University of Southern Queensland
Faculty of Engineering and Surveying

Electromagnetic Linear Actuator - Design, Manufacture and Control

A dissertation submitted by

Justin John Grimm

In fulfilment of the requirements of

Courses ENG4111 and 4112 Research Project

Towards the degree of

Bachelor of Engineering (Mechatronic)

Submitted: October 2009

Abstract

This dissertation seeks to investigate different methods of obtaining high thrust forces from an electromagnetic linear actuator. It contains two prototypes, one using a variable reluctance path to create physical movement of the actuator and another which uses permanent magnets and coils to create movement.

The permanent magnet prototype contains thirteen coils, each independently controlled by a pulse-width modulated signal and investigation is made into controlling these coils with respect to the position of the actuator. A software programme was written for a microcontroller to control the position which is interfaced to a personal computer via a serial communications link.

Investigation is made into the theory behind both prototypes and simulated models were developed and analysed.

Results were not as high as originally anticipated for Prototype 2 but with further work these forces can be significantly increased.

Disclaimer

University of Southern Queensland

Faculty of Engineering and Surveying

ENG4111 & ENG4112 *Research Project*

Limitations of Use

The council of the University of Southern Queensland, its Faculty of Engineering and Surveying, and the staff of the University of Southern Queensland, do not accept any responsibility for the truth, accuracy or completeness of material contained within or associated with this dissertation.

Persons using all or any part of this material do so at their own risk, and not at the risk of the council of the University of Southern Queensland, its Faculty of Engineering and Surveying or the staff of the University of Southern Queensland.

This dissertation reports an educational exercise and has no purpose or validity beyond this exercise. The sole purpose of the course pair entitled "Research Project" is to contribute to the overall education within the student's chosen degree program. This document, the associated hardware, software, drawings, and other material set out in the associated appendices should not be used for any other purpose: if they are so used, it is entirely at the risk of the user.

Prof Frank Bullen
Dean
Faculty of Engineering and Surveying

Certification

I certify that the ideas, designs and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.

I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

Justin John Grimm

Student Number: 0031210459

______________________________ (Signature)

28th October 2009 (Date)

Acknowledgements

I acknowledge Dr. Sam Cubero from the University of Southern Queensland for creating the topic of this dissertation and for his help and guidance throughout this research project.

Justin Grimm

Contents

Figures

Tables

1. Introduction

This dissertation investigates the design, manufacture and control of two different methods of producing variable thrust forces from an electromagnetic linear actuator.

Linear actuators have been commercially available for many years with the principal of operation being mainly pneumatic, hydraulic and electromechanical. Electromagnetic types have not had mainstream commercial viability due to the relative low forces produced. This research project seeks to improve the magnitude and control of these forces.

Several ideas were implemented including:

- Alternative actuator shaft shapes,
- Vernier scale style positioning of the control coils,
- Control strategy of tubular linear synchronous motor type actuator.

These ideas were integrated into the designs.

The background and history behind the linear actuator and electromagnetics in general is investigated in Section 2. Section 3 introduces the reader to the methodologies behind the experiments conducted in this paper and Sections 4 and 5 discuss the two prototypes developed.

1.1 Statement of Task

The goal of this project is to obtain high, variable thrust forces from a linear actuator comparable to pneumatic or hydraulic actuators using only electromagnetic components. The linear actuator should be similar in size and stroke to a standard 300mm pneumatic unit. The actuator should be fully electrically powered.

The actuator should be fully controllable by a low-cost microcontroller which should be monitored by a programme implemented in the Windows environment.

1.2 Objectives

Key objectives of the project are as follows:

- Design and build an electromagnetic linear actuator;
- The energy source should be purely electrical;
- The principle of operation of the actuator should be electromagnetic;
- The stroke should be similar to an off-the-shelf pneumatic type; and
- Position and thrust forces should be able to be controlled through a microcontroller.

1.3 Abbreviations

The following abbreviations are used in this document:

Abbreviation	Description
DC	Direct Current
DIL	Dual In Line package
GUI	Graphical User Interface
PID	Proportion, Integral and Derivative
PM	Permanent Magnet
PWM	Pulse Width Modulation
SCI	Serial Communications Interface
USART	Universal Synchronous/Asynchronous Receiver/Transmitter

Table 1-1: Abbreviations

1.4 Definitions

General Term	Definition
A/D Converter	Analogue to Digital Converter
Flux Path	The path the lines of force in an electric or magnetic field take
Magnetic Field	A field or force associated with a changing magnetic field
Permeability	The ability of a material to allow a substance (magnetic flux) to pass through it

General Term	Definition
Reluctance	The resistance of a closed circuit magnetic loop to magnetic flux
TO-220	A type of package that MOSFETS and other integrated devices are housed
Weber	Unit of magnetic flux

Table 1-2: Definitions

2. Background

2.1 Electromagnetism

2.1.1 Brief History

Magnetism is a phenomenon whereby a magnetised material can attract or repel other materials at a distance as well as cause other effects such as generating the flow of electricity in electrical wires. Only specific materials can be magnetised such as nickel, iron, cobalt and gadolinium. Magnetism is widely used to convert mechanical energy to electrical energy.

The first human interaction with magnetism probably occurred in pre-historic times in the form of static electricity and Lodestones. The first definite record is from Thales of Miletus about 585 BC in describing Lodestone where he said that ‘Lodestone attracts iron because it has a soul’ (Fowler, 1997). Since this time there have been many theories on the source of magnetism, some scientifically based whilst others have been based on divine influences such as the gods.

William Gilbert (1540-1603) was the first person to study the science of magnetism in earnest and published his book on the subject (De Magnete) in 1600. He was the first to understand that the Earth is in fact a magnet and produced experiments to prove his theory.

2.1.1.1 The Lodestone

The Lodestone is a naturally occurring magnetic rock and is a form of magnetite which in turn is a form of iron ore. The magnetite is thought to have been magnetised during lightning strikes. It is thought that the Lodestone was first discovered on Mount Ida in Crete.

Figure 2-1: Lodestone *(Geology.com, 2009)*

It is popular belief that Lodestone was first discovered around 2700 BC. The Chinese were apparently the first to exploit the properties of lodestone and there is evidence of a primitive non-navigational compass being used for ritual ploughing in around 100 AD (Magnetism Group, Physics Dept, Trinity College Dublin). By the year 1040 there is evidence that the Chinese used a magnetic iron leaf for the purpose of navigation.

2.1.1.2 Ferromagnetic Materials and the Magnetic Domain Theory

There a three classes of magnetic materials; diamagnetic, paramagnetic and ferromagnetic.

The spin of an electron around the nucleus of an atom combined with its angular momentum creates a magnetic moment which in turn creates a magnetic field. In most non-ferromagnetic materials the electrons are in pairs and the magnetic moments cancel out. Therefore a net magnetic moment from an atom can only occur with elements that contain electrons in unpaired spins. (Ferromagnetism, 2009).

Ferromagnetic materials contain many atoms with electrons in unpaired spins. They also contain areas (domains) of up to 1mm in size which are naturally created when the atoms bond to form the material. These domains contain billions of atoms which have their magnetic moments naturally aligned. In an un-magnetised state, these ferromagnetic materials contain domains with a random orientation and the magnetic effect of each domain is essentially cancelled out. When the material is magnetised most of these domains are aligned thereby adding the magnetic effect of each domain. This produces a net magnetic field in the material (Ferromagnetism, 2009).

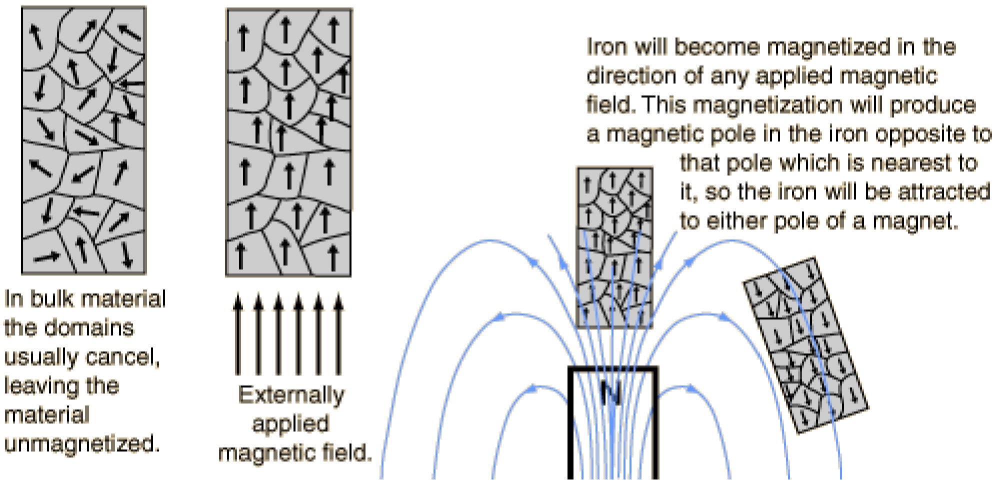


Figure 2-2: Domains in magnetic materials *(Nave, 2006)*

Magnetic Properties of Ferromagnetic Materials					
Material	Treatment	Initial Relative Permeability	Maximum Relative Permeability	Coercive Force (oersteds)	Remanent Flux Density (gauss)
Iron, 99.8% pure	Annealed	150	5000	1	13,000
Iron, 99.95% pure	Annealed in hydrogen	10,000	200,000	0.05	13,000
78 Permalloy	Annealed, quenched	8,000	100,000	0.05	7,000
Super permalloy	Annealed in hydrogen, controlled	100,000	1,000,000	0.002	7,000

Magnetic Properties of Ferromagnetic Materials					
	cooling				
Cobalt, 99% pure	Annealed	70	250	10	5,000
Nickel, 99% pure	Annealed	110	600	0.7	4,000
Steel, 0.9% C	Quenched	50	100	70	10,300
Steel, 30% Co	Quenched	...	...	240	9,500
Alnico 5	Cooled in magnetic field	4	...	575	12,500
Silmanal	Baked	...	...	6,000	550
Iron, fine powder	Pressed	...	...	470	6,000

Table 2-1: Properties of Ferromagnetic Materials *(Georgia State University, 2005)*

2.1.1.3 Diamagnetic and Paramagnetic Materials

Diamagnetic materials have the effect of slightly repelling a magnet due to the applied magnetic field altering the orbital velocity of the electrons around the nuclei and thus changing the magnetic dipole moment in a direction opposing the external field. Diamagnetic materials have a magnetic permeability of less than μ_0 (Diamagnetism, 2009).

Paramagnetic materials exhibit a slight magnetic attraction to an external magnetic field whilst in the presence of the field but do not retain any magnetisation when the field is removed. This is due to thermal motion causing the spins to become randomly oriented at normal room temperature. These materials have a low permeability but which is greater than μ_0 (Paramagnetism, 2009).

Material	Relative Permeability
Paramagnetic	
Iron oxide (FeO)	721

Material	Relative Permeability
Iron ammonium alum	67
Uranium	41
Platinum	27
Tungsten	7.8
Cesium	6.1
Aluminium	3.2
Lithium	2.4
Magnesium	2.2
Sodium	1.72
Oxygen gas	1.19
Diamagnetic	
Ammonia	0.74
Bismuth	-15.6
Mercury	-1.9
Silver	-1.6
Carbon (diamond)	-1.1
Carbon (graphite)	-0.6
Lead	-0.8
Sodium chloride	-0.4
Copper	0
Water	0.09

Table 2-2: Properties of Diamagnetic and Paramagnetic Materials *(Georgia State University, 2005)*

2.1.2 Current Electromagnetic Theory

Magnetism and electricity are understood to be fundamentally interlinked as described in Einstein's theory on special relativity. One cannot exist without the other and a phenomenon observed to be purely electrical by one observer may appear to be electrical and magnetic by another observer in a different reference frame.

An excellent approximation of magnetic fields (ignoring some quantum effects) is given by Maxwell's equations which state that magnetism is seen whenever electrically charged particles are in motion, for example in electric current or from the electrons orbiting around an atom's nucleus. They can also arise from quantum-mechanical spin which produce magnetic dipoles. (Wikipedia, 2009).

Magnetic fields have the effect of creating a force which is defined as follows:

$$\mathbf{F} = q(\mathbf{v} \times \mathbf{B})$$

Where:

q is the electric charge of the particle

$\mathbf{v}$ is the velocity vector of the particle

$\mathbf{B}$ isthe magnetic field

Since this equation is a cross-product, the resultant force is perpendicular to both the magnetic field and the velocity of the particle. A tool for determining the direction of the resultant force vector is by using the right-hand rule where the direction of the thumb is the resultant force vector, the index finger points along the particles velocity vector and the middle finger follows the magnetic field vector.

2.1.2.1 Magnetic Field around a Wire

A magnetic field forms around a long wire when current is passed through it. The magnetic field is in the form of concentric circles around the wire and can be approximated using Ampere's Law:

$$B = \frac{\mu_0 I}{2\pi r}$$

Where:

$\mu_0 = 4\pi * 10^{-7}$

I is the current in the wire in coulombs per second

r is the radial distance from the wire

B is the magnetic field

Figure 2-3 depicts the magnetic field around a current-carrying wire.

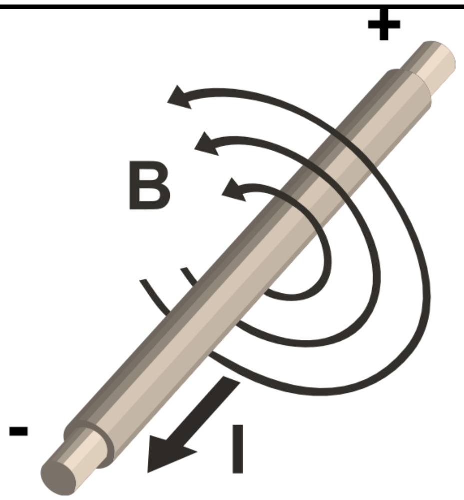


Figure 2-3: Magnetic Field around a wire *(Stannered, 2007)*

2.1.2.2 Solenoids

A solenoid is defined as a series of loops of a single wire (see Figure 2-4). When an electric current is applied to this wire the magnetic fields circling the wire along its length tend to reform into a magnetic field that moves up the centre of the solenoid, around the outside and back to the start of the solenoid (see Figure 2-5). Due to Weber's law there are no 'sources' or 'sinks' in a magnetic field, therefore magnetic fields are always a loop.

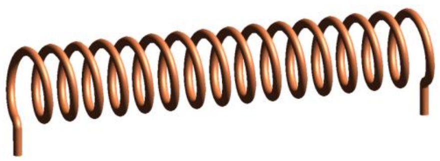


Figure 2-4: Solenoid *(Zureks, 2008)*

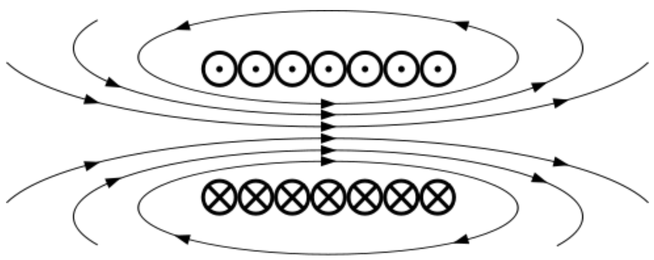


Figure 2-5: Magnetic field created by a solenoid *(Nogueira, 2006)*

The magnetic field strength in the solenoid can be approximated from Ampere's Law as follows:

$$B = \frac{\mu_0 IN}{L}$$

Where:

$\mu_0 = 4\pi * 10^{-7}$

I is the current in the wire in coulombs per second

N is the number of turns in the solenoid

L is the axial length of the solenoid

The magnetic field in a solenoid is greatly increased when an iron core is added. This is due to the magnetic domains of the ferromagnetic material lining up with the driving magnetic field (Georgia State University, 2005). The relative permeability of the ferromagnetic material k is the magnifying factor for the magnetic field. This value tells us how much greater the magnetic field is within the material compared to the magnetic field in an air gap.

When the magnetic path is fully filled with a ferromagnetic material except for a small air gap, the magnetic field across the air gap causes a force which tends to close that air gap. This force can be calculated and used in industry to move a movable iron core which can operate a set of electrical contacts or a valve actuator. Common in industry is the electromagnetic relay (see Figure 2-6).

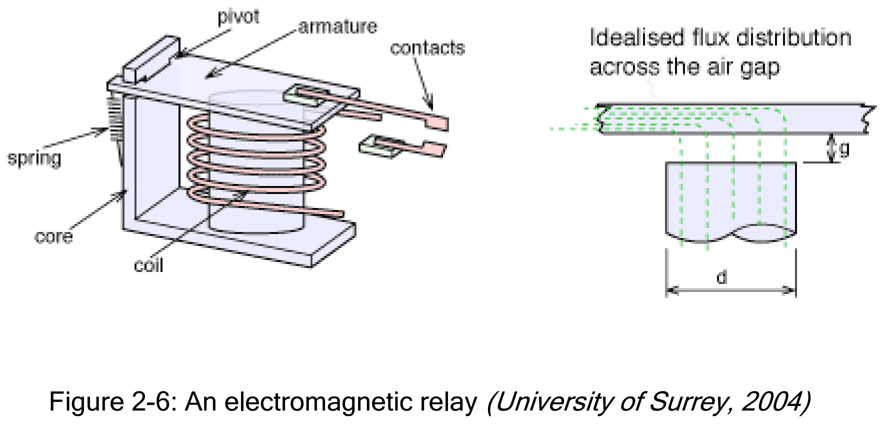


Figure 2-6: An electromagnetic relay *(University of Surrey, 2004)*

The magnetic field across the air gap can be calculated as follows (ignoring the reluctance in the ferromagnetic core):

$$B_g = \frac{{F_m}^2 \mu_0}{g}$$

Equation 2-1

Where:

B_g is the magnetic field in the air gap (Tesla)

F_m is the magnetomotive force $= NI$

g is the air gap

The force across the air gap can be calculated as follows:

$$F = \frac{{B_g}^2 A}{2\mu_0}$$

Equation 2-2

Where:

F is the force in the air gap

A is the cross sectional area of the air gap

2.1.2.2.1 Determining the Direction of the Magnetic Field

To determine the direction of the magnetic field in a solenoid the right-hand rule is used. The fingers of the right hand are wrapped around the coil in the direction of the current flow. The direction of north is determined by the direction that the thumb is pointing.

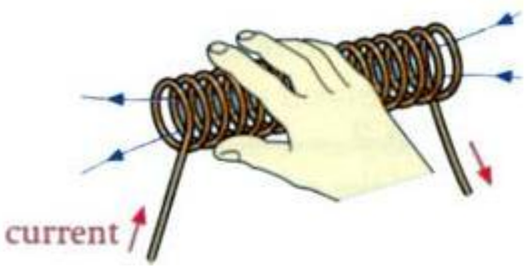


Figure 2-7: The right-hand rule *(Magnetism Hand Rules)*

2.1.2.3 Coil Design

Two of the factors determining the strength of a magnetic field generated by a solenoid are:

- The number of turns in the coil, and
- The current in the coil.

These two parameters are multiplied together to give the magnetomotive force or the current density of the coil. For any given coil volume there is a limit on the current density in that coil. For instance one can increase the number of turns in a coil by decreasing the cross sectional area of the wire that is used but decreasing the wire size also increases its resistance therefore reducing the current flow in the coil for the same applied voltage potential.

2.1.2.3.1 Winding Factor

Small air gaps are formed when wire of a circular cross section is used for winding coils. Figure 2-8 shoes a cross section of a tightly would coil and the air gaps between the windings. Figure 2-9 shows how this airspace can be calculated with respect to the wire radius.

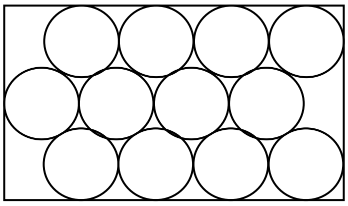


Figure 2-8: Cross section of coil.

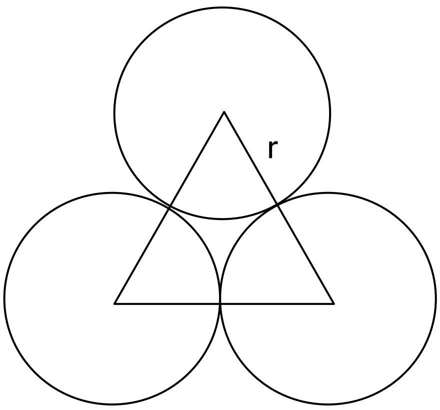


Figure 2-9: Finding the air space between windings.

The area of the equilateral triangle is:

$$A_t = \frac{2r^2\sqrt{3}}{4}$$

The area of a 60 degree circular sector is:

$$A_s = \frac{\pi r^2}{6}$$

The area of the air space in the middle of the triangle is:

$$\begin{aligned} A &= A_t - 3A_s \\ &= \frac{2r^2\sqrt{3}}{4} - \frac{3\pi r^2}{6} \end{aligned}$$

Therefore, since there are two air spaces per winding:

$$A_a = \frac{2r^2\sqrt{3}}{2} - \pi r^2$$

Equation 2-3

Since the air gap equation has terms containing the square of the radius, it can be shown that using multiple stands of smaller diameter wire in parallel results in a higher current density since there is less wasted air space between successive windings.

2.2 Linear Actuators

2.2.1 Introduction

Linear actuators are devices that convert different forms of energy into linear motion as opposed to rotary motion as in electric motors. They are used in many industries such as automotive, industrial and consumer goods. They vary in application from accurate linear measuring devices to producing high force hydraulic actuators in earthmoving machines.

2.2.2 Different Types and Uses

Linear actuators fall in to several main categories, usually classified by their energy source:

Energy Source	Advantages	Disadvantages	Typical example
Mechanical	Cheap, mechanical advantage	Requires manual labour to actuate	Car jack
Electro-mechanical	Easy to control	Susceptible to mechanical failure due to moving parts	Automotive electric chair adjustors
Pneumatic	Reasonably cheap, medium force output, air freely available	Require pressurised air to operate	Industrial control valve
Hydraulic	High force output	Require pressurised hydraulic energy source	Lift actuator for front-end loader bucket
Electromagnetic	Low friction loss and minimal moving parts	Difficult to control position, low force output	Solenoid valve

Table 2-3: Different types of linear actuators.

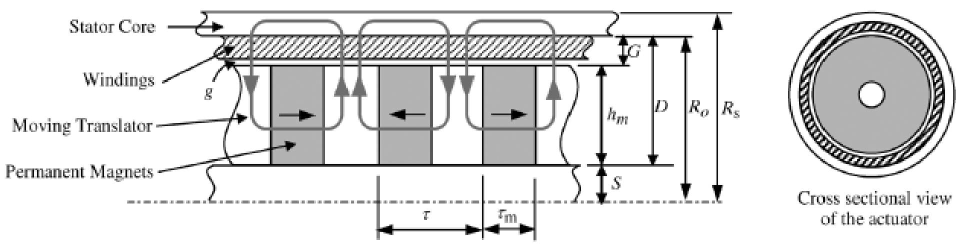
A typical pneumatically powered linear actuator is shown in Figure 2-10.

Figure 2-10: Typical Pneumatic Actuator *(Machine-Design.com)*

2.2.3 Tubular Linear Synchronous Motor

Some research has been done in the area of tubular linear synchronous motors and discussions in this section will be centred on the research paper produced by Haiwei Lu, Jianguo Zhu, Zhiwei Lin, and Youguang Guo from the Faculty of Engineering, University of Technology, Sydney, NSW 2007, Australia. Their paper is titled 'A Miniature Short Stroke Linear Actuator—Design and Analysis'.

Their design was aimed at the miniature robotics market and therefore has a short stroke length of 6mm. The design uses permanent magnets and is arranged similar to a brushless DC motor but laid out flat so as to provide linear motion instead of rotational. The resultant force from their experiments was around 2.5 N.



DIMENSIONS OF THE ACTUATOR UNIT: mm

Stator Outer Radius (R_s)	16
Stator Inner Radius (R_o)	13.75
Stator Length (L_s)	34
Air Gap (g)	0.4
Translator Radius (h_m+S)	11.25
Translator Length (L_a)	20
Shaft Radius (S)	2.5

Figure 2-11: Half-section of the actuator configuration *(Haiwei Lu, 2008)*

In order to keep the package volume small, the design looked at using back EMF sensing techniques to control the driving current to each coil for position and force control. Force calculations were derived from Lorentz force law and the electromagnetic force of the phase coils under the poles determined analytically and then confirmed numerically using finite element methods.

As far as the goal of this project is concerned it is possible that a similar design to this, only on a larger scale, can be used. A potential candidate can be designed fitting the physical size criteria and then similar equations as used in the project by Lu et al.

2.3 DC Motor Control

2.3.1 Overview

DC motor control is the term given to the task of controlling the speed, torque and/or force produced by a DC motor. DC motors are generally classified as either:

- Internally commutated, or
- Externally commutated.

Internally commutated DC motors contain brushes which are used to alternate the polarities of the internal electromagnets contained in the rotor relative to the angle of the rotor against the stator of the motor. As the rotor rotates the polarity of the electromagnet is reversed to facilitate the push-pull interaction between the electromagnets and the permanent magnets contained in the stator. The advantages of internally commutated DC motors are low initial cost and simple control. The disadvantage is the higher maintenance costs due to the friction between the brushes and the commutator.

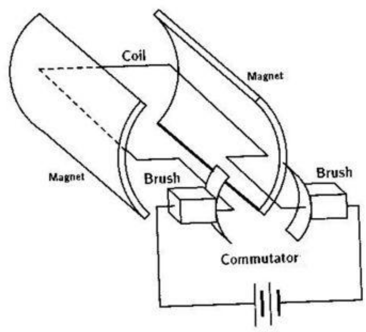


Figure 2-12: Simple Brushed DC Motor *(Laboratory for Intelligent Mechanical Systems)*

Externally commutated DC motors generally have the permanent magnets contained as part of the rotor and the electromagnets contained in the stator. Commutation is achieved in the control circuitry. Advantages of the externally commutated DC motor are in their reliability due to minimal friction, disadvantages include the cost of initial purchase and the elaborate control strategy required.

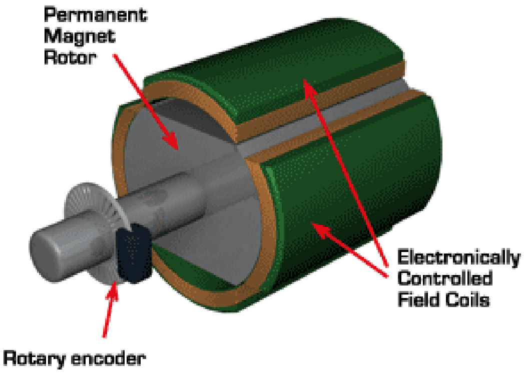


Figure 2-13: Simple Brushless DC Motor *(Zero Emission Vehicles Australia)*

Prototype two of this dissertation uses a variation on the brushless DC motor where the coils are effectively rolled out and laid flat and the permanent magnet armature is moved laterally to the coils by controlling the coil electromagnetic polarity with respect to its position.

2.3.2 Microcontroller

A microcontroller is a device that integrates a microprocessor and a range of peripheral devices such as multiple USARTs, A/D converters and PWM controllers into the one package. They are useful as a low cost controlling device for the purposes of controlling a dedicated function or task.

Manufacturers of microcontrollers include the following:

- Microchip,
- Atmel,
- Zilog, and
- Motorola

Microcontrollers generally have several ports that can provide TTL level inputs and outputs to interface to the 'real world'. It is through these ports that devices such as h-bridge controllers can be driven and where inputs such as position feedback are interfaced.

2.3.3 H-Bridge

An H-Bridge is a term used to describe an electronic motor controller which controls the direction and amplitude of current through a DC motor.

2.3.3.1 Basic H-Bridge controller

Figure 2-14 shows the general arrangement of the H-Bridge controller.

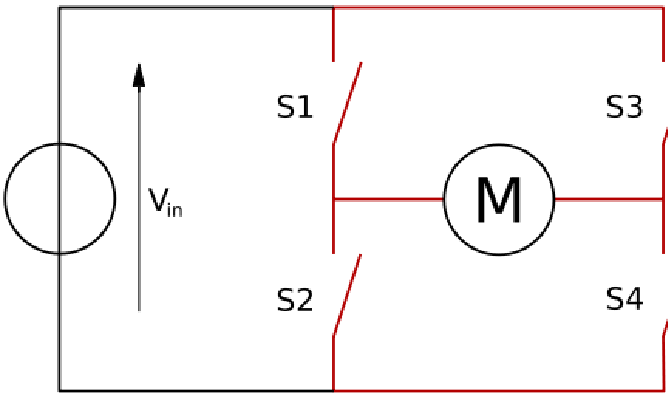


Figure 2-14: H-Bridge configuration *(Buttay, H-Bridge, 2006)*

Figure 2-15 shows the basic states of the H-Bridge. This shows how different switch positions can control the current direction.

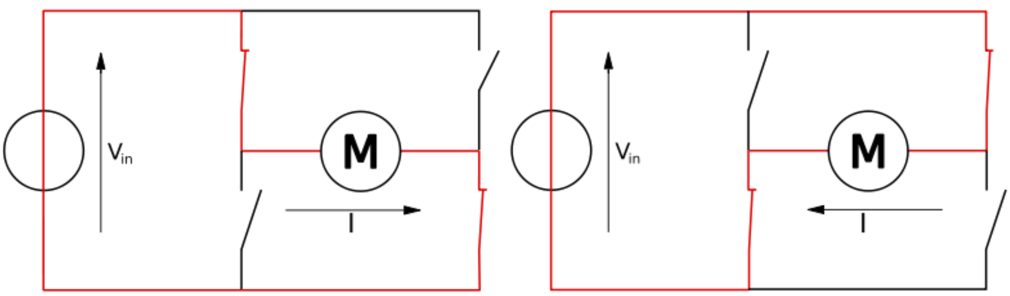


Figure 2-15: Basic states of the H-Bridge *(Buttay, H bridge.svg, 2006)*

By controlling the switches S1 to S4 the direction of current through the motor can be controlled. Table 2-4 shows the resultant control with reference to difference switch positions. All positions not shown on this list are illegal and may cause a short circuit.

S1	S2	S3	S4	Result
Off	Off	Off	Off	Motor off
On	Off	Off	On	Current flows in motor in the forward direction
Off	On	On	Off	Current flows in motor in the reverse direction
Off	On	Off	On	Brake motor
On	Off	On	Off	Brake motor

Table 2-4: H-Bridge control.

2.3.3.2 MOSFETs

Control of the current amplitude can be achieved by replacing the switches in the circuit in Figure 2-14 with high speed electronic switches such as MOSFETs. Figure 2-16 depicts the circuit symbol of a P-channel enhancement MOSFET.

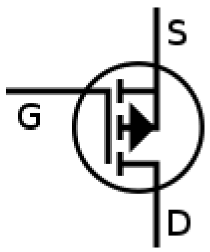


Figure 2-16: P-channel enhancement MOSFET *(jjbeard, 2006)*

Pin	Name
G	Gate
S	Source
D	Drain

Table 2-5: MOSFET pins.

MOSFETs are devices that can control relatively high currents by varying the voltage signal to its gate. MOSFET is an acronym for Metal-Oxide Semiconductor Field-Effect Transistor.

The gate in a MOSFET connects to a plate that is separated from a semiconductor substrate by an insulator such as metal oxide or more recently a polycrystalline silicon layer. This arrangement is similar to a capacitor.

For an n-type MOSFET, when the gate is at a positive voltage with respect to the semiconductor substrate an electric field is generated which tends to repel the holes in the semiconductor. As the voltage difference increases, minority carrier electrons are attracted from the source towards the semiconductor interface creating an electron inversion layer. This allows a current to flow between the source and drain regions when a voltage difference is applied. Figure 2-17 shows the arrangement for an n-type MOSFET. A p-type device is similar to an n-type except the voltages and charges are opposite and the main substrate is made of n-type semiconductor material.

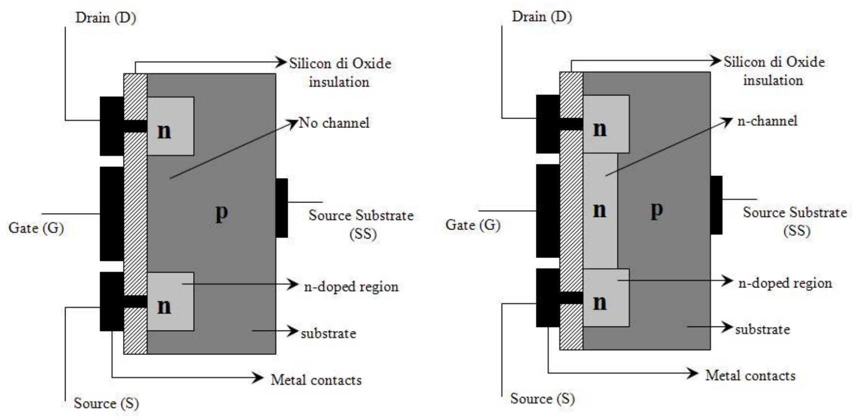


Figure 2-17: N-type MOSFET (a) Off state, (b) On state *(Wikipedia)*

MOSFETs are typically used as the switching devices for h-bridge circuits because of their high switching speed and low control current. If the devices are switched fast enough the average current can also be controlled giving the h-bridge the ability to control the current flow as well as the direction.

2.3.3.3 Integrated H-Bridge Devices

Devices are now available that contain a full h-bridge on a chip. These devices feature the following:

- A pulse-width modulated input pin that switches the internal h-bridge MOSFETs,
- Over current protection,
- Current sensing,
- Brake input pin, and a
- Direction pin.

Figure 2-18 shows a typical integrated h-bridge controller.

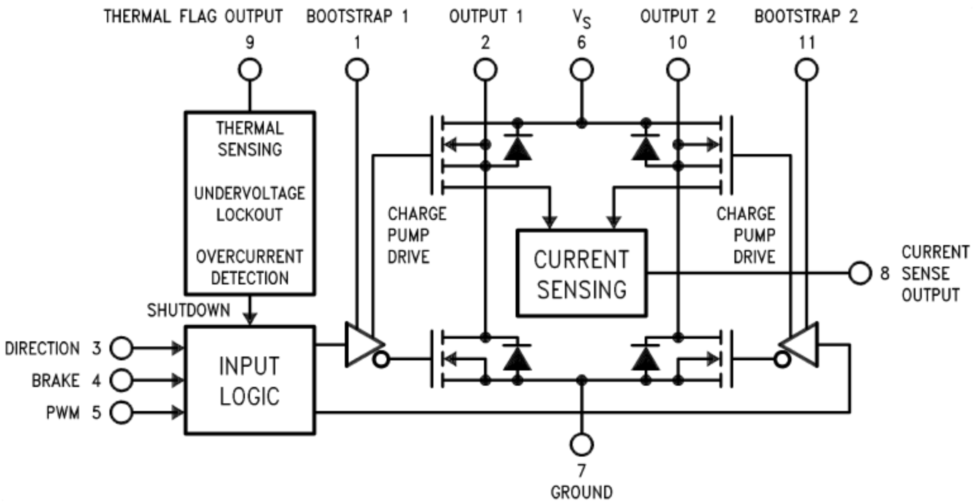


Figure 2-18: LMD18200 functional diagram *(National Semiconductor, 2005)*

2.3.4 Pulse Width Modulation

Pulse width modulation (PWM) is a technique generally used to allow a digital device to generate a simulated analogue signal. It is used extensively where the control of an analogue signal is required by a digital device, such as a microcontroller. The h-bridge circuit often utilises PWM to control the average current to the DC motor.

2.3.4.1 PWM Theory

PWM is a signal which is either on or off. It has a cycle period which is usually of a fixed time. It also has a duty cycle which refers to the percentage of time that the signal is turned on in the cycle period.

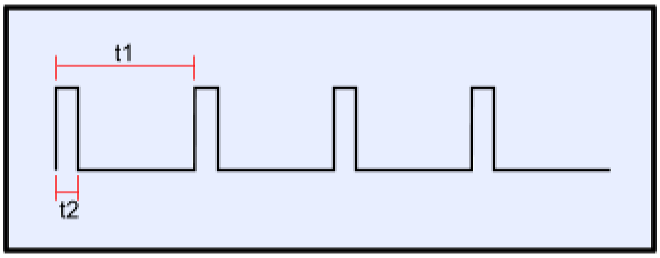


Figure 2-19: The PWM signal *(PCB Heaven, 2008)*

Figure 2-19 shows the structure of the PWM signal. Time t1 is the cycle period and t2 is the duty cycle. Figure 2-20 shows three different duty cycles, 10 percent, 40 percent and 90 percent. A 10 percent duty cycle means that the signal is high for 10 percent of the cycle period and similarly a 90 percent duty cycle means that the signal is on for 90 percent of the cycle time. The average voltage for the signal corresponds to the duty cycle. I.e. a 10 percent duty cycle gives an average voltage of 10 percent of the full-scale signal voltage.

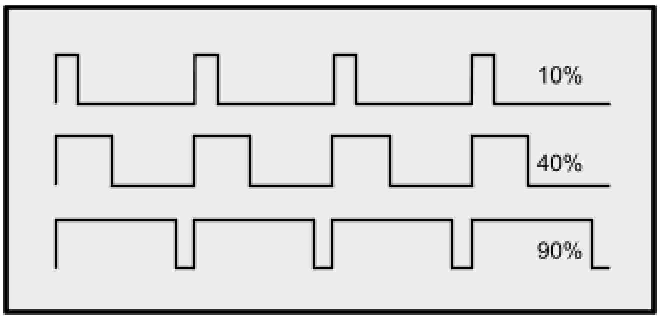


Figure 2-20: Different duty cycles *(PCB Heaven, 2008)*

When a suitable filter is applied to the PWM signal, an analogue value can be approximated. Figure 2-21 shows a representative filter circuit and Figure 2-22 shows the resulting voltages across the capacitor after filtering.

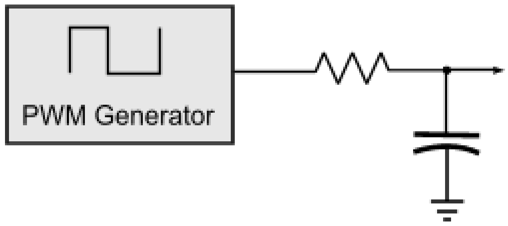


Figure 2-21: Filter circuit *(PCB Heaven, 2008)*

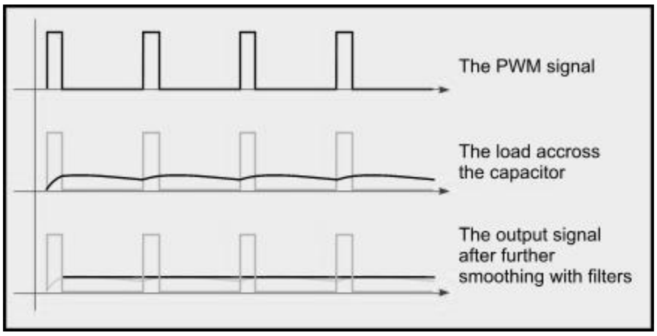


Figure 2-22: Results of filtering *(PCB Heaven, 2008)*

By using these techniques a relatively simple PWM signal from a microcontroller can control large currents in power motor circuits.

2.3.4.2 PWM Generation

Many microcontrollers come with on-board hardware PWM generators that require little in the way of setup and use minimal software resources when the programme is running. An example of this is the PIC16F877A microcontroller from Microchip Inc.

This device has two on-board modules which can be configured for either capture, compare or for PWM. In PWM mode the user sets the cycle period in the initialisation routine of the programme and sets the duty cycle as required in the main section.

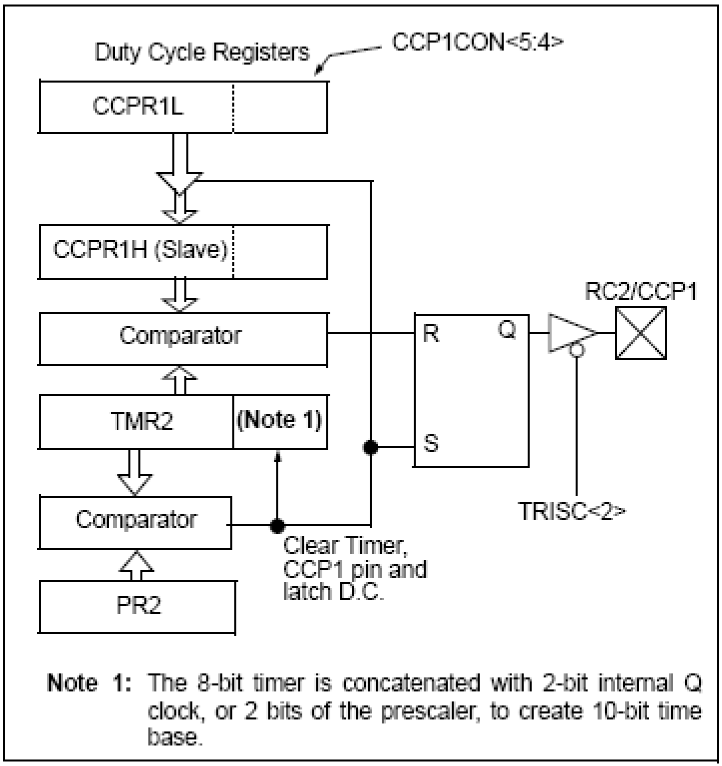


Figure 2-23: PIC16F877A PWM block diagram *(Microchip Inc, 2009)*

2.3.5 Feedback Control

Since the current through the coils of a DC motor can be controlled by an h-bridge driver circuit and the speed (or position) of the motor can be easily measured, a control scheme can be implemented to control the speed (or position) of the motor.

A typical feedback control system performs the following actions:

- Measure the speed (or position),
- Determine the error between the speed or position and the setpoint,
- Pass the error through a PID control algorithm and determine an appropriate output signal to reduce the error, and
- Drive the output to the motor.

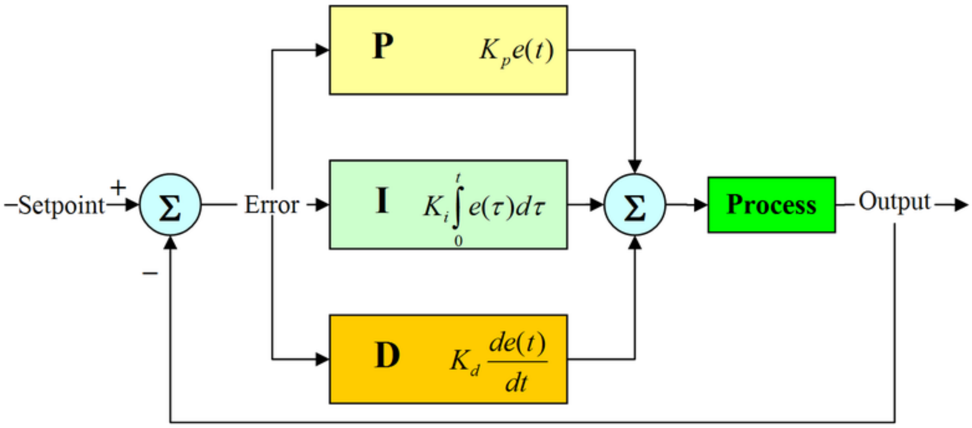


Figure 2-24: PID controller (SilverStar, 2006).

3. Methodologies

The methodology utilised in this project was:

- Research current advances in using electromagnetic components in linear actuators,
- Settle on some design ideas,
- Simulate the design ideas with ease of manufacturing in mind,
- Finalise the design selections based on the simulations,
- Manufacture selected designs,
- Test the manufactured items, and
- Report on the results.

3.1 Simulation Software

Maxwell SV version 3.1.04 was used throughout this project to simulate and refine the designs.

Maxwell SV is a free subset of Maxwell 2D and is an electromagnetic field simulation software for the design of electromagnetic and electromechanical devices using the finite element method.

4. Prototype 1 – Solenoid Type Actuator

Figure 4-1: Prototype 2.

Prototype 1 was developed to test the force output from a solenoid/slug type electromagnetic linear actuator. This design uses no permanent magnets but relies on the strength of the electromagnetic field generated by its coils. A relatively low reluctance path has been designed for the electromagnetic field except for a purposely designed air gap. It is the closing of this air gap by the forces generated from the magnetic field that generates movement in the actuator.

Although the prototype was designed, simulated and manufactured, the unit was not assembled and tested due to time restraints.

4.1 Design

There are three basic items that determine the strength of a magnetic field in an electromagnet:

- The number of turns in the solenoid,
- The current in the coil,
- The reluctance of the flux path.

The intention of prototype 1 was to maximise the current density in the solenoid coils whilst providing a low reluctance path for the magnetic field to travel. The design was investigated using Maxwell simulation software and a design was then chosen and manufactured.

4.1.1 Analytical Design

To calculate the force in a solenoid design with a steel slug the following equation can be used:

$$F = \mu \frac{dB}{dX}$$

(Nikunj shah, Rob Jamieson, 2006) Equation 4-1

The magnetic field can be calculated by:

$$B = \frac{\mu_o NIR^2}{2} \left\{ \frac{1}{{R^2 + x^2}^{\frac{3}{2}}} + \frac{1}{{R^2 + x^2 - 2sx + s^2}^{\frac{3}{2}}} \right\}$$

(Cabrillo College) Equation 4-2

The change in the magnetic field can be calculated by:

$$\frac{dB(x)}{dX} = \frac{\mu_o NIR^2}{2} \left\{ \frac{3x}{{R^2 + x^2}^{\frac{5}{2}}} + \frac{3(x-s)}{{R^2 + x^2 - 2sx + s^2}^{\frac{5}{2}}} \right\}$$

(Cabrillo College) Equation 4-3

Where

μ_0 = permeability of free space $= 4\pi * 10^{-7}$ H/m

μ = permeability $= 875 * 10^{-6}$ H/m

N = number of turns

I = current = 1.2 A

L = length $= 0.115 * 2 = 0.230$ m

x = position from the centre of the left coil

s = position of centre of right coil

r = radius = 0.035 m

B = magnetic field strength

The resultant force for each position for Prototype 1 was calculated using the above equations and a graph was generated as follows:

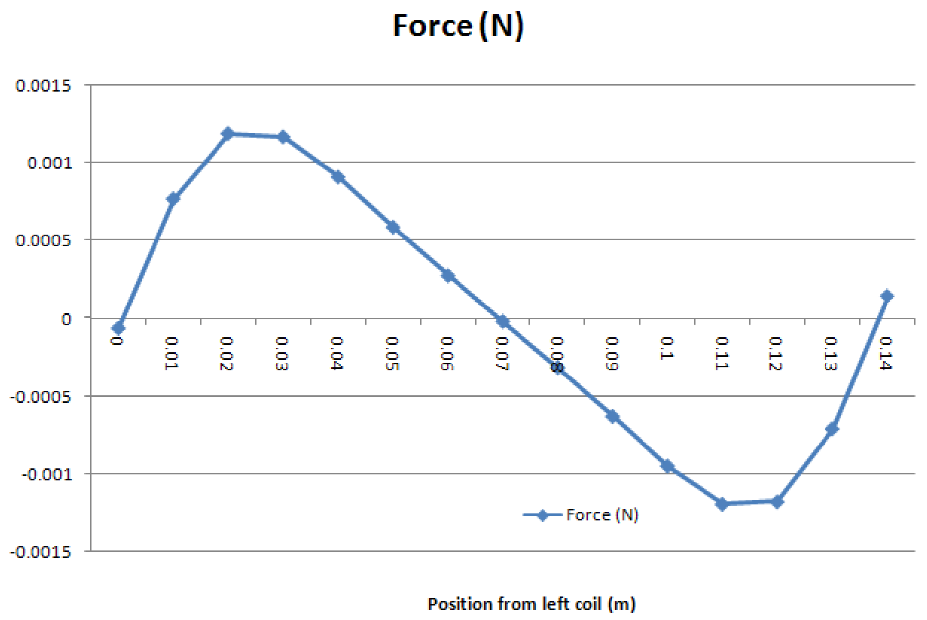


Figure 4-2: Graph showing analytical results.

These results are low compared to the simulation results due to the calculation not accounting for a ferromagnetic reluctance path.

4.1.2 Coil Design

The coils were designed to utilise the maximum available volume. A nylon coil former was designed and manufactured to contain the coil for ease of assembly.

4.1.3 Designs Considered

It can be shown that the force generated in the air gap within the path of a magnetic field is inversely proportional to the square of the width of the air gap and directly proportional to the cross-sectional area of the air gap. Therefore this design sought to:

- Minimise the air gap width whilst still providing adequate actuator stroke length, and
- Maximise the cross-sectional area of the air gap within reasonable limits.

Multiple designs were considered and tested using the simulation software.

4.1.3.1 Tapered Actuator Design

A tapered actuator allows greater lateral movement of the actuator whist minimising the air gap width. This design is depicted below. Note that the coils are coloured pink and the actuator is coloured red.

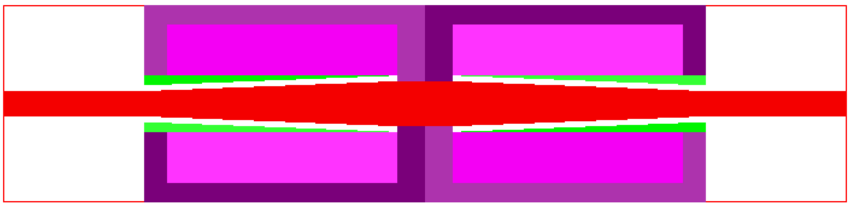


Figure 4-3: Tapered actuator design (lateral section).

4.1.3.2 Stepped Actuator Design

This design allows for a low reluctance path for the magnetic field and maximises the effective cross-sectional area of the air gap. This design was chosen for Prototype 1 due to its high force output and its ease of manufacture.

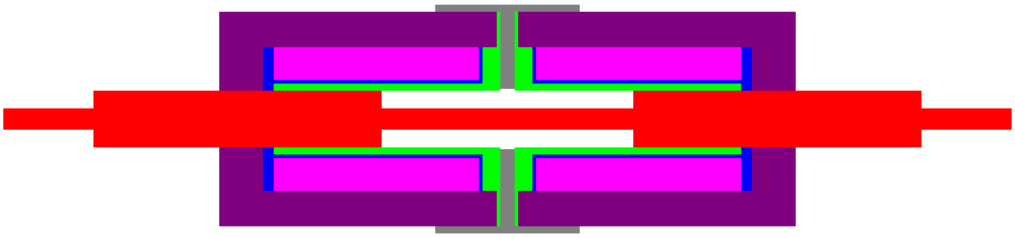


Figure 4-4: Stepped actuator design (lateral section).

4.2 Simulation

4.2.1.1 Current Density

The calculations below determine the current density in the coil for the purposes of simulation.

Coil window area:

$$A_c = 114 \ \ 18 \ \ = 2052 \text{ mm}^2$$

Wire size:

$$A_w = \pi r^2 = \pi(0.25)^2 = 0.1963 \text{ mm}^2$$

Theoretical turns in coil window:

$$T_t = \frac{2052}{0.1963} = \ 10453$$

Air space per winding:

$$A_a = \frac{2r \ ^2 \sqrt{3}}{2} - \pi r^2 = \frac{0.5 \ ^2 \sqrt{3}}{2} - \pi(0.25)^2 = 0.0202 \text{ mm}^2$$

Therefore area taken by each winding:

$$A_t = 0.1963 + 0.0202 = 0.2165 \text{ mm}^2$$

Calculated turns:

$$T_c = \frac{2052}{0.2165} = 9478$$

Current carrying capacity of 0.2mm^2 copper wire:

$$I_{CCP} = 1.2 \text{ A}$$

Therefore current density:

$$I_D = 9478 \;\; 1.2 \quad = 11374 \text{ At}$$

4.2.1.2 Simulation Report

This section shows the simulation parameters and mesh results for the first simulation for the stepped actuator design. Other simulations were carried out on this design with a changing actuator position. Only the left hand coil was energised.

Software	Maxwell SV version 3.1.04
Solver	Magnetostatic
Drawing	RZ Plane
Solver Passes Required	3
Triangles	1659
Energy Error	0.63%
Source 1 (right coil)	0 A
Source 2 (left coil)	11374 At

Table 4-1: Simulation parameters.

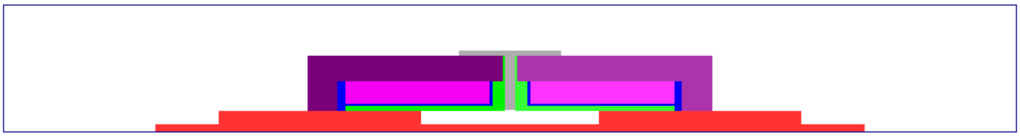


Figure 4-5: Design 3 (lateral half section).

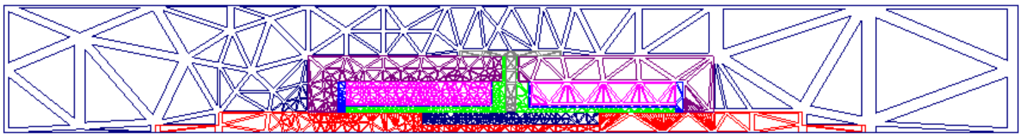


Figure 4-6: Mesh for Design 3 (lateral half section).

4.2.1.3 Simulation Results

The simulation showed a resultant maximum force on the actuator of 233 N at -60mm. When the actuator was set at a position of 90mm it can be seen from Figure 4-8 that the coil tends to attract the right-most actuator lobe.

Figure 4-7 shows the flux path generated by the coil when the actuator is centred. The resultant force at this position is 140 N.

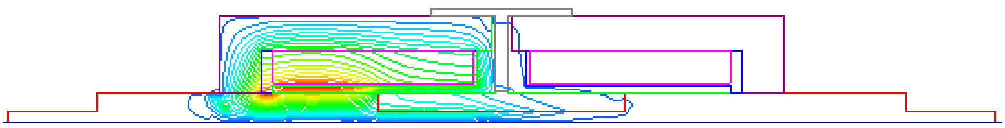


Figure 4-7: Flux path when the actuator is centred.

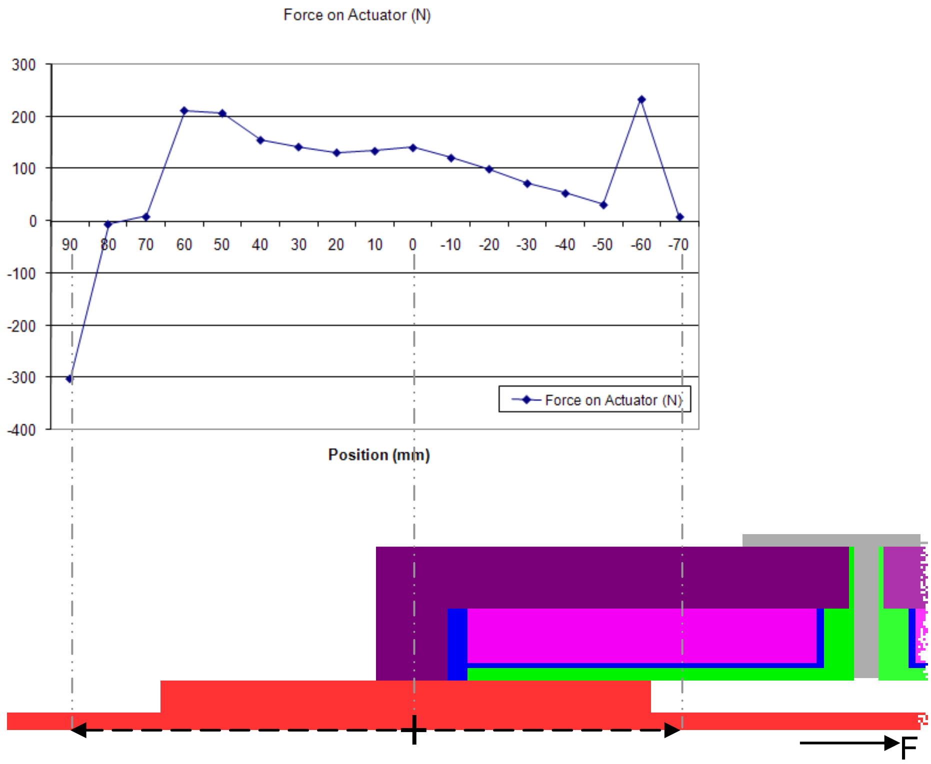


Figure 4-8: Graph of position versus force for Prototype 1 simulation.

4.3 Construction

Prototype 1 was constructed from the following components:

Item	Part	Material	Quantity	Purpose
1	Actuator	AISI 1010 Steel	1	Moving actuator
2	Stator	AISI 1010 Steel	2	Path for electromagnetic flux
3	Bobbin	Nylon	2	To contain coil windings
4	Solenoid	0.2mm2 enamelled copper wire	2	To generate electromagnetic flux
5	Inner Stator	AISI 1010 Steel	2	Path for electromagnetic flux
6	Joiner	Aluminium	1	Join the two halves of the assembly

Table 4-2: Part list for Prototype 1.

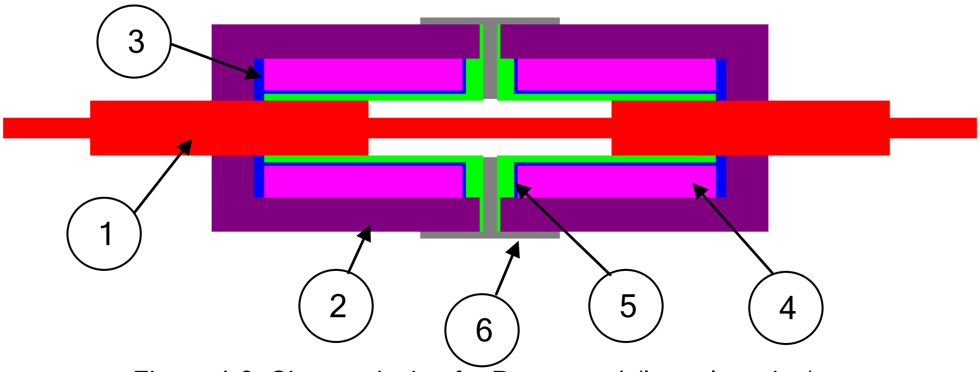


Figure 4-9: Chosen design for Prototype 1 (lateral section).

Figure 4-10: Machined parts for Prototype 1.

All parts were machined at the USQ mechanical workshop. The coils were hand wound with 5900 turns each of $0.2mm^2$ enamelled copper wire using the apparatus shown below.

Figure 4-11: Coil winding apparatus.

4.4 Conclusions

This prototype was not assembled due to time constraints. Even though it was not assembled it is clear that unit has a high mass for its size which may, or may not pose a problem depending on the application. Medium levels of force output is expected from this design.

4.4.1 Power Supply Problems

The unit was originally designed using $0.5mm^2$ copper wire but the only wire available to the author at the time was $0.2mm^2$. This caused a problem since the original coils were designed with a voltage of around 55 VDC, the new coils now needed around 90 VDC due to the higher wire resistance. With higher voltages the power supply design becomes more complex (and expensive). The original design also was to use h-bridge drivers with a maximum supply voltage of around 55 VDC, new h-bridge drivers also needed to be sought.

5. Prototype 2 - Tubular Linear Synchronous Motor

Prototype 2 is a tubular linear synchronous motor design which uses permanent magnets and steel cores as the armature and coils and an outer steel sheath as the stator. The armature is made up from grade N42 neodymium permanent magnets which are stacked N-S-S-N-N-S etc. and which are separated by steel spacers the same size and shape as the permanent magnets. The stator has thirteen coils each 4mm wide and separated by non-magnetic insulators which are 3mm wide.

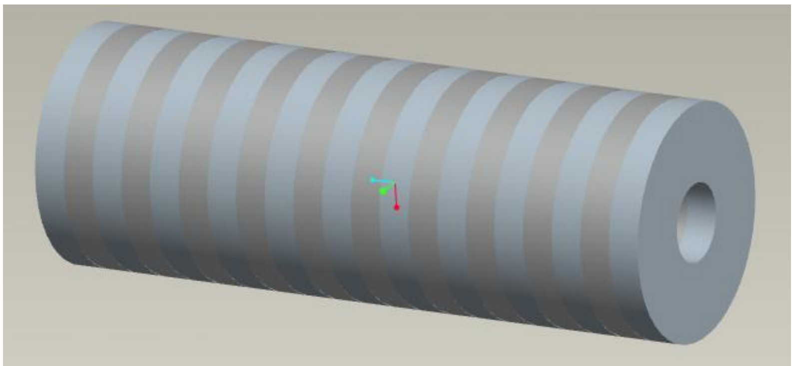


Figure 5-1: Armature showing alternate permanent magnet and steel spacer construction.

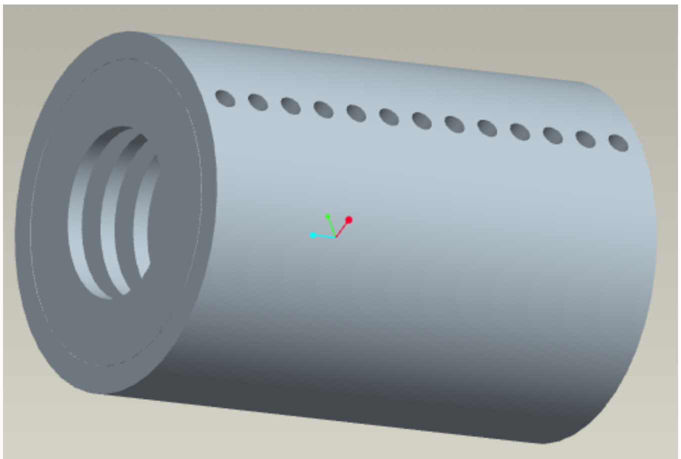


Figure 5-2: Stator showing the internal non-magnetic spacers which separate the thirteen coils.

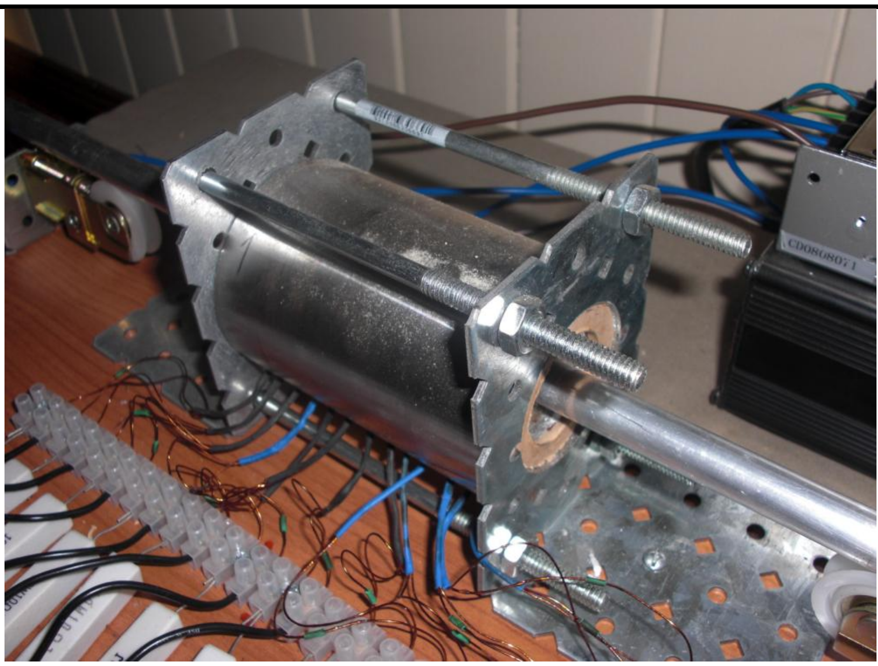


Figure 5-3: Image of the constructed prototype.

5.1 Design

5.1.1 Lorentz Force

Lorentz force states that when an electric current in a wire is exposed to a magnetic field it experiences Lorentz Force:

$$\mathbf{F}_e(t) = i(t).\ \mathbf{I} \times \mathbf{B}_g$$

Equation 5-1

where
$\mathbf{F}_e$ = Electromagnetic force vector, in newtons
$\mathbf{I}$ = Vector of length in direction of current i in wire
$\mathbf{B}_g$ = flux density of field in air gap

Considering the lateral section view of the single coil and PM design of Figure 5-4, it can be seen that the current vector **I** and the magnetic field vector **B** are orthogonal and therefore maximum force is produced. The magnitude of the electromagnetic force is (Shujun Zhang):

$$f_e(t) = N \cdot B_g \cdot \pi \cdot D \cdot i(t)$$

Equation 5-2

where
N = Number of turns of the stator winding
D = Diameter of the coil
B_g = Flux density of the PM

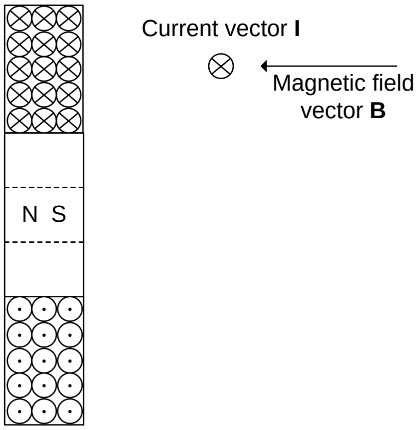


Figure 5-4: Lateral section of a single coil and PM design.

Therefore, for a single coil/permanent magnet arrangement where the magnet is of type N42 with a magnetic flux density of 1.32 T:

$$f_e(t) = N \cdot B_g \cdot \pi \cdot D \cdot i(t)$$
$$f_e(t) = 104 \quad 1.32 \quad \pi \quad 0.031 \quad 1.1$$
$$= 14.71 \text{ N}$$

This is consistent with the values obtained through the simulation.

5.1.2 Hardware Design

The initial design of the hardware for the actuator was completed in Maxwell SV simulation software. Several designs were simulated and the design with the greatest force output, based on available parts, was selected.

5.1.2.1 Vernier Type Scale

The relationship between the positioning of the permanent magnets and the coils was designed to be similar to a vernier scale relationship. A vernier scaled instrument is a tool used to measure length or angles to a high accuracy. In a vernier scaled instrument, the fixed section of the device has a set of uniform divisions that line up with another set of uniform divisions on the sliding section. The scale for the divisions on the sliding section is a constant fraction (usually 9/10) of the fixed scale. The two scales together are used to accurately find fractions of the divisions of the main scale.

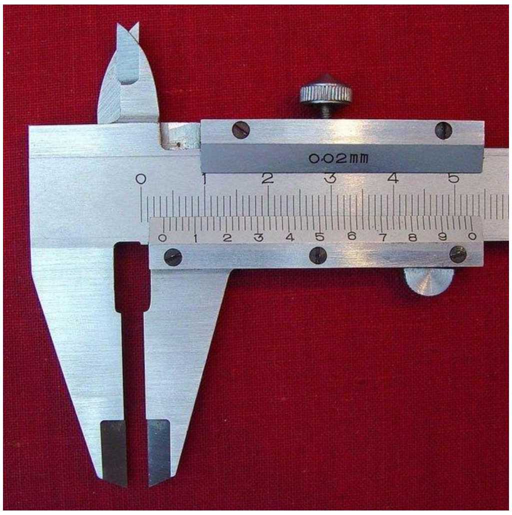


Figure 5-5: Vernier calipers *(ArtMechanic, 2004)*

In the same fashion the distance between each coil is slightly less that the distance between the permanent magnets. This way finer control can be gained over the position of the armature and in all positions at least one of the permanent magnet/coil pairs will be interacting with maximum magnetic field strength.

5.1.3 Neodymium Permanent Magnets

The neodymium permanent magnet is made from an alloy of neodymium, iron and boron and is currently the strongest type of permanent magnet (Wikipedia, 2009). Neodymium magnets are graded on the material they are made from. Generally the higher the number (N42 as opposed to N38), the higher the strength of the magnet. N42 permanent magnets were chosen for this project due to their low cost and high availability.

Figure 5-6 shows the specifications of the N42 neodymium magnet.

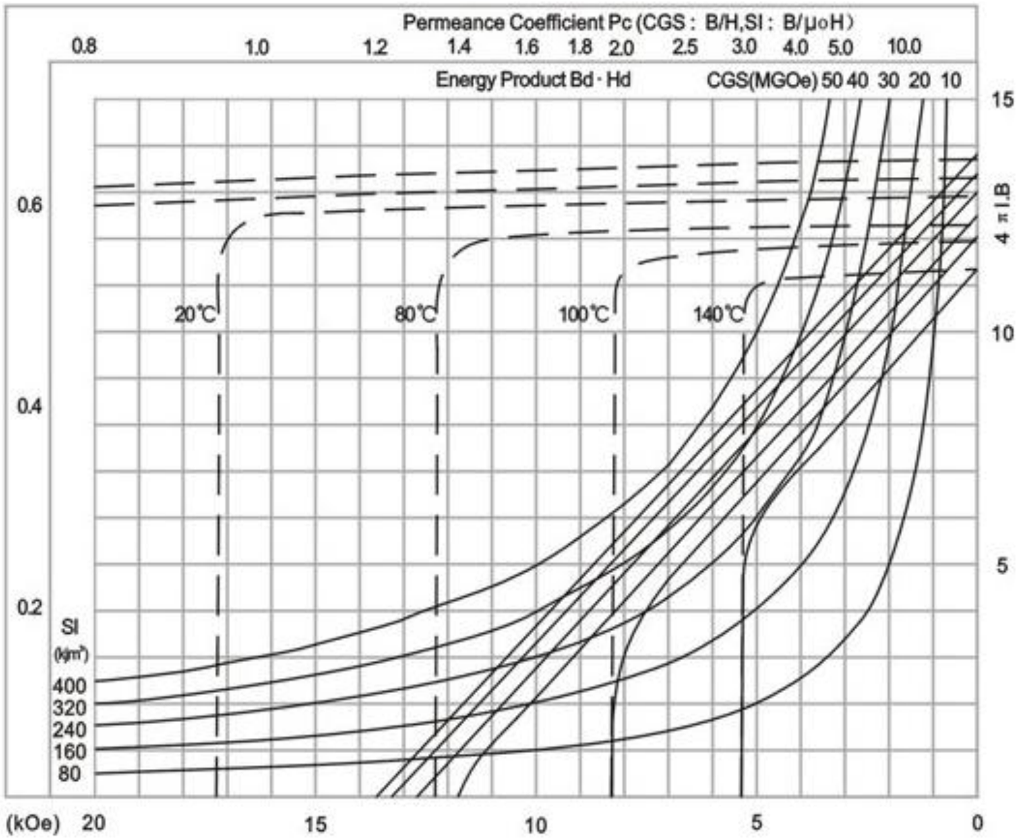


Figure 5-6: Typical specifications for N42 permanent magnets *(Eng-Tips Forums)*

5.1.4 H-Bridge

This design uses thirteen integrated h-bridge controller devices (part name: LMD18200 by National Semiconductor) which contain a MOSFET h-bridge circuit along with a control circuit which accepts a PWM signal and a direction input to control the direction and average current supplied to each coil.

The LMD18200 functional diagram is repeated below.

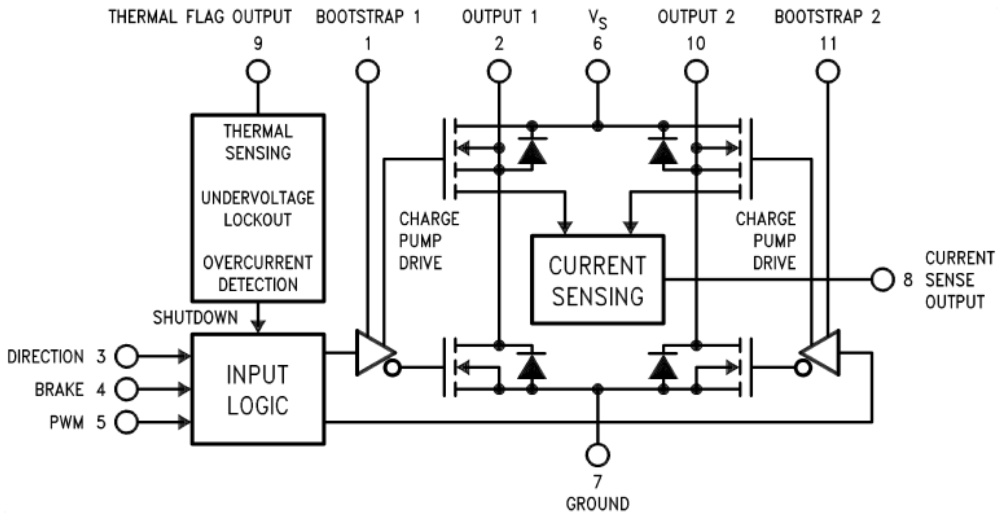


Figure 5-7: LMD18200 functional diagram *(National Semiconductor, 2005)*

The LMD18200 can be supplied in either an 11 lead TO-220 or 24 lead DIL package. The DC motor or coil is connected between pins 2 and 10 of the device and direction and average current is controlled via pin 3 (direction) and/or pin 5 (PWM).

The device has two modes of operation, namely:

- Locked anti-phase PWM control, or
- Sign/magnitude PWM control.

In the sign/magnitude PWM control mode, amplitude data is contained in the PWM signal and direction information is contained in the direction signal. The direction of the current to the motor is reversed when the signal to the direction pin is reversed. A 0% duty cycle on the PWM signal supplies 0% average current to the motor, 100% supplies 100% average current to the motor.

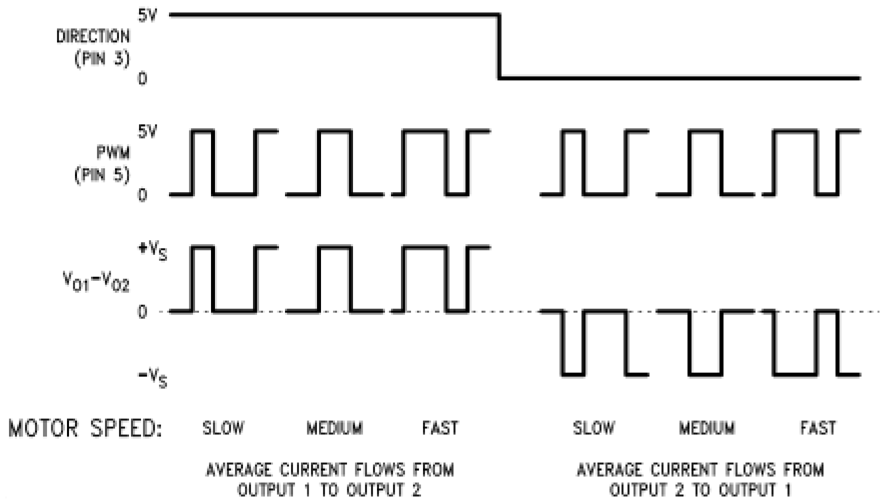


Figure 5-8: Sign/magnitude PWM control *(National Semiconductor, 2005)*

In the locked anti-phase PWM control mode, the PWM data contains both amplitude and direction information. The PWM pin is held high and the PWM signal is applied to the direction pin. Therefore a 50% duty cycle supplies an average current of zero to the motor. A duty cycle of 0% supplies an average current of 100% in one direction and 100% supplies an average current of 100% in the reverse direction.

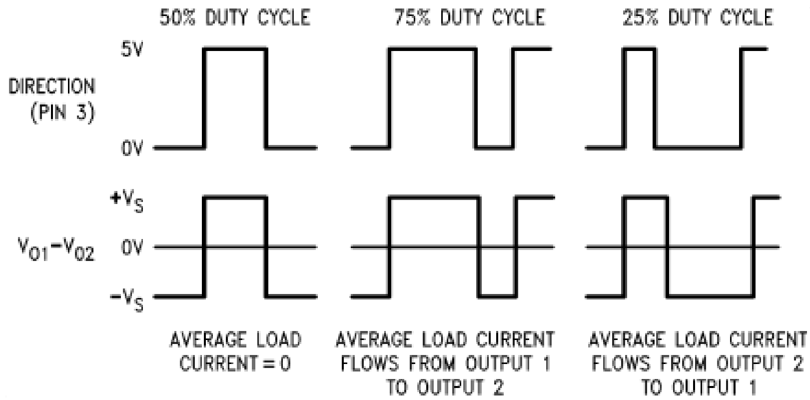


Figure 5-9: Locked anti-phase PWM control *(National Semiconductor, 2005)*

Prototype 2 uses the locked anti-phase PWM control mode because it uses less I/O pins on the microcontroller as a dedicated direction output is not required.

5.1.5 Electronic Hardware Design

Figure 5-12 shows a schematic of the electronic circuit design. The development system used was the SimmStick platform where the main power supply and serial communications circuitry is on a small motherboard () and the processor is on a daughterboard (Figure 5-11). These pre-made printed circuit boards were sourced from Dontronics and required the assembly of the components on to the circuit board before it was able to be used. This design approach was adopted to minimise the complexity of manufacturing a printed circuit board and to provide a known proven platform to work from.

Parts of the circuit in Figure 5-12 are taken from the SimmStick DT003 schematic (McKenzie, 1997) and the SimmStick DT106b schematic (Dontronics, 1999) not including the h-bridge circuits.

A Microchip PIC16F877A was selected as the microcontroller due to its low cost and availability. The circuit diagram was drawn using Eagle Layout Editor v5.4.2 Lite Edition.

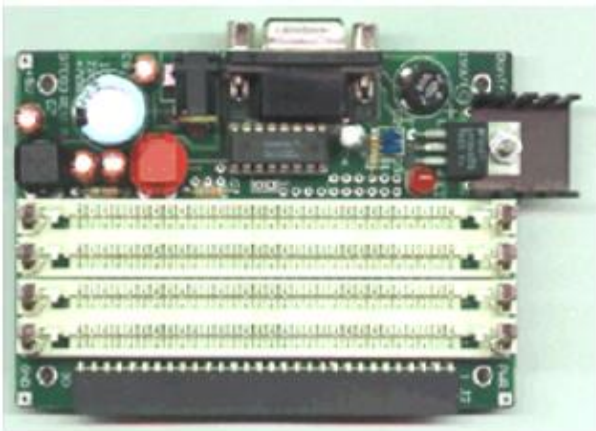


Figure 5-10: DT106 motherboard *(Dontronics, 2009)*

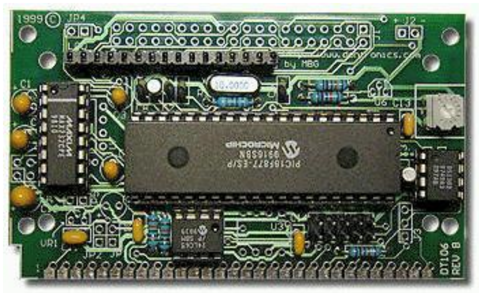


Figure 5-11: DT106 processor daughterboard *(Dontronics, 2009)*

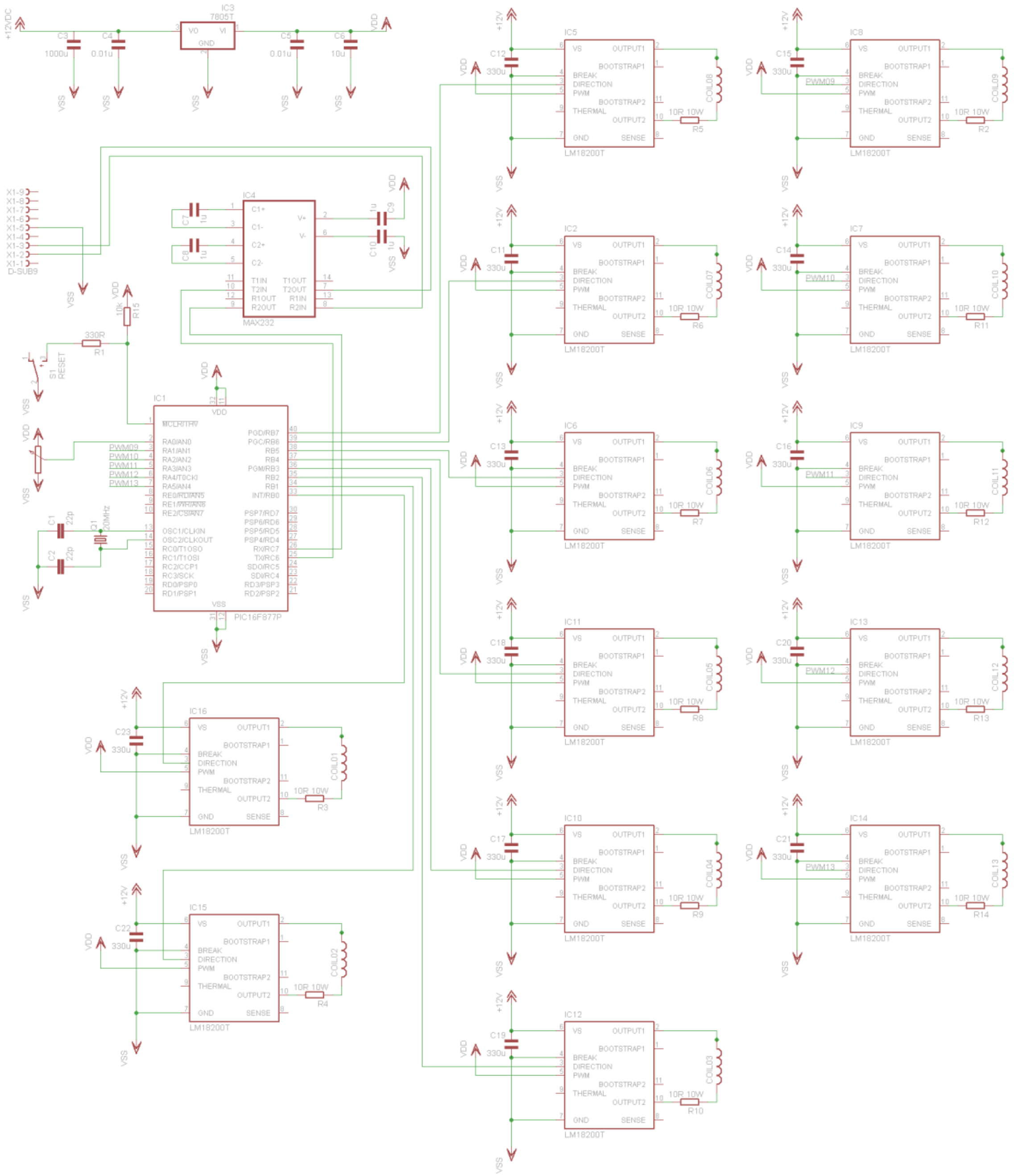


Figure 5-12: Prototype 2 control schematic.

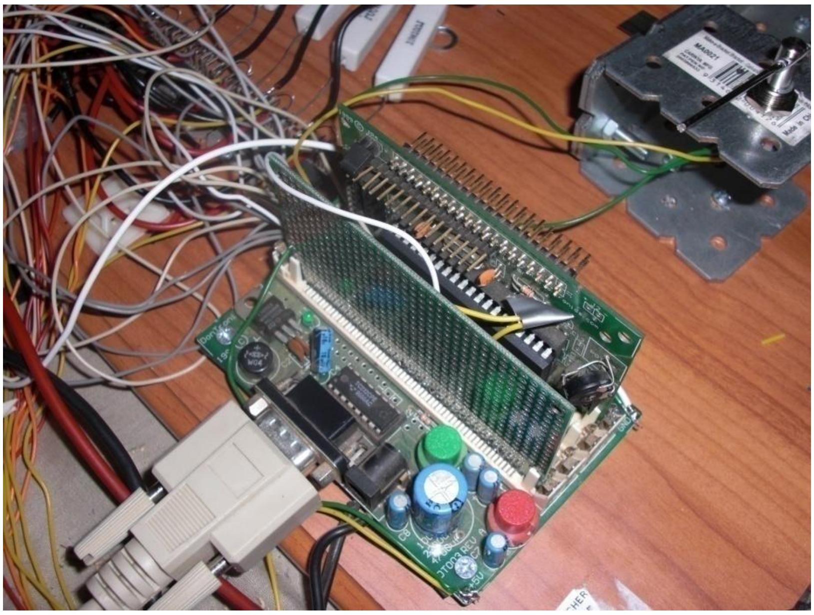


Figure 5-13: Image of the prototype microcontroller and development board.

5.1.6 Software Design

All microcontroller software was compiled in Microchip MPLAB v8.33.00.00 and written in Microchip assembly language for speed of operation and so that the timing could be effectively controlled. A complete listing of the microcontroller code can be found in Appendix D. Figure 5-14 shows the flowchart of the structure of the code.

On startup all internal registers are initialised to a known state then the peripherals such as the communications interface (SCI), analogue to digital converter (A/D) and the interrupt controller are configured. The programme then checks the SCI for a received command from the host computer. If a command is received this is then serviced, else the programme will go on to measure the actuator position via the A/D converter.

The position error is then calculated and setpoints determined for each of the thirteen driver coils. Coil drive data is then retrieved from a table look-up to drive the coils based on the current position and the requested position.

In the meantime the interrupt service routine (ISR) is called every 31.25 us whereby each of the thirteen PWMs are built.

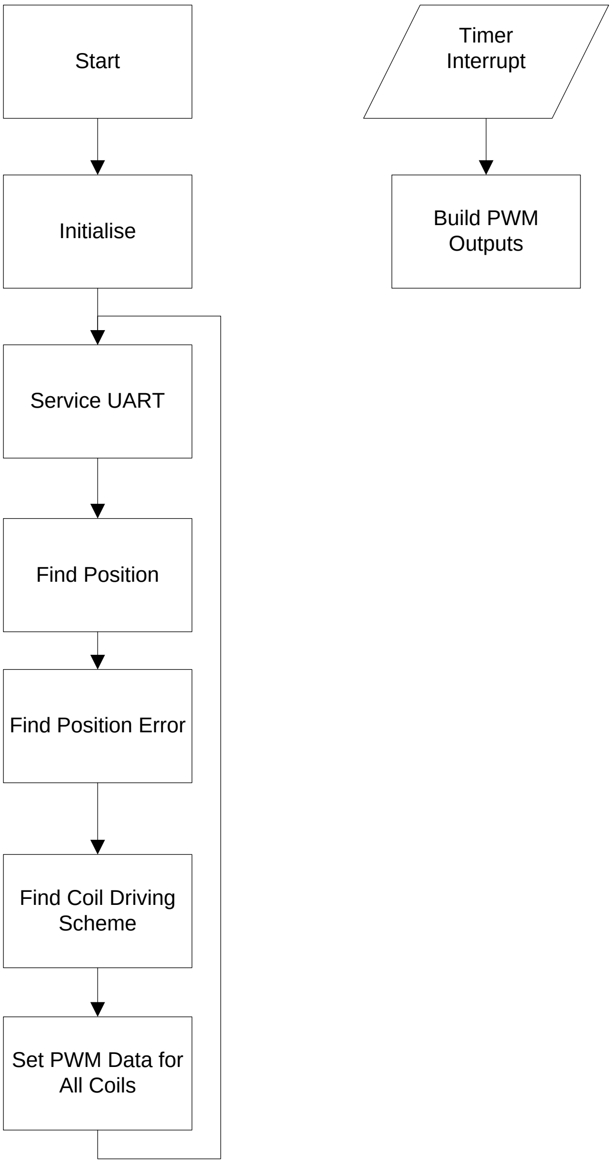


Figure 5-14: Software flow diagram.

5.1.6.1 Motor Control Scheme

Due to the complex nature of the coil assemblies a suitable coil driving scheme was developed by inspection. To facilitate deriving the scheme an assembly drawing was created for every possible position of the armature within the stator to a resolution of 1mm. Three sets of data were then derived based on what direction to drive each of the thirteen coils if the desired movement was:

- In one direction,
- In the opposite direction, and
- To hold the armature in place

.

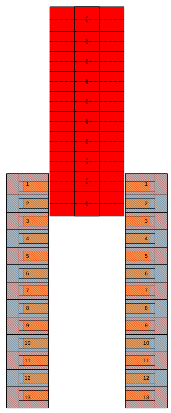


Figure 5-15: One of the 189 drawings developed of the stator/armature assembly to determine the drive required to move the armature.

	Coil Number	1			2			3			4			
	Coil config to provide max force to go in said direction	Up	Down	Hold	Up	Down	Hold	Up	Down	Hold	Up	Down	Hold	Up
	Rotor Position													
Fully Up	0	1	-1	0	1	-1	0	1	-1	0	1	-1	0	1
	1	1	-1	0	1	-1	0	1	-1	0	1	-1	0	1
	2	1	-1	0	1	-1	0	1	-1	0	1	-1	0	1
	3	1	-1	0	1	-1	0	1	-1	0	1	-1	0	1
	4	1	-1	0	1	-1	0	1	-1	0	1	-1	0	1
	5	1	-1	0	1	-1	0	1	-1	0	1	-1	0	1
	6	1	-1	0	1	-1	0	1	-1	0	1	-1	0	1
	7	1	-1	0	1	-1	0	1	-1	0	1	-1	0	1
	8	1	-1	0	1	-1	0	1	-1	0	1	-1	0	1
	9	1	-1	0	1	-1	0	1	-1	0	1	-1	0	1
	10	-1	-1	1	1	-1	0	1	-1	0	1	-1	0	1
	11	-1	1	0	1	-1	0	1	-1	0	1	-1	0	1
	12	-1	1	0	1	-1	0	1	-1	0	1	-1	0	1
	13	-1	1	0	1	-1	0	1	-1	0	1	-1	0	1
	14	-1	1	0	1	-1	0	1	-1	0	1	-1	0	1
	15	-1	1	0	1	-1	0	1	-1	0	1	-1	0	1
	16	-1	1	0	1	-1	0	1	-1	0	1	-1	0	1
	17	-1	1	0	-1	-1	1	1	-1	0	1	-1	0	1
	18	-1	1	0	-1	1	0	1	-1	0	1	-1	0	1
	19	-1	-1	1	-1	1	0	1	-1	0	1	-1	0	1
	20	1	-1	0	-1	1	0	1	-1	0	1	-1	0	1
	21	1	-1	0	-1	1	0	1	-1	0	1	-1	0	1

Figure 5-16: Partial diagram of the table developed to determine the polarity and drive current of each coil based on the armature location and requested position.

Colour Code	Operation
(green)	To raise position, set PWM on that coil to 0-15, to lower position, set PWM on that coil to 16-31, to hold position set PWM on that coil to zero. Code = 01
(red)	To raise position, set PWM on that coil to 15-31, to lower position, set PWM on that coil to 0-15, to hold position set PWM on that coil to zero. Code = 10
(blue)	To raise position, set PWM on that coil to 0-15, to lower position, set PWM on that coil to 0-15, to hold position set PWM on that coil to 15-31. Code = 11

Figure 5-17: Legend of what the driving code means in relation to the PWM signal.

This data was then entered into the control programme in a look-up table format. The general operation of this section of code is to determine the position in millimetres of the armature and use this position as an offset in the look-up table to determine the driving scheme for the thirteen coils. The table lookup retrieves four bytes of data which contain the drive information for the thirteen coils. This data is then interpreted and PWM set points generated for each coil. The interrupt service routine then builds the PWM signals.

For example, if the position of the armature is at 20mm and the position setpoint is set to 40mm, the code decides that the armature should go ‘down’ to reach the setpoint, retrieves the data for position ‘20mm’ from the lookup table and decides that the PWM for coil 1 should be between 0 and 15 counts (0-50%). The exact value between 0 and 15 is based on the magnitude of the error. The same process is completed for each coil.

5.1.6.2 Position Measurement

The linear position is measured by a precision potentiometer. The potentiometer provides the microcontroller with a voltage signal between 0 and 5 VDC which is proportional to the angle of the potentiometer. The voltage signal is converted to a value between 0 and 1023 by an internal analogue to digital (A/D) converter with 10-bit resolution.

Since the linear position of the armature relates to the potentiometer angle by a sine function, a suitable way of converting the raw A/D count to this linear position needed to be implemented in the microcontroller code. Normally in a high level programming environment, trigonometric functions are available but since this code was written in assembler these functions were not available. Therefore a lookup table was implemented to convert the A/D count to a position value in millimetres. A lookup table was implemented due to its fast operation and because there was enough spare programme space in the microcontroller.

Figure 5-18 shows the measured geometry of the potentiometer in relation to the armature shaft.

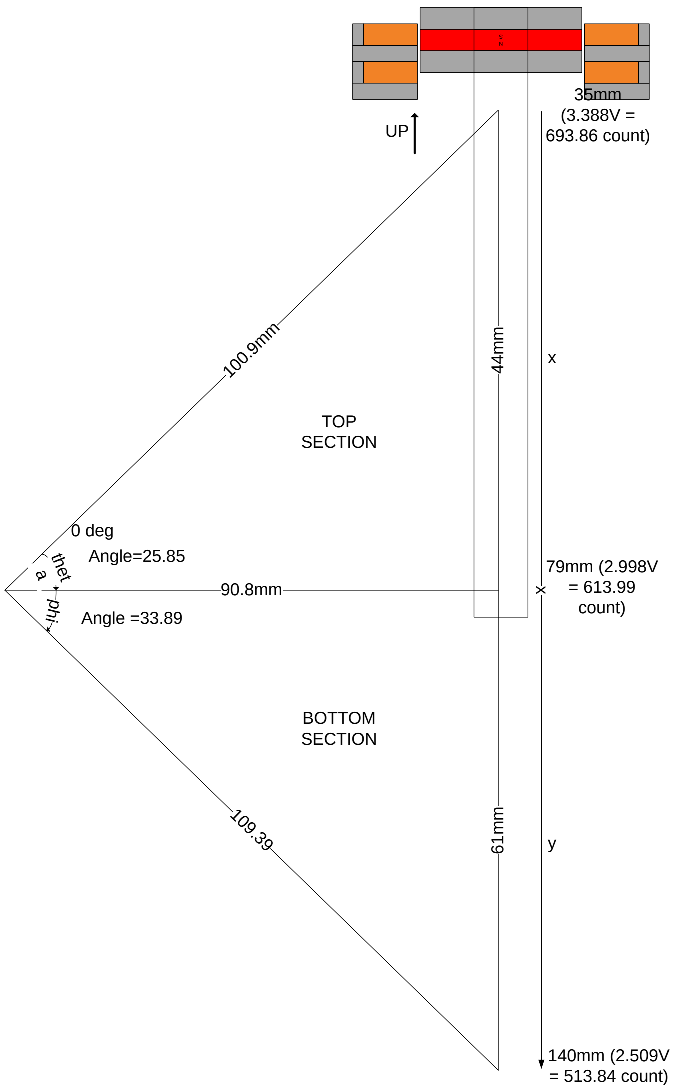


Figure 5-18: Geometry of the potentiometer in relation to the armature shaft.

The armature position in relation the potentiometer angle was determined as follows:

Top section:

$$\text{Voltage range} = 0.39\ \text{V}$$

$$\text{Count range} = \frac{0.39}{5} * 1024 = 79.87\ \text{counts}$$

$$\text{Angle of section} = 25.85°$$

$$\text{Resolution} = \frac{79.87}{25.85} = 3.090\ \text{counts/degree}$$

$$\text{For counts} \geq 613.99$$

$$\theta = \frac{693.86 - \text{actual count}}{3.090}$$
$$x = 79 - 100.9 * \sin(25.85 - \theta)$$

Bottom section:

$$\text{Voltage range} = 0.489 \text{ V}$$
$$\text{Count range} = \frac{0.489}{5} * 1024 = 100.15 \text{ counts}$$
$$\text{Angle of section} = 33.89°$$
$$\text{Resolution} = \frac{100.15}{33.89} = 2.955 \text{ counts/degree}$$

$$\text{For counts} < 613.99$$

$$\phi = \frac{613.99 - \text{actual count}}{2.955}$$
$$y = 79 + 109.39 * \sin(\phi)$$

The lookup table was developed separately to the microcontroller. Each entry of the table represents an A/D count between 0 and 1023 with each A/D count referring to an equivalent position in mm of the armature. Figure 5-19 shows a partial diagram of the table used to find the position of the armature with respect to the A/D count.

AD count	Angle (degrees)	Position (mm)	PIC Instruction and Hex value
0	207.78	188.39	retlw 0xBC
1	207.4416	188.39	retlw 0xBC
2	207.1032	188.39	retlw 0xBC
3	206.7648	188.39	retlw 0xBC
4	206.4264	188.39	retlw 0xBC
5	206.088	188.39	retlw 0xBC
6	205.7496	188.39	retlw 0xBC
7	205.4112	188.39	retlw 0xBC
8	205.0728	188.39	retlw 0xBC
9	204.7343	188.39	retlw 0xBC

Figure 5-19: Partial diagram of the table developed to determine position of the armature shaft with respect to the A/D count from the potentiometer.

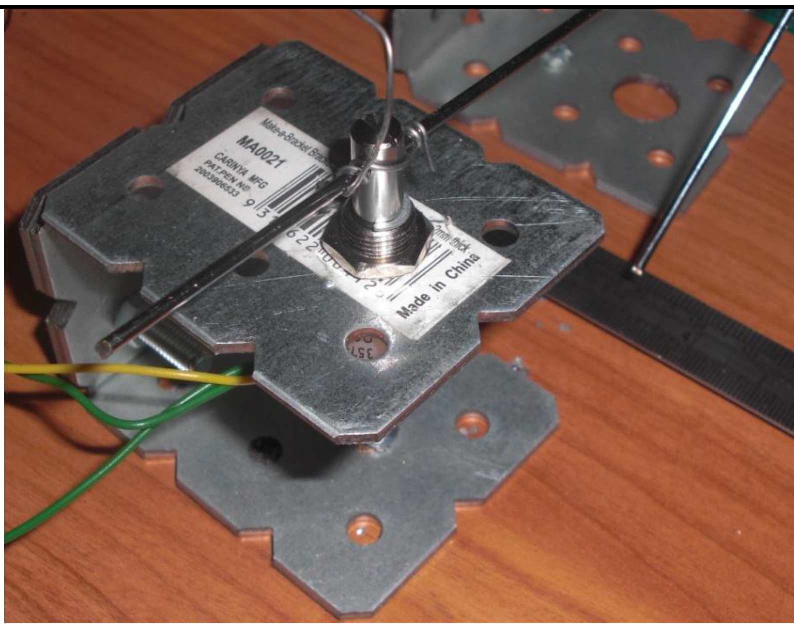


Figure 5-20: Image of the prototype potentiometer arrangement.

5.1.6.3 PWM Generation

The PIC16F877A microcontroller contains only two hardware PWM controllers. Since this design uses thirteen, another method of producing the PWM signals needed to be devised.

The method selected was to implement a timer interrupt in the microcontroller that interrupts the code execution after a set amount of time and then runs an interrupt service routine. This routine builds the thirteen PWM signals based on the PWM set points generated in the main routine.

Each call to the interrupt service routine (ISR) corresponds to the resolution time of the PWM signal. When the ISR is called the output pins corresponding to each coil are either driven on or off depending on the requirements. On each call the PWM signals are built, after the 32nd ISR call the PWM period corresponding to 1 ms is completed and the cycle is restarted. This corresponds to a PWM frequency of 1kHz.

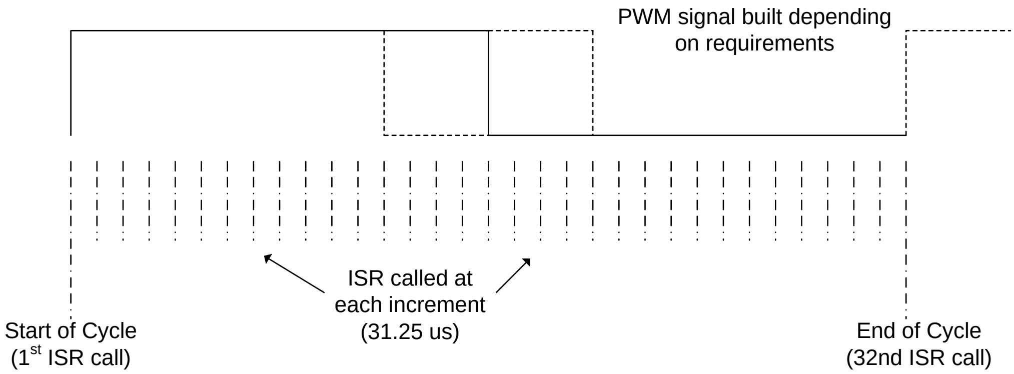


Figure 5-21: Diagram of the building of the PWM signal through successive timer interrupts.

The time between successive calls to the interrupt service routine (ISR) was selected to be 31.25 us which allows a PWM resolution of 32 steps with a PWM frequency of 1 kHz. One instruction cycle for the PIC16F877A microcontroller is:

$$Tcy = \frac{4}{20*10^{6}} = 200*10^{-9}\,\text{s} = 200\ \text{ns}$$

The number of instructions cycles in the ISR is a maximum of 85 (by inspection), therefore the time taken to complete the ISR is:

$$T_{ISR} = 85*200 = 17*10^{3}\ \text{ns} = 17\ \mu\text{s}$$

This leaves 14.25 us to service the main code which contains the position measurement and control routines.

5.1.7 Monitoring Software

Monitoring software was built using Microsoft Visual Basic v4.0 Enterprise Edition. The application consists of a simple terminal interface which writes an entered hexadecimal value to the RS232 port. When a value is received from the microcontroller it is placed in a text box for viewing. There are two hexadecimal command codes as described below:

- 0xf7. Request position in millimetres.

- 0xf8. Request current position setpoint in millimetres.

When either of these codes is sent to the microcontroller, it responds with the current position or the current position setpoint respectively.

The programme also features a trending function which, when enabled, continually polls the microprocessor for position information. The received position along with the position setpoint is then trended.

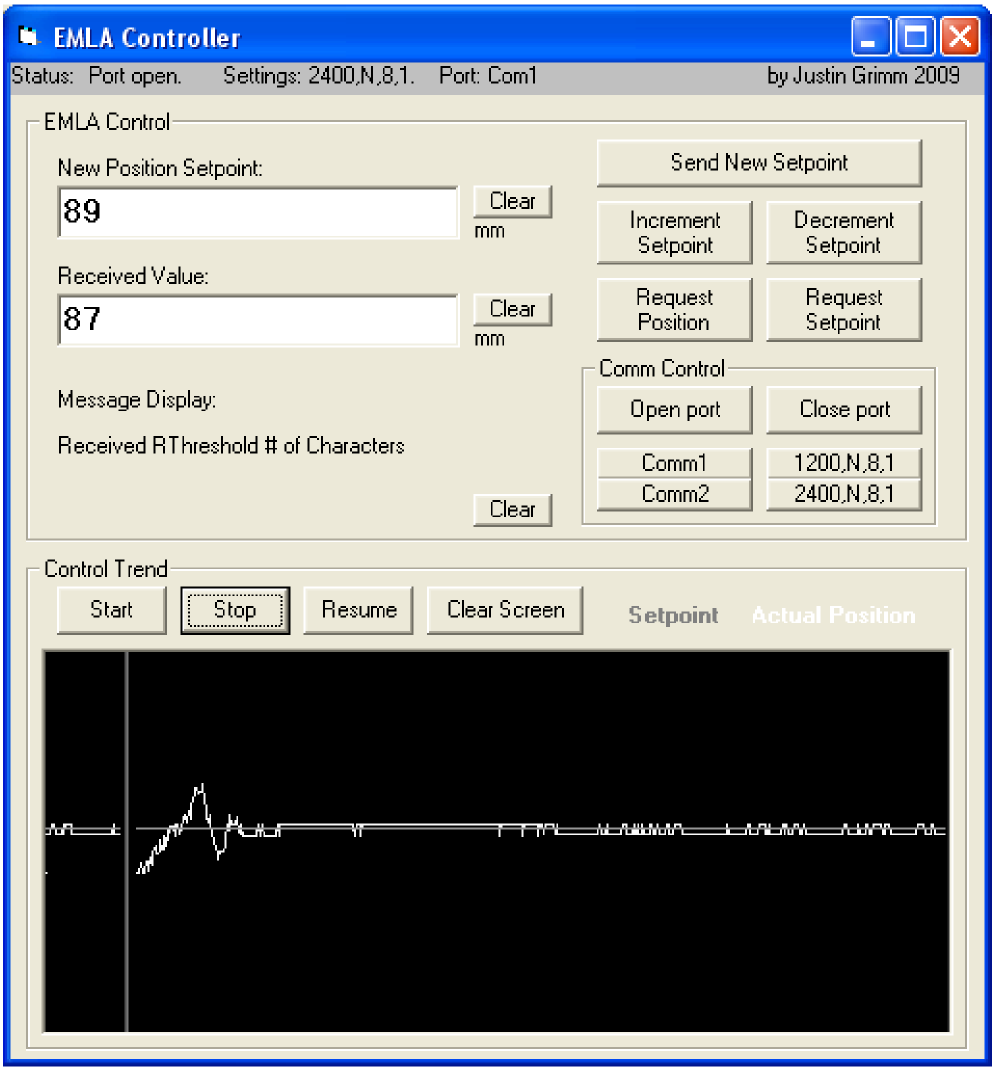


Figure 5-22: Screenshot of the terminal programme.

5.2 Simulation

5.2.1.1 Current Density

The calculations below determine the current density in the coils for the purposes of simulation.

Coil window area:

$$A_c = (51-31)\ \ 4 \ \ = 80\ \text{mm}^2$$

Wire size:

$$A_w = \pi r^2 = \pi(0.25)^2 = 0.1963\ \text{mm}^2$$

Theoretical turns in coil window:

$$T_t = \frac{80}{0.1963} = 407$$

Air space per winding:

$$A_a = \frac{2r\ ^2 \sqrt{3}}{2} - \pi r^2 = \frac{0.5\ ^2 \sqrt{3}}{2} - \pi(0.25)^2 = 0.0202\ \text{mm}^2$$

Therefore area taken by each winding:

$$A_t = 0.1963 + 0.0202 = 0.2165\ \text{mm}^2$$

Calculated turns:

$$T_c = \frac{80}{0.2165} = 369$$

Current carrying capacity of 0.2mm^2 copper wire:

$$I_{CCP} = 1.2\ \text{A}$$

Therefore current density:

$$I_D = 369\ \ 1.2 \ \ = 442\ \text{At}$$

5.2.1.2 Simulation Report

This section shows the simulation parameters and mesh results for the tubular linear synchronous motor. Multiple simulations were carried out on this design with a changing actuator position. All coils were energised as per the control philosophy.

Software	Maxwell SV version 3.1.04
Solver	Magnetostatic
Drawing	RZ Plane
Solver Passes Required	10
Triangles	2449
Energy Error	0.909%
Source for coils	442 At

Table 5-1: Simulation parameters.

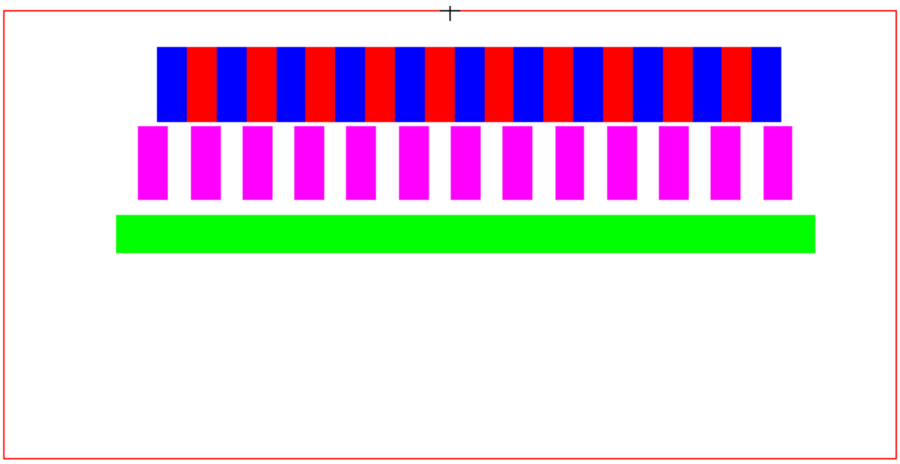


Figure 5-23: Lateral half section.

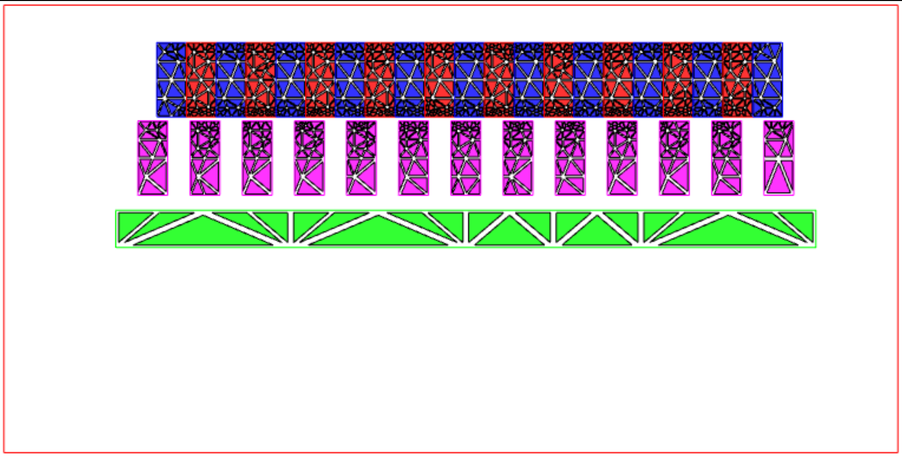


Figure 5-24: Mesh plot (lateral half section).

5.2.1.3 Simulation Results

The simulation showed a resultant maximum force on the actuator of 53.1 N at 90mm. Figure 5-25 shows the flux path generated by the coil when the actuator is centred. The resultant force at this position is 38.2 N.

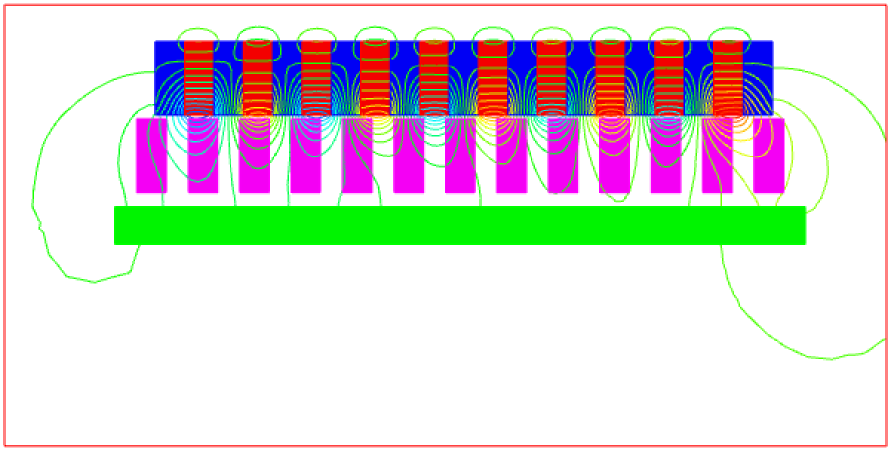


Figure 5-25: Flux path when the actuator is centred.

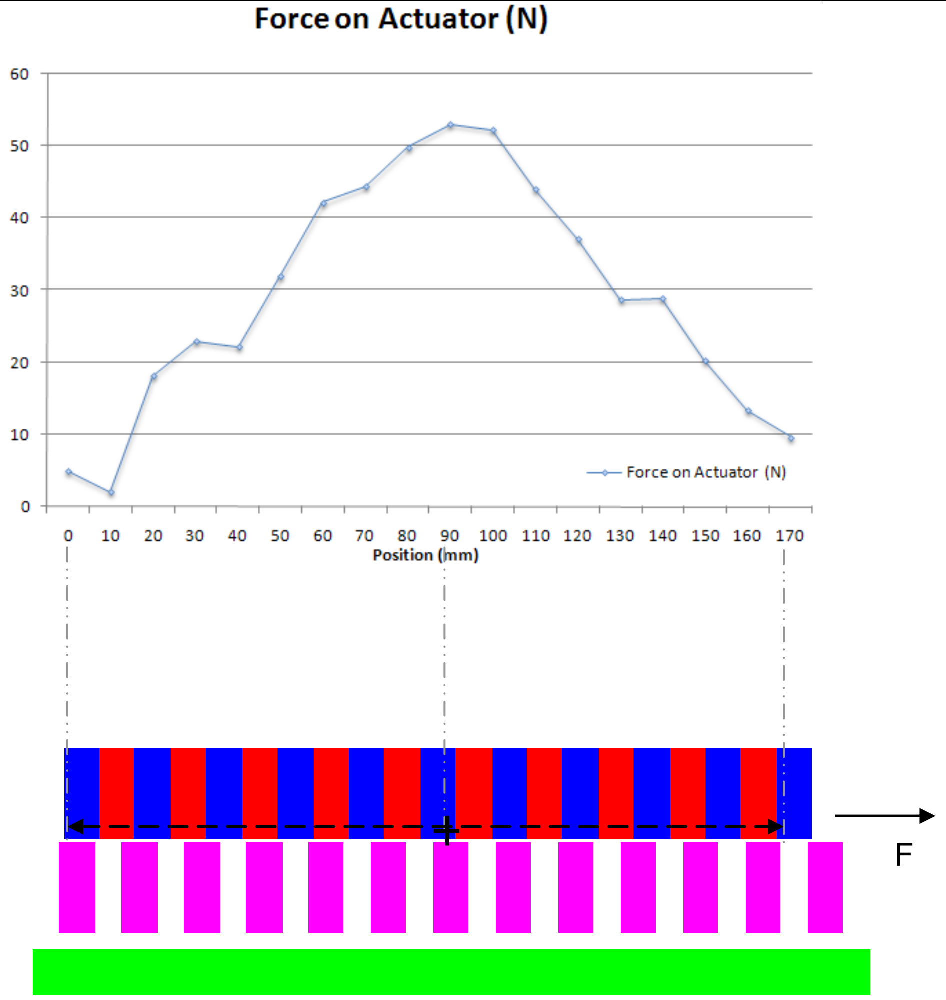


Figure 5-26: Graph of position versus force for Prototype 2 simulation.

5.3 Construction

Prototype 2 was constructed from the following components:

Item	Part	Material	Quantity	Purpose
1	Stator	AISI 1010 Steel	1	Path for electromagnetic flux
2	Spacer	AISI 1010 Steel	11	Spacer between magnets, path for electromagnetic flux
3	Coil	Copper Wire	13	Generate controllable electromagnetic fields
4	Actuator Rod	Aluminium	1	Used to hold armature assembly together
5	Neodymium Magnet	N42 Neodymium Permanent Magnet	10	To create required magnetic fields

Table 5-2: Part list for Prototype 2.

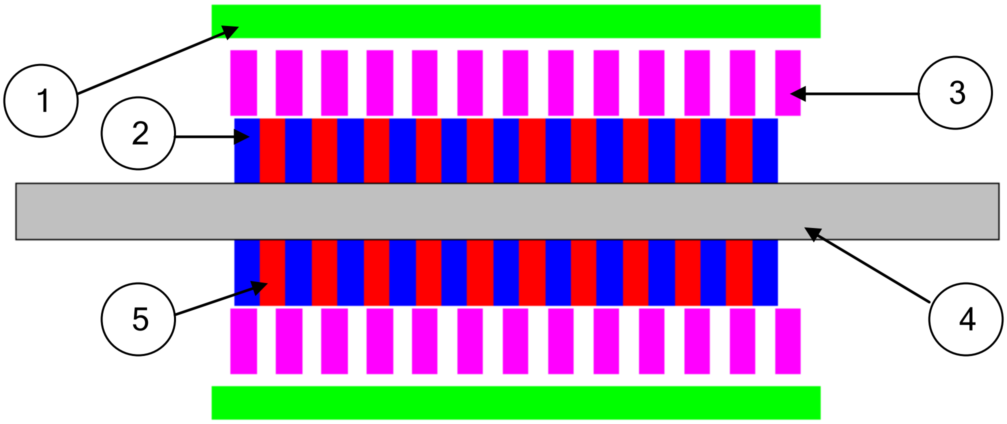


Figure 5-27: Chosen design for Prototype 2 (lateral section).

5.3.1 Coils

All thirteen coils were hand wound using 0.2mm^2 enamelled copper wire. Each coil was wound around a former and packed with potting resin which was used to allow the coil to support itself. Each coil has a total of 104 turns.

The coil forming assembly was prepared by coating the individual parts in light grease and then applying aluminium foil to all parts. This was done so that the former can be easily disassembled after the resin has cured due to the problem of the resin bonding to the bare metal.

Figure 5-28 shows the coil forming tool used to hold the coil whilst the resin is curing. Figure 5-29 shows the coil after it has been removed from the former. Some aluminium foil can be seen on the coil.

Figure 5-28: Coil forming apparatus.

Figure 5-29: Coil after removal from the former.

Figure 5-30: Coil ready for assembly.

Coil resistance was measured to be between 1.1 and 1.2 ohms. Using ohms law to calculate the voltage required to cause 1.2A of current flow:

$$\begin{aligned} V_{coil} &= I_{coil} R_{coil} \\ &= 1.2(1.1) \\ &= 1.32 \text{ V} \end{aligned}$$

Since the design uses a 12 volt power supply, a current limiting resistor was used in series with each coil. The value and power rating for these resistors were calculated as follows:

$$V_{drop} = V_{ps} - V_c = 12 - 1.32 = 10.68 \text{ V}$$

$$R_{res} = \frac{V_{drop}}{I_{coil}} = \frac{10.68}{1.2} = 8.9\ \Omega$$

$$P_{res} = V_{drop} I_{coil} = 10.68(1.2) = 12.8 \text{ W}$$

A 10 ohm 10W resistor was chosen to limit the current in the coils.

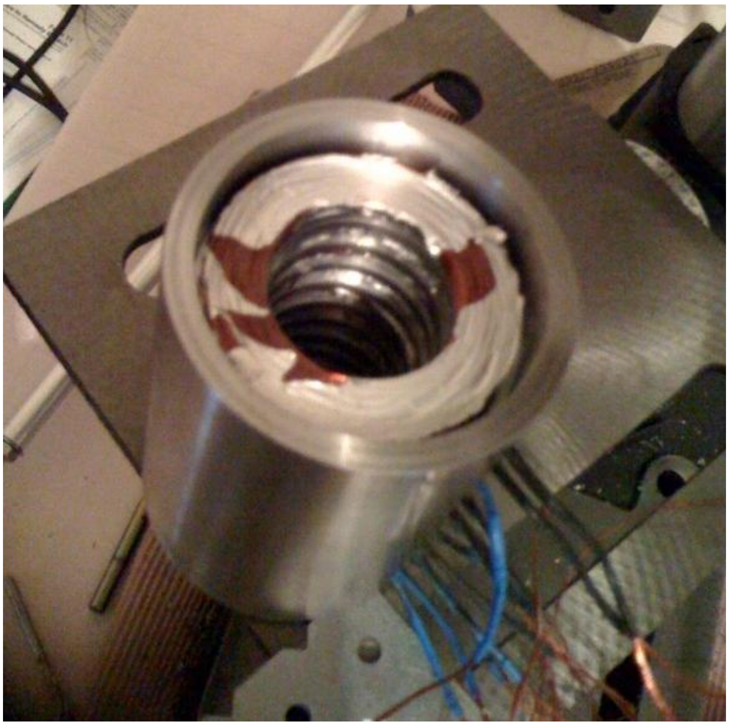


Figure 5-31: Coils being assembled into the stator.

5.3.2 Hardware Layout

The hardware for the project was fixed to a piece of timber approximately 600 mm long and 400 mm wide. The actuator assembly was then fixed to this board with mounting brackets. The h-bridge drivers were also fixed directly to the board along with the potentiometer and the controller board.

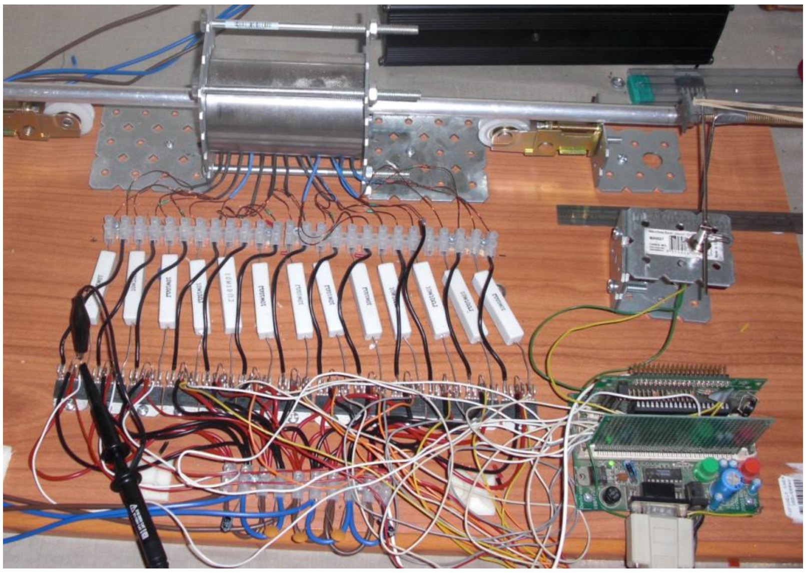


Figure 5-32: Hardware layout.

5.4 Testing and Results

5.4.1 Testing

Initial testing of the control electronics was completed with the power to the coils turned off. The microcontroller was powered up and the outputs to all thirteen coils checked against the coil drive chart (

Figure 5-16). The position of the armature was then changed to see if the coil control changed as expected.

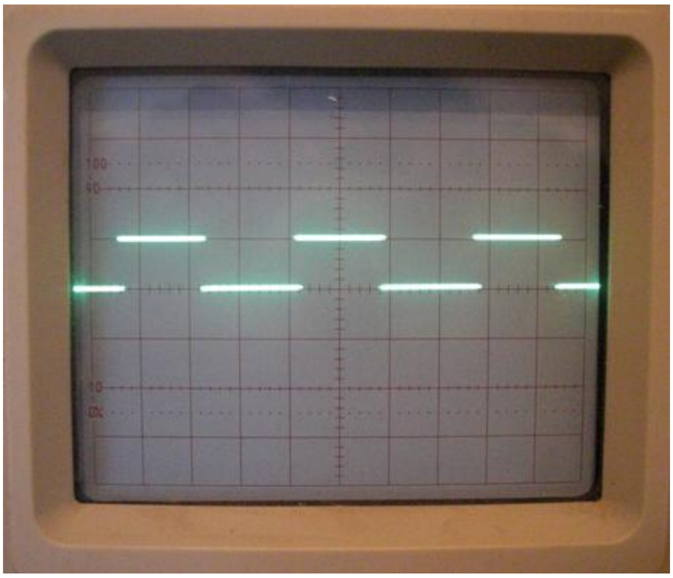


Figure 5-33: Cathode ray oscilloscope trace of the output to the first h-bridge driver.

Coil power was then turned on and the first thing that was noticed was the audible sound of the frequency of the coil drives which are at 1 kHz. A position setpoint was then sent to the microcontroller via the terminal application and a lack of position control was observed.

Several attempts were made to control the position which all failed. The microcontroller software was checked and some adjustments made to the coil drive routine, this then achieved some control over the position. More checks were made on the position measurement apparatus and it was found that there was around 7% hysteresis in the potentiometer.

The potentiometer was changed out for a precision type and the geometry rechecked for the population of the A/D count vs. position chart (Figure 5-19). There were some errors found in the chart and these were rectified. Control of the position was then achieved.

5.4.2 Measured Forces

Forces were measured with a spring type weight scale. The scale was attached to the actuator and the setpoint changed to be the furtherest distance away from the actual position. The force exerted was then measured.

The maximum force measured was around 8 N. The author expects greater forces; up to 20 N could be produced by this prototype with a finer control of the coil drives.

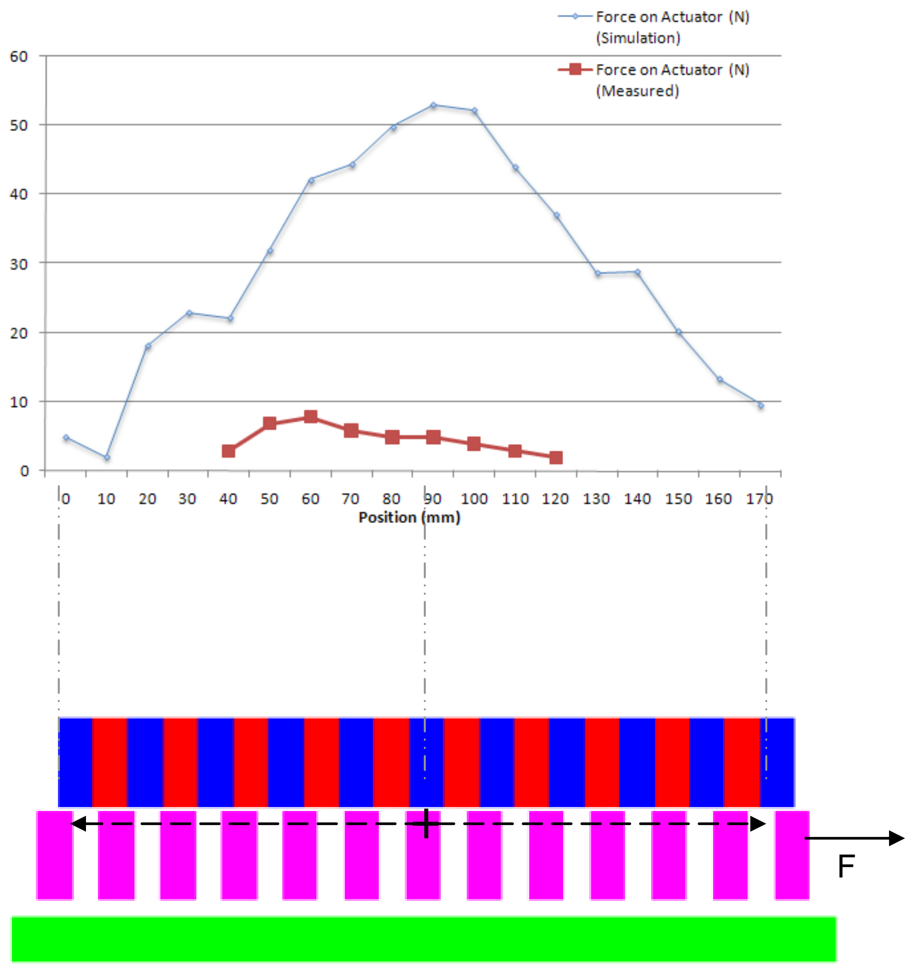


Figure 5-34: Graph of position versus force for Prototype 2 (measured and simulation).

5.4.3 Control

The actuator provides reasonable position control as depicted by Figure 5-35. This control could be improved by providing a more accurate way of measuring the position of the armature. An accuracy of better than 0.5 mm would be required to adequately control the coil drivers. Due to the inaccuracies of the position measurement the actuator suffers some stalling typically when it is at specific positions. This is due to the coil driving code being one or two millimetres different than where it should be and therefore providing the incorrect drive signals.

In the trend below the white line is the actual position and the grey line is the setpoint.

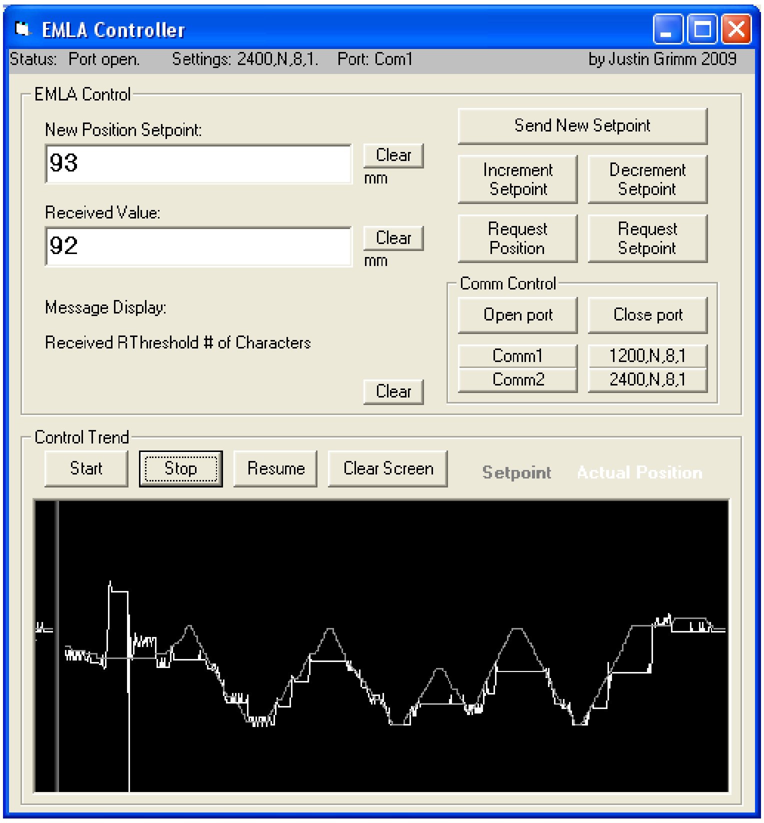


Figure 5-35: Trend of setpoint versus position.

1.1.1.1.3.1Appendix F contains a video of the actuator being controlled.

5.4.4 Problems

5.4.4.1 Coil Manufacture

Initially there were some issues with the manufacture of the coils. The epoxy resin mix bonded to the side of the coil forming tool which made disassembly impossible without damaging the coil.

The first attempt at rectifying the problem was to use a non-stick Vaseline gel to coat the former to stop the bonding effect. This did not work as expected and the epoxy resin again stuck to the former.

The next attempt saw the former coated in the Vaseline gel and then coated in general purpose aluminium foil. This allowed the moulded coil to come away from the former without damaging the coil.

5.4.4.2 Cogging Force

The initial design of the actuator used steel spacers between the coils to act as pole pieces. When the actuator was assembled it was found to be difficult to insert the armature into the stator due to the close proximity of the pole pieces to the permanent magnets. It was found that the attraction between these two components to be far greater than the force generated by the magnetic field could overcome.

It was decided to remove the steel pole pieces and replace them with non-ferromagnetic wooden spacers. This alleviated the cogging issue but also reduced the potential resultant axial force that could be generated according to the simulation results.

5.4.4.3 Hysteresis in Potentiometer

To control the position of the armature effectively its position must be known to an accuracy of 0.5 mm. When the actuator was tested it was found that controlling of the position was difficult. After some investigation it was found that the hysteresis of the position potentiometer was in the order of 7% of range which equated to around 10 mm. Dismantling the potentiometer found that the wiper has some degree of lateral movement so that moving it in one direction shifted the wiper to a different position than when it is in the opposite direction.

A precision potentiometer was sourced and proved to have a hysteresis of around 1%. To partially remove this error a routine was implemented in code to detect the direction of travel of the potentiometer and the 1% error was then added or subtracted from the position depending on this direction.

5.5 Conclusions

It has been shown that a Tubular Linear Synchronous Motor can be easily designed, simulated, manufactured and controlled. With tighter measurement of the position there can be tighter control and therefore greater resultant forces.

5.6 Future Work

Future work that could be done to Prototype 2 may include:

- Design better position feedback to have a tolerance of <0.5 mm.
- Investigate the relationship between the inside diameter of steel pole pieces and cogging force. Having steel pole pieces may improve the force output.
- Manufacture the actuator under tighter tolerances so that the air gap can be reduced.
- Investigate higher current densities in the coils.

6. Discussion

Prototype 1 and Prototype 2 both show that there is merit in continued investigation into these types of linear electromagnetic actuators.

Prototype 1 is a solenoid type actuator which contains no permanent magnets but relies on minimising the reluctance path to create movement in the armature. It can be seen by the design that heavy components are needed to provide the least amount of reluctance in the stator assembly. This may cause some issues due to the high mass of the device. The prototype also requires reasonably high power to operate, in the order of 200 W. This may or may not be a problem depending on the application. In robotics, where the power supply is on board the unit, this level of power may not be sustainable for extended periods of time.

Prototype 2 is a tubular linear synchronous motor where the armature is made from a series of permanent magnets separated by steel spacers and the stator contains multiple independently controlled coils. Movement in the armature is generated by controlling the polarity of the coils based on the position and thereby creating an interaction between the magnetic fields of the permanent magnets and the coils. This design is smaller that Prototype 1 and requires less power but also requires a control system which is more complex.

Prototype 2 has the most merit out of the two designs. With more investigation the author believes that the unit can produce greater forces than experienced in the prototype by tightening the measurement and control and by investigating higher flux densities in the coils.

7. Conclusions

The outcomes of the project specification were mostly achieved. A functioning electromagnetic linear actuator was designed and constructed, it was made from purely electromagnetic components, it is similar in size to an off-the-shelf 300 mm pneumatic actuator and it is controlled by a microcontroller system. The only item it did not fully achieve is that it did not produce the high forces anticipated in the project specification.

There is merit in the continued development of this project, especially in the area of tubular linear synchronous motors. Tighter manufacturing controls will produce a linear machine with tighter tolerances and in turn a greater force output.

8. References

(n.d.). Retrieved September 30, 2009, from Eng-Tips Forums: http://www.eng-tips.com/viewthread.cfm?qid=240251&page=1

ArtMechanic. (2004, March 18). *Vernier scale*. Retrieved October 4, 2009, from Wikipedia: http://en.wikipedia.org/wiki/Vernier_scale

Buttay, C. (2006, June 9). *H bridge.svg*. Retrieved August 15, 2009, from Wikipedia: http://en.wikipedia.org/wiki/File:H_bridge.svg

Buttay, C. (2006, June 9). *H-Bridge*. Retrieved August 15, 2009, from Wikipedia: http://en.wikipedia.org/wiki/H-bridge

Cabrillo College. (n.d.). *Chapter 29: Magnetic Fields due to Currents.* Retrieved October 5, 2009, from Cabrillo College: http://www.cabrillo.edu/~cfigueroa/4B/4Bexamples/4Bexample29.pdf

Diamagnetism. (2009, April 27). Retrieved May 4, 2009, from Wikipedia: http://en.wikipedia.org/wiki/Diamagnetic

Dontronics. (2009, June 23). *DT003 SimmStick Power Supply and Comms Platform* . Retrieved August 23, 2009, from Dontronics: http://www.dontronics.com/dt003.html

Dontronics. (2009, June 23). *DT106 28/40 Pin PICmicro on a SimmStick* . Retrieved August 23, 2009, from Dontronics: http://www.dontronics.com/dt106.html

Dontronics. (1999, September 11). *DT106 28/40 Pin PICmicro on a SimmStick*. Retrieved August 23, 2009, from Dontronics: http://www.dontronics.com/pdf/dt106bct.pdf

Ferromagnetism. (2009, April 30). Retrieved May 4, 2009, from Wikipedia: http://en.wikipedia.org/wiki/Ferromagnetic

Fowler, M. (1997). *Historical Beginnings of Theories of Electricity and Magnetism*. Retrieved May 4, 2009, from Galileo and Einstein: http://galileoandeinstein.physics.virginia.edu/more_stuff/E&M_Hist.html

Geology.com. (2009). *Magnetite and Lodestones*. Retrieved May 4, 2009, from Geology.com: http://geology.com/minerals/magnetite.shtml

Georgia State University. (2005). *Electromagnet*. Retrieved May 4, 2009, from Hyperphysics: http://hyperphysics.phy-astr.gsu.edu/hbase/magnetic/elemag.html#c4

Georgia State University. (2005). *Magnetic Properties*. Retrieved May 4, 2009, from HyperPhysics: http://hyperphysics.phy-astr.gsu.edu/HBASE/Tables/magprop.html

Haiwei Lu, J. Z. (2008, April 4). *A Miniature Short Stroke Linear Actuator—Design and Analysis.* Retrieved April 5, 2009, from http://ieeexplore.ieee.org.ezproxy.usq.edu.au/stamp/stamp.jsp?arnumber=04475333

jjbeard. (2006, June 1). *MOSFET*. Retrieved August 15, 2009, from Wikipedia: http://en.wikipedia.org/wiki/MOSFET

Laboratory for Intelligent Mechanical Systems. (n.d.). *400px-Motor_Commutators*. Retrieved September 30, 2009, from Laboratory for Intelligent Mechanical Systems: http://hades.mech.northwestern.edu/images/thumb/c/cf/Motor_Commutators.jpg/400px-Motor_Commutators.jpg

Machine-Design.com. (n.d.). Retrieved May 23, 2009, from Machine-Design.com: http://images.machinedesign.com/images/archive/71776lowfrictio_00000049729.jpg

Magnetism Group, Physics Dept, Trinity College Dublin. (n.d.). *Myths and Origins: Child A encounters the lodestone*. Retrieved May 4, 2009, from Magnetism Through the Ages: http://www.tcd.ie/Physics/Schools/what/materials/magnetism/one.html

Magnetism Hand Rules. (n.d.). Retrieved August 15, 2009, from Magnetism Hand Rules: http://www.waowen.screaming.net/Maghandrules.htm

McKenzie, D. (1997, April 29). *DT003 SimmStick Power Supply and Comms Platform* . Retrieved August 23, 2009, from Dontronics: http://www.dontronics.com/pdf/dt03as.pdf

Microchip Inc. (2009). *PIC16F877A.* Retrieved August 15, 2009, from Microchip: http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en010242

National Semiconductor. (2005, April). *LMD18200.pdf.* Retrieved August 15, 2009, from National Semiconductor: http://www.national.com/ds/LM/LMD18200.pdf

Nave, C. (2006). *Magnetic Domains*. Retrieved August 15, 2009, from HyperPhysics: http://hyperphysics.phy-astr.gsu.edu/hbase/solids/ferro.html#c5

Nikunj shah, Rob Jamieson. (2006). Electromagnetic Linear Actuator. 28.

Nogueira, N. (2006). *File:Solenoid.svg*. Retrieved May 4, 2009, from Wikipedia: http://en.wikipedia.org/wiki/File:Solenoid.svg

Paramagnetism. (2009, April 23). Retrieved May 4, 2009, from Wikipedia: http://en.wikipedia.org/wiki/Paramagnetism

PCB Heaven. (2008). *PWM Modulation*. Retrieved August 15, 2009, from PCB Heaven: http://pcbheaven.com/wikipages/PWM_Modulation/

Shujun Zhang, L. E. (n.d.). *Modeling and Control for Tubular Linear Permanent Synchronous Machines with Gas Springs in Drilling Applications.* Retrieved October 2, 2009, from Institutt for elkraftteknikk: http://www.elkraft.ntnu.no/eno/Papers2008/Zhang-ICEMS.pdf

SilverStar. (2006, October 26). *PID Controller*. Retrieved August 14, 2009, from Wikipedia: http://en.wikipedia.org/wiki/PID_controller

Stannered. (2007, February 6). *Magnetic Field.* Retrieved May 23, 2009, from Wikipedia: http://en.wikipedia.org/wiki/Magnetic_field

University of Surrey. (2004). *The force produced by a magnetic field*. Retrieved May 4, 2009, from Department of Electronic Engineering: http://info.ee.surrey.ac.uk/Workshop/advice/coils/force.html#MPF

Wikipedia. (2009, April 30). *Magnetism*. Retrieved May 4, 2009, from Wikipedia: http://en.wikipedia.org/wiki/Magnetism

Wikipedia. (n.d.). *MOSFET*. Retrieved August 15, 2009, from Wikipedia: http://en.wikipedia.org/wiki/MOSFET

Wikipedia. (2009, September 30). *Neodymium magnet*. Retrieved September 30, 2009, from Wikipedia: http://en.wikipedia.org/wiki/Neodymium_magnet

Zero Emission Vehicles Australia. (n.d.). Retrieved September 30, 2009, from Zero Emission Vehicles Australia: http://www.zeva.com.au/tech/motors/BLDC.gif

Zureks. (2008). *File:Solenoid-1.png*. Retrieved May 4, 2009, from Wikipedia: http://en.wikipedia.org/wiki/File:Solenoid-1.png

Appendix A Project Specification

University of Southern Queensland

FACULTY OF ENGINEERING AND SURVEYING

ENG4111/4112 Research Project
PROJECT SPECIFICATION

FOR: **JUSTIN GRIMM (0031210459)**

TOPIC: **09-134 Electromagnetic Linear Actuator**

SUPERVISOR: **Dr Sam Cubero**

SPONSERSHIP: **Nil**

PROJECT AIM: **Design, develop and test high speed, compliant linear actuators for general purpose motion control applications.**

PROGRAMME: **(Issue A, 24 March 2009)**

1. **Research background information relation to electromagnetic linear actuators and electromagnetics in general.**
2. **Choose two designs and model using electromagnetic modelling software.**
3. **Design electronic control circuits and interface software for the purpose of controlling the actuator.**
4. **Manufacture the electromagnetic linear actuator and control circuitry.**
5. **Test the design and identify areas for improvement.**
6. **Submit dissertation.**

AGREED ……………………. **(student)** Dr Sam Cubero ……………………….. **(supervisor)**

Date: 24 / 3 / **2009** **Date: / / 2009**

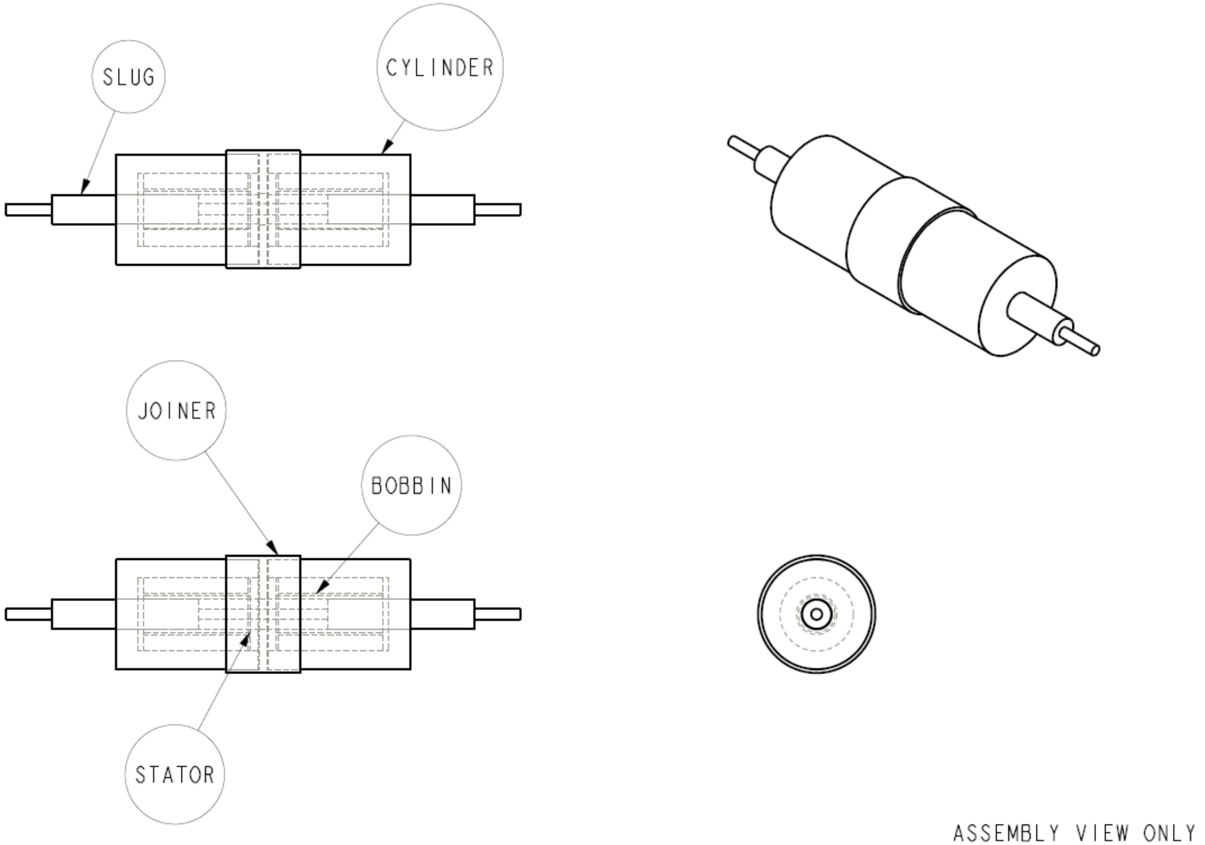
Examiner/Co-Examiner: ……………………………………………… R.A. Morris

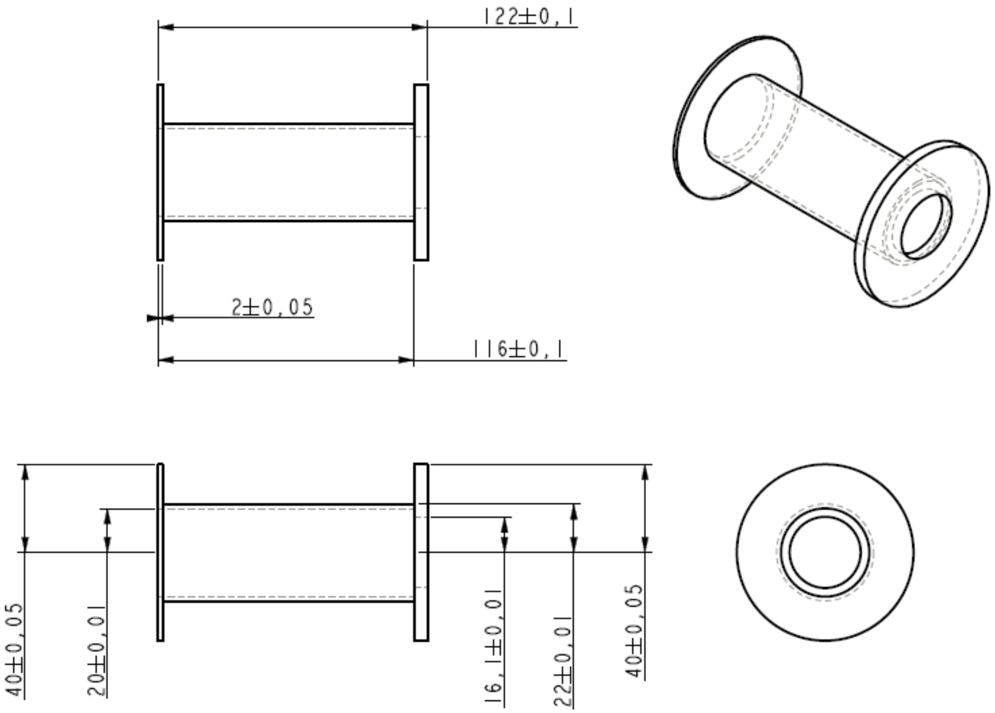
29/4/09

Co-Examiner:

Appendix B Drawings

Appendix B.1 Prototype 1


SLUG
CYLINDER
JOINER
BOBBIN
STATOR
ASSEMBLY VIEW ONLY

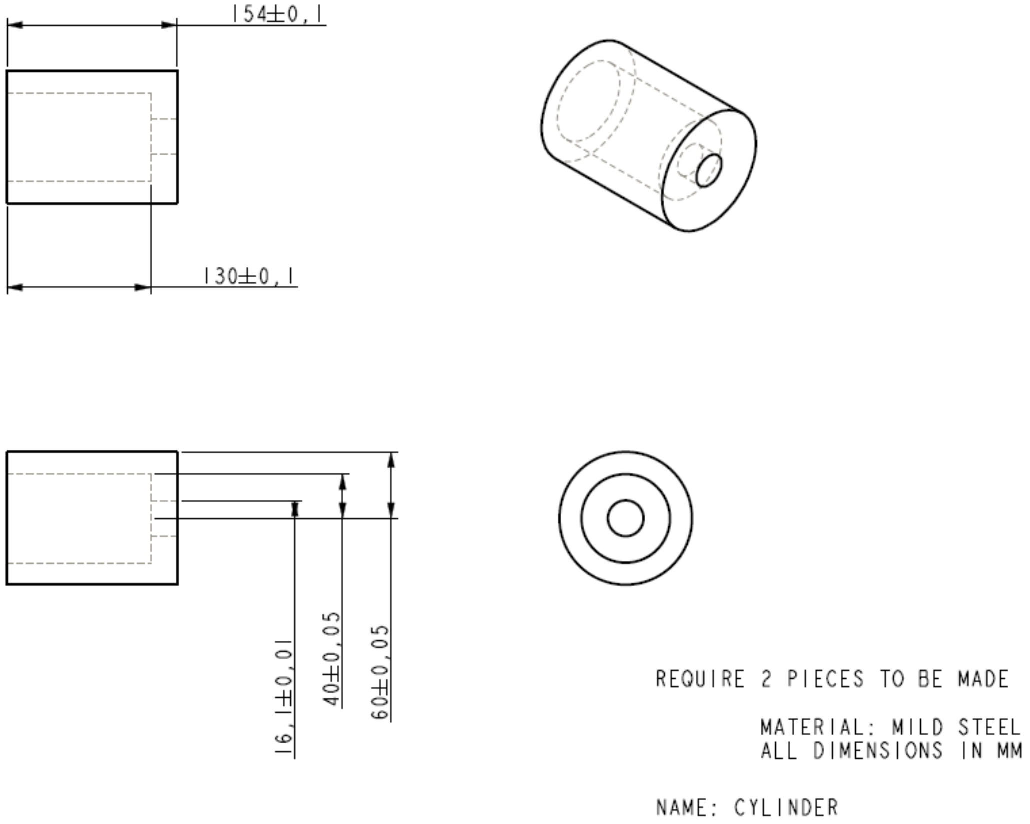


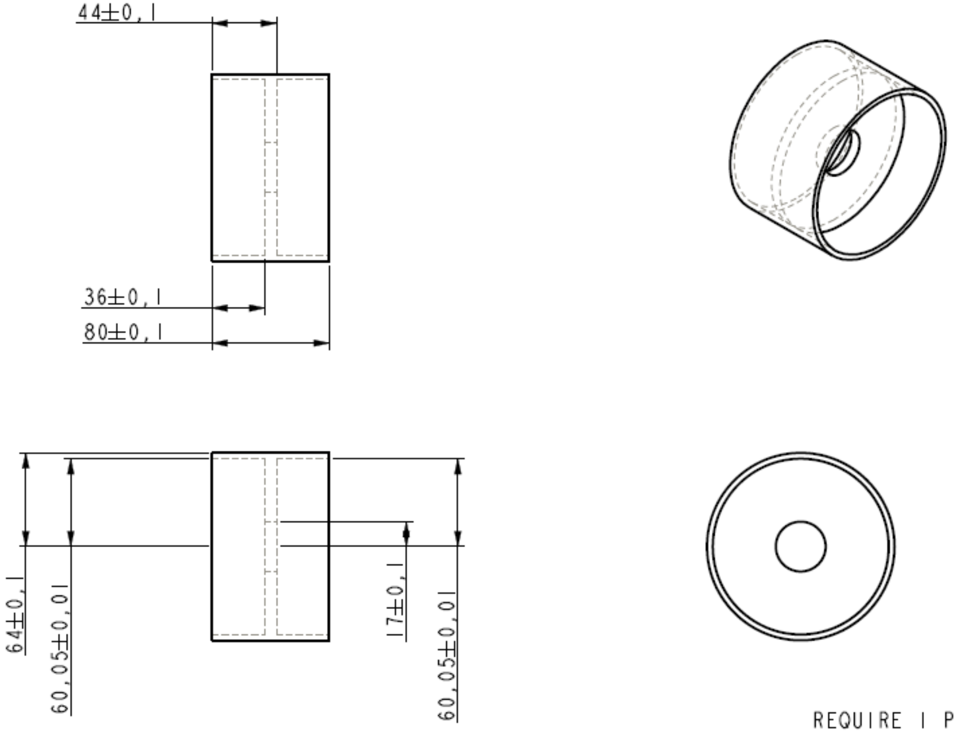
REQUIRE 2 PIECES TO BE MADE

MATERIAL: PLASTIC

ALL DIMENSIONS IN MM

NAME: BOBBIN

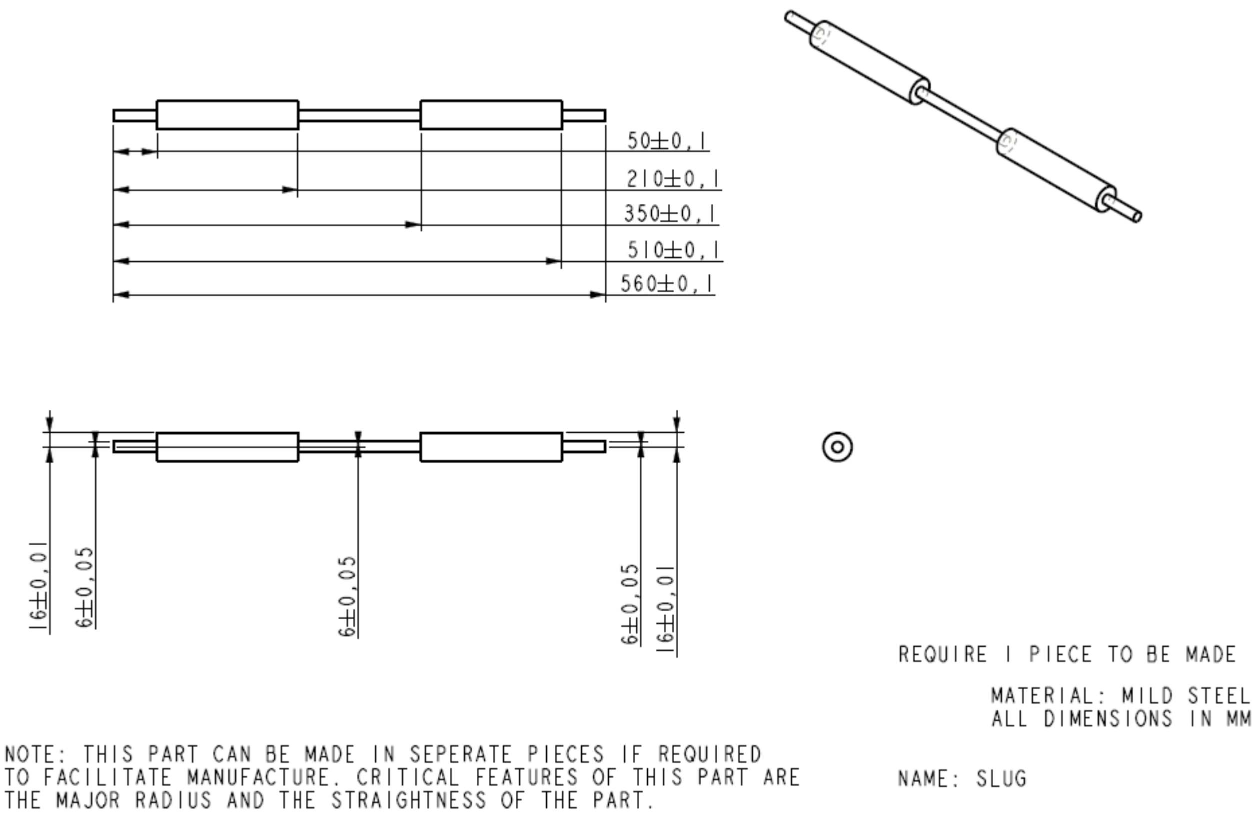

154±0,1
130±0,1
16,1±0,01
40±0,05
60±0,05
REQUIRE 2 PIECES TO BE MADE
MATERIAL: MILD STEEL
ALL DIMENSIONS IN MM
NAME: CYLINDER

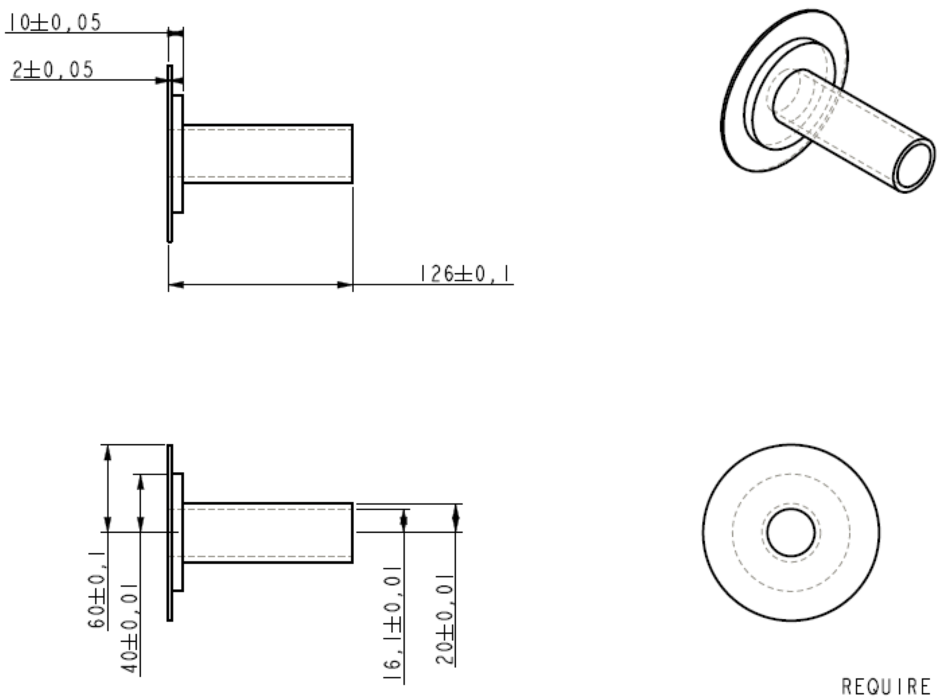


REQUIRE 1 PIECE TO BE MADE

MATERIAL: ALUMINIUM
ALL DIMENSIONS IN MM

NAME: JOINER


50±0,1
210±0,1
350±0,1
510±0,1
560±0,1
16±0,01
6±0,05
6±0,05
6±0,05
16±0,01
REQUIRE 1 PIECE TO BE MADE
MATERIAL: MILD STEEL
ALL DIMENSIONS IN MM
NOTE: THIS PART CAN BE MADE IN SEPERATE PIECES IF REQUIRED
TO FACILITATE MANUFACTURE. CRITICAL FEATURES OF THIS PART ARE
THE MAJOR RADIUS AND THE STRAIGHTNESS OF THE PART.
NAME: SLUG

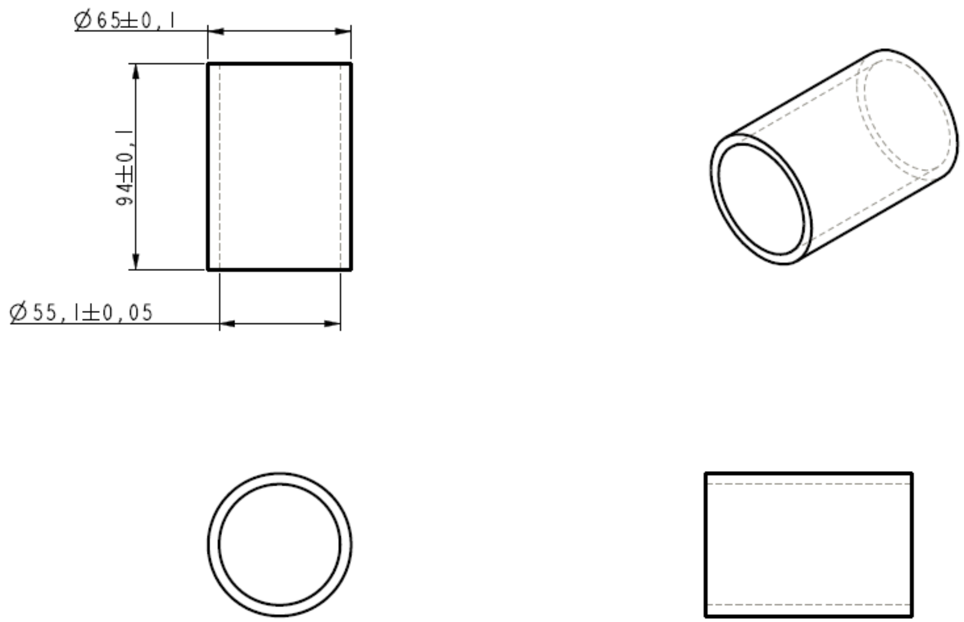


REQUIRE 2 PIECES TO BE MADE

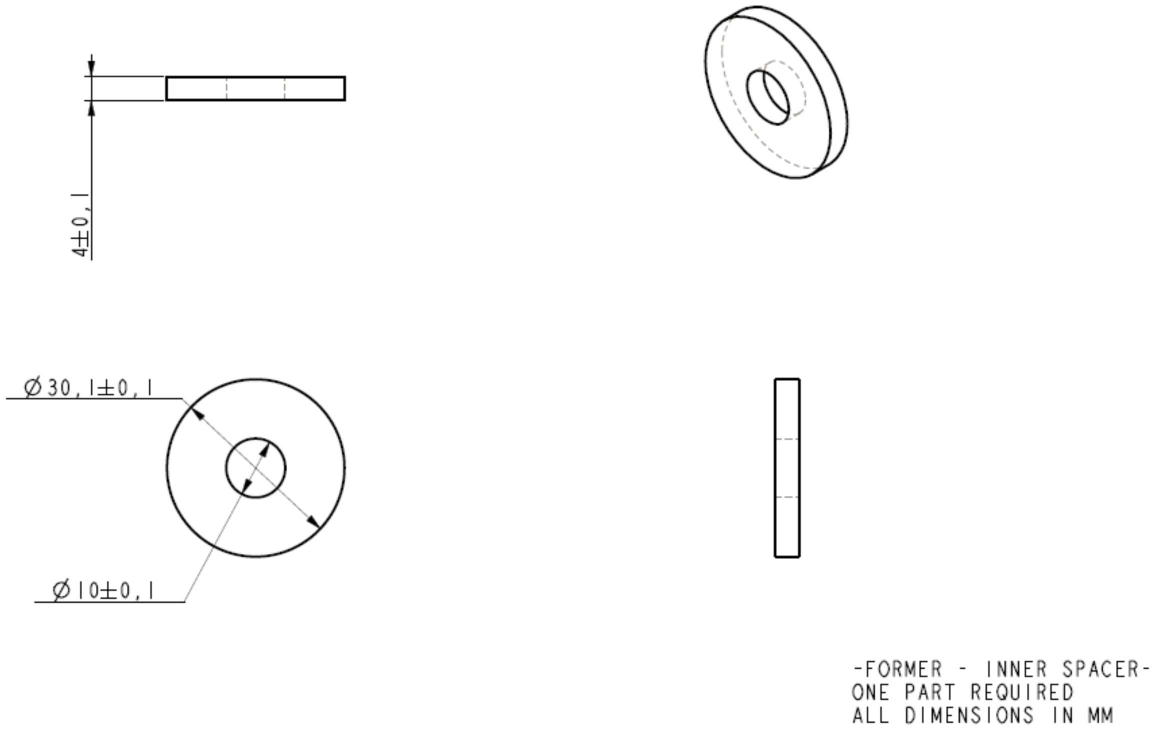
MATERIAL: MILD STEEL
ALL DIMENSIONS IN MM

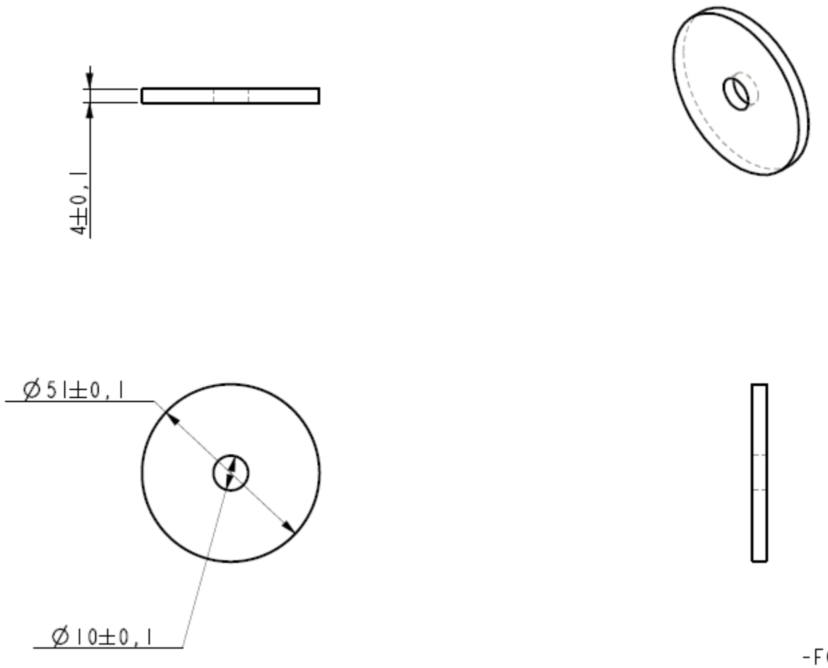
NAME: STATOR

Appendix B.2 Prototype 2

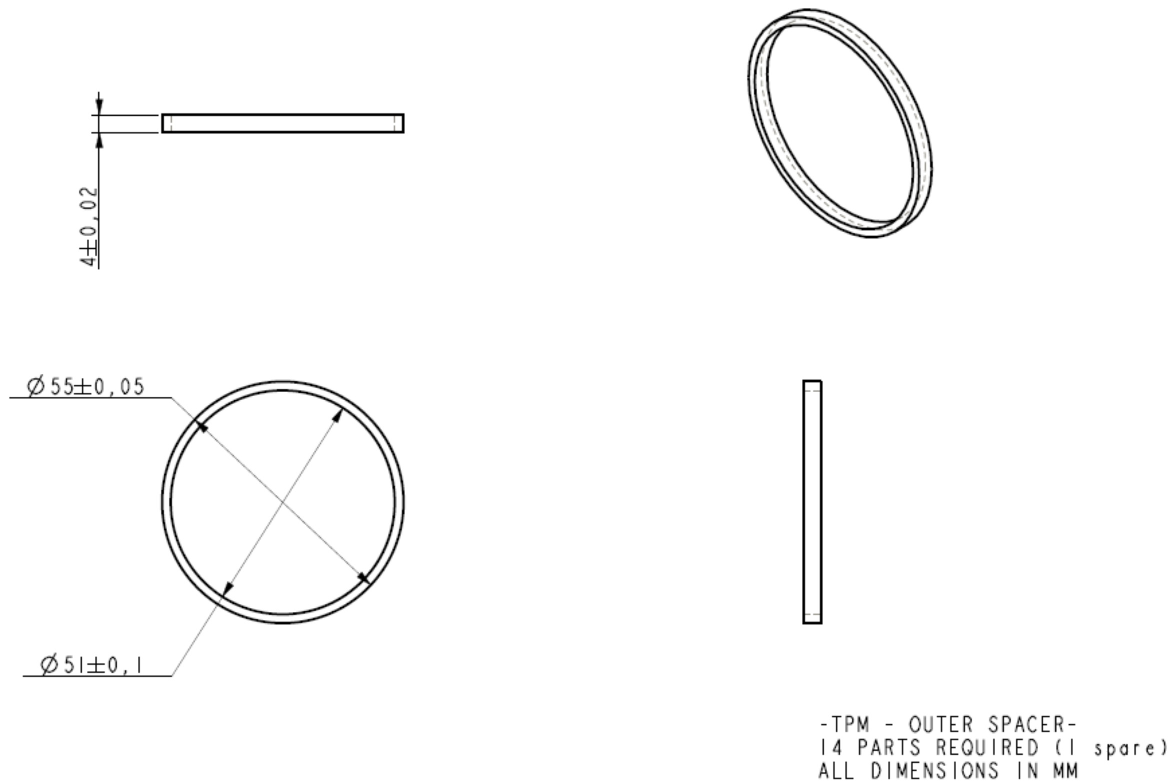


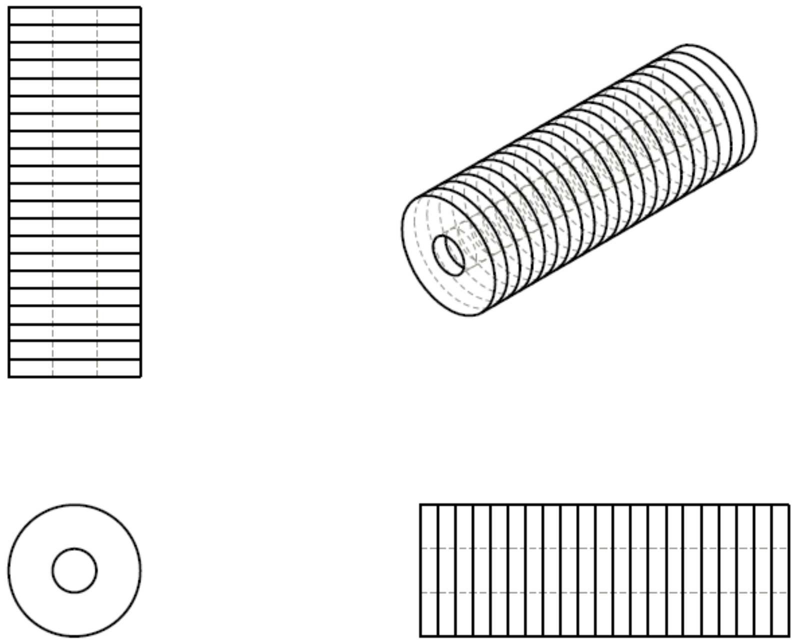
-TPM - OUTER CYLINDER-
ONE PART REQUIRED
ALL DIMENSIONS IN MM


4±0,1
Ø30,1±0,1
Ø10±0,1
-FORMER - INNER SPACER-
ONE PART REQUIRED
ALL DIMENSIONS IN MM

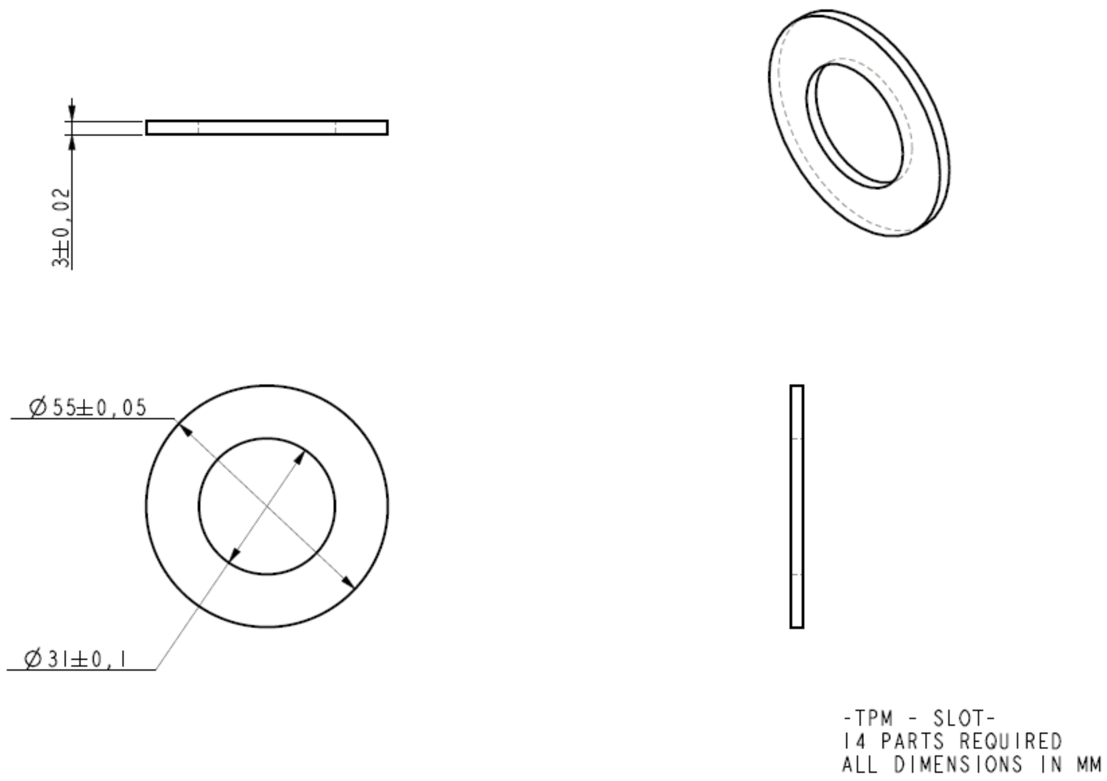


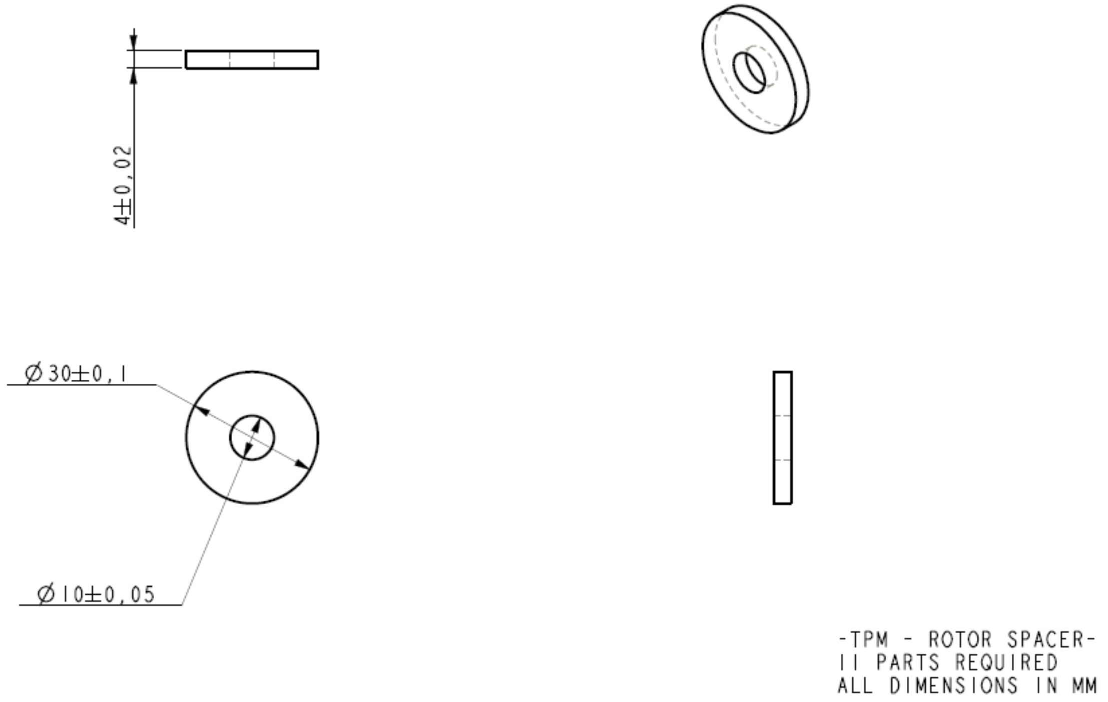
-FORMER - OUTER-
2 PARTS REQUIRED
ALL DIMENSIONS IN MM

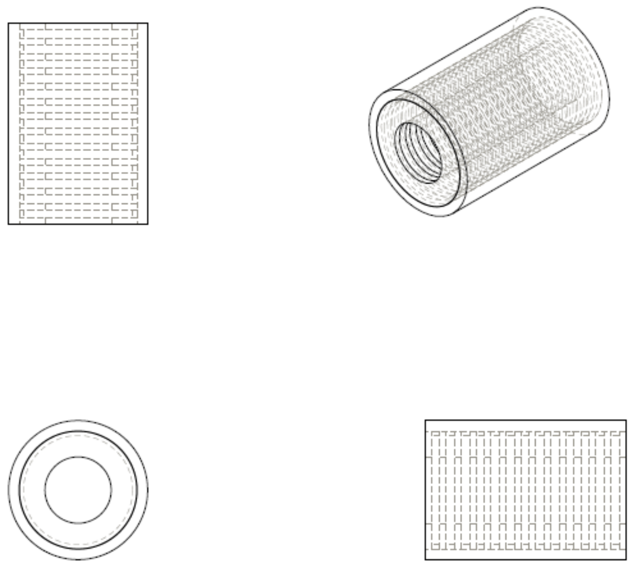

4±0,02
Ø55±0,05
Ø51±0,1
-TPM - OUTER SPACER-
14 PARTS REQUIRED (1 spare)
ALL DIMENSIONS IN MM



FOR INFORMATION ONLY
ASSEMBLY OF THE ROTOR
INNER SPACERS ARE USED TO SEPARATE THE PERMANENT MAGNETS


3±0,02
Ø55±0,05
Ø31±0,1
-TPM - SLOT-
14 PARTS REQUIRED
ALL DIMENSIONS IN MM


4±0,02
Ø30±0,1
Ø10±0,05
-TPM - ROTOR SPACER-
11 PARTS REQUIRED
ALL DIMENSIONS IN MM



FOR INFORMATION ONLY
VIEW OF ASSEMBLED TUBULAR PM LINEAR MOTOR
THE CYLINDER PART HOUSES THE SLOTS AND THE OUTER SPACERS
THE ROTOR WILL BE INSERTED INTO THE CENTRE OF THIS ASSEMBLY

Appendix C Simulations

This appendix contains the electronic simulation files for Prototypes 1 and 2.

Appendix D Microcontroller Code Listing

```
;**********************************************************************
;
;    Filename:
;    ELMA.asm
;
;    Date:
;    30/09/09
;
;    File Version:
;    1.0
;
;    Author:
;    Justin Grimm
;
;    Multiplier subroutine Adapted from
;    http://www.convict.lu/Jeunes/Math/Fast_operations.htm
;    Claude Baumann 2002; updated 2006 thanks to L. Armstrong
;
;**********************************************************************
;
;**********************************************************************
;
;    Notes:     Port assignments               tris
;
;       RA0            AN0     Position             1
;       RA1            AN1     PWM09                0
;       RA2            AN2     PWM10                0
;       RA3            AN3     PWM11                0
;       RA4                    PWM12                0
;       RA5            AN4     PWM13                0
;       RB0                    PWM1                 0
;       RB1                    PWM2                 0
;       RB2                    PWM3                 0
;       RB3                    PWM4                 0
;       RB4                    PWM5                 0
;       RB5                    PWM6                 0
;       RB6                    PWM7                 0
;       RB7                    PWM8                 0
;       RC0                                         0
```

```
;       RC1                                     0
;       RC2                                     0
;       RC3                                     0
;       RC4                                     0
;       RC5                                     0
;       RC6                 232/tx              0
;       RC7                 232/rx              1
;       RD0                                     0
;       RD1                                     0
;       RD2                                     0
;       RD3                                     0
;       RD4                                     0
;       RD5                                     0
;       RD6                                     0
;       RD7                                     0
;       RE0             AN5                     0
;       RE1             AN6                     0
;       RE2             AN7                     0
;
;***********************************************************************
;
; rs232 receive commands
;
;
; 0xf7  position requested
; 0xf8  sp requested
;
;***********************************************************************



;***** VARIABLE DEFINITIONS

w                       equ     0
f                       equ     1
w_temp                  EQU      0x20                   ; variable used for context saving
status_temp             EQU      0x21                   ; variable used for context saving
pclath_temp             EQU     0x22                    ; variable used for context saving
pclath                  equ     0x0a
status                  equ     0x03
z                       equ     2
c                       equ     0
dc                      equ     1
rp0                     equ     5
```

```
tmr0        equ   0x01
pcl         equ   0x02
fsr         equ   0x04
porta       equ   0x05
pot1        equ   0
PWM09       equ   1
PWM10       equ   2
PWM11       equ   3
PWM12       equ   4
PWM13       equ   5
portb       equ   0x06
PWM01       equ   0
PWM02       equ   1
PWM03       equ   2
PWM04       equ   3
PWM05       equ   4
PWM06       equ   5
PWM07       equ   6
PWM08       equ   7
portc       equ   0x07
portd       equ   0x08
porte       equ   0x09
intcon      equ   0x0b
rbif        equ   0
intf        equ   1
t0if        equ   2
rbie        equ   3
inte        equ   4
t0ie        equ   5
peie        equ   6
gie         equ   7
pir1        equ   0x0c
tmr1if      equ   0
tmr2if      equ   1
ccp1if      equ   2
sspif       equ   3
txif        equ   4
rcif        equ   5
adif        equ   6
pspif       equ   7
pir2        equ   0x0d
tmr1l       equ   0x0e
tmr1h       equ   0x0f
t1con       equ   0x10
```

```
tmr2            equ     0x11
t2con           equ     0x12
sspbuf          equ     0x13
sspcon          equ     0x14
ccpr1l          equ     0x15
ccpr1h          equ     0x16
ccp1con         equ     0x17
rcsta           equ     0x18
txreg           equ     0x19
rcreg           equ     0x1a
ccpr2l          equ     0x1b
ccpr2h          equ     0x1c
ccp2con         equ     0x1d
adresh          equ     0x1e
adcon0          equ     0x1f
option_reg      equ     0x81
trisa           equ     0x85
trisb           equ     0x86
trisc           equ     0x87
trisd           equ     0x88
trise           equ     0x89
pie1            equ     0x8c
pie2            equ     0x8d
pcon            equ     0x8e
pr2             equ     0x92
sspadd          equ     0x93
sspstat         equ     0x94
txsta           equ     0x98
txen            equ     5
spbrg           equ     0x99
adresl          equ     0x9e
adcon1          equ     0x9f
txdata          equ     0x23                                    ; data to be transmitted over sci
rcsave          equ     0x24                                    ; register to save received byte drom sci
pos_sp          equ     0x25                                    ; the position setpoint in mm
position        equ     0x26                                    ; actual position in mm
pos_error       equ     0x27                                    ; absolute position error
flags           equ     0x28                                    ; general purpose flags register
move_up_req     equ     0                                       ; means position > pos_sp
pos_add         equ     1                                       ; added to position for hysteresis adjustment last scan
tempx16         equ     0x29                                    ; multiplier 1 for 16 bit multiplier sub
tempy16         equ     0x2a                                    ; multiplier 2 for 16 bit multiplier sub
wait4           equ     0x2b                                    ; Temporary register
temp1           equ     0x2c                                    ; Temporary register
```

```
result16_L      equ     0x2d                                    ; Result LSB for 16 bit multiplier sub
result16_H      equ     0x2e                                    ; Result MSB for 16 bit multiplier sub
tmr0_cnt        equ     0x2f                                    ; count to determine when the ISR has been called 32 times.
                                                                ; This is the 1kHz PWM period.
PWM01_SP        equ     0x30                                    ; PWM setpoints for each coil. 0=fully reverse, 16=off, 31=fully forward.
PWM02_SP        equ     0x31
PWM03_SP        equ     0x32
PWM04_SP        equ     0x33
PWM05_SP        equ     0x34
PWM06_SP        equ     0x35
PWM07_SP        equ     0x36
PWM08_SP        equ     0x37
PWM09_SP        equ     0x38
PWM10_SP        equ     0x39
PWM11_SP        equ     0x3a
PWM12_SP        equ     0x3b
PWM13_SP        equ     0x3c
drive_byte1     equ     0x3d                                    ; Drive data from table read
drive_byte2     equ     0x3e
drive_byte3     equ     0x3f
drive_byte4     equ     0x4a
tempx16_H       equ     0x4b                                    ; temporary registers from multiplier routine
tempy16_H       equ     0x4c
adresl_reg      equ     0x4d                                    ; holding register for A/D converter low byte
pos_last        equ     0x4e                                    ; last scan position
pos_temp        equ     0x4f                                    ; temporary position location



;**********************************************************************
; PROGRAM START

        ORG     0x000                                           ; processor reset vector
        clrf    pclath                                          ; ensure page bits are cleared
        goto    boot                                            ; go to beginning of program


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;ISR
;;;;This interrupt vector is called every 31.25us (1kHz/32) and services the PWM outputs.
;;;;PWM is 0-100% duty cycle with 50% being off, 0% being fully reverse and 100% being fully forward.
;;;;Resolution is a total of 32 steps, 0-31 being 0-100% duty cycle. Therefore 0-15 is 100% to 0% reverse
```

```
;;;;and 16-31 is 0% to 100% forward.

        ORG     0x004                                   ; interrupt vector location

        MOVWF   w_temp                                  ;Copy W to TEMP register
        SWAPF   status,w                                ;Swap status to be saved into W
        CLRF    status                                  ;bank 0, regardless of current bank, Clears IRP,RP1,RP0
        MOVWF   status_temp                             ;Save status to bank zero STATUS_TEMP register
        MOVF    pclath,w                                ;Only required if using pages 1, 2 and/or 3
        MOVWF   pclath_temp                             ;Save PCLATH into W
        CLRF    pclath                                  ;Page zero, regardless of current page

isr
        movlw   0x5d                                    ; 256-156-2-5 = 0x5d. Timer transitions FF to 00 after 156 instruction
                                                        ; cycles = 31.2us.
        movwf   tmr0
        incfsz  tmr0_cnt,f                              ; increment the PWM period counter, skip next instruction if transition to zero.
                                                        ; (2 Tcy)
        goto    pwm_period_not_fin                      ; 1ms period not finished (2 Tcy)
        goto    pwm_period_fin                          ; 1ms period finished (2 Tcy)

pwm_period_not_fin
        movlw   0xe1                                    ; tmr0_cnt - 0xe1 -> temp1
        subwf   tmr0_cnt,w                              ; result is 0-31.
        movwf   temp1

        movf    temp1,w
        subwf   PWM01_SP,w                              ; PWM01_SP - temp1 -> w. If c = 0 then temp1 > PWM01_SP and the PWM output
                                                        ; should be 0.
        btfss   status,c                                ; test carry flag, if zero then clear PWM output.
        bcf     portb,PWM01

        movf    temp1,w
        subwf   PWM02_SP,w                              ; PWM02_SP - temp1 -> w. If c = 0 then temp1 > PWM02_SP and the PWM output
                                                        ; should be 0.
        btfss   status,c                                ; test carry flag, if zero then clear PWM output.
        bcf     portb,PWM02

        movf    temp1,w
        subwf   PWM03_SP,w                              ; PWM03_SP - temp1 -> w. If c = 0 then temp1 > PWM03_SP and the PWM output
                                                        ; should be 0.
        btfss   status,c                                ; test carry flag, if zero then clear PWM output.
        bcf     portb,PWM03
```

```
        movf    temp1,w
        subwf   PWM04_SP,w                              ; PWM04_SP - temp1 -> w. If c = 0 then temp1 > PWM04_SP and the PWM output
                                                        ; should be 0.
        btfss   status,c                                ; test carry flag, if zero then clear PWM output.
        bcf     portb,PWM04

        movf    temp1,w
        subwf   PWM05_SP,w                              ; PWM05_SP - temp1 -> w. If c = 0 then temp1 > PWM05_SP and the PWM output
                                                        ; should be 0.
        btfss   status,c                                ; test carry flag, if zero then clear PWM output.
        bcf     portb,PWM05

        movf    temp1,w
        subwf   PWM06_SP,w                              ; PWM06_SP - temp1 -> w. If c = 0 then temp1 > PWM06_SP and the PWM output
                                                        ; should be 0.
        btfss   status,c                                ; test carry flag, if zero then clear PWM output.
        bcf     portb,PWM06

        movf    temp1,w
        subwf   PWM07_SP,w                              ; PWM07_SP - temp1 -> w. If c = 0 then temp1 > PWM07_SP and the PWM output
                                                        ; should be 0.
        btfss   status,c                                ; test carry flag, if zero then clear PWM output.
        bcf     portb,PWM07

        movf    temp1,w
        subwf   PWM08_SP,w                              ; PWM08_SP - temp1 -> w. If c = 0 then temp1 > PWM08_SP and the PWM output
                                                        ; should be 0.
        btfss   status,c                                ; test carry flag, if zero then clear PWM output.
        bcf     portb,PWM08

        movf    temp1,w
        subwf   PWM09_SP,w                              ; PWM09_SP - temp1 -> w. If c = 0 then temp1 > PWM09_SP and the PWM output
                                                        ; should be 0.
        btfss   status,c                                ; test carry flag, if zero then clear PWM output.
        bcf     porta,PWM09

        movf    temp1,w
        subwf   PWM10_SP,w                              ; PWM10_SP - temp1 -> w. If c = 0 then temp1 > PWM10_SP and the PWM output
                                                        ; should be 0.
        btfss   status,c                                ; test carry flag, if zero then clear PWM output.
        bcf     porta,PWM10

        movf    temp1,w
        subwf   PWM11_SP,w                              ; PWM11_SP - temp1 -> w. If c = 0 then temp1 > PWM11_SP and the PWM output
```

```
                                                ; should be 0.
        btfss   status,c                        ; test carry flag, if zero then clear PWM output.
        bcf     porta,PWM11

        movf    temp1,w
        subwf   PWM12_SP,w                      ; PWM12_SP - temp1 -> w. If c = 0 then temp1 > PWM12_SP and the PWM output
                                                ; should be 0.
        btfss   status,c                        ; test carry flag, if zero then clear PWM output.
        bcf     porta,PWM12

        movf    temp1,w
        subwf   PWM13_SP,w                      ; PWM13_SP - temp1 -> w. If c = 0 then temp1 > PWM13_SP and the PWM output
                                                ; should be 0.
        btfss   status,c                        ; test carry flag, if zero then clear PWM output.
        bcf     porta,PWM13

        goto    end_isr


pwm_period_fin
        movlw   0xe0                            ; reset counter to 256-32=0xe0
        movwf   tmr0_cnt
        movlw   0xff                            ; set all PWM outputs to 1 on port b
        movwf   portb
        movlw   0x1f                            ; set PWM9, 10, 11, 12 and 13 to 1 on port a
        iorwf   porta,f


end_isr
        bcf     intcon,t0if                     ; clear interrupt flag

        MOVF    pclath_temp, W                  ;Restore PCLATH
        MOVWF   pclath                          ;Move W into PCLATH
        SWAPF   status_temp,w                   ;Swap STATUS_TEMP register into W
                                                ;(sets bank to original state)
        MOVWF   status                          ;Move W into STATUS register
        SWAPF   w_temp,f                        ;Swap W_TEMP
        SWAPF   w_temp,w                        ;Swap W_TEMP into W

        retfie                                  ; return from interrupt
```

```
;;;; ISR
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;; INITIALISE


init
        bcf     intcon,gie
        clrf    porta
        clrf    portb
        clrf    portc
        clrf    portd
        clrf    porte
        clrf    position
        clrf    wait4
        bsf     status,rp0                                      ; bank1
        movlw   0x01                                            ; set port directions- ra0 analogue in
        movwf   trisa
        movlw   0x00                                            ; 00000000
        movwf   trisb
        movlw   0x80                                            ; 10000000
        movwf   trisc
        movlw   0x00                                            ; 00000000
        movwf   trisd
        movlw   0x00
        movwf   trise
        movlw   0x89                                            ; RB P/U disabled, prescaler WDT 2:1, int on falling edge, timer 0
                                                                ; uses internal clock
        movwf   option_reg
        bcf     0x83,rp0                                        ; bank0
        movlw   0x59                                            ; initialise position setpoint and last position to 89mm
        movwf   pos_sp
        movwf   pos_last
        clrf    result16_L
        clrf    result16_H
        clrf    tmr0_cnt
        clrf    temp1

initad
        movlw   0x81                                            ; Fosc/32 clk,channel ra0,a/d on
        movwf   adcon0                                          ;
        bsf     status,rp0                                      ; bank1
        movlw   0x8e                                            ; High bits from result are in lower bits of ADRESH
        movwf   adcon1                                          ; set ra0 to analog
```

```
initsci                                                 ; txif is set when tx finished, to tx, load txreg check txif
        movlw   0x81                                    ; 2400 baud at 20Mhz
        movwf   spbrg
        movlw   0x20                                    ; clr brgh (low baud rate),async op, enabled
        movwf   txsta                                   ;
        bcf     0x83,5                                  ; bank0
        movlw   0x80                                    ; enable serial
        movwf   rcsta                                   ;
        clrf    txreg                                   ; clr rx tx regs
        clrf    rcreg                                   ;
        movlw   0x90                                    ; enable receive
        movwf   rcsta

initint
        clrf    intcon                                  ; disable all interrupts
        bsf     intcon,t0ie                             ; enable interrupt on timer 0
        movlw   0x00                                    ; disable peripheral interrupts
        movwf   pie1

inittmr0
        movlw   0x62                                    ; 256-156-2 = 0x61. Timer transitions FF to 00 after 156 instruction cycles = 31.2us.
        movwf   tmr0
        movlw   0xe0                                    ; load 256-32=0xe0 into ISR counter
        movwf   tmr0_cnt                                ;
        movlw   0xff                                    ; set all PWM outputs to 1 on port b
        movwf   portb
        movlw   0x1f                                    ; set PWM9, 10, 11, 12 and 13 to 1 on port a
        iorwf   porta,f

        bsf     intcon,gie                              ; enable global interrupts (temp disabled)

        return

;;;; INITIALISE
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;; SUBROUTINES

;; Read analogue input.
;; Inputs: None
;; Outputs: Result in ADRESH and ADRESL.
;; Length: 17 Tcy + wait time for A/D conversion.

get_ad                                                  ; result in ADRES reg
```

```
        movlw   0x02                                            ; wait 2us sampling time
        movwf   wait4
adloop
        nop
        decfsz  wait4,f
        goto    adloop
        bsf     adcon0,2                                        ; start ad conversion
waitad
        btfsc   adcon0,2                                        ; wait for a/d conversion
        goto    waitad
        return

tx_data
        movf    txdata,w                                        ; move transmit data to tx
        movwf   txreg
wait_tx
        btfsc   pir1,txif                                       ; tx finished?
        return                                                  ; yes
        goto    wait_tx                                         ; no


;; Multiply 2 * 8 bit registers. Adapted from http://www.convict.lu/Jeunes/Math/Fast_operations.htm
;; tempx16 * tempy16 -> result16_H and result16_L

mul16
        clrf    result16_L
        clrf    result16_H
        clrf    tempx16_H
        clrf    tempy16_H
mul16_loop
        btfsc   tempy16,0
        call    add16
        bcf     status,c
        rrf     tempy16_H,f
        rrf     tempy16,f
        bcf     status,c
        rlf     tempx16,f
        rlf     tempx16_H,f
        movf    tempy16,f
        btfss   status,z
        goto    mul16_loop
        movf    tempy16_H,f
```

```
        btfss  status,z
        goto   mul16_loop
        return
add16
        movf   tempx16,w
        addwf  result16_L
        btfsc  status,c
        incf   result16_H
        movf   tempx16_H,w
        addwf  result16_H
        return

;;;; SUBROUTINES
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;; MAIN PROGRAMME

boot
       clrwdt
       call    init                                       ; initialise device, a/d, sci, int

main

check_sci_rec                                             ; check if a byte received over the sci
       btfss   pir1,rcif                                  ; check for received bit
       goto    get_position                               ; no
       bcf     pir1,rcif                                  ; yes, clr flag
       movlw   0x80                                       ; disable receive
       movwf   rcsta
       movf    rcreg,w                                    ; save received byte
       movwf   rcsave

       btfss   rcsave,7                                   ; see if msb 1, valid data requests have the msb as 1.
       goto    _pos_sp_rec                                ; no, not valid data request, must be position setpoint. Position setpoint is 0-178mm

       movlw   0xf7                                       ; see if current position data wanted
       xorwf   rcsave,w
       btfsc   status,z
       goto    send_position                              ; yes

       movlw   0xf8                                       ; see if setpoint data wanted
       xorwf   rcsave,w
       btfsc   status,z
       goto    send_sp                                    ; yes
```

```
_pos_sp_rec
        movf    rcsave,w                                ; copy position setpoint to pos_sp register
        movwf   pos_sp
        goto    check_sci_cont                          ; continue

send_position
        movf    position,w                              ; send position data to PC.
        movwf   txdata
        call    tx_data
        goto    check_sci_cont

send_sp
        movf    pos_sp,w                                ; send position data to PC.
        movwf   txdata
        call    tx_data
        goto    check_sci_cont

check_sci_cont
        movlw   0x90                                    ; enable receive
        movwf   rcsta


get_position
        call    get_ad                                  ; get analogue input raw data

        bsf     status,rp0                              ; bank1
        movf    adresl,w                                ; save adresl to bank 0
        bcf     0x83,5                                  ; bank0
        movwf   adresl_reg

        movlw   0xfe                                    ; clamp adresl_reg to a max of 254 if above 254
        subwf   adresl_reg,w                            ; adresl_reg - 0xfe -> w
        btfss   status,c                                ; see if adresl_reg < 0xfe
        goto    get_position_cont                       ; yes, continue
        movlw   0xfe                                    ; no, set adresl_reg to 0xfe
        movwf   adresl_reg

get_position_cont

position_cont2
        movlw   0x00                                    ; see if ad result in bank 0
        xorwf   adresh,w
        btfsc   status,z
        goto    get_pos_bank0                           ; yes
```

```
        movlw   0x01                                        ; see if ad result in bank 1
        xorwf   adresh,w
        btfsc   status,z
        goto    get_pos_bank1                               ; yes
        movlw   0x02                                        ; see if ad result in bank 2
        xorwf   adresh,w
        btfsc   status,z
        goto    get_pos_bank2                               ; yes
        movlw   0x03                                        ; see if ad result in bank 3
        xorwf   adresh,w
        btfsc   status,z
        goto    get_pos_bank3                               ; yes
        goto    get_pos_bank3                               ; error



get_pos_bank0
        movlw   0x04
        movwf   pclath
        movf    adresl_reg,w
        call    pos_table1                                  ; returns with rotor position in mm in w
        movwf   position                                    ; save actual position to register
;       clrf    pclath
        goto    control
get_pos_bank1
        movlw   0x05
        movwf   pclath
        movf    adresl_reg,w
        call    pos_table2                                  ; returns with rotor position in mm in w
        movwf   position                                    ; save actual position to register
;       clrf    pclath
        goto    control
get_pos_bank2
        movlw   0x06
        movwf   pclath
        movf    adresl_reg,w
        call    pos_table3                                  ; returns with rotor position in mm in w
        movwf   position                                    ; save actual position to register
;       clrf    pclath
        goto    control
get_pos_bank3
        movlw   0x07
        movwf   pclath
        movf    adresl_reg,w
```

```
        call    pos_table4                          ; returns with rotor position in mm in w
        movwf   position                            ; save actual position to register
;       clrf    pclath
        goto    control

control

        movf    pos_last,w                          ; see if there was a change in position
        xorwf   position,w
        btfss   status,z
        goto    _pos_adj_req                        ; yes
        btfsc   flags,pos_add                       ; no, see if last scan added or subtracted from position
        goto    _pos_gt_last                        ; added, retain added value
        goto    _pos_lt_last                        ; subtracted, retain added value
_pos_adj_req
        movf    position,w                          ; save raw position
        movwf   pos_temp
        subwf   pos_last,w                          ; see if moving up or down, pos_last - position -> w
        btfsc   status,c                            ; see if carry = 0 (means position > pos_last)
        goto    _pos_lt_last                        ; position is less than last position
_pos_gt_last
        movlw   0x02                                ; add 2 to position to account for hysteresis in pot
        addwf   position,f
        bsf     flags,pos_add                       ; set 'position adjustment added' flag
        goto    _pos_adj_cont
_pos_lt_last
        movlw   0x02                                ; subtract 2 from position to account for hysteresis in pot
        subwf   position,f
        bcf     flags,pos_add                       ; clear 'position adjustment added' flag
_pos_adj_cont
        movf    pos_temp,w                          ; save raw position to position last
        movwf   pos_last

        movf    position,w                          ; find position absolute error
        subwf   pos_sp,w                            ; pos_sp - position -> w
        movwf   pos_error                           ; save position error
        bsf     flags,move_up_req                   ; clear flag ;;reversed
        btfss   status,c                            ; see if carry = 0 (means position > pos_sp)
        goto    _inverse_error                      ; yes, find inverse of error
        goto    _control_cont                       ; no, continue
_inverse_error
        movf    pos_sp,w                            ; find position absolute error
        subwf   position,w                          ; position - pos_sp -> w
        movwf   pos_error                           ; save position error
```

```
        bcf     flags,move_up_req                           ; reversed

_control_cont
;       bcf     status,c                                    ; divide pos_error by 2. Code below scales a max error of 178mm to 22
;       rrf     pos_error,f
;       bcf     status,c                                    ; divide pos_error by 2
;       rrf     pos_error,f
;       bcf     status,c                                    ; divide pos_error by 2
;       rrf     pos_error,f                                 ; **commented out to increase gain**
        movlw   0x0f                                        ; clamp error to a max of 15 if above 15
        subwf   pos_error,w                                 ; pos_error - 0x0f -> w
        btfss   status,c                                    ; see if pos_error < 0x0f
        goto    get_drive_data                              ; yes, continue
        movlw   0x0f                                        ; no, set pos_error to 0x0f
        movwf   pos_error

get_drive_data
        movlw   0x04                                        ; Multiply current position by 4 to get correct index in table
        movwf   tempx16
        movf    position,w
        movwf   tempy16
        call    mul16
        movf    result16_H,w                                ; Set index 2 most significant bits
        movwf   pclath
        bcf     pclath,4                                    ; set page 1
        bsf     pclath,3
        movlw   0x01
        addwf   result16_L,f                                ; increment index
        btfsc   status,c
        incf    result16_H,f
        movf    result16_L,w                                ; Set index 8 least siginificant bits

        call    drive_table                                 ; get coil drive data
        movwf   drive_byte1                                 ; save byte 1

        movlw   0x01
        addwf   result16_L,f                                ; increment index
        btfsc   status,c
        incf    result16_H,f
        movf    result16_H,w                                ; Set index 2 most significant bits
        movwf   pclath
        bcf     pclath,4                                    ; set page 1
        bsf     pclath,3
        movf    result16_L,w                                ; Set index 8 least siginificant bits
```

```
        call    drive_table                                 ; get coil drive data
        movwf   drive_byte2                                 ; save byte 2

        movlw   0x01
        addwf   result16_L,f                                ; increment index
        btfsc   status,c
        incf    result16_H,f
        movf    result16_H,w                                ; Set index 2 most significant bits
        movwf   pclath
        bcf     pclath,4                                    ; set page 1
        bsf     pclath,3
        movf    result16_L,w                                ; Set index 8 least siginificant bits
        call    drive_table                                 ; get coil drive data
        movwf   drive_byte3                                 ; save byte 3

        movlw   0x01
        addwf   result16_L,f                                ; increment index
        btfsc   status,c
        incf    result16_H,f
        movf    result16_H,w                                ; Set index 2 most significant bits
        movwf   pclath
        bcf     pclath,4                                    ; set page 1
        bsf     pclath,3
        movf    result16_L,w                                ; Set index 8 least siginificant bits
        call    drive_table                                 ; get coil drive data
        movwf   drive_byte4                                 ; save byte 4

        clrf    pclath                                      ; Clear page index

apply_control

_check_coil_1
        btfss   drive_byte1,7                               ; see if msb = 0
        goto    _check_coil_1_msb_0                         ; yes
        btfss   drive_byte1,6                               ; no, see if type 10
        goto    _check_coil_1_10                            ; yes,
_check_coil_1_type_11                                       ; get here if type 11
        movlw   0x1f
        movf    pos_error,f                                 ; see if error = 0
        btfsc   status,z
        goto    _check_coil_1_type_11_zerror; yes
        movf    pos_error,w                                 ; no, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM01_SP                                    ; move result to coil PWM setpoint (0-15)
```

```
        goto    _check_coil_1_end                           ; continue
_check_coil_1_type_11_zerror
        movwf   PWM01_SP                                    ; yes, drive coil positive
        goto    _check_coil_1_end                           ; continue
_check_coil_1_msb_0
        btfss   drive_byte1,6                               ; see if type 01
        goto    _check_coil_1_end                           ; no, get here if error
_check_coil_1_01                                            ; yes
        movlw   0x0f                                        ; move 15 to w in case error = 0
        movf    pos_error,f                                 ; see if error = 0
        btfsc   status,z
        goto    _check_coil_1_01_zerror                     ;
        btfss   flags,move_up_req                           ; no, see if up movement required
        goto    _check_coil_1_01_dn_rq                      ; no, down movement required
        movf    pos_error,w                                 ; yes, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM01_SP                                    ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_1_end
_check_coil_1_01_zerror
        movwf   PWM01_SP                                    ; yes, drive coil to zero
        goto    _check_coil_1_end
_check_coil_1_01_dn_rq
        movf    pos_error,w                                 ; yes, 0x0f + pos_error -> w
        addlw   0x0f
        movwf   PWM01_SP                                    ; move result to coil PWM setpoint (16-31)
        goto    _check_coil_1_end
_check_coil_1_10
        movlw   0x0f                                        ; move 15 to w in case error = 0
        movf    pos_error,f                                 ; see if error = 0
        btfsc   status,z
        goto    _check_coil_1_10_zerror ;
        btfss   flags,move_up_req                           ; no, see if up movement required
        goto    _check_coil_1_10_dn_rq                      ; no, down movement required
        movf    pos_error,w                                 ; yes, 0x0f + pos_error -> w
        addlw   0x0f
        movwf   PWM01_SP                                    ; move result to coil PWM setpoint (16-31)
        goto    _check_coil_1_end
_check_coil_1_10_zerror
        movwf   PWM01_SP                                    ; yes, drive coil to zero
        goto    _check_coil_1_end
_check_coil_1_10_dn_rq
        movf    pos_error,w                                 ; yes, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM01_SP                                    ; move result to coil PWM setpoint (0-15)
```

```
        goto    _check_coil_1_end
_check_coil_1_end

_check_coil_2
        btfss   drive_byte1,5                               ; see if msb = 0
        goto    _check_coil_2_msb_0                         ; yes
        btfss   drive_byte1,4                               ; no, see if type 10
        goto    _check_coil_2_10                            ; yes,
_check_coil_2_type_11                                       ; get here if type 11
        movlw   0x1f
        movf    pos_error,f                                 ; see if error = 0
        btfsc   status,z
        goto    _check_coil_2_type_11_zerror; yes
        movf    pos_error,w                                 ; no, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM02_SP                                    ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_2_end                           ; continue
_check_coil_2_type_11_zerror
        movwf   PWM02_SP                                    ; yes, drive coil positive
        goto    _check_coil_2_end                           ; continue
_check_coil_2_msb_0
        btfss   drive_byte1,4                               ; see if type 01
        goto    _check_coil_2_end                           ; no, get here if error
_check_coil_2_01                                            ; yes
        movlw   0x0f                                        ; move 15 to w in case error = 0
        movf    pos_error,f                                 ; see if error = 0
        btfsc   status,z
        goto    _check_coil_2_01_zerror                     ;
        btfss   flags,move_up_req                           ; no, see if up movement required
        goto    _check_coil_2_01_dn_rq                      ; no, down movement required
        movf    pos_error,w                                 ; yes, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM02_SP                                    ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_2_end
_check_coil_2_01_zerror
        movwf   PWM02_SP                                    ; yes, drive coil to zero
        goto    _check_coil_2_end
_check_coil_2_01_dn_rq
        movf    pos_error,w                                 ; yes, 0x0f + pos_error -> w
        addlw   0x0f
        movwf   PWM02_SP                                    ; move result to coil PWM setpoint (16-31)
        goto    _check_coil_2_end
_check_coil_2_10
        movlw   0x0f                                        ; move 15 to w in case error = 0
```

```
        movf    pos_error,f                             ; see if error = 0
        btfsc   status,z
        goto    _check_coil_2_10_zerror                 ;
        btfss   flags,move_up_req                       ; no, see if up movement required
        goto    _check_coil_2_10_dn_rq                  ; no, down movement required
        movf    pos_error,w                             ; yes, 0x0f + pos_error -> w
        addlw   0x0f
        movwf   PWM02_SP                                ; move result to coil PWM setpoint (16-31)
        goto    _check_coil_2_end
_check_coil_2_10_zerror
        movwf   PWM02_SP                                ; yes, drive coil to zero
        goto    _check_coil_2_end
_check_coil_2_10_dn_rq
        movf    pos_error,w                             ; yes, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM02_SP                                ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_2_end
_check_coil_2_end

_check_coil_3
        btfss   drive_byte1,3                           ; see if msb = 0
        goto    _check_coil_3_msb_0                     ; yes
        btfss   drive_byte1,2                           ; no, see if type 10
        goto    _check_coil_3_10                        ; yes,
_check_coil_3_type_11                                   ; get here if type 11
        movlw   0x1f
        movf    pos_error,f                             ; see if error = 0
        btfsc   status,z
        goto    _check_coil_3_type_11_zerror; yes
        movf    pos_error,w                             ; no, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM03_SP                                ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_3_end                       ; continue
_check_coil_3_type_11_zerror
        movwf   PWM03_SP                                ; yes, drive coil positive
        goto    _check_coil_3_end                       ; continue
_check_coil_3_msb_0
        btfss   drive_byte1,2                           ; see if type 01
        goto    _check_coil_3_end                       ; no, get here if error
_check_coil_3_01                                        ; yes
        movlw   0x0f                                    ; move 15 to w in case error = 0
        movf    pos_error,f                             ; see if error = 0
        btfsc   status,z
        goto    _check_coil_3_01_zerror                 ;
```

```
        btfss   flags,move_up_req                   ; no, see if up movement required
        goto    _check_coil_3_01_dn_rq              ; no, down movement required
        movf    pos_error,w                         ; yes, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM03_SP                            ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_3_end
_check_coil_3_01_zerror
        movwf   PWM03_SP                            ; yes, drive coil to zero
        goto    _check_coil_3_end
_check_coil_3_01_dn_rq
        movf    pos_error,w                         ; yes, 0x0f + pos_error -> w
        addlw   0x0f
        movwf   PWM03_SP                            ; move result to coil PWM setpoint (16-31)
        goto    _check_coil_3_end
_check_coil_3_10
        movlw   0x0f                                ; move 15 to w in case error = 0
        movf    pos_error,f                         ; see if error = 0
        btfsc   status,z
        goto    _check_coil_3_10_zerror             ;
        btfss   flags,move_up_req                   ; no, see if up movement required
        goto    _check_coil_3_10_dn_rq              ; no, down movement required
        movf    pos_error,w                         ; yes, 0x0f + pos_error -> w
        addlw   0x0f
        movwf   PWM03_SP                            ; move result to coil PWM setpoint (16-31)
        goto    _check_coil_3_end
_check_coil_3_10_zerror
        movwf   PWM03_SP                            ; yes, drive coil to zero
        goto    _check_coil_3_end
_check_coil_3_10_dn_rq
        movf    pos_error,w                         ; yes, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM03_SP                            ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_3_end
_check_coil_3_end


_check_coil_4
        btfss   drive_byte1,1                       ; see if msb = 0
        goto    _check_coil_4_msb_0                 ; yes
        btfss   drive_byte1,0                       ; no, see if type 10
        goto    _check_coil_4_10                    ; yes,
_check_coil_4_type_11                               ; get here if type 11
        movlw   0x1f
        movf    pos_error,f                         ; see if error = 0
```

```
        btfsc   status,z
        goto    _check_coil_4_type_11_zerror; yes
        movf    pos_error,w                              ; no, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM04_SP                                 ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_4_end                        ; continue
_check_coil_4_type_11_zerror
        movwf   PWM04_SP                                 ; yes, drive coil positive
        goto    _check_coil_4_end                        ; continue
_check_coil_4_msb_0
        btfss   drive_byte1,0                            ; see if type 01
        goto    _check_coil_4_end                        ; no, get here if error
_check_coil_4_01                                         ; yes
        movlw   0x0f                                     ; move 15 to w in case error = 0
        movf    pos_error,f                              ; see if error = 0
        btfsc   status,z
        goto    _check_coil_4_01_zerror                  ;
        btfss   flags,move_up_req                        ; no, see if up movement required
        goto    _check_coil_4_01_dn_rq                   ; no, down movement required
        movf    pos_error,w                              ; yes, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM04_SP                                 ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_4_end
_check_coil_4_01_zerror
        movwf   PWM04_SP                                 ; yes, drive coil to zero
        goto    _check_coil_4_end
_check_coil_4_01_dn_rq
        movf    pos_error,w                              ; yes, 0x0f + pos_error -> w
        addlw   0x0f
        movwf   PWM04_SP                                 ; move result to coil PWM setpoint (16-31)
        goto    _check_coil_4_end
_check_coil_4_10
        movlw   0x0f                                     ; move 15 to w in case error = 0
        movf    pos_error,f                              ; see if error = 0
        btfsc   status,z
        goto    _check_coil_4_10_zerror                  ;
        btfss   flags,move_up_req                        ; no, see if up movement required
        goto    _check_coil_4_10_dn_rq                   ; no, down movement required
        movf    pos_error,w                              ; yes, 0x0f + pos_error -> w
        addlw   0x0f
        movwf   PWM04_SP                                 ; move result to coil PWM setpoint (16-31)
        goto    _check_coil_4_end
_check_coil_4_10_zerror
        movwf   PWM04_SP                                 ; yes, drive coil to zero
```

```
        goto    _check_coil_4_end
_check_coil_4_10_dn_rq
        movf    pos_error,w                         ; yes, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM04_SP                            ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_4_end
_check_coil_4_end

_check_coil_5
        btfss   drive_byte2,7                       ; see if msb = 0
        goto    _check_coil_5_msb_0                 ; yes
        btfss   drive_byte2,6                       ; no, see if type 10
        goto    _check_coil_5_10                    ; yes,
_check_coil_5_type_11                               ; get here if type 11
        movlw   0x1f
        movf    pos_error,f                         ; see if error = 0
        btfsc   status,z
        goto    _check_coil_5_type_11_zerror; yes
        movf    pos_error,w                         ; no, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM05_SP                            ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_5_end                   ; continue
_check_coil_5_type_11_zerror
        movwf   PWM05_SP                            ; yes, drive coil positive
        goto    _check_coil_5_end                   ; continue
_check_coil_5_msb_0
        btfss   drive_byte2,6                       ; see if type 01
        goto    _check_coil_5_end                   ; no, get here if error
_check_coil_5_01                                    ; yes
        movlw   0x0f                                ; move 15 to w in case error = 0
        movf    pos_error,f                         ; see if error = 0
        btfsc   status,z
        goto    _check_coil_5_01_zerror             ;
        btfss   flags,move_up_req                   ; no, see if up movement required
        goto    _check_coil_5_01_dn_rq              ; no, down movement required
        movf    pos_error,w                         ; yes, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM05_SP                            ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_5_end
_check_coil_5_01_zerror
        movwf   PWM05_SP                            ; yes, drive coil to zero
        goto    _check_coil_5_end
_check_coil_5_01_dn_rq
        movf    pos_error,w                         ; yes, 0x0f + pos_error -> w
```

```
        addlw   0x0f
        movwf   PWM05_SP                                ; move result to coil PWM setpoint (16-31)
        goto    _check_coil_5_end
_check_coil_5_10
        movlw   0x0f                                    ; move 15 to w in case error = 0
        movf    pos_error,f                             ; see if error = 0
        btfsc   status,z
        goto    _check_coil_5_10_zerror                 ;
        btfss   flags,move_up_req                       ; no, see if up movement required
        goto    _check_coil_5_10_dn_rq                  ; no, down movement required
        movf    pos_error,w                             ; yes, 0x0f + pos_error -> w
        addlw   0x0f
        movwf   PWM05_SP                                ; move result to coil PWM setpoint (16-31)
        goto    _check_coil_5_end
_check_coil_5_10_zerror
        movwf   PWM05_SP                                ; yes, drive coil to zero
        goto    _check_coil_5_end
_check_coil_5_10_dn_rq
        movf    pos_error,w                             ; yes, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM05_SP                                ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_5_end
_check_coil_5_end

_check_coil_6
        btfss   drive_byte2,5                           ; see if msb = 0
        goto    _check_coil_6_msb_0                     ; yes
        btfss   drive_byte2,4                           ; no, see if type 10
        goto    _check_coil_6_10                        ; yes,
_check_coil_6_type_11                                   ; get here if type 11
        movlw   0x1f
        movf    pos_error,f                             ; see if error = 0
        btfsc   status,z
        goto    _check_coil_6_type_11_zerror; yes
        movf    pos_error,w                             ; no, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM06_SP                                ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_6_end                       ; continue
_check_coil_6_type_11_zerror
        movwf   PWM06_SP                                ; yes, drive coil positive
        goto    _check_coil_6_end                       ; continue
_check_coil_6_msb_0
        btfss   drive_byte2,4                           ; see if type 01
        goto    _check_coil_6_end                       ; no, get here if error
```

```
_check_coil_6_01                                        ; yes
       movlw   0x0f                                     ; move 15 to w in case error = 0
       movf    pos_error,f                              ; see if error = 0
       btfsc   status,z
       goto    _check_coil_6_01_zerror                  ;
       btfss   flags,move_up_req                        ; no, see if up movement required
       goto    _check_coil_6_01_dn_rq                   ; no, down movement required
       movf    pos_error,w                              ; yes, 0x0f - pos_error -> w
       sublw   0x0f
       movwf   PWM06_SP                                 ; move result to coil PWM setpoint (0-15)
       goto    _check_coil_6_end
_check_coil_6_01_zerror
       movwf   PWM06_SP                                 ; yes, drive coil to zero
       goto    _check_coil_6_end
_check_coil_6_01_dn_rq
       movf    pos_error,w                              ; yes, 0x0f + pos_error -> w
       addlw   0x0f
       movwf   PWM06_SP                                 ; move result to coil PWM setpoint (16-31)
       goto    _check_coil_6_end
_check_coil_6_10
       movlw   0x0f                                     ; move 15 to w in case error = 0
       movf    pos_error,f                              ; see if error = 0
       btfsc   status,z
       goto    _check_coil_6_10_zerror                  ;
       btfss   flags,move_up_req                        ; no, see if up movement required
       goto    _check_coil_6_10_dn_rq                   ; no, down movement required
       movf    pos_error,w                              ; yes, 0x0f + pos_error -> w
       addlw   0x0f
       movwf   PWM06_SP                                 ; move result to coil PWM setpoint (16-31)
       goto    _check_coil_6_end
_check_coil_6_10_zerror
       movwf   PWM06_SP                                 ; yes, drive coil to zero
       goto    _check_coil_6_end
_check_coil_6_10_dn_rq
       movf    pos_error,w                              ; yes, 0x0f - pos_error -> w
       sublw   0x0f
       movwf   PWM06_SP                                 ; move result to coil PWM setpoint (0-15)
       goto    _check_coil_6_end
_check_coil_6_end

_check_coil_7
       btfss   drive_byte2,3                            ; see if msb = 0
       goto    _check_coil_7_msb_0                      ; yes
       btfss   drive_byte2,2                            ; no, see if type 10
```

```
        goto    _check_coil_7_10                        ; yes,
_check_coil_7_type_11                                   ; get here if type 11
        movlw   0x1f
        movf    pos_error,f                             ; see if error = 0
        btfsc   status,z
        goto    _check_coil_7_type_11_zerror; yes
        movf    pos_error,w                             ; no, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM07_SP                                ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_7_end                       ; continue
_check_coil_7_type_11_zerror
        movwf   PWM07_SP                                ; yes, drive coil positive
        goto    _check_coil_7_end                       ; continue
_check_coil_7_msb_0
        btfss   drive_byte2,2                           ; see if type 01
        goto    _check_coil_7_end                       ; no, get here if error
_check_coil_7_01                                        ; yes
        movlw   0x0f                                    ; move 15 to w in case error = 0
        movf    pos_error,f                             ; see if error = 0
        btfsc   status,z
        goto    _check_coil_7_01_zerror                 ;
        btfss   flags,move_up_req                       ; no, see if up movement required
        goto    _check_coil_7_01_dn_rq                  ; no, down movement required
        movf    pos_error,w                             ; yes, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM07_SP                                ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_7_end
_check_coil_7_01_zerror
        movwf   PWM07_SP                                ; yes, drive coil to zero
        goto    _check_coil_7_end
_check_coil_7_01_dn_rq
        movf    pos_error,w                             ; yes, 0x0f + pos_error -> w
        addlw   0x0f
        movwf   PWM07_SP                                ; move result to coil PWM setpoint (16-31)
        goto    _check_coil_7_end
_check_coil_7_10
        movlw   0x0f                                    ; move 15 to w in case error = 0
        movf    pos_error,f                             ; see if error = 0
        btfsc   status,z
        goto    _check_coil_7_10_zerror                 ;
        btfss   flags,move_up_req                       ; no, see if up movement required
        goto    _check_coil_7_10_dn_rq                  ; no, down movement required
        movf    pos_error,w                             ; yes, 0x0f + pos_error -> w
        addlw   0x0f
```

```
        movwf   PWM07_SP                                        ; move result to coil PWM setpoint (16-31)
        goto    _check_coil_7_end
_check_coil_7_10_zerror
        movwf   PWM07_SP                                        ; yes, drive coil to zero
        goto    _check_coil_7_end
_check_coil_7_10_dn_rq
        movf    pos_error,w                                     ; yes, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM07_SP                                        ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_7_end
_check_coil_7_end


_check_coil_8
        btfss   drive_byte2,1                                   ; see if msb = 0
        goto    _check_coil_8_msb_0                             ; yes
        btfss   drive_byte2,0                                   ; no, see if type 10
        goto    _check_coil_8_10                                ; yes,
_check_coil_8_type_11                                           ; get here if type 11
        movlw   0x1f
        movf    pos_error,f                                     ; see if error = 0
        btfsc   status,z
        goto    _check_coil_8_type_11_zerror; yes
        movf    pos_error,w                                     ; no, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM08_SP                                        ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_8_end                               ; continue
_check_coil_8_type_11_zerror
        movwf   PWM08_SP                                        ; yes, drive coil positive
        goto    _check_coil_8_end                               ; continue
_check_coil_8_msb_0
        btfss   drive_byte2,0                                   ; see if type 01
        goto    _check_coil_8_end                               ; no, get here if error
_check_coil_8_01                                                ; yes
        movlw   0x0f                                            ; move 15 to w in case error = 0
        movf    pos_error,f                                     ; see if error = 0
        btfsc   status,z
        goto    _check_coil_8_01_zerror                         ;
        btfss   flags,move_up_req                               ; no, see if up movement required
        goto    _check_coil_8_01_dn_rq                          ; no, down movement required
        movf    pos_error,w                                     ; yes, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM08_SP                                        ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_8_end
```

```
_check_coil_8_01_zerror
        movwf   PWM08_SP                                ; yes, drive coil to zero
        goto    _check_coil_8_end
_check_coil_8_01_dn_rq
        movf    pos_error,w                             ; yes, 0x0f + pos_error -> w
        addlw   0x0f
        movwf   PWM08_SP                                ; move result to coil PWM setpoint (16-31)
        goto    _check_coil_8_end
_check_coil_8_10
        movlw   0x0f                                    ; move 15 to w in case error = 0
        movf    pos_error,f                             ; see if error = 0
        btfsc   status,z
        goto    _check_coil_8_10_zerror                 ;
        btfss   flags,move_up_req                       ; no, see if up movement required
        goto    _check_coil_8_10_dn_rq                  ; no, down movement required
        movf    pos_error,w                             ; yes, 0x0f + pos_error -> w
        addlw   0x0f
        movwf   PWM08_SP                                ; move result to coil PWM setpoint (16-31)
        goto    _check_coil_8_end
_check_coil_8_10_zerror
        movwf   PWM08_SP                                ; yes, drive coil to zero
        goto    _check_coil_8_end
_check_coil_8_10_dn_rq
        movf    pos_error,w                             ; yes, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM08_SP                                ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_8_end
_check_coil_8_end


_check_coil_9
        btfss   drive_byte3,7                           ; see if msb = 0
        goto    _check_coil_9_msb_0                     ; yes
        btfss   drive_byte3,6                           ; no, see if type 10
        goto    _check_coil_9_10                        ; yes,
_check_coil_9_type_11                                   ; get here if type 11
        movlw   0x1f
        movf    pos_error,f                             ; see if error = 0
        btfsc   status,z
        goto    _check_coil_9_type_11_zerror; yes
        movf    pos_error,w                             ; no, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM09_SP                                ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_9_end                       ; continue
```

```
_check_coil_9_type_11_zerror
        movwf   PWM09_SP                                    ; yes, drive coil positive
        goto    _check_coil_9_end                           ; continue
_check_coil_9_msb_0
        btfss   drive_byte3,6                               ; see if type 01
        goto    _check_coil_9_end                           ; no, get here if error
_check_coil_9_01                                            ; yes
        movlw   0x0f                                        ; move 15 to w in case error = 0
        movf    pos_error,f                                 ; see if error = 0
        btfsc   status,z
        goto    _check_coil_9_01_zerror                     ;
        btfss   flags,move_up_req                           ; no, see if up movement required
        goto    _check_coil_9_01_dn_rq                      ; no, down movement required
        movf    pos_error,w                                 ; yes, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM09_SP                                    ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_9_end
_check_coil_9_01_zerror
        movwf   PWM09_SP                                    ; yes, drive coil to zero
        goto    _check_coil_9_end
_check_coil_9_01_dn_rq
        movf    pos_error,w                                 ; yes, 0x0f + pos_error -> w
        addlw   0x0f
        movwf   PWM09_SP                                    ; move result to coil PWM setpoint (16-31)
        goto    _check_coil_9_end
_check_coil_9_10
        movlw   0x0f                                        ; move 15 to w in case error = 0
        movf    pos_error,f                                 ; see if error = 0
        btfsc   status,z
        goto    _check_coil_9_10_zerror                     ;
        btfss   flags,move_up_req                           ; no, see if up movement required
        goto    _check_coil_9_10_dn_rq                      ; no, down movement required
        movf    pos_error,w                                 ; yes, 0x0f + pos_error -> w
        addlw   0x0f
        movwf   PWM09_SP                                    ; move result to coil PWM setpoint (16-31)
        goto    _check_coil_9_end
_check_coil_9_10_zerror
        movwf   PWM09_SP                                    ; yes, drive coil to zero
        goto    _check_coil_9_end
_check_coil_9_10_dn_rq
        movf    pos_error,w                                 ; yes, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM09_SP                                    ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_9_end
```

```
_check_coil_9_end

_check_coil_10
        btfss   drive_byte3,5                           ; see if msb = 0
        goto    _check_coil_10_msb_0                    ; yes
        btfss   drive_byte3,4                           ; no, see if type 10
        goto    _check_coil_10_10                       ; yes,
_check_coil_10_type_11                                  ; get here if type 11
        movlw   0x1f
        movf    pos_error,f                             ; see if error = 0
        btfsc   status,z
        goto    _check_coil_10_type_11_zerror; yes
        movf    pos_error,w                             ; no, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM10_SP                                ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_10_end                      ; continue
_check_coil_10_type_11_zerror
        movwf   PWM10_SP                                ; yes, drive coil positive
        goto    _check_coil_10_end                      ; continue
_check_coil_10_msb_0
        btfss   drive_byte3,4                           ; see if type 01
        goto    _check_coil_10_end                      ; no, get here if error
_check_coil_10_01                                       ; yes
        movlw   0x0f                                    ; move 15 to w in case error = 0
        movf    pos_error,f                             ; see if error = 0
        btfsc   status,z
        goto    _check_coil_10_01_zerror                ;
        btfss   flags,move_up_req                       ; no, see if up movement required
        goto    _check_coil_10_01_dn_rq                 ; no, down movement required
        movf    pos_error,w                             ; yes, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM10_SP                                ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_10_end
_check_coil_10_01_zerror
        movwf   PWM10_SP                                ; yes, drive coil to zero
        goto    _check_coil_10_end
_check_coil_10_01_dn_rq
        movf    pos_error,w                             ; yes, 0x0f + pos_error -> w
        addlw   0x0f
        movwf   PWM10_SP                                ; move result to coil PWM setpoint (16-31)
        goto    _check_coil_10_end
_check_coil_10_10
        movlw   0x0f                                    ; move 15 to w in case error = 0
        movf    pos_error,f                             ; see if error = 0
```

```
        btfsc   status,z
        goto    _check_coil_10_10_zerror                ;
        btfss   flags,move_up_req                       ; no, see if up movement required
        goto    _check_coil_10_10_dn_rq                 ; no, down movement required
        movf    pos_error,w                             ; yes, 0x0f + pos_error -> w
        addlw   0x0f
        movwf   PWM10_SP                                ; move result to coil PWM setpoint (16-31)
        goto    _check_coil_10_end
_check_coil_10_10_zerror
        movwf   PWM10_SP                                ; yes, drive coil to zero
        goto    _check_coil_10_end
_check_coil_10_10_dn_rq
        movf    pos_error,w                             ; yes, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM10_SP                                ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_10_end
_check_coil_10_end

_check_coil_11
        btfss   drive_byte3,3                           ; see if msb = 0
        goto    _check_coil_11_msb_0                    ; yes
        btfss   drive_byte3,2                           ; no, see if type 10
        goto    _check_coil_11_10                       ; yes,
_check_coil_11_type_11                                  ; get here if type 11
        movlw   0x1f
        movf    pos_error,f                             ; see if error = 0
        btfsc   status,z
        goto    _check_coil_11_type_11_zerror; yes
        movf    pos_error,w                             ; no, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM11_SP                                ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_11_end                      ; continue
_check_coil_11_type_11_zerror
        movwf   PWM11_SP                                ; yes, drive coil positive
        goto    _check_coil_11_end                      ; continue
_check_coil_11_msb_0
        btfss   drive_byte3,2                           ; see if type 01
        goto    _check_coil_11_end                      ; no, get here if error
_check_coil_11_01                                       ; yes
        movlw   0x0f                                    ; move 15 to w in case error = 0
        movf    pos_error,f                             ; see if error = 0
        btfsc   status,z
        goto    _check_coil_11_01_zerror                ;
        btfss   flags,move_up_req                       ; no, see if up movement required
```

```
        goto    _check_coil_11_01_dn_rq                 ; no, down movement required
        movf    pos_error,w                             ; yes, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM11_SP                                ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_11_end
_check_coil_11_01_zerror
        movwf   PWM11_SP                                ; yes, drive coil to zero
        goto    _check_coil_11_end
_check_coil_11_01_dn_rq
        movf    pos_error,w                             ; yes, 0x0f + pos_error -> w
        addlw   0x0f
        movwf   PWM11_SP                                ; move result to coil PWM setpoint (16-31)
        goto    _check_coil_11_end
_check_coil_11_10
        movlw   0x0f                                    ; move 15 to w in case error = 0
        movf    pos_error,f                             ; see if error = 0
        btfsc   status,z
        goto    _check_coil_11_10_zerror                ;
        btfss   flags,move_up_req                       ; no, see if up movement required
        goto    _check_coil_11_10_dn_rq                 ; no, down movement required
        movf    pos_error,w                             ; yes, 0x0f + pos_error -> w
        addlw   0x0f
        movwf   PWM11_SP                                ; move result to coil PWM setpoint (16-31)
        goto    _check_coil_11_end
_check_coil_11_10_zerror
        movwf   PWM11_SP                                ; yes, drive coil to zero
        goto    _check_coil_11_end
_check_coil_11_10_dn_rq
        movf    pos_error,w                             ; yes, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM11_SP                                ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_11_end
_check_coil_11_end


_check_coil_12
        btfss   drive_byte3,1                           ; see if msb = 0
        goto    _check_coil_12_msb_0                    ; yes
        btfss   drive_byte3,0                           ; no, see if type 10
        goto    _check_coil_12_10                       ; yes,
_check_coil_12_type_11                                  ; get here if type 11
        movlw   0x1f
        movf    pos_error,f                             ; see if error = 0
        btfsc   status,z
```

```
        goto    _check_coil_12_type_11_zerror; yes
        movf    pos_error,w                            ; no, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM12_SP                               ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_12_end                     ; continue
_check_coil_12_type_11_zerror
        movwf   PWM12_SP                               ; yes, drive coil positive
        goto    _check_coil_12_end                     ; continue
_check_coil_12_msb_0
        btfss   drive_byte3,0                          ; see if type 01
        goto    _check_coil_12_end                     ; no, get here if error
_check_coil_12_01                                      ; yes
        movlw   0x0f                                   ; move 15 to w in case error = 0
        movf    pos_error,f                            ; see if error = 0
        btfsc   status,z
        goto    _check_coil_12_01_zerror               ;
        btfss   flags,move_up_req                      ; no, see if up movement required
        goto    _check_coil_12_01_dn_rq                ; no, down movement required
        movf    pos_error,w                            ; yes, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM12_SP                               ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_12_end
_check_coil_12_01_zerror
        movwf   PWM12_SP                               ; yes, drive coil to zero
        goto    _check_coil_12_end
_check_coil_12_01_dn_rq
        movf    pos_error,w                            ; yes, 0x0f + pos_error -> w
        addlw   0x0f
        movwf   PWM12_SP                               ; move result to coil PWM setpoint (16-31)
        goto    _check_coil_12_end
_check_coil_12_10
        movlw   0x0f                                   ; move 15 to w in case error = 0
        movf    pos_error,f                            ; see if error = 0
        btfsc   status,z
        goto    _check_coil_12_10_zerror               ;
        btfss   flags,move_up_req                      ; no, see if up movement required
        goto    _check_coil_12_10_dn_rq                ; no, down movement required
        movf    pos_error,w                            ; yes, 0x0f + pos_error -> w
        addlw   0x0f
        movwf   PWM12_SP                               ; move result to coil PWM setpoint (16-31)
        goto    _check_coil_12_end
_check_coil_12_10_zerror
        movwf   PWM12_SP                               ; yes, drive coil to zero
        goto    _check_coil_12_end
```

```
_check_coil_12_10_dn_rq
        movf    pos_error,w                         ; yes, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM12_SP                            ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_12_end
_check_coil_12_end

_check_coil_13
        btfss   drive_byte4,7                       ; see if msb = 0
        goto    _check_coil_13_msb_0                ; yes
        btfss   drive_byte4,6                       ; no, see if type 10
        goto    _check_coil_13_10                   ; yes,
_check_coil_13_type_11                              ; get here if type 11
        movlw   0x1f
        movf    pos_error,f                         ; see if error = 0
        btfsc   status,z
        goto    _check_coil_13_type_11_zerror; yes
        movf    pos_error,w                         ; no, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM13_SP                            ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_13_end                  ; continue
_check_coil_13_type_11_zerror
        movwf   PWM13_SP                            ; yes, drive coil positive
        goto    _check_coil_13_end                  ; continue
_check_coil_13_msb_0
        btfss   drive_byte4,6                       ; see if type 01
        goto    _check_coil_13_end                  ; no, get here if error
_check_coil_13_01                                   ; yes
        movlw   0x0f                                ; move 15 to w in case error = 0
        movf    pos_error,f                         ; see if error = 0
        btfsc   status,z
        goto    _check_coil_13_01_zerror ;
        btfss   flags,move_up_req                   ; no, see if up movement required
        goto    _check_coil_13_01_dn_rq             ; no, down movement required
        movf    pos_error,w                         ; yes, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM13_SP                            ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_13_end
_check_coil_13_01_zerror
        movwf   PWM13_SP                            ; yes, drive coil to zero
        goto    _check_coil_13_end
_check_coil_13_01_dn_rq
        movf    pos_error,w                         ; yes, 0x0f + pos_error -> w
        addlw   0x0f
```

```
        movwf   PWM13_SP                                        ; move result to coil PWM setpoint (16-31)
        goto    _check_coil_13_end
_check_coil_13_10
        movlw   0x0f                                            ; move 15 to w in case error = 0
        movf    pos_error,f                                     ; see if error = 0
        btfsc   status,z
        goto    _check_coil_13_10_zerror                        ;
        btfss   flags,move_up_req                               ; no, see if up movement required
        goto    _check_coil_13_10_dn_rq                         ; no, down movement required
        movf    pos_error,w                                     ; yes, 0x0f + pos_error -> w
        addlw   0x0f
        movwf   PWM13_SP                                        ; move result to coil PWM setpoint (16-31)
        goto    _check_coil_13_end
_check_coil_13_10_zerror
        movwf   PWM13_SP                                        ; yes, drive coil to zero
        goto    _check_coil_13_end
_check_coil_13_10_dn_rq
        movf    pos_error,w                                     ; yes, 0x0f - pos_error -> w
        sublw   0x0f
        movwf   PWM13_SP                                        ; move result to coil PWM setpoint (0-15)
        goto    _check_coil_13_end
_check_coil_13_end


end_program
        goto    main


;;;; MAIN PROGRAMME
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;; TABLE DATA

;; Table data for the angle-to-position (sine) conversion is located between 0x0400 and 0x07ff
;; pclath should be cleared as we are in page 0
;; Inputs: AD conversion results in ADRESL and ADRESH
;; Outputs: position in mm in w (0=fully up)
;;
;; The table has been built using the following steps:
;; 1. Find fully up voltage (-89mm). Vup.
;; 2. Find fully down voltage (+89mm). Vdn.
;; 3. Find centre voltage. Vc.
;; 4. Configure wiring so that Vup<Vdn
;; 5. Range = (Vdn-Vup)/90 volts per degree, or [(Vdn-Vup)/90]*(1024/5) counts per degree
;; 6. Zero = (Vup/5)*1024 counts
```

```
;; 7. Angle = [(AD count – Zero count) / counts per degree]-45
;; 8. Position x=125.86*sin(angle)+89
;; 9. Build Table. I.e:
;;    AD Count : Position (mm)
;;    AD count : 125.86*sin([(AD count – Zero count) / counts per degree]-45)+89
;;
;; Position is 0mm fully up and 178mm fully down.




        ORG     0x0400
pos_table1
        addwf   pcl,f
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
```

```
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
```

```
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
```

```
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
```

```
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
```

```
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
```

```
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
;       retlw 0xBC




        ORG     0x0500                  ; Address
pos_table2
        addwf   pcl,f
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
```

```
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
retlw 0xBC
```

```
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBC
        retlw 0xBB
        retlw 0xBB
        retlw 0xBB
        retlw 0xBB
        retlw 0xBB
        retlw 0xBB
        retlw 0xBB
        retlw 0xBB
        retlw 0xBB
        retlw 0xBB
        retlw 0xBB
```

```
        retlw 0xBB
        retlw 0xBB
        retlw 0xBA
        retlw 0xBA
        retlw 0xBA
        retlw 0xBA
        retlw 0xBA
        retlw 0xBA
        retlw 0xBA
        retlw 0xBA
        retlw 0xB9
        retlw 0xB9
        retlw 0xB9
        retlw 0xB9
        retlw 0xB9
        retlw 0xB9
        retlw 0xB9
        retlw 0xB8
        retlw 0xB8
        retlw 0xB8
        retlw 0xB8
        retlw 0xB8
        retlw 0xB8
        retlw 0xB7
        retlw 0xB7
        retlw 0xB7
        retlw 0xB7
        retlw 0xB7
        retlw 0xB6
        retlw 0xB6
        retlw 0xB6
        retlw 0xB6
        retlw 0xB6
        retlw 0xB5
        retlw 0xB5
        retlw 0xB5
        retlw 0xB5
        retlw 0xB4
        retlw 0xB4
        retlw 0xB4
        retlw 0xB4
        retlw 0xB3
        retlw 0xB3
        retlw 0xB3
```

```
        retlw 0xB3
        retlw 0xB2
        retlw 0xB2
        retlw 0xB2
        retlw 0xB2
        retlw 0xB1
        retlw 0xB1
        retlw 0xB1
        retlw 0xB0
        retlw 0xB0
        retlw 0xB0
        retlw 0xB0
        retlw 0xAF
        retlw 0xAF
        retlw 0xAF
        retlw 0xAE
        retlw 0xAE
        retlw 0xAE
        retlw 0xAD
        retlw 0xAD
        retlw 0xAD
        retlw 0xAC
        retlw 0xAC
        retlw 0xAC
        retlw 0xAB
        retlw 0xAB
        retlw 0xAB
        retlw 0xAA
        retlw 0xAA
        retlw 0xAA
        retlw 0xA9
        retlw 0xA9
        retlw 0xA9
        retlw 0xA8
        retlw 0xA8
        retlw 0xA8
        retlw 0xA7
        retlw 0xA7
        retlw 0xA6
        retlw 0xA6
        retlw 0xA6
        retlw 0xA5
        retlw 0xA5
        retlw 0xA4
```

```
        retlw 0xA4
        retlw 0xA4
        retlw 0xA3
        retlw 0xA3
        retlw 0xA2
        retlw 0xA2
        retlw 0xA2
        retlw 0xA1
        retlw 0xA1
        retlw 0xA0
        retlw 0xA0
        retlw 0x9F
        retlw 0x9F
        retlw 0x9F
        retlw 0x9E
        retlw 0x9E
        retlw 0x9D
        retlw 0x9D
        retlw 0x9C
        retlw 0x9C
        retlw 0x9B
        retlw 0x9B
        retlw 0x9A
        retlw 0x9A
        retlw 0x9A
        retlw 0x99
        retlw 0x99
        retlw 0x98
        retlw 0x98
        retlw 0x97
        retlw 0x97
        retlw 0x96
        retlw 0x96
        retlw 0x95
        retlw 0x95
        retlw 0x94
        retlw 0x94
        retlw 0x93
        retlw 0x93
        retlw 0x92
        retlw 0x92
        retlw 0x91
        retlw 0x91
        retlw 0x90
```

```
        retlw 0x90
        retlw 0x8F
        retlw 0x8F
        retlw 0x8E
        retlw 0x8E
;       retlw 0x8D




        ORG     0x0600
pos_table3
        addwf   pcl,f
        retlw 0x8C
        retlw 0x8C
        retlw 0x8B
        retlw 0x8B
        retlw 0x8A
        retlw 0x8A
        retlw 0x89
        retlw 0x89
        retlw 0x88
        retlw 0x88
        retlw 0x87
        retlw 0x86
        retlw 0x86
        retlw 0x85
        retlw 0x85
        retlw 0x84
        retlw 0x84
        retlw 0x83
        retlw 0x83
        retlw 0x82
        retlw 0x81
        retlw 0x81
        retlw 0x80
        retlw 0x80
        retlw 0x7F
        retlw 0x7F
        retlw 0x7E
        retlw 0x7D
        retlw 0x7D
        retlw 0x7C
```

```
        retlw 0x7C
        retlw 0x7B
        retlw 0x7A
        retlw 0x7A
        retlw 0x79
        retlw 0x79
        retlw 0x78
        retlw 0x77
        retlw 0x77
        retlw 0x76
        retlw 0x76
        retlw 0x75
        retlw 0x74
        retlw 0x74
        retlw 0x73
        retlw 0x73
        retlw 0x72
        retlw 0x71
        retlw 0x71
        retlw 0x70
        retlw 0x70
        retlw 0x6F
        retlw 0x6E
        retlw 0x6E
        retlw 0x6D
        retlw 0x6C
        retlw 0x6C
        retlw 0x6B
        retlw 0x6B
        retlw 0x6A
        retlw 0x69
        retlw 0x69
        retlw 0x68
        retlw 0x67
        retlw 0x67
        retlw 0x66
        retlw 0x66
        retlw 0x65
        retlw 0x64
        retlw 0x64
        retlw 0x63
        retlw 0x62
        retlw 0x62
        retlw 0x61
```

```
        retlw 0x61
        retlw 0x60
        retlw 0x5F
        retlw 0x5F
        retlw 0x5E
        retlw 0x5D
        retlw 0x5D
        retlw 0x5C
        retlw 0x5B
        retlw 0x5B
        retlw 0x5A
        retlw 0x59
        retlw 0x59
        retlw 0x58
        retlw 0x58
        retlw 0x57
        retlw 0x56
        retlw 0x56
        retlw 0x55
        retlw 0x54
        retlw 0x54
        retlw 0x53
        retlw 0x52
        retlw 0x52
        retlw 0x51
        retlw 0x50
        retlw 0x50
        retlw 0x4F
        retlw 0x4E
        retlw 0x4E
        retlw 0x4D
        retlw 0x4D
        retlw 0x4C
        retlw 0x4C
        retlw 0x4B
        retlw 0x4B
        retlw 0x4A
        retlw 0x49
        retlw 0x49
        retlw 0x48
        retlw 0x48
        retlw 0x47
        retlw 0x47
        retlw 0x46
```

```
        retlw 0x45
        retlw 0x45
        retlw 0x44
        retlw 0x44
        retlw 0x43
        retlw 0x43
        retlw 0x42
        retlw 0x41
        retlw 0x41
        retlw 0x40
        retlw 0x40
        retlw 0x3F
        retlw 0x3F
        retlw 0x3E
        retlw 0x3D
        retlw 0x3D
        retlw 0x3C
        retlw 0x3C
        retlw 0x3B
        retlw 0x3B
        retlw 0x3A
        retlw 0x3A
        retlw 0x39
        retlw 0x38
        retlw 0x38
        retlw 0x37
        retlw 0x37
        retlw 0x36
        retlw 0x36
        retlw 0x35
        retlw 0x35
        retlw 0x34
        retlw 0x33
        retlw 0x33
        retlw 0x32
        retlw 0x32
        retlw 0x31
        retlw 0x31
        retlw 0x30
        retlw 0x30
        retlw 0x2F
        retlw 0x2F
        retlw 0x2E
        retlw 0x2D
```

```
        retlw 0x2D
        retlw 0x2C
        retlw 0x2C
        retlw 0x2B
        retlw 0x2B
        retlw 0x2A
        retlw 0x2A
        retlw 0x29
        retlw 0x29
        retlw 0x28
        retlw 0x28
        retlw 0x27
        retlw 0x27
        retlw 0x26
        retlw 0x26
        retlw 0x25
        retlw 0x24
        retlw 0x24
        retlw 0x23
        retlw 0x23
        retlw 0x22
        retlw 0x22
        retlw 0x21
        retlw 0x21
        retlw 0x20
        retlw 0x20
        retlw 0x1F
        retlw 0x1F
        retlw 0x1E
        retlw 0x1E
        retlw 0x1D
        retlw 0x1D
        retlw 0x1C
        retlw 0x1C
        retlw 0x1B
        retlw 0x1B
        retlw 0x1A
        retlw 0x1A
        retlw 0x19
        retlw 0x19
        retlw 0x18
        retlw 0x18
        retlw 0x18
        retlw 0x17
```

```
        retlw 0x17
        retlw 0x16
        retlw 0x16
        retlw 0x15
        retlw 0x15
        retlw 0x14
        retlw 0x14
        retlw 0x13
        retlw 0x13
        retlw 0x12
        retlw 0x12
        retlw 0x11
        retlw 0x11
        retlw 0x11
        retlw 0x10
        retlw 0x10
        retlw 0x0F
        retlw 0x0F
        retlw 0x0E
        retlw 0x0E
        retlw 0x0D
        retlw 0x0D
        retlw 0x0D
        retlw 0x0C
        retlw 0x0C
        retlw 0x0B
        retlw 0x0B
        retlw 0x0A
        retlw 0x0A
        retlw 0x0A
        retlw 0x09
        retlw 0x09
        retlw 0x08
        retlw 0x08
        retlw 0x08
        retlw 0x07
        retlw 0x07
        retlw 0x06
        retlw 0x06
        retlw 0x06
        retlw 0x05
        retlw 0x05
        retlw 0x04
        retlw 0x04
```

```
        retlw 0x04
        retlw 0x03
        retlw 0x03
        retlw 0x02
        retlw 0x02
;       retlw 0x02




        ORG     0x0700
pos_table4
        addwf   pcl,f
        retlw 0x01
        retlw 0x01
        retlw 0x01
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
```

```
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
retlw 0x00
```

```
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
```

```
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
```

```
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
	retlw 0x00
```

```
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00
        retlw 0x00




;; Table data for position vs driving force. This data drives the coils depending on the position.
;; Data consists of 4 bytes for every position. These
;; bytes hold driving data for each of the 13 coil arrangements. The data for each coil is 2 bits
;; long which represents one of four possible driving scenarios.
;; 00 = zero - no driving force
;; 01 = to raise position, set PWM on that coil to 0-15, to lower position, set PWM on that coil to 16-31, to hold position set
;; PWM on that coil to zero
;; 10 = to raise position, set PWM on that coil to 15-31, to lower position, set PWM on that coil to 0-15, to hold position set
;; PWM on that coil to zero
;; 11 = to raise position, set PWM on that coil to 0-15, to lower position, set PWM on that coil to 0-15, to hold position set
;; PWM on that coil to 15-31
```

```
;;
;; For position XXX mm
;;
;; 1st Byte
;; Bit 76     54     32     10
;;     coil01 coil02 coil03 coil04
;;
;; 2nd Byte
;; Bit 76     54     32     10
;;     coil05 coil06 coil07 coil08
;;
;; 3rd Byte
;; Bit 76     54     32     10
;;     coil09 coil10 coil11 coil12
;;
;; 4th Byte
;; Bit 76     54     32     10
;;     coil13 N/A     N/A     N/A
;;


       ORG     0x0800                ; Address 2048
drive_table
       movwf   pcl ;,f
                                              ; Position 0mm
       retlw B'10101010'
       retlw B'10101010'
       retlw B'10101010'
       retlw B'10000000'
       retlw B'10101010'
       retlw B'10101010'
       retlw B'10101010'
       retlw B'10000000'
       retlw B'10101010'
       retlw B'10101010'
       retlw B'10101010'
       retlw B'10000000'
       retlw B'10101010'
       retlw B'10101010'
       retlw B'10101010'
       retlw B'10000000'
       retlw B'10101010'
       retlw B'10101010'
       retlw B'10101010'
```

```
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'11101010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'01101010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'01101010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'01101010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'01101010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'01101010'
        retlw B'10101010'
        retlw B'10101010'
```

```
        retlw B'10000000'
        retlw B'01101010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'01111010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'01011010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'11011010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10011010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10011010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10011010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10011010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10011110'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10010110'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10110110'
        retlw B'10101010'
        retlw B'10101010'
```

```
        retlw B'10000000'
        retlw B'11100110'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'01100110'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'01100110'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'01100110'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'01100111'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'01100101'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'01101101'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'01111001'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'11011001'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10011001'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10011001'
        retlw B'10101010'
        retlw B'10101010'
```

```
        retlw B'10000000'
        retlw B'10011001'
        retlw B'11101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10011001'
        retlw B'01101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10011011'
        retlw B'01101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10011110'
        retlw B'01101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10110110'
        retlw B'01101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'11100110'
        retlw B'01101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'01100110'
        retlw B'01101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'01100110'
        retlw B'01111010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'01100110'
        retlw B'01011010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'01100110'
        retlw B'11011010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'01100111'
        retlw B'10011010'
        retlw B'10101010'
```

```
        retlw B'10000000'
        retlw B'01101101'
        retlw B'10011010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'01111001'
        retlw B'10011010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'11011001'
        retlw B'10011010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10011001'
        retlw B'10011110'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10011001'
        retlw B'10010110'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10011001'
        retlw B'10110110'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10011001'
        retlw B'11100110'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10011011'
        retlw B'01100110'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10011110'
        retlw B'01100110'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10110110'
        retlw B'01100110'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'11100110'
        retlw B'01100111'
        retlw B'10101010'
```

```
        retlw B'10000000'
        retlw B'01100110'
        retlw B'01100101'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'01100110'
        retlw B'01101101'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'01100110'
        retlw B'01111001'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'01100110'
        retlw B'11011001'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'01100111'
        retlw B'10011001'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'01101101'
        retlw B'10011001'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'01111001'
        retlw B'10011001'
        retlw B'11101010'
        retlw B'10000000'
        retlw B'11011001'
        retlw B'10011001'
        retlw B'01101010'
        retlw B'10000000'
        retlw B'10011001'
        retlw B'10011011'
        retlw B'01101010'
        retlw B'10000000'
        retlw B'10011001'
        retlw B'10011110'
        retlw B'01101010'
        retlw B'10000000'
        retlw B'10011001'
        retlw B'10110110'
        retlw B'01101010'
```

```
        retlw B'10000000'
        retlw B'10011001'
        retlw B'11100110'
        retlw B'01101010'
        retlw B'10000000'
        retlw B'10011011'
        retlw B'01100110'
        retlw B'01101010'
        retlw B'10000000'
        retlw B'10011110'
        retlw B'01100110'
        retlw B'01111010'
        retlw B'10000000'
        retlw B'10110110'
        retlw B'01100110'
        retlw B'01011010'
        retlw B'10000000'
        retlw B'11100110'
        retlw B'01100110'
        retlw B'11011010'
        retlw B'10000000'
        retlw B'01100110'
        retlw B'01100111'
        retlw B'10011010'
        retlw B'10000000'
        retlw B'01100110'
        retlw B'01101101'
        retlw B'10011010'
        retlw B'10000000'
        retlw B'01100110'
        retlw B'01111001'
        retlw B'10011010'
        retlw B'10000000'
        retlw B'01100110'
        retlw B'11011001'
        retlw B'10011010'
        retlw B'10000000'
        retlw B'01100111'
        retlw B'10011001'
        retlw B'10011110'
        retlw B'10000000'
        retlw B'01101101'
        retlw B'10011001'
        retlw B'10010110'
```

```
        retlw B'10000000'
        retlw B'01111001'
        retlw B'10011001'
        retlw B'10110110'
        retlw B'10000000'
        retlw B'01011001'
        retlw B'10011001'
        retlw B'11100110'
        retlw B'10000000'
        retlw B'11011001'
        retlw B'10011011'
        retlw B'01100110'
        retlw B'10000000'
        retlw B'10011001'
        retlw B'10011110'
        retlw B'01100110'
        retlw B'10000000'
        retlw B'10011001'
        retlw B'10110110'
        retlw B'01100110'
        retlw B'10000000'
        retlw B'10011001'
        retlw B'11100110'
        retlw B'01100111'
        retlw B'10000000'
        retlw B'10011011'
        retlw B'01100110'
        retlw B'01100101'
        retlw B'10000000'
        retlw B'10011110'
        retlw B'01100110'
        retlw B'01101101'
        retlw B'10000000'
        retlw B'10010110'
        retlw B'01100110'
        retlw B'01111001'
        retlw B'10000000'
        retlw B'10110110'
        retlw B'01100110'
        retlw B'11011001'
        retlw B'10000000'
        retlw B'10100110'
        retlw B'01100111'
        retlw B'10011001'
```

```
        retlw B'10000000'
        retlw B'10100110'
        retlw B'01101101'
        retlw B'10011001'
        retlw B'10000000'
        retlw B'10100110'
        retlw B'01111001'
        retlw B'10011001'
        retlw B'11000000'
        retlw B'10100110'
        retlw B'11011001'
        retlw B'10011001'
        retlw B'01000000'
        retlw B'10100111'
        retlw B'10011001'
        retlw B'10011011'
        retlw B'01000000'
        retlw B'10100101'
        retlw B'10011001'
        retlw B'10011110'
        retlw B'01000000'
        retlw B'10101101'
        retlw B'10011001'
        retlw B'10110110'
        retlw B'01000000'
        retlw B'10101001'
        retlw B'10011001'
        retlw B'11100110'
        retlw B'01000000'
        retlw B'10101001'
        retlw B'10011011'
        retlw B'01100110'
        retlw B'01000000'
        retlw B'10101001'
        retlw B'10011110'
        retlw B'01100110'
        retlw B'01000000'
        retlw B'10101001'
        retlw B'10110110'
        retlw B'01100110'
        retlw B'01000000'
        retlw B'10101001'
        retlw B'11100110'
        retlw B'01100110'
```

```
        retlw B'11000000'
        retlw B'10101001'
        retlw B'01100110'
        retlw B'01100111'
        retlw B'10000000'
        retlw B'10101011'
        retlw B'01100110'
        retlw B'01101101'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'01100110'
        retlw B'01111001'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'01100110'
        retlw B'11011001'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'01100111'
        retlw B'10011001'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'01101101'
        retlw B'10011001'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'01111001'
        retlw B'10011001'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'01011001'
        retlw B'10011001'
        retlw B'11000000'
        retlw B'10101010'
        retlw B'11011001'
        retlw B'10011011'
        retlw B'01000000'
        retlw B'10101010'
        retlw B'10011001'
        retlw B'10011110'
        retlw B'01000000'
        retlw B'10101010'
        retlw B'10011001'
        retlw B'10110110'
```

```
        retlw B'01000000'
        retlw B'10101010'
        retlw B'10011001'
        retlw B'11100110'
        retlw B'01000000'
        retlw B'10101010'
        retlw B'10011011'
        retlw B'01100110'
        retlw B'01000000'
        retlw B'10101010'
        retlw B'10011110'
        retlw B'01100110'
        retlw B'01000000'
        retlw B'10101010'
        retlw B'10010110'
        retlw B'01100110'
        retlw B'01000000'
        retlw B'10101010'
        retlw B'10110110'
        retlw B'01100110'
        retlw B'11000000'
        retlw B'10101010'
        retlw B'10100110'
        retlw B'01100111'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10100110'
        retlw B'01101101'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10100110'
        retlw B'01111001'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10100110'
        retlw B'11011001'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10100111'
        retlw B'10011001'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10100101'
        retlw B'10011001'
```

```
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10101101'
        retlw B'10011001'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10101001'
        retlw B'10011001'
        retlw B'11000000'
        retlw B'10101010'
        retlw B'10101001'
        retlw B'10011011'
        retlw B'01000000'
        retlw B'10101010'
        retlw B'10101001'
        retlw B'10011110'
        retlw B'01000000'
        retlw B'10101010'
        retlw B'10101001'
        retlw B'10110110'
        retlw B'01000000'
        retlw B'10101010'
        retlw B'10101001'
        retlw B'11100110'
        retlw B'01000000'
        retlw B'10101010'
        retlw B'10101001'
        retlw B'01100110'
        retlw B'01000000'
        retlw B'10101010'
        retlw B'10101011'
        retlw B'01100110'
        retlw B'01000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'01100110'
        retlw B'01000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'01100110'
        retlw B'11000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'01100111'
```

```
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'01101101'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'01111001'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'01011001'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'11011001'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10011001'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10011001'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10011001'
        retlw B'11000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10011011'
        retlw B'01000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10011110'
        retlw B'01000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10010110'
        retlw B'01000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10110110'
```

```
        retlw B'01000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10100110'
        retlw B'01000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10100110'
        retlw B'01000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10100110'
        retlw B'01000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10100110'
        retlw B'11000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10100111'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10100101'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10101101'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10101001'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10101001'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10101001'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10101001'
```

```
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10101001'
        retlw B'11000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10101001'
        retlw B'01000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10101011'
        retlw B'01000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'01000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'01000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'01000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'01000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'01000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'01000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'11000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10101010'
```

```
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10101010'
        retlw B'10000000'

        end                             ; directive 'end of program'
```

Appendix E Terminal Programme Code Listing

```
Dim flg4 As Integer
Dim flg5 As Integer
Option Explicit
Dim ax As Integer
Dim ay As Integer
Dim bx As Integer
Dim kx As Integer
Dim ky As Integer
Dim ly As Integer
'trend
Dim MV As Integer
Dim MVold As Integer
Dim tr1 As Integer
Dim tr2 As Integer
Dim mvvar As Integer
Dim PV As Integer
Dim PVold As Integer
Dim pvvar As Integer
Dim SP As Integer
Dim SPold As Integer
Dim spvar As Integer
''pid''
Dim Gvar As Single
Dim intvar As Single
Dim dervar As Single
Dim scanvar As Integer
Dim de As Integer
Dim e As Integer
Dim scans As Integer
Dim dmv As Integer
Dim laste As Integer
Dim intcomp As Double
Dim gaindelay As Integer
Dim pgvar As Single
Dim plvar As Integer
Dim dpv As Integer
Dim lastpv As Integer
Dim Position As Integer
Dim outchar As String
Dim spvariable As Integer
Private Sub Command1_Click()
If MSComm1.PortOpen = False Then
MsgBox ("Error, Port not opened")
Exit Sub
Else
outchar = Chr$(Text1.Text)
spvariable = Asc(outchar)
MSComm1.Output = outchar
End If
End Sub
Private Sub Command10_Click()
Label7.Caption = ""
End Sub
Private Sub Command11_Click()
If MSComm1.PortOpen = False Then
MsgBox ("Error, Port not opened")
Exit Sub
Else
Text1.Text = Text1.Text + 2
outchar = Chr$(Text1.Text)
spvariable = Asc(outchar)
MSComm1.Output = outchar
End If
End Sub
```

```
Private Sub Command12_Click()
If MSComm1.PortOpen = False Then
MsgBox ("Error, Port not opened")
Exit Sub
Else
SimpTerm - 2
Text1.Text = Text1.Text - 2
outchar = Chr$(Text1.Text)
spvariable = Asc(outchar)
MSComm1.Output = outchar
End If
End Sub
Private Sub Command13_Click()
If MSComm1.PortOpen = False Then
MsgBox ("Error, Port not opened")
Exit Sub
Else
outchar = Chr$("&h" + "f7")
MSComm1.Output = outchar
End If
'End If
End Sub
Private Sub Command14_Click()
If MSComm1.PortOpen = False Then
MsgBox ("Error, Port not opened")
Exit Sub
Else
outchar = Chr$("&h" + "f8")
MSComm1.Output = outchar
End If
End Sub
Private Sub Command15_Click()
Text3.Text = ""
End Sub
Private Sub Command16_Click()
Picture1.Cls
End Sub
Private Sub Command17_Click()
Timer4.Enabled = False
Timer2.Enabled = False
Timer3.Enabled = False
Command13.Enabled = True
Command14.Enabled = True
End Sub
Private Sub Command18_Click()
Timer4.Enabled = True
Timer2.Enabled = True
Timer3.Enabled = True
Command13.Enabled = False
Command14.Enabled = False
End Sub
Private Sub Command2_Click()
Text1.Text = ""
End Sub
Private Sub Command3_Click()
kx = 1000
ay = 35
Picture1.Cls
Timer4.Enabled = True
Timer2.Enabled = True
Timer3.Enabled = True
tr1 = -20
Command13.Enabled = False
Command14.Enabled = False
End Sub
Private Sub Command4_Click()
MSComm1.CommPort = 1
Label6.Caption = " Port: Com1"
```

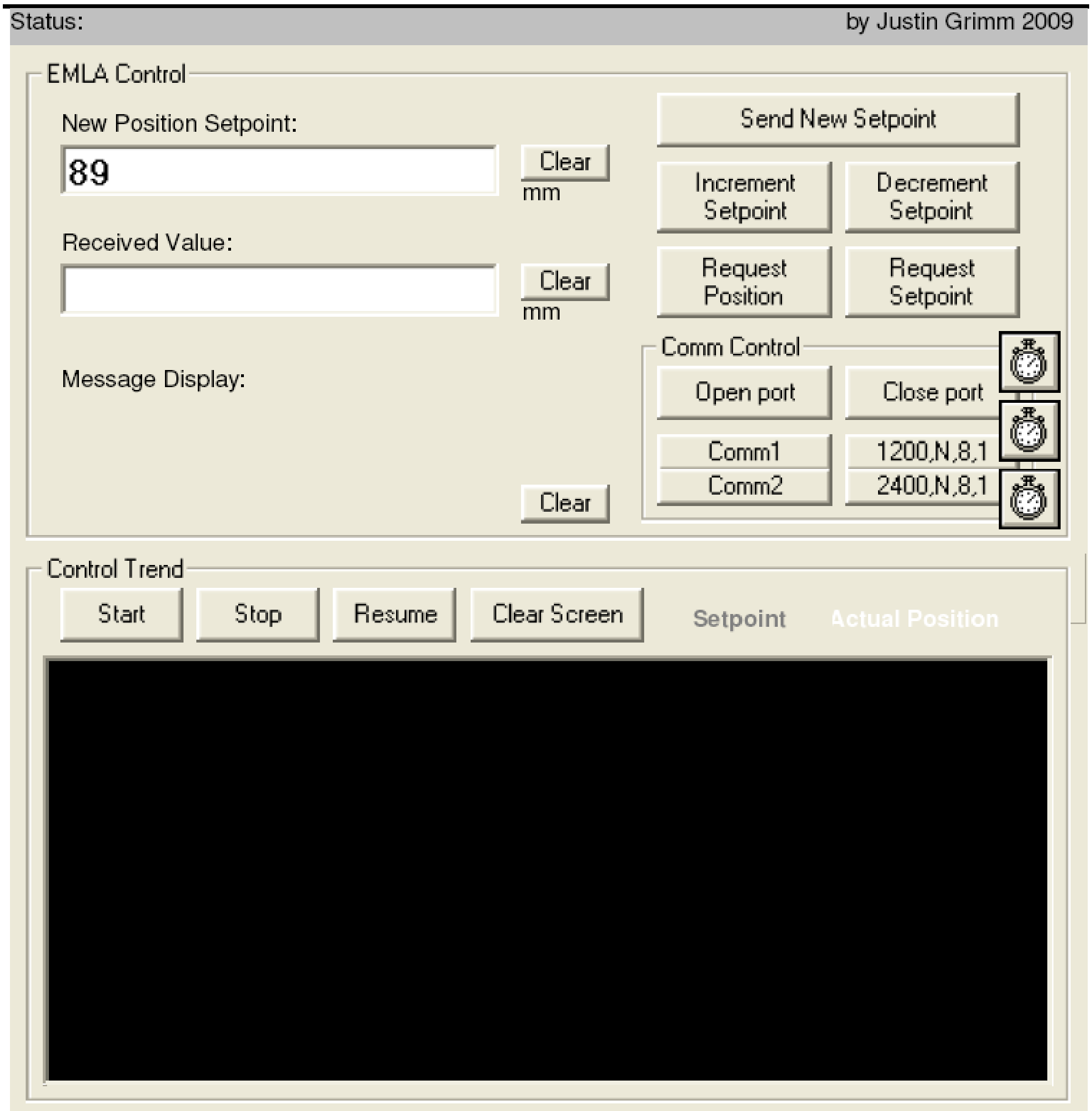
```
SimpTerm - 3
End Sub
Private Sub Command5_Click()
MSComm1.CommPort = 2
Label6.Caption = " Port: Com2"
End Sub
Private Sub Command6_Click()
MSComm1.Settings = "1200,N,8,1"
Label5.Caption = " Settings: 1200,N,8,1."
End Sub
Private Sub Command7_Click()
MSComm1.Settings = "2400,N,8,1"
Label5.Caption = " Settings: 2400,N,8,1."
End Sub
Private Sub Command8_Click()
MSComm1.PortOpen = False
Label4.Caption = "Port closed."
End Sub
Private Sub Command9_Click()
MSComm1.PortOpen = True
Label4.Caption = "Port open."
End Sub
Private Sub Form_Load()
MSComm1.RThreshold = 1
Label4.Caption = "Port closed."
Label5.Caption = " Settings: 2400,N,8,1."
Label6.Caption = " Port: Com1"
MSComm1.SThreshold = 0
flg4 = 1
flg5 = 0
Picture1.DrawWidth = 1.3
FillStyle = 0
Timer4.Enabled = False
Timer2.Enabled = False
Timer3.Enabled = False
kx = 1000
ay = 35
Timer4.Interval = 100 'sets speed of chart (mS/twip)
mvvar = 0 'dummy value for testing MV trend
pvvar = 0
spvar = 0
e = 0
laste = 0
scans = 0
Timer2.Interval = 500
Timer3.Interval = 1040
tr1 = -20
spvariable = 89
MSComm1.PortOpen = True
Label4.Caption = "Port open."
End Sub
Private Sub MSComm1_OnComm()
Select Case MSComm1.CommEvent
' Errors
Case comBreak ' A Break was received.
Label7.Caption = "Break Received"
' Code to handle a BREAK goes here.
Case comCDTO ' CD (RLSD) Timeout.
Label7.Caption = "CD Timeout"
Case comCTSTO ' CTS Timeout.
Label7.Caption = "CTS Timeout"
Case comDSRTO ' DSR Timeout.
Label7.Caption = "DSR Timeout"
Case comFrame ' Framing Error
Label7.Caption = "Framing Error"
Case comOverrun ' Data Lost.
Label7.Caption = "Data Lost"
SimpTerm - 4
```

```
Case comRxOver ' Receive buffer overflow.
Label7.Caption = "Receive Overflow"
Case comRxParity ' Parity Error.
Label7.Caption = "Parity Error"
Case comTxFull ' Transmit buffer full.
Label7.Caption = "Transmit Buffer Full"
' Events
Case comEvCD ' Change in the CD line.
Label7.Caption = "Change in CD Line"
Case comEvCTS ' Change in the CTS line.
Label7.Caption = "Change in CTS Line"
Case comEvDSR ' Change in the DSR line.
Label7.Caption = "Change in DSR Line"
Case comEvRing ' Change in the Ring Indicator.
Label7.Caption = "Change in Ring Indicator"
Case comEvReceive ' Received RThreshold # of chars.
Dim inchar As Byte
inchar = Asc(MSComm1.Input)
Text3.Text = inchar
Position = inchar
PV = (Position * -18) + 2970
SP = (spvariable * -18) + 2970
tr2 = tr1 + 10
Picture1.Line (tr2 + 50, 0)-(tr2 + 50, 3015), RGB(0, 0, 0)
Picture1.Line (tr2 + 60, 0)-(tr2 + 60, 3015), RGB(0, 0, 0)
Picture1.Line (tr2 + 70, 0)-(tr2 + 70, 3015), RGB(0, 0, 0)
Picture1.Line (tr2 + 80, 0)-(tr2 + 80, 3015), RGB(0, 0, 0)
Picture1.Line (tr2 + 90, 0)-(tr2 + 90, 3015), RGB(0, 0, 0)
Picture1.Line (tr2 + 100, 0)-(tr2 + 100, 3015), RGB(0, 0, 0)
Picture1.Line (tr2 + 10, 0)-(tr2 + 10, 3015), RGB(0, 0, 0)
Picture1.Line (tr2 + 20, 0)-(tr2 + 20, 3015), RGB(40, 40, 40)
Picture1.Line (tr2 + 30, 0)-(tr2 + 30, 3015), RGB(80, 80, 80)
Picture1.Line (tr2 + 40, 0)-(tr2 + 40, 3015), RGB(120, 120, 120)
Picture1.Line (tr1, PVold)-(tr2, PV), RGB(250, 250, 250) '(0, 255, 0)
Picture1.Line (tr1, SPold)-(tr2, SP), RGB(150, 150, 150) '(255, 255,
0)
tr1 = tr1 + 10
PVold = PV
SPold = SP
If tr2 = 7000 Then
tr1 = 0
End If
Label7.Caption = "Received RThreshold # of Characters"
Case comEvSend ' There are S(Threshold number of
' characters in the transmit buffer.
Label7.Caption = "SThreshold # of Characters in Transmit Buffer"
End Select
End Sub
Private Sub Timer4_Timer()
If MSComm1.PortOpen = False Then
MsgBox ("Error, Port not opened")
Timer4.Enabled = False
Timer2.Enabled = False
Timer3.Enabled = False
Exit Sub
Else
outchar = Chr$("&h" + "f7")
MSComm1.Output = outchar
End If
End Sub
Public Sub vertlines()
vertlines:
If kx < 7095 Then
SimpTerm - 5
ky = 35
ly = 3015
Picture1.Line (kx, ky)-(kx, ly), RGB(0, 0, 255)
kx = kx + 1000
```

```
GoTo vertlines
End If
End Sub
Public Sub horizlines()
horizlines:
If ay < 3015 Then
ax = 0
bx = 7095
Picture1.Line (ax, ay)-(bx, ay), RGB(0, 0, 255)
ay = ay + 750
GoTo horizlines
End If
End Sub

Public Sub vertlines()
vertlines:
If kx < 5000 Then
ky = 35
ly = 2035
Picture1.Line (kx, ky)-(kx, ly), RGB(0, 0, 255)
kx = kx + 1000
GoTo vertlines
End If
End Sub

Module2 - 1
Public Sub vertlines()
vertlines:
If kx < 5000 Then
ky = 35
ly = 2035
Picture1.Line (kx, ky)-(kx, ly), RGB(0, 0, 255)
kx = kx + 1000
GoTo vertlines
End If
End Sub
```


Status:
by Justin Grimm 2009
EMLA Control
New Position Setpoint:
89
Clear
mm
Received Value:
Clear
mm
Message Display:
Clear
Send New Setpoint
Increment Setpoint
Decrement Setpoint
Request Position
Request Setpoint
Comm Control
Open port
Close port
Comm1
1200,N,8,1
Comm2
2400,N,8,1
Control Trend
Start
Stop
Resume
Clear Screen
Setpoint
Actual Position

Appendix F Media

This appendix contains an electronic video showing the action of Prototype 2.

Appendix G Electronic Design Files

This appendix contains the schematic editor files for Prototype 2.

Appendix H Datasheets

This appendix contains the following electronic data sheets:

- PIC16F877A microcontroller,
- DT03AS PIC development board,
- DT106BCT PIC development board,
- LMD18200 h-bridge controller,
- PIC16 instruction set,
- PIC16 table reads.