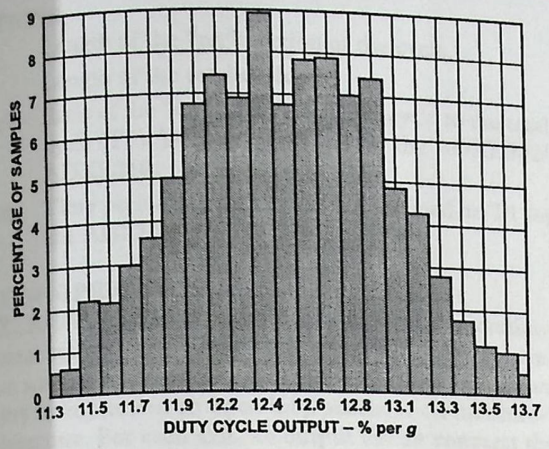


Figure 8. Typical Sensitivity per g at +25°C

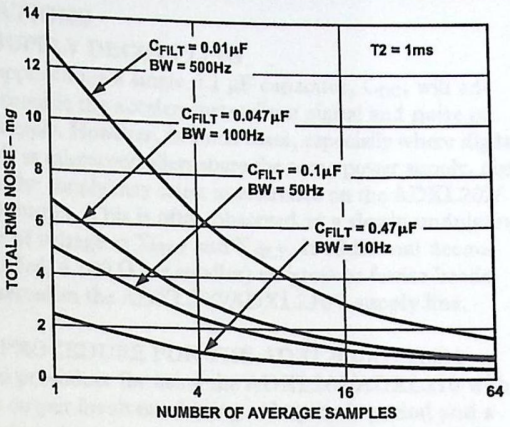


Figure 10. Typical Noise at Digital Outputs

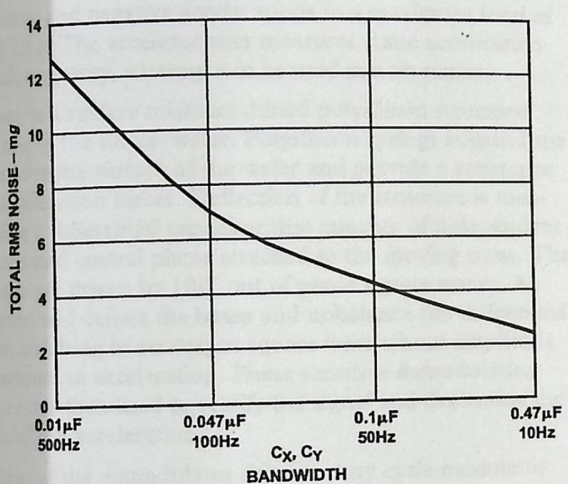


Figure 9. Typical Noise at X_{FILT} Output

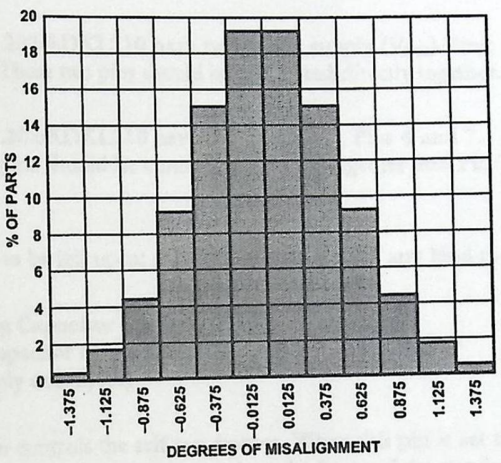


Figure 11. Rotational Die Alignment

DEFINITIONS

T1	Length of the "on" portion of the cycle.
T2	Length of the total cycle.
Duty Cycle	Ratio of the "on" time (T1) of the cycle to the total cycle (T2). Defined as T1/T2 for the ADXL202/ADXL210.
Pulsewidth	Time period of the "on" pulse. Defined as T1 for the ADXL202/ADXL210.

THEORY OF OPERATION

The ADXL202/ADXL210 are complete dual axis acceleration measurement systems on a single monolithic IC. They contain a polysilicon surface-micromachined sensor and signal conditioning circuitry to implement an open loop acceleration measurement architecture. For each axis, an output circuit converts the analog signal to a duty cycle modulated (DCM) digital signal that can be decoded with a counter/timer port on a microprocessor. The ADXL202/ADXL210 are capable of measuring both positive and negative accelerations to a maximum level of ±2 *g* or ±10 *g*. The accelerometer measures static acceleration forces such as gravity, allowing it to be used as a tilt sensor.

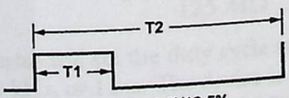
The sensor is a surface micromachined polysilicon structure built on top of the silicon wafer. Polysilicon springs suspend the structure over the surface of the wafer and provide a resistance against acceleration forces. Deflection of the structure is measured using a differential capacitor that consists of independent fixed plates and central plates attached to the moving mass. The fixed plates are driven by 180° out of phase square waves. An acceleration will deflect the beam and unbalance the differential capacitor, resulting in an output square wave whose amplitude is proportional to acceleration. Phase sensitive demodulation techniques are then used to rectify the signal and determine the direction of the acceleration.

The output of the demodulator drives a duty cycle modulator (DCM) stage through a 32 kΩ resistor. At this point a pin is available on each channel to allow the user to set the signal bandwidth of the device by adding a capacitor. This filtering improves measurement resolution and helps prevent aliasing.

After being low-pass filtered, the analog signal is converted to a duty cycle modulated signal by the DCM stage. A single resistor sets the period for a complete cycle (T2), which can be set between 0.5 ms and 10 ms (see Figure 12). A 0 *g* acceleration produces a nominally 50% duty cycle. The acceleration signal can be determined by measuring the length of the T1 and T2 pulses with a counter/timer or with a polling loop using a low cost microcontroller.

An analog output voltage can be obtained either by buffering the signal from the X_{FILT} and Y_{FILT} pin, or by passing the duty cycle signal through an RC filter to reconstruct the dc value.

The ADXL202/ADXL210 will operate with supply voltages as low as 3.0 V or as high as 5.25 V.



A(*g*) = (T1/T2 – 0.5)/12.5%
0*g* = 50% DUTY CYCLE
T2(s) = R_{SET}(Ω)/125MΩ

Figure 12. Typical Output Duty Cycle

APPLICATIONS

POWER SUPPLY DECOUPLING

For most applications a single 0.1 µF capacitor, C_{DC}, will adequately decouple the accelerometer from signal and noise on the power supply. However, in some cases, especially where digital devices such as microcontrollers share the same power supply, digital noise on the supply may cause interference on the ADXL202/ADXL210 output. This is often observed as a slowly undulating fluctuation of voltage at X_{FILT} and Y_{FILT}. If additional decoupling is needed, a 100 Ω (or smaller) resistor or ferrite beads, may be inserted in the ADXL202/ADXL210's supply line.

DESIGN PROCEDURE FOR THE ADXL202/ADXL210

The design procedure for using the ADXL202/ADXL210 with a duty cycle output involves selecting a duty cycle period and a filter capacitor. A proper design will take into account the application requirements for bandwidth, signal resolution and acquisition time, as discussed in the following sections.

V_{DD}

The ADXL202/ADXL210 have two power supply (V_{DD}) Pins: 13 and 14. These two pins should be connected directly together.

COM

The ADXL202/ADXL210 have two commons, Pins 4 and 7. These two pins should be connected directly together and Pin 7 grounded.

V_{TP}

This pin is to be left open; make no connections of any kind to this pin.

Decoupling Capacitor C_{DC}

A 0.1 µF capacitor is recommended from V_{DD} to COM for power supply decoupling.

ST

The ST pin controls the self-test feature. When this pin is set to V_{DD}, an electrostatic force is exerted on the beam of the accelerometer. The resulting movement of the beam allows the user to test if the accelerometer is functional. The typical change in output will be 10% at the duty cycle outputs (corresponding to 800 m*g*). This pin may be left open circuit or connected to common in normal use.

Duty Cycle Decoding

The ADXL202/ADXL210's digital output is a duty cycle modulator. Acceleration is proportional to the ratio T1/T2. The nominal output of the ADXL202 is:

0 *g* = 50% *Duty Cycle*

Scale factor is 12.5% Duty Cycle Change per *g*

The nominal output of the ADXL210 is:

0 *g* = 50% *Duty Cycle*

Scale factor is 4% Duty Cycle Change per *g*

These nominal values are affected by the initial tolerance of the device including zero *g* offset error and sensitivity error.

T2 does not have to be measured for every measurement cycle. It need only be updated to account for changes due to temperature, (a relatively slow process). Since the T2 time period is shared by both X and Y channels, it is necessary only to measure it on one channel of the ADXL202/ADXL210. Decoding algorithms for various microcontrollers have been developed. Consult the appropriate Application Note.

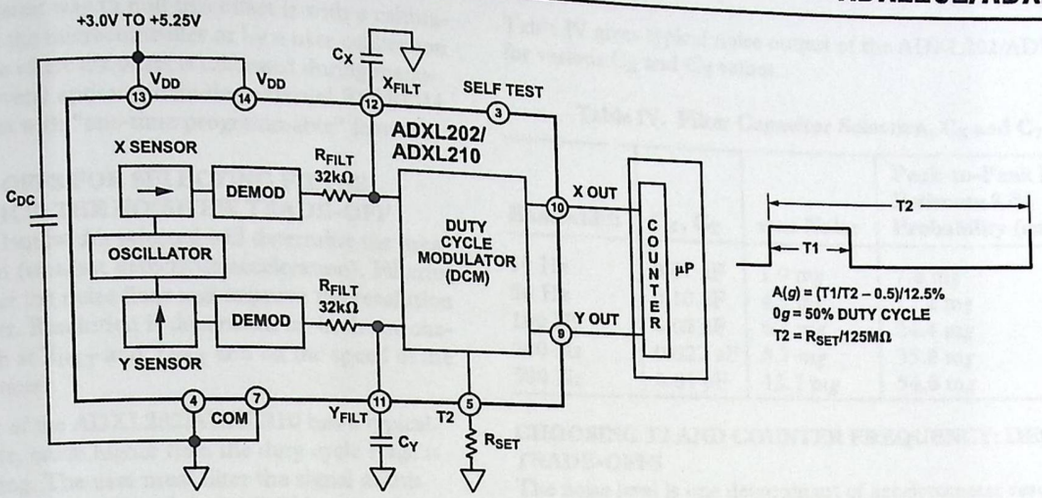


Figure 13. Block Diagram

Setting the Bandwidth Using C_X and C_Y

The ADXL202/ADXL210 have provisions for bandlimiting the X_{FILT} and Y_{FILT} pins. Capacitors must be added at these pins to implement low-pass filtering for antialiasing and noise reduction. The equation for the 3 dB bandwidth is:

$$F_{-3\,dB} = \frac{1}{\left(2\,\pi\,(32\,k\Omega) \times C(x,y)\right)}$$

or, more simply, $F_{-3\,dB} = \frac{5\,\mu F}{C_{(X,Y)}}$

The tolerance of the internal resistor (R_{FILT}), can vary as much as ±25% of its nominal value of 32 kΩ; so the bandwidth will vary accordingly. A minimum capacitance of 1000 pF for $C_{(X, Y)}$ is required in all cases.

Table I. Filter Capacitor Selection, C_X and C_Y

Bandwidth	Capacitor Value
10 Hz	0.47 μF
50 Hz	0.10 μF
100 Hz	0.05 μF
200 Hz	0.027 μF
500 Hz	0.01 μF
5 kHz	0.001 μF

Setting the DCM Period with R_{SET}

The period of the DCM output is set for both channels by a single resistor from R_{SET} to ground. The equation for the period is:

$$T2 = \frac{R_{SET}\,(\Omega)}{125\,M\Omega}$$

A 125 kΩ resistor will set the duty cycle repetition rate to approximately 1 kHz, or 1 ms. The device is designed to operate at duty cycle periods between 0.5 ms and 10 ms.

Table II. Resistor Values to Set T2

T2	R_{SET}
1 ms	125 kΩ
2 ms	250 kΩ
5 ms	625 kΩ
10 ms	1.25 MΩ

Note that the R_{SET} should always be included, even if only an analog output is desired. Use an R_{SET} value between 500 kΩ and 2 MΩ when taking the output from X_{FILT} or Y_{FILT}. The R_{SET} resistor should be place close to the T2 Pin to minimize parasitic capacitance at this node.

Selecting the Right Accelerometer

For most tilt sensing applications the ADXL202 is the most appropriate accelerometer. Its higher sensitivity (12.5%/*g* allows the user to use a lower speed counter for PWM decoding while maintaining high resolution. The ADXL210 should be used in applications where accelerations of greater than ±2 *g* are expected.

MICROCOMPUTER INTERFACES

The ADXL202/ADXL210 were specifically designed to work with low cost microcontrollers. Specific code sets, reference designs, and application notes are available from the factory. This section will outline a general design procedure and discuss the various trade-offs that need to be considered.

The designer should have some idea of the required performance of the system in terms of:

Resolution: the smallest signal change that needs to be detected.

Bandwidth: the highest frequency that needs to be detected.

Acquisition Time: the time that will be available to acquire the signal on each axis.

These requirements will help to determine the accelerometer bandwidth, the speed of the microcontroller clock and the length of the T2 period.

When selecting a microcontroller it is helpful to have a counter timer port available. The microcontroller should have provisions for software calibration. While the ADXL202/ADXL210 are highly accurate accelerometers, they have a wide tolerance for

initial offset. The easiest way to null this offset is with a calibration factor saved on the microcontroller or by a user calibration for zero *g*. In the case where the offset is calibrated during manufacture, there are several options, including external EEPROM and microcontrollers with "one-time programmable" features.

DESIGN TRADE-OFFS FOR SELECTING FILTER CHARACTERISTICS: THE NOISE/BW TRADE-OFF

The accelerometer bandwidth selected will determine the measurement resolution (smallest detectable acceleration). Filtering can be used to lower the noise floor and improve the resolution of the accelerometer. Resolution is dependent on both the analog filter bandwidth at X_{FILT} and Y_{FILT} and on the speed of the microcontroller counter.

The analog output of the ADXL202/ADXL210 has a typical bandwidth of 5 kHz, much higher than the duty cycle stage is capable of converting. The user must filter the signal at this point to limit aliasing errors. To minimize DCM errors the analog bandwidth should be less than 1/10 the DCM frequency. Analog bandwidth may be increased to up to 1/2 the DCM frequency in many applications. This will result in greater dynamic error generated at the DCM.

The analog bandwidth may be further decreased to reduce noise and improve resolution. The ADXL202/ADXL210 noise has the characteristics of white Gaussian noise that contributes equally at all frequencies and is described in terms of µg per root Hz; i.e., the noise is proportional to the square root of the bandwidth of the accelerometer. It is recommended that the user limit bandwidth to the lowest frequency needed by the application, to maximize the resolution and dynamic range of the accelerometer.

With the single pole roll-off characteristic, the typical noise of the ADXL202/ADXL210 is determined by the following equation:

$$Noise\,(rms) = \left(500\ \mu g/\sqrt{Hz}\right) \times \left(\sqrt{BW \times 1.5}\right)$$

At 100 Hz the noise will be:

$$Noise\,(rms) = \left(500\ \mu g/\sqrt{Hz}\right) \times \left(\sqrt{100 \times (1.5)}\right) = 6.12\ mg$$

Often the peak value of the noise is desired. Peak-to-peak noise can only be estimated by statistical methods. Table III is useful for estimating the probabilities of exceeding various peak values, given the rms value.

Table III. Estimation of Peak-to-Peak Noise

Nominal Peak-to-Peak Value	% of Time that Noise Will Exceed Nominal Peak-to-Peak Value
2.0 × rms	32%
4.0 × rms	4.6%
6.0 × rms	0.27%
8.0 × rms	0.006%

The peak-to-peak noise value will give the best estimate of the uncertainty in a single measurement.

Table IV gives typical noise output of the ADXL202/ADXL210 for various C_X and C_Y values.

Table IV. Filter Capacitor Selection, C_X and C_Y

Bandwidth	C_X, C_Y	rms Noise	Peak-to-Peak Noise Estimate 95% Probability (rms × 4)
10 Hz	0.47 µF	1.9 *mg*	7.6 *mg*
50 Hz	0.10 µF	4.3 *mg*	17.2 *mg*
100 Hz	0.05 µF	6.1 *mg*	24.4 *mg*
200 Hz	0.027 µF	8.7 *mg*	35.8 *mg*
500 Hz	0.01 µF	13.7 *mg*	54.8 *mg*

CHOOSING T2 AND COUNTER FREQUENCY: DESIGN TRADE-OFFS

The noise level is one determinant of accelerometer resolution. The second relates to the measurement resolution of the counter when decoding the duty cycle output.

The ADXL202/ADXL210's duty cycle converter has a resolution of approximately 14 bits; better resolution than the accelerometer itself. The actual resolution of the acceleration signal is, however, limited by the time resolution of the counting devices used to decode the duty cycle. The faster the counter clock, the higher the resolution of the duty cycle and the shorter the T2 period can be for a given resolution. The following table shows some of the trade-offs. It is important to note that this is the resolution due to the microprocessors's counter. It is probable that the accelerometer's noise floor may set the lower limit on the resolution, as discussed in the previous section.

Table V. Trade-Offs Between Microcontroller Counter Rate, T2 Period and Resolution of Duty Cycle Modulator

T2 (ms)	R_{SET} (kΩ)	ADXL202/ ADXL210 Sample Rate	Counter-Clock Rate (MHz)	Counts per T2 Cycle	Counts per *g*	Resolution (mg)
1.0	124	1000	2.0	2000	250	4.0
1.0	124	1000	1.0	1000	125	8.0
1.0	124	1000	0.5	500	62.5	16.0
5.0	625	200	2.0	10000	1250	0.8
5.0	625	200	1.0	5000	625	1.6
5.0	625	200	0.5	2500	312.5	3.2
10.0	1250	100	2.0	20000	2500	0.4
10.0	1250	100	1.0	10000	1250	0.8
10.0	1250	100	0.5	5000	625	1.6

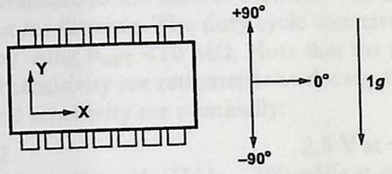
STRATEGIES FOR USING THE DUTY CYCLE OUTPUT WITH MICROCONTROLLERS

Application notes outlining various strategies for using the duty cycle output with low cost microcontrollers are available from the factory.

USING THE ADXL202/ADXL210 AS A DUAL AXIS TILT SENSOR

One of the most popular applications of the ADXL202/ADXL210 is tilt measurement. An accelerometer uses the force of gravity as an input vector to determine orientation of an object in space.

An accelerometer is most sensitive to tilt when its sensitive axis is perpendicular to the force of gravity, i.e., parallel to the earth's surface. At this orientation its sensitivity to changes in tilt is highest. When the accelerometer is oriented on axis to gravity, i.e., near its +1 *g* or –1 *g* reading, the change in output acceleration per degree of tilt is negligible. When the accelerometer is perpendicular to gravity, its output will change nearly 17.5 *mg* per degree of tilt, but at 45° degrees it is changing only at 12.2 *mg* per degree and resolution declines. The following table illustrates the changes in the X and Y axes as the device is tilted ±90° through gravity.



X AXIS ORIENTATION TO HORIZON (°)	X OUTPUT		Y OUTPUT (*g*)	
	X OUTPUT (*g*)	Δ PER DEGREE OF TILT (*mg*)	Y OUTPUT (*g*)	Δ PER DEGREE OF TILT (*mg*)
–90	–1.000	–0.2	0.000	17.5
–75	–0.966	4.4	0.259	16.9
–60	–0.866	8.6	0.500	15.2
–45	–0.707	12.2	0.707	12.4
–30	–0.500	15.0	0.866	8.9
–15	–0.259	16.8	0.966	4.7
0	0.000	17.5	1.000	0.2
15	0.259	16.9	0.966	–4.4
30	0.500	15.2	0.866	–8.6
45	0.707	12.4	0.707	–12.2
60	0.866	8.9	0.500	–15.0
75	0.966	4.7	0.259	–16.8
90	1.000	0.2	0.000	–17.5

Figure 14. How the X and Y Axes Respond to Changes in Tilt

A DUAL AXIS TILT SENSOR: CONVERTING ACCELERATION TO TILT

When the accelerometer is oriented so both its X and Y axes are parallel to the earth's surface it can be used as a two axis tilt sensor with a roll and a pitch axis. Once the output signal from the accelerometer has been converted to an acceleration that varies between –1 *g* and +1 *g*, the output tilt in degrees is calculated as follows:

$$Pitch = ASIN\ (Ax/1\ g)$$
$$Roll = ASIN\ (Ay/1\ g)$$

Be sure to account for overranges. It is possible for the accelerometers to output a signal greater than ±1 *g* due to vibration, shock or other accelerations.

MEASURING 360° OF TILT

It is possible to measure a full 360° of orientation through gravity by using two accelerometers oriented perpendicular to one another (see Figure 15). When one sensor is reading a maximum change in output per degree, the other is at its minimum.

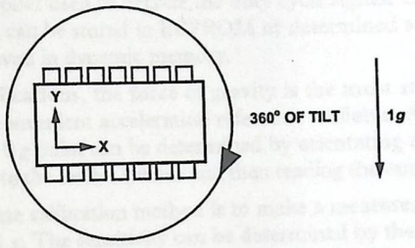


Figure 15. Using a Two-Axis Accelerometer to Measure 360° of Tilt

USING THE ANALOG OUTPUT

The ADXL202/ADXL210 was specifically designed for use with its digital outputs, but has provisions to provide analog outputs as well.

Duty Cycle Filtering

An analog output can be reconstructed by filtering the duty cycle output. This technique requires only passive components. The duty cycle period (T2) should be set to 1 ms. An RC filter with a 3 dB point at least a factor of 10 less than the duty cycle frequency is connected to the duty cycle output. The filter resistor should be no less than 100 kΩ to prevent loading of the output stage. The analog output signal will be ratiometric to the supply voltage. The advantage of this method is an output scale factor of approximately double the analog output. Its disadvantage is that the frequency response will be lower than when using the X_{FILT}, Y_{FILT} output.

X_{FILT}, Y_{FILT} Output

The second method is to use the analog output present at the X_{FILT} and Y_{FILT} pin. Unfortunately, these pins have a 32 kΩ output impedance and are not designed to drive a load directly. An op amp follower may be required to buffer this pin. The advantage of this method is that the full 5 kHz bandwidth of the accelerometer is available to the user. A capacitor still must be added at this point for filtering. The duty cycle converter should be kept running by using R_{SET} <10 MΩ. Note that the accelerometer offset and sensitivity are ratiometric to the supply voltage. The offset and sensitivity are nominally:

0 *g* Offset = $V_{DD}/2$	2.5 V at +5 V
ADXL202 Sensitivity = (60 mV × V_S)/*g*	300 mV/*g* at +5 V, V_{DD}
ADXL210 Sensitivity = (20 mV × V_S)/*g*	100 mV/*g* at +5 V, V_{DD}

USING THE ADXL202/ADXL210 IN VERY LOW POWER APPLICATIONS

An application note outlining low power strategies for the ADXL202/ADXL210 is available. Some key points are presented here. It is possible to reduce the ADXL202/ADXL210's average current from 0.6 mA to less than 20 µA by using the following techniques:

1. Power Cycle the accelerometer.
2. Run the accelerometer at a Lower Voltage, (Down to 3 V).

Power Cycling with an External A/D

Depending on the value of the X_{FILT} capacitor, the ADXL202/ADXL210 is capable of turning on and giving a good reading in 1.6 ms. Most microcontroller based A/Ds can acquire a reading in another 25 µs. Thus it is possible to turn on the ADXL202/ADXL210 and take a reading in <2 ms. If we assume that a 20 Hz sample rate is sufficient, the total current required to take 20 samples is 2 ms × 20 samples/s × 0.6 mA = 24 µA average current. Running the part at 3 V will reduce the supply current from 0.6 mA to 0.4 mA, bringing the average current down to 16 µA.

The A/D should read the analog output of the ADXL202/ADXL210 at the X_{FILT} and Y_{FILT} pins. A buffer amplifier is recommended, and may be required in any case to amplify the analog output to give enough resolution with an 8-bit to 10-bit converter.

Power Cycling When Using the Digital Output

An alternative is to run the microcontroller at a higher clock rate and put it into shutdown between readings, allowing the use of the digital output. In this approach the ADXL202/ADXL210 should be set at its fastest sample rate (T2 = 0.5 ms), with a 500 Hz filter at X_{FILT} and Y_{FILT}. The concept is to acquire a reading as quickly as possible and then shut down the ADXL202/ADXL210 and the microcontroller until the next sample is needed.

In either of the above approaches, the ADXL202/ADXL210 can be turned on and off directly using a digital port pin on the microcontroller to power the accelerometer without additional components. The port should be used to switch the common pin of the accelerometer so the port pin is "pulling down."

CALIBRATING THE ADXL202/ADXL210

The initial value of the offset and scale factor for the ADXL202/ADXL210 will require calibration for applications such as tilt measurement. The ADXL202/ADXL210 architecture has been designed so that these calibrations take place in the software of the microcontroller used to decode the duty cycle signal. Calibration factors can be stored in EEPROM or determined at turn-on and saved in dynamic memory.

For low *g* applications, the force of gravity is the most stable, accurate and convenient acceleration reference available. A reading of the 0 *g* point can be determined by orientating the device parallel to the earth's surface and then reading the output.

A more accurate calibration method is to make a measurements at +1 *g* and –1 *g*. The sensitivity can be determined by the two measurements.

To calibrate, the accelerometer's measurement axis is pointed directly at the earth. The 1 *g* reading is saved and the sensor is turned 180° to measure –1 *g*. Using the two readings, the sensitivity is:

Let A = Accelerometer output with axis oriented to +1 *g*
Let B = Accelerometer output with axis oriented to –1 *g then:*

$$Sensitivity = [A - B]/2\ g$$

For example, if the +1 *g* reading (A) is 55% duty cycle and the –1 *g* reading (B) is 32% duty cycle, then:

$$Sensitivity = [55\% - 32\%]/2\ g = 11.5\%/g$$

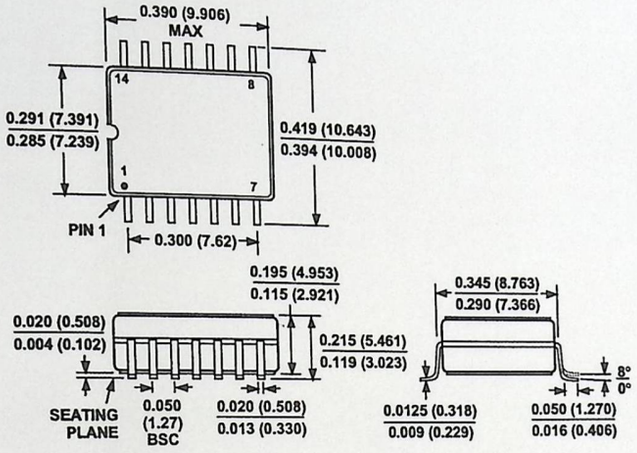
These equations apply whether the output is analog, or duty cycle.

Application notes outlining algorithms for calculating acceleration from duty cycle and automated calibration routines are available from the factory.

OUTLINE DIMENSIONS

Dimensions shown in inches and (mm).

14-Lead CERPAK
(QC-14)



C3037b–2–4/99

PRINTED IN U.S.A.

Appendix B: Source code of PIC program

```
#include<p18f4550.h>

#include<adc.h>
#include<usart.h>
#include<delays.h>
#include <timers.h>

#pragma config WDT = OFF
#pragma config LVP = OFF
#pragma config DEBUG = OFF
#pragma config PWRT = OFF
#pragma config FOSC = XT_XT

#pragma config ICPRT = OFF
#pragma config BOR = OFF
#pragma config CPUDIV = OSC1_PLL2
#pragma config PLLDIV = 1
#pragma config VREGEN = OFF
#pragma config XINST = OFF

float DutyCycleX (int period);
float DutyCycleY (int period);
int Period (void);
int round (float x);
void readings(void);


void send(const rom char *data);

void send(const rom char *data)
{
        char c;

        while ((c = *data++))
        {
                while (PIR1bits.TXIF == 0) //busy
                        ;
                TXREG = c;//send
        }
        ;
}



void main(void)
{
```

```
char temp1[8];
char temp2[26];
char c;


int i;
int Zero_Before = 1 ;
int DutyCyclePeriod;
int LeftPressCounter = 0;

char byte=0xAA;
int TolRight = 5;
int TolLeft = -5;
int TolDown = 5;
int TolUP = -5;
float DutyX,DutyY;
int number = 0;
int result;
        char ch;


TRISD = 0b11111111;
TRISA = 0;
PORTA = 0;


ADCON1= 0b10000000;


OpenUSART (USART_TX_INT_OFF &
        USART_RX_INT_OFF &
        USART_ASYNCH_MODE &
        USART_EIGHT_BIT &
        USART_CONT_RX &
        USART_BRGH_HIGH, 25);

OpenTimer0( TIMER_INT_OFF &
                    T0_8BIT &
                    T0_SOURCE_INT &
                    T0_PS_1_4 );


  TRISC = 0b11000000;
  RCSTAbits.SPEN = 1;
```

```
getsUSART(temp1,6);

A1:send("AT+BTSCAN\r");

getsUSART(temp1,6);

if(temp1[2] != 'O')
{
        send("ERROR\r");
        c = RCREG;
        c = RCREG;
        RCSTAbits.CREN = 0;
        RCSTAbits.CREN = 1;

        goto A1;
}

//\r\nok\r\n

 getsUSART(temp2,24); //wait for connection


DutyCyclePeriod = Period();



while(1)
{

        byte = 0x00;


        if(PORTDbits.RD7 || PORTDbits.RD6 || LeftPressCounter > 30)   // Enable for
mouse MOVE or scroll
        {
                DutyX =  DutyCycleX (DutyCyclePeriod);
                DutyY =  DutyCycleY (DutyCyclePeriod);


                if ( round(DutyX) > 124 + TolRight)     // Right
                {
                        byte = byte | 0b00101000 ;
```

```
}
else if (round(DutyX) <= 124 + TolLeft) // Left
{
        byte = byte | 0b00100000 ;


}


if ( round(DutyY) > 149 + TolDown) //98   // DOWN
{
        byte = byte | 0b00010000 ;



}
else if (round(DutyY) < 149 + TolUP)  //105  //up
{
        byte = byte | 0b00010100 ;


}

}

if(PORTDbits.RD3)               // Left
{       byte = byte | 0b00000010 ;
        LeftPressCounter++;

}

else
        LeftPressCounter = 0;


if(PORTDbits.RD4)               // Right
{
        byte = byte | 0b00000001 ;

}                               // Scroll
if(PORTDbits.RD6)
{
        byte = byte | 0b01000000 ;
```

```
            }


 Delay1KTCYx(3);

       if (byte == 0x00)
            {
                  if(Zero_Before == 0 )  // Previous byte wasn't zero
                  {
                        while(BusyUSART());
                        WriteUSART(byte);
                        Zero_Before = 1 ;
                  }

            }
            else   // Byte isn't zero
            {
                  while (BusyUSART());
                  WriteUSART(byte);
                  Zero_Before = 0 ;
            }




}


 CloseUSART();

 }


 float DutyCycleX (int period)
 {  char ch;
       int Ton,Tsum,i=1;
       Tsum = 0;

       for(i=1;i<=5;i++)
       {
    while (PORTDbits.RD0);
       if(!PORTDbits.RD0)
       {
```

```
        while (!PORTDbits.RD0);
        TMR0L = 0x00;

        while(PORTDbits.RD0);
        Ton = ReadTimer0() ;
        Tsum += Ton;

    }

        }
        return Tsum/(5*1.0);
}

float DutyCycleY (int period)
{   char ch;
        int Ton,Tsum,i=1;
        Tsum = 0;

        for(i=1;i<=5;i++)
        {
    while (PORTDbits.RD1);
        if(!PORTDbits.RD1)
        {


        while (!PORTDbits.RD1);
        TMR0L = 0x00;

        while(PORTDbits.RD1);
        Ton = ReadTimer0() ;
        Tsum += Ton;

     }

        }
        return Tsum/(5*1.0);
  }


  int Period (void)
  {
        // Calculate the period of Xout and Yout
        int period;
        if(PORTDbits.RD0)
      while(PORTDbits.RD0);
```

```
  while(!PORTDbits.RD0);
  TMR0L=0x00;
  while(PORTDbits.RD0);
  while(!PORTDbits.RD0);
        period=TMR0L;
        return period;

}


int round (float x)
{
 return (int)(x+0.5);
}
```

Appendix C: Source code of PC program

```
Imports System.IO.Ports
Imports System
Imports System.Threading

Public Class Eglove
    Dim com_port As String = "Com5"
    Dim Acceleration As Double = 0.05
    Dim Speed As Double = 5

    Dim WithEvents serialPort1 As SerialPort = New _
     System.IO.Ports.SerialPort(com_port, 9600, Parity.None, 8,
StopBits.One)

    Private Declare Sub mouse_event Lib "user32.dll" ( _
    ByVal dwFlags As Int32, _
    ByVal dx As Int32, _
    ByVal dy As Int32, _
    ByVal cButtons As Int32, _
    ByVal dwExtraInfo As Int32 _
    )

    Private Declare Function SetCursorPos Lib "user32.dll" ( _
    ByVal X As Int32, _
    ByVal Y As Int32 _
    ) As Boolean

    Public Const MOUSEEVENTF_LEFTDOWN = &H2
    Public Const MOUSEEVENTF_LEFTUP = &H4
    Public Const MOUSEEVENTF_MIDDLEDOWN = &H20
    Public Const MOUSEEVENTF_MIDDLEUP = &H40
    Public Const MOUSEEVENTF_RIGHTDOWN = &H8
    Public Const MOUSEEVENTF_RIGHTUP = &H10
    Public Const MOUSEEVENTF_MOVE = &H16

    Public Left_Down As Boolean
    Public Right_Down As Boolean
    Public Middle_Down As Boolean
    Dim AccelerationX As Double = 1.0
    Dim AccelerationY As Double = 1.0
    Dim AccelEnable As Boolean = False

    Private Sub ExitToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ExitToolStripMenuItem.Click
        serialPort1.Close()
        End
    End Sub

    Private Sub HideToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
HideToolStripMenuItem.Click
        Me.Hide()
        Me.NotifyIcon1.Visible = True
    End Sub
```

```
    Private Sub AboutToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
AboutToolStripMenuItem.Click
        MsgBox("Glove Program v1.0" & vbCrLf & vbCrLf & "Graduation
Project 2007/2008")

    End Sub

    Private Sub CheckBox1_CheckedChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles CheckBox1.CheckedChanged
        If Me.CheckBox1.CheckState = CheckState.Checked Then
            Me.GroupBox1.Enabled = True
        Else
            Me.GroupBox1.Enabled = False
        End If

    End Sub

    Private Sub ComboBox1_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ComboBox1.SelectedIndexChanged
        com_port = Me.ComboBox1.SelectedItem.ToString

    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        CheckForIllegalCrossThreadCalls = False

        serialPort1.PortName = com_port
        Try
            If serialPort1.IsOpen = False Then serialPort1.Open()

        Catch ex As Exception
            MsgBox(ex.Message & "open")
        End Try

        Left_Down = False
        Right_Down = False

        If Me.CheckBox1.CheckState = CheckState.Checked Then
            Acceleration = 0.05 + Me.TrackBar1.Value / 100
            Speed = 5 + Me.TrackBar2.Value * 3
            AccelEnable = True
        Else
            AccelEnable = False

        End If
        Me.Button1.Enabled = False
        Me.ComboBox1.Enabled = False
    End Sub

    Private Sub TrackBar1_Scroll(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles TrackBar1.Scroll
        Acceleration = 0.05 + Me.TrackBar1.Value / 100

    End Sub
```

```
Private Sub port_DataReceived(ByVal sender As Object, ByVal e As _
    System.IO.Ports.SerialDataReceivedEventArgs) Handles
serialPort1.DataReceived

        Dim readbyte As Byte
        Dim i, j, num As Integer

        For i = 1 To serialPort1.BytesToRead()
            readbyte = serialPort1.ReadByte()

            num = Integer.Parse(readbyte.ToString())

            Dim str As String
            str = IntegerToBits_8(num)

            Dim chArray As Char()
            chArray = New Char(8) {}
            chArray = str.ToCharArray()


            Dim temp As Char
            For j = 0 To 3
                temp = chArray(j)
                chArray(j) = chArray(7 - j)
                chArray(7 - j) = temp
            Next
            MouseOperation(chArray)
        Next

    End Sub
    Private Function IntegerToBits_8(ByVal num As Integer) As String

        Dim temp, i As Integer
        Dim bits As Integer()
        bits = New Integer(8) {}
        Dim n1 As Integer

        For i = 0 To 7
            bits(i) = 0
        Next

        i = 7
        n1 = 128
        temp = num
        While (i >= 0)
            If (temp.Equals(n1)) Then
                bits(i) = 1
                i = 0  ' break the loop
            ElseIf (temp > n1) Then
                bits(i) = 1
                temp = temp - n1

            End If
            i = i - 1
            n1 = n1 / 2
        End While
```

```
    Dim str As String
    str = ""
    i = 7
    While (i >= 0)
        str = str & bits(i).ToString()
        i = i - 1
    End While

    Return str

End Function

Private Sub MouseOperation(ByVal chArray As Char())

    Dim x, y As Integer
    Dim i As Integer

    If AccelEnable = False Then
        AccelerationX = 2
        AccelerationY = 2
    End If

    If (chArray(0).ToString.Equals("1")) Then
        ' " Right Click "
        If Right_Down = False Then
            mouse_event(MOUSEEVENTF_RIGHTDOWN, 0, 0, 0, 0)
            Right_Down = True
        End If

    Else
        ' " Right not "
        If Right_Down = True Then
            mouse_event(MOUSEEVENTF_RIGHTUP, 0, 0, 0, 0)
            Right_Down = False
        End If

    End If

    If (chArray(1).ToString.Equals("1")) Then
        ' " Left Click "
        If Left_Down = False Then
            mouse_event(MOUSEEVENTF_LEFTDOWN, 0, 0, 0, 0)
            Left_Down = True
        End If

    Else
        '  " Left not "
        If Left_Down = True Then
            mouse_event(MOUSEEVENTF_LEFTUP, 0, 0, 0, 0)
            Left_Down = False
        End If


    End If
```

```
If (chArray(6).ToString.Equals("1")) Then
    '  " Scroll Click "
    If Middle_Down = False Then
        mouse_event(MOUSEEVENTF_MIDDLEDOWN, 0, 0, 0, 0)
        Middle_Down = True
    End If

Else
    ' " Scroll not "
    If Middle_Down = True Then
        mouse_event(MOUSEEVENTF_MIDDLEUP, 0, 0, 0, 0)
        Middle_Down = False
    End If

End If


If (chArray(4).ToString.Equals("1")) Then
    If (chArray(2).ToString.Equals("1")) Then
        ' " Up "
        For i = 1 To round(AccelerationY)
            x = Windows.Forms.Cursor.Position.X
            y = Windows.Forms.Cursor.Position.Y
            SetCursorPos(x, y - 1)
        Next
        If (AccelerationY < Speed And AccelEnable) Then
            AccelerationY = AccelerationY + Acceleration
        End If


    Else
        '  " down "
        For i = 1 To round(AccelerationY)
            x = Windows.Forms.Cursor.Position.X
            y = Windows.Forms.Cursor.Position.Y
            SetCursorPos(x, y + 1)
        Next
        If (AccelerationY < Speed And AccelEnable) Then
            AccelerationY = AccelerationY + Acceleration
        End If

    End If
Else
    AccelerationY = 1.0
End If


If (chArray(5).ToString.Equals("1")) Then
    If (chArray(3).ToString.Equals("1")) Then
        '  " right "
        For i = 1 To 1 * round(AccelerationX)
            x = Windows.Forms.Cursor.Position.X
            y = Windows.Forms.Cursor.Position.Y
            SetCursorPos(x + 1, y)
        Next
        If (AccelerationX < Speed And AccelEnable) Then
```

```
                AccelerationX = AccelerationX + Acceleration
            End If


        Else
            ' " left"
            For i = 1 To 1 * round(AccelerationX)
                x = Windows.Forms.Cursor.Position.X
                y = Windows.Forms.Cursor.Position.Y
                SetCursorPos(x - 1, y)
            Next
            If (AccelerationX < Speed And AccelEnable) Then
                AccelerationX = AccelerationX + Acceleration
            End If

        End If

    Else
        AccelerationX = 1.0
    End If
End Sub
Private Function round(ByVal num As Double) As Integer
    Dim num1 As Double
    num1 = num + 0.5
    Return CInt(num1)
End Function

Private Sub TrackBar2_Scroll(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles TrackBar2.Scroll
    Speed = 5 + Me.TrackBar2.Value * 3
End Sub

Private Sub NotifyIcon1_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles NotifyIcon1.Click
    Me.NotifyIcon1.Visible = False
    Me.Show()
    WindowState = FormWindowState.Normal
End Sub

Private Sub HelpMeToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
HelpMeToolStripMenuItem.Click
    MessageBox.Show("To use program, please do the following
steps:" & vbCrLf & vbCrLf & _
    "1. Determine connection port " & vbCrLf & "2. Press start
button to start mouse work" & vbCrLf & _
    "....." &
    "....." &
vbCrLf & vbCrLf & "To make mouse acceleration check the Box" & _
    vbCrLf & "And determine the acceleration and speed you need " &
vbCrLf & _
    "Then press Button apply to perform the operation" & vbCrLf & _
    "....." &
vbCrLf & vbCrLf & _
    "To miimize the program to tray go to file -> Hide or use
Keyboard shortcut Ctrl+H" & vbCrLf & _
    "To show help go to Help ->Help Me or use Keyboard shortcut
Ctrl+F1" & vbCrLf & _
```

```
        "To exit go to File-> Exit or use Keyboard Shortcut Ctrl+x",
"Help Me", MessageBoxButtons.OK, MessageBoxIcon.Information)

    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        If Me.CheckBox1.CheckState = CheckState.Checked Then
            Acceleration = 0.05 + Me.TrackBar1.Value / 100
            Speed = 5 + Me.TrackBar2.Value * 3
            AccelEnable = True
        Else
            AccelEnable = False
        End If
    End Sub


End Class
```