

**ENG 499  
PROJECT  
FINAL YEAR REPORT  
HAND POSITIONED  
MOUSE**

Tutor Name: Dr Thimm Georg Lothar  
Name : Tay Boon Kwong Roland  
Course : ENG 499  
SIM PI NO : E0401574  
DATE : 02/11/2007

## **Abstract**

This project attempts to have a different approach of how a person uses a mouse to interact with his or her computer. A glove is to be designed and worn on the hand of a computer operator to provide a mouse function for a computer. Instead of using LED sensors and ball trackers, a pair of accelerometer sensors is mounted on the finger; each of the sensors is used to detect a negative or positive direction along the located orientation line of the finger. The apparatus senses the direction of the computer operator's body and transmits corresponding commands to the computer to move the cursor in the video display in a corresponding direction. In addition, this invention of the hand positioned mouse is ideal for playing computer games and interacting with services such as web TV. This project will involve both hardware and software components. The hardware design includes schematic drawing; part selection and circuit layout and the programming includes the learning of the Machine/Assembly language and the use of compilers for micro controllers.

## **Acknowledgments**

I have taken information, advice, and inspiration from a number of individuals and like to take this opportunity to thank several people. The support of which has made this project report possible. To my advisor for this project, Dr Thimm Georg Lothar, who had provided countless insights and advice. To the people, in the electrical labs, who had provided the usage of their equipments and components. To my friends and colleagues, who had to put up with me when I got frustrated over my thesis work. To my boss, Mr Thon Pang Yong, who has agreed to lend his workshop and equipments for my project.

Lastly, I would like to thank my family for their support and encouragement, not only for this thesis, but also for all my years at university.

# TABLE OF CONTENTS

	<b>Page</b>	
<b>ABSTRACT</b>	I	
<b>ACKNOWLEDGEMENTS</b>	II	
<b>TABLE OF CONTENTS</b>	III	
<b>LIST OF FIGURES</b>	V	
<b>LIST OF TABLES</b>	VI	
<b>CHAPTER 1 Introduction</b>		
1.1	Background of Problems	1
1.2	Project Objectives	1
1.3	Proposed Approach	2
1.4	Scope of Project	3
<b>CHAPTER 2 Literature Review And Usage</b>		
2.1	Introduction to the Development of Computer Mouse	4
2.2	Background and Context	5
2.3	Investigation of a Mouse Requirement	6
2.3.1	Sample Serial Port Device. How Serial Mouse Works?	6
2.3.2	Methods of Interface	7
2.4	Development of Micro controller	8
2.4.1	Architecture of Micro controller Board	9
2.4.2	CPU Device Overview	10
2.5	Common Serial Protocols used in Peripheral Device	11
2.5.1	USART	12
<b>CHAPTER 3 System Design</b>		
3.1	Hardware	14
3.1.1	5V DC Power Supply Unit	14
3.1.2	Selection of a Most Suitable Micro-controller	16

3.1.3	Comparison of Different Micro-controller	17
3.1.4	Selecting the Microchip PIC18F452 Micro controller	19
3.1.5	Sensors Selection	19
3.1.6	Button Selection	20
3.1.7	MAX232 TTL TO RS-232 Converter	21
3.2	Software Design	22
3.2.1	Interfacing Micro controller to Personal Computer	22
3.2.2	Software Development Tools	23
3.2.3	Writing the software	25
3.2.4	Types of Analog to Digital Conversion	25

## **CHAPTER 4 System Implementation and Detailed Design**

4.1	Hardware Implementation	27
4.1.1	Pin Assignment for the Micro-controller	27
4.1.2	Accelerometer Hardware	29
4.1.3	TTL Signal to RS-232 Signal generation	29
4.2	Software Implementation	30
4.3	Accelerometer Software Design	30
4.4	Interfacing Design (Software)	30
4.5	Analog to Digital Conversion Sequence (Software)	32
4.6	X-Axis Movement Algorithm	33
4.7	Y-Axis Movement Algorithm	35

## **CHAPTER 5 Testing and Results**

5.1	Types of Testing	37
5.2	Initial Test Strategy Used	39
5.3	Timing	41
5.4	Analog to Digital Conversion Results	41

## **CHAPTER 6 Conclusion**

6.1	System Limitation	43
6.2	Conclusion	43
6.3	Recommendation for Future Work	44

## **CHAPTER 7 Critical Review and Reflections**

7.1	Problem Encountered	45
-----	---------------------	----

7.2	Skill Review	46
7.1	Reflection	46

<b>References</b>		48
-------------------	--	----

### **Appendix**

I	(Gantt Chart) Project Schedule	50
II	ASCII Character Map	51
III	Material List and Cost	54
IV	Application source code listing	55

### **List of Figures**

Figure 2.1	The first computer mouse invented in the 1960s	4
Figure 2.2	PIC18F452 (40/44-PIN) Block Diagram	9
Figure 2.3	PIC18F452 (40/44-PIN) Connection	10
Figure 2.4	UART Data Frame	13
Figure 3.1	Functional Blocks Diagram of Hand Positioned Mouse	14
Figure 3.2	L7805CV Voltage Regulator	15
Figure 3.3	L7805CV Voltage Regulator Circuit Connection	15
Figure 3.4	Actual looks of the push buttons	20
Figure 3.5	MAX232 Pin Configurations and Typical Operating	21
Figure 3.6	Signal Diagram of RS232	23
Figure 3.7	MPLAB IDE Integrated Development Environments	24
Figure 3.8	Set-up of the ICD 2 connect to a PC via USB cable and to a PIC18 micro controller via a ICD cable.	24
Figure 4.1	Hand Positioned Mouse Circuit Diagram	28
Figure 4.2	Actual Layout of the Components for Hand Positioned Mouse	28
Figure 4.3	Flow Chart of the Hand Positioned Mouse	31
Figure 4.4	Flow Chart of the Hand Positioned Mouse ADC conversion	32
Figure 5.1	Picture of Hand Positioned Mouse	37
Figure 5.2	Close Up on the Circuitry	37
Figure 5.3	Close Up on the Sensor and Buttons	38
Figure 5.4	Voltage Testing on Vinnic Branded Battery	39
Figure 5.5	Connection of the Two Push Buttons	40
Figure 5.6	Testing with the Buttons	40
Figure 5.7	Sending 'M' to PC	41
Figure 6.1	Reset Button	43
Figure 7.1	Soldering on the Printed Circuit Board	46

### **List of Tables**

Table 1.1	Comparison between Electrical Simulation and Building Prototype	3
Table 2.1	RS232 serial connections	7
Table 2.2	PIC18F452 (40/44-PIN) Features	11
Table 3.1	Comparison of 3 types of Micro-controller	17
Table 3.2	Comparison of 3 types of Micro-controller by cost and software support	17
Table 4.1	Mouse System and MICROSOFT Protocols	29
Table 4.2	NEGATIVE X-Axis Movement Algorithms	34
Table 4.3	POSITIVE X-Axis Movement Algorithms	34
Table 4.4	NEGATIVE Y-Axis Movement Algorithms	35
Table 4.5	POSITIVE Y-Axis Movement Algorithms	36

## **Chapter 1 Introduction**

### **1.1 Background of Problems**

In today's computer technology, a simple mouse, consisting of a small case, is used to function as a pointing device by detecting two-dimensional motion relative to its supporting surface. In olden days, classic ball is used to turn wheels to indicate movement in the hardware. Recently, optical mice with small LED used to detect changes in the surface have become more popular. But many people who play computer games will find it frustrating to apply force to move back and forth on mouse on a surface. This project attempts to have a different approach of how a person uses a mouse to interact with his or her computer. It will also provide a solution to the frustration by any gamer.

### **1.2 Project Objectives**

The aim of this project is to design and develop a motion-sensing glove that will be worn on the hand of a computer operator. The glove will act as a computer mouse function for a computer having a video display highlighted by a cursor. The glove-like apparatus, which senses the direction of movement of a computer operator's hand, will transmits information to the computer to control the cursor movement in the video display in a corresponding direction. The glove will also come with a button that is activated by thumb to represent mouse click functions.

The Project will include:

- Evaluations of project methods, meet user requirements, and generate designs.
- Gathering system specifications and make plans to co-ordinate my activities.
- Evaluation of software development to be implemented onto the hardware interface.
- Perform measurement strategies to check the system and testing for system reliability.
- Perform literature review search by gathering data from various sources.
- Perform modifications on system's shortcomings when necessary.
- Reviewing the need for future expansion to the completed project.

- Providing some recommendations such as alternative solutions or enhancement of the completed project.

### **1.3 Proposed Approach**

The main approach is to design and develop a hand positioned controlled mouse that meets the user's requirements and specifications.

Before deciding on the types of approach, there are number of factors that need to be considered.

Technology –This project will involve both hardware and software components. Most the components are easily available in the market. Therefore, searching for the best choice of components will be carried out. A suitable micro controller will be selected according to the defined requirements. The comparison of the hardware will be based on the performance and specifications of the various types of components.

Ease of implementation –Assemble of components should be easy.

Cost – The budget of building a prototype should be kept as low as possible.

Product requirements – The apparatus should be light and comfortable to wear on the hand. Requirements and specifications will be obtained from external source, group interview in my work areas as well as from the use of literature review.

	Recommended rating value			Electrical simulation or modeling	Build physical prototype
	1	2	3		
Technology	Existing application	Mixed of existing and new application	New application	2	3
Ease of implementation	Easy	Moderate	Difficult	3	1
Cost	Cheap	Moderate	Expensive	1	2
Product requirements	Known	Know partially	Unknown	3	1
Total Scores				9	7

Table 1.1 Comparison between Electrical Simulation and Building Prototype

Besides, I have to ensure that building this project will not have any similarities or duplications of other mouse systems used. To ensure that the system can perform up to expectation, measurement will have to be carried out. The scoring is rated from 1 to 3 with 1 representing low risk and 3 high risks. The total scores for building a prototype is lower than the Electrical simulation.

Method to be employed:

1. Specification definition
2. Hardware design and development
3. Software design and development
4. Implementation
5. Integration and testing

#### 1.4 Scope of Project

Other goals of the project were to develop skills in hardware design and programming. The hardware design included schematic drawing; part selection and circuit layout and the programming included the learning of the Machine/Assembly language and the use of compilers for micro controllers. All these skills were not touched upon in regular classes are taken here.

## Chapter 2 Literature review

### 2.1 Introduction to the Development of Computer Mouse

The first computer mouse which was invented by Dr. Engelbart in 1963-64 was part of an experiment to find better ways to "point and click" on a display screen [22]. The first mouse had only one button and was carved out of wood due to space restrictions. It used two gear wheels perpendicular to each other. The rotation of each wheel translated into motion along one axis. At that time, Engelbart envisaged that users would type on a mini-keyboard with five keys with the other and hold the mouse continuously in one hand.



Figure 2.1 The first computer mouse invented in the 1960s

While working for Xerox PARC, Bill English, also known as the builder of the original mouse, invented the ball mouse in 1972. This was the main type of mechanical mouse in use before the introduction of the optical mouse. A small ball is found inside the mouse, which causes a rolling motion to small wheels that transmit data to a sensor. The sensor then transmits the required data back to the computer which causes the cursor to move in the required direction [29].

In early 1980, as computing power grew cheaper, optical mice came in two different varieties:

- 1) To detect grid lines printed with infrared ink on a special metallic surface, it used an infrared LED and a four-quadrant infrared sensor.
- 2) To track the motion of light dots in a dark field of a printed paper or similar mouse pad, it used a 16-pixel visible-light image sensor with integrated motion detection on the same chip.

In today's market, optical mice are among the most popular mice. A small LED or laser emits light to read the grid as the mouse passes over the surface of the mouse pad. Optical mice are found to be more accurate than regular mice. Mouse manufacturers like Logitech and Microsoft have fixed the issue, as the LED or laser can be extremely bright [29].

As early as 1998, to enhance the improvements, the Sun SPARCstation servers and workstations Sun Microsystems came up with a laser mouse. It was the latest computer technology available for mice at that time. The laser inside could deliver up to 20x the performance compare to an equipped optical mouse. Laser mice work perfectly well on any surface without problems. However, laser mice did not enter the mainstream market until 2004 [29].

Over the past decades, the mouse retains its basic shape and function in moving a cursor within a graphical user interface for users to easily and effectively use their computers.

## **2.2 Background and Context**

The mouse typically controls the exact location of a cursor, which is shown on the monitor. The cursor moves in a similar manner, as the mouse is moving. When the mouse is moving, there is a tracking ball that rotates within the mouse. According to the rotation, an electrical signal or pulse is produced from the sensors, and the signal and pulse is feed back to the computer, which causes the cursor to move according. In addition, the switching functions of the mouse, which can be controlled by two or three buttons, are usually used to activate a function or command identified by the cursor location. The switching functions can also be used to control a whole range of different functions and certain software features, such as highlighting in a word processor.

Although the interface (serial, PS2, USB, wireless, Etc) of the mouse to the computer has changed numerous times, the actual mechanics of the mouse has undergone no significant change in 2 decades! The sliding motions of the mouse on a table are translated in sliding motions of the mouse-pointer. However, there are many tasks,

which require the use of both, keyboard and mouse, and many people find it difficult to use both devices at the same time as a hand has to move back and forth between the two devices: the movement requires time and a repositioning of the hand, which implies that the eyes have to be taken off the screen. At the moment, there are several different sensing systems, which are known and used. One of the solutions is to place the tracking ball or the touch panel at the center of the keyboard, which is developed by some computer companies.

Computer games and web TV are also getting very popular among children and adult, in which most games, which often require a cursor control with a mouse. The present invention allow the mouse to be dragged across a surface or having the track ball to be rotated. One disadvantage is that the fast response and comfort is not there. Through research and interviews, many game challengers, suggest that the track ball and switch functions should be located on the hand to be completely free from a desktop or keyboard. This provides a more convenient means of playing a game or surfing the web while sitting in a chair or just lie in the bed in front of the TV.

The new invention will overcome these disadvantages. The tilt sensor, which acts as the tracking ball, is mount on the hand and it is conveniently located and can be operated without removing the fingers from a type position. With the efficient mean of simultaneous cursor control and types, it enables the computer user to complete tasks faster, thus short time is require to produce a piece of work. There is no need to move the hand back and forth between the keyboard and the mouse. Both hands can be kept on the keyboard at all times. The principal advantage of this invention is the utilization of the dexterity of the thumb against the index finger for mouse or track ball operation. This invention can be used with any application requiring a mouse or track ball operation.

### **2.3 Investigation of a Mouse Requirement.**

To design and build this system, there are few issues to consider.

#### **2.3.1 Sample Serial Port Device. How Serial Mouse Works?**

A simple PC mouse controlling system has the following parts:

Sensors and buttons -> Mouse's Micro controller -> RS232 Communication link -  
>Personal computer driver.

LED sensors or ball trackers are commonly used as movement detectors for sensing mouse movement. The mouse is also mounted with button switches for sensing the button states. Mouse controller reads the state of those sensors and buttons and passes this information into the mouse controller. The mouse controller then sends a packet of data to the computer using a RS232 communication standard cable. The mouse driver in the computer received that packet of data and decodes the information from it. Based on the information, the computer moves the cursor accordingly [13].

### 2.3.2 Methods of Interface

#### Microsoft serial mouse protocol and RS232 serial connection.

The voltage, which is required by the RS232, is +12V for high and 0V for low signals, respectively. RS232 idles at +12V. A serial level shifter chip must be used to allow the micro controller to communicate with the computer since the micro controller outputs 5V high and 0V low (TTL level). The computer expects an identifier string (data at 1200 baud, 7N1) from the micro controller. When the USB cable is plugged onto the USB port of the computer, the micro controller must send multiple "M" for the Microsoft serial scroll mouse to be detected. In order to be considered you as identified, the computer will display message, which tell the user that the Microsoft serial mouse had been detected.

DB-9 Pin	IDC internal pin name*	Name	Dir	Description
1	1	CD	←	Carrier Detect
2	3	RXD	←	Receive Data
3	5	TXD	→	Transmit Data
4	7	DTR	→	Data Terminal Ready
5	9	GND	—	System Ground
6	2	DSR	←	Data Set Ready
7	4	RTS	→	Request to Send
8	6	CTS	←	Clear to Send
9	8	RI	←	Ring Indicator

Table 2.1 RS232 serial connections

The packet format is as follows:

Packet 0 - x 1 LB RB X7 Y4 X7 X6

Packet 1 - x 0 Y5 Y4 Y3 Y2 Y1 X0

Packet 2 - x 0 X5 Y4 Y3 Y2 Y1 Y0

LB and RB represents the left and right mouse button status. A binary 1 registered on the micro controller will represent that the button has been pushed and a binary 0 represents not pushed. Y7-Y0 (From most significant to least significant) represents the two's complement mouse delta Y, with positive Y pointing to the bottom of the screen. X7-X0 (From most significant to least significant), which has the same format as Y, represents the two's complement mouse delta X, with positive X pointing to the right of the screen. The first bit in every packet is an extra stop bit; some implementations set it as high or low. Since a genuine Microsoft serial mouse is going to be implemented, the bit has to set it as 1.

#### **2.4 Development of Micro controller**

Since a micro controller controls the Hand Positioned Mouse's main control system, it would be wise to have a deep understanding in the electronics, programming and architecture of the micro-controller technology before a most suitable micro-controller can be select in Chapter 3.

The period between 1970 and 1980 witness a huge growth in microprocessor technology. The computers, which were developed in the 1950's, were quickly replaced with more reliable and smaller Large-Scale Integration chips. The first micro controller that was developed had a 4-bit architecture, which means that the processor could deal with 4-bit width of data. These 4-bit micro controllers were cheap and have fast response. It also consumed low power and could handle just about any data processing application or any control imaginable in those days.

With more complicated designs, there was a need for smaller and low power systems. 8-bit micro-controller, which still dominates the market, was introduced. The typical 8-bit microprocessor has an 8-bit wide data bus and a 16-bit wide address bus. This means that 8 bit microprocessors can directly access  $2^8$  or 256 memory locations. About 55% of all CPUs sold in the world are 8-bit microcontrollers

With today's advanced VLSI technologies, 16, 32 and 64-bit architectures micro controllers were introduced. They are cheaper, faster, consume lower power, and are more powerful. These micro-controllers are usually reserved for supercomputer and minicomputer applications.

Modern micro-controllers still retain this RAM/ROM/IO structure along with data and address buses. This configuration allows the most expandability and allows the highest performance when designing large systems

### 2.4.1 Architecture of Micro controller Board

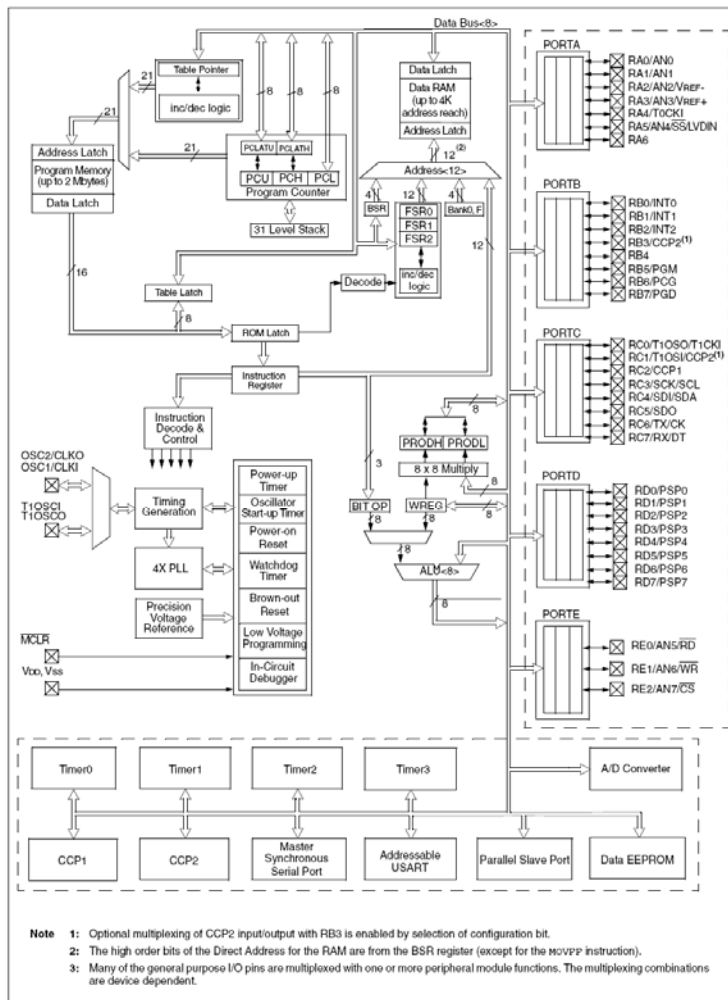


Figure 2.2 PIC18F452 (40/44-PIN) Block Diagram

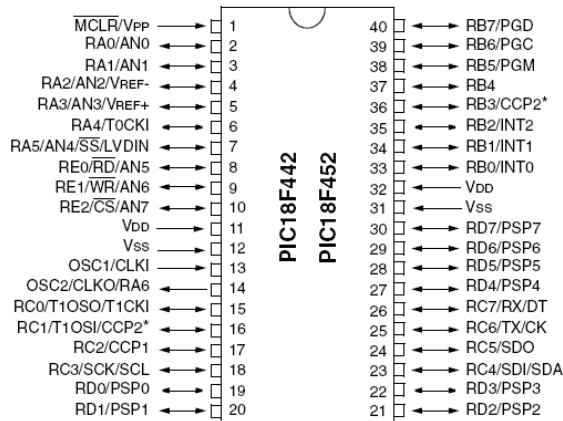


Figure 2.3 PIC18F452 (40/44-PIN) Connection

## 2.4.2 CPU Device Overview

PIC18F452 CPU is a high performance RISC CPU [25]. It has a C compiler optimized architecture/instruction set. It has a linear program memory addressing to 32 Kbytes and a linear data memory addressing to 1.5 Kbytes. It support up to 10 MIPs operation:

- DC - 40 MHz osc./clock input
- 4 MHz - 10 MHz osc./clock input with PLL active

The CPU is based on 16-bit wide instructions, 8-bit wide data path and has priority levels for interrupts.

### It has some peripheral features which includes-

- High current sink/source 25 mA/25 mA
- Three external interrupt pins
- Timer0 module: 8-bit/16-bit timer/counter with 8-bit programmable prescaler
- Timer1 module: 16-bit timer/counter
- Timer2 module: 8-bit timer/counter with 8-bit period register (time-base for PWM)
- Timer3 module: 16-bit timer/counter
- Secondary oscillator clock option - Timer1/Timer3
- Two Capture/Compare/PWM (CCP) modules.
  - I2C™ Master and Slave mode
- Addressable USART module:
  - Supports RS-485 and RS-232

**It has some Analog features which includes-**

- Compatible 10-bit Analog-to-Digital Converter module (A/D) with:
  - Fast sampling rate
- Programmable Low Voltage Detection (PLVD)

Features	PIC18F452
Operating Frequency	DC - 40 MHz
Program Memory (Bytes)	32K
Program Memory (Instructions)	16384
Data Memory (Bytes)	1536
Data EEPROM Memory (Bytes)	256
Interrupt Sources	18
I/O Ports	Ports A, B, C, D, E
Timers	4
Capture/Compare/PWM Modules	2
Serial Communications	MSSP, Addressable USART
Parallel Communications	PSP
10-bit Analog-to-Digital Module	8 input channels
RESETS (and Delays)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)
Programmable Low Voltage Detect	Yes
Programmable Brown-out Reset	Yes
Instruction Set	75 Instructions
Packages	40-pin DIP 44-pin PLCC 44-pin TQFP

Table 2.2 PIC18F452 (40/44-PIN) Features

**2.5 Common Serial Protocols used in Peripheral Device**

Understanding the common integrated serial buses in micro controller plays an important part for implementation of the Hand Positioned Mouse and there are several protocols to be chosen [25]. The common integrated serial buses in micro controllers are

- 1) Universal Asynchronous Receiver Transmission (UART)
- 2) Serial Peripheral Interface (SPI),

- 3) Inter-integrated Circuit (I<sup>2</sup>C),
- 4) Universal Serial Bus (USB),
- 5) Automotive serial bus
- 6) Controller Area Network (CAN).

The main advantage of the serial bus is that when new devices are added onto a serial network, it is easy to implement without affecting the existing devices on the network. Defective device on the network can also be easily traced and replaced without disturbing the network which makes it simpler.

This project utilize on the one serial buses:

### **2.5.1 USART**

The Universal Synchronous Asynchronous Receiver Transmission, which is also known as a Serial Communications Interface or SCI, is a general-purpose serial data bus for Synchronous /Asynchronous communication [27]. The Universal Synchronous Asynchronous Receiver Transmitter (USART) module is one of the two serial I/O modules. The USART can be configured as a full duplex asynchronous system, which can communicate with peripheral devices, such as CRT terminals and personal computers. It can also be configured as a half-duplex synchronous system that can communicate with peripheral devices, such as A/D or D/A integrated circuits, serial EEPROMs, etc.

For transmission at 9600 baud, each bit has a bit time of 1/9600 second. UARTs integrated in micro controllers can transfer data at speeds ranging from a few hundred bits per second up to 1.5Mbps.

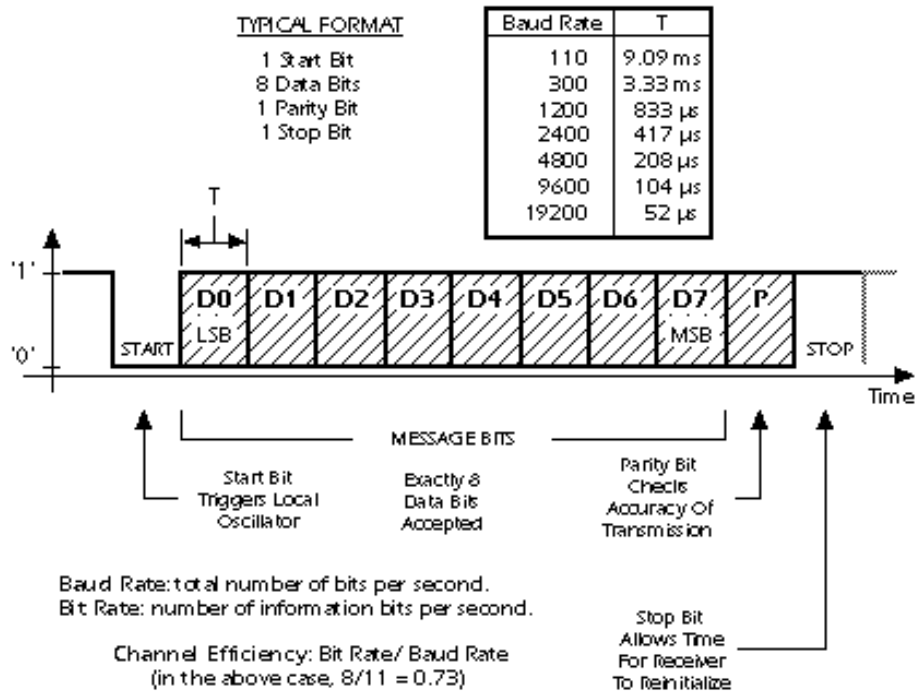


Figure 2.4 UART Data Frame

## Chapter 3 System Design

### 3.1 Hardware

#### Block Diagram

The hardware for the Hand Positioned Mouse is composed of five subsystems:

- 1) 5V DC Power Supply Unit
- 2) Micro-controller
- 3) Button Detection
- 4) Motion Detection
- 5) RS-232 Signal generation

Each of the mentioned subsystems will undergo analysis and design. It should be noted that while these appear independent, they are in fact very intertwined.

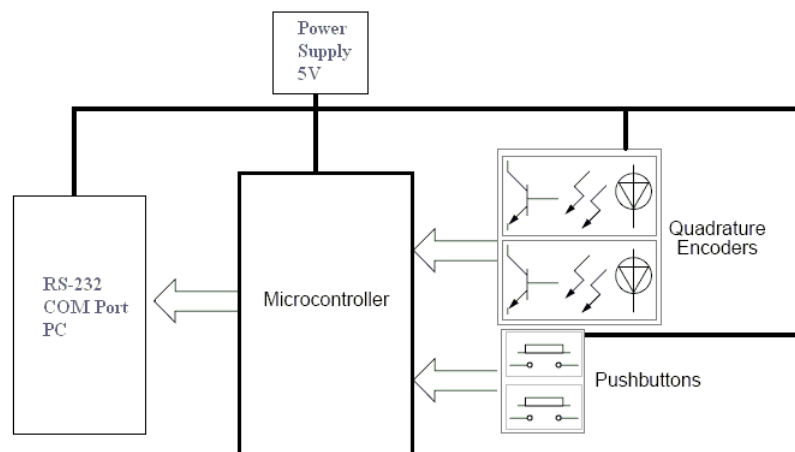


Figure 3.1 Functional Blocks Diagram of Hand Positioned Mouse

#### 3.1.1 5V DC Power Supply Unit

The project will be battery powered operating off a 9V battery. It is because the project's digital circuits require 5V DC; therefore a voltage regulator chip will be used.

The 5V DC Power Supply Unit Circuit for the Hand Positioned Mouse is composed of 4 different components;

- a) L7805CV Voltage Regulator
- b) 0.33uF Capacitor
- c) 0.1uF Capacitor

d) 9V Battery

a) L7805CV Voltage Regulator

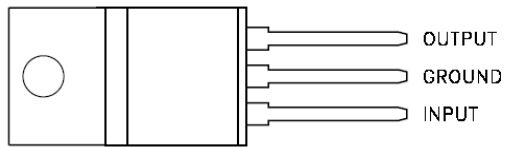


Figure 3.2 L7805CV Voltage Regulator

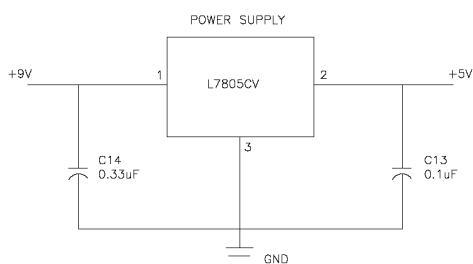


Figure 3.3 L7805CV Voltage Regulator Circuit Connection

A L7805CV voltage regulator provides a constant DC output voltage and contains circuitry that continuously holds the output voltage at the design value regardless of changes in load current or input voltage (this assumes that the load current and input voltage are within the specified operating range for the part) [28].

The LM7805CV fixed voltage regulator series has built-in thermal overload protection, which prevents the device from being damaged due to excessive junction temperature.

The regulators also contain internal short-circuit protection which limits the maximum output current, and safe-area protection for the pass transistor which reduces the short circuit current as the voltage across the pass transistor is increased.

Although the internal power dissipation is automatically limited, the maximum junction temperature of the device must be kept below  $+125^{\circ}\text{C}$  in order to meet data sheet specifications.

b&c) 0.33uF and 0.1uF Capacitor

The capacitor is a circuit element that stores electrical charges. It is used in this circuit to help to keep the voltage regulator's output voltage constant over time. The rate of changes of the voltage across a capacitor is proportional to the current flowing out of

it divided by the capacitance. Therefore, the larger the capacitor's value, the smaller the changes in voltage at the output of the regulator of the regulator over time for a fixed current drain.

#### d) Battery

A normal 9V PP3 battery is used in this project.

### **3.1.2 Selection of a Most Suitable Micro-controller**

When designing a micro-controller system, there are many more considerations than merely the choice of micro-controller.

Micro-controller is commonly used to develop embedded equipments in the fast growing technology world today. It can be very portable, small in size and used for real time applications. Micro-controller, which is chosen, must be able to respond to external trigger in microseconds through interrupts and able to analyse the real time data. Most of the micro controller has features like:

- A/D converter
- Digital I/O
- Pulse Generator
- USB, SPI, I<sup>2</sup>C ports
- External triggers
- RS232 interface
- Input captures
- Flash memory

The Hand Positioned Mouse developed using a micro controller must be very portable.

### 3.1.3 Comparison of Different Micro-controller

No	Features	PIC18F452 Microchip	MC68HC11E2 Motorola	Z8X1621 Zilog
1	Micro controller speed	DC-40MHz	2MHz	20MHz
2	General purpose I/O	32 digital I/O 10 Analog I/O	22 digital I/O 8 Analog I/O	31 digital I/O 8 Analog I/O
3	Timer	4 8/16-bits timers/counters	16-bit, 3-4 IC, 4-5 OC, RTI, pulse accumulator	3 16-bits timer/counters
4	Memory size i) Data memory ii) Program memory	512 bytes SRAM 256bytes EEPROM 4 Kbytes FLASH	256 bytes RAM 2048bytes EEPROM	2 KB RAM 16Kbytes Flash
5	Interrupt Feature i) Edge triggering	falling/rising	Falling/Rising	No.
6	Operating Voltage	2.5V to 5.5V	3.0V to 5.5V	3V to 3.6V
7	MSSP (Serial communication)	Enhanced USART	SCI and SPI	UARTs with IrDA SPI
8	Development tools & kit	MPLAB IDE software	Motorola Modular Development System (MMDS11)	Z8 Encore!
9	Emulator Debugger	MPLAB ICD2	M68EM11E20	ZDS II Integrated Development Environment
10	Evaluation boards	PICDEM 2 board	M68HC11EVBU Universal Evaluation Board	Z8 Encore 64K series Development Kit
11	Local Support	Yes	No	No

Table 3.1 Comparison of 3 types of Micro-controller

No.	Micro-controller	Software support	Ease of implementation	Cost
1	PIC18F452 Microchip	Good	Easy	Average
2	C68HC11E2 Motorola	Average	Good	Above Average
3	Z8X1621 Zilog	Not Available	Good	Average

Table 3.2 Comparison of 3 types of Micro-controller by cost and software support

Other factors under consideration when designing an embedded system include the size of the printed circuit board, multi-voltage power supplies (today's chips use a combination of +5, +3.3, +3, and now even +1.8 volts for power), electromagnetic

Tay Boon Kwong Roland  
Hand Positioned Mouse

E0401574  
07/BEHE/12

interference shielding, FCC certification, etc.

### **3.1.4 Selecting the Microchip PIC18F452 Micro controller**

The Microchip "PIC" is a very popular micro-controller in the hobbyist market. It has less than 33 instructions and only a few registers. It is a true RISC processor.

PICs are used in very small, cost-sensitive applications. They have a number of sleep modes and use very little power during operation. There is an 8 pins version of the PIC for the smallest applications. The most popular PIC micro-controllers are 16 to 28 pins.

The Microchip PIC18F452 Micro controller is selected for this project based on the following:

- Low cost
- Simple
- Ease of implementation
- Availability of development tools for hardware and software.
- High-speed Flash technology
- Lower power consumption
- Digital signal processing capabilities.

### **3.1.5 Sensors Selection**

Sensors give the application the means to perceive its environment. The Hand Positioned Mouse processes the information received from its sensors and reacts in a predetermined manner according to the design of the control system. Now let's look at some of the sensors that can be use on the Hand Positioned Mouse.

#### **1) Mechanical Tilt and Vibration Sensors**

These sensors are designed to consume extreme low power, virtually no power when these sensors are not moving. It generates a digital output when moved. It requires no signal conditioning and are fully passive. It can be used in a triggering circuit that draws as little as 0.25uA of continuous current. The surface-mount tilt sensor provides low cost solutions, reliable, simple for omni directional movement sensing. When activity is sensed, they provide intelligent power management to interrupt and "wake-up" a micro-controller.

#### **2) Inclinometers**

Inclinometers are the smallest and lowest power, fully calibrated, modules found in the electronic industry. These inclinometers are built to combine 2- or 3-axis measurement algorithms with the reliability and performance of silicon MEMS accelerometers to achieve best performance. These sensors are commonly used in applications such as pipeline inspection, vehicle leveling, ROV navigation, satellite positioning, physical therapy and rehabilitation throughout the world.

### 3) Accelerometers

Accelerometers are commonly used in the prototyping phase of a design. These sensors allow easy capture of shock, vibration and tilt to a computer for recording and analysis. These sensors are commonly found in wireless data acquisition, process monitoring and remote sensing applications where low cost, calibrated acceleration measurement is required.

#### 3.1.6 Button Selection

Pushbuttons come in various sizes, ratings and terminations and these buttons allow virtually unlimited configurations possibilities [24]. Some of the examples of the push buttons are as follows:

- 1) Through-Hole Mount Toggles
- 2) Through-Hole Mount Rockers
- 3) Through-Hole Mount Pushbuttons
- 4) Surface Mount Toggles
- 5) Surface Mount Pushbuttons
- 6) G Indicator

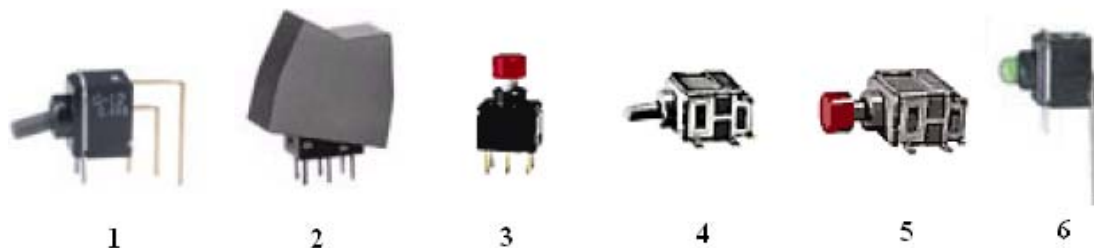


Figure 3.4 Actual looks of the push buttons

There are some optional considerations for the button selections.

- a) Type of poles
  - 1) SPDT
  - 2) DPDT
- b) PC Terminals
  - 1) Straight
  - 2) Right Angle
  - 3) Vertical
- c) Plunger Position
  - 1) Alternate
  - 2) Momentary
- d) Cap colour
  - 1) Black
  - 2) White
  - 3) Red

### 3.1.7 MAX232 TTL TO RS-232 Converter

The MAX232 is an IC chip, which converts TTL signal from the micro-controller to RS232 signal for the purpose of interface with the personal computer. Here are three detailed descriptions of the components, which contributes to the function of the IC chip [27][23].

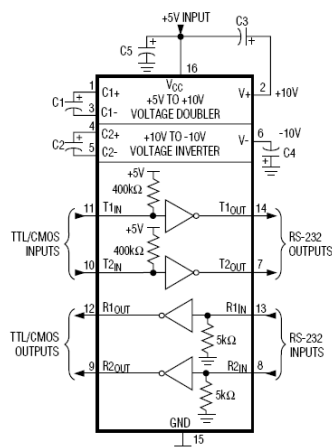


Figure 3.5 MAX232 Pin Configurations and Typical Operating

### **1) Dual Charge-Pump Voltage Converter**

The MAX232 have two internal charge-pumps that convert +5V to  $\pm 10V$  for RS-232 driver operation. As shown in figure 3.5, the first converter uses capacitor C1 to double the +5V input to +10V. The second converter uses capacitor C2 to invert +10V to -10V on C4 at the V- output. The IC chip also allows a small amount of power to be drawn from the +10V (V+) and -10V (V-) outputs to power external circuitry.

### **2) RS-232 Receivers**

EIA/TIA-232E and V.28 specifications define a voltage level greater than 3V as a logic 0, so all receivers invert. Input thresholds are set at 0.8V and 2.4V, so receivers respond to TTL level inputs as well as EIA/TIA-232E and V.28 levels.

### **3) RS-232 Drivers**

Input thresholds are both TTL and CMOS compatible. The driver output voltage swing is  $\pm 8V$  when loaded with a RS-232 receiver signal and  $VCC = +5V$ .

## **3.2 Software Design**

The approach to software design is to develop the software and interface with the Microsoft Serial mouse driver in the computer.

### **3.2.1 Interfacing Micro controller to Personal Computer**

The USART is the key component of the serial communication port on a computer. The RS232 standard defines the logic “1” signal to be between -3 and -25 volts, and the logic “0” signal to be between 3 and 25 volts with respect to ground signal. The range of voltages between -3 volts and +3 volts is considered a transition region for which a signal state is not assigned. Figure 3.6 shows a signal diagram of the RS232.

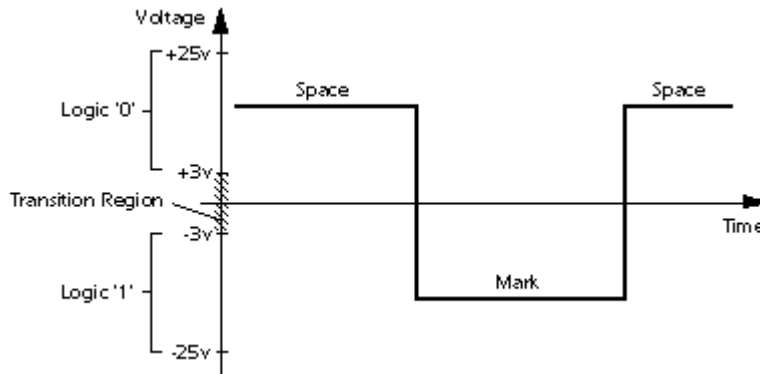


Figure 3.6 Signal Diagram of RS232

It requires a RS232 driver to convert the voltage levels when USART in a micro controller is connected with the PC. The MAX232 driver IC from Maxim is then used for converting the voltage level and for connecting the USART to the serial port of the computer.

### 3.2.2 Software Development Tools

There are three Microchip Development tools[19][20][21][26]. They are:

- a) MPLAB Integrated Development Environment (IDE) Version 7.60
- b) MPLAB ICD 2 In-Circuit Debugger
- c) MPLAB MPASM compiler

Figure 3.7 shows a picture of MPLAB Integrated Development Environment version 7.60. It is a low cost development tool. The software is used for debugging and programming of PIC18 enhanced Flash micro controller (P18F452) in this project. It can also provide real time debugging of source code by setting breakpoints at certain stages in the program. MPLAB ICD 2 is a hardware device, which is used to download the source code into the target micro-controller.

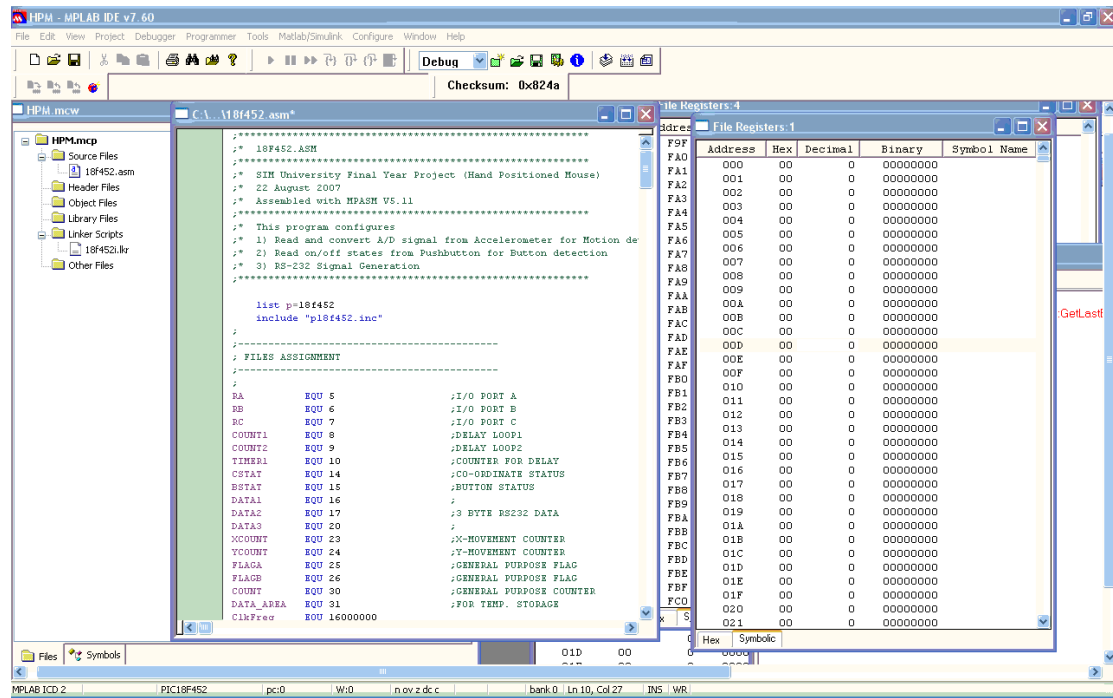


Figure 3.7 MPLAB IDE Integrated Development Environments

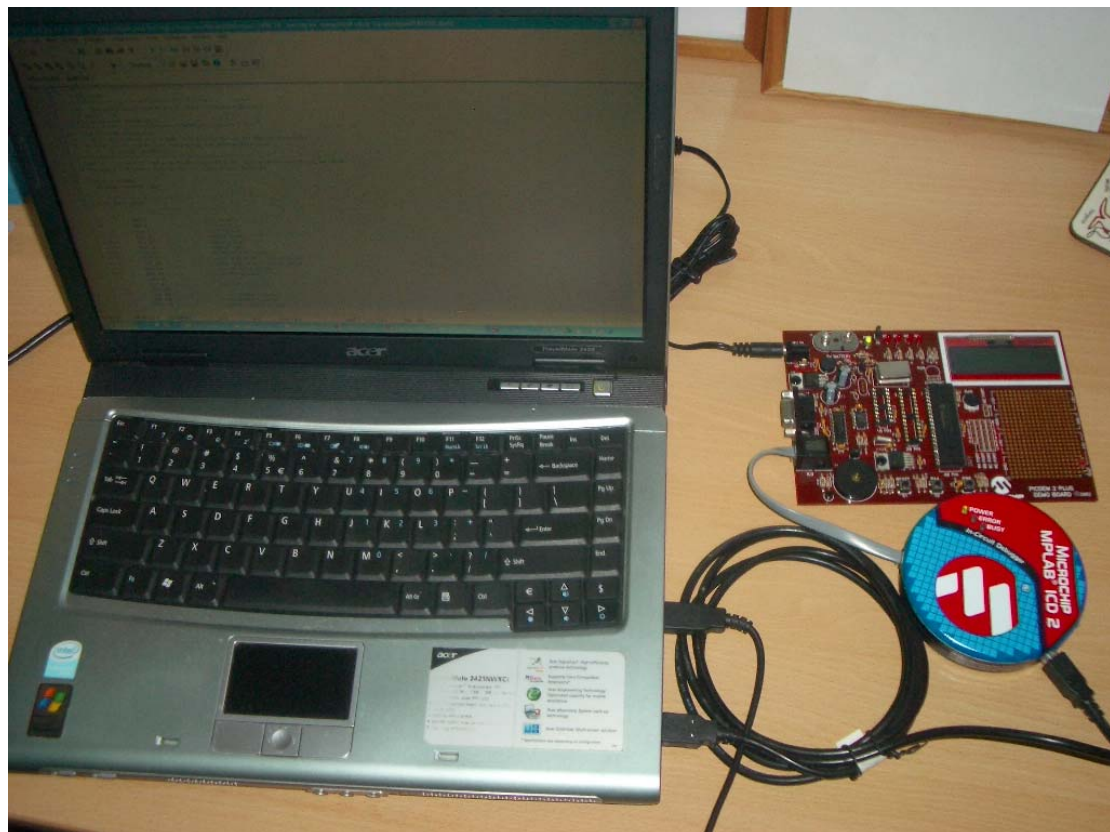


Figure 3.8 Set-up of the ICD 2 connect to a PC via USB cable and to a PIC18 micro controller via a ICD cable.

### 3.2.3 Writing the software

There are quite a few ways to create software for the Micro-controller. The program can easily be created using MPLAB from Microchip or a simple text editor.

The PIC chip does not really care what method was used to write the software because it only understands raw HEX code. After the conversion took place, it is then downloaded into the chip, by using an appropriate programmer.

This is a sample of HEX code.

```
:100009832975279579993250880250250208528050  
:10000000000000JHRHJ32797579595UKJH585898598  
:100029T7349U3IHKJ35H3757395983737843753975  
:1089349U3HJ53J3U5I357395935853485895835839
```

### 3.2.4 Types of Analog to Digital Conversion

These are many ways of implementing an ADC and below are some of ADC structures commonly used.

- 1) Direct conversion/Flash ADC
- 2) A successive approximation ADC
- 3) Ramp-compare ADC
- 4) Delta-encoded ADC
- 5) Pipeline ADC
- 6) Sigma-Delta ADC

#### 1) Direct conversion ADC

Direct conversion ADC or flash ADC uses a comparator, which fires for each decoded voltage range. The comparator bank then feeds a logic circuit that generates a code for each voltage range. Direct conversion usually has only 8 bits of resolution (256 comparators) or fewer and it is very fast. But it has one disadvantage, as it is a large and expensive circuit to build.

#### 2) A successive approximation ADC

A successive approximation ADC uses a comparator by constantly comparing the input voltage to the output of an internal analog to digital converter to reject ranges of voltages, eventually settling on a final voltage range. It will compare until the best approximation is achieved. In this process, a binary value of the approximation is then stored in a successive approximation register, which is used as a reference voltage for comparisons.

### 3) Ramp-compare ADC

A ramp-compare ADC produces a saw-tooth signal that ramps up, then quickly falls to zero. When the ramp signal begins, a timer will start counting. When the ramp voltage matches the input, a comparator fires, and the timer's value is recorded. The advantage of the Ramp-compare ADC is when it is used to compare second signal, it just requires another comparator, and another registers to store the voltage value. A micro-controller together with one resistor and capacitor can be used to implement a very simple ramp-converter.

### 4) Delta-encoded ADC

A delta-encoded ADC used an up-down counter, which feeds an analog to digital converter (ADC). The comparator, which controls the counter, is fed by the input signal and the ADC. The circuit then uses a negative feedback from the comparator to adjust the counter until the ADC's output has little margin to the input signal.

### 5) Pipeline ADC

A pipeline ADC, which is also called sub-ranging quantizer, uses two or more steps of sub-ranging. A coarse conversion is done at first stage. The difference to the input signal is then determined with an analog to digital converter (ADC) in the second stage. At the last step, the results are combined and the difference is then converted to finer results.

### 6) Sigma-Delta ADC

A Sigma-Delta ADC, which is also known as a Delta-Sigma ADC, over-samples the desired signal by a large factor and filters the desired signal band. The resulting signal, which is generated along with the error by the discrete levels of the Flash, is subtracted from the input to the filter and fed back into the system.

With the above ADC, Direct conversion/Flash ADC is the simplest and most suitable for the Hand Positioned Project.

## **Chapter 4 System Implementation and Detailed Design**

### **4.1 Hardware Implementation**

#### **4.1.1 Pin Assignment for the Micro-controller**

##### **Overview**

The Micro Chip CPU I/O pins are illustrated in Figure 4.1 below and the function of each is described in the following paragraphs.

##### **Port A (ADC/Input Port)**

PIN 2 Bit 0: X-axis Motion

PIN 3 Bit 1: Y-axis Motion

##### **Port B (ADC/Input Port)**

PIN 39 Bit 6: Right side Push Button

PIN 40 Bit 7: Left side Push Button

##### **Power Supply**

PIN 11: +5V DC

PIN 32: +5V DC

PIN 12: 0V DC

PIN 31: 0V DC

##### **Oscillator for Timer**

PIN 13 OSC1 Oscillator crystal clock input

PIN 14 OSC2 Oscillator crystal clock output

##### **RS232 Signal to PC COM Port**

PIN 25 TX TO MAX232

PIN 26 RX TO MAX232



### 4.1.2 Accelerometer Hardware

The accelerometers were relatively easy to set up. In this project, one MMA1260D and one MMA2260D were used to detect a negative or positive direction along the located orientation line of the finger. The MMA1260D is used to measure tilt movement along the Y-axis where as the MMA2260D measures X-axis tilt. In order to have a constant and stable signal between Micro-controller inputs and accelerometer outputs, it is recommended that a low pass filter is required. To have a design as simple as possible, other suggestion for testing, which includes connections without low pass filter. [14][27].

### 4.1.3 TTL Signal to RS-232 Signal generation

The main function of the TTL Signal to RS-232 conversion block is to carry out the necessary conversion so that the PIC18F452 can communicate with the PC. Data bytes are sent to MAX232 IC chip and being converted to signals meeting the RS-232 standard, which the PC can understand [23].

The data has the following form:

- Start bit (always low or high)
- Data byte (from Least Significant bit to Most Significant bit)
- Parity bit (high for an even number of high bits in the data byte and low for an odd number)
- Stop bit (always high)

Bit Position	Mouse System Format*								Microsoft Format*							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 1	1	0	0	0	0	L	M	R	1	1	L	R	Y7	Y6	X7	X6
Byte 2	X7	X6	X5	X4	X3	X2	X1	X0	0	0	X5	X4	X3	X2	X1	X0
Byte 3	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	0	0	Y5	Y4	Y3	Y2	Y1	Y0
Byte 4	X7	X6	X5	X4	X3	X2	X1	X0								
Byte 5	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0								

\* L = Left Key Status      1 = Pressed      X7-X0 = X-Axis Movement Data  
M = Middle Key Status    0 = Released      Y7-Y0 = Y-Axis Movement Data  
R = Right Key Status

Table 4.1 Mouse System and MICROSOFT Protocols

## **4.2 Software Implementation**

The project file name, HPM.mcp, is created with Microchip software, MPLAB IDE tool for the Hand Positioned Mouse.

There are two sub-files in the HPM.mcp

- 1) Main source file - 18f452.asm
- 2) Linker script - 18f452.lkr

The main source file, 18f452.asm, contains all the necessary codes for micro-controller and the linker script is reserved for the resources used by the MPLAB ICD 2.

The MPLAB IDE tool can also be used to build a HEX file for debugging and transferring program code into the PIC18f452 IC chip for debugging. When the program is successfully debugged, the PIC18f452 MCU is programmed for stand-alone operation.

## **4.3 Accelerometer Software Design**

The 10-bit Analog to Digital Converter (ADC) of the micro-controller reads a voltage output from the X and Y-axis accelerometer, compares it to a 5V reference, and outputs a 8 bit binary variable. Therefore, if the accelerometers are not moving (reading of 0G), has a 2.5V output, the ADC value would be about 127. The accelerometer with the model number MMA1260D (which measures the tilt along the Y axis) is used for X axis position and the MMA2260D (which measures tilt along the X axis) is used for the Y-axis position. When the hand is tilted right, the MMA1260D reads positive G values, and outputs a voltage. The same theory also applies for the MMA2260D accelerometer [14][15].

## **4.4 Interfacing Design (Software)**

When the USB cable is plugged into the USB's COM port, six repeated sequences of "M" had to be sent to the PC. It is an indication to the PC that a Microsoft serial scroll mouse has been connected [10].

After the mouse has been established with the PC, the right and the left button are sampled on every cycle. When either the right or the left button is pressed, it will pass the information from the micro-controller to the PC and do the necessary actions. (Selection, open of file and close of program...etc)

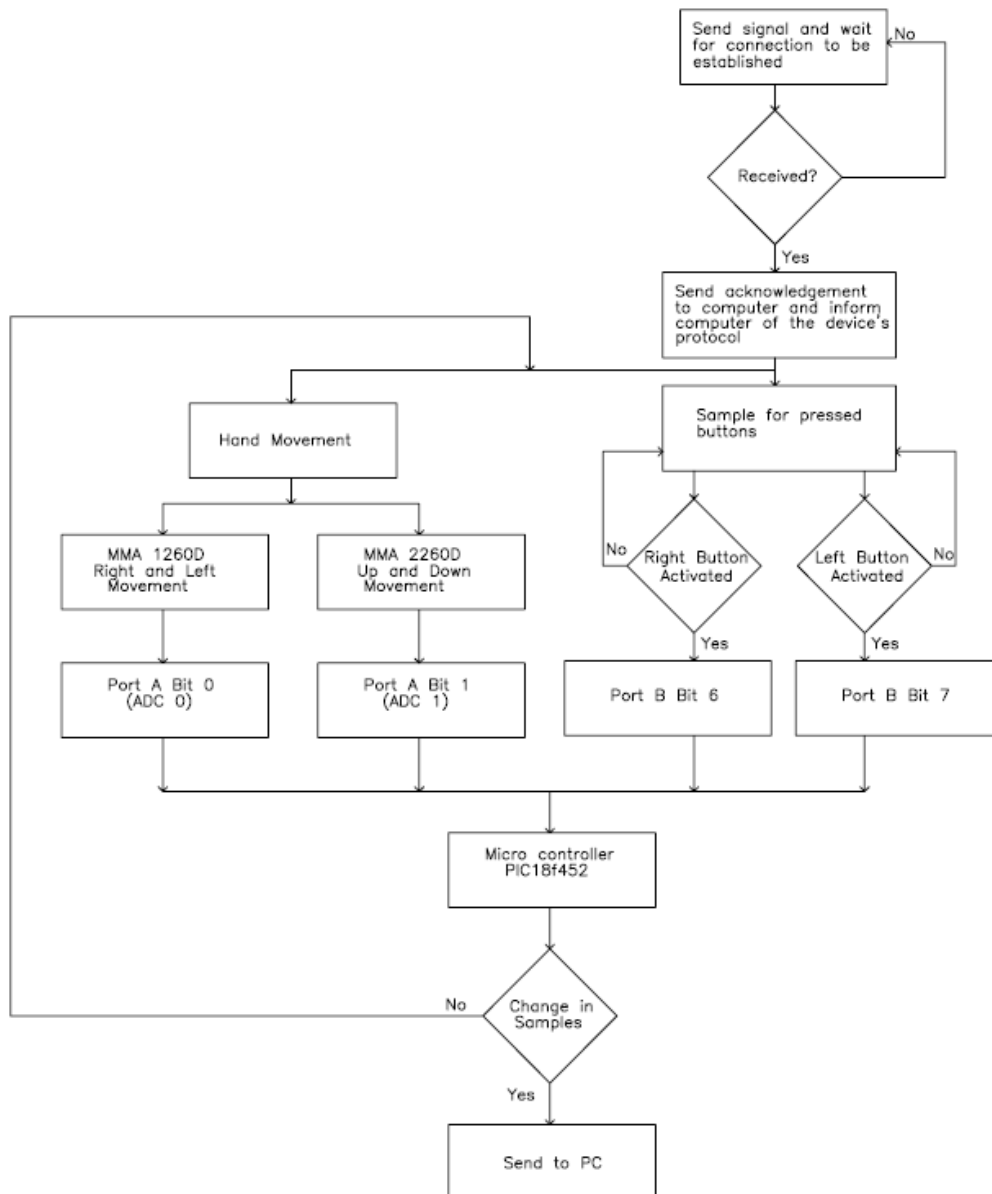


Figure 4.3 Flow Chart of the Hand Positioned Mouse

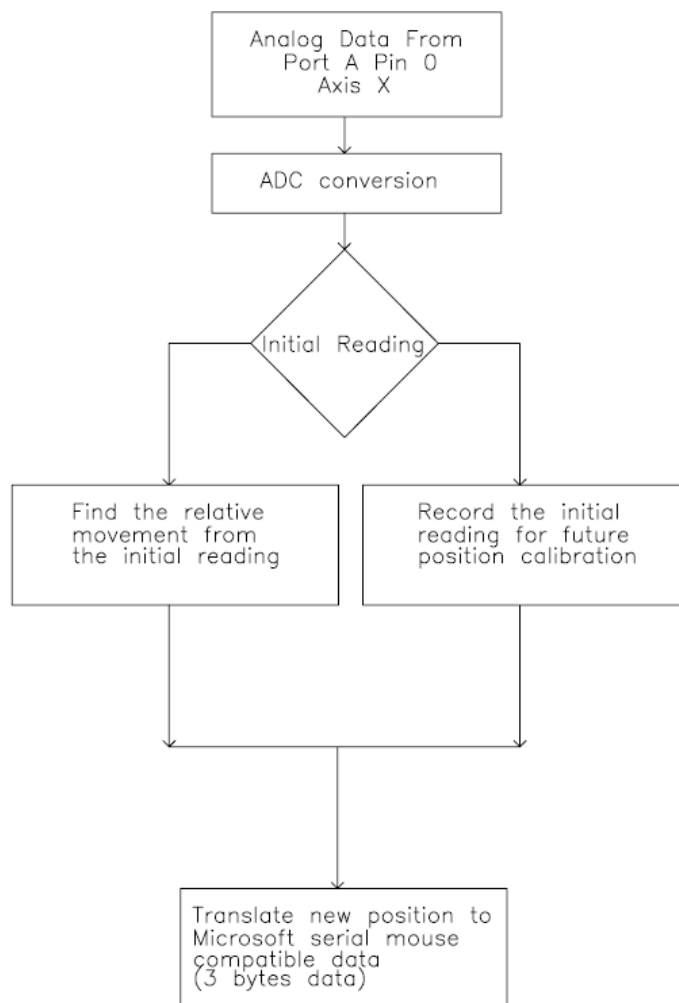


Figure 4.4 Flow Chart of the Hand Positioned Mouse ADC conversion

#### 4.5 Analog to Digital Conversion Sequence (Software)

The following steps should be followed for doing an A/D conversion:

1. Configure the A/D module:

- Configure analog pins, voltage reference and digital I/O (ADCON1)
- Select A/D input channel (ADCON0)
- Select A/D conversion clock (ADCON0)
- Turn on A/D module (ADCON0)

2. Configure A/D interrupt (if desired):

- Clear ADIF bit
- Set ADIE bit
- Set GIE bit

- Set PEIE bit
3. Wait the required acquisition time.
  4. Start conversion:
    - Set GO/DONE bit (ADCON0)
  5. Wait for A/D conversion to complete, by either:
    - Polling for the GO/DONE bit to be cleared (interrupts disabled)OR
    - Waiting for the A/D interrupts.
  6. Read A/D Result registers (ADRESH/ADRESL); clear bit ADIF if required.
  7. For next conversion, go to step 1 or step 2 as required. The A/D conversion time per bit is defined as TAD. A minimum wait of 2 TAD is required.

#### **4.6 X-Axis Movement Algorithm**

The movement counters are 8-bit 2's complement integers, where the most significant bit appears as a sign bit in bit X7 of the movement data packet.

X7 = 0 → represent the negative value

X7 = 1 → represent the positive value

X6-X0 will represent the data for relative movement.

If the X is positive, it implies that the mouse is moving to the right. If the X is negative, it implies a movement to the left.

These counters are always updated when the mouse reads its input.

NEGATIVE X DIRECTION									
SEQUENCE	BIT								
	X7	X6	X5	X4	X3	X2	X1	X0	
1	0	0	0	0	1	0	0	0	8
2	0	0	0	1	0	0	0	0	16
3	0	0	0	1	1	0	0	0	24
4	0	0	1	0	0	0	0	0	32
5	0	0	1	0	1	0	0	0	40
6	0	0	1	1	0	0	0	0	48
7	0	0	1	1	1	0	0	0	56
8	0	1	0	0	0	0	0	0	64
9	0	1	0	0	1	0	0	0	72
10	0	1	0	1	0	0	0	0	80
11	0	1	0	1	1	0	0	0	88
12	0	1	1	0	0	0	0	0	96
13	0	1	1	0	1	0	0	0	104
14	0	1	1	1	0	0	0	0	112
15	0	1	1	1	1	0	0	0	120
16	0	1	1	1	1	1	1	1	128

Table 4.2 NEGATIVE X-Axis Movement Algorithms

POSITIVE X DIRECTION									
SEQUENCE	BIT								
	X7	X6	X5	X4	X3	X2	X1	X0	
1	1	0	0	0	1	0	0	0	8
2	1	0	0	1	0	0	0	0	16
3	1	0	0	1	1	0	0	0	24
4	1	0	1	0	0	0	0	0	32
5	1	0	1	0	1	0	0	0	40
6	1	0	1	1	0	0	0	0	48
7	1	0	1	1	1	0	0	0	56
8	1	1	0	0	0	0	0	0	64
9	1	1	0	0	1	0	0	0	72
10	1	1	0	1	0	0	0	0	80
11	1	1	0	1	1	0	0	0	88
12	1	1	1	0	0	0	0	0	96
13	1	1	1	0	1	0	0	0	104
14	1	1	1	1	0	0	0	0	112
15	1	1	1	1	1	0	0	0	120
16	1	1	1	1	1	1	1	1	128

Table 4.3 POSITIVE X-Axis Movement Algorithms

#### 4.7 Y-Axis Movement Algorithm

Similarly for the Y-Axis Movement Algorithm, the movement counters are also 8-bit 2's complement integers, where the most significant bit appears as a sign bit in bit Y7 of the movement data packet.

Y7 = 0 → represent the negative value

Y7 = 1 → represent the positive value

Y6-Y0 will represent the data for relative movement.

If the Y is positive, it implies that the mouse is moving down. If the Y is negative, it implies then mouse is moving up.

NEGATIVE Y DIRECTION									
SEQUENCE	BIT								
	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	
1	0	0	0	0	1	0	0	0	8
2	0	0	0	1	0	0	0	0	16
3	0	0	0	1	1	0	0	0	24
4	0	0	1	0	0	0	0	0	32
5	0	0	1	0	1	0	0	0	40
6	0	0	1	1	0	0	0	0	48
7	0	0	1	1	1	0	0	0	56
8	0	1	0	0	0	0	0	0	64
9	0	1	0	0	1	0	0	0	72
10	0	1	0	1	0	0	0	0	80
11	0	1	0	1	1	0	0	0	88
12	0	1	1	0	0	0	0	0	96
13	0	1	1	0	1	0	0	0	104
14	0	1	1	1	0	0	0	0	112
15	0	1	1	1	1	0	0	0	120
16	0	1	1	1	1	1	1	1	128

Table 4.4 NEGATIVE Y-Axis Movement Algorithms

POSITIVE Y DIRECTION									
SEQUENCE	BIT								
	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	
1	1	0	0	0	1	0	0	0	8
2	1	0	0	1	0	0	0	0	16
3	1	0	0	1	1	0	0	0	24
4	1	0	1	0	0	0	0	0	32
5	1	0	1	0	1	0	0	0	40
6	1	0	1	1	0	0	0	0	48
7	1	0	1	1	1	0	0	0	56
8	1	1	0	0	0	0	0	0	64
9	1	1	0	0	1	0	0	0	72
10	1	1	0	1	0	0	0	0	80
11	1	1	0	1	1	0	0	0	88
12	1	1	1	0	0	0	0	0	96
13	1	1	1	0	1	0	0	0	104
14	1	1	1	1	0	0	0	0	112
15	1	1	1	1	1	0	0	0	120
16	1	1	1	1	1	1	1	1	128

Table 4.5 POSITIVE Y-Axis Movement Algorithms

## Chapter 5 Testing and Results

### 5.1 Types of Testing

Testing is an essential process and is concerned with the verification that the program is working properly. Based on the requirements in the document, many attempts are done to test every components or modules that the software is composed of. There are several ways to test a piece of software. The most common are the following:

- a) Black Box Testing
- b) White Box Testing



Figure 5.1 Picture of Hand Positioned Mouse

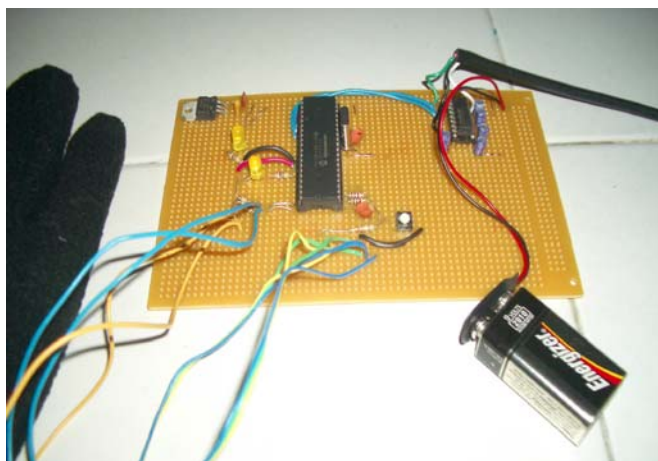


Figure 5.2 Close Up on the Circuitry



Figure 5.3 Close Up on the Sensor and Buttons

**Black Box Testing** –It is also known as testing the functional requirements of the system. This type of testing is the same as input and output verification. To ensure that the component is working correctly, the system should provide the user with the expected output. In this type of testing, the full assemble code is not tested.

**White Box Testing** – This type of testing is based on a small portion of the program that needs to be tested. This time, the full assemble code will be fully tested. The results are documented and need to be studied in greater details when the testing has been completed. The outcome of such study will reveal if the software, described in the requirements, behaves in a correct way. The report must state

- 1) What was being tested?
- 2) Were the requirements were met?
- 3) Was the testing successful and was there any error revealed?

A system can be tested at several levels.

- 1) A routine can be implemented in the program to verify whether it is in a working condition. This is done by creating test data.
- 2) The system integration process begins when all the components are implemented, checked and put together to form the final system. When this process is completed, the whole system is checked again and again to make sure there is no 'bug' in the software. In this way, interface problems may be discovered, as well as other types of errors.

## 5.2 Initial Test Strategy Used

The system was mounted on the breadboard for initial test. The tests that were used were generated from the document where the requirements of the system were.

### 1) Voltage Regulator

For testing the voltage system, Black box testing was used. The voltage regulator chosen should be able to regulate a 9V input to  $\pm 5V$ . But in the initial test result recorded, the voltage obtained from the regulator was about 4.62V. With a quick check on the manual, the voltage applied should be proportional to the voltage output. To get a constant output of 5V, the input of 9V should be achieved. With the statement, three brands of 9V battery were tested.

- 1) Vinnic output – 8.11V (Cheapest in cost)
- 2) Eveready output- 9.12V (Normal)
- 3) Energizer output- 9.02V (Expensive)

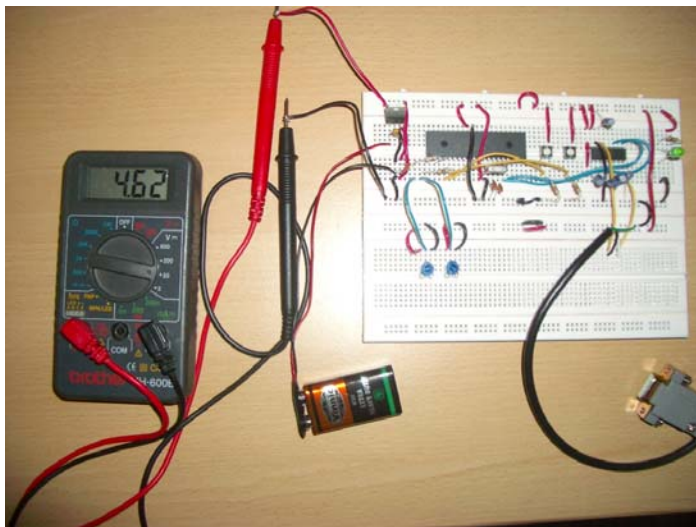


Figure 5.4 Voltage Testing on Vinnic Branded Battery

With the readings obtained from the multi-meter, Energizer branded battery was chosen as it provided a stable voltage of 9V. The whole system was provided with a constant of 4.93V and the voltage met the requirements of the micro-controller and signal converter.

### 2) Input Signal Capturing

There were two buttons on the circuit that needed to be tested. For testing the voltage system, Black box testing was used. The two buttons were connected to the 4.7K $\Omega$

pull up resistors. The voltage of 5V was supplied into the input port of the micro-controller when the button was not pressed. The voltage of 0V was supplied into the input port of the micro-controller when the button was pressed. This type of connection had one advantage, the micro-controller was always fed with a high value, so any loose connection in the circuit, would register a low value when the breadboard was moved from place to place.

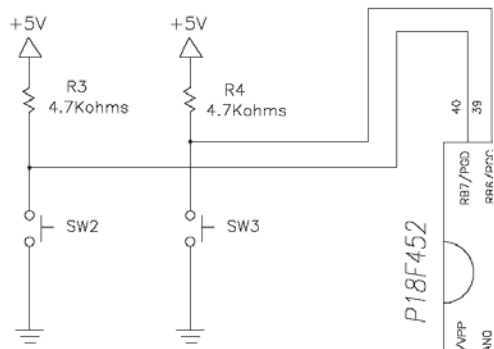


Figure 5.5 Connection of the Two Push Buttons

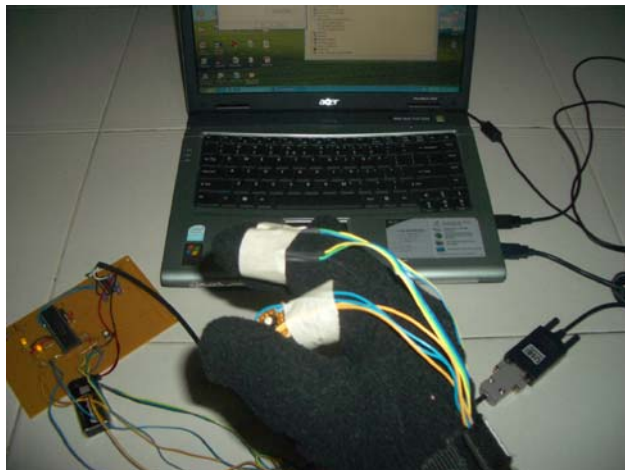


Figure 5.6 Testing with the Buttons

### 3) Interfacing signal with the PC.

The signal from the micro-controller has to be converted and send to the PC. For testing the communication system, White box testing was used. This was an important step because if communication was not established, the whole system would not work.

According to Microsoft document, if a Microsoft Serial mouse was to be detected, the micro-controller had to send a series of 'M' to the PC.

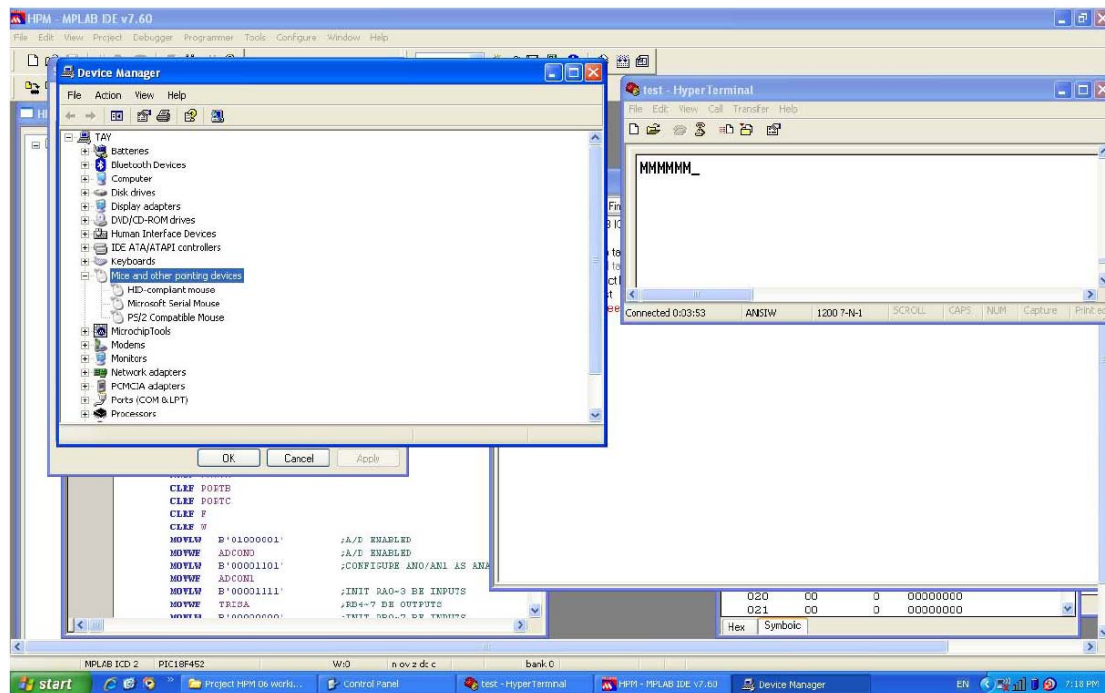


Figure 5.7 Sending 'M' to PC

Six 'M', with the interval of 1 second, were sent to the PC. Upon successful detection, Microsoft serial Mouse would be ready for usage. The status of the established Mouse could be monitored through Device Manager on the PC.

### 5.3 Timing

Timing issues posed some problems when writing many portions of the code. Since there were no available information from the Microsoft website or book, the timing had to be on a 'trial and error' method. After many days of testing, it was found that during initialisation with the PC, an interval of one second between each signal sent provide the best results. During sending of data, a timing of 0.1 second was recommended.

### 5.4 Analog to Digital Conversion Results

The software implementation for Analog to Digital Conversion took up most of the time for testing. By using direct conversion/Flash ADC theory and using potentiometer for testing on the breadboard, the conversion of the voltage to twos complement number could be achieved. But I had problems with the accelerometer readings. The stable reading was not able to be achieved due to the reference for the

angle of the Earth's gravitational force and the sensitivity of the mouse movement. If the hardware was moving slowly, it was fine. But when it came to rapid movement, the hardware was not fast enough to send the response to the input.

## Chapter 6 Conclusion

### 6.1 System Limitation

When a voltage of 5V is applied to the micro-controller, the program will start to run on its own. However, as it is difficult to synchronise micro-controller with the PC when the USB cable is plugged into the USB COM port, a reset button was built to reset the program. When the USB cable is attached to the COM port, the reset button must be pressed to make sure that the program starts from the beginning and send multiple 'M' to the PC for it to identify the mouse.

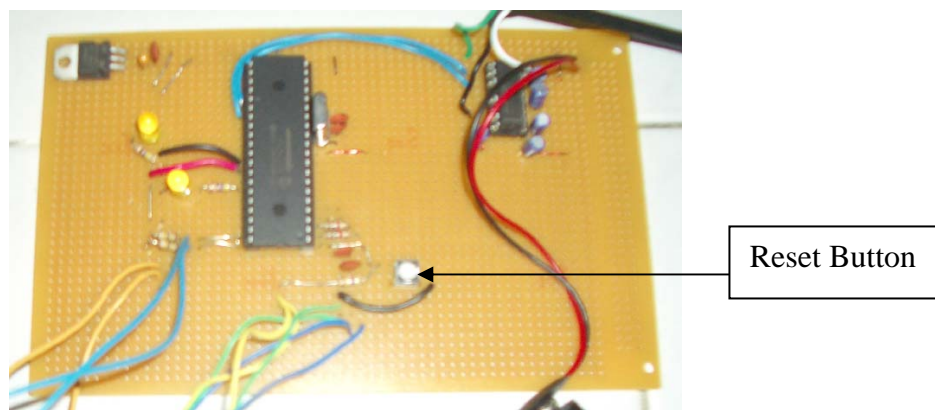


Figure 6.1 Reset Button

### 6.2 Conclusion

The development of the mouse's hardware is successfully assembled and completed. But the result of this project was not as good as expected. The implementation for the movement of the cursor did not really follow the path that I had expect and planned for. This is due to the lack of understanding on the accelerometer; the mouse would make jagged movements. At certain times, the speed of the mouse changes randomly because of speedy changes in velocity and other angle of glove's movements. Due to many unexpected problems arise, a lot of the intended functionality had to be cut from the original design. Despite of the many changes made, a working hardware was finally achieved, which satisfied two out of the three original objectives.

- 1) The mouse is able to interface and detect by the computer.
- 2) The two buttons are able to perform to their expected functions.
- 3) The movement of the cursor would make jagged movements.

Many problems had already been considered during the design phase, meaning problems that occurred were dealt with quickly and effectively. The Hand Positioned Mouse is certainly an accomplishment for me after 8 months of hard work. I am quite familiar with motion control using accelerometers, electrical systems, microcontrollers, and Microchip software on the machine-code level or higher levels. In addition, I have gained knowledge with budgeting, project management and systems engineering. I have also learned about every aspect of design. Hand Positioned Mouse has been the most rewarding and beneficial undertaking of my engineering career.

### **6.3 Recommendation for Future Work**

My plan for the Hand Positioned Mouse is simple: To make the movement of the mouse more stable. Of course, to accomplish this goal, much work remains to be done. The full set of mouse movement algorithms would grant the hardware more power than it currently possesses. Three or more buttons can be added into the design to allow for more functions, something that is missing from the current implementation. A new complex printed circuit board can be designed. Before releasing the final drafts of the boards, the design can be printed on paper or cardboard with header strips and connectors attached. This is to ensure that the components fit together mechanically. Headers can be forced through paper, or they can be placed into cut or drilled holes in cardboard. Cables can then be affixed directly to the headers and held in place on the boards while the designer can assemble everything together. In this way, electrical connections can be verified, mechanical spacing can also be checked, and the board layout can still be adjusted readily.

Other optional modifications to the Mouse's design include:

- The installation of a more sophisticated microprocessor.
- A smaller power circuit.

## Chapter 7 Critical Review and Reflections

### 7.1 Problem Encountered

There have been many problem encountered during the project such as eccentricities of the hardware, the unreliability of the sensor readings and compatibility problems. But some of problems encountered are easily solved and other remains unexplained.

#### 1. During the Design stage

During the design stage, a lot of time was spent on getting and gathering ideas by Searching of articles and journals from website and reading of books. It was a painful experience. There were a lot of different sensors and micro-controllers to choose from. After getting the layout drawn, each detail of design was ready to be implemented. But on some occasions, some components were not available anymore, so the design has to change numerous times until the components are within specifications.

#### 2. During the initial stage (using breadboard)

During the testing of the hardware on the breadboard, there are several occasions that the personnel computer failed to receive signals from the micro-controller. It is not due to the unreliability of the hardware but there are some reasons behind it. Firstly, the components are mounted onto the breadboard and are link up with insecure wires. Most of the time, when the breadboard moves from place to place, the insecure wires will become loose. This will lead to the lost in signal between the PC and the hardware. Secondly, the uncertainty in the battery life is the most common reason for failure. To solve the problem, the voltage source is connected parallel with a LED to make it light. A resistor is connected in series with the LED to limit the current at a safe level because in most LED's the maximum current is about 20mA. As the LED grow dimmer, I will know the battery is running low.

#### 3. During the implementation stage (using printed circuit board)

During the testing of the hardware on the printed circuit board, there are several occasions that the personnel computer failed to receive signals from the micro-

controller. The most common reason is because the soldering of the components onto the printed circuit board is not up to standard. Some of the connections have huge solder on it, which causes the links to 'short-circuit'. If the short-circuit is between VDD and GND, the components will be damaged, so extra precautions are needed when doing soldering. After spending some time on soldering, I manage to get the job done without damaging any of the components.

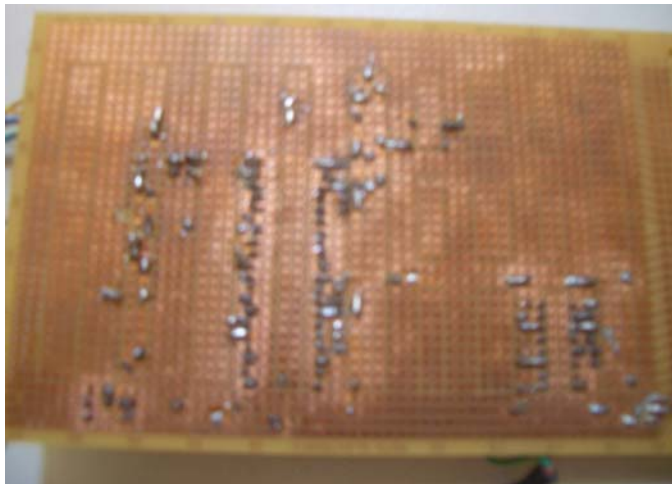


Figure 7.1 Soldering on the Printed Circuit Board

## 7.2 Skill Review

I have achieved the skills of getting and gathering ideas by searching of articles and journals from website, reading of books and project guidance and consultation from tutor. Some of the knowledge acquired through UniSIM courses are useful, for example, MTZS262 Putting computer systems to work, has prepared me in the understanding programming easily. I have also mastered the skill in testing, measurement, troubleshooting and modification on the hardware and software for Hand Positioned Mouse. Skills of assessing and evaluating project progress through setting of targets and deadlines are also achieved.

## 7.3 Reflection

I strongly believe that I have managed my time reasonably well and have achieved an understanding of many aspects of project management. In particular, I had encountered the difficulty of estimating the time, which is required to complete each task assigned. To my stroke of luck, the code libraries which is available from the

Microchip enabled me to concentrate on the development of the Hand Positioned Mouse's main program. In terms of system design, the approach was adopted to allow gradual development of system components and their implementation in a concise manner. The hardware design included schematic drawing; part selection and circuit layout and the programming included the learning of the Machine/Assembly language and the use of compilers for micro controllers. All these skills were not touched upon in regular classes are taken here. At the beginning of this project, I had little understanding on Microsoft computer mouse, and knew nothing about programming in microchip's assemble language although I am a programmer. At the end of the project, software code has been produced and compiled without errors or warnings. I have experienced a steep and a long learning curve throughout the development process as it was very disheartening at the start when most of the design items did not perform. To ensure that the system can perform up to expectation, measurement will have to be carried out. But I believe my gain from this research process has been immense. I think I have built a solid foundation for further improvement. I also learnt a lot about system development and researching during the project. Besides, I have to ensure that building this project will not have any similarities or duplications of other mouse systems used.

Overall I think the time spent has proved to be a valuable educational experience.

## References

- [1] Adam Chapweske, The PS/2 Mouse Interface, <http://www.computer-engineering.org/ps2mouse/> (Current Oct14, 2007)
- [2] Aseem kohli and Karthik, The Accelerometer Based Mouse, Cornell University, 2005
- [3] Bogdan Solca, Of Mice and Men... and PCs, <http://news.softpedia.com/news/Of-Mice-and-Men-and-PCs-43129.shtml> (Current Oct14, 2007)
- [4] Christopher W. Kung and Peter Wang, Gmouse, Cornell University, 2002
- [5] Craig Peacock, Interfacing the Serial/RS-232 port, <http://www.beyondlogic.org/serial/serial1.htm#40> (Current Oct14, 2007)
- [6] David Wu and Eric Lee, 3D gForce Mouse, Cornell University, 2004
- [7] John Becher, PIC16F87x Mini Tutorial, <http://www.epemag.wimborne.co.uk/pictutorial.pdf> (Current Oct14, 2007)
- [8] Mark Sullivan, PIC12C671 Mouse, <http://www.nalanda.nitc.ac.in/industry/datasheets/microchip/Mchipweb/Download/Appnote/Category/12CXXX/Masters/sCircuit/MP075.pdf> (Current Oct14, 2007)
- [9] Marshall Brain and Carmen Carmack, How Computer Mice work, <http://computer.howstuffworks.com/mouse.htm> (Current Oct14, 2007)
- [10] Petr Simandl, PC Mouse, <http://www.simandl.cz/stranky/elektro/mouse/mouse.htm> (Current Oct14, 2007)
- [11] Salvatore Isaja, Serial Mouse Driver, <http://freedos-32.sourceforge.net/showdoc.php?page=sermouse> (Current Oct14, 2007)
- [12] S.J. Katzen, "The Quintessential PIC Micro-controller", Assembly Language Code Building Tools, Ch 8 pg213-246, Springer second edition, 2003.
- [13] Tomi Engdahl, PC serial port (RS-232 DE9) pinout, [http://pinouts.ru/SerialPorts/Serial9\\_pinout.shtml](http://pinouts.ru/SerialPorts/Serial9_pinout.shtml) (Current Oct14, 2007)
- [14] Freescale Semiconductor,  $\pm 1.5g$  X-Axis Micromachined Accelerometer, [http://www.freescale.com/files/sensors/doc/data\\_sheet/MMA2260D.pdf](http://www.freescale.com/files/sensors/doc/data_sheet/MMA2260D.pdf) (Current Oct14, 2007)
- [15] Freescale Semiconductor, Low G Micromachined Accelerometer, [http://www.freescale.com/files/sensors/doc/data\\_sheet/MMA1260D.pdf](http://www.freescale.com/files/sensors/doc/data_sheet/MMA1260D.pdf) (Current Oct14, 2007)

- [16] Microchip, Serial Port Utilities, Application Note, AN-547,  
<http://ww1.microchip.com/downloads/en/AppNotes/00547c.pdf> (Current Oct14, 2007)
- [17] Microchip, Crystal Oscillator Basics and crystal Selection for rfPIC™ and PICmicro® Device, Application Note, AN-826,  
<http://ww1.microchip.com/downloads/en/AppNotes/00826a.pdf> (Current Oct14, 2007)
- [18] Microchip, Implementing a Simple Serial Mouse Controller, Application Note, AN-519, <http://ww1.microchip.com/downloads/en/AppNotes/00519c.pdf> (Current Oct14, 2007)
- [19] MPLAB IDE, User's Guide, Microchip Technology Incorporated, 2006.
- [20] MPLAB IDE, Quick Start Guide, Microchip Technology Incorporated, 2006.
- [21] MPLAB ICD 2 In-Circuit Debugger User's Guide, Microchip Technology Incorporated, 2005.
- [22] National Inventors Hall of Fame, Hall of Fame/inventor profile-Douglas Engelbart, [http://www.invent.org/hall\\_of\\_fame/53.html](http://www.invent.org/hall_of_fame/53.html) (Current Oct14, 2007)
- [23] MAXIM, +5V+Powered, Multichannel RS-232 Drivers/Receivers,  
<http://pdfserv.maxim-ic.com/en/ds/MAX220-MAX249.pdf> (Current Oct14, 2007)
- [24] NKK Switches, G3B (Ultra-Mini Push Button Switches),  
<http://www.nkkswitches.com/switchcategory.asp?S3=3> (Current Oct14, 2007)
- [25] PIC18FXX2 Data Sheet, High Performance, Enhanced Flash Micro-controllers with 10-Bit A/D, Microchip Technology Incorporated, 2006.
- [26] PICDEM™ 2 Plus Demonstration Board User's Guide, Microchip Technology Incorporated, 2006.
- [27] Sparkfun Electronics, Beginning Embedded Electronics, Lecture 4- UART and serial Communication, <http://www.sparkfun.com/commerce/present.php?BEE-4-UART> (Current Oct14, 2007)
- [28] ST Microelectronics, Positive Voltage Regulators (L7800 Series),  
<http://www.ece.unb.ca/Courses/EE4333/CPD/DataSheets/L7805.pdf> (Current Oct14, 2007)
- [29] Wikipedia, Mouse (computing), [http://en.wikipedia.org/wiki/Computer\\_mouse](http://en.wikipedia.org/wiki/Computer_mouse) (Current Oct14, 2007)

## Appendix

### Appendix I (Gantt Chart) Project Schedule

Project Activities	Dur	Timeline																																					
		Feb				Mar				Apr				May				Jun				Jul				Aug				Sep				Oct					
Starting day of the Week		5	12	19	26	5	12	19	26	2	9	16	23	30	7	14	21	28	4	11	18	25	2	9	16	23	30	8	15	22	29	6	13	20	27	4	11	18	25
<b>Literature review</b>																																							
Reading, Library session, Internet search,	10 weeks	[Blue bar from Feb 5 to Apr 16]																																					
<b>WorkShop</b>																																							
Literature Review and Project Methods	1 day	[Blue bar at Mar 19]																																					
Design strategies and Report writing skills	1 day	[Blue bar at Apr 2]																																					
Labview and National Instrument Applications	1 day	[Blue bar at Apr 9]																																					
Introduction to MATLAB Applications for FYP	1 day	[Blue bar at Apr 16]																																					
Introduction to microcontroller design for FYP	1 day	[Blue bar at Apr 23]																																					
Briefing on submission of Final Report and Oral Presentation needs	1 day	[Blue bar at Sep 6]																																					
<b>Define Hardware specification</b>																																							
Technical discussion Tutor	4 weeks	[Blue bar from May 7 to Jun 4]																																					
Sourcing for technical material	10 weeks	[Blue bar from May 14 to Jul 16]																																					
<b>Hardware design and development</b>																																							
Designing of hardware according to test specification	4 weeks	[Blue bar from Jun 11 to Jun 25]																																					
Drawing of hardware schematic layout using OrCAD	4 weeks	[Blue bar from Jun 18 to Jul 2]																																					
Purchase parts	4 weeks	[Blue bar from Jun 25 to Jul 9]																																					



58	:	00111010	3A	116	t	01110100	74
59	;	00111011	3B	117	u	01110101	75
60	<	00111100	3C	118	v	01110110	76
61	=	00111101	3D	119	w	01110111	77
62	>	00111110	3E	120	x	01111000	78
63	?	00111111	3F	121	y	01111001	79
64	@	01000000	40	122	z	01111010	7A
65	A	01000001	41	123	{	01111011	7B
66	B	01000010	42	124		01111100	7C
67	C	01000011	43	125	}	01111101	7D
68	D	01000100	44	126	~	01111110	7E
69	E	01000101	45	127	•	01111111	7F
70	F	01000110	46	128	•	10000000	80
71	G	01000111	47	129	•	10000001	81
72	H	01001000	48	130	•	10000010	82
73	I	01001001	49	131	•	10000011	83
74	J	01001010	4A	132	•	10000100	84
75	K	01001011	4B	133	•	10000101	85
76	L	01001100	4C	134	•	10000110	86
77	M	01001101	4D	135	•	10000111	87
78	N	01001110	4E	136	•	10001000	88
79	O	01001111	4F	137	•	10001001	89
80	P	01010000	50	138	•	10001010	8A
81	Q	01010001	51	139	•	10001011	8B
82	R	01010010	52	140	•	10001100	8C
83	S	01010011	53	141	•	10001101	8D
84	T	01010100	54	142	•	10001110	8E
85	U	01010101	55	143	•	10001111	8F
86	V	01010110	56	144	•	10010000	90
87	W	01010111	57	145	•	10010001	91
88	X	01011000	58	146	•	10010010	92
89	Y	01011001	59	147	•	10010011	93
148	•	10010100	94	202	Ê	11001010	CA
149	•	10010101	95	203	Ë	11001011	CB
150	•	10010110	96	204	Ì	11001100	CC
151	•	10010111	97	205	Í	11001101	CD
152	•	10011000	98	206	Î	11001110	CE
153	•	10011001	99	207	Ï	11001111	CF
154	•	10011010	9A	208	Ð	11010000	D0
155	•	10011011	9B	209	Ñ	11010001	D1
156	•	10011100	9C	210	Ò	11010010	D2
157	•	10011101	9D	211	Ó	11010011	D3
158	•	10011110	9E	212	Ô	11010100	D4
159	•	10011111	9F	213	Õ	11010101	D5
160		10100000	A0	214	Ö	11010110	D6
161	ı	10100001	A1	215	×	11010111	D7
162	ø	10100010	A2	216	∅	11011000	D8
163	£	10100011	A3	217	Ù	11011001	D9
164	¤	10100100	A4	218	Ú	11011010	DA
165	¥	10100101	A5	219	Û	11011011	DB
166	ı	10100110	A6	220	Ü	11011100	DC
167	§	10100111	A7	221	Ý	11011101	DD
168	¨	10101000	A8	222	Þ	11011110	DE
169	©	10101001	A9	223	ß	11011111	DF
170	ª	10101010	AA	224	à	11100000	E0

171	«	10101011	AB	225	á	11100001	E1
172	¬	10101100	AC	226	â	11100010	E2
173		10101101	AD	227	ã	11100011	E3
174	®	10101110	AE	228	ä	11100100	E4
175	˘	10101111	AF	229	å	11100101	E5
176	°	10110000	B0	230	æ	11100110	E6
177	±	10110001	B1	231	ç	11100111	E7
178	²	10110010	B2	232	è	11101000	E8
179	³	10110011	B3	233	é	11101001	E9
180	´	10110100	B4	234	ê	11101010	EA
181	µ	10110101	B5	235	ë	11101011	EB
182	¶	10110110	B6	236	ì	11101100	EC
183	•	10110111	B7	237	í	11101101	ED
184	¸	10111000	B8	238	î	11101110	EE
185	¹	10111001	B9	239	ï	11101111	EF
186	º	10111010	BA	240	ð	11110000	F0
187	»	10111011	BB	241	ñ	11110001	F1
188	¼	10111100	BC	242	ò	11110010	F2
189	½	10111101	BD	243	ó	11110011	F3
190	¾	10111110	BE	244	ô	11110100	F4
191	¿	10111111	BF	245	õ	11110101	F5
192	À	11000000	C0	246	ö	11110110	F6
193	Á	11000001	C1	247	÷	11110111	F7
194	Â	11000010	C2	248	ø	11111000	F8
195	Ã	11000011	C3	249	ù	11111001	F9
196	Ä	11000100	C4	250	ú	11111010	FA
197	Å	11000101	C5	251	û	11111011	FB
198	Æ	11000110	C6	252	ü	11111100	FC
199	Ç	11000111	C7	253	ý	11111101	FD
200	È	11001000	C8	254	þ	11111110	FE
201	É	11001001	C9	255	ÿ	11111111	FF

## Appendix III Material List and Cost

PROJECT	HAND POSITIONED MOUSE WITH PUSHBUTTON		Qty			COS (SING)	
MACHINE	HAND POSITIONED MOUSE WITH PUSHBUTTON		1				
UNIT ASSEMBLY	JAN 07-BEHE-12 ELECTRICAL CONTROL		1				
Item Part	Description	Maker	Maker Spec	Qty	Supplier	COS (SING)	
1	JAN 07-BEHE-12-001	DIP (MICROCONTROLLER)	MICROCHIP	PIC18F452	1	MICROCHIP	18.55
2	JAN 07-BEHE-12-002	±1.5g X-AXIS MICROMACHINED ACCELEROMETER	MOTOROLA	MMA2260D (SENSOR)	1	FREESCALE	24.09
3	JAN 07-BEHE-12-003	LOW G MICROMACHINED ACCELEROMETER	MOTOROLA	MMA1260D (SENSOR)	1	FREESCALE	24.09
4	JAN 07-BEHE-12-004	5V-Powered, Multichannel RS-232 Drivers/Receivers	MAXIM-IC	MAX232CPE	1	MAXIM-IC	7.05
5	JAN 07-BEHE-12-005	RESISTOR, 0.25W 5% 1K	MULTICOMP	MCF 0.25W 1K	4	FARNELL	0.16
6	JAN 07-BEHE-12-006	CAPACITOR, 22PF 50V	MULTICOMP	MCCHU5220J5	2	FARNELL	0.16
7	JAN 07-BEHE-12-007	CAPACITOR, 10NF 50V	MULTICOMP	MC FYU5103Z5	2	FARNELL	0.10
8	JAN 07-BEHE-12-008	CAPACITOR, 100NF 50V	MULTICOMP	MC FYU6104Z6	4	FARNELL	0.84
9	JAN 07-BEHE-12-009	CAPACITOR, 1UF 25V	SANYO	25SH1M	5	FARNELL	5.05
10	JAN 07-BEHE-12-010	DB 9 FEMALE CONNECTOR C/W PLASTIC COVER	ALTITUDE	CHROME TYPE	1	ALTITUDE	1.20
11	JAN 07-BEHE-12-011	CAPACITOR, 0.33UF 35V	MULTICOMP	CB1V334M2ACB	1	FARNELL	0.46
12	JAN 07-BEHE-12-012	VOLTAGE REGULATOR	NATIONAL SEMICON	L7805CV	1	FARNELL	1.33
13	JAN 07-BEHE-12-013	IC SOCKET, DIL 0.6 40WAY"	MULTICOMP	2227-40-06-05	1	FARNELL	0.50
14	JAN 07-BEHE-12-014	CONNECTOR, BATTERY PP3 PK10	MULTICOMP	440005P	1	FARNELL	2.96
15	JAN 07-BEHE-12-015	EUROCARD,FR2,spacing 2.54mm	ROTH ELEKTRONIK	RE315-HP	1	FARNELL	8.98
16	JAN 07-BEHE-12-016	CRYSTAL, 16.000000MHZ	C-MAC FREQUENCY PRODUCTS	LF A161A	1	FARNELL	1.63
17	JAN 07-BEHE-12-017	RESISTOR, 0.25W 5% 22R	MULTICOMP	MCF 0.25W 22R	1	FARNELL	0.04
18	JAN 07-BEHE-12-018	IC SOCKET, DIL 16 WAY	TYCO ELECTRONICS / AUGAT	816-AG11D-ESL-LF	1	FARNELL	1.77
19	JAN 07-BEHE-12-019	SWITCH, SPNO FLAT	OMRON	B3F-1000	2	FARNELL	2.04
20	JAN 07-BEHE-12-020	RESISTOR, 0.25W 5% 4.7R	MULTICOMP	MCF 0.25W 4.7R	2	FARNELL	0.08
21	JAN 07-BEHE-12-021	ADP UNIT, IC PROGRAMMER	MICROCHIP DIRECT	DV164006	1	MICROCHIP	398.00
						499.08	

## Appendix IV Application source code listing

```

;*****
;* 18F452.ASM
;*****
;* SIM University Final Year Project (Hand Positioned Mouse)
;* 02 NOVEMBER 2007
;* Assembled with MPASM V5.11
;*****
;* This program configures
;* 1) Read and convert A/D signal from Accelerometer for Motion detection
;* 2) Read on/off states from Pushbutton for Button detection
;* 3) RS-232 Signal Generation
;*****
;* Version 1.0 070822 - Able to communicate with PC. Microsoft Serial Mouse detected.
;* Version 1.1 070905 - Able to sense right and left button click
;* Version 1.2 070916 - Able to detect X and Y Axis Movement

        list p=18f452
        include "p18f452.inc"
;
;-----
; FILES ASSIGNMENT
;-----
;
COUNT1      EQU 8           ;DELAY LOOP1
COUNT2      EQU 9           ;DELAY LOOP2
COUNT3      EQU 10          ;DELAY LOOP2
TIMER1       EQU 11          ;COUNTER FOR DELAY
DETECT       EQU 12          ;LOOP FOR DETECT SERIAL MOUSE
DETECT1      EQU 13          ;LOOP FOR DETECT SERIAL MOUSE
DATA_X       EQU 16          ;DATA OF X-AXIS
DATA_Y       EQU 17          ;DATA OF Y-AXIS
COUNT       EQU 18          ;GENERAL PURPOSE COUNTER
ClkFreq      EQU 16000000
BAUD1200     EQU 207         ;((10*ClkFreq/(64*9600))+5/10-1
TXSTA_INIT   EQU 0X20        ;SEE PAGE 168 IN P18F452
                                MANUAL(TRANSMIT ENABLE)
RCSTA_INIT   EQU 0X90        ;SEE PAGE 169 IN P18F452 MANUAL(SERIAL
                                PORT/CONTINUOUS RECEIVE ENABLE)
F            EQU 1           ;
;
;-----
; Program Address
;-----
;
RESET_VECTOR CODE 0x000      ;RESET ADDRESS
        GOTO START

        CODE 0x0002A

START

;
;-----
; Initalise
;-----
;
INIT
        CLRF TRISA
        CLRF TRISB
        CLRF TRISC
        CLRF PORTA
        CLRF PORTB
        CLRF PORTC

```

```

CLRF F
CLRF W
MOVLW B'01111111'      ;INIT RA0~6 BE INPUTS
MOVWF TRISA            ;
MOVLW B'11000000'      ;INIT RB0~7 BE OUTPUTS
MOVWF TRISB
MOVLW B'00000000'      ;INIT RC0~1 BE OUTPUTS
MOVWF TRISC
MOVLW B'11000111'      ;TMR0 prescaler, 1:256
MOVWF T0CON

;
;-----
; Setup Communication with Laptop or PC
;-----
;

INIT_COM

    MOVLW B'11001111'
    MOVWF SPBRG
    MOVLW TXSTA_INIT
    MOVWF TXSTA        ; ENABLE TRANSMISSION (TXEN)
    MOVLW RCSTA_INIT
    MOVWF RCSTA        ;(SERIAL PORT/CONTINUOUS RECEIVE ENABLE)
    BCF   PIE1,4
    MOVLW 06H
    MOVWF DETECT
    MOVLW 01H
    MOVWF DETECT1
    GOTO  SETUP_COM

SETUP_COM

POLLTXIS1
    BTFSS PIR1,4
    GOTO  POLLTXIS1
    BCF   PORTB,1
    MOVLW B'01001101'
    MOVWF TXREG
    CALL  DELAY1
    CALL  DELAY1
    BSF   PORTB,1
    DECFSZ DETECT,1
    GOTO  SETUP_COM
    GOTO  SETUP_COM_DELAY

;
;-----
; Detection of Button pressed
;-----
;
SETUP_COM_DELAY

    CALL  DELAY1
    BTFSS PORTB,6
    GOTO  RBUTTON_SET
    BTFSS PORTB,7
    GOTO  LBUTTON_SET
    GOTO  X_AXIS
    GOTO  SETUP_COM_DELAY

;
;-----
; Right Button
;-----
;

RBUTTON_SET
    MOVLW B'11100000'      ; BYTE 1
    MOVWF TXREG            ; SEND TO COM PORT
    CALL  DELAY2
    MOVLW B'10000000'      ; BYTE 2
    MOVWF TXREG            ; SEND TO COM PORT
    CALL  DELAY2
    MOVLW B'10000000'      ; BYTE 3

```

```

MOVWF TXREG           ; SEND TO COM PORT
CALL DELAY2
MOVLW B'11000000'    ; BYTE 1
MOVWF TXREG           ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10000000'    ; BYTE 2
MOVWF TXREG           ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10000000'    ; BYTE 3
MOVWF TXREG           ; SEND TO COM PORT
CALL DELAY2
MOVLW B'11100000'    ; BYTE 1
MOVWF TXREG           ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10000000'    ; BYTE 2
MOVWF TXREG           ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10000000'    ; BYTE 3
MOVWF TXREG           ; SEND TO COM PORT
CALL DELAY2
MOVLW B'11000000'    ; BYTE 1
MOVWF TXREG           ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10000000'    ; BYTE 2
MOVWF TXREG           ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10000000'    ; BYTE 3
MOVWF TXREG           ; SEND TO COM PORT
CALL DELAY2
GOTO SETUP_COM_DELAY

;
;-----
; Left Button
;-----
;
;
LBUTTON_SET
MOVLW B'11010000'    ; BYTE 1
MOVWF TXREG           ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10000000'    ; BYTE 2
MOVWF TXREG           ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10000000'    ; BYTE 3
MOVWF TXREG           ; SEND TO COM PORT
CALL DELAY2
MOVLW B'11000000'    ; BYTE 1
MOVWF TXREG           ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10000000'    ; BYTE 2
MOVWF TXREG           ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10000000'    ; BYTE 3
MOVWF TXREG           ; SEND TO COM PORT
CALL DELAY2
GOTO SETUP_COM_DELAY

;
;-----
; X_AXIS
;-----
;
;
X_AXIS
MOVLW B'01000001'    ;A/D ENABLED (BIT 5 to 3 = 000, AN0 SELECTED)
MOVWF ADCON0          ;A/D ENABLED
MOVLW B'00000100'    ;CONFIGURE AN0/AN1/AN3 AS ANALOG CHANNEL
MOVWF ADCON1
BSF ADCON0,GO

WAITX
BTFS PIR1,ADIF        ;A/D CONVERTER INTERRUPT FLAG BIT
GOTO WAITX            ;1=AN A/D CONVERSION COMPLETED(MUST BE
                      ;CLEAR IN SOFTWARE)

```

```

        SWAPF  ADRESH,W
        ANDLW  0X1F
        MOVWF  DATA_X

        BCF   ADCON0,GO

        GOTO  NEGATIVE_X

;
;-----
;NEGATIVE_X
;-----
;
;
NEGATIVE_X

        MOVLW B'00000001'
        CPFSEQ DATA_X,W
        GOTO   NX2
        GOTO   XN1
NX2
        MOVLW B'00000010'
        CPFSEQ DATA_X,W
        GOTO   NX3
        GOTO   XN2
NX3
        MOVLW B'00000011'
        CPFSEQ DATA_X,W
        GOTO   NX4
        GOTO   XN3
NX4
        MOVLW B'00000100'
        CPFSEQ DATA_X,W
        GOTO   NX5
        GOTO   XN4
NX5
        MOVLW B'00000101'
        CPFSEQ DATA_X,W
        GOTO   NX6
        GOTO   XN5
NX6
        MOVLW B'00000110'
        CPFSEQ DATA_X,W
        GOTO   NX7
        GOTO   XN6
NX7
        MOVLW B'00000111'
        CPFSEQ DATA_X,W
        GOTO   NX8
        GOTO   XN7
NX8
        MOVLW B'00001000'
        CPFSEQ DATA_X,W
        GOTO   NX9
        GOTO   XN8
NX9
        MOVLW B'00001001'
        CPFSEQ DATA_X,W
        GOTO   NX10
        GOTO   XN9
NX10
        MOVLW B'00001010'
        CPFSEQ DATA_X,W
        GOTO   NX11
        GOTO   XN10
NX11
        MOVLW B'00001011'
        CPFSEQ DATA_X,W
        GOTO   NX12
        GOTO   XN11
NX12
        MOVLW B'00001100'
        CPFSEQ DATA_X,W
        GOTO   NX13
        GOTO   XN12

```

```

;SWAP A/D RESULT NIBBLE
;MASK OFF LOWER 3 BITS
;WRITE A/D RESULT TO DATA_X

```

```

NX13      MOVLW B'00001101'
           CPFSEQ DATA_X,W
           GOTO  NX14
           GOTO  XN13

NX14      MOVLW B'00001110'
           CPFSEQ DATA_X,W
           GOTO  NX15
           GOTO  XN14

NX15      MOVLW B'00001111'
           CPFSEQ DATA_X,W
           GOTO  POSITIVE_X
           GOTO  XN15

XN1       MOVLW B'11000000'      ; BYTE 1
           MOVWF TXREG          ; SEND TO COM PORT
           CALL  DELAY2
           MOVLW B'10001000'    ; BYTE 2
           MOVWF TXREG          ; SEND TO COM PORT
           CALL  DELAY2
           MOVLW B'10000000'    ; BYTE 3
           MOVWF TXREG          ; SEND TO COM PORT
           CLRF  DATA_X
           GOTO  Y_AXIS

XN2       MOVLW B'11000000'    ; BYTE 1
           MOVWF TXREG          ; SEND TO COM PORT
           CALL  DELAY2
           MOVLW B'10010000'    ; BYTE 2
           MOVWF TXREG          ; SEND TO COM PORT
           CALL  DELAY2
           MOVLW B'10000000'    ; BYTE 3
           MOVWF TXREG          ; SEND TO COM PORT
           CLRF  DATA_X
           GOTO  Y_AXIS

XN3       MOVLW B'11000000'    ; BYTE 1
           MOVWF TXREG          ; SEND TO COM PORT
           CALL  DELAY2
           MOVLW B'10011000'    ; BYTE 2
           MOVWF TXREG          ; SEND TO COM PORT
           CALL  DELAY2
           MOVLW B'10000000'    ; BYTE 3
           MOVWF TXREG          ; SEND TO COM PORT
           CLRF  DATA_X
           GOTO  Y_AXIS

XN4       MOVLW B'11000000'    ; BYTE 1
           MOVWF TXREG          ; SEND TO COM PORT
           CALL  DELAY2
           MOVLW B'10100000'    ; BYTE 2
           MOVWF TXREG          ; SEND TO COM PORT
           CALL  DELAY2
           MOVLW B'10000000'    ; BYTE 3
           MOVWF TXREG          ; SEND TO COM PORT
           CLRF  DATA_X
           GOTO  Y_AXIS

XN5       MOVLW B'11000000'    ; BYTE 1
           MOVWF TXREG          ; SEND TO COM PORT
           CALL  DELAY2
           MOVLW B'10101000'    ; BYTE 2
           MOVWF TXREG          ; SEND TO COM PORT
           CALL  DELAY2
           MOVLW B'10000000'    ; BYTE 3
           MOVWF TXREG          ; SEND TO COM PORT
           CLRF  DATA_X
           GOTO  Y_AXIS

XN6       MOVLW B'11000000'    ; BYTE 1
           MOVWF TXREG          ; SEND TO COM PORT
           CALL  DELAY2
           MOVLW B'10110000'    ; BYTE 2

```

```

MOVWF TXREG          ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10000000'    ; BYTE 3
MOVWF TXREG          ; SEND TO COM PORT
CLRF DATA_X
GOTO Y_AXIS

XN7
MOVLW B'11000000'    ; BYTE 1
MOVWF TXREG          ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10111000'    ; BYTE 2
MOVWF TXREG          ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10000000'    ; BYTE 3
MOVWF TXREG          ; SEND TO COM PORT
CLRF DATA_X
GOTO Y_AXIS

XN8
MOVLW B'11000001'    ; BYTE 1
MOVWF TXREG          ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10000000'    ; BYTE 2
MOVWF TXREG          ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10000000'    ; BYTE 3
MOVWF TXREG          ; SEND TO COM PORT
CLRF DATA_X
GOTO Y_AXIS

XN9
MOVLW B'11000001'    ; BYTE 1
MOVWF TXREG          ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10001000'    ; BYTE 2
MOVWF TXREG          ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10000000'    ; BYTE 3
MOVWF TXREG          ; SEND TO COM PORT
CLRF DATA_X
GOTO Y_AXIS

XN10
MOVLW B'11000001'    ; BYTE 1
MOVWF TXREG          ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10010000'    ; BYTE 2
MOVWF TXREG          ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10000000'    ; BYTE 3
MOVWF TXREG          ; SEND TO COM PORT
CLRF DATA_X
GOTO Y_AXIS

XN11
MOVLW B'11000001'    ; BYTE 1
MOVWF TXREG          ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10011000'    ; BYTE 2
MOVWF TXREG          ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10000000'    ; BYTE 3
MOVWF TXREG          ; SEND TO COM PORT
CLRF DATA_X
GOTO Y_AXIS

XN12
MOVLW B'11000001'    ; BYTE 1
MOVWF TXREG          ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10100000'    ; BYTE 2
MOVWF TXREG          ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10000000'    ; BYTE 3
MOVWF TXREG          ; SEND TO COM PORT
CLRF DATA_X
GOTO Y_AXIS

XN13
MOVLW B'11000001'    ; BYTE 1

```

```

MOVWF TXREG           ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10101000'    ; BYTE 2
MOVWF TXREG           ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10000000'    ; BYTE 3
MOVWF TXREG           ; SEND TO COM PORT
CLRF DATA_X
GOTO Y_AXIS

XN14
MOVLW B'11000001'    ; BYTE 1
MOVWF TXREG           ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10110000'    ; BYTE 2
MOVWF TXREG           ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10000000'    ; BYTE 3
MOVWF TXREG           ; SEND TO COM PORT
CLRF DATA_X
GOTO Y_AXIS

XN15
MOVLW B'11000001'    ; BYTE 1
MOVWF TXREG           ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10111000'    ; BYTE 2
MOVWF TXREG           ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10000000'    ; BYTE 3
MOVWF TXREG           ; SEND TO COM PORT
CLRF DATA_X
GOTO Y_AXIS

XN16
MOVLW B'11000001'    ; BYTE 1
MOVWF TXREG           ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10111111'    ; BYTE 2
MOVWF TXREG           ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10000000'    ; BYTE 3
MOVWF TXREG           ; SEND TO COM PORT
CLRF DATA_X
GOTO Y_AXIS

;
;-----
; POSITIVE_X
;-----
;

POSITIVE_X

MOVWF B'00010001'
CPFSEQ DATA_X,W
GOTO PX2
GOTO XN1

PX2
MOVWF B'00010010'
CPFSEQ DATA_X,W
GOTO PX3
GOTO XN2

PX3
MOVWF B'00010011'
CPFSEQ DATA_X,W
GOTO PX4
GOTO XN3

PX4
MOVWF B'00010100'
CPFSEQ DATA_X,W
GOTO PX5
GOTO XN4

PX5
MOVWF B'00010101'
CPFSEQ DATA_X,W
GOTO PX6

```

```

GOTO    XN5
PX6
MOVLW  B'00010110'
CPFSEQ DATA_X,W
GOTO    PX7
GOTO    XN6
PX7
MOVLW  B'00010111'
CPFSEQ DATA_X,W
GOTO    PX8
GOTO    XN7
PX8
MOVLW  B'00011000'
CPFSEQ DATA_X,W
GOTO    PX9
GOTO    XN8
PX9
MOVLW  B'00011001'
CPFSEQ DATA_X,W
GOTO    PX10
GOTO    XN9
PX10
MOVLW  B'00011010'
CPFSEQ DATA_X,W
GOTO    PX11
GOTO    XN10
PX11
MOVLW  B'00011011'
CPFSEQ DATA_X,W
GOTO    PX12
GOTO    XN11
PX12
MOVLW  B'00011100'
CPFSEQ DATA_X,W
GOTO    PX13
GOTO    XN12
PX13
MOVLW  B'00011101'
CPFSEQ DATA_X,W
GOTO    PX14
GOTO    XN13
PX14
MOVLW  B'00011110'
CPFSEQ DATA_X,W
GOTO    PX15
GOTO    XN14
PX15
MOVLW  B'00011111'
CPFSEQ DATA_X,W
GOTO    NEGATIVE_Y
GOTO    XN15

XP1
MOVLW  B'11000010'    ; BYTE 1
MOVWF  TXREG          ; SEND TO COM PORT
CALL   DELAY2
MOVLW  B'10001000'    ; BYTE 2
MOVWF  TXREG          ; SEND TO COM PORT
CALL   DELAY2
MOVLW  B'10000000'    ; BYTE 3
MOVWF  TXREG          ; SEND TO COM PORT
CLRF   DATA_X
GOTO   Y_AXIS

XP2
MOVLW  B'11000010'    ; BYTE 1
MOVWF  TXREG          ; SEND TO COM PORT
CALL   DELAY2
MOVLW  B'10010000'    ; BYTE 2
MOVWF  TXREG          ; SEND TO COM PORT
CALL   DELAY2
MOVLW  B'10000000'    ; BYTE 3
MOVWF  TXREG          ; SEND TO COM PORT
CLRF   DATA_X
GOTO   Y_AXIS

XP3

```

```

        MOVLW B'11000010'    ; BYTE 1
        MOVWF TXREG          ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10011000'    ; BYTE 2
        MOVWF TXREG          ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10000000'    ; BYTE 3
        MOVWF TXREG          ; SEND TO COM PORT
        CLRF  DATA_X
        GOTO  Y_AXIS
XP4
        MOVLW B'11000010'    ; BYTE 1
        MOVWF TXREG          ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10100000'    ; BYTE 2
        MOVWF TXREG          ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10000000'    ; BYTE 3
        MOVWF TXREG          ; SEND TO COM PORT
        CLRF  DATA_X
        GOTO  Y_AXIS
XP5
        MOVLW B'11000010'    ; BYTE 1
        MOVWF TXREG          ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10101000'    ; BYTE 2
        MOVWF TXREG          ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10000000'    ; BYTE 3
        MOVWF TXREG          ; SEND TO COM PORT
        CLRF  DATA_X
        GOTO  Y_AXIS
XP6
        MOVLW B'11000010'    ; BYTE 1
        MOVWF TXREG          ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10110000'    ; BYTE 2
        MOVWF TXREG          ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10000000'    ; BYTE 3
        MOVWF TXREG          ; SEND TO COM PORT
        CLRF  DATA_X
        GOTO  Y_AXIS
XP7
        MOVLW B'11000010'    ; BYTE 1
        MOVWF TXREG          ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10111000'    ; BYTE 2
        MOVWF TXREG          ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10000000'    ; BYTE 3
        MOVWF TXREG          ; SEND TO COM PORT
        CLRF  DATA_X
        GOTO  Y_AXIS
XP8
        MOVLW B'11000011'    ; BYTE 1
        MOVWF TXREG          ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10000000'    ; BYTE 2
        MOVWF TXREG          ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10000000'    ; BYTE 3
        MOVWF TXREG          ; SEND TO COM PORT
        CLRF  DATA_X
        GOTO  Y_AXIS
XP9
        MOVLW B'11000011'    ; BYTE 1
        MOVWF TXREG          ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10001000'    ; BYTE 2
        MOVWF TXREG          ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10000000'    ; BYTE 3
        MOVWF TXREG          ; SEND TO COM PORT
        CLRF  DATA_X

```

```

GOTO   Y_AXIS
XP10
    MOVLW B'11000011'   ; BYTE 1
    MOVWF TXREG         ; SEND TO COM PORT
    CALL  DELAY2
    MOVLW B'10010000'   ; BYTE 2
    MOVWF TXREG         ; SEND TO COM PORT
    CALL  DELAY2
    MOVLW B'10000000'   ; BYTE 3
    MOVWF TXREG         ; SEND TO COM PORT
    CLRF  DATA_X
    GOTO  Y_AXIS
XP11
    MOVLW B'11000011'   ; BYTE 1
    MOVWF TXREG         ; SEND TO COM PORT
    CALL  DELAY2
    MOVLW B'10011000'   ; BYTE 2
    MOVWF TXREG         ; SEND TO COM PORT
    CALL  DELAY2
    MOVLW B'10000000'   ; BYTE 3
    MOVWF TXREG         ; SEND TO COM PORT
    CLRF  DATA_X
    GOTO  Y_AXIS
XP12
    MOVLW B'11000011'   ; BYTE 1
    MOVWF TXREG         ; SEND TO COM PORT
    CALL  DELAY2
    MOVLW B'10100000'   ; BYTE 2
    MOVWF TXREG         ; SEND TO COM PORT
    CALL  DELAY2
    MOVLW B'10000000'   ; BYTE 3
    MOVWF TXREG         ; SEND TO COM PORT
    CLRF  DATA_X
    GOTO  Y_AXIS
XP13
    MOVLW B'11000011'   ; BYTE 1
    MOVWF TXREG         ; SEND TO COM PORT
    CALL  DELAY2
    MOVLW B'10101000'   ; BYTE 2
    MOVWF TXREG         ; SEND TO COM PORT
    CALL  DELAY2
    MOVLW B'10000000'   ; BYTE 3
    MOVWF TXREG         ; SEND TO COM PORT
    CLRF  DATA_X
    GOTO  Y_AXIS
XP14
    MOVLW B'11000011'   ; BYTE 1
    MOVWF TXREG         ; SEND TO COM PORT
    CALL  DELAY2
    MOVLW B'10110000'   ; BYTE 2
    MOVWF TXREG         ; SEND TO COM PORT
    CALL  DELAY2
    MOVLW B'10000000'   ; BYTE 3
    MOVWF TXREG         ; SEND TO COM PORT
    CLRF  DATA_X
    GOTO  Y_AXIS
XP15
    MOVLW B'11000011'   ; BYTE 1
    MOVWF TXREG         ; SEND TO COM PORT
    CALL  DELAY2
    MOVLW B'10111000'   ; BYTE 2
    MOVWF TXREG         ; SEND TO COM PORT
    CALL  DELAY2
    MOVLW B'10000000'   ; BYTE 3
    MOVWF TXREG         ; SEND TO COM PORT
    CLRF  DATA_X
    GOTO  Y_AXIS
XP16
    MOVLW B'11000011'   ; BYTE 1
    MOVWF TXREG         ; SEND TO COM PORT
    CALL  DELAY2
    MOVLW B'10111111'   ; BYTE 2
    MOVWF TXREG         ; SEND TO COM PORT
    CALL  DELAY2
    MOVLW B'10000000'   ; BYTE 3

```

```

MOVWF TXREG          ; SEND TO COM PORT
CLRF  DATA_X
GOTO  Y_AXIS

;
-----
; Y_AXIS
-----
;
;
Y_AXIS
MOV LW B'01001001'   ;A/D ENABLED (BIT 5 to 3 = 001, AN1 SELECTED)
MOVWF ADCON0         ;A/D ENABLED
MOV LW B'00000100'   ;CONFIGURE ANO/AN1/AN3 AS ANALOG CHANNEL
MOVWF ADCON1
BSF  ADCON0,GO

WAITY
BTFSS PIR1,ADIF     ;A/D CONVERTER INTERRUPT FLAG BIT
GOTO  WAITY         ;1=AN A/D CONVERSION COMPLETED(MUST BE
                    ;CLEAR IN SOFTWARE)

SWAPF ADRESH,W      ;SWAP A/D RESULT NIBBLE
ANDLW 0X1F          ;MASK OFF LOWER 3 BITS
MOVWF DATA_X       ;WRITE A/D RESULT TO DATA_X

BCF  ADCON0,GO

GOTO  NEGATIVE_Y

;
-----
; NEGATIVE_Y
-----
;
;
NEGATIVE_Y
NY1
MOV LW B'00000001'
CPFSEQ DATA_Y,W
GOTO  NY2
GOTO  YN1

NY2
MOV LW B'00000010'
CPFSEQ DATA_Y,W
GOTO  NY3
GOTO  YN2

NY3
MOV LW B'00000011'
CPFSEQ DATA_Y,W
GOTO  NY4
GOTO  YN3

NY4
MOV LW B'00000100'
CPFSEQ DATA_Y,W
GOTO  NY5
GOTO  YN4

NY5
MOV LW B'00000101'
CPFSEQ DATA_Y,W
GOTO  NY6
GOTO  YN5

NY6
MOV LW B'00000110'
CPFSEQ DATA_Y,W
GOTO  NY7
GOTO  YN6

NY7
MOV LW B'00000111'
CPFSEQ DATA_Y,W
GOTO  NY8
GOTO  YN7

NY8
MOV LW B'00001000'
CPFSEQ DATA_Y,W

```

```

        GOTO  NY9
        GOTO  YN8
NY9
        MOVLW B'00001001'
        CPFSEQ DATA_Y,W
        GOTO  NY10
        GOTO  YN9
NY10
        MOVLW B'00001010'
        CPFSEQ DATA_Y,W
        GOTO  NY11
        GOTO  YN10
NY11
        MOVLW B'00001011'
        CPFSEQ DATA_Y,W
        GOTO  NY12
        GOTO  YN11
NY12
        MOVLW B'00001100'
        CPFSEQ DATA_Y,W
        GOTO  NY13
        GOTO  YN12
NY13
        MOVLW B'00001101'
        CPFSEQ DATA_Y,W
        GOTO  NY14
        GOTO  YN13
NY14
        MOVLW B'00001110'
        CPFSEQ DATA_Y,W
        GOTO  NY15
        GOTO  YN14
NY15
        MOVLW B'00001111'
        CPFSEQ DATA_Y,W
        GOTO  POSITIVE_Y
        GOTO  YN15

YN1
        MOVLW B'11000000'      ; BYTE 1
        MOVWF TXREG           ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10000000'      ; BYTE 2
        MOVWF TXREG           ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10001000'      ; BYTE 3
        MOVWF TXREG           ; SEND TO COM PORT
        CLRF  DATA_Y
        GOTO  SETUP_COM_DELAY

YN2
        MOVLW B'11000000'      ; BYTE 1
        MOVWF TXREG           ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10000000'      ; BYTE 2
        MOVWF TXREG           ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10010000'      ; BYTE 3
        MOVWF TXREG           ; SEND TO COM PORT
        CLRF  DATA_Y
        GOTO  SETUP_COM_DELAY

YN3
        MOVLW B'11000000'      ; BYTE 1
        MOVWF TXREG           ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10000000'      ; BYTE 2
        MOVWF TXREG           ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10011000'      ; BYTE 3
        MOVWF TXREG           ; SEND TO COM PORT
        CLRF  DATA_Y
        GOTO  SETUP_COM_DELAY

YN4
        MOVLW B'11000000'      ; BYTE 1
        MOVWF TXREG           ; SEND TO COM PORT
        CALL  DELAY2

```

```
        MOVLW B'10000000'    ; BYTE 2
        MOVWF TXREG          ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10100000'    ; BYTE 3
        MOVWF TXREG          ; SEND TO COM PORT
        CLRF  DATA_Y
        GOTO  SETUP_COM_DELAY
YN5
        MOVLW B'11000000'    ; BYTE 1
        MOVWF TXREG          ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10000000'    ; BYTE 2
        MOVWF TXREG          ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10101000'    ; BYTE 3
        MOVWF TXREG          ; SEND TO COM PORT
        CLRF  DATA_Y
        GOTO  SETUP_COM_DELAY
YN6
        MOVLW B'11000000'    ; BYTE 1
        MOVWF TXREG          ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10000000'    ; BYTE 2
        MOVWF TXREG          ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10110000'    ; BYTE 3
        MOVWF TXREG          ; SEND TO COM PORT
        CLRF  DATA_Y
        GOTO  SETUP_COM_DELAY
YN7
        MOVLW B'11000000'    ; BYTE 1
        MOVWF TXREG          ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10000000'    ; BYTE 2
        MOVWF TXREG          ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10111000'    ; BYTE 3
        MOVWF TXREG          ; SEND TO COM PORT
        CLRF  DATA_Y
        GOTO  SETUP_COM_DELAY
YN8
        MOVLW B'11000100'    ; BYTE 1
        MOVWF TXREG          ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10000000'    ; BYTE 2
        MOVWF TXREG          ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10000000'    ; BYTE 3
        MOVWF TXREG          ; SEND TO COM PORT
        CLRF  DATA_Y
        GOTO  SETUP_COM_DELAY
YN9
        MOVLW B'11000100'    ; BYTE 1
        MOVWF TXREG          ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10000000'    ; BYTE 2
        MOVWF TXREG          ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10001000'    ; BYTE 3
        MOVWF TXREG          ; SEND TO COM PORT
        CLRF  DATA_Y
        GOTO  SETUP_COM_DELAY
YN10
        MOVLW B'11000100'    ; BYTE 1
        MOVWF TXREG          ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10000000'    ; BYTE 2
        MOVWF TXREG          ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10010000'    ; BYTE 3
        MOVWF TXREG          ; SEND TO COM PORT
        CLRF  DATA_Y
        GOTO  SETUP_COM_DELAY
YN11
        MOVLW B'11000100'    ; BYTE 1
```

```

MOVWF TXREG          ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10000000'    ; BYTE 2
MOVWF TXREG          ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10011000'    ; BYTE 3
MOVWF TXREG          ; SEND TO COM PORT
CLRF DATA_Y
GOTO SETUP_COM_DELAY

YN12
MOVLW B'11000100'    ; BYTE 1
MOVWF TXREG          ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10000000'    ; BYTE 2
MOVWF TXREG          ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10100000'    ; BYTE 3
MOVWF TXREG          ; SEND TO COM PORT
CLRF DATA_Y
GOTO SETUP_COM_DELAY

YN13
MOVLW B'11000100'    ; BYTE 1
MOVWF TXREG          ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10000000'    ; BYTE 2
MOVWF TXREG          ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10101000'    ; BYTE 3
MOVWF TXREG          ; SEND TO COM PORT
CLRF DATA_Y
GOTO SETUP_COM_DELAY

YN14
MOVLW B'11000100'    ; BYTE 1
MOVWF TXREG          ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10000000'    ; BYTE 2
MOVWF TXREG          ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10110000'    ; BYTE 3
MOVWF TXREG          ; SEND TO COM PORT
CLRF DATA_Y
GOTO SETUP_COM_DELAY

YN15
MOVLW B'11000100'    ; BYTE 1
MOVWF TXREG          ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10000000'    ; BYTE 2
MOVWF TXREG          ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10111000'    ; BYTE 3
MOVWF TXREG          ; SEND TO COM PORT
CLRF DATA_Y
GOTO SETUP_COM_DELAY

YN16
MOVLW B'11000100'    ; BYTE 1
MOVWF TXREG          ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10000000'    ; BYTE 2
MOVWF TXREG          ; SEND TO COM PORT
CALL DELAY2
MOVLW B'10111111'    ; BYTE 3
MOVWF TXREG          ; SEND TO COM PORT
CLRF DATA_Y
GOTO SETUP_COM_DELAY

```

```

;
;-----
; POSITIVE_Y
;-----
;

```

POSITIVE\_Y

PY1

Tay Boon Kwong Roland  
Hand Positioned Mouse

E0401574  
07/BEHE/12

```
      MOVLW B'00010001'  
      CPFSEQ DATA_Y,W  
      GOTO PY2  
      GOTO YP1  
PY2  
      MOVLW B'00010010'  
      CPFSEQ DATA_Y,W  
      GOTO PY3  
      GOTO YP2  
PY3  
      MOVLW B'00010011'  
      CPFSEQ DATA_Y,W  
      GOTO PY4  
      GOTO YP3  
PY4  
      MOVLW B'00010100'  
      CPFSEQ DATA_Y,W  
      GOTO PY5  
      GOTO YP4  
PY5  
      MOVLW B'00010101'  
      CPFSEQ DATA_Y,W  
      GOTO PY6  
      GOTO YP5  
PY6  
      MOVLW B'00010110'  
      CPFSEQ DATA_Y,W  
      GOTO PY7  
      GOTO YP6  
PY7  
      MOVLW B'00010111'  
      CPFSEQ DATA_Y,W  
      GOTO PY8  
      GOTO YP7  
PY8  
      MOVLW B'00011000'  
      CPFSEQ DATA_Y,W  
      GOTO PY9  
      GOTO YP8  
PY9  
      MOVLW B'00011001'  
      CPFSEQ DATA_Y,W  
      GOTO PY10  
      GOTO YP9  
PY10  
      MOVLW B'00011010'  
      CPFSEQ DATA_Y,W  
      GOTO PY11  
      GOTO YP10  
PY11  
      MOVLW B'00011011'  
      CPFSEQ DATA_Y,W  
      GOTO PY12  
      GOTO YP11  
PY12  
      MOVLW B'00011100'  
      CPFSEQ DATA_Y,W  
      GOTO PY13  
      GOTO YP12  
PY13  
      MOVLW B'00011101'  
      CPFSEQ DATA_Y,W  
      GOTO PY14  
      GOTO YP13  
PY14  
      MOVLW B'00011110'  
      CPFSEQ DATA_Y,W  
      GOTO PY15  
      GOTO YP14  
PY15  
      MOVLW B'00011111'  
      CPFSEQ DATA_Y,W  
      GOTO SETUP_COM_DELAY  
      GOTO YP15
```

```
YP1
    MOVLW B'11001000'    ; BYTE 1
    MOVWF TXREG          ; SEND TO COM PORT
    CALL  DELAY2
    MOVLW B'10000000'    ; BYTE 2
    MOVWF TXREG          ; SEND TO COM PORT
    CALL  DELAY2
    MOVLW B'10001000'    ; BYTE 3
    MOVWF TXREG          ; SEND TO COM PORT
    CLRF  DATA_Y
    GOTO  SETUP_COM_DELAY

YP2
    MOVLW B'11001000'    ; BYTE 1
    MOVWF TXREG          ; SEND TO COM PORT
    CALL  DELAY2
    MOVLW B'10000000'    ; BYTE 2
    MOVWF TXREG          ; SEND TO COM PORT
    CALL  DELAY2
    MOVLW B'10010000'    ; BYTE 3
    MOVWF TXREG          ; SEND TO COM PORT
    CLRF  DATA_Y
    GOTO  SETUP_COM_DELAY

YP3
    MOVLW B'11001000'    ; BYTE 1
    MOVWF TXREG          ; SEND TO COM PORT
    CALL  DELAY2
    MOVLW B'10000000'    ; BYTE 2
    MOVWF TXREG          ; SEND TO COM PORT
    CALL  DELAY2
    MOVLW B'10011000'    ; BYTE 3
    MOVWF TXREG          ; SEND TO COM PORT
    CLRF  DATA_Y
    GOTO  SETUP_COM_DELAY

YP4
    MOVLW B'11001000'    ; BYTE 1
    MOVWF TXREG          ; SEND TO COM PORT
    CALL  DELAY2
    MOVLW B'10000000'    ; BYTE 2
    MOVWF TXREG          ; SEND TO COM PORT
    CALL  DELAY2
    MOVLW B'10100000'    ; BYTE 3
    MOVWF TXREG          ; SEND TO COM PORT
    CLRF  DATA_Y
    GOTO  SETUP_COM_DELAY

YP5
    MOVLW B'11001000'    ; BYTE 1
    MOVWF TXREG          ; SEND TO COM PORT
    CALL  DELAY2
    MOVLW B'10000000'    ; BYTE 2
    MOVWF TXREG          ; SEND TO COM PORT
    CALL  DELAY2
    MOVLW B'10101000'    ; BYTE 3
    MOVWF TXREG          ; SEND TO COM PORT
    CLRF  DATA_Y
    GOTO  SETUP_COM_DELAY

YP6
    MOVLW B'11001000'    ; BYTE 1
    MOVWF TXREG          ; SEND TO COM PORT
    CALL  DELAY2
    MOVLW B'10000000'    ; BYTE 2
    MOVWF TXREG          ; SEND TO COM PORT
    CALL  DELAY2
    MOVLW B'10110000'    ; BYTE 3
    MOVWF TXREG          ; SEND TO COM PORT
    CLRF  DATA_Y
    GOTO  SETUP_COM_DELAY

YP7
    MOVLW B'11001000'    ; BYTE 1
    MOVWF TXREG          ; SEND TO COM PORT
    CALL  DELAY2
    MOVLW B'10000000'    ; BYTE 2
    MOVWF TXREG          ; SEND TO COM PORT
    CALL  DELAY2
    MOVLW B'10111000'    ; BYTE 3
    MOVWF TXREG          ; SEND TO COM PORT
```

```

        CLRF   DATA_Y
        GOTO  SETUP_COM_DELAY
YP8
        MOVLW B'11001100'      ; BYTE 1
        MOVWF TXREG            ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10000000'      ; BYTE 2
        MOVWF TXREG            ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10000000'      ; BYTE 3
        MOVWF TXREG            ; SEND TO COM PORT
        CLRF  DATA_Y
        GOTO  SETUP_COM_DELAY
YP9
        MOVLW B'11001100'      ; BYTE 1
        MOVWF TXREG            ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10000000'      ; BYTE 2
        MOVWF TXREG            ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10001000'      ; BYTE 3
        MOVWF TXREG            ; SEND TO COM PORT
        CLRF  DATA_Y
        GOTO  SETUP_COM_DELAY
YP10
        MOVLW B'11001100'      ; BYTE 1
        MOVWF TXREG            ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10000000'      ; BYTE 2
        MOVWF TXREG            ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10010000'      ; BYTE 3
        MOVWF TXREG            ; SEND TO COM PORT
        CLRF  DATA_Y
        GOTO  SETUP_COM_DELAY
YP11
        MOVLW B'11001100'      ; BYTE 1
        MOVWF TXREG            ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10000000'      ; BYTE 2
        MOVWF TXREG            ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10011000'      ; BYTE 3
        MOVWF TXREG            ; SEND TO COM PORT
        CLRF  DATA_Y
        GOTO  SETUP_COM_DELAY
YP12
        MOVLW B'11001100'      ; BYTE 1
        MOVWF TXREG            ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10000000'      ; BYTE 2
        MOVWF TXREG            ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10100000'      ; BYTE 3
        MOVWF TXREG            ; SEND TO COM PORT
        CLRF  DATA_Y
        GOTO  SETUP_COM_DELAY
YP13
        MOVLW B'11001100'      ; BYTE 1
        MOVWF TXREG            ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10000000'      ; BYTE 2
        MOVWF TXREG            ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10101000'      ; BYTE 3
        MOVWF TXREG            ; SEND TO COM PORT
        CLRF  DATA_Y
        GOTO  SETUP_COM_DELAY
YP14
        MOVLW B'11001100'      ; BYTE 1
        MOVWF TXREG            ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10000000'      ; BYTE 2
        MOVWF TXREG            ; SEND TO COM PORT
        CALL  DELAY2

```

```
        MOVLW B'10110000'    ; BYTE 3
        MOVWF TXREG          ; SEND TO COM PORT
        CLRF  DATA_Y
        GOTO  SETUP_COM_DELAY
YP15
        MOVLW B'11001100'    ; BYTE 1
        MOVWF TXREG          ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10000000'    ; BYTE 2
        MOVWF TXREG          ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10111000'    ; BYTE 3
        MOVWF TXREG          ; SEND TO COM PORT
        CLRF  DATA_Y
        GOTO  SETUP_COM_DELAY
YP16
        MOVLW B'11001100'    ; BYTE 1
        MOVWF TXREG          ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10000000'    ; BYTE 2
        MOVWF TXREG          ; SEND TO COM PORT
        CALL  DELAY2
        MOVLW B'10111111'    ; BYTE 3
        MOVWF TXREG          ; SEND TO COM PORT
        CLRF  DATA_Y
        GOTO  SETUP_COM_DELAY

;
;-----
; Delay subroutine
;-----
;

DELAY1
        MOVLW 06H
        MOVWF COUNT3
LOOP1 DECFSZ COUNT1,1
        GOTO  LOOP1
        DECFSZ COUNT2,1
        GOTO  LOOP1
        DECFSZ COUNT3,1
        GOTO  LOOP1
        RETURN

DELAY2
        MOVLW 01H
        MOVWF COUNT3
LOOP2 DECFSZ COUNT1,1
        GOTO  LOOP2
        DECFSZ COUNT2,1
        GOTO  LOOP2
        DECFSZ COUNT3,1
        GOTO  LOOP2
        RETURN

END
```