AN INVESTIGATION OF IMPROVED MYOELECTRIC PROSTHESIS CONTROL USING MICROPROCESSORS

by

Peter D. Hortensius

A Thesis
presented to the University of Manitoba
in partial fulfillment of the
requirements for the degree of
Master of Science
in
Electrical Engineering

Winnipeg, Manitoba, 1984

AN INVESTIGATION OF IMPROVED MYOELECTRIC PROSTHESIS CONTROL USING MICROPROCESSORS

BY

PETER D. HORTENSIUS

A thesis submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

ABSTRACT

The feasibility of a microcomputer based myoelectric limb controller is demonstrated. This limb controller consists of two microprocessors; one responsible for data acquisition and motor control and the other for use as a signal processor. A four function prosthetic limb (hand, wrist, elbow, and humeral) intended for use by above elbow amputees is controlled by the limb controller which uses only 410 mA of current from a 7.2 volt battery. Software implemented on the limb controller has duplicated the present technology in clinical myoelectric limb controllers (i.e. two and three state control) and presents the possibility for more advanced uses of the prosthetic limb such as multifunction and preprogrammed movements. A four coefficient autoregressive model of the electromyographic signal is implemented on the limb controller, but it does not presently provide sufficiently accurate control for use in a clinical system.

ACKNOWLEDGEMENTS

This thesis would not have been possible without the assistance of several people. I would first like to thank the staff at the Rehabilitation Center for Children for their many hours of effort in creating some of the equipment used to develop and test the prosthesis. I would especially like to thank Mr. F. Hreczuch for his aid in technical suggestions and acquisition of the various components used for the limb controller, Mr. A. O. Quanbury for his supervision while I was at the Rehabilitation Center, and to Mr. B. Nichol for constructing the mockup prosthetic limb used for this project. Several people at the University of Manitoba also deserve mention; Dr. S. Onyshko without whose aid and advice I would never have been able to write this thesis; Dr E. Shwedyk for his advice in the area of signal analysis, Dr. A. B. Thornton-Trump for the use of his computing facilities, and to the technical staff of the Department of Electrical Engineering.

I would also like to thank those who provided funding for this project. The Rehabilitation Center for Children which funded the prosthetic limb and limb controller components and the National Sciences and Engineering Research Council for personal funding.

DEDICATION

This thesis is dedicated to the Glory of God and to my wife, Gloria, whose love and patience during the trying times of this thesis made the problems easier to face.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

ADC	Analog to Digital Converter; a device that converts a analog voltage input into a digital output proportional to its magnitude.
Ahr	Ampere-hour; unit of electricity, equal to 1 ampere of current delivered over one hour.
Bit	Basic unit of measurement for computer memories. One bit stores a single unit of information.
Bus	A set of logic lines that transfer information to various components within a computer.
Bus transceiver	A device which can transfer information, in either direction, from one computer bus to another computer bus based on a control input.
Byte	A unit of size for computer memories equal to 8 bits.
CD80C86	CMOS version of the popular Intel 8086 microprocessor manufactured by the Harris Corporation. Used in this project as the signal processor.
Clocked latch	A device that transfers the information present at its input to its output when the control input changes logic state.
CMOS	Complementary Metal Oxide on Silicon; a method of chip fabrication that, along with other differences, uses much less power than other chip fabrication processes.
CSA	Canadian Standards Association; a group that sets specifications for manufactured goods that will be used by the public and industry.
EMG	Electromyogram; a signal that can be detected on the skin surface when a muscle is contracted.

EPROM	Electrically Programmable Read Only Memory; a nonvolitile computer memory that can be programmed by electrical pulses.
HC or HCMOS	High speed CMOS; an implementation of CMOS logic that combines high speed with low power usage.
Hz	Hertz; a unit of measurement equal to the number of times an event occurs per second.
K	A measure of computer storage capacity equal to 1024.
MC146805E2	Low power CMOS version of the MC6805 microprocessor manufactured by Motorola. Used in this project as the data acquisition supervisor, the motor controller, and the controller supervisor.
Memory bank	A designation for a block of computer memory that acts as a single unit of memory.
MOSFET	Metal Oxide on Silicon Field Effect Transistor; a transistor with low voltage drop and low gate drive current (i.e. high input impedance).
NMOS	N carrier, Metal Oxide on Silicon; a chip fabrication process that is used in many modern microprocessors.
Nyquist criterion	A fundamental theory of signal processing which states that a signal must be sampled at a minimum sampling rate of two times its maximum frequency component to remove aliasing.
On-chip	An indication that the device it refers to is physically contained on the chip presently being discussed.
On resistance	The resistance of a device when it is turned on, this usually corresponds to the voltage drop across the device.
Otto Bock	A large manufacturer of orthopaedic aids such as prostheses.

RAM	Random Access Memory; a computer memory that can be accessed in a random order. Memory data can be changed during program execution so it is used for variable storage but it is volitile in nature.
TTL	Transistor Transistor Logic; a chip fabrication process commonly found in many digital logic circuits.
usecs	A unit of measurement equal to one millionth of a second.
VLSI	Very Large Scale Integration; a technique used in the manufacturing of computer components that combines many smaller components into one component.
Word	When used in reference to computer memory is a unit of size equal to 16 bits.

CHAPTER 1

INTRODUCTION

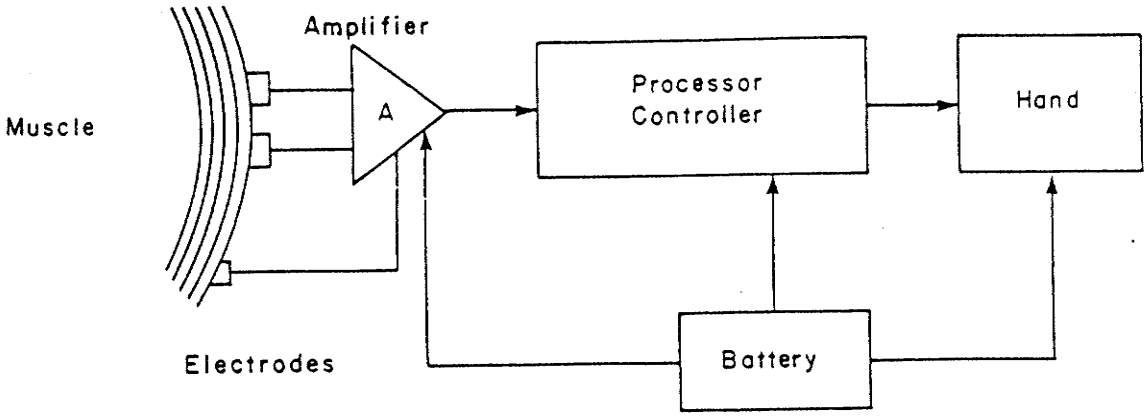
From the beginning of time human beings have used tools to overcome their deficiencies in dealing with their environment. From the cutting edge of the knife to the automobile these tools have become extensions of their natural abilities. However, for the amputee, before many of these tools can be fully used a more basic tool must be made, a prosthetic limb to replace the natural limb that is missing. These prosthetic limbs have ranged from the "peg leg" of Treasure Island's Captain Long John Silver and the hook of Peter Pan's Captain Hook to the complex powered upper and lower limb prostheses of today. While many advances in the area of self contained limb prostheses have occurred, it is generally agreed that there is still a great deal of research and development that must be conducted into the control and powering of prosthetic limbs to achieve natural limb function and cosmetic appeal. This is especially true for upper limb prostheses where the functionality and cosmetics are more readily evident than in the lower limb.

The design of an upper limb prosthesis and limb controller is a difficult task that must take into account many different factors. An excellent discussion on the

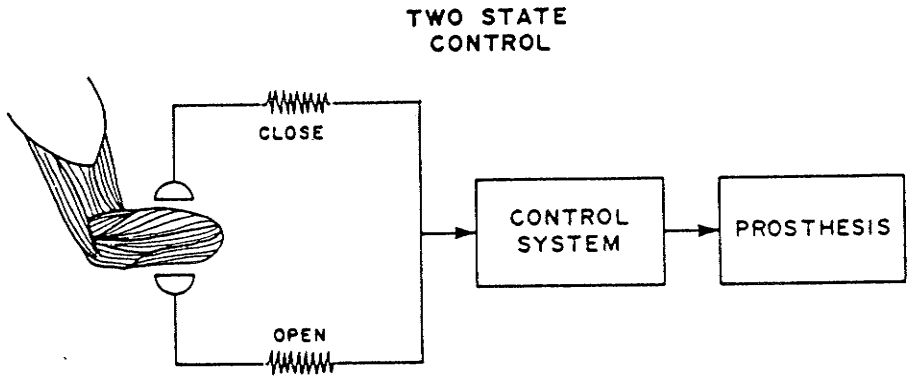
problems associated with the development of an upper limb prosthesis is given by Jacobsen et al. [1]. One of the most promising areas of research for upper limb prostheses is the use of biopotential signals, such as the electromyographic signal, to control the prosthesis. This method uses muscles that still are present on the arm, but can no longer perform a useful function, to control the artificial limb. For example, for a below elbow amputee, an artificial hand could be controlled by the now useless hand and wrist flexors and extensors. As either muscle group contracts the resulting electromyographic signal can be detected and control accomplished (i.e. open or close hand). The first electromyographically controlled prosthesis was designed by Reiter [2] in 1948. Since then many other attempts have been made at achieving a more complete prosthesis but only a few commercial systems are now in existence. Notable advances in externally powered (i.e. battery or some other form of energy) myoelectric artificial limbs since Reiter's first attempt include: the IBM arm [3], the Boston elbow [4], the Veteran Administration elbow [5], the Otto Bock hand [6], the Fidelity hand [7], [8], the Italian arm [9], the New York University elbow [10], the Variety Village elbow [11], the University of New Brunswick hand [12], and the Utah Arm [13]. Recent work to produce new commercial prostheses has occurred in Japan [14], [15], [16], at the University of Utah with the Utah elbow [17], and at several other centers around the world [18], [19], and [20].

While all the above advances in powered upper limb prosthetic limbs have increased either the availability, functionality, or cosmetic appeal of prosthetic limbs they all rely on essentially the same method of control when used with myoelectric limb controllers. This control scheme consists of amplifying the electromyographic signal of a suitable muscle to a useable voltage, calculating the power of the muscle activity using analog filtering techniques, using a predetermined threshold to determine whether or not there is muscle activity, and turning a joint motor on or off based on the muscle activity. This makes the interface between the user and the prosthetic limb essentially that of an on/off switch (i.e. muscle contracts - joint motor turns on, muscle relaxes - joint motor turns off). This technique also implies the need for a second controlling muscle to give joint motion in the opposite direction (i.e. "open" and "close" motions are needed in a practical prosthesis). Generally an antagonist pair of muscles, such as the biceps and triceps, are used as the controlling muscles. A block diagram of a typical myoelectric prosthesis control system is given in Figure 1.

Some of the newer limb controllers use three state control from each muscle (i.e. same muscle controls open, close, and off) and some five state control schemes have been proposed [21]. Yet, even with five reliable states of control complete natural control of prosthetic limbs will



Block diagram of a Myoelectric Control Unit



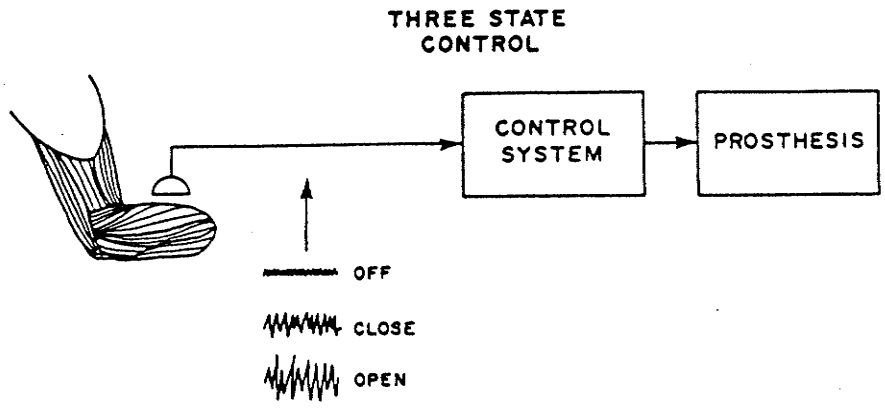


Figure 1 A Typical Myoelectric Prosthesis Control System.

never be achieved because of the many degrees of freedom which exist in the natural human limb. In fact, the natural human limb has approximately 87 gross degrees of freedom with many more internal degrees of freedom [22]. In addition, many amputees have problems with the remaining musculature of the amputated limb (e.g. spasticity, contractures, weakness, etc..) and cannot use the present day myoelectric prostheses properly if at all. What is needed is a control scheme that will allow the user to have more complete control of the prosthesis ranging from the direction of motion to the strength and speed of that motion. Yet this control scheme must also be more robust to problems that can affect the remaining musculature of the amputee. This obviously implies that robust limb controllers with many degrees of freedom must be developed for use with prosthetic limbs.

At the present time there are many promising control algorithms under development around the world ranging from observations on the biomechanics of natural limb motion to complex statistical methods that require a great deal of computational power. More information on some of these new techniques can be found by consulting the following references; Scott [23], Jacobsen [24], Wood [25], Simpson [26], Childress [27], Hogan [28], Deluca [29], Stein [30], Graupe [31], Sardis [32], Jerard [33], and Shwedyk [34]. However, many of these control schemes are computationally,

or decision, orientated and cannot be used in present myoelectric limb controllers. This is because present limb controllers use analog circuits, such as filters and threshold detectors, to operate. What is needed is to design, implement and test a prosthetic limb and limb controller upon which these control schemes can be implemented without redesigning the limb controller each time a new control algorithm is used. The obvious answer is to design a microprocessor based computer interface between the user and the prosthetic limb. This limb controller will allow complex and simple control algorithms to be implemented by merely changing the limb controller software which is much more cost and time effective than changing the limb controller hardware each time. The objective of this thesis is the design, implementation, and demonstration of such a myoelectric limb controller using microprocessor based computer hardware powerful enough to implement most of the above control algorithms. In addition, the feasibility or infeasibility of using microcomputers as myoelectric limb controllers is to be investigated.

1.1 Prosthetic Limb and Controller Overview

The goal of this thesis is to create the necessary hardware and software to implement a variety of modern myoelectric prosthesis control methods. The resulting computer will be used to show the feasibilty, or

infeasibility, of myoelectric prosthesis control using computers. However, it should be remembered that the resulting prosthetic limb and limb controller is created to be used as a demonstration project and design tool and not as a final end product. This means that in the final implementation of the prosthesis, sections of the present limb controller may not be necessary. Yet, a final product will not have to be redesigned. This is because the present design is set up to allow the removal of sections of the limb controller, that in the future may be deemed unnecessary, without necessitating a complete redesign of the limb controller.

The limb controller presently consists of five printed circuit boards each approximately 23 cm. (9 in.) square. These boards are stacked on top of each other creating final dimensions of 23 x 23 x 21 cm.. The device can be powered by any battery between 7.2 and 15 volts that can deliver at least 1.2 ampere-hours (Ahrs) of current and can withstand a maximal surge of 5 to 6 amps. There is an on/off switch that can be easily modified into any other form of switch, such as a microswitch, that can be installed in the prosthesis. As well, the limb controller presently has an on board battery charger with user adjustable output voltage and current levels.

The controlling computer of the limb controller is actually made from two microprocessors. One microprocessor, a CD80C86, is dedicated to signal analysis while the second microprocessor, a MC146805E2, controls the data acquisition of the myoelectric signal, controls the prosthesis motors, and acts as the limb controller supervisor. This allows the controlling computer to react much more quickly than if only one microprocessor were used to gather the data, process the data, and implement the desired control of the prosthetic limb. The two microprocessors communicate via a shared memory with the CD80C86 operating at a clock rate of 4.77 MHz and the MC146805E2 operating at 2.38 MHz. Memory consists of 4k words of control memory and 2k words of local variable memory for the signal microprocessor, 4k bytes of control memory and 128 bytes of local variable memory for the system supervisor and 4k words of shared variable memory set up into two banks of 2k words each. Four channels of myoelectric signals can be acquired at a maximum sampling rate of 10 kHz for one channel to 2.5 kHz for all four channels. The myoelectric signals are acquired using dry electrodes and are amplified to about 10 volts peak to peak before being digitized by a 12 bit analog to digital converter. The digitized data are then stored in the computer memory. Motor control is presently only "open/close" in nature but provisions have been made to allow for proportional control to be used in the future. This hardware allows the limb controller to operate in real

time running under most of the control algorithms mentioned in the preceding and succeeding pages.

While almost any upper limb prosthesis will demonstrate the principles needed for this thesis, an above elbow prosthesis is used because this type of prosthesis is where microprocessor control can provide the greatest increase in function. The prosthetic limb allows the user the three most used natural functions associated with the upper limb (elbow flexion/extension, forearm pronation/supination, and hand grasp/release). In addition, humeral rotation (normally associated with the shoulder but lost with most above elbow amputees [35]) is included as a fourth motion. The limb controller is connected to the user and prosthetic limb by an umbilical cord that passes the myoelectric signals to, and control signals from, the limb controller. A picture of the prosthesis showing both the prosthetic limb and limb controller is shown in Figure 2.

At present, the user effects limb movement and control in the following manner:

1. The user contracts or relaxes a muscle group(s) in a way that corresponds to the desired action.
2. The electromyographic signal(s) are digitized by the data acquisition section and stored in memory.

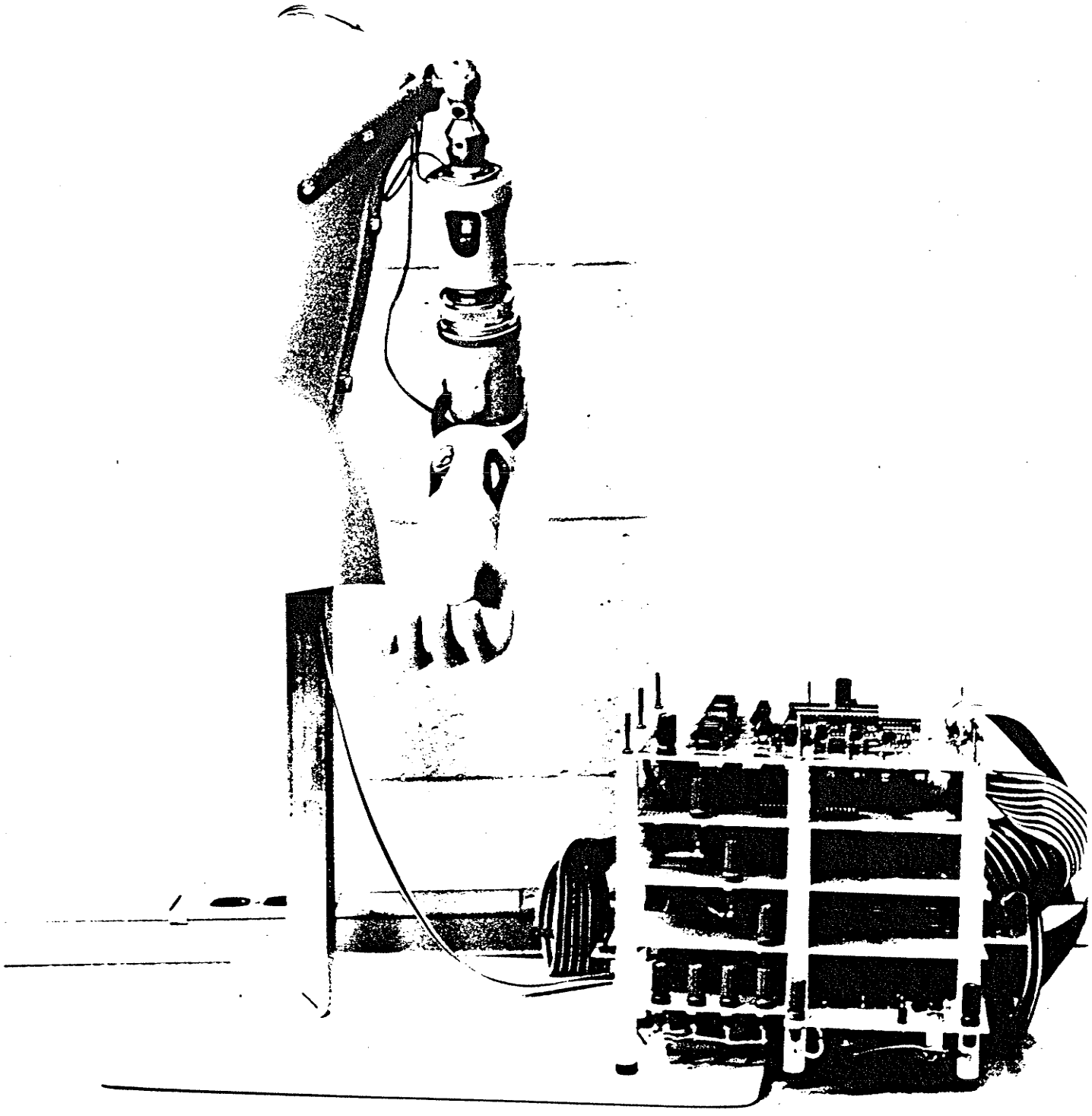


Figure 2 Picture of Prosthesis with Limb Controller.

3. The signal microprocessor detects the change in the muscle activity and informs the motor controller of the change in action required.
4. The motor controller effects the desired action by turning on or off the appropriate motor(s).

The following chapters describe the limb controller construction and operation in general terms. Three other technical manuals, not released with this thesis, are referenced in the Bibliography and give much more detailed descriptions of the limb controller hardware and software. A short overview of the research undertaken for this project is given below.

1.2 Overview of Research

The research undertaken for this thesis does not chronologically follow the order given in the following chapters. The first part of the investigation involved an extensive literature search to find other projects in the myoelectric limb control area that had similar goals or objectives. This survey revealed that very few attempts had been made to incorporate microprocessor based technology into a myoelectric limb controller. As well, any attempts that been made were dated by several years. This showed that the research into the limb controller hardware could not rely to any great extent on previous work. However, several

promising control algorithms requiring computer based decision and mathematical abilities were found. These were more closely investigated by computer simulation and the autoregressive model described in Chapter 3 was found to be the most promising. In investigating the control algorithms several programs were written to aid in future studies of control algorithms before they are actually implemented on the limb controller.

The next phase of the project involved the hardware design of the limb controller. A first prototype was designed and constructed. This prototype demonstrated that the fundamental principles behind the hardware would work. However, problems with the implementation necessitated the design and construction of a second more robust prototype. This is the limb controller presently available for use in further research and investigation.

The third phase of the project involved the writing of the limb controller software much of which was adapted from the software written to investigate the control algorithms. Original software included the operating system of the limb controller and the multichannel variance control algorithms.

The final phase of the research involved verification of the correct operation of the limb controller hardware and software. This was not a complete test, because no clinical

trials were made with the limb controller. However, these tests conclusively verified the correct operation of the prosthesis hardware and software design and demonstrated the potential of the limb controller.

CHAPTER 2

HARDWARE DESIGN

In designing a controller for any purpose the basic system requirements must first be defined. For this limb controller there are several obvious requirements and a few other more subtle ones. These are summarized below:

i) Power - The limb controller power requirement must be low enough to allow it to be powered by portable rechargeable batteries.

ii) Size - The limb controller must be small enough to allow it to fit into a prosthetic limb.

iii) Lightweight - The limb controller must be lightweight because it will be carried by the user for extended periods of time and also the connection between the user and prosthesis, through the remaining limb stump, can only bear loads not exceeding a few kilograms.

iv) Natural - The limb controller must be natural to operate, or easily learned, because the user will not use the prosthesis if it is difficult or awkward to use.

v) Flexible - The controlling output from the limb controller control must be flexible enough to allow the limb controller to be used with most types of externally powered upper limb prostheses.

vi) Powerful - The limb controller must be capable of handling the various processing requirements that may be placed on it by different control algorithms.

vii) Responsive - The limb controller must respond quickly to the user's commands to avoid any time lag that may detract from the effectiveness of the control algorithm.

The above requirements are all necessary in a final version of the limb controller but some are not necessary in a first prototype. In this thesis the fundamental problems

of the limb controller are addressed while those that are not so fundamental are left to be done in further work. The fundamental requirements are all of the above list except that of the size requirement. A few years ago the size constraint would have been a major constraint on the design, but with the advances in very large scale integration (VLSI) and hybridization techniques this has become a secondary design problem. The above requirements place some strong constraints on the design and these will be discussed in the following sections. What follows is not a detailed discussion of the limb controller hardware, but rather a more general discussion of the design of this myoelectric limb controller. Details of the limb controller implementation and operation are given in the limb controller hardware manual given in the Bibliography.

2.1 Power Supply

One of the most crucial components in a myoelectric prosthesis is the power supply to the controlling electronics and driving motors. An ideal power supply would deliver power to the limb for an entire day and would be able to be quickly recharged. Present myoelectric limb controllers use power supplies derived from batteries that last from 4 to 8 hours depending on the amount of use. These batteries typically are rated at about 250 milliamphours (mAhr) and deliver about 1 mA continuously to the

controlling electronics of the limb controller. The motors used to produce limb movement are the main power drains of a myoelectric prosthesis. The current drains vary from 50 mA under no load conditions for a hand motor to a maximum of 3500 mA when an elbow motor is in a stall condition. Therefore, for present clinical limb controllers the more the limb is used the shorter the battery life (i.e. battery life is heavily dependent on limb usage). However, for this prosthesis, the constant current drain on the battery by the limb controller is much larger than conventional limb controllers. Therefore, for the designed prosthesis (limb controller and prosthetic limb) the battery life is almost independent of use. Thus, one of the major problems in this project was to find a method of supplying power to the limb controller while still giving long battery life. An additional constraint on the battery selection is the weight and size of the battery. If a battery is too large or too heavy the prosthetist cannot place the battery in the prosthesis. In addition, the user of the prosthesis will object because of cosmetic reasons or because the prosthesis will be come too heavy to wear.

The battery selected for the prosthesis is a Saft Pack Battery number 120-198 rated at 1.2 volts and 1.2 Ahr. Six of these batteries were combined in series to form a 7.2 volt 1.2 Ahr battery. The overall weight is 500 grams which is just over the recommended maximum, of 450 grams,

suggested by a prosthetist. The discharge curve of the battery is shown in Figure 3. Notice that the battery voltage drops off quickly when the battery is about to be discharged. Thus, this choice of battery is good because the operating voltage is present for a maximal length of time. For prototype testing a stronger battery is used. This battery is a gel cell battery number GC826 rated at 8.0 volts and 2.6 Ahrs. The weight of this battery, 1100 grams, prohibits its use in clinical trials.

The limb controller requires a five volt supply because of the microcomputer components in it. This is generated by a five volt regulator placed across the battery supply lines. The regulator is rated to deliver 5 volts at 1.5 amps and so can easily supply the current requirements of the limb controller. There are other techniques of supplying power to a portable limb controller, such as running the control circuitry directly from the battery. This was not used in this limb controller for several reasons. The primary reason is to maintain voltage supply accuracy to the digital computer circuitry. This is necessitated because several of the computer components require the supply voltage to be within 10 % of their nominal supply voltage of 5 volts. A secondary reason to use a regulated 5 volt supply is to achieve some isolation between the joint motor voltage supply and the computer voltage supply. The joint motors of the prosthetic limb are powered directly from the battery

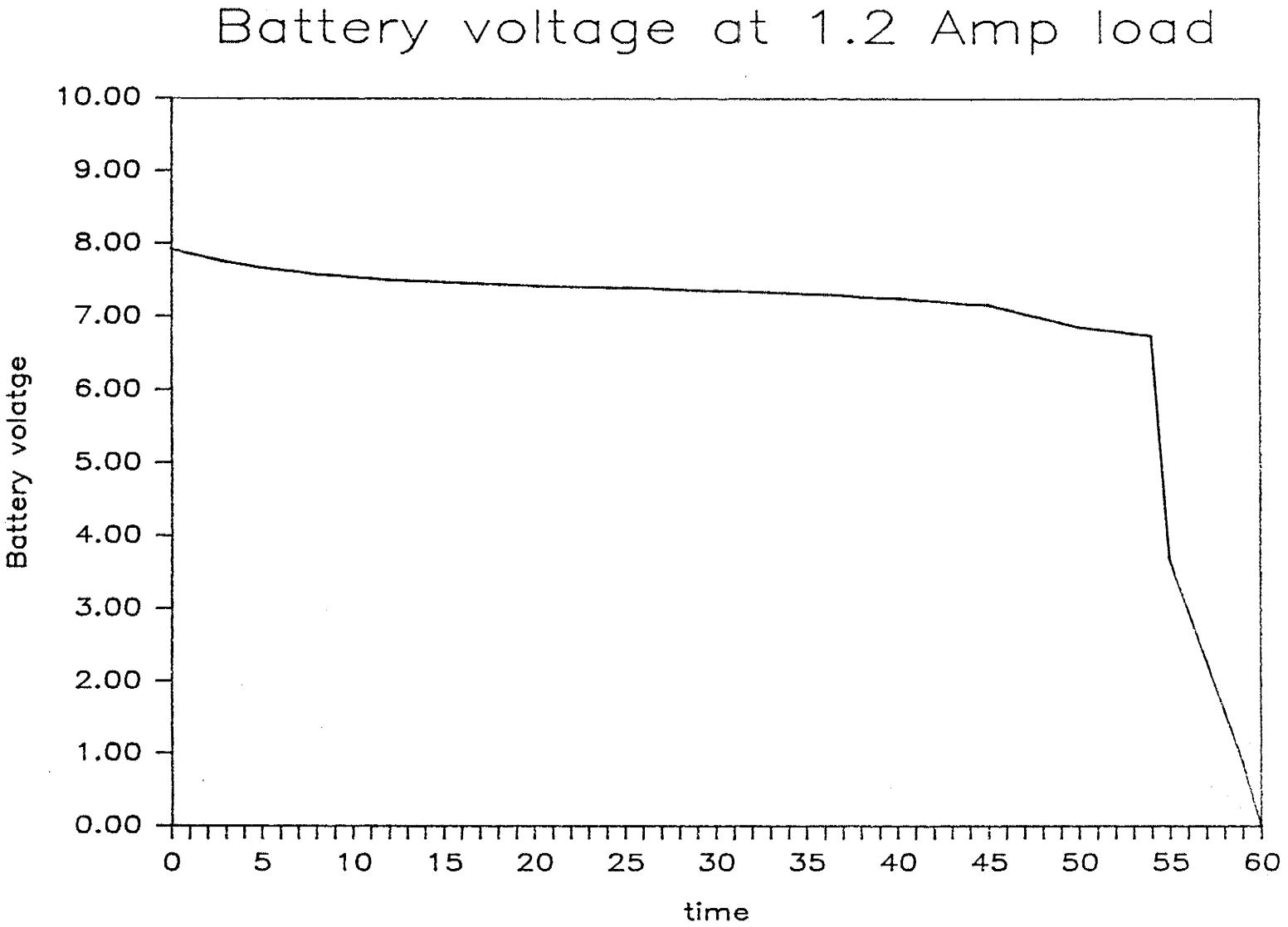


Figure 3 Discharge Curve of Prosthesis Battery.

and tend, like all dc motors, to produce electrical noise on their voltage supply lines. This noise can be a problem for digital circuitry and is disastrous for the accuracy of the analog circuitry. Isolation through a five volt regulator can help to remove this noise and so a regulated supply is used. It should be noted that noise from the limb motors is also present on the electrical ground of the limb controller but can be sufficiently reduced by proper grounding techniques.

It was found early in the development of the limb controller that electrical isolation between the digital and analog areas of the limb controller would be a problem. Digital circuits set up a great deal of switching noise on power supply lines that can reduce the accuracy of an analog circuit. A simple way to reduce this problem is to use separate power supplies for each area but this uses too much space, increases weight, and also increases the complexity. The best solution found was to use a dc-dc converter to isolate the two areas. The dc-dc power converter used converts 5 volts to +15 and -15 volts with up to 40 mA per supply line. This gives the isolation necessary for accurate operation of the analog circuits and meets weight and size constraints since the dc-dc converter is only slightly larger than a standard 24 pin chip. However, as shown in Figure 4, the converter is not very efficient at low current levels and uses the most current of any

component in the entire limb controller under static conditions (i.e. when no motors are running). For a future version of this limb controller a more efficient method of generating +15 and -15 volts is needed. The author experimented with several different dc-dc converters but found that many of the more efficient dc-dc converters either could not supply the necessary maximum of 20 mA per supply line, used too much space, or were too heavy.

For accurate digitization of the EMG signal a precision voltage reference of about 0.01% is needed. A dc-dc converter cannot supply this accuracy and so a 10 volt precision regulator is used. In addition, the +10 volts is also used to supply some of the motor on/off electronics, but because the gate drive current of the motor control MOSFETs (described later on in this chapter) is so low the accuracy of the reference is not impaired.

The current drain of the various components of the limb controller is outlined in Table 1. Notice that the dc-dc converter for the +15 and -15 volts uses 110 mA or about 27% of the current supplied to the limb controller compared to the control microcomputers which use only about 120 mA. Therefore, improvements in the power supply section to the limb controller, especially in the supply to the data acquisition section, will greatly increase the battery lifetime and thereby the amount of time that the prosthesis

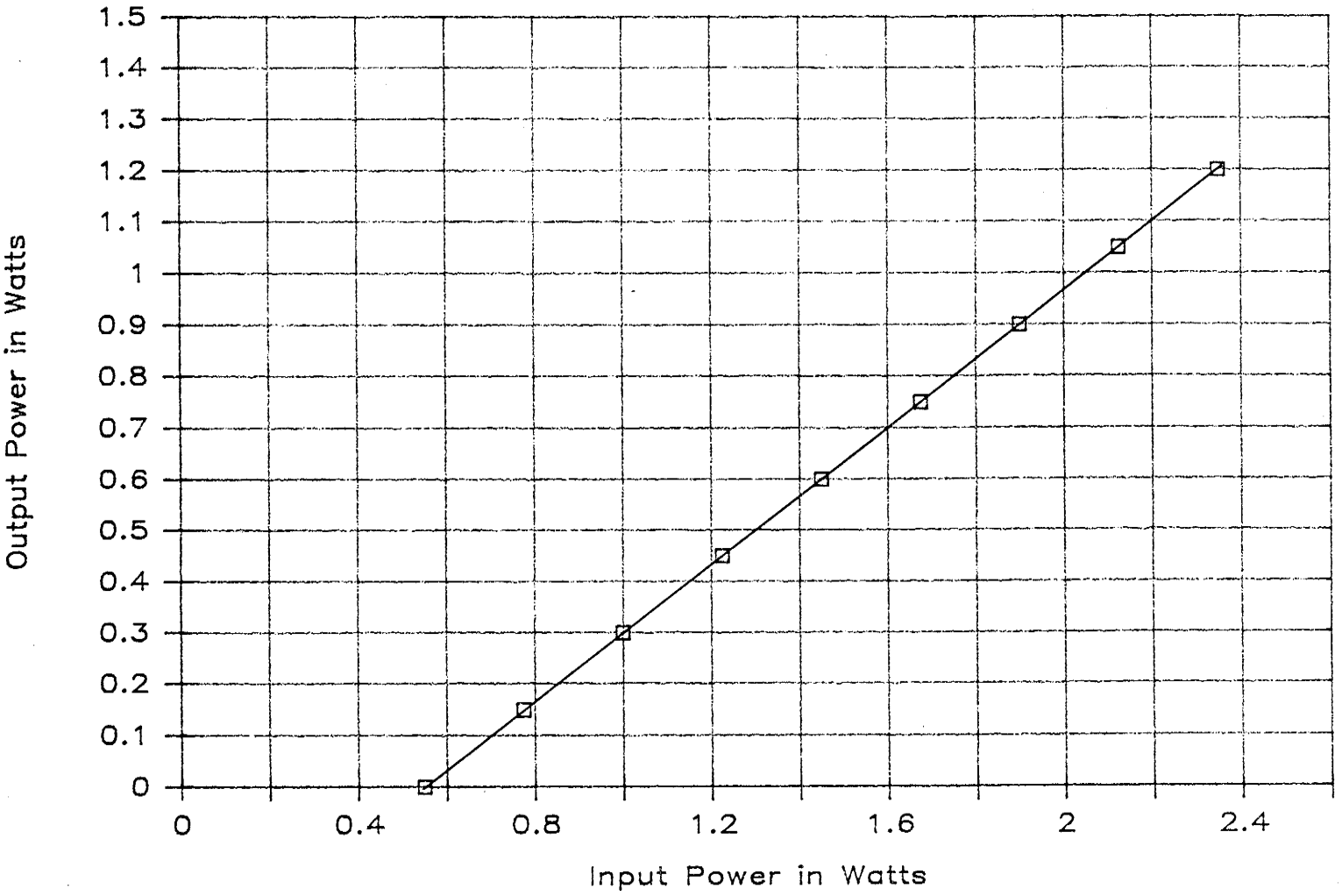


Figure 4 Output Power versus Input Power of dc-dc Converter.

is usable to the amputee between battery chargings.

An on board battery charger is included with adjustable output voltage and current levels. This charger facilitates ease of testing and increases portability of the limb controller because all that is required to charge the battery is to plug the ac power pac into a standard 120 volt ac outlet and place the limb controller into the charge mode.

2.2 Electromyographic Signal Acquisition

The electromyographic (EMG) signal is produced by action potentials as they pass along muscle fibers that have been activated. As more and more action potentials are evoked in the muscle (corresponding to a stronger contraction) the measurable voltage from the EMG signal increases. The frequency range of the EMG signal is on the order of 5 to 2000 hertz (Hz), with the main power band of the signal in the range from 20 to 200 Hz [36].

There are two ways of measuring the EMG signal. One method involves the placement of intramuscular needle electrodes. The other method involves the placement of electrodes on the skin surface. For this project the use of intramuscular needle electrodes was rejected mainly because of safety reasons as well as the difficulty in finding test

Device	current used from battery
Static load dc-dc converter	110 mA
Data acquisition board without electrodes	190 mA
Data acquisition board with all electrodes	270 mA
Motor control board	100 mA
MC146805E2 microcomputer board	45 mA
Shared memory board	30 mA
CD80C86 microcomputer board	45 mA
Total microcomputer load	120 mA
Total limb controller load without electrodes	410 mA
Total limb controller load with all electrodes	490 mA

Table 1 Limb Controller Current Usage.

subjects for such a prosthesis. When surface electrodes are used the levels of the EMG signals are low, having peak amplitudes of 0.1 to 1.0 mV. Therefore, a device that detects the EMG signal must have a fairly large gain and low noise at the input stage over the bandwidth of the EMG signal. As well, the interface at the skin surface must conduct as little current as possible from the skin (i.e. otherwise the current drain will, in effect, load the EMG signal down) so that the measurement is as true as possible. In addition, there are many safety factors set by the Canadian Standards Association (CSA) that are a requirement in any electronic device that is to be directly connected to a human being. For further information on these standards the reader should consult CSA document 125-1979.

For this project it was decided that four EMG signal channels would be made available to the limb controller. The use of four channels provides more information than the conventional single or dual channel myoelectric prostheses, and also increases the number of movements that can be selected by the user. In addition, some of the new EMG signal processing algorithms perform better with more electrode sites [37]. Therefore, the use of four channels will allow further research into multichannel EMG signal processing. However, the use of more than four channels creates a very complex electrode placement problem for the prosthetist and also may provide too many variables for the

limb controller to examine on a real time basis. Even the use of four electrode sites creates severe problems, but prosthetists have indicated that the problems could be overcome if the increased functionality of the prosthetic limb merited the extra effort.

The electrode at the skin interface is a standard differential electrode made by Otto Bock (a large manufacturer of artificial limbs and other aids for the handicapped) and is used because of its proven durability, low dc offset, availability, and its familiarity to medical personnel working with amputees. As well, the electrode is easily positioned in the prosthetic limb and can be modified by attaching different electronic circuitry to the electrode. All electrodes used in modern myoelectric prostheses are dry electrodes (i.e. no conducting gel is used between the skin surface and the electrode) because prolonged use of conducting gel may cause skin reaction problems, not to mention the inconvenience of placing conducting gel on the electrode each time it is to be used. Dry electrodes are inherently noisy because of poor electrical contact between the electrodes and the skin surface. Therefore, good electrical contact to the skin should be made by snugly pressing the electrode to the skin surface. Noise problems can also be reduced to acceptable levels by amplification right at the electrode site, thereby removing the 60 Hz noise problems often associated with low

impedance cables leading away from a high impedance source.

The above problems are inherent in dry electrodes placed on the skin surface and will not be solved until a better EMG detection technique for use with prostheses and not requiring conducting gel is devised. An example of such a technique is the implanting of electrodes in the muscle site and transmitting the EMG signal to a receiver on the prosthesis [38]. This method has been demonstrated in a laboratory setting, but it is not yet clinically usable for prosthetic devices.

All components used for amplifying the EMG signal must have low power consumption and this necessitates use of CMOS and other low power operational amplifiers. Initially it was thought that an electrode amplifier from an existing myoelectric limb controller could be used, but in these amplifiers the EMG signal is filtered as it is amplified. For this project it was decided that the filtering of the EMG signal must be as flexible as possible so as not to preclude an EMG signal processing technique because of filtering at the electrode site. An additional benefit is the possibility of using digital filters implemented in the software of the limb controller to aid in prosthesis control. All this necessitated the design of a custom dry electrode amplifier. A standard instrumentation amplifier with some minor modifications is used at the electrode site.

A photograph of the designed EMG amplifier and Otto Bock electrode is shown in Figure 5. Also shown in Figure 6 is the special electrode clip that was designed to aid in the placement of the electrodes on the skin surface during testing.

From the electrode site the amplified EMG signals are transmitted to the limb controller via cabling. Cables have a reputation for becoming a nuisance, but they were considered to be adequate for demonstration and initial testing purposes. In a future version of the limb controller a less cumbersome way of connecting the electrodes to the limb controller, such as radio transmission of the EMG signals, will have to be devised. As well, the size of the electrode amplifiers will have to be reduced to make the package easier to place in a prosthesis. This could be easily done by using smaller components (e.g. eighth watt resistors) in the electrode amplifiers.

At the limb controller the four channels of EMG are further amplified and filtered. The filtering is necessary to remove any extraneous low and high frequency waveforms that may have been induced on the EMG signal as it was acquired and amplified. In addition, filtering of the EMG signal will remove any aliasing that may be caused by the sampling rate of the analog to digital converter provided that the sampling rate meets the Nyquist criteria. This

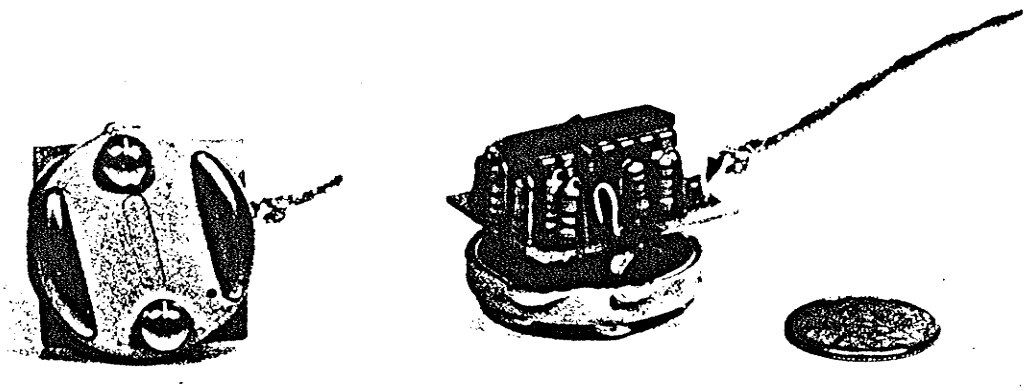


Figure 5 Otto Bock Electrode and Designed Electrode Amplifier.

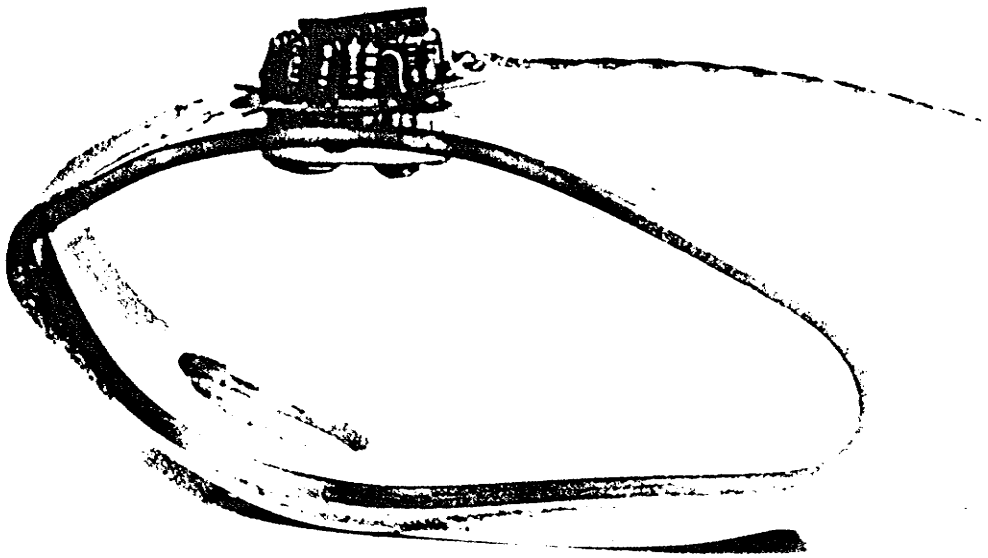


Figure 6 Picture of Electrode in Electrode Clip.

allows the easy addition of any special hardware filtering desired and makes a fully amplified EMG signal available at all times to the analog to digital converter. The use of four analog to digital converters is too inefficient from a space and power use point of view so the signals are multiplexed through an analog data selector before being digitized. The channel to be converted is selected by software in order to give increased flexibility to the conversion process. This also eases the use of a different number of channels between the different signal processing algorithms.

The analog to digital converter must be accurate and fast to give maximum signal processing ability to the limb controller. While only a few EMG signal processing algorithms suggest a conversion accuracy, or bit width, it was decided that a 12 bit analog to digital converter would be necessary to give sufficient accuracy and dynamic range to the conversion. Since four channels of EMG signal are used, the analog to digital converter should ideally be able to sample all four channels at the maximum frequency component. This would mean a 4 kHz sampling rate since the EMG signal has frequency components up to 2 kHz (i.e. must satisfy the Nyquist criteria). For all four channels to be converted this means that the analog to digital converter should be able to operate at a minimum of a 16 kHz, or 62.5 usecs per conversion. This sampling rate is easily within

the present technology for analog to digital converters, but only for higher power devices. For CMOS and other low power analog to digital converters the best available for this project is an analog to digital converter that can operate at a 10 kHz sampling rate. As well, the sampling rate can not be significantly increased by reducing the accuracy and using an 8 bit analog to digital converter. However, this limitation of sampling rate does not create a loss in performance of the limb controller. For single and dual EMG signal channels the limb controller will sample at the maximum rates needed and for four EMG signal channels valid sampling up to the 1.25 kHz components of the EMG signal can be given.

An obvious point comes into the picture at this time. If the limb controller were able to sample all four channels at 4 kHz what would happen to all the data? For a limb controller that must operate in real time the obvious answer is that it is unlikely that the limb controller would be able to do anything other than store this tremendous amount of data in a memory, assuming that the available memory is large enough for 16,000 twelve bit samples per second. As well, filtering on the data acquisition board is presently set up to output the main power band of the EMG signal (i.e. 20 to 200Hz). While this filtering can be easily adjusted it shows that the maximum possible sampling rate provided is fast enough to sample the EMG signal.

2.3 Limb Control Computer

Once the EMG signal has been acquired and digitized the decision as to what it means (i.e. initiate an action, complete an action, or remain dormant) must be made. For this myoelectric prosthesis the decision is made by the limb controller. The limb controller is divided into three different areas:

1) the data acquisition supervisory section.
2) the EMG signal processing section.
3) the motor controller section.

The data acquisition supervisor and the motor controller use the same microprocessor, the MC146805E2, while the signal processing uses a separate microprocessor, the CD80C86. The use of two microprocessors in a system such as this with the low power and small space requirements may seem somewhat extravagant, but the result is not as wasteful as may be initially thought. The MC146805E2 is an eight (8) bit microprocessor with sixteen (16) input/output lines. Therefore, it is not hard to interface this device to other digital devices because of the many control and data lines available. However, its data processing ability is poor because of a limited instruction set that is customized for use in control applications. As a result, the MC146805E2 can

become cumbersome and slow when doing any mathematical analysis, such as the simple 4 channel variance calculations or the complex autoregressive modelling described in Chapter 3. On the other hand, the CD80C86 is a sixteen (16) bit microprocessor that has no input/output lines but can do many mathematical functions including multiplication and division quickly and efficiently. Therefore, while the CD80C86 has good data processing abilities it requires additional circuitry to interface to the motors and data acquisition section because it has no input/output lines. Thus, a combination of the two devices, where the MC146805E2 gathers the data and implements the control output that the CD80C86 has determined is actually an efficient system. A block diagram of the limb controller is given in Figure 7.

Separating the limb controller into three different sections is useful because of the prototype nature of the device. In the future, if it is decided that only data acquisition and a simple signal processing algorithm is all that is really needed for a computer controlled myoelectric prosthesis, these sections can be easily "copied" from the present design without the need for redesigning the limb controller.

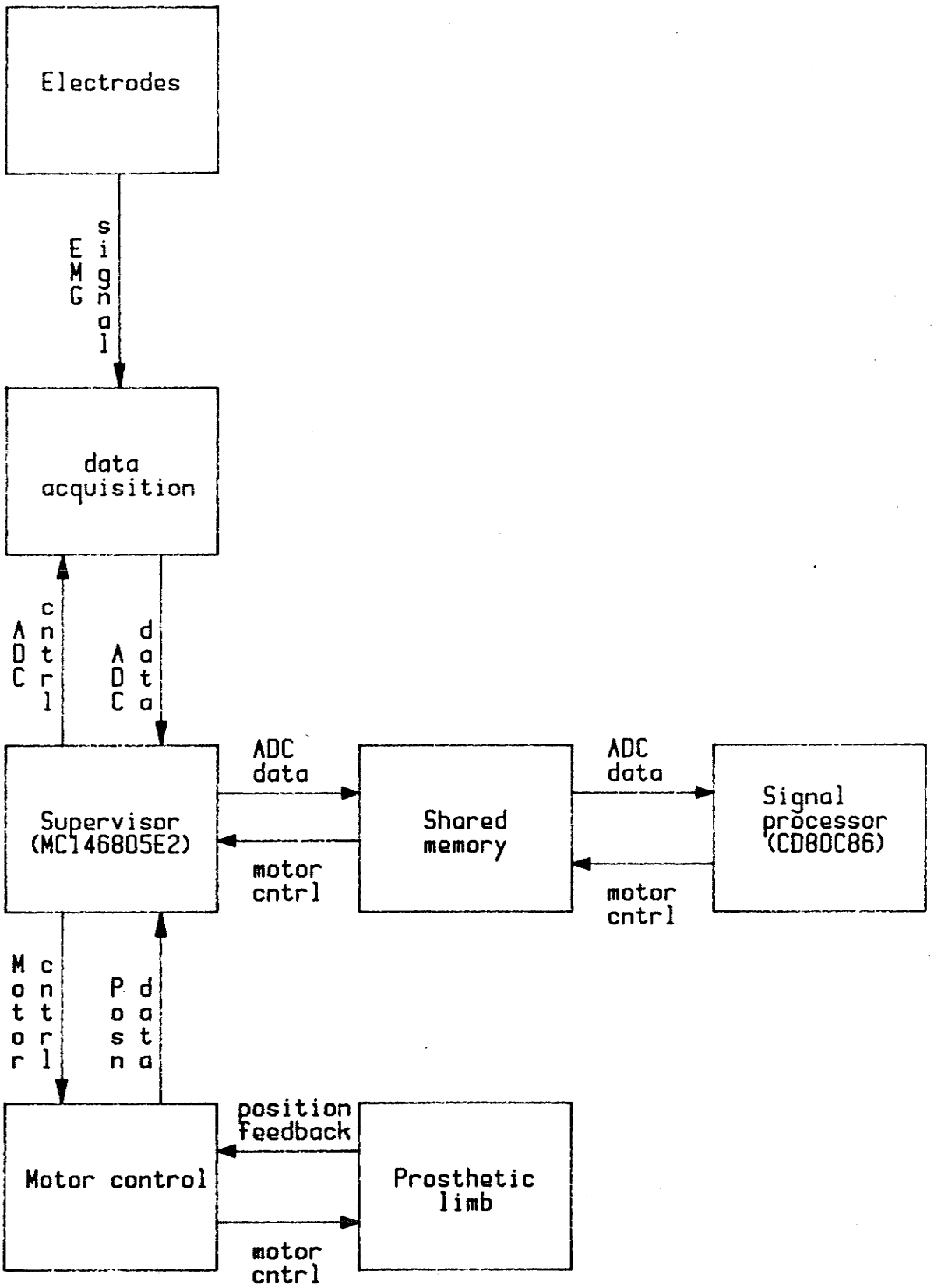


Figure 7 Block Diagram of Limb Controller Microcomputer.

2.3.1 Data Acquisition Supervisory Section

The MC146805E2 as outlined above was selected primarily because of its many input/output lines. However, it was also selected because it has a 13 bit external multiplexed address bus, 128 bytes of on-chip RAM, an on-chip timer and is a CMOS device using only 35 mW of power at a full operating speed of 5 MHz. The above attributes make it ideal for use in a prototype system where external addressing of the control memory is required to facilitate program development, yet data storage is small and does not necessarily require a separate data memory.

The MC146805E2 has complete control over the data acquisition section of the limb controller to give maximum flexibility in EMG signal acquisition. Any combination of the four signal channels may be used as well as different sampling frequencies for the four channels. While all of these abilities will probably never need to be used it adds to the potential of the limb controller to handle a variety of signal processing algorithms. It was decided that the internal timer of the MC146805E2 would be used to implement the sampling rates. The clock frequency of the analog to digital converter can be used to drive the internal timer and knowing the frequency of sampling desired, the timer can be set up to interrupt the microprocessor at the rate of the sampling frequency. The microprocessor will then output the

start conversion pulse and also select the data channel to be used. Three of the input/output lines are used to control the data channel selection and start of conversion.

The completion of the data channel digitization initiates a hardware interrupt of the microprocessor. This interrupt is cleared by the hardware when reading the A/D data register. The A/D data register is mapped to any odd (high data byte) or even (low data byte) address in the range 0000 to 0FFF. This reduces the hardware necessary to map the register to a unique address. The A/D data register is read from the data bus rather than from the input/output lines in order to leave the lines free for use in other limb controller interfacing (e.g. motor control). When the data is read it may either be placed immediately into the shared memory or be processed by the MC146805E2.

The MC146805E2 uses a 4k x 8 CMOS electrically programmable read only memory (EPROM) as its control memory. This is mapped to the upper 4k of memory and provides enough control memory to handle most processing and control requirements that may be placed on the MC146805E2. The shared memory is mapped to the lower 4k of memory and shares some locations with the on-chip RAM. Therefore, the internal memory of the MC146805E2 is mirrored in the shared memory. When reading the A/D data register the shared memory is disabled by setting an input/output line, but it is not

disabled when writing the data to the shared memory even if the input/output line is set. This saves on interfacing circuitry as well as time because the operation is essentially: set control line, read data, write data, clear control line.

Interfacing to the memories is done by clocked latches, bus transceivers, and of course control logic. The interfacing is over designed to protect the microprocessor and other components from damage due to errors in design. This was done in the prototype limb controller so that down time from destroyed components could be minimized. However, when the limb controller is to be implemented into another form it is suggested that several interfacing circuits be removed to save space and complexity.

2.3.2 EMG Signal Processing Section

The CD80C86 microprocessor is used as the signal processor for the limb controller. In addition, it may also be used for any numerically intensive analysis of any limb motion (e.g. multifunction preprogrammed movements with feedback) or for possible applications in robotics. The CD80C86 was selected because its instruction set lends itself to numeric processing (eg. built-in instructions for signed and unsigned 16 bit multiplication and division) and it also has a low power consumption of about 30 mA at 5 MHz.

The signal processing software that has been developed for the NMOS 8086 is quite extensive and all of this software may, of course, be used on the CD80C86 because it is a plug in replacement for the NMOS 8086. As well, the Harris Corporation manufactures CMOS versions of the full MCS 86 microprocessor family line. Therefore, the clock, reset, and other support circuitry needed by the CD80C86, or possibly needed in the future (e.g. an interrupt handler or bus arbitrator), are available in CMOS.

The CD80C86 uses a 4k x 16 control memory, made from two CMOS EPROMs, that is mapped to the upper 8k of memory. This provides sufficient control memory for most signal processing applications, but if more control memory is needed several large (64 kbit to 256 kbit) CMOS EPROMs have recently become available and these could easily be added to the limb controller. A local memory is required because of the interrupt table in the lowest 4 pages of memory needed by the CD80C86 for operation. It could be argued that this extra local memory will waste space, but some local memory is essential to give the CD80C86 some variable storage for data that could not be stored in the shared memory (eg. global variables, information about the last signal processed, position of prosthesis variables, etc...). With the size of memory storage that is now available in CMOS RAMs this does not significantly add to the physical size over using a microprocessor that would not need an interrupt

table.

The CD80C86 acts as a slave to the MC146805E2 in that it will execute only when told to by the MC146805E2. This may seem backward in that the low performance microprocessor is supervising the high performance microprocessor, but because of the dedicated nature of the task that is being performed by the CD80C86, this is in fact a logical choice. The nonmaskable interrupt (NMI) of the CD80C86 is used to start the signal processing. This interrupt is used over the maskable interrupt because when a maskable interrupt is invoked the CD80C86 requests an interrupt vector on the data bus. This of course requires additional support circuitry that should be avoided. The only way to avoid this circuitry is to use the nonmaskable interrupt which does not request an interrupt vector to be placed on the data bus.

Shared memory allows the two processors to communicate via variables in memory. This saves on additional circuitry to inform the two microprocessors of what the other is presently doing. Bus conflicts to the shared memory are removed by using one of the control lines from the MC146805E2 to switch the two microprocessors between the two banks of shared memory. Therefore, the MC146805E2 fills up one bank with data, while the CD80C86 is processing the data in the other bank of memory. This allows the two processors to work completely in parallel without having to worry about

bus conflicts. Since the shared memory is connected to each microprocessor in a mutually exclusive fashion the EMG signal data cannot be analyzed until all the data in the sample has been gathered. The operation is essentially: the MC146805E2 fills a bank of RAM with data; the CD80C86 is then selected to the memory and begins to process the signal, finally the MC146805E2 selects the bank back and implements the control requested. This creates a weakness in the limb controller in that EMG signal data cannot be analyzed as it is gathered, but a full sample of data must be available before analysis. However, the simplicity of this solution over the hardware and software complexity of having a large (4k x 16) CMOS two port shared memory makes this limitation acceptable. In addition, the size of the CD80C86 local RAM memory enables the CD80C86 to store enough information about the sample data that it is processing to allow the MC146805E2 to store only a portion of the data sample before switching memories and requesting the CD80C86 to start processing the EMG signal data. When the shared memory banks are switched, the CD80C86 can continue with the rest of the sample using the information stored in its local RAM memory.

2.3.3 Motor Control Section

The motor control is under the direct control of the MC146805E2. Using the input/output lines of the

microprocessor the 81 (four joints each with three possible motions) possible motions of the prosthetic limb can be selected. The motion selection output from the microprocessor is placed into a latch so that the motion can be frozen and the input/output lines used for another purpose. External logic is used to decode the output of the MC146805E2 and give proper joint motor direction. Eight control lines are used to control the limb motion with each joint motor using two lines. If both lines are high then an open motion is selected, if both lines are low then a close motion is selected and finally, if the two lines are unequal (i.e. one line high, the other low) then no motion occurs. This is easy to program and will aid in future work by allowing medical personnel, or others who are generally inexperienced in microcomputer programming, to make motion studies of the prosthesis.

The power to the motors is supplied through MOSFETs which give low "on resistance", low gate drive current, and high current carrying capability. This allows the motor MOSFET switches to be turned on or off directly by the CMOS decoding logic. The MOSFETs used were found to require an on voltage higher than 5 volts at the gate input to give low "on resistance" so the on voltage is shifted up to 10 volts to give the desired low "on resistance". The direction of the motor can be controlled by switching the polarity of the voltage that is supplied to the motor using the switching

technique shown in Figure 8. Therefore, each motor requires 4 MOSFETs to be operated.

A feature that will lead to improved controllability of prosthetic limbs will be to have positional feedback from the prosthetic limb. To aid in future studies in this area 4 analog to digital converters have been included to monitor the position of the limb. Several different methods of measuring the limb position were studied, but it was found that none of the methods could give reliable and robust feedback. Therefore, in the future, before the positional feedback capabilities of the limb controller can be fully used, a transducer to measure the limb position must be designed.

Four CMOS analog to digital converters are used for positional feedback. These are eight bit analog to digital converters which are referenced to the digital five volt supply. This leads to inaccuracies in the conversion, but because the joint position does not need to be accurately measured to eight bits of resolution these analog to digital converters can be used without a precision reference. If, in the future, joint positions need to be accurately measured then simply using a precision 5 volt reference across the battery supply lines to power the analog to digital converters will increase the accuracy to the full eight bits.

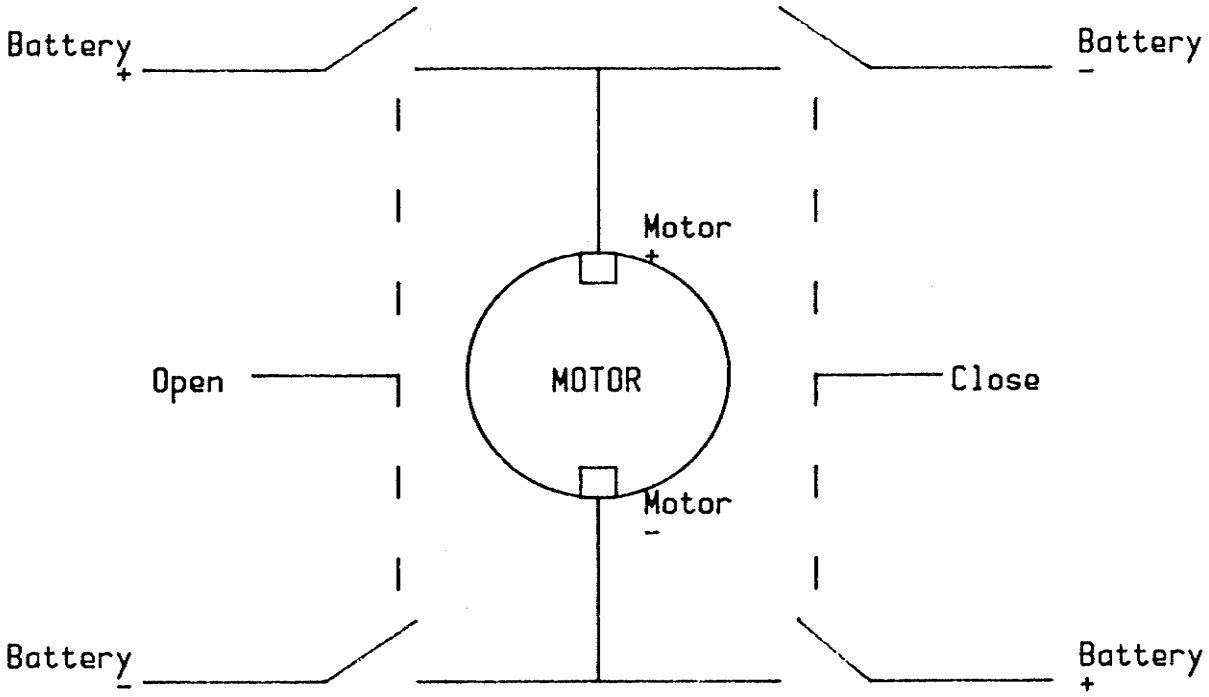


Figure 8 Polarity Switching for dc Motor Direction Control.

The motor position analog to digital converters are set to continuously convert the joint positions so that the MC146805E2 does not have to wait for the conversion to be completed when it requests joint position information. The conversion time is about 120 usec at a clock rate of 600 kHz. Therefore, the conversion frequency is 8.3 kHz which is more than adequate to follow joint motor motion. Only 7 bits of the analog to digital converter are fed into the MC146805E2 because the inaccuracies of the 5 volt voltage reference reduce the accuracy to 7 bits (remember it is not a precision reference but is the 5 volt supply line). However, this is still more than adequate to give accurate positional information about the prosthetic limb. The hand has a range of motion of about 135 degrees, giving a feedback resolution of about 1 degree. The results are similar for the elbow. For the humeral and wrist rotators the range of motion is 360 degrees which gives a feedback resolution of about 3 degrees.

The positional feedback data are fed back into the MC146805E2 on the same input/output lines as the motor control output, since the motor control can be frozen by the motor control data latch. The positional feedback analog to digital converters are not memory mapped because the address decoding circuits required would use too much space. Instead the analog to digital converter is selected by two of the

MC146805E2 input/output lines and the selected joint position data are fed into the MC146805E2.

It should also be noted that incorporating positional feedback capabilities into the limb controller opens the door for use in robotic applications where feedback is a necessary part of the control.

2.4 Limb Controller Implementation

The limb controller as seen from the above discussion is simply one microcomputer built from two microprocessors that uses EMG signals to control four motors. The limb controller could have been implemented with standard TTL and NMOS computer components, but because of the power constraints these components could not be used. The alternative is to use CMOS with its much lower power consumption and larger noise margins. However, until recently CMOS devices were slow and could not support the operation of a microcomputer bus. New advances in the design of CMOS circuits have led to the development of high speed series of CMOS products or the "HC" series as it is more commonly known. These components use much less power than TTL or NMOS devices and yet give approximately the same performance. The power saving of HCMOS over TTL or NMOS components is shown graphically in Figure 9. Notice that the HCMOS component's power consumption is dependent on the

frequency of operation while the other forms of logic are relatively fixed. This shows that to reduce power consumption by the limb controller only the speed of operation needs to be reduced. Since both microprocessors are CMOS they can be operated all the way down to static operation (i.e. very low frequency). Yet even at the clock rates of this limb controller the power consumption is about a tenth of the consumption of the other logic implementations. An excellent discussion of the relative merits of HCMOS over other forms of logic is produced by the Motorola Corporation [39] and the National Semiconductor Corporation [40].

The limb controller was initially implemented by a wire wrap board. This proved that the design was workable, but because of noise problems and also because the performance of the limb controller was slowly being degraded by the electrical problems inherent in wire wrap boards (eg. short circuits between the wires) it was decided that a more permanent and robust limb controller be implemented. This should not be used to infer that the wire wrap board was doomed to failure from the start because usually wire wrapping is an excellent method of building a single prototype system. An excellent discussion on the advantages and disadvantages of wire wrapped circuit boards is given in [41]. In this case the wire wrapped board failed because the layout of some of the circuits resulted in several potential

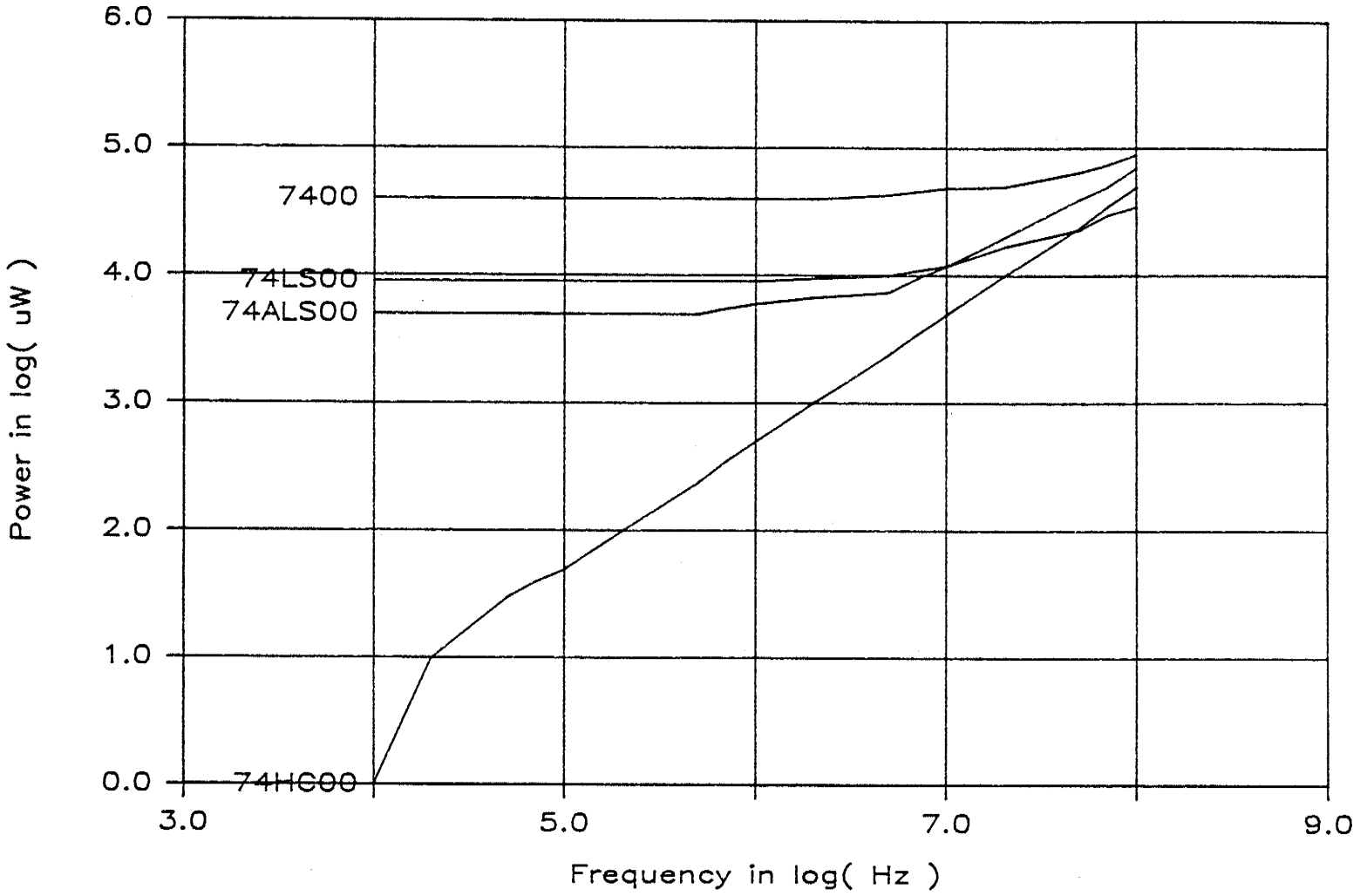


Figure 9 Power Consumption of Various Logic Families.

short circuits. As can be expected, following Murphy's Law, these potential short circuits began to short circuit. Photographs of the first limb controller prototype are given in Figures 10, 11, and 12.

A second prototype was designed and the result is the set of five printed circuit boards pictured in Figures 13 to 17. These boards are stacked upon each other to give a more workable size than if they are on one large printed circuit board. The printed circuit board layout negatives and the addendums for the layout are given in the limb controller hardware manual given in the Bibliography. The boards are stacked in order from top to bottom as:

1) CD80C86 computer board
2) Shared memory board
3) MC146805E2 computer board
4) Motor control and position feedback board
5) EMG signal acquisition and power supply board.

The boards communicate by using three vertical buses made from ribbon cable. The first vertical bus connects the EMG signal acquisition board to the motor control and position feedback board. This vertical bus passes the power and analog to digital converter data to the motor control and position feedback board and receives control data for the EMG signal conversion and data register output. The next

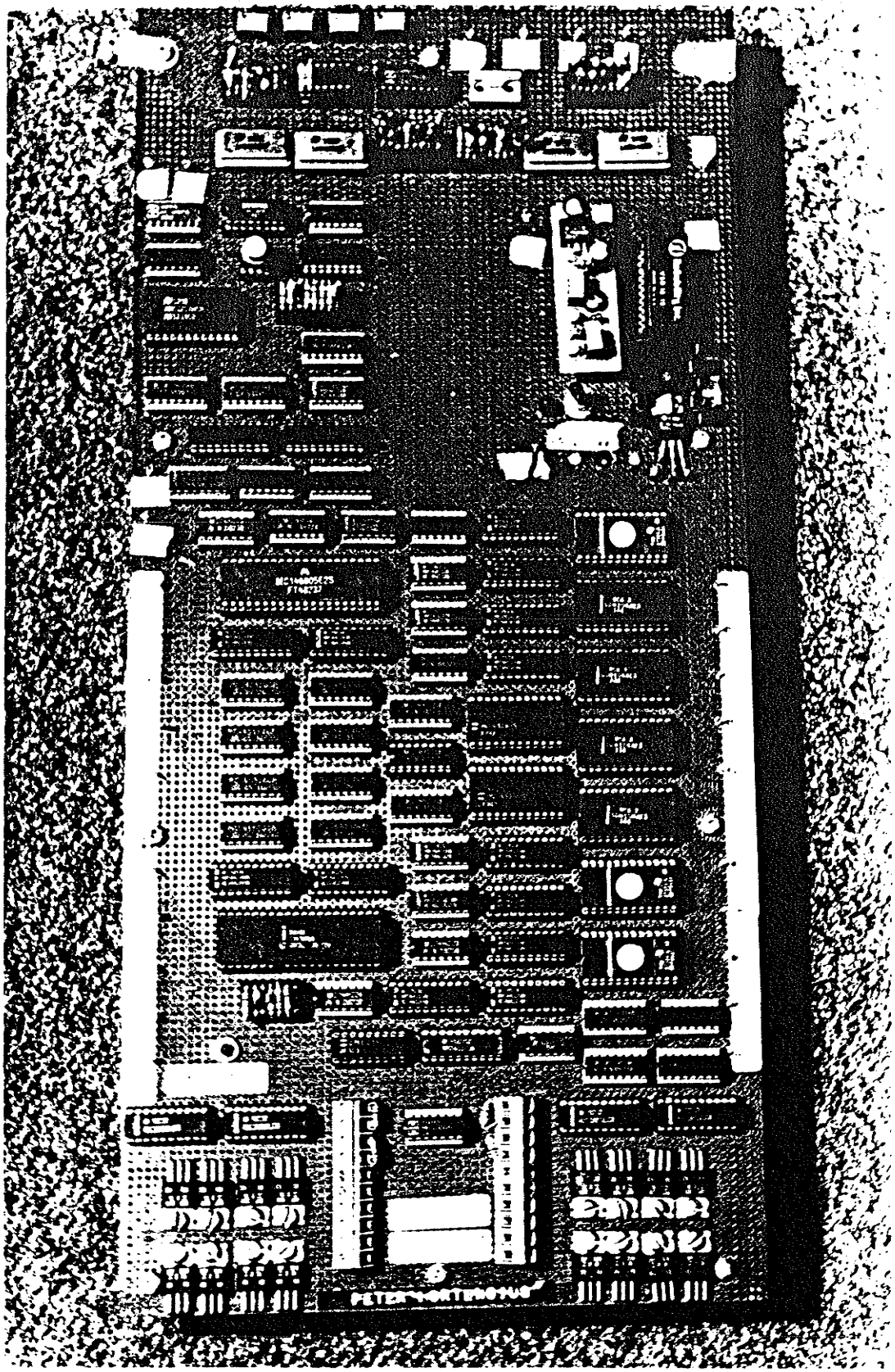


Figure 10 Top Side of Wire Wrapped Prototype 1.

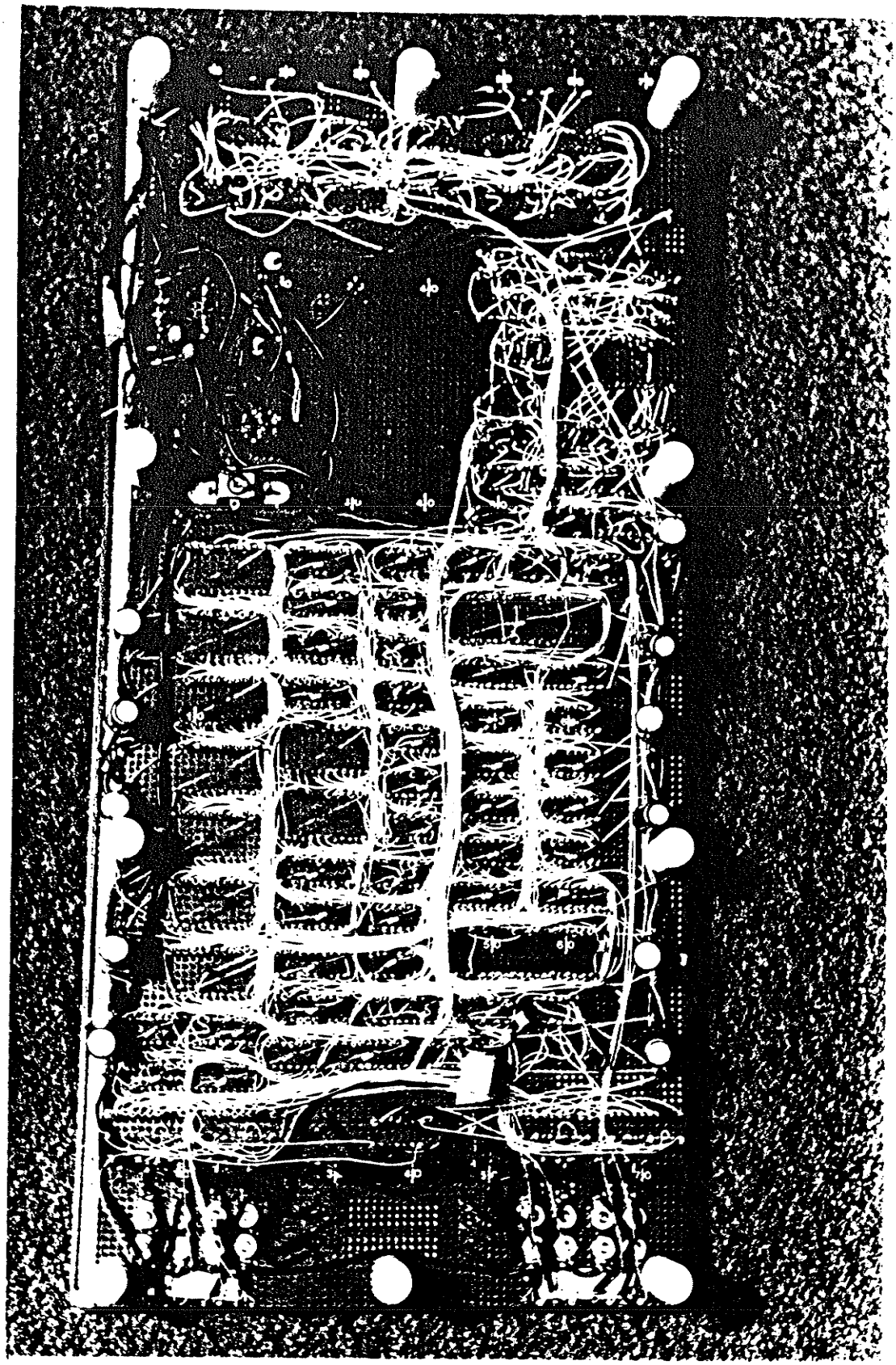


Figure 11 Bottom Side of Wire Wrapped Prototype 1.

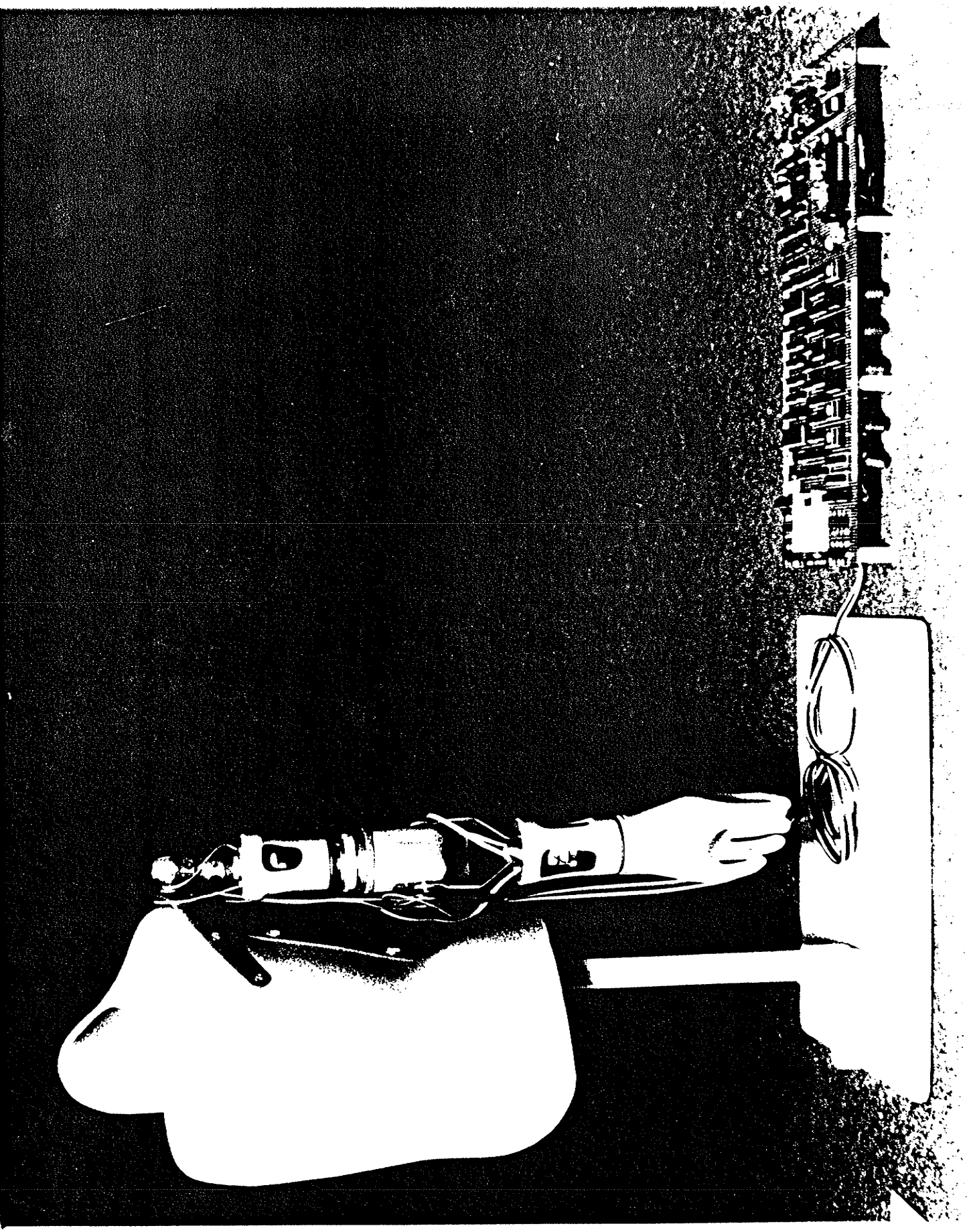


Figure 12 Prototype 1 with Prosthetic Limb.

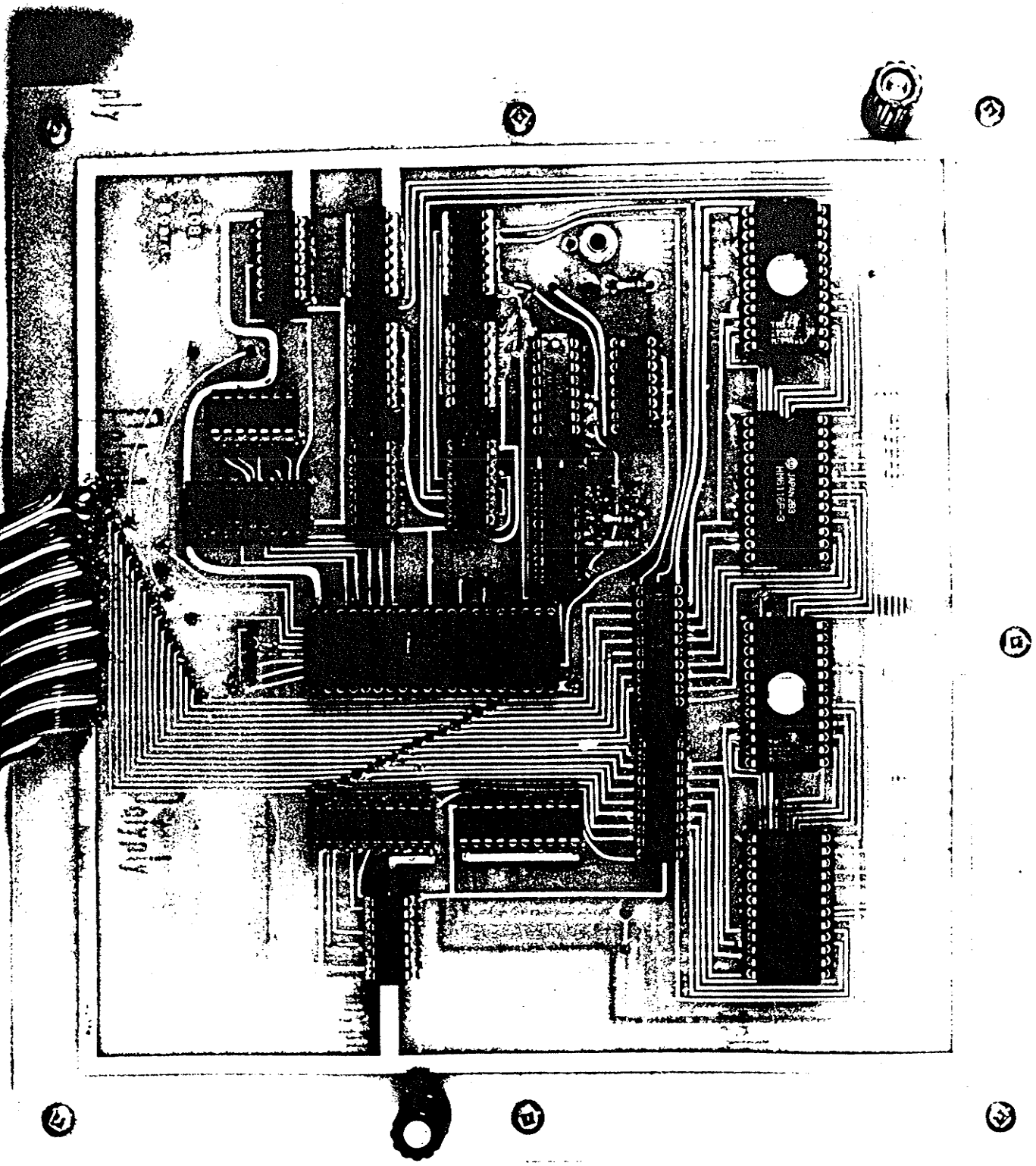


Figure 13 CD80C86 Microcomputer Board.

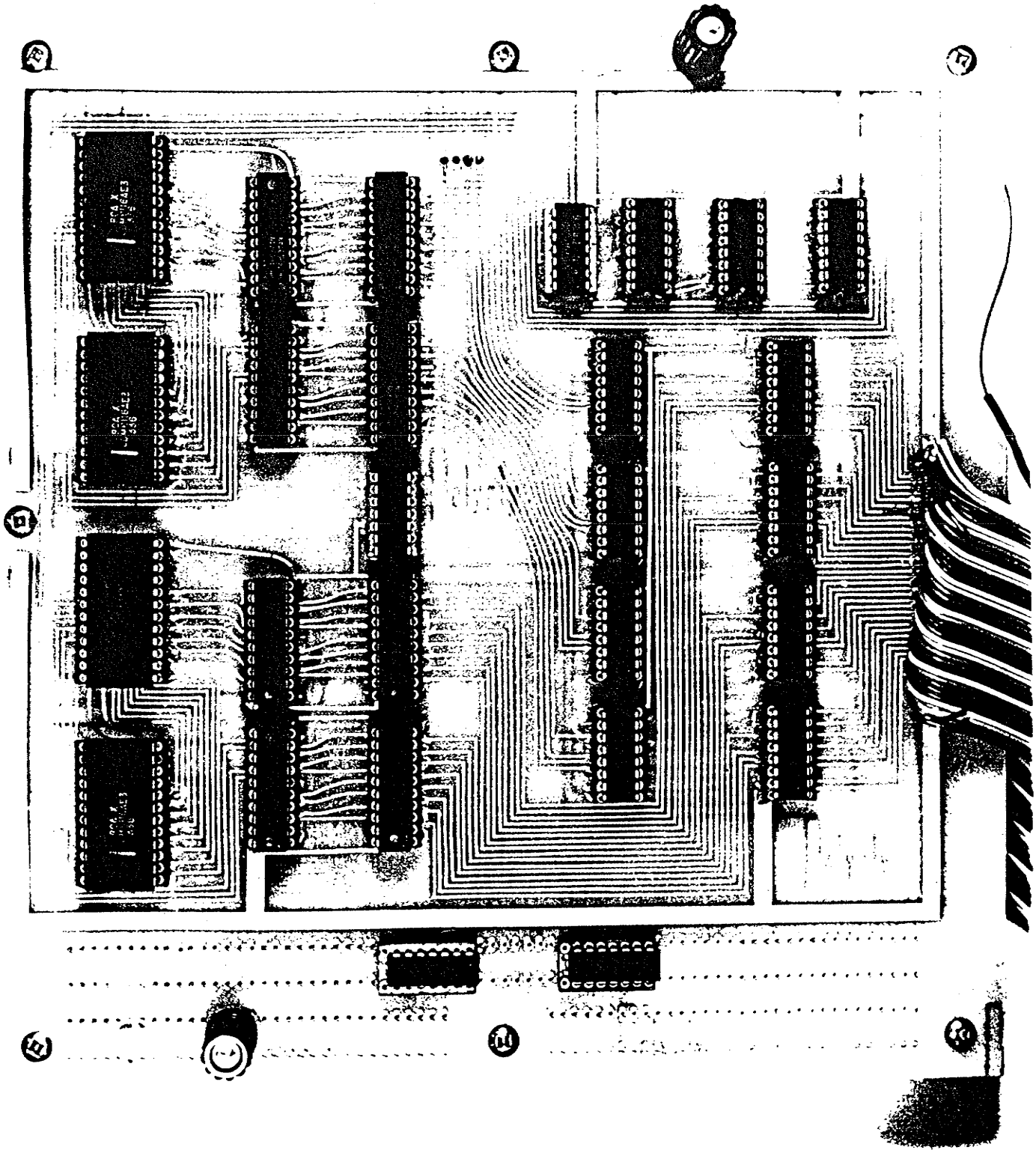


Figure 14 Shared Memory Board.

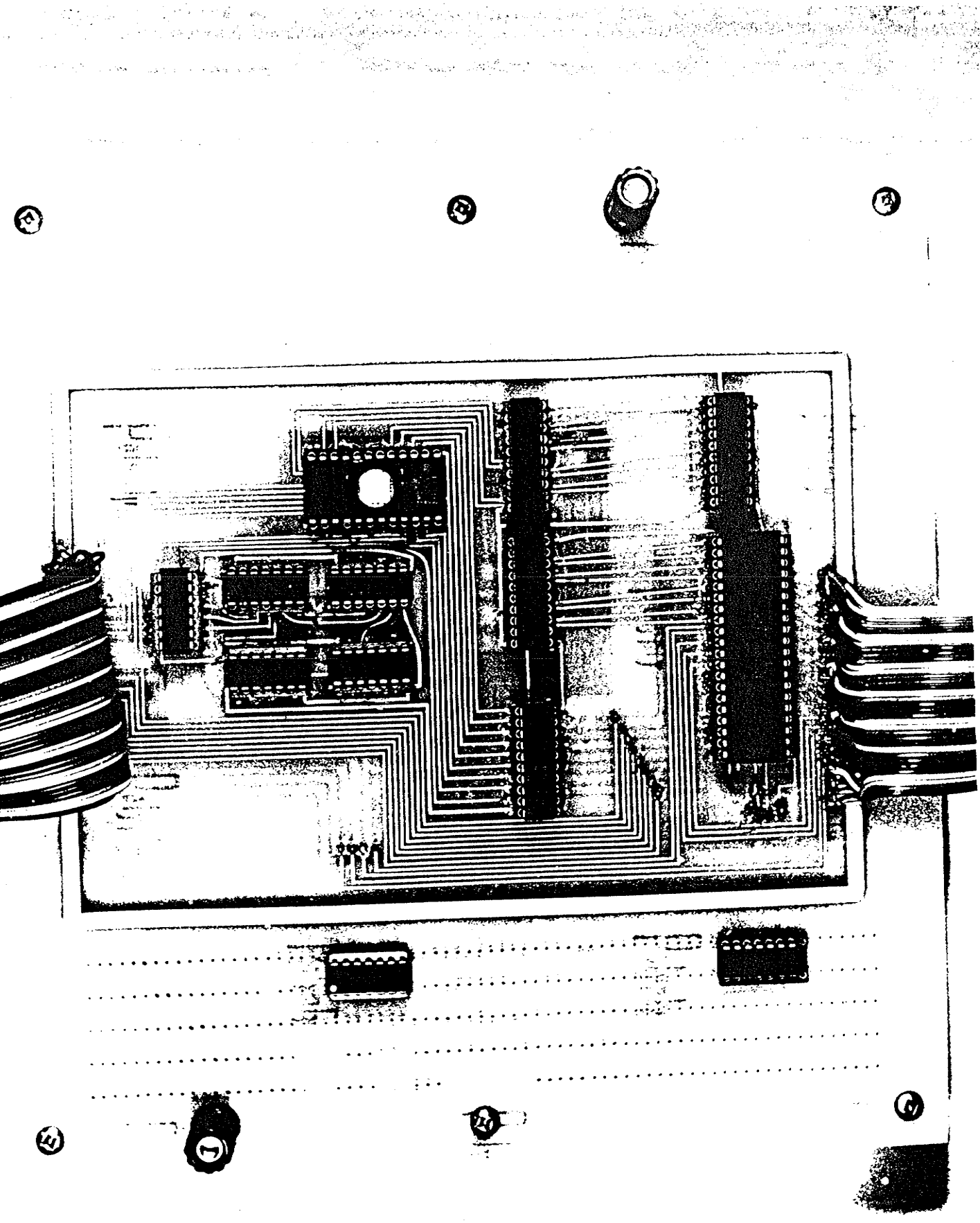


Figure 15 MC146805E2 Microcomputer Boord.

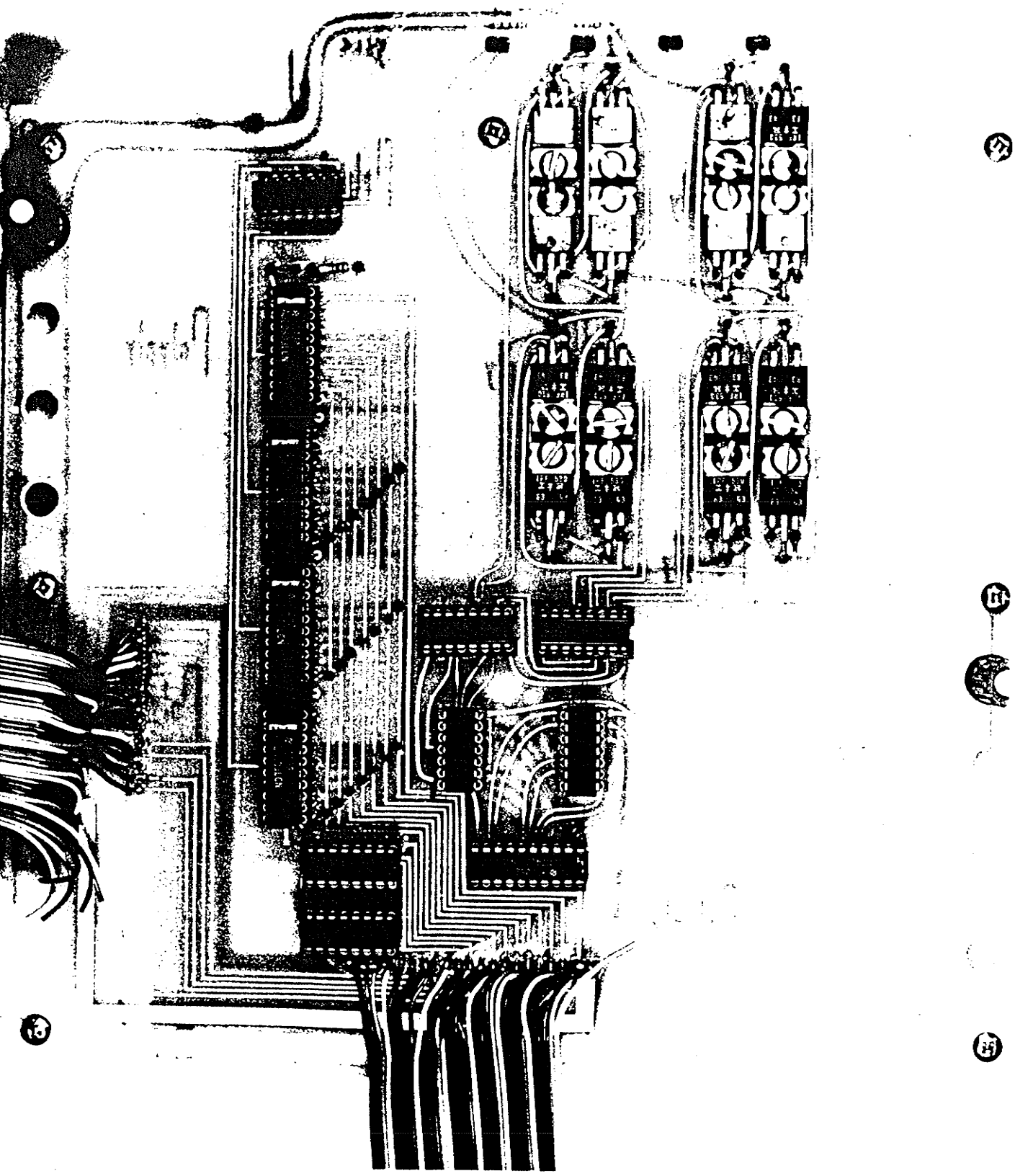


Figure 16 Motor Control and Positional Board.

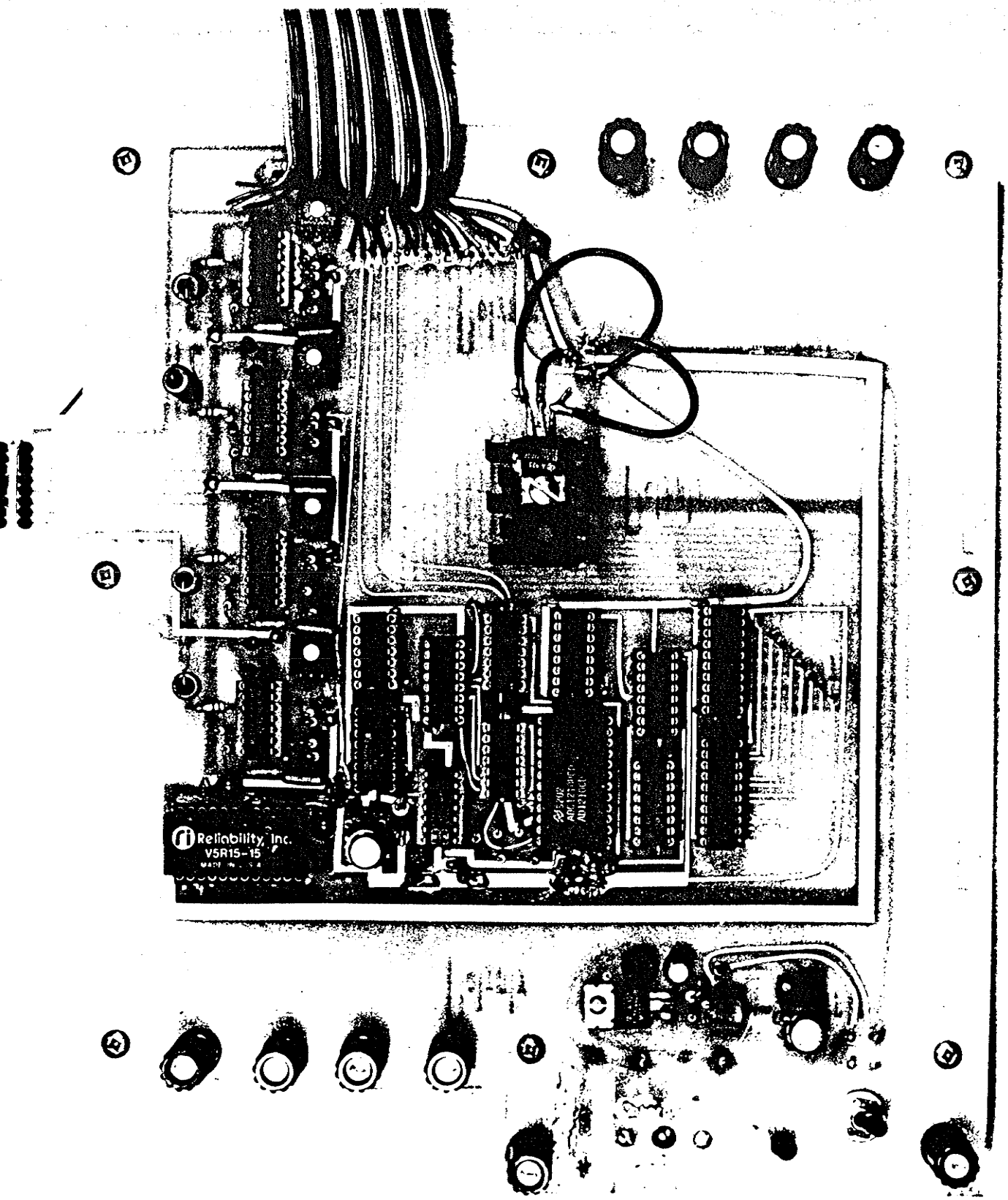


Figure 17 Data Acquisition and Power Supply Board.

vertical bus connects the motor control and position feedback board to the MC146805E2 computer board. This bus connects the data and control lines needed by the two boards to operate and also delivers the power lines to the rest of the limb controller. The last vertical bus is also the largest vertical bus. This bus connects the MC146805E2, shared memory, and CD80C86 boards. It contains both the MC146805E2 and the CD80C86 address/data buses as well as all the control signals that must be passed to the shared memory board from the two computer boards. In addition, power lines are passed by these vertical buses.

To aid in noise reduction the +5 volt and ground supply lines are passed to the circuit boards by using 14 gauge wire connected to terminal posts on each board. This provides solid voltage supply lines for each board and significantly reduces the noise present on the voltage supply lines.

CHAPTER 3

LIMB CONTROLLER SOFTWARE

The software of the limb controller is as important to its operation as the hardware. This is because the hardware cannot be fully utilized without good software to control its operation. It should be realized that the hardware was not designed and then the software conceived. Like most microcomputers and microcontrollers the hardware and software were developed in tandem. Before any change in limb controller hardware was made the effect on the operation of the software was determined and the final decision was based on the optimal system with both software and hardware taken into account.

As with the hardware design, before writing the system software some basic system requirements must be determined. These basic software requirements can be summarized as follows:

i) Simple to use - The software must be simple to operate because if the software makes the prosthesis difficult to operate the prosthesis will not be used.

ii) Flexible - The basic limb controller software must be flexible enough to do a variety of tasks with little or no modification.

iii) Powerful - The software must be able to implement a variety of control algorithms without needing extensive revisions.

The above requirements are all necessary to the limb controller software but some need to be more fully clarified before the software design is discussed. The words "simple to use" often give a general idea but not a specific idea of what is meant. For this project "simple to use" is defined as: the user needs to understand no more than to press the reset switch to start operation of the limb controller. The limb controller will either have no requirements on power up or will take the user through a calibration procedure by modelling the actions necessary on the prosthetic limb.

Flexible is meant in the sense that for a variety of signal processing algorithms to be implemented on the CD80C86 microprocessor the MC146805E2 microprocessor software should need no, or very little, modification. This requirement focuses on the MC146805E2 software more so than on the CD80C86 signal processing software.

The final basic requirement is that of powerful. This is also a relative term but specifically for this thesis it will be defined as how many different kinds of signal processing algorithms can be operated within the basic system software environment without needing to rewrite portions of the software. This requirement centers on both the framework around the signal processing software on the CD80C86 and the limb controller supervisory software on the MC146805E2.

The following discussion concerns the various programs that were written for the limb controller. Some of these programs were written for direct implementation on the limb controller while others were written to provide a framework for future software development. All of these programs are necessary to do present or future work with the limb controller. The programs are categorized by the microprocessor on which they operate.

3.1 Limb Controller Supervisory Software

The MC146805E2, as discussed previously, is used as the limb controller supervisor in addition to its duties for data acquisition and joint motor control. Its interface to the limb controller is defined in Chapter 2 and so in discussing the software the actual control line, or hardware circuit, that is used to effect an action will not be

mentioned. For example, when indicating that the limb controller supervisor is about to start an analog to digital conversion it will be stated that way rather than state that the MC146805E2 is about to start an analog to digital conversion by toggling input/output control line PB6 high then low. This should aid in easing the wordiness of the discussion.

The basic software requirements defined previously weigh most heavily on the software written for the limb controller supervisor. This is because of the basic nature of the tasks performed by the limb controller supervisory software. It could be thought of as being analogous to the basic input and output software (commonly known as the BIOS) of any microcomputer and is the most important piece of software on the microcomputer because no other programs could operate without it. Yet it is rarely, if ever, changed. The limb controller supervisory software must gather the EMG signal data for use by the signal processor, output the motor control, and control the operation of the limb controller. A general flowchart of the limb controller supervisory software is given in Figure 18.

Any discussion of software must start at the beginning and in this case the beginning will be defined as upon reset. Once the limb controller has been reset the entire limb controller configuration must be redefined. The

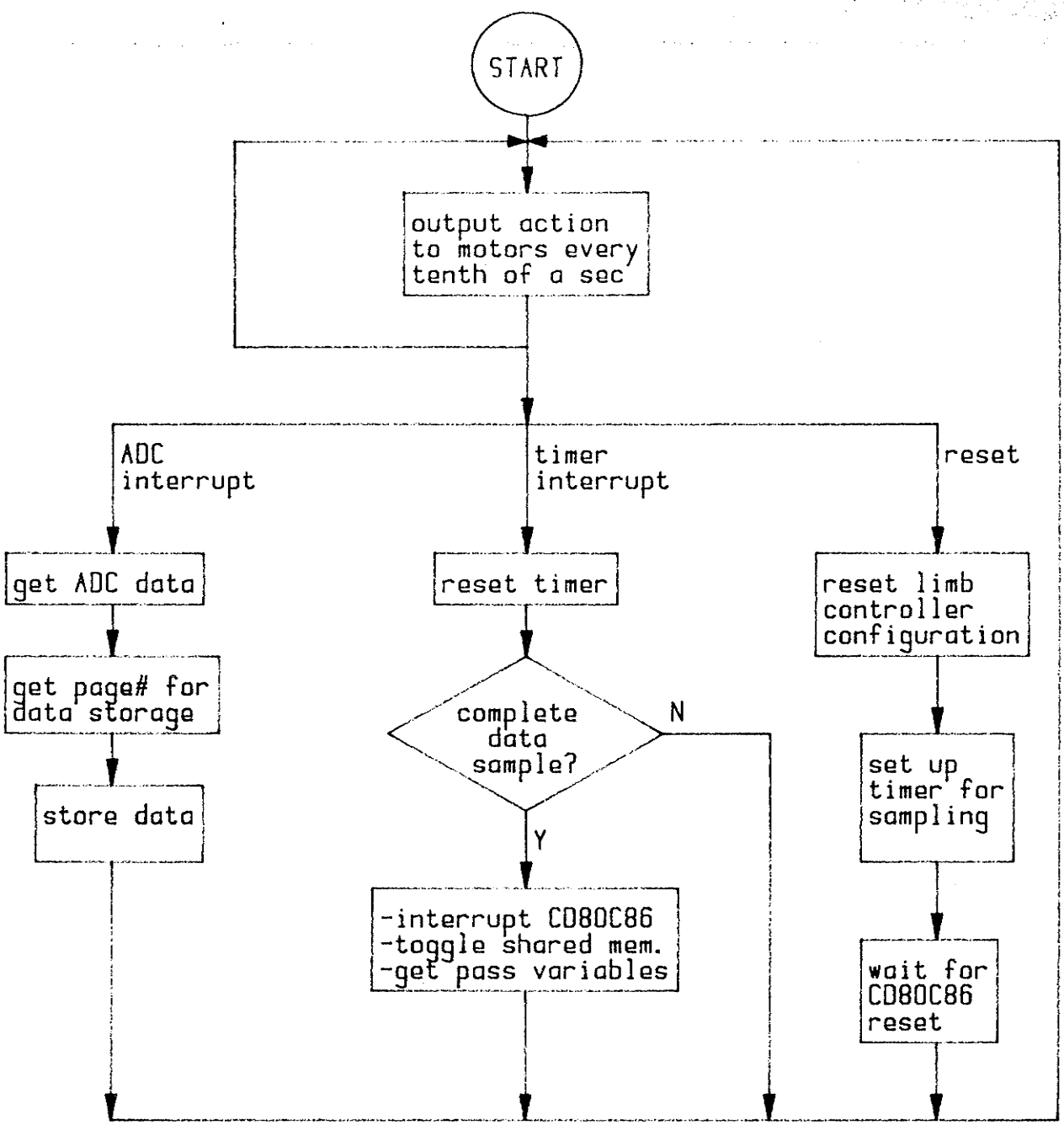


Figure 18 Flowchart of Limb Controller Supervisory Software.

input/output control lines must be redefined to output, shared memory be reset to shared memory 0, system variables reset to their starting values, and the system timer reset. All the obvious tasks that a reset routine must do. As well, the reset routine needs to ensure that no false analog to digital conversions will be started so one analog to digital conversion is done to clear any pending interrupts from the conversion process. In order to allow the CD80C86 to reset a wait loop of about 0.5 seconds is executed. The processor then moves on to the main routine of the limb controller supervisory software.

It was decided that updating the joint motor control every tenth of a second would be sufficient to achieve good joint motor control. This is because of the reaction time of the user is of course much slower than one tenth of a second and also because some possible preprogrammed motions may need this rapid updating time in order to achieve sufficient resolution in their movements. However, a technique to allow preprogrammed and/or multifunction movements and movements superimposed on other movements had to be developed. The solution was to define a set of pass motor control variables from the signal processor and a set of internal motor control variables. An illustration to show the use of the pass variables is given in Figure 19. The pass motor control variables will request a new action for a joint motor only if the most significant bit (i.e. bit 7) of the variable is

high, otherwise the control request will be ignored. The internal variables keep track of the joint motor direction and the time left for its operation. Bit 6 of the internal variables defines the joint motor direction with a low indicating a close motion and a high indicating an open motion. The remaining bits 0 to 5 are used to time the joint motor control action. The signal processor when requesting a joint motor control action sets bit 7 in the pass variable as well as bit 6 for direction and controls the duration of the action by using bits 0 to 5. Since each action is updated every tenth of a second this gives a minimum action duration of a tenth of a second and a maximum action duration of 6.4 seconds. Sufficient resolution for fine joint motor control will be available because a joint motor will only move a small distance in a tenth of a second, but in 6.4 seconds the full range of motion can be executed.

The strength of this technique for joint motor control becomes evident when it is realized that this means automatic shut off of a joint motion is accomplished saving on battery life. As well, once an action is started it cannot be stopped until it times out or another request for motion of that joint motor occurs. This means actions can be superimposed upon one another. For example, if the user requests 1.5 seconds of elbow flexion (approximately 90 degrees) and as the motion is executing it is also decided that the hand must be opened. The user requests this action

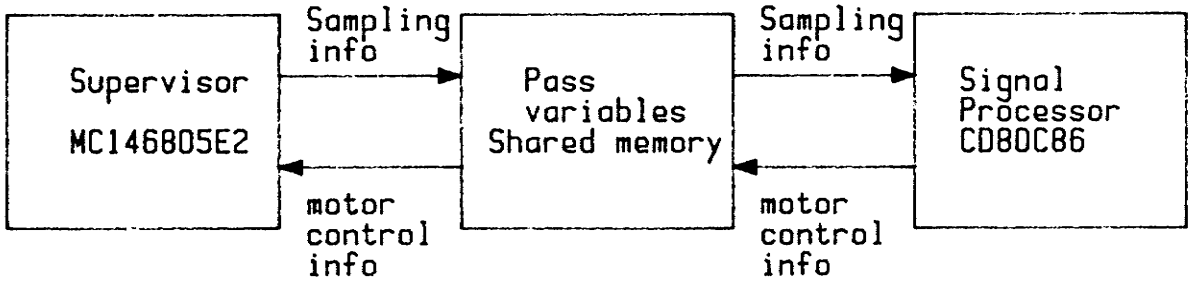


Figure 19 Operation of Pass Variables.

and the hand is opened as the elbow is flexing, thereby superimposing the hand motion onto the elbow movement.

For preprogrammed movements this technique is also easily applied. The preprogrammed pass variable from the signal processor must have bit 7 set to indicate a new preprogrammed motion request and then using bits 0 to 5 select the preprogrammed movement desired. This gives a range of 64 possible preprogrammed movements. The program upon detection of a preprogrammed movement request enters a routine which will select the preprogrammed movement. This is done by replacing the values in the pass variables from the signal processor with values that select the action desired. For example, if preprogrammed motion 9 corresponded to elbow flexion for 2.0 seconds, wrist supination for 1.6 seconds, and hand closing for 0.5 seconds, then upon a request for preprogrammed motion 9 the values corresponding to the above action would be placed into the pass variables for the elbow, wrist, and hand joint motors. These joint motor actions would then be carried out until they timed out or a new request for a joint motor action is given.

Multifunction movements are simply simultaneous implementations of superimposed actions. For example, if the user selects wrist pronation with hand opening, the two pass variables for wrist and hand are set to the corresponding values for that action. The two actions will carry on

individually until they time out or a new action for that joint motor is requested.

The timer of the MC146805E2 is used to interrupt the limb controller supervisory software whenever a new sample of EMG data is to be taken. The timer value can be set to any value desired based on an input frequency of either the MC146805E2 clock or the analog to digital converter clock. For example, if a sampling rate of 1 kHz were desired this could be done by selecting 130 kHz clock of the analog to digital converter, setting the prescaler to 32 and setting the timer to the value 4. The timer interrupt would then occur every 985 usecs or about 1.016 kHz. The timer routine first resets the timer to the starting value and restarts it. Next, the timer interrupt routine resets the channel count to start at the first channel and initiates the conversion immediately. This produces one limitation in that channel 0 is always converted first but this should not cause any problems and if it does the change is very simple to make. The timer routine also checks if the required number of data samples has been taken, and if so, resets the data count and data storage variables and executes the signal processor interrupt procedure. If there are still more data samples to be taken the routine returns to the main program.

The signal processor interrupt procedure must load the pass variables to the signal processor with the required values. For example, in three state EMG control the two power ratios and the number of data samples are passed to the signal processor. The shared memory is then toggled and the nonmaskable interrupt of the CD80C86 asserted, thereby interrupting the signal processor. The final action is to load the pass variables from the signal processor into a temporary set of local variables. At this location the meaning of the pass variables from the signal processor can be redefined if necessary by either modifying the pass variable value or storing it in another location. The routine then returns execution to the main program.

The MC146805E2 is interrupted whenever an analog to digital conversion has been completed. This invokes the limb controller supervisor interrupt routine, which first checks if there are more channels to be converted and if so, selects the next channel and starts the conversion. Next, the routine reads the high and low data bytes of the analog to digital converter and stores them in temporary variables. The MC146805E2 does not provide a convenient way to store a large amount of incoming information because it assumes a small data memory requirement. This problem can be overcome by specialized subroutines corresponding to each page in memory. Therefore, once the data bytes have been read the subroutine corresponding to the current page of data storage

is invoked and the data stored. After execution the interrupt routine returns to the main program. It should be noted that an analog to digital conversion takes approximately 100 usecs or about 212 machine cycles. Therefore, the interrupt routine should execute in that time period or the analog to digital data may be overwritten. At present, the interrupt routine takes about 137 machine cycles or about 64 usecs, well within the necessary time limit.

If the reader desires more information about the operation of the limb controller supervisory software please consult the limb controller software manual given in the Bibliography..

3.2 Signal Processing Software

The basic software requirements mentioned at the start of this chapter do not apply to the signal processor software as strictly as to the limb controller supervisory software. This is because each new unique signal processing and decision algorithm will need to be implemented in a different way. The limb controller supervisory software provides all the utilities that the signal processor needs to invoke action requests of the joint motors without needing to be actually interfaced to the joint motors. Therefore, all that the signal processing software needs to

do is determine actions and output the results to the pass variables.

The programs for the signal processor were all developed on an IBM Personal Computer or related compatible. This microcomputer uses an Intel 8088 as the microprocessor and uses dynamic RAM for memory. However, this does not cause any compatibility problems between the limb controller and the IBM PC because the 8088 is simply an eight bit data bus version of the 8086. This means that the instructions are the same. It merely takes longer to read from and write to memory using an 8088 instead of an 8086 because 8 bits instead of 16 bits of data are used at a time. The use of dynamic RAM necessitates a refresh cycle every 2 msecs, thereby causing a delay in processing every 2 msecs. This is invisible to the programmer and program so it also does not present any visible incompatibilities with the limb controller CD80C86 setup. In fact the only difference will be the speed of execution. The clock rate of the IBM PC/XT is 4.77 MHz; the same as the limb controller. Therefore, since the CD80C86 uses a 16 bit data bus and the limb controller uses static RAM memory requiring no refresh cycle, programs will execute faster on the limb controller CD80C86 than on the IBM PC/XT 8088. It would be difficult to say how much faster, but they will definitely execute faster. This means that if the program execution time on the IBM PC/XT during development is fast enough, then it will be

even faster on the limb controller. Therefore, the IBM PC/XT gives what could be compared to a worst case execution time for any signal processing software developed on it. The IBM PC/XT used for development also has a multichannel analog to digital converter as a peripheral device, thereby permitting full testing of the signal processing software before being used in the limb controller.

The limb controller signal processing software is contained within a shell program that is used only on limb controller reset. The program is called FRAME. The purpose of the FRAME program is to provide a standard reset sequence for use by all signal processing programs. A general flowchart of the FRAME program is given in Figure 20. Its execution is very simple. On limb controller reset the signal processor must assume that it must reconstruct its processing environment by rebuilding the interrupt table located in the four lowest pages of memory. The FRAME program sets any flags that need to be set as an indicator of limb controller reset and then begins to rebuild the interrupt table. There are 256 interrupt vectors in the interrupt vector table with only a few being predefined. The predefined interrupts consist of a divide by 0 error interrupt for use with the DIV and IDIV instructions, a nonmaskable interrupt vector, an overflow interrupt for use with the INTO instruction, and three other vectors that are used with debugging programs. The rest of the interrupts are

accessed with either the INT instruction for software interrupts or are used on maskable processor interrupts. Since there are no maskable interrupts in this system and no debugging program is in operation, all the vectors are defined to a return from interrupt instruction except the divide by zero, the overflow, and nonmaskable interrupts. The divide by 0 and the overflow interrupts are vectored to a routine that will process the error condition. The nonmaskable interrupt is used to start the signal processing algorithm and so its vector points to the signal processing software. After the interrupt vector table has been loaded the signal processor waits for a nonmaskable interrupt to occur and then begins execution of the signal processing software. After completion of the signal processing program the execution is passed back to wait for another nonmaskable interrupt to occur.

3.2.1 EMG Signal Power Processing

The EMG signal may be modelled as a zero mean Gaussian process with a controllable variance [42]. Therefore, most myoelectric limb controllers use the variance as the parameter to be used for control. It should be noted that the variance of the EMG signal is also the power of the EMG signal. For this limb controller the variance is calculated rather than derived by filters as in present day analog limb controllers.

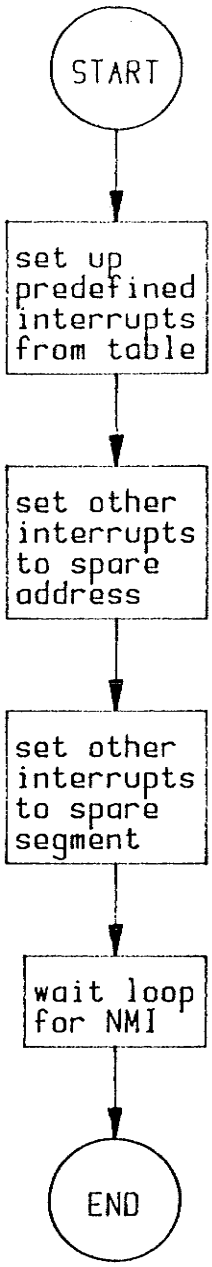


Figure 20 Flowchart of FRAME Program.

The mean of the measured EMG signal should be zero except that some offset may be introduced by the measurement system. The mean of the measured EMG signal may be calculated as:

$$u = \frac{1}{N} \sum_{i=1}^{N} s_i \qquad [1]$$

where
- u - mean of EMG signal
- s_i - EMG signal at time i
- N - number of samples taken

The mean may then be used to calculate the variance as follows:

$$\sigma^2 = \sum_{i=1}^{N} \frac{(s_i - u)^2}{N-1} \qquad [2]$$

where
- σ^2 - variance of EMG signal
- u - mean of EMG signal
- s_i - EMG signal at time i
- N - number of samples taken

This value may then be used to control the prosthetic limb. For example, if the power increases over a certain threshold then the limb is activated; if it has not reached the threshold value then no action is taken. This models two state EMG control. For three state control two thresholds are used to give two stages of activation for each EMG signal. This extends over any number of control states, but as the number of control states is increased the difficulty of prosthetic limb control is increased. Present clinical

systems use a maximum of three state control.

Two signal power processing programs were written for the limb controller signal processor. These were for two and three state control. The two programs operate in a similar manner, but the exact method of implementation is different in order to test various aspects of the limb controller. These differences are not important to this discussion and can be found in the limb controller software manual given in the Bibliography..

The first signal power processing program that was written was the two state control program called STATE2 and this was followed by the three state control program called STATE3. For this discussion only the STATE3 program will be discussed because the two programs are very similar. A general flowchart of the STATE3 program is given in Figure 21. The first task that the STATE3 program must perform is to clear its own internal joint variables and get the pass variable parameters passed to it from the limb controller supervisory program that runs on the MC146805E2 (remember that the STATE3 program does not begin execution until the signal processor is interrupted by the limb controller supervisory program). The parameters passed to the signal processor are the two signal power threshold values, the number of channels of EMG signal used, and the number of data items in the sample. After setting the pass variables

to the limb controller supervisory program to the default values the STATE3 program calculates the sample mean using the method of equation [1]. Next the variance is calculated as described using equation [2]. The variance is then stored using the channel number as an offset into an array. For normal operation this is done into a general array, but for the first few passes through this routine the variance is stored in a comparison array. This is needed to give a base value for relaxed EMG signal power corresponding to each EMG signal channel. The program then checks if more channel variances are to be calculated and if so, calculates them.

The STATE3 program must then determine if an action has been requested by the operator. This is done by comparing the base variances determined in the first few passes of the program to the variances just calculated. If the ratio of a channel variance is less than the first power ratio value then no action is taken. If the ratio falls between the two power ratio values then a close action is taken and if the ratio is greater than the second power ratio then an open action is taken. The results corresponding to each channel are then stored in the return pass variables. Once the program has outputted all the actions for the number of EMG signal channels used then the signal processing returns to the FRAME program and waits for the next interrupt to restart the STATE3 program.

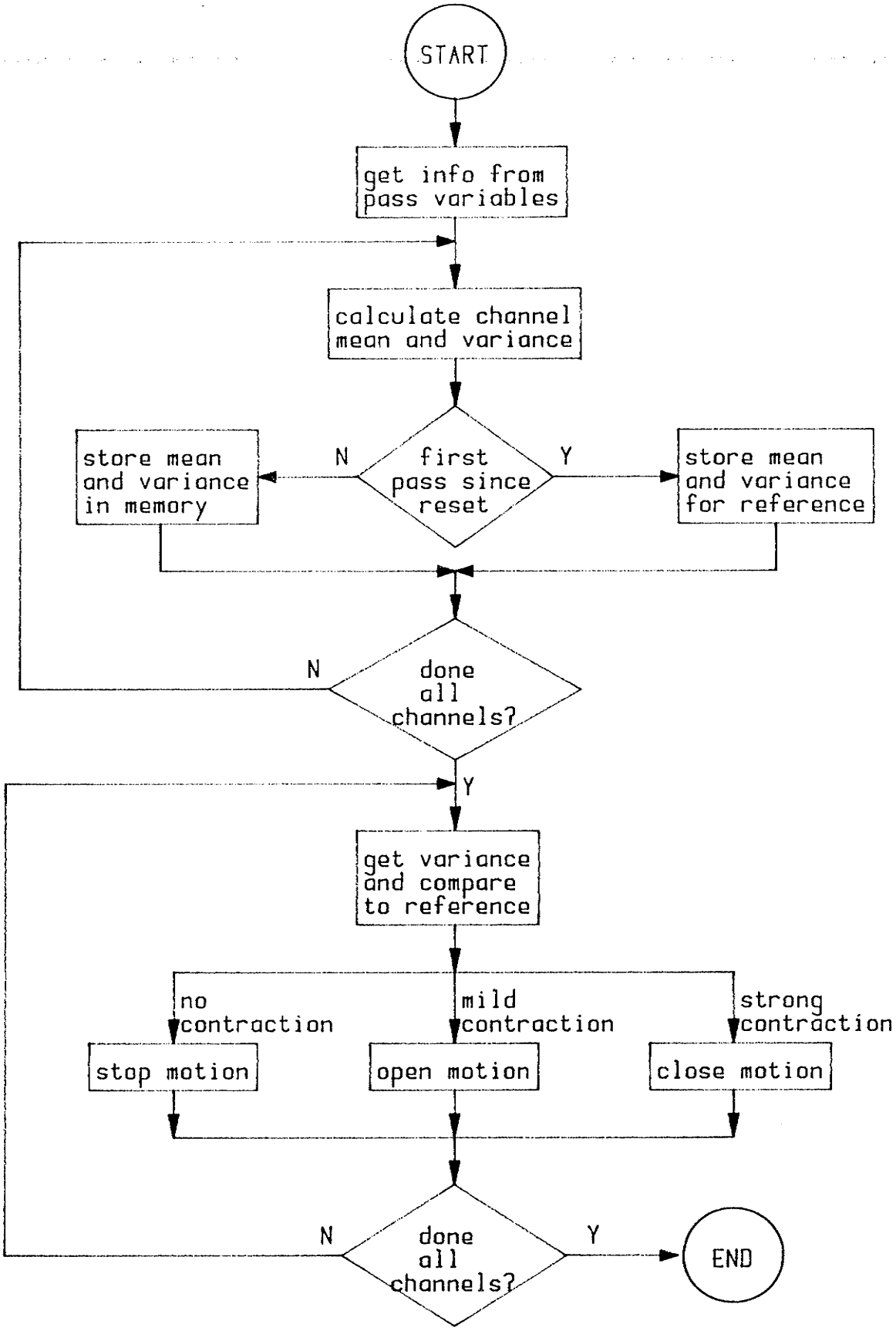


Figure 21 Flowchart of STATE3 Program.

3.2.2 EMG Autoregressive Model

The final program to be implemented on the limb controller signal processor is an autoregressive model of the EMG signal. The use of an autoregressive model to identify unique temporal parameters of the EMG signal was first proposed by Graupe in 1973 [43], [44], and [45]. The control algorithm was implemented on a microprocessor based limb controller dedicated to this control algorithm. An 8080 was used for this work. While the 8080 was a powerful microprocessor in 1973, today it is considered a low performance microprocessor and is rarely used in newly designed systems. This device is also not a CMOS device so its final implementation into a limb controller would create power supply problems. Graupe found that the response time of the limb controller was much too slow (around 2.5 seconds) because the 8080 could not calculate the autoregressive coefficients quickly enough. He is presently implementing the control algorithm on a 68000 based microcomputer and estimates the response time to be reduced to an acceptable 200 to 300 msecs.

The autoregressive model coefficient determination used is the method of least squares. This method is similar to the one used by Graupe [46]. The autoregressive model of a signal is based on the following equations taken from Makhoul [47] where the signal at time n is estimated as a

linear combination of past values:

$$-\tilde{s}_n = -\sum_{k=1}^{p} a_k s_n \qquad [3]$$

where
- $\tilde{s}_n$ - estimated EMG signal at time n
- n - discrete time
- p - model order
- k - coefficient number
- a_k - autoregressive coefficients

The error between the actual value of the signal and the predicted value is given by:

$$e_n^2 = s_n - \tilde{s}_n = s_n + \sum_{k=1}^{p} a_k s_n \qquad [4]$$

where
- s_n - value of the EMG signal at time n
- e_n^2 - error at time n

Obviously, in order to make the model as accurate as possible the error must be minimized. For a random signal, such as the measured EMG signal the total mean squared error can be stated as:

$$E = \mathcal{E}\, e_n^2 = \mathcal{E}\, \{ (s_n + \sum_{k=1}^{p} a_k s_n)^2 \} \qquad [5]$$

where
- E - total mean squared error

It is clear that E can be minimized by:

$$\frac{dE}{da_i} = 0 \qquad 1 \leq i \leq p \qquad [6]$$

Partial differentiating equation [5] with respect to a , the following is derived:

$$\sum_{k=1}^{p} a_k \mathcal{E} \{ s_{n-k} \; s_{n-i} \} = -\mathcal{E} (s_n \, s_{n-i}) \qquad 1 \leq i \leq p \qquad [7]$$

Therefore, we now have a set of p equations with p unknowns which, when solved, will give a minimum value for the total squared error E as defined in equation [5]. The error for each coefficient value can then be stated by expanding equation [5] and substituting in equation [7]:

$$E = \mathcal{E} (s_n^2) + \sum_{k=1}^{p} a_k \mathcal{E} (s_n \, s_{n-i}) \qquad [8]$$

The EMG signal can be considered a stationary random signal as long as its mean and variance are not changing with time. For the EMG signal it has already been stated that it is a zero mean Gaussian process. However, for the limb controller a slight offset is introduced by the measurement system. This offset is constant and therefore the mean does not change over time. Since the variance of the EMG signal can be controlled it can be stated that for a given contraction the variance will not be changing over time. This, of course, assumes a constant contraction where

the limb is not moving, a situation which is especially true for the amputee where the muscle insertions are missing for the muscle being measured and so no limb motion can occur. In addition, present limb controllers, especially those using three state control, use a constant EMG signal power or variance for control. Thus, for further discussion the EMG signal will be considered as a stationary signal.

For a stationary random process the autocorrelation of that process is equal to:

$$R(i-k) = \mathbf{E}\ (s_{n-k}\, s_{n-i}) \qquad [9]$$

where R(i) - autocorrelation of the process

Using equation [9], equations [7] and [8] can be reduced to:

$$\sum_{k=1}^{p} a_k R(i-k) = -R(i) \qquad 1 \le i \le p \qquad [10]$$

$$E = R(0) + \sum_{k=1}^{p} a_k R(k) \qquad [11]$$

It should also be noted that the autocorrelation function is also an even function i.e.

$$R(-i) = R(i) \qquad [12]$$

Expanding equation [10] we obtain its matrix form:

$$\begin{bmatrix} R_0 & R_1 & R_2 & \cdots & R_{p-1} \\ R_1 & R_0 & R_1 & \cdots & R_{p-2} \\ R_2 & R_1 & R_0 & \cdots & R_{p-3} \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ R_{p-1} & R_{p-2} & R_{p-3} & \cdots & R_0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \cdot \\ \cdot \\ \cdot \\ a_p \end{bmatrix} = - \begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ \cdot \\ \cdot \\ \cdot \\ R_p \end{bmatrix} \qquad [13]$$

Equation [13] can be solved in a variety of ways such as the Gauss reduction method or Cholesky decomposition. However, these methods are general and require a great deal of computations. If we examine the autocorrelation matrix it can be seen that it is symmetric and the elements along the diagonal are identical. In addition, the column vector on the right hand side of [13] uses the same elements as the autocorrelation matrix. Using these facts Durbin [48] derived a quick and efficient algorithm to solve for the autoregression coefficients. This method requires only 2p storage locations and $p^2 + O(p)$ operations. This is a big saving over the Gauss reduction method which requires p^2 storage locations and $p^3/3 + O(p^2)$ operations and the Cholesky method which requires $p^2/2$ storage locations and $p^3/6 + O(p^2)$ operations. Durbin's recursive method can be specified as:

$$E_0 = R_0 \qquad [14a]$$

$$k_i = -\left[R(i) + \sum_{j=1}^{i-1} (a_k * R(i-j)) \right] / E_i \qquad [14b]$$

$$a_i^{(i)} = k_i \qquad [14c]$$

$$a_j^{(i)} = a_j^{(i-1)} + k_i a_j^{(i-1)} \qquad 1 \le j \le i-1 \qquad [14d]$$

$$E_i = (1-k_i^2) * E_{i-1} \qquad [14e]$$

Equations [14b] to [14e] are solved recursively for i=1,2, ...,p and the final solution is given by:

$$a_j = a_j^{(i)} \qquad 1 \le j \le p \qquad [15]$$

The major computational load in the calculation of the autoregressive coefficients is the calculation of the autocorrelation coefficients provided N>>p which is usually the case. Therefore, any method that may reduce the number of computations, especially those of multiplication, required to calculate the autocorrelation coefficients would speed up the processing. One of the more common methods has been attributed to Kendall. The results of his work show that if we assume N (the number of samples) even, then the expression for autocorrelation can be expressed as:

$$R(k) = \sum_{m=0}^{(N-k)/2-1} [s(2m)+s(2m+k+1)] * [s(2m+1)+s(2m+k)] - A(k) - B(k) \qquad k \text{ even} \quad [16]$$

$$R(k) = \sum_{m=0}^{(N-k-1)/2-1} [s(2m)+s(2m+k+1)] * [s(2m+1)+s(2m+k)] - A(k) - B(k) + s(N-1-k)*s(N-1) \qquad k \text{ odd} \quad [17]$$

where
- k - coefficient number
- s - EMG signal
- A(k) - correction factor
- B(k) - correction factor
- R(k) - autocorrelation coefficient
- N - number of data samples

A(k) and B(k) can be obtained by the recursive relations:

$$A(k) = A(k+2) + s(N-2-k) * s(N-1-k) \qquad k \text{ even} \quad [18]$$

with initial condition: A(N)=0.

$$A(k) = A(k+1) \qquad k \text{ odd} \quad [19]$$

with initial condition: A(N-1)=0.

$$B(k) = B(k+2) + s(k) * s(k+1) \qquad k \text{ even} \quad [20]$$

with initial condition: B(N)=0.

$$B(k) = B(k+2) + s(k) * s(k+1) \qquad k \text{ odd} \quad [21]$$

with initial condition: B(N-1)=0.

Using equations [16] and [17] it can be shown that the number of multiplications required to compute R(k) is

approximately (N-k-1)/2, about half the number normally required by direct evaluation. This decrease in multiplications is done at the expense of about 50% more additions, but since addition is a much quicker operation than multiplication the overall result is significantly faster computation of the autocorrelation coefficients.

While the claims of Graupe to have achieved reliable control of prosthetic limbs with up to 6 degrees of freedom have been verified by Doerschuk et al [49], it was decided to do a preliminary investigation of the possibilities of using an autoregressive model for this limb controller. If this method would be reliable it would be an ideal demonstration of the abilities of this limb controller to advance the state of the art in clinical myoelectric limb controllers.

A PDP 11 minicomputer with an analog data acquisition subsystem was used to gather EMG signal data from 7 test subjects. These subjects were all physically normal young persons in the age range from 19 to 26 years of age. Six males and one female were used. The age and sex should create insignificant changes in results, but the use of physically normal people was selected because of ease in finding test subjects and because this would be a comparison of fairly similar subjects. Data were taken for six movements and a relaxation calibration trial from an

electrode situated at approximately the insertion of the deltoid muscle on the lateral side of the left arm. The six actions were divided into two sets of three similar actions. Low power and high power contractions for elbow extension, elbow flexion, and wrist supination were measured and the EMG signal data gathered. The subject's arm was fixed in place so as not to allow any movement and thereby introduce nonstationary properties into the measured EMG signal. The data were filtered to give a bandpass of 10 to 1000 Hz and was sampled at 2500 Hz for 820 msecs. The subjects were tested five times each with a rest period of 5 minutes in between tests to allow testing for repeatability. It should be realized that this was not the most stringent of test conditions but was adequate for preliminary investigation. After gathering of the EMG signal data, the data were transferred to the university mainframe computer for analysis. The program called ARTEST (found in the limb controller software development manual given in the Bibliography) was used to analyze the data and output the coefficients. Tests were done to simulate various sampling rates and sampling periods. The results varied widely between subjects, but for each subject the results were consistent with the exception of the first subject used in the test procedure. The discrepancy could be explained by the fact that this was the first test subject used so that, in fact, the testing procedure was not as controlled as for the later subjects. The results for one of the subjects is

given in Tables 2 to 6.

Graupe found that when using four coefficients the best muscle discriminations patterns were found for a sampling frequency of 500 Hz and a sampling duration of about 200 msecs [50]. From the data given in the tables it can be seen that for this sampling frequency and duration good discrimination between actions can be obtained. For example, if the results of 500 Hz for 204 msecs (table 6c) are examined, the low power actions can be distinguished from the high power actions by the lower variance value. Action EE-LP is uniquely defined by much more negative a1 and a3 coefficients; action EE-HP is uniquely defined by a moderately negative a1 coefficient and a high positive a4 coefficient; action WS-LP is uniquely defined by a positive a1 coefficient and a high positive a4 coefficient. For high power actions the coefficients used to distinguish actions are different, but the method of analysis is the same. It is interesting to note that for all sampling rates and durations, except for those of 819 msecs, good discrimination between actions could be obtained; unlike Graupe's results where he found 500 Hz and approximately 200 msecs to give best discrimination. Additional tests were performed to check on the effects of using different numbers of autoregressive coefficients. It was found that reducing the number of coefficients sometimes would not yield sufficient information on which to uniquely define an

EE-Elbow Extension EF- Elbow Flexion WS- Wrist Supination

Action:	EE-LP	EF-LP	WS-LP	EE-HP	EF-HP	WS-HP
a1	-0.390	-0.225	-0.127	-0.524	-0.391	-0.260
a2	-0.126	-0.118	-0.032	+0.362	+0.111	+0.221
a3	-0.050	-0.068	+0.002	-0.080	+0.072	+0.028
a4	-0.055	-0.038	+0.020	+0.066	+0.047	+0.094
var	+0.020	+0.019	+0.026	+0.161	+0.046	+0.059

Table 2a Data sampled at 2500 Hz for 819 msecs.

Action:	EE-LP	EF-LP	WS-LP	EE-HP	EF-HP	WS-HP
a1	-0.404	-0.283	-0.225	-0.528	-0.338	-0.265
a2	-0.177	-0.099	-0.039	+0.349	+0.123	+0.179
a3	-0.077	-0.056	-0.019	-0.089	+0.066	+0.039
a4	-0.067	-0.057	+0.057	+0.054	-0.004	+0.067
var	+0.025	+0.027	+0.014	+0.196	+0.051	+0.047

Table 2b Data sampled at 2500 Hz for 409 msecs

Action:	EE-LP	EF-LP	WS-LP	EE-HP	EF-HP	WS-HP
a1	-0.535	-0.260	-0.221	-0.532	-0.246	-0.250
a2	-0.192	-0.106	-0.047	+0.374	+0.100	+0.214
a3	+0.001	-0.067	+0.021	-0.086	+0.112	+0.044
a4	-0.080	-0.061	+0.028	+0.055	-0.041	+0.067
var	+0.038	+0.038	+0.014	+0.181	+0.054	+0.060

Table 2c Data sampled at 2500 Hz for 204 msecs.

Action:	EE-LP	EF-LP	WS-LP	EE-HP	EF-HP	WS-HP
a1	-0.661	-0.458	-0.290	-0.507	-0.121	-0.303
a2	-0.159	-0.082	-0.078	+0.332	+0.053	+0.177
a3	+0.014	-0.044	+0.089	-0.154	+0.114	+0.048
a4	-0.042	-0.025	+0.051	+0.079	-0.068	+0.025
var	+0.063	+0.047	+0.014	+0.173	+0.058	+0.065

Table 2d Data sampled at 2500 Hz for 102 msecs.

Table 2 Results for data sampled at 2500 Hz with various sampling durations.

EE-Elbow Extension EF- Elbow Flexion WS- Wrist Supination

Action:	EE-LP	EF-LP	WS-LP	EE-HP	EF-HP	WS-HP
a1	-0.317	-0.168	-0.033	+0.136	-0.009	+0.203
a2	-0.110	-0.106	+0.008	+0.088	+0.070	+0.129
a3	-0.008	+00.023	-0.064	+0.078	+0.081	+0.055
a4	+0.010	+0.021	-0.079	+0.088	+0.013	-0.010
var	+0.019	+0.020	+0.026	+0.171	+0.045	+0.061

Table 3a Data sampled at 1250 Hz for 819 msecs.

Action:	EE-LP	EF-LP	WS-LP	EE-HP	EF-HP	WS-HP
a1	-0.335	-0.148	-0.073	+0.116	+0.046	+0.183
a2	-0.148	-0.120	-0.004	+0.060	-0.006	+0.077
a3	-0.050	+0.022	-0.037	+0.092	+0.052	-0.004
a4	-0.004	-0.015	-0.214	+0.110	+0.064	-0.047
var	+0.026	+0.028	+0.014	+0.206	+0.052	+0.048

Table 3b Data sampled at 1250 Hz for 409 msecs.

Action:	EE-LP	EF-LP	WS-LP	EE-HP	EF-HP	WS-HP
a1	-0.482	-0.156	-0.087	+0.199	+0.082	+0.204
a2	-0.166	-0.141	-0.015	+0.060	-0.078	+0.058
a3	-0.014	+0.060	-0.021	+0.086	-0.008	-0.029
a4	+0.016	-0.080	-0.255	+0.094	+0.113	-0.012
var	+0.036	+0.041	+0.013	+0.200	+0.057	+0.061

Table 3c Data sampled at 1250 Hz for 204 msecs.

Action:	EE-LP	EF-LP	WS-LP	EE-HP	EF-HP	WS-HP
a1	-0.607	-0.259	-0.185	+0.187	+0.083	+0.127
a2	-0.140	-0.161	+0.049	+0.021	-0.154	+0.024
a3	+0.046	+0.138	-0.047	+0.088	-0.035	-0.007
a4	+0.006	-0.113	-0.373	+0.152	+0.174	-0.008
var	+0.061	+0.053	+0.013	+0.205	+0.060	+0.067

Table 3d Data sampled at 1250 Hz for 102 msecs.

Table 3 Results for data sampled at 1250 Hz with various sampling durations.

EE-Elbow Extension EF- Elbow Flexion WS- Wrist Supination

Action:	EE-LP	EF-LP	WS-LP	EE-HP	EF-HP	WS-HP
a1	-0.299	-0.158	-0.022	+0.174	+0.131	+0.138
a2	-0.117	-0.025	-0.021	+0.089	+0.084	+0.012
a3	+0.029	+0.036	-0.054	+0.139	-0.041	-0.001
a4	+0.030	+0.138	+0.021	+0.063	+0.029	+0.027
var	+0.019	+0.017	+0.029	+0.147	+0.049	+0.056

Table 4a Data sampled at 833 Hz for 819 msecs.

Action:	EE-LP	EF-LP	WS-LP	EE-HP	EF-HP	WS-HP
a1	-0.386	-0.163	-0.033	+0.210	+0.125	+0.178
a2	-0.006	+0.000	+0.013	+0.136	+0.043	-0.062
a3	-0.033	-0.018	-0.216	+0.201	-0.013	+0.029
a4	+0.015	+0.111	+0.078	+0.072	+0.103	-0.005
var	+0.025	+0.024	+0.015	+0.174	+0.055	+0.045

Table 4b Data sampled at 833 Hz for 409 msecs.

Action:	EE-LP	EF-LP	WS-LP	EE-HP	EF-HP	WS-HP
a1	-0.447	-0.176	-0.050	+0.171	+0.175	+0.166
a2	-0.034	+0.034	+0.044	+0.067	-0.051	-0.129
a3	-0.049	-0.040	-0.147	+0.242	-0.121	+0.103
a4	+0.010	+0.084	+0.144	+0.128	+0.158	-0.059
var	+0.038	+0.036	+0.015	+0.151	+0.059	+0.056

Table 4c Data sampled at 833 Hz for 204 msecs.

Action:	EE-LP	EF-LP	WS-LP	EE-HP	EF-HP	WS-HP
a1	-0.598	-0.168	-0.042	+0.111	+0.105	+0.127
a2	+0.059	+0.050	+0.036	+0.161	-0.028	-0.033
a3	-0.076	-0.083	-0.352	+0.190	-0.043	+0.082
a4	+0.047	+0.079	+0.075	+0.079	+0.107	-0.048
var	+0.063	+0.050	+0.014	+0.152	+0.059	+0.049

Table 4d Data sampled at 833 Hz for 102 msecs.

Table 4 Results for data sampled at 833 Hz with various sampling durations.

EE-Elbow Extension EF- Elbow Flexion WS- Wrist Supination

Action:	EE-LP	EF-LP	WS-LP	EE-HP	EF-HP	WS-HP
a1	-0.187	-0.092	+0.011	+0.051	+0.077	+0.042
a2	+0.002	+0.026	-0.082	+0.088	+0.010	+0.015
a3	+0.015	+0.089	+0.021	-0.036	-0.008	-0.034
a4	-0.106	+0.123	-0.151	+0.024	+0.000	-0.013
var	+0.022	+0.023	+0.031	+0.165	+0.042	+0.062

Table 5a Data sampled at 625 Hz for 819 msecs.

Action:	EE-LP	EF-LP	WS-LP	EE-HP	EF-HP	WS-HP
a1	-0.225	-0.108	+0.003	+0.032	+0.027	+0.060
a2	-0.064	+0.038	-0.184	+0.093	+0.013	-0.027
a3	+0.044	+0.046	+0.033	+0.013	+0.050	+0.040
a4	-0.098	+0.142	-0.112	+0.033	+0.022	+0.086
var	+0.031	+0.035	+0.015	+0.202	+0.047	+0.051

Table 5b Data sampled at 625 Hz for 409 msecs.

Action:	EE-LP	EF-LP	WS-LP	EE-HP	EF-HP	WS-HP
a1	-0.383	-0.114	-0.051	+0.050	+0.032	+0.052
a2	-0.015	+0.001	-0.189	+0.088	+0.025	+0.020
a3	+0.037	+0.089	+0.094	-0.027	+0.115	+0.068
a4	-0.123	+0.166	-0.123	+0.040	-0.027	+0.160
var	+0.043	+0.050	+0.015	+0.206	+0.051	+0.062

Table 5c Data sampled at 625 Hz for 204 msecs.

Action:	EE-LP	EF-LP	WS-LP	EE-HP	EF-HP	WS-HP
a1	-0.448	-0.192	+0.001	-0.037	+0.010	-0.009
a2	+0.026	+0.134	-0.346	+0.122	+0.122	-0.035
a3	+0.018	+0.002	+0.193	-0.024	+0.143	+0.074
a4	-0.089	+0.256	-0.033	+0.030	-0.153	+0.314
var	+0.075	+0.059	+0.016	+0.195	+0.047	+0.067

Table 5d Data sampled at 625 Hz for 102 msecs.

Table 5 Results for data sampled at 625 Hz with various sampling durations.

EE-Elbow Extension EF- Elbow Flexion WS- Wrist Supination

Action:	EE-LP	EF-LP	WS-LP	EE-HP	EF-HP	WS-HP
a1	-0.164	-0.191	-0.006	+0.123	+0.100	+0.071
a2	-0.008	+0.013	+0.045	+0.177	+0.035	-0.048
a3	-0.153	+0.047	-0.053	-0.042	-0.081	-0.024
a4	-0.001	+0.077	+0.047	-0.015	+0.010	-0.022
var	+0.018	+0.021	+0.029	+0.163	+0.041	+0.059

Table 6a Data sampled at 500 Hz for 819 msecs.

Action:	EE-LP	EF-LP	WS-LP	EE-HP	EF-HP	WS-HP
a1	-0.312	-0.246	+0.096	+0.174	+0.097	+0.090
a2	-0.045	+0.004	+0.010	+0.178	+0.087	-0.014
a3	-0.215	-0.013	-0.054	-0.042	-0.076	+0.080
a4	+0.106	+0.170	+0.196	+0.001	+0.045	+0.143
var	+0.021	+0.031	+0.014	+0.210	+0.050	+0.047

Table 6b Data sampled at 500 Hz for 409 msecs.

Action:	EE-LP	EF-LP	WS-LP	EE-HP	EF-HP	WS-HP
a1	-0.489	-0.320	+0.102	+0.167	+0.020	+0.021
a2	-0.013	-0.024	-0.096	+0.134	+0.048	+0.028
a3	-0.186	-0.062	-0.034	+0.034	-0.046	+0.186
a4	+0.168	+0.224	+0.248	+0.043	+0.064	+0.119
var	+0.030	+0.045	+0.015	+0.1999	+0.052	+0.061

Table 6c Data sampled at 500 Hz for 204 msecs.

Action:	EE-LP	EF-LP	WS-LP	EE-HP	EF-HP	WS-HP
a1	-0.590	-0.459	+0.131	+0.287	-0.141	+0.012
a2	+0.051	-0.121	-0.071	+0.089	+0.023	-0.017
a3	-0.225	-0.024	+0.001	+0.086	-0.038	+0.186
a4	+0.231	+0.123	+0.335	+0.176	-0.142	+0.074
var						

Table 6d Data sampled at 500 Hz for 102 msecs.

Table 6 Results for data sampled at 500 Hz with various sampling durations.

action. Using more coefficients yielded more information, but the discrimination between actions was not increased enough to justify the extra processing time needed to calculate the extra autoregressive coefficients.

The results as a whole were encouraging enough to try to implement the autoregressive model on a microcomputer to test its response time and susceptibility to a smaller word length. The program was implemented on an IBM PC microcomputer because, as discussed previously, it gives a similar environment to that of the limb controller. Four programs were used to test the autoregressive model on the IBM PC and are called EMGDISK, EMGHEX, EMGAR, and ARVERIFY. These programs are discussed more completely later on in this chapter.

3.2.3 EMG Autoregressive Model Processing

After testing of the autoregressive model was completed it was decided to implement it on the limb controller. The resulting program is called ARMODEL and its general flowchart is given in Figure 22. This program was mainly implemented to show the processing power of the limb controller's signal processor. The program begins in a similar manner to the STATE3 program by resetting and clearing pass and internal control variables. Next, the program takes the data sample and using the method outlined

above calculates the autocorrelation and autoregressive coefficients. The control output is determined by finding what action the resulting coefficients most closely resemble, provided that they fall within a certain range of the action coefficients. The reference action coefficients are calculated each time the limb controller is reset. In a clinical limb controller it may be adequate to store the coefficients in permanent memory, but for this demonstration device it is best that a calibration run be executed on reset to allow easy modification of electrode placement and test subject usage. The calibration run requests the action that it wishes the user to execute by outputting the same action on the prosthetic limb. This creates a simple learning process because all the user has to do is imitate the actions of the prosthetic limb. It should be remembered that for normal subjects the limb on which the electrode is placed should be made immobile so as not to introduce nonstationary properties to the EMG signal.

3.3 EMG Signal Processing Development Software

A variety of programs were needed to test the proposed use of the autoregressive model of the EMG signal. It is not the intention of the author to discuss the operation of these programs at length in this section, but rather, to give the reader a general impression as to the operation and relationship between these programs. A more detailed

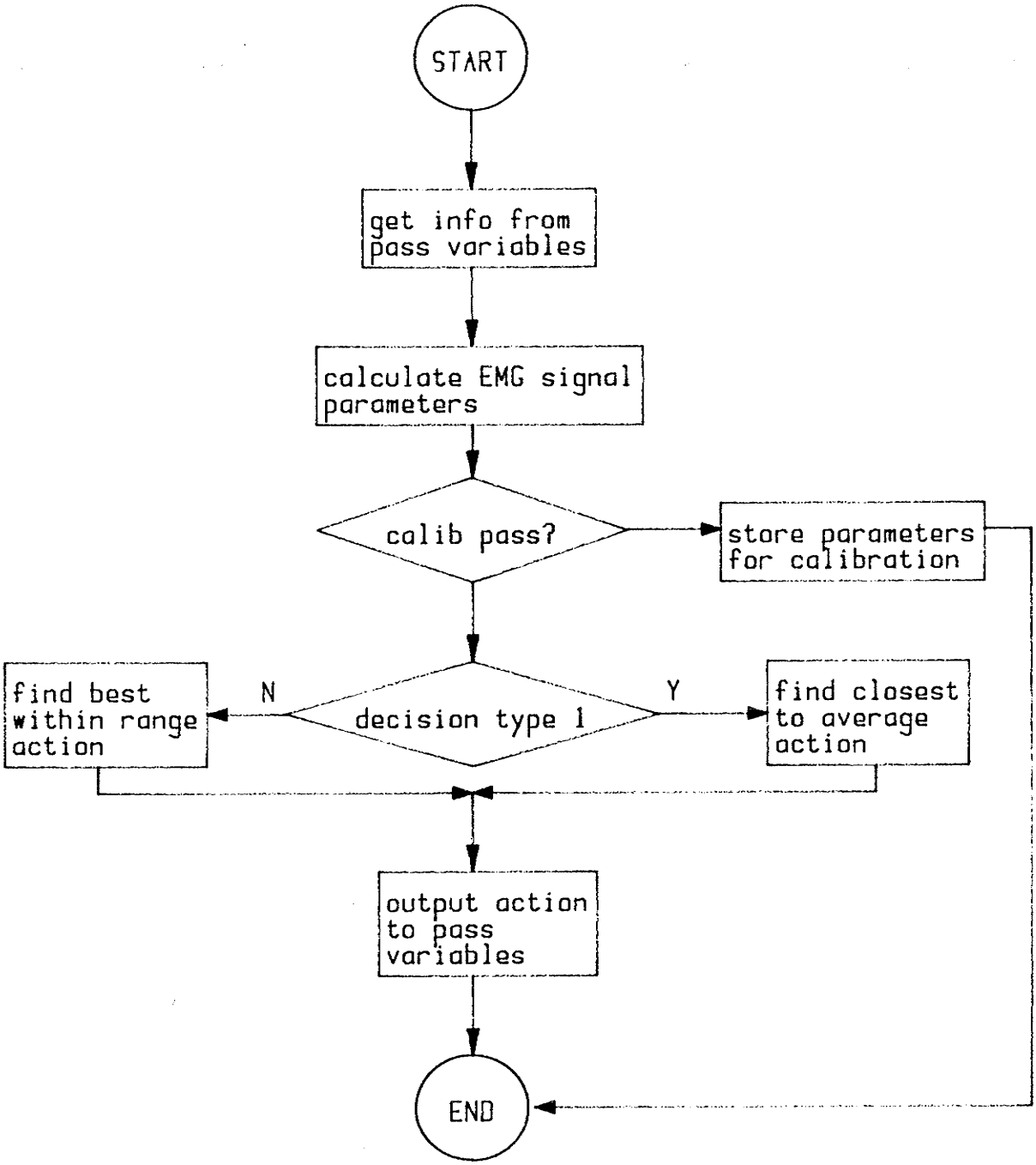


Figure 22 Flowchart of ARMODEL Program.

discussion of these programs is given in the limb controller software development manual given in the Bibliography.

The development programs create the shell for the development of a full EMG signal processing test network of possible control algorithms before they are implemented for clinical testing on the limb controller. The programs are written mostly in assembler because of the speed advantage and also because there is presently no higher language support that makes it convenient to down load the program code to the limb controller from the IBM PC used to develop the software. Any programs used for software development that were written in a higher language were not intended to be used on the limb controller.

The first program is the ARTEST program. This program was used for the initial testing of the autoregressive model on the university mainframe computer and a general flowchart describing its operation is given in Figure 23. The program calculates the autoregressive coefficients using the method given in Section 3.2.2. The program simulates different sampling rates and sampling durations by selecting data from the EMG data files in differing orders. For example, knowing that the sampling rate used was 2500 Hz a rate of 1250 Hz can be simulated by selecting every second data item. Differing rates can be simulated by similar techniques. Counting the number of data items selected and knowing the

sampling frequency being simulated, the sampling duration can also be varied. The results of the program are outputted to a data file for possible further study. The results given in Tables 2 to 6 show an example of the output from this program.

After initial testing it was decided to do further tests on a microcomputer to simulate the model's performance on the limb controller. The computer selected was an IBM PC/XT for reasons outlined previously. Two data acquisitions programs were written to acquire EMG signal data and store it on disk for further analysis by other programs. The first data acquisition program is the EMGHEX program whose general flowchart is given in Figure 24. The program first requests the sampling rate and duration desired for sampling as well as the data file to be used for storage. Next, the analog to digital converter is setup for the sampling rate and duration requested and the EMG signal data are acquired. For this program the EMG signal data are stored on disk exactly as it would be in memory. This data can then be used by the EMGAR program.

The second data acquisition program is the EMGDISK program and a general flowchart is given in Figure 25. This program operates in a similar manner to the EMGHEX program for acquiring the data but the storage on disk is in a much different manner. The data are converted to a binary coded

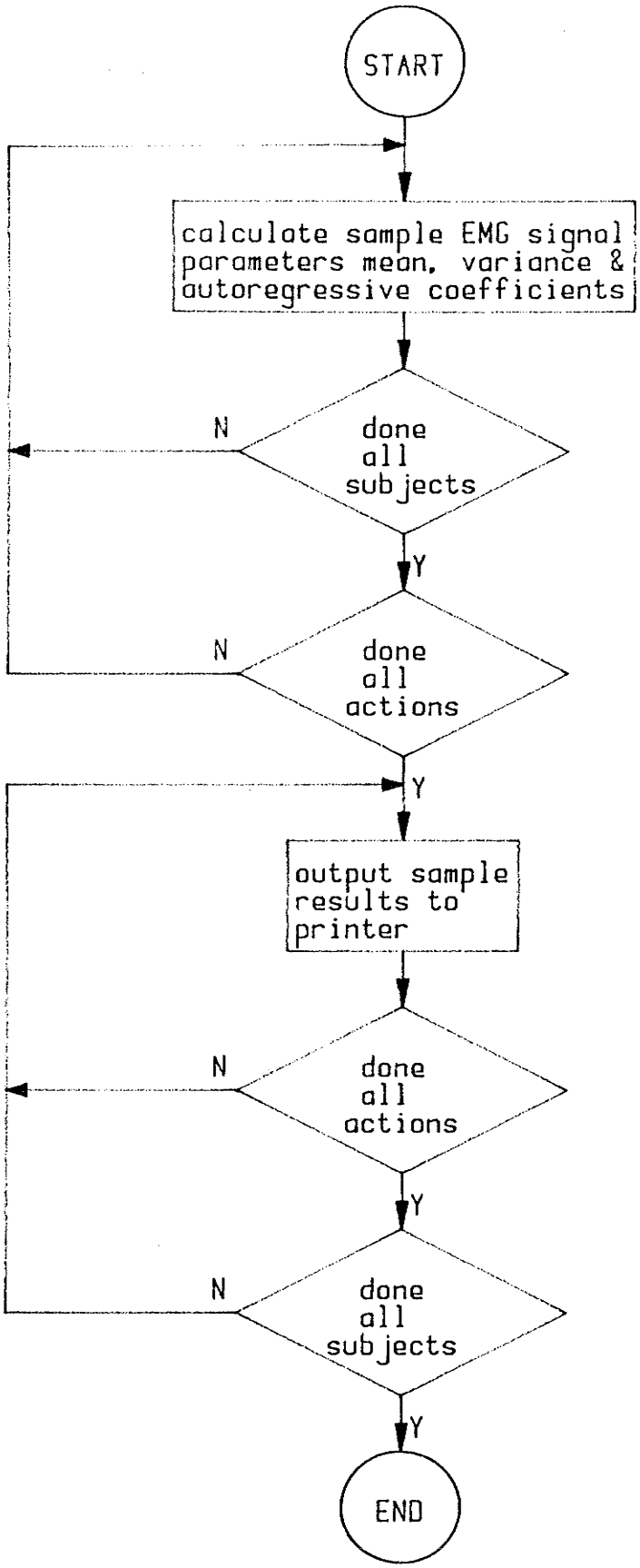


Figure 23 Flowchart of ARTEST Program.

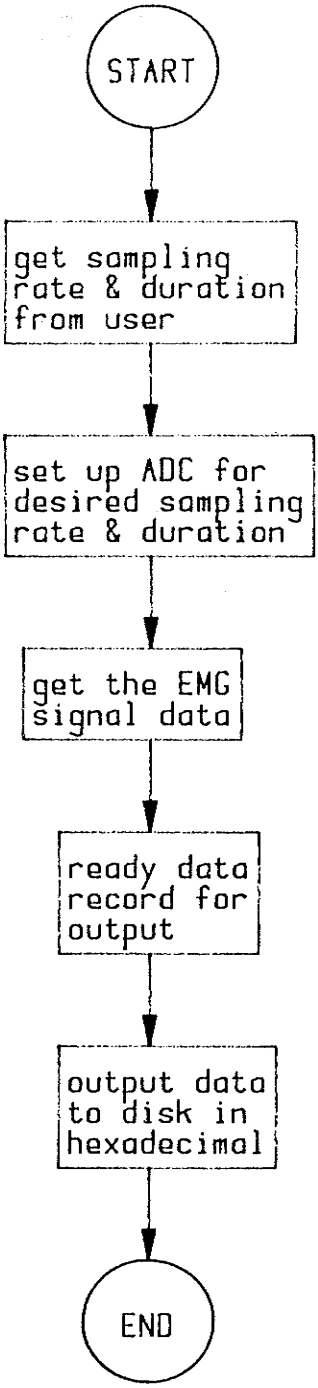


Figure 24 Flowchart of EMGHEX Program.

data (BCD) form before it is stored on disk. This allows testing by higher language programs without necessitating a conversion routine from hexadecimal to BCD. The resulting data can then be used by programs similar to the ARVERIFY program.

The EMGAR program is used to test the autoregressive model coefficient calculation and decision algorithms. A general flowchart of the EMGAR program is given in Figure 26. The program first requests the sampling rate and duration desired and also, whether the data are to be acquired directly from a test subject or if the data are to be taken from disk. In addition, the program requests whether a debugging output is to be given or a feedback display. The debug display outputs the autoregressive coefficients along with the action selected on screen while the feedback display gives a bar graph display of the action selected versus other actions for training purposes. The EMGAR program then sets up the analog to digital converter and acquires the EMG signal data or gets the data from disk.

Next the program calculates the autoregressive coefficients using the technique described in Section 3.2.2. After the coefficients have been calculated they are readied for output. If this is a calibration run then the program exits to this procedure. However, for normal operation if the debug display has been selected the coefficients are

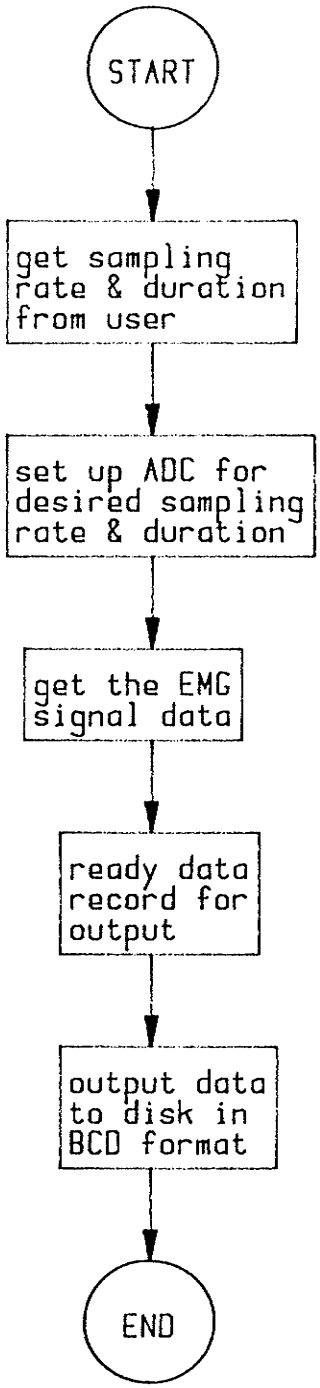


Figure 25 Flowchart of EMGDISK Program.

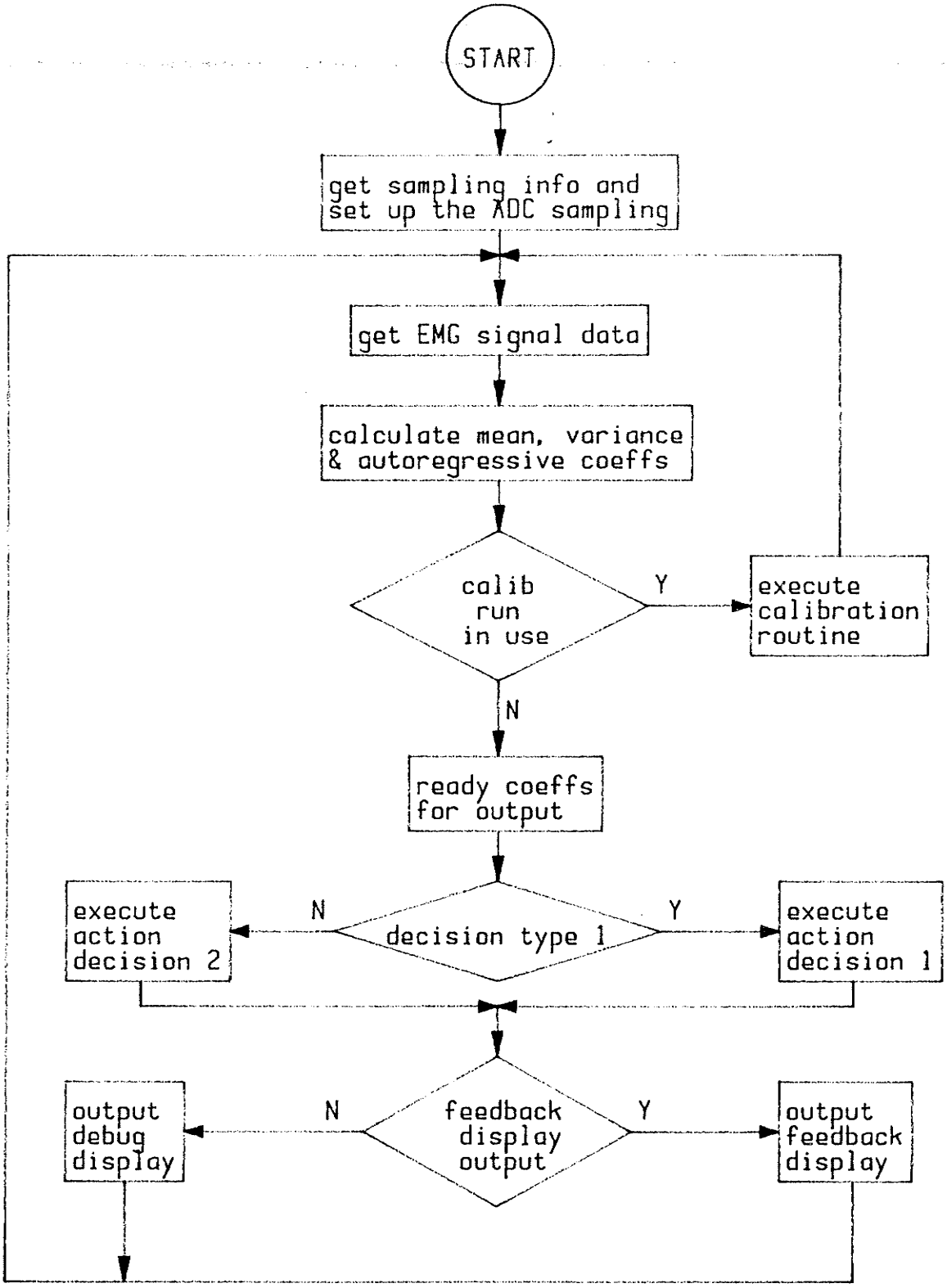


Figure 26 Flowchart of EMGAR Program.

outputted. An example of the debug display output is given in Figure 27. Two action decision algorithms are available for use as selected by the user. Decision type 1 bases its decision on finding which action the coefficients just calculated correspond most closely, while decision type 2 counts the number of times that each coefficient came within the range of the action coefficients as calculated by the calibration procedure. The action to which the most coefficients came within range is outputted as the decision. If the feedback display is selected the bar graph output is loaded with the various action counts from either one of the two decision algorithms. The results outputted to the screen allow the user to see what action is selected and also how closely the actions were to being classified as another action. An example of this display is given in Figure 28.

The calibration procedure of the EMGAR program is used to allow the user to calibrate the various autoregressive coefficients for each action. The user may select the number of trials the calibration is to be taken over and also has control over when the EMG signal data are to be taken. After all the coefficients have been calculated the EMGAR program calculates the average coefficient value, the power level to define low and high power actions and finally the range around the calculated average coefficient. The results may then be outputted to the printer for a hard copy of the calibration procedure or for further testing. An example of

```
Hit any key when ready to continue
Option A-ACTION DECISION,C-CALIBRATE,R-CHANGE SAMPLING RATE,D-CHANGE DATA TYPE
MEAN= 0000   VAR= 0001   A1= FFF8   A2= 8001   A3= FFF5   A4= 3FFF
    EELP         EBLP         WSLP         EEHP         EBHP         WSHP
     1            3            0            0            0            0
MEAN= 0000   VAR= 0001   A1= FFF8   A2= 8001   A3= FFF5   A4= 3FFF
    EELP         EBLP         WSLP         EEHP         EBHP         WSHP
     1            3            0            0            0            0
MEAN= 0000   VAR= 0001   A1= FFF8   A2= 8001   A3= FFF5   A4= 3FFF
    EELP         EBLP         WSLP         EEHP         EBHP         WSHP
     1            3            0            0            0            0
MEAN= 0000   VAR= 0001   A1= FFF8   A2= 8001   A3= FFF5   A4= 3FFF
    EELP         EBLP         WSLP         EEHP         EBHP         WSHP
     1            3            0            0            0            0
MEAN= 0000   VAR= 0001   A1= FFF8   A2= 8001   A3= FFF5   A4= 3FFF
    EELP         EBLP         WSLP         EEHP         EBHP         WSHP
     1            3            0            0            0            0
MEAN= 0000   VAR= 0001   A1= FFF8   A2= 8001   A3= FFF5   A4= 3FFF
    EELP         EBLP         WSLP         EEHP         EBHP         WSHP
     1            3            0            0            0            0
MEAN= 0000   VAR= 0001   A1= FFF8   A2= 8001   A3= FFF5   A4= 3FFF
    EELP         EBLP         WSLP         EEHP         EBHP         WSHP
     1            3            0            0            0            0
MEAN= 0000   VAR= 0001   A1= FFF8   A2= 8001   A3= FFF5   A4= 3FFF
    EELP         EBLP         WSLP         EEHP         EBHP         WSHP
     1            3            0            0            0            0
```

Figure 27 Example of EMGAR Debug Display.

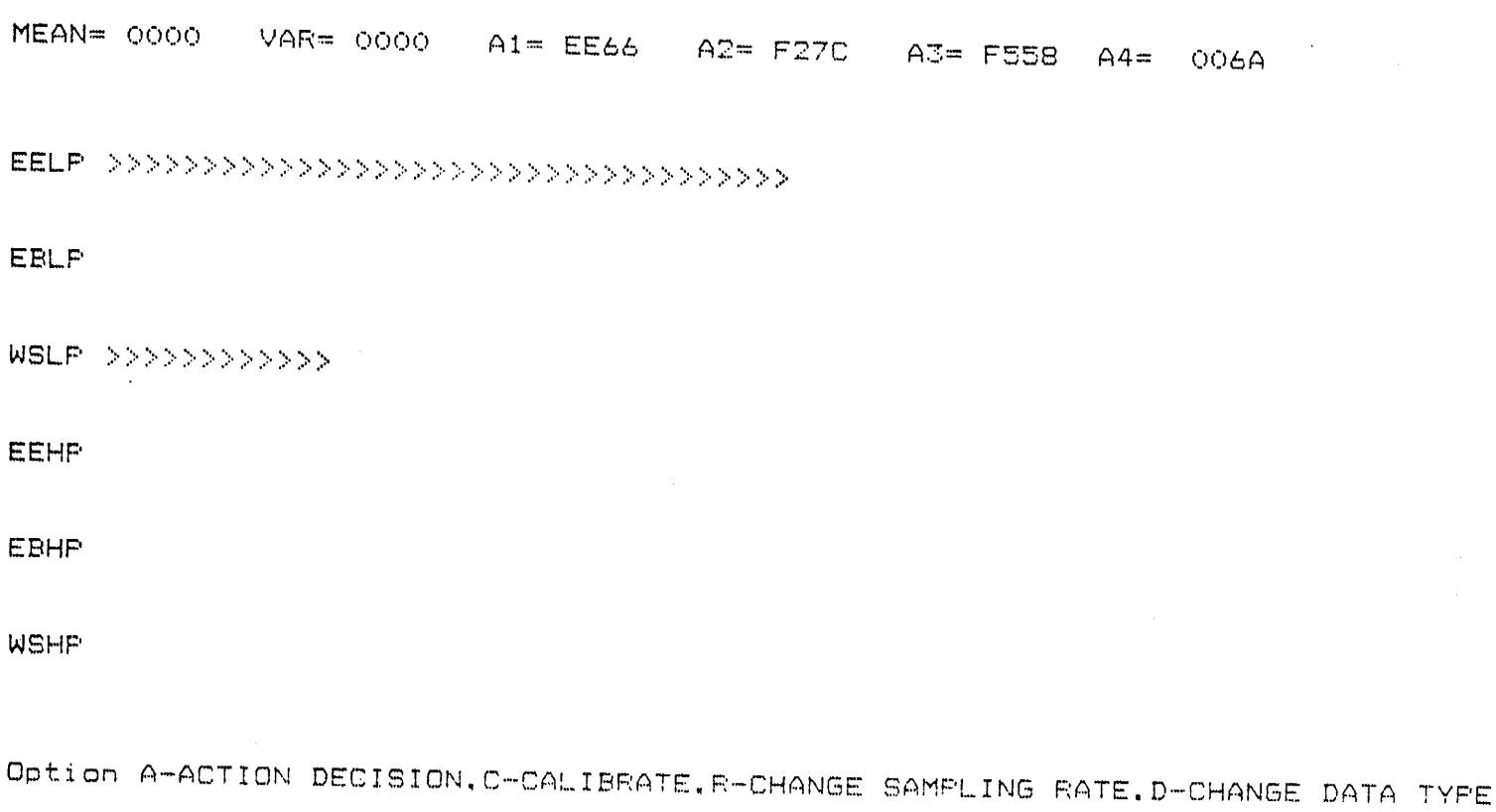


Figure 28 Example of EMGAR Feedback Display.

```
Low power ELBOW EXTENSION test
VAR= 0001   A1= 0C06   A2= DF29   A3= E508   A4= F095
VAR= 0001   A1= E15C   A2= F805   A3= F875   A4= 07CF
Averages  are
VAR= 0001   A1= F6B1   A2= EB97   A3= EEBF   A4= FC32
Ranges are
VAR= 0000   A1= 1555   A2= 0C6E   A3= 09B7   A4= 0B9D

Low power ELBOW FLEXION test
VAR= 0004   A1= CD5F   A2= E9B1   A3= 0674   A4= 0D5F
VAR= 0003   A1= E7B3   A2= AEC5   A3= 0A53   A4= 2B40
Averages  are
VAR= 0003   A1= DA89   A2= CC3B   A3= 0863   A4= 1C4F
Ranges are
VAR= 0001   A1= 0D2A   A2= 1D76   A3= 01F0   A4= 0EF1

Low power WRIST SUPINATION test
VAR= 0001   A1= E421   A2= EA02   A3= F693   A4= 07D2
VAR= 0001   A1= EA3D   A2= E76C   A3= EFA7   A4= 0864
Averages  are
VAR= 0001   A1= E72F   A2= E8B7   A3= F31D   A4= 081B
Ranges are
VAR= 0000   A1= 030E   A2= 014B   A3= 0376   A4= 0049

High power ELBOW EXTENSION test
VAR= 002C   A1= C02B   A2= 1CDF   A3= F8E9   A4= 0629
VAR= 0048   A1= BC4B   A2= 18C3   A3= 06D2   A4= 0076
Averages  are
VAR= 003A   A1= BE3B   A2= 1AD1   A3= FFDE   A4= 034F
Ranges are
VAR= 000E   A1= 01F0   A2= 020E   A3= 06F5   A4= 02DA

High power ELBOW FLEXION test
VAR= 0075   A1= A226   A2= 019F   A3= 487D   A4= D98F
VAR= 00A9   A1= A985   A2= 0A8C   A3= 1ED4   A4= F865
Averages  are
VAR= 008F   A1= A5D6   A2= 0615   A3= 33A8   A4= E8FA
Ranges are
VAR= 001A   A1= 03B0   A2= 0477   A3= 14D5   A4= 0F6B

High power WRIST SUPINATION test
VAR= 0004   A1= D1DA   A2= F3D3   A3= 0E5F   A4= 053A
VAR= 0007   A1= D2CE   A2= F495   A3= 00CC   A4= 12B8
Averages  are
VAR= 0005   A1= D254   A2= F434   A3= 0795   A4= 0BF9
Ranges are
VAR= 0002   A1= 007A   A2= 0061   A3= 06CA   A4= 06BF
```

Figure 29 Example EMGAR Calibration Output.

the calibration output is given in Figure 29.

The final program, ARVERIFY, is in itself not very useful, but is included as an example of how a higher level language program can use the data files created by the EMGHEX and EMGDISK program. The ARVERIFY program was used to verify the correct operation of the EMGAR coefficient calculation and a general flowchart is given in Figure 30. Notice that to read data from data files created by the EMGDISK program only a standard read process is needed because of the way the data has been formatted by the EMGDISK program. In addition, the sampling rate and duration are given in the first two data items respectively. However, as shown in the ARVERIFY program, data files created by the EMGHEX program can also be used as long as the EMG signal data are converted from hex to decimal when it is read from the data file.

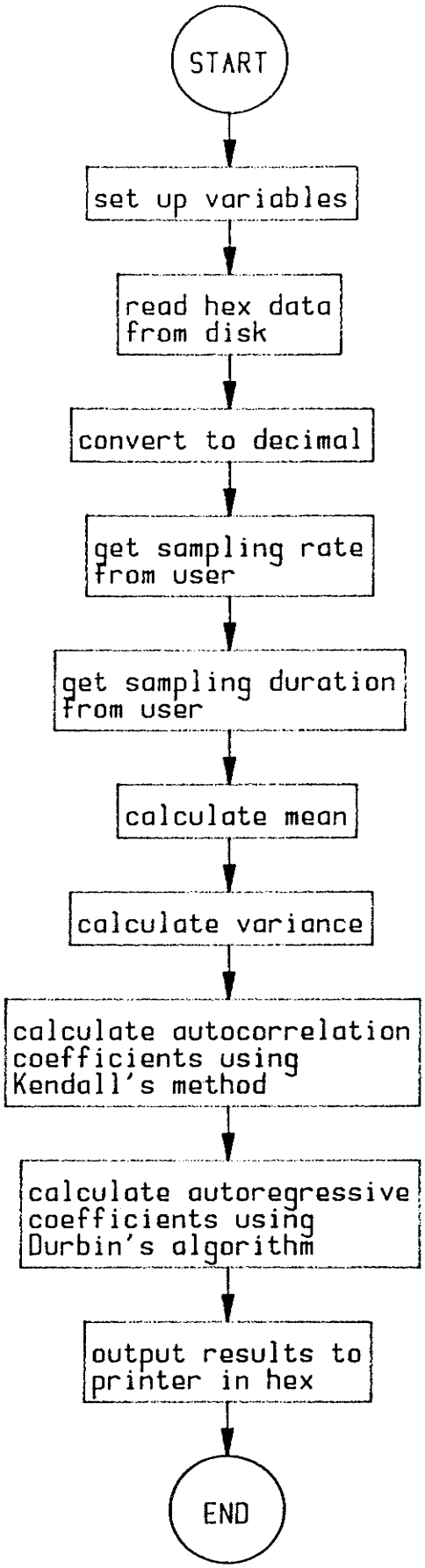


Figure 30 Flowchart of ARVERIFY Program.

CHAPTER 4

TESTING: RESULTS AND DISCUSSION

The testing and evaluation of the limb controller was based on the controller's ability to effect the required function than on some numerical performance criteria because numerical results would be inapplicable here. This is because this limb controller is at a very preliminary stage of design and will need many refinements before it will be ready to be implemented for full clinical trials. It was decided that meeting the goal of developing a computer based myoelectric limb controller would be shown if the following criteria could be met:

1) it could be demonstrated that all sections of the limb controller operated as designed.

2) that the software could duplicate the present technology of clinical myoelectric limb controllers (i.e. two and three state control from several muscle sites).

3) correct operation could be accomplished for several users.

The second goal of examining the feasibility or infeasibility of using microcomputers as myoelectric limb controllers will be discussed more on the basis of the author's experiences in the design of this limb controller than on a full-fledged evaluation procedure. The projects that will follow this initial development project will need to include a more complete evaluation of the adequacy of using microcomputers in myoelectric limb controllers.

The limb controller presently available is the second prototype of a computer based myoelectric limb controller. The first prototype discussed in Chapter 2 under Limb Controller Implementation was also designed and built as part of this thesis and so in the discussion of test results both controllers will be included.

4.1 Test Procedures

As mentioned above the limb controller was tested on the basis of "Does it work?". That is, all possible operations and modes of the limb controller hardware and software were tested. The order of testing was as follows:

1. verification of the operation of each microcomputer within the limb controller both separately and together using the shared memory.

2. verification of the operation of the data acquisition subsystem, testing both the analog and digital circuits.

3. verification of the operation of the joint motor control and limb position feedback subsystem with and without microprocessor control.

4. selection of limb actions using a variable voltage supply to select the desired actions; - single function, multifunction, and preprogrammed movements.

5. implementation of two state control, three state control, and autoregressive model control algorithms.

4.2 Results and Discussion

The first four test procedures deal with verifying the operation of the hardware of the limb controller. This was accomplished by running test programs on the limb controller computers and insuring that the microprocessors, memory (both shared and local), and control circuitry operated in a correct manner. The hardware could be easily tested by using logic analyzers, oscilloscopes, multimeters, and a variety of other test equipment, but the correct operation of the limb controller software demands closer scrutiny.

Testing of two and three state control was done by first verifying that the control algorithm could be implemented and then operated by the author. After it was found that these control algorithms could be operated by the author, several subjects were used to verify that the device could be operated by a wide range of users, most with little or no technical knowledge as to the operation of the limb controller. Two state control was the easiest of the two control methods, with the subject typically able to move the prosthetic limb at will after only a few tries. Three state control was more difficult to learn, but after several minutes most subjects could operate the prosthetic limb with a reasonable degree of accuracy. For three state control it was found that the two power thresholds had to be customized to each user. This is similar to clinical three state limb controllers where the threshold values must be adjusted for each individual.

The autoregressive model provided less than desirable control because even the best user could only attain up to 50% accuracy on the EMGAR program written on the IBM PC and less than this using the ARMODEL program on the limb controller. This is much less than expected from the tests done initially. However, this could be explained by the fact that fixed point arithmetic was used in the autoregressive coefficient calculation instead of floating point arithmetic. Fixed point arithmetic speeds up the coefficient

calculation, but allows inaccuracies to creep into the calculation. Fixed point is necessary to maintain the processing speed because floating point calculations take much more time than fixed point calculations. Therefore, until these problems with the autoregressive model can be addressed and solved the autoregressive model will not provide good prosthesis control. However, the potential is there and it is suggested that further work on the autoregressive model be done.

The autoregressive model running on the limb controller does display one very important characteristic. This the ability of the limb controller to implement computationally orientated algorithms and execute them in real time. It also shows that the control memory is large enough to handle complex control programs.

The limb controller does demonstrate that a computer based myoelectric limb controller is feasible and, in fact, is a very desirable device. The limb controller provides many facilities that conventional myoelectric limb controllers do not, and never will be able to. Once the hardware on this controller was debugged and operational the present technology for limb controllers (i.e. two and three state control) was implemented in software in approximately three days. As well, this could be customized for each user with quick and relatively simple software changes. This

development time for software is of course much faster than for comparable hardware development. The limb controller provides facilities that cannot be duplicated by any other non-computer based limb controller such as preprogrammed multifunctional movements, superimposed movements, and limb position feedback. These potential advances in myoelectric prosthesis control were barely utilized in this project and the potential for the limb controller is almost limitless. There are some problems such as power consumption and size, but as discussed previously these can be overcome, the power consumption by some design changes and miniaturization of sections of the limb controller by using VLSI, hybridization, and other miniaturization techniques.

The author is quite convinced from his experiences in designing the limb controller, and from observing the prosthetic limb industry as an electrical engineer in a medical world, that clinical computer based myoelectric limb controllers will occur within the next several years. These devices will probably not be as sophisticated as this limb controller, but they will incorporate many of the ideas and control advantages presented in this project. It is the author's hope that this project may be used to help in creating more advanced myoelectric limb controllers that will provide amputees with improved prostheses, allowing them a chance to better utilize those tools that normal human beings take for granted.

CHAPTER 5

RECOMMENDATIONS

The microcomputers of the limb controller work as designed, but there are several things that must be corrected in a third prototype of the limb controller. There is a large amount of noise on the power supply lines of the computers. This noise enters the data acquisition subsection despite its isolation through two dc-dc converters. This causes additional inaccuracies to those normally present in the measurement of the EMG signal. The inaccuracies are not large because the noise is of very high frequency, but it is conceivable that the analog to digital converter gives occasional incorrect measurements because of this high frequency noise. It was found that the noise is caused by the 82C84 clock driver chip used to supply the clock frequencies for the limb controller. This noise seems to be inherent in the 82C84 because several chips were tried. The noise is peculiar because it has a large fundamental at 20 MHz. In the third prototype it is suggested that a different method of clock generation be examined so as to remove this noise.

A great deal of logic is needed to control the operation of the limb controller support circuitry. In the

future, programmable array logic (PAL) should be used to reduce the chip count and thereby save space and possibly power.

The data acquisition subsection works well and is very accurate considering the noise problems on the power supply lines. However, as mentioned previously it accounts for most of the power usage by the limb controller. Therefore, in the future a method of data acquisition using less power should be developed. This should involve some of the expensive very low power CMOS operational amplifiers rather than the less expensive low power JFET operational amplifiers.

The electrode interface to the limb controller presently consists of shielded cables. This is awkward and should be replaced by a less cumbersome technique which could include transmission of the EMG signals by small transmitters.

The joint motor control subsection operates as it should except that there is at present no method for implementing proportional control of the joint motors. Proportional control is an up and coming feature of clinical myoelectric limb controllers and so in a third limb controller prototype the ability to implement proportional control should be included.

Limb position feedback data acquisition works as it should. However, there is presently no transducer to measure the position of each joint. Therefore, a transducer to measure the position of each joint should be developed to make use of the limb position feedback abilities.

Software additions should include the implementation of several practical preprogrammed and multifunction movements. This should involve research to find several practical preprogrammed movements as well as developing an easy to use technique for invoking those actions. As well, several promising new control algorithms should be implemented and tested, in addition to refining the autoregressive model technique already implemented.

There is also a need to develop a model to make the prosthetic limb motion more natural. This could be done by examining models of human limb motion and developing a model of the prosthetic limb motion and then correlating the two models.

CHAPTER 6

SUMMARY AND CONCLUSIONS

To summarize a large and extensive project, such as this one, is never an easy task for an author. The temptation to gloss over what to the author is obvious is continuous as is the temptation to move to the other extreme and explain everything in detail to the reader. Throughout this thesis the author has attempted to shy away from the detail and present the overall concepts and results of the work. In summarizing this project this too is the author's desire.

The most important result of the work is the demonstration of the feasibility of using both low computing power and high computing power microcomputers in clinical myoelectric prostheses. This feasibility ranges from the power consumption of less than 500 mA (with the very real possibility of reducing this substantially) to the increased functionality of the prosthesis that this limb controller provides. The advantages that this type of limb controller provides over present analog myoelectric limb controllers are; increased availability to previously medically ineligible amputees (eg. spastics, etc...); easy

implementation of custom prosthesis control algorithms through software; provisions for a variety of preprogrammed movements; ability to give superimposed movements; joint position feedback; and easier development of new clinical prosthesis control algorithms.

The limb controller in its present form is designed to be used as a developmental tool for future prosthesis control algorithms and also for hardware development of new microcomputer based myoelectric limb controllers. It allows the implementation and testing of various limb control algorithms in real time rather than in computer simulation and shows the hardware processing requirements needed for real time implementation of these control algorithms on microprocessors. Since the control algorithms are developed in software the development time for new control algorithms is relatively quick as compared to hardware development. A demonstration of the above was the implementation of the two and three state control algorithms in just three days and also the implementation of the autoregressive model control algorithm in several weeks.

The two and three state control algorithms worked as well as those in use in present clinical limb controllers thereby demonstrating that the limb controller can already provide as good control as any present clinical myoelectric limb controller. However, the autoregressive model of the

EMG signal did not provide sufficiently accurate control to be presently considered for implementation in a clinical device. The algorithm does show great promise and continued work should provide better control.

REFERENCES

[1] Jacobsen et al., Development of the Utah Artificial Arm, IEEE Transactions on Biomedical Engineering, vol BME-29, no. 4, April 1982, pp. 249-269.

[2] Reiter R., Eine Neue Eltrokunsthand, Grenzbebiete der Medizin, vol 4, no. 133, 1948.

[3] Alderson S.W., The electric arm in Human Limbs and Their Substitutes, Klopsteg and Wilson, Ed. New York: Hakner, 1968, pp. 359-410 (reprint of McGraw-Hill, 1954).

[4] Product Literature on Boston Elbow, Liberty Mutual Insurance Company, Boston, Mass.

[5] Mason C.P., Design of a Powered Prosthetic Arm for the Above Elbow Amputee, Bulletin of Prosthetic Research., Fall 1972.

[6] Peizer E., Wright D.W., Mason C., and Pirrello T., The Otto Bock Hand in Guidelines for Standards for Externally Powered Hands, Bulletin of Prosthetic Research, Fall 1969.

[7] Product Literature on Fidelity Hand. Chicago, IL: Fidelity Electronics.

[8] Childress et al., VA/NU Myoelectric System for Below Elbow Amputees Bulletin of Prosthetic Research, Fall 1972 pp. 232.

[9] Schimdl H., The INAIL-CECA Prostheses, Orthodics, Prosthetics, vol 27, March 1973, pp. 6-12.

[10] Product Literature on New York Elbow, New York University, New York, New York.

[11] McLaurin et al., Annual Report., Prosthetic Research and Training Unit, Ontario Crippled Children's Centre, Toronto Ont. October 1970.

[12] Scott R.N. et al., Myoelectric Control Systems - Progress reports 5-14, University of New Brunswick, Canada, 1970.

[13] Jacobsen S.C., Jerard R.B., and Knutti D.F., Development and Control of the Utah Arm, Proc. Fifth Int. Symp. Ext. Cont. Human Extremities, Yugoslavia, Aug 25-30, 1975.

[14] Funakubo H. et al, Application of Microcomputer to the Total Arm Prosthesis and to the Environmental Control System for Bedridden Patient, Uses of Computers in Aiding the Disabled, J. Raviv (editor), North-Holland Publishing Co. 1982.

[15] Funakubo H. et al, The Control of an Electrically Powered Orthosis Microcomputers and Multi-Commands, Proceedings of 4th Annual Conference on Rehabilitation Engineering, Washington D.C., 1981.

[16] Funakubo H. et al., Total Arm Prosthesis Driven by 12 Micro-Motors, Pocketable Microcomputer and Voice and Look-Sight Microcommanding system, Proceedings of International Conference on Rehabilitation Engineering, Toronto, Canada, pp. 39-42.

[17] Jacobsen et al., Development of the Utah Artificial Arm, IEEE Transactions on Biomedical Engineering, vol BME-29, no. 4, April 1982, pp. 249-269.

[18] Schimdl H., The INAIL-CECA Prostheses, Orthodics, Prosthetics, vol 27, March 1973, pp. 6-12.

[19] Product Literature - Otto Bock Myoelectric Limbs. Otto Bock Corporation, Germany.

[20] Scott R.N. et al. Myoelectric Control Systems Progress Reports #5 - 17, University of New Brunswick, Canada.

[21] Paciga J.E., Richard P.D., and Scott R.N., Error Rate in Five-State Myoelectric Control Systems, Medical and Biological Engineering and Computing, vol. 18, 1980, pp. 287-290.

[22] Jacobsen et al., Development of the Utah Artificial Arm, IEEE Transactions on Biomedical Engineering, vol BME-29, no. 4, April 1982, pp. 249-269.

[23] Scott R.N. et al, Myoelectric Control Systems Progress Report 17, University of New Brunswick, Canada, 1980.

[24] Jacobsen S.C. and Mann R.W., Control Systems for Artificial Arms, Proceedings of IEEE Conference Systems, Man, and Cybernetics, Nov. 1973.

[25] Wood J.E., Theoretical Formalism for the Kinesiological Trajectories of a Computer Simulated Neuro-musculo-skeletal System, Doctoral Dissertation M.I.T., Cambridge MA.

[26] Simpson D.C., The Choice of Control System for Multimovement Prosthesis: Extended Physiological Proprioception (E.P.P.), The Control of Upper- Extremity Prostheses and Orthoses, chapter 15, Herberts, Peter et al., Ed., C.C. Thomas Springfield IL. 1974.

[27] Childress D. et al., VA/NU Myoelectric Control System for Below Elbow Amputees: Contractor Report, Bulletin of Prosthetic Research, vol. 10-18, pp. 225, Fall 1972.

[28] Hogan N., Myoelectric Prosthesis Control: Optimal Estimation Applied to EMG and the Cybernetic Considerations for its use in a Man - Machine Interface, Ph.D. dissertation, M.I.T. Cambridge MA. 1976.

[29] DeLuca C.J., Control of Upper Limb Prostheses: A Case for Neuroelectric Control, Journal of Medical Engineering Technology, vol. 2 March 1978.

[30] Stein R.B. et al., New Approaches for the Control of Powered Prostheses, Particularly High-Level Amputees, Bulletin of Prosthetic Research, vol. 17, pp. 10-33, Spring 1980.

[31] Graupe E., Control of Upper Limb Prostheses in Several Degrees of Freedom, Bulletin of Prosthetic Research, pp. 226-235, Fall 1975.

[32] Sardis G.N. and T.P. Gootee, EMG Pattern Analysis and Classification for a Prosthetic Arm, IEEE Transactions on Biomedical Engineering. vol. BME-29, pp. 403-412, June 1982.

[33] Jerard J.E. and Jacobsen S.C., Laboratory Evaluation of a Unified Theory for Simultaneous Multiple Axis Artificial Arm Control, ASME Transactions, vol. 102, pp. 199-207, Aug. 1980.

[34] Shwedyk E and Fleisher S., Sequential Multistate EMG Signal Processor, IEEE Transactions on Biomedical Engineering. vol. BME-26, pp. 549-556, Oct. 1979.

[35] Nichol B., Personal communication with author, Rehabilitation Center for Children, Winnipeg, Canada, June 1983.

[36] Winter David A., Biomechanics of Human Movement John Wiley and Sons Inc. New York N.Y., 1979 pp. 134.

[37] Doerschuk P., Gustafson D., and Willsky A., Upper Extremity Limb Function Discrimination Using EMG Signal Analysis, IEEE Transactions on Biomedical Engineering, vol. BME-30, no. 1, pp. 18-29.

[38] DeLuca C.J., Control of Upper Limb Prostheses: A Case for Neuroelectric Control, Journal of Medical Engineering Technology, vol. 2 March 1978.

[39] High-Speed CMOS Logic Databook, Motorola Inc., 1983, chapter 4, pp. 1-15.

[40] MM54HC/74HC High-Speed CMOS Family Databook, National Semiconductor Corp., 1983, chapter 2, pp. 1-53.

[41] Mangerei A., Wire Wrapping and Proto-System Techniques, Byte, vol. 6,no. 5, May 1981, pp. 152-170.

[42] Shwedyk, E. et al. A Non-stationary Model for the Electromyogram, IEEE Transactions on Biomedical Engineering, BME-24, 1977.

[43] Graupe D. and Cline W., Functional Separation of EMG signals via ARMA Identification Methods for Prosthesis Control Purposes, IEEE Transactions on Systems Science and Cybernetics, March 1975 vol. SMC-5, pp. 252-258.

[44] Graupe D., Magnussen J., and Beex A., A Microprocessor System for Multifunctional Control of Upper Limb Prosthesis via EMG Signal Identification, IEEE Transactions on Automatic Control, Aug 1978, vol. AC-23, pp. 538-544.

[45] Graupe D., Control of an Artificial Limb in Three Degrees of Freedom, Bulletin of Prosthetic Research, Fall 1973, No. 10-20, pp. 331-332.

[46] Graupe D., Salahi J., and Kohn K., Multifunctional Prosthesis and Orthosis Control via Microcomputer Identification of Temporal Pattern Differences in Single Site Myoelectric Signals, Journal of Biomedical Engineering, Jan 1982, vol. 2, pp.17-22.

[47] Makhoul J., Linear Prediction: A Tutorial Review, Proceedings of the IEEE, April 1975, vol. 63, pp. 561-580.

[48] Durbin J., The Fitting of Time Series Models, Rev. Inst. Int. Statistics, 1960, vol. 28, no. 3, pp. 233-243.

[49] Doerschuk P., Gustafson D., and Willsky A., Upper Extremity Limb Function Discrimination Using EMG Signal Analysis, IEEE Transactions on Biomedical Engineering, vol. BME-30, no. 1, pp. 19.

[50] Graupe D., Salahi J., and Kohn K., Multifunctional Prosthesis and Orthosis Control via Microcomputer Identification of Temporal Pattern Differences in Single Site Myoelectric Signals, Journal of Biomedical Engineering, Jan 1982, vol. 2, pp.17-22.

BIBLIOGRAPHY

[1] Hortensius P.D., An Investigation of Improved Myoelectric Prosthesis Control using Microprocessors Volume 2 Limb Controller Hardware Manual, Rehabilitation Center for Children, Winnipeg, Manitoba, Canada, September 1984, 75 pages, illustrated.

[2] Hortensius P.D., An Investigation of Improved Myoelectric Prosthesis Control using Microprocessors Volume 3 Limb Controller Software Manual, Rehabilitation Center for Children, Winnipeg, Manitoba, Canada, September 1984, 151 pages, illustrated.

[3] Hortensius P.D., An Investigation of Improved Myoelectric Prosthesis Control using Microprocessors Volume 4 Limb Controller Software Development Manual, Rehabilitation Center for Children, Winnipeg, Manitoba, Canada, September 1984, 228 pages, illustrated.