DIGITAL CONTROL OF AN ELECTRO DISCHARGE MACHINING (EDM) SYSTEM

by

Azli Yahya

(B. Eng.(Hons.), MSc.)

A Doctoral Thesis
Submitted in partial fulfilment of the requirements
for the award of Doctor of Philosophy of the
Loughborough University

November 2005

ABSTRACT

This thesis presents a model of the complete Electro Discharge Machining (EDM) system and the design and implementation of a digital controller for the servomotor control and the gap voltage and current pulse power generator. A Matlab/Simulink simulation is used to investigate the EDM system model behaviour and based on the simulation results, a compensated EDM control system is designed. Simulation studies were also carried out to predict the material removal rate of a steel workpiece in mm^3/min. The control software of the EDM control process and servo system control was performed mainly in software with minimal hardware implementation. The control hardware consists of an eZdsp, user-interface device and analogue signal processing and interfacing circuit. The eZdsp communicates with the user-interface device by sending the information/instruction to the LCD screen while the user-interface device uses push button switches to communicate with the eZdsp. It is shown that one DSP microcontroller can be used to provide the control functions for the EDM system.

The experimental studies of the Electro Discharge Machining process using a copper electrode, a graphite electrode and steel workpiece materials are presented in tabular and graphical forms. The analysis of the experimental results show that the material removal rate is influenced by the process parameters such as the gap current I_{gap}, gap voltage V_{arc}, pulse on-time t_{on}, and sparking frequency F_s as well as the material properties of the electrode and the workpiece. Comparison studies between simulation and experimental results show reasonable agreement. Further improvement was made to the EDM process model based on the comparison studies. As a result, the predicted material removal rate using the improved EDM process model shows better agreement with experimental results.

ACKNOWLEDGEMENTS

This thesis was made possible through funding by the Ministry of Science Malaysia and support from the Universiti Teknologi Malaysia. It was a privilege to have the chance to research in the U.K.

I would like to express my thanks to Dr. Carl Manning for all his invaluable help, direction, expertise and kindness as well as encouragement to complete my thesis.

I am also indebted to the technical staffs and members of the Electrical Department of Loughborough University for their support while doing my laboratory experiments.

Last but not least, my humble gratitude and deepest appreciation goes to my beloved wife Norhalimah Idris, mother, in-laws, siblings, other family and friends for all their unconditional support, love, counsel, encouragement, prayer, patience, compassion and sacrifices. I dedicate this thesis to all of you who have helped me over the years.

TABLE OF CONTENTS

CHAPTER 4: HARDWARE DEVELOPMENT OF THE EDM SYSTEM 75

CHAPTER 5: EXPERIMENTAL STUDIES 87

LIST OF SYMBOLS AND ABBREVIATIONS

A/D	Analogue to Digital
A_E	Electrode inverse area
AEDG	Abrasive Electrical Discharge Grinding
C	Dimensionless constant
DC	Direct Current, A
d_{ls}	Leadscrew pitch
DSP	Digital Signal Processor
EDG	Electro Discharge Grinding
EDM	Electro Discharge Machining
F_s	Sparking frequency, kHz
FSM	Finite State Machine
I/O	Input / Output
I_a	Armature current, A
$I_a$peak	Peak armature current, A
IC	Integrated circuit
I_{gap}	Gap current, A
I_L	Inductor current, A
I_{ref}	Reference current, A
ISR	Interrupt Service Routine
J_m	Motor inertia, kg-m^2
K_n	Variable indices
K_{avg}	Average gap voltage proportional gain
K_e	Back e.m.f constant, V.s/rad
K_f	Total frictional torque coefficient, Nm.s/rad
K_{fl}	Frictional loss coefficient, Nm.s/rad

K_{ic}	Current integral gain
K_{iv}	Velocity integral gain
K_p	Proportional gain
K_{pc}	Current proportional gain
K_{pv}	Velocity proportional gain
K_t	Torque constant, Nm/A
K_{tac}	Tachometer coefficient, V/krpm
L_a	Armature inductance, mH
N_{dv}	Number of dimensionless variables
N_{gl}	Number of teeth at gear load
N_{gm}	Number of teeth at gear motor
N_v	Number of variables
PB	Push Button
PI	Proportional Integral
PWM	Pulse Width Modulation
R_a	Armature resistance, Ω
R_{DM}	Determinant rank
RERF	Reduce Erosion Rate Factor
S_n	State
T_c	Coulomb friction torque, Nm
t_D	Time delay, µs
t_{down}	Down time, s
t_{elect}	Electrical time constant, ms
T_f	Frictional torque, Nm
T_{fcoul}	Coulomb friction, Nm
T_{fric}	Torque friction, Nm
T_{fvisc}	Viscous friction, Nm
T_l	Load torque, Nm
T_m	Motor torque, Nm

t_{mech}	Mechanical time constant, ms
t_{off}	Pulse off-time, µs
t_{on}	Pulse on-time, µs
Tpeak	Peak torque, Nm
T_s	Static friction, Nm
t_{up}	Up time, s
V_a	Armature voltage, V
V_{arc}	Gap voltage, V
V_b	Back e.m.f, V
V_{gap_avg}	Average gap voltage, V
V_{gap_ref}	Gap voltage reference, V
V_{max}	Maximum voltage, V
$V_{N,man}$	Manual speed control, V
V_{oc}	Open circuit voltage, V
V_{tac}	Tacho voltage, V
z	Electrode position, m
$\dot{V}$	Material removal/erosion rate, mm^3/min
ω	Angular velocity, rad/s
δ	Gap width, m
α	Material properties factor, m^3/J
ϖ	Peak acceleration, rad/s^2
τ	Time constant, s
ξ	Workpiece surface position, m
π_n	Dimensionless variables
ω_m	Motor speed, rad/s

LIST OF FIGURES

LIST OF TABLES

LIST OF FLOWCHARTS

PREFACE

THE REQUIREMENT FOR THIS RESEARCH

This research is the result of an investigation to develop a model of a complete Electro Discharge Machining (EDM) system and uses the model in a computer simulation to predict the performance of EDM systems. Simulated results from the model are used to aid in the design and development of a DSP based controller for the servomotor control and the current generator control. The digital controller is to replace the conventional analogue controllers used in existing EDM systems.

THE SCOPE OF THIS RESEARCH

The core of this research considers a Die-sinking EDM system with a DC servomotor and an existing high efficiency switch-mode current generator with fast current pulse rise and fall times. The EDM process, which is a sequence of sparks is a discrete event system. Existing EDM process models were investigated with the view to selecting the most suitable model for simulation and in the event of a suitable model not being found, to develop one for accurate EDM process modelling.

The DC servomotor axis drive is a continuous-time system with considerably friction. Its purpose is to maintain accurate electrode position (10μm to 50μm gap length) in the z-axis relative to the moving eroded workpiece surface. The servomotor drive system should thus be modelled to accurately describe the electro-mechanical system dynamics subject to position and velocity control. Both the EDM process model and the servomotor model are combined into an EDM system model that can be used to accurately predict the material removal rate of the EDM system.

The final objective of this research is to develop a compact single DSP based controller for the EDM system. The results from the simulation should show good agreement with those from the practical system.

STRUCTURE OF THE THESIS

Chapter 1 gives an introduction to EDM systems and conducts a comprehensive literature survey of such systems.

Chapter 2 describes the modelling and simulation studies of a Die-sinking EDM system.

Chapter 3 explains the digital control strategy adopted for the EDM system. A TMS320LF2407A processor of DSP microcontroller was chosen for software implementation. The design strategy for current generator control is also explained.

Chapter 4 presents the system's hardware development. The circuit for the analogue-to-digital interface, feedback signal conditioning and various DSP on-chip peripherals are discussed. The design and implementation of a digital user-interface controller is explained.

Chapter 5 presents the material removal rate experimental results. Various experiments were undertaken for different EDM process parameters.

Chapter 6 compares the simulation results with the experimental results. The comparison was poor in many instances and investigation showed that the EDM process model used in the simulation was inadequate. A new improved EDM process model was then developed to identify the factors that influences material removal rate.

Chapter 7 concludes the studies and gives suggestion for further research.

CHAPTER 1

INTRODUCTION

1.1 ELECTRO DISCHARGE MACHINING (EDM) SYSTEMS

Electro Discharge Machining (EDM) is a process of electric conductive material removal using an accurately controlled electrical discharge (spark) through a small gap (approximately 10 to 50 microns) filled with dielectric fluid between an electrode and a workpiece. The technique allows machining high-strength and wear-resistant materials such as high-strength alloys, polycrystalline diamond and ceramic (ultra-hard conductive material) since the hardness of the workpiece has no effect on the process. Unlike the traditional cutting and grinding processes, which depends on the force generated by a harder tool to remove the softer material workpiece, the EDM process is free from contact force and chatter vibration. Furthermore, EDM technique permits the machining to be done even after the hardening process. The EDM process has been used in high precision machining of metals, and to date, there are several different types of EDM systems that were developed for specific industrial applications. EDM applications ranging from drilling micro-holes that are smaller than a human hair to the machining large automotive dies [1]. The two most common EDM systems are Die-sinking EDM and Wire EDM. Of the two, the Die-sinking EDM system presents the more challenging task as regards to the current generator and the servo system control requirements. For this reason, the Die-sinking EDM system is chosen to be the main focus of this research. However, many other EDM systems such as Milling, Grinding, Abrasive Grinding and Wire Grinding exist but these are less popular due to their limited application [1-2]. For completeness, the following sub-section gives a brief description of these EDM systems.

1.1.1 Types of EDM systems

EDM Milling uses cylindrical electrodes to perform the electrical discharge erosion in the form of milling. Consecutive passes of the electrode down to just less than the desired depth forms a mould cavity [3]. Basically, the method of material removal is similar to Die-sinking EDM. EDM Milling allows complex shapes to be machined using simple shaped electrodes. This is illustrated in Figure 1.1 where a custom-made electrode is used in Die-sinking EDM. However the main limitation in the EDM Milling process is that complex shapes with sharp corners cannot be machined due to the rotating electrode [4].

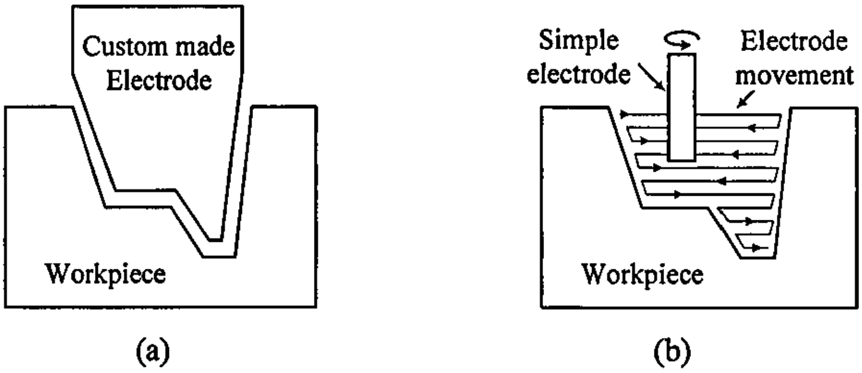


Figure 1.1 Electrode comparisons of (a) Die-sinking EDM and (b) EDM Milling

Electrical Discharge Grinding (EDG) is a non-contact process that is similar to Die-sinking EDM but uses a rotating conductive (i.e. carbon) wheel as the tool electrode. The name of EDG originated many years ago due to the original equipment that resembled a tool and cutter grinder [5]. EDG is also known as Rotary EDM. This is an alternative process for sharpening tungsten carbide and diamond tipped cutting tools such as polycrystalline diamond. The sparking area is sprayed with a dielectric fluid. The rotating electrode helps to keep the gap width free from debris and reduce electrode wear. Basic features of the EDG system are shown in the following Figure 1.2.

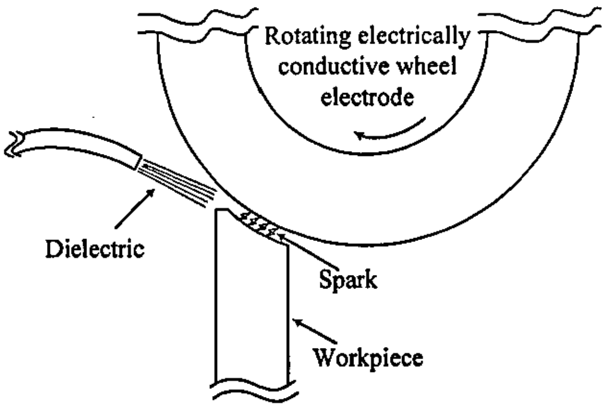


Figure 1.2 Basic features of the EDG system

A hybrid process of Abrasive Electrical Discharge Grinding (AEDG) removes advanced ultra-hard materials by a combination of action of electrical discharge erosion and mechanical grinding. A diagram illustrating the AEDG process is shown in Figure 1.3.

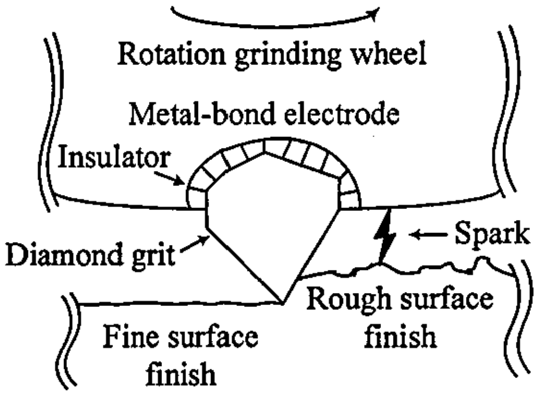


Figure 1.3 Electrical discharge and grinding process

This process is useful for machining polycrystalline conducting diamond materials. Electrical discharges perform the material removal and the mechanical grinding uses diamond grain grit for fine surface finish. The spark discharges, thermally soften the workpiece material in the grinding zone. According to [6], the grinding action removes the debris from craters and also decreases the chances of resolidification of molten material, leading to a higher material removal rate.

Wire EDMing was introduced around 1970, twenty years later than Die-sinking EDM [7]. The principle of Wire EDM is similar to Die-sinking EDM. The electrical sparks cause material removal, and the dielectric circulates through a deionizing system in the cooling module [8]. The system uses a travelling thin wire (approximately 50 to 300 microns) to cut through a workpiece as discharges takes place [9]. The wire electrode wears as it cuts and is therefore continuously replaced as it moves horizontally. The vertical path of the cutting direction is controlled by a computer program for particular shapes such as molds for IC lead frames. Figure 1.4 shows the basic features of Wire EDM.

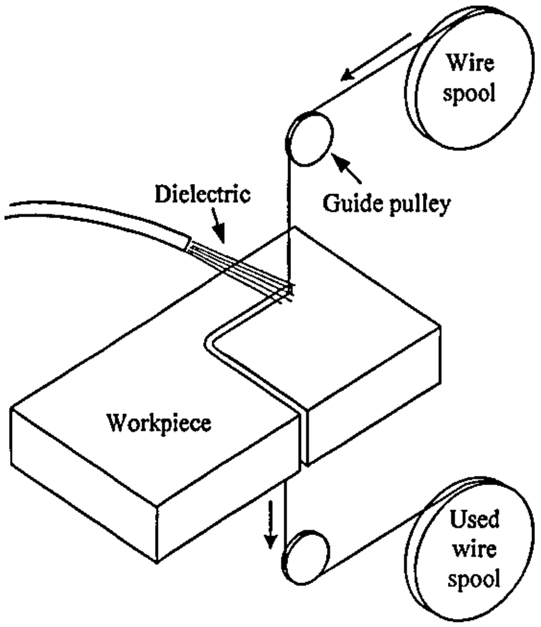


Figure 1.4 Basic features of the Wire EDM system.

The wire electrode wears during machining due to the electrical spark discharge and the combination of wire wear and the thermal load on the wire increases the risk of wire rupture. Some research has been carried out to improve EDMing wire electrode properties

[10-11]. The wire core material uses high electrical conductivity and high strength thermal resistance materials, while its coated layer uses a high concentration zinc (coated with zinc). The combination of both core and coated materials thus increase the electrical and thermal lower resistance strength of the wire electrode. However, preventing wire rapture by means of improving wire material has added an extra cost to Wire EDM. Alternatively, a control strategy can be applied to Wire EDM system to reduce the risk of wire rupture [12-13].

1.2 DIE-SINKING EDM SYSTEM

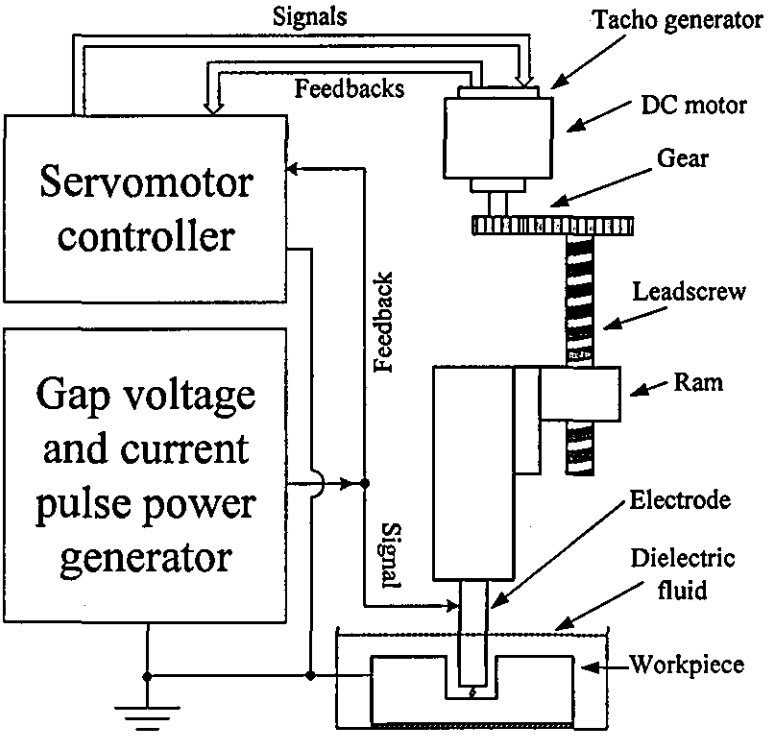
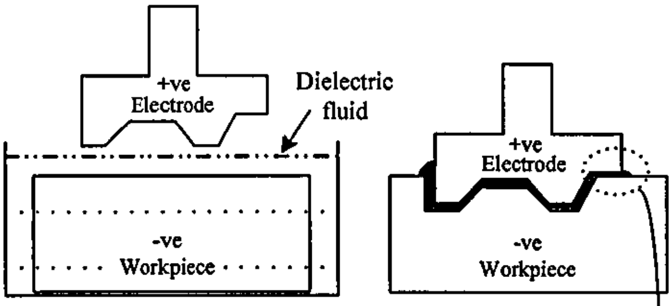


Figure 1.5 Die-sinking EDM system

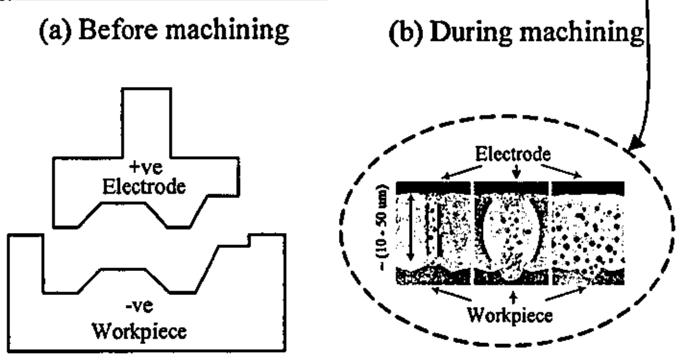
The Die-sinking EDM system is shown in Figure 1.5. The system consists of a servo system made up of a servomotor, leadscrew and ram holding the tool electrode and a gap voltage and current pulse power generator. The electrode and the workpiece are immersed in a tank containing dielectric fluid.

1.2.1 Die-sinking EDM Process

The process of removing material in Die-sinking EDM is shown in Figure 1.6. An electrical potential from the gap voltage and current pulse generator is created between the electrode and the workpiece. The electrode is slowly lowered towards the workpiece but there is no flow of current because of the strong insulating properties of the dielectric fluid.



(a) Before machining (b) During machining



(c) After machining

Figure 1.6 (a) Electrode and workpiece before machining (b) Development of plasma reaction (c) Formation of the workpiece

When the gap between the electrode and the workpiece is sufficiently small (10 - 50μm), said gap being controlled by the position control servo system, an electrical spark occurs in the gap between the electrode and the workpiece. In this process, which is also known as a discharge, current produces I^2R loss, which converted into heat. The surface of the material is intensely heated in the area of the discharge channel. If the flow of current is interrupted the discharge channel collapses very quickly. Consequently the molten metal on the surface of the material evaporates explosively and takes liquid material with it down to a certain depth. A small crater is formed. If one discharge is followed by another, new craters are formed next to the previous ones and the workpiece surface is constantly eroded. These particles immediately resolidify into small spheres and flushed away by the dielectric fluid.

Material removal rate is defined as the cavity volume removed by the spark, divided by the sum of the discharge pulse on-time t_{on} and pulse off-time t_{off} for each cycle of operation [14].

$$\text{Material removal rate} = \text{Cavity volume removed by spark} / (t_{on}+t_{off}) \qquad (1.1)$$

The spark phenomenon is shown in Figure 1.7.

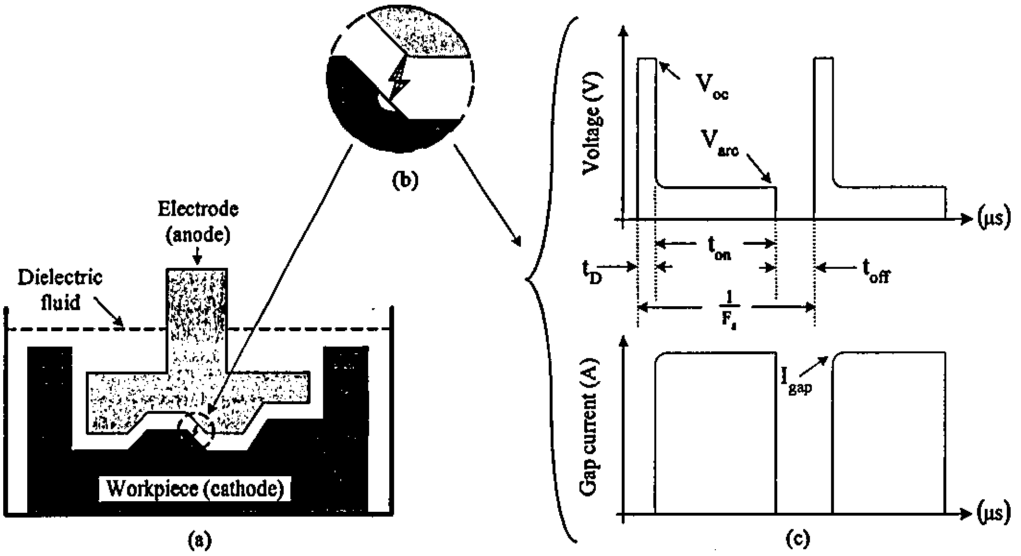


Figure 1.7 (a) Electrode and workpiece in dielectric fluid (b) Cross-sectional area showing electrical spark between electrode and workpiece (c) Profile of voltage across and current in spark gap

The profile of voltage across and current in the spark gap is illustrated in Figure 1.7(c). First, a high voltage V_{oc} is applied during the delay time t_D. During this time, a high electric field is formed between the electrode and workpiece (at the highest field strength). At the end of the delay time, the insulating effect of the dielectric fluid breaks down, current begins to flow whilst the voltage falls, signalling the start of the discharge phase. The spark is thus formed and machining takes place during the on-time t_{on} with a machining current I_{gap} and a voltage across the gap V_{arc}. At the end of the on-time, the flow of current is interrupted and the desired insulating electric properties of the dielectric fluid are recovered during the off-time t_{off}.

1.2.2 Electrodes and dielectric fluids in EDM

The electrode material properties are important factors to consider when choosing the electrodes. Those with the lowest electrical resistivity and the highest melting point are preferred. Traditionally, electrodes have been manufactured from metallic materials such as copper, tungsten, brass and steel. For non-metallic materials, graphite is preferred. Copper is normally chosen for electrode fabrication due to the fact that it is low in cost, easy to machine and widely available.

Research by [15] into electrode fabrication, used copper-tungsten for the electrode material due to its high wear resistance and better electrical conductive properties. A comparison study of micro-hole machining between tungsten-carbide electrodes and copper electrodes [16] revealed that better surface finish and lower electrode wear are obtained while machining with the copper electrode, whilst higher material removal is achieved when using the tungsten-carbide electrode. Tungsten-carbide possesses good wear resistance and high temperature resistance properties and its high stiffness makes it easy to machine in small diameters [17]. Sancez et al. [18] conducted research into EDM technology for advanced ceramics using three different electrode materials, namely pure graphite, copper-graphite and copper-tungsten. In their study, it was highlighted that the highest electrode wear occurs when using pure graphite followed by copper-graphite and copper-tungsten respectively. In addition, they also report that the worst surface conditions are obtained when using a graphite electrode.

Tool electrodes are manufactured using various techniques such as etching, stamping, turning, grinding and EDM processing [19-22]. The turning (lathe) machine easily cuts the tool electrode for EDM Milling since the shape is cylindrical [3]. However, this method is not applicable for complicated shapes. A set of gear components fabricated from multiple machining operations of turning, wire EDM and Die-sinking EDM is shown in Figure 1.8 [23].

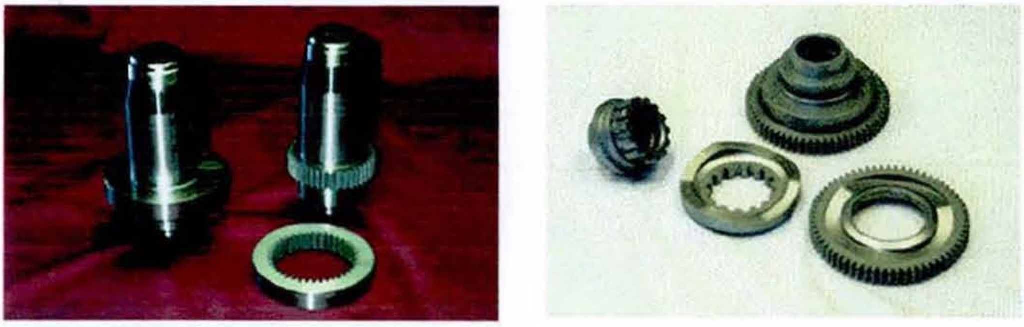


Figure 1.8 Gear-set components fabricated from multiple machining

The lathe machine cuts a blank tool electrode in a cylindrical shape. The blank tool electrode is then machined by Wire EDM to produce a geared tool electrode. Later, the gear tool electrode is use in Die-sinking EDM to fabricate the gear mould.

Research [24] states that Wire grinding EDM is an effective method to fabricate microelectrodes to very small diameters. A cylindrical electrode has been successfully machined down to 20μm in diameter and a rectangular electrode down to 15μm in corner radius. The EDM processing for the mesh electrode and the block electrode methods are also applicable for fabrication of microelectrodes. The mesh electrode method is used to produce multiple electrodes but machining to micrometre levels using this method is tedious due to the flushing difficulty. The block electrode method on the other hand is a preferable approach for producing microelectrodes due to its low cost and simple preparation compared to other methods [25]. However, machining electrodes via the EDM process is time consuming and could account for more than 50 percent of the total machining costs [26-27].

Electrode wear in EDM is unavoidable. However, with proper machining setup, the electrode erosion rate can be kept down to a minimal level. The term 'electrode wear' is referred to some amount of material removed from the tool electrode whilst machining. Electrode wear is in fact, a serious drawback of the EDM process. Research has been conducted to minimize electrode wear. However, such methods increase machining time [2], [28]. Nevertheless, most electrodes can be reused several times before losing their shape and dimensional accuracy.

Mineral oil hydrocarbon is the most common dielectric fluid used in EDM systems. However, the choice of dielectric fluid used depends on the EDM tasks. The dielectric fluid must have sufficient fluidity to enable it to flow freely between the narrow gap between the electrode and the workpiece [22], [29]. For example, high viscosity mineral oil is suitable for roughing, whilst low viscosity mineral oil is used in finishing tasks. In extremely small parts in wire EDM applications, water is used as the dielectric fluid. In this process, water is continuously de-ionized while being circulated in order to reduce its electrical conductivity to a suitable level [30]. It is important for the dielectric fluid to provide the electrical insulation, cool the electrode-workpiece, inflammable and flush away the erosion debris simultaneously. Furthermore, the dielectric fluid must ionize and de-ionize as quickly as possible, so that sparks can occurs with short pulse on and off times. The ability to continuously remove or flush away the eroded particles from the immediate sparking zone is one of the main criteria in choosing the dielectric fluid. Poor flushing results in electrode-workpiece bridges and short-circuits, which leads to thermal damage on the workpiece surface [31]. Open flushing is the most common method used in EDM processes and the technique is applied in this research. Figure 1.9 illustrates the open flushing technique in the EDM process.

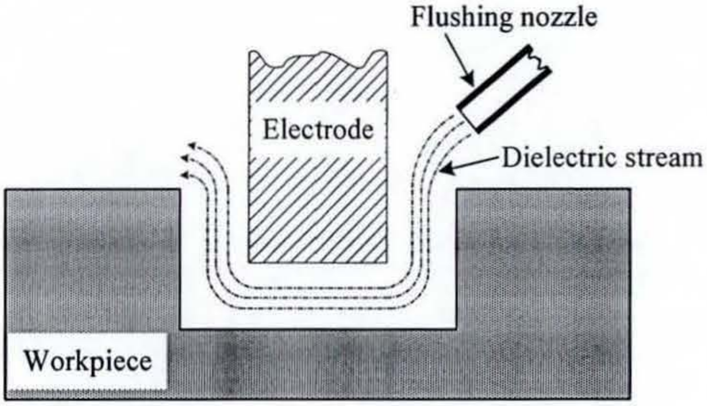
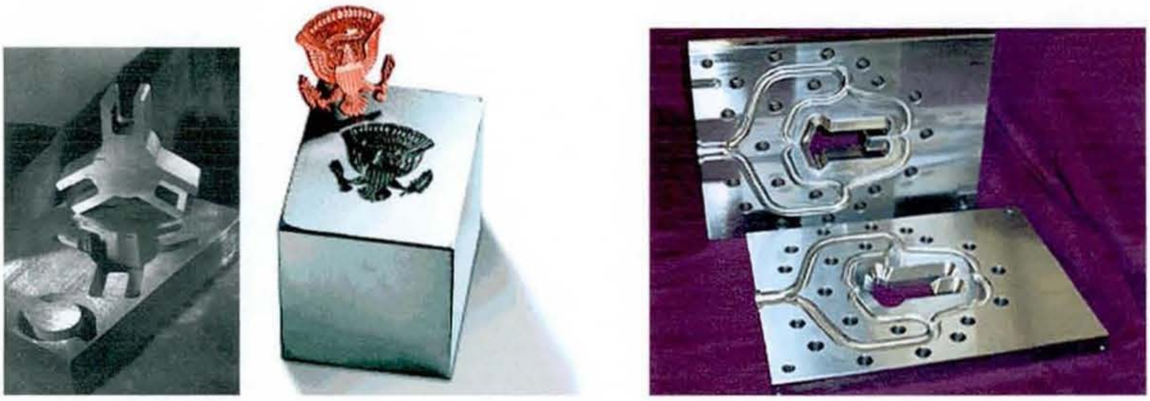


Figure 1.9 Open flushing

There are several other flushing methods used in EDM systems. For example in pressure flushing, the dielectric is pushed through a flushing hole in the electrode from the top direction or the workpiece from below. Suction flushing is suitable for a fine finish task where the eroded particles are sucked from the electrode-workpiece gap. For complicated shaped electrodes and complex tasks, a combined suction and pressure flushing is used.

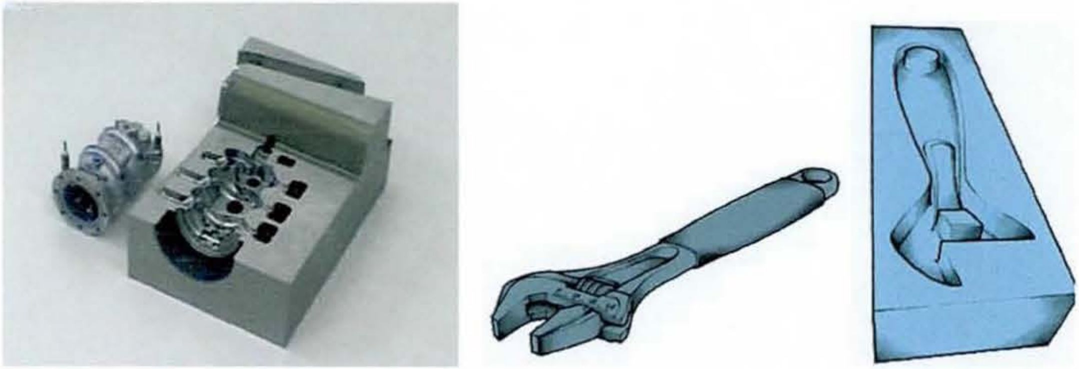
1.2.3 Workpiece finish

One of the major manufacturing industries using the EDM process is in the making of dies and molds. The main concern in the making of dies and moulds is the quality regarding the dimensional, geometrical and surface accuracy. A technical brochure [32] highlighted that better quality of surface finish is achieved by reducing the discharge energy. Low machining current however, produces relatively low material removal, thus increasing machining time. In some workshop practice, a rougher surface is initially machined with higher discharge energy followed by a finer one with reduced discharge energy. This procedure proves to be time saving of overall erosion process in relation to the degree of machining accuracy. Some of the samples are shown in Figure 1.10 [23].



(a) Emblems

(b) Tooling component



(c) Injector module

(d) Adjustable wrench

Figure 1.10 Sample of dies and moulds

A study [33] in material cracking behaviour reveals that machining parameters are directly related to surface damage. Experimental observation indicates that an increase in pulse on-time t_{on} enlarges the sparking zone, thus increases the number of cracks on the surface. A technical manual [29] elaborates that a short pulse off-time t_{off} gives positive impact on surface finish and increases erosion rate. In addition, the report explains that the surface finish is also influenced by the working gap. Increased discharge energy widens the working gap, and surface roughness becomes more pronounced.

The EDM process is also used in micro machining technology. The micro Die-sinking EDM process has become an important requirement [24], [28], [34]. In addition, this type of micro EDM is well known for its ability to machine holes down to diameters of 5μm [17], [20]. In machining micro holes, material removal rate is low due to the inefficient removal of debris. However, the flushing technique using ultrasonic vibration helps to improve the situation [35-37].

1.3 CONCLUSION

EDM is a non-traditional precision machining process. The technique is widely used in manufacturing industries ranging from large automotive dies down to drilling micro-holes in injector units. The EDM process removes material due to spark energy, thus allowing machining of high-strength and wear-resistant materials. Several different types of EDM systems are available, differentiated mainly by their particular applications. However, the method of removing material is fundamentally the same in all EDM based systems. The two most common EDM systems are Die-sinking and Wire EDM. Die-sinking EDM uses a tool electrode while Wire EDM uses a wire in machining a workpiece.

The Die Sinking EDM uses a high precision positioning control system to position the electrode away from the workpiece. The servomotor controller maintains the gap length in the range of 10μm to 50μm by processing the feedback signals from the servomotor and the gap voltage. An electrical potential from the gap voltage and current pulse generator unit is applied between the electrode and workpiece, which causes the insulating properties of the dielectric fluid in the gap to breakdown resulting in a spark current flow between the electrode and the workpiece. Material removal in the EDM process is dominated by the machining parameters such as on-time, off-time and gap current. High workpiece removal rate and low electrode wear can be obtained by proper machining set-up. However, it is difficult to satisfy these two conditions since other factors, such as electrode-material properties, dielectric fluid properties and the flushing technique influence, the EDM process.

The EDM process dominates the dies and moulds manufacturing industries, where the main concern is in regards to the dimensional, geometrical and surface accuracy. Lower discharge energy improves surface finish but increases machining time and cost. It is known that EDM is a slow material removal process compared to traditional machining such as turning, grinding and milling. However EDM can machine complex shapes and the hardness of the workpiece has no effect on the process. Free from contact force and chatter vibration, makes the EDM process suitable for micro parts and micro device fabrication. Micro Die-sinking EDM is able to machine holes down to a diameter of 5μm but is unable to flush the debris efficiently. As a result, it slows down the material removal rate.

In brief, Chapter 1 serves to explain the factors surrounding the issues of EDM systems and their applications. In the following chapter, a modelling and simulation study of a Die Sinking EDM system is presented.

CHAPTER 2

MODELLING AND SIMULATION STUDIES OF AN EDM SYSTEM

Modelling is a well-established engineering process and plays an important role in designing an engineering system. It is important that the model of the system is accurately developed as it provides the basis whereby the fundamental behaviour of the system can be determined. This chapter describes the development of the EDM system model. A dimensional analysis technique is applied in the development of the EDM process model and the servo system is modelled according to the differential equations of Newton's and Kirchhoff's laws. A Matlab/Simulink simulation is used to analyze the EDM system model behaviour and system compensation is performed in order to improve system performance. The simulation studies are also used to ascertain the erosion rate of material removal in $\mathrm{mm}^3/\mathrm{min}$.

2.1 EDM PROCESS MODEL

The EDM process is a technique for removing material using a complex combination of electrical, thermal and mechanical effects. A comprehensive description of the material removal process has been explained in section 1.2.1 of Chapter 1. Some mathematical models of the process have been developed previously based on the boundary condition of the plasma formed between the cathode (workpiece) and the anode (electrode). These models are a cathode erosion model [14] and an anode erosion model [38] and are based on the thermophysical properties of the plasma applied over the temperature range from solid to liquid melt. However, these models are presented in a complex relationship between the material and the plasma and are not compatible for use in the Matlab/Simulink simulation study intended by the author. In this work, the EDM process model is developed based on three sub models, which are a material removal rate model, a breakdown model and an average gap voltage model. The material removal rate model is developed from a Dimensional Analysis technique as discussed in the following sub section.

2.1.1 Material removal rate model using Dimensional Analysis

Application of Dimensional Analysis to determine material removal rate $\dot{V}$ can be presented by an equation of the form,

$$\dot{V} = f(x_1, x_2, x_3 \ldots\ldots\ldots\ldots x_n) \qquad (2.1)$$

A number of parameters have been identified that effect removal rate $\dot{V}$ [14], [29], [32], [38-40]. These parameters are the discharge pulse on-time t_{on}, the sparking frequency F_s $(1/(t_{on} + t_{off} + t_D))$, the gap current I_{gap}, the gap voltage V_{arc}, and the material properties factor α. Parameters t_{on}, F_s, I_{gap} and V_{arc} are shown in Figure 1.7. The α parameter is a factor representing the material properties of the electrode and the workpiece. Thus equation (2.1) can be formulated:

$$\dot{V} = f(t_{on}, V_{arc}, F_s, I_{gap}, \alpha) \qquad (2.2)$$

All of the parameters in equation (2.2) can be expressed in terms of four fundamental quantities of length L, mass M, time T and current I where the dimensions of the parameters are shown in Appendix 1. This leads to the following dimensional dependency matrix shown in equation (2.3).

$$\begin{array}{c|cccccc} & \dot{V} & t_{on} & V_{arc} & F_s & I_{gap} & \alpha \\ \hline L & 3 & 0 & 2 & 0 & 0 & 1 \\ M & 0 & 0 & 1 & 0 & 0 & -1 \\ T & -1 & 1 & -3 & -1 & 0 & 2 \\ I & 0 & 0 & -1 & 0 & 1 & 0 \end{array} \qquad (2.3)$$

Observation of the dimensional matrix shows that the number of variables N_v is six, the determinant is non-zero and the rank R_{DM} is four. Thus, the number of dimensionless variables N_{dv} needed to characterise the system is two as shown in equation (2.4).

$$N_{dv} = N_v - R_{DM} = 6 - 4 = 2 \qquad (2.4)$$

Following Langhaar [41], enables the dimensional matrix to be rewritten in the form:

$$\begin{array}{ccccccc} & k_1 & k_2 & k_3 & k_4 & k_5 & k_6 \\ & \dot{V} & t_{on} & V_{arc} & F_s & I_{gap} & \alpha \\ \hline L & 3 & 0 & 2 & 0 & 0 & 1 \\ M & 0 & 0 & 1 & 0 & 0 & -1 \\ T & -1 & 1 & -3 & -1 & 0 & 2 \\ I & 0 & 0 & -1 & 0 & 1 & 0 \end{array} \qquad (2.5)$$

where k_1, k_2, k_3, k_4, k_5 and k_6 are the indices of the variables in equation (2.2). The following set of homogenous linear algebraic equations can then be derived from consideration of the four rows of this matrix.

$$3k_1 + 2k_3 + k_6 = 0 \tag{2.6}$$

$$k_3 - k_6 = 0 \tag{2.7}$$

$$-k_1 + k_2 - 3k_3 - k_4 + 2k_6 = 0 \tag{2.8}$$

$$-k_3 + k_5 = 0 \tag{2.9}$$

which may be solved for k_3, k_4, k_5 and k_6 in terms of k_1 and k_2. Two dimensionless variables written as π_1 and π_2 are now introduced [42], with k_1 and k_2 assigned values of dimensionless variables π_1 and π_2 as shown in the square identity matrix of equation (2.10).

$$\begin{array}{ccccccc} & k_1 & k_2 & k_3 & k_4 & k_5 & k_6 \\ & \dot{V} & t_{on} & V_{arc} & F_s & I_{gap} & \alpha \\ \hline \pi_1 & 1 & 0 & -1 & 0 & -1 & -1 \\ \pi_2 & 0 & 1 & 0 & 1 & 0 & 0 \end{array} \tag{2.10}$$

↑
Square identity matrix

The values for k_2, k_3, k_4, k_5 and k_6 shown in the matrix of equation (2.10) were obtained as follows.

For row π_1: $k_1 = 1$ and $k_2 = 0$

Equation (2.7) can be rearranged to be $k_3 = k_6$. Substituting for k_6 in terms of k_3 into equation (2.6) results in $k_3 = -1$. Thus, $k_6 = -1$ and from equation (2.8), $k_5 = -1$. By replacing all known k_n values in equation (2.8), k_4 is found to be zero.

Similarly for row π_2: $k_1 = 0$ and $k_2 = 1$

Substituting these values into equations (2.6) to (2.9), gives $k_3 = 0$, $k_4 = 1$, $k_5 = 0$ and $k_6 = 0$. The set of dimensionless parameters resulting from the matrix of equation (2.10) is as follows:

$$\pi_1 = \frac{\dot{V}}{V_{arc} I_{gap} \alpha} \tag{2.11}$$

$$\pi_2 = t_{on} F_s \tag{2.12}$$

According to Buckingham's theorem, the dimensionless parameters are related as follows:

$$\pi_1 = f(\pi_2) \tag{2.13}$$

where f is a function of the dimensionless parameter π_2. As far as dimensions are concerned, the monomial power form is always applicable [42]. Thus, equation (2.13) can be expressed as:

$$\pi_1 = C\pi_2^n \tag{2.14}$$

where C is a dimensionless constant and n is an unknown power. Combining equations (2.11) and (2.12) into equation (2.14), gives the following equation:

$$\frac{\dot{V}}{V_{arc} I_{gap} \alpha} = C(t_{on} F_s)^n \qquad (2.15)$$

Mathematically, the constant index n can be solved using an Indicial Method discussed in [43]. However, experimental evidence [42], [44] enables equation (2.15) to be simplified in that the constant index n can be made equal to 1 since the material removal rate is proportional to both the pulse duration t_{on} and the sparking frequency F_s. Therefore, the material removal rate is given by equation (2.16):

$$\dot{V} = C\alpha V_{arc} I_{gap} t_{on} F_s \qquad (2.16)$$

The dimensionless constant C is determined from analysis of experimental data [14] as explained below.

The first step in evaluating C is to rearrange equation (2.16) as follows:

$$C = \frac{\dot{V}}{\alpha V_{arc} I_{gap} t_{on} F_s} \qquad (2.17)$$

Then, C is calculated by inserting the experimental data into equation (2.17). From [14], an $\alpha = 2 \times 10^{-12}$ m^3/J and V_{arc}=25V were used to calculate the C value at I_{gap} of 8.5A, 12.5A, 18A, 25A, 36A and 50A. Thus, the average value of C was found as 1.74.

2.1.2 Empirical breakdown model

When the gap width between the electrode and the workpiece is sufficiently small, discharge takes place. Experimental observations [45] on this breakdown phenomenon show that the gap position δ is nonlinearly related to the ignition delay time t_D and to a lesser extent on dielectric fluid flushing velocity. Figure 2.1 shows reproducted data from reference [45], of the average ignition delay in (μs) and the gap width in (μm) for a typical flushing velocity of 1m/s.

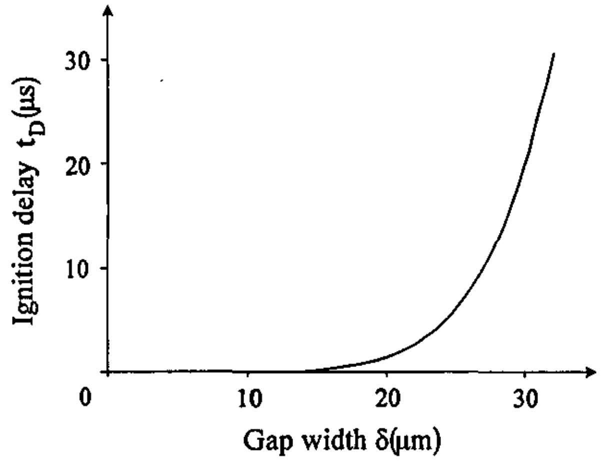


Figure 2.1 Experimental curve of breakdown model

Using this data, a curve fitting technique [46] was used to generate the following equation relating the ignition delay to the gap width:

$$t_D = v \times \delta^n \tag{2.18}$$

where,

$$v = 1.04 \times 10^{25} \text{ and } n = 6.57$$

2.1.3 Average gap voltage model

Regulation of the gap width is achieved in the prototype (Chapter 4) by using the average gap voltage as an indirect measure indicator of the gap width. An average gap voltage is calculated according to the following equation;

$$V_{gap\_avg} = \frac{(V_{max} \times t_D) + (V_{arc} \times t_{on})}{t_{on} + t_{off} + t_D} \tag{2.19}$$

This is then filtered with a filter time constant τ. The three sub-sectioned EDM process models have been combined to form the complete Simulink EDM process model as shown in Figure 2.2.

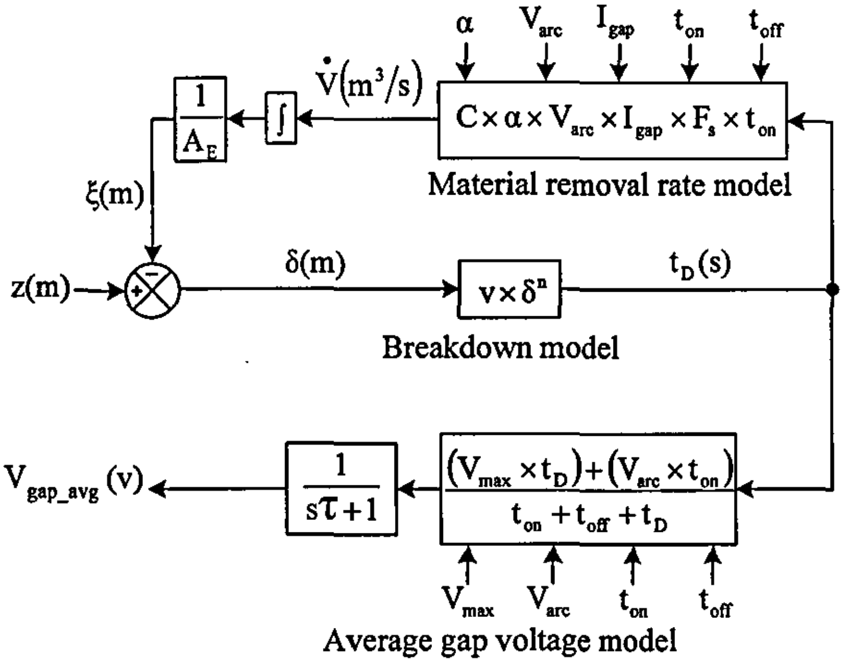

α
V_{arc}
I_{gap}
t_{on}
t_{off}
$\dot{V}(m^3/s)$
$C \times \alpha \times V_{arc} \times I_{gap} \times F_s \times t_{on}$
Material removal rate model
$\frac{1}{A_E}$
∫
ξ(m)
z(m)
+
-
δ(m)
$v \times \delta^n$
$t_D(s)$
Breakdown model
V_{gap_avg} (v)
$\frac{1}{s\tau+1}$
$\frac{(V_{max} \times t_D) + (V_{arc} \times t_{on})}{t_{on} + t_{off} + t_D}$
V_{max}
V_{arc}
t_{on}
t_{off}
Average gap voltage model

Figure 2.2 Model of EDM process

The gap width δ is the error between the electrode position z and the workpiece surface position ξ. The time delay t_D is a function of the gap width as shown by the breakdown model and is one of the parameters that affects the material removal rate. By taking the integral of the material removal rate, volumetric material removal is obtained. The volumetric material removal is then divided by the electrode area A_E in order to obtain the workpiece surface position.

2.2 SERVO SYSTEM MODEL

The servo system consists of two major subsystems; an armature voltage controlled permanent magnet DC motor with its controller and a lead-screw load containing the tool electrode. The leadscrew load consists of gears, lead screw shaft and ram. The gears are located between the motor shaft and the lead-screw shaft with the ram holding the electrode. Figure 1.5 in Chapter 1 illustrates the mechanical structures of the EDM servo system. The servo system model transfer function is calculated based on the total inertia of the system.

2.2.1 DC motor model

Verification of the DC motor model is carried out through a comparison analysis between simulation and experimental test. The Simulink model is used in simulation analysis and the results are compared with the information provided from the manufacturer data sheet for this particular DC motor. A practical test on the DC motor is conducted for further validation of the motor. In general, the equivalent circuit as shown in Figure 2.3 represents the DC motor.

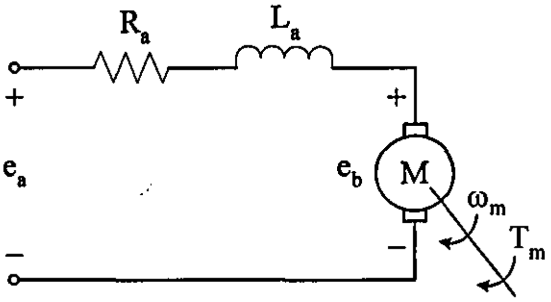


Figure 2.3 DC motor equivalent circuit

A combination of Newton's Second Law of Motion and Kirchhoff's Voltage Law gives the following differential equations [47-49].

$$e_a(t) = L_a \frac{di_a(t)}{dt} + R_a i_a(t) + e_b(t) \qquad \text{(V)} \quad (2.20)$$

$$T_m(t) = J_m \frac{d\omega_m(t)}{dt} + K_f \omega_m(t) \qquad \text{(Nm)} \quad (2.21)$$

where,

$$e_b(t) = K_e \omega_m(t) \qquad \text{(V)} \quad (2.22)$$

$$T_m(t) = K_t i_a(t) \qquad \text{(Nm)} \quad (2.23)$$

A Simulink model using a Transfer Function approach is used to represent the DC motor in block diagram form and the parameters are defined in a Matlab m-file (see Appendix 2). The Laplace Transform of equations (2.20) to (2.23) gives the Simulink model depicted in Figure 2.4. The DC motor position $\theta_m(s)$ is obtained by taking the integral of the motor speed $\omega_m(s)$.

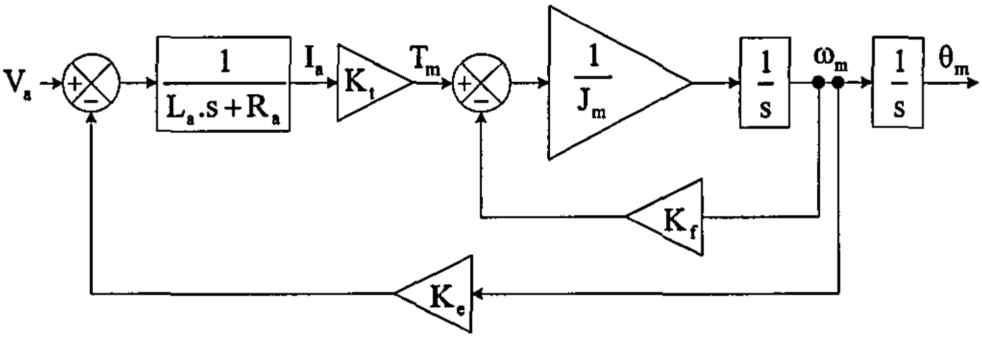


Figure 2.4 DC motor Simulink model

2.2.1.1 DC motor linear model

An initial linear model of the DC motor as shown in Figure 2.5 is used in simulation analysis.

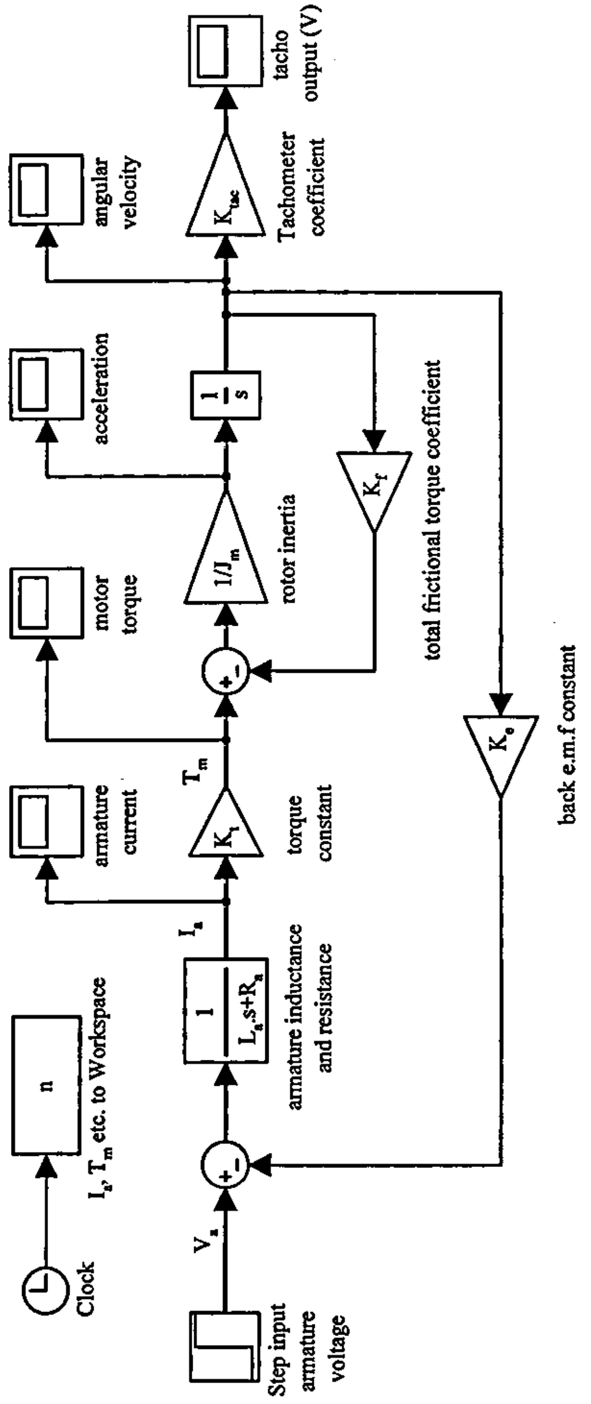


Figure 2.5 Simulink linear model of unloaded DC motor

The parameters for the model are listed in Table 2.1. All of the parameters, except K_f are manufacturer's data sheet values.

Table 2.1 DC motor parameters

		From data sheet		Use in simulation	
Motor parameter	Symbol	Value	Units	Value	Units
Armature inductance	L_a	3	mH	3×10^{-3}	H
Armature resistance	R_a	0.9	Ω	0.9	Ω
Torque constant	K_t	1.38	kg.cm/A	0.1356	Nm/A
Rotor inertia	J_m	2.3×10^{-3}	kg.cm-s^2	0.2255×10^{-3}	kg.m^2
Viscous friction coefficient	K_f	-	-	0.268×10^{-3}	Nm.s/rad
Static friction	T_s	0.44	kg.cm	-	Nm
Back e.m.f contant	K_e	14.2	V/krpm	0.1356	V.s/rad

In some cases, the units obtained for a parameter from the manufacture data sheet were not expressed in the standard metric unit of SI. Thus, those units as shown in Table 2.1 have been converted to the appropriate SI unit for consistency purpose in simulation analysis. K_f was obtained from tests carried out on the motor as explained in the following subsection.

2.2.1.2 Friction coefficient, K_f

In a DC motor, any resistance to movement results in the force acting in opposition to the rotational motion. This term is known as frictional torque. A typical frictional torque-speed characteristic is shown in Figure 2.6.

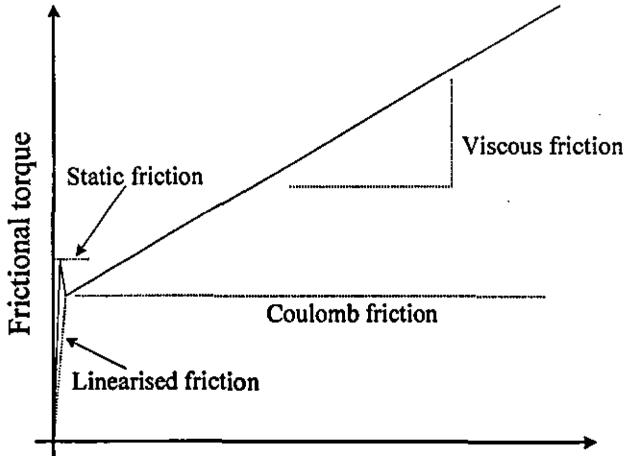


Figure 2.6 Components of friction

Three friction component of the motor torque are Static, Coulomb and Viscous. These are affected by bearings, brush material, air gap flux density, lubricant and the magnetic circuit configuration [50-52]. Static friction or so-called stiction torque is the torque required for the drive to commence rotation. As the drive starts to rotate (break-away), the initial static friction falls quickly to a value known as the Coulomb friction. The viscous friction then takes over and varies proportionally to the angular velocity of the drive. In many cases, viscous friction is non-linear. However for simplicity of simulation, the frictional torque for the motor is defined by the following linear relationship [53]:

$$\text{Frictional torque, } T_f = K_f.\omega \qquad (2.24)$$

where K_f is the friction coefficient in (Nm.s/rad) and ω is the angular velocity in rad/s. Neglecting static friction does not give rise to significant simulation errors, providing the motor is not required for the motor to accelerated to high velocities and decelerated quickly in a repetitive process. It should be noted that coefficients for Coulomb and Viscous friction are not provided in the manufacturer data sheet. According to Figure 2.4, friction coefficient K_f is multiplied with the angular velocity in order to provide the total frictional torque to the motor. The friction coefficient is found through the experimental tests on the motor. In an open loop test, 24V DC was applied to the armature. This is, the maximum voltage that will be applied to the motor armature by the servo amplifier in the practical servo control system as explained in Chapter 4. The motor was allowed to accelerate to a steady state velocity without load. Fixed to the motor is a tachometer supplied by the manufacturer, with a coefficient of 7V/krpm. The tachometer output was measured at 11.46V and the armature current was measured at 0.34A. An SI conversion gives the tachometer coefficient of 0.067V/rad.s^{-1}. Knowing that at steady state, the motor torque is equal to load torque (frictional torque losses in the un-loaded case), K_f is calculated as follows;

$$T_f = T_m = K_t . I_a = 0.1356 \times 0.34 = 0.0461\text{Nm}$$

Rearranging equation (2.24);

$$K_f = \frac{T_f}{\omega} = \frac{0.1356 \times 0.34}{11.46 / 0.067} = 0.268 \times 10^{-3}\text{Nm.s/rad} \qquad (2.25)$$

2.2.1.3 Linear model verification

Simulation analysis is carried out using the parameters presented in Table 2.1 in order to verify the model parameters. This first simulation was undertaken with a step input of 48V (rated armature voltage) in order to determine from simulation the maximum angular velocity, peak armature current, peak torque, peak acceleration, and the mechanical and electrical time constants. A comparison with the manufacturer's data sheet values will indicate the validity or otherwise of the selected model parameters. Various simulation results are shown in Figures 2.7-2.10 and the results are summarised in Table 2.2.

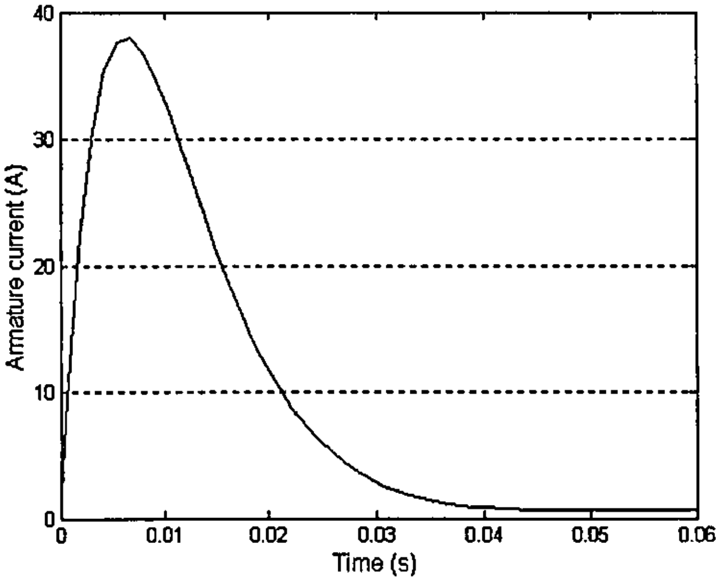


Figure 2.7 Time response of armature current

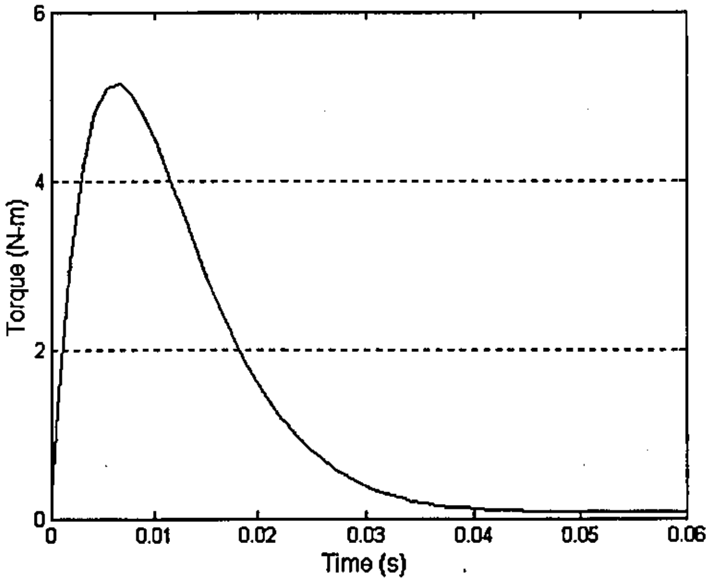


Figure 2.8 Time response of torque

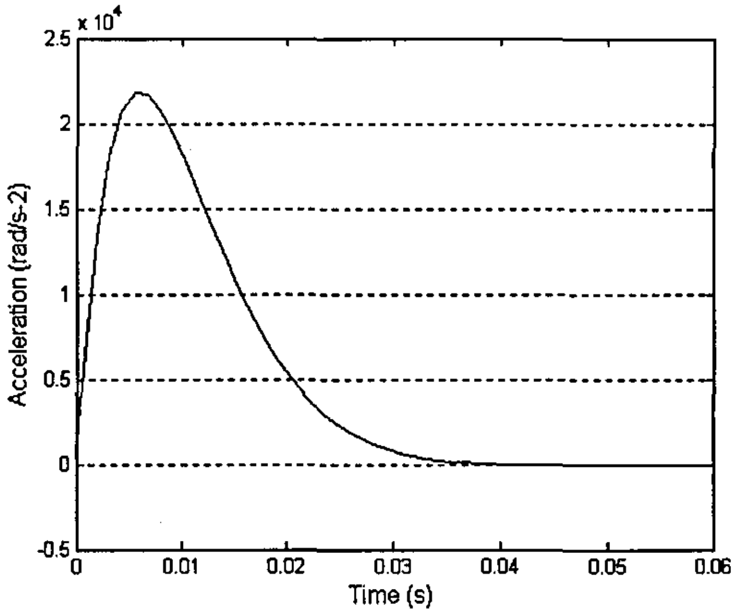


Figure 2.9 Time response of acceleration

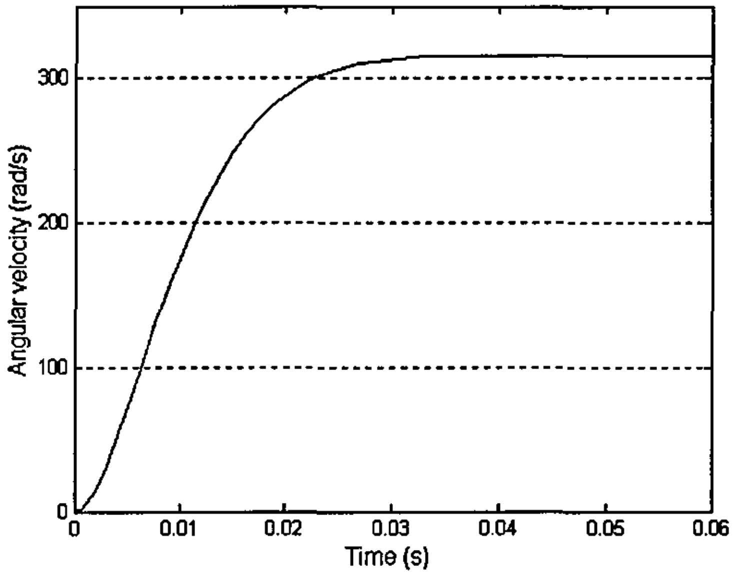


Figure 2.10 Time response of angular velocity

Table 2.2 Comparison between manufacturer data sheet and simulation data

Motor parameters	Symbol	Unit	Value stated on data sheet	Measured value from simulation
Angular velocity	ω	$rads^{-1}$	314.16	315.4
Peak armature current	$I_a$peak	A	40	38.1
Peak torque	Tpeak	Nm	4.7	5.2
Peak acceleration	ϖ	$rads^{-2}$	21×10^3	21.8×10^3
Mech. time constant	t_{mech}	ms	12	11.5
Elect. time constant	t_{elect}	ms	3	2.1

For a given step input of 48V amplitude, the results show only slight differences when compared. Thus, for the linear model, the parameters used in the simulation are adequate to represent the fundamental motor model in the simulation analysis aimed at predicting peak values of the motor variables as shown in Table 2.2. However, further analysis will be carried out for low level armature voltage amplitudes to check the performance consistency of the model at low velocities.

Practical tests were therefore undertaken on the motor with armature voltages in the range of 1.17V to 24V. The voltage as measured from the tachometer output represents the motor velocity. The angular velocity in rad/s can be obtained by dividing the tachometer voltage and by the tachometer coefficient of 0.067. The results are then compared with the simulation results for the same input magnitude. The results are shown in tabular form in Table 2.3 and in graphical form in Figure 2.11.

Table 2.3 Comparison results for armature voltages in the range of 1.17V to 24V

Experimental DC motor open loop test results							
Applied Armature voltages, V_a (V)	1.17	2	5	10	15	20	24
Measured current, I_a (A)	0.22	0.23	0.25	0.29	0.31	0.33	0.34
Measured tacho voltage, V_{tac} (V)	0.18	0.6	1.98	4.48	6.86	9.35	11.32

Simulink DC motor open loop test results							
Applied Armature voltages, V_a (V)	1.17	2	5	10	15	20	24
Measured current, I_a (A)	0.02	0.03	0.06	0.13	0.19	0.26	0.33
Measured tacho voltage, V_{tac} (V)	0.57	0.97	2.43	4.87	7.31	9.74	11.68

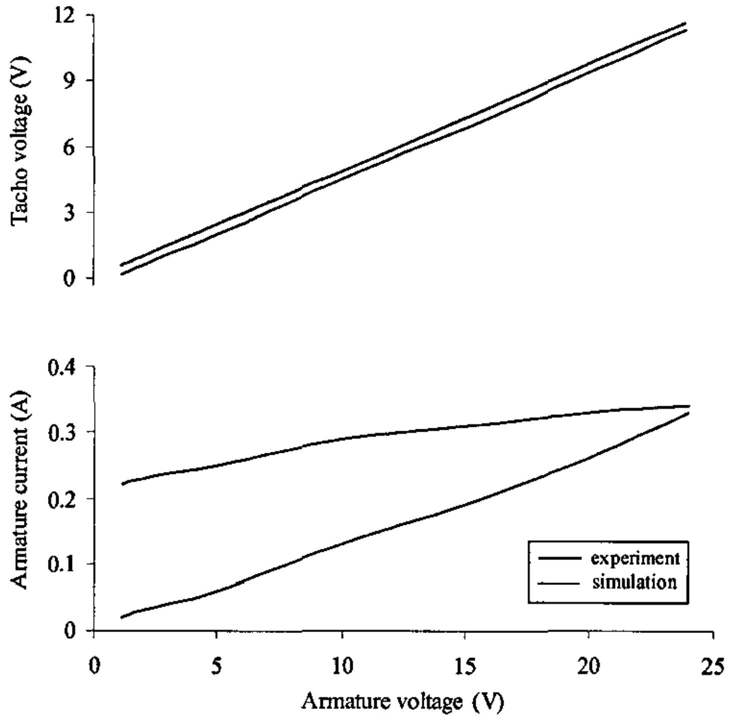


Figure 2.11 Open loop test results of DC motor

The difference between experimental and simulation results for the armature current is clearly seen in Figure 2.11, especially at low voltages. However, as the applied armature voltage increases, the simulation results for armature current tend to match with the practical results. This indicates that at low velocities, the resulting linear frictional losses in the linear model do not account correctly for the dominant Coulomb friction. Thus, to give a true representation of the actual DC motor behaviour at low applied armature voltages, a nonlinear model needs to be developed.

2.2.1.4 DC motor nonlinear model

A nonlinear model of the DC motor is based on the nonlinear behaviour of frictional losses. The two most common frictions that are always considered in the nonlinear model are viscous and Coulomb friction. These friction phenomena have been mentioned briefly in section 2.2.1.2 with the aid of Figure 2.6. The Viscous friction is proportional to the angular velocity and in the model it is always considered as a linear function with respect to the change of the angular velocity [54]. Coulomb friction on the other hand does not depend on the angular velocity but exists subject to the sign (+/-) of the angular velocity. A general description to describe those frictions in the nonlinear DC motor model is shown in equation (2.26).

$$T_{fric} = \begin{Bmatrix} T_{fcoul} + T_{fvisc} & \omega_m(t) > 0 \\ 0 & \omega_m(t) = 0 \\ -T_{fcoul} - T_{fvisc} & \omega_m(t) < 0 \end{Bmatrix} \qquad (2.26)$$

Thus equation (2.21) becomes, for the nonlinear model;

$$T_m(t) = J_m \frac{d\omega_m(t)}{dt} + T_{fvisc} + T_{fcoul}(\omega_{sign}) \qquad (2.27)$$

where

$$T_{fvisc} = K_f . \omega(t) \qquad (2.28)$$

The linear model of the DC motor in Simulink is therefore modified accordingly in order to include the nonlinear friction as shown in Figure 2.12.

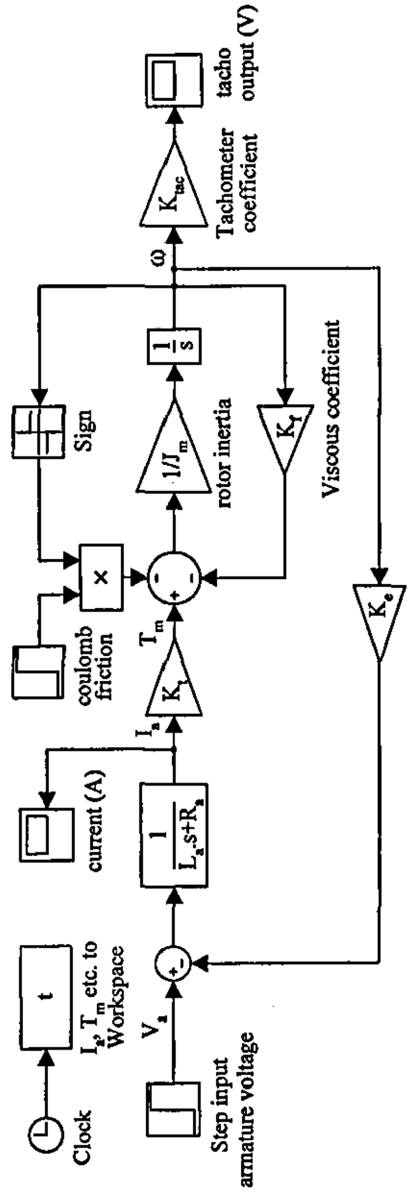


Figure 2.12 Simulink nonlinear model of a DC motor

The Coulomb and Viscous frictions were determined from analysis of open loop motor tests. The test results and the calculated friction components are shown in Table 2.4. The armature voltage of 1.17V was applied to commence rotation and viscous friction losses as rotation commences were assumed to be zero. The resulting calculated friction torque was therefore assumed to be purely Coulomb friction torque. The results of current and angular velocity were obtained for a range of armature voltages.

Table 2.4 Determination of coulomb and viscous friction

Measured during open loop velocity tests							
Applied Armature Voltages, V_a (V)	1.17	2	5	10	15	20	24
Measured current, I_a (A)	0.22	0.23	0.25	0.29	0.31	0.33	0.34
Measured tacho voltage, V_{tac} (V)	0.18	0.6	1.98	4.48	6.86	9.35	11.32
Equivalent angular Velocity (rad/s)	2.69	8.97	29.62	67.02	102.63	139.87	169.34
Torque (Nm)	0.029	0.031	0.033	0.039	0.042	0.045	0.046
Coulomb friction torque loss (Nm)	0.03						
Viscous friction torque loss (Nm)	-	0.001	0.003	0.009	0.012	0.015	0.016
Approximating K_f to linear curve (Nm.s/rad)	9.66×10^{-5}						

The Coulomb friction torque as calculated above is an estimation, and will have to be assesed in simulation. To improve the most accuracy of the simulation results, the frictional components were adjusted slightly. The Coulomb friction is chosen to be 0.035Nm and the viscous friction coefficient is 9.66×10^{-5}Nm.s/rad. A complication arises when changing the rotational direction in a single simulation test. The complication is one of instability of the motor speed around the zero speed. To overcome the problem, a look-up table is used to represent the frictional losses. Due to the fact that static and Coulomb frictional losses are 0.04Nm and 0.035Nm respectively, when rotation begins, the look-up table linearises the value of the friction losses between zero and Coulomb friction loss as shown in Figure 2.6.

With the new frictional loss representation, error in the armature current was reduced but still proved to be unacceptable. A potential source of the inaccuracy is the value of the armature resistance chosen for the simulations, which neglects brush voltage drops [55]. To give an indication of the effective armature resistance, a voltage of 1.5V DC was applied to the motor terminals with the rotor locked in a stationary position, the current flowing through the armature was measured at 0.526A. Performing a fundamental calculation using Ohm's Law;

$$\text{Resistance, } R = \frac{V}{I} = \frac{1.5}{0.526} = 2.85\Omega$$

The armature resistance quoted on the data sheet for the DC motor is 0.9Ω. The measured differences are due to the brush voltage drop and reduced quality of the electrical contact with the comutator for small currents. The brush voltage drop is a non-linear parameter and difficult to model. The simulated brush voltage drop effect was modelled by using a fixed value of resistance of 2.85Ω (1.95Ω, added to the armature resistance).

The revised DC motor parameters used to produce DC motor simulation results are given in Table 2.5.

Table 2.5 Revised DC motor parameters used in the Simulink model

Motor parameter	Symbol	Used in simulation	
		Value	Units
Armature inductance	L_a	3×10^{-3}	H
Armature resistance	R_a	2.85	Ω
Inertia of rotor and gear motor	J_m	0.2255×10^{-3}	$Kg.m^2$
Back e.m.f contant	K_e	0.1356	V.s/rad
Torque constant	K_t	0.1356	Nm/A
Viscous friction coefficient	K_f	9.66×10^{-5}	Nm.s/rad
Coulomb friction torque	T_c	0.035	Nm

The revised parameters were used in the nonlinear Simulink model shown in Figure 2.12. The simulation results for tacho voltage and motor current are shown in Table 2.6.

Table 2.6 Simulation results for armature voltages in the range of 1.17V to 24V

Simulink nonlinear DC motor open loop test results							
Applied Armature voltages, V_a (V)	1.17	2	5	10	15	20	24
Measured current, I_a (A)	0.26	0.265	0.28	0.31	0.33	0.35	0.37
Measured tacho voltage, V_{tac} (V)	0.21	0.61	2.07	4.49	6.92	9.35	11.3

Figure 2.13 shows the graphical comparison between the simulation and the experimental results for the DC motor without load. It can be seen that the Simulink model results now compare favourably with the practical results.

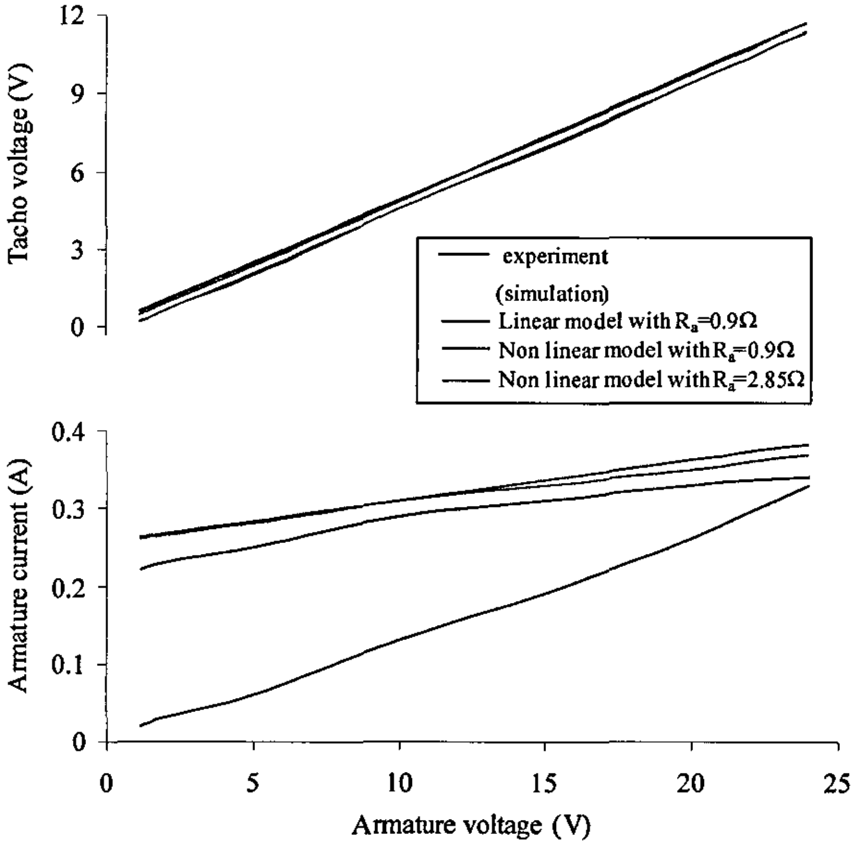


Figure 2.13 Open loop test results of nonlinear DC motor

2.2.2 DC motor model with EDM mechanical load

Open loop tests as described in section 2.2.2.2 of the DC motor with mechanical load were also performed to determine the load torque and the viscous friction losses for the model with load. The mechanical system model equations are used to determine the inertias of the EDM mechanical system. The calculated inertias are later included in the Simulink model for simulation analysis of the DC motor with mechanical load.

2.2.2.1 Mechanical system inertias and load torque

The mechanical system inertias are calculated based on the mechanical EDM system illustrated in Figure 2.14.

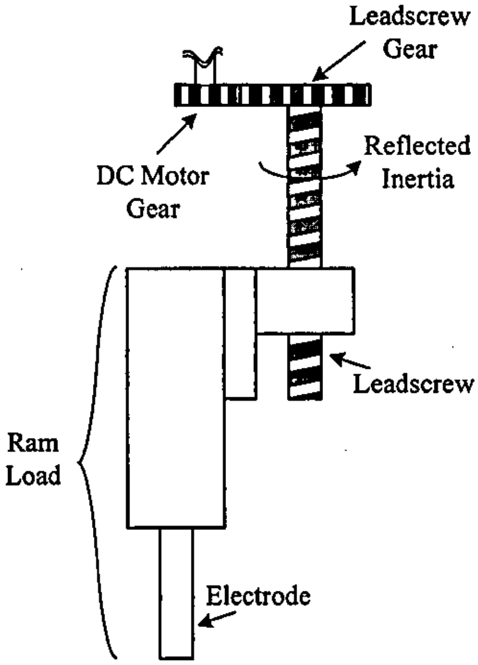


Figure 2.14 Components of mechanical EDM system

Calculations of the mechanical system's inertias were carried out according to Newton's Second Law of Motion [47], [56]. Component efficiencies, conversion factors and material densities were taken from a data sheet supplied in reference [57]. Due to the complexity and number of calculations performed, the complete mechanical system's inertias calculation is documented in Appendix 3.

2.2.2.2 Load torque T_l and frictional loss coefficient K_{fl} of DC motor with load

Open loop tests are carried out by applying a range of armature voltages to the motor and the results in terms of tacho voltage and current are measured. From the results shown in Table 2.7, load torque and viscous friction losses were determined. From these, a viscous friction coefficient was estimated.

Table 2.7 Determination of load torque and viscous friction coefficient

Measured during open loop test							
Applied Armature voltages, V_a (V)	2.25	3	5	10	15	20	24
Measured current, I_a (A)	0.89	0.91	0.98	1.1	1.22	1.31	1.36
Measured tacho voltage, V_{tac} (V)	0.23	0.53	1.48	3.87	6.24	8.67	10.57
Equivalent angular velocity (rad/s)	3.44	7.92	22.14	57.89	93.35	129.7	158.1
Motor torque T_m (Nm)	0.121	0.123	0.133	0.149	0.165	0.177	0.184
Load torque T_l at 'break away' (Nm)	0.121						
Viscous friction torque loss (Nm)	-	0.0027	0.0122	0.0285	0.0447	0.0569	0.0637
K_{fl} (Nm.s/rad)	40.62×10^{-5}						

It should be noted that the calculations from section 2.2.2.1 result in a load torque reflected back to the motor, which is only 2.436×10^{-3}Nm greater than motor torque produced at the 'break away' condition, according to Table 2.7. The Simulink model to represent the DC motor with mechanical load is shown in Figure 2.15.

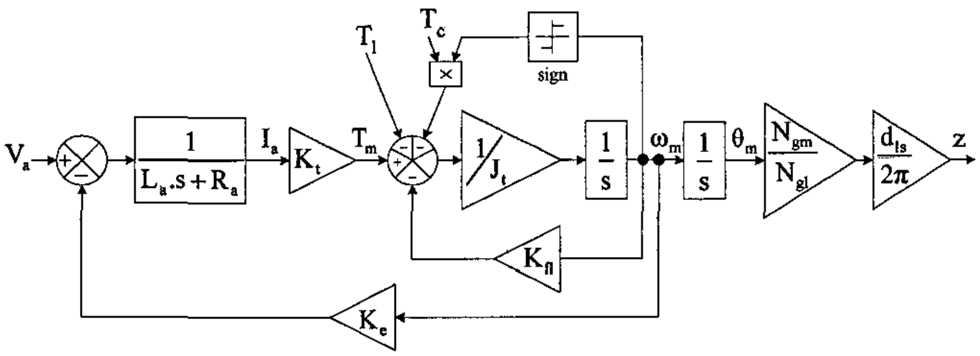


Figure 2.15 Simulink nonlinear model of a DC motor with load

The model has been extended to include the electrode position, which is required for later simulations. The gain block N_{gm}/N_{gl} represents the gear reduction. The final gain block represents the conversion of angular position of the leadscrew to linear position of the electrode z. The experimental and simulation results are depicted in Table 2.8. The graphical results can be seen in Figure 2.16. The results show close agreement.

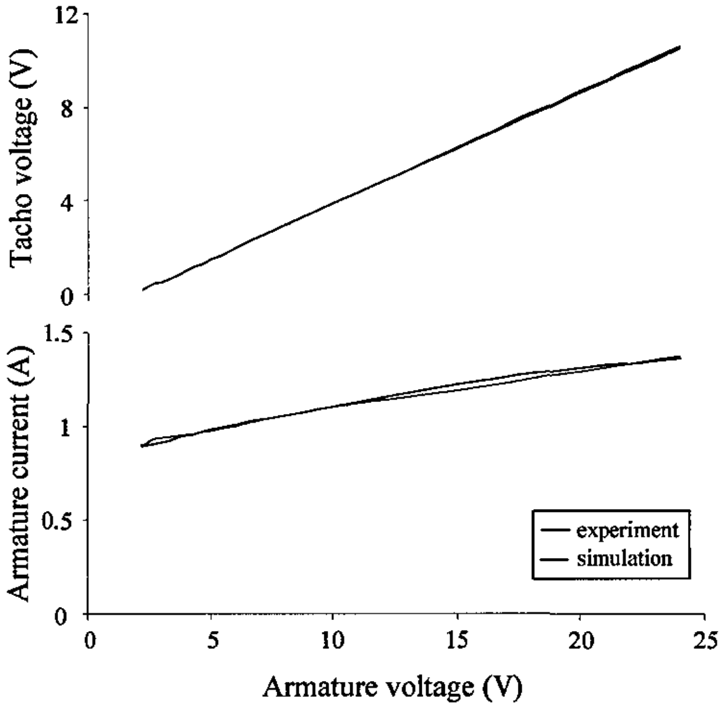


Figure 2.16 Open loop test results of a nonlinear DC motor with load

Table 2.8 Experimental and simulation open loop test results

Experimental results							
Applied Armature voltages, V_a (V)	2.25	3	5	10	15	20	24
Measured current, I_a (A)	0.89	0.91	0.98	1.1	1.22	1.31	1.36
Measured tacho voltage, V_{tac} (V)	0.23	0.53	1.48	3.87	6.24	8.67	10.57
Simulation results							
Measured current, I_a (A)	0.9	0.93	0.97	1.1	1.19	1.29	1.37
Measured tacho voltage, V_{tac} (V)	0.22	0.55	1.5	3.86	6.22	8.58	10.47

2.3 MULTI-LOOP CONTROLLER DESIGN

The proposed control scheme uses a multi-loop controller consisting of a current loop controller, a velocity loop controller and an average gap voltage loop controller as shown in block diagram form of Figure 2.17.

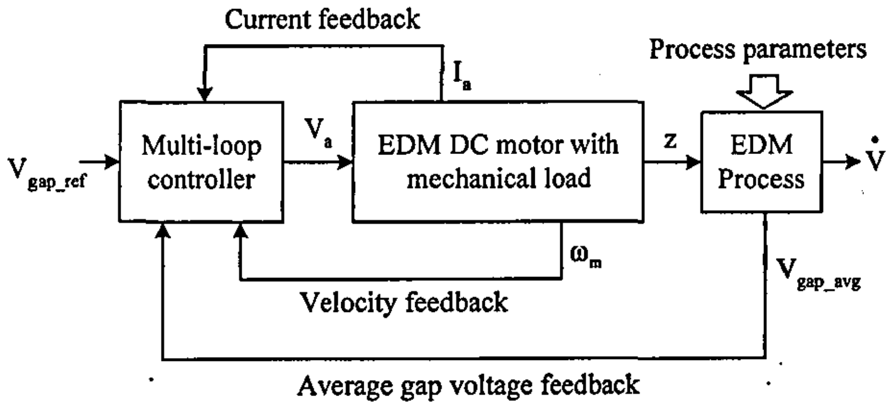


Figure 2.17 Block diagram of multi-loop control system

The current and velocity loops are inner loops with the current loop being the innermost loop. The average gap voltage loop is the outer loop and is used to indirectly control the position z of the electrode. The system controllers are designed via a frequency-domain and a time-domain simulation analysis using the Matlab/Simulink. The frequency response analysis provides information on the system stability and transient and steady state responses [47][58-60]. The time domain simulation allows the response to be analysed in term of steady-state error, rise time and overshoot percentage.

2.3.1 Current loop controller design

The current loop, also known as the torque loop, is introduced to control the motor torque so that the motor reacts quickly to overcome the system load torque. A Proportional-Integral (PI) controller shown in equation (2.32) is placed in cascade with the servomotor as shown in Figure 2.18.

$$PI_c = K_{pc} + \frac{K_{ic}}{s} \qquad (2.32)$$

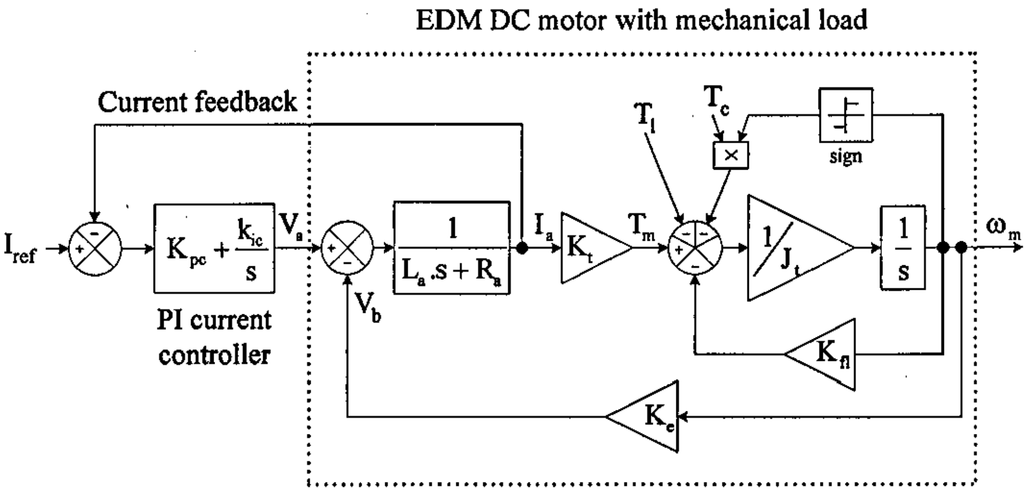


Figure 2.18 Current loop control

The current loop transfer function PI_c can be converted to a pole-zero equation to enable design parameters to be easily defined, and the resulting diagram is shown in Figure 2.19 [61].

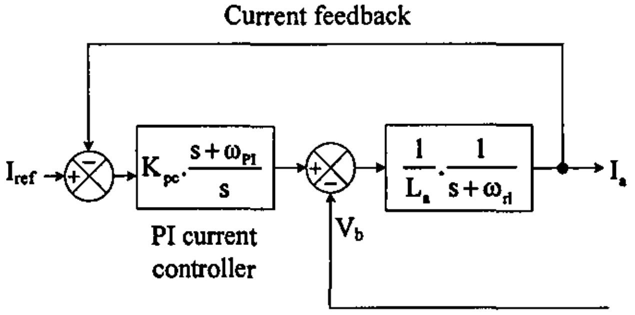

Current feedback
I_{ref}
+
−
$K_{pc}.\frac{s+\omega_{PI}}{s}$
PI current controller
+
−
V_b
$\frac{1}{L_a}.\frac{1}{s+\omega_{rl}}$
I_a

Figure 2.19 Pole-zero current loop diagram

The pole-zero controller representation has proportional gain K_{pc}, one zero and one pole at the origin. As shown in Figure 2.19, ω_{PI_c} and ω_{rl} are defined as follows:

$$\omega_{PI}(rad/s) = \frac{K_{ic}}{K_{pc}} \tag{2.33}$$

$$\omega_{rl}(rad/s) = \frac{R_a}{L_a} \tag{2.34}$$

By using pole-zero cancellation, ω_{PI} is made equal to ω_{rl}. K_{ic} is then equal to $K_{pc} \times \frac{R_a}{L_a}$. Therefore, the response of the system can be obtained for any chosen value of K_{pc}. Simulation tests using Figure 2.18 were performed and the value of K_{pc} of 12 was chosen as it gave fast transient response with minimal current overshoot. The resulting open loop and closed loop frequency response characteristics are shown in Figure 2.20 and Figure 2.21 respectively.

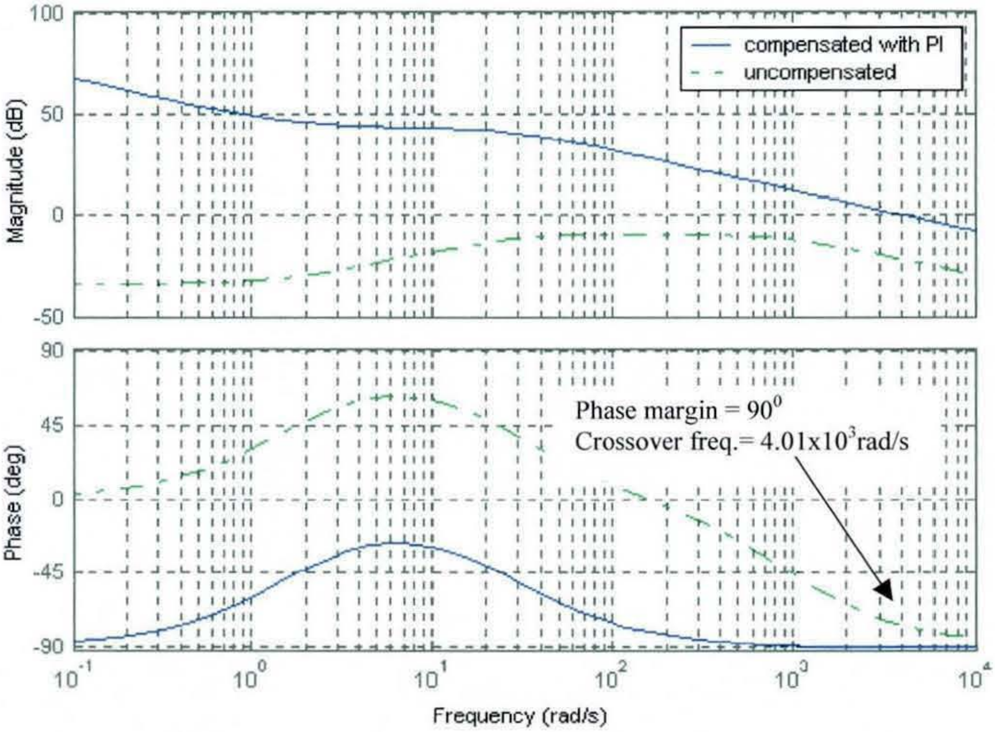


Figure 2.20 Open loop frequency response characteristics of current loop

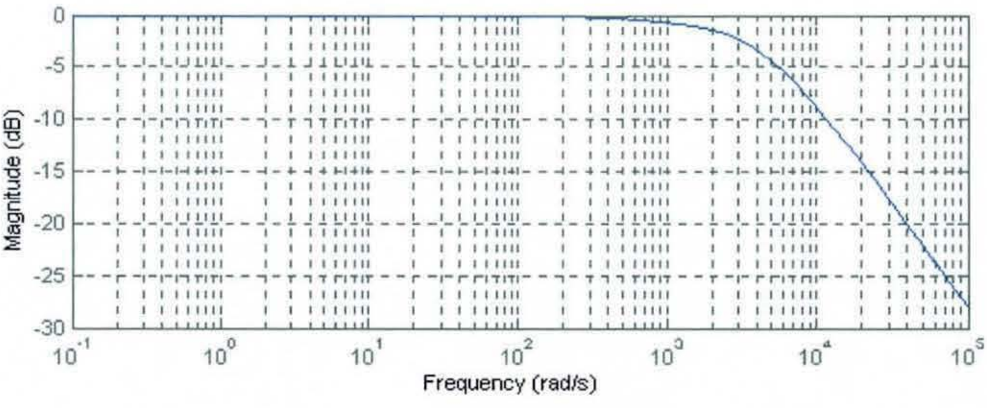


Figure 2.21 Closed-loop frequency response characteristics of compensated current loop

In Figure 2.20, the frequency response of the compensated current loop shows a large improvement compared to the uncompensated current loop. The DC gain is at 67.5dB and the low frequency range is adequately high to retain the steady-state accuracy. The gain crossover frequency is increased to 4.01 x 10^3rad/s. This indicates that the current loop has a fast response to ensure the motor torque rises quickly to overcome the load torque of the servo system. The phase margin is measured at 90^0, which is sufficient to maintain the required relative stability of the servo system. In Figure 2.21, the closed-loop bandwidth at -3dB is found to be 3.94 x 10^3rad/s. The result shows that the closed-loop bandwidth is large enough to ensure the fast time response required.

2.3.2 Velocity loop controller design

The velocity or speed loop controller is designed as the second stage of the multi-loop control of the EDM system. Figure 2.22 shows the block diagram of the EDM servo motor control consisting of the inner current loop and the velocity loop.

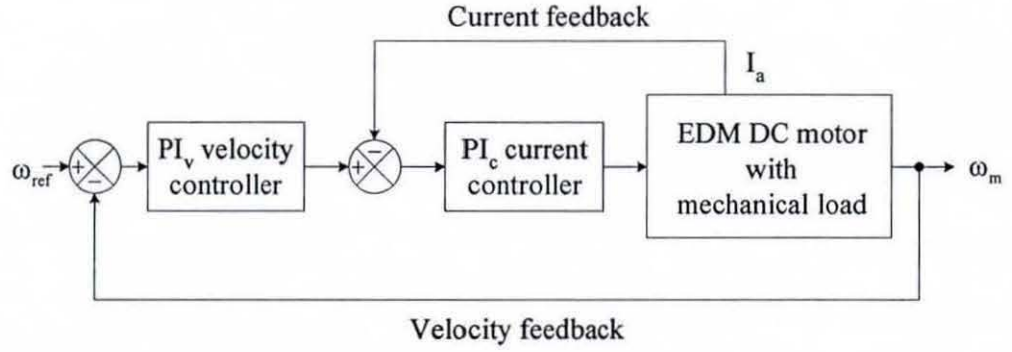

Current feedback
I_a
ω_{ref}
+
−
PI_v velocity controller
+
−
PI_c current controller
EDM DC motor with mechanical load
ω_m
Velocity feedback

Figure 2.22 Speed loop control

Like the current loop, the velocity loop also uses a PI controller of the form:

$$PI_v = K_{pv} + \frac{K_{iv}}{s} \tag{2.35}$$

Initially an open-loop simulation test was performed with P_{iv} set equal to 1 in order to obtain basic information about the frequency response performance of the velocity loop. From the simulation, a gain crossover frequency of 526rad/s was obtained and this is insufficient to ensure a fast transient response. To obtain a higher gain crossover frequency, the proportional gain of the velocity controller K_{pv} was set to 5 and this results in a gain crossover frequency of 2.31 x 10^3rad/s, giving a much faster transient response. The controller is further tuned by adding an integral gain K_{iv} of 111. The PI controller increases the open loop DC gain and the low frequency range, thus better steady state accuracy is achieved. The phase margin of 59.6^0 is sufficient to retain the required relative stability. The resulting open-loop frequency response characteristic is shown in Figure 2.23.

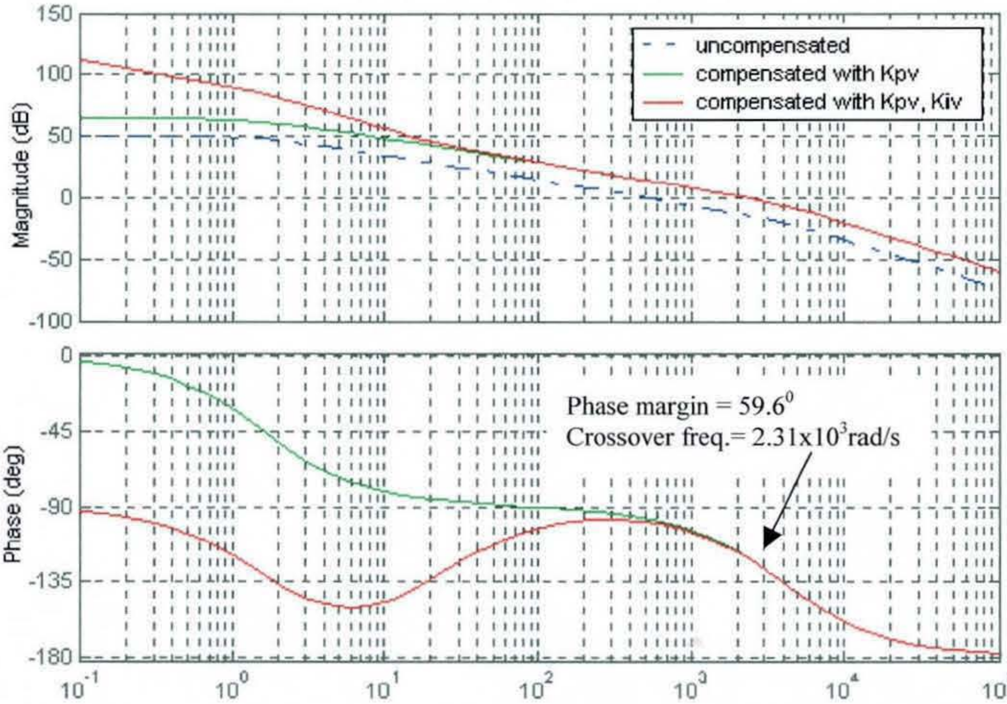


Figure 2.23 Open loop frequency response characteristics of velocity loop

In the close-loop frequency response as shown in Figure 2.24, a small resonant peak of 0.333dB exists at the resonant frequency but the system stability is maintained. It should be noted that a large excessive resonant peak indicates poor system stability. Figure 2.24 also shows a bandwidth of 3.65 x 10^3rad/s indicating a fast transient response. The bandwidth of the system can be extended further, by increasing the K_{pv} value. However, higher bandwidth could pick up undesirable high-frequency noise and affect control performance.

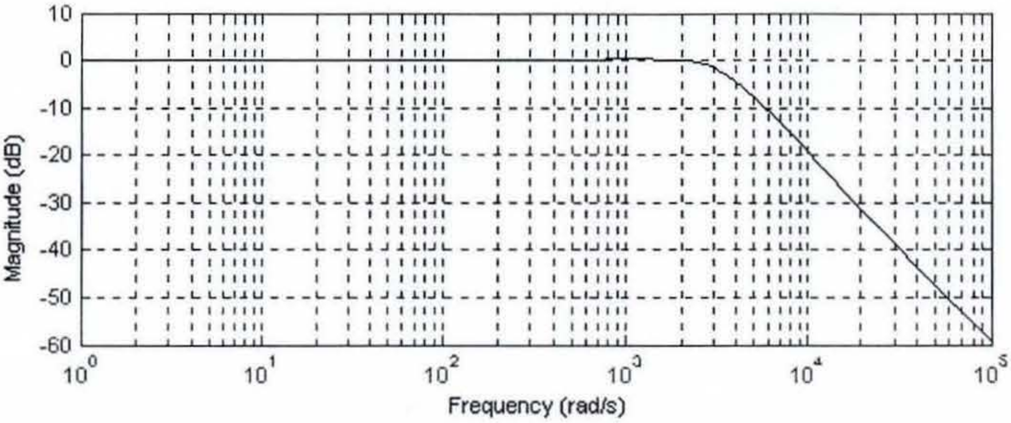


Figure 2.24 Closed-loop frequency response characteristic of compensated velocity loop

2.3.3 Anti-windup scheme due to saturation limit

In practical implementation, the armature voltage of the EDM servomotor is limited to $\pm 24V$ and the armature current to $\pm 4A$. Therefore, saturation limits are used within the Simulink model to limit the applied voltage and armature current to the above values as shown in the block diagram of Figure 2.25.

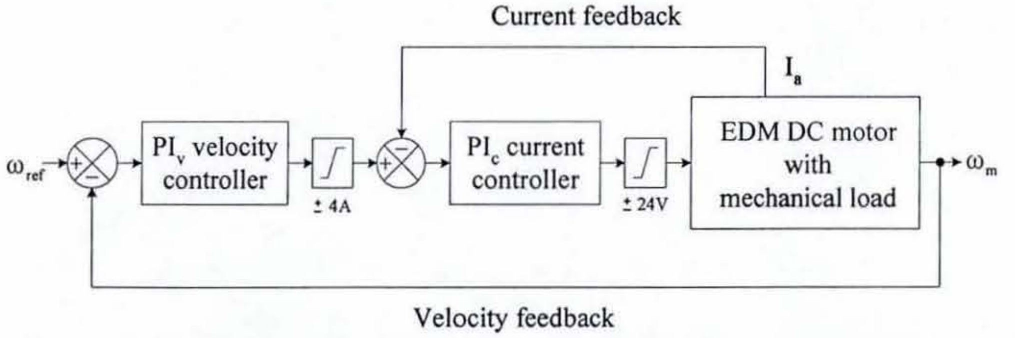

Current feedback
I_a
ω_{ref}
+
−
PI_v velocity controller
± 4A
+
−
PI_c current controller
± 24V
EDM DC motor with mechanical load
ω_m
Velocity feedback

Figure 2.25 Simulink model with saturation limit

Figure 2.26 shows the velocity response to a 100rad/s step input. The response shows a large overshoot for a long period of time that may lead to a deterioration of response and even instability.

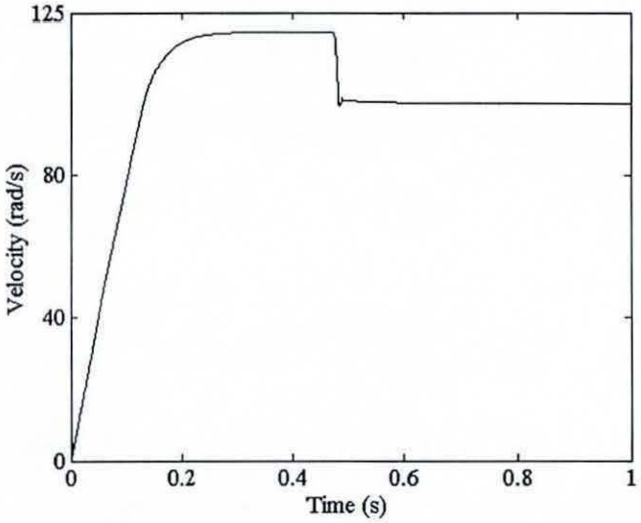


Figure 2.26 Velocity step response of Simulink model incorporating saturation limit

The cause of the problem lies on the applied voltage and armature current saturation limit and the integral part of the PI control loop. The error between the demand/reference and the feedback loop is multiplied with the PI gain and the error is integrated over a longer duration by the integral mode. The integrator output continues to increase (wind-up) resulting in an increased to the drive signal until the integrator output begins to reduce (wind-down). To correct the wind-up problem, an anti wind-up scheme within the PI controllers was applied [62,63].

The block diagram of the Simulink model incorporating the saturation limits and the anti wind-up scheme within the PI controller is shown in Figure 2.27. The velocity response to a 100rad/s step input as illustrated in Figure 2.28 shows a response without overshoot and zero steady state error.

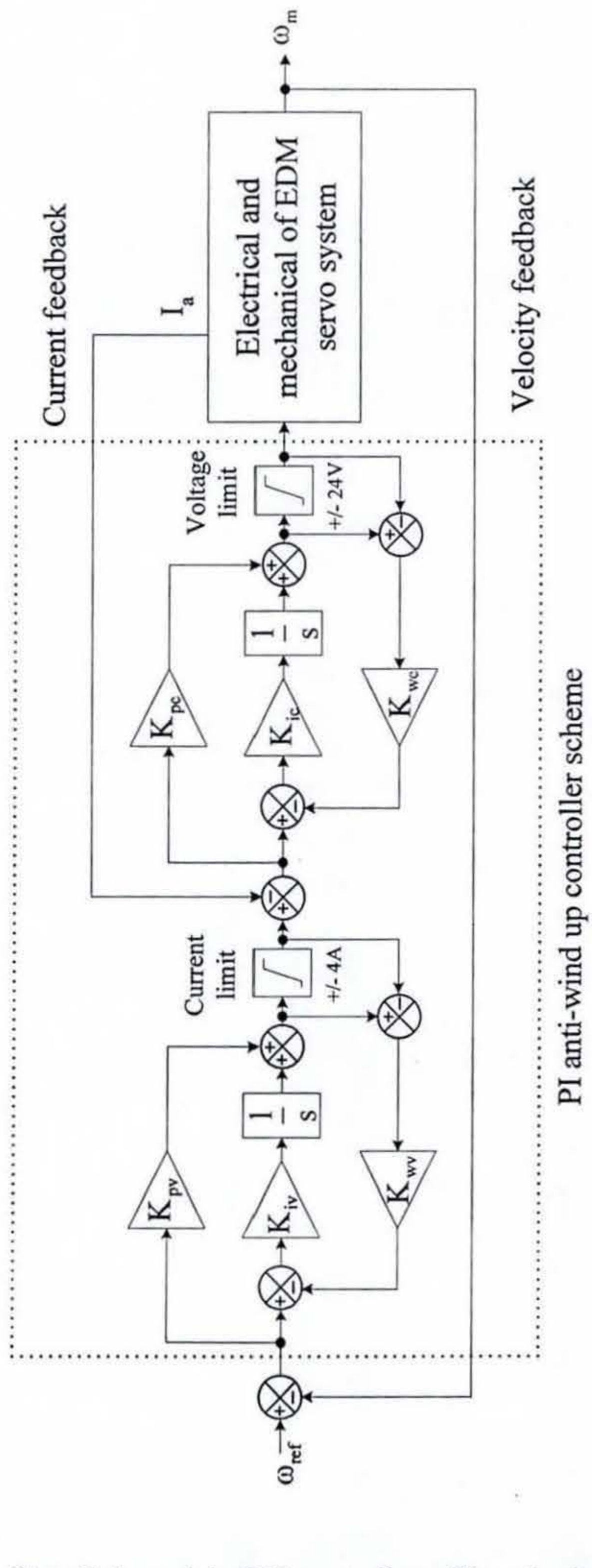


Figure 2.27 Simulink model of PI controller with anti wind-up scheme

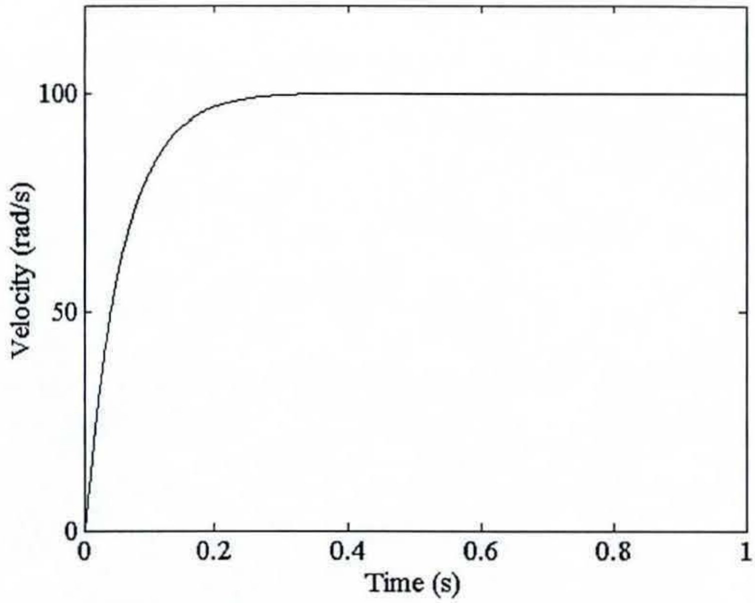


Figure 2.28 Velocity response using anti wind-up scheme

2.3.4 Average gap voltage loop controller design

A general servo position control system is a point-to-point position control system where the actuator is coordinated to yield a desired motion for a specific end [64]. Generally the desired position is known and can be measured directly. Although position control in the EDM system is philosophically similar due to the fact that the position is controlled to maintain a certain gap width, the gap width cannot however be measured directly. In the EDM system, the average gap voltage is used as a feedback parameter as shown in Figure 2.29 in order to indirectly control the gap width.

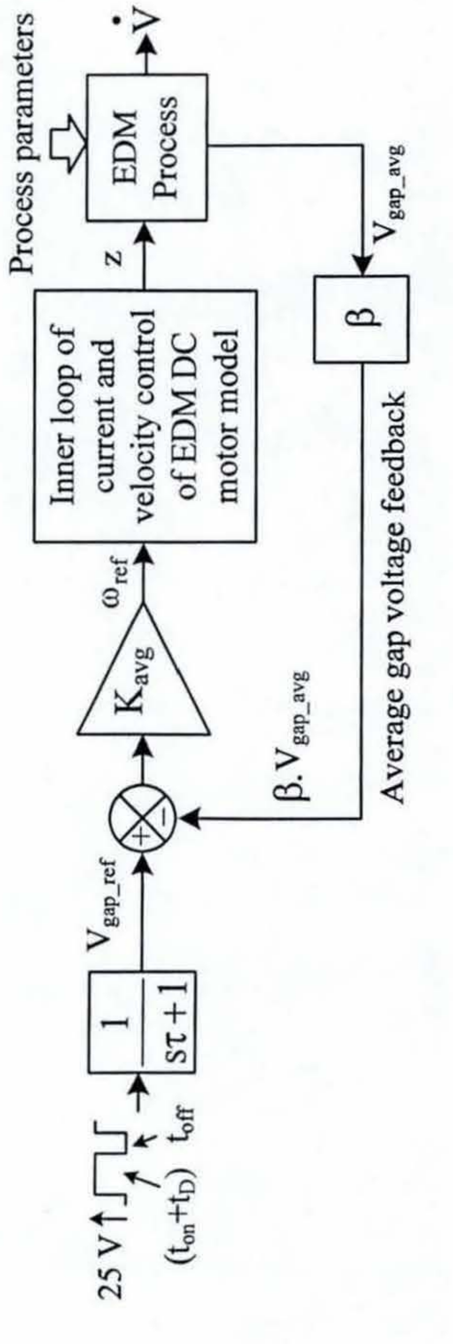


Figure 2.29 Block diagram of average gap voltage control

In positioning the electrode accurately, the average gap voltage controller is designed to meet certain response requirements such as no overshoot and zero steady state error. In order to design the controller, EDM process parameters from [14] are used in the simulation analysis. These parameters are defined in a Matlab m-file as follows I_{gap}=25A, V_{max}=160V, V_{arc}=25V, t_{on}=110μs, t_{off}=4.2μs, F_S=8.605 kHz, α =2 x 10^{-12} m^3/J and C=1.74.

In the Simulink model shown in Figure 2.29, the average gap voltage reference V_{gap_ref} is obtained by filtering a PWM waveform of amplitude V_{arc} and PWM duty ratio $(t_D+t_{on})/(t_D+t_{on}+t_{off})$. A portion of the average gap voltage $\beta.V_{gap\_avg}$ is used as the feedback quantity. β is under manual control in order to control the amount of the average gap voltage feedback to the controller. β is set to a value between 0 and 1. β effectively controls the value of t_D. A β value of one effectively gives t_D a zero value. A β value of zero results in open circuit gap operation (i.e. gap breakdown does not occur) where t_D occupies the whole of the on-time period, as can be inferred from Figure 1.7. The set value of β corresponds to a chosen ignition delay time t_D giving an appropriate gap width δ from the relationship given in (2.18). β is adjusted to give a time delay t_D of 2μs. In a time domain analysis, the proportional gain K_{avg} of the average gap voltage controller is tuned to 0.31.

The response of the average gap voltage control loop is shown in Figure 2.30 where the feedback signal follows the demand with no overshoot and where the steady state error tends to zero. Thus the proportional controller with K_{avg} =0.31 for the average gap voltage control loop is adequate to achieve a satisfactory performance. Further simulation results in Figure 2.31 show the response of the steady state values of electrode position z, workpiece surface position ξ, gap width δ and time delay t_D.

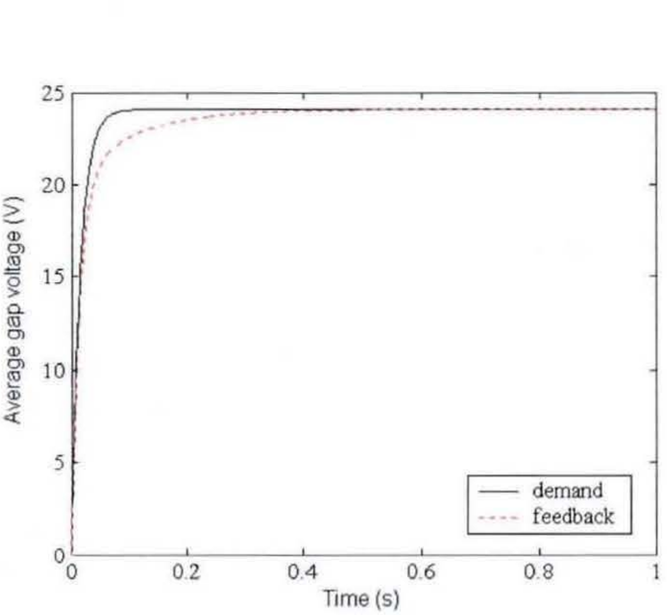


Figure 2.30 Response of average gap voltage

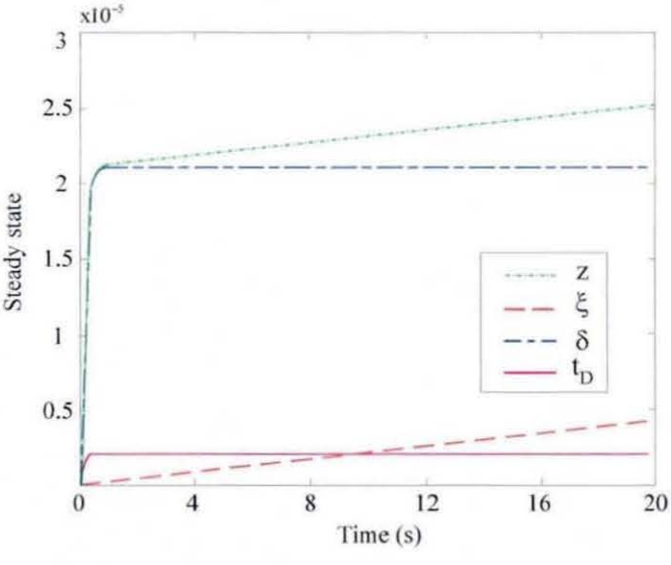


Figure 2.31 Steady state response of electrode position z, workpiece surface position ξ, gap width δ and time delay t_D

The figure shows that the electrode position z is continuously moving while the depth of the workpiece surface position ξ is simultaneously increasing. The gap width δ is maintained at 21.13μm which corresponds to the fixed control time delay t_D of 2μs.

2.4 SIMULATION RESULTS OF MATERIAL REMOVAL RATE

This section describes a series of Matlab/Simulink simulations of the complete EDM system shown in Figure 2.17 for different gap currents in order to obtain predicted of material removal rates. The gap currents are 4A, 6A, 8.5A, 12.5A, 18A, 25A, 36A and 50A. The material properties factor α is 2×10^{-12} m^3/J, the constant C is 1.74 and the open gap voltage is 160V [14].

For each value of gap current, simulations were run at different 'on' and 'off' times. The results are recorded as process 1, 2, etc. In simulation, the erosion rate is measured in m^3/s, thus a conversion factor of 60×10^9 is used to present the results in the more practical unit of mm^3/min. The simulation data and results are presented in tabular and graphical forms as shown in Table 2.9 and Figure 2.32 respectively.

Table 2.9 Simulation data and results of material removal rate

Process	t_{on} (μs)	t_{off} (μs)	F_s (kHz)	$\dot{V}$ (mm^3/min)							
				at I_{gap}=4A	at I_{gap}=6A	at I_{gap}=8.5A	at I_{gap}=12.5A	at I_{gap}=18A	at I_{gap}=25A	at I_{gap}=36A	at I_{gap}=50A
1	2	4	125	5.22	7.83	-	-	-	-	-	-
2	3	4	111.11	6.96	10.44	14.79	21.75	-	-	-	-
3	4	4	100	8.35	12.53	17.75	26.1	37.58	52.2	-	-
4	6	4	83.33	10.44	15.66	22.19	32.63	46.98	62.25	93.96	130.5
5	12	4	55.55	13.92	20.88	29.58	43.5	62.64	87	125.28	174
6	25	4	32.25	16.84	25.26	35.78	52.65	75.77	105.24	151.55	210.48
7	50	6	17.24	18	27	38.25	56.25	81	112.5	162	225
8	100	12	8.77	18.32	27.47	38.92	57.24	82.42	114.47	164.84	228.95
9	200	25	4.41	18.40	27.59	39.09	57.49	82.78	114.98	165.57	229.96
10	400	50	2.21	18.48	27.72	39.27	57.74	83.15	115.49	166.3	230.97

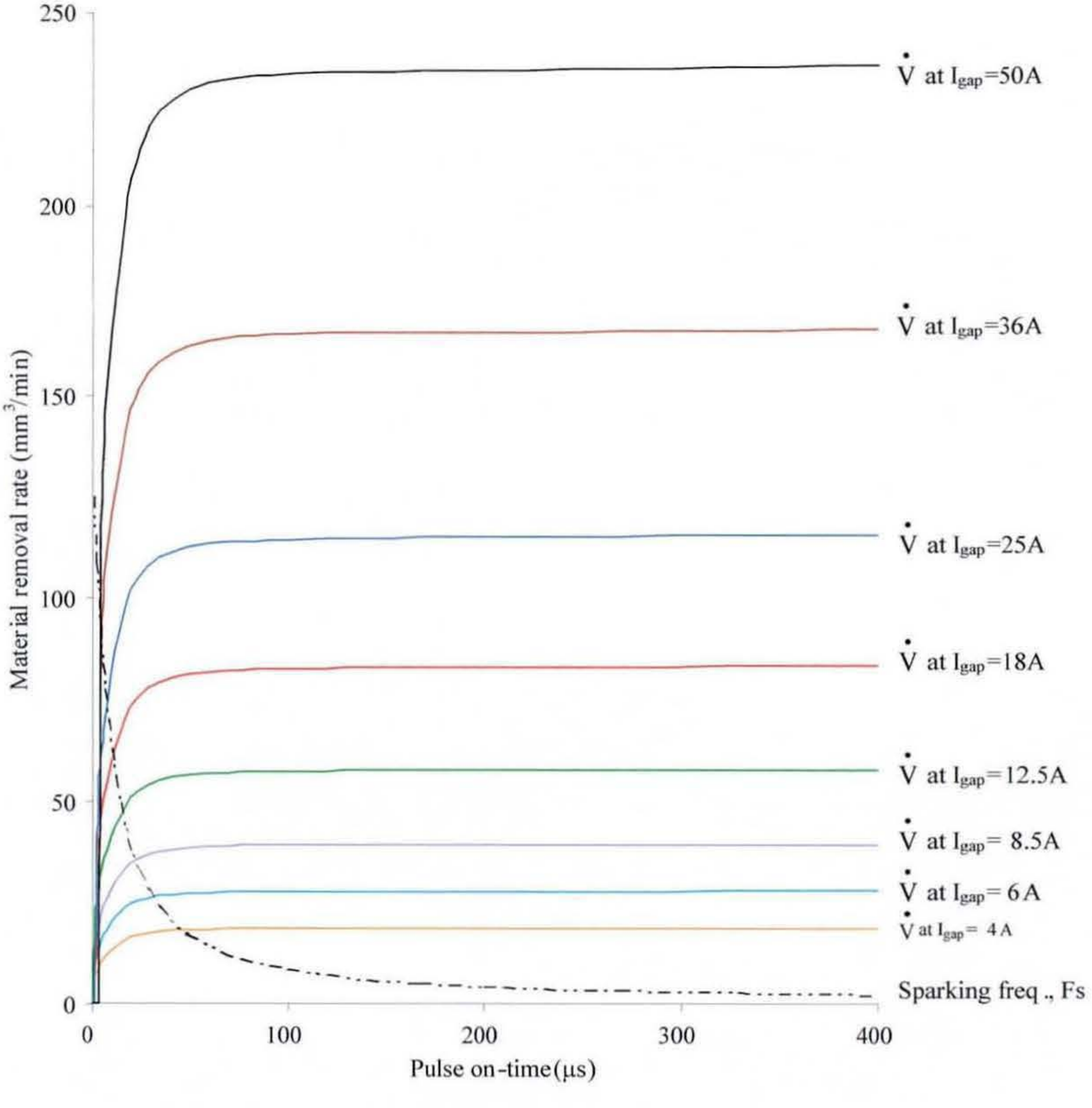


Figure 2.32 Material removal rate simulation results

Graphs of material removal rate and sparking frequency versus pulse on-time as depicted in Figures 2.32 show that erosion with a low current gives a low rate of material removal, while conversely a high current gives a high rate of material removal. Detail discussions of the material removal rate results are presented in Chapter 6 when a comparison is made between the simulation and the experimental results.

2.5 CONCLUSION

The development of an EDM system model has been detailed within this chapter. The EDM system model consists of two major parts, namely the EDM process model and the servo system model. The EDM process model was developed from sub-models such as the material removal rate model, the empirical breakdown model and the average gap voltage model. The material removal rate model was developed using a dimensional analysis technique. The empirical breakdown model on the other hand was identified from the experimental observations by previous researchers. The average gap voltage model was constructed and used to indirectly measure the gap width in term of voltage level. The model of the servo system was developed in Simulink using the Transfer Function approach. The DC motor and the EDM mechanical system parameters were identified from theory and from experiments and the validity of the model was verified through comparison analysis between experimental and simulation results. A suitable controller was designed using frequency and time domain analysis and applied to control the EDM servo system in a multi-loop controller. An anti-wind up scheme is also incorporated within the controller in order to overcome the complicated response due to saturation. The complete EDM system model consisting of the EDM process model and the servo system model were then used to produce the simulation results of material removal rate. A comprehensive explanation on modelling and simulation has been presented in Chapter 2. The following chapter is dedicated to the development and implementation of control system software using a DSP microcontroller.

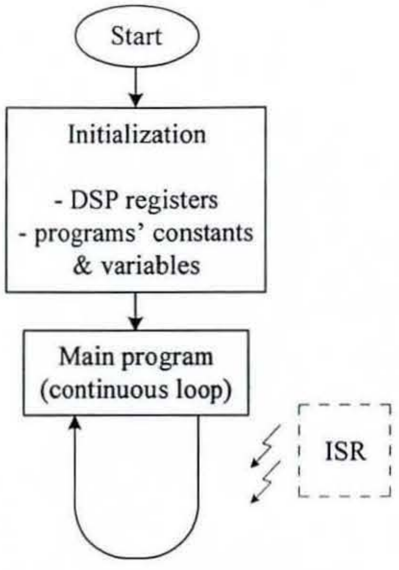
CHAPTER 3

DSP BASED CONTROL STRATEGY FOR AN EDM SYSTEM

A digital control strategy for the EDM system was developed and implemented within an eZdsp^{TM} LF2407 (eZdsp) microcontroller development board. The microcontroller uses a TMS320LF2407A fixed point Digital Signal Processor (DSP) for software implementation and various on-chip interface components [65,66]. Control software was designed using the structure programming methodology that combines a flowchart and program structure diagrams for clear description of the program code [58,67]. The code, written in C language, was compiled, debugged and then downloaded into the DSP on-chip Flash ROM using a dedicated C2000 Tools Code Composer for the eZdsp [68]. Explanations on the DSP peripherals using jargon words such as CMPRx, ACTRx, TxCNT etc. may require the reader to refer to reference [69].

3.1 CONTROL STRATEGY OF EDM SYSTEM

Flowchart 3.1 shows the overall program structure. At start, the program initializes DSP registers, various programs' constants and variables. After activating a real-time timer interrupt service routine ISR, it moves to a 'continuous loop' to execute the main program. The structure diagram of the main program is shown in Figure 3.1.


Start
Initialization
- DSP registers
- programs' constants & variables
Main program (continuous loop)
ISR

Flowchart 3.1 Overall program structure

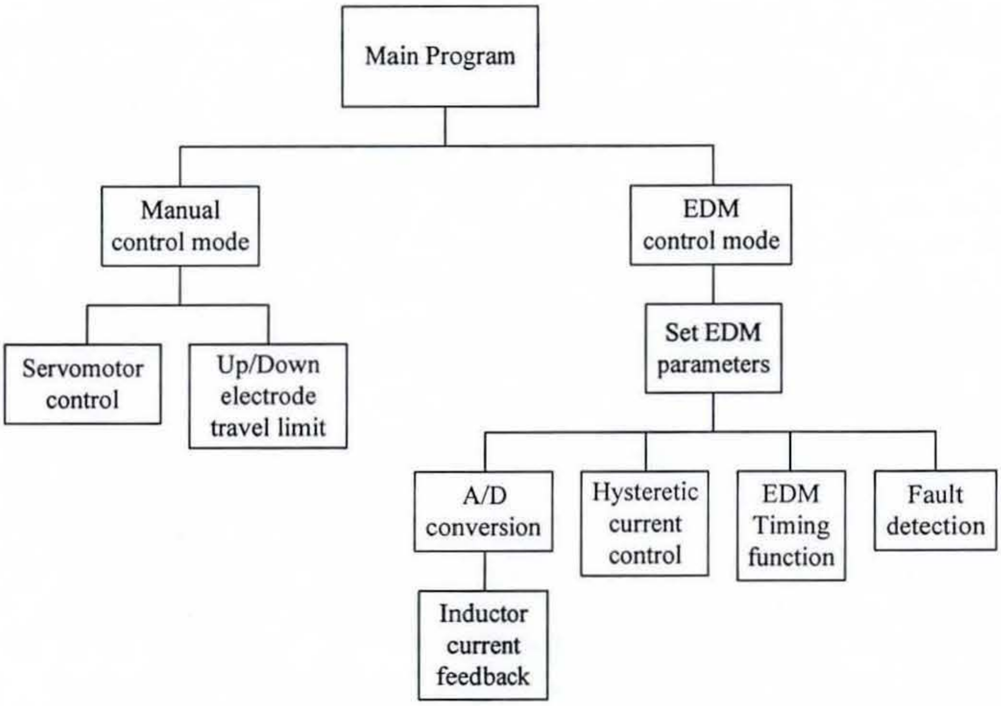
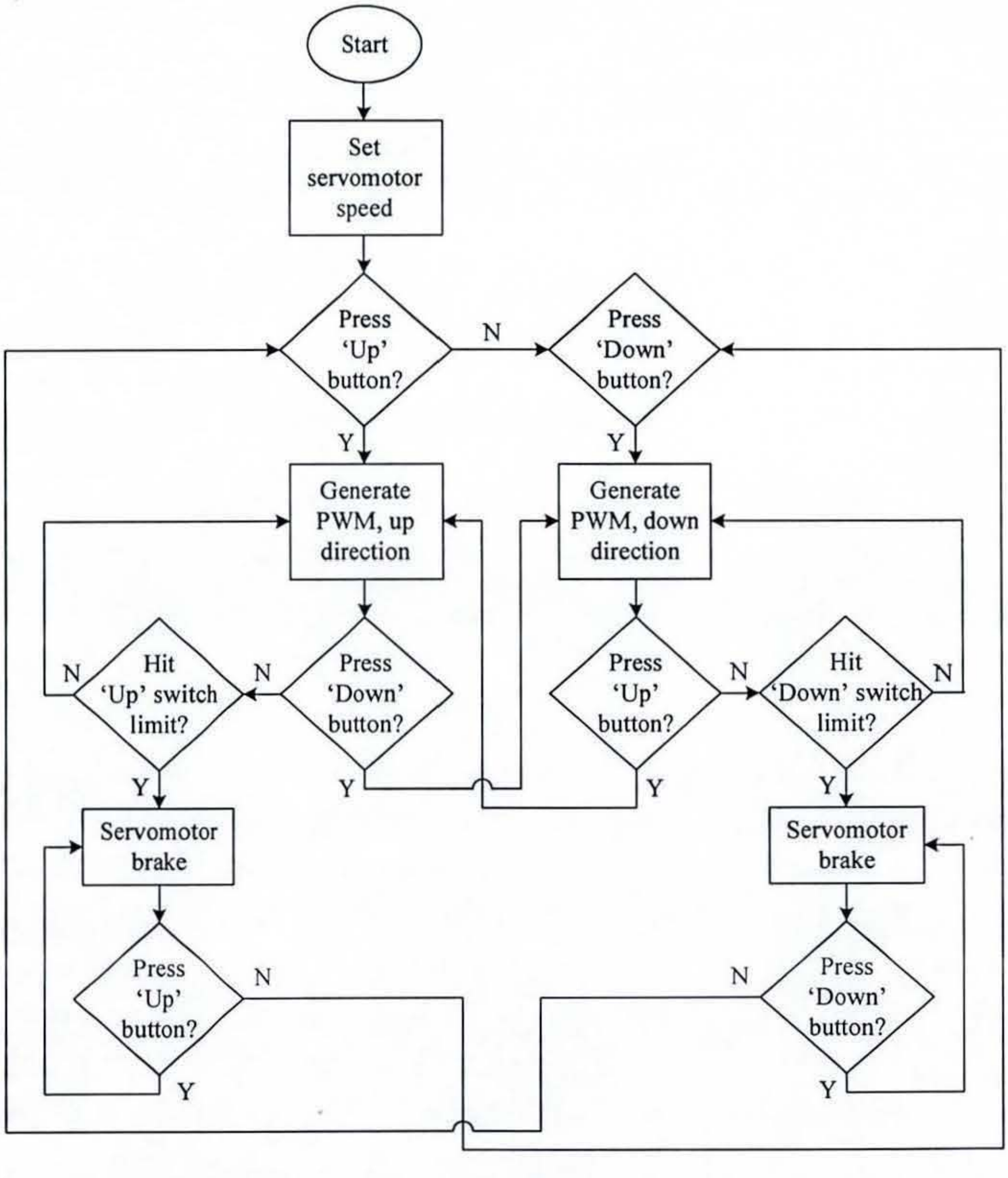


Figure 3.1 Block structure diagram of main program

In the continuous loop of the main program, the user selects either manual or EDM control modes of operation. In the manual control mode, the program executes the servomotor control software with controllable speed. The software code detects the up and down limits of the electrode position as shown in Flowchart 3.2.



Flowchart 3.2 Up/Down limit of electrode position

In the EDM control mode where the EDM process takes place, at first, the user sets the EDM parameters such as t_{on}, t_{off}, t_{up}, t_{down} and I_{gap}. The parameters are then saved but can be updated to new values. These parameters are pre-defined in order to execute the next program code. The ISR executes the servomotor control program code while the main program is running in the EDM control mode. The program structure diagram of the ISR is shown in Figure 3.2.

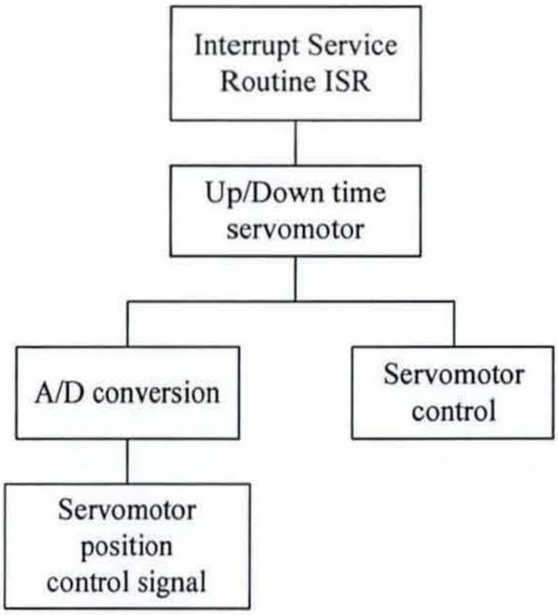

Interrupt Service Routine ISR
Up/Down time servomotor
A/D conversion
Servomotor control
Servomotor position control signal

Figure 3.2 Block structure diagram of the ISR

The input/output (I/O) data that is processed by the eZdsp are shown in Figure 3.3. V_c is the control signal for the servomotor PWM controller, V_L is the inductor control feedback signal, and $V_{N,man}$ is the servomotor manual speed control signal. The inductor current control signal is a PWM signal that controls a MOSFET switch Q_1, whilst the EDM timing control signal is a PWM signal that control a power MOSFET switch Q_2 as discussed in Chapter 4 (see Figure 4.8). The servomotor control signals PWM_1 and PWM_2 are used to control power MOSFET switches in an H-bridge configuration for the servomotor control (see Figure 4.6). The complete EDM program code with comments can be found in Appendix 4.

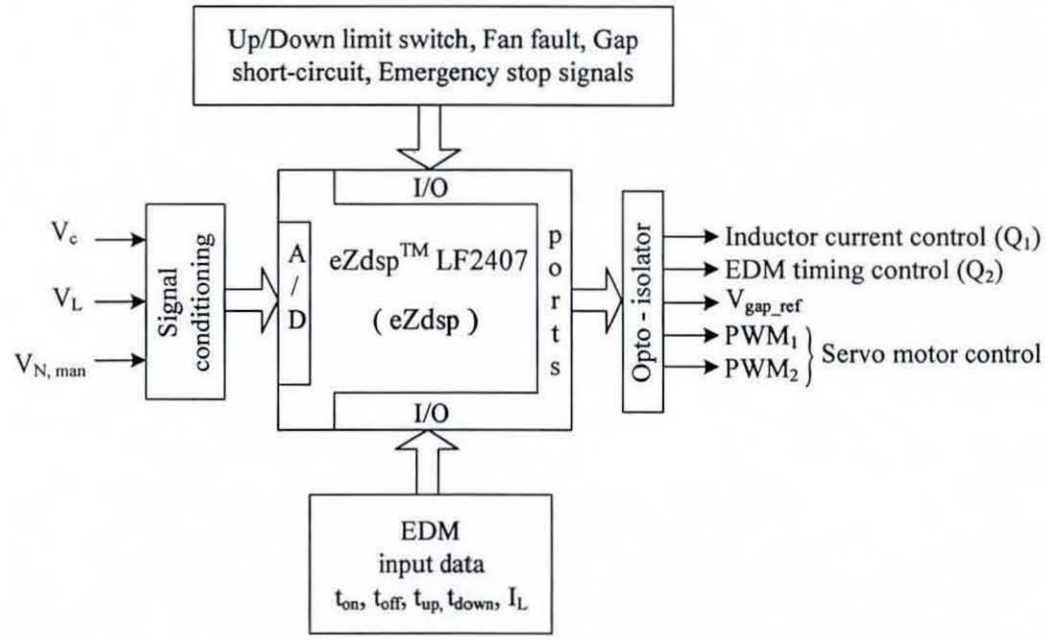

Up/Down limit switch, Fan fault, Gap short-circuit, Emergency stop signals
I/O
V_c
V_L
$V_{N, man}$
Signal conditioning
A / D
eZdsp™ LF2407
(eZdsp)
ports
Opto - isolator
Inductor current control (Q_1)
EDM timing control (Q_2)
V_{gap_ref}
PWM_1
PWM_2
Servo motor control
I/O
EDM
input data
t_{on}, t_{off}, t_{up}, t_{down}, I_L

Figure 3.3 Input and output signal processes by the eZdsp

3.1.1 Software design for the user-interface control device and EDM parameters selection

A user-interface control device is used to communicate between the user and the eZdsp in order to select/set the EDM parameters digitally. Eight-bit signal transfer is used to send the data to the eZdsp. The software part of the LCD does not require any programming except some LCD register initialization. The software implementation for the user-interface control device and EDM parameters selection is designed according to the Finite State

Machine FSM [70]. In this way, the number of switches on the user-interface device is minimized. The control states of the FSM are shown in Figure 3.4.

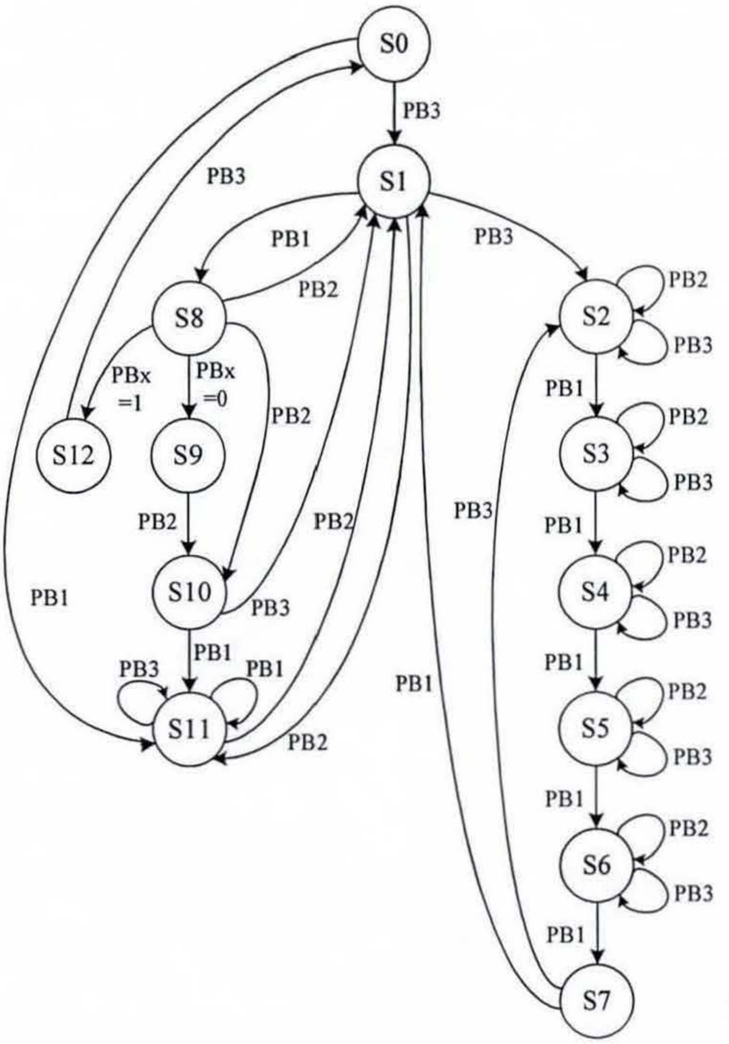
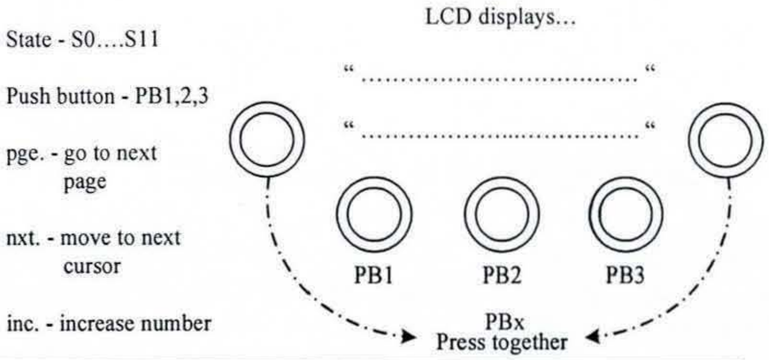


Figure 3.4 Control state for the user-interface device and EDM parameters selection

For every next state, the user-interface device displays the information and instruction on the LCD screen as shown in Figure 3.5.



State	LCD displays...
S0	EDM CONTROLLER Manual Menu
S1	Main Menu Run Manual Set up
S2	On time: _ _ _ _ µs pge. nxt. inc.
S3	Off time: _ _ _ _ µs pge. nxt. inc.
S4	Gap current: _ _ A pge. nxt. inc.
S5	Up time: _ _ . _ s pge. nxt. inc.
S6	Down time: _ _ . _ s pge. nxt. inc.
S7	Save setting? yes no
S8	← Press to start → stop
S9	System running stop
S10	System stop manual menu
S11	Manual control up menu down
S12	System stop "FAULTY ctrl"

Figure 3.5 LCD display information for state selection

There are twelve programmed states (S0 to S12) and screen displays. At S0, the first line of the LCD screen shows the display information of "EDM CONTROLLER" with the instruction "Manual Menu". Pressing PushButton1 PB1 leads to the manual control of the servomotor in state S11 as explained in section 3.1. Pressing PushButton3 PB3, leads to the main menu selection, whether to run/activate the EDM process or control the servomotor manually or setting up the process parameters. S2 to S6 allows the user to select the process parameters.

For example, state 2 displays the "On time: _ _ _ _ μs" information on the LCD screen. The LCD screen also displays the instruction of "pg. nxt. inc." on the second line of the screen. This represents which of the push button is active. Selecting the "pg." (by pressing the assign button PB1) brings the next page of display. In other word, the program code executes the next state program, displaying other information and instruction. The "nxt." instruction places the cursor (blink cursor) to the assign position. Then, pressing the "inc." instruction button, the number 0 to 9 appears on the cursor. The process parameters are then saved at S7.

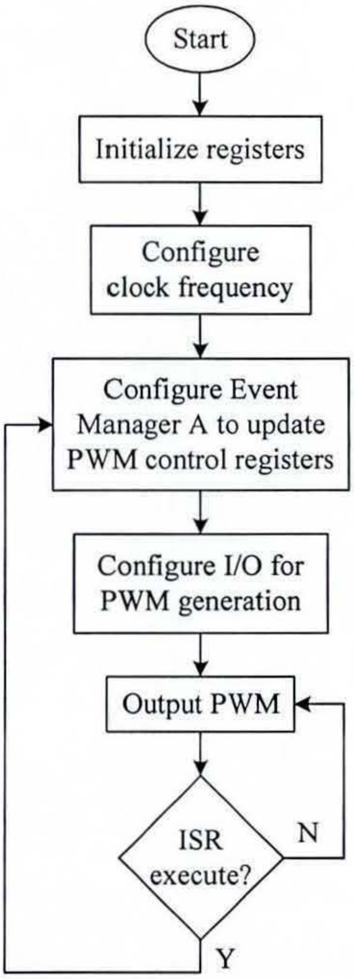
In state S1, pressing PB1 leads to state S8 where the user has to press two buttons together (PBx) to run/activate the EDM process. However the EDM process activation is subject to a faultless condition as explained in section 3.1.5. If any faults occur, the state goes to S12 or else to S9. At S9, the LCD screen displays "System running" indicating that the EDM process is being activated. Subroutine functions were used in the main program as required since some of the displays share the same characters and words.

3.1.2 Software design for EDM servo system control

The signals from motor current, velocity and gap width feedback are to be processed by the analogue EDM signal processing circuit (discuss in Chapter 4). The output of this circuit V_c

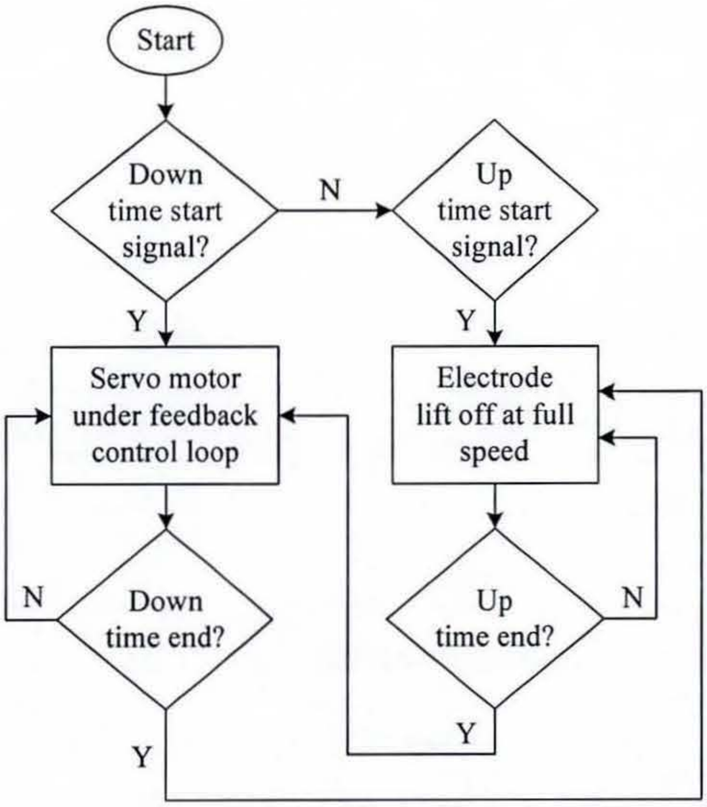
is the control signal for the digital servomotor PWM controller. Only the analogue control signal V_c is digitised in order to obtain the highest possible resolution with the 10-bit eZdsp's A/D module of the eZdsp microcontroller.

Program code was developed to digitize the output signal V_c of the analog controller. The digitised signal is then used to generate the PWM waveform. The PWM waveform is a 20kHz Symmetric Fixed Frequency type that is generated by initializing and configuring specific DSP registers and modules respectively as shown in Flowchart 3.3 [71].


Start
Initialize registers
Configure clock frequency
Configure Event Manager A to update PWM control registers
Configure I/O for PWM generation
Output PWM
ISR execute?
N
Y

Flowchart 3.3 Configuring eZdsp modules to generate the PWM waveform

The control registers of Event Manager A that changed the PWM duty cycle are updated every time the ISR executes within the main program. The servomotor is under the feedback control loop mode within the down time t_{down} period (EDM machining time), while in the up time t_{up} period the electrode lifts off at full speed allowing gap debris to be flushed away. The strategy for the Up/Down time control is shown in Flowchart 3.4.



Flowchart 3.4 Up/Down time control strategy

3.1.3 Software design for EDM timing control

For EDM timing control, the parameters t_{on} and t_{off} control the pulse duration of switch Q_2 (see Figure 4.8). In practice, Q_2 is turned 'on' to divert the inductor current from the gap during the pause off-time t_{off}. During the pulse on-time t_{on}, Q_2 is turn 'off' and inductor current flows into the gap during gap breakdown. The relation between EDM control and the Q_2 control is shown in Figure 3.6.

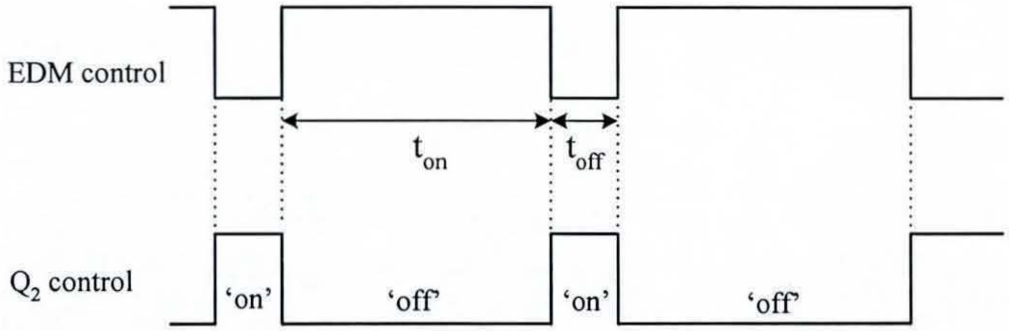
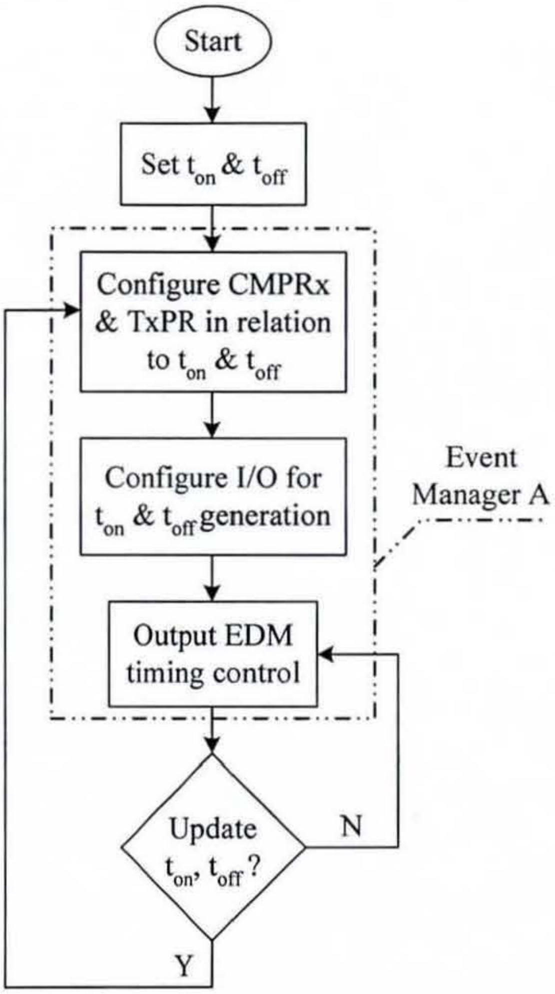


Figure 3.6 Relation between EDM control pulse duration and Q_2 control switch

The method of generating the t_{on} and t_{off} times for EDM timing control is similar to that explained in section 3.1.2 except that the frequency varies according to the summed values of t_{on} and t_{off}. A control strategy for EDM timing control is depicted in Flowchart 3.5. By knowing the t_{on} and t_{off} values as pre-defined by the user, the frequency of the waveform can be calculated internally. The waveform is generated internally by configuring various associate registers of Event Manager A. The frequency and the duty cycle of the waveform rely on the register values of CMPRx and TxPR. These register values vary according to the selection of t_{on} and t_{off}. Within Event Manager A, the I/O port must be correctly configured to choose the PWM output since most of the I/O and PWM modules sharing the same port.


Start
Set t_{on} & t_{off}
Configure CMPRx & TxPR in relation to t_{on} & t_{off}
Configure I/O for t_{on} & t_{off} generation
Output EDM timing control
Update t_{on}, t_{off}?
N
Y
Event Manager A

Flowchart 3.5 EDM timing control strategy

3.1.4 Software design for EDM hysteretic current control

The inductor current I_L is controlled according to the hysteretic current control technique. The control scheme switches Q_1 'on' and 'off' as necessary to maintain the chosen current level. An illustration of hysteretic current control is shown in Figure 3.7.

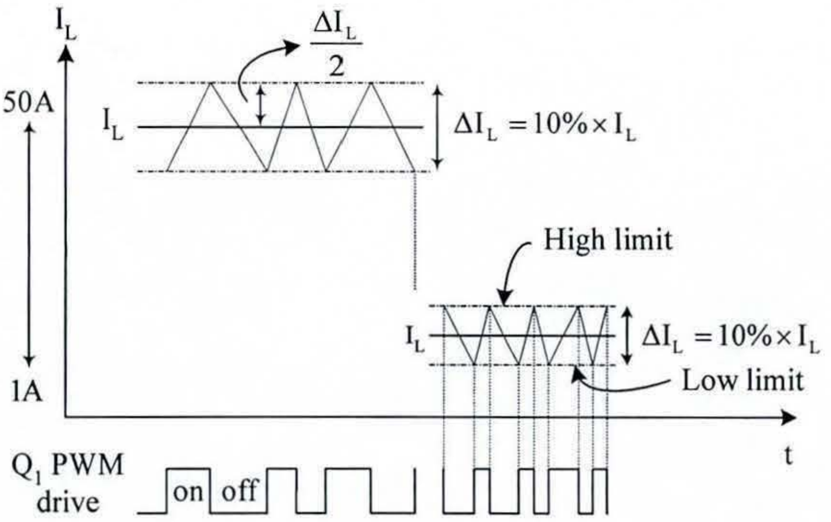


Figure 3.7 Hysteretic current control scheme

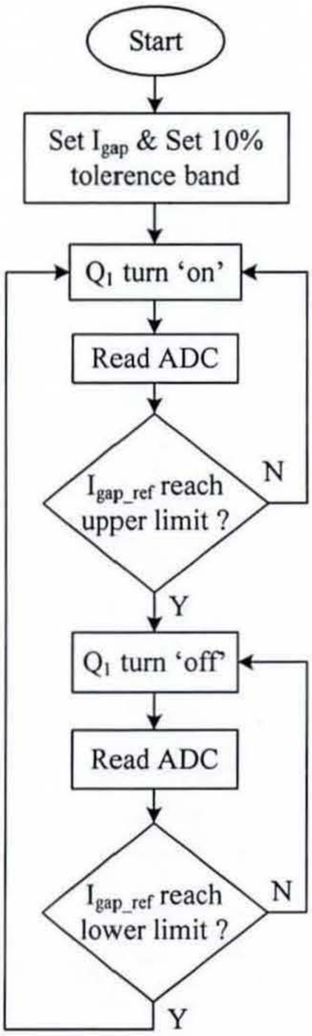
The inductor current increases when Q_1 is on and decreases when Q_1 is off. The hysteretic controller switches Q_1 'on' whenever the inductor current reaches the lower limit according to equation (3.1).

$$I_{L\text{ low limit}} = I_L - 10\% \times \left(I_L / 2 \right) \tag{3.1}$$

The controller switches Q_1 'off' whenever the inductor current reaches the upper limit according to equation (3.2).

$$I_{L\text{ high limit}} = I_L + 10\% \times \left(I_L / 2 \right) \tag{3.2}$$

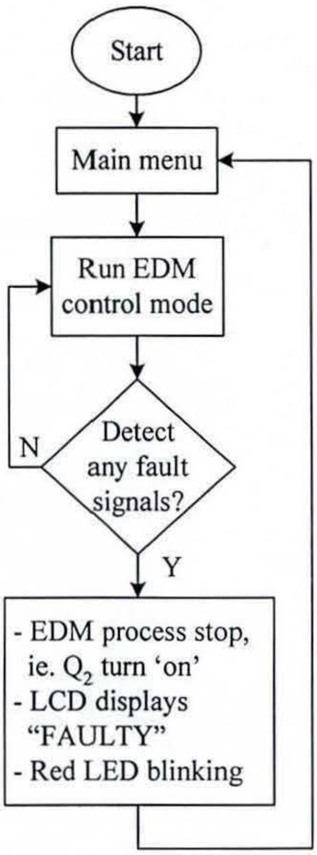
A strategy control to implement hysteretic current control is shown in Flowchart 3.6. The gap current I_{gap} is selected digitally via the user-interface device. The gap current variation is set to a 10% $_{\text{(peak-to-peak)}}$ tolerance band. The processor reads the gap current feedback A/D module and evaluates the feedback according to the 10% tolerance band as defined by equations (3.1) and (3.2). Then logic '0' or '1' is send to the selected I/O port (see Figure 4.4) to activate the Q_1 switch.


Start
Set I_{gap} & Set 10% tolerence band
Q_1 turn 'on'
Read ADC
I_{gap_ref} reach upper limit ?
N
Y
Q_1 turn 'off'
Read ADC
I_{gap_ref} reach lower limit ?
N
Y

Flowchart 3.6 Hysteretic current control strategy

3.1.5 Software design for EDM fault-detection

In an existing analogue EDM circuit controller, three fault-detection signals output were available. These signals recognise the emergency push button, the fan fault and the electrode/servo contact. These signals were used in a control strategy for EDM fault-detection as shown in Flowchart 3.7.



Flowchart 3.7 EDM fault detection

If any of the fault signals are triggered, the program code halts the EDM process i.e Q_2 switch is turn 'on' and the system exits the EDM control mode. The LCD's screen on the user-interface control device displays a "FAULTY" message with a red LED indication.

After clearing the fault, the user can re-start the process or go to the main menu for changing/up-dating parameters.

3.2 CONCLUSION

The EDM system is controlled digitally using the eZdsp microcontroller development board. The microcontroller uses a DSP TMS320LF2407A fixed point processor as a platform to develop the digital control scheme for the EDM system. Flowcharts and program structure diagrams were employed as a guideline to design and implement the program code (software). A user-interface device was used to communicate by digitally set the EDM process parameters to the processor, and to control the servomotor manually. The LCD screen displays the selected information and instructions via an eight-bit data transfer from the eZdsp. In manual control mode, the program executes the PWM waveform for driving the servomotor with controllable speed, while in EDM control mode, the servomotor operates in a control loop. For real-time control, the program code for servomotor control is executed via the ISR.

In the main program, the software executes several other programs as explained in sections 3.1.1 to 3.1.5. Software design for the user-interface device and process parameters selection uses a Finite State Machine approach in order to reduce the use of hardware switching and eZdsp I/O ports. The EDM timing control program employs variable PWM software control to generate the pulse times t_{on} and t_{off}. The hysteretic current controller executes program code to generate a switching 'on' and 'off' signal for Q_2 depending on the feedback value of the gap current. The software for fault-detection signal detects the rising edge of the digital input, which then stops the continuous loop program immediately. The control strategy of the EDM system has been designed and implemented in software, while the hardware implementation for the control strategy is discussed in Chapter 4.

CHAPTER 4

HARDWARE DEVELOPMENT OF THE EDM SYSTEM

A block diagram of the EDM system hardware is shown in Figure 4.1. The system consists of three block subsystems; a digital controller unit, a servo system and a gap voltage and current pulse power generator.

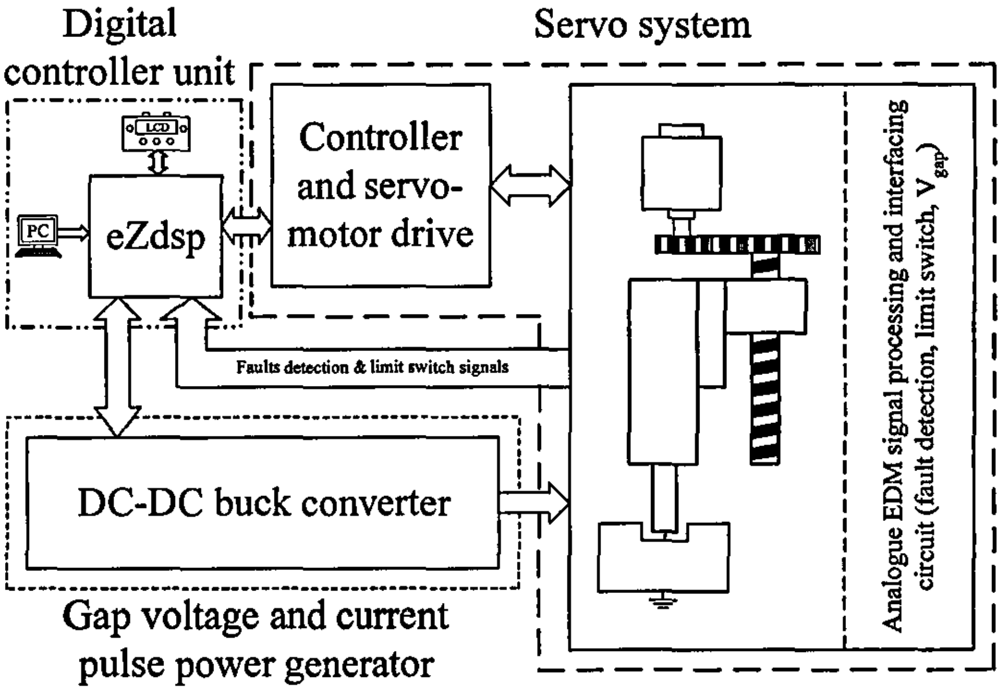

Digital controller unit
Servo system
LCD
PC
eZdsp
Controller and servo-motor drive
Faults detection & limit switch signals
DC-DC buck converter
Gap voltage and current pulse power generator
Analogue EDM signal processing and interfacing circuit (fault detection, limit switch, V_{gap})

Figure 4.1 Block diagram of EDM system hardware

The digital control unit uses the eZdsp microcontroller to process the control signals for the servomotor control and the gap voltage and current pulse power generator. The analogue EDM circuit board provides opto-isolated fault and V_{gap} signals as explained in section 3.1.5. The following subsections describe the EDM block subsystems.

4.1 DIGITAL CONTROLLER UNIT

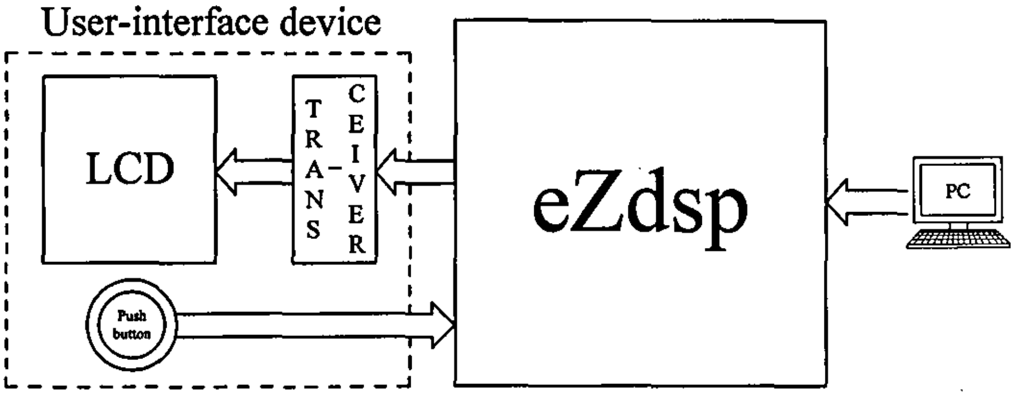


Figure 4.2 Block diagram of digital controller unit

The digital control unit consists of the eZdsp microcontroller development board and the user-interface device as shown in the block diagram of Figure 4.2. The software code is downloaded to the eZdsp's Flash ROM via a serial cable from the personal computer (PC). The eZdsp communicates to the user-interface device by sending the information/instruction to the LCD screen while the user-interface device uses push-button switches to communicate with the eZdsp. The user-interface device is equipped with two dial potentiometer for fine adjustment of $V_{gap\_f/b}$ and manual speed control $V_{N,man}$.

4.1.1 eZdsp

The eZdsp is a microcontroller development board (5.25" x 3") that uses a 3.3V DC supply, with a multi-layered printed circuit board as shown in Figure 4.3.

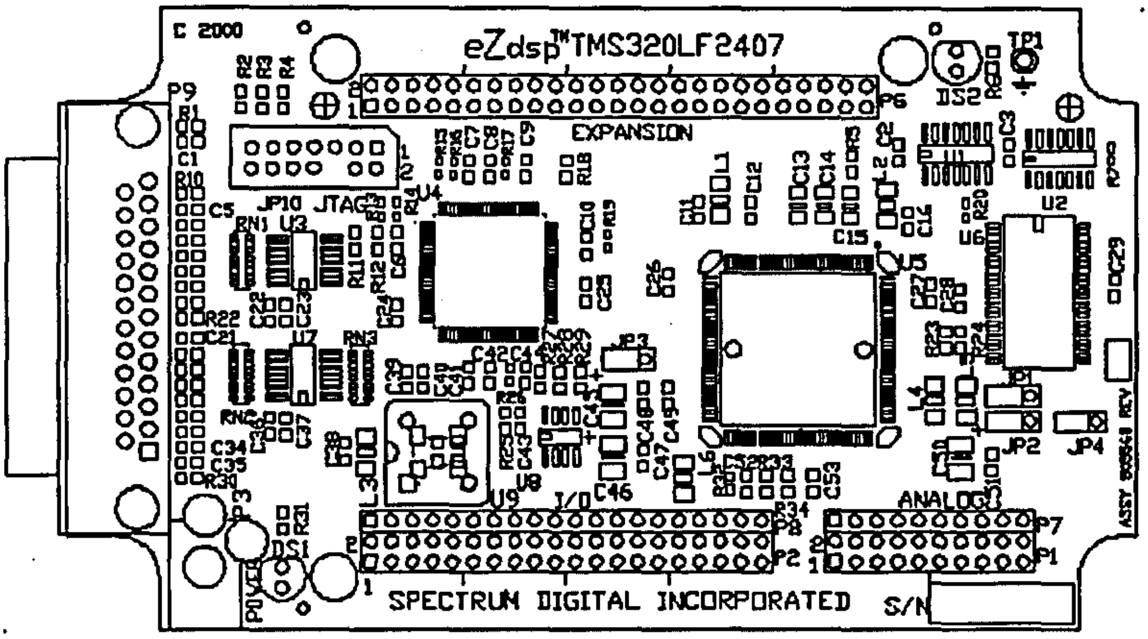


Figure 4.3 eZdsp microcontroller development board

The eZdsp operates from a parallel printer port, therefore no internal adapter card is required to communicate between a host PC and the eZdsp. The board is equipped with peripheral units for motor control applications [72,73]. A hardware layout function of the eZdsp is shown in Figure 4.4. It has six digital ports (A to F). Each port has eight I/O units. However, only five ports (A, B, C, E, F) are available for receiving and producing digital signal. The manufacturer reserves Port D internally. Nevertheless, all the I/O units share other functions. For example, I/O unit 6 of port A has two functions; PWM1 or digital output 6 can only use one function at any time. Two A/D units were used to convert the analogue inductor current feedback signal V_L and the servo system PWM control signal V_c.

One A/D unit was used for the servomotor manual speed control input signal $V_{N,man}$. Fifteen digital I/O units were connected to the LCD unit; eight for data communication and seven for unit initialization. Ten digital input units were used for Up/Down limit switches, push button switches and fault detection signals. Seven digital output units were used; two units were allocated for servomotor system PWM signals, two units for switching control of the gap voltage and current pulse power generator and another three units for LED indication of running and failing system.

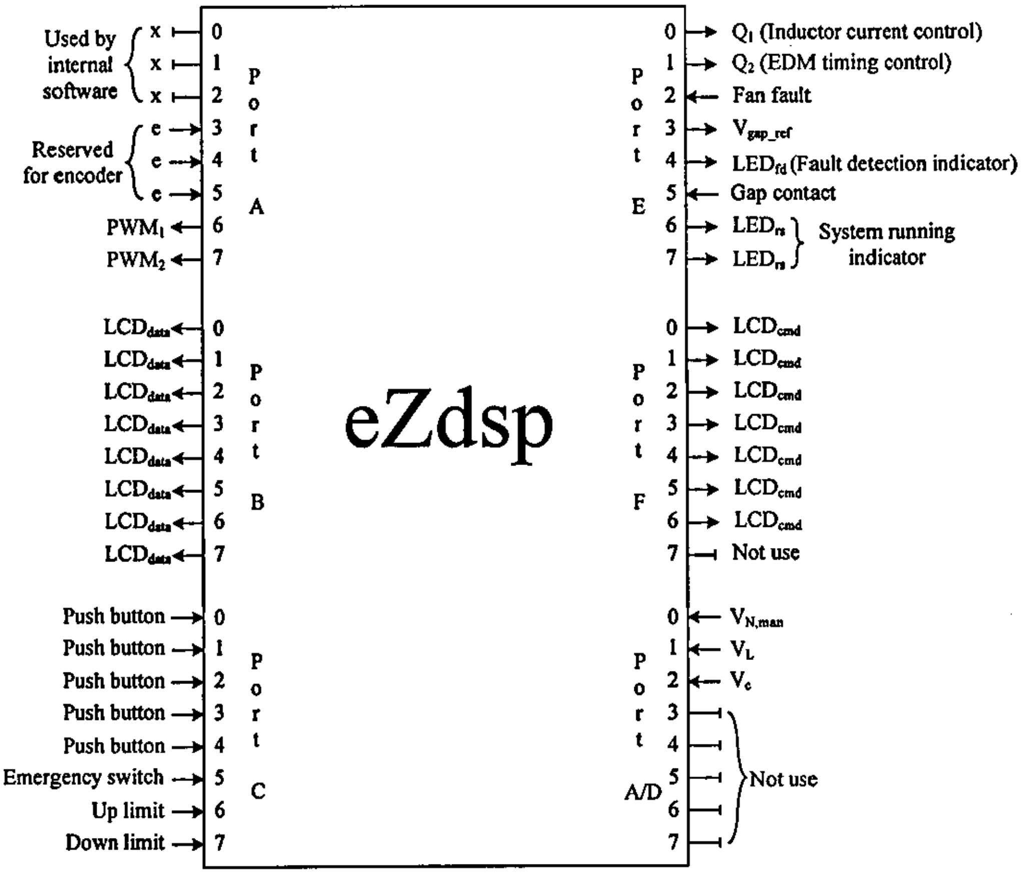


Figure 4.4 Hardware layout function of the eZdsp

4.1.2 User-interface device

The user-interface device is made up of an LCD unit for display purposes and push button switches for digital parameter selection. As shown in Figure 4.2, a transceiver was used between the eZdsp output signals and the LCD input signals. Figure 4.5 shows the circuit diagram of the user-interface device.

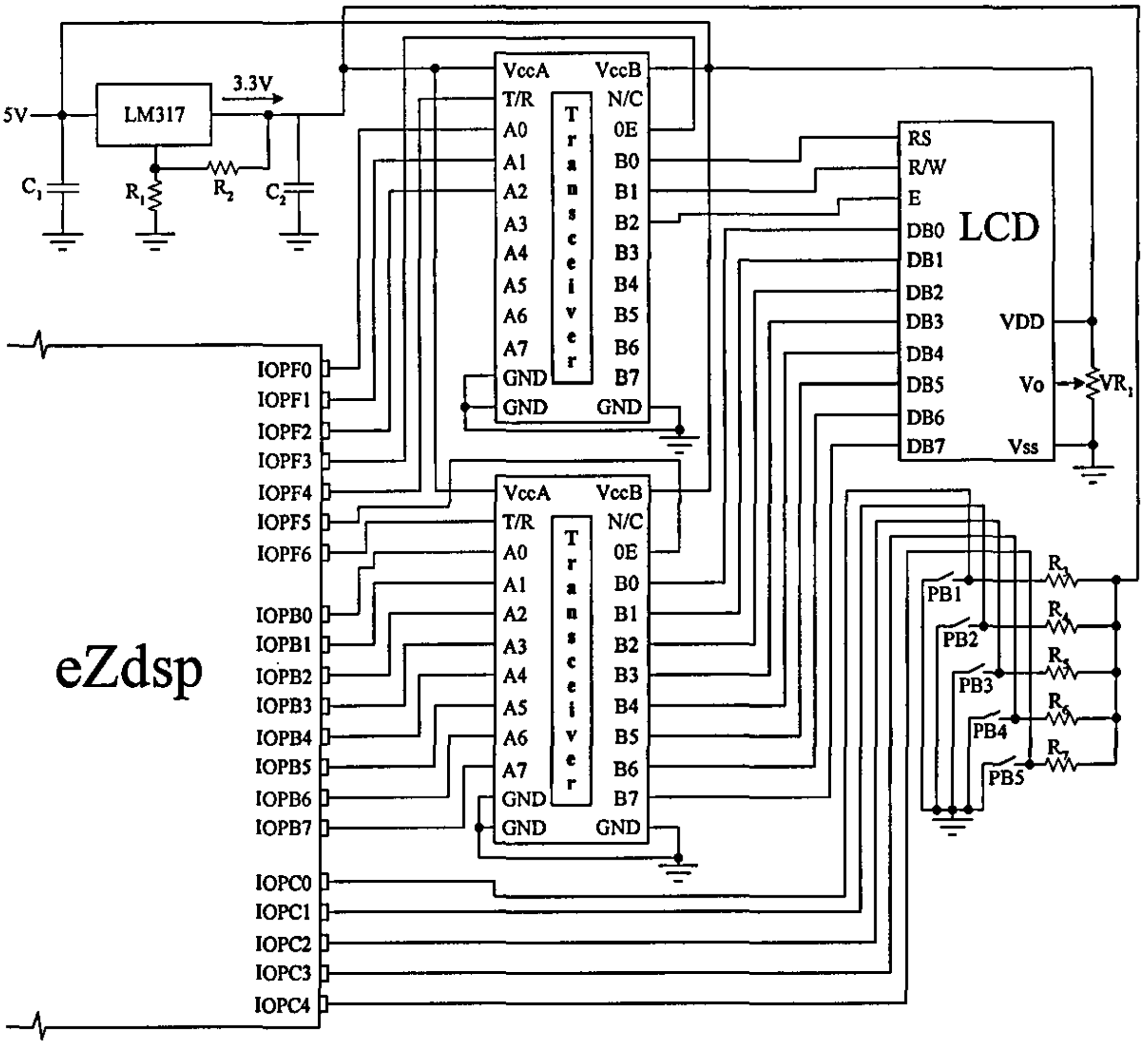


Figure 4.5 Circuit diagram of the user-interface device

4.1.2.1 Push button switch

Five 'push-to-make' push button switches were used for selecting the parameter values in digital form. These switches perform different tasks according to the assigned state position as explained in section 3.1.1. The digital input ports are holding logic '1' as initial input signals (see Figure 4.5). Pressing the button sends logic '0' to the eZdsp input port that indicates a signal to activate a particular software function.

4.1.2.2 LCD display unit

The display information/instruction unit uses a Powertip 1602 series LCD. The LCD is capable of displaying sixteen characters in two lines (16 x 2). As explained in Chapter 3, the LCD does not require any special programming except some signals initialization. The LCD accepts an 8-bit signal for displaying characters. To display the characters at a specific place on the LCD screen, command (3-bit) and data character (8-bit) signals are sent via the eZdsp output ports.

4.1.2.3 Transceiver

Direct connection between the eZdsp and the LCD unit cannot be established due to their different voltage levels. The output signals from the eZdsp is 3.3V DC maximum while, the input signal to the LCD unit is 4.5V DC minimum. Therefore, a signal level (voltage) interface is used to effect signal transfer. The interface device is an 8-bit Dual Supply Translating Transceiver 74LVX3245. The side connected to the eZdsp is supplied with 3.3V DC and the side connected to the LCD is supplied with 5V DC. Two ICs were used to accommodate the 3-bit command signals and the 8-bit data signals.

4.2 SERVO SYSTEM

In the servo system shown in Figure 4.1, rotational motion of DC motor rotor is reduced by the gear system n_g to linearly drive the electrode position z towards the workpiece. The PWM signals from the eZdsp are connected to the DMOS H-bridge DC servomotor power driver inputs to control the servomotor. An LEM Hall effect current sensor is used to measure the motor current I_m and provides the current feedback signal to the controller. The signal from the tacho-generator provides the speed information to the controller and the gap voltage V_{gap} signal that is measured between the electrode and the workpiece is fedback to the average gap voltage controller. Signals to the eZdsp are conditioned in order to obtain compatible voltage levels from the analogue circuits.

4.2.1 DC servomotor power drive circuit

The DC servomotor power drive circuit uses an L6203 48V 5A DMOS Full Bridge driver as shown in Figure 4.6. IN1 and IN2 are the driver's PWM inputs with 24V DC supply used to drive the motor via the H bridge. These inputs are connected to the eZdsp's I/O ports via opto-isolator components that provide voltage level isolation between the eZdsp and the power drive circuit. The circuit is also equipped with an over-current protection circuit to 'shut-down' the driver's operation by sending a disable signal (logic '0') to the ENABLE input pin. The current sensing circuit uses an LEM LA 25-NP (5A max) to sense the motor current I_m.

4.2.2 Analogue EDM signal processing and interfacing circuit

The analogue control scheme in Figure 2.29 is implemented as in Figure 4.7 using operational amplifiers and associate components. With reference to Figure 4.7 and Figure 2.29, the gap reference voltage V_{gap_ref} used to indirectly set the gap width is obtained from

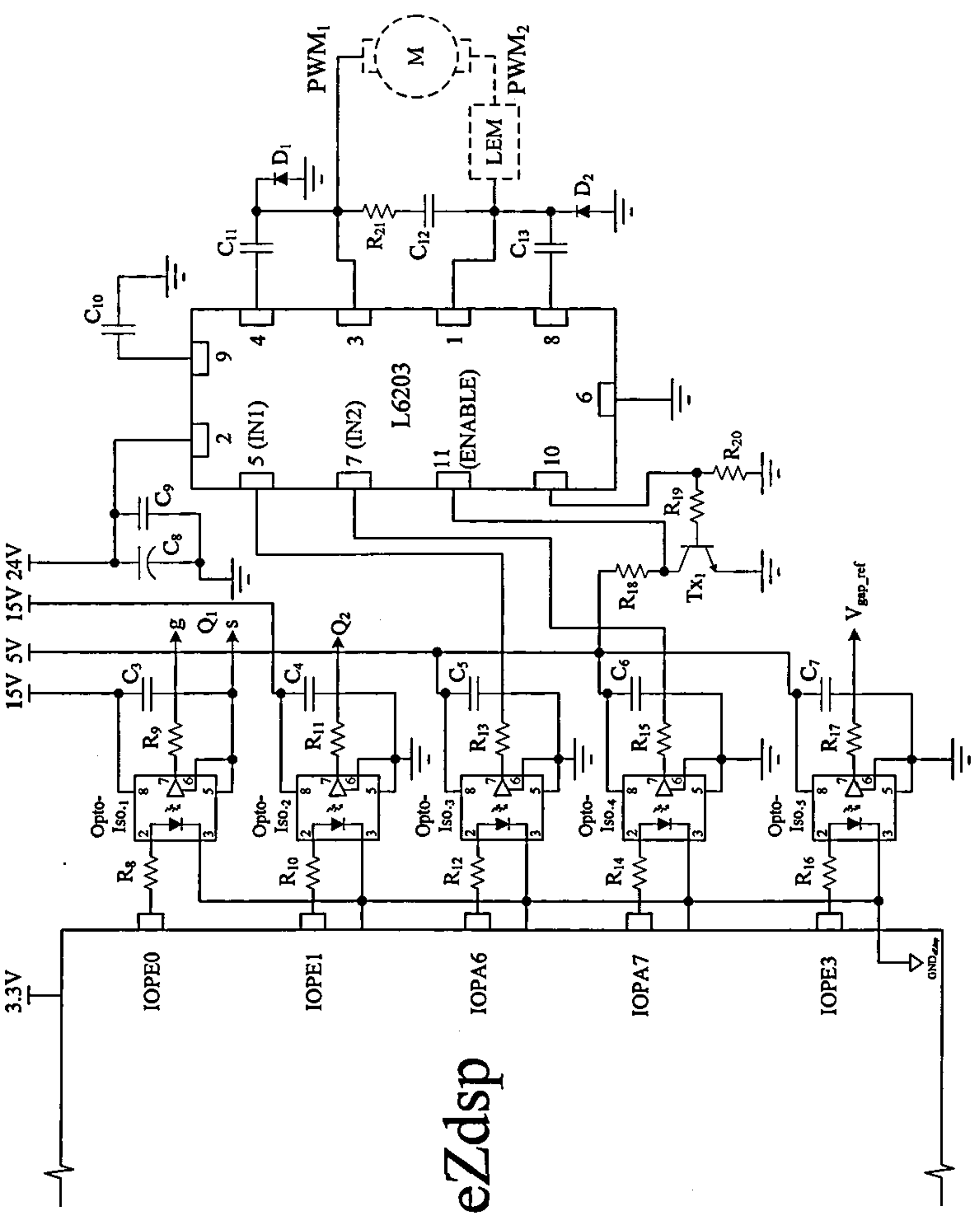


Figure 4.6 DC servomotor power drive circuit

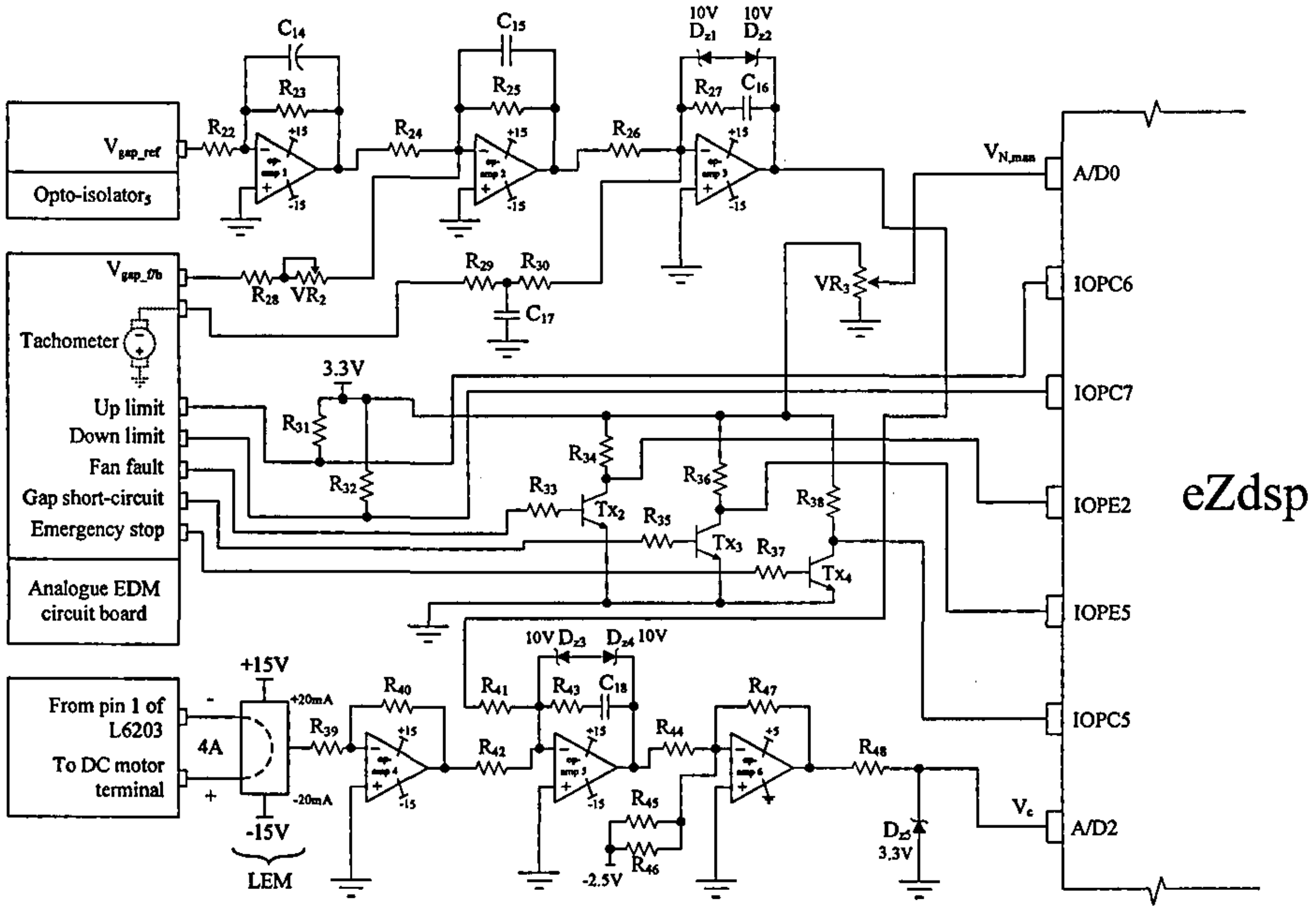


Figure 4.7 Analogue EDM signal processing and interface circuit diagram

the microcontroller via opto-isolator$_5$. op-amp1 and the associated component R_{22}, R_{23} and C_{14} amplify and smooth this signal. The V_{gap} feedback voltage $V_{gap\_f/b}$ is obtained from the analogue EDM circuit board. op-amp2 performs the summing and proportional amplification of the error using components R_{24}, R_{25},R_{28},VR_2 and C_{15}. C_{15} is used to suppress noise and also provides smoothing of the error signal. VR_2 is used to fine tuned the gap voltage feedback level. The output of op-amp2 is the speed reference for the servomotor. The speed reference and the tachometer output from the analogue EDM circuit board are fed to the Proportional Integrator PI circuit consisting of op-amp3 and associated components R_{26}, R_{27}, R_{30} and C_{16}. R_{29} and C_{17} filter any noise on the tachometer output. D_{z1} and D_{z2} limit the output of op-amp3 to just over +/- 10V thus preventing op-amp saturation. The output of op-amp3 is the current reference for the servomotor. The current feedback signal is obtained via an LEM current transducer, op-amp4, and resistors R_{39} and R_{40}. The current error function and PI function are performed by op-amp5 and components R_{41}, R_{42}, R_{43} and C_{18}. D_{z3} and D_{z4} limit the output of op-amp5 to just over +/-10V. The output of op-amp5 is conditioned via op-amp6 and components R_{44} to R_{48} and D_{z5}. The output of op-amp5 is the PWM control signal V_c to the eZdsp and is limited to 3.3V. VR_3 is used to control the servomotor speed in the manual mode. Fault detection is performed by switching networks consists of transistor and resistor components. The fan fault signal for example, turns 'on' Tx_2 via R_{33} and R_{34} sending a logic signal '0' to the eZdsp.

4.3 GAP VOLTAGE AND CURRENT PULSE POWER GENERATOR

The gap voltage and current pulse generator is the buck converter shown in Figure 4.8 and consists of MOSFET switches $Q_{1,2}$, diodes $D_{1,2}$ and inductor L. The output quantity being control is the inductor current up to 50A maximum. The hysteretic control method as explained in section 3.1.4 is used to control the current. The maximum gap voltage is fixed by V_d to 160V, which is the input voltage to the buck converter. Q_2 is turned on to divert the inductor current from the gap during the pulse off-time t_{off}. During the pulse on-time t_{on} and also for the period t_D, Q_2 is turn off and inductor current flows into the gap during gap breakdown or back to the supply via D_2 during a gap open circuit condition. The above voltage and current pulse generator is a new topology applied to EDM and results in efficient gap voltage and current generation without the attendant power loss associated with the use of a switch power resistor that is used to generate the current in previous EDM system [74]. The above topology gives fast voltage and current pulse rise and fall times. A signal conditioning circuit is used to interface V_L signal level from the existing signal generator circuit to eZdsp's A/D1. Details operation of the gap voltage and current pulse power generator appear in Appendix 5.

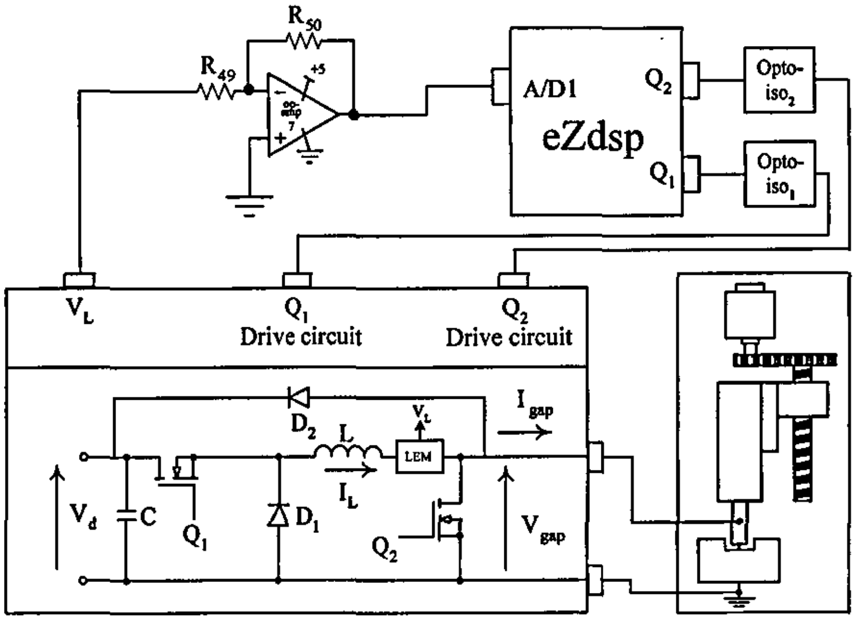


Figure 4.8 Gap voltage and current pulse power generator topology and circuit diagram

4.4 CONCLUSION

The eZdsp microcontroller development board was used in order to utilize the TMS320LF2407A DSP fixed point processor for digital control and to minimize the associate hardware components of the EDM control system. All of the existing hardware EDM system control was implemented in software terms and includes the process parameter signal generation as well as the servomotor control function. Operational amplifiers were used for analogue EDM signal processing and interfacing circuitry. The analogue signals are conditioned to the 3.3V level required by the eZdsp. The user-interface device uses an LCD and push button switches for displaying purposes and for the selection of process parameters as well as for manual and start/stop control functions. A transceiver was used to effect transfer of information between the eZdsp and the LCD. The hardware component list is shown in Appendix 6.

CHAPTER 5

EXPERIMENTAL STUDIES

This chapter describes the experimental studies of the EDM process when using a copper electrode and a graphite electrode with a steel workpiece. Experimental results of the material removal rate from a steel workpiece are recorded and presented in tabular form. A widely used BP200 hydrocarbon mineral oil was employed as the dielectric fluid. An 'Open flushing' condition (see Figure 1.9) was applied to circulate the dielectric fluid between the electrode and the workpiece.

A cylindrical electrode of diameter 20mm is used. The open gap voltage at maximum V_{max} is 160V. For each gap current value, several experiments were conducted at different 'on' and 'off' times in order to adjust the sparking frequency F_s for maximum material removal rate. Experiments at particular gap current settings were recorded as process 1, 2, 3, etc. When breakdown occurs, the average gap voltage feedback is fine tuned by adjustment of VR_2 in Figure 4.7 to give a time delay t_D of 2μs. The gap voltage V_{arc} falls to about 25V and the gap current rises to the selected constant value.

Figure 5.1 shows an oscillographic recording of gap current and gap voltage for the experimental system. The digital storage scope used to capture the waveforms was not capable of sampling and displaying the 2μs delay period. The delay period is clearly seen in Figure 5.1 where an adjustment was made in the servo gap width control potentiometer VR_2.

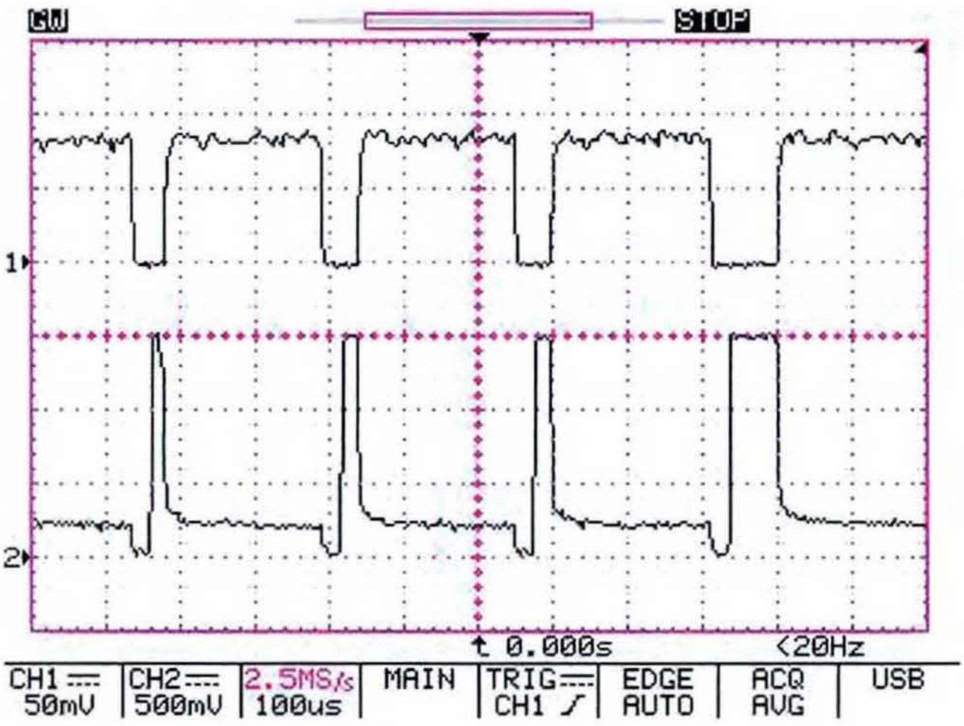


Figure 5.1 Showing EDM operation, gap current and gap voltage waveforms CH1:I_{gap}=5A/div CH2:V_{gap}=50V/div Time=100μs/div

Figure 5.2 is an oscillographic recording of gap voltage and servo system control voltage with VR_2 set to provide too large a gap width, leading to open circuit EDM conditions. Adjusting VR_2 to reduce the gap width for normal EDM condition leads to the oscillographic recording of Figure 5.3.

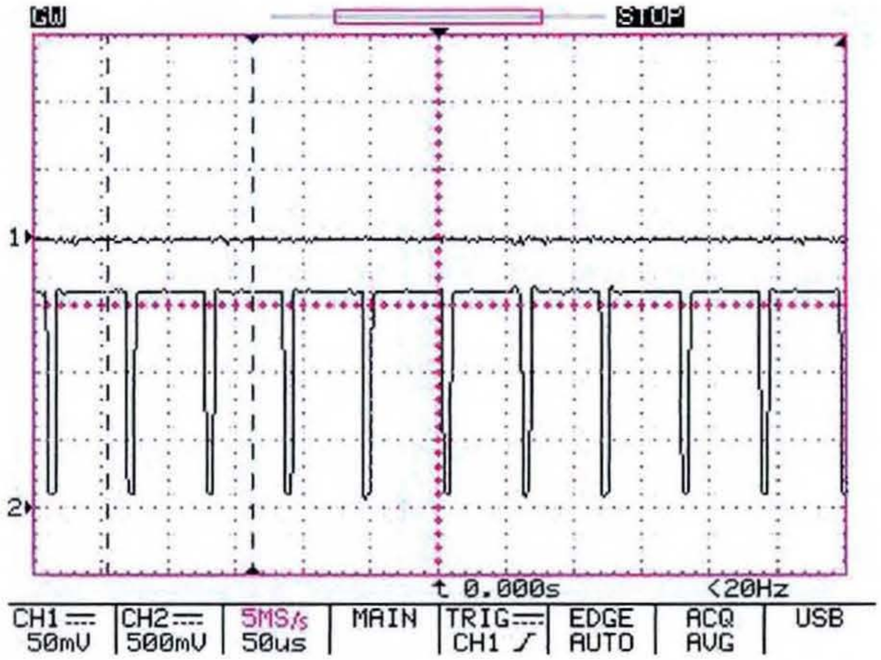


Figure 5.2 Showing open circuit operation, servo system control voltage V_c and gap voltage waveforms CH1:V_c=50mV/div CH2:V_{gap}=50V/div Time=50μs/div

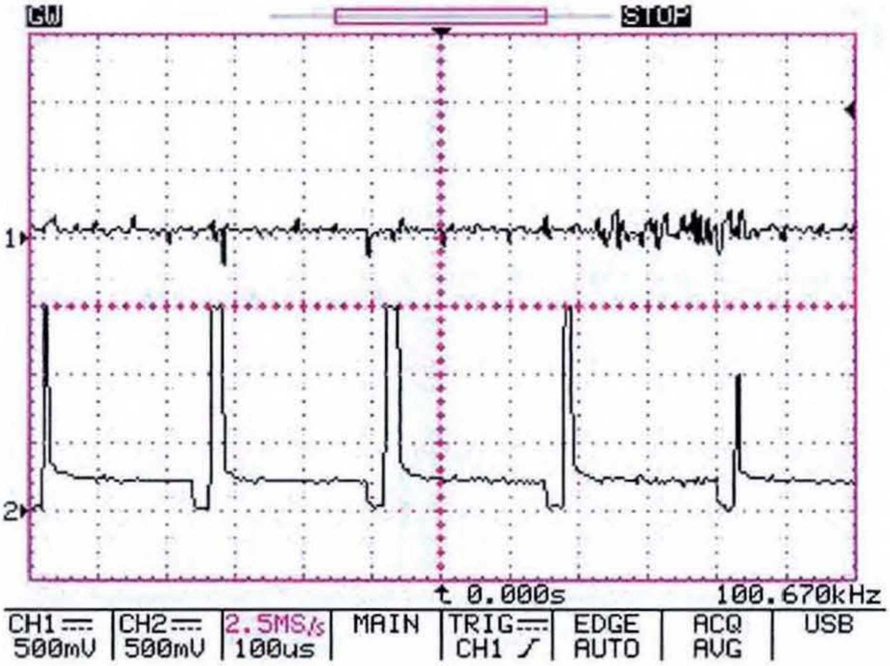
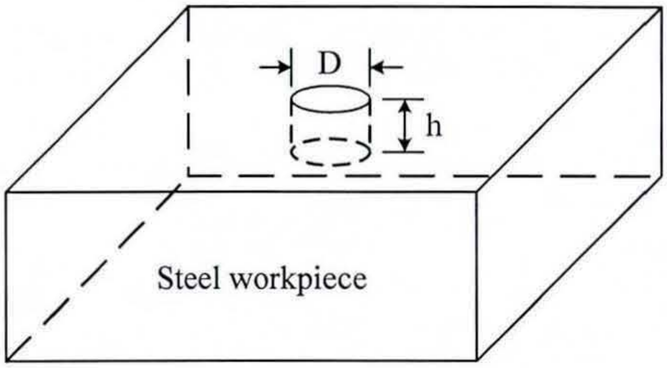


Figure 5.3 Showing EDM operation, servo system control voltage V_c and gap voltage waveforms CH1:V_c=500mV/div CH2:V_{gap}=50V/div Time=100μs/div

The removed material for a given period is measured in cubic millimetres per minute, is of a cylindrical form and calculated as detailed in Figure 5.4. The diameter D and height h were accurately measured using a precision digital dial calliper. The following sections 5.1 and 5.2 present the material removal rate results of the steel workpiece for copper and graphite electrodes respectively.



$$\text{Volume of material removal} = \left(\frac{\pi D^2}{4}\right) \times h$$

Figure 5.4 Method used to measure material removed

5.1 MATERIAL REMOVAL RATE RESULTS FOR A STEEL WORKPIECE AND A COPPER ELECTRODE

The gap currents were selected at 4A, 6A, 8.5A, 12.5A, 18A, 25A, 36A and 50A. The following Table 5.1 shows the process parameters of the experimental data selected and the experimental results of the material removal rate.

Table 5.1 Experimental data and results of a steel workpiece and a copper electrode

Process	t_{on} (μs)	t_{off} (μs)	F_s (kHz)	$\dot{V}$ (mm^3/min)							
				at I_{gap}=4A	at I_{gap}=6A	at I_{gap}=8.5A	at I_{gap}=12.5A	at I_{gap}=18A	at I_{gap}=25A	at I_{gap}=36A	at I_{gap}=50A
1	2	4	125	4	7	-	-	-	-	-	-
2	3	4	111.11	6	9	11	16	-	-	-	-
3	4	4	100	8	11	16	20	16	46	-	-
4	6	4	83.33	10	12	21	31	42	60	72	82
5	12	4	55.55	13	19	23	43	54	81	111	143
6	25	4	32.25	15	23	31	48	68	99	137	170
7	50	6	17.24	17	26	36	52	79	126	181	218
8	100	12	8.77	19	23	38	54	86	126	175	250
9	200	25	4.41	13	21	33	47	72	110	151	221
10	400	50	2.21	12	19	29	43	65	90	141	200

5.2 MATERIAL REMOVAL RATE RESULTS FOR A STEEL WORKPIECE AND A GRAPHITE ELECTRODE

The gap currents were selected at 6A, 8A, 12.5A, 18A, 25A, 37.5A and 50A. The following Table 5.2 shows the process parameters of the experimental data selected and the experimental results of the material removal rate.

Table 5.2 Experimental data and results of a steel workpiece and a graphite electrode

Process	t_{on} (μs)	t_{off} (μs)	F_s (kHz)	$\dot{V}$ (mm^3/min)						
				at I_{gap}=6A	at I_{gap}=8A	at I_{gap}=12.5A	at I_{gap} =18A	at I_{gap}=25A	at I_{gap}=37.5A	at I_{gap} =50A
1	3	4	111.11	9	-	-	-	-	-	-
2	4	4	100	11	19	-	-	-	-	-
3	6	4	83.33	14	21	28	-	-	-	-
4	12	4	55.55	17	29	39	53	66	-	-
5	18	6	38.46	19	31	42	66	89	-	-
6	25	6	30.3	22	35	46	75	100	105	209
7	38	6	21.74	21	36	49	78	107	174	222
8	50	6	17.24	-	34	47	81	115	180	235
9	75	12	11.24	-	-	33	79	111	168	243
10	100	18	8.33	-	-	-	61	85	141	237
11	200	25	4.41	-	-	-	47	64	133	180
12	400	50	2.21	-	-	-	-	-	116	175

5.3 DISCUSSION

Tables 5.3 and 5.4 show the optimum values of process parameters t_{on}, t_{off} and the steel-workpiece material removal rate results at specific gap currents. From the tables, graphs of the material removal rate as a function of average gap power $\left(I_{gap} \times V_{arc} \times \frac{t_{on}}{t_{on} + t_{off} + t_D}\right)$ were plotted as shown in Figures 5.5 and 5.6.

Table 5.3 Process parameters at optimum value of material removal rate for a copper electrode and a steel workpiece

Experiment	I_{gap} (A)	t_{on} (μs)	t_{off} (μs)	$\dot{V}$ (mm^3/min)
2	4	100	12	19
3	6	50	6	26
4	8.5	100	12	38
5	12.5	100	12	54
6	18	100	12	86
7	25	50	6	126
8	36	50	6	181
9	50	100	12	250

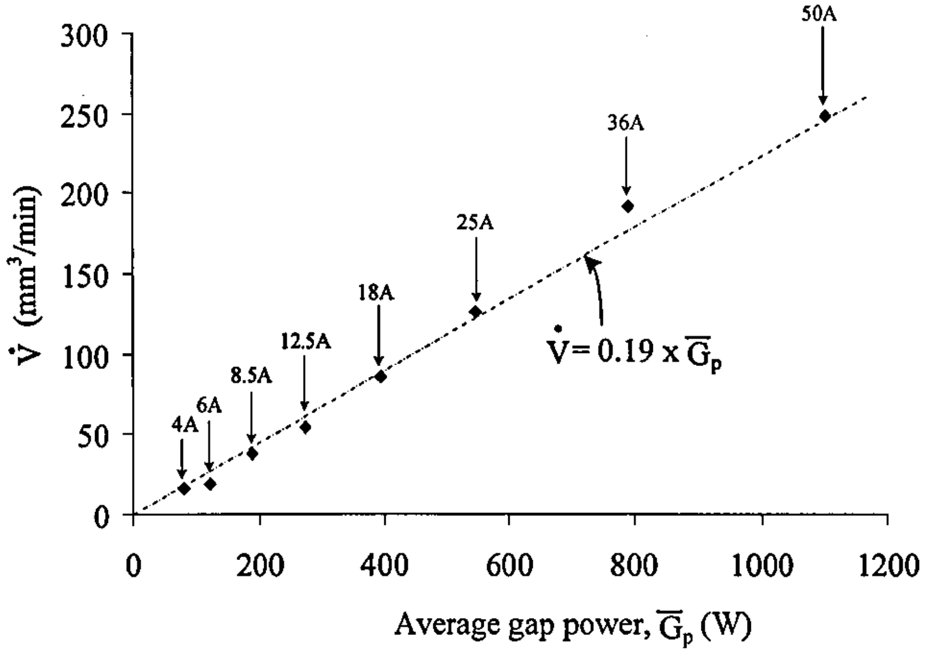


Figure 5.5 Optimum material removal rate as a function of average gap power for a copper electrode and a steel workpiece

Table 5.4 Process parameters at optimum value of material removal rate for a graphite electrode and a steel workpiece

Experiment	I_{gap} (A)	t_{on} (μs)	t_{off} (μs)	$\dot{V}$ (mm^3/min)
1	6	18	6	19
2	8	38	6	36
3	12.5	38	6	49
4	18	50	6	81
5	25	50	6	115
6	37.5	50	6	180
7	50	75	18	243

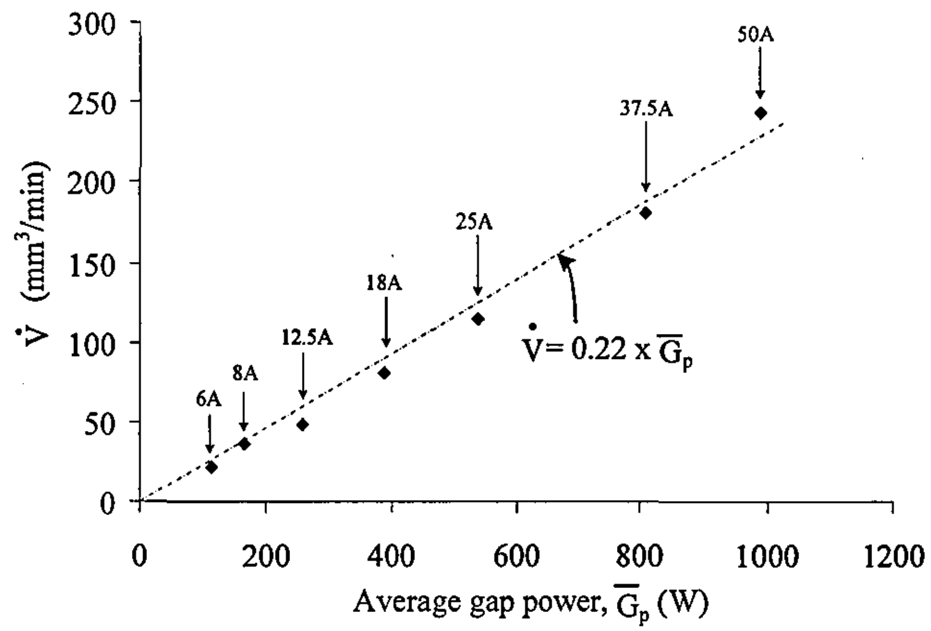


Figure 5.6 Optimum material removal rate as a function of average gap power for a graphite electrode and a steel workpiece

At gap currents of 12.5A, 18A, 25A and 50A, the optimum value of material removal rate for a copper-electrode is 54mm^3/min, 86mm^3/min, 126mm^3/min, 248mm^3/min respectively and for a graphite-electrode, the optimum material removal rate is 49mm^3/min, 81mm^3/min, 115mm^3/min, 243mm^3/min, respectively. The experimental results show that a higher material removal rate of the steel workpiece is achieved using the copper electrode material compared to graphite electrode material. Even at the same level of the applied gap current, a different amount of material removal rate is observed which is mainly related to the thermal conductivity of both electrode and workpiece [75]. Thus, proper parameter setting for t_{on} and t_{off} is necessary in order to obtain optimum material removal rate.

From the experimental data and results at particular gap current, as shown in Tables 5.1 and 5.2, it can be noticed that as the pulse on-time increases, material removal rate also increases up to the optimum value of specific pulse on-time. Beyond this value, material removal rate starts decreasing rapidly. This is due to the long pulse duration that diminishes pressure and energy of the plasma channel over the molten material of the electrode [74]. As a consequence, this phenomenon brings instability to the process that reduces the material removal rate. The graph in Figure 5.5 and 5.6 illustrate that the material removal rate increases with the increase of average gap power. The graph also indicates that the material removal rate linearly increases with the average gap power. Therefore, based on the experimental evidence, the material removal rate $\dot{V}$ is proportional to the gap current I_{gap}, gap voltage V_{arc}, pulse on-time t_{on} and sparking frequency F_s.

5.4 CONCLUSION

Copper and graphite electrodes have been used in material removal rate experiments with a steel workpiece material. It was found that a copper electrode results in a higher material removal rate when compared to a graphite electrode. These two electrodes are different in material properties. Copper is a metallic material while graphite is a non-metallic material. As reviewed in Chapter 1, the electrode material properties are an important factor to consider when choosing the electrode. Those materials with the lowest electrical resistivity and the highest melting point are preferred. For metallic workpiece materials, a copper electrode is preferred and for a non-metallic workpiece material, a graphite electrode is favoured. A graphite electrode produces a lower material removal rate and is more expensive, compared to a copper electrode. However a graphite electrode is still sometimes used because of its high resistance to heat and this makes it suitable for machining high melting point workpiece materials such as high temperature alloys[75].

CHAPTER 6

COMPARISON BETWEEN SIMULATION AND EXPERIMENTAL RESULTS

This chapter compares the simulation results generated in Chapter 2 with the experimental results produced in Chapter 5.

6.1 SIMULATION AND EXPERIMENTAL RESULTS

Simulation and experimental results of material removal rate (mm^3/min) versus t_{on} (μs) for a copper electrode and a steel workpiece are shown in Figure 6.1. Simulation results show reasonable agreement with the experimental results except at short and long pulse on-time. The discrepancies of material removal rate at short and long pulse on-times are due to the low plasma flushing efficiency and material resolidification respectively. The plasma flushing efficiency is the fraction of the theoretical melt volume actually removed upon collapse of the plasma. As the current increases so does the optimum pulse time for maximum erosion rate. This causes the energy contained in the plasma, which is proportional to $I_{gap}t_{on}$, to increase significantly. Low energy plasma fails to build up sufficient pressure during its relatively short pulse time to expel all of the melt into the dielectric, thus lowering the plasma flushing efficiency. The resolidification phenomena arise as follows. At the end of the on-time, a pause period t_{off} begins when power is terminated to the workpiece. During this period, a violent collapse of the plasma channel and the vapor bubble occurs, causing the superheated, molten liquid on the surface of both electrodes to explode into the liquid dielectric. While some of this material is carried away by the dielectric, the remainder of the melt in the cavities resolidifies in place, waiting to be removed by a later spark. As a consequence of this, the erosion rate reduces at long pulse on-times.

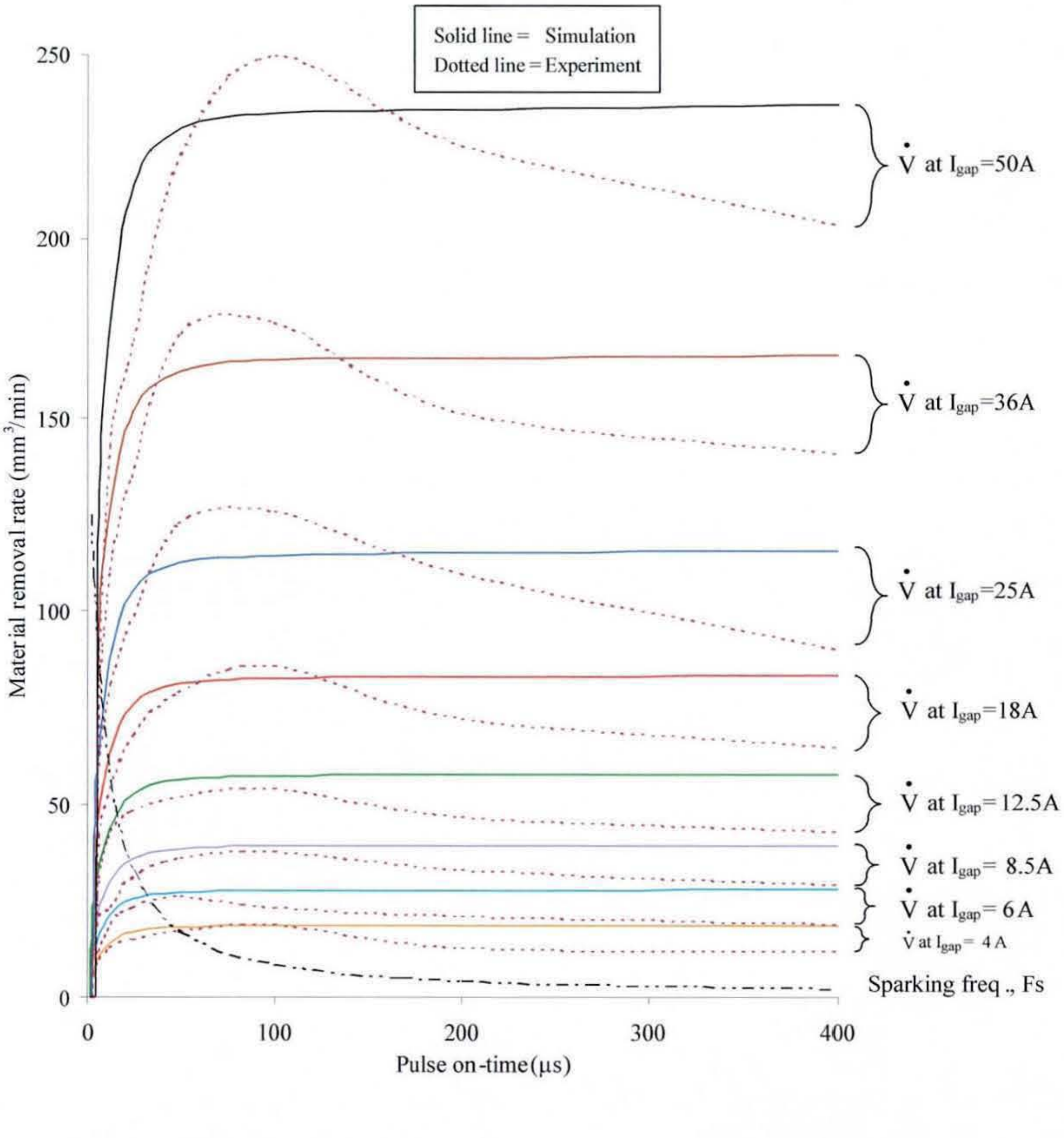


Figure 6.1 Material removal rate versus pulse on-time of simulation and experiment

6.2 IMPROVED EDM PROCESS SIMULATION MODEL

The simulation results use equation (2.16) in order to predict the material removal rate. However, equation (2.16) has been developed without considering the reduced erosion rate factor, [RERF] that accounts for the low plasma flushing efficiency at short pulse on-times and the material resolidification at long pulse on-times. The plasma flushing efficiency and the resolidification effects can be taken into account with the addition of a dimensionless equation, formulated from the experimental results. The following section explains the [RERF] generation.

6.2.1 Generation of [RERF]

Regression analysis is utilised in order to obtain an equation that enables the theoretical results to match the experimental results at short and long pulse on-times. The reduced erosion rate factor [RERF] is introduced into equation (2.16) as shown in equation (6.1).

$$\dot{V} = C\alpha V_{arc} I_{gap} t_{on} F_s [RERF] \qquad (6.1)$$

Equation (6.1) is rearranged as shown in equation (6.2) to evaluate [RERF].

$$[RERF] = \frac{\dot{V}}{C\alpha V_{arc} I_{gap} t_{on} F_s} \qquad (6.2)$$

The experimental data of the eight EDM processes listed in Table 6.1 was used in equation (6.2) to evaluate the average value for [RERF] with V_{arc}=25V, α=2 x 10^{-12} and C=1.74.

Table 6.1 Experimental data and [RERF] value

Process	t_{on} (μs)	F_s (kHz)	I_{gap}=4A		I_{gap}=6A		I_{gap}=8.5A		I_{gap}=12.5A		I_{gap}=18A		I_{gap}=25A		I_{gap}=36A		I_{gap}=50A		Average [RERF]
			$\dot{V}\left(\frac{mm^3}{min}\right)$	[RERF]	$\dot{V}\left(\frac{mm^3}{min}\right)$	[RERF]	$\dot{V}\left(\frac{mm^3}{min}\right)$	[RERF]	$\dot{V}\left(\frac{mm^3}{min}\right)$	[RERF]	$\dot{V}\left(\frac{mm^3}{min}\right)$	[RERF]	$\dot{V}\left(\frac{mm^3}{min}\right)$	[RERF]	$\dot{V}\left(\frac{mm^3}{min}\right)$	[RERF]	$\dot{V}\left(\frac{mm^3}{min}\right)$	[RERF]	
1	2	125	4	0.76	7	0.89	-	-	-	-	-	-	-	-	-	-	-	-	0.83
2	3	111.1	6	0.86	9	0.86	11	0.74	16	0.73	-	-	-	-	-	-	-	-	0.80
3	4	100	8	0.95	11	0.87	16	0.90	20	0.76	16	0.42	46	0.88	-	-	-	-	0.80
4	6	83.33	10	0.95	12	0.76	21	0.94	31	0.95	42	0.89	60	0.91	72	0.76	82	0.62	0.85
5	12	55.55	13	0.93	19	0.91	23	0.77	43	0.98	54	0.86	81	0.93	111	0.88	143	0.82	0.89
6	25	32.25	15	0.89	23	0.91	31	0.86	48	0.91	68	0.89	99	0.94	137	0.90	170	0.80	0.89
7	50	17.24	17	0.94	26	0.96	36	0.94	52	0.92	79	0.97	126	1.12	181	1.11	218	0.96	0.99
8	100	8.77	19	1.03	21	0.76	38	0.97	54	0.94	86	1.04	126	1.10	175	1.06	250	1.09	1.00
9	200	4.41	13	0.70	23	0.83	33	0.84	47	0.81	72	0.86	110	0.95	151	0.91	221	0.96	0.86
10	400	2.21	12	0.65	19	0.68	29	0.73	43	0.74	65	0.78	90	0.78	141	0.84	200	0.86	0.76

A ratio of t_{on}/t_D is established in order to obtain a dimensionless [RERF]. Plotting the average value of [RERF] so obtained against t_{on}/t_D (see Figure 6.2) and using regression analysis, leads to the following equation;

$$[\mathrm{RERF}] = 0.81 + 8.37\times10^{-3}\left(\frac{t_{on}}{t_D}\right) - 9.66\times10^{-5}\left(\frac{t_{on}}{t_D}\right)^2 + 2.71\times10^{-7}\left(\frac{t_{on}}{t_D}\right)^3 \tag{6.3}$$

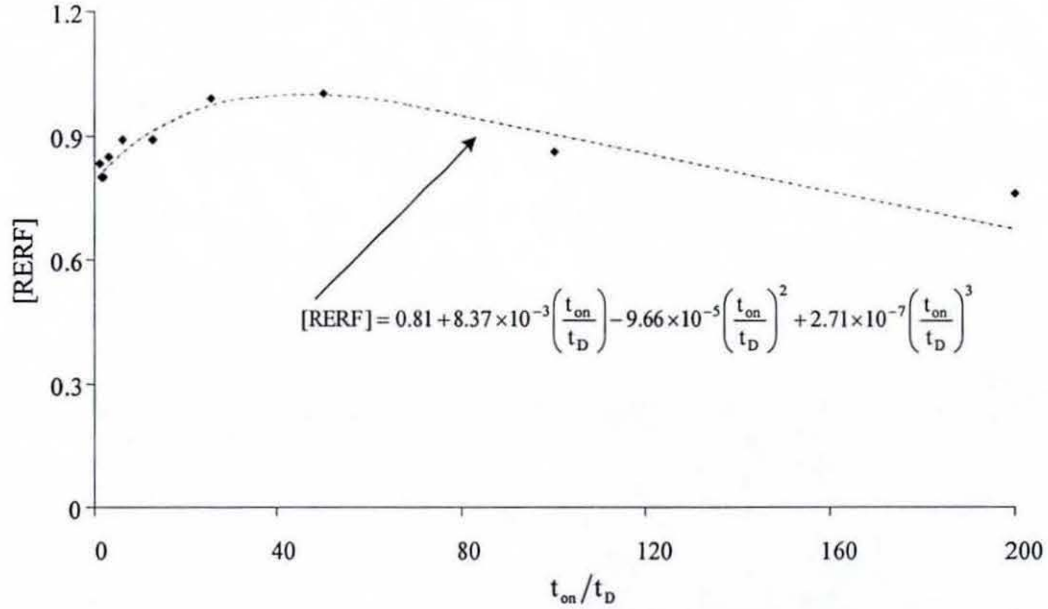


Figure 6.2 [RERF] value as a function of t_{on}/t_D

Equation (6.3) is valid for t_{on} up to 400μs, which is the maximum on-time used for the experimental results. Combining equation (2.16) and (6.3) gives the final equation for material removal rate as

$$\dot{V} = C\alpha V_{arc} I_{gap} t_{on} F_s \left[0.81 + 8.37\times10^{-3}\left(\frac{t_{on}}{t_D}\right) - 9.66\times10^{-5}\left(\frac{t_{on}}{t_D}\right)^2 + 2.71\times10^{-7}\left(\frac{t_{on}}{t_D}\right)^3\right]. \tag{6.4}$$

6.2.2 Improved simulation model versus experimental result

The improved simulation model uses the new material removal rate equation (6.4). Graphs of material removal rate and sparking frequency versus pulse on-time for various gap current are shown in the following Figure 6.3 with V_{arc}=25V, α=2 x 10^{-12} and C=1.74.

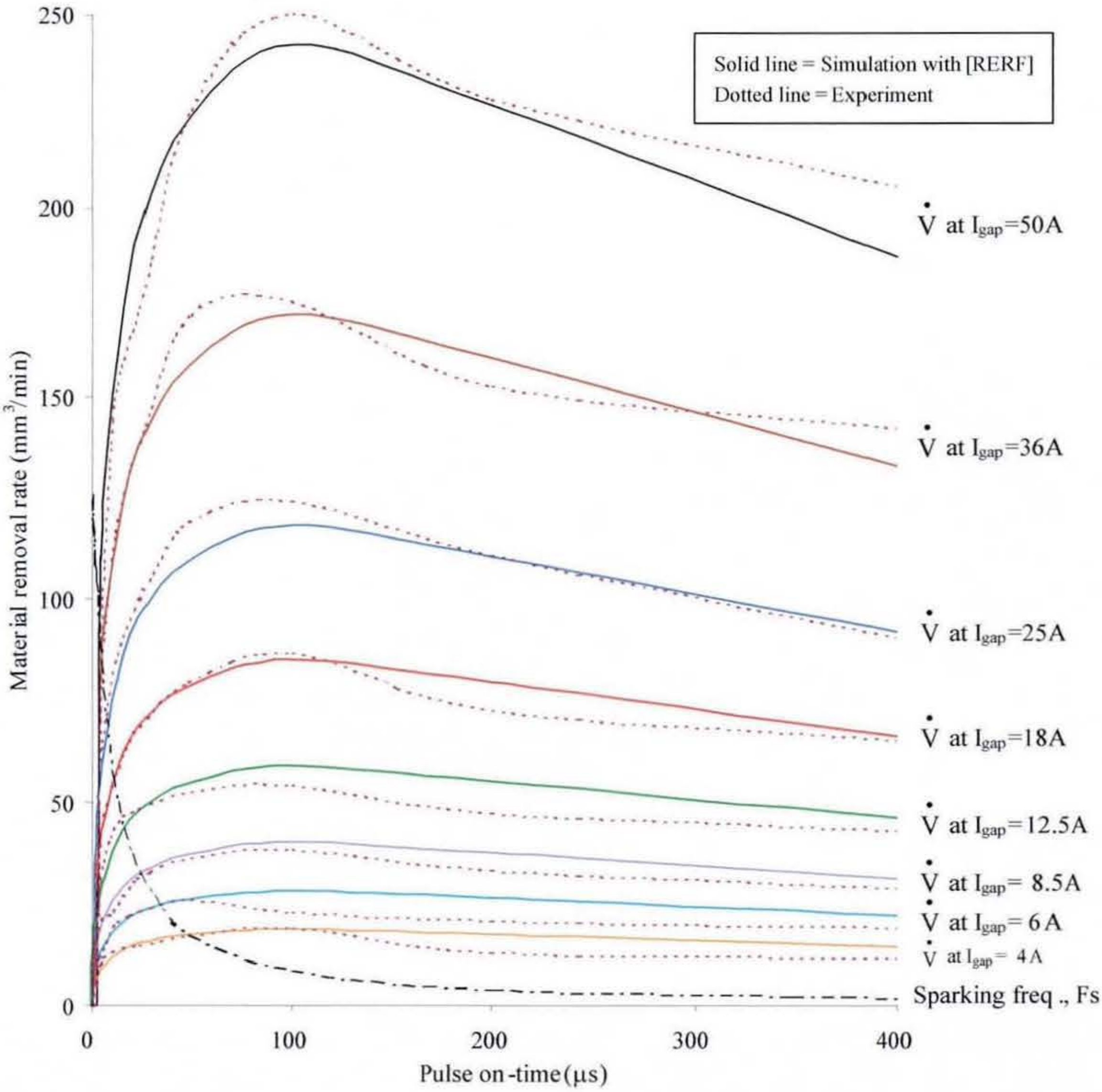


Figure 6.3 Material removal rate versus pulse on-time of simulation with [RERF] and experiment

Simulation results show acceptable agreement with the experimental results after taking into account the reduce erosion rate factor [RERF]. Equation (6.4) is only valid for t_D equal to 2μs. A new [RERF] equation must be regenerated if t_D is set to a different value.

6.3 CONCLUSION

Comparison studies between simulation and experimental results have been presented in this chapter. The results of the comparison using the original material removal rate equation developed using dimensional analysis reveal reasonable agreement of material removal rate in the region of optimum pulse on-time. However at short and long pulse on-times, large discrepancies occur. This discrepancy is a result of low plasma flushing efficiency at short pulse on-times and material resolidification at long pulse on-times. Therefore an improved EDM process model was developed to include the reduce erosion rate factor [RERF] that accounts for the low plasma flushing efficiency at short pulse on-times and material resolidification at long pulse on-times. The [RERF] factor was constructed from analysis of experimental results. Using the improved EDM process model, the predicted material removal rate shows better agreement with the experimental results.

CHAPTER 7

CONCLUSION AND SUGGESTIONS FOR FURTHER RESEARCH

This chapter concludes the study and discusses some suggestions for future research. The overall research activities are emphasized and several limitations are also highlighted.

7.1 CONCLUSION

The research presented in this thesis proposes a model of the complete Electro Discharge Machining (EDM) system and the design and implementation of a digital controller for the servomotor control and the gap voltage and current pulse power generator. The complete EDM system model consists of two sub models, namely an EDM process model and the servo system model. The EDM process model was developed using a dimensional analysis technique and the servo system model was developed using the differential equations of Newton's and Kirchhoff's laws. The complete EDM system model was used in a Matlab/Simulink simulation to investigate the EDM system model behaviour. The results of the simulation were used to aid in the design of the compensated EDM control system. Simulation studies were then carried out to predict the material removal rate in mm^3/min.

The design and development of the digital EDM control system were performed mainly in software with minimal hardware. The control software was designed using the structure programming methodology that combines a flowchart and program structure diagrams for clear description of program code. EDM control processes such as timing control t_{on}, t_{off}, hysteretic current control I_{gap} and servo system control t_{up}, t_{down} are implemented in software. The control hardware consists of an eZdsp, user-interface device and analogue signal processing and interfacing circuit. The eZdsp communicates to the user-interface device by sending the information/instruction to the LCD screen while the user-interface device uses push-button switches to communicate with the eZdsp. The software implementation for the user-interface device and EDM process parameter selection is designed using a Finite State Machine FSM, thus minimising the number of switches on the user-interface device. Digital EDM parameter selection using a keypad would be an ideal and practical way, but was not used because the eZdsp's I/O has been fully utilised to accommodate all the necessary signals from/to the eZdsp. The eZdsp uses 3.3V low voltage for operation but requires additional components for signal compatibility between the user-interface device and the eZdsp.

The analogue EDM signal processing circuit processes motor current, velocity and gap width feedback signals. The output of this circuit V_c is the control signal for the digital servomotor PWM controller. Only the analogue control signal V_c is digitised in order to obtain the highest possible resolution with the 10-bit eZdsp's A/D. The analogue signal processing and interfacing circuit was implemented using operational amplifier circuitry. The signals from the eZdsp to the power drive circuit were opto-coupled to isolate the voltage levels between these devices. It has been shown in this research that a single DSP microcontroller can be used to provide the control functions for the EDM system.

The experimental studies of the EDM process using a copper electrode, a graphite electrode and steel workpiece materials were presented in tabular and graphical form. Analysis of the experimental results shows that the material removal rate is influenced by the process parameters such as the gap current I_{gap}, gap voltage V_{arc}, pulse on time t_{on} and sparking frequency F_s as well as the material properties of the electrode and the workpiece. In comparison studies, simulation results of material removal rate show reasonable agreement with experimental results, except at short and long pulse on-times. The discrepancy between the simulation and experimental results is due to the low plasma flushing efficiency at short pulse on-times and material resolidification at long pulse on-times. To correct the discrepancy, a reduction erosion rate factor [RERF], developed from regression analysis, that accounts for the low plasma flushing efficiency at short pulse on-times and material resolidification at long pulse on-times, is introduced into the EDM process model. As a result, predicted material removal rate using the improved EDM process model shows acceptable agreement with experimental results.

7.2 SUGGESTIONS FOR FURTHER RESEARCH

It has been demonstrated that the research objectives have been achieved throughout the research work. However, there are several suggestions that can be addressed by further research such as:

1. The present controller is not programmed to detect the presence of an arc when machining. Some form of arc detection/prevention system should be investigated in order to prevent arcs from damaging the workpiece. Such a system could be implemented via a fuzzy logic or neural network controller.

2. Although the user has the ability to control EDM gap width, no indication is given to the user concerning gap width. A visual of display of gap width could be implemented indicating too large gap, too small gap and ideal gap.

3. Since removal rate is proportional to pulse current, a higher power gap voltage and current pulse power generator could be developed.

4. Investigate a linear motor servo system to replace the DC servomotor. It is acknowledged that the linear motor is superior in positioning system performance but expensive when compared to the DC servomotor. A better positioning system would lead to a stable process, resulting in a higher material removal rate.

All the above suggestions would enhance the performance of the EDM system but would be subject to cost constraints.

REFERENCES

[1] Different types of machining processes that uses EDM 2004 (www.edmtt.com/articlesreports/edmprocesses.html)

[2] Dimov S, Pham D T, Ivanov A and Popov K 2003 Tool-path generation system for micro-electro discharge machining milling *Proc. Instn Mech. Engrs* vol. 217 part B, pp. 1633 – 1637

[3] Bleys P, Kruth J P and Lauwers B 2004 Monitoring and control of electrode wear in Electrical Discharge Milling *The VIIth International Conference on Monitoring and Automatic Supervision in Manufacturing AC'04 (Poland)*

[4] Center for non-traditional manufacturing research (www.unl.edu/nmrc/Diesinking/edmill/edmilling.htm)

[5] Vectaspark Electro discharge grinding system www.vectaspark.com

[6] Jain V K and Mote R G 2004 On the temperature and specific energy during electrodischarge diamond grinding (EDDG) *Advanced Manufacturing Technology*

[7] Brown J 1998 Advance machining technology handbook (*New York: London; McGraw-Hill*)

[8] A reference to understanding, selecting and using wire on wire-cut EDM machines 1995 *Intech EDM* (www. Intech-edm.com)

[9] Petrovic D, Chatzittheodoridis E, Popovic G, Medico O D, Almansa A, Brenner W, Detter H, Martins R and Fortunato E 2001 Design of a mechanical gripper for assembly of microparts *XXX Convegno Nazionale AIAS-Alghero (SS)*

[10] Aoyama S, Tamura K, Sato T, Kimura T, Sawahata K and Nagai T 1999 High-performance coated wire electrodes for high-speed cutting and accurate machining *Hitachi Cable Review* no.18 pp. 75-80

[11] Kuroda H, Aoyama S, Kimura T, Sawahata K and Sato T 2003 Development of high-performance coated wire electrodes for high-speed cutting and accurate machining *Hitachi Cable Review* no.22 pp. 51-56

[12] Lauwers B, Kruth J P, Bleys Ph, Coppenolle V, Stevens L and Derighetti 1998 Wire rupture prevention using on-line pulse localisation in WEDM *12th Int. Symp. for Electromachining (Germany)*, pp 203-209

[13] Andrzej N and Zbigniew S 2004 Using random sequences for Wedm process states discrimination *The VIIth International Conference on Monitoring and Automatic Supervision in Manufacturing AC'04 (Poland)*

[14] Dibinto D D, Eubank P T, Patel M R and Barrufet M A 1989 Theoretical models of the electrical discharge machining process. I. A simple cathode erosion model *J. Appl. Phys.* 66 4095-4103

[15] Ravi N and Huang H 2002 Fabrication of symmetrical section microfeatures using the electro-discharge machining block electrode method *J. Micromech. Microeng. vol. 12 pp. 905-910*

[16] Her M G and Weng F T 2001 Micro-hole machining of copper using the electrode-discharge machining process with a tungsten carbide electrode compared with a copper electrode *Int. J. Adv. Manuf. Technol. vol. 17 pp. 715-719*

[17] Weng F T and Her M G 2002 Study of the batch production of micro parts using the EDM process *Int. J. Adv. Manuf. Technol. 19 266-270*

[18] Sanchez J A, Cabanes I, Lopez L N and Lamikiz A 2001 Development of optimum electrodischarge machining technology for advanced ceramics *Int. J. Adv. Manuf. Technol. vol. 18 pp. 897-905*

[19] Reynaerts D, Meeusen W, Brussel H V, Reyntjens S and Puers R 1999 Production of seismic mass suspensions in silicon by electro-discharge machining *J. Micromech. Microeng. vol. 9 pp. 206-210*

[20] Weng F T 2004 Fabrication of microelectrodes for EDM machining by a combined etching process *J. Micromech. Microeng. 14 N1-N4*

[21] Stampfl J, Leitgeb R, Cheng Y L and Prinz F B 2000 Electro-discharge machining of mesoscopic parts with electroplated copper and hot-pressed silver tungsten electrodes *J. Micromech. Microeng. vol. 10 pp. 1-6*

[22] Operating Manual Joemars machinery electric industrial co. ltd.

[23] www.automatededm.com/wire_edm.html

[24] Meeusen W, Reynaerts D, Peirs J, Brussel H V, Dierickx V and Driesen W 2001 The machining of freeform micro moulds by Micro EDM; work in progress *The 12th MicroMechanics Europe Workshop, Cork*

[25] Ravi N and Chuan S X 2002 The effects of electro-discharge machining block electrode method for microelectrode machining *J. Micromech. Microeng. vol. 12 pp. 532-540*

[26] Arthur A, Dickens P M and Cobb R C 1996 Using rapid prototyping to produce electrical discharge machining electrodes *Rapid Prototyping Journal* Vol. 2 pp. 4-12

[27] Arthur A and Dickens P M 1998 The measurement of heat distribution in stereolithography electrodes during electro-discharge machining *Int. J. Prod. Res.* Vol. 36 no. 9 pp. 2451-2461

[28] Narasimhan J, Yu Z and Rajurkar K P 2004 Tool wear compensation and path generation in micro and macro EDM *Transactions of NAMRI/SME* Vol. 32 pp. 1-8

[29] BP Technical Brochure 1988 Dielectric fluids for Electro-Discharge Machining, United Kingdom, MP 639/88

[30] Ekmekci B, Elkoca O, Koltuk F, Tekkaya A E and Erden A 2002 Metallurgical properties of electric discharge machined surfaces *6th Biennial Conference on Engineering Systems Design and Analysis, Turkey*

[31] Wang W M and Rajurkar K P 1990 Monitoring, Modeling and control of EDM *Mon. and ctrl. For manfc. Proc., ASME PED* Vol. 44 pp. 393-406

[32] Technical Brochure 1999 Important facts about spark erosion OEL-HELD GmbH http://www.oel-held.de

[33] Ekmekci B, Elkoca O, Tekkaya A E and Erden A 2002 Observations on cracking behaviour of micro alloy steel in Electric Discharge Machine (EDM) *The Tenth International Conference on Machine Design and Production, Turkey*

[34] Peirs J, Reynaets D, Verplaetsen F, Norman F and Lefever S 2003 Development of a micro gas turbine for electric power generation *The 14th MicroMechanics Europe Workshop, Netherlands*

[35] Yeo S H and Tan L K 1999 Effects of ultrasonic vibrations in micro electro-discharge machining of microholes *J. Micromech. Microeng. 9 345-352*

[36] Huang H, Zhang, Zhou L and Zheng H Y 2003 Ultrasonic vibration assisted electro-discharge machining of microholes in Nitinol *J. Micromech. Microeng. 13 693-700*

[37] Yu Z Y, Rajurkar K P and Tandon A 2004 Study of 3D micro-ultrasonic machining *Journal of Manufacturing Science and Engineering vol. 126 pp. 727-732*

[38] Patel M R, Barrufet M A, Eubank P T and Dibinto D D 1989 Theoretical models of the electrical discharge machining process. II. The anode erosion model *J. Appl. Phys.* 66 4104 – 4111

[39] Qu J, Shih A J and Scattered R O 2002 Development of the cylindrical wire electrical discharge machining process, part 1: Concept, Design, and material removal rate *Journal of Manufacturing Science and Enginnering* Vol. 124 pp. 702-707

[40] Altpeter F, Cors J and longchamp R 1998 EDM modeling for control *12th Int. Symp. for Electromachining (Germany)* , pp 149 – 155

[41] Langhaar HL 1951 Dimensional Analysis and Theory of Models *(New York: Wiley)*

[42] Szirtes T 1997 Applied Dimensional Analysis and Modeling *(New York: McGraw-Hill)*

[43] Douglas J F 1969 An introduction to dimensional analysis for engineers *(London: Sir Isaac Pitman & Sons Ltd.)*

[44] Buntat Z, Harry J E and Smith I R 2003 Application of dimensional analysis to ozone production by pulsed streamer discharge in oxygen *J. Phys. D: Appl. Phys.* 36 1553-1557

[45] Kocher M and Guglielmetti P 1999 Modelling and simulation of an EDM die sinking machine *Proc. ISEM XI (Lausanne) pp 105-113*

[46] Sigmaplot software www.systat.com/products/SigmaPlot/

[47] Kuo B C 1995 Automatic control systems *(Prentice Hall, Englewood Cliffs, New Jersey 07632)*

[48] Lyshevski S E 2001 Control system theory with engineering applications *(Birkhauser Boston)*

[49] Charles L P and Royce D H 2000 Feedback control systems *(Prentice-Hall)*

[50] Ramsbotham P 2005 Speed and position control of a *DC* Servomotor *Loughborough University Final Year Project Report*

[51] Kelly R 2000 A measurement procedure for Viscous and Coulomb friction *IEEE Trans. On Instr. And Meas.* Vol. 49, no. 4, pp. 857-861

[52] Fusing mechanics and electronics to design intelligent machines www.mechatronics.me.vt.edu/book/Section3/motormodelling.html

[53] Ge S S, Lee T H and Ren S X 2001 Adaptive friction compensation of servo mechanisms *International Journal of Systems Science vol. 32, no. 4, pp.523-532*

[54] Cho Y H, Lee W S and Kim J H 2003 Performance evaluation of nonlinear character friction control *ICCAS2003 Korea* pp. 2551-2554

[55] Oreb B F, Duffy R M and Steel W H 1984 Motorised focusing for optical benches *J. Phys. Sci. Instrum.* vol. 17, pp. 193-195

[56] LinTech Technical report positioning systems www.lintechmotion.com

[57] The smart motion cheat sheet www.electromate.com

[58] Zhang Y 1997 High performance DSP-based servo drive control for a limited-angel torque motor *PhD Thesis, Department of Electronic and Electrical Engineering, Loughborough University*

[59] Neulinger C 1995 High speed laser cutting machine for thin materials *MSc Thesis, Department of Mechanical Engineering, Loughborough University*

[60] Dorf R C and Bishop R H 1995 Modern control systems *7th Edition, Addison-Wesley Publishing Company, Inc.*

[61] Fairchild application note 2004 A guide to the design of current feedback control www.fairchild.com

[62] Franklin G F and Powell J D 1994 Feedback control of dynamic systems *Addison-Wesley Publishing Company.*

[63] Lecture of control theory www.leotechion.ac.il/courses/CT/Lectures/lect10.pdf

[64] Ibrahim Z 1999 Fuzzy logic control of PWM inverter-fed sinusoidal permanent magnet synchronous motor drives *Ph.D thesis, Liverpool John Moores University, UK.*

[65] eZdsp™ LF2407 (Technical reference) - http://www.ti.com/sc/docs/products/dsp

[66] TMS320LF2407 2000 (Evaluation module) - http://www.ti.com/sc/docs/products/dsp

[67] Cooling J E 1997 Real-time software systems, an introduction to structured and object-oriented design *PWS Publishing Company.*

[68] Code composer user's guide (Spru296) - http://www.ti.com/sc/docs/products/dsp

[69] TMS320LF/LC240xA DSP controllers reference guide, System and peripherals (Spru357a) - http://www.ti.com/sc/docs/products/dsp

[70] Katz R H 1993 Contemporary logic design *Prentice Hall*

[71] TMS320C24x PWM full compare in symmetric mode (Spra369) - http://www.ti.com/sc/docs/products/dsp

[72] Xiao J, Dulimarta H, Xi N, Tummala R L 2000 Controller design for an autonomous Wallclimbing Micro-robot based on TI 320LF2407 DSP chip *DSPS Fest 2000 (Texas)*

[73] Xiao J, Yu Z, Dulimarta H, Xi N, Tummala R L 2000 DSP solution for Wall-climber Micro-robot control using TMS320LF2407 chip *The 43rd Midwest Symposium on Circuits and Systems (Michigan)*

[74] Behrens A and Witzak M P 1998 An hierarchical process control system for highly efficient electro-discharge machining *12th Int. Symp. for Electromachining (Germany)*, pp 149 – 155

[75] Sommer C 2001 EDM electrodes – a matter of choosing wisely, different materials affect machining http://www.manufacturingcenter.com

APPENDIX 1

Dimension of parameters

Variables	**Symbols**	**Dimensions**
Material removal rate	$\dot{V}$	m^3s^{-1}
Pulse on-time	t_{on}	s
Gap voltage	V_{arc}	$m^2\,kg\,s^{-3}A^{-1}$
Sparking frequency	F_s	s^{-1}
Gap current	I_{gap}	A
Material properties factor	α	$m\,kg^{-1}s^2$

APPENDIX 2

Matlab m-file

% EDM servomotor parameters
L_a = 3e-3; % inductance (H)
R_a = 2.85; % resistance (ohm)
K_t = 0.1356; % torque constant (Nm/A)
K_e = 0.1356; % back emf constant (V.s/rad)
J_m = 0.2255e-3; % motor inertia
J_t = 0.2545 e-3; % motor and mechanical load inertias (kg-m^2)
K_f = 0.268e-3; % viscous friction coefficient of linear model (Nm.s/rad)
K_{fnl} = 9.66e-5; % viscous friction coefficient of nonlinear model (Nm.s/rad)
K_{fl} = 40.6264e-3; % viscous friction coefficient of model with load (Nm.s/rad)
N_{gm} = 15; % number of gear teeth at motor
N_{gl} = 60; % number of gear teeth at load
d_{ls} = 0.005; % leadscrew pitch (m/rev)
K_{tac}=0.06684; % tacho coefficient (V.s/rad)

% Current control loop PI parameters
K_{pc} = 12; % proportional gain
K_{ic} = 11400; % integrator gain
K_{wc} = 44; % anti wind-up gain

% Velocity control loop PI parameters
K_{pv} = 5; % proportional gain
K_{iv} = 111; % integrator gain
K_{wv} = 44; % anti wind-up gain

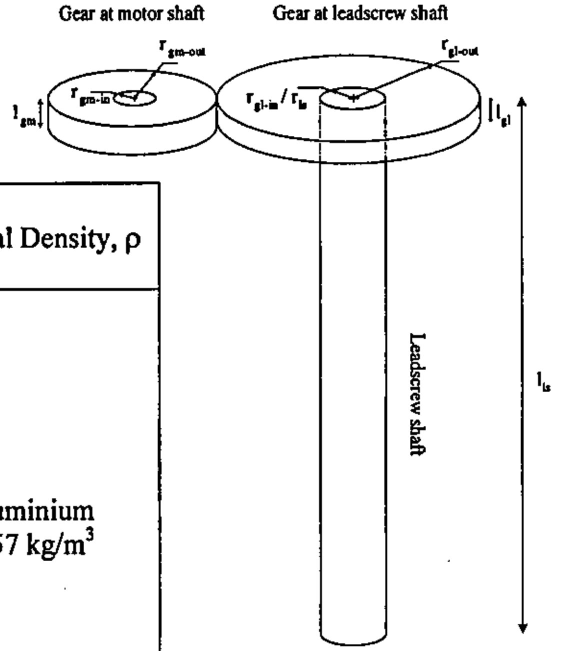
% Average gap voltage control loop P parameters
K_{avg} = 0.31; % proportional gain

% EDM process parameters
% Values are changeable for different processes. The following values are chosen for
% tuning the average gap voltage controller parameter.
V_{arc}= 25; % gap voltage
V_{max}= 160; % maximum voltage
t_{on} = 110e-6; % pulse on-time
t_{off} = 4.2e-6; % pulse off-time
I_{gap} = 25; % gap current
A_p = 2e-12; % α, material removal properties
C = 1.74; % dimensionaless constant
A = 0.01; % surface area for cylindrical electrode
n = 6.57; % parameters for breakdown model
v = 1.04e+25;
τ =15.5e-3; % time constant filter

APPENDIX 3

Mechanical system inertias calculation

Consider a solid cylinder of leadscrew shaft and a hollow cylinder of gear at motor shaft and at leadscrew shaft.



Component	Measured Parameter	Material Density, ρ
Gear at motor shaft	Length, l_{gm} 0.038m Radius, $r_{gm\text{-}in}$ 0.005m $r_{gm\text{-}out}$ 0.01m No. of teeth, N_{gm} 15	Aluminium 2657 kg/m^3
Gear at leadscrew shaft	Length, l_{gl} 0.02m Radius, $r_{gl\text{-}in}$ 0.0075m $r_{gl\text{-}out}$ 0.047m No. of teeth, N_{gl} 60	
Leadscrew	Length, l_{ls} 0.37m Radius, r_{ls} 0.0075m Screw nut efficiency, e_s 0.5 Length / rev, d_{ls} 0.005m/rev (rev is dimensionless)	Steel 7833kg/m^3
Ram load	Weight, w 30kg	

The total equivalent inertia J_t as seen by the EDM DC motor is given in the following equation (2.29);

$$J_t = (J_m + J_{gm} + J_{gl\mapsto m} + J_{ls\mapsto m} + J_{rl\mapsto m}) \text{ kg-m}^2 \qquad (2.29)$$

where,

J_m, inertia of the DC motor

J_{gm}, inertia of the gear at motor shaft

$J_{gl\mapsto m}$, reflected inertia of gear at leadscrew shaft load to the DC motor

$J_{ls\mapsto m}$, reflected inertia of the leadscrew to the DC motor

$J_{rl\mapsto m}$, reflected inertia of the ram load to the DC motor

Inertia of the DC motor, J_m;

$$J_m = 0.2255\times10^{-3}\text{kg-m}^2 \text{ (data sheet value)}$$

Inertia of the gear at motor shaft, J_{gm};

$$\begin{aligned} J_{gm} &= \frac{\pi\times l_{gm}\times\rho_a\times\left(r_{gm-out}{}^4 - r_{gm-in}{}^4\right)}{2} \\ &= \frac{\pi\times0.038\text{m}\times2657\,\text{kg/m}^3\times\left(0.01^4-0.005^4\right)\text{m}^4}{2} \\ &= 1.48\times10^{-6}\,\text{kg-m}^2 \end{aligned}$$

Reflected inertia of gear at leadscrew shaft load to the DC motor, $J_{gl\mapsto m}$;

$$J_{gl\mapsto m} = \left(\frac{N_{gm}}{N_{gl}}\right)^2 \times \frac{\pi \times l_{gl} \times \rho_a \times \left(r_{gl-out}{}^4 - r_{gl-in}{}^4\right)}{2}$$

$$= \left(\frac{15}{60}\right)^2 \times \frac{\pi \times 0.02\text{m} \times 2657\,\text{kg/m}^3 \times \left(0.047^4 - 0.0075^4\right)\text{m}^4}{2}$$

$$= 2.5441 \times 10^{-5}\,\text{kg-m}^2$$

Reflected inertia of the leadscrew to the DC motor, $J_{ls\mapsto m}$;

$$J_{ls\mapsto m} = \left(\frac{N_{gm}}{N_{gl}}\right)^2 \times \frac{\pi \times l_{ls} \times \rho_s \times r_{ls}{}^4}{2}$$

$$= \left(\frac{15}{60}\right)^2 \times \frac{\pi \times 0.37\text{m} \times 7833\,\text{kg/m}^3 \times 0.0075^4\,\text{m}^4}{2}$$

$$= 0.9002 \times 10^{-6}\,\text{kg-m}^2$$

Reflected inertia of the ram load to the DC motor, $J_{rl\mapsto m}$;

$$J_{rl\mapsto m} = \left(\frac{N_{gm}}{N_{gl}}\right)^2 \times w \times \left(\frac{d_{ls}}{2 \times \pi}\right)^2$$

$$= \left(\frac{15}{60}\right)^2 \times \left(\frac{30\text{kg}}{0.9}\right) \times \left(\frac{0.005\,\text{m/rev}}{2 \times \pi}\right)^2$$

$$= 1.1873 \times 10^{-6}\,\text{kg-m}^2$$

Thus,

$$J_t = [(0.2255 \times 10^{-3}) + (1.4868 \times 10^{-6}) + (2.5441 \times 10^{-5}) + (0.9002 \times 10^{-6}) + (1.1873 \times 10^{-6})]\ \text{kg-m}^2$$

$$= 0.25 \times 10^{-3} \text{ kg-m}^2$$

Load Torque

The load torque reflected to the DC motor from the mechanical system is calculated by considering torques due to sliding friction and gravity at the load side. The load torque at the load side of the gear load is given by;

$$T_{l \to s} = \left(\frac{d_{ls} \times (F_g + F_{fr})}{2\pi \times e_s} \right) + T_p \qquad (2.30)$$

where;

F_g, force due to gravity = $W_l \times \sin(\gamma)$

F_{fr}, force due to sliding friction = $\mu . \times W_l \times \cos(\gamma)$

T_p, preload torque = 0 (zero in this case, Nm)

W_l, weight of load (ram and electrode, N)

μ, coefficient of sliding friction

d_{ls}, lead of screw (m/rev)

γ, angle of screw inclination from horizontal (degrees)

e_s, efficiency of screw nut

Since the leadscrew in the physical system is situated in the vertical plane, γ is equal to 90^0. Therefore the force due to sliding friction, F_{fr} becomes zero and the force due to gravity is at maximum;

$$F_g = W_l (30\text{kg} \times 9.8 \text{ m/s}^2) \times \sin (90^0)$$
$$= 294 \text{ N}$$

Thus, the load torque at the load side of the gear load, $T_{l \to s} = \frac{0.005 \times 294}{2 \times \pi \times 0.5} = 0.46\text{Nm}$ (un-reflected to the motor). The following equation is applied in order to obtain the reflected load torque;

$$T_{refl} = \frac{T_{l \to s}}{\left({}^{N_{gl}}\!/\!_{N_{gm}} \right) \times e_g} \qquad (2.31)$$

where;

e_g, efficiency of gear system (95%)

Therefore;

$$T_{refl} = \frac{T_{l \to s}}{\left({}^{N_{gl}}\!/\!_{N_{gm}} \right) \times e_g} = \frac{0.4679}{4 \times 0.95} = 0.123\text{Nm}$$

APPENDIX 4

EDM program codes

```
/*****************************************************************
*       Filename: edm.c                                          *
*                                                                *
*       Author: Azli Yahya, Loughborough university              *
*                                                                *
*       Description: Main program for eZdsp                      *
*                                                                *
*****************************************************************/

/*** Address Definitions ***/
#include        "f2407_c.h"
#include        "edm1.h"

/*** Constant Definitions ***/

/*** LCD Definitions ***/

unsigned int a, b, c,j, d, q,f,v,w=0,PF;
static unsigned char ai=0, aj=0;
int t_on, t_off,t_up ,t_down;
int state=0;
int cnt1=0;
int count_on=0,count_1a=1,count_2a=1,count_3a=1,count_4a=1;
int count_off=0,count_1b=1,count_2b=1,count_3b=1,count_4b=1;
int count_current=0,count_1c=1,count_2c=1;
int count_up=0,count_1d=1,count_2d=1,count_3d=1,count_4d=1;
int count_down=0,count_1e=1,count_2e=1,count_3e=1,count_4e=1;

int value_1a=0,value_2a=0,value_3a=0,value_4a=0;
int value_1b=0,value_2b=0,value_3b=0,value_4b=0;
int value_1c=0,value_2c=0;
int value_1d=0,value_2d=0,value_4d=0;
int value_1e=0,value_2e=0,value_4e=0;

#define  period                 0
#define pwm_duty_0              0
#define timer2_per          3125        /* 10ms timer2 period with a 1/64 timer prescaler and
                                        40MHz CPUCLK */
#define motor_stop             500
#define motor_reverse  100    /* 10% duty cycle*/
```

```
void main(void)
{
        dsp_setup();
        lcd_display_main();     /*use Final State Machine concept */

   while(1){  /*continuous loop program */
                lcd_display_function(); /* call pushbutton LCD function */

                if(w==1){
                        faulty();           /* function to detect the switch failure */
                        hysteretic_current();}

                if(w==2)manual_mode();    }
                }


/*****************  INTERRUPT SERVICE ROUTINES    *******************/

interrupt void timer2_isr(void) /* generate up / down time */
{                       /* timer2 interrupt every 1mS */
        ai++;
        if(ai==100){
                aj++;
                ai=0;
                if(aj<=t_up)
                        reverse_motor();
        else if((t_up+1)<=aj && aj<=(t_down+t_up)){
                        if(sense_pushbutton(0x0080)==0)CMPR1=500; /*if hits the limit,
                                                                        motor stop */
                        else error_controller(); }
                if(aj==(t_down+t_up))aj=0; }

        EVAIFRB = EVAIFRB & 0x0001;    /* clear T2PINT flag */ }

void manual_mode(){
        T1PR=1000;    /*set timer period*/
        MCRA = 0x00C0;  /*select as pwm*/
        if(sense_pushbutton(0x0010)){
                if(sense_pushbutton(0x0040)==0)CMPR1=500;
                else{   ACTRA = 0x0009;         /*forward pwm */
                        CMPR1=(float)(0.48828125*ReadADC(0)+500);}}  /*eqn refers
                                        to 0v=500CMPR1 & 3.3V=1000CMPR1*/

        else if(sense_pushbutton(0x0004)){
                if(sense_pushbutton(0x0080)==0)CMPR1=500;
                else{   ACTRA = 0x0006;         /*reverse pwm for reverse drive */
                        CMPR1=(float)(0.48828125*ReadADC(0)+500);}}
        else CMPR1=500;  /*no press means motor stop hence 50% duty cycle */
        state=17;}         /*after completing the manual cmd, go back to */
                           /*start the system or change setup*/

void hysteretic_current(){
        int Iref_a, H_limit_d, L_limit_d;         /* define high and low limit constants */
        float d_limit, I_d, H_limit, L_limit,Iref_d;

        Iref_a = (value_1c*10)+value_2c;          /* set I ref */

        d_limit = (float)Iref_a*(0.1);            /*calculate limit digitally */
        I_d = (float)d_limit/2;
        H_limit = (float)Iref_a + I_d;
        L_limit = (float)Iref_a - I_d;
        H_limit_d = (float)H_limit*18.804363;
        L_limit_d = (float)L_limit*18.804363;

        if (ReadADC(0) < L_limit_d) PEDATDIR = 0xFF01;     /* on Q5 */
        else if (ReadADC(0) >= H_limit_d) PEDATDIR = 0xFF00; /* off Q5 */
        }

void faulty(){
        if ((sense_pushbutton(0x0040)&&sense_pushbutton(0x0080)
                &&sense_FF(0x0001))==0)
                        {        /* ..if any of the switches goes low, led off */
                                 /* LED indicates switch fail   */

                                 lcd_cmd(0xFF01);       /*clear display command*/
                                 state=12;
                                 delay_mS(80);
                                 fault_system();
                                 w=0;
                                 setting_off();    /* Thus, start button must be pressed again*/
                                                   } /*out from the main loop */
                                 else w=1;}

void lcd_display_main(){
        PFDATDIR=0xFF08;
        delay_mS(50);
        lcd_initialize();
        lcd_cmd(0xFF0C);      /* Lcd blink */
        lcd_cmd(0xFF0E);      /* cursor on */
        display_edm_controller();        /*state=0*/
        state=0;}

void run_system(){ /*get ton, toff, up, down time digitally */
        t_on = (value_1a*1000)+(value_2a*100)+(value_3a*10)+value_4a;
        t_off = (value_1b*1000)+(value_2b*100)+(value_3b*10)+value_4b;
        t_up = (value_1d*100)+(value_2d*10)+(value_4d*1);
        t_down = (value_1e*100)+(value_2e*10)+(value_4e*1);
        generate_Q6_Q7();}/*run */

void setting_off(){
        MCRA = 0x00C0;          /*select as pwm */
        T1PR=1000;              /*motor hold */
        CMPR1=500;
        T2PR=0;                 /*off timer 2 & 3*/
        T3PR=0;
        count_1();
        ai=0;                   /*ai & aj is declared globally to reset these counter (ai */
        aj=0;}                  /* & aj) after stop button is pressed.*/

void setting_on(){
        MCRA = 0x00C0; /*return to select pwm*/
        T1PR=1000;
        T2PR=20000;}/*on timer1 & 2*/

void error_controller(){
        ACTRA = 0x0006;
        CMPR1= (float)(0.9765625*ReadADC(2)); /* varies according to error*/          }

void reverse_motor(){
        ACTRA = 0x0006; /*reverse pwm for reverse drive */
        CMPR1=(float)(0.48828125*ReadADC(0)+500);          }

void count_1(){
        count_on=0;
        count_off=0;
        count_current=0;
        count_up=0;
        count_down=0;}

void pwm_0(){  /* SCSR1 must be masked/set to 0x06FD for correct freq*/
        T1PR = 0;  /* This apllies to all pwm_xx */
        CMPR1 = 0;
        }

void lcd_cmd(int cmd){  /* LCD initializations*/
        delay_mS(6);
        PFDATDIR = 0xFF0C;/*PFDATDIR | 0xFF04;   /* RS = 0 RW= 0 E =1 */
        PBDATDIR = cmd;        /* command - initialize display */
        PFDATDIR = 0xFF09;/*PFDATDIR | 0xFF01;   /* RS=1 E =0  */
        }

void lcd_data(int data){
```

```
                PFDATDIR = 0xFF0D;/*PFDATDIR | 0xFF05;  /* E=1 RS=1 */
                PBDATDIR = data;  /* data E */
                PFDATDIR = 0xFF09;/*PFDATDIR | 0xFF01;  /* RS = 1  E=0*/
                delay_mS(6);}

void lcd_initialize() {
        lcd_cmd(0xFF3C);                /* Function set 8-bit, 2 lines, 5x10 dots */
        lcd_cmd(0xFF08);                /* Lcd off  ) */
        lcd_cmd(0xFF01);                /* clear display ) */
        lcd_cmd(0xFF0C);        /* Lcd on  ) */
        lcd_cmd(0xFF02);                /* Lcd home ) */
        lcd_cmd(0xFF0C);        /* Lcd blink off ) */
        delay_mS(5);}


void cursorposition(int c){
        /* 80h is the address for cursor position. 80h plus address LCD
        of 01h,where at start of 1st line. Thus=81h. */
        PFDATDIR = 0xFF0C;/*PFDATDIR | 0xFF04;   /* RS = 0 RW= 0 E =1 */
        PBDATDIR = c;           /* command - cursor position to 1st line, 2nd block */
        PFDATDIR = 0xFF09;/*PFDATDIR | 0xFF01;   /* RS=1 E =0  */
        delay_mS(5);    }

/* LCD subroutines functions */
void time(){
        lcd_data(0xFF74); /*time*/
        lcd_data(0xFF69);
        lcd_data(0xFF6D);
        lcd_data(0xFF65);
        lcd_data(0xFF3A);}

void setup(){
        lcd_data(0xFF53); /*Setup*/
        lcd_data(0xFF65);
        lcd_data(0xFF74);
        lcd_data(0xFF75);
        lcd_data(0xFF70);}

void uS(){
        lcd_data(0xFF5F); /*____uS*/
        lcd_data(0xFF5F);
        lcd_data(0xFF5F);
        lcd_data(0xFF5F);
        lcd_data(0xFFE4);
        lcd_data(0xFF73);}

void pge(){
        lcd_data(0xFF50); /*pge*/
        lcd_data(0xFF67);
        lcd_data(0xFF65);
        lcd_data(0xFF7E); }

void nxt(){
        lcd_data(0xFF4E); /*nxt*/
        lcd_data(0xFF78);
        lcd_data(0xFF74);
        lcd_data(0xFF7E); }

void inc(){
        lcd_data(0xFF49); /*inc*/
        lcd_data(0xFF6E);
        lcd_data(0xFF63);
        lcd_data(0xFF7E);}

void on(){
        lcd_data(0xFF4F); /*on*/
        lcd_data(0xFF6E); }

void off(){
        lcd_data(0xFF4F); /*off*/
        lcd_data(0xFF66);
        lcd_data(0xFF66);}

void Stop(){
        lcd_data(0xFF53); /*Stop*/
        lcd_data(0xFF74);
        lcd_data(0xFF6F);
        lcd_data(0xFF70);}

void Current(){
        lcd_data(0xFF43); /*Current*/
        lcd_data(0xFF75);
        lcd_data(0xFF72);
        lcd_data(0xFF72);
        lcd_data(0xFF65);
        lcd_data(0xFF6E);
        lcd_data(0xFF74);
        lcd_data(0xFF3A); }

void amps(){
        lcd_data(0xFF5F); /*amps*/
        lcd_data(0xFF5F);
        lcd_data(0xFF41);
        lcd_data(0xFF6D);
        lcd_data(0xFF70);
        lcd_data(0xFF73); }
```

```
void Press(){
        lcd_data(0xFF50); /*Press*/
        lcd_data(0xFF72);
        lcd_data(0xFF65);
        lcd_data(0xFF73);
        lcd_data(0xFF73);}

void Run(){
        lcd_data(0xFF52); /*Run*/
        lcd_data(0xFF75);
        lcd_data(0xFF6E);}

void Up(){
        lcd_data(0xFF55); /*Up*/
        lcd_data(0xFF70); }

void Down(){
        lcd_data(0xFF44); /*Down*/
        lcd_data(0xFF77);
        lcd_data(0xFF6E); }

void sec(){
        lcd_data(0xFF5F);  /*sec*/
        lcd_data(0xFF5F);
        lcd_data(0xFF2E);
        lcd_data(0xFF5F);
        lcd_data(0xFF73);
        lcd_data(0xFF63);}

void view(){
        lcd_data(0xFF56);       /*View*/
        lcd_data(0xFF69);
        lcd_data(0xFF65);
        lcd_data(0xFF77);}

void system(){
        lcd_data(0xFF53);       /*system*/
        lcd_data(0xFF79);
        lcd_data(0xFF73);
        lcd_data(0xFF74);
        lcd_data(0xFF65);
        lcd_data(0xFF6D);}

void error(){
        lcd_data(0xFF45);       /* error*/
        lcd_data(0xFF72);
        lcd_data(0xFF72);
        lcd_data(0xFF6F);
        lcd_data(0xFF72);}

void input(){
        lcd_data(0xFF49);       /*input */
        lcd_data(0xFF6E);
        lcd_data(0xFF70);
        lcd_data(0xFF75);
        lcd_data(0xFF74);}

void menu(){
        lcd_data(0xFF4D); /*Menu*/
        lcd_data(0xFF65);
        lcd_data(0xFF6E);
        lcd_data(0xFF75);}

void manual(){
        lcd_data(0xFF4D); /*Manual*/
        lcd_data(0xFF61);
        lcd_data(0xFF6E);
        lcd_data(0xFF75);
        lcd_data(0xFF61);
        lcd_data(0xFF6C);}

void control(){
        lcd_data(0xFF43); /*control*/
        lcd_data(0xFF6F);
        lcd_data(0xFF6E);
        lcd_data(0xFF74);
        lcd_data(0xFF72);
        lcd_data(0xFF6F);
        lcd_data(0xFF6C);}

void fault(){                   /*fault */
        lcd_data(0xFF22);
        lcd_data(0xFF46);
        lcd_data(0xFF41);
        lcd_data(0xFF55);
        lcd_data(0xFF4C);
        lcd_data(0xFF54);
        lcd_data(0xFF59);
        lcd_data(0xFF21);
        lcd_data(0xFF22);}

void reset(){                   /*reset */
        lcd_data(0xFF52);
        lcd_data(0xFF73);
        lcd_data(0xFF74);
        lcd_data(0xFF3F);}

void fault_system(){/*"FAULTY"*/
        delay_mS(2);
        cursorposition(0xFF83);
        system();
        cursorposition(0xFF8A);
        Stop();
        cursorposition(0xFFC0);
        menu();
        cursorposition(0xFFC6);
        fault();}

void display_edm_controller() {
        delay_mS(4);  /*200*/
        cursorposition(0xFF81);
        lcd_data(0xFF45);   /*EDM*/
        lcd_data(0xFF44);
        lcd_data(0xFF4D);
        cursorposition(0xFF85);
        lcd_data(0xFF43);  /*CONTROLLER*/
        lcd_data(0xFF4F);
        lcd_data(0xFF4E);
        lcd_data(0xFF54);
        lcd_data(0xFF52);
        lcd_data(0xFF4F);
        lcd_data(0xFF4C);
        lcd_data(0xFF4C);
        lcd_data(0xFF45);
        lcd_data(0xFF52);
        cursorposition(0xFFC0);
        manual();
        cursorposition(0xFFCC);
        menu(); }

void display_main_menu(){
        /* 80h is the address for cursor position. 80h plus address LCD
        of 03h,where at start of 1st line. Thus=83h. */

        delay_mS(5);
        cursorposition(0xFF83);
        lcd_data(0xFF4D); /*Main*/
        lcd_data(0xFF61);
        lcd_data(0xFF69);
        lcd_data(0xFF6E);
        cursorposition(0xFF89);
        menu();
```

```
/* 80h is the address for cursor position. 80h plus address LCD
  of 40h,where at start of 2nd line. Thus=C0h. */
            cursorposition(0xFFC0);
            Run();
            cursorposition(0xFFC5);
            reset();
            cursorposition(0xFFCB);
            setup(); }

void display_manual_mode(){
            delay_mS(5);
            cursorposition(0xFF81);
            manual();
            cursorposition(0xFF88);
            control();
            cursorposition(0xFFC0);  /*Fwd*/
            lcd_data(0xFF46);
            lcd_data(0xFF77);
            lcd_data(0xFF64);
            cursorposition(0xFFC6);
            lcd_data(0xFF4D);  /*Main*/
            lcd_data(0xFF61);
            lcd_data(0xFF69);
            lcd_data(0xFF6E);
            cursorposition(0xFFCD);  /*Rvr*/
            lcd_data(0xFF52);
            lcd_data(0xFF76);
            lcd_data(0xFF72);}

void display_press_start(){
            /* 80h is the address for cursor position. 80h plus address LCD
            of 01h,where at start of 1st line. Thus=81h. */

            delay_mS(5);
            cursorposition(0xFF81);
            Press();                        /*Press*/
            cursorposition(0xFF87);
            lcd_data(0xFF74);               /*to*/
            lcd_data(0xFF6F);
            cursorposition(0xFF8A);
            lcd_data(0xFF73);               /*start*/
            lcd_data(0xFF74);
            lcd_data(0xFF61);
            lcd_data(0xFF72);
            lcd_data(0xFF74);
            cursorposition(0xFFC0);
            lcd_data(0xFF7F);
            cursorposition(0xFFC6);
```

```
        menu();                         /*Stop*/
        cursorposition(0xFFCF);
        lcd_data(0xFF7E); }

void system_running(){
                /* 80h is the address for cursor position. 80h plus address LCD
                of 01h,where at start of 1st line. Thus=81h. */
        delay_mS(1);
        cursorposition(0xFF81);
        system();
        cursorposition(0xFF88);
        lcd_data(0xFF52);               /*running*/
        lcd_data(0xFF75);
        lcd_data(0xFF6E);
        lcd_data(0xFF6E);
        lcd_data(0xFF69);
        lcd_data(0xFF6E);
        lcd_data(0xFF67);
        cursorposition(0xFFC6);
        Stop();                         /*Stop*/
                        }

void display_on_time(){
   cursorposition(0xFF80);
        on();                           /*on*/
        cursorposition(0xFF83);
        time();                         /*time*/
        cursorposition(0xFF8A);
        uS();                                   /*uS*/
        cursorposition(0xFFC0);
        pge();                                  /*pge*/
        cursorposition(0xFFC6);
        nxt();                                  /*nxt*/
        cursorposition(0xFFCC);
        inc();                                  /*inc*/
                }

void display_off_time(){
   cursorposition(0xFF80);
        off();                                  /*off*/
        cursorposition(0xFF84);
        time();                                 /*time*/
        cursorposition(0xFF8A);
        uS();                                   /*sec*/
        cursorposition(0xFFC0);
        pge();                                  /*pge*/
        cursorposition(0xFFC6);
        nxt();                                  /*nxt*/
```

```
        cursorposition(0xFFCC);
        inc();                          /*inc*/
                }

void current(){
        cursorposition(0xFF80);
        Current();                      /*Current*/
        cursorposition(0xFF8A);
        amps();                         /*amps*/
        cursorposition(0xFFC0);
        pge();                          /*pge*/
        cursorposition(0xFFC6);
        nxt();                          /*nxt*/
        cursorposition(0xFFCC);
        inc();                          /*inc*/
                }

void save_settings(){
        int no[]={0xFF4E,0xFF6F};
        delay_mS(2);
        cursorposition(0xFF81);
        lcd_data(0xFF53);               /*save*/
        lcd_data(0xFF61);
        lcd_data(0xFF76);
        lcd_data(0xFF65);
        cursorposition(0xFF86);
        lcd_data(0xFF53);               /*settings?*/
        lcd_data(0xFF65);
        lcd_data(0xFF74);
        lcd_data(0xFF74);
        lcd_data(0xFF69);
        lcd_data(0xFF6E);
        lcd_data(0xFF67);
        lcd_data(0xFF73);
        lcd_data(0xFF3F);
        cursorposition(0xFFC0);
        lcd_data(0xFF59);               /*yes*/
        lcd_data(0xFF65);
        lcd_data(0xFF73);
        cursorposition(0xFFCE);
        lcd_data(0xFF6E);               /*no*/
        lcd_data(0xFF6F);
           }

void system_stop(){
        delay_mS(2);
        cursorposition(0xFF82);
        system();
        cursorposition(0xFF8A);
        Stop();
        cursorposition(0xFFC0);
        manual();
        cursorposition(0xFFCC);
        menu(); }

void up_time(){
        delay_mS(2);
        cursorposition(0xFF80);
        Up();                               /*up*/
        cursorposition(0xFF83);
        time();                             /*time*/
        cursorposition(0xFF8A);
        sec();                              /*sc*/
        cursorposition(0xFFC0);
        pge();                              /*pge*/
        cursorposition(0xFFC6);
        nxt();                              /*nxt*/
        cursorposition(0xFFCC);
        inc();          }

void down_time(){
        delay_mS(2);
        cursorposition(0xFF80);
        Down();                             /*on*/
        cursorposition(0xFF84);
        time();                             /*time*/
        cursorposition(0xFF8A);
        sec();                              /*sc*/
        cursorposition(0xFFC0);
        pge();                              /*pge*/
        cursorposition(0xFFC6);
        nxt();                              /*nxt*/
        cursorposition(0xFFCC);
        inc();          }

/*push button function */
int next_cursor_on(void){     /* function for digital inc. */
                switch (count_on){
                case 0: if(count_on==0){
                                        delay_mS(600);
                                        lcd_cmd(0xFF8A);
                                        count_on=1;}
                                break;
                case 1: if(count_on==1){
                                        delay_mS(600);
                                        lcd_cmd(0xFF8B);
                                        count_on=2;}
                              break;
                case 2: if(count_on==2){
                                        delay_mS(600);
                                        lcd_cmd(0xFF8C);
                                        count_on=3;}
                              break;
                case 3: if(count_on==3){
                                        delay_mS(600);
                                        lcd_cmd(0xFF8D);
                                        count_on=4; }
                              break;
                case 4: if(count_on==4){
                                        delay_mS(600);
                                        lcd_cmd(0xFF8A);
                                        count_on=1;}
                              break;
                return count_on;  }
    }

int next_cursor_off(void){              /* function for digital inc. */

        switch (count_off){
                case 0:  if(count_off==0){
                                        delay_mS(600);
                                        lcd_cmd(0xFF8A);
                                        count_off=1; }
                              break;
                case 1:  if(count_off==1){
                                        delay_mS(600);
                                        lcd_cmd(0xFF8B);
                                        count_off=2;}
                              break;
                case 2: if(count_off==2){
                                        delay_mS(600);
                                        lcd_cmd(0xFF8C);
                                        count_off=3; }
                              break;
                case 3: if(count_off==3){
                                        delay_mS(600);
                                        lcd_cmd(0xFF8D);
                                        count_off=4;     }
                              break;
                case 4: if(count_off==4){
                                        delay_mS(600);
                                        lcd_cmd(0xFF8A);
                                        count_off=1; }
                              break;
                return count_off;  }

        }

int next_cursor_current(void){   /* function for digital inc. */

        switch (count_current){
                case 0:  if(count_current==0){
                                                delay_mS(600);
                                                lcd_cmd(0xFF8A);
                                                count_current=1;}
                                break;
                case 1:  if(count_current==1){
                                                delay_mS(600);
                                                lcd_cmd(0xFF8B);
                                                count_current=2; }
                                break;
                case 2: if(count_current==2){
                                                delay_mS(600);
                                                lcd_cmd(0xFF8A);
                                                count_current=1;  }
                                break;
                return count_current;  }

   }

int next_cursor_up(void){        /* function for digital inc. */

        switch (count_up){
                case 0:  if(count_up==0){
                                                delay_mS(600);
                                                lcd_cmd(0xFF8A);
                                                count_up=1; }
                                break;
                case 1:  if(count_up==1){
                                                delay_mS(600);
                                                lcd_cmd(0xFF8B);
                                                count_up=2;  }
                                break;
                case 2: if(count_up==2){
                                                delay_mS(600);
                                                lcd_cmd(0xFF8C);
                                                count_up=3; }
                                break;
                case 3: if(count_up==3){
                                                delay_mS(600);
                                                lcd_cmd(0xFF8D);
                                                count_up=4;}
                                break;
                case 4: if(count_up==4){
                                        delay_mS(600);
                                        lcd_cmd(0xFF8A);
                                        count_up=1;}
                                break;
                return count_up;   }

        }

int next_cursor_down(void){                 /* function for digital inc. */

        switch (count_down){
                case 0:  if(count_down==0){
                                        delay_mS(600);
                                        lcd_cmd(0xFF8A);
                                        count_down=1; }
                                break;
                case 1:  if(count_down==1){
                                        delay_mS(600);
                                        lcd_cmd(0xFF8B);
                                        count_down=2;}
                                break;
                case 2: if(count_down==2){
                                        delay_mS(600);
                                        lcd_cmd(0xFF8C);
                                        count_down=3;}
                                break;
                case 3: if(count_down==3){
                                        delay_mS(600);
                                        lcd_cmd(0xFF8D);
                                        count_down=4;}
                                break;
                case 4: if(count_down==4){
                                        delay_mS(600);
                                        lcd_cmd(0xFF8A);
                                        count_down=1;}
                                break;
                return count_down;   }
        }

int increament_on0(void){          /* function for digital inc. */
                if(sense_pushbutton(0x0004)){
                        delay_mS(500);
                        lcd_data(0xFF20);
                        lcd_cmd(0xFF0D);
                        lcd_cmd(0xFF10);}}
```

```
int increament_on1(void){
                if(sense_pushbutton(0x0004)){
                        delay_mS(500);
                        lcd_data(0xFF30+count_1a);
                        lcd_cmd(0xFF0D);
                        lcd_cmd(0xFF10);
                        value_1a=count_1a;
                        count_1a=count_1a+1;
                   if(count_1a==10)count_1a=0;
                return value_1a;} }

int increament_on2(void){
                if(sense_pushbutton(0x0004)){
                        delay_mS(500);
                        lcd_data(0xFF30+count_2a);
                        lcd_cmd(0xFF0D);
                        lcd_cmd(0xFF10);
                        value_2a=count_2a;
                        count_2a=count_2a+1;
                   if(count_2a==10)count_2a=0;
                return value_2a;}  }

int increament_on3(void){
        if(sense_pushbutton(0x0004)){
                        delay_mS(500);
                        lcd_data(0xFF30+count_3a);
                        lcd_cmd(0xFF0D);
                        lcd_cmd(0xFF10);
                        value_3a=count_3a;
                        count_3a=count_3a+1;
                   if(count_3a==10)count_3a=0;
        return value_3a;}   }

int increament_on4(void){
        if(sense_pushbutton(0x0004)){
                        delay_mS(500);
                        lcd_data(0xFF30+count_4a);
                        lcd_cmd(0xFF0D);
                        lcd_cmd(0xFF10);
                        value_4a=count_4a;
                        count_4a=count_4a+1;
                   if(count_4a==10)count_4a=0;
        return value_4a;}  }

int increament_off1(void){
                if(sense_pushbutton(0x0004)){
                        delay_mS(500);
                        lcd_data(0xFF30+count_1b);
```

```
                              lcd_cmd(0xFF0D);
                              lcd_cmd(0xFF10);
                              value_1b=count_1b;
                              count_1b=count_1b+1;
                              if(count_1b==10)count_1b=0;
                    return value_1b;}  }


int increament_off2(void){
   if(sense_pushbutton(0x0004)){
                              delay_mS(500);
                              lcd_data(0xFF30+count_2b);
                              lcd_cmd(0xFF0D);
                              lcd_cmd(0xFF10);
                              value_2b=count_2b;
                              count_2b=count_2b+1;
                              if(count_2b==10)count_2b=0;
          return value_2b;} }


int increament_off3(void){
   if(sense_pushbutton(0x0004)){
                              delay_mS(500);
                              lcd_data(0xFF30+count_3b);
                              lcd_cmd(0xFF0D);
                              lcd_cmd(0xFF10);
                              value_3b=count_3b;
                              count_3b=count_3b+1;
                              if(count_3b==10)count_3b=0;
          return value_3b;}  }


int increament_off4(void){
   if(sense_pushbutton(0x0004)){
                              delay_mS(500);
                              lcd_data(0xFF30+count_4b);
                              lcd_cmd(0xFF0D);
                              lcd_cmd(0xFF10);
                              value_4b=count_4b;
                              count_4b=count_4b+1;
                              if(count_4b==10)count_4b=0;
          return value_4b;}  }


int increament_current1(void){
   if(sense_pushbutton(0x0004)){
                              delay_mS(500);
                              lcd_data(0xFF30+count_1c);
```

```
                        lcd_cmd(0xFF0D);
                        lcd_cmd(0xFF10);
                        value_1c=count_1c;
                        count_1c=count_1c+1;
                        if(count_1c==10)count_1c=0;
        return value_1c;} }


int increament_current2(void){
  if(sense_pushbutton(0x0004)){
                        delay_mS(500);
                        lcd_data(0xFF30+count_2c);
                        lcd_cmd(0xFF0D);
                        lcd_cmd(0xFF10);
                        value_2c=count_2c;
                        count_2c=count_2c+1;
                        if(count_2c==10)count_2c=0;
        return value_2c;} }

int increament_up1(void){
                if(sense_pushbutton(0x0004)){
                        delay_mS(500);
                        lcd_data(0xFF30+count_1d);
                        lcd_cmd(0xFF0D);
                        lcd_cmd(0xFF10);
                        value_1d=count_1d;
                        count_1d=count_1d+1;
                   if(count_1d==10)count_1d=0;
                return value_1d;} }

int increament_up2(void){
                if(sense_pushbutton(0x0004)){
                        delay_mS(500);
                        lcd_data(0xFF30+count_2d);
                        lcd_cmd(0xFF0D);
                        lcd_cmd(0xFF10);
                        value_2d=count_2d;
                        count_2d=count_2d+1;
                   if(count_2d==10)count_2d=0;
                return value_2d;}  }

int increament_up4(void){
        if(sense_pushbutton(0x0004)){
                        delay_mS(500);
                        lcd_data(0xFF30+count_4d);
                        lcd_cmd(0xFF0D);
                        lcd_cmd(0xFF10);
                        value_4d=count_4d;
```

```
            count_4d=count_4d+1;
          if(count_4d==10)count_4d=0;
    return value_4d;}  }

int increament_down1(void){
        if(sense_pushbutton(0x0004)){
            delay_mS(500);
            lcd_data(0xFF30+count_1e);
            lcd_cmd(0xFF0D);
            lcd_cmd(0xFF10);
            value_1e=count_1e;
            count_1e=count_1e+1;
          if(count_1e==10)count_1e=0;
        return value_1e;} }

int increament_down2(void){
        if(sense_pushbutton(0x0004)){
            delay_mS(500);
            lcd_data(0xFF30+count_2e);
            lcd_cmd(0xFF0D);
            lcd_cmd(0xFF10);
            value_2e=count_2e;
            count_2e=count_2e+1;
          if(count_2e==10)count_2e=0;
        return value_2e;}  }

int increament_down4(void){
    if(sense_pushbutton(0x0004)){
            delay_mS(500);
            lcd_data(0xFF30+count_4e);
            lcd_cmd(0xFF0D);
            lcd_cmd(0xFF10);
            value_4e=count_4e;
            count_4e=count_4e+1;
          if(count_4e==10)count_4e=0;
    return value_4e;}  }

int sense_pushbutton(int pb) {    /*pushbutton shares with limit switch*/
    unsigned char value;              /*electro contact & emergency */
    value = PCDATDIR & pb;
    if (value == 0x0000) return 1;
    else return 0;
    }

int sense_FF(int sf) {    /*Fan Fault, FF switch */
    unsigned char value;
    value = PDDATDIR & sf;
    if (value == 0x0000) return 1;
```

```
        else return 0;
        }

void delay_mS(k){
        for (j=0;j<k;j++){
                unsigned int i;         /* 1 mS delay with  SCSR1 = 0x00FD;*/
                for (i=0;i<4000;i++); } } /* this is generated by software*/

void delay_s(f){
        int x;
        for(x=0;x<f;x++){
                delay_mS(1);
                if(sense_pushbutton(0x0008)){
                        delay_mS(50);
                lcd_cmd(0xFF01);
                state=16;
                system_stop();
                        PFDATDIR = 0xFF00;
                        break;  }
                                        }       }

unsigned int ReadADC(int adc_channel) {  /* Read ADC */
        unsigned int adc_value;
        ADCTRL2 = 0x2000;
        if (adc_channel==0) {
                adc_value=RESULT0>>6 & 0x03FF; }
        else if (adc_channel==1) {
                adc_value=RESULT1>>6 & 0x03FF; }
        else if (adc_channel==2) {
                adc_value=RESULT2>>6 & 0x03FF; }
        return adc_value;       }

void auto_up_down_stop(){
        /*stop associate timer*/
        CMPR1=500; /*set pwm to 50% DC so motor stop*/}

void dsp_setup(){
        /*** Configure the System Control and Status registers ***/
   SCSR1 = 0x00FD; /*40MHz clock freq */
   SCSR2 = (SCSR2 | 0x000B) & 0x000F;

/*** Disable the watchdog timer ***/
   WDCR  = 0x00E8;

/*** Setup external memory interface for LF2407 EVM ***/
   WSGR = 0x0040;

/*** Setup shared I/O pins ***/
```

```
   MCRA = 0x00C0; /* activate pwm1 & pwm2 */
   MCRB = 0x0000;              /* group B pins */
   MCRC = 0x000A; /* pwm7=Q6, pwm9=Q7 */

/*** Configure IO pin as an input ***
  PCDATDIR = 0x0000;
  PCDATDIR = PADATDIR | 0x4000;  /*set pwm1 o/p  *
  PDDATDIR = 0x0000;

/*** Configure IO pin as an output **/
   PFDATDIR = 0xFF00;
   PBDATDIR = 0xFF00;
   PEDATDIR = 0xFF00;
   PADATDIR = 0xFF00;

/*** Setup the core interrupts ***/
   IMR = 0x0000;              /* clear the IMR register */
   IFR = 0x003F;              /* clear any pending core interrupts */
   IMR = 0x0004;              /* enable desired core interrupts */

/*** Setup the event manager interrupts ***/
   EVAIFRA = 0xFFFF;          /* clear all EVA group A interrupts */
   EVAIFRB = 0xFFFF;          /* clear all EVA group B interrupts */
   EVAIFRC = 0xFFFF;          /* clear all EVA group C interrupts */
   EVAIMRA = 0x0000;          /* enable desired EVA group A interrupts */
   EVAIMRB = 0x0001;          /* enable desired EVA group B interrupts */
   EVAIMRC = 0x0000;          /* enable desired EVA group C interrupts */

   EVBIFRA = 0xFFFF;          /* clear all EVB group A interrupts */
   EVBIFRB = 0xFFFF;          /* clear all EVB group B interrupts */
   EVBIFRC = 0xFFFF;          /* clear all EVB group C interrupts */
   EVBIMRA = 0x0000;          /* enable desired EVB group A interrupts */
   EVBIMRB = 0x0000;          /* enable desired EVB group B interrupts */
   EVBIMRC = 0x0000;         /* enable desired EVB group C interrupts */

/*** Enable global interrupts ***/
   asm(" CLRC INTM");         /* enable global interrupts */

/**** ADC Setup ****/
   ADCTRL1 = 0x4000;
   ADCTRL1 = 0x2F82;
   MAX_CONV = 0x0002;
   CHSELSEQ1 = 0x0210;
   ADCTRL2 = 0x0000;

/*** Setup timers 1 and 2, and the PWM configuration ***/
   T1CON = 0x0000;            /* disable timer 1 */
   T2CON = 0x0000;            /* disable timer 2 */
   GPTCONA = 0x0000;              /* configure GPTCONA */

/* Timer 1: configure to clock 20kHz PWM1 & PWM2 pin */
/* Asymmetric PWM, user set carrier frequency & duty cycle */
   T1CNT = 0x0000;              /* clear timer counter */
   T1PR = 0;    /* initialized timer period to 0 */
   DBTCONA = 0x0000;            /* deadband units off */
   CMPR1 = 0;/*500;   /* initialized PWM1 & PWM2 duty cycle */
   ACTRA = 0x0009;   /*pwm1 active low & pwm2 active high  */
   COMCONA = 0x8200;            /* configure COMCON register */
   T1CON = 0x0940;              /* Assymetric & prescaler 2 */

/* Timer 3, generate Q6 & Q7 */
/* Timer 3: configure to clock PWM7 & PWM9 pin */
/* Asymmetric PWM, user set carrier frequency & duty cycle */
/* Q7 always 2.2uS delay start from Q6 */
   T3CNT = 0x0000;              /* clear timer counter */
   T3PR = period;               /* initialized timer period to 0 */
   DBTCONA = 0x0000;            /* deadband units off */
   CMPR4 = pwm_duty_0;          /* initialized PWM7 duty cycle */
   CMPR5 = pwm_duty_0;          /* initialized PWM9 duty cycle */
   ACTRB = 0x0021;    /* use PWM7(Q6) & PWM9(Q7) invert ,PWM2 pin set
                                active high */
   COMCONB = 0x8200;
   T3CON = 0x1240;    /* Assymetric & prescaler 2 */

/* Timer 2: configure to generate a 0.01s periodic interrupt */
   T2CNT = 0x0000;              /* clear timer counter */
   T2PR = 0;                    /* initialized timer period */
   T2CON = 0x0840;              /* configure T2CON register */

 /*** Enable global interrupts ***/
   asm(" CLRC INTM");             /* enable global interrupts */
}

void lcd_display_function(){
         switch(state){
                  case 0: if(state==0&&sense_pushbutton(0x0004)){
                                    delay_mS(50);
                                    lcd_cmd(0xFF01); /*clear display command*/
                                    state=1;
                                    display_main_menu();}
                           else if(state==0&&sense_pushbutton(0x0010)){
                                    delay_mS(50);
                                    lcd_cmd(0xFF01); /*clear display command*/
                                    state=11;
                                    display_manual_mode();
```

```
                              w=2; /*start manual mode program in loop */ }
                    break;

          case 1: if(state==1&&sense_pushbutton(0x0010)){
                              delay_mS(50);
                              lcd_cmd(0xFF01);
                              state=8;
                              display_press_start(); }
                    else if(state==1&&sense_pushbutton(0x0008)){
                              delay_mS(50);
                              lcd_cmd(0xFF01); /*clear display command*/
                              state=11;
                              display_manual_mode();
                              w=2; /*start manual mode program in loop */ }
                    else if(state==1&&sense_pushbutton(0x0004)){
                              delay_mS(50);
                              lcd_cmd(0xFF01);
                              state=2;
                              count_1(); /*reset */
                              display_on_time();
                              cursorposition(0xFF8A);
                              lcd_data(0xFF30+value_1a);
                              cursorposition(0xFF8B);
                              lcd_data(0xFF30+value_2a);
                              cursorposition(0xFF8C);
                              lcd_data(0xFF30+value_3a);
                              cursorposition(0xFF8D);
                              lcd_data(0xFF30+value_4a);
                              cursorposition(0xFF89);
                              lcd_cmd(0xFF0D);   }
                    break;

case 2: if(state==2&&sense_pushbutton(0x0010)){
                              delay_mS(50);
                              lcd_cmd(0xFF01);
                              state=3;
                              display_off_time();
                              cursorposition(0xFF8A);
                              lcd_data(0xFF30+value_1b);
                              cursorposition(0xFF8B);
                              lcd_data(0xFF30+value_2b);
                              cursorposition(0xFF8C);
                              lcd_data(0xFF30+value_3b);
                              cursorposition(0xFF8D);
                              lcd_data(0xFF30+value_4b);
                              cursorposition(0xFF89);
                              lcd_cmd(0xFF0D);}
                    if(state==2&&sense_pushbutton(0x0008))next_cursor_on();
```

```
        if(state==2&&sense_pushbutton(0x0004)){
            if (count_on==0) increament_on0();
            else if (count_on==1) increament_on1();
            else if (count_on==2) increament_on2();
            else if (count_on==3) increament_on3();
            else if (count_on==4) increament_on4();}
            break;

case 3: if(state==3&&sense_pushbutton(0x0010)){
            delay_mS(50);
            lcd_cmd(0xFF01);
            state=4;
            current();
            cursorposition(0xFF8A);
            lcd_data(0xFF30+value_1c);
            cursorposition(0xFF8B);
            lcd_data(0xFF30+value_2c);
            cursorposition(0xFF89);
            lcd_cmd(0xFF0D);}
            if(state==3&&sense_pushbutton(0x0008))
                next_cursor_off();
            if(state==3&&sense_pushbutton(0x0004)){
            if (count_off==1) increament_off1();
            else if (count_off==2) increament_off2();
            else if (count_off==3) increament_off3();
            else if (count_off==4) increament_off4();}
            break;

case 4: if(state==4&&sense_pushbutton(0x0010)){
            delay_mS(50);
            lcd_cmd(0xFF01);
            state=5;
            up_time();
            cursorposition(0xFF8A);
            lcd_data(0xFF30+value_1d);
            cursorposition(0xFF8B);
            lcd_data(0xFF30+value_2d);
            cursorposition(0xFF8D);
            lcd_data(0xFF30+value_4d);
            cursorposition(0xFF89);
            lcd_cmd(0xFF0D);}
            if(state==4&&sense_pushbutton(0x0008))
                next_cursor_current();
            if(state==4&&sense_pushbutton(0x0004)){
            if (count_current==1) increament_current1();
            else if (count_current==2) increament_current2();}
            break;
```

```
case 5: if(state==5&&sense_pushbutton(0x0010)){
                        delay_mS(50);
                        lcd_cmd(0xFF01);
                        state=6;
                        down_time();
                        cursorposition(0xFF8A);
                        lcd_data(0xFF30+value_1e);
                        cursorposition(0xFF8B);
                        lcd_data(0xFF30+value_2e);
                        cursorposition(0xFF8D);
                        lcd_data(0xFF30+value_4e);
                        cursorposition(0xFF89);
                        lcd_cmd(0xFF0D);}
                        if(state==5&&sense_pushbutton(0x0008))
                                next_cursor_up();
                        if(state==5&&sense_pushbutton(0x0004)){
                        if (count_up==1) increament_up1();
                        else if (count_up==2) increament_up2();
                        else if (count_up==4) increament_up4();}
                        break;

case 6:if(state==6&&sense_pushbutton(0x0010)){
                        delay_mS(50);
                        lcd_cmd(0xFF01);
                        state=7;
                        save_settings();   }
                        if(state==6&&sense_pushbutton(0x0008))
                                next_cursor_down();
                        if(state==6&&sense_pushbutton(0x0004)){
                        if (count_down==1) increament_down1();
                        else if (count_down==2) increament_down2();
                        else if (count_down==4) increament_down4();}
                        break;

case 7: if(state==7&&sense_pushbutton(0x0004)){
                        delay_mS(50);
                        lcd_cmd(0xFF01);
                        state=2;
                        count_1();
                        display_on_time();
                        cursorposition(0xFF8A);
                        lcd_data(0xFF30+value_1a);
                        cursorposition(0xFF8B);
                        lcd_data(0xFF30+value_2a);
                        cursorposition(0xFF8C);
                        lcd_data(0xFF30+value_3a);
                        cursorposition(0xFF8D);
                        lcd_data(0xFF30+value_4a);
                        cursorposition(0xFF89);
                        lcd_cmd(0xFF0D);   }
                        else if(state==7&&sense_pushbutton(0x0010)){
                        delay_mS(50);
                        lcd_cmd(0xFF01);
                        state=1;
                        display_main_menu();
                        count_0();}
                        break;

case 8: if(state==8&&sense_pushbutton(0x0002)&&sense_pushbutton(0x0020)){
                        delay_mS(50);
                        lcd_cmd(0xFF01);
                        state=9;
                        system_running();
                        lcd_cmd(0xFF0D);
                        run_system();
                        setting_on();
                        w=1;}            /*w=1, to main loop, activate detect faulty*/
                        else if(state==8&&sense_pushbutton(0x0008)){
                        delay_mS(50);
                        lcd_cmd(0xFF01);
                        state=1;
                        display_main_menu();
                        lcd_cmd(0xFF0D); }
                        break;

case 9: if(state==9&&sense_pushbutton(0x0008)){
                        delay_mS(50);
                        lcd_cmd(0xFF01);
                        state=10;
                        system_stop();
                        setting_off();}
                        break;

case 10:if(state==10&&sense_pushbutton(0x0004)){
                        delay_mS(50);
                        lcd_cmd(0xFF01);
                        state=1;
                        display_main_menu();}
                        else if(state==10&&sense_pushbutton(0x0010)){
                        delay_mS(50);
                        lcd_cmd(0xFF01);
                        state=11;
                        display_manual_mode();
                        w=2; /*start manual mode program in loop */ }
                        break;
```

```
        case 11: if(state==11&&sense_pushbutton(0x0008)){
                                delay_mS(50);
                                lcd_cmd(0xFF01);
                                display_main_menu();
                                state=1;
                                w=0; /*stop manual mode program in loop */
                                lcd_cmd(0xFF0D);}
                                break;

        case 12: if(state==12&&sense_pushbutton(0x0010)){
                                delay_mS(50);
                                lcd_cmd(0xFF01);
                                state=0;
                                display_edm_controller();
                                lcd_cmd(0xFF0D);}
                                break;
                }
}
```

```
/*****************************************************************
* Filename: f2407_c.h                                            *
*                                                                *
* Author: Azli Yahya, Loughborough University.                   *
*                                                                *
* Description: LF2407A DSP register definitions for EDM C-code.  *
*****************************************************************/

/* Core registers */
#define IMR           *(volatile unsigned int *)0x0004  /* Interrupt mask reg */
#define GREG          *(volatile unsigned int *)0x0005  /* Global memory allocation reg */
#define IFR           *(volatile unsigned int *)0x0006  /* Interrupt flag reg */

/* System configuration and interrupt registers */
#define PIRQR0        *(volatile unsigned int *)0x7010  /* Peripheral interrupt request reg 0 */
#define PIRQR1        *(volatile unsigned int *)0x7011  /* Peripheral interrupt request reg 1 */
#define PIRQR2        *(volatile unsigned int *)0x7012 /* Peripheral interrupt request reg 2 */
#define PIACKR0       *(volatile unsigned int *)0x7014/* Peripheral interrupt
              acknowledge reg 0 */
#define PIACKR1       *(volatile unsigned int *)0x7015  /* Peripheral interrupt
              acknowledge reg 1 */
#define PIACKR2       *(volatile unsigned int *)0x7016  /* Peripheral interrupt
              acknowledge reg 2 */
#define SCSR1    *(volatile unsigned int *)0x7018  /* System control & status reg 1 */
#define SCSR2    *(volatile unsigned int *)0x7019  /* System control & status reg 2 */
#define DINR     *(volatile unsigned int *)0x701C  /* Device identification reg */
#define PIVR     *(volatile unsigned int *)0x701E  /* Peripheral interrupt vector reg */

/* Watchdog timer (WD) registers */
#define WDCNTR        *(volatile unsigned int *)0x7023  /* WD counter reg */
#define WDKEY         *(volatile unsigned int *)0x7025  /* WD reset key reg */
#define WDCR          *(volatile unsigned int *)0x7029  /* WD timer control reg */

/* Serial Peripheral Interface (SPI) registers */
#define SPICCR        *(volatile unsigned int *)0x7040 /* SPI configuration control reg */
#define SPICTL        *(volatile unsigned int *)0x7041  /* SPI operation control reg */
#define SPISTS        *(volatile unsigned int *)0x7042  /* SPI status reg */
#define SPIBRR        *(volatile unsigned int *)0x7044  /* SPI baud rate reg */
#define SPIRXEMU      *(volatile unsigned int *)0x7046  /* SPI emulation buffer reg */
#define SPIRXBUF      *(volatile unsigned int *)0x7047  /* SPI serial receive buffer reg */
#define SPITXBUF      *(volatile unsigned int *)0x7048 /* SPI serial transmit buffer reg */
#define SPIDAT        *(volatile unsigned int *)0x7049  /* SPI serial data reg */
#define SPIPRI        *(volatile unsigned int *)0x704F  /* SPI priority control reg */

/* SCI registers */
#define SCICCR        *(volatile unsigned int *)0x7050 /* SCI communication control reg */
#define SCICTL1       *(volatile unsigned int *)0x7051  /* SCI control reg 1 */
```

```
#define SCIHBAUD     *(volatile unsigned int *)0x7052 /* SCI baud-select reg, high bits */
#define SCILBAUD     *(volatile unsigned int *)0x7053  /* SCI baud-select reg, low bits */
#define SCICTL2      *(volatile unsigned int *)0x7054  /* SCI control reg 2 */
#define SCIRXST      *(volatile unsigned int *)0x7055  /* SCI receiver status reg */
#define SCIRXEMU     *(volatile unsigned int *)0x7056 /* SCI emulation data buffer reg */
#define SCIRXBUF     *(volatile unsigned int *)0x7057  /* SCI receiver data buffer reg */
#define SCITXBUF     *(volatile unsigned int *)0x7059  /* SCI transmit data buffer reg */
#define SCIPRI       *(volatile unsigned int *)0x705F  /* SCI priority control reg */

/* External interrupt configuration registers */
#define XINT1CR    *(volatile unsigned int *)0x7070  /* Ext interrupt 1 config reg */
#define XINT2CR    *(volatile unsigned int *)0x7071  /* Ext interrupt 2 config reg */

/* Digital I/O registers */
#define MCRA         *(volatile unsigned int *)0x7090  /* I/O mux control reg A */
#define MCRB         *(volatile unsigned int *)0x7092  /* I/O mux control reg B */
#define MCRC         *(volatile unsigned int *)0x7094  /* I/O mux control reg C */
#define PADATDIR     *(volatile unsigned int *)0x7098  /* I/O port A data & dir reg */
#define PBDATDIR     *(volatile unsigned int *)0x709A  /* I/O port B data & dir reg */
#define PCDATDIR     *(volatile unsigned int *)0x709C  /* I/O port C data & dir reg */
#define PDDATDIR     *(volatile unsigned int *)0x709E  /* I/O port D data & dir reg */
#define PEDATDIR     *(volatile unsigned int *)0x7095  /* I/O port E data & dir reg */
#define PFDATDIR     *(volatile unsigned int *)0x7096  /* I/O port F data & dir reg */

/* Analog-to-Digital Converter (ADC) registers */
#define ADCTRL1    *(volatile unsigned int *)0x70A0  /* ADC control reg 1 */
#define ADCTRL2    *(volatile unsigned int *)0x70A1  /* ADC control reg 2 */
#define MAX_CONV             *(volatile unsigned int *)0x70A2  /* Maximum conversion
                             channels reg */
#define CHSELSEQ1            *(volatile unsigned int *)0x70A3  /* Channel select
                             sequencing control reg 1 */
#define CHSELSEQ2            *(volatile unsigned int *)0x70A4  /* Channel select
                             sequencing control reg 2 */
#define CHSELSEQ3            *(volatile unsigned int *)0x70A5  /* Channel select
                             sequencing control reg 3 */
#define CHSELSEQ4            *(volatile unsigned int *)0x70A6  /* Channel select
                             sequencing control reg 4 */
#define AUTO_SEQ_SR          *(volatile unsigned int *)0x70A7  /* Autosequence
                             status reg */
#define RESULT0  *(volatile unsigned int *)0x70A8  /* Conversion result buffer reg 0 */
#define RESULT1  *(volatile unsigned int *)0x70A9  /* Conversion result buffer reg 1 */
#define RESULT2  *(volatile unsigned int *)0x70AA /* Conversion result buffer reg 2 */
#define RESULT3  *(volatile unsigned int *)0x70AB /* Conversion result buffer reg 3 */
#define RESULT4  *(volatile unsigned int *)0x70AC /* Conversion result buffer reg 4 */
#define RESULT5  *(volatile unsigned int *)0x70AD /* Conversion result buffer reg 5 */
#define RESULT6  *(volatile unsigned int *)0x70AE /* Conversion result buffer reg 6 */
#define RESULT7  *(volatile unsigned int *)0x70AF /* Conversion result buffer reg 7 */
#define RESULT8  *(volatile unsigned int *)0x70B0  /* Conversion result buffer reg 8 */
```

```
#define RESULT9  *(volatile unsigned int *)0x70B1  /* Conversion result buffer reg 9 */
#define RESULT10*(volatile unsigned int *)0x70B2 /* Conversion resultbuffer reg 10 */
#define RESULT11*(volatile unsigned int *)0x70B3 /* Conversion result buffer reg11 */
#define RESULT12 *(volatile unsigned int *)0x70B4 /*Conversion result buffer reg12 */
#define RESULT13 *(volatile unsigned int *)0x70B5 /*Conversion result buffer reg13 */
#define RESULT14 *(volatile unsigned int *)0x70B6/*Conversion result buffer reg 14 */
#define RESULT15 *(volatile unsigned int *)0x70B7/*Conversion result buffer reg 15 */
#define CALIBRATION  *(volatile unsigned int *)0x70B8  /* Calibration result reg */

/* Controller Area Network (CAN) registers */
#define MDER          *(volatile unsigned int *)0x7100 /* CAN mailbox direction/enable reg */
#define TCR           *(volatile unsigned int *)0x7101  /* CAN transmission control reg */
#define RCR           *(volatile unsigned int *)0x7102  /* CAN receive control reg */
#define MCR           *(volatile unsigned int *)0x7103  /* CAN master control reg */
#define BCR2          *(volatile unsigned int *)0x7104  /* CAN bit config reg 2 */
#define BCR1          *(volatile unsigned int *)0x7105  /* CAN bit config reg 1 */
#define ESR           *(volatile unsigned int *)0x7106  /* CAN error status reg */
#define GSR           *(volatile unsigned int *)0x7107  /* CAN global status reg */
#define CEC           *(volatile unsigned int *)0x7108  /* CAN trans and rcv err counters */
#define CAN_IFR       *(volatile unsigned int *)0x7109  /* CAN interrupt flag reg */
#define CAN_IMR       *(volatile unsigned int *)0x710a  /* CAN interrupt mask reg */
#define LAM0_H        *(volatile unsigned int *)0x710b  /* CAN local acceptance mask
              MBX0/1 */
#define LAM0_L        *(volatile unsigned int *)0x710c  /* CAN local acceptance mask
              MBX0/1 */
#define LAM1_H        *(volatile unsigned int *)0x710d  /* CAN local acceptance mask
              MBX2/3 */
#define LAM1_L        *(volatile unsigned int *)0x710e  /* CAN local acceptance mask
              MBX2/3 */
#define MSGID0L       *(volatile unsigned int *)0x7200  /* CAN message ID for mailbox
              0 (lower 16 bits) */
#define MSGID0H       *(volatile unsigned int *)0x7201  /* CAN message ID for
              mailbox 0 (upper 16 bits) */
#define MSGCTRL0      *(volatile unsigned int *)0x7202  /* CAN RTR and DLC for
              mailbox 0 */
#define MBX0A         *(volatile unsigned int *)0x7204 * CAN 2 of 8 bytes of mailbox 0*/
#define MBX0B         *(volatile unsigned int *)0x7205/* CAN 2 of 8 bytes of mailbox 0 */
#define MBX0C         *(volatile unsigned int *)0x7206/* CAN 2 of 8 bytes of mailbox 0 */
#define MBX0D         *(volatile unsigned int *)0x7207/* CAN 2 of 8 bytes of mailbox 0*/
#define MSGID1L       *(volatile unsigned int *)0x7208  /* CAN message ID for mailbox
              1 (lower 16 bits) */
#define MSGID1H       *(volatile unsigned int *)0x7209  /* CAN message ID for
              mailbox 1 (upper 16 bits) */
#define MSGCTRL1      *(volatile unsigned int *)0x720A  /* CAN RTR and DLC for
              mailbox 1 */
#define MBX1A         *(volatile unsigned int *)0x720C/* CAN 2 of 8 bytes of mailbox 1 */
#define MBX1B         *(volatile unsigned int *)0x720D/* CAN 2 of 8 bytes of mailbox 1 */
#define MBX1C         *(volatile unsigned int *)0x720E/* CAN 2 of 8 bytes of mailbox 1 */
```

```
#define MBX1D        *(volatile unsigned int *)0x720F/* CAN 2 of 8 bytes of mailbox 1 */
#define MSGID2L      *(volatile unsigned int *)0x7210  /* CAN message ID for mailbox 2
                     (lower 16 bits) */
#define MSGID2H      *(volatile unsigned int *)0x7211  /* CAN message ID for
                     mailbox 2 (upper 16 bits) */
#define MSGCTRL2     *(volatile unsigned int *)0x7212  /* CAN RTR and DLC for
                     mailbox 2 */
#define MBX2A        *(volatile unsigned int *)0x7214  /* CAN 2 of 8 bytes of mailbox 2 */
#define MBX2B        *(volatile unsigned int *)0x7215  /* CAN 2 of 8 bytes of mailbox 2 */
#define MBX2C        *(volatile unsigned int *)0x7216  /* CAN 2 of 8 bytes of mailbox 2 */
#define MBX2D        *(volatile unsigned int *)0x7217  /* CAN 2 of 8 bytes of mailbox 2 */

#define MSGID3L      *(volatile unsigned int *)0x7218  /* CAN message ID for mailbox
                     3 (lower 16 bits) */
#define MSGID3H      *(volatile unsigned int *)0x7219  /* CAN message ID for
                     mailbox 3 (upper 16 bits) */
#define MSGCTRL3     *(volatile unsigned int *)0x721A  /* CAN RTR and DLC for
                     mailbox 3 */
#define MBX3A        *(volatile unsigned int *)0x721C  /* CAN 2 of 8 bytes of mailbox 3 */
#define MBX3B        *(volatile unsigned int *)0x721D /* CAN 2 of 8 bytes of mailbox 3 */
#define MBX3C        *(volatile unsigned int *)0x721E /* CAN 2 of 8 bytes of mailbox 3 */
#define MBX3D        *(volatile unsigned int *)0x721F /* CAN 2 of 8 bytes of mailbox 3 */
#define MSGID4L      *(volatile unsigned int *)0x7220  /* CAN message ID for mailbox 4
                     (lower 16 bits) */
#define MSGID4H      *(volatile unsigned int *)0x7221  /* CAN message ID for
                     mailbox 4 (upper 16 bits) */
#define MSGCTRL4     *(volatile unsigned int *)0x7222  /* CAN RTR and DLC for
                     mailbox 4 */
#define MBX4A        *(volatile unsigned int *)0x7224  /* CAN 2 of 8 bytes of mailbox 4 */
#define MBX4B        *(volatile unsigned int *)0x7225  /* CAN 2 of 8 bytes of mailbox 4 */
#define MBX4C        *(volatile unsigned int *)0x7226  /* CAN 2 of 8 bytes of mailbox 4 */
#define MBX4D        *(volatile unsigned int *)0x7227 /* CAN 2 of 8 bytes of mailbox 4 */
#define MSGID5L      *(volatile unsigned int *)0x7228  /* CAN message ID for mailbox
                     5 (lower 16 bits) */
#define MSGID5H      *(volatile unsigned int *)0x7229  /* CAN message ID for
                     mailbox 5 (upper 16 bits) */
#define MSGCTRL5     *(volatile unsigned int *)0x722A  /* CAN RTR and DLC for
                     mailbox 5 */
#define MBX5A        *(volatile unsigned int *)0x722C  /* CAN 2 of 8 bytes of mailbox 5 */
#define MBX5B        *(volatile unsigned int *)0x722D /* CAN 2 of 8 bytes of mailbox 5 */
#define MBX5C        *(volatile unsigned int *)0x722E /* CAN 2 of 8 bytes of mailbox 5 */
#define MBX5D        *(volatile unsigned int *)0x722F /* CAN 2 of 8 bytes of mailbox 5 */

/* Event Manager A (EVA) registers */
#define GPTCONA      *(volatile unsigned int *)0x7400  /* GP timer control reg A */
#define T1CNT        *(volatile unsigned int *)0x7401  /* GP timer 1 counter reg */
#define T1CMPR       *(volatile unsigned int *)0x7402  /* GP timer 1 compare reg */
#define T1PR         *(volatile unsigned int *)0x7403  /* GP timer 1 period reg  */
```

```
#define T1CON        *(volatile unsigned int *)0x7404  /* GP timer 1 control reg */
#define T2CNT        *(volatile unsigned int *)0x7405  /* GP timer 2 counter reg */
#define T2CMPR       *(volatile unsigned int *)0x7406  /* GP timer 2 compare reg */
#define T2PR         *(volatile unsigned int *)0x7407  /* GP timer 2 period reg */
#define T2CON        *(volatile unsigned int *)0x7408  /* GP timer 2 control reg */
#define COMCONA      *(volatile unsigned int *)0x7411  /* Compare control reg A */
#define ACTRA        *(volatile unsigned int *)0x7413/* Compare action control reg A */
#define DBTCONA      *(volatile unsigned int *)0x7415/* Dead-band timer control reg A */
#define CMPR1        *(volatile unsigned int *)0x7417  /* compare reg 1 */
#define CMPR2        *(volatile unsigned int *)0x7418  /* compare reg 2 */
#define CMPR3        *(volatile unsigned int *)0x7419  /* compare reg 3 */
#define CAPCONA      *(volatile unsigned int *)0x7420  /* Capture control reg A */
#define CAPFIFOA     *(volatile unsigned int *)0x7422  /* Capture FIFO status reg A */
#define CAP1FIFO     *(volatile unsigned int *)0x7423  /* Capture Channel 1 FIFO top */
#define CAP2FIFO     *(volatile unsigned int *)0x7424  /* Capture Channel 2 FIFO top */
#define CAP3FIFO     *(volatile unsigned int *)0x7425  /* Capture Channel 3 FIFO top */
#define CAP1FBOT     *(volatile unsigned int *)0x7427  /* Bottom reg of capture FIFO
                     stack 1 */
#define CAP2FBOT     *(volatile unsigned int *)0x7427  /* Bottom reg of capture FIFO
                     stack 2 */
#define CAP3FBOT     *(volatile unsigned int *)0x7427  /* Bottom reg of capture FIFO
                     stack 3 */
#define EVAIMRA      *(volatile unsigned int *)0x742C  /* EVA interrupt mask reg A */
#define EVAIMRB      *(volatile unsigned int *)0x742D  /* EVA interrupt mask reg B */
#define EVAIMRC      *(volatile unsigned int *)0x742E  /* EVA interrupt mask reg C */
#define EVAIFRA      *(volatile unsigned int *)0x742F  /* EVA interrupt flag reg A */
#define EVAIFRB      *(volatile unsigned int *)0x7430  /* EVA interrupt flag reg B */
#define EVAIFRC      *(volatile unsigned int *)0x7431  /* EVA interrupt flag reg C */

/* Event Manager B (EVB) registers */
#define GPTCONB      *(volatile unsigned int *)0x7500  /* GP timer control reg B */
#define T3CNT        *(volatile unsigned int *)0x7501  /* GP timer 3 counter reg */
#define T3CMPR       *(volatile unsigned int *)0x7502  /* GP timer 3 compare reg */
#define T3PR         *(volatile unsigned int *)0x7503  /* GP timer 3 period reg */
#define T3CON        *(volatile unsigned int *)0x7504  /* GP timer 3 control reg */
#define T4CNT        *(volatile unsigned int *)0x7505  /* GP timer 4 counter reg */
#define T4CMPR       *(volatile unsigned int *)0x7506  /* GP timer 4 compare reg */
#define T4PR         *(volatile unsigned int *)0x7507  /* GP timer 4 period reg */
#define T4CON        *(volatile unsigned int *)0x7508  /* GP timer 4 control reg */
#define COMCONB      *(volatile unsigned int *)0x7511  /* Compare control register B */
#define ACTRB        *(volatile unsigned int *)0x7513  /* Compare action control
                     register B */
#define DBTCONB      *(volatile unsigned int *)0x7515 /* Dead-band timer control reg B */
#define CMPR4        *(volatile unsigned int *)0x7517  /* Compare reg 4 */
#define CMPR5        *(volatile unsigned int *)0x7518  /* Compare reg 5 */
#define CMPR6        *(volatile unsigned int *)0x7519  /* Compare reg 6 */
#define CAPCONB      *(volatile unsigned int *)0x7520  /* Capture control reg B */
#define CAPFIFOB     *(volatile unsigned int *)0x7522  /* Capture FIFO status reg B */
```

```
#define CAP4FIFO     *(volatile unsigned int *)0x7523  /* Capture channel 4 FIFO top */
#define CAP5FIFO     *(volatile unsigned int *)0x7524  /* Capture channel 5 FIFO top */
#define CAP6FIFO     *(volatile unsigned int *)0x7525  /* Capture channel 6 FIFO top */
#define CAP4FBOT     *(volatile unsigned int *)0x7527  /* Bottom reg of capture FIFO
             stack 4 */
#define CAP5FBOT     *(volatile unsigned int *)0x7527  /* Bottom reg of capture FIFO
             stack 5 */
#define CAP6FBOT     *(volatile unsigned int *)0x7527  /* Bottom reg of capture FIFO
             stack 6 */
#define EVBIMRA      *(volatile unsigned int *)0x752C  /* EVB interrupt mask reg A */
#define EVBIMRB      *(volatile unsigned int *)0x752D  /* EVB interrupt mask reg B */
#define EVBIMRC      *(volatile unsigned int *)0x752E  /* EVB interrupt mask reg C */
#define EVBIFRA      *(volatile unsigned int *)0x752F  /* EVB interrupt flag reg A */
#define EVBIFRB      *(volatile unsigned int *)0x7530  /* EVB interrupt flag reg B */
#define EVBIFRC      *(volatile unsigned int *)0x7531  /* EVB interrupt flag reg C */

/* I/O space mapped registers */
#define FCMR portFF0F                  /* Flash control mode register */
ioport unsigned int portFF0F;          /* C2xx compiler specific keyword */
#define WSGR portFFFF                  /* Wait-state generator reg */
ioport unsigned int portFFFF;          /* C2xx compiler specific keyword */
```

```
/*****************************************************************
* Filename: edm1.h                                               *
*                                                                *
* Author: Azli Yahya, Loughborough University.                   *
*                                                                *
* Description: Header file for EDM C-code.                       *
*****************************************************************/

/*PROTOTYPE DEFINITIONS*/

void error_input();
void run_system();
void disable_interrupts();
void dsp_setup();
void event_manager_init();
void enable_interrupts();
void start_main_program();
unsigned int ReadADC();
void select_Imax();
void delay_sec();
void delay_mS();
void lcd_clear_display();
void lcd_initialize();
void lcd_8_bit();
void display_main_menu();
void display_on_time();
void display_off_time();
void display_on_off();
void display_current();
void display_edm_controller();
void display_press_start();
void sensor();
void current();
void system_running();
void save_settings();
void movecursor();
void lcd_cmd();
void cursorposition();
int next_cur_on(void);
int next_cur_off(void);
int next_cur_curr(void);
int inc_no1(void);
int increament_on1(void);
int increament_on2(void);
int increament_on3(void);
int increament_off1(void);
int increament_off2(void);
int increament_off3(void);
int increament_current1(void);
int increament_current2(void);
int up_timetest(void);
int down_timetest(void);

void error_controller();
void inc_no3();
void lcd_data();
void view_on_time();
void view_off_time();
void view_current();
void on_off_time();
void up_time();
void down_time();
void view_up_time();
void view_down_time();
void run_settings();
void system_stop();
void count_0();
void hysteretic_current();

void pwm_0();
void current();
int sense_pushbutton();
void setting_off();
void setting_on();
void generate_Q6_Q7();
void lcd_display_main();
void lcd_display_function();
void manual_mode();
void fault_system();
void faulty();
void reverse_motor();
```

```
/*****************************************************************
* Filename: edm.cmd                                              *
*                                                                *
* Author: Azli Yahya, loughborough University.                   *
*                                                                *
* Description: LF2407A DSP memory allocation for EDM C-code.     *
*****************************************************************/

MEMORY
{
 PAGE 0 :   /* program memory */
  VECS:       org=00000h,    len=00040h
  FLASH:     org=00044h,              len=07FBCh
  EXTPROG:  org=08800h,               len=07800h

 PAGE 1 :   /* data memory */
  B2:                  org=00060h,              len=00020h
  B0:                  org=00200h,              len=00100h
  B1:                  org=00300h,              len=00100h
  SARAM:               org=00800h,              len=00800h
  EXTDATA:  org=08000h,    len=08000h       }

                       /* specify sections */
SECTIONS
{
/* Sections generated by the C-compiler */
  .text: >        FLASHPAGE  0
  .cinit: >       FLASHPAGE   0
  .const:         >       B1              PAGE  1
  .switch: >      FLASHPAGE  0
  .bss:  >        B1              PAGE  1
  .stack:         >       SARAM           PAGE  1
  .sysmem:        >       B1              PAGE  1

/* Sections declared by the user */
  vectors:        > VECS                  PAGE  0 }
```

APPENDIX 5

Pre-existing gap voltage and current pulse power generator

The pre-existing gap voltage and current pulse power generator circuit shown in Figure 4.8 was designed to supply a specify current into the gap during machining. The topology is a buck converter consisting of Q_1, D_1 and inductor L with parallel switch Q_2 and feedback diode D_2. The inductor current is controlled using the hysteretic method described in section 3.1.4. The inductor current I_L is measured with the aid of LEM Hall effect current transducer. The output of the current transducer is the isolated V_L proportional to the inductor current. The PWM pulses from the hysteretic controller in eZdsp is fed to Q_1 from the IOPE0 port via opto-isolator1 as shown in Figure 4.6. The 15V floating supply for Q_1 gate drive is obtained from a circuit shown below in Figure A6.1.

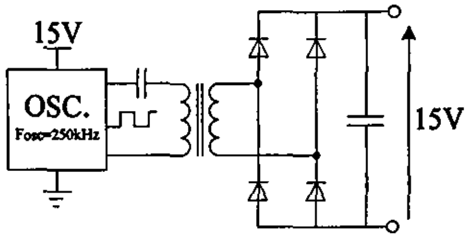


Figure A6.1 Floating supply for Q_1 gate drive

This consists of a simple square wave oscillator running at 250kHz feeding a primary winding of a small Toroidal transformer (OD=16mm) via a 1μF ceramic DC blocking capacitor. The secondary voltage of the transformer is full wave rectified and filtered by a 1μF ceramic capacitor. The buck converter at any time can see a short circuit when Q_2 is 'on', a load when the gap breaks down and an open circuit when the gap does not breaks down. During short circuit at the buck converter output, the inductor current flows through Q_2. When the gap breakdown, the inductor current flows through the gap as shown in Figure 4.8. During open circuit, the inductor current is feedback to the supply via D_2, thus limiting the maximum gap voltage to V_d.

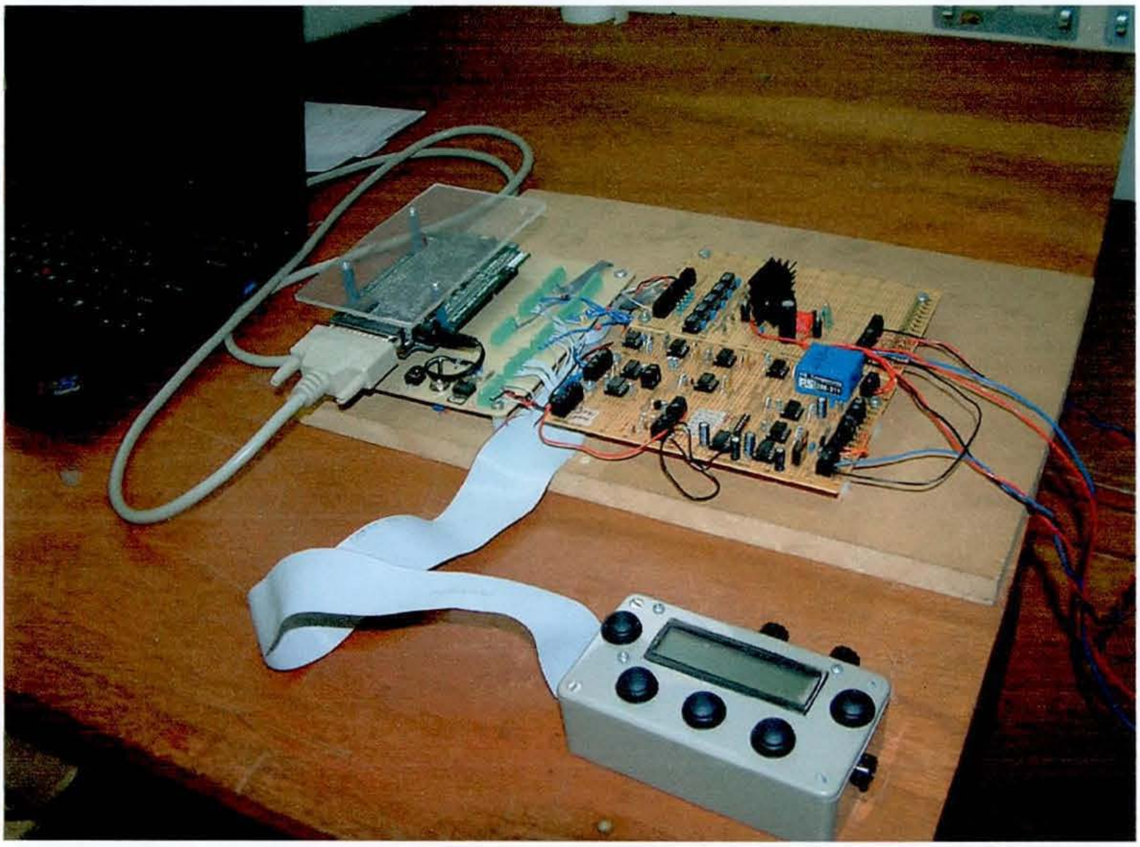
APPENDIX 6

Hardware components list

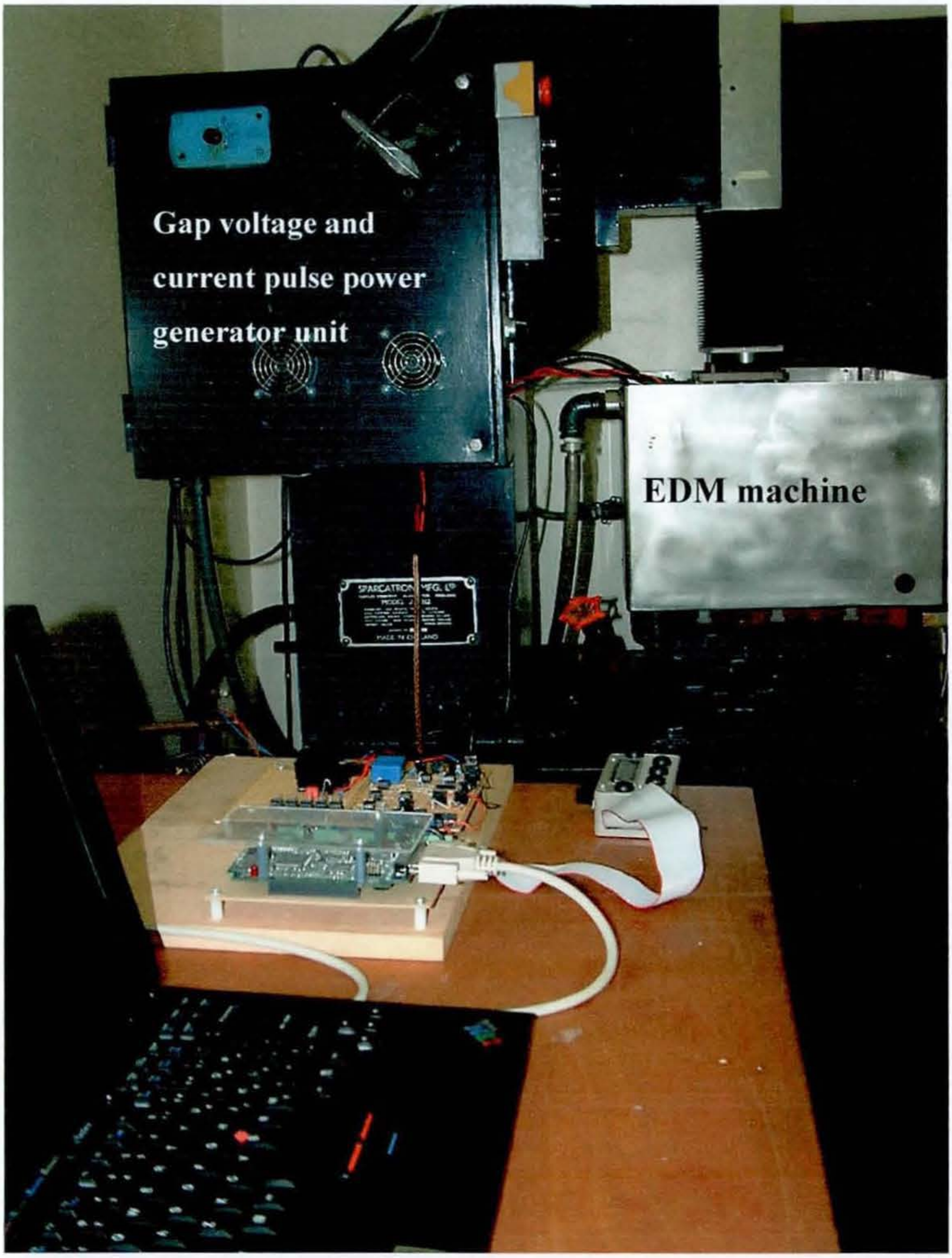
Components	
Semiconductor	
eZdsp^{TM} TMS320LF2407 microcontroller	1 unit
LCD PC1602-H series Power Tip	1 unit
Transceiver 74LVX3245	2 units
L6203 DMOS Full Bridge Driver	1 unit
LM317 voltage regulator	1 unit
Transistor Tx_1 to Tx_4	ZTX302
Diode zener D_{z1} to D_{z4}	BZX79C10
Diode zener D_{z5}	BZX79C3V3
Operational amplifier op-amp1 to op-amp 5, op-amp7	TL071
Operational amplifier op-amp6	MAX473
Opto-isolator opto-iso.$_1$, opto-iso.$_2$	HCPL3120
Opto-isolator opto-iso.$_3$ to opto-iso.$_5$	740L6010
Resistor	Ω, 1%, 0.6W
R_1	220
R_2	680
R_3 to R_7	20k
R_9, R_{11}, R_{21}	10
R_8, R_{10}, R_{12}, R_{14}, R_{16}	560
R_{13}, R_{15}, R_{17}	330
R_{18}, R_{19}, R_{48}	1k

R_{20}	0.15, 5%, 5W
R_{22}	4.99k
R_{23}, R_{24}, R_{33}, R_{35}, R_{37}, R_{41}, R_{45}, R_{46}, R_{49}	10k
R_{25}	1.58k
R_{26}	1.6k
R_{27}	453k
R_{28}	4.7k
R_{29}, R_{50}	8.2k
R_{30}	1.8k
R_{31}, R_{32}, R_{34}, R_{36}, R_{38}, R_{47}	3.3k
R_{39}, R_{40}	510
R_{42}	6.49k
R_{43}	3.24k
R_{44}	27k
VR_1, VR_3	5k
VR_2	100k
<u>Capacitor</u>	<u>F, 25V</u>
C_1, C_2 (ceramic)	1μ
C_3 to C_7, C_9, C_{17} (ceramic)	0.1μ
C_8 (electrolytic)	680μ
C_{10} (plastic)	220n
C_{11}, C_{13} (plastic)	15n
C_{12} (ceramic)	22n
C_{14} (ceramic)	10μ
C_{15} (ceramic)	0.47μ
C_{16}, C_{18} (ceramic)	470n

Current transducer	1 unit
LEM LTS25 NP	1 unit
LEM HX 50-NP	
Push button switches	
Series 59 - miniature sealed	5 units
DC servomotor	
M818T(031) San Driver	1 unit



Hardware photo shot showing user interface device, analogue processing and interface circuit, eZdsp and PC.



Hardware photo shot showing eZdsp EDM controller attached to the EDM machine

LIST OF PUBLICATIONS

1. Yahya A and Manning C D 2003 Modelling and Simulation of Die-sinking Electro Discharge Machine *PREP2003 Exeter, UK.*
2. Yahya A and Manning C D 2003 Modelling and Simulation of Die-sinking Electro Discharge Machine *Third Asian Conference on Industrial Automation and Robotics Bangkok, Thailand* pp. 1-4.
3. Yahya A and Manning C D 2003 Modelling, Simulation and Controller Design for Electro Discharge Machine System *Electronic Systems and Control Division Research (mini conference), Electronic and Electrical Dept. Loughborough University, UK.*
4. Yahya A and Manning C D 2004 Determination of Material Removal Rate of an Electro-Discharge Machine Using Dimensional Analysis *Journal of Physics D: Applied Physics* 37 pp. 1467-1471.
5. Yahya A and Manning C D 2005 Material Removal Rate Modelling of an Electro Discharge Machine *Plasma and Pulse Power Group (mini conference), Electronic and Electrical Dept. Loughborough University, UK.*