LAMASS: A LARGE-MOTION ACTIVE SUSPENSION SYSTEM FOR POWERED WHEELCHAIRS

by

William W. Li

B.A.Sc., Simon Fraser University, 1993

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCES

in the School

of

Engineering Science

SIMON FRASER UNIVERSITY

December 1997

Approval

Name: William W. Li

Degree: M.A.Sc.

Title of thesis: LAMASS: A Large-Motion Active Suspension System for Powered Wheelchairs

Examining Committee:

Chair: Dr. William A. Gruver
Professor
School of Engineering Science

Senior Supervisor:

Dr. Shahram Payandeh
Associate Professor
School of Engineering Science

Supervisor:

Dr. Andrew H. Rawicz
Professor
School of Engineering Science

Examiner:

Dr. John Jones
Associate Professor
School of Engineering Science
Simon Fraser University

Date Approved: December 11, 1997

ABSTRACT

The design of a large-motion active suspension system for powered wheelchairs is described and discussed. A common problem with powered wheelchairs is their inability to navigate steep or rough terrain. As the seat support is typically rigidly fixed to the chassis, travelling along or across a slope can result in the wheelchair's user sliding uncomfortably from side to side. In more extreme cases, it is possible to lose control of the wheelchair as the center of gravity of the combined user and wheelchair system tilts out beyond the footprint formed by the wheels contacting with the ground. By moving the seat and user, which typically comprise half of the mass of the wheelchair, relative to the wheelchair chassis, it is possible to maintain equivalent stability with a smaller footprint or enhanced stability for a given chassis as the wheelchair tilts according to the terrain it moves across. Any device which accomplishes this type of motion, though, must satisfy strict constraints with regards to safety, reliability, production cost, power consumption, and size. One important constraint is the need to devise the mechanism such that it fits within the space already beneath the wheelchair's seat without adding bulk. One solution is to use a self-levelling type of active suspension to provide the motion. A proof-of-concept of the Large Motion Active Suspension System (LaMASS) has been built to test out this idea. LaMASS moves a scale model of a human user according to input from inertial sensors mounted on the wheelchair chassis, attempting to keep the user level regardless of the external acceleration input. A design technique to satisfy the design constraints is discussed, and the results of the LaMASS controller are analyzed.

Dedication

This thesis is dedicated to the memory of Dr. Tom Richardson, whose example inspired me to learn more, whose support brought out my best, and whose lessons will guide me for the rest of my life.

Acknowledgements

I would like to thank the guidance, support, and encouragement of my supervisor, Dr. Payandeh, without whom this work would not have been completed. I would also like to thank Simon Fraser University for providing the funding which made the construction of the proof-of-concept for the LaMASS system possible, through the Prototype Development Fund and the Wighton Endowment Fund. The wheelchair platform was kindly donated by Dr. Andrew Rawicz. As well, Dick and the rest of the guys in the Science Shop have always been helpful to me whenever I needed anything quickly. I would also like to thank Harry Boehm for helping me through the inevitable experimental blowups. Bolko Rawicz and Greg Hall also deserve a vote of thanks for working with me in the early stages of this project.

Last, but not least, I want to thank my lovely significant other Jennifer for always believing in me.

Table of Contents

List of Tables

List of Figures

Foreword

It seems that only in this past decade has the level of technology seen in assistive devices started catching up with the levels seen in products destined for the general consumer market. The reasons for these circumstances are many and variegated, including the ergonomic challenges in designing for the physically disabled, the small market size for such devices, and the general lack of data on the needs of the disabled. These circumstances have been changing slowly, with the growth in awareness of the rights of the disabled through such means as the Americans with Disabilities Act (ADA) and the prominence of centres for the study of assistive technology, including the Trace Centre at University of Wisconsin and the local Neil Squire Foundation, to name a few.

Materials technology originally designed for bicycles has been applied to manual wheelchairs to make lighter, faster, more usable conveyances for everyday living. The efforts of the Trace Centre to establish a standard protocol for serial communications among the various subcomponents in a powered wheelchair trail the establishment of such standards as CANBUS in the automobile industry. Computer technology not available a couple of decades ago has made it possible for people such as Stephen Hawking to make full and meaningful contributions to society.

Although the scope of this project is somewhat modest, it is my intention that this work serve as the basis for the development of a commercial system which can then be made widely available. It is my belief that commercialization of such a product is the best means by which this technology can be made widely available.

Chapter 1

Introduction

Independence is one of the key factors which contribute to quality of life. According to [1], mobility along with communication plays a critical role in maintaining independence, and the degree of mobility is directly related to one's level of independence. Powered wheelchairs are the primary means by which many physically disabled people extend the limits of their mobility both in and out of their home and work environments. These limits, though, are circumscribed by the range of places the wheelchairs can safely and comfortably take the user. Common physical limits to mobility include sidewalk curbs, stairs, and sloped roads. Although the wheelchairs themselves may be capable of negotiating the slopes and bumps, they may not be able to do so without causing mild to extreme discomfort to the user as the wheelchair pitches and rolls over the obstacles in its path. Powered wheelchair users more than most others are vulnerable to such motions, as they tend to have fairly little upper torso strength [2].

One solution, to immobilize the wheelchair users in their seats, is generally not prescribed by clinicians [3], as doing so can lead to medical or other complications. The

alternative is to strive for some means of keeping the user upright as much as possible. This thesis investigates the means by which an active levelling system might be designed in order to compensate for the tilt and acceleration of the wheelchair, and examines the results of one such design.

1.1 Motivation

As a powered wheelchair user drives along the road in a wheelchair, he or she will encounter disturbances in his or her path which will subject him or her to accelerations in six degrees of freedom (Figure 1.1), as the wheelchair irregularities in the road. Deviations in the road surface from absolute levelness relative to the local gravitational gradient such as slopes, ramps, curbs, and speed bumps all excite disturbances in the wheelchair's attitude and position.

Depending on the severity of the vibration, these accelerations may be enough to cause perceptions of mild discomfort to severe pain [4]. This is a common problem which has long been studied in the context of the automobile. The concept of introducing suspension to wheelchairs in the same manner as automobiles, in which a damping element is contained within a mechanical linkage between the vibration source (the ground) and the isolated platform (the user), is a fairly new concept for the wheelchair industry, with such suspension systems found widely available only within the past decade. A brief survey of commercially-available assistive products from 1991 [5] found only one wheelchair which claimed to have a suspension system. A search of the U.S. Patent database revealed very few patents [6, 7, 8, 9, 10, 11, 12] dealing with wheelchair

suspension systems. Most wheelchairs then and even now rely on the seat cushion to provide most of the shock absorption.

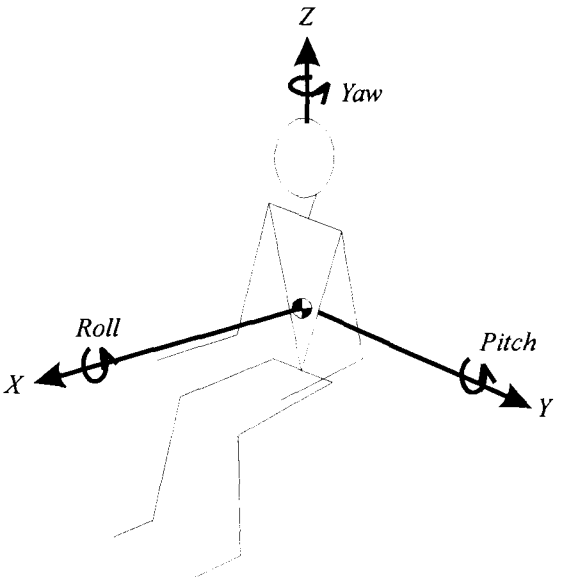


Figure 1.1: Human co-ordinate frame ($\mathscr{H}$), origin at seated person CoG

The reason for this state of affairs is that wheelchair designers have traditionally been more concerned with the prevention of pressure ulcers. Prolonged exposure of body tissues to concentrated pressure and shear, such as from sitting in a single position for too long on a hard surface, can cause pressure ulcers. Accordingly, much research has been conducted into the choosing of a seat cushion to distribute pressures evenly across the skin, to prevent pressure concentrations, and many different products which fill this need are commercially available [5, 13]. Unfortunately, an even pressure distribution does not necessarily imply vibration isolation as a design objective.

A second issue of importance for wheelchair users is the maintenance of proper posture [14]. Proper positioning in the seat is important for medical, psychological, and functional reasons. Maintaining a balanced, upright posture when travelling on a side

slope (causing a roll rotation) can be difficult for someone without adequate upper body strength, such as those who use powered wheelchairs. Unless some intervention is made, it can be easy for a powered wheelchair user's body to rotate out of proper posture if the slope is severe enough. To date, those wheelchairs which have addressed the problem of travel on a slope have done so by one of two methods: restraint via seat belts and tilt of the seat. Both methods, though, address primarily pitch rotations, as well as having other shortcomings which will be discussed later.

From anecdotal evidence, most wheelchair users will try to avoid areas which cannot be travelled comfortably, including slopes and bumpy terrain. Often, an appreciably large amount of circumnavigation is required to get to one's destination, if it is reachable at all. Such limits on travel have a great impact an individual's freedom of mobility and independence.

It is no surprise, therefore, that one of the respondents to a survey on powered wheelchair usage [15] (summarized in Appendix VI) replied at some length complaining about this very situation:

> I have a lot of trouble with balance on a side slope, and conventional body supports really get in the way. If the seat could keep itself level automatically at low chair speeds, it would be more than a help. I know a lot of quadriplegics have the same problems as I do, even when just travelling along a sidewalk that is tilted to the side.
>
> 38 year-old with spinal muscular dystrophy [15]

In a general sense, the type of system described above would be classified under the heading of self-levelling systems, a special class of vibration isolation systems. The difference between self-levelling systems and the shock absorbing suspension systems

found in most automobiles is that the goal of the former is to control displacement of the isolated platform relative to an inertial reference, whereas the latter uses as its reference the isolated platform itself. The differences between these two vibration isolation schemes is further discussed in section 2.3.

Ideally, such a self-levelling system for powered wheelchairs would provide at least 10°-20° of tilt in roll and pitch. For stair-climbing wheelchairs, as much as 45° of pitch might be necessary, both forwards and backwards. Scaled to the size of an automobile for comparison, such an angular displacement can easily be seen to be a very large motion.

The concept of tilting a platform to keep the driver of a vehicle upright over steep terrain is not a new one. For example, [16] describes a cross-country vehicle which uses a set of hydraulic cylinders to keep its cabin upright independent of the chassis. Active suspension for individual seats is also an area of increasing research. For example, [17] describes a mechanically-controlled hydraulic seat suspension system for use in tractors.

Logically, the ability to shift the driver relative to the wheelchair chassis should aid in maintaining tipping stability, by changing the vehicle geometry to compensate for the terrain. Indeed, wheelchairs such as [18] which allow for such a function are already in existence. Automatic compensation, however, has primarily been the domain of larger vehicles such as [19]. Other authors [20, 21, 22, 23] have suggested that such a shift can have large effects on the stability of the wheelchair, in the case of manual wheelchairs.

There are thus three issues related to the encounter of road disturbances:

1) small vibrational disturbances leading to discomfort or pain

2) pressure ulcers due to concentrated pressure at a single point for an extended time

3) postural changes from slumping of the body as the wheelchair's attitude changes

As both (1) and (2) have been largely addressed by available products, the motivation for this thesis is to investigate the means by which a self-levelling system might be built to address issue (3), examining the issues which are involved in such a design and presenting some results from a proof-of-concept. The next section presents an overview of existing systems.

1.2 Overview of Existing Systems

As stated previously, there are two main ways by which posture may be maintained for wheelchair users: restraints and tilting seats.

Restraints in wheelchairs are similar to seat belts commonly found in automobiles, whether of the two-point lap-belt variety, the three-point type with shoulder harness, or the full five-point web favoured by race car drivers. Of these, only the five-point web gives some measure of protection from roll rotations. Unfortunately, in doing so, this type of restraint also prevents the wheelchair user from easily shifting around to access his or her environment or to relieve pressure build-ups.

Alternatively, a form-fitting orthosis seat custom-moulded to an individual's body may be prescribed in certain cases. One example is manufactured by Summit Seating Systems (Skokie IL) shown below in Figure 1.2. Such seats perform even better than the five-point harness in maintaining an individual's posture, while the custom moulding helps distribute pressures evenly around the user's body. Such seating systems, though,

are generally only prescribed in the more extreme cases in which an individual has no other recourse to maintain posture, as prolonged use can lead to muscular atrophy and other associated problems, which in turn further aggravate the existing postural problems. As with overly-restrictive harness systems, restraints designed to maintain posture can, paradoxically, worsen the situation.

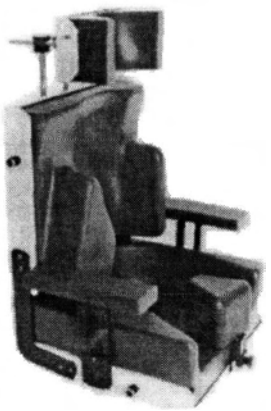


Figure 1.2: Summit Seating Systems orthosis seat [5]

Tilting systems, commonly referred to in the wheelchair industry as tilt-in-space seating systems, are found primarily as options on more expensive wheelchairs. The tilt angle is generally controlled manually, and the seat is usually tilted to either relieve pressure or to tilt the seat back before negotiating a slope or sidewalk curb. These tilt mechanisms handle only pitch rotations.

Tilt-in-space seating systems are to be distinguished from reclining seating systems, which simply tilt the seat back down for pressure relief. More advanced recliners utilize a kinematic linkage to reduce the shear forces on the person's back and bottom as the seat back reclines [1]. Such reclining systems, though, are generally used more for pressure relief with the wheelchair in a stationary position than as a means of maintaining balance.

One example of a wheelchair with a tilt-in-space system is the Zippie P500 by Quickie Designs Inc. (Fresno, CA), which pivots about an axis located at the rear bottom corner of the seat (Figure 1.3). As will be demonstrated in Chapter 3, this can lead to problems with tipping stability of the loaded wheelchair while in the tilted position.

Figure 1.3: Zippie P500 tilt-in-space chair by Quickie. [24]

Examples of wheelchairs which attempt to circumvent this problem include the Permobil Chairman (Timrå, Sweden) shown in Figure 1.4 and the Invacare Tarsys seating system (Cleveland, OH) shown in Figure 1.5. In the case of the Chairman, a mechanism is used to lift the seat back up over the chassis as it tilts back, in order to keep the center of gravity (CoG) between the wheels. Similarly, the Tarsys seating system slides the seat back and forth on a rail as the seat tilts back. Of the two systems, the Tarsys is more flexible, as it allows for two degrees of freedom of independently-controllable motion.

Having the ability to shift the CoG forwards and backwards while tilting the seat is important to maximizing tipping stability while travelling up and down slopes, resulting in either more stability for a given chassis frame size or a need for a smaller wheelbase.

Figure 1.4: Permobil Chairman powered wheelchair with Multi-Positioning Seat [25]

Figure 1.5: Invacare weight-shifting Tarsys tilt-in-space seat in tilted position. [26]

Both the Invacare Tarsys and the Quest ACCESS seat tilt systems are two degree-of-freedom mechanisms consisting of a rotational joint mounted on a sliding platform. Although both of these mechanisms perform their functions admirably, it would be preferable to not have a translational linkage in the mechanism, as these tend to have more problems with wear and reliability than rotational elements, especially in the harsh environment being part of a wheelchair would subject them to, including weather, humidity, and temperature extremes. Additionally, neither system is designed with a high duty cycle in mind, which allows for the use of the long guideways which run almost the entire length of the wheelchairs. In practice, tilting with both systems is only

an infrequent event, certainly not with the same duty cycle as, for example, the main drive motors run.

There have also been a few stair-climbing wheelchairs on the market such as the Quest Technologies Corporation's (Sunnyvale, CA) ACCESS wheelchair, shown in Figure 1.6, in which the seat pitches forwards and backwards automatically as the wheelchair climbs up and down stairs and other slopes. With a price tag of $25,000 US in 1992 prices, this wheelchair is far beyond the means of the vast bulk of the population of powered wheelchair users. And, although the stair-climbing facility is undoubtedly very useful, an increasing number of places in North America are becoming wheelchair accessible.

Figure 1.6: Quest ACCESS stair-climbing wheelchair with tiltable seat. [5]

1.3 Contributions

There is a need for a self-levelling suspension system which compensates for both roll and pitch and can fit onto an existing conventional powered wheelchair. In order to design such a system, it is necessary to first build up expertise in designing devices for

use in wheelchairs and characterize the behaviour of wheelchairs as they travel on the road. This information can then be used to design a suspension system.

One of the goals of this work was to make the system economical to produce, which would lower the potential sales price and thus make it more readily available to a wider population than other like efforts in the past. By concentrating on a single component in the wheelchair as opposed to trying to redesign the entire wheelchair, the chances of a successful product are increased.

The approach taken was to use modern, consumer-grade inertial sensors to measure the absolute motion parameters of the wheelchair chassis. This information, fed through a set of economical high-speed microcontrollers, would then be used to control the attitude of a seat mock-up. By concentrating on a 1-dof design, results derived from this experiment could be used to help design a similar controller for a commercial 1-dof tilt-in-space seating system using the same set of inertial sensors and microcontrollers.

The LaMASS proof-of-concept device, mounted on a conventional powered wheelchair chassis, provided a testbed to test the efficacy of using an accelerometer pair, a piezoelectric vibrating gyroscope, and a liquid level tilt sensor, singly and in combination, in order to control the attitude and position of a seat mock-up. The position and attitude were determined by a closed kinematic chain 4-bar mechanism optimized using a genetic algorithm method. To control the stepper motor which drives the mechanism, a fuzzy-logic computationally-inexpensive algorithm was used to limit the step rate according to the required torque at different points in the path of the motion. The main contribution of this thesis is the inertial sensing controller system designed using economical, consumer-grade parts.

1.4 Thesis Organization

This thesis presents the work on the Large Motion Active Suspension System as a study on the design of an active suspension system for use in the powered wheelchair application, and explores the issues involved in such a design as well as design techniques which would allow such a device to go to production in a timely and economic manner.

Chapter 2 presents a brief background on selected physiological aspects dealing with wheelchair use, as well as a lexicon of terms and concepts dealing with wheelchairs, in order to present a context for the design. Chapter 3 develops and presents some of the motivating theory behind self-levelling suspension systems for powered wheelchairs. Chapter 4 examines the design process of the suspension mechanism, including using a genetic algorithm technique for optimization of the design parameters within the design constraints. Chapter 5 details the equipment which was built in order to realize the design. Chapter 6 discusses some of the results from the design and relates these results to their impact on a potential production model. Chapter 7 begins with the conclusion, and then discusses future work to be done in order to further this work.

Chapter 2

Background

This chapter introduces some terms and concepts which are specific to the rehabilitative engineering field, especially that concerning powered wheelchairs and their use. As well, some background material concerning vibration isolation systems is presented.

2.1 Wheelchair Mechanics: Terms and Concepts

Powered wheelchairs (also referred to as simply power wheelchairs by some authors) can be thought of as small vehicles in which the driver sits (Figure 2.1). Virtually all powered wheelchairs run on electric power. Like an automobile, the powered wheelchair possesses three or four wheels, motors, and a battery pack. The driver steers the wheelchair through a control interface commonly mounted either on the armrest or headrest, with a top speed which is usually less than around 15-16 km/h, depending on the manual dexterity and cognitive abilities of the driver. Unlike the common car, though, the driver makes up a significant fraction of the total loaded vehicle

weight. An adult might weigh 65 kg or more. Together with the seat (~18 kg) which the driver is usually belted into, about half of the total weight of a typical loaded powered wheelchair (occupant plus wheelchair) is composed of or directly attached to the driver. This difference is enough to justify the development of new equations describing wheelchair behaviour, as a subset of general vehicle dynamics, even though many of the results developed for larger vehicles apply to wheelchairs with little modification.

For ergonomic reasons, the driver must be kept a reasonable height off the ground to allow for easier navigation and improved self-esteem [14]. Raising the driver raises the person's centre of gravity (CoG) and subsequently the CoG of the loaded wheelchair. At the same time, the footprint of the wheelchair must be kept small enough to allow for easy manoeuvrability (e.g., egress and ingress through doorways). The result is a compromise among size, manoeuvrability, and structural stability.

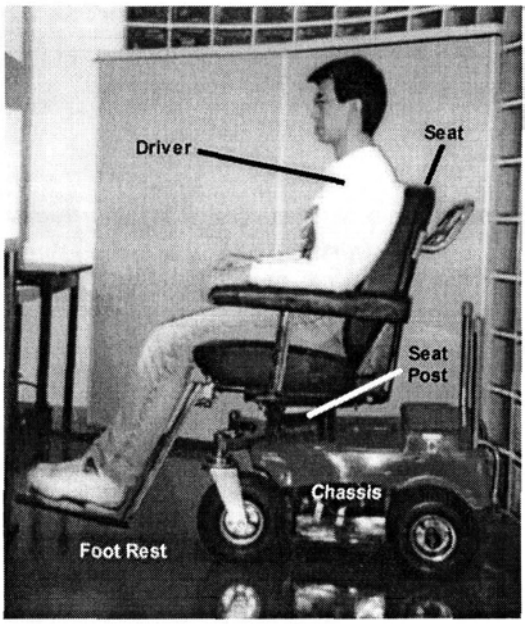


Figure 2.1: Powered wheelchair nomenclature

2.1.1 Wheelchair Footprint

The footprint of a wheelchair, as defined in [27], is the area formed by the perpendicular projections of the points of contact of the wheels onto a level floor beneath the wheelchair, as shown in Figure 2.2. As is further expounded upon in section 3.2, a larger footprint leads to greater tipping stability which, loosely defined, is a measure of the wheelchair's ability to resist tipping. Static tipping stability is maintained so long as the projection of the location of the center of gravity of the wheelchair onto the level floor remains within the footprint.

One consequence of linearity in projection is that the geometric center of the footprint is the projection of the geometric center of the area formed in plane with the four wheels. This implies that a line connecting the center of the footprint with a point directly above it will pass through this other center point, the origin of the $\mathcal{G}$ frame defined in the next section.

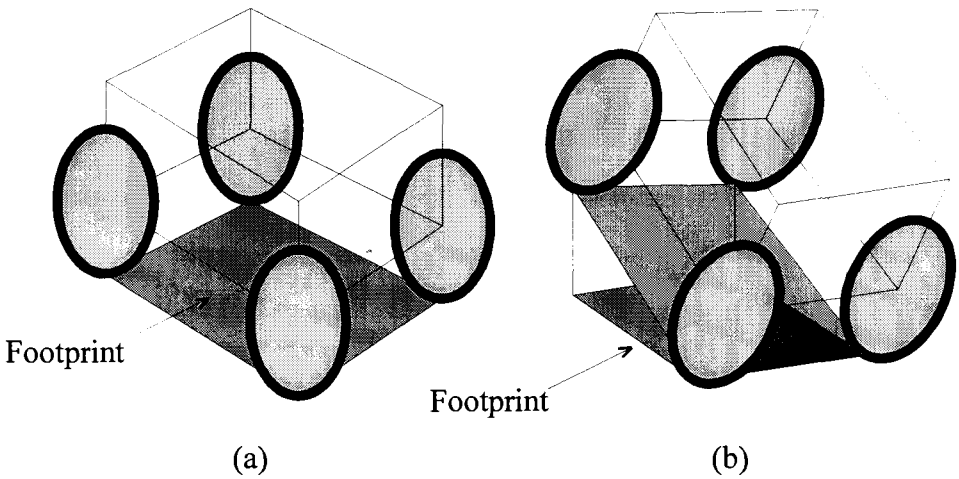


Figure 2.2: Wheelchair footprint for level (a) and tilted (b) surfaces

2.1.2 Co-ordinate Systems

Most of the mechanism analysis performed in this thesis takes place with respect to the wheelchair co-ordinate frame ($\mathcal{G}$), which is rigidly affixed to the chassis, and has as its origin the point equidistant from the contact point of each of the four wheels with level ground and lying in plane with all four of these points, as shown in Figure 2.3. The positive X-axis points forward, positive Y points to the left, and positive Z points straight up, perpendicular to the plane formed by the contact points of the four wheels, the assumption being that these four contact points are indeed coplanar. With a person sitting directly upright on the wheelchair's seat, $\mathcal{H}$ from Figure 1.1 will be parallel to $\mathcal{G}$. Note that $\mathcal{G}$ is not necessarily an inertial frame, as it can accelerate relative to $\mathcal{W}$, the inertial world co-ordinate frame.

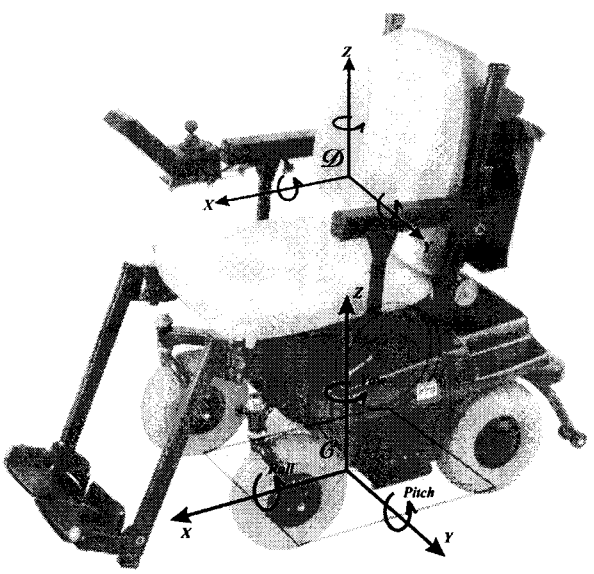


Figure 2.3: Wheelchair coordinate frame ($\mathcal{G}$) and seat-driver coordinate frame ($\mathcal{D}$)

$\mathcal{D}$, the seat-driver co-ordinate frame, has its origin located at the center-of-gravity of the combined system consisting of the human driver sitting in the seat with footrests

(essentially, everything except the chassis). The axes of $\mathscr{D}$ are defined to be parallel with those of $\mathscr{G}$ when the seat is located at the neutral position, where the seat is level with respect to the chassis, as with conventional fixed-seat wheelchairs. In practice, the origin of $\mathscr{D}$ will be located slightly below that of $\mathscr{H}$, as the mass of the seat and footrests, located below the driver, pulls the seat-driver system CoG downwards. Furthermore, movement of the driver within the seat will cause the origin of $\mathscr{D}$ to move as well. The vector which describes the origin of $\mathscr{G}$ relative to $\mathscr{D}$ is $\vec{R}_{\mathscr{G}\mathscr{D}}$ (not shown in the diagram for reasons of clarity).

2.1.3 Types of Powered Wheelchairs

The wheelchair shown in Figure 2.1 above is a powered wheelchair manufactured by Fortress, and is typical of the power-base style of wheelchair which has become popular over the past decade. This style is characterized by a chassis, also known as a power base, with small wheels with a modular seat mounted on top. The seat interfaces with the chassis at the seat post via a standard mechanical interface. Typically, a variety of seating systems are available and easily interchangeable with one another to allow the wheelchair to be customized to the individual users' needs.

The other main style of powered wheelchair is the dual manual/powered style, such as the Fortress Commuter powered wheelchair shown in Figure 2.4 below, which is similar in structure to the traditional manual wheelchair, with the addition of a joystick control and a set of motors. Keeping the large rear wheel with handrims in this design allows users to manually propel themselves. For this type of wheelchair, the seat is often an integral part of the frame, not easily exchanged with seating systems from other

manufacturers, if at all. This second type of wheelchair is typically prescribed only for those with some upper body movement, generally as a means to extend the available travelling range for a given individual.

Figure 2.4: Fortress Commuter wheelchair, dual manual/power drive. [28]

The third major type of powered wheelchair is more commonly referred to as a scooter. Electric scooters can have either four wheels or three, such as the Fortress 2001LX shown in Figure 2.5 below. These scooters are designed for use by marginal walkers in order to extend their range of travel outside the home, and as such are designed to be easy to get in and out of as well as to not look like conventional wheelchairs [1]. Included in this group are people such as the elderly, who make up a sizeable portion of the mobility disabled community. [29]

Figure 2.5: Fortress 2001LX electric scooter. [28]

2.2 Physiology of Mobility Disability

The 1991 Health and Activity Limitation Survey (HALS) conducted by Statistics Canada [30] defines a mobility disability as a disability which limits a person's ability to move around in performing his or her daily tasks. Similar definitions hold for the other four major classes listed on the HALS: agility, hearing, seeing, and speaking. According to the survey, 10.8% of Canadians over the age of 15 years have some form of mobility disability, or some 2.9 million people. Of these, perhaps 10% might have a mobility disability severe enough to warrant the prescription of a power wheelchair of some type.

[1] suggests that powered wheelchairs can serve in two capacities: as either primary transportation for the severely mobility disabled or as a means to extend the range and functionality of marginal ambulators who can use a manual wheelchair but are greatly restricted in rate and distance due to lack of energy, such as those with mild quadriplegia. Because mobility is so central to our daily lives, the wheelchair will form the primary environment for most wheelchair users for much of each day. Without moving from the chair, it is more likely that pressure ulcers will form.

Pressure ulcers are also known as decubitus ulcers, ischemic ulcers, or bed sores. Pressure ulcers are characterized by breakdown in the skin and neighbouring tissues, primarily as a result of localized pressure and shear concentrations [31]. These pressure ulcers may lead to infections or other medical complications, and are thus to be avoided if at all possible. Decubitus is generally not observed in able-bodied people, as they unconsciously move around to relieve local pressure concentrations, even while in an

apparently still seated position. For most wheelchair users, such motions must be made consciously, if such motions are possible at all.

Loss of upright posture can have negative consequences for the driver ranging from discomfort to loss of control over the wheelchair [32]. People who use powered wheelchairs do so because they do not have the requisite upper body strength or endurance to use a manual wheelchair, and hence may not to be able to bring their bodies back to an upright position if their own position in their seats changes as a result of an attitudinal change in the wheelchair chassis. [33] Additionally, although most wheelchair drivers will have seatbelts and hence are in little danger of sliding out of their seats, a tilted seat can result in uncomfortable and potentially physiologically harmful shear forces at the bearing skin surfaces. In fact, devices such as [34] have been designed specifically to combat such debilitating effects.

2.3 Effects of Vibration on the Human Body

If one treats the human body as a viscoelastic mechanical system, one finds that the various parts of the body have different resonance frequencies, summarized in For example, a human subject can feel pain and discomfort in the abdominal region as a result of resonance vibrations at 4-12 Hz. Because of where the resonance frequencies lie, most of the harmful effects of vibrations acting on human subjects, from motion disease to difficulty breathing and deterioration of hand-eye co-ordination, takes place in the frequency range between 1 and 10-15 Hz. For example, at 2 Hz, a vibration displacement of 40 mm (acceleration displacement of 6.3 m/s^2) would be considered

unpleasant at long duration, while a vibration displacement of 200 mm (corresponding to 31.6 m/s^2) would be considered unpleasant for a short duration.

Table 2.1 below, based on data from [4]. As all of these frequency ranges overlap at least partially within the vibrational frequency spectra experienced by a vehicle moving at wheelchair speeds, the vibrations felt by the wheelchair user is of no little concern.

For example, a human subject can feel pain and discomfort in the abdominal region as a result of resonance vibrations at 4-12 Hz. Because of where the resonance frequencies lie, most of the harmful effects of vibrations acting on human subjects, from motion disease to difficulty breathing and deterioration of hand-eye co-ordination, takes place in the frequency range between 1 and 10-15 Hz. For example, at 2 Hz, a vibration displacement of 40 mm (acceleration displacement of 6.3 m/s^2) would be considered unpleasant at long duration, while a vibration displacement of 200 mm (corresponding to 31.6 m/s^2) would be considered unpleasant for a short duration.

Table 2.1: Resonance frequencies of the human body

Body Part	Resonance Frequency
Eyes	12-17 Hz
Throat	6-27 Hz
Chest	2-12 Hz
Feet, Hands	2-8 Hz
Head	8-27 Hz
Face and Jaws	4-27 Hz
Lumbar part of spine	4-14 Hz
Abdomen	4-12 Hz

2.4 Types of Suspension Systems

The task of a vibration isolation system is to isolate an object from an external vibrational excitation input. The term "suspension system" has come to be associated with vibration isolation systems employed in an automobile from the method by which the vehicle body is suspended from the vehicle's wheels, although not all vehicles necessarily have this type of arrangement. In this thesis, "suspension system" and "vibration isolation system" will be used interchangeably. In general, the vibration isolation element will lie between the source of the excitation force and the isolated platform. In a vehicle, the source of the excitation force is the road, as shown in Figure 2.6.

As the vehicle moves along the road, any unevenness in the road surface can be interpreted as the road moving upwards with a velocity V_0 relative to an inertial reference

frame. In so doing, the road exerts a force $\vec{F}_E$ on the isolation element through the wheel which contacts the ground. The vibration isolation element exerts a force on the vehicle, with a resultant transmitted force, $\vec{F}_T$, which causes the platform to move with a velocity V_P. Ideally, the transmitted force $\vec{F}_T = 0$ when the force exerted by the vibration isolation element exactly cancels out the excitation force $\vec{F}_E$.

There are many variations on what exists inside the box labelled "isolation element" in the figure, too numerous to list here. Instead, this section concentrates mainly on the classes of isolation systems.

Vehicle suspension systems are usually categorized according to the amount of energy which is input into the system in order to effect vibration isolation, or the lack thereof. The traditional automobile suspension system is a passive system which utilizes only passive, energy-dissipating elements. In the past 10-15 years, much interest has arisen into the properties of active suspension systems, which use active elements such as motors to provide the damping force. Semi-active suspension systems use an active controller to control the damping properties of what would otherwise be a regular passive suspension. The next sections describe and contrast these types of vibration isolation systems.

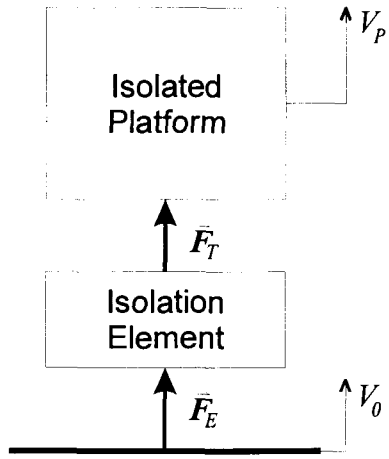


Figure 2.6: Generic vibration isolation system

2.4.1 Passive Suspension

The familiar passive shock absorber suspension system is found on most automobiles. In these passive vibration isolation systems, the isolation element consists of a spring in parallel with a damping element. Both the spring and damping elements may or may not be linear. One of the main purposes of this arrangement in automobiles is to damp out unwanted road harshness. A second important function is to promote better road handling through such tactics as equalization of weight distribution on the tires and control of body attitude. As [35] indicates, there is a trade-off involved between these functions. This trade-off is discussed further in section 3.3.

Passive suspension systems are the most common type found in vehicles, as they can be manufactured inexpensively, tend to be fairly lightweight, and require no external sources of energy to operate. Common passive suspension systems employ viscous fluid, metallic springs, or a combination of both as the vibration isolation element. The elements' parameters do not, as a rule, change with time, although they may individually

have non-linear response characteristics. As vibration isolation elements, these passive damping elements serve as low-pass filters.

For example, for a linear spring and damper in parallel as the isolation element, we get the following classic second-order Laplace transfer function, the plot of which is shown below in Figure 2.7.

$$\frac{V}{V_0} = \frac{2\zeta\omega_n s + \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{2.1}$$

where ζ is the damping ratio and ω_n is the natural frequency.

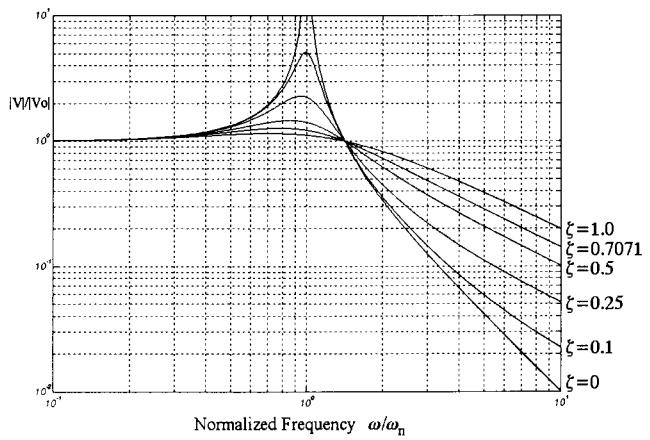


Figure 2.7: Normalized second-order logarithmic frequency response characteristics

2.4.2 Active Suspension

In a fully active suspension system, the vibration isolation element of Figure 2.6 is replaced by some form of actuator which exerts forces on the isolated platform in order to achieve some desired optimum response. For example, according to [35], the optimal response for the second-order active isolator corresponding to equation (2.1) is:

$$\frac{V}{V_0} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{2.2}$$

Equation (2.2) results in a response in which increasing the damping ratio does not affect high-frequency isolation (Figure 2.8). This response is not achievable using any combination of parameters for the passive isolation system described in the previous section.

Active suspension systems allow for protection of objects from low-frequency vibrational excitations outside the range of existing passive vibration isolation systems or from vibrational inputs with time-varying characteristics, such as those found in roads. In particular, the frequency characteristics of the vibratory excitation due to road unevenness will change as the vehicle's velocity changes. A good approximation has the road's frequency spectra expanding in the frequency domain to cover higher frequencies as the vehicle's velocity doubles. (If we consider the vehicle to be standing still and the roadway to be moving along underneath causing excitations with some function f(t), then doubling the speed of the vehicle causes the excitations to change to f(2t).)

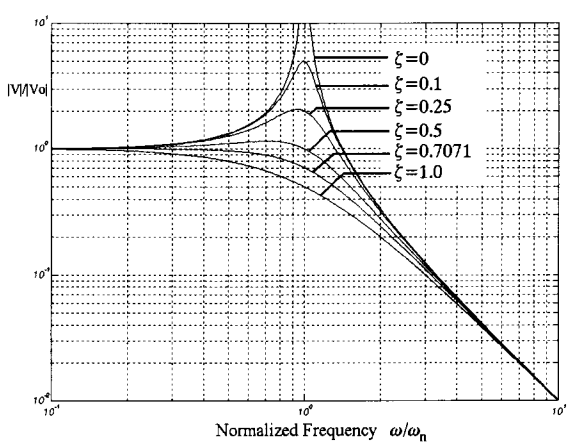


Figure 2.8: Normalized frequency response of an optimal active vibration isolator

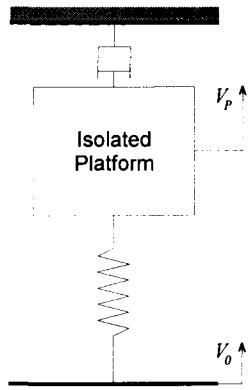


Figure 2.9: Skyhook damper system

With an active suspension system, it is possible to use vehicle velocity relative to the ground or other data measured relative to an inertial reference as additional inputs to the controller and change the frequency characteristic of the vibration isolation system in an adaptive way. Using an inertial reference allows for the creation of what has been dubbed a "skyhook" damper by [35], as shown in Figure 2.9 above. The skyhook damper arrangement has superior frequency characteristics as compared to the topology shown in Figure 2.6.

Two of the active isolation topologies described in [4] are disturbance compensation with feed forward (Figure 2.10) and deviation compensation (Figure 2.11). There are, of course, literally hundreds of different topologies possible which have been documented in the literature. In the case of disturbance compensation with feed forward, a disturbance, Z_0, enters the servo system C, which compensates according to some control law K based on information from sensor D_{vib}. The resulting motion X is then transferred to the isolated platform.

In the case of deviation compensation, the disturbance Z_0 acts directly on the isolated platform. However, the sensor D_0 picks up the current motion X of the platform. When

compared to a reference value Y and passed through a controller P, a control signal U_0 is generated to compensate for the disturbance.

The disadvantage of using fully active suspension systems is their cost in terms of power, something which is usually in limited supply on a moving vehicle. In order to effect vibration isolation, power must constantly be supplied to the actuation elements (C in Figure 2.10 or P in Figure 2.11).

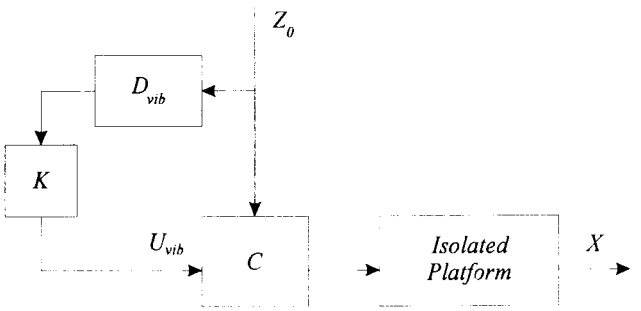


Figure 2.10: Active isolation via disturbance compensation with feed forward

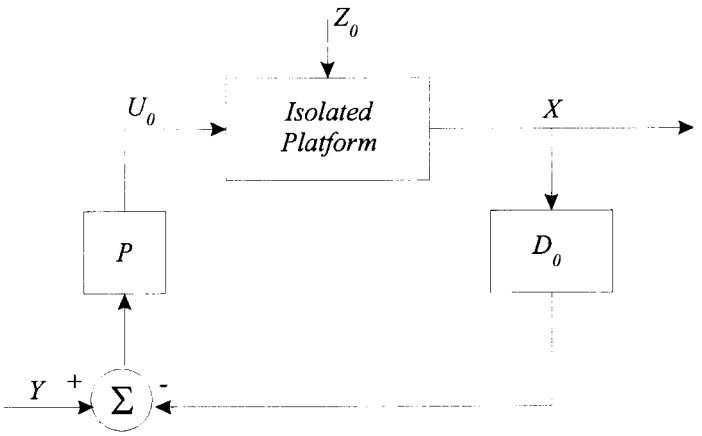


Figure 2.11: Active isolation via deviation compensation

We can estimate the power required in an ad hoc manner by considering the simple case of a rotational actuator which must isolate a 51 kg person and 20 kg seat from a rotational vibration input of 1.2 Hz and amplitude 20° applied at the base of the wheelchair, at the origin of the wheelchair chassis frame $\mathcal{C}$. This combination is an

estimate based on an average speed of 9 mph, or 4 m/s, running up a standard 20° sidewalk curb cut. For this 51 kg person, calculations based on [36] suggest a moment of inertia of approximately 4 kg·m^2 about the person's center of mass, depending on the exact configuration of the body. Adding the seat's moment of inertia results in an estimate of I_{cm}=6 kg·m^2. For the rotation about the origin of $\mathcal{O}$, the seat-driver CoG at the origin of $\mathcal{D}$ will have a moment arm of 0.7 m. Applying the parallel axis theorem results in an effective moment of inertia of I=40 kg·m^2. To calculate power, we will assume that a compensation rotation is made about the origin of $\mathcal{O}$.

The power P is equal to the product $\tau\omega$, the product of the torque exerted and the angular velocity. The torque τ is also equal to the product $I\cdot\alpha$, where α is the angular acceleration. Thus, the instantaneous power will be equal to $P=I\cdot\alpha\cdot\omega$. Assuming pure sinusoidal motion at 1.2 Hz, peak power will be generated at

$$P_{peak} = \tfrac{1}{2} I \alpha_{peak} \omega_{peak} \tag{2.3}$$

The result is a peak power of $P_{peak} \sim$ 1.1 kW. By contrast, a typical wheelchair's motors consume a total of around 20-30A at 24V, for a power consumption of 0.6-0.7 kW. Naturally, moving at slower speeds (as one would probably do in heading up a curb cut) would require a smaller consumption of power. Nonetheless, it is clear that using a fully active suspension system on a powered wheelchair would more than double the power consumption, drastically cutting down on vital running time for the wheelchair. The situation is even worse when one considers that most real-world mechanical systems do not have energy efficiencies even close to 100%, thus requiring a power input of between 5 to 10 times P_{peak} in order to accomplish the stated motion, assuming a typical electric motor drive.

2.4.3 Semi-Active Suspension

A variation on the active suspension is the semi-active suspension, which uses an active sensing and control element to control the damping properties of a passive suspension system. Using such a scheme allows the vehicle to retain many of the benefits of the fully active suspension system such as adaptivity to time-varying or random input disturbances without the large power drain required by active suspensions. For example, as the vehicle speeds up and the peak input disturbance frequency increases far past the vibration isolator's natural frequency, it may be desirable to decrease the system's damping ratio, increasing resonant response but decreasing the passive damper's high frequency response. At low velocities, or with road vibration frequencies in the neighbourhood of the system resonant frequency, one could then increase the damping ratio, sacrificing some high frequency response for increased damping. Alternatively, one may want to have a stiffer system, corresponding to a system with higher damping, at higher road velocities in order to improve road handling.

2.4.4 Self-Levelling Systems

One subclass of active vibration isolation systems consists of self-levelling systems. While other vibration isolation systems respond to an excitatory force, the goal of a self-levelling system is to respond to a change in displacement. For example, [37] describes the self-levelling suspension system for a Citroen automobile in which a pair of hydraulic accumulators senses a change in level of one of the vehicle's wheels and uses that information to change the amount of fluid in a piston, moving the wheel.

Chapter 3

Theory and Modelling

It is important, in attempting to keep the driver level with LaMASS, that the tipping stability of the wheelchair, defined as the susceptibility of the wheelchair to tipping, not be compromised. As with medical instruments and other devices which come in direct contact with a live person, the standards for safety are very high. In order to ensure the safety of the device, it is necessary to understand its behaviour under various conditions. This chapter develops some of the underlying physical equations which describe the motion of a self-levelling system in the context of LaMASS.

3.1 Modelling

Data gathered by [38] suggests that tipping is one of the biggest concerns for wheelchair users in motion. As defined in the ISO standards on wheelchair stability [39,40], tipping stability is associated with the loss of driver control over the wheelchair's behaviour inherent in the lifting of one or more wheels, as it is through the wheels of the vehicle that the wheelchair is steered and driven.

Knowing when one or more wheels lose contact with the ground is important because the wheels are necessary for control of the direction of the wheelchair's direction, especially in turning on a slope, as pointed out in [41]. Furthermore, the loss of even one contact point out of the four which most powered wheelchairs have can cut down the tipping stability drastically by reducing the effective footprint covered by the wheelchair, as [27] has noted.

In this and the next section, some of the conditions which lead to tipping, as well as some simplified models which can be used to predict tipping, will be developed. The primary model which will be used is a two-dimensional one where a 65 kg human is sitting on a Fortress powered wheelchair. In the single-mass two-dimensional model (Figure 3.1) the driver and wheelchair are modelled together as one lump mass, with a certain mass and moment of inertia. The masses of the various portions of the wheelchair were derived from experimental measurements. The mass distribution model of the human was supplied with the two-dimensional mechanical simulation software [42], which is substantially the same as [36].

Because the criteria used to evaluate tipping stability deal with motion and structural stability in a single direction at a time, the models presented here are fairly simple two-dimensional ones. Assuming decoupling of pitch (forward/back rotation) and roll (rotation left/right), these two-dimensional models may be applied to certain three-dimensional cases, such as the ISO tip stability testing procedures [39, 40].

In the static stability test [39], a fully-loaded wheelchair (including a test dummy as the driver) is placed upon a tiltable platform. The wheelchair is gently tilted in several different directions and the stability limit recorded for each. The stability limit for a

particular configuration (e.g., locked brakes, wheelchair facing downhill) is defined as the tilt angle at which one or more wheels of the wheelchair slide along the platform due to insufficient friction between the wheel and the test plane or where the wheelchair tips over.

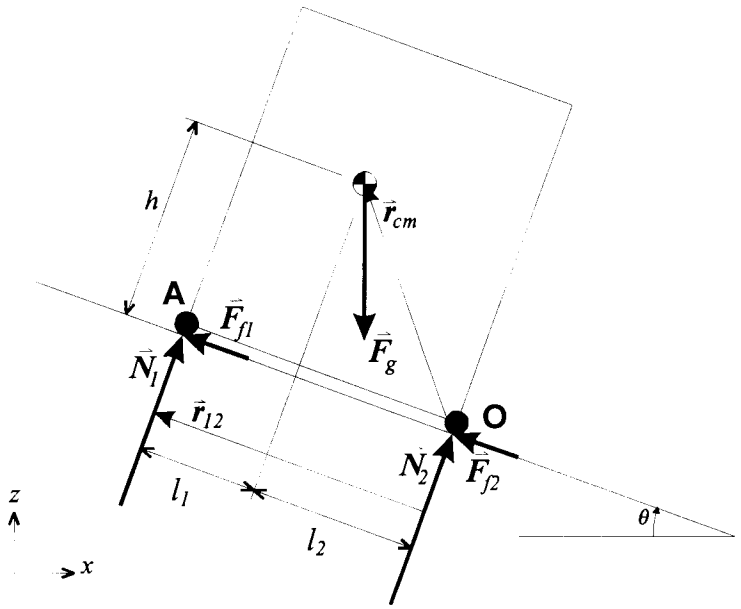


Figure 3.1: Single-mass wheelchair model

The dynamic stability test [40] is similar, with the exception being that the wheelchair is accelerated and braked going up and down, respectively, a tilted surface. The stability limit is again defined by the tilt angle at which the wheels are observed to lift off the surface in braking or accelerating or where the wheelchair tips over.

The loaded wheelchair can be modelled as a single mass M, with a moment of inertia I_{cm} about its centre of mass. The forces acting on M are the gravitational force $\vec{F}_g = M\vec{g}$, the surface normal reaction forces $\vec{N}_1$ and $\vec{N}_2$, and the frictional forces $\vec{F}_{f1}$ and $\vec{F}_{f2}$ with coefficients of friction μ_1 and μ_2.

The displacement vector $\vec{r}_{cm}$ to the centre of mass from point **O** is given by the complex vector:

$$\vec{r}_{cm} = (-l_1 + h)e^{i\theta} \tag{3.1}$$

while the vector $\vec{r}_{12}$ from the downhill contact point to the uphill contact is similarly given by:

$$\vec{r}_{12} = -(l_1 + l_2)e^{i\theta} \tag{3.2}$$

The model shown in Figure 3.1 serves equally well for analysis of transverse (side-slope) and longitudinal (down-slope) problems. In this model, there are only two points of contact with the slope at **O** and **A**, each corresponding to one pair of wheels. (e.g., in a longitudinal problem, the points of contact are the front and back pairs of wheels.)

We can disregard any moment of inertia of the wheels, as the wheels tend to be quite small on powered wheelchairs and turn relatively slowly, and hence possess a negligible moment as compared with the rest of the system.

An extension of the single-mass model separates the driver and seat from the chassis (Figure 3.2). In many powered wheelchairs, the seat is customized or capable of being customized to suit the needs of the user, separate from the chassis which contains the motors, wheels, and batteries. It is a practical approach to insert a mechanism between the seat and the chassis in order to position the seat and driver for maximal tipping stability.

$$\begin{aligned} \vec{r}_{cm} &= \frac{m_c}{M}\vec{r}_c + \frac{m_d}{M}(\vec{r}_c + \vec{r}_{cd}) \\ &= \vec{r}_c + \frac{m_d}{M}\vec{r}_{cd} \end{aligned} \tag{3.3}$$

which gives us the location of the centre of mass for the combined system $\vec{r}_{cm}$, where $M = m_c + m_d$, the sum of the masses of the chassis (m_c) and the seat-driver system (m_d).

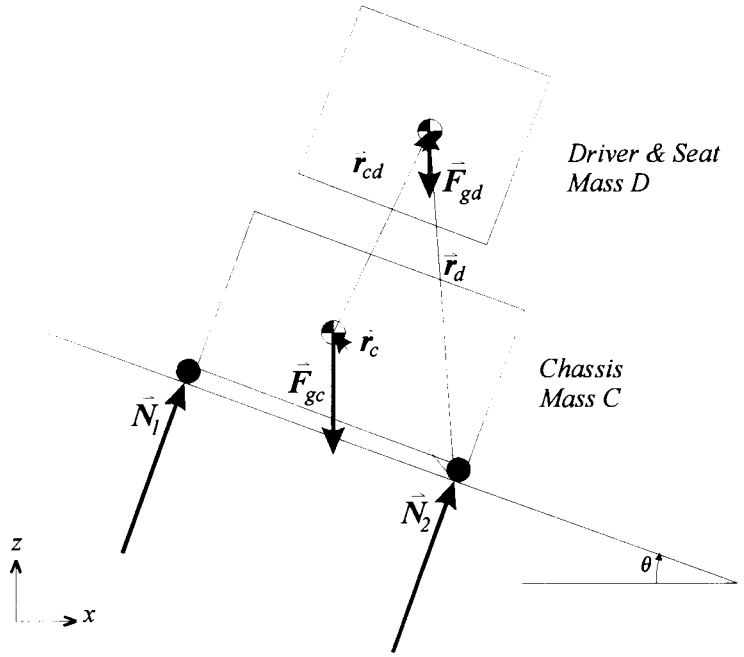


Figure 3.2: Two-mass wheelchair model

Consider the case of a wheelchair accelerating to the left on a level plane, as shown in Figure 3.3. A reference frame attached to the wheelchair would be non-inertial due to the acceleration. An inertial frame can be constructed by the addition of a fictitious external force $\vec{F}_O = -m\vec{a}$ acting in the opposite direction to the acceleration. The resulting configuration, pictured on the right side of Figure 3.3, is analogous to the static tilted models with a slightly larger net force $|\vec{F}_R| = |\vec{F}_O + M\vec{g}|$ replacing the gravitational force $M\vec{g}$ from the model of Figure 3.1. The angle $\theta_a = \theta_0$, the tilt angle in the single-force model. It is not difficult to see by inspection that acceleration on a slope θ can be modelled with a mass on an incline with an angle equal to the sum of θ_a and θ. Thus,

the analysis of the wheelchair on an incline presented here can be applied to more general situations.

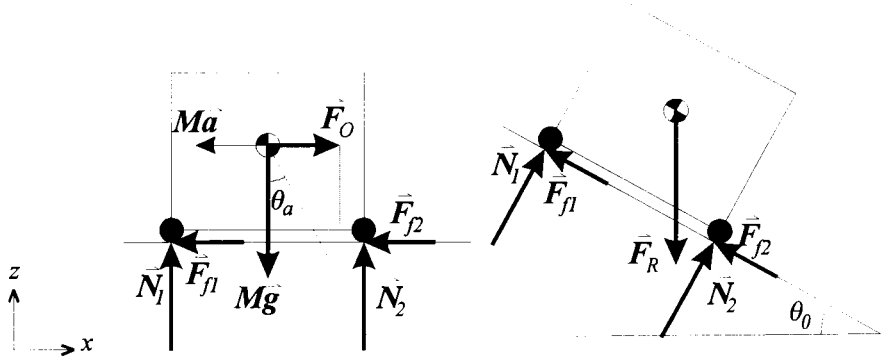


Figure 3.3: Accelerating noninertial vs. tilted inertial frames

As an example, we can obtain the side tipping, or roll, stability, by considering the addition of centripetal acceleration to the wheelchair. To model this acceleration using the static model, we can add a centrifugal pseudo-force $\vec{F}_O$ in the x-direction to the gravitational force pulling down on the system, in similar fashion to [27], and considering this system to be quasi-static. This new set-up can then be used to calculate roll stability. In a three-dimensional model which would extrapolate from this work, it is essential that consideration be taken of pitch as well as roll stability.

3.2 Static Tipping Stability

Although the tipping stability of the wheelchair is measured by the angle of tilt required to actually tip the wheelchair, it can also be applied to more general situations as a margin to unrecoverable tipping. Recoverable tipping of the wheelchair ends up with the wheelchair returning to its original position and orientation, while unrecoverable tipping leads to rather catastrophic results.

3.2.1 Static Tipping Criterion

As a first step in calculating the tipping stability limits of the model, in a derivation similar to [20], consider the sums of forces and moments in the model shown in Figure 3.1. In the static situation, these sums will be identical to 0. Assuming that $l_1 < l = l_1 + l_2$ (l, the distance between the two contact points, is fixed for a given system configuration.) and that $\theta \geq 0$, the point of rotation will be about the downhill contact point **O**. $\vec{N}_2$, $\vec{F}_{f1}$, and $\vec{F}_{f2}$ act through **O**, and hence do not contribute a moment in equation (3.5).

$$0 = \vec{N}_1 + \vec{N}_2 - M\vec{g} \tag{3.4}$$

$$0 = (\vec{r}_{cm} \times M\vec{g}) + (\vec{r}_{12} \times \vec{N}_1) \tag{3.5}$$

Equation (3.4) can be rewritten in terms of its components. Specifically, we look at the component of force along the direction normal to the ramp surface:

$$0 = \left|\vec{N}_1\right| + \left|\vec{N}_2\right| - \left|M\vec{g}\right| \cos\theta \tag{3.6}$$

Equation (3.5) will have a z-component only, perpendicular to the two-dimensional model plane:

$$0 = Mg(-l_1 \cos\theta + h \sin\theta) - (l_1 + l_2)\left|\vec{N}_1\right| \tag{3.7}$$

At the limit of stability, $\vec{N}_2 = \vec{0}$ as point **A** just lifts away from the ground, so equation (3.7) simplifies into:

$$\tan\theta_{crit} = \frac{l_1}{h} \tag{3.8}$$

Equation (3.8) gives the tipping stability limit, the angle θ_{crit} at which the wheelchair first starts to tip over. This static tipping stability criterion can also be applied to roll stability while turning, by adding an extra centrifugal force component applied at the CoG, which will contribute additional tipping moment.

3.2.2 Relationship Between System Centre of Gravity and Tipping Stability

For a given configuration of the single-mass model, the location of the system CoG is fixed. With the two-mass model, though, we are able to move the system CoG by moving system $\mathcal{D}$ relative to system $\mathcal{C}$. Because the mass of the driver-seat system is roughly equivalent to that of the chassis, equation (3.3) suggests that for any given movement of system $\mathcal{D}$ relative to system $\mathcal{C}$, the system CoG will move half as much in the same direction. We will make use of this property in order to actively shift the system CoG to maximize tipping stability by maximizing the zone of stability.

It is apparent from equation (3.8) that an increased height h of the CoG above the ground will lower θ_{crit} and hence decrease the tipping stability, which is an intuitively obvious result. Furthermore, increasing the horizontal displacement l_1 of the CoG seems to increase the θ_{crit}. However, one cannot increase l_1 without bound.

For $l_1 < l$, the wheelchair retains tipping stability for the one-sided zone of stability $\theta \in [0, \theta_{crit}]$. If one makes $l_1 > l$, though (forcing relative displacement l_2 negative), the region of stability flips around to $\left[\theta_{crit}, \frac{\pi}{2}\right]$, which is not generally desirable. This fits with simple physics and the observations of [27] that the projection of the CoG upon the

horizontal plane must remain within the footprint formed by the projection of the bounded area between the ground contact points onto the horizontal plane.

Because $l = l_1 + l_2$, increasing the system l_1 by moving mass $\varnothing$ decreases the system l_2. It is possible to calculate a tipping stability criterion for the uphill side, which will give us a two-sided zone of stability $\theta \in [-\theta_{crit2}, \theta_{crit1}]$ where

$$\tan(\theta_{crit2}) = \frac{l - l_1}{h} = \frac{\pi}{2} - \theta_{crit1} \tag{3.9}$$

so long as $l_1 < l$. The total zone of stability will be given by:

$$\theta_{crit1} + \theta_{crit2} = \frac{\pi}{2} \tag{3.10}$$

For a given chassis tilt angle θ_c, the optimal stability zone is chosen such that θ_c is equidistant from the two stability limit angles. This choice will be given by choosing the parameters such that:

$$\theta_c = \tfrac{1}{2}(\theta_{crit1} - \theta_{crit2}) = \theta_{crit1} - \frac{\pi}{4} \tag{3.11}$$

which will centre the system CoG inside the wheelchair's footprint.

3.2.3 Formulation of Dynamic Tipping Stability

In the dynamic stability test of [40], the wheelchair is braked suddenly going downhill. In the worst case, the wheelchair comes to a sudden halt rather than decelerating smoothly, similar to impact with a low obstacle going downhill, as in Figure 3.4. In such a case, the rear wheels of the wheelchair will definitely leave the ramp. We will determine the response to this impulse input by looking at the extent to which the wheelchair tips and the conditions required for such tipping.

Subscript 0 refers to the situation immediately prior to impact, subscript 1 refers to the situation immediately after impact, and subscript 2 refers to the situation some time after impact.

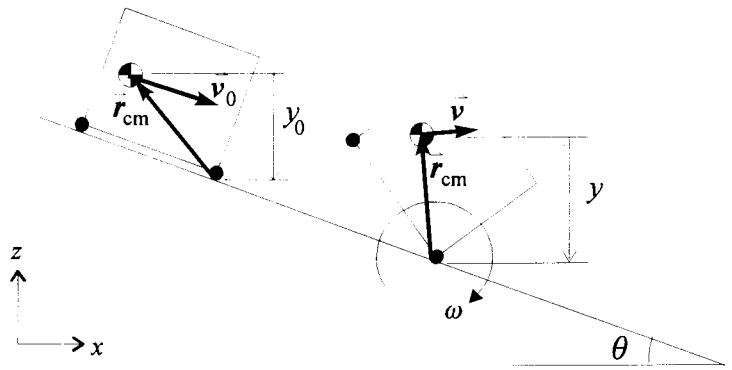


Figure 3.4: Wheelchair plastic collision

3.2.4 Threshold Unrecoverable Tipping Speed

Consider a wheelchair moving downhill with a translational speed v_0 and rotational speed ω_0=0 when the brakes are applied suddenly. In such a case, we can find the speeds v_1 and ω_1 immediately post-impact by summing the system momenta before and after the impact [43]. Assuming this to be a plastic collision in which the movement of the wheelchair's wheels is totally stopped, we apply the principle of conservation of angular momentum:

$$(\boldsymbol{M}\boldsymbol{v}_0)\boldsymbol{h} = (\boldsymbol{M}\boldsymbol{v}_1)|\vec{r}_{cm}| + \boldsymbol{I}_{cm}\omega_1 \tag{3.12}$$

Because the wheelchair starts rotating about the point of impact, we consider that $v_1 = |\vec{r}_{cm}|\omega_1$. Combining this expression with equation (3.12) leads to:

$$\boldsymbol{M}\boldsymbol{v}_0\boldsymbol{h} = \left(\boldsymbol{I}_{cm} + \boldsymbol{M}|\vec{r}_{cm}|^2\right)\omega_1 \tag{3.13}$$

which yields an expression for the total kinetic energy post-impact:

$$\begin{aligned} T_1 &= \tfrac{1}{2} M v_1^{\,2} + \tfrac{1}{2} I_{cm} \omega_1^{\,2} \\ &= \tfrac{1}{2} \left(I_{cm} + M \left| \vec{r}_{cm} \right|^2 \right) \omega_1^{\,2} \end{aligned} \tag{3.14}$$

Note that the wheelchair has lost some of its kinetic energy to the plastic collision, so that in general $T_1 < T_0$. We will comment more on this fact in the next section.

We can calculate the conditions required to induce unrecoverable tipping of the wheelchair, where the wheelchair's tilt angle in world co-ordinates has reached the limit of static stability θ_{crit} with zero speed. Tipping of the wheelchair up to this point will be recoverable, as the wheelchair will be statically stable even at maximum tip, and hence recover its initial position and orientation. If the initial kinetic energy is enough to push the wheelchair past this point, then the wheelchair will have no static stability.

It is desirable to maximize this threshold, so that for any given braking situation, one will have the maximum possible margin to unrecoverable tipping. Applying conservation of energy post-impact,

$$T_1 + V_1 = T_2 + V_2 \tag{3.15}$$

where $V_1 = Mgy_1$ and $V_2 = Mgy_2$. $y_1 = y_0$ because the wheelchair hasn't moved yet immediately after the impact. In unrecoverable tipping, the wheelchair must pass through the position where $y_2 = |\vec{r}_{cm}|$. Since we are trying to find the minimum speed which will lead to unrecoverable tipping, we will consider the case where $T_2 = 0$, i.e., where the wheelchair has just enough kinetic energy left after impact to raise the system to the threshold of unrecoverable tipping. Solving equation (3.15) for T_1 and substituting the expressions from equations (3.13) and (3.14) yields:

$$T_1 = \frac{\frac{1}{2}(Mv_0 h)^2}{I_{cm} + Mr_{cm}^2} = Mg(y_2 - y_0) \tag{3.16}$$

The geometry in Figure 3.4 suggests that:

$$\begin{aligned} y_0 &= |\vec{r}_{cm}| \sin\left(\theta + \frac{\pi}{2} - \theta_{crit}\right) \\ &= |\vec{r}_{cm}| \cos(\theta_{crit} - \theta) \end{aligned} \tag{3.17}$$

At the threshold of tipping, $y = r_{cm}$. Substituting into equation (3.16) along with equation (3.17), and solving for $v_{thresh} = v_0$ gives us:

$$v_{thresh} = \sqrt{\frac{2g|\vec{r}_{cm}|}{Mh^2}\left(I_{cm} + M|\vec{r}_{cm}|^2\right)(1 - \cos(\theta_{crit} - \theta))} \tag{3.18}$$

Equation (3.18) is an expression for the maximum speed the wheelchair can be travelling at when the brakes are applied past which the wheelchair will tip completely over in an unrecoverable fashion. As expected, increasing the slope angle θ or the height of the system CoG h will decrease v_{thresh}.

The system moment of inertia I_{cm} is related to the moments of inertia I_c and I_d of the two components of the two-mass model through the parallel-axis theorem and judicious application of equation (3.3):

$$\begin{aligned} I_{cm} &= I_c + I_d + m_c|\vec{r}_{cm} - \vec{r}_c|^2 + m_d|\vec{r}_{cm} - \vec{r}_d|^2 \\ &= I_c + I_d + \frac{m_c m_d}{m_c + m_d}|\vec{r}_{cd}|^2 \end{aligned} \tag{3.19}$$

The implication of equation (3.19) is that by increasing the separation $|\vec{r}_{cd}|$ of the two component masses, we can increase I_{cm} and hence the resistance of the wheelchair to tipping. Furthermore, we get a double effect on v_{thresh}, as increasing $|\vec{r}_{cd}|$ will also

increase $|\vec{r}_{cm}|$ in equation (3.18). Note, however, that in increasing $|\vec{r}_{cd}|$, we must be careful not to also decrease θ_{crit}, as doing so will simply decrease v_{thresh} in equation (3.18).

3.2.5 Maximum Tip Criterion

Another criterion we can use is the maximum amount of tip Δy which is generated upon braking, which is a measure of how much time the wheelchair's wheels spend off the ground and out of play as control surfaces for the wheelchair. It is desirable to minimize Δy as decreasing the amount of time the wheelchair's wheels spend away from the ground increases the control the driver has over a given braking situation.

$\Delta y = y_2 - y_0$, so from equations (3.16) and (3.17):

$$\begin{aligned} \Delta y &= \frac{M v_0^{\,2} h^2}{2g\left(I_{cm} + M|\vec{r}_{cm}|^2\right)} \\ &= \frac{T_0}{g}\frac{h^2}{\left(I_{cm} + M|\vec{r}_{cm}|^2\right)} \end{aligned} \tag{3.20}$$

Note that equation (3.20) is independent of the ramp angle θ. It is only the amount of kinetic energy on impact which affects the height Δy which the wheelchair gains.

We can rewrite equation (3.20) in more usable terms if we recall from the single-mass model that $|\vec{r}_{cm}|^2 = h^2 + l_1^{\,2}$. Applying this expression for $|\vec{r}_{cm}|^2$ to equation (3.20), we get the ratio of change in potential energy ΔU at maximum tilt to kinetic energy on impact:

$$\frac{\Delta U}{T_0} = \frac{Mh^2}{\left(I_{cm} + M\left|\vec{r}_{cm}\right|^2\right)} = \frac{1}{1+\left(\frac{I_{cm} + Ml_1^2}{Mh^2}\right)} < 1 \qquad (3.21)$$

The ratio of equation (3.21) calculates the fraction of the kinetic energy originally available which is not dissipated in the plastic collision, and hence the ratio is always less than unity. In the real world, the plastic collision in this model could correspond to braking via some dissipative friction process or to sudden stopping when hitting some irregularity in the ground such as a curb. It is obviously of benefit to have as much kinetic energy as possible dissipated in the braking process and as little as possible transformed into potential energy in raising the rear end of the wheelchair off the ground.

To decrease the ratio of equation (3.21), one notes that increases in the system moment of inertia I_{cm} and the horizontal displacement of the system CoG (relative to the baseline between the two ground contact points) l_1 both have a desirable (decreasing) effect on the ratio, while increasing the vertical displacement h has the expected undesirable effect (increasing the ratio). As I_{cm} will increase with increases in $\left|\vec{r}_{cd}\right|$, we can conclude that in order to increase the wheelchair tipping stability, we should increase $\left|\vec{r}_{cd}\right|$ in such a manner as to increase l_1 and decrease or keep constant h, so the primary adjustment is one of sliding the driver-seat system back-and-forth. It is still desirable, though, to have some rotation of the driver-seat system in order to reduce shear forces, as pointed out in section 2.2.

3.2.6 Effect of Active Control on Stability

According to [36], approximately half of the mass in humans (mean 56.5% percentage by weight among the population sampled) is located in the torso minus the limbs. As it is the torso which is secured into the seat, it is fairly safe to treat the driver and seat combination as one rigid body.

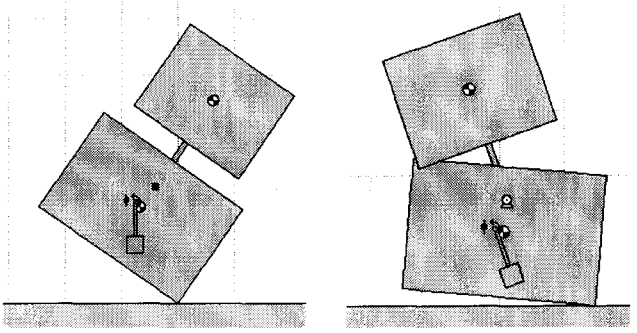


Figure 3.5: Simulation of person on wheelchair with active levelling

A simple simulation which can accomplish the goals of improving tipping stability and limiting shear forces on the driver is shown in the simulation trace of Figure 3.5. Here, the two-mass model has been used to demonstrate active control over system centre of gravity. Each simulation has been captured at the point of maximum system tilt. The model at the left has the two masses rigidly joined together, and is similar to the single-mass model. The model at the right has the second mass under active control, via a torque applied at the indicated pivot point.

The mass of system $\mathscr{D}$ in this model is 40 kg, with a moment of inertia of 27 kg m^{-2}, while the mass of system $\mathscr{C}$ is 48 kg, with a moment of inertia of 52 kg m^{-2}. The CoG of system $\mathscr{D}$ is 3.275 m off the ground, while the CoG of system C is 1.0 m off the ground.

The l_1 parameter is 1.5 m. In both cases, the systems were started at a speed of 4.478 m/s, which is the threshold tipping speed for the rigid system.

The active mechanism itself consists of a powered rotational joint, similar to those found in manually-controlled tiltable wheelchair seats, which pitches the driver and seat back and forth. Because the driver-seat system CoG is 0.25m above this rotational joint, a rotation in either direction will have the effect of changing l_1 and decreasing h. Furthermore, the resulting change in attitude of the driver-seat system can be used to reduce shear forces if the alignment is such that the head of the driver is always pointed away from the net force vector on the system. The net force vector is a vector sum of gravitational pull combined with any forces experienced as a result of acceleration. The sensor used is a damped pendulum mounted on the wheelchair chassis. This pendulum allows the direct measurement of the net force vector on the wheelchair at any time.

Dynamically, the driver-seat system in this example behaves like an inverted pendulum, and a simple PD controller is used to control the tilt of the driver-seat system to match the tilt angle of the pendulum. As the comparison in Figure 3.6 shows, this simple controller does an adequate job of keeping the chassis near the ground, allowing the driver to recover control of the wheelchair after stopping much sooner than if the seat-to-chassis connection were rigid.

Furthermore, as seen in Figure 3.7, the addition of the controller decreases the peak value of the driver rotation in an absolute sense, relative to the inertial world frame $\mathcal{W}$.

The disadvantage of this simplistic controller and mechanism, though, is that the average deceleration force experienced by the driver under the controlled case is greater than that experienced for the rigid connection model. This disparity is a simple

consequence of the much shorter stopping distance which the driver moves through in coming to a complete halt for the controlled vs. the rigid connection model, and cannot be avoided except by allowing the driver to move further under the controlled case than he or she does in the rigid connection case, which would compromise tipping stability. This increase in force is not necessarily an insurmountable design problem, however, as an intelligent reorientation of the driver allows him or her to take the force on the body surfaces which are already evenly supported: the bottom and the back, cushioned by the seat.

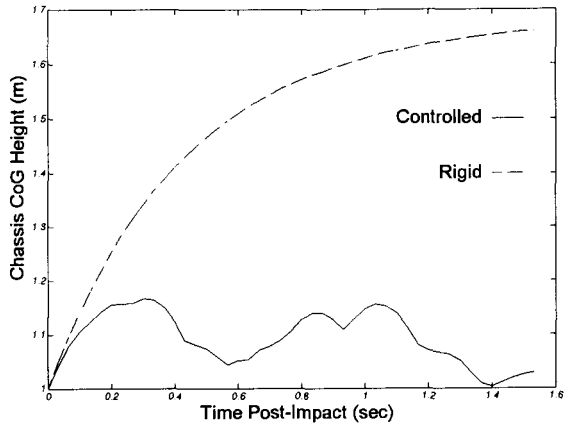


Figure 3.6: Comparison of chassis CoG height post-impact

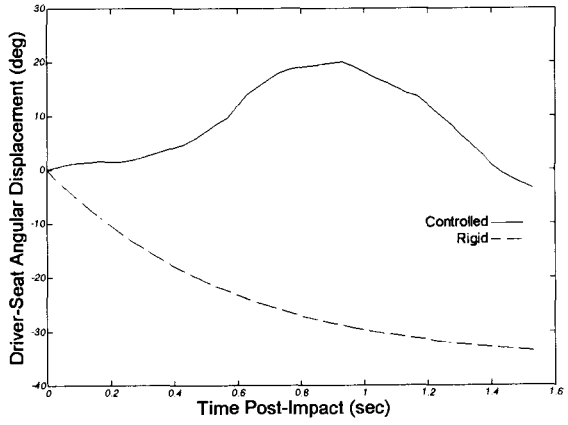


Figure 3.7: Comparison of driver-seat system rotation post-impact

3.3 Suspension Systems: Tipping Stability vs. Vibration Isolation

[35] and [44] discuss some of the design conflicts which arise as a result of the various design requirements imposed on suspension systems. For example, softer springs may make for a smoother ride but will increase the degree of body attitude change as the vehicle accelerates, whether in a straight line or around a curve. The converse is also true. This situation is easily understood through a force diagram such as that in Figure 3.8, which presents a simplified illustration of the forces on the vehicle body just as it starts to turn a corner (left), and a little while after the corner has been begun, when the vehicle has tilted on its suspension by an angle θ (right).

At the beginning, the vehicle body is level. Together, $F_1 + F_2 = F_G$, the gravitational force on the vehicle body. Assuming a simple spring with a linear restoring force as the suspension on each of the two wheels at both bottom corners of the box representing the vehicle body, the upward forces (F_1, F_2) at each suspension point will be the same. (Note that F_1 and F_2 are not the same as the normal forces to the wheels.) Including the frictional forces F_{F1} and F_{F2}, there will be a net torque clockwise, which will lead to the situation shown in the second diagram, in which the vehicle has tilted slightly to the right by an angle θ and compressing spring 2 more than spring 1, resulting in a restoring torque to force the vehicle back to its original attitude (i.e., no tilt).

Note that a higher spring constant (i.e., stiffer springs) results in a greater restoring torque for the same amount of tilt. It is desirable to keep the wheel-to-ground forces balanced as such a configuration allows for the optimum use of the available tire contact

surface, maximizing the vehicle's grip on the road. Furthermore, as was pointed out in the previous sections, increased tilt of the vehicle decreases the footprint and decreases the tipping stability. The conflict between tipping stability and vibration isolation cannot be perfectly resolved using a purely passive suspension system [35].

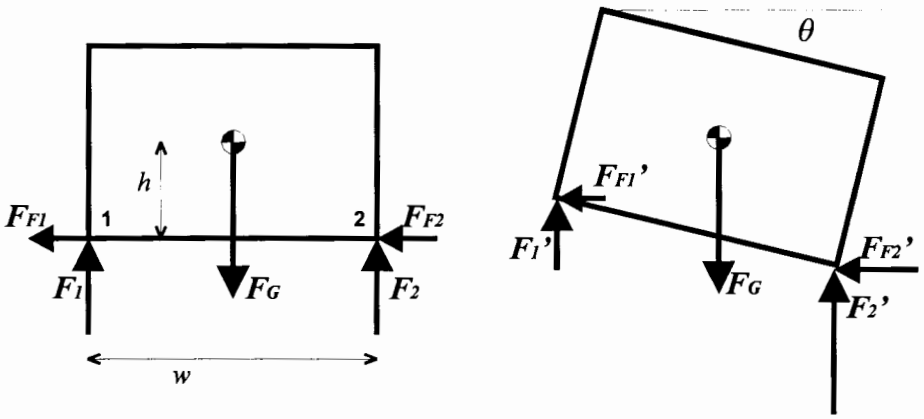


Figure 3.8: Forces applied to vehicle with suspension during cornering.

Chapter 4

Mechanical Design

This chapter discusses the design, optimization, and analysis of a closed-chain kinematic linkage which performs some of the required functions of the suspension, keeping the driver in the seat level while preventing the wheelchair from tipping over.

4.1 Problem Definition

The ultimate goal of this project is to isolate a platform — the seat-driver system — from the tilt of the wheelchair chassis in pitch and roll and provide some measure of relief from accelerations in the three translational degrees of freedom. Further, given that the driver must experience some acceleration as the wheelchair moves along, the suspension system should attempt to orient the driver such that he or she experiences forces which:

1) tend to keep him or her in the seat of the wheelchair,

2) minimize shear forces on his or her skin, and

3) pull at his or her body in such a way that the resulting pressures are evenly distributed across the body.

One condition which readily satisfies both conditions is to ensure that, regardless of the direction of the net force vector defined in section 3.1, the head of the driver, $\mathcal{H}_z$, is either pointed in the opposite direction to the net force vector $\vec{F}_R$ or tilted slightly back, as shown for a two-dimensional example in Figure 4.1. Here, the arc radiating from the CoG indicates the permissible range of directions in which $\vec{F}_R$ is allowed to point without letting the driver slip out of the seat. For safety reasons, though, the situation of the picture on the left in Figure 4.1 is preferred, as travel will usually take place in a direction roughly perpendicular to $\vec{F}_R$. The upright attitude of the driver shown on the left allows the driver to more easily see and navigate the approaching terrain than the reclining attitude of the driver shown on the right.

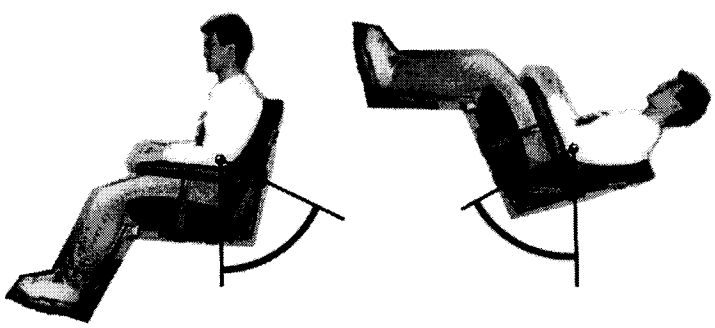


Figure 4.1: Permissible directions for acceleration force vector

As pointed out in previous sections, the cushions in the seat are much better at restraining the driver from moving under the presence of external forces than a seat belt restraint. For the general 6-dof case, the seat would need to yaw and roll in addition to the pitching motion indicated in Figure 4.1.

In addition to orienting the driver correctly, it is important that the suspension system keep the projection of the seat-driver system CoG $\mathcal{D}$ along the direction of $\bar{F}_R$ onto the plane of the wheelchair's footprint as close to the geometric center of the footprint as possible. Based on the results of section 3.2, doing so will enhance the wheelchair's tipping stability.

As a test case for the design methodology, this thesis will concentrate on the design of a simpler two-dimensional, single degree-of-freedom mechanism whose responsibility will be to keep the moving platform upright at all times and the seat-driver system CoG directly above the centre of the wheelchair's footprint by moving and reorienting seat-driver system along a prescribed path.

Using a single degree-of-freedom (1-dof) mechanism allows for the establishment of baseline results which can be used further on in the project without having to worry about interactions between different degrees of rotational and translational motion. Furthermore, development and analysis on a 1-dof system allows direct comparison to existing 1-dof tilt wheelchair seat tilt-in-space systems, to determine the efficacy of adding active tilt sensing to what is essentially a tilt-in-space mechanism. Finally, a successful demonstration of the control algorithm and techniques on a 1-dof system may allow for wider acceptance of the system through integration with existing tilt-in-space products.

Existing designs such as the Zippie P500 (Figure 1.3) allow the user to tilt backwards as much as 40° to 45° under manual control. This design, though, is limited to tilting back only, as it was designed with pressure relief in mind. Furthermore, it fails to keep the total wheelchair CoG centered within the chassis' footprint as the seat tilts back,

unlike the Invacare Tarsys (Figure 1.5), as a result of which more restrictions are placed on the tilting than with the Tarsys. The Tarsys, though, suffers from the drawback of not being able to tilt forward, which could be important when ascending an incline, as it too is designed primarily for pressure relief, unlike the Quest ACCESS (Figure 1.6) which is designed to actively level the driver.

Keeping the seat-driver CoG (origin of $\mathcal{D}$) directly over the center of the wheelchair's footprint and ensuring that the $\mathcal{H}_z$ axis always points opposite to $\vec{F}_R$ can be accomplished by defining a path which ensures that the z-axis of $\mathcal{D}$, $\mathcal{D}_z$, is always aligned with the displacement vector $\vec{r}_{cd}$ which locates the origin of $\mathcal{D}$ in the frame of $\mathcal{C}$. Figure 4.2 shows how the co-ordinate frame $\mathcal{D}$, which locates the seat and driver, is defined for three positions. Considering that one of the design objectives is to allow both uphill and downhill travel, it makes sense to define a minimum of three such positions as the prescribed positions: at a forward tilt of 20° ($\vec{R}_{h1}$) for uphill travel, at the neutral position ($\vec{R}_{h2}$) for level travel, and at a backwards tilt of 20° ($\vec{R}_{h3}$) for downhill travel. The two tilted positions define the limits of required motion, and so are definitely necessary. It is also important to have the neutral position defined as one of the prescribed points in the process of mechanism synthesis, as the wheelchair will spend most of its time in travel over level ground, where no tilt of the seat-driver system is necessary. By defining one of the three prescribed points at the neutral position, we ensure that the balance of the wheelchair as a whole will be optimal for level travel.

Another kinematic constraint is that the seat and driver must be kept clear of the ground at all times so that the footrests don't hit the ground if the seat is tilted forward

while running the wheelchair uphill. As well, it is desirable to keep the seat and driver as low to the ground as possible at all times, as decreasing CoG height above ground increases the tipping stability, as demonstrated in equation (3.18). This condition will drive the choice of the lengths of the vectors $\vec{R}_{h1}$, $\vec{R}_{h2}$, and $\vec{R}_{h3}$ in the design optimization stage.

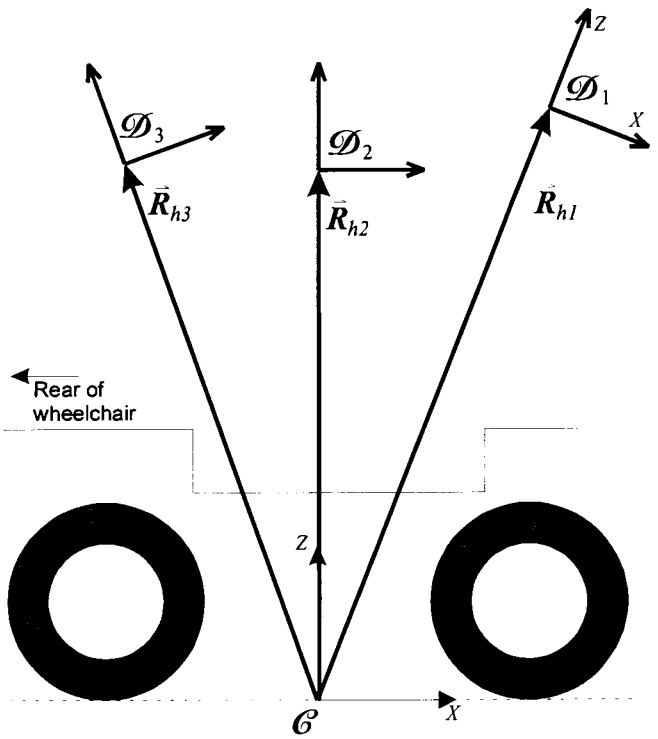


Figure 4.2: Motion definition vectors showing three prescribed positions

It is also important to ensure that any mechanism resulting from the design process be able to fit within the physical space available underneath the seat. These boundaries are illustrated in Figure 4.3, which shows the space available for a power-base-style wheelchair chassis with rear-drive motors, in a well directly above the wheelchair batteries. The vector $\vec{r}_{LO}$ locates the bottom of the available space relative to the ground. The two vectors $\vec{r}_{s1}$ and $\vec{r}_{s2}$ locate the front and rear top corners of the well beneath the

seat. Within this well, a rectangular bounded area whose opposite corners are given by the co-ordinates (x_{FRT}, z_{TOP}) and (x_{BAK}, z_{BOT}) defines the space in which it is possible to locate the ground link of the mechanism. These co-ordinates may or may not reach as far as the boundaries of the well, although in this particular design example they don't. The exact value of the vectors $\vec{r}_{s1}$, $\vec{r}_{s2}$, and $\vec{r}_{LO}$ depends on the wheelchair geometry, while the co-ordinates (x_{FRT}, z_{TOP}) and (x_{BAK}, z_{BOT}) are determined by both the wheelchair geometry and the physical size of the mechanism's components (motors, linkages, etc). Numerical values for the Fortress power base chassis used in this thesis are given in section 4.4 below on dimensional optimization.

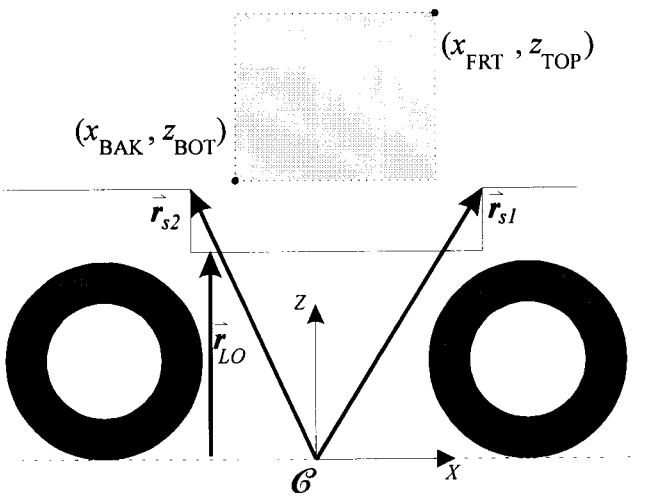


Figure 4.3: Boundary conditions on mechanism in space beneath seat

Another kinematic criterion insists that the mechanism remain in the same kinematic configuration throughout the generated motion, that the required motion not take the mechanism through a kinematic inversion point, or singularity, as mechanisms become difficult to control around a singularity.

In summary, this thesis will concentrate on designing for active level compensation for wheelchair pitch, as these results can be applied to further development on a multi-degree-of-freedom system.

4.2 Type Synthesis

The motion task described above is a classic example of motion planning with a single degree of freedom confined to a single plane, the *x-z* plane, in the wheelchair co-ordinate system $\mathcal{G}$. To solve this problem, a two-dimensional 1-dof mechanism may be employed. Fortunately, design for such a mechanism has long been established in the literature [45]. The simplest closed kinematic chain mechanism which will perform the task belongs to the class of mechanisms known as 4-bars. There are six varieties of 4-bar mechanisms consisting of: RRRR, TRRR, TRTR, TTRR, TTTR, and TTTT, where T denotes a translational joint, while R denotes a rotational one. (Figure 4.4)

Note that two very simple open-link mechanisms (T and R) are also able to generate the appropriate path. The R mechanism, while directly fulfilling the specification keeping the origin of $\mathcal{D}$ directly above $\mathcal{G}$, is not very practical, as it requires that a physical rotational joint be located at the origin of $\mathcal{G}$, placing it right at ground level. The T mechanism requires that a curved guideway be fabricated for the range of motion, which can be an onerous manufacturing step.

Furthermore, rotational joints can be made lighter than a translational prismatic joint, which requires guideways. A rotational linkage can use heavier materials (e.g., steel) at the joint and lighter materials to form the body of the linkage (e.g., aluminum). All that having been said, the only choice left among the 4-bars is the 4R mechanism.

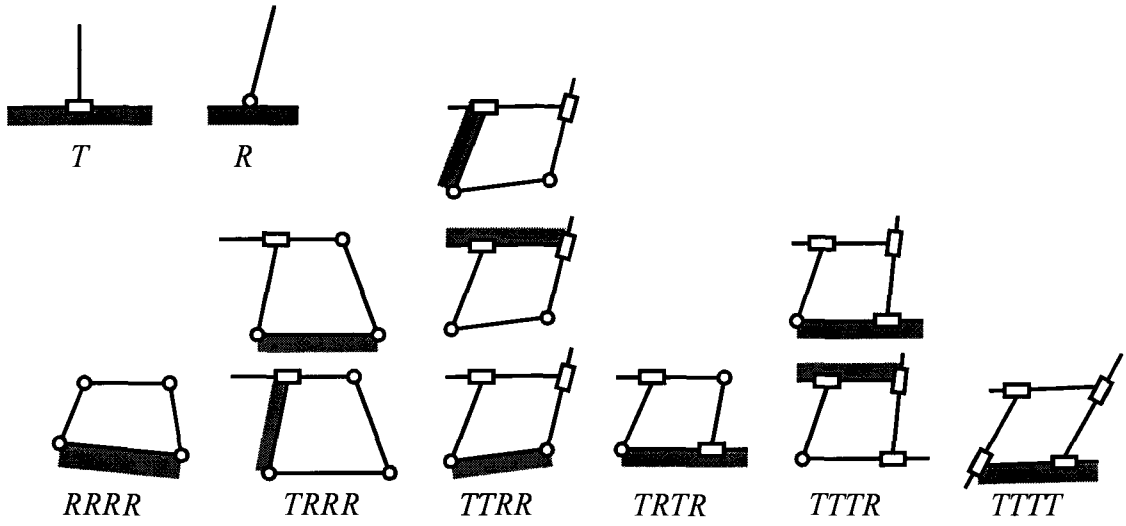


Figure 4.4: Simple 1-dof mechanisms

4.3 Dimensional Synthesis

In this section, the kinematic equations of motion of the 4R mechanism are developed and presented. The notation and some arguments of [45] are followed. Figure 4.5 describes the links and link angles for a generic four-bar mechanism, providing definitions for some of the symbols which will be used later on.

The inset photographs in Figure 4.6 show a side view of the mechanism as implemented on the LaMASS proof-of-concept.

In the following discussion, variables without a prime mark (e.g., θ_2) refer to the initial conditions, whereas those with the prime mark (e.g., θ_2') refer to conditions at some later configuration. The vectors are specified with respect to the co-ordinate frame $\mathcal{A}$. Given the link vectors at an initial, known configuration, wholly defined by complex vectors $\vec{r}_1$, $\vec{r}_2$, $\vec{r}_3$, $\vec{r}_4$, $\vec{r}_5$, and $\vec{r}_P$, it is desired to calculate the new link vectors $\vec{r}_1'$, $\vec{r}_2'$, $\vec{r}_3'$, $\vec{r}_4'$, $\vec{r}_5'$, and $\vec{r}_P'$ at another configuration of the mechanism, where:

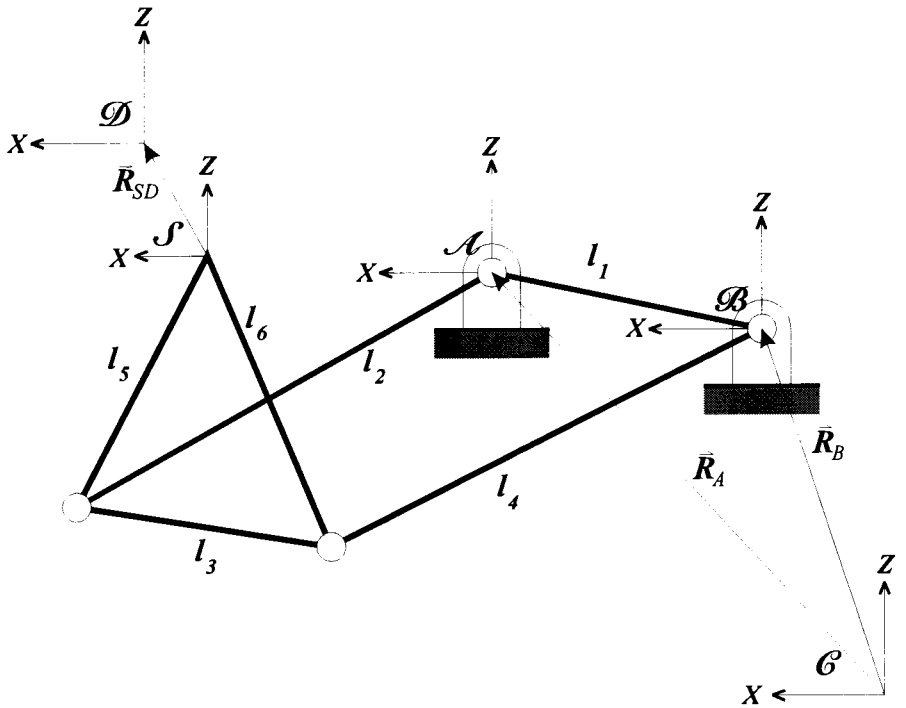


Figure 4.5: Four-bar mechanism with coordinate references

$$
\begin{aligned}
\vec{r}_1' &= \vec{r}_1 \\
\vec{r}_2' &= \vec{r}_2 e^{i(\theta_2' - \theta_2)} \\
\vec{r}_3' &= \vec{r}_3 e^{i(\theta_3' - \theta_3)} \\
\vec{r}_4' &= \vec{r}_4 e^{i(\theta_4' - \theta_4)} \\
\vec{r}_5' &= \vec{r}_3' e^{i\alpha} = \vec{r}_5 e^{i(\theta_3' - \theta_3)} \\
\vec{r}_P' &= \vec{r}_5' e^{i\beta} = \vec{r}_P e^{i(\theta_3' - \theta_3)}
\end{aligned}
\tag{4.1}
$$

Complex vectors are used here in order to provide a more compact notation given the amount of rotation which is used in the following arguments. Note that both $\alpha = \arg(\vec{r}_5') - \arg(\vec{r}_3')$ and $\beta = \arg(\vec{r}_P') - \arg(\vec{r}_5')$ are constants, such that vectors $\vec{r}_5'$ and $\vec{r}_P'$ maintain fixed attitudes relative to $\vec{r}_3'$ and to each other. As link 1 is fixed, and the link lengths do not change (e.g., $\|\vec{r}_2\| = \|\vec{r}_2'\|$), then in order to determine the new configuration, it is necessary to determine the new link angles θ_2', θ_3', and θ_4'. If we

consider link 2 to be the input link, then θ_2' will be the known input angle, and the problem becomes one of determining θ_3' and θ_4' based on θ_2'.

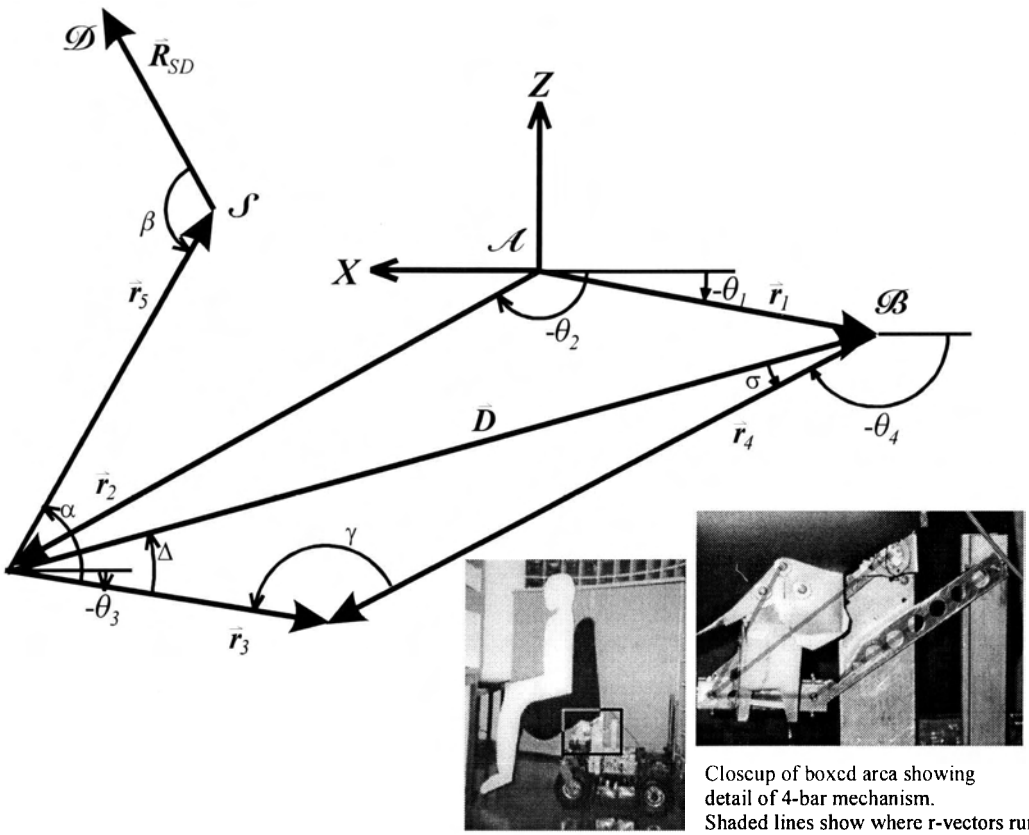


Figure 4.6: Linkage vector diagram.

In Figure 4.6, we define

$$\gamma = \text{Arg}(\vec{r}_4) - \text{Arg}(\vec{r}_3) \tag{4.2}$$

$$\alpha = \text{Arg}(\vec{r}_5) - \text{Arg}(\vec{r}_3) \tag{4.3}$$

and

$$\begin{aligned} \vec{D} &= \vec{r}_1' - \vec{r}_2' \\ &= \vec{r}_1 - \vec{r}_2 e^{i(\theta_2' - \theta_2)} \end{aligned} \tag{4.4}$$

where θ_2' = new link 2 angle, and θ_2 = original link 2 angle. As θ_2' is given, then $\vec{D}$ is known as well. Application of the law of cosines results in values for the angles Δ and σ.

$$\Delta = \text{sgn}(\gamma)\cos^{-1}\left(\frac{r_3^2 + D^2 - r_4^2}{2r_3 D}\right) \tag{4.5}$$

and

$$\sigma = \text{sgn}(\gamma)\cos^{-1}\left(\frac{r_4^2 + D^2 - r_3^2}{2r_4 D}\right) \tag{4.6}$$

These two angles can then be used to directly calculate the quantities of interest:

$$\theta_3 = \arg(\vec{D}) + \Delta \tag{4.7}$$

$$\theta_4 = \arg(\vec{D}) + (\pi - \sigma) \tag{4.8}$$

Of most interest to this problem is the equation (4.7), the expression for θ_3, as θ_3 bears a simple and fixed relationship to P. A sample graph of θ_3 as a function of θ_2 can be found in Figure 4.8.

We can also use the results of equation (4.2), as comparing the signs of the angle γ for each of the three precision positions lets us determine if the mechanism has gone through a kinematic inversion or not in between. A sign change indicates that l_2 and l_3 change configuration relative to one another. If the signs at all three points are the same, the implication is that the mechanism remains in the same kinematic configuration throughout the motion.

Although the forward problem of calculating link geometry and, specifically, the output link angle, given an input link angle for this single degree of freedom mechanism is easy, the inverse problem is not so obvious.

$$\omega_3 = -\omega_2 \frac{r_2}{r_3}\frac{\sin(\theta_4 - \theta_2)}{\sin(\theta_4 - \theta_3)} \tag{4.9}$$

and

$$\omega_4 = \omega_2 \frac{r_2}{r_4}\frac{\sin(\theta_3 - \theta_2)}{\sin(\theta_3 - \theta_4)} \tag{4.10}$$

Differentiating equations (4.9) and (4.10) results in the respective angular accelerations of links 3 and 4, as follows:

$$\alpha_3 = \frac{-r_2\alpha_2 \sin(\theta_4 - \theta_2) + r_2\omega_2{}^2 \cos(\theta_4 - \theta_2) + r_3\omega_3{}^2 \cos(\theta_4 - \theta_3) - r_4\omega_4{}^2}{r_3 \sin(\theta_4 - \theta_3)} \quad (4.11)$$

$$\alpha_4 = \frac{r_2\alpha_2 \sin(\theta_3 - \theta_2) - r_2\omega_2{}^2 \cos(\theta_3 - \theta_2) + r_4\omega_4{}^2 \cos(\theta_3 - \theta_4) - r_3\omega_3{}^2}{r_4 \sin(\theta_3 - \theta_4)} \quad (4.12)$$

For the four-bar mechanism in Figure 4.6, it is possible to draw three free-body diagrams, one for each of the moving links (2, 3, and 4), as shown in Figure 4.7 below. The co-ordinate frame indicated by the axes X-Z is an inertial reference frame, fixed with respect to the ground. Each link *j* is joined via frictionless pin joints to links *n* and *m* at either end. For link *j*, there are three externally-applied forces: $\vec{F}_{kj}$ and $\vec{F}_{lj}$ (the forces applied on link *j* by adjacent links *k* and *l*) and $\vec{F}_{gj}$ (applied force due to gravity on link *j*). In all three cases, the gravitational force vector points in the same direction but with differing magnitude, such that $\theta_{g2} = \theta_{g3} = \theta_{g4}$ and $\|\vec{F}_{gj}\| = m_j g$, where m_j is the mass of link *j* and g is the gravitational acceleration.

Defining the inertia force for the j^{th} link $\vec{F}_{Oj} = -m_j \vec{A}_{gj}$ where m_j is the mass of link *j* and $\vec{A}_{gj}$ is the resultant acceleration of the link center of mass, we can write the following dynamic equilibrium force equation:

$$\vec{F}_{12} + \vec{F}_{32} + \vec{F}_{g2} + \vec{F}_{O2} = 0 \quad (4.13)$$

Breaking this down into components yields:

$$\begin{aligned} F_{12x} + F_{32x} + F_{g2}\cos\theta_{g2} + F_{O2x} &= 0 \\ F_{12z} + F_{32z} + F_{g2}\sin\theta_{g2} + F_{O2z} &= 0 \end{aligned} \quad (4.14)$$

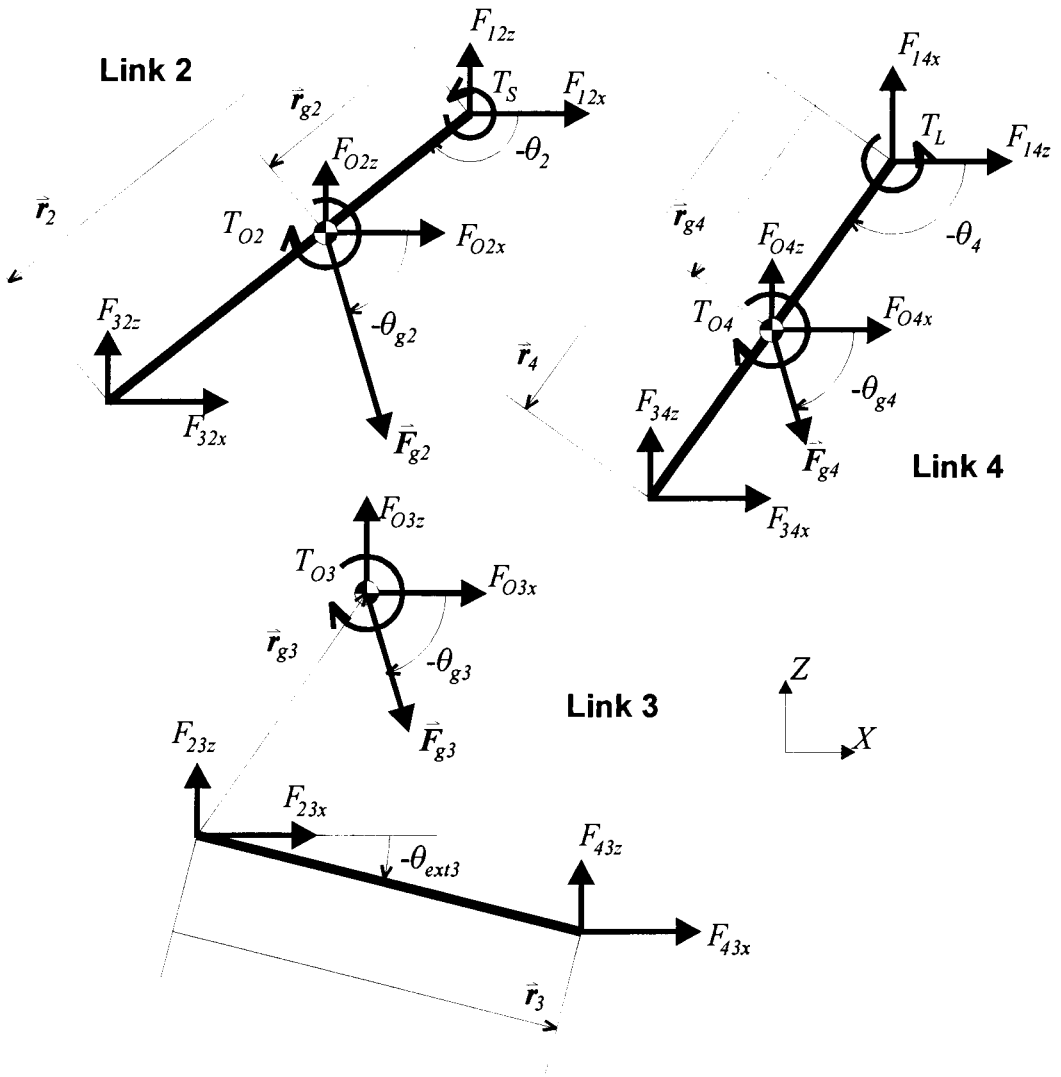


Figure 4.7: Free-body diagrams of four-bar mechanism linkages.

Similar equations can be written for links 3 and 4, thus:

$$\begin{aligned} \boldsymbol{F}_{43x} + \boldsymbol{F}_{23x} + \boldsymbol{F}_{g3}\cos\theta_{g3} + \boldsymbol{F}_{O3x} &= 0 \\ \boldsymbol{F}_{43z} + \boldsymbol{F}_{23z} + \boldsymbol{F}_{g3}\sin\theta_{g3} + \boldsymbol{F}_{O3z} &= 0 \end{aligned} \tag{4.15}$$

$$\begin{aligned} \boldsymbol{F}_{14x} + \boldsymbol{F}_{34x} + \boldsymbol{F}_{g4}\cos\theta_{gg} + \boldsymbol{F}_{O4x} &= 0 \\ \boldsymbol{F}_{14z} + \boldsymbol{F}_{34z} + \boldsymbol{F}_{g4}\sin\theta_{g4} + \boldsymbol{F}_{O4z} &= 0 \end{aligned} \tag{4.16}$$

In addition to the force equations, we need to write equations for the moment of each force on each link about its centre of mass. For example:

$$\begin{aligned}\vec{M}_{F12} &= \left(-\vec{r}_{g2}\right)\times \vec{F}_{12} \\ &= -r_{g2x}F_{12z} + r_{g2z}F_{12x}\end{aligned} \tag{4.17}$$

Similar equations can be written for each of the other five moment-generating forces shown in the diagram. These expressions for moment are used in the torque equations of dynamic equilibria. If we take $T_{Oj} = I_{Oj}\alpha_j$ then:

$$\begin{aligned} T_{F2} + T_s + M_{F12} + M_{F32} + T_{O2} &= 0 \\ T_{F3} + M_{F23} + M_{F43} + T_{O3} &= 0 \\ T_{F4} + T_L + M_{F34} + M_{F14} + T_{O4} &= 0 \end{aligned} \tag{4.18}$$

where T_L is the load torque due to external loading, T_{Fj} is the total frictional torque exerted on each link, and T_S is the driving torque at link 2 required to make the links move at the stated velocities and accelerations.

Combining equations (4.14), (4.15), (4.16), and (4.18), we have a system of equations which can be written in matrix form as:

$$[\mathbf{F_I}] = [\mathbf{L}][\mathbf{F_B}] - [\mathbf{F_g}] \Rightarrow [\mathbf{F_B}] = [\mathbf{L}]^{-1}\left([\mathbf{F_I}] + [\mathbf{F_g}]\right) \tag{4.19}$$

where the matrix variables $[\mathbf{F_I}]$, $[\mathbf{L}]$, $[\mathbf{F_B}]$, and $[\mathbf{F_g}]$ are defined as:

$$[\mathbf{F_I}] = \begin{bmatrix} F_{02x} \\ F_{02z} \\ T_{02} \\ F_{03x} \\ F_{03z} \\ T_{03} \\ F_{04x} \\ F_{04z} \\ T_{04} + T_L \end{bmatrix} \tag{4.20a}$$

$$[\mathbf{L}]=\begin{bmatrix} -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ -r_{g2z} & r_{g2x} & -1 & (r_{g2z}-r_{2z}) & (r_{2x}-r_{g2x}) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -r_{g3z} & r_{g3x} & (r_{g3z}-r_{3z}) & (r_{3x}-r_{g3x}) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & (r_{4z}-r_{g4z}) & (r_{g4x}-r_{4x}) & -r_{g4z} & r_{4x} \end{bmatrix} \quad (4.20b)$$

$$[\mathbf{F_B}]=\begin{bmatrix} F_{12x} \\ F_{12z} \\ T_S \\ F_{23x} \\ F_{23y} \\ F_{34x} \\ F_{34z} \\ F_{14x} \\ F_{14z} \end{bmatrix} \quad (4.20c)$$

$$[\mathbf{F_g}]=\begin{bmatrix} F_{g2}\cos\theta_{g2} \\ F_{g2}\sin\theta_{g2} \\ T_{F2} \\ F_{g3}\cos\theta_{g3} \\ F_{g3}\sin\theta_{g3} \\ T_{F3} \\ F_{g4}\cos\theta_{g4} \\ F_{g4}\sin\theta_{g4} \\ T_{F4} \end{bmatrix} \quad (4.20d)$$

Equations (4.19) and (4.20a-d) allow us to calculate the inter-link forces and, especially, the input torque required to generate a particular motion profile.

In order to determine the lengths of the four linkages, it is useful to employ the ground-pivot specification technique [46], in which the locations of the two ground-pivots and the locations and orientations of the coupling linkage 3 for three precision points are specified. This method is useful for this problem because the space in which

the mechanism's ground pivots are allowed is constrained to lie directly beneath the seat of the user, within the box drawn in Figure 4.3. Using this design method allows us to directly use this boundary condition. In section 4.4 below, this property is used to perform a search of the linkage space for the optimal mechanism, hence the choice of the ground-pivot specification design technique here in order to facilitate later optimization.

The following is a brief summary of the ground-pivot specification method as it applies here. Considering the vector diagrams in Figure 4.2, Figure 4.5, and Figure 4.6, we can write two systems of three equations, one for each of the three precision points. The three vectors which define the location of the precision points relative to frame $\mathcal{G}$ are:

$$\begin{aligned} \vec{R}_{h1} &= h_1 e^{i\psi_1} \\ \vec{R}_{h2} &= h_2 e^{i\psi_2} \\ \vec{R}_{h3} &= h_3 e^{i\psi_3} \end{aligned} \tag{4.21}$$

Let

$$\begin{aligned} \vec{W}_A &= \vec{r}_2 \\ \vec{Z}_A &= \vec{r}_5 \end{aligned} \tag{4.22}$$

and

$$\begin{aligned} \vec{W}_B &= \vec{r}_4 \\ \vec{Z}_B &= \vec{r}_6 \\ &= \vec{r}_5 - \vec{r}_3 \\ &= \vec{r}_5 - \left(\vec{r}_1 + \vec{r}_4 - \vec{r}_2\right) \\ &= \vec{r}_5 + \vec{r}_2 + \vec{R}_A - \vec{R}_B - \vec{W}_B \end{aligned} \tag{4.23}$$

The vector $\vec{R}_i$ (i = A or B) locates the ground pivot within frame $\mathcal{G}$, while vectors $\vec{W}_i$ and $\vec{Z}_i$, respectively, define half of the 4-bar. (i.e., either links l_2 and l_5 or links l_4 and l_6)

$$\begin{aligned}
\vec{W}_i + \vec{Z}_i &= \vec{R}_{h1} = h_1 e^{i\psi_1} - \vec{R}_i \\
\vec{W}_i e^{i\beta_2} + \vec{Z}_i e^{i\phi_2} &= \vec{R}_{h2} = h_2 e^{i\psi_2} - \vec{R}_i \\
\vec{W}_i e^{i\beta_3} + \vec{Z}_i e^{i\phi_3} &= \vec{R}_{h3} = h_3 e^{i\psi_3} - \vec{R}_i
\end{aligned} \tag{4.24}$$

The system of equations (4.24) can be rewritten in matrix form and solved by setting the determinant equal to 0, as in:

$$\begin{vmatrix} 1 & 1 & \vec{R}_{h1} \\ e^{i\beta_2} & e^{i\phi_2} & \vec{R}_{h2} \\ e^{i\beta_3} & e^{i\phi_3} & \vec{R}_{h3} \end{vmatrix} = 0 \tag{4.25}$$

Equation (4.25) results in:

$$\left(\vec{R}_{h3} e^{i\phi_2} - \vec{R}_{h3} e^{i\phi_3}\right) + e^{i\beta_2}\left(\vec{R}_{h1} e^{i\phi_2} - \vec{R}_{h3}\right) + e^{i\beta_3}\left(\vec{R}_{h2} - \vec{R}_{h1} e^{i\phi_2}\right) = 0 \tag{4.26}$$

or

$$D_1 + D_2 e^{i\beta_2} + D_3 e^{i\beta_3} = 0 \tag{4.27}$$

if we define:

$$\begin{aligned}
\vec{D}_1 &= \vec{R}_{h3} e^{i\phi_2} - \vec{R}_{h3} e^{i\phi_3} \\
\vec{D}_2 &= \vec{R}_{h1} e^{i\phi_2} - \vec{R}_{h3} \\
\vec{D}_3 &= \vec{R}_{h2} - \vec{R}_{h1} e^{i\phi_2}
\end{aligned} \tag{4.28}$$

A geometric argument in [46] yields the result:

$$\beta_2 = 2\arg(-\vec{D}_1) - \arg(\vec{D}_2) - \arg(\vec{D}_2 e^{i\phi_2}) \tag{4.29}$$

and

$$\beta_3 = 2\arg(-\vec{D}_1) - \arg(\vec{D}_3) - \arg(\vec{D}_3 e^{i\phi_3}) \tag{4.30}$$

We can then put this result back into the system of equations (4.24):

$$\begin{aligned}
\vec{W}\left(e^{i\beta_2} - 1\right) + \vec{Z}\left(e^{i\phi_2} - 1\right) &= \vec{R}_{h2} - \vec{R}_{h1} = \vec{\delta}_2 \\
\vec{W}\left(e^{i\beta_3} - 1\right) + \vec{Z}\left(e^{i\phi_3} - 1\right) &= \vec{R}_{h3} - \vec{R}_{h1} = \vec{\delta}_3
\end{aligned} \tag{4.31}$$

Solving (4.31) for the vectors $\vec{W}$ and $\vec{Z}$ through applying Cramer's rule on the 2x2 matrix formed from (4.28) results in:

$$\vec{W} = \frac{\begin{vmatrix} \vec{\delta}_2 & e^{i\phi_2} - 1 \\ \vec{\delta}_3 & e^{i\phi_3} - 1 \end{vmatrix}}{\begin{vmatrix} e^{i\beta_2} - 1 & e^{i\phi_2} - 1 \\ e^{i\beta_3} - 1 & e^{i\phi_3} - 1 \end{vmatrix}} \tag{4.32}$$

and

$$\vec{Z} = \frac{\begin{vmatrix} e^{i\beta_2} - 1 & \vec{\delta}_2 \\ e^{i\beta_3} - 1 & \vec{\delta}_3 \end{vmatrix}}{\begin{vmatrix} e^{i\beta_2} - 1 & e^{i\phi_2} - 1 \\ e^{i\beta_3} - 1 & e^{i\phi_3} - 1 \end{vmatrix}} \tag{4.33}$$

Equations (4.32) and (4.33) have been encoded into the FOURBAR.M MATLAB routine (see Appendix V) and as a result, it is possible to determine a possible 4-bar mechanism which takes $\mathcal{D}$ through each of the three precision points at $\vec{R}_{h1}$, $\vec{R}_{h2}$, and $\vec{R}_{h3}$ for any pair of ground pivots within the boundary area.

Equations (4.7) through (4.12) allow us to use the initial configuration provided by equations (4.28) through (4.33) (applied twice: once for each half of the 4-bar) to generate a kinematic motion profile as the driving link l_2 is swept around. Finally, equations (4.19) and (4.20) are used to calculate the static torque required to generate this motion. The result is shown in Figure 4.8, which shows a torque plot for one choice of ground pivots. An arbitrary input test function of zero angular acceleration with an angular velocity of 1 rad/sec over the range of angles for angle θ_2 was used. The ground pivots chosen are, in fact, the result of optimization described in the following section.

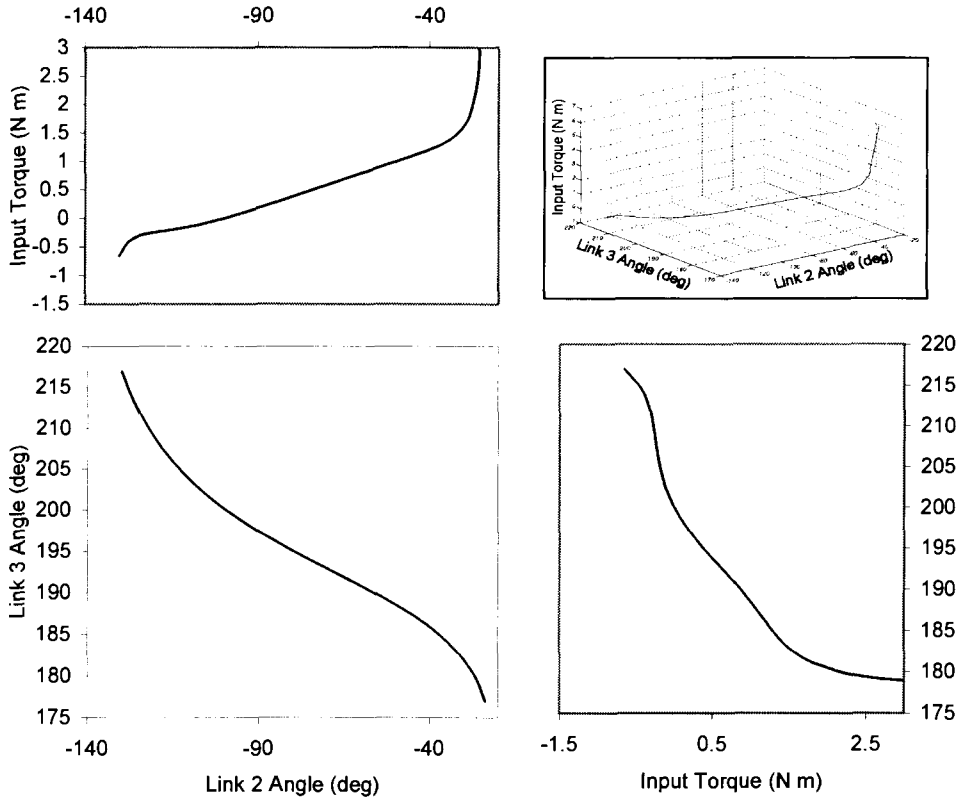


Figure 4.8: Input torque vs. link angle, for level mechanism.

4.4 Dimensional Calculation and Optimization

The results of the previous section, while not exactly defining the mechanism, do allow for the generation of a mechanism which will meet most of the design criteria set out in section 4.1. In this section, we consider the remaining design criteria and how best to meet them.

The first criterion we will examine is the one involving boundary conditions outlined in Figure 4.3. The shaded bounding box in this figure represents the area beneath the seat which can be used to house a mechanism for the suspension. Because of the finite size of motors, supports, and so on, a clearance of 0.02m is left between the bounding box and

the walls of the well. It is a rather simple argument to say that having both ground pivots lie inside the bounding box is a necessary but not sufficient condition for the mechanism to remain within the bounding box.

Measuring the Fortress power base chassis which forms the base for the LaMASS hardware, we arrive at the dimensions given in Table 4.1. Using these dimensions, we can readily use equations (4.28) through (4.33) of the ground pivot specification design technique to design any one of an infinite number of mechanisms whose ground pivots both lie inside the shaded box in Figure 4.3.

Table 4.1: Geometry parameter set for Fortress power base chassis

Parameter	**Dimension (x,z)**
$\vec{r}_{s1}$	(0.2125m, 0.343m)
$\vec{r}_{s2}$	(-0.1595m, 0.343m)
$\vec{R}_{LO}$	(0, 0.263m)
(x_{FRT}, z_{TOP})	(0.1705m, 0.5630m)
(x_{BAK}, z_{BOT})	(-0.1395m, 0.323 m)

To further limit the selection of ground pivots, we can use the other bounding criterion, that the parts of the mechanism do not hit the inside surface of the well. To check for an intersection, we use the values of the vectors $\vec{r}_{s1}$, $\vec{r}_{s2}$, and $\vec{R}_{LO}$. For a selected ground pivot $\vec{R}_i$ with its corresponding link vectors $\vec{W}_i$ and $\vec{Z}_i$, we can check to see if the following conditions are true:

$$\begin{aligned} \left\|\vec{R}_i - \vec{r}_{s1}\right\| &\le \left\|\vec{W}_i\right\| \\ \left\|\vec{R}_i - \vec{r}_{s2}\right\| &\le \left\|\vec{W}_i\right\| \\ \vec{R}_{i(Z)} - \left\|\vec{W}_i\right\| &\le \vec{R}_{LO(Z)} \end{aligned} \tag{4.34}$$

Equations (4.34) check to see if, by swinging either link 2 or link 4 around their ground pivots, it is possible to intersect either the floor of the well or the two corners.

For a given design, we must choose the location of the two ground pivots plus the three vectors which define the precision points. If, for simplicity, we fix the angles of the three precision point vectors at +20°, 0°, and -20°, we find that there are a total of seven numbers to be found which when defined completely specify the mechanism. Optimizing the design requires defining a fitness function of some sort which is dependent on these seven numbers.

Optimizing the design requires first observing the boundary conditions:

1) ground pivots both inside bounding box

2) no intersection of mechanism with wheelchair well, per equation (4.34)

3) no changing of kinematic configuration over motion path

A fourth boundary condition requires the length of link 3 to be no less than 0.07m, which is the smallest dimension which can still allow for the insertion of the bearings into either end for the pin joints.

For this thesis, the function which was chosen to be optimized was the average of static torques over the complete range of motion, with the input link 2 rotating at a steady 1 rad/sec with no angular acceleration. More sophisticated schemes could, for example, involve evaluations of the acceleration at the person's CoG.

Genetic algorithm optimization refers to a class of techniques by which an answer is evolved from a population of potential solutions, and are often used as function optimizers. One advantage of genetic algorithms over gradient descent for function optimization is that it is not necessary to have a closed-form differentiable expression for the function to be optimized [47]. Genetic algorithm techniques can also converge faster, under many circumstances, than other non-derivative techniques such as simulated annealing.

Genetic algorithms are characterized by the use of chromosomes and fitness functions. In order to apply genetic algorithm techniques to this case, it is necessary to first choose the encoding of the chromosomes within the sample population. Fortunately, as the results from above demonstrate, each four-bar mechanism for this problem solution can be uniquely determined using five vectors: the location of the two ground-pivots and the three locations of the tracer point throughout the path. Following [48], who suggests that chromosomes with real-valued parameters, or genes, can be more efficient than the binary representation, the chromosome of each member of the sample population is chosen to consist of seven real-valued numbers, giving the *xz* co-ordinates $\vec{R}_A$ and $\vec{R}_B$ of the two ground pivots, and the radial distances h_1, h_2, and h_3 from the wheelchair co-ordinate frame origin $\mathcal{O}$ defined earlier for each of three angular positions. Thus, each chromosome is uniquely identified with a single four-bar mechanism.

Accordingly, in the following discussion, "member of the population," "chromosome," and the corresponding four-bar mechanism represented by this member will be used interchangeably as the situation warrants, and will be understood to imply one another.

Having defined the chromosome, we can now define the fitness function. In order to represent the boundary conditions, the fitness function assigns highly negative values to members of the sample population which lie outside the boundaries. Because members are chosen according to a sorting function which allows those members with the best, most positive fitness values to reproduce, those members which lie outside the boundary conditions will have a very small, yet finite chance of reproduction. This chance is allowed in case some of these members contain a useful gene which would serve to take the system solution out of some local maximum. If the member passes all the boundary condition tests, then the inverse of the average of the input torque (calculated via equation (2.20) is calculated and used as the fitness of the member.

A routine using the GAOT (Genetic Algorithm Optimization Toolbox) set of routines for MATLAB [48] was used to perform the optimization (Appendix V). A population was randomly seeded with 1000 members. For each successive generation, crossover is performed using the arithmetic (linear interpolation) method, while mutation is performed using a uniform distribution over the search space.

Figure 4.9 plots the fitness of the best member of the population as a function of generation. As Figure 4.9 shows, the population solution converges very quickly. The best fitness, as shown in Figure 4.10, continues to improve, although the solution has levelled out by about generation 130. Further testing with trials running out to as much as 1000 generations shows no appreciable increase in the solution quality for the parameters used. Figure 4.11 shows the mean fitness of the population as the solution progresses. The result after 200 generations was a mechanism with the following properties:

Table 4.2: Final parameter set after GA optimization

$$\bar{R}_A = (0.1246\text{m}, 0.5564\text{m})$$
$$\bar{R}_B = (0.0089\text{m}, 0.5381\text{m})$$
$$h_1 = 0.7001\text{m}$$
$$h_2 = 0.5000\text{m}$$
$$h_3 = 0.5258\text{m}$$

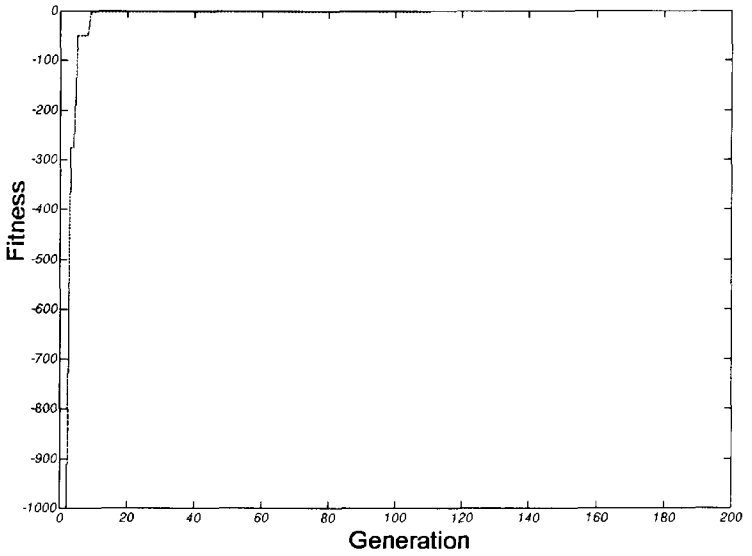


Figure 4.9: Plot of best fitness as a function of generation

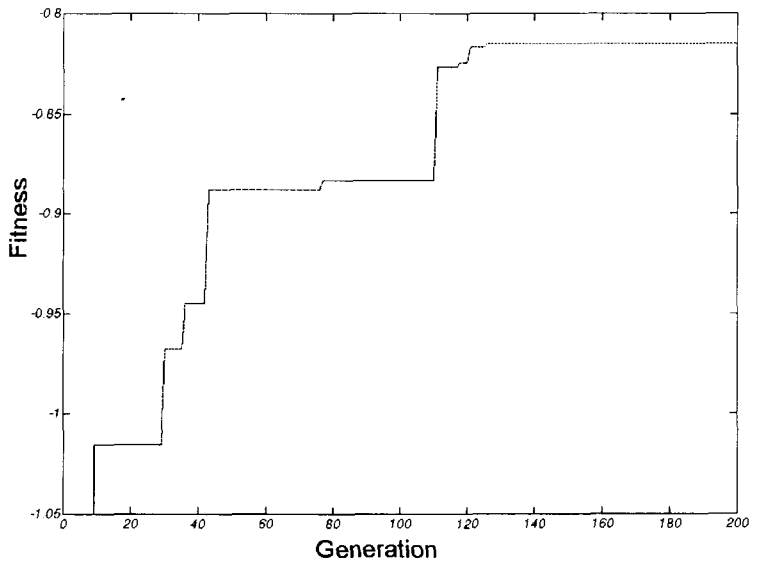


Figure 4.10: Plot of best fitness vs. generation, close-up

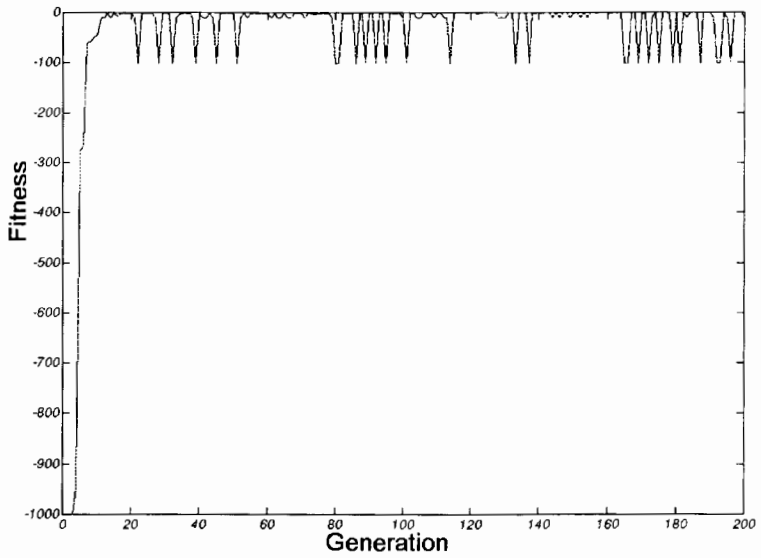


Figure 4.11: Plot of mean fitness vs. generation

Chapter 5

Implementation

A Fortress powered wheelchair, shown in Figure 2.1, was donated as the base on which to build the LaMASS proof-of-concept. This wheelchair is typical of many powered wheelchairs on the market today, with a separate chassis and seat. Working with the chassis alone and replacing the seat with a much lighter mock-up allowed the use of relatively safer, lower-power motors as compared to a full-sized system which would be capable of moving a full-sized person.

5.1 System Overview

The block diagram for the LaMASS proof-of-concept, shown in Figure 5.1 below, is primarily a feedforward system. Table 5.1 lists the symbols used in Figure 5.1 along with their meanings. In the nomenclature of [4], LaMASS is a deviation compensation system with feed forward, in which both the input disturbance and a suitably processed control signal based on the input disturbance are fed through a compensating servo system, which then transmits some (hopefully smaller for a properly-designed system) amount of

vibration to the isolated object. In this case, the servo system is comprised of the suspension mechanism and the closed-loop motor controller which controls the position of the suspension mechanism. As noted earlier, one of the disadvantages of using this feedforward scheme is that some knowledge of the incoming disturbance signal must be assumed. Although a functioning speedometer was placed on one of the wheels in an attempt to use speed data to help predict the incoming frequency spectra, this approach was never realized during this thesis due to time constraints. The wheel speedometer had been intended to supplement the data from the inertial data package on chassis motion.

In order to make a feedback system such as the disturbance compensator topology work, it is necessary to use a state estimator for the position, acceleration, and other motion characteristics of the seat-driver system. The reason why this is necessary is because, unlike the case with the wheelchair chassis, which can be quite easily instrumented, it is quite difficult to persuade a driver sitting on a seat that they need to have an inertial reference platform bolted somewhere to their person in order to measure their motion.

Each block shown in Figure 5.1 represents a discrete, hardware subsystem listed in the next sections.

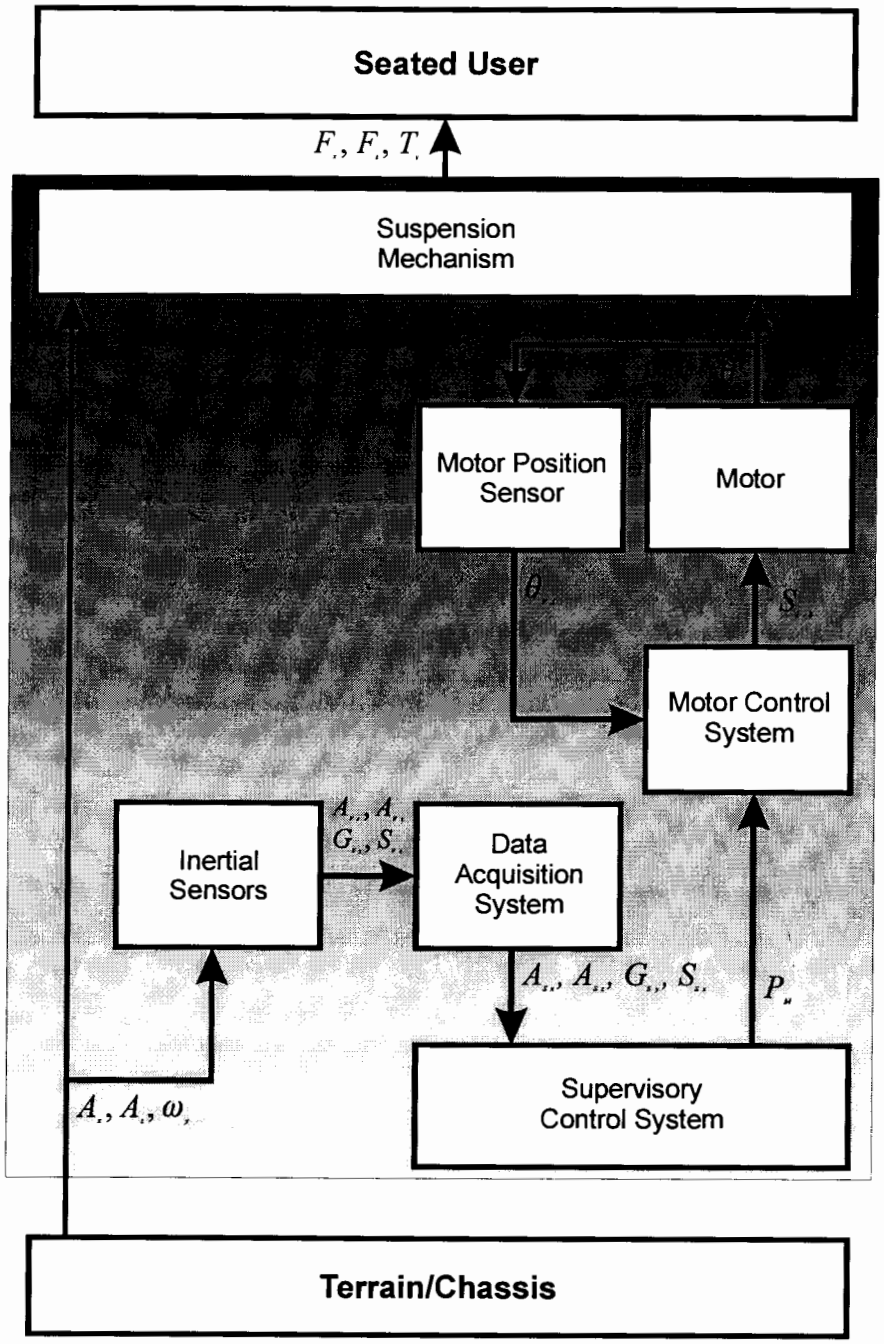


Figure 5.1: System block diagram showing information flow.

Table 5.1: Symbols used in system block diagram.

Symbol	Definition
A_x	x-component of chassis acceleration
A_z	z-component of chassis acceleration
A_{Sx}	x-component of chassis acceleration, digitized signal
A_{Sz}	z-component of chassis acceleration, digitized signal
A_{Vx}	x-component of chassis acceleration, raw inertial sensor signal
A_{Vz}	z-component of chassis acceleration, raw inertial sensor signal
F_x	x-component of force exerted on seated user
F_z	z-component of force exerted on seated user
G_{Sy}	digitized gyroscope inertial sensor signal
G_{Vy}	raw gyroscope inertial sensor signal
P_M	commanded motor position
S_N	step signal, to one of N=1..8 lines
S_{Sx}	digitized inertial tilt (roll: around x-axis) sensor signal
S_{Sy}	digitized inertial tilt (pitch: around y-axis) sensor signal
S_{Vx}	raw inertial tilt (roll: around x-axis) sensor signal
S_{Vy}	raw inertial tilt (pitch: around y-axis) sensor signal
T_y	torque about the y-axis exerted on seated user
θ_2	angle of link 2 in suspension mechanism
θ_{v2}	raw voltage signal directly proportional to link 2 angle
ω_y	rotational velocity about y-axis (pitch)

5.2 Hardware Subsystems

For testing and prototyping purposes, each of the system blocks in Figure 5.1 has been implemented using a separate piece of hardware. The following sections describe the function of each hardware subsystem shown in Figure 5.3. Schematic diagrams for the motor positioning board, motor controller board, data acquisition board, and power supply can be found in Appendix I.

All the hardware and mechanisms for LaMASS are bolted to a base plate of aluminum 0.5-inch thick. This aluminum plate is in turn securely bolted down to the main central steel spine of the wheelchair chassis framework. A bracket attached to the

rear bulkhead of the well provides additional stiffness and solidity to the base plate, which can be seen in Figure 5.2.

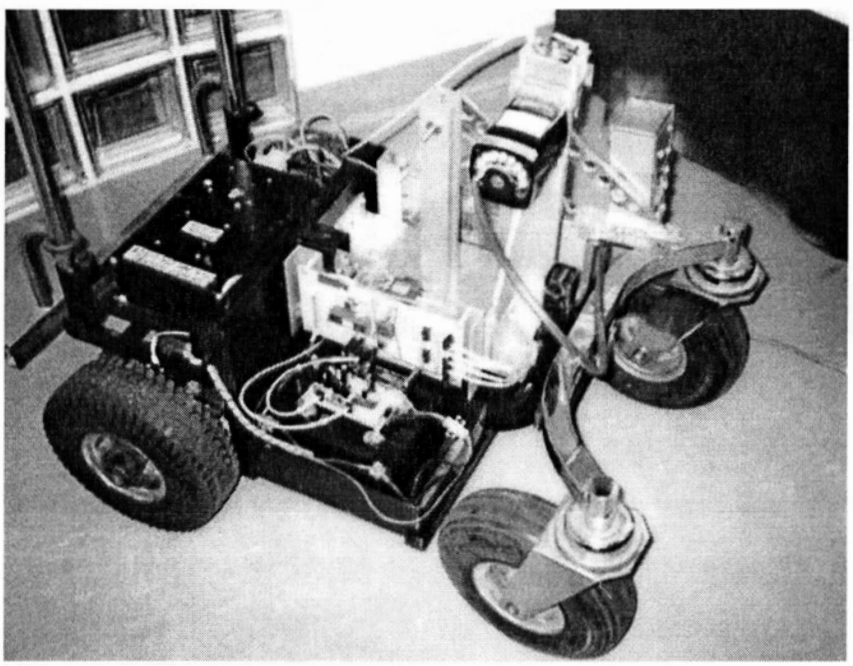


Figure 5.2: Overhead view of wheelchair chassis with LaMASS proof-of-concept.

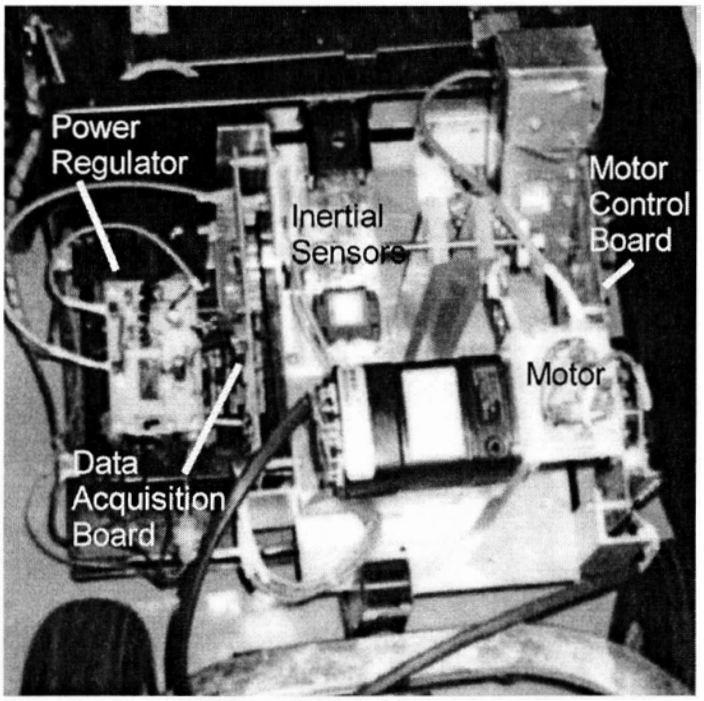


Figure 5.3: Close-up showing layout.

5.2.1 Power Supply

The power regulator board, shown in Figure 5.4 below, is responsible for generating and distributing power at the correct voltages to the other hardware components. One of the driving concepts behind the design of the power supply is that it should utilize the existing power source if possible.

The data acquisition (section 5.2.2) and motor control (section 5.2.4) boards each require regulated +12V, +5V, and -12V power at sub-amp current levels. The +5V supply is used for most of the logic circuitry. The linearly regulated ±12V supply is used primarily for op amp amplifiers and buffers in the analog conditioning circuitry. In addition, the motor driver board requires a high-current line (maximum 6A, typical 3A) at 12V, which does not require regulation.

In order to minimize the current drain by the electronics, a National Semiconductor LM2825N-5.0 integrated power supply was chosen to provide +5V. The LM2825N-5.0 is a switching regulator which contains all necessary circuitry except for a couple of external capacitors on board, allowing for a very small form factor. Because it is a switching regulator, the LM2825N-5.0 is able to deliver +5V with well over 80% efficiency.

It is important that the regulated power lines are kept as noise-free as possible, as these lines are used as references for some data acquisition in addition to providing power. For such a purpose, it would have been preferable to perform all power regulation on each individual board. It was, however, decided that doing so was unwarranted for this proof-of-concept.

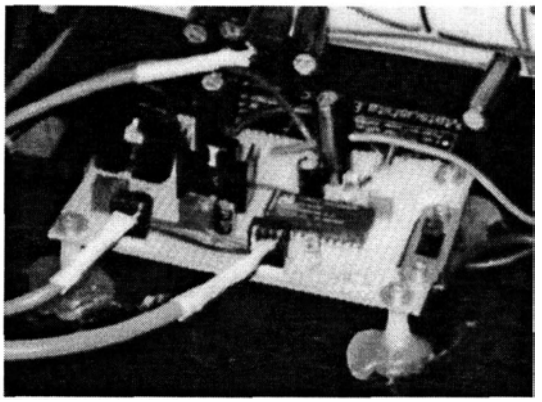


Figure 5.4: Power regulator board.

5.2.2 Inertial Sensor Platform

The inertial sensor platform consists of a 0.5-inch aluminum slab with a milled flat face to which are attached a number of inertial sensors (Figure 5.5). This slab is rigidly attached at its four corners to the LaMASS base plate which is in turn rigidly attached to the wheelchair chassis via a row of bolts and a bracket. The inertial slab, as shown in Figure 5.3, is mounted directly over this row of bolts and hence directly over the wheelchair chassis framework, as close to the middle of the wheelchair as possible.

The inertial sensor platform includes three different types of sensor for measuring the motion of the wheelchair chassis: a liquid-level tilt sensor, an X-Z accelerometer assembly, and a piezoelectric vibrating gyroscope.

The liquid-level tilt sensor is the model SSY0090 supplied by Spectron (Hauppauge, NY), and is capable of sensing up to ±45° of tilt, with an accuracy of ±5° out to 45° and ±2° out to 30°, using a ±12V supply. Data from a similar sensor shows that the sensor has a low-pass characteristic with a natural frequency of ~16 Hz. The SSY0090 is

arranged on the inertial slab to measure both pitch and roll, although only the pitch channel is currently being used for control.

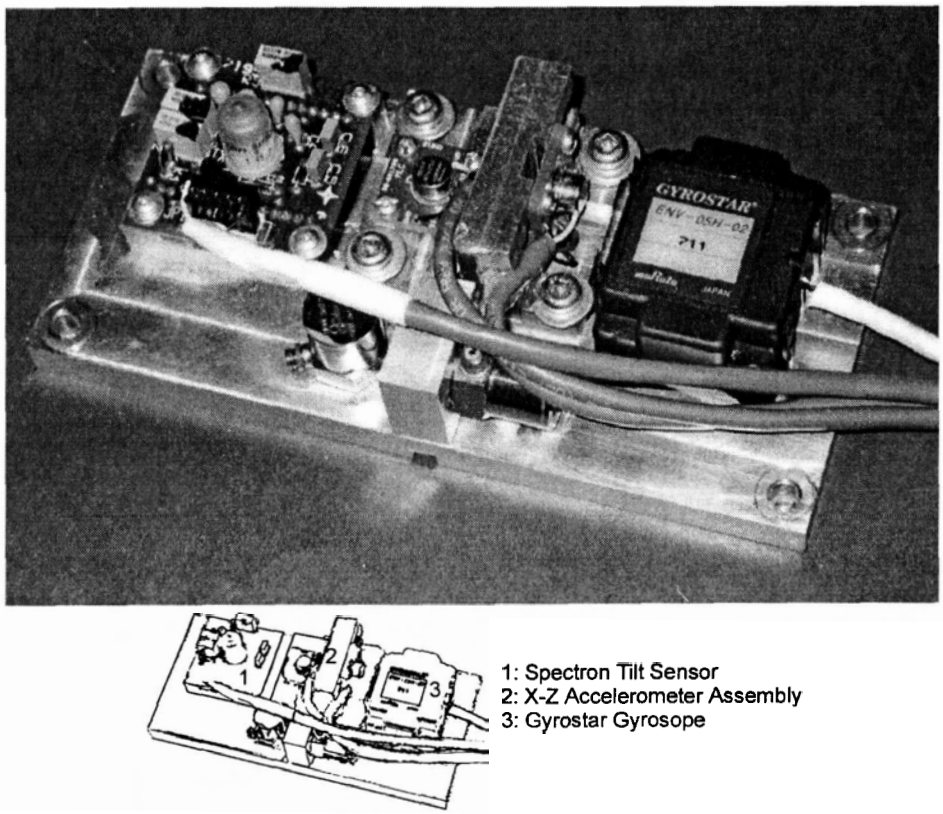


Figure 5.5: Inertial sensor platform.

The piezoelectric vibrating gyroscope is the model ENV-05H-02 Gyrostar supplied by Murata Electronics. A vibrating prism inside is used as the sensing element by measuring the phase difference of surface acoustical waves across the prism. The signal from the Gyrostar ranges from 0 to 5V, with 2.50V the output at zero angular velocity. The scale factor for the Gyrostar is rated at 22.2 mV/°/sec at room temperatures, although changing temperature can cause a drift in this scale factor with time. In fact, a drift due to self-heating (from the vibration) was observed in the Gyrostar. The Gyrostar is arranged on the inertial slab to measure pitch.

The accelerometer assembly consists of two Analog Devices ADXL-05 (±5g max) sensor packages mounted with their axes perpendicular to one another: one pointing in the X-direction, the other in the Z-direction. The assembly is made with small, thick pieces of aluminum in order to minimize resonance frequencies in the region where the sensor packages are sensitive (<100 Hz). Both sensor packages are configured to measure accelerations as slow as 0 Hz (DC). As a result, the two sensor packages together output signals proportional to the X- and Z-components of any applied acceleration. Taking the ATAN2 of the signals from these two sensors gives the angle of the applied acceleration.

The signals from these sensors are fed into the data acquisition board.

5.2.3 Data Acquisition Board

The data acquisition board (Figure 5.6) is responsible for digitizing inertial sensor data and relaying the results to the supervisory controller. An LM12H458 data acquisition system (DAS) is used to collect and digitize the buffered and scaled signals from the inertial sensors. Because its signals are relative to a ±12V supply, the Spectron tilt sensor's signals must be fed through a level shifter and scaler before going to the LM12H458, which is set up to acquire up to 8 signals between 0V and +5V. The LM12H458 handles the data gathering semi-autonomously and signals the PIC16C74A microcontroller when data is ready.

The PIC16C74A is responsible purely for interfacing between the supervisory controller and the LM12H458. For each time the supervisory controller sends a

command byte to the data acquisition board, the board decodes the command, then sends back either a single or a stream of 12-bit plus sign data, in a set specific order.

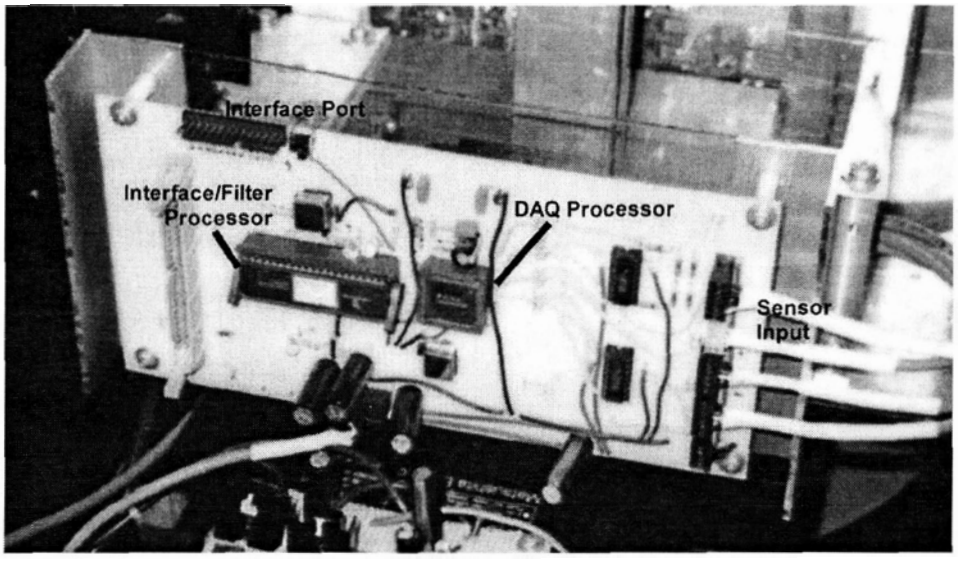


Figure 5.6: Data acquisition board.

Although some pre-filtering functions had been planned for the microcontroller, experiments showed that it was simply too slow to handle the task effectively. The code for running the microcontroller can be found in Appendix II. The PIC16C74A could probably be replaced by a lesser microcontroller, as its A/D functions aren't used in this design.

5.2.4 Supervisory Controller

The supervisory controller in the current set-up is a Pentium-P100 desktop computer running MS-DOS v6.21. It interfaces with both the data acquisition board and the motor control board via a National Instruments PC-DIO-96PnP digital interface card, which has 96 lines of digital I/O. A special 100-pin ribbon cable connector at the back of the PC-DIO-96PnP card turns into two regular 50-pin flat cable headers via a special adapter. In

the proof-of-concept system another adapter changes the 50-pin header into a 14-pin MTA connector via 15 feet of cable.

Both the data acquisition and motor control boards are plugged into the PC-DIO-96PnP at the same time, allowing the supervisory controller to use data acquired from one to control the other. The supervisory controller is also responsible for logging data to disk for post-experiment analysis.

Rather than using the rather slow NI-DAQ software library supplied with the card, direct access to the card's port is used. The code for performing this access and handling the handshaking protocol with each of the two boards is found in DIO_FACE.CPP, in Appendix IV.

5.2.5 Motor Control Board

The motor control board (Figure 5.7) is responsible for performing low-level servo control of the stepper motor which drives the LaMASS mechanism. The PIC16C74A microcontroller code which performs this task can be found in Appendix III. The microcontroller switches the stepper's phases in a software-determined manner based on the commands it receives.

Eight lines from the microcontroller run to eight optoisolators, as is shown in Appendix I. These lines switch a much larger voltage via the IGBT transistors to control 0.75A of current. The optoisolators keep the delicate electronics isolated from the large voltage swings as the motor is stepped around a circle. The two halves of the board have totally separate boards.

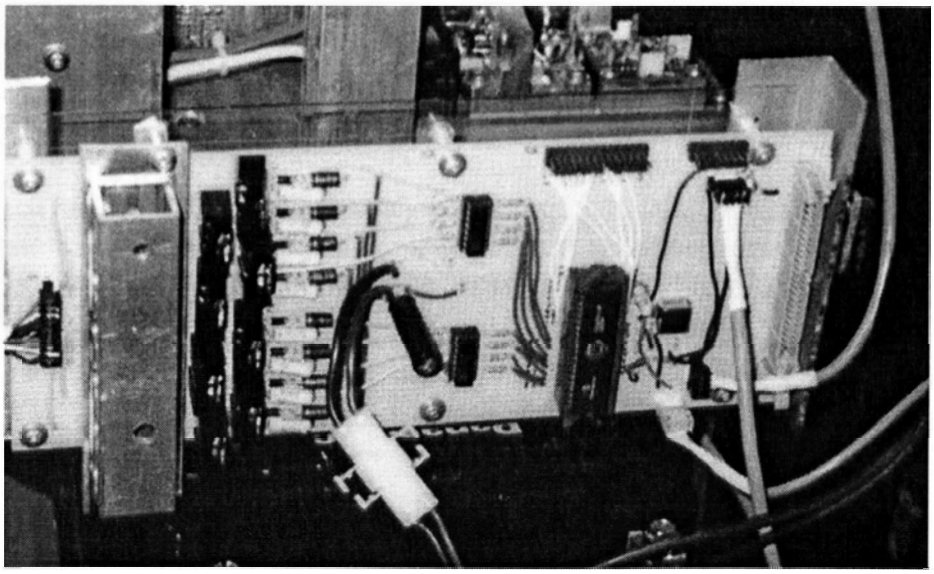


Figure 5.7: Motor control board

For feedback, a signal from the motor position board is fed into one of the PIC16C74A's analog input lines, where it is converted into an 8-bit number. As well, an optical sensor package bouncing light off an encoder pattern attached to the right rear drive wheel sends pulses to the microcontroller through one of its capture pulse ports as the drive wheel spins. By timing the inter-pulse interval, it is possible to provide a rough speed estimate. This feature, though, is currently disabled due to lack of microcontroller processing time.

Each time the supervisory controller sends a 16-bit motor position command followed by a 16-bit step rate to the motor controller, the motor controller sends back up a 16-bit motor position index equal to the sum of the previous 32 8-bit motor position numbers, which creates in effect a moving-average filter. The effective range of the motor position index is from about 8000 near the back end of the tilt range to almost 0 at the front end.

5.2.6 Motor and Motor Position Board

A stepper motor was chosen as a transition to using a DC brushless motor. Geared DC brushed motors were rejected as they suffer from problems of wear and reliability. Essentially, the fewer contacting parts, the better.

The motor chosen is an 8-phase Responsyn HDM-150-2000-8 stepper motor (Figure 5.8) with 2000 steps per revolution, for a step size of 0.18°. The stepper motor is rated for a holding torque of ~1 N·m. at the maximum current of 0.75 A. Maximum torque is generated for four active phases [49], and so the motor's phases are almost always in one of eight different configurations, shown in Table 5.2 below, in order to give maximum torque as the motor rotates. Rotation of the motor is achieved by sequentially changing the motor's step pattern from one pattern to the next.

The motor is operated in an open-loop style, in which the eight phase engagement patterns are stepped through at a set rate without checking to see if the motor has actually hit the last position or not, trusting that for the given load torque, the step rate is slow enough to ensure that the motor moves properly.

Table 5.2: Bit pattern of eight different step patterns

	Bit Pattern							
0	0	0	0	0	1	1	1	1
1	0	0	0	1	1	1	1	0
2	0	0	1	1	1	1	0	0
3	0	1	1	1	1	0	0	0
4	1	1	1	1	0	0	0	0
5	1	1	1	0	0	0	0	1
6	1	1	0	0	0	0	1	1
7	1	0	0	0	0	1	1	1

A motor position board with a MAX6250 precision +5V reference scales and offsets a signal from a potentiometer connected in-line with the motor shaft to provide position feedback. The MAX6250 provides a stable, steady voltage at the top end of the potentiometer relative to ground, which results in a clean, consistent signal. The offset and gain are set in order to provide almost the full 0-5V swing allowed by the PIC16C74A's A/D converter as the mechanism tilts from one extreme to the other.

As mentioned previously, the motor position index which is sent back to the supervisory controller is a 16-bit number which consists of the sum of the previous 32 8-bit samples of the signal from the motor position board. Assuming that the signal from the motor position board is either noise-free or else noisy with the addition of a zero-mean random process, the summing of 32 samples is designed to improve the signal-to-noise ratio and increase the position bit resolution, as there are approximately 500 steps in the full allowed range of travel for the mechanism, but only 256 different voltage levels in the full scale 8-bit A/D.

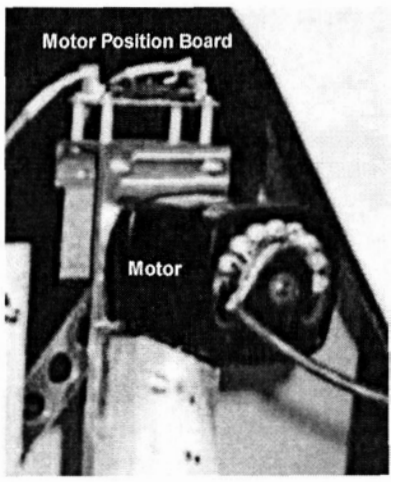


Figure 5.8: Detail of stepping motor and motor position scaling board.

5.3 Closed-Loop Motor Control

It is possible to obtain reasonable results from the lowest-level motor controller using a fairly simple bang-bang type of position controller with feedback, similar to the open-loop stepper motor controllers illustrated in [49] and [50] in which the motor is simply stepped for a precalculated number of steps to get to the desired position, assuming no slippage. For this controller, the current motor position index is compared with the commanded motor position index in order to determine when to stop. The sign of the comparison is used to determine which direction to step the motor. If the two values are within a deadband of each other, then the motor is stopped and the current step pattern is held. The deadband chosen is 20, equivalent to about 1.5 steps. In the presence of an external loading torque, the actual position of a stepper motor under open-loop control will have some static position error θ_e according to [49]:

$$\theta_e = \frac{\sin^{-1}\left(-\frac{T_L}{T_{PK}}\right)}{p} \qquad (5.1)$$

where T_L is the load torque, T_{PK} is the peak static torque, and p is the number of rotor teeth. For the stepper motor used, there are 2000 steps and 8 phases, so $p = 250$. Although the actual angular amount may be small due to the large number of rotor teeth, equation (5.1) implies that the actual position of a stepper motor may be as far as 2 step angles away from where the step is centered.

Although a stiffer control would have been possible using a fully closed-loop feedback controller, it was not possible to implement such a controller due to a combination of excessive noise and an overly-fine step angle. Using an algorithm to vary

the step rate allows the motor to be operated near the edge of its response characteristic with little additional overhead, as the maximum torque which can be supplied decreases as the step rate increases. Figure 5.9 shows a graph of some experimentally-derived effective minimum step intervals (in units of 51.2 μs) as a function of motor position index, and the four-rule fuzzy function which smoothly interpolates between these values. A fuzzy-logic algorithm was chosen as it allows for a close, smooth fit to several points with a fairly low computational cost. Although the current supervisory controller is implemented on a Pentium-100 desktop PC, it is desired to replace the desktop PC with a more inexpensive microcontroller for future development, thus motivating the search for computationally inexpensive control algorithms.

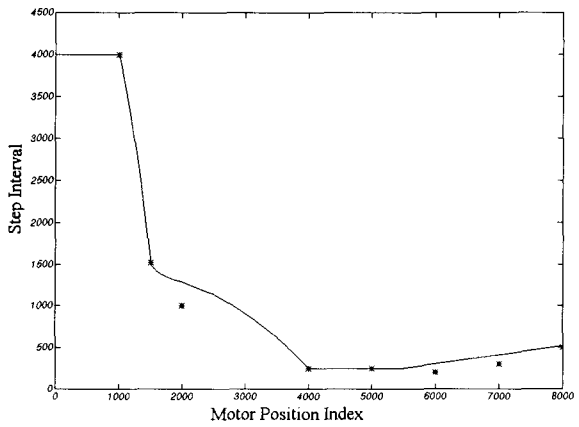


Figure 5.9: Experimental minimum step intervals and fuzzy approximating function

The fuzzy-logic algorithm uses the simple Takagi-Sugeno model [51] of rules R^i:

$$\text{Ri: if } x \text{ is } A^i \text{ then } y \text{ is } B^i \tag{5.2}$$

Linguistically, the four rules were:

If *motor position index* is **tiny**, then *step interval* is **huge**.

If *motor position index* is **small**, then *step interval* is **large**.

If *motor position index* is **medium**, then *step interval* is **small**.

If *motor position index* is **big**, then *step interval* is **medium**. (5.3)

Figure 5.10 shows the trapezoidal fuzzy membership functions A^i which define the linguistic notions of tiny, small, medium, and big for motor position index. Small, medium, large, and huge step interval were defined to be crisp numbers b^i belonging to the ordered set {250, 520, 1500, 4000}. The calculation used to obtain the inferred value, $\hat{y}$, is given by equation (5.4):

$$\hat{y} = \frac{\sum_{i=1}^{4} A^i(x) b^i}{\sum_{i=1}^{4} A^i(x)} \tag{5.4}$$

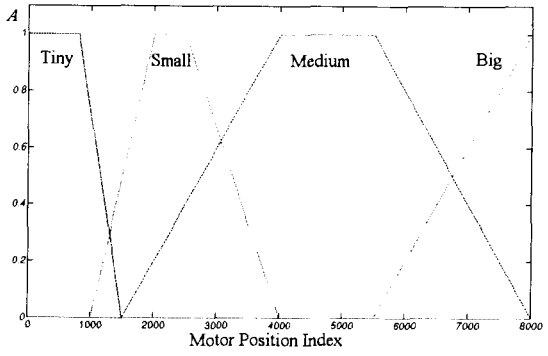


Figure 5.10: Fuzzy membership functions

5.4 Supervisory Control

The supervisory controller used here is fairly simple, as the main purpose of this set of experiments was to investigate the usage of the inertial sensors. The basic scheme of the controller was to command the motor to tilt the mechanism directly to the pitch angle calculated from the inertial sensors, essentially a P-type controller even though the error

in position is never calculated for control purposes. Control was tried using each of the inertial sensors singly as the feedback and with a fusion of all three sensors, the results of which are presented in the next chapter.

From the data acquisition board, the inertial sensor data came in the form of 16-bit signed words (of which only 12 plus the sign are significant). To put the data into a standard form, each of these signed integers was multiplied by the factor (5 / 4096), which yields the voltage of the signal. Because the data is really signed information which has been placed into the 0-5V range by the A/D process, each sensor's zero-tilt offset was subtracted away from its value.

At the beginning of each control trial, the wheelchair is meant to sit on a level surface so that the controller can get current information on the zero-tilt offset of each sensor, which has been observed to drift slightly with time and temperature, and as the batteries drain. By recalibrating at the beginning of each control run, it is possible to avoid having one set of fixed zero-tilt offsets which would need minor adjustments with each run.

The scaling laws used to convert the sensor data to obtain the tilt are:

$$\theta_A = \frac{180°}{\pi} \tan^{-1}\left(\frac{A_z - A_{xoff}}{A_x - A_{xoff}}\right) \tag{5.5}$$

$$\theta_G = \sum \frac{(G_y - G_{yof})}{0.0222\text{V}/°/\text{sec}} \Delta t \tag{5.6}$$

$$\theta_S = (S_y - S_{yoff})28.0°/\text{V} \tag{5.7}$$

where A_x, A_z, G_y, and S_y are the accelerometer (X & Z), gyroscope, and tilt sensor signal strengths in volts, and A_{xoff}, A_{zoff}, G_{yoff}, and S_{yoff} are the corresponding zero-tilt offsets.

Notice that, in equation (5.5), the x-accelerometer offset is subtracted from the z-accelerometer data. This is done because the x-accelerometer is free of the DC bias of gravity which is seen in the z-accelerometer's signal.

The 0.0222 V/°/sec constant in equation (5.6) is the typical conversion value for the Gyrostar at room temperature. Unfortunately, it was not possible to complete the jig for characterizing angular rates to sufficient quality to perform an actual calibration of the inertial sensors. The 28.0°/V constant in equation (5.7) was derived from cross-checking the results from the accelerometers. Because both accelerometers run off the same power supply and their corresponding measurements are divided into one another in equation (5.5), any fluctuations in the power supply would tend to cancel out, leaving the calculated angle in equation (5.5) subject to errors in only alignment and process control between acceleration sensors. Thus, the accelerometer-derived angle should be the most accurate of the three.

The input signals are run through a moving average filter of length 64 before the tilt calculation in order to get rid of high frequency components in the signal.

Having calculated the tilt angle required to level out the seat, the next step is to calculate the required motor position index. Table 5.3 below lists a few observations correlating motor position index to tilt angle. A linear interpolation function is used to calculate the motor position index required. This number, along with the step interval calculated by the fuzzy logic algorithm, is then sent to the motor control board.

In the case of individual sensor control, only one of θ_A, θ_G, or θ_S at a time is used to calculate the required motor position index. For the case of combined sensor control, a rather naive mean of all three angles is used.

Table 5.3: Motor position index and observed seat tilt angle

Motor Position Index	Tilt Angle (deg)
7343	-20
6686	-10
4881	0
1558	10
5	20

Chapter 6

Results

A number of trials were undertaken in order to determine the behaviour of the levelling system as the wheelchair was driven onto and off of a test ramp with inclination 10°. For all of these experiments, the lightweight human model was strapped in place. The primary purpose of these trials was to collect data on wheelchair dynamic behaviour and to compare the results of the different inertial sensors.

6.1 Motor Controller Characterization

The first test performed was to examine the signal coming from the motor positioning board which the motor control microcontroller converts to an 8-bit number. In a series of trials, the motor was locked from moving by energizing all phases. The raw output of the motor position board was then recorded to disk. A typical stream of such 8-bit bytes recorded over 0.45 seconds is plotted in Figure 6.1. In order to increase the resolution of the signal, groups of 32 samples were averaged together and the resulting signal plotted in Figure 6.2. Assuming that the noise in the signal is white and zero-

mean, using the information from more than one sample should allow us to increase the signal-to-noise ratio and allow us to increase the resolution of the quantized signal. It is, however, clear from both Figures 6.1 and 6.2 that such is not the case. Indeed, plotting the histogram of Figure 6.2 in Figure 6.3, we see what appears to be two distributions: a main one centered at 156.08 and a much smaller one centered at 155.88. Examination of data from other runs confirms this behaviour, although the location of the small secondary distribution moves around within ±0.1.

Examination of Figure 6.1 suggests that the secondary distribution is solely the result of the large negative data spikes which can be seen below the main mass of the data. The magnitude of the negative data spikes range up to -176mV (5V = 255 raw motor position). It is not obvious what causes this effect, although it is consistent over time. Locking of the motor would tend to rule out mechanical causes, leaving some unknown source in the electronics somewhere.

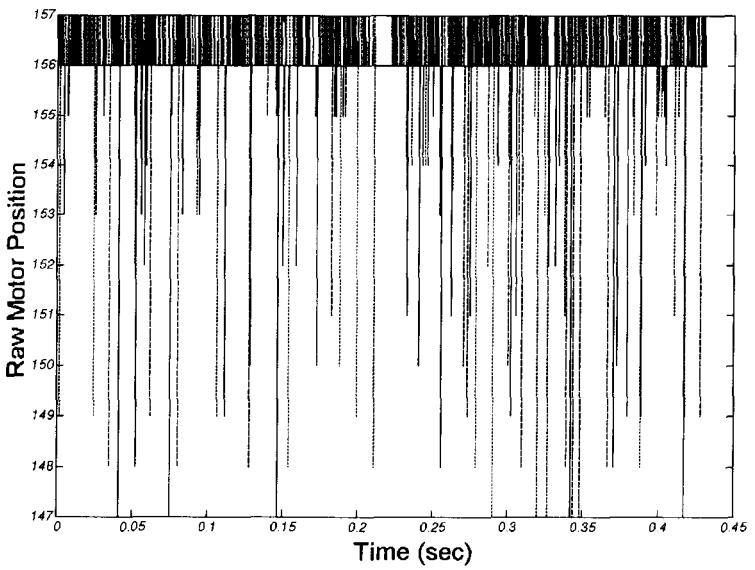


Figure 6.1: Graph of raw motor position data, mechanism locked.

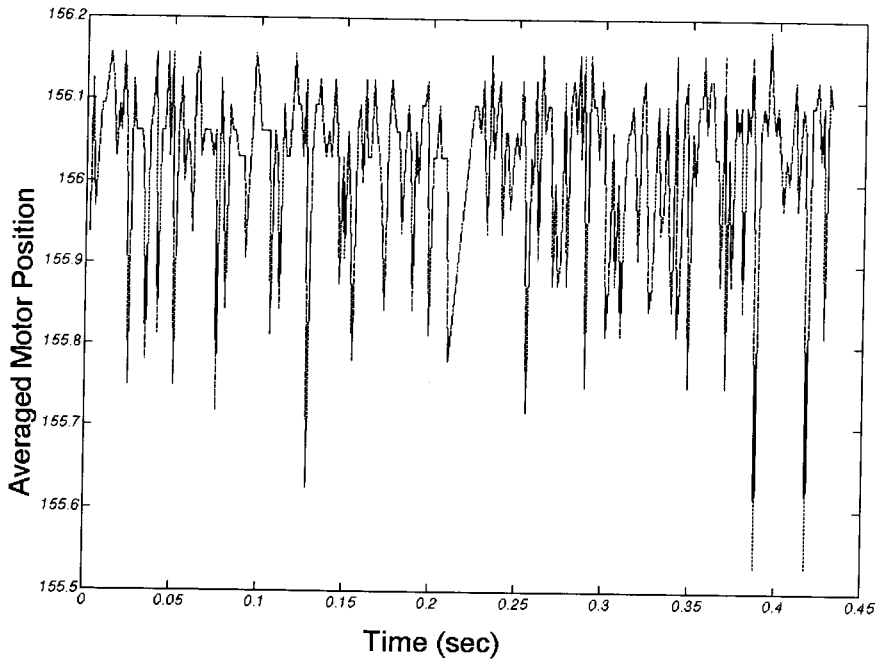


Figure 6.2: Filtered raw motor position for locked motor

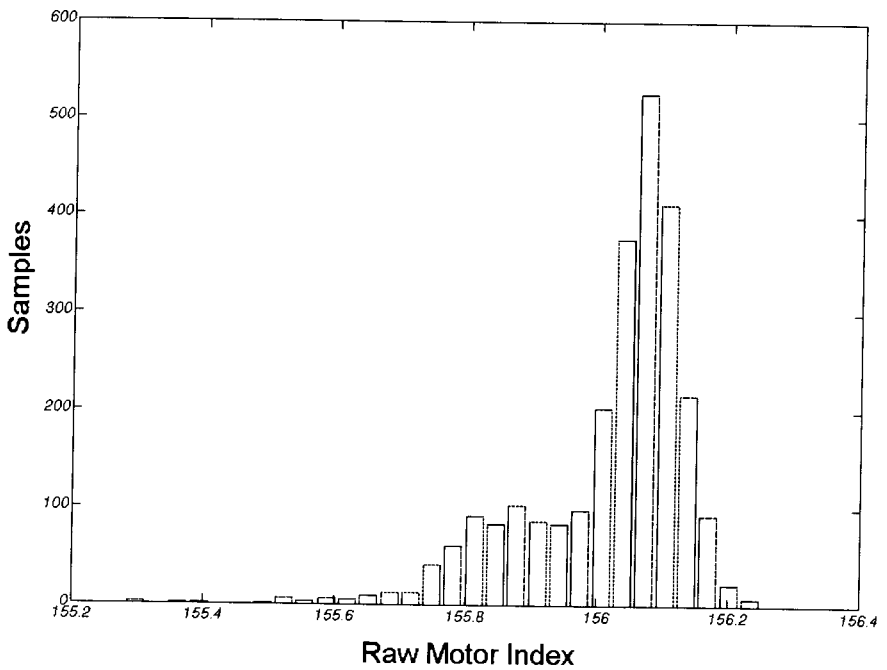


Figure 6.3: Histogram of filtered raw data.

The next experiment which was tried was an attempt to characterize the motor's behaviour in the frequency domain for sinusoidal input. One such result, for a sinusoid of frequency 0.1 Hz, is shown in Figure 6.4 below. The solid line is the recorded motor position index, while the dotted line is the commanded input position. There is clearly a great resemblance here. As the frequency increases, though, the signal quickly degrades. Figure 6.5 shows the results for a sinusoid of frequency 1.7 Hz. Note the gaps plotted in the command motor position trace, corresponding to time in which no data was recorded, as the logging and control program wrote the data from memory to disk for storage. The continuity of the other trace recording actual motor position is an artifact of the plotting process, as no data was actually recorded in between the bursts. Clearly, the motor is slewing as it hurries to catch up to the motor command, even though it never quite does.

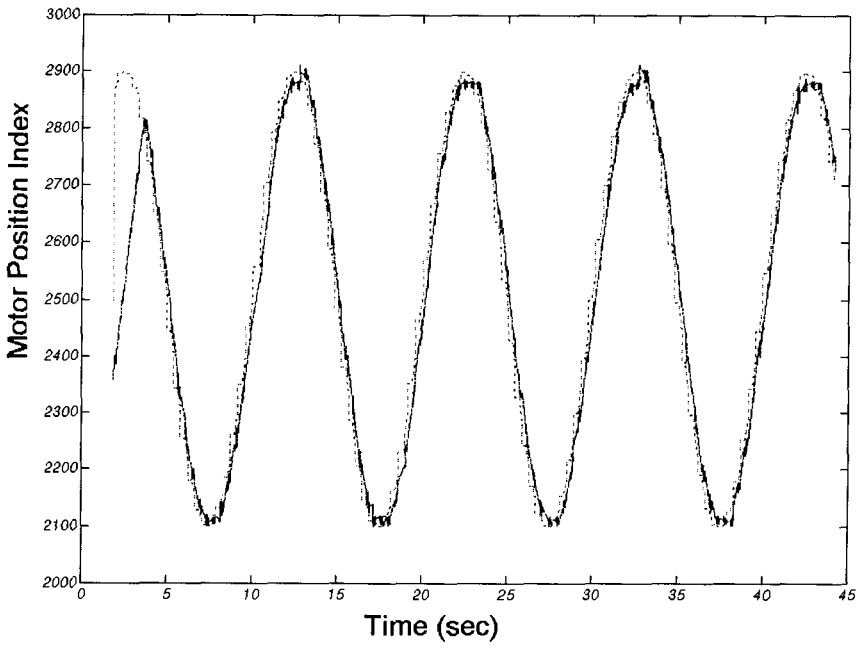


Figure 6.4: Comparison of input motor command to output motor position at 0.1 Hz

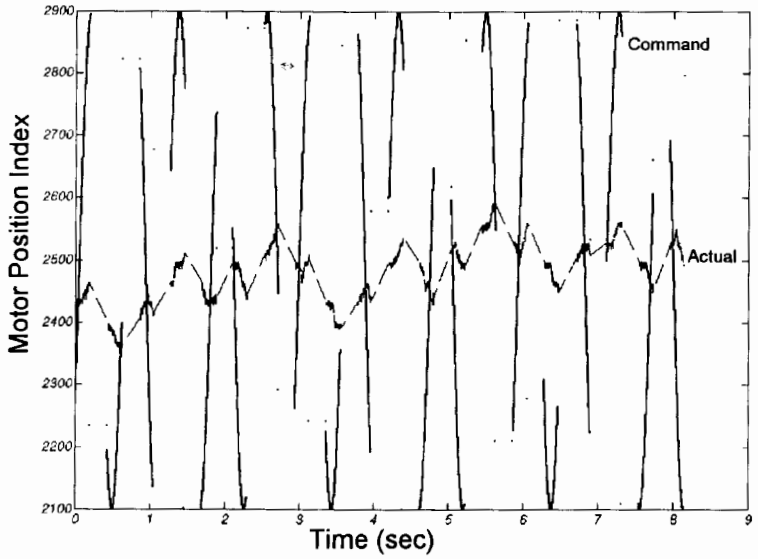


Figure 6.5: Comparison of input motor command to output motor position at 1.7 Hz

Based on this data, a plot of the frequency characteristic, of sorts, can be produced, as in Figure 6.6. The veracity of the graph for higher frequencies is suspect, though, because of the obvious nonlinearities in the system. Nonetheless, we can conclude with some confidence that the closed-loop motor servo loop has a lowpass characteristic with a break frequency just a little bit above 0.1 Hz.

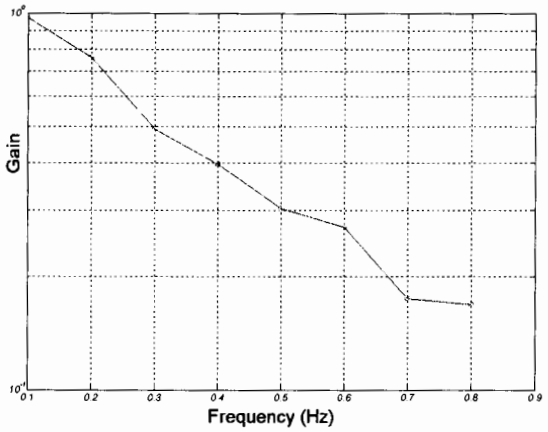


Figure 6.6: Frequency transfer characteristic of motor driver

6.2 Accelerometer Control

As noted in section 5.4, one of the control methods was to use data from just one sensor type as the feedback information. Figure 6.7 is a plot comparing input (dotted line) and output (solid line) motor position for such a control. In this experiment, the wheelchair was driven partially up the ten-degree ramp (t=10 sec) at which point it was stopped (t=20 sec) and then backed back down the ramp (t=25 sec), stopped (t = 35 sec), then driven slowly back up again. In the motor position index graphs, decreasing motor position index indicates a tilting forward of the seat-driver system relative to the wheelchair, as in Figure 6.8.

For small motions, the motor servo has no problem following the command, although the servo loop does slew, lagging seriously behind in a couple of places, most noticeably at the 30 second mark. There does, however, seem to be excessive high-frequency components present in the command signal, indicating that perhaps insufficient filtering was applied.

As Figure 4.8 suggests, though, increasing or decreasing tilt angle away from the neutral position for the LaMASS mechanism increases the input torque required. If the tilt is enough, then small jostling can cause the motor to lose its hold and slip, as in Figure 6.9. This figure shows the motor slipping suddenly at 35 sec and again at 39 sec after it has made its way back to where it should be, despite the slowing of the motor step rate. At these angles, the motor is right at the limit of static peak torque as far as input torque required to move the seat is concerned.

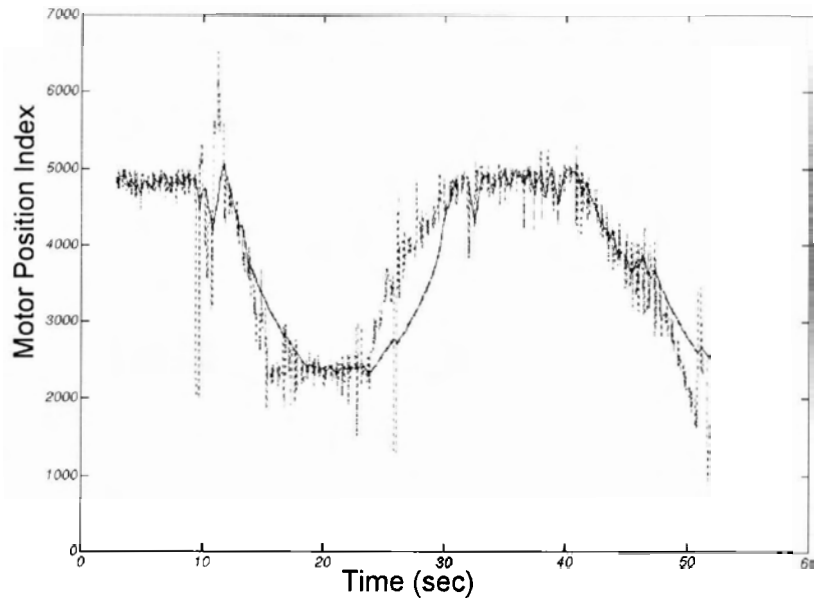


Figure 6.7: Comparison of motor command and output for accelerometer control

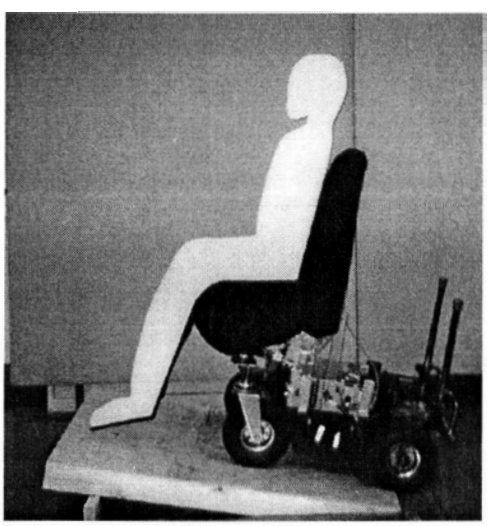


Figure 6.8: Wheelchair going up slope showing compensatory tilt

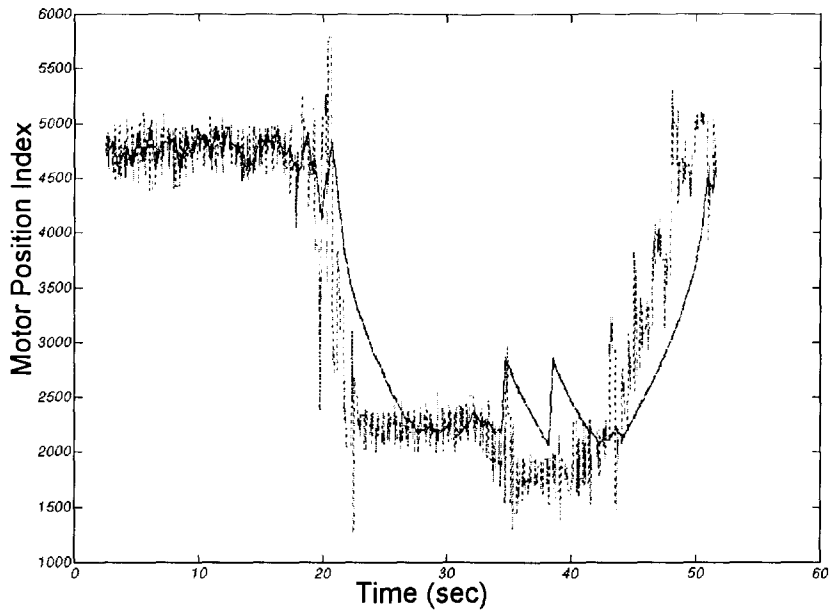


Figure 6.9: Comparison of motor position and command, trial B

6.3 Gyroscope Control

The second set of experiments relied on the gyroscope to provide the feedback information on tilt. As seen in Figure 6.10, the motor command trace (dotted) is very smooth, allowing the motor position (solid) to follow easily. Similar problems as encountered in the previous section with high forward tilt angles causing slippage were recorded here also. The graph in Figure 6.10 is from a run where the angle was not pushed as high and the wheelchair was driven more carefully to avoid jostling.

The gyroscope control is the smoothest of all the control types because the process of integration is itself a lowpass procedure. The disadvantage to this smoothness, though, is that the controller is too slow to handle the ramp at normal speeds.

Note that the mechanism does not return at the end to the original position, even though the wheelchair in reality is sitting back on level ground. To understand what

happened, we can look at Figure 6.11, which shows inertially-derived tilt angle data from all three inertial sensors for the trial in Figure 6.10. It is immediately obvious that although the calculated angle from both the accelerometers and the tilt sensor match very closely, the gyroscope-derived angle trace matches in shape only, and instead parallels the accelerometer-derived angle, which *does* return to 0°.

To understand this phenomenon, we look at Figure 6.12, which shows the gyroscope rotational rate data from which the rotational displacement of Figure 6.11 is calculated. At 7 seconds, where the gyroscope-derived angle data trace diverges from the other two, we see that there is a spike in the trace in Figure 6.12. This corresponds to the impulse felt when the wheelchair first hits the leading edge of the ramp. Due to fabrication limitations, this leading edge does not narrow down continuously to a knife edge, but instead terminates abruptly, leaving a small 0.5-inch vertical curb for the wheelchair to mount before moving along the ramp. The spike in Figure 6.12 shows the wheelchair mounting this vertical lip. Zooming in, we get Figure 6.13, which has, like Figure 6.5, been plotted to show during which time periods no data was collected due to disk access by the supervisory controller PC. It seems apparent from this graph that the gyroscope trace would inevitably diverge, as we have clearly lost the high-frequency information which would have shown up in the gaps between recording. In addition, even without this programming artifact, other observations by the author as well as others [52] suggest that self-heating effects in the gyroscope cause the rate gain constant to drift with time, which would also lead to an offset in the integrated signal.

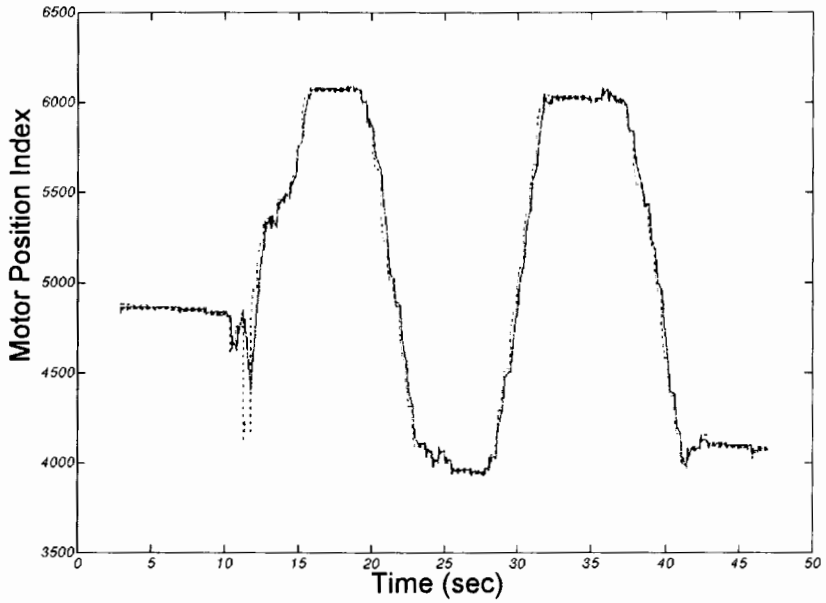


Figure 6.10: Comparison of motor input and output, for gyroscope control

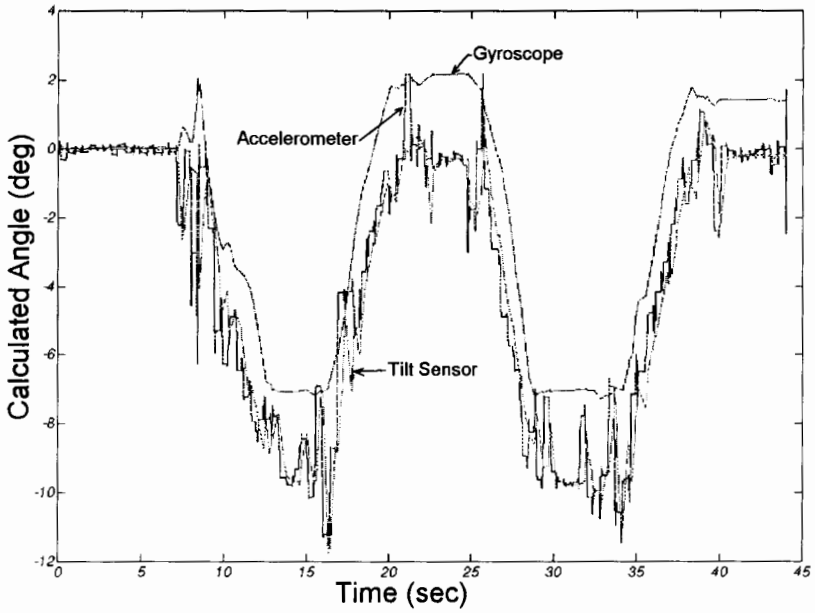


Figure 6.11: Inertial data, for gyroscope control

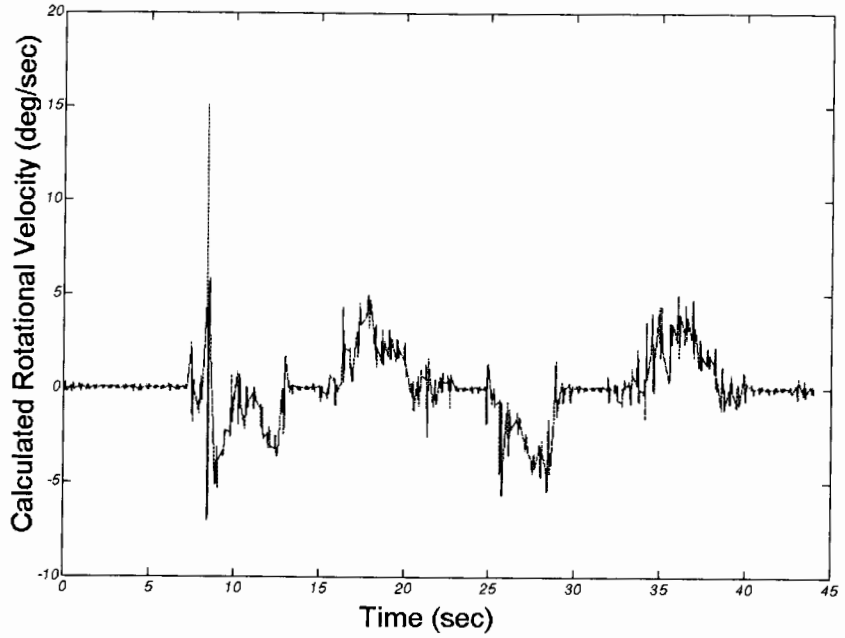


Figure 6.12: Inertial data showing gyroscope-derived rotational rate

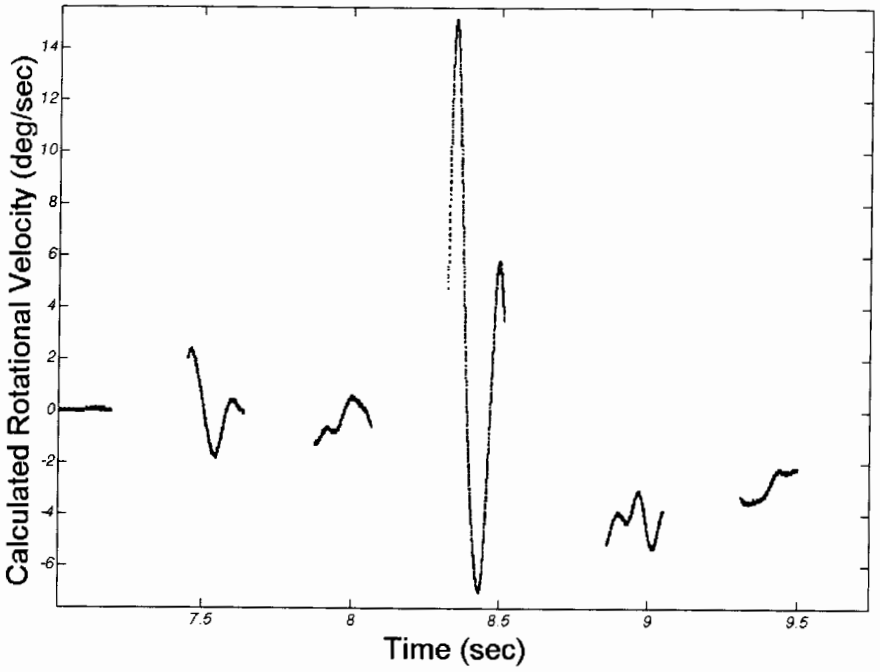


Figure 6.13: Inertial gyroscope data showing details, for gyroscope control

6.4 Tilt Sensor Control

The third single-sensor control uses information from the tilt sensor. As with the other two sets of experiments, the wheelchair was started out on level ground, then driven slowly up and down the ramp. In addition, another set of experiments was conducted to test the controller's behaviour in tilting the seat back. This was accomplished by the simple expedient of starting the wheelchair on the ramp so that zero would be reset there, then driving forwards down the ramp, as in Figure 6.14. The resulting motor position (solid) and command (dotted) plot is shown in Figure 6.15.

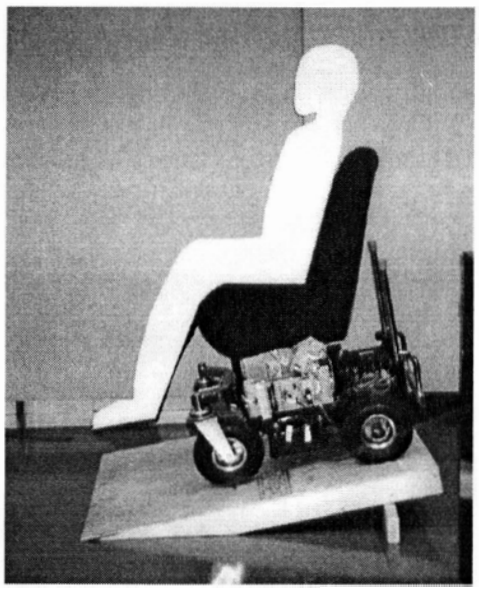


Figure 6.14: Wheelchair going down ramp, under tilt back compensation

The two big spikes observed in the motor position at 10 sec and 32 sec correspond to the shock excitation of the wheelchair's front wheel contacting the ground at the bottom of the ramp. Looking at Figure 6.16 and comparing traces suggests that the severity of the spike is an artifact from liquid movements inside the tilt sensor. Another interesting

observation is that, for these graphs, the gyroscope-calculated angle matches very closely the other two traces despite the shock excitation.

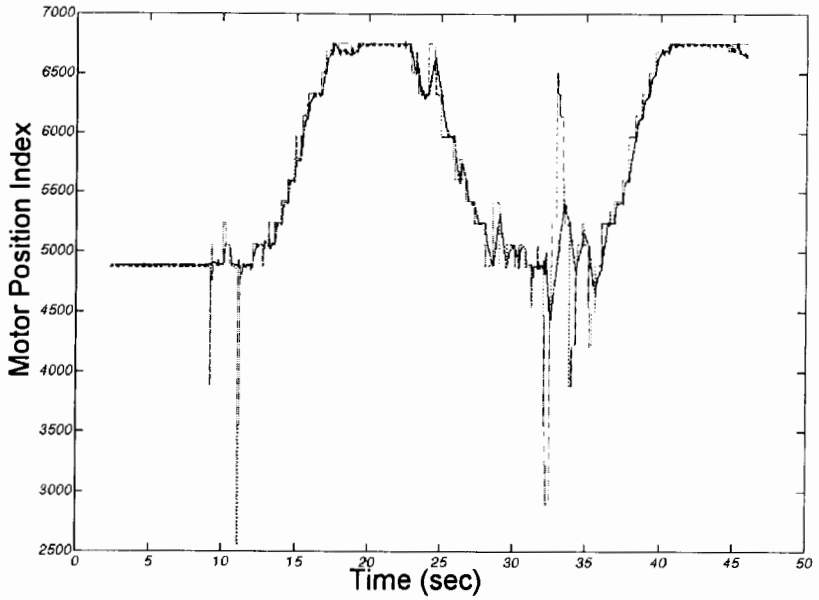


Figure 6.15: Comparison of motor input and output, for tilt sensor control

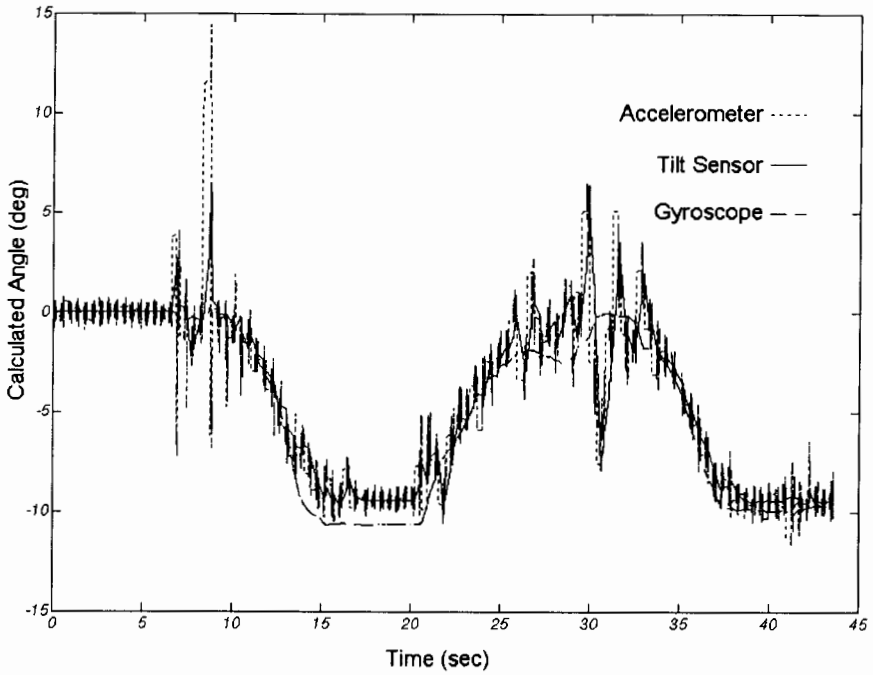


Figure 6.16: Inertial data comparison, for tilt sensor control

6.5 Combined Sensor Control

Finally, the average of angles calculated from all three sensors was used as the feedback. Again, the wheelchair was driven both up and down the ramp, starting alternately on level ground and tilted on the ramp. A graph comparing motor position and command is shown in Figure 6.17. Clearly, this trace is superior in most ways to the single-sensor data, as the command trace both shows some higher-frequency details indicative of fast rise time and is smoother, with less noise than in, for example, the command trace of the accelerometer-based controller, allowing the motor to more easily track the control.

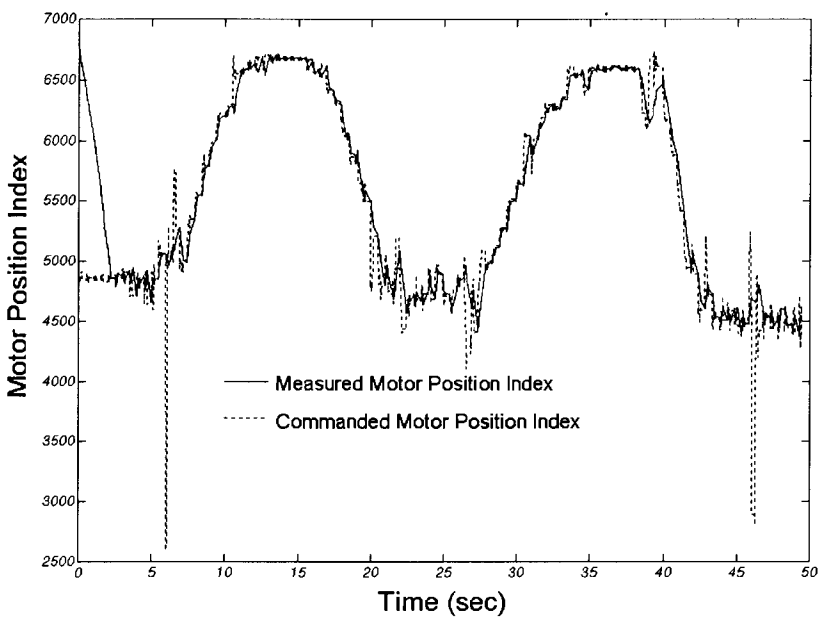


Figure 6.17: Comparison of motor position data for combined sensor control

As Figure 6.18 shows, though, the occasional glitch in the accelerometer and tilt sensor data still shows up as a result of shock excitation which usually contains too little

energy to move the person in the chair, but, lowpass-filtered by the data acquisition system and the supervisory controller, is enough to cause LaMASS to jostle alarmingly.

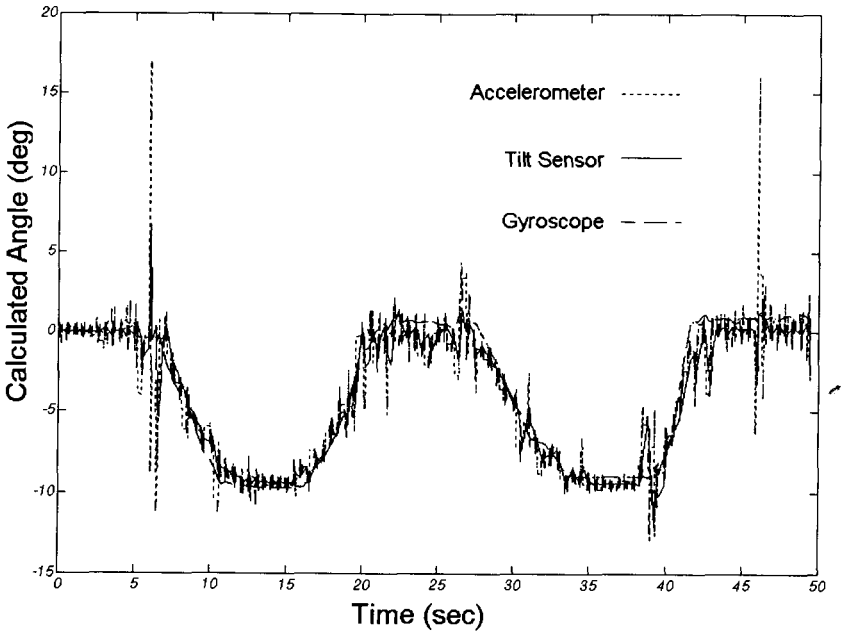


Figure 6.18: Inertial data, combined sensor control

6.6 Summary

Four different control approaches were attempted, using each of the inertial sensor types singly and all together. Both the accelerometer and tilt sensor data demonstrated excessive noise, suggesting that heavier filtering may provide a good solution. The gyroscope provided smooth data, but extremely lowpassed and too slow for use without some form of prediction. Additionally, offsets in the gyroscope data caused as a result of both programming artifacts and gyroscope-intrinsic effects cause the integrated rotational rate data to drift over time as well. The combined data control seemed to work best, combining aspects of all three.

Chapter 7

Conclusions and Future Work

In this chapter, the conclusions of the work completed on the LaMASS proof-of-concept are drawn, and suggestions for future work on the next phases are made. The suggestions discussed include: mechanical redesign, controller redesign, and use of semi-active vibration isolation elements.

7.1 Conclusions

A proof-of-concept for the Large Motion Active Suspension System (LaMASS) for powered wheelchairs has been completed and tested. The LaMASS proof-of-concept device uses inertial data from a small inertial package mounted on the wheelchair chassis to calculate chassis pitch information. The pitch angle is then used directly to direct a servo control loop to reposition the lightweight seat mock-up.

The system is able to compensate for chassis pitch for slow speeds and small angles less than about 10° by using a suitably lowpassed sensor such as the gyroscope. Speeding up the controller, though, appears also to introduce a great deal of noise to the

output, noise which results in the mechanism rarely, if ever, staying still even when not moving. The origin of the noise points towards the sensors, although whether the noise is intrinsic to the sensors, is a true mechanical vibration, or is a result of fluctuations in the power supply is not clear.

Compared to the raw data from either the accelerometer or the tilt sensor, the gyroscope has the quietest signal. Although the tilt sensor also has a quiet signal, the Spectron sensor suffers from long settling time (up to 500 msec) which can lead to problems when the mechanical excitation input is of high frequency or shock character. Both the gyroscope and accelerometers appears to be able to handle shock excitation well, quickly settling down immediately after the shock, allowing for the possibility of analog or digital filtering to take care of the signal conditioning without having to worry about the sensor's frequency characteristics.

The accelerometers seem to be the noisiest of the three sensor sets. This may, however, be a fabrication problem associated with the design of the electronics used, as the buffer for the signals is located about 2 feet of cable away from the sensor package itself. The sensor package is, furthermore, in an unshielded carrier, which may promote noise pickup. Analog Devices also sells the same sensor pre-packaged in both one-, two-, and three-axis models. Using these pre-packaged sensors, combined with better cable shielding, may solve most of the noise problem.

A program has been written in MATLAB to analyze this mechanism's kinematic and static attributes, to calculate the torques and forces required to move the wheelchair seat through a particular path. This analysis allows the use of a simple genetic algorithm optimization technique to optimize the design. For any other mechanical design, the

technique is simple enough that, given equations describing the dynamics, it would be easy to optimize a design using the same technique.

7.2 Future Work

In the following sections, some design areas which will be investigated in the next phases of the project are discussed.

7.2.1 Hardware Design Considerations

This section discusses a few of the criteria which need to be considered in the development of a full-sized self-levelling suspension based on the work in this thesis, but which did not play a large role in the design of the proof-of-concept hardware due to the difference in payload. (mock-up vs. real live human being)

7.2.1.1 Safety and Reliability

Safety and reliability are the two primary design factors when dealing with any assistive device: the former because of the device's proximity to a human being, the latter due to the constant use (and occasional abuse) such a device receives. Wheelchairs typically remain in near-constant use throughout the user's waking day, whether as a means of transportation or simply as a seat to sit on.

The chattering observed for the accelerometer-driven control and the offset drift of the purely gyroscope-driven control are both unacceptable in a real system. Especially the accelerometers and tilt sensor are sensitive to sudden shocks. A better control system is obviously needed.

7.2.1.2 Power Consumption

Virtually all powered wheelchairs in use today are powered by electric batteries, most often of the 12V lead-acid deep-discharge type. The unsealed type of lead-acid battery, such as those found in most automobiles, are referred to as wet cells, whereas sealed lead-acid batteries are referred to as gel cells. The primary purpose of these batteries is to provide energy to drive the motors. The motors on a typical powered wheelchair can collectively sink as much as 20-30A [53]. For scooters and power-base type wheelchairs, running out of power will essentially render the individual immobile until the batteries can be recharged. Consequently, any device which draws electrical power from the main batteries should contain some sort of cut-off to prevent drawing additional power when the batteries are in a low-charge condition.

For this reason, the Invacare Tarsys comes with its own battery pack to power its own motors. As pointed out earlier in section 2.4.2, moving a full-sized person around demands a great deal of power, comparable to that required by the main motors themselves. The disadvantage of this design is that an additional 50 lbs. of batteries alone is added to the weight of the wheelchair, which can have an adverse affect on the wheelchair's range.

7.2.1.3 Size

Currently, the electronics are sprawled out, taking far more space than they need to. A simple consolidation of the boards into a single board, combining the two microcontrollers into one, would save even more space, with the added bonus of reducing the overhead spent on external data communications protocols.

7.2.2 Addition of Instrumentation Sensors

Currently, there is no instrumentation on the seat which would measure the effectiveness of LaMASS as either a vibration isolation or self-levelling system. The addition of sensors on the seat itself would allow better verification of the experimental results, as calculation of seat accelerations from motor position data is subject to error from mechanical backlash between the input link 2 and the output link 3 of the LaMASS mechanism which the seat is rigidly attached to.

7.2.3 Use of Larger Human Mass Model

The next step in the development is to increase the power scale to allow for a design which would move something closer to human scale. The size and power considerations mentioned in a previous section would then also come to the fore.

7.2.4 Extension to Spatial Mechanism

The next natural step is to change the mechanical configuration to allow combined pitch-roll motions. One possible model is a 3-dof Stewart platform, whose kinematics and dynamics are already well-known [54,55]. Switching to the Stewart platform configuration would also suggest the use of prismatic, translational linkages, such as those described in the next section.

7.2.5 Utilizing a Semi-Active Suspension

One thing which is clear is that a totally active suspension system is very costly in terms of power consumed. One method of decreasing the energy cost is by using a semi-

active system, in which a controller actively modifies the passive damping properties of structural elements in the suspension mechanism [35]. Because no energy is expended in moving the payload platform, a significantly smaller amount of energy is required.

Such actively-controlled damping elements can be fluid or solid, although solid damping elements tend to suffer from wear problems. Many semi-active suspension systems employ a fluid damping element whose damping ratio is controlled via valves.

Electro- and magneto-rheological fluids provide a safe, reliable, and energy-efficient method of changing the damping properties of a linear or rotational damping unit [56]. In an electro-rheological fluid, an applied electric field causes suspended particles in the fluid to align with one another, increasing the viscosity and hence the damping factor. A magneto-rheological fluid is similar, but utilizes a magnetic field to vary its viscosity.

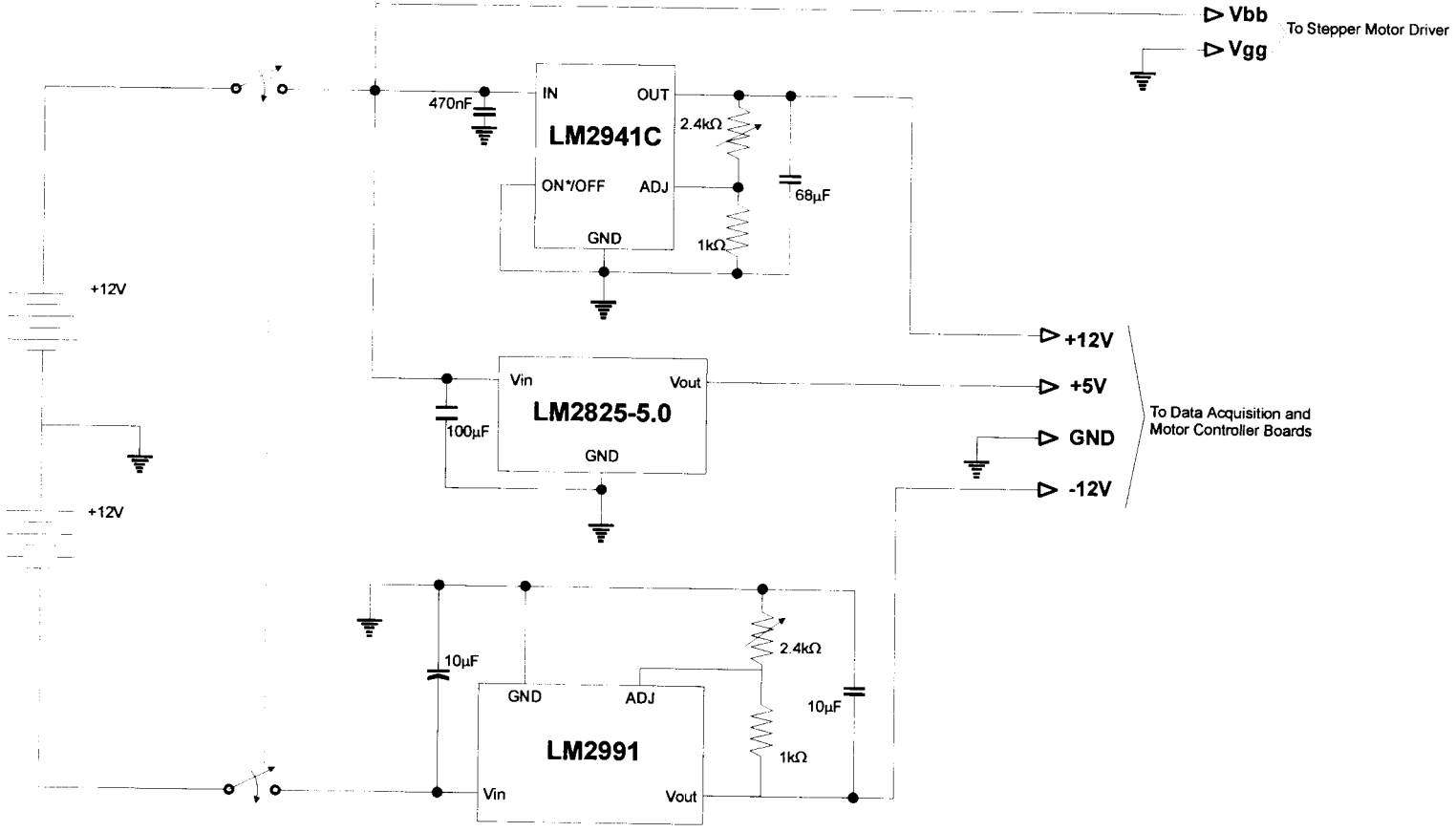
Recently, Lord Corporation (Cary, NC) has introduced a line of magneto-rheological fluid-based damping elements under the trade name of Rheonetic™ Magnetic Fluid Systems. Lord's magneto-rheological fluids are 20 to 50 times stronger than electro-rheological fluids, and require much lower voltages in order to generate the magnetic fields. With a millisecond response time and a high yield stress (up to 90 KPa), this product makes a suitable candidate for an semi-active damping element. Indeed, Lord's product brochure [57] describes a tractor seat active damping system which has been designed using its product already.

Appendix I

Electronic Schematics

This section contains schematic diagrams for the three main electronic subsystems of the LaMASS proof-of-concept referred to in section 0: power supply, data acquisition board, and motor controller board.

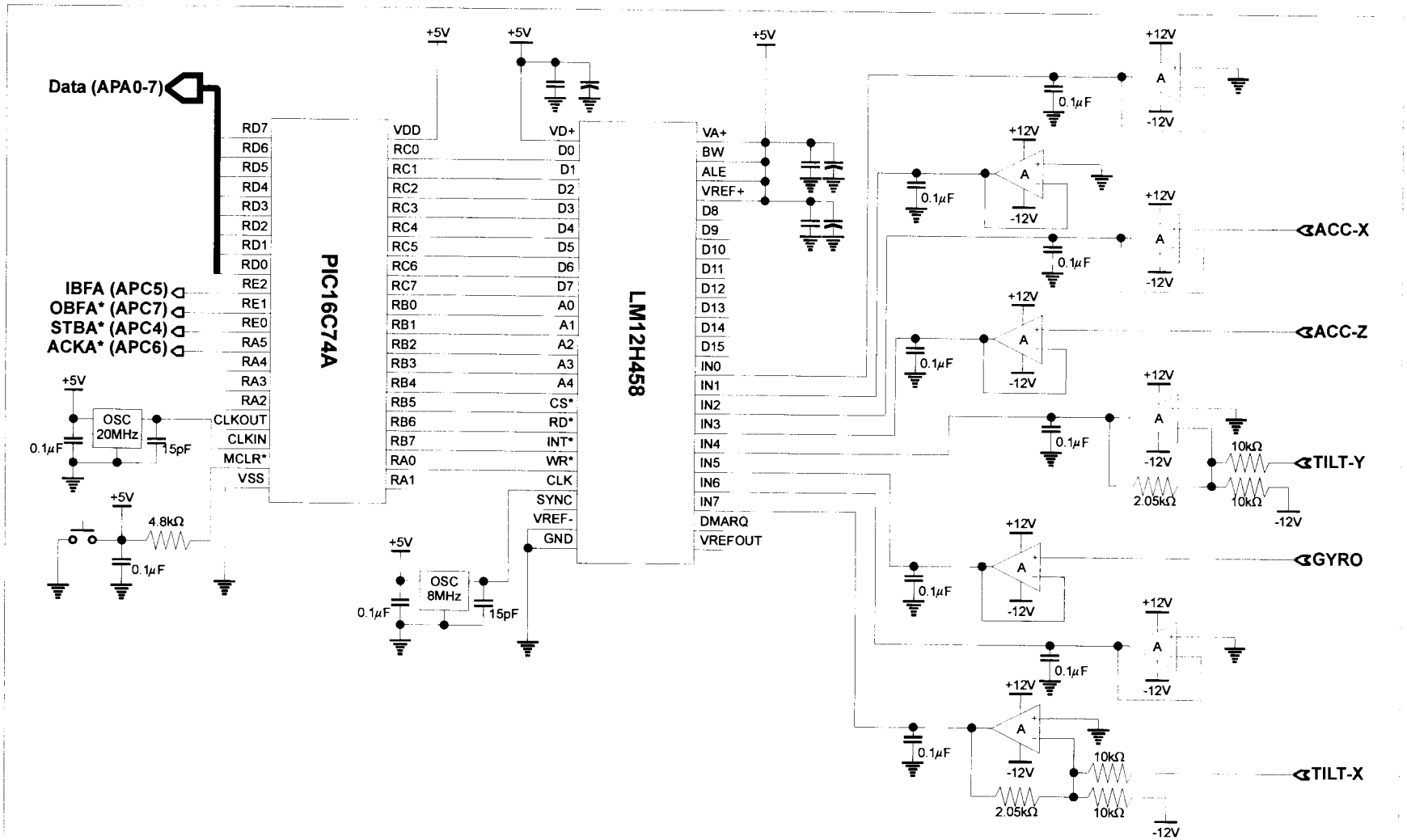
The power supply board supplies power to each of the other two boards via a keyed, 4-conductor Panduit Mas-Con style connector. Both the data acquisition and motor controller boards have two alternate data interfaces, each connected to one another. The supervisory PC can connect to these boards via either a 50-connector locking 3M style ribbon cable connector or through a 14-conductor Mas-Con connector.



LaMASS Power Supply Board

William Li
School of Engineering Science, Simon Fraser University

rev. 1.0
June 25, 1997

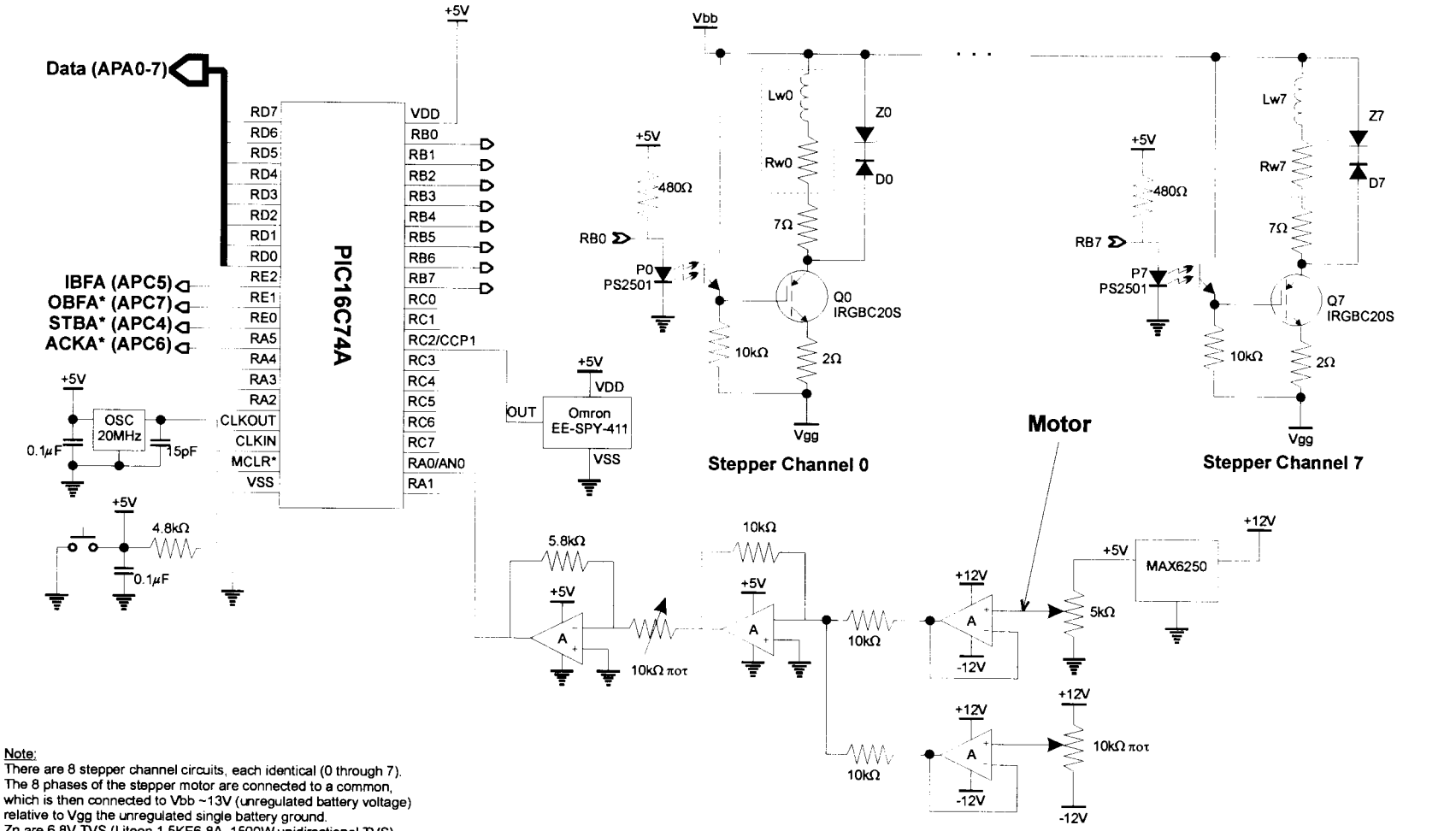


Notes:
A = TL074
OSC = CTS clock oscillator (half-size)

LaMASS Data Acquisition Board

William Li
School of Engineering Science, Simon Fraser University

rev. 1.0
June 25, 1997



Note:
There are 8 stepper channel circuits, each identical (0 through 7).
The 8 phases of the stepper motor are connected to a common, which is then connected to Vbb ~13V (unregulated battery voltage) relative to Vgg the unregulated single battery ground.
Zn are 6.8V TVS (Liteon 1.5KE6.8A, 1500W unidirectional TVS)
Dn are ultra fast switching rectifiers UF1004DICT-ND
Lwn is nominally 24mH
Rwn is nominally 10Ω
Omron EE-SPY-411 is a self-contained reflective optical sensor (light-on) with built-in amplifier, used to calculate wheel speed.

LaMASS Motor Control and Positioning Subsystems

William Li
School of Engineering Science, Simon Fraser University

rev. 1.0
July 6, 1997

Appendix II

Code for Acquiring Data

The following code was written and implemented on a Microchip PIC16c74A microcontroller as shown in Appendix I for interfacing with both the LM12H458 data acquisition chip and with the supervisory PC controller. Upon reset, the code resets and restarts the LM12H458, then waits for a request for data from the supervisory PC controller, at which point the most current data is sent back to the PC. A timed interrupt is responsible for gathering data from the LM12H458 at an effective rate of just under 8 kHz.

The include file referred to, <p16Cxx.inc>, is a standard header file published by MicroChip which defines the labels for all the status registers on each microcontroller.

```
      LIST    p = 16C74,  f = INHX8M, n = 66
;
;*****************************************************************
;
; Author: William W. Li
; Date:   August 8, 1997
;
; This program implements reading and writing to the LM12H458 Data
```

```
; Acquisition System chip.
;
; Connections to the LM12H458:
; ----------------------------
;     LM12H458          16c74
;     --------          -----
;     2-9   D0-7  <-->  RC0-7 Data bus, 8-bit
;     19    RD*   <---  RB6   Read enable, active low
;     20    WR*   <---  RA0   Write enable, active low
;     21    CS*   <---  RB5   Chip select, active low (halts LM12H458)
;     31    INT*  --->  RB7   interrupt request, active low
;     24-28 A0-4  <---  RB0-4 address lines
;
; The READ_DAS subroutine reads the memory location of the LM12H458
; which is located at the 5-bit address contained in the REGLMADDR
; register upon invocation of the routine.  (the upper 3 bits are
; masked out)  Upon return, the W register contains the 8-bit piece of
; data contained at that address.
;
; The WRITE_DAS subroutine writes the 8-bit contents of the W register
; to the memory location in the LM12H458 at the address pointed to by
; the REGLMADDR register.
;
; This program tests the parallel port handling abilities of the
; PIC16c74, implementing an interface to the OKI 82C55A CMOS
; programmable peripheral interface in Mode 2 (bidirectional), used on
; the National Instruments PC-DIO-96/PnP board.  Port A of the 82c55a
; is connected to port D of the 16c74, and is used for the
; bidirectional data transfer.  Other control lines which are connected
; are as follows:
;
;     82c55a                  16c74
;     ------                  -----
;     C7    OBFA* --->  RE1   output buffer full flag
;     C6    ACKA* <---  RA5   acknowledge
;     C5    IBFA  --->  RE2   input buffer full flag
;     C4    STBA* <---  RE0   strobe
;     C3    INTRA --->  n/c   interrupt request
;     port A (0-7)      <-->  port D (0-7)
;
; Write Exchange:
; ---------------
; The 82C55A sends a high-to-low transition on the OBFA* (RE1) line,
; which signals that a piece of data is ready for reception by the
; 16c74.  The 16c74 then sends a high-to-low transition on the ACKA*
; (RA5) line, at which point (up to 150ns after this transition) the
; 82c55a will put the data onto the bus at Port D.  Before this, the
; 82c55a's output buffers are tristated into a high-impedance state.
; ACKA* must be low for a minimum of 100 ns.  OBFA* will rise a maximum
; of 150 ns after ACKA* falls.  Port A will be return to a floating
; condition between 20 ns and 250 ns after the rising edge of ACKA* is
; detected.
;
; Read Exchange:
; --------------
; The 82c55a waits for the 16c74 to signal a high-to-low on the STBA*
; (RE0) line, which must be down for a minimum of 100 ns.  The data
```

```
; which is put out onto port D by the 16c74 will be latched upon the
; rising edge of STBA*, and must have been present at the port for a
; minimum of 20 ns before the rising edge of STBA*.  The data must
; remain on port D for a minimum of 50 ns after the rising edge of
; STBA* to allow the 82c55a to latch the data properly.  IBFA goes high
; a maximum of 150 ns after STBA* falls and indicates that data has
; been fetched into the input latch.
;
; Mod as of Aug 8: 2 bytes, in low-byte, high-byte order are put back
; out after the initial write.
;
;*****************************************************************
;
;
; HARDWARE SETUP
;       None
;
;
            INCLUDE <pl6Cxx.inc>
;
        __CONFIG ( _CP_OFF & _WDT_OFF & _HS_OSC & _PWRTE_OFF )
;
;*****************************************************************
;*****       Register definitions
;*****************************************************************
;
; REG0 through 7 hold the filtered values of the results from the A/D
; conversion. Each register is a 2-byte, 16-bit register, although only
; the lower 12 bits (0-11) are ever used, in practice.  The upper 4
; bits store sign data, and will be either all 1 or all 0.  The bytes
; marked "L" are the LSB, the ones marked "H" are MSB.  Each register
; corresponds to a particular channel of the LM12H458 (0 through 7).
; For the current implementation, channels 0, 1, and 6 are not used,
; and have their inputs shorted to ground.  The connections of the
; channels are:
;     Ch 0: Strain Gauge connector pin 3 (grounded via connector)
;     Ch 1: Strain Gauge connector pin 4 (grounded via connector)
;     Ch 2: ADXL "X"
;     Ch 3: ADXL "Z"
;     Ch 4: Spectron "Y"
;     Ch 5: Gyrostar
;     Ch 6: N/C (grounded at buffer)
;     Ch 7: Spectron "X"
; This set of registers is what the external controller reads when it
; requires data.  This table is updated at regular intervals by the
; same interrupt routine which gets data from the LM12H458.
;
REGSTART    EQU   H'30'       ; REGSTART is the same as REG0L
  CBLOCK    REGSTART
            REG0L, REG0H
            REG1L, REG1H
            REG2L, REG2H
            REG3L, REG3H
            REG4L, REG4H
            REG5L, REG5H
            REG6L, REG6H
            REG7L, REG7H
```

```
      ENDC
;
;
  CBLOCK    H'20'
            TEMPW               ; W storage used by interrupt handler
            TEMPSTATUS          ; STATUS storage used by int handler
            TEMPFSR             ; FSR storage used by interrupt handler
            TEMPTRANS           ; swap register used exclusively by TRANS
            TEMPREAD
            SHADA               ; Shadow register for RA
            SHADB               ; Shadow register for RB
            SHADC               ; Shadow register for RC
            SHADD               ; Shadow register for RD
            SHADE               ; Shadow register for RE
            REGLMADDR           ; Read/Write address from/to LM12H458
            REGTABPTR           ; pointer to register filter table
            ACCaLO
            ACCaHI
            ACCbLO
            ACCbHI
      ENDC
;
REG2START   EQU   H'50'
REG3START   EQU   H'58'
REG4START   EQU   H'70'
REG5START   EQU   H'60'
REG6START   EQU   H'68'
REGSUMSTART EQU   H'40'
;
;
;*************************************************************************
;*****      Register redirections
;*************************************************************************
;
;
PORTLED     EQU   PORTA         ; 2 LEDs exist for test purposes
TRISLED     EQU   TRISA         ; located at A2 and A4
SHADLED     EQU   SHADA
;
;
PORTLM      EQU   PORTC         ; redirects lines to appropriate ports
PORTRDLM    EQU   PORTB         ; for controlling LM12H458
PORTWRLM    EQU   PORTB
PORTCSLM    EQU   PORTA
PORTINTLM   EQU   PORTB
PORTADDRLM  EQU   PORTB
;
TRISLM      EQU   TRISC         ; redirects lines to appropriate ports
TRISRDLM    EQU   TRISB         ; for controlling LM12H458
TRISWRLM    EQU   TRISB
TRISCSLM    EQU   TRISA
TRISINTLM   EQU   TRISB
TRISADDRLM  EQU   TRISB
;
SHADLM      EQU   SHADC         ; redirects lines to appropriate shadow
SHADRDLM    EQU   SHADB         ; registers for controlling LM12H458
SHADWRLM    EQU   SHADB
```

```
SHADCSLM    EQU   SHADA
SHADINTLM   EQU   SHADB
SHADADDRLM  EQU   SHADB
;
;
PORTPAR     EQU   PORTD       ; redirects lines to appropriate ports
PORTOBF     EQU   PORTE       ; for parallel port interface to PC-DIO96
PORTACK     EQU   PORTA
PORTIBF     EQU   PORTE
PORTSTB     EQU   PORTE

TRISPAR     EQU   TRISD       ; redirects lines to appropriate ports
TRISOBF     EQU   TRISE       ; for parallel port interface to PC-DIO96
TRISACK     EQU   TRISA
TRISIBF     EQU   TRISE
TRISSTB     EQU   TRISE

SHADPAR     EQU   SHADD       ; redirects lines to shadow registers
SHADOBF     EQU   SHADE       ; for parallel port interface to PC-DIO96
SHADACK     EQU   SHADA
SHADIBF     EQU   SHADE
SHADSTB     EQU   SHADE
;
;
;*****************************************************************************
;*****      Constant definitions
;*****************************************************************************
;
; Common Constants
; ----------------
allTx       EQU   H'00'       ; disable all tristate buffers
allRx       EQU   H'FF'       ; enable all tristate buffers
;
; Test LED Constants
; ------------------
led0out     EQU   H'02'
led1out     EQU   H'03'
led2out     EQU   H'04'
;
; Parallel Interface Handshaking Constants
; ----------------------------------------
ackpar      EQU   H'05'       ; ACK* line is RA5
stbpar      EQU   H'00'       ; STB* line is RE0
obfpar      EQU   H'01'       ; OBF* line is RE1
ibfpar      EQU   H'02'       ; IBF line is RE2
;
; LM12H458 Interface Constants
; ----------------------------
cslm        EQU   H'00'       ; CS* line is RA0
readlm      EQU   H'06'       ; RD* line is RB6
writelm     EQU   H'07'       ; WR* line is RA7
intlm       EQU   H'00'       ; INT* line is RB0
addr0lm     EQU   H'01'       ; ADDR0-4 lines RB1 through RB5
addr1lm     EQU   H'02'
addr2lm     EQU   H'03'
addr3lm     EQU   H'04'
addr4lm     EQU   H'05'
```

```
;
; LM12H458 Memory Addresses
; -------------------------
; Note that 'L' designates low byte on the 16-bit registers, while 'H'
; designates high byte.  Each register is 16-bits wide (except the
; instruction RAM registers, which are really 48 bits wide, selectable
; via the RAM pointer in the configuration register).  The 0th bit A0
; selects between low and high bytes.  On 16-bit data bus systems, A0
; is a don't care.
;
insram0L    EQU   B'00000'    ; Instruction RAM 0 through 7
insram0H    EQU   B'00001'
insram1L    EQU   B'00010'
insram1H    EQU   B'00011'
insram2L    EQU   B'00100'
insram2H    EQU   B'00101'
insram3L    EQU   B'00110'
insram3H    EQU   B'00111'
insram4L    EQU   B'01000'
insram4H    EQU   B'01001'
insram5L    EQU   B'01010'
insram5H    EQU   B'01011'
insram6L    EQU   B'01100'
insram6H    EQU   B'01101'
insram7L    EQU   B'01110'
insram7H    EQU   B'01111'
;
configregL  EQU   B'10000'    ; Configuration Register
configregH  EQU   B'10001'
interegL    EQU   B'10010'    ; Interrupt Enable Register
interegH    EQU   B'10011'
intstatregL EQU   B'10100'    ; Interrupt Status Register
intstatregH EQU   B'10101'
timerregL   EQU   B'10110'    ; Timer Register
timerregH   EQU   B'10111'
fiforegL    EQU   B'11000'    ; Conversion FIFO register
fiforegH    EQU   B'11001'
limitregL   EQU   B'11010'    ; Limit Status Register
limitregH   EQU   B'11011'
;
; Register Bit Fields
; -------------------
; Configuration Register
;
IOsel       EQU   H'07'
autoZeroEC  EQU   H'06'
chanMask    EQU   H'05'
standby     EQU   H'04'
fullCal     EQU   H'03'
autoZero    EQU   H'02'
resetLM     EQU   H'01'
startLM     EQU   H'00'
;
diagLM      EQU   H'03'
testLM      EQU   H'02'
;
; Interrupt Enable & Status Registers on LM12H458
```

```
;
intWD       EQU   H'00'
intSEQlim   EQU   H'01'
intFIFOlim  EQU   H'02'
intAutoZero EQU   H'03'
intFullCalib      EQU   H'04'
intPause    EQU   H'05'
intLowSupply      EQU   H'06'
intActive   EQU   H'07'
;
; Control word bit fields
ctrlsum     EQU   H'03'
ctrlraw     EQU   H'03'
ctrlall     EQU   H'04'
ctrlsngl    EQU   H'04'
ctrlzero    EQU   H'05'
ctrlcalib   EQU   H'06'
ctrlreset   EQU   H'07'
;
;
;*************************************************************************
;*****      Start macro definitions here.
;*************************************************************************
;
; Macro for transferring data from one register to another.
; W register gets trashed
TRNTRSH           MACRO fromReg, toReg
            MOVF  fromReg, W
            MOVWF toReg
            ENDM
;
; Macro for transferring data from one register to another with no side
; effects other than the status bits, as long as nobody else uses
; TEMPTRANS
TRANS       MACRO fromReg, toReg
            MOVWF TEMPTRANS
            TRNTRSH     fromReg, toReg
            MOVF  TEMPTRANS, W
            ENDM
;
; Macro for setting bank 1 mode
SET_BANK_1  MACRO
            BSF   STATUS, RP0       ; Bank 1
            ENDM
;
; Macro for setting bank 0 mode
SET_BANK_0  MACRO
            BCF   STATUS, RP0       ; Bank 0
            ENDM
;
; Macro for setting tristate bits
SETTRIS           MACRO whichbuffer, whatvalue
            SET_BANK_1
            MOVLW whatvalue
            MOVWF whichbuffer
            SET_BANK_0
            ENDM
```

```
;
; Macro for setting tristate bits for special case all Tx (0)
SETTX       MACRO whichbuffer
            SET_BANK_1
            CLRF  whichbuffer
            SET_BANK_0
            ENDM
;
; Macro for putting address on address bus with RD*, WR* lines high
; Note that contents of W get trashed.
; ***** The 3 MSB in REGLMADDR MUST be 0 *****
PUT_ADDR_ON_BUS   MACRO
            MOVF  SHADADDRLM, W     ; First, blank out the 5 bits which
            ANDLW B'11000001'       ; make up the address field in the
            IORWF REGLMADDR, W      ; port which holds the LM address.
            MOVWF SHADADDRLM        ; Then, OR in the address field.
            MOVWF PORTADDRLM        ; Next, move result to the port.
            ENDM
;
; Macro for loading up the REGLMADDR register
; W register gets trashed
LDADDR            MACRO value
            MOVLW value
            MOVWF REGLMADDR
            BCF   STATUS, C         ; Clear carry bit
            RLF   REGLMADDR, F      ; rotate left one bit
            ENDM
;
; Macro for writing a literal out to DAS
; W register gets trashed
WRLIT       MACRO address, value
            LDADDR      address
            MOVLW value
            CALL  WRITE_DAS
            ENDM
;
; Macro for reading from DAS.  Inlined code for speed only.
; Behaviour is just like the READ_DAS routine, except that the
; result is placed in the indirect register.
; W register gets trashed.
RDDAS       MACRO
            PUT_ADDR_ON_BUS
            BCF   SHADRDLM, readlm
            TRNTRSH     SHADRDLM, PORTRDLM      ; Now, lower RD*, after
min 20 ns
;
            MOVF  PORTLM, W   ; at least 80 ns later, get data to W
            MOVWF INDF
;
            BSF   SHADRDLM, readlm
            TRNTRSH     SHADRDLM, PORTRDLM      ; re-raise RD* to
signal end of read
            ENDM
;
; Macro for lowering the CS* line
; W register gets trashed
LOWERCS           MACRO
```

```
            BCF   SHADCSLM, cslm
            TRNTRSH     SHADCSLM, PORTCSLM      ; Lower CS*
            ENDM
;
; Macro for raising the CS* line
; W register gets trashed
RAISECS           MACRO
            BSF   SHADCSLM, cslm
            TRNTRSH     SHADCSLM, PORTCSLM      ; Raise CS*
            ENDM
;
;
; Macro for enabling GIE
SETGIE            MACRO
            BSF   INTCON, GIE       ; Enable global interrupt
            ENDM
;
;*****************************************************************
;       Double Precision  Addition ( ACCb + ACCa -> ACCb )
;
M_add       MACRO
            movf    ACCaLO, W
            clrf  ACCbHI
            addwf   INDF, F               ;add lsb
            btfsc   STATUS,C        ;set up carry
            incf    ACCbHI, F
            incf  FSR, F
            movf    ACCaHI, W
            addwf ACCbHI, W         ;add in carry
            addwf   INDF, F               ;add msb
            ENDM
;
;*****************************************************************
;         Double Precision Subtraction ( ACCb - ACCa -> ACCb )
;
M_sub       MACRO
            comf    ACCaLO, F       ; negate ACCa ( -ACCa -> ACCa )
            incf    ACCaLO, F
            btfsc   STATUS, Z
            decf    ACCaHI, F
            comf    ACCaHI, F       ; then, add
            M_add
            ENDM
;
;
; Macro for adding or subtracting something to sum table
; Adds the entry pointed to at (fromtbl + contents of REGTABPTR lower
; nybble) to the sum-table entry at (REGSUMSTART + index*2).  Each
; entry is 16-bit, with low-byte going first.  The result is placed
; back in the sum table.  Whether addition or subtraction is done
; depends on the sign of which.
;
ADDTBL      MACRO fromtbl, index, which
            MOVF  REGTABPTR, W
            ADDLW fromtbl
            MOVWF FSR
            MOVF  INDF, W
```

```
            MOVWF ACCaLO
            INCF  FSR, F
            MOVF  INDF, W
            MOVWF ACCaHI
;
            MOVLW REGSUMSTART
            ADDLW (index * 2)
            MOVWF FSR

      IF (which > 0)
            M_add       ; Finally, add ACCb to ACCa
      ELSE
            M_sub
      ENDIF
            ENDM
;
;
;*************************************************************************
;*****       Start program here.
;*************************************************************************
;
      ORG   H'0000'                 ; set up start of program
      GOTO  START
;
      ORG   H'0004'                 ; set up interrupt vector
      GOTO  INTERRUPT_HANDLER ; (not implemented)
;
      ORG   H'0005'
;
START                               ; POWER_ON Reset (Beginning of program)
      CLRF  STATUS                  ; Do initialization (Bank 0)
CLRGIE0
      BCF   INTCON, GIE             ; Disable global interrupt
      BTFSC INTCON, GIE             ; Global interrupt disabled?
      GOTO  CLRGIE0
      CLRF  INTCON
;
; Clear RAM to all zeroes.
; Registers with suffix "L" hold the low byte, "H" registers hold the
; high byte.
;
      MOVLW H'20'                   ; Clear lower page
      MOVWF FSR
CLEAR_TABLE_0
      CLRF  INDF                    ; Clear memory location
      INCF  FSR, F
      BTFSS FSR, 7                  ; Keep going until we've passed 0x7F.
      GOTO  CLEAR_TABLE_0
;
      MOVLW H'A0'                   ; Clear upper page
      MOVWF FSR
CLEAR_TABLE_1
      CLRF  INDF                    ; Clear memory location
      INCF  FSR, F
      BTFSS STATUS, Z               ; Keep going until we've passed 0xFF.
      GOTO  CLEAR_TABLE_1
;
```

```
      MOVLW REG2START          ; Set up indirection table
      MOVWF REG2L
      MOVLW REG3START
      MOVWF REG3L
      MOVLW REG4START
      MOVWF REG4L
      MOVLW REG5START
      MOVWF REG5L
      MOVLW REG6START
      MOVWF REG6L

      BSF   SHADACK, ackpar          ; raise ACK* line
      BSF   SHADSTB, stbpar          ; raise STB* line
      BSF   SHADCSLM, cslm           ; raise CS* line to disable CS on
                                     ; LM12H458.
      BSF   SHADRDLM, readlm         ; raise RD* line to disable read
      BSF   SHADWRLM, writelm        ; raise WR* line to disable write
      TRNTRSH     SHADA, PORTA       ; After this initialization, we
      TRNTRSH     SHADB, PORTB       ; won't ever be referring to the
      TRNTRSH     SHADC, PORTC       ; ports by "PORTx" again.  Instead,
      TRNTRSH     SHADD, PORTD       ; each line will get its own port
      TRNTRSH     SHADE, PORTE       ; ID, e.g. "PORTRDLM".  This makes
                                     ; for heftier but more maintainable
                                     ; code.
;
; Next, ensure that all the lines are behaving as digital I/O and are
; set to read or write as appropriate.  Any line which is not actively
; used is set to Rx, or high impedance (tristated) state.
;
      SET_BANK_1
      MOVLW B'00000111'        ; set RAx and REx to digital mode
      MOVWF ADCON1

      MOVLW allRx              ; set all I/O lines to Rx first
      MOVWF TRISA
      MOVWF TRISB
      MOVWF TRISC
      MOVWF TRISD
      MOVLW B'00000111'        ; TRISE is a special case as the upper
      MOVWF TRISE              ; 5 bits are used.

      BCF   TRISRDLM, readlm   ; Set the following lines to Tx
      BCF   TRISWRLM, writelm
      BCF   TRISCSLM, cslm
      BCF   TRISADDRLM, addr0lm
      BCF   TRISADDRLM, addr1lm
      BCF   TRISADDRLM, addr2lm
      BCF   TRISADDRLM, addr3lm
      BCF   TRISADDRLM, addr4lm
      BCF   TRISACK, ackpar
      BCF   TRISSTB, stbpar
      BCF   TRISLED, led1out
      BCF   TRISLED, led2out
      SET_BANK_0
;
; Set up LM12H458 to do acquisition of 8 samples continuously
      WRLIT configregH, B'00000000' ; Set Diagnostic Mode off,
```

```
                                    ; RAM Ptr 00

      WRLIT insram0L, B'00001000'   ; Set up channel 2: ADXL "X"
      WRLIT insram1L, B'00001100'   ; Set up channel 3: ADXL "Z"
      WRLIT insram2L, B'00011000'   ; Set up channel 6: Spectron "X"
      WRLIT insram3L, B'00010101'   ; Set up channel 5: Gyrostar

      WRLIT insram0H, B'00000010'   ; Set up acq. times and precision
      WRLIT insram1H, B'00000000'
      WRLIT insram2H, B'00000000'
      WRLIT insram3H, B'00000000'

      WRLIT configregL, B'10100010' ; Reset LM12H458, stop operation

WAIT_ON_CALIB
      BTFSC PORTINTLM, intlm
      GOTO  WAIT_ON_CALIB

      LDADDR      intstatregL
      CALL  READ_DAS

      WRLIT interegL, B'00011100'   ; Set FIFO limit (INT2),
                                    ; ZERO (INT3), CALIB (INT4)
      WRLIT interegH, B'00100000'   ; Set FIFO limit to 5

      WRLIT timerregL, H'0A'  ; Set up timer for 125us period (8 kHz)
      WRLIT timerregH, H'00'

      SET_BANK_1              ; Set up interrupt handling
      BCF   OPTION_REG, INTEDG
      SET_BANK_0
      CLRW
      MOVWF INTCON
      BSF   INTCON, INTE

      WRLIT configregL, B'10100010' ; Reset
      WRLIT configregL, B'10100001' ; Start acquisition

      SETGIE                        ; Enable interrupts
      WRLIT configregL, B'10101001' ; Start full calibration

WAIT_FOR_OBF_HIGH_0     ; Wait for OBF (RE1) high-to-low transition
      BTFSS PORTOBF, obfpar
      GOTO  WAIT_FOR_OBF_HIGH_0


;*************************************************************************
;*****      Start main loop here.
;*************************************************************************
MAINLOOP

WAIT_FOR_OBF_LOW
      BTFSC PORTOBF, obfpar
      GOTO  WAIT_FOR_OBF_LOW

CLRGIE1
      BCF   INTCON, GIE       ; Disable global interrupt
```

```
      BTFSC INTCON, GIE       ; Global interrupt disabled?
      GOTO  CLRGIE1

      BCF   SHADACK, ackpar   ; clear ACK*. Leave ACK* line down for
      TRANS SHADACK, PORTACK  ; to give the computer time to finish

      TRNTRSH     PORTPAR, TEMPREAD ; Transfer parallel port contents
                                    ; to TEMPREAD

      BSF   SHADACK, ackpar         ; raise ACK* again, after which we
      TRNTRSH     SHADACK, PORTACK  ; are free to put the return value
                                    ; in D onto the bus

      SETTX TRISPAR

      BTFSS TEMPREAD, ctrlreset
      GOTO  CHECK_CALIB

      WRLIT configregL, B'10100010' ; Reset
      WRLIT configregL, B'10100001' ; Start acquisition
      GOTO  DONE_AQ

CHECK_CALIB
      BTFSS TEMPREAD, ctrlcalib
      GOTO  CHECK_ZERO

      WRLIT configregL, B'10101001' ; Start full calibration
      GOTO  DONE_AQ

CHECK_ZERO
      BTFSS TEMPREAD, ctrlzero
      GOTO  CHECK_ALL

      WRLIT configregL, B'10100101' ; Start auto-zero
      GOTO  DONE_AQ

CHECK_ALL
      BTFSS TEMPREAD, ctrlall
      GOTO  CHECK_RAW

      BCF   SHADLED, led2out
      TRNTRSH     SHADLED, PORTLED
      GOTO  DONE_AQ

CHECK_RAW
      BTFSC TEMPREAD, ctrlraw
      GOTO  GET_SUM

      MOVLW B'00000111'       ; strip away all but lower 3 bits (0-7)
      ANDWF TEMPREAD, F
      BCF   STATUS, C
      RLF   TEMPREAD, W
      ADDLW REGSTART          ; get raw data value address
      MOVWF FSR
      MOVF  INDF, W
      ADDWF REGTABPTR, W
      MOVWF FSR               ; stick the result in the FSR pointer
```

```
      GOTO  CONTINUE_AQ

GET_SUM
      MOVLW B'00000111'       ; strip away all but lower 3 bits (0-7)
      ANDWF TEMPREAD, F
      BCF   STATUS, C
      RLF   TEMPREAD, W
      ADDLW REGSUMSTART       ; add in offset to bottom of sum table
      MOVWF FSR               ; stick the result in the FSR pointer

CONTINUE_AQ
      MOVF  INDF, W           ; load the register indexed into W
      CALL  OUTPUT_DATA
      INCF  FSR, F      ; reference next value in table, high byte
      MOVF  INDF, W     ; transfer the high byte to parallel port

WAIT_FOR_IBF_LOW
      BTFSC PORTIBF, ibfpar         ; wait till IBF goes low,
      GOTO  WAIT_FOR_IBF_LOW        ; indicating the host has read the
                                    ; data, before sending more.

      CALL  OUTPUT_DATA

DONE_AQ
      SETTRIS     TRISPAR, allRx
      SETGIE
      GOTO  MAINLOOP

;*************************************************************************
;*****      Subroutines start here.
;*************************************************************************
;
READ_DAS                      ; Read REGLMADDR, return in W
      PUT_ADDR_ON_BUS
      LOWERCS
      BCF   SHADRDLM, readlm
      TRNTRSH     SHADRDLM, PORTRDLM      ; Lower RD*, after min 20 ns

      MOVF  PORTLM, 0         ; at least 80 ns later, get data to W
      MOVWF TEMPREAD

      BSF   SHADRDLM, readlm
      TRNTRSH     SHADRDLM, PORTRDLM      ; re-raise RD* to signal end
      RAISECS                             ; of read
      MOVF  TEMPREAD, 0
      RETURN

WRITE_DAS                     ; Write W to REGLMADDR location
      MOVWF PORTLM            ; Latch output data into reg C
      PUT_ADDR_ON_BUS         ; (which is still tristated)
      LOWERCS

      BCF   SHADWRLM, writelm
      TRNTRSH     SHADWRLM, PORTWRLM      ; Now, lower WR*, after
                                          ; min 20 ns

      SETTRIS     TRISC, allTx
```

```
      BSF   SHADWRLM, writelm
      TRNTRSH     SHADWRLM, PORTWRLM      ; Raise WR* to latch data at
      SETTRIS     TRISC, allRx            ; LM12H458
      RAISECS
      RETURN


;*****************************************************************
; OUTPUT_DATA
;     Outputs one byte of data to the host (PC-DIO-96), waits
;     until we're sure the host has latched the data.
;     The data byte is found in the W register.
;     This routine assumes that PORTPAR is already set for Tx.
;
OUTPUT_DATA
      MOVWF PORTPAR
      BCF   SHADSTB, stbpar         ; lower STB* line
      TRNTRSH     SHADSTB, PORTSTB

WAIT_FOR_IBF_HIGH
      BTFSS PORTIBF, ibfpar         ; loop till IBF goes high
      GOTO  WAIT_FOR_IBF_HIGH       ; indicating host has latched data

      BSF   SHADSTB, stbpar         ; re-raise STB*
      TRNTRSH     SHADSTB, PORTSTB
      RETURN


;*****************************************************************
; Interrupt Handler
;*****************************************************************
;
INTERRUPT_HANDLER
      MOVWF TEMPW
      SWAPF STATUS, W
      SET_BANK_0
      MOVWF TEMPSTATUS
      TRNTRSH     FSR, TEMPFSR

      BTFSS INTCON, INTF
      GOTO  HANDLE_OTHER

      BCF   INTCON, INTF            ; clear interrupt flag
      BSF   SHADLED, led1out
      TRNTRSH     SHADLED, PORTLED

      ADDTBL      REG2START, H'02', -1    ; subtract out oldest values
      ADDTBL      REG3START, H'03', -1    ; from sum in sum table
      ADDTBL      REG6START, H'06', -1
      ADDTBL      REG5START, H'05', -1

      LOWERCS

      LDADDR      intstatregL       ; see what kind of interrupt it is
      MOVLW TEMPREAD
      MOVWF FSR
```

```
        RDDAS

        BTFSS INDF, intFIFOlim
        GOTO  OTHER_LM_INT
;
; The data is set up to be acquired in the following order:
;       Ch 2: ADXL "X"
;       Ch 3: ADXL "Z"
;       Ch 4: Spectron "Y" (no longer being acquired)
;       Ch 6: Spectron "X"
;       Ch 5: Gyrostar
; Data is acquired in a low-byte, high-byte order and placed
; directly into the appropriate table.
;
        LDADDR      fiforegL    ; ADXL X: Ch 2
        MOVF  REGTABPTR, W
        ADDLW REG2START
        MOVWF FSR
        RDDAS
        LDADDR      fiforegH
        INCF  FSR, F
        RDDAS

        LDADDR      fiforegL    ; ADXL Z: Ch 3
        MOVF  REGTABPTR, W
        ADDLW REG3START
        MOVWF FSR
        RDDAS
        LDADDR      fiforegH
        INCF  FSR, F
        RDDAS

        LDADDR      fiforegL    ; Spectron X: Ch 6
        MOVF  REGTABPTR, W
        ADDLW REG6START
        MOVWF FSR
        RDDAS
        LDADDR      fiforegH
        INCF  FSR, F
        RDDAS

        LDADDR      fiforegL    ; Gyrostar (Y): Ch 5
        MOVF  REGTABPTR, W
        ADDLW REG5START
        MOVWF FSR
        RDDAS
        LDADDR      fiforegH
        INCF  FSR, F
        RDDAS

        ADDTBL      REG2START, H'02', +1    ; add in newest values to sum
        ADDTBL      REG3START, H'03', +1    ; in sum table
        ADDTBL      REG6START, H'06', +1
        ADDTBL      REG5START, H'05', +1
        INCF  REGTABPTR, F
        INCF  REGTABPTR, F
        MOVLW B'00000110'
```

```
        ANDWF REGTABPTR, F

        BCF   SHADLED, led1out
        TRNTRSH     SHADLED, PORTLED

        RAISECS
        GOTO  DONE_INTERRUPT

OTHER_LM_INT                    ; Some other LM interrupt has occurred
        RAISECS
        BSF   SHADLED, led2out
        BCF   SHADLED, led1out
        TRNTRSH     SHADLED, PORTLED
        GOTO  DONE_INTERRUPT

HANDLE_OTHER
        BCF   INTCON, T0IF
        BCF   INTCON, RBIF
        GOTO  DONE_INTERRUPT

DONE_INTERRUPT
        TRNTRSH     TEMPFSR, FSR
        SWAPF TEMPSTATUS, W
        MOVWF STATUS
        SWAPF TEMPW, F
        SWAPF TEMPW, W
        RETFIE
;
;
;
; End of Program Memory
;
    IFDEF   __16C71
PROG_MEM_END    EQU     0x3FF
    ENDIF
;
    IFDEF   __16C71A
PROG_MEM_END    EQU     0x3FF
    ENDIF
;
    IFDEF   __16C73
PROG_MEM_END    EQU     0xFFF
    ENDIF
;
    IFDEF   __16C74
PROG_MEM_END    EQU     0xFFF
    ENDIF
;
org     PROG_MEM_END    ; End of Program Memory
ERR_LP_1    GOTO    ERR_LP_1    ; If you get here your program was lost
;
;
    end
```

Appendix III

Code for Controlling Motor

This appendix contains the code implemented on a PIC16c74A to act as the low-level motor controller in the motor control subsystem. The primary function of this code is to step the motor until its position as measured via the AN0 analog input line, matches a preset value. The motor step rate is set at some fixed rate. The PC controller interface is responsible for obtaining new motor set points, setting new step rates, and reporting back a filtered version of the motor position, known as the motor position index as it is comprised of the successive sums of 32 samples of the analog input. There is also provision in the code (not currently used) for obtaining a number from which the wheel speed can be calculated, via an optical encoder affixed to the right rear drive wheel.

As before, <p16Cxx.inc> is a standard header file published by MicroChip which defines the labels for all the status registers on each microcontroller.

```
      LIST    P = 16C74,  F = INHX8M, n = 66

WHEEL_TIMER_ON    equ   0x00  ; disabled wheel speedometer

;*****************************************************************************
;
; Author: William W. Li
; Date:   August 8, 1997
;
; Updated: Sept 18, 1997
;     Added code to include time stamp on each piece of data sent back.
; Updated: Sept 19, 1997
;     Streamlined data transfer code to bring it in line with LMFACE4B,
;     allowing for more interruptable code during data transfer.
; Updated: Sept 22, 1997
;     Changed deadband constants after having removed some of the noise
;     from the motor position potentiometer.
; Updated: Sept 23, 1997
;     Fixed bug where MSB of ports were always set to Tx mode.
; Updated: Nov. 8, 1997
;     Made wheel time optional (WHEEL_TIMER_ON) and added controller
;     interface: motor step interval.
;
; This program does a sample bang-bang controller on the stepping
; motor.  The PC-DIO96 sends one 16-bit control word to the PIC16c74 in
; low-byte, high-byte format, which is interpreted as a motor position
; command.  Next, another 16-bit control is sent down to the PIC16C74
; (low, high) which is the motor step interval, in units of timer 0
; time ticks, currently 51.2 us.
; Then, the following data are sent back up:
;     CNT_LO, CNT_HI          counter, timestamp
;     AN_POSLO, AN_POSHI      position of the motor, in summed
;                             (32 samples) moving-average format.
;     WHL_TIMELO,WHL_TIME_HI,WHL_TIMEOVR  24-bit  wheel time counter
;
; An interrupt running in the background constantly acquires data from
; the ADC on AN0 (RA0 pin), which is coming from the potentiometer
; mounted on the motor shaft.  This data is echoed back up to the
; computer.
;
; In addition to the standard parallel interface to the PC-DIO-96,
; there are the following connections:
;
;     Motor ACBDA*C*B*D*      <--   RB0-7 (CCW from back of motor)
;     Motor potentiometer     -->   RA0 / AN0
;     Wheel speed click -->   RC2
;
; This program tests the parallel port handling abilities of the
; PIC16c74, implementing an interface to the OKI 82C55A CMOS
; programmable peripheral interface in Mode 2 (bidirectional), used on
; the National Instruments PC-DIO-96/PnP board.  Port A of the 82c55a
; is connected to port D of the 16c74, and is used for the
; bidirectional data transfer.  Other control lines which are connected
; are as follows:
;
;     82c55a                  16c74
;     ------                  -----
```

```
;      C7    OBFA* --->  RE1   output buffer full flag
;      C6    ACKA* <---  RA5   acknowledge
;      C5    IBFA  --->  RE2   input buffer full flag
;      C4    STBA* <---  RE0   strobe
;      C3    INTRA --->  n/c   interrupt request
;      port A (0-7)      <-->  port D (0-7)
;
; Write Exchange:
; --------------
; The 82C55A sends a high-to-low transition on the OBFA* (RE1) line,
; which signals that a piece of data is ready for reception by the
; 16c74.  The 16c74 then sends a high-to-low transition on the ACKA*
; (RA5) line, at which point (up to 150ns after this transition) the
; 82c55a will put the data onto the bus at Port D.  Before this, the
; 82c55a's output buffers are tristated into a high-impedance state.
; ACKA* must be low for a minimum of 100 ns.  OBFA* will rise a maximum
; of 150 ns after ACKA* falls.  Port A will be return to a floating
; condition between 20 ns and 250 ns after the rising edge of ACKA* is
; detected.
;
; Read Exchange:
; -------------
; The 82c55a waits for the 16c74 to signal a high-to-low on the STBA*
; (RE0) line, which must be down for a minimum of 100 ns.  The data
; which is put out onto port D by the 16c74 will be latched upon the
; rising edge of STBA*, and must have been present at the port for a
; minimum of 20 ns before the rising edge of STBA*.  The data must
; remain on port D for a minimum of 50 ns after the rising edge of
; STBA* to allow the 82c55a to latch the data properly.  IBFA goes high
; a maximum of 150 ns after STBA* falls and indicates that data has
; been fetched into the input latch.
;
;*****************************************************************
;
;
; HARDWARE SETUP
;       None
;
;
            INCLUDE <p16Cxx.inc>
;
        __CONFIG ( _CP_OFF & _WDT_OFF & _HS_OSC & _PWRTE_OFF )
;
;
      CBLOCK      H'20'
            TEMPW               ; 20 W storage used by interrupt handler
            TEMPSTATUS          ; 21 STATUS storage used by int handler
            TEMPFSR             ; 22 FSR storage used by int handler
            SHADA               ; 23 Shadow register for RA
            SHADB               ; 24 Shadow register for RB
            SHADC               ; 25 Shadow register for RC
            SHADD               ; 26 Shadow register for RD
            SHADE               ; 27 Shadow register for RE

      ; The first time the motor action routine is called after reset,
      ; MOT_CMD is set to be equal to AN_POS.  Because MOT_TIME_CNTR is
      ; initially set to a value greater than 32 (number of items in
```

```
; the AN_POS moving average table), this gives the moving average
; filter for AN_POS time to wind up to its running value.
; The MOT_FLAG motfirst flag is used to determine whether or not
; it's the first time out: MOT_FLAG:motfirst is 1 before the
; initialization, and 0 thereafter.
; MOT_TIME_INTERVAL is the time between step lengths, measured in
; units of 51.2 us, the overflow time of TMR0 at 1:1.  Each time
; the TMR0 interrupt is generated, MOT_TIME_CNTR is decremented.
; When MOT_TIME_CNTR hits 0, a step change is performed and
; MOT_TIME_CNTR is reset to MOT_TIME_INTERVAL.

     MOT_CMDLO          ; 28 Motor position command
     MOT_CMDHI          ; 29
     MOT_FLAG           ; 2A
     MOT_TIME_CNTRLO    ; 2B Counter to adjust TMR0 time ticks
     MOT_TIME_CNTRHI    ; 2C
     MOT_TIME_INTLO     ; 2D value to reset counter to
     MOT_TIME_INTHI     ; 2E

; TMR0 is set to 1:1 ratio, or 200ns clock ticks, for the TMR0
; interrupt when the TMR0 counter clicks over 0xFF.  Each time
; the TMR0 interrupt is triggered, the value in MOT_TIME_CNTR is
; decremented.  Action is taken on the motor when MOT_TIME_CNTR
; counts down to 0.  Sampling is done at the TMR0 interrupt
; interval.  Every 32 samples are averaged together by summing
; and the result is put into AN_POSLO/HI, which is thus updated
; every 32*256*200ns, or 1.6384 ms.  The running sum is kept in
; AN_RUNLO/HI, and is transferred over to AN_POSLO/HI only when
; AN_TBL_PTR clicks over 32.

     AN_POSLO           ; 2F Analog motor position register
     AN_POSHI           ; 30
     AN_RUNLO           ; 31
     AN_RUNHI           ; 32
     AN_TBL_PTR         ; 33 pointer into REG_AN table
     AN_TEMPLO          ; 34 temp arith. register used by M_sub
     AN_TEMPHI          ; 35

; WHL_TIME (LO/HI/OVR) is the 22-bit counter for the wheel speed.
; It holds the time as at the last capture event. i.e., the last
; time the wheel encoder tripped the capture line.  If bit 6 in
; OVR is set, then a timeout condition has occurred.  Once bit 6
; in OVR gets set, no further incrementation of OVR takes place
; until the next capture event.  Bit 7 in OVR is used to indicate
; that a new piece of wheel time data has been acquired.  If it
; is not set, then the external controller knows that the
; WHL_TIME data is old.  To convert, multiply the total value of
; WHL_TIME by 1.6us, which is the length of one clock tick on
; timer 1 in the 1:8 prescale mode.  The resulting time is the
; time between the last black-to-white transition and the current
; one.  What happens is that TIMER 1 is running and can trigger
; an interrupt, as can a CCP1 event.  When CCP1 is triggered, the
; interrupt handler clears TIMER 1.  If CCP1 gets triggered
; before TIMER 1 overflows, then the TIMER 1 interrupt handler
; never gets called.  If TIMER 1 gets triggered, though, we know
; that CCP1 isn't likely to be happening anytime soon, so we
; raise OVR.  When CCP1 happens again, OVR gets cleared.  Each
```

```
        ; time the PC queries the microcontroller, WHL_TIMELO, HI, OVR
        ; get sent back up in addition to the motor position difference
        ; (AN_TEMP) information. In this case, instead of just setting
        ; OVR, we'll have the TIMER 1 interrupt handler increment CNT.
        ; Then, the CCP1 IH will copy CNT to OVR when CCP1 finally gets
        ; triggered.

              WHL_TIMELO        ; 36
              WHL_TIMEHI        ; 37
              WHL_TIMEOVR       ; 38
              WHL_TIMECNT       ; 39

        ; CNT_LO/HI are used to provide a unique timestamp for each
        ; packet of data sent up to the external controller.  CNTLO/HI
        ; counts up at a rate of roughly 51.2us per tick, the interrupt
        ; period of timer 0, which triggers a motor position data
        ; acquisition.

              CNTLO             ; 3A
              CNTHI             ; 3B

        ; EXTCNTLO/HI, EXTMOTCMDLO/HI, EXTAN_POSLO/HI, and
        ; EXTWHLTIMELO/HI/OVR are what the external controller interfaces
        ; with, for reasons of avoiding data contention.  Each time a
        ; motor command is sent down, the current data packet of counter,
        ; motor position, and wheel position data is sent back up.

              EXTCNTLO          ; 3C
              EXTCNTHI          ; 3D
              EXTMOT_CMDLO      ; 3E
              EXTMOT_CMDHI      ; 3F
              EXTAN_POSLO       ; 40
              EXTAN_POSHI       ; 41
              EXTWHL_TIMELO     ; 42
              EXTWHL_TIMEHI     ; 43
              EXTWHL_TIMEOVR    ; 44
              EXTMOT_TIME_INTLO ; 45
              EXTMOT_TIME_INTHI ; 46

        ENDC

;
; REG_AN is the start of the table of analog motor positions.
; Each motor position is a single 8-bit number, acquired via the
; onboard ADC.  There are a total of 32 entries in this table.

REG_AN            EQU   H'50'

; REG_STEP is the start of the table of 9 step excitation values (for
; each of the 8 possible sets of 4-on/4-off coil excitation patterns,
; plus one for all off).

REG_STEP    EQU   H'70'       ; stores table of step excitations

; STEP_PTR is a value from 0 to 8 inclusive which indicates the current
; stepper motor coil excitation pattern.
```

```
STEP_PTR    EQU   H'79'       ; pointer to step excitation table


      CBLOCK      0xA0

            AARGA0                  ; A0
            AARGA1                  ; A1
            AARGB0                  ; A2
            AARGB1                  ; A3
            AARGC0                  ; A4
            AARGC1                  ; A5
            AARGD0                  ; A6
            AARGD1                  ; A7
            TEMPDIV                 ; A8
            LOOPCNT                 ; A9

      ENDC

;
;
;
PORTLED           EQU   PORTC       ; redirects port to access LED at
RC0
TRISLED           EQU   TRISC
SHADLED           EQU   SHADC

PORTMOT           EQU   PORTB       ; redirects port for motor control
TRISMOT           EQU   TRISB
SHADMOT           EQU   SHADB

PORTPAR           EQU   PORTD       ; redirects lines to appropriate
ports
PORTOBF           EQU   PORTE       ; for parallel port interface to
PC-DIO96
PORTACK           EQU   PORTA
PORTIBF           EQU   PORTE
PORTSTB           EQU   PORTE
;
TRISPAR           EQU   TRISD       ; redirects lines to appropriate
ports
TRISOBF           EQU   TRISE       ; for parallel port interface to
PC-DIO96
TRISACK           EQU   TRISA
TRISIBF           EQU   TRISE
TRISSTB           EQU   TRISE
;
SHADPAR           EQU   SHADD       ; redirects lines to appropriate
                                    ; shadow registers
SHADOBF           EQU   SHADE       ; for parallel port interface to
PC-DIO96
SHADACK           EQU   SHADA
SHADIBF           EQU   SHADE
SHADSTB           EQU   SHADE
;
;
;*****************************************************************************
;*****      Constant definitions
```

```
;*****************************************************************************
;

overflag    EQU   H'07'

; Analog Position Table
; ---------------------
an_tbl_size       EQU   H'20'             ; must be power of 2
tbl_ptr_mask      EQU   (an_tbl_size-1)   ; mask controls when
                                          ; AN_TBL_PTR wraps around
;
;
; Step Excitation Table
; ---------------------
step_ptr_mask     EQU   B'00000111'       ; STEP_PTR can take on only
                                          ; one of 8 values, one for
                                          ; each motor phase.

step0       EQU   B'00001111'
step1       EQU   B'00011110'
step2       EQU   B'00111100'
step3       EQU   B'01111000'
step4       EQU   B'11110000'
step5       EQU   B'11100001'
step6       EQU   B'11000011'
step7       EQU   B'10000111'
step8       EQU   B'00000000'
;
deenergized EQU   H'08'       ; STEP_PTR = 8 goes to step8 (all off)
;
; Motor Mode Flags (for MOT_FLAG)
; -------------------------------
motfirst    EQU   H'00'       ; signal first time through
motfree     EQU   H'01'       ; signal off motor
deadband    EQU   H'18'

initmotorcntlo    EQU   H'60'
initmotorcnthi    EQU   H'00'
runmotorcntlo     EQU   H'00'
runmotorcnthi     EQU   H'08'
;

;
; Parallel Interface Handshaking Constants
; ----------------------------------------
ackpar      EQU   H'05'       ; ACK* line is RA5
stbpar      EQU   H'00'       ; STB* line is RE0
obfpar      EQU   H'01'       ; OBF* line is RE1
ibfpar      EQU   H'02'       ; IBF line is RE2
;
;
led0        EQU   H'00'       ; LED at RC0
led1        EQU   H'01'       ; LED at RC1
;
;
newdata     EQU   H'07'       ; new data in WHL_TIMEOVR
;
```

```
;
;*****************************************************************************
;*****      Start macro definitions here.
;*****************************************************************************
;
; Macro for transferring data from one register to another.
; W register gets trashed
TRNTRSH          MACRO fromReg, toReg
            MOVF  fromReg, W
            MOVWF toReg
            ENDM
;
;
; Macro for setting bank 1 mode
SET_BANK_1  MACRO
            BSF   STATUS, RP0       ; Bank 1
            ENDM
;
; Macro for setting bank 0 mode
SET_BANK_0  MACRO
            BCF   STATUS, RP0       ; Bank 0
            ENDM
;
; Macro for setting tristate bits
SETTRIS          MACRO whichbuffer, whatvalue
            SET_BANK_1
      IF (whichbuffer == TRISE)
            MOVLW (whatvalue & B'00000111')
      ELSE
            MOVLW whatvalue
      ENDIF
            MOVWF whichbuffer
            SET_BANK_0
            ENDM
;
;
; Macro for setting tristate bits to all Tx (0's)
SETTX       MACRO whichbuffer
            SET_BANK_1
            CLRF  whichbuffer
            SET_BANK_0
            ENDM
;
;
; Macro for setting tristate bits to all Rx (1's)
SETRX       MACRO whichbuffer
            SET_BANK_1
      IF (whichbuffer == TRISE)
            MOVLW B'00000111'
      ELSE
            MOVLW H'FF'
      ENDIF
            MOVWF whichbuffer
            SET_BANK_0
            ENDM
;
; Macro for enabling GIE
```

```
SETGIE          MACRO
          BSF   INTCON, GIE       ; Enable global interrupt
          ENDM

;*****************************************************************
; CLRGIE disables the global interrupt

CLRGIE          MACRO
          LOCAL LOOP

LOOP      BCF   INTCON, GIE       ; Disable global interrupt
          BTFSC INTCON, GIE       ; Global interrupt disabled?
          GOTO  LOOP
          ENDM

;
; Macro for getting data from the parallel interface.
; First, lower the ACK* line, then get the data, then re-raise ACK*
GETDATA         MACRO dest
          BCF   SHADACK, ackpar         ; clear ACK*, then leave ACK*
                                        ; line down for a bit to give
          TRNTRSH     SHADACK, PORTACK  ; the computer time to finish
          NOP
          TRNTRSH     PORTPAR, dest     ; transfer parallel port data
                                        ; to destination
          BSF         SHADACK, ackpar   ; raise ACK* again
          TRNTRSH     SHADACK, PORTACK
          ENDM

;*****************************************************************
; Macro for waiting for OBF high-to-low transition
; indicating acknowledge has been received.

WAIT_FOR_OBF_HIGH  MACRO
          LOCAL WAIT

WAIT      BTFSS PORTOBF, obfpar
          GOTO  WAIT
          ENDM

;*****************************************************************
; WAIT_FOR_OBF_LOW waits until the OBF line goes low, indicating
; that the 82c55a has data ready to put onto the bus.

WAIT_FOR_OBF_LOW  MACRO
          LOCAL WAIT

WAIT      BTFSC PORTOBF, obfpar
          GOTO WAIT
          ENDM

;
;*****************************************************************
;       Double Precision  Addition ( ACCb + ACCa -> ACCb )
; This macro adds the 8-bit value in W to the 16-bit value
; in ACCb, ACCb+1.  The result is placed in ACCb.
;
```

```
M_add8          MACRO ACCb
            ADDWF ACCb, F           ; add lsb
            BTFSC STATUS, C   ; add in carry
            INCF  (ACCb+1), F
            ENDM
;
;
;*****************************************************************
;         Double Precision Subtraction ( ACCb - ACCa -> ACCb )
; This macro subtracts the 8-bit value located in W from the
; 16-bit value in ACCb, ACCb+1.  The result is placed back in ACCb.
;
M_sub8          MACRO ACCb
            SUBWF ACCb, F           ; subtract LSB
            BTFSS STATUS, C
            DECF  (ACCb+1), F ; subtract borrow
            ENDM
;
;
; Macro for double-precision addition on two 16-bit values.
; ACCb and ACCa are both to be arranged in the standard low-byte,
; high-byte fashion.  The result is placed in ACCb.  ACCa remains
; untouched.
;
M_add16         MACRO ACCb, ACCa
            MOVF  ACCa, W
            ADDWF ACCb, F           ; add LSB
            BTFSC STATUS, C   ; add in carry bit
            INCF  (ACCb+1), F
            MOVF  (ACCa+1), W
            ADDWF (ACCb+1), F
            ENDM
;
;
; Macro for double-precision subtraction on two 16-bit values.
; ACCb and ACCa are both to be arranged in the standard low-byte,
; high-byte fashion.  The result is placed in ACCb.
; ACCb <-- ACCb - ACCa
;
M_sub16         MACRO ACCb, ACCa
            MOVF  ACCa, W
            SUBWF ACCb, F           ; subtract LSB
            BTFSS STATUS, C
            DECF  (ACCb+1), F ; subtract borrow
            MOVF  (ACCa+1), W
            SUBWF (ACCb+1), F ; subtract MSB
            ENDM
;
;
; Macro for adding or subtracting something to running sum.  Adding or
; subtracting from the sum is dependent on the sign of the which
; parameter to this macro. (positive = add, otherwise subtract) The
; table is located starting at REG_AN, assumed to be in the same page
; as everything else.  The pointer to this table is located at
; AN_TBL_PTR, so that the FSR is calculated as
; (REG_AN + val(AN_TBL_PTR)).  This will be the location of an 8-bit
```

```
; value.  The 16-bit sum is located at where.
;
ADDTBL          MACRO which, where
            MOVLW REG_AN
            ADDWF AN_TBL_PTR, W
            MOVWF FSR
            MOVF  INDF, W

      IF (which > 0)
            M_add8      where       ; Finally, add ACCb to ACCa
      ELSE
            M_sub8      where
      ENDIF
            ENDM


;
;
;*************************************************************************
;*****      Start program here.
;*************************************************************************
;
      ORG   H'0000'                 ; set up start of program
      GOTO  START

      ORG   H'0004'                 ; set up interrupt vector
      GOTO  INTERRUPT_HANDLER ;

      ORG   H'0005'

START                               ; POWER_ON Reset (Beginning of program)
      CLRF  STATUS                  ; Do initialization (Bank 0)
      CALL  CLRGIE
      CLRF  INTCON
;
; Clear RAM to all zeroes.
; Registers with suffix "L" hold the low byte, "H" registers hold the
; high byte.
;
      MOVLW H'20'                   ; Clear lower page
      MOVWF FSR
CLEAR_TABLE_0
      CLRF  INDF                    ; Clear memory location
      INCF  FSR, F
      BTFSS FSR, 7                  ; Keep going until we've passed 0x7F.
      GOTO  CLEAR_TABLE_0

      MOVLW H'A0'                   ; Clear upper page
      MOVWF FSR
CLEAR_TABLE_1
      CLRF  INDF                    ; Clear memory location
      INCFSZ      FSR, F            ; Keep going until we've passed 0xFF.
      GOTO  CLEAR_TABLE_1
;
; Set up step excitation table
;
```

```
        MOVLW step0
        MOVWF (REG_STEP)
        MOVLW step1
        MOVWF (REG_STEP + 1)
        MOVLW step2
        MOVWF (REG_STEP + 2)
        MOVLW step3
        MOVWF (REG_STEP + 3)
        MOVLW step4
        MOVWF (REG_STEP + 4)
        MOVLW step5
        MOVWF (REG_STEP + 5)
        MOVLW step6
        MOVWF (REG_STEP + 6)
        MOVLW step7
        MOVWF (REG_STEP + 7)
        MOVLW step8             ; special: all zeroes
        MOVWF (REG_STEP + 8)
;
; Set up motor constants

        BSF   MOT_FLAG, motfirst      ; signal the first time through on
reset
        MOVLW initmotorcntlo          ; give enough time to settle down
        MOVWF MOT_TIME_CNTRLO
        MOVLW initmotorcnthi
        MOVWF MOT_TIME_CNTRHI

        MOVLW runmotorcntlo
        MOVWF MOT_TIME_INTLO
        MOVLW runmotorcnthi
        MOVWF MOT_TIME_INTHI
;
; Set up shadow registers
;
        BSF   SHADACK, ackpar         ; raise ACK* line
        BSF   SHADSTB, stbpar         ; raise STB* line
        TRNTRSH     SHADA, PORTA      ; After this initialization, we
        TRNTRSH     SHADB, PORTB      ; won't ever be referring to the
        TRNTRSH     SHADC, PORTC      ; ports by "PORTx" again.  Instead,
        TRNTRSH     SHADD, PORTD      ; each line will get its own port
        TRNTRSH     SHADE, PORTE      ; ID, e.g. "PORTRDLM".  This makes
                                      ; for heftier but more maintainable
                                      ; code.


; Next, ensure that all the lines are behaving as digital I/O and are
; set to read or write as appropriate.  Any line which is not actively
; used is set to Rx, or high impedance (tristated) state.
;
        SET_BANK_1
        MOVLW B'00000100'       ; set RA5,2 and REx to digital mode
        MOVWF ADCON1

        MOVLW B'00000111'       ; TRISE is a special case as the upper
        MOVWF TRISE             ; 5 bits are used.
        MOVLW H'FF'             ; set all I/O lines to Rx first
```

```
        MOVWF TRISD
        MOVWF TRISC
        MOVWF TRISB
        MOVWF TRISA

        CLRF  TRISMOT           ; Clear those lines which will be Tx
        BCF   TRISACK, ackpar   ; All Rx lines will be in high-impedance
        BCF   TRISSTB, stbpar   ; mode
        BCF   TRISLED, led0
        BCF   TRISLED, led1

        SET_BANK_0

        MOVLW B'10000001'       ; Set TAD for 20MHz osc, turn on ADC,
        MOVWF ADCON0            ; and select AN0

        CLRF  PIR1              ; Clear interrupt flags for peripheral
        CLRF  PIR2              ; devices.
;
; Set up timer 0, no prescale value.  Overflow will take place
; every 256 * 200ns = 51.2 us.
;
        CLRWDT
        SET_BANK_1
        BSF   OPTION_REG, PSA         ; Set WDT prescale

        BCF   OPTION_REG, PS2         ; Set prescale to 1:1
        BCF   OPTION_REG, PS1
        BCF   OPTION_REG, PS0

        BCF   OPTION_REG, T0CS
        SET_BANK_0
        CLRF  TMR0

  IF  WHEEL_TIMER_ON == 1
; Set up timer 1: 1:8 prescale value, timer 1 on, oscillator shut off
; then clear timer 1

        MOVLW B'00110001'
        MOVWF T1CON
        CLRF  TMR1L
        CLRF  TMR1H

; Set up capture CCP1 on falling edges

        MOVLW B'00000100'
        MOVWF CCP1CON

  ENDIF

        BSF   INTCON, T0IE            ; Enable Timer 0 interrupts
        BSF   INTCON, PEIE            ; Enable peripheral interrupts

        SET_BANK_1
        BSF   PIE1, ADIE        ; Enable ADC-done interrupt

  IF  WHEEL_TIMER_ON == 1
```

```
      BSF   PIE1, CCP1IE            ; Enable CCP1 interrupt
      BSF   PIE1, TMR1IE
  ENDIF

      SET_BANK_0
      SETGIE
      WAIT_FOR_OBF_HIGH

MAINLOOP
      WAIT_FOR_OBF_LOW
      CLRGIE
      GETDATA     EXTMOT_CMDLO

      WAIT_FOR_OBF_HIGH       ; Wait for the OBF high-to-low
      SETGIE                  ; transition.
      WAIT_FOR_OBF_LOW

      CLRGIE            ; Disable the interrupts while we get data.
      GETDATA     EXTMOT_CMDHI

      WAIT_FOR_OBF_HIGH       ; Wait for the OBF high-to-low
      SETGIE                  ; transition.
      WAIT_FOR_OBF_LOW

      CLRGIE
      GETDATA     EXTMOT_TIME_INTLO

      WAIT_FOR_OBF_HIGH       ; Wait for the OBF high-to-low
      SETGIE                  ; transition.
      WAIT_FOR_OBF_LOW

      CLRGIE
      GETDATA     EXTMOT_TIME_INTHI

; As soon as we get the motor command, we will freeze the interrupts
; and transfer the current data packet consisting of CNT, which holds
; the time stamp for the AN_POS motor position data, AN_POS, and
; WHL_TIME.  We will also, while we're at it, transfer the new command
; into the motor command registers.  Although it's not important that
; all three of these pieces of data be time-registered with one
; another, it is important that we time register CNT with AN_POS, and
; each of the 8-bit parts of each register with the other 8-bit parts
; of the same register, so we might as well do it all in one big shot.
; After we're done, clear the new data flag on WHL_TIMEOVR to indicate
; that we've transferred the data out.  If it was already clear, then
; clearing it again won't make a difference anyways.

      TRNTRSH     EXTMOT_CMDLO, MOT_CMDLO
      TRNTRSH     EXTMOT_CMDHI, MOT_CMDHI

      TRNTRSH     EXTMOT_TIME_INTLO, MOT_TIME_INTLO
      TRNTRSH     EXTMOT_TIME_INTHI, MOT_TIME_INTHI

      TRNTRSH     CNTLO, EXTCNTLO
      TRNTRSH     CNTHI, EXTCNTHI
      TRNTRSH     AN_POSLO, EXTAN_POSLO
      TRNTRSH     AN_POSHI, EXTAN_POSHI
```

```
  IF  WHEEL_TIMER_ON == 1
      TRNTRSH     WHL_TIMELO, EXTWHL_TIMELO
      TRNTRSH     WHL_TIMEHI, EXTWHL_TIMEHI
      TRNTRSH     WHL_TIMEOVR, EXTWHL_TIMEOVR
      BCF   WHL_TIMEOVR, newdata
  ENDIF

; When all done, then we can re-enable the interrupts.

      SETGIE

; Now that we're done transferring, we can output all the data in the
; packet.

      SETTX TRISPAR           ; Get ready to output data.

      MOVF  EXTCNTLO, W       ; Output all the data in the right order.
      CALL  OUTPUT_DATA
      MOVF  EXTCNTHI, W
      CALL  OUTPUT_DATA

      MOVF  EXTAN_POSLO, W
      CALL  OUTPUT_DATA
      MOVF  EXTAN_POSHI, W
      CALL  OUTPUT_DATA

  IF  WHEEL_TIMER_ON == 1
      MOVF  EXTWHL_TIMELO, W
      CALL  OUTPUT_DATA
      MOVF  EXTWHL_TIMEHI, W  ; transfer the high byte to parallel port
      CALL  OUTPUT_DATA
      MOVF  EXTWHL_TIMEOVR, W ; transfer the high byte to parallel port
      CALL  OUTPUT_DATA
  ENDIF

      SETRX TRISPAR

      GOTO  MAINLOOP


;*************************************************************************
;*****      Subroutines start here.
;*************************************************************************


;*************************************************************************
; OUTPUT_DATA
;     Outputs one byte of data to the host (PC-DIO-96), waits
;     until we're sure the host has latched the data.
;     The data byte is found in the W register.
;     This routine assumes that PORTPAR is already set for Tx.
;
OUTPUT_DATA
      MOVWF PORTPAR
      CLRGIE
      BCF   SHADSTB, stbpar         ; lower STB* line
```

```
      TRNTRSH     SHADSTB, PORTSTB

WAIT_FOR_IBF_HIGH
      BTFSS PORTIBF, ibfpar   ; loop till IBF goes high
      GOTO  WAIT_FOR_IBF_HIGH ; indicating 82c55a has latched data

      BSF   SHADSTB, stbpar         ; re-raise STB*
      TRNTRSH     SHADSTB, PORTSTB

      SETGIE

; Wait for the IBF line to go low, indicating that the host
; PC has read the data which was latched by the 82c55a port
; interface in the PC-DIO96.

WAIT_FOR_IBF_LOW
      BTFSC PORTIBF, ibfpar
      GOTO  WAIT_FOR_IBF_LOW
      RETURN


;*****************************************************************
;
INTERRUPT_HANDLER
      MOVWF TEMPW
      SWAPF STATUS, W
      SET_BANK_0
      MOVWF TEMPSTATUS
      TRNTRSH     FSR, TEMPFSR

      BTFSC INTCON, T0IF      ; Check to see if it's a timer interrupt
      GOTO  HANDLE_TIMER_0

      BTFSC PIR1, ADIF        ; Check to see if it's an ADC-done int
      GOTO  HANDLE_ADC_DONE

  IF  WHEEL_TIMER_ON == 1
      BTFSC PIR1, TMR1IF      ; Check to see if Timer 1 has overflowed
      GOTO  HANDLE_TIMER_1

      BTFSC PIR1, CCP1IF      ; See if the wheel encoder has tripped
      GOTO  HANDLE_CAPTURE
  ENDIF

      GOTO  HANDLE_OTHER
;
HANDLE_TIMER_0
      BCF   INTCON, T0IF      ; Clear interrupt flag
      BSF   ADCON0, GO        ; then start ADC

; Update time counter, which happens every time timer 0 overflows from
; FF to 00.  Timer 0 was earlier set up to tick at 200ns, so it
; overflows in 51.2us.

      INCFSZ      CNTLO, F
      GOTO  CHECK_MOT_TIME
      INCF  CNTHI, F
```

```
CHECK_MOT_TIME
      MOVLW 0x01                    ; decrement 16-bit motor counter
      M_sub8      MOT_TIME_CNTRLO   ; and see if it's all zero

      MOVF  MOT_TIME_CNTRLO, F      ; check low byte
      BTFSS STATUS, Z
      GOTO  DONE_INTERRUPT
      MOVF  MOT_TIME_CNTRHI, F      ; then check high byte
      BTFSS STATUS, Z               ; if both zero, reset motor counter
      GOTO  DONE_INTERRUPT

      TRNTRSH     MOT_TIME_INTLO, MOT_TIME_CNTRLO
      TRNTRSH     MOT_TIME_INTHI, MOT_TIME_CNTRHI

; First, check to see if it's the first time after a reset
; If it is, we want to set this current, initial position as
; the default resting place of the motor, to ensure that we
; stay here on power-up.

      BTFSS MOT_FLAG, motfirst
      GOTO  REGULAR_MOTOR

      BCF   MOT_FLAG, motfirst      ; clear the flag from now on
      TRNTRSH     AN_POSLO, MOT_CMDLO
      TRNTRSH     AN_POSHI, MOT_CMDHI

      CLRF  STEP_PTR
      GOTO  ENERGIZE_MOTOR

REGULAR_MOTOR
; Look at the MSB of MOT_CMDHI to see if it's high (i.e., MOT_CMD
; negative).  We use this to indicate that we want to deenergize the
; motor coils.  The motor remains in this state until another, positive
; motor command is sent to it.

      BTFSS MOT_CMDHI, H'07'
      GOTO  MOTOR_CONTROL

      MOVLW deenergized
      MOVWF STEP_PTR
      GOTO  ENERGIZE_MOTOR

MOTOR_CONTROL

      TRNTRSH     AN_POSLO, AN_TEMPLO     ; Transfer AN_POS to AN_TEMP
      TRNTRSH     AN_POSHI, AN_TEMPHI     ; subtract (AN_TEMP-MOT_CMD)
      M_sub16     AN_TEMPLO, MOT_CMDLO

      BTFSS       AN_TEMPHI, H'07'        ; Look at the sign bit to see
                                          ; if it's negative
      GOTO  POSITIVE_CHECK

; Now, add the deadband value because we know the number is negative.
; If we end up with a positive value, we know that we're within the
; deadband.  Otherwise, if the result is negative (MSB set), then we're
; not.
```

```
      MOVLW deadband
      M_add8      AN_TEMPLO
      BTFSS AN_TEMPHI, H'07'
      GOTO  IN_DEADBAND

NEG_ROTATION
      INCF  STEP_PTR, W       ; Negative, so decrease rotation
      GOTO  FIX_PTR

; If the difference is positive, subtract away the deadband number.
; If the result is negative, then we're within the deadband.

POSITIVE_CHECK
      MOVLW deadband
      M_sub8      AN_TEMPLO
      BTFSC AN_TEMPHI, H'07'
      GOTO  IN_DEADBAND

POS_ROTATION
      DECF  STEP_PTR, W       ; Positive, so increase rotation
      GOTO  FIX_PTR                 ; and fall through to FIX_PTR

IN_DEADBAND
      MOVF  STEP_PTR, W       ; Load up current step pointer and
                              ; fall through

FIX_PTR
      ANDLW step_ptr_mask     ; Mask to get rid of spurious bits
      MOVWF STEP_PTR          ; then fall through to energize motor

ENERGIZE_MOTOR
      MOVLW REG_STEP          ; Load up the proper coil energization
      ADDWF STEP_PTR, W       ; pattern from the step table, and
      MOVWF FSR               ; send it to the motor.
      TRNTRSH     INDF, PORTMOT
      GOTO  DONE_INTERRUPT
;
HANDLE_ADC_DONE
      BCF   PIR1, ADIF        ; Clear ADC-done interrupt flag

;     ADDTBL      -1          ; Subtract the oldest table value,
                              ; for moving average filter

      MOVF  AN_TBL_PTR, W           ; Get pointer to table,
      ADDLW REG_AN                  ; add offset to start of table
      MOVWF FSR
      TRNTRSH     ADRES, INDF       ; Get converted ADC value

      ADDTBL      +1, AN_RUNLO      ; Add in the newest value

      INCF  AN_TBL_PTR, W     ; Increment and mask table pointer
      ANDLW tbl_ptr_mask      ; so that it wraps around for 32 values
      MOVWF AN_TBL_PTR

      BTFSS STATUS, Z         ; Check to see if we've gotten all 32
      GOTO  DONE_INTERRUPT    ; If not, return out of here.
```

```
      TRNTRSH     AN_RUNLO, AN_POSLO      ; Transfer running sum to
      TRNTRSH     AN_RUNHI, AN_POSHI      ; output register
      CLRF  AN_RUNLO                      ; Clear running sum
      CLRF  AN_RUNHI

      GOTO  DONE_INTERRUPT

;
; If we've triggered the timer 1 interrupt, then we must have timed out
; If this is the case, then check to see if WHL_TIMECNT has timed out
; too by checking bit 6.  If not, then increment WHL_TIMECNT.  Note
; that bit 7 in WHL_TIMEOVR is reserved to indicate that new wheel data
; has arrived.  Because the HANDLE_CAPTURE routine copies WHL_TIMECNT
; to WHL_TIMEOVR, it is imperative that WHL_TIMECNT not trample on
; bit 7.
;
; Currently disabled function, set/disabled with WHEEL_TIMER_ON expr.

  IF  WHEEL_TIMER_ON == 1

HANDLE_TIMER_1
      BCF   PIR1, TMR1IF            ; Reset interrupt flag

      BTFSS WHL_TIMECNT, H'06'
      INCF  WHL_TIMECNT, F

      GOTO  DONE_INTERRUPT

; If we've triggered a capture, then reset the timer.

HANDLE_CAPTURE
      BCF   PIR1, CCP1IF            ; Reset interrupt flag

      BCF   T1CON, TMR1ON           ; Stop and reset timer 1
      CLRF  TMR1L
      CLRF  TMR1H
      BSF   T1CON, TMR1ON           ; Restart timer 1
      BCF   PIR1, TMR1IF            ; Reset timer 1 flag, just in case

      TRNTRSH     CCPR1L, WHL_TIMELO
      TRNTRSH     CCPR1H, WHL_TIMEHI
      TRNTRSH     WHL_TIMECNT, WHL_TIMEOVR
      BSF   WHL_TIMEOVR, newdata    ; Set new data flag
      CLRF  WHL_TIMECNT

; Toggle LED to show that we've come here

      BTFSS SHADLED, led0
      GOTO  LIGHT_ON1
      BCF   SHADLED, led0
      TRNTRSH     SHADLED, PORTLED
      GOTO  DONE_INTERRUPT

LIGHT_ON1
      BSF   SHADLED, led0
      TRNTRSH     SHADLED, PORTLED
```

```
      GOTO  DONE_INTERRUPT    ; Fall through to exit interrupt code

  ENDIF

;
HANDLE_OTHER
      GOTO  DONE_INTERRUPT    ; Fall through to exit interrupt code
;
DONE_INTERRUPT
      TRNTRSH     TEMPFSR, FSR
      SWAPF TEMPSTATUS, W
      MOVWF STATUS
      SWAPF TEMPW, F
      SWAPF TEMPW, W
      RETFIE
;
; End of Program Memory
;
    IFDEF   __16C71
PROG_MEM_END    EQU     0x3FF
    ENDIF
;
    IFDEF   __16C71A
PROG_MEM_END    EQU     0x3FF
    ENDIF
;
    IFDEF   __16C73
PROG_MEM_END    EQU     0xFFF
    ENDIF
;
    IFDEF   __16C74
PROG_MEM_END    EQU     0xFFF
    ENDIF
;
;
        org     PROG_MEM_END        ; End of Program Memory
ERR_LP_1    GOTO    ERR_LP_1     ; If you get here your program was
lost
;
;
    end
```

Appendix IV

Code for Supervisory Control

This appendix contains Borland C++ code used on the supervisory PC controller. After some initial setup, this code settles into a loop in which first inertial data is acquired from the data acquisition board, then this data is used to calculate an appropriate motor position set point, followed by a motor position command issued to the motor control board. The interface with the two boards takes place via a National Instruments PC-DIO-96 digital interface card. There are two main modules and one header file in this code. The DIO_FACE.CPP and DIO_FACE.H files contain code to interface with the PC-DIO-96 card directly, without going through the relatively slower National Instruments NI-DAQ library. CONTROL.CPP contains the acquisition, logging, and control code.

DIO_FACE.H

```
//////////////////////////////////////////////////////////////////
//
// DIO_FACE
//
// This class encapsulates methods for reading and writing
// single- and double-byte words from and to the PC-DIO96
```

```
// digital I/O board.  DIO_INT is optimized for single-word
// access to the PC-DIO96, and accesses the board directly
// via the port address area in memory.
//
////////////////////////////////////////////////////////////////

typedef enum { PORTA, PORTB } port_type;
typedef enum { FALSE = 0, TRUE = 1 } boolean;
typedef enum { lohi, hilo } order_type;

class dio_face
{
      public:
            dio_face();                                    // default
constructor
            dio_face(port_type port_sel);
            dio_face(dio_face& c_face);
            ~dio_face();

            const port_type get_port() const;
            void set_port(const port_type port_sel);

            dio_face& operator =(dio_face& c_face);
            dio_face& operator =(unsigned char output_byte);
            dio_face& operator =(unsigned int output_word);

            const unsigned char inbyte();
            const unsigned int inword();

            void set_timeout(const unsigned int new_timeout);
            const unsigned int get_timeout() const;
            const int get_err() const;

      private:
            static boolean port_open;

            port_type io_port;
            unsigned int portPA;
            unsigned int portPC;
            unsigned int timeout;
            int err_status;
            order_type byte_order;

            void config_port();

            void send_byte(const unsigned char output_byte);
            void send_word(const unsigned int output_word);
};

inline const port_type dio_face::get_port() const
      {
      return (io_port);
      }

inline const unsigned int dio_face::get_timeout() const
      {
      return (timeout);
```

```
      }

inline const int dio_face::get_err() const
      {
      return (err_status);
      }
```

DIO_FACE.CPP

```
//////////////////////////////////////////////////////////////////
//
// DIO_FACE
//
// This class encapsulates methods for reading and writing
// single- and double-byte words from and to the PC-DIO96
// digital I/O board.  DIO_INT is optimized for single-word
// access to the PC-DIO96, and accesses the board directly
// via the port address area in memory.
//
// The very first time an instance of a dio_face object is
// declared, we configure the ports for bidirectional I/O.
//
//////////////////////////////////////////////////////////////////

#include <conio.h>
#include "dio_face.h"

boolean dio_face::port_open = FALSE;

const unsigned int default_timeout = 1000;

// Port addresses for the PC-DIO96 in its default location.

const unsigned int BASE_ADDRESS = 0x180;
const unsigned int APORTAoffset = 0x00;
const unsigned int APORTBoffset     = 0x01;
const unsigned int APORTCoffset     = 0x02;
const unsigned int ACNFGoffset      = 0x03;
const unsigned int BPORTAoffset     = 0x04;
const unsigned int BPORTBoffset     = 0x05;
const unsigned int BPORTCoffset     = 0x06;
const unsigned int BCNFGoffset      = 0x07;

// The default constructor sets up for port A.
// We check to see if the ports have been configured
// already by some other instance, and if not, we do it here.

dio_face::dio_face()
      : timeout (default_timeout),
        err_status (0),
        byte_order (lohi)
      {
      set_port(PORTA);
```

```
      if (port_open == FALSE)
            config_port();
      }

dio_face::dio_face(port_type port_sel)
      : timeout (default_timeout),
        err_status (0),
        byte_order (lohi)
      {
      set_port(port_sel);

      if (port_open == FALSE)
            config_port();
      }

dio_face::dio_face(dio_face& c_face)
      : timeout (c_face.timeout),
        err_status (c_face.err_status),
        byte_order (c_face.byte_order)
      {
      portPA = c_face.portPA;
      portPC = c_face.portPC;
      io_port = c_face.io_port;

      if (port_open == FALSE)
            config_port();
      }

dio_face::~dio_face()
      {
      }


// Set up both ports to be mode 2 I/O (bidirectional)
void dio_face::config_port()
      {
      outp((BASE_ADDRESS + ACNFGoffset), 0xC0);
      outp((BASE_ADDRESS + BCNFGoffset), 0xC0);

      port_open = TRUE;
      }

void dio_face::set_port(const port_type port_sel)
      {
      switch (port_sel)
            {
            case PORTA:
                  portPA = BASE_ADDRESS + APORTAoffset;
                  portPC = BASE_ADDRESS + APORTCoffset;
                  break;
            case PORTB:
                  portPA = BASE_ADDRESS + BPORTAoffset;
                  portPC = BASE_ADDRESS + BPORTCoffset;
                  break;
            }
      }
```

```
dio_face& dio_face::operator =(dio_face& c_face)
      {
      portPA = c_face.portPA;
      portPC = c_face.portPC;
      io_port = c_face.io_port;

      return (*this);
      }

// If we set the dio_face object equal to a single byte,
// that means that we want to output that byte to the port.
dio_face& dio_face::operator =(unsigned char output_byte)
      {
      send_byte(output_byte);
      return (*this);
      }

dio_face& dio_face::operator =(unsigned int output_word)
      {
      send_word(output_word);
      return (*this);
      }

const unsigned char dio_face::inbyte()
      {
      unsigned int wait_count = 0;

      // Wait for strobe signal
      while ((!(inp(portPC) & 0x20)) && (wait_count < timeout))
            wait_count++;

      if (wait_count >= timeout)
            {
            err_status = -2;
            return (0);
            }
      else
            {
            err_status = 0;
            }

      // Then, get the data and return it.

      return (inp(portPA));
      }

const unsigned int dio_face::inword()
      {
      unsigned int word;
      unsigned char onebyte, twobyte;

      onebyte = inbyte();
      twobyte = inbyte();

      if (byte_order == lohi)
            word = (unsigned int)onebyte + ((unsigned int)twobyte <<
8);
```

```
      else
            word = (unsigned int)twobyte + ((unsigned int)onebyte <<
8);

      return (word);
      }

void dio_face::set_timeout(const unsigned int new_timeout)
      {
      timeout = new_timeout;
      }

void dio_face::send_byte(const unsigned char output_byte)
      {
      unsigned int wait_count = 0;

      // Write out a byte to the port, first
      outp(portPA, output_byte);

      // Then, wait for the acknowledge signal
      while ((!(inp(portPC) & 0x80)) && (wait_count < timeout))
            wait_count++;

      if (wait_count >= timeout)
            err_status = -1;
      else
            err_status = 0;
      }


// 16-bit words are sent in either low/high or high/low order
void dio_face::send_word(const unsigned int output_word)
      {
      unsigned char onebyte, twobyte;

      if (byte_order == lohi)
            {
            onebyte = output_word & 0xff;
            twobyte = (output_word & 0xff00) >> 8;
            }
      else
            {
            twobyte = output_word & 0xff;
            onebyte = (output_word & 0xff00) >> 8;
            }

      send_byte(onebyte);
      send_byte(twobyte);
      }
```

CONTROL.CPP

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "dio_face.h"

// Data packet size (in 16-bit words) from the LM12H458 data
// acquisition board.
const unsigned int numdata = 4;
const unsigned int numtilt = 64;


// Output one 16-bit motor control word to the motor control
// board, then get back:
//   16-bit clock ticks (at 25.6 us)
//   16-bit motor position (filtered)
//  16-bit wheel position data, followed by 7-bit highest byte.
int motorcmd(unsigned int motor_cmd,
             unsigned int motor_int,
             unsigned int *motor_position,
             long int *mot_count)
{
     dio_face motor_port(PORTB);
     unsigned char inbyte;

     motor_port = motor_cmd;
     if (motor_port.get_err() != 0)
          return (motor_port.get_err());

     motor_port = motor_int;
     if (motor_port.get_err() != 0)
          return (motor_port.get_err());

     *mot_count = motor_port.inword();
     *motor_position = motor_port.inword();

     return(0);
}

// Get a data packet of inertial data from the data acquisition board.
int get4data(unsigned int *data, long int *count, long int lastcount)
{
     dio_face adc_port(PORTA);
     unsigned int index;

     // Write out a byte to the port, first, to signal we
     // want to get a data packet.  0x18 means sum, all
     adc_port = (unsigned char) 0x18;
     if (adc_port.get_err() != 0)
          return(adc_port.get_err());

     *count = adc_port.inword();
     if (adc_port.get_err() != 0)
          return(adc_port.get_err());
```

```
    if (*count == lastcount)  // Don't expect any more data.
        return (1);

    for (index = 0; index < numdata; index++)
        {
        data[index] = adc_port.inword();
        if (adc_port.get_err() != 0)
            return(adc_port.get_err());
        }

    return(0);
}

unsigned int interp(double tilt)
{
    const double table[5][2] = {
                        { 7343.0, -20.0 },
                        { 6686.0, -10.0 },
                        { 4881.0,   0.0 },
                        { 1558.0,  10.0 },
                        {    5.0,  20.0 } };

    int index;
    int ok = 1;
    unsigned int retval;
    double retmot = 0.0;

    if (tilt <= table[0][1])
        retmot = table[0][0];
    else
        {
        index = 0;
        while (ok)
            {
            if ( (tilt > table[index][1]) && (tilt <=
table[index+1][1]) )
                {
                retmot = table[index][0] + ( (tilt - table[index][1])
* (table[index+1][0] - table[index][0]) / (table[index+1][1] -
table[index][1]) );
                }

            index++;
            if (index > 3)    ok = 0;
            }

        if (tilt >= table[4][1])    retmot = table[4][0];
        }

    retval = (unsigned int) retmot;
    return (retval);
}

double filtered_tilt(unsigned int *curdata,
                        int clear,
                        unsigned long int axavg,
                        unsigned long int gyavg,
```

```
                        unsigned long int sxavg,
                        double dt)
    {
    static double ax[numtilt], az[numtilt], tilt[numtilt];
    static unsigned int tilt_index;
    static double tilt_gyro;
    const double gyrofactor = 0.0222;
    const double sxfactor = -27.9725;
    const double conv = 5.0 / 4096.0;
    unsigned int index;
    double tilt_mean, ax_mean, az_mean;

    double gyro;
    double sx;

    ax[tilt_index] = ((double)curdata[0] - (double)axavg) * conv;
    az[tilt_index] = ((double)curdata[1] - (double)axavg) * conv;
    gyro = ((double)curdata[2] - (double)gyavg) * conv / gyrofactor;
    sx = ((double)curdata[3] - (double)sxavg) * conv * sxfactor;

    if (clear == 1)
        {
        tilt_index = 0;
        tilt_gyro = 0;
        for (index = 0; index< numtilt; index++)
            tilt[index] = 0;
        return (0);
        }

    ax_mean = 0;
    az_mean = 0;
    for (index = 0; index < numtilt; index++)
        {
        ax_mean += ax[index];
        az_mean += az[index];
        }

    tilt_gyro += gyro * dt;

    tilt[tilt_index] = (((atan2(az_mean, ax_mean)*180/M_PI) + 90)
                        + (tilt_gyro)
                        + sx) / 3;

    tilt_index++;
    if (tilt_index >= numtilt)      tilt_index = 0;

    tilt_mean = 0;
    for (index = 0; index < numtilt; index++)
        tilt_mean += tilt[index];

    tilt_mean /= numtilt;

    return (tilt_mean);
    }

double trapezoid(const double x, double *bound)
    {
```

```
    if ((x <= bound[0]) || (x >= bound[3]))
        return(0);
    else if (x < bound[1])
        return((x - bound[0]) / (bound[1] - bound[0]));
    else if (x > bound[2])
        return((bound[3] - x) / (bound[3] - bound[2]));
    else
        return(1.0);
    }

unsigned int fuzzy(unsigned int position)
    {
    const double x = (double) position;
    const unsigned int numrules = 4;
    double rule[numrules][4] = {
                { 0, 0, 800, 1500 },
                { 1000, 2000, 2500, 4000 },
                { 1500, 4000, 5500, 8000 },
                { 5500, 8000, 10000, 10000 }  };

    double b[numrules] = {
                                    4000,
                                    1500,
                                    250,
                                    520   };

    double A[numrules];
    double Asum = 0;
    double Bsum = 0;

    for (unsigned int index = 0; index < numrules; index++)
        {
        A[index] = trapezoid(x, rule[index]);
        Asum += A[index];
        Bsum += b[index] * A[index];
        }

    return ((unsigned int) (Bsum / Asum));
    }

int main(int argc, char *argv[])
{
    const unsigned int samples = 1024;

    long int old_count = -1;
    unsigned int curdata[numdata];
    long int curcount, curmot_count, lastmot_count, stepper;
    unsigned int curmotor_cmd, curmotor_int, curposition;

    double dt = 0;
    double t = 0;
    double last_t = 0;

    int err = 0;
    unsigned int ok = 1;
    double tilt;
```

```
unsigned int index, sampindex;

unsigned int trials = 10;
char fname[13] = "xxx0000.log";
FILE *logfile;
int log_on = 0;

long int *mot_count;
unsigned int *motor_cmd, *position;
unsigned int *dataax, *dataaz, *datagy, *datasx;

unsigned long int axavg, gyavg, sxavg;

axavg = 0;
gyavg = 0;
sxavg = 0;

dataax = new unsigned int[samples];
dataaz = new unsigned int[samples];
datagy = new unsigned int[samples];
datasx = new unsigned int[samples];

mot_count = new long int[samples];
motor_cmd = new unsigned int[samples];
position = new unsigned int[samples];

for (index=0; index < numdata; index++)
     curdata[index] = 0;

for (index=0; index < samples; index++)
     {
     dataax[index] = 0;
     dataaz[index] = 0;
     datagy[index] = 0;
     datasx[index] = 0;
     position[index] = 0;
     motor_cmd[index] = 0;
     mot_count[index] = 0;
     }

printf("Calibrating sensors.  Please wait.\n");
for (index = 0; index < 5000; index++)
     {
     err = get4data(curdata, &curcount, old_count);

     if (err < 0)
          {
          printf("Startup data error = %i\n", err);
          exit(-1);
          }

     old_count = curcount;

     if ((err != 0) && (index > 0))
          {
          index--;
```

```
        }
    else
        {
        axavg += curdata[0];
        gyavg += curdata[2];
        sxavg += curdata[3];
        }
    }

axavg /= 5000;
gyavg /= 5000;
sxavg /= 5000;

if (argc > 2)
    {
    trials = atoi(argv[2]);
    if (trials > 9999)
        {
        printf("Maximum number of trials 9999.\n");
        return (-1);
        }
    }
printf("Trials = %i\n", trials);

if (argc > 1)
    {
    log_on = 1;
    sprintf(fname, "%.4s%04i.log", argv[1], 0);
    printf("Log on %s\n", fname);

    logfile = fopen(fname, "wb");
    fwrite(&samples, sizeof(samples), 1, logfile);
    }

filtered_tilt(curdata,1, axavg, gyavg, sxavg, dt);
tilt = 0;
stepper = 0;
lastmot_count = 0;

for (sampindex = 0; sampindex < trials; sampindex++)
    {
    if (ok == 0) break;

    if ( (sampindex > 0) && ((sampindex % 20) == 0) )
        {
        fclose(logfile);
        sprintf(fname, "%.4s%04i.log", argv[1], sampindex);
        logfile = fopen(fname, "wb");
        fwrite(&samples, sizeof(samples), 1, logfile);
        }

    for (index = 0; index < samples; index++)
        {
        if (ok == 0) break;

        err = get4data(curdata, &curcount, old_count);
```

```
            if (err < 0)
                 {
                 ok = 0;
                 printf("data error = %i\n", err);
                 break;
                 }

            dataax[index] = curdata[0];
            dataaz[index] = curdata[1];
            datagy[index] = curdata[2];
            datasx[index] = curdata[3];
            old_count = curcount;

            dt = t - last_t;
            tilt = filtered_tilt(curdata, 0, axavg, gyavg, sxavg, dt);
            curmotor_cmd = interp(tilt);

            err = motorcmd(curmotor_cmd,
                                 curmotor_int,
                                 &curposition,
                                 &curmot_count);

            if ((lastmot_count - curmot_count) > 600)
                 stepper += 65536;
            lastmot_count = curmot_count;
            last_t = t;
            t = 51.2e-6 * (double)(curmot_count + stepper);

            if (log_on == 0)
                 {
                 printf("[CT]=%4lx [AX]=%4x [AZ]=%4x [GY]=%4x
[SX]=%4x\n",
                                 curcount, curdata[0], curdata[1],
curdata[2], curdata[3]);
//               printf("CMD: %4i (%4i) -> %4i\n",
//                               curmotor_cmd, curmotor_int, curposition);
                 }

            motor_cmd[index] = curmotor_cmd;
            position[index] = curposition;
            mot_count[index] = curmot_count;

            curmotor_int = fuzzy(curposition);

            if (err < 0)
                 {
                 ok = 0;
                 printf("motor error = %i\n", err);
                 }
            }

        if (log_on)
            {
            fwrite(&sampindex, sizeof(sampindex), 1, logfile);

            fwrite(mot_count, sizeof(mot_count[0]), samples, logfile);
            fwrite(position, sizeof(position[0]), samples, logfile);
```

```
            fwrite(motor_cmd, sizeof(motor_cmd[0]), samples, logfile);

            fwrite(dataax, sizeof(dataax[0]), samples, logfile);
            fwrite(dataaz, sizeof(dataaz[0]), samples, logfile);
            fwrite(datagy, sizeof(datagy[0]), samples, logfile);
            fwrite(datasx, sizeof(datasx[0]), samples, logfile);

            printf("%4i\n", sampindex);
            }
        }

    fclose(logfile);

    delete mot_count;
    delete position;
    delete motor_cmd;
    delete dataax;
    delete dataaz;
    delete datagy;
    delete datasx;

    return (0);
}
```

Appendix V

Code for Mechanism Kinematics

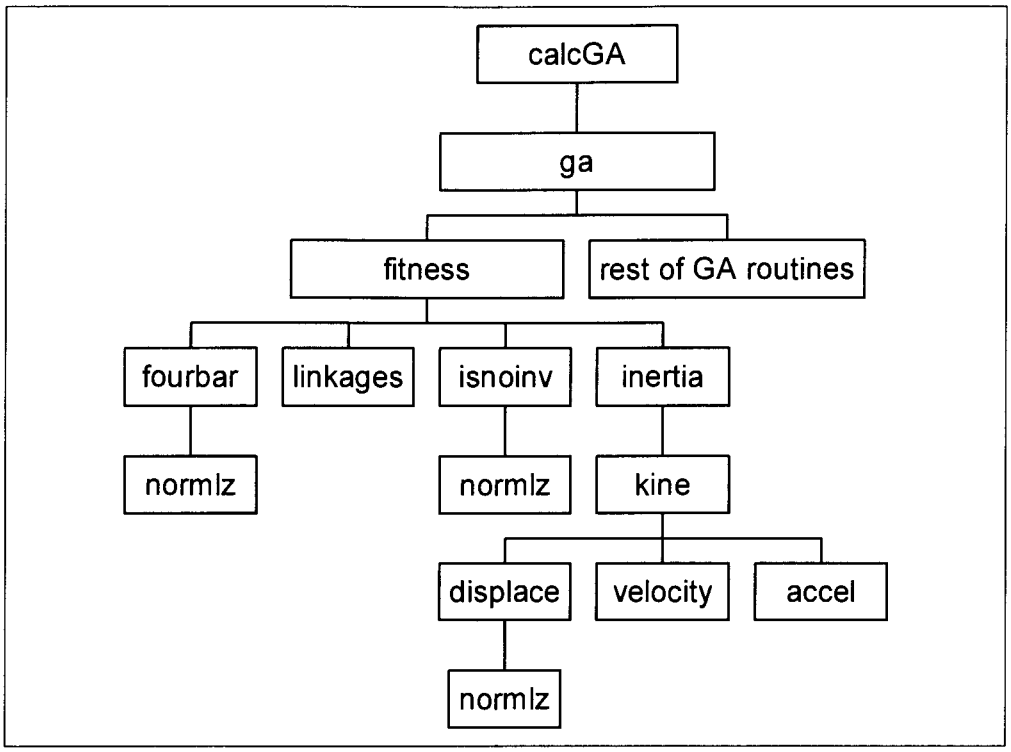
This appendix lists MATLAB code used to perform calculations on the angular position, velocity, acceleration, and torque under a gravitational field of a 4R-type four-bar kinematic linkage. The highest-level routine is responsible for applying the genetic algorithm optimization to the mechanism.

The following code is listed:

CALCGA.M	Calls all other routines
FITNESS.M	Calculates the fitness of a given individual in a population based on its kinematic characteristics.
FOURBAR.M	Calculates the mechanism link vectors at the initial position for half the mechanism given the ground pivot and three position vectors.
LINKAGES.M	Calculates the length of each linkage given four link vectors.
ISNOINV.M	Determines whether or not the mechanism undergoes a kinematic inversion in moving between three specified points.
INERTIA.M	Calculates the static torque required to move the mechanism at a given position, velocity, and acceleration.

KINE.M	Calculates the angular position, velocity, and acceleration for all links for a given range of input link angles.
DISPLACE.M	Calculates the angular position for all links.
VELOCITY.M	Calculates the angular velocity for all links given the position.
ACCEL.M	Calculates the angular acceleration given position and velocity.
NORMLZ.M	Utility to confine angles to within $\pm\pi$ radians.
GA.M	Genetic algorithm optimization function, from GAOT [48].

Table V.1 Software Organization Chart for GA Optimizer



CALCGA.M

```
function [BstX, endPop, bestSols, trace, initPop] = calcga(gen,
options, initPop, mutFN)

if (nargin < 1)
   gen = 1000;
end

if (nargin < 2)
```

```
   options = [0 0];
end

if (length(options) < 2)
   options = [options 0];
end

if (length(options) < 2)
   options = [options 0];
end

bounds = [-0.07 0.1705 ; 0.4230 0.5630 ; -0.07 0.1705 ; 0.4230 0.5630 ;
0.7 0.8 ; 0.5 0.8];

if (nargin < 3)
   initPop = initialz(1000, bounds, 'fitness');
end

if (nargin < 4)
   mutFN = 'unifm';
end

[BstX, endPop, bestSols, trace] = ga(bounds, 'fitness', [options(1)],
initPop, [5e-7 1 options(2)], 'maxgent', gen, 'normgeos', [0.08],
['arithx'], [2], mutFN, [2 1 3]);
```

FITNESS.M

```
function [val, sol] = fitness(sol,options)

% This function calculates the fitness of a solution for
% a given pair of pivot points rA and rB, and a defined
% set of 3 set locations R1, R2, and R3, defined by
% the h vector.  R1, R2, and R3 are vectors originating
% at the origin of the XZ plane located on the ground
% at the midpoint between the front and rear wheels of
% the wheelchair.  Z points up and X points forward.
% The lengths of the vectors are h(1), h(2), and h(3),
% respectively.  R1 points theta degrees forward from
% vertical, R2 points straight up along the Z axis, and
% R3 points theta degrees back.
% Each of the vectors is a single complex number, with
% the real component pointing along the x direction and
% the imaginary component along the z.

if (length(options) < 1)
   options = [0];
end

theta = 20 * pi / 180;
pi2 = 0.5 * pi;

ra = sol(1) + i*sol(2);
```

```
rb = sol(3) + i*sol(4);
h = [sol(5) 0.5 sol(6)];

R1 = h(1)*exp(i*(pi2 - theta));
R2 = h(2)*exp(i*(pi2));
R3 = h(3)*exp(i*(pi2 + theta));

Ba = [0 0 0];
Bb = Ba;

[Wa, Za, Ba] = fourbar(ra, theta, R1, R2, R3);
[Wb, Zb, Bb] = fourbar(rb, theta, R1, R2, R3);


% Rs1 defines the top back corner of the front bulkhead,
% with a 2cm clearance (actually, 2.8cm diagonal).
% Rs2 defines the top front corner of the rear bulkhead.
% Rlo defines the clearance to the bottom of the space.

Rs1 = (0.2125-0.02) + i*(0.08+0.02+0.263);
Rs2 = (-0.1595+0.02) + i*(0.08+0.02+0.263);
Rlo = i*(0.025+0.263);

% First, do the boolean pass/fail tests.
% Test 1: Clearance
% If any of the six clearance tests fails, T goes to 0 and
% we get out of here without doing any more time-consuming
% calculations.

AWa = abs(Wa);
AWb = abs(Wb);

T1a = abs(ra - Rs1) - AWa;     % Back corner clearance
T1b = abs(rb - Rs1) - AWb;

T2a = abs(ra - Rs2) - AWa;     % Front corner clearance
T2b = abs(rb - Rs2) - AWb;

T3a = imag(ra - i*AWa - Rlo); % Bottom clearance
T3b = imag(rb - i*AWb - Rlo);

T = (T1a > 0) * (T1b > 0) * (T2a > 0) * (T2b > 0) * (T3a > 0) * (T3b >
0);

if (T <= 0)
   val = -100000;
else

% Test 2: Inversion
% Check to see if the configuration in all three set points is the
same.
% i.e., we do not go through a kinematic inversion point.  Although it
is
% conceivable that we will go through an inversion and come back out
again
% between any two of the three set points, it is very unlikely for this
% particular problem domain.  If it does turn out to be a problem later
```

```
% on for other kinematic arrangements, we can always rewrite the code
% within the isnoinv routine without changing anything here.
% If this test is failed, then we must also fail the mechanism, as
% the four bar would pass through a kinematic inversion point, which
% wreaks havoc with smooth control.  As well, the kinematic inversion
% point is a singularity in the manipulator, requiring essentially
% infinite amounts of torque to get past statically.  (i.e., we
couldn't
% control the mechanism effectively if this inversion point is anywhere
% within the control space)

   lr = linkages(ra,Wa,Za, rb,Wb,Zb);
   [OK, gam] = isnoinv(lr, Ba,Bb);

   if (OK == 0)

      val = -10000;

   else

% Test 3: Length of 3rd bar.
% There is a tendency to reduce the torque by reducing the length of
% bar 3 (floater).  We wish to ensure that there is a minimum
separation
% of 0.07m (7cm) between the pivots.

      pr = abs(lr(3));

      if (pr < 0.07)

         val = -1000;

      else

% With the limit tests out of the way, we can develop an actual fitness
% coefficient.  The goal is to optimize the mechanical design of the
% four-bar, such that we minimize the distance hx away from the X-Z
% origin for all positions of the platform, maximize the range of
% positions over which we can supply torque, and minimize the amount
% of torque required at each position.

% For each (ra, rb, h1, h3) individual in the population, we can
% calculate the displacement, velocity, acceleration, and static
% torque for the input link (ground pivot at ra) for a distribution
% of points from one limit to the other.  The test function we will
% be using will consist of driving the input linkage at a constant
% angular velocity.  This is probably about as good as anything else,
% as far as choosing a test function goes.  We could use a more complex
% velocity/acceleration profile, such as that resulting from
compensating
% for a constant velocity charge onto and up a ramp, but we would need
% to use a dynamic torque model instead of the current static torque
one.

% We don't care, especially, about the sign of the torque, just its
% absolute value.  We can cheat a little by taking the mean of the
% absolute value of the torque profile we generate and using this as
```

```
% our metric.  The typical torque characteristic curve for this type
% of four-bar looks like a tangent function, with a couple of vertical
% asymptotes.


          bmeshsize = 100;

          beta = Ba(1):( (Ba(3)-Ba(1)) / (bmeshsize-1) ):Ba(3);
          a2 = zeros(size(beta));
          w2 = sign(Ba(3)-Ba(1))*ones(size(beta));

          [Ts, T,W,A] = inertia(a2,w2,beta,lr,-90);

          val = -mean(abs(Ts));

          if (options(1) == 1)
             fprintf(1,'%f\n', val);
          end

       end

    end

end
```

FOURBAR.M

```
function [W,Z, beta] = fourbar(Ra,theta,R1,R2,R3)

% Ra is a single complex number which encodes a vector,
% t is the angular limit (front and back), in degrees.
% Last updated July 27, 1997.

R1a = R1 - Ra;
R2a = R2 - Ra;
R3a = R3 - Ra;

del2 = R2a - R1a;
del3 = R3a - R1a;

a2 = theta;
a3 = 2*theta;

D1 = R3a*exp(i*a2) - R2a*exp(i*a3);
D2 = R1a*exp(i*a3) - R3a;
D3 = R2a - R1a*exp(i*a2);

beta(2) = normlz(2*angle(-D1) - angle(D2) - angle(D2*exp(i*a2)));
beta(3) = normlz(2*angle(-D1) - angle(D3) - angle(D3*exp(i*a3)));

A = [(exp(i*beta(2)) - 1)    (exp(i*a2) - 1) ;
     (exp(i*beta(3)) - 1)    (exp(i*a3) - 1) ];

res = inv(A) * [del2 ; del3];
```

```
W = res(1);
Z = res(2);
```

LINKAGES.M

```
function [r, lr, ar] = linkages(R1,W1,Z1, R2,W2,Z2)

r = zeros(6,1);

r(1) = R2 - R1;
r(2) = W1;
r(4) = W2;
r(5) = Z1;
r(6) = -Z2;
r(3) = r(1) + r(4) - r(2);

lr = abs(r);
ar = atan2(imag(r), real(r));
```

ISNOINV.M

```
function [OK, gam] = isnoinv(r,Ba,Bb)

gam = [0 0 0];
gam(1) = normlz( angle(r(4)) - angle(r(3)) );

r2 = r;
r2(2) = r2(2) * exp(i*Ba(2));
r2(4) = r2(4) * exp(i*Bb(2));
r2(3) = r2(1) + r2(4) - r2(2);
gam(2) = normlz(angle(r2(4)) - angle(r2(3)));

r3 = r;
r3(2) = r3(2) * exp(i*Ba(3));
r3(4) = r3(4) * exp(i*Bb(3));
r3(3) = r3(1) + r3(4) - r3(2);
gam(3) = normlz(angle(r3(4)) - angle(r3(3)));

OK = ( sign(gam(1)) == sign(gam(2)) )*( sign(gam(2)) == sign(gam(3)) );
```

INERTIA.M

```
function [Ts, T,W,A] = inertia(a2,w2,beta,r, gangle)

if (nargin < 5)
   gangle = -90;
end
```

```
i = sqrt(-1);
lr = abs(r);
ar = atan2(imag(r),real(r));

[A,W,T] = kine(a2,w2,beta,r,lr,ar);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Calculate inertial forces, return column vector Fb
% [F12x ; F12y ; Ts ; F23x ; F23y ; F34x ; F34y ; F14x ; F14y]
% a2 = angular acceleration of input link r2
% w2 = angular velocity of r2 (rel to inertial frame)
% beta = angular displacement of r2 rel. to tilt forward posn.
% r = vector of 6 complex vectors describing rl through r6
%     in tilt forward position #1.
%
% First, define common physical constants.
%
Aldensity = 2713; % kg/m^3 aluminum alloy density
Brdensity = 8609; % kg/m^3 brass density
g = 9.81;
sg = sin(gangle*pi/180);
cg = cos(gangle*pi/180);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Define Link 2 physical parameters
% Link 2 will be a simple Al rectangular bar with
% length (r2 + a bit on the ends), thickness 6.56mm,
% and width ~2.8cm.  Note that the pivot-to-pivot distance,
% though, is still defined by r(2).
% The density modifier is to account for the weight-saving
% holes to be drilled in the bar at uniform intervals along
% its length.  The CM of this bar will thus be in the middle.
% All units of length in meters, mass in kg, force in N.

l2len = lr(2) + (2 * 0.014);  % length
l2wid = 0.028;                % width
l2thk = 0.00656;        % thickness
l2densMod = 1.0;        % density modifier

m2 = l2densMod * Aldensity * l2len * l2wid * l2thk;    % mass of link 2
I2 = (m2 / 12) * ((l2len*l2len) + (l2wid*l2wid));      % moment of
inertia

rg2 = (lr(2) * 0.5);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Define Link 3 physical parameters
% Link 3 physically is an alumninum bar with four chunks of
% aluminum attached to the ends (2 on each end, 75.7g per pair).
% In addition, a chunk of posterboard sits on this link.

l3len = lr(3) - (2 * 0.023);  % length
l3wid = 0.028;                % width
l3thk = 0.00656;        % thickness
```

```
l3densMod = 1.0;        % density modifier

m3 = l3densMod * Aldensity * (l3len * l3wid * l3thk);
m3 = m3 + 0.154;        % weight of chunks
m3 = m3 + 0.350;        % weight of posterboard

I3 = (m3 / 12) * ((l3len*l3len) + (l3wid*l3wid));
I3 = I3 + 2*(0.0757 * (lr(3)*0.5 - 0.023 + 0.005)^2); % I of chunks
I3 = I3 + 2*(0.0757/12)*(0.025^2 + 0.06^2);

rg3 = r(5)*exp(i*(-ar(3)));

I3 = I3 + 0.350*(abs(rg3 + (0.116 + 0.180*i)*exp(i*( (pi/9)-ar(3))
))).^2;
I3 = I3 + (0.0537 * 1.25);    % I of posterboard

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Define Link 4 physical parameters
% Link 4 is pretty much the same as link 2, a plain
% rectangular bar.  We may later on want to add the
% moments of inertia of the brass rods at the ends.

l4len = lr(4) + (2 * 0.014);  % length
l4wid = 0.028;                % width
l4thk = 0.00656;        % thickness
l4densMod = 1.0;        % density modifier

m4 = l4densMod * Aldensity * l4len * l4wid * l4thk;    % mass of link 4
I4 = (m4 / 14) * ((l4len*l4len) + (l4wid*l4wid));      % moment of
inertia

rg4 = (lr(4) * 0.5) + i*(0);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Test code for Ex.5.3 from Erdman & Sandor
% Override mass, moment of inertia, and
% link definitions.
%
% rg2 = (lr(2) * 0.5);
% m2 = 0.8 / 2.205;
% I2 = 0.012 * ( 0.0254 * 9.81 / 2.205 );
%
% rg3 = (lr(3) * 0.5);
% m3 = 2.4 / 2.205;
% I3 = 0.119 * ( 0.0254 * 9.81 / 2.205 );
%
% rg4 = lr(4) * 0.5;
% m4 = 1.4 / 2.205;
% I4 = 0.038 * ( 0.0254 * 9.81 / 2.205 );
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

phi2 = angle(rg2)*ones(size(T(2,:)));
phi3 = angle(rg3)*ones(size(T(3,:)));
phi4 = angle(rg4)*ones(size(T(4,:)));
ri2 = lr(2).*exp(i*T(2,:));
```

```
rgi2 = rg2.*exp(i*T(2,:));
ri3 = lr(3).*exp(i*T(3,:));
rgi3 = rg3.*exp(i*T(3,:));
ri4 = lr(4).*exp(i*T(4,:));
rgi4 = rg4.*exp(i*T(4,:));


% Now that we've defined all the nasty physical constants,
% we can figure out what all the accelerations, inertial forces,
% and inertial torques are.

Ag2 = abs(rgi2).*(i*A(2,:) - W(2,:).^2).* exp(i.*(T(2,:) + phi2));
Ag4 = abs(rgi4).*(i*A(4,:) - W(4,:).^2).* exp(i.*(T(4,:) + phi4));
A2 = abs(ri2).*(i*A(2,:) - W(2,:).^2).* exp(i.*(T(2,:)));
A4 = abs(ri4).*(i*A(4,:) - W(4,:).^2).* exp(i.*(T(4,:)));
Ag3 = ((abs(rgi3)/lr(3)) .* exp(i.*phi3) .* (A4 - A2)) + A2;

Fo2 = -m2 * Ag2;
Fo3 = -m3 * Ag3;
Fo4 = -m4 * Ag4;

To2 = -I2 * A(2,:);
To3 = -I3 * A(3,:);
To4 = -I4 * A(4,:);

Fi = [      real(Fo2) ;
      imag(Fo2) ;
      To2 ;
      real(Fo3) ;
      imag(Fo3) ;
      To3 ;
      real(Fo4) ;
      imag(Fo4) ;
      To4 ];

Fg = [      m2*g*cg ;
      m2*g*sg ;
      0 ;
      m3*g*cg ;
      m3*g*sg ;
      0 ;
      m4*g*cg ;
      m4*g*sg ;
      0];

for n=1:length(T(2,:)),
   L = [-1, 0, 0, 1, 0, 0, 0, 0, 0 ;
        0, -1, 0, 0, 1, 0, 0, 0, 0 ;
        -imag(rgi2(n)), real(rgi2(n)), -1, imag(rgi2(n)-ri2(n)),
real(ri2(n)-rgi2(n)), 0, 0, 0, 0 ;
        0, 0, 0, -1, 0, 1, 0, 0, 0 ;
        0, 0, 0, 0, -1, 0, 1, 0, 0 ;
        0, 0, 0, -imag(rgi3(n)), real(rgi3(n)), imag(rgi3(n)-ri3(n)),
real(ri3(n)-rgi3(n)), 0, 0 ;
        0, 0, 0, 0, 0, -1, 0, -1, 0 ;
        0, 0, 0, 0, 0, 0, -1, 0, -1 ;
```

```
        0, 0, 0, 0, 0, imag(ri4(n)-rgi4(n)), real(rgi4(n)-ri4(n)), -
imag(rgi4(n)), real(rgi4(n)) ];

   Fb = L\(Fi(:,n) + Fg);
   Ts = [Ts Fb(3)];
end
```

KINE.M

```
function [A, W, T] = kine(a2, w2, beta, r,lr,ar)

% beta is displacement angle measured relative
% to a zero at the starting (tilt forward) position.
% beta must be a row vector.
% w2 and a2 must be row vectors of the same
% size as beta.

[n,m] = size(beta);
if (n > m)
   beta = beta';
end

[n,m] = size(w2);
if (n > m)
   w2 = w2';
end

[n,m] = size(a2);
if (n > m)
   a2 = a2';
end

[t1,t2,t3,t4,P] = displace(r,lr,ar, beta);
[w3,w4] = velocity(w2, t1,t2,t3,t4, lr);
[a3,a4] = accel(a2,w2,w3,w4, t1,t2,t3,t4, lr);

T = [t1 ; t2 ; t3 ; t4];
blank = zeros(size(t1));

W = [blank ; w2 ; w3 ; w4 ];

A = [blank ; a2 ; a3 ; a4 ];
```

DISPLACE.M

```
function [t1 , t2 , t3 , t4 , P] = displace(r,lr,ar, beta, index)

% Calculates displacements for a given range of beta, where
% beta is a row vector of input angles for the joint angle
% index, where index is between 1 and 4.  If index is omitted,
% it is 2 by default.
```

```
if (nargin < 5)
   index = 2;
end

if (index < 1)
   index = 1;
end

if (index > 4)
   index = 4;
end

[n,m] = size(beta);
if (n > m)
   beta = beta';
end

t1 = ones(size(beta)) * ar(1);
t2 = zeros(size(beta));
t3 = zeros(size(beta));
t4 = zeros(size(beta));

if (index == 1)
elseif (index == 2)

   t2 = beta + ar(2);
   gamma = normlz(ar(4) - ar(3));
   alpha = normlz(ar(5) - ar(3));

   D = r(1) - r(2)*exp(sqrt(-1)*beta);
   lD = abs(D);
   aD = atan2(imag(D), real(D));

   delta = sign(gamma)*abs(acos( (lr(3).^2 + lD.^2 - lr(4).^2) ./
(2*lr(3).*lD) ));
   sigma = sign(gamma)*abs(acos( (lr(4).^2 + lD.^2 - lr(3).^2) ./
(2*lr(4).*lD) ));

   t3 = aD + delta;
   t4 = aD + (pi - sigma);

   P = r(2).*exp(sqrt(-1)*(t2-ar(2))) + r(5).*exp(sqrt(-1)*(t3 -
ar(3)));


elseif (index == 3)
elseif (index == 4)

   t4 = beta;

else
end
```

VELOCITY.M

```
function [w3, w4] = velocity(w2, t1,t2,t3,t4, lr)

w3 = -w2.*(lr(2)/lr(3)).*sin(t4-t2)./sin(t4-t3);
w4 = w2.*(lr(2)/lr(4)).*sin(t3-t2)./sin(t3-t4);
```

ACCEL.M

```
function [a3,a4] = accel(a2,w2,w3,w4, t1,t2,t3,t4, lr)

a3 = (-lr(2)*a2.*sin(t4-t2) + lr(2)*(w2.^2).*cos(t4-t2) +
lr(3)*(w3.^2).*cos(t4-t3) - lr(4)*(w4.^2)) ./ (lr(3)*sin(t4-t3));

a4 = (lr(2)*a2.*sin(t3-t2) - lr(2)*(w2.^2).*cos(t3-t2) +
lr(4)*(w4.^2).*cos(t3-t4) - lr(3)*(w3.^2)) ./ (lr(4)*sin(t3-t4));
```

NORMLZ.M

```
function angleout = normlz(anglein)

while (anglein > pi)
   anglein = anglein - (2*pi);
end

while (anglein <= -pi)
   anglein = anglein + (2*pi);
end

angleout = anglein;
```

Appendix VI

Survey on Powered Wheelchair Usage

In March of 1994, a survey on powered wheelchair usage [15] was sent out via the Internet news group misc.handicap to solicit opinions on possible research directions for improving powered wheelchairs from actual users. A total of 15 replies were received over the course of a couple of months. The results of this survey combined with anecdotal data gathered from members of the local wheelchair-using community drove the initial direction of research on this thesis. This appendix contains the text of this survey.

On average, the respondents were fairly satisfied with their current wheelchairs. Virtually all the respondents have used other wheelchairs in the past, and so have some basis of comparison for making suggestions on wheelchair improvements. Common concerns seen among the respondents' replies were reliability and access to service, mobility over different terrain types, and price. These concerns fit well with similar comments made by local members of the disabled community.

Hello, we are a group of engineering (and other) students from Simon Fraser University who are planning to design a powered wheelchair that focusses more on the individual than on mechanics and electronics. We have dubbed our project the Personal Vehicle and at this stage of the development we are researching various existing wheelchairs and how they meet the needs of the public.

We would like to ask that anyone currently using a powered wheelchair please take a moment and respond to the questions that follow. We would be very grateful for any input you could give us now or in the future. You can send your responses directly to us, Greg or Bolko at whall@sfu.ca (Greg) or brawicz@sfu.ca (Bolko).

Thank you very much for your time and your input.

Survey of Powered Wheelchair Users

1. What make of powered wheelchair do you currently use?
2. How satisfied are you with your current wheelchair?
3. What things do you like/dislike about your wheelchair?
4. Have you used other wheelchairs before your current one?

   If so, what thing(s) prompted you to switch to your current wheelchair?
5. What performance features would you like to see on a powered wheelchair in order to make it more usable in daily living?

Demographics
Age:
Brief description of disability:

Once again thank you very much for your time and participation. The information you have provided us with will be very useful in the development of The Personal Vehicle.

Sincerely,

Bolko and Greg

Responsible for
Public Interest Research for the
Personal Vehicle Design Team

References

[1] C.G. Warren, "Powered mobility and its implications," *J. Rehab. Res. Dev.,* Clinical Supplement #2, pp. 74-85, 1990.

[2] K.T. Ragnarsson, "Prescription considerations and a comparison of conventional and lightweight wheelchairs," *J. Rehab. Res. Dev.,* Clinical Supplement #2, pp.8-16, 1990.

[3] M.W. Ferguson-Pell, "Seat cuhion selection," *J. Rehab. Res. Dev.,* Clinical Supplement #2, pp.49-73, 1990.

[4] K.V. Frolov and F.A. Furman, *Applied Theory of Vibration Isolation Systems.* New York: Hemisphere, 1990.

[5] *The Illustrated Directory of Handicapped Products, 1991-92.* M.S. Behzad, ed. Trio Publications, 1991.

[6] D.F. Wilkes, "Vehicular suspension system," U.S. Patent 4,247,127, Jan. 1981.

[7] W.N. Hosaka, "Wheelchair," U.S. Patent 4,455,031, Jun. 1984.

[8] G.Y. Duffy, Jr., "Folding wheelchair with improved frame and suspension system," U.S. Patent 4,861,056, Aug. 1989.

[9] M.J. Quintile, "Suspension for seat of powered wheel chair," U.S. Patent 5,145,020, Sept. 1992.

[10] B.T. Peters, "Multi-terrain wheelchair," U.S. Patent 5,507,513, Apr. 1996.

[11] B.H. Thibodeau, "All-terrain, all-weather wheelchair," U.S. Patent 5,518,081, May 1996.

[12] P.J. Scheulderman, "Wheelchair," U.S. Patent 5,564,512, Oct. 1996.

[13] R.J. Kwiatkowski and R.M. Iñigo, "A closed loop automated seating system," *J. Rehab. Res. Dev.*, vol. 30, no. 4, pp. 393-404 1993.

[14] R. Plautz and B. Timen, "Positioning can make the difference," *Nursing Homes*, pp. 30-33, Jan/Feb 1992.

[15] B. Rawicz, W.G. Hall, and W.W. Li, "Survey of Powered Wheelchair Users," unpublished, May, 1994.

[16] J.E. Ericsson, "Cross-country vehicle or machine," U.S. Patent 4,991,673, Feb. 1991.

[17] H.W. Van Gerpen, "Active seat suspension control system," U.S. Patent 4,363,377, Dec. 1982.

[18] G.G. Goertzen, N.J. Curran, and J.H. Molnar, "Powered wheelchair with adjustable center of gravity and independent suspension," U.S. Patent 5,575,348, Nov. 1996.

[19] G. Liprandi et al., "Anticentrifugal active lateral suspension for railway vehicles," U.S. Patent 5,454,329, Oct. 1995.

[20] G.G. Majaess, R.L. Kirby, S.A. Ackroyd-Stolarz, and P.B. Charlebois, "Influence of seat position on the static and dynamic forward and rear stability of occupied wheelchairs," *Arch. Phys. Med. Rehabil.*, vol. 74, pp. 977-982, 1993.

[21] R.L. Kirby, M.T. Sampson, F.A.V. Thoren, and D.A. MacLeod, "Wheelchair stability: effect of body position," *J. Rehab. Res. Dev.*, vol. 32, no. 4, pp. 367-372, 1995.

[22] R.L. Kirby, B.D. Ashley, S.A. Ackroyd-Stolarz, "Static rear and forward stability of occupied wheelchairs: effect of the magnitude and position of added loads," (abstract) *Arch. Phys. Med. Rehabil.*, vol. 73, p. 1014, 1992.

[23] R.L. Kirby, S.M. Atkinson, and E.A. MacKay, "Static and dynamic forward stability of occupied wheelchairs: influence of elevated footrests and forward stabilizers," *Arch. Phys. Med. Rehabil.*, vol. 70, pp. 681-686, 1989.

[24] Sunrise Medical, Quickie Division webpage, http://www.quickie.com

[25] Permobil web page, http://www.permobil.com

[26] Invacare web page, http://www.invacare.com

[27] R.A. Cooper and M. MacLeish, "Racing wheelchair roll stability while turning: a simple model," *J. Rehab. Res. and Dev.*, vol. 29, pp. 23-30, 1992.

[28] *Fortress Mini Catalog*, 1992.

[29] J.B. Redford, "Seating and wheeled mobility in the disabled elderly population," *Arch. Phys. Med. Rehabil.*, vol. 74, pp. 877-885, 1993.

[30] Statistics Canada, *Health and Activities Limitation Survey*. 1991.

[31] G.M. Yarkony, "Pressure ulcers: a review," *Arch. Phys. Med. Rehabil.*, vol. 75, pp. 908-917, 1994.

[32] C.L. George, "The posture montior: an automated prompting device for body alignment," *IEEE J. Spec. Educ.*, vol. 10, no. 2, pp. 147-152, 1992.

[33] J. Vandyke, personal communication, May 2, 1994.

[34] J.M. Koerlin, J.L. Tausz, "Zero shear recliner/tilt wheelchair seat," U.S. Patent 5,297,021, Mar. 1994.

[35] D. Karnopp, "Active and semi-active vibration isolation," *Trans. ASME, Spec. 50th Anniv. Spec.*, vol. 117, June 1995.

[36] W.T. Dempster, *Space Requirements of the Seated Operator: Geometrical, Kinematic, and Mechanical Aspects of the Body With Special Reference to the Limbs*. WADC Tech Report 55-159. Wright-Patterson Air Force Base, OH: Wright Air Development Center, July 1955.

[37] M. Alirand, H. Lachaize, M. Lebrun, "Study and anlysis of an active self levelling suspension," *IEEE Sys. Man Cyb. 1993 Int'l Conf.*, pp. 222-227, 1993.

[38] R.L. Kirby, S.A. Ackroyd-Stolarz, M.G. Brown, S.A. Kirkland, and D.A. MacLeod, "Wheelchair-related accidents caused by tips and falls among noninstitutionalized

users of manually propelled wheelchairs in Nova Scotia," *Am. J. Phys. Med. Rehabil.*, vol. 73, no. 5, pp. 319-330, 1994.

[39] *Wheelchairs – Part 1: Determination of static stability.* Stockholm: International Organization for Standardization, 1986: ISO 7176-1.

[40] *Wheelchairs – Part 2: Determination of dynamic stability of electric wheelchairs.* Stockholm: International Organization for Standardization, 1990: ISO 7176-2.

[41] C.E. Brubaker, C.A. McLaurin, and I.S. McClay, "Effects of side slope on wheelchair performance," *J. Rehabil. Res. Dev.*, vol. 23, no. 2, pp. 55-57, 1986.

[42] *Working Model v3.0.3.* Knowledge Revolution, San Mateo, California, 1995.

[43] F.P. Beer and E.R. Johnston Jr., Vector Mechanics for Engineers: Dynamics (2nd S.I. Metric Edition). McGraw-Hill, Singapore, 1990.

[44] A.W. Burton, A.J. Truscott, and P.E. Wellstead, "Analysis, modelling, and control of an advanced automotive self-levelling suspension system," *IEEE Proc. Control Theory Appl.*, vol. 142, no. 2, March 1995.

[45] A.G. Erdman and G.N. Sandor, *Mechanism Design: Analysis and Synthesis.* Englewood Cliffs, New Jersey: Prentice-Hall, 1991.

[46] R.J. Loerch, A.G. Erdman, G.N. Sandor, and A. Midha, "Synthesis of four-bar linkages with specified ground pivots," *Proc 4th Applied Mechanisms Conference (Chicago, November 1975).* pp. 10.1-10.6, Stillwater, Oklahoma: Oklahoma State University, 1975.

[47] R. Fletcher, *Practical Methods of Optimization.* 2nd ed. Chichester: Wiley, 1987.

[48] C.R. Houck, J.A. Joines, and M.G. Kay, "A genetic algorithm for function optimization: a MATLAB implementation," NCSU-IE TR 95-09, 1995.

[49] P.P. Acarnley, *Stepping Motors: A Guide to Modern Theory and Practice*. 3rd ed. London: Peter Peregrinus, 1992.

[50] T. Kenjo, *Stepping Motors and their Microprocessor Controls*. Oxford: Clarendon, 1984.

[51] M. Sugeno and T. Yasukawa, "A fuzzy-logic-based approach to qualitative modelling," *IEEE Trans. Fuzzy Sys.*, vol. 1, no. 1, pp. 7-31, 1993.

[52] B. Barshan and H.F. Durrant-Whyte, "Inertial navigation systems for mobile robots," *IEEE Trans. Rob. Aut.*, vol. 11, no. 3, pp. 328-342, 1995.

[53] R.A. Cooper, D.P. VanSickle, S.J. Albright, K.J. Stewart, M. Flannery, and R.N. Robertson, "Power wheelchair range testing and energy consumption during fatigue testing," *J. Rehab. Res.*, vol. 32, no. 3, pp. 255-263, 1995.

[54] K.M. Lee and D.K. Shah, "Kinematic analysis of a three-degrees-of-freedom in-parallel actuated manipulator," *IEEE J. Rob. Aut.*, vol. 4, no. 3, pp. 354-360, 1988.

[55] K.M. Lee and D.K. Shah, "Dynamic analysis of a three-degrees-of-freedom in-parallel actuated manipulator," *IEEE J. Rob. Aut.*, vol. 4, no. 3, pp. 361-367, 1988.

[56] W.A. Bullough, "Electro-rheological fluids: an introduction for biomedical applications," *J. Biomed. Eng.*, vol. 13, pp. 234-238, 1991.

[57] *Advanced Magnetic Fluid Formulations: Rheonetic Magnetic Fluids & Systems*. Lord Corporation, 1997.