Position Sensing and Control of a Linear Synchronous Motor

by

Tracy M. Clark

S.B., Electrical Engineering
Massachusetts Institute of Technology
(1983)

S.M., Electrical Engineering
Massachusetts Institute of Technology
(1990)

Submitted to the Department of
Electrical Engineering and Computer Science
in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Science
at the
Massachusetts Institute of Technology
May, 1995

Signature of Author _____
May 26, 1995

Certified by _____
Richard D. Thornton
Thesis Supervisor

Accepted by _____
Frederic R. Morgenthaler
Chairman, Departmental Committee on Graduate Students

Position Sensing and Control of a Linear Synchronous Motor

by

Tracy M. Clark

Submitted to the Department of
Electrical Engineering and Computer Science
on May 26, 1995
in partial fulfillment of the requirements for the
degree of Doctor of Science

Abstract

Linear Synchronous Motor (LSM) technology is well suited to providing propulsion for a wide range of surface transportation systems, from low speed freight systems using wheels for suspension to high speed magnetically levitated (MagLev) passenger vehicles. There are a variety of technical challenges associated with designing and implementing a multiple vehicle, multiple zone LSM that have not been fully addressed in prior research. A number of contributions relevant to this field of research have resulted from this thesis project.

Measurements of the vehicle position with an accuracy on the order of one electrical degree (which corresponds to a displacement of about 1 cm in a high speed MagLev system) must be made available to the wayside electronics to ensure smooth and efficient synchronous motor operation. A novel position sensing system is developed which utilizes signals injected into the stator winding by a transducer mounted onboard the vehicle to determine the vehicle position. A nonlinear observer is used to track the vehicle state.

A wide bandwidth controller is required to maintain synchronous motor operation. A two-tiered control architecture is employed to realize an effective and inexpensive controller implementation. Dedicated zone controllers which handle the high-speed data-intensive computations required to sustain LSM synchronization are distributed along the guideway, and each zone controller concentrates its effort on a single vehicle. A higher level central dispatch controller views each vehicle in a more abstract manner, and performs the scheduling and routing tasks necessary to ensure a coordinated flow of traffic and the proper spacing between all of the vehicles in a region.

A helical shape is used for the LSM stator winding. This winding shape is simpler and cheaper to manufacture and improves the electrical characteristics of the motor.

A laboratory scale demonstration system has been constructed to develop and test these innovations. Measurements of the position sensor indicate an error of only 0.1°. The control architecture supports the control of vehicles around a multiple-zone guideway. Inter-zone communication links allow a coordinated hand-off of control as a vehicle crosses a zone boundary with no discontinuity in motor thrust.

Thesis Committee: Professor Richard D. Thornton (chairman)
Professor James K. Roberge
Professor Jeffrey H. Lang

Acknowledgements

The completion of this thesis would not have been possible without the aid of many people. I would like to take this opportunity to thank the following people and groups for their help and support.

This project was supported by the Federal Railway Administration, including work completed as part of the National MagLev Initiative. In addition, support was provided by the Volpe National Transportation Systems Center under a contract with the MIT Center for Transportation Studies. George Anagnostopoulos, Mike Coltman, and Lena Valavani of the Volpe Center were helpful throughout this project.

I would also like to thank the members of my thesis committee for their efforts. Professor Thornton, the chairman, and Professors Lang and Roberge all contributed to the success of this project.

The students, faculty, and staff in the Laboratory for Electromagnetic and Electronic Systems (LEES) made the work involved with this project more pleasurable. They were willing to offer advice, equipment, or just plain good humor to keep things moving smoothly. Vivian Mizuno was particularly helpful at eradicating any bureaucratic or administrative problems that arose, and Professor James Kirtley led me to the helical winding technique. I am also grateful for the equipment furnished by the friendly staff working in the Undergraduate Teaching Laboratory. The MIT Plasma Fusion Center generously provided space when the demonstration system outgrew the LEES lab.

I am most indebted to the students who provided a great deal of the labor required to construct the first prototype and demonstration systems. The MIT undergraduate research opportunities program (UROP) provides a valuable service both to the MIT research community and to the students. Dan Zahn and Matt Trevithick constructed some of the electronics, Victor Liau and Umair Khan wrote early versions of the controller code, Ken Peters worked on the guideway design, Tri Tran and Andrew Chen machined and assembled components for the vehicles, and Lawrence Chang, Steve Schlueter, and David Rodriguez machined and assembled most of the guideway. Perhaps the most important benefits gained from this project are the educational opportunities it provided for me as well as the graduate and undergraduate students who assisted in the construction and testing of the demonstration system.

Table of Contents

List of Figures

2. Demonstration System Design

3. Position Sensor

4. Wayside Electronics Design

Appendices

List of Tables

1. Introduction

background

The quest for a faster or a more efficient means of transporting people and freight has engaged inventors, researchers, and engineers through the ages. Developments in the 20th century include the birth of new modes of transportation such as airlines and automobiles as well as the refinement of older modes of transportation such as rail. Improvements in transportation technology foster the growth of business, which increases the demand for further transportation improvement. The lifestyle of today's population relies on a variety of transportation services; rail, trucks, and ships to haul freight; airlines and automobiles for business or vacation travel; automobiles, busses, and light rail for commuting to work. The demand for increased transportation capacity in urban areas is large and growing, but the availability of room for expansion in these urban locations is tight. Increasing the system capacity for commuting into or travel between urban areas requires innovation, simply building a larger system using conventional ideas is not practical and will not meet future demands.

The focus of this thesis is to develop techniques suitable for use in controlling a linear synchronous motor (LSM) in a surface transportation system. In addition, an improved method of producing the stator winding for the LSM was discovered while constructing the demonstration system guideway. The material presented in this thesis concentrates on a high speed passenger oriented transportation system, but most of the concepts apply directly to freight and lower speed passenger systems. The research activity associated with the development of high speed passenger service is largely centered in Japan and Europe. A survey of systems in service and under development is reviewed in the following section to provide a context for the concepts presented in this thesis [1].

current research

The Shinkansen 'bullet train' is a high speed rail system connecting the major cities in Japan. It is used primarily by commuters, and operates with vehicles that offer either express service between major terminals or local service to small terminals. When originally placed in service in 1964, the top speed capability was 200 km/h (124 mph). The current system has a top speed of 274 km/h (170 mph) and an experimental series vehicle is under development that reaches a top speed of 350 km/h (217 mph). The Shinkansen uses rotary electric motors to provide traction via steel wheels on

rails. Propulsion power is transferred to the vehicle through a catenary. The experimental series utilizes propulsion motors in passenger cars; 10 of the 16 cars making up a train set are powered. The daily ridership of this system is quite large; 400,000 passengers travel on 260 trains each day covering a distance that circles the globe 3 times. A maintenance crew is dispatched nightly to realign and repair the rail bed. The maintenance costs associated with operation of this system are relatively high, approximately equal to energy costs.

The Japanese MLU002 is a high speed magnetically levitated (MagLev) passenger system. This system operates in a test facility and is not intended for immediate commercial application. The MLU002 represents the current state of an evolving series of systems. It incorporates an electrodynamic suspension (EDS) to provide lift and guidance forces. A superconducting magnet array on board the vehicle induces currents in the null-flux suspension and guidance coils located in the sidewalls of the trough shaped guideway. Interaction of the coil currents and the field of the magnet array creates an 'electromagnetic spring' that maintains the correct relative position of the vehicle with respect to guideway. The vehicle nominally operates at a distance of 10 cm from the guideway. An active guideway LSM provides the thrust to accelerate the vehicle. Wayside power electronics circuitry drives current in the multiple phase stator winding embedded in the guideway. The stator current interacts with the field from the vehicle magnet array (the same array used for the suspension system) to produce a propulsive force on the vehicle. The superconducting vehicle magnets are located in bogies placed at the ends of each car, and occupy approximately 25% of the vehicle length. This magnet placement was designed to allow the maximum distance between passengers and the magnets, thus reducing the magnetic field level in the vicinity of the passengers. The MLU002 operates at speeds approaching 500 km/h (311 mph).

The Transrapid TR07 is currently operational at the test facility in Emsland, Germany. This system was primarily designed to provide high speed MagLev passenger service, but a vehicle can be configured to haul freight. An electromagnetic suspension (EMS) system is utilized to generate lift and guidance forces. A port and a starboard array of electromagnets (made of copper wire wound on a steel core) produce an attractive force to a pair of laminated steel rails mounted on the guideway. An onboard control system senses the gap between the vehicle and the guideway and drives the electromagnets with the current level necessary to maintain a nominal gap of 8 mm. An active guideway LSM is used to produce a propulsive force for the vehicle. Wayside power electronics circuitry drives current in a multiple

phase stator winding embedded in slots in the laminated steel rails on the guideway. The interaction between the stator current and the field from the vehicle magnets (the same magnets used in the suspension system) results in vehicle thrust. The vehicle magnets are placed along the entire length of the vehicle, resulting in a very even weight distribution. The steel used in the magnetic circuit of the suspension and LSM shields the passenger compartment from large magnetic fields. The Transrapid TR07 has been operated at a maximum speed of 450 km/h (280 mph) at the Emsland test facility. A commercial link between Hamburg and Berlin is planned for operation in 2005.

Germany has also developed a high speed rail system, the InterCity Express (ICE) train. This system uses rotary electric motors to provide traction through steel wheels and propulsion power is transferred to the vehicle via an overhead catenary. The ICE was designed to provide relatively high speed passenger service through the somewhat hilly terrain in Germany, and has sealed cabins to shield passengers from pressure gradients as the vehicle enters or exits a tunnel. The vehicle normally operates at a maximum speed of 300 km/h (186 mph). An experimental version of this system can reach a top speed of 405 km/h (252 mph).

The Train à Grande Vitesse (TGV) system was developed in France. The TGV is a high speed rail system designed to provide fast and comfortable transportation for passengers. The vehicle is propelled by rotary electric motors that produce traction through steel wheels. Propulsion power is transferred to the vehicle via an overhead catenary and pantograph. The TGV operates at a top speed of 300 km/h (186 mph) in normal service, and has reached a speed of 515 km/h (320 mph) in a test run.

The terminal transport at the Birmingham, England airport is used to shuttle passengers from the terminal to baggage area. This low speed system uses an EMS MagLev suspension to generate lift and guidance forces and a linear induction motor (LIM) to provide propulsion. The vehicle travels at a speed of 54 km/h (34 mph) with a levitation gap of 15 mm. Perhaps the most notable feature of this system is its impressively low operating cost. Since there is no contact between vehicle and guideway and virtually no moving parts on the vehicle, the required maintenance is minimal. The control system is fully automated to eliminate the need for an operator; thus the energy usage dominates the operating cost of the system.

The Skytrain transit system in Vancouver, British Columbia is unique in that it uses conventional steel wheels and rails for suspension and guidance, and a LIM for propulsion. The LIM propulsion system provides traction in all weather conditions with an energy consumption claimed to be

30% of that used by an equivalent rotary motor and gearbox. This system is designed for urban transit and operates at a speed of 86 km/h (53 mph).

research trends

Research and development of high speed passenger transportation systems has been spread geographically through Germany, France, Japan, Sweden, Italy, England, Canada, and the United States, and temporally over the past three decades. A broad range of ideas, problems, and solutions have resulted; some innovations have proved practical while others pass with little recognition. There are some basic trends that appear repeatedly through the body of research and bear recognition.

Higher vehicle speed is a continuing goal of transportation research and development. Speed is an important metric for gauging both the degree of technical achievement and the market worth of a system. (One could easily argue that too much emphasis is placed on this single performance metric.) While speeds just over 161 km/h (100 mph) were obtainable in the early 1900's, actual operating speeds were much slower. High performance rail systems are currently operating near 322 km/h (200 mph) and MagLev systems achieve speeds over 402 km/h (250 mph).

Modern vehicle designs are more aerodynamically streamlined than their earlier counterparts. Lower aerodynamic drag may stem partially from a more thorough understanding of the principles involved or from the more sophisticated computational models available with improving computer hardware, but the primary motivation is linked to the quest for higher vehicle speed. Aerodynamic drag increases dramatically as speed increases, thus a streamlined vehicle offers higher speed with the same propulsion capability. The traction available from a steel wheel on rail interface is limited by the coefficient of friction, thus the propulsion capability of conventional rail systems cannot be increased simply by using a motor with a larger power rating. An aerodynamically clean vehicle also produces less noise, especially at higher vehicle speeds where aerodynamic noise dominates. Increased passenger comfort and decreased environmental impact are derived from improved vehicle aerodynamics.

Improvements in materials, equipment, and design have resulted in lighter vehicle weight. A lighter vehicle is helpful in reducing both the capital and the operating costs associated with the guideway, since a lighter and more distributed weight profile results in a smaller structural load on the guideway, less wear on the rails, and a lower power requirement for accelerating the vehicle. In addition, a lighter vehicle produces less noise and vibration through guideway contact.

A few systems, including the Swedish X2000 and the Italian Pendolino, have incorporated a tilting mechanism on the vehicle. The tilting capability provides passengers with a coordinated turn, thus allowing the vehicle to travel through tight radius turns at high speed while maintaining an acceptable level of passenger comfort.

Several systems include propulsion capability in most or all of the cars making up the vehicle rather than the conventional approach of placing a locomotive at the front (and perhaps the rear) of the vehicle to propel passive passenger cars. A powered passenger car configuration evenly distributes the weight profile of the vehicle, which helps to reduce guideway loading and wear. Distributing traction over many wheels also increases the limit on propulsion capability.

All of the modern high speed passenger transportation systems utilize an electric motor for propulsion. Virtually all wheeled systems rely on rotary motors located on the vehicle for traction, and all of the MagLev systems use either a LIM or LSM for propulsion. The German Transrapid designs used a LIM for propulsion in earlier models, but advances in power electronics, control electronics, and superconducting magnets have allowed more efficient LSMs to replace the LIMs. Many of the advantages of LSM technology are also applicable to wheeled transportation systems, but a passenger system combining LSM propulsion with wheeled suspension has not yet been developed.

Automated control systems are becoming more common due to both an improvement in the capability of inexpensive digital control hardware and an increase in the level of sophistication required to operate vehicles at higher speeds and tighter headways. The limitations associated with human computational capability, attention span, and reaction time are inadequate to meet the demands of higher speed, higher capacity transportation systems, and the role of the train operator is either disappearing or being limited to the handling of emergency procedures.

Demands on system capacity are especially intense in urban areas. The currently most common method of increasing system capacity is to increase the length of the vehicle, since decreasing headway imposes higher demands on the control system and on vehicle acceleration and deceleration capability. In systems using steel wheel / rail traction (this includes virtually all systems in current commercial use), the ability to increase acceleration and deceleration rates is limited. As the use of linear motor propulsion becomes more widespread, capacity increase via shorter headway should be possible and result in improved passenger service.

The primary emphasis of the research effort in this project is to develop techniques for controlling a LSM in a transportation system. The major issues addressed are the need for an accurate method of sensing the position of the vehicle relative to the stator winding and a control architecture capable of supporting the operation of a number of vehicles travelling on a multiple-zone guideway. In addition, a technique to simplify the manufacture of the stator winding for the LSM was developed while constructing a laboratory scale demonstration system.

LSM performance advantages

Linear synchronous motor technology provides some interesting features that can enhance the performance of a transportation system. LSM technology is particularly attractive for use in high capacity systems where a short headway is necessary or in systems that must be capable of climbing steep grades. The following paragraphs will outline some of the key performance features of a transportation system propelled by a LSM and contrast its characteristics with a rotary electric motor based traction system.

acceleration / deceleration

Perhaps the most important benefit of the LSM is that it can provide large forces to the vehicle to enable rapid acceleration or deceleration. The traction force available from a conventional electric rotary motor driving steel wheels on rails is limited by the coefficient of friction between the wheel and the rail. (At very high speeds the dynamic interaction between the steel wheel and rail further reduces the available traction.) The LSM produces a thrust on the vehicle through the interaction between the field of the vehicle magnet array and the current in the stator winding embedded in the guideway. The force available through this interaction is limited by practical or economic rather than physical restrictions. While rail guns or launchers can generate accelerations of many g's using LSM technology, there are significant costs (*e.g.* passenger comfort, power electronics rating, guideway structure, vehicle structure) that limit the acceleration level used in a passenger transportation system to a more reasonable 0.25 to 0.35 g's. By contrast, the acceleration level achievable in a steel wheel / rail system is on the order of 0.05 to 0.1 g's. The increased thrust capability of the LSM enhances the performance of a transportation system in many ways.

Rapid deceleration improves system safety by decreasing the time and the distance required to stop a moving vehicle, especially in the event of an emergency. Increased safety margins allow the system headway to be reduced, thus increasing the capacity of the system. When coupled with the short

reaction time of an automated control system, the rapid deceleration capability afforded with LSM propulsion can result in vehicle headway of a fraction of a second in low speed (50 km/h / 31 mph) applications or a fraction of a minute in high speed (500 km/h / 311 mph) applications.

Off-line loading and unloading of vehicles is an important feature in a high capacity system since the time delay incurred at a station stop significantly degrades the line throughput in an on-line terminal system. A propulsion motor with high thrust capability helps to reduce the length of the on and off ramps required to accelerate the vehicle when leaving and decelerate the vehicle when entering the station. Less real estate and less guideway translate into a reduction in station cost. The availability of rapid acceleration minimizes the impact of merging on the line speed.

The grade climbing ability of the vehicle is directly related to available motor thrust. A LSM propulsion system significantly enhances the performance of a vehicle over uneven terrain. Simplified site preparation, elimination of elevated guideway trusses, and the ability to pass over or under existing infrastructure may result from increased grade climbing capability.

The LSM topology is interesting in that the vehicle contains only a portion of the propulsion system. The bulk of the motor is not onboard the vehicle; the stator winding is embedded in the guideway and the power electronics are located at the wayside. The thrust capability of the LSM can be tailored to meet local requirements. At a steep grade or near a station the stator winding and power electronics can be upgraded to higher power levels; and on a straight, level segment the guideway can be fitted with less expensive hardware. The vehicle is not required to carry the burden of a motor and power electronics capable of meeting the peak thrust demand of the entire route. This feature helps to reduce the weight of the vehicle and the cost of the infrastructure, and also allows the system to be upgraded as local capacity requirements increase.

As mentioned previously, aerodynamic drag increases dramatically as vehicle speed increases. A propulsion system with a high thrust output is required to operate at high speeds. The limited thrust available through steel wheel / rail traction restricts the maximum operating speed. This limitation is compounded by a system right of way with many turns. The vehicle must slow down in a tight radius turn to ensure safety and passenger comfort. Limited acceleration out of and deceleration into turns reduces the ratio of the average trip speed to the maximum vehicle speed, preventing the system from operating near its full speed potential.

power transfer

The vehicle in a system using an active guideway LSM is 'passive'; it requires no power to produce the propulsion force. This feature eliminates the need for a means of transferring large amounts of power from the wayside to the vehicle. (In a high speed transportation system the propulsion motor rating is on the order of several MWatts.) Safety concerns associated with a third rail (a shock hazard) or an overhead catenary (electromagnetic field exposure) are mitigated with LSM propulsion. The cost of maintenance due to wear of the catenary or rail is eliminated. This savings is particularly important at high speeds; the reliability of power transfer in the megawatt range to a vehicle moving at 500 km/h (311 mph) is known to be a problem. The catenary of the French TGV was destroyed in a single run at a speed of 515 km/h (320 mph).

The passive vehicle with LSM propulsion does not require bulky power handling hardware onboard the vehicle. Eliminating the power transformer and power electronics circuitry results in a lighter vehicle, which helps to reduce guideway load and wear.

system configuration

As the name implies, the armature winding of an active guideway LSM is located on the guideway. Thus the armature is the stationary, or stator winding, and it consists of a number of phases wound in a periodic pattern attached to the guideway structure. The stator is wound using stranded cables of copper or aluminum; and the phases are spatially displaced from one another, distributed evenly along a period of the winding. When each phase is driven with a periodic current waveform displaced in time, a travelling wave results. The velocity of the travelling wave is the product of the spatial period of the stator winding and the temporal frequency of the stator drive. The velocity and position of the travelling wave are determined by the frequency and phase of the stator drive. The placement and orientation of the stator winding may vary to suit system constraints such as performance, cost, or interaction with the suspension system. Common stator configurations include a single winding placed in a horizontal plane directly below the vehicle, a port / starboard pair of windings placed in a vertical plane attached to each side of the guideway, or a port / starboard pair of windings embedded in a horizontal plane within a pair of guideway rails.

The excitation field of the LSM is located onboard the vehicle, and consists of an array of magnets. For the motor considered in this thesis, the field is constant (*i.e.* the LSM considered is a singly fed machine). In a large, high speed vehicle the field array may be implemented with superconducting

coils operating in the persistent mode or with conventional copper and iron electromagnets. Superconducting coils are generally required in a system with a large gap between the vehicle and the guideway (*e.g.* MLU002) because a large number of field Ampere-turns are necessary to produce a sufficient magnetic field strength at the stator winding. A smaller gap system (*e.g.* Transrapid TR07) uses a magnetic circuit made up of the steel magnet poles on the vehicle and the laminated steel guideway rails to concentrate the field, and copper coils are used. Rare earth permanent magnet materials can also produce impressive fields, and may offer a less expensive alternative for smaller, lower power systems [2]. It is important to note that the 'active guideway' LSM might equally well be described as a 'passive vehicle' LSM. The vehicle does not require power (or power processing equipment) for the purpose of developing a propulsive force. The placement and orientation of the field array must be commensurate with the stator winding and often also with a suspension mechanism.

A propulsive force on the field array (and of course a reaction force on the stator winding) results from the interaction of the magnetic flux from the field array and the currents in the stator winding. A constant motor force is maintained by controlling the relative position of the travelling current pattern in the stator with respect to the vehicle field array. A stationary, or synchronous, physical relationship between the stator current pattern and the field array results in a constant motor thrust with a magnitude proportional to the amplitude of the stator current. The process of maintaining the position of the stator current relative to the vehicle field is known as 'field oriented control', and allows the LSM to be operated with maximum efficiency.

The placement of the stator current pattern is determined by the phase of the currents driving the stator winding. The stator currents are produced by a wayside power inverter. The controller used to regulate the inverter must know the position of the vehicle field array in order to command a stator current that places the stator travelling wave pattern in the proper position. To effectively control the synchronism of the motor, the accuracy of the vehicle position measurement used to derive the stator current must be on the order of 1 electrical degree of the motor period, which corresponds to a displacement of about 1 cm in a high speed system. This required level of accuracy is quite precise when compared to other transportation modes (conventional rail, trucks, airlines), and the development of specialized hardware to meet this requirement is an important research concern.

Since the stator current pattern must be synchronized to the vehicle position, multiple vehicles may not be placed on the stator winding. (It is

impractical to simultaneously synchronize the stator current pattern to vehicles travelling at different speeds.) The stator winding is divided into zones in order to accommodate multiple vehicles within the system. The stator winding in each zone is electrically isolated from and independent of other zones. A complete set of wayside electronics (controller, power electronics, and demodulation hardware) is associated with each guideway zone. The vehicle spacing is controlled such that there is never more than one vehicle within a given zone's boundaries at any time.

The controller in each guideway zone communicates with its neighboring zone controllers to facilitate a coordinated hand-off of control as a vehicle crosses a zone boundary. The stator currents in both zones are matched as the vehicle crosses the zone boundary, thus no discontinuity in vehicle thrust occurs as the vehicle passes from one zone to the next. In addition, each zone controller communicates with a central dispatch controller to report the measured position and velocity of a vehicle and to receive position or velocity commands. The central dispatch controller is responsible for tracking the position and velocity of all vehicles and issuing the appropriate vehicle profiles to control vehicle spacing and routing.

Splitting the control architecture into this hierarchical configuration simplifies the design of the necessary control and communication hardware. The zone controller must deal with synchronizing the LSM drive, which requires accurately measuring the vehicle position and performing a number of computations several thousand times each second. This task is most effectively carried out by a dedicated, specialized controller located in each guideway zone to eliminate the need for large data transfers between each zone and a centralized controller located remotely. The central dispatch controller is not concerned with the detailed computations requires to synchronize the stator drive in each guideway zone; it views each vehicle in a more abstract sense. The position and velocity measurements of each vehicle are required by the central controller at a data rate of perhaps once a second, thus a wide bandwidth communication link to each zone is not necessary.

Many important system design issues such as the distribution of propulsion power, structural considerations, operation in failure modes, or provision for system maintenance are not discussed in this thesis. Information regarding these issues may be found in [3,4,5].

thesis objectives

The primary objective of this research project was to develop, test, and demonstrate techniques suitable for controlling a LSM-based traction drive. The topics targeted for development in this project included a position

sensing technique capable of accurately measuring the position of the vehicle relative to the stator at all vehicle speeds, a protocol to facilitate a coordinated hand-off of control as a vehicle crosses a zone boundary, a control architecture capable of performing the high speed, detailed computations required to maintain synchronous motor drive as well as providing the higher level scheduling and dispatch functions, and automated control routines to regulate the position or velocity profile of the vehicle. These basic features are required to operate a simple transportation system with LSM propulsion. In addition, a hardware and software platform flexible enough to allow the implementation and testing of future control concepts was desired.

A laboratory scale demonstration system has been constructed to develop and test some of the techniques presented in this thesis. A practical method for the manufacture of a multiple phase stator winding for the LSM was developed during the construction of the demonstration system. The demonstration system may also be used in future projects to explore issues such as fault tolerant operation or self-monitoring of the system to detect wear or degradation. The demonstration system is approximately 1/50 scale relative to a full size high speed MagLev transportation system. The model vehicles travel around an oval shaped guideway layout 33.6 meters in length. The oval consists of two 180° banked curves with a 4 meter radius and two straight sections 4 meters in length. Thus the guideway occupies an area 8 meters wide and 12 meters long.

The stator windings are oriented in a vertical plane and attach to both sides of the guideway, forming a port/starboard motor configuration. (As in the proposed full size system, the port and starboard motors are connected in series rather than being driven independently.) The stators utilize a helical winding technique and are wound with six phases. The guideway is divided into four zones, and each zone contains a full set of electronics including an analog signal processing board, a microcontroller board, a six-phase power inverter, and a chopper. A dc power distribution cable and a serial communication cable link each zone. The vehicles use wheels for suspension and guidance and an array of neodymium iron boron (NdFeB) permanent magnets to develop the excitation field for the LSM. The vehicles are made up of short bogies linked by articulated joints to allow the vehicle to negotiate sharp turns. A vehicle may be either one, two, or four bogies in length. The design and construction of the guideway and vehicle are detailed in Chapter 2. The design of the wayside electronics hardware is detailed in Chapter 4.

contributions

There are many technical challenges in designing a multiple zone, multiple vehicle LSM that have not been fully addressed in prior research. Several contributions relevant to this field of research have resulted from this thesis project.

A novel position measurement system has been developed and implemented. This position system can detect the position of the vehicle relative to the stator winding with an accuracy of approximately 0.1° of an electrical period. This level of accuracy corresponds to a distance of approximately 25 μm (0.001 in) in the model system, or a distance of 1 mm (0.04 in) for a full scale high speed MagLev type system. The measurement system samples the position of the vehicle 3200 times per second, and provides accurate data over the full range of vehicle speeds. By tracking the vehicle position over time, the system also provides a measurement of the vehicle velocity. The hardware used to implement the position sensing system is relatively inexpensive. It consists of a transducer mounted on the vehicle to inject signals into the stator winding, an analog signal processing board at the wayside to extract and demodulate the transducer signals from the guideway, and a digitally implemented nonlinear observer to track the vehicle position and velocity based on the demodulated signals.

The demonstration system is the first implementation of a linear synchronous motor using a helically wound stator. This winding technique has been used in slotless rotary machines and offers an improvement in electrical performance. However, the primary motivation for the use of this winding technique in a linear motor is ease of manufacture. The helical LSM stator can easily be wound on a simple machine since complicated weaving patterns to deal with end turns are not required. The difficulty in manufacturing a conventional stator winding increases as the number of motor phases increases, but the helical technique extends easily to any number of motor phases. The stator winding used in the demonstration system is based on a stock, commercially available rectangular cross section litz wire. (The dimension of the winding period, or 'lay', and the aspect ratio of the width and thickness of the litz wire used for the stator winding differ from stock offerings. The stator winding was produced by making small modifications to the machine used to wind the standard size rectangular litz wire.) A single helical winding replaces two narrow conventional stator windings in the motor configuration used in the demonstration system, which further reduces the manufacturing cost. Most importantly, the performance and manufacturing advantages of the helical stator winding used in the demonstration system are also applicable to the full scale system.

The two-tiered control architecture used in the demonstration system offers some significant advantages. The distributed zone controllers perform the detailed, high speed computations necessary to synchronize the motor control, and the central dispatch controller coordinates the global movement of a number vehicles. This hierarchical approach minimizes the need for wide bandwidth data transfer and allows the system to be expanded easily. The demonstration system is the first to provide a coordinated transfer of control as a vehicle crosses a zone boundary with no discontinuity in the propulsive force applied to the vehicle by the LSM.

2. Demonstration System Design

overview

This chapter outlines the basic design of the laboratory-scale system constructed to develop, test, and demonstrate the control techniques detailed in this thesis. The first section provides an overview of the zone controller, and it is followed by sections describing the design of the guideway, the vehicle, and the LSM.

zone controller

The zone controller is a part of a hierarchical control architecture, which is depicted in block diagram form in Figure 2-1. The guideway is divided into zones to allow a number of vehicles to to travel on the guideway. Since a synchronous motor is used for propulsion only a single vehicle may reside on a stator winding. The stator winding in each zone is isolated from its neighbors, and thus may be driven independently. The system must be operated with a vehicle headway larger than the length of a zone, thus the zone length is a tradeoff between system capacity, cost, and motor efficiency.

The hierarchical control architecture uses a dedicated controller placed in each guideway zone to perform the detailed computations required to drive the stator of the LSM in an efficient, synchronous manner. Placing this control hardware locally eliminates the need to transfer large amount of data to a centralized control facility. Each zone controller is responsible for sensing the position and velocity of a vehicle within the zone boundaries, and for computing the stator drive necessary to force the vehicle to follow a defined velocity or position profile. To maintain a constant thrust on the vehicle the stator must be driven in a manner such that the travelling current pattern in the stator winding remains synchronized with the vehicle field array. The zone controller measures the vehicle position and adjusts the stator drive several thousand times per second to accomplish this task.

Each zone controller must communicate with neighboring zone controllers to facilitate a coordinated hand-off of control as a vehicle crosses a zone boundary. As a vehicle approaches a zone boundary a message containing the position, velocity, and stator current magnitude associated with the vehicle is sent to the neighboring zone controller. The neighboring controller prepares to receive the approaching vehicle by driving its stator with currents identical to those in the zone containing the vehicle, thus the vehicle experiences no thrust discontinuity as it crosses the zone boundary.

If the zone is subdivided into blocks, the zone controller must control the operation of stator winding switches as the vehicle crosses block boundaries. The use of block switching improves the efficiency of the LSM and may help to reduce the VA rating required of the power electronics [4].

The central controller coordinates the global movement of a number of vehicles, and specifies the velocity or position profile for each vehicle. The central controller is responsible for determining the proper headway, scheduling, and routing of all vehicles. The central controller deals with vehicles in a more abstract sense than the zone controller, and can perform its task with vehicle data that is both less precise and less frequently sampled than the measurements required by the zone controller.

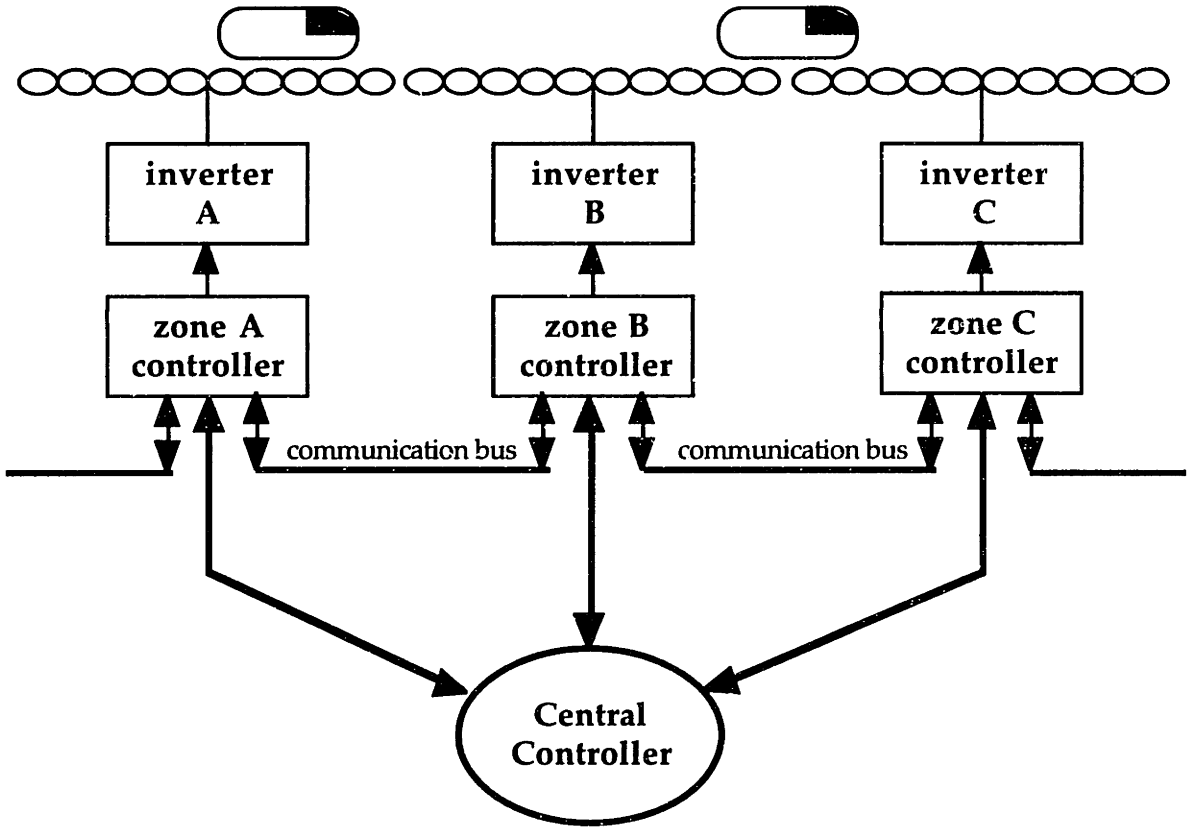


Figure 2-1. Hierarchical control architecture

zone switching

A guideway zone in the demonstration system consists of a section of track approximately 8 meters in length (for a full scale MagLev system the zone length would be approximately 4 kilometers), a set of power inverters to drive the track windings, and a controller. Control of the vehicle must be

passed from one zone controller to the next when the vehicle crosses a zone boundary. This hand-off of control is accomplished in the following manner (refer to Figure 2-2).

When the vehicle nears a zone boundary and reaches position X, controller A notifies controller B of the approaching vehicle. Controller B synchronizes its inverter drive signals to controller A (*i.e.* controller A acts as a master, controller B as a slave). Thus the track windings in zone B carry current waveforms identical in both magnitude and phase to the corresponding windings in zone A.

At position Y the vehicle crosses the zone boundary and the vehicle transducer injects the position dependent signals into the stator windings of zone B. Controller B assumes the role of master, with controller A as its slave.

When the vehicle is entirely within zone B (at position Z), controller B relieves controller A of its role as a slave. Controller A shuts down the drive to its track windings and waits for the next vehicle.

The communication bus linking adjacent zone controllers is implemented with a 19.2 kBaud serial interface. A 10-byte message containing vehicle position and velocity, stator current magnitude, and a time stamp is used to communicate the information necessary to synchronize the slave controller to the master.

Measured error responses of the zone controller's position sensing system resulting from crossing a zone boundary are shown in Figures 2-3 through 2-5. (Refer to Chapter 3 for a description of the dynamics of the nonlinear observer used in the position sensing system.) Figure 2-3 shows the observer error response with the inter-zone communication bus intact. The zone controller receiving the vehicle is informed of the vehicle position and velocity just before the vehicle enters the zone. This information allows the zone controller to accurately predict the position profile of the vehicle, and the peak observer error is less than 6°. The error is mainly due to misalignment of the guideway joint at the zone boundary. (Note that the 'spike' preceding the observer error response is a measurement artifact.)

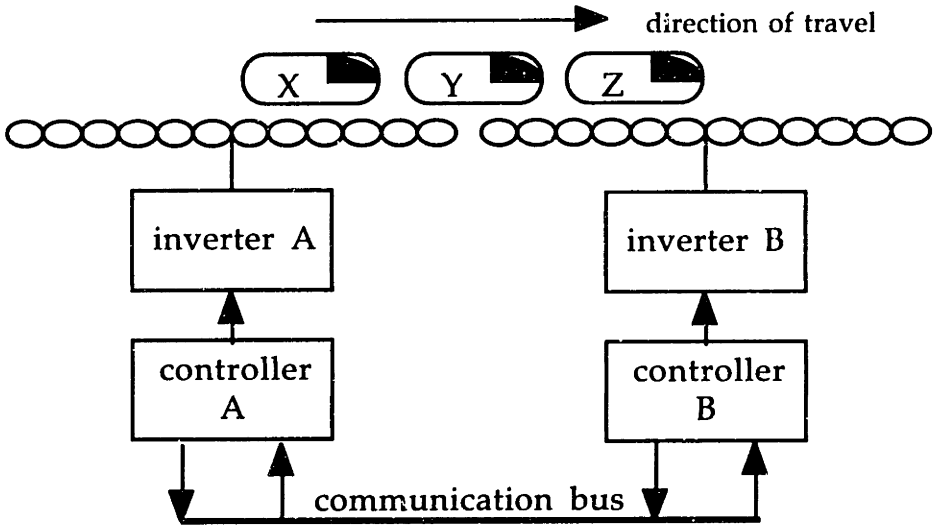


Figure 2-2. Zone switching pictorial

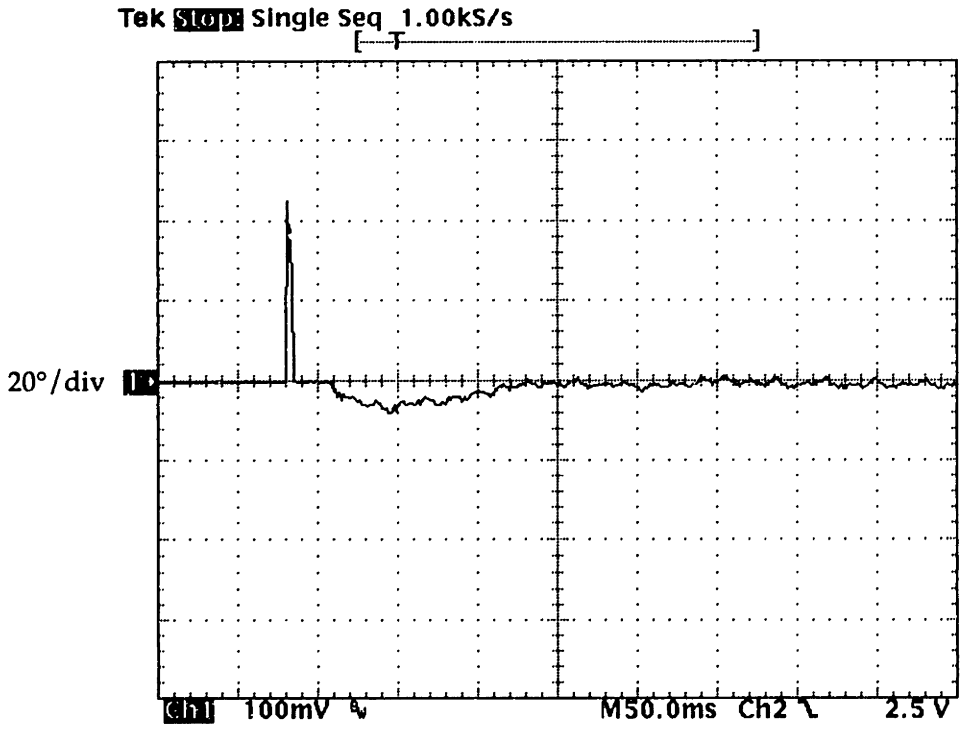


Figure 2-3. Observer error response at zone boundary (inter-zone communication intact)

Figures 2-4 and 2-5 show the observer error response with a disabled communication bus. In Figure 2-4 the zone controller receiving the vehicle is informed of the approximate velocity of the approaching vehicle but has no position information. The position sensing system locks onto the actual vehicle position quickly. The peak observer error is random (it could be

anywhere in the range of $-\pi$ to π). The scope plot shown is for a relatively large error of nearly 90°. The plot shows the $\mathrm{Sin}(\theta_{error})$, thus the 20°/div scale factor only applies for small errors. Figure 2-5 shows the observer error response with a disabled communication bus and no position or velocity information. The position sensor must lock to both the velocity and the position of the vehicle. The large initial velocity error (a full-speed error is induced for this plot) causes the position sensor to slip several cycles before locking to the vehicle position. The responses shown in Figures 2-3 through 2-5 were taken with the minimum recommended position sensor signal amplitude, which degrades the dynamics of the position sensing system. The response shown in Figure 2-5 represents the worst case transient for the position sensing system, and the sensing system requires slightly over 1 second to lock to the vehicle position.

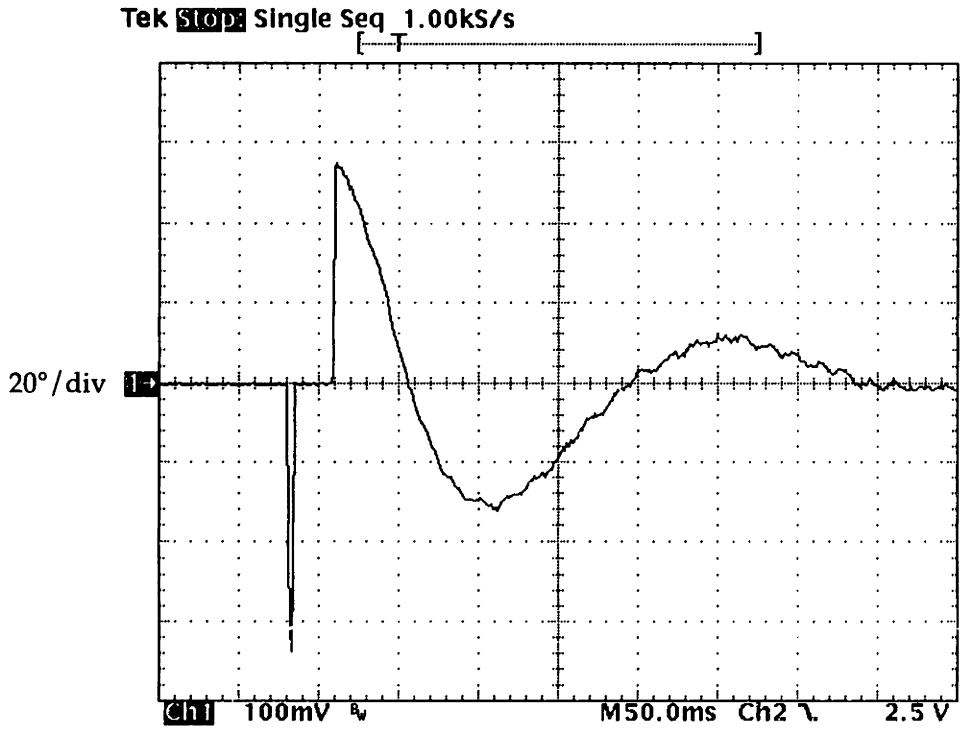


Figure 2-4. Observer error response at zone boundary (inter-zone communication disabled - velocity known)

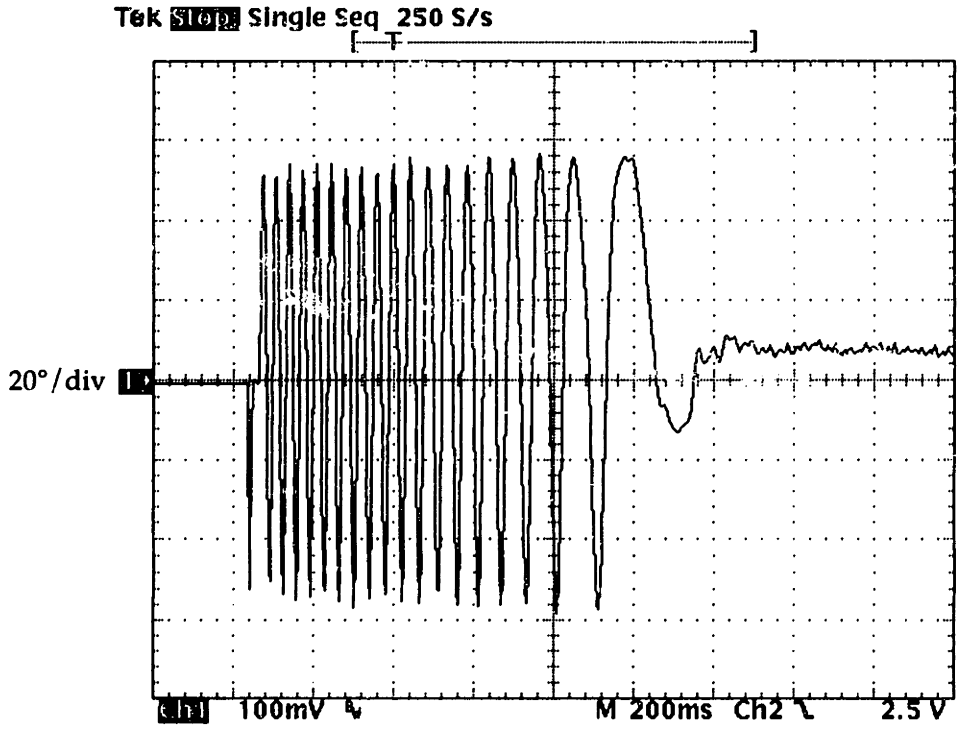


Figure 2-5. Observer error response at zone boundary
(inter-zone communication disabled - velocity unknown)

block switching

Block switching is a technique used to minimize the power dissipated as I^2R losses in the stator windings. The effective stator winding resistance and inductance values are reduced by energizing only a portion of the stator windings within a zone. The reduction in stator winding impedance results in increased LSM efficiency. In addition, vehicles illegally entering a block enter a region with shorted stator windings, and are therefore subjected to maximum dynamic braking. This feature improves the safety of the system by automatically decelerating a vehicle that illegally attempts to follow the lead vehicle at a smaller than allowable distance. The guideway design of the demonstration system is quite flexible, and it allows the stator winding in each zone to be divided into as many as four blocks. However, the switch hardware and controller software required to implement block switching have not been incorporated into the demonstration system.

Figure 2-6 shows a pictorial of a block switching scenario. Switch A and B each represent a bank of individual switches to short each phase of the stator winding. The block switching operation is as follows:

When the vehicle is at position X, switch A is open and switch B is closed. The stator winding in the left portion of the zone is energized, and the stator winding in the right portion of the zone is shorted (and thus dissipates no power).

When the vehicle reaches position Y, switch A is closed, and switch B remains closed. The stator winding in the center portion of the zone is energized, while the stator windings in the outer portions of the zone are shorted.

When the vehicle reaches position Z, switch A remains closed and switch B is opened. The stator winding in the right portion of the zone is energized, and the stator winding in the left portion of the zone is shorted.

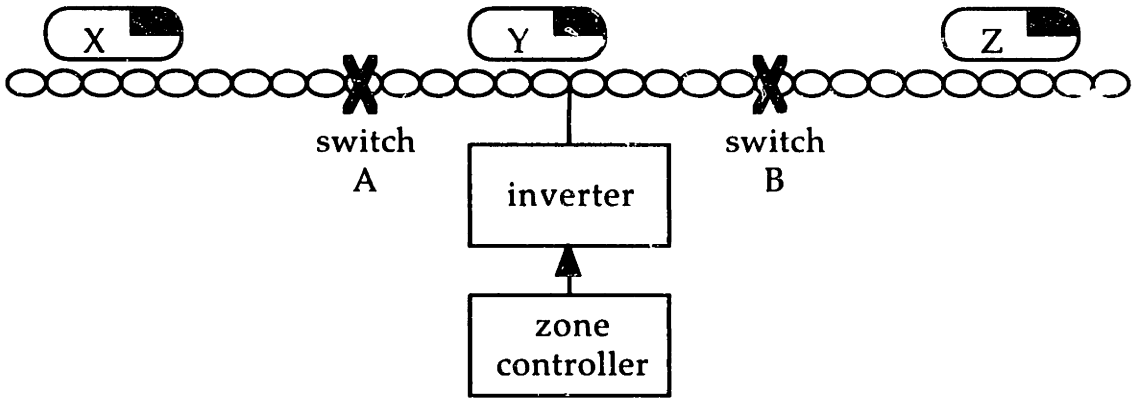


Figure 2-6. Block switching pictorial

propulsion control

The propulsion control loop for the system is shown in block diagram form in Figure 2-7. The compensator, the observer, and the power electronics control algorithm are all implemented digitally on a Motorola 68HC16 micro-controller. A digital implementation provides a flexible platform that can be extended to allow sophisticated features such as adaptive control, parameter estimation, or condition monitoring.

The desired vehicle profile is a function of position to allow speed changes for hills, turns, stations, weather, or traffic conditions (to model the full-scale system). The appropriate vehicle profile is computed by the central controller, and down-loaded to the zone controller.

The compensator can be programmed to force the vehicle to follow either a position or a velocity profile. The compensator is implemented using straightforward digital control techniques.

The power electronics controller computes the appropriate state for each of the switches in the power electronics module by comparing the set-point current to the measured chopper current and by using an inverter commutation look-up table indexed by the estimated vehicle position. The power electronics control algorithm is described in Chapter 4.

The propulsion controller utilizes a nonlinear state observer to compute the vehicle position and velocity based on measurements of the signals induced in the stator windings by the vehicle transducer. The observer implementation is described in Chapter 3. Discrete-time filter techniques could be used instead of an observer to derive velocity information. There are, however, some disadvantages associated with the filter technique. Substantial low-pass filtering is necessary to reject position sensor noise, and the dynamics of the low-pass filter degrade the performance of the velocity control loop. In addition, the filter design is complicated by the nonlinear, periodic nature of the sensed position. (The electrical position has a discontinuity, or wrap-around, at a position of 2π.)

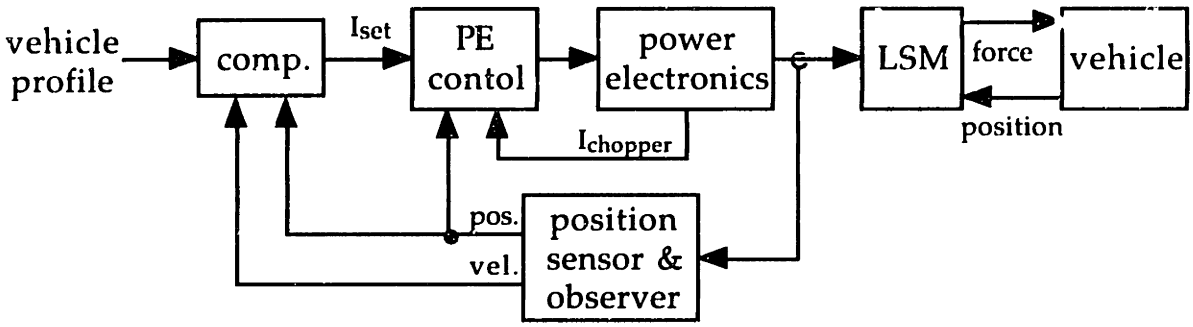


Figure 2-7. Block diagram of propulsion control loop

The inner control loop formed by the position sensor, the motor commutation algorithm (a part of the power electronics controller), the inverter (a part of the power electronics circuitry), the motor, and the vehicle performs a task similar to the mechanical commutator on a dc motor. This commutation loop maintains synchronism of the stator current pattern in the guideway to the field array on the vehicle with the optimum torque angle. The nonlinear relationship between the current driving the stator phases and the thrust produced by the motor is transformed into a linear relationship between the current driving the inverter and the motor thrust. Indeed, the dynamics associated with the transfer function linking inverter current and motor thrust are beyond the desired bandwidth of the propulsion control loop, and the motor is modelled simply as a proportional constant. The thrust constant K_t for the LSM used in the model system (for a four bogey vehicle) is 0.47 Newtons/Ampere. The mass of a four bogey vehicle is 1.25 kg, so the LSM can be modelled as producing an acceleration of 0.38 m/sec^2 (or 0.039 g's) for each Ampere of stator current.

The motor commutation loop is enclosed by a current control loop, which is implemented with the chopper (a part of the power electronics

circuitry) and the chopper control algorithm (a part of the power electronics controller). The chopper provides only a two quadrant range of operation; the chopper current can be in either direction but its output voltage must be positive. Four quadrant motor operation is obtained by allowing the chopper control algorithm to flip the state of all of the inverter switches, thus negating the voltage applied to the stator. A 'negative' stator current occurs when the chopper current is negative and the inverter commutation pattern is normal (this operation corresponds to a dynamic braking mode, the LSM is being used as a generator); or when the chopper current is positive and the inverter commutation pattern is flipped (the LSM is operating as a motor travelling in a reverse direction). The dynamics associated with the current control loop are much faster than the desired propulsion control bandwidth.

The current control loop is enclosed by a velocity control loop. A proportional compensator is used to compute the appropriate stator current from the difference between the desired and the measured vehicle velocities. A filter pole placed at a frequency of 100 rad/sec (*i.e.* τ_p = .01 sec) smooths the compensator output. The stator current set-point value I_{set} is limited to the range of ±10 Amperes. A block diagram of the velocity control loop is shown in Figure 2-8. The power electronics, the LSM, the vehicle mass, and the integration relating acceleration to velocity have been combined into a single block labelled 'motor'. The value of the velocity compensator gain constant K_v is 35 A sec/m, which results in a loop crossover frequency of 13.3 rad/sec with a phase margin of 82° [6].

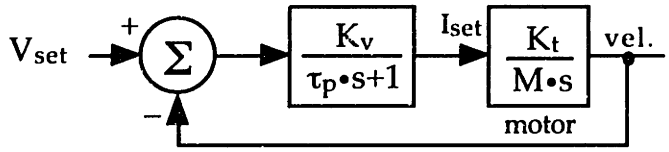


Figure 2-8. Velocity control loop

A position control loop can enclose the velocity loop; this loop configuration allows the controller to track a cruise input position profile with no steady state velocity error and provides a direct method for controlling the spacing between vehicles. A proportional compensator is used to compute the velocity set-point value based on the difference between the desired and the measured vehicle positions. The velocity set-point value V_{set} is limited to the range of ±2.5 m/sec. A block diagram of the position control loop is shown in Figure 2-9. The value of the position compensator gain constant K_p is 6.25 sec^{-1}, which results in a position control loop crossover frequency of 5.7 rad/sec with a phase margin of 63°.

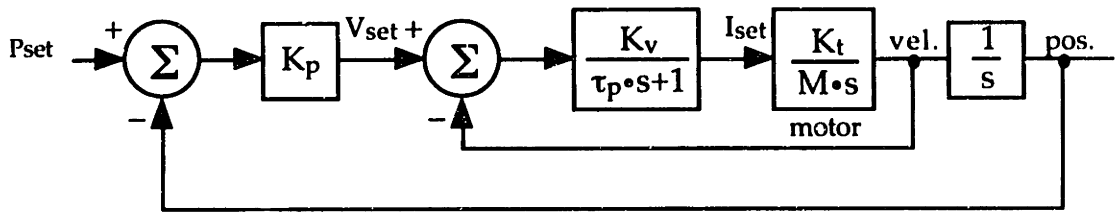


Figure 2-9. Position control loop

The desired position profile P_{set} can be specified in a variety of ways; the method adopted in the model vehicle system uses a series of piecewise cubic segments. A cubic polynomial function of time is used because it is easy for the zone controller to compute the value of the polynomial at any value of time, and the four coefficients specifying each cubic segment can be chosen so that position, velocity, and acceleration are continuous at the segment boundaries. A segment length of 1 second is used; thus the central controller, which computes the desired position profile for every vehicle, must download only four coefficients each second to each zone controller. A low speed serial interface between each zone controller and the central controller is sufficient to transfer the position profile data.

guideway design

The guideway is based on the oval-shaped layout shown in Figure 2-10. The oval consists of a pair of 180°, 4-meter radius curves connected by 4-meter long straightaways. The guideway is divided into four zones, and each zone may be optionally subdivided into as many as four blocks. Each zone contains a complete set of control and power electronics.

The oval shape of the guideway allows constant speed operation of the vehicle. The zone and block divisions of the guideway allow testing of control algorithms for zone and block switching, as well as allowing simultaneous operation of up to three vehicles. (A central controller, implemented on a computer, is required to schedule vehicle profiles for multiple vehicle operation.) Operation with multiple vehicles provides the opportunity for testing algorithms to control inter-vehicle spacing. The mechanical characteristics of the guideway are based on a design outlined in reference [7].

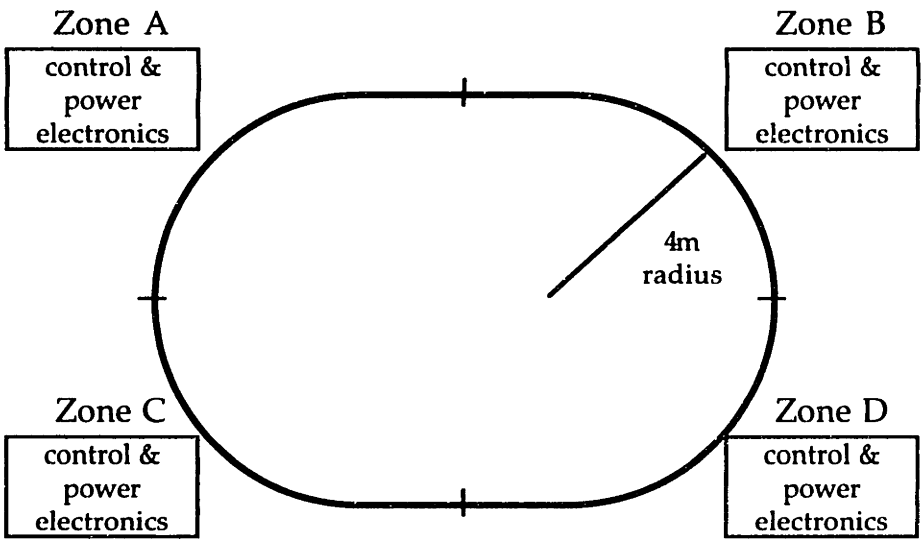
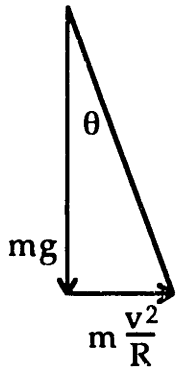


Figure 2-10. General layout of guideway

guideway banking

The full-size MagLev vehicle has the ability to actively control the bank angle of the passenger compartment, and it can provide the passengers with coordinated turn conditions over a wide range of vehicle operating speeds. The model vehicle does not have banking control capabilities, so the guideway bank angle is chosen to provide a coordinated turn when the model vehicle is operating at a nominal cruise speed. The full-size vehicle has a cruise speed of approximately 130 m/sec, which corresponds to a 1:50 scale

model cruise speed of 2.6 m/sec. As shown in Figure 2-11, the appropriate bank angle for a vehicle operating at a speed of 2.6 m/sec in a 4-meter radius turn is approximately 10°.



$$\tan\theta = \frac{v^2/R}{g}$$

$$or \quad \theta = \tan^{-1}\left[\frac{(2.6\ m/sec)^2\ /\ 4m}{9.8\ m/sec^2}\right] \approx 10°$$

Figure 2-11. Bank angle required for coordinated turn

transition curve

The vehicle is not subjected to any lateral acceleration while travelling over a straight section of guideway, and is exposed to a constant lateral acceleration while cruising through a constant-radius curve. A discontinuity in acceleration, or 'jerk' occurs if a straight section of guideway is connected to a constant-radius curved section of guideway. To prevent this discontinuity, a transition curve is placed between the straight section and the constant-radius curved section of guideway. The transition curve varies smoothly in radius of curvature from infinity (to match the straight section), to 4 meters (to match the constant radius curved section). In addition, the bank angle of the transition curve varies smoothly from zero to 10°. A more detailed description of the transition curve can be found in [8].

guideway construction

A cross section of the guideway structure is shown in Figure 2-12. The principal structural member of the guideway is the vertical fiberglass web. The top of this web supports the suspension wheels of the vehicle. The stator windings for the port and starboard LSMs are attached to the sides of the web. The guidance rails protrude from both sides of the web just below the stator windings, these rails provide lateral stability for the vehicle. The guideway is constructed in 2.1 meter long segments to allow disassembly for storage or relocation. The oval layout is made up of a total of 16 segments; 4 straight segments and 12 curved segments. The outline and holes in the fiberglass web of the curved segments were machined on a numerically controlled milling machine to produce the required shape for the compound curves.

The guideway web is supported by a weighted pedestal and clamp mount. The pedestal height and clamp mechanism angle are adjustable to allow for curves and banks in the guideway. A side view of a segment of guideway with support pedestals is shown in Figure 2-13.

Slots for a Hall effect sensor are cut into both ends of each guideway segment. (The sensors are only placed at the center joint of each guideway zone, but the Hall sensor slots are available at all joints.) The Hall effect sensors are placed near the bottom edge of the stator winding. In this location the magnetic field from the poles of a passing vehicle field array are strong but the magnetic field from the stator current is weak. The wires to connect to the sensor fit in a vertical groove cut in the end of the guideway web.

A two-conductor cable for the distribution of dc propulsion power to each of the zone controllers and a four-conductor modular phone cable for communications between neighboring zones are run along the guideway.

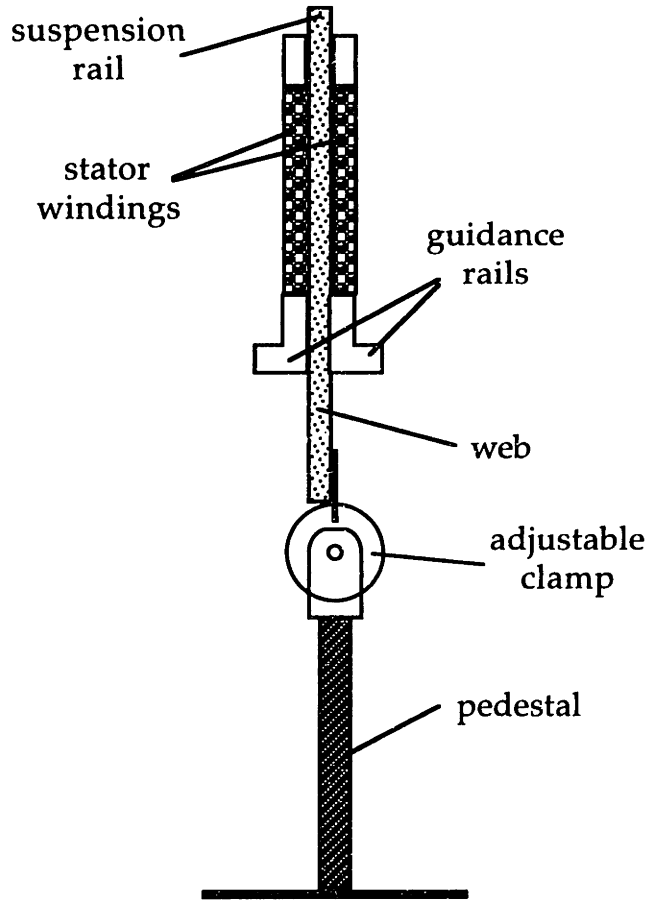


Figure 2-12. Cross-sectional view of guideway structure

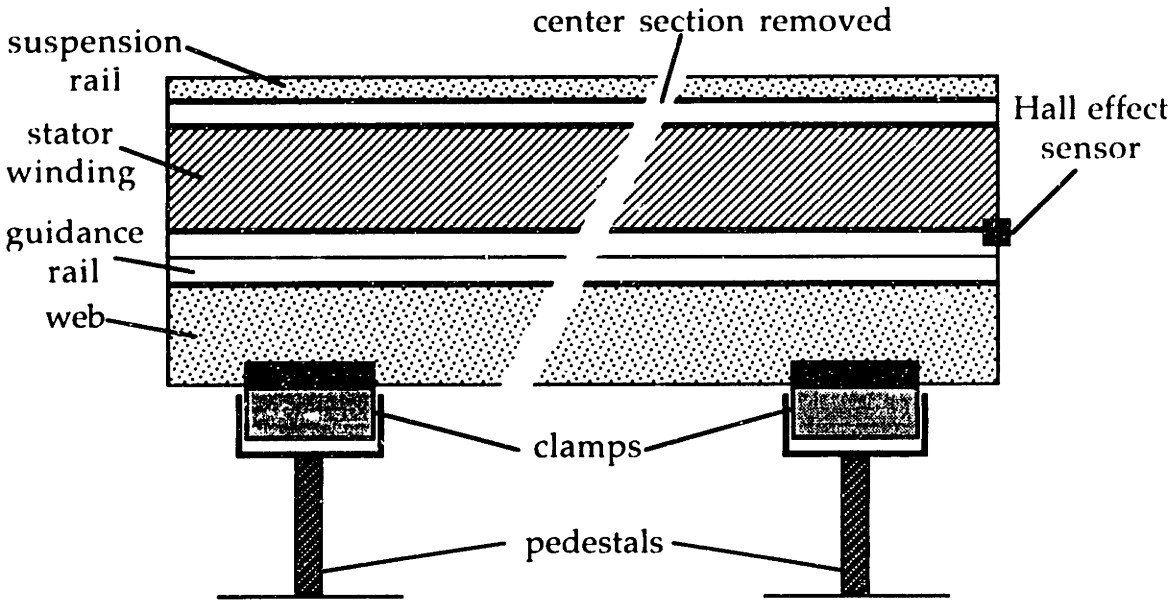


Figure 2-13. Side view of guideway section with pedestal mounts

vehicle design

The demonstration vehicle under-carriage is made up of four bogies (vehicles of one or two bogies in length may also be used). The bogies are attached to one another with articulated joints, thus allowing the vehicle to negotiate tight turns. Each bogey has both suspension and guidance wheels, and port and starboard magnet modules. The bogies wrap around the top of the guideway structure, with the suspension wheels supported by the rail on top of the guideway web and the guidance wheels supported by rails protruding from the sides of the guideway web. A four bogey vehicle has a mass of 1.25 kg.

Each magnet module consists of eight neodymium iron boron (NdFeB) rare earth magnets arranged as a linear array of four dipoles of alternating polarity, glued to a steel plate. The magnets are rectangular in shape with the dimensions 0.25" thick x 0.375" tall x 0.625" long, with the magnetization vector along the 0.25" dimension. The vertical dipole spacing is 0.5" between centers and the horizontal pole pitch separating the dipoles is 1.875 cm (0.738") between centers. The steel plate serves both as a structural member of the bogey and as a path for magnetic flux. The field strength of the magnet array is 0.5 Tesla at the pole face and 0.25 Tesla at the center of the guideway web (measured at the middle of a pole face).

Cut-away end and side views of a vehicle bogey are shown in Figures 2-14 and 2-15. Figure 2-16 shows a side view of the demonstration vehicle, made up of four articulated bogies, the position sensing inductor, and the driver circuitry.

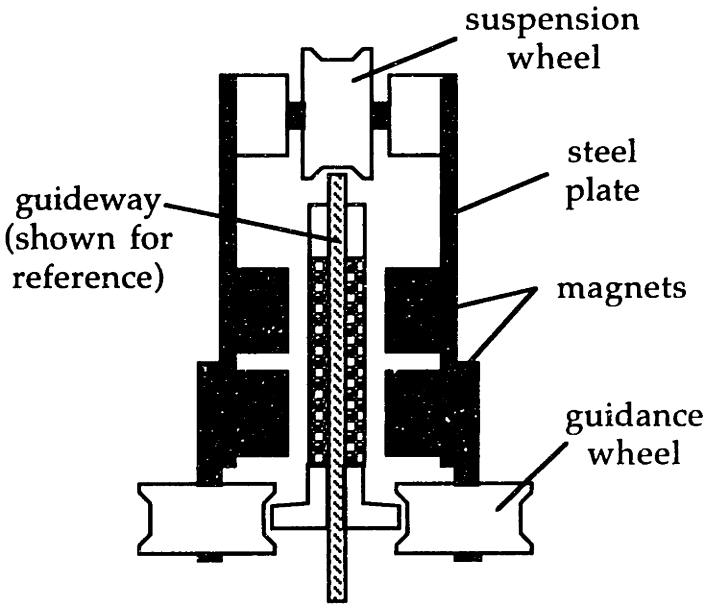


Figure 2-14. Bogey structure (end view)

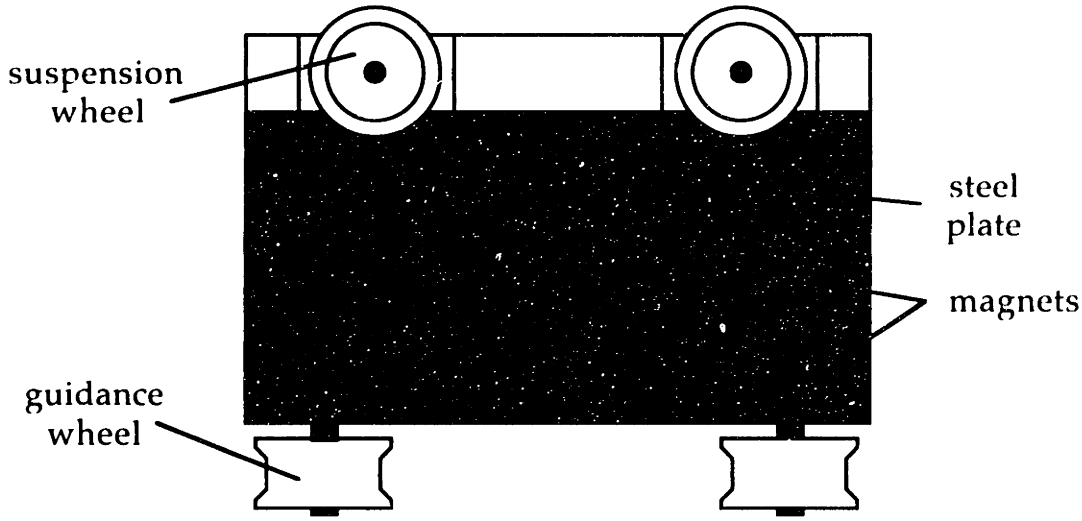


Figure 2-15. Bogey structure (cut-away side view)

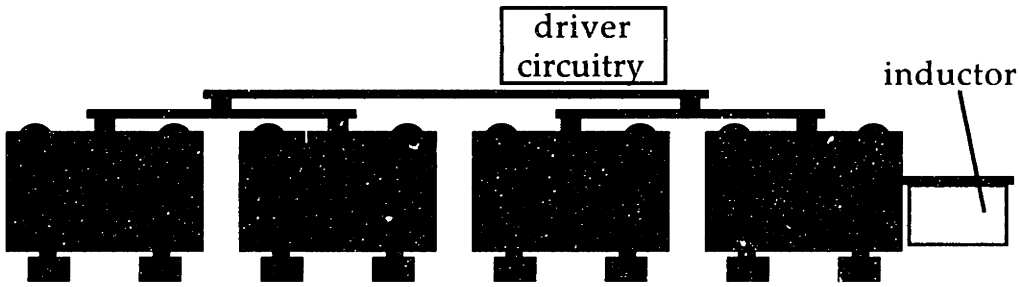


Figure 2-16. Side view of test vehicle

vehicle electronics

The electronic circuitry on board the vehicle consists of a simple, low-power oscillator that drives the inductor used to induce position dependent signals into the stator winding. The inductor is constructed by winding 96 turns of 7/30 litz wire onto each of a pair of ferrite 'C' cores. The cores are Hitachi part number SB7C-FUR50 with the legs cut to a length of 0.625 inches. Each core is mounted in a bracket attached to the vehicle side plates. The gap between the poles of the cores is approximately 0.5 inches. This spacing allows adequate clearance with respect to the stator windings, and results in an inductance of approximately 387 μH. Figure 2-17 shows the core modification and the inductor construction.

The peak energy stored in the magnetic field of the inductor when driven with a 25.6 kHz, 8 Volt sine wave is 3.2 μJoules. A 0.1 μF resonating capacitor is placed in parallel with the inductor to minimize the reactive load presented to the driver circuitry.

The tank circuit and the oscillator circuitry are shown in the schematic diagram in Figure 2-18. The first stage is a 25.6 kHz crystal oscillator and the second stage is an amplifier used to boost the output of the oscillator to drive the tank circuit. The diode and 1 μF capacitor detect the (negative) peak value of the tank drive voltage, which is fed back to set the bias of the oscillator stage. The feedback network controls the magnitude of the output drive to a value approximately 1 Volt less than the supply voltage. Power for the onboard circuitry is supplied by a 9 Volt battery, and the current drawn is nominally 4 mAmps.

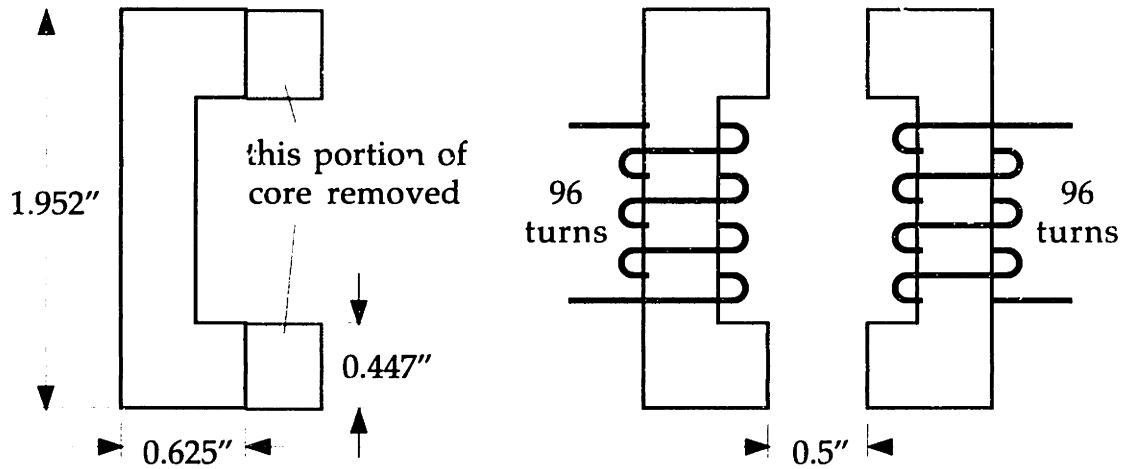


Figure 2-17. Vehicle transducer construction

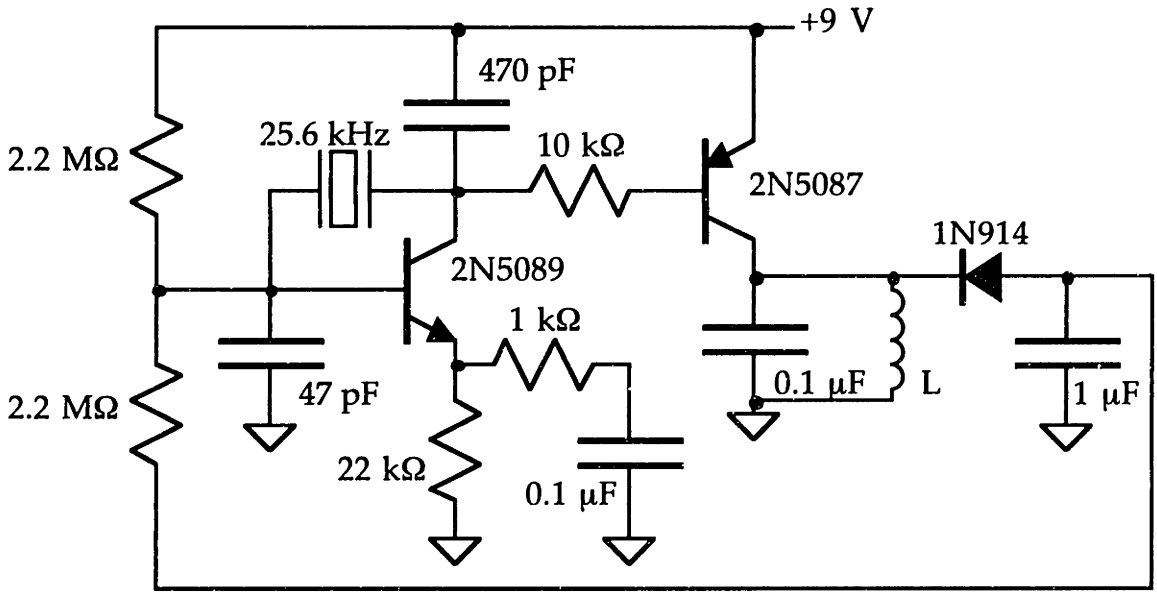


Figure 2-18. Oscillator schematic

LSM design

The linear motor used to provide the propulsive force to the vehicle is a multiple phase, permanent magnet synchronous motor. A set of inverters drive the stationary windings of the motor (located in the guideway) with variable frequency, variable magnitude waveforms. This stator drive produces a magnetic field pattern that moves along the guideway as a travelling wave. The velocity and position of the travelling wave are determined by the frequency and phase of the currents driven through the motor windings. The interaction of the travelling magnetic field pattern produced by the windings (the stator field) and the field produced by permanent magnets located in the vehicle (the excitation field) results in a propulsive and/or a normal force on the vehicle. Depending on the orientation of the LSM, the normal force may be useful for levitation, heave damping [9], or guidance. The normal forces from the port and starboard motor windings cancel one another in the motor configuration used in the demonstration system described in this thesis. By independently controlling the port and starboard LSMs, it would be possible to generate a guidance force with the model system. However, this mode of operation is not being considered.

Research literature concerning the design and analysis of motors suitable for propulsion of a MagLev system is readily available; see [4,10,11,12]. Further development of this research topic was not a primary goal of this thesis. However, an improved method of manufacturing the stator winding of the LSM was developed during the construction of the demonstration system. A short discussion of LSM features is included here for reference.

motor configuration

A typical LSM configuration is shown in Figure 2-19. The field excitation is provided by a linear array of magnet poles of alternating polarity. This magnet array would be implemented by a set of superconducting coils operating in the persistent mode in a large motor such as that found in a full scale high speed MagLev vehicle. For the small scale test system, an array of rectangularly shaped neodymium iron boron (NdFeB) permanent magnets provide the field excitation. For clarity, only one phase of the stator winding is shown in Figure 2-19. The cross-turn portions of the stator winding are perpendicular to both the magnetic field and the direction of the desired propulsive force. Since the reaction force (this is the force acting on the stator) produced is the cross-product of the magnetic field vector and the current vector, this orientation of the cross-turn provides the maximum possible force in the desired direction for a given stator current. The end-turn

portions of the stator winding are oriented along the direction of the desired force, connecting adjacent cross-turns. As a result of this orientation, the end-turns cannot produce a propulsive force in the desired direction. (Depending on the orientation of the LSM, the component of force from the end-turns can be used for levitation or guidance.)

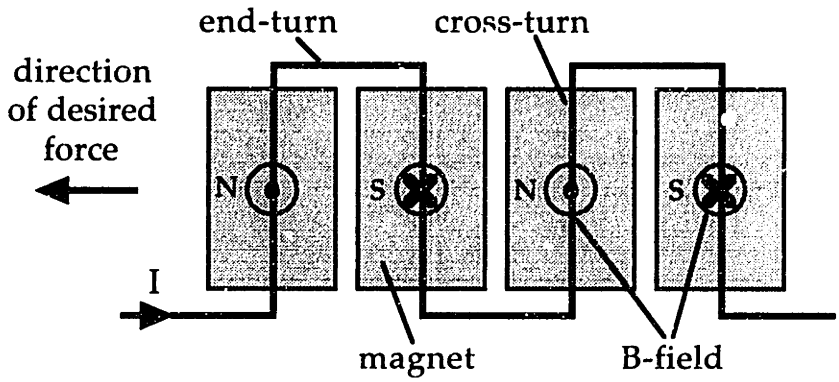


Figure 2-19. LSM with conventional stator winding
(only one phase of the stator winding is shown for clarity)

A helically wound LSM is depicted in Figure 2-20. The magnet array used to provide the field excitation for the motor is exactly the same as the field array used with the conventionally wound stator. This winding configuration has no end-turns, and the cross-turns are placed in a diagonal direction. Thus the force vector produced by the current flowing in a cross-turn of the helical winding contains a component in the desired direction and a normal component. The normal component forces associated with adjacent cross-turns are in opposing directions, and therefore cancel. The propulsion force components associated with adjacent cross-turns are in the same direction.

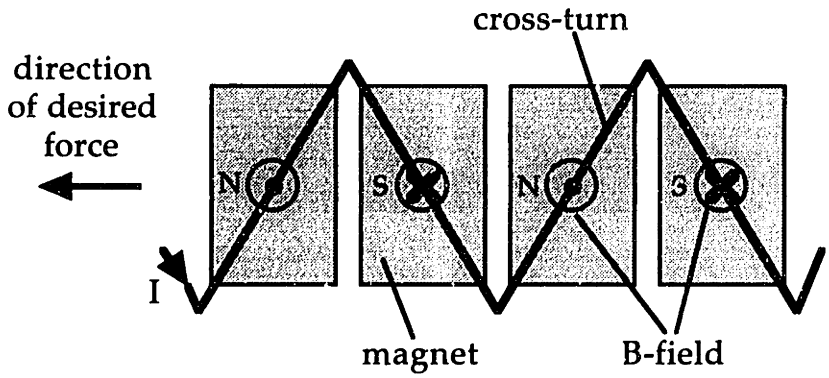


Figure 2-20. LSM with helical stator winding
(only one phase of the stator winding is shown for clarity)

The conventional winding's advantage of straight cross-turns which produce the maximum force in the desired direction per unit stator current is offset by the disadvantage of end-turns which produce no force in the desired direction. Each turn of the helical winding links less of the field flux than a comparable conventional winding but the length of the helical winding path is shorter, resulting in a smaller winding resistance. A detailed comparison of the two winding techniques [13] shows that the helical configuration is favorable for machines without iron slots. However, the electrical performance advantages of the helical winding are not the primary reason for selecting this configuration. The helical winding is simpler and cheaper to manufacture.

The model vehicle system has a stator winding on both sides of the guideway, and a field magnet array on both the port and starboard side of the vehicle. The port and starboard stator windings are connected in series and are driven by a single inverter, so the two motors act as a unit.

far field roll-off

In the full-size MagLev vehicle, magnetic field strengths on the order of 1 Tesla are required in the region of the stator winding to produce the necessary propulsive force. To avoid the health risks associated with exposure to large magnetic fields, the field strength allowable in the passenger compartment is orders of magnitude smaller [14]. The weight associated with ferromagnetic shielding is undesirable, thus it is beneficial to employ a LSM configuration that provides a sharp roll-off in far field intensity. One method of achieving this goal is to 'split' the LSM as shown in Figure 2-21. The field excitation is provided by a linear array of magnetic dipoles of alternating polarity, and the stator winding is split to place currents of the appropriate direction under each field pole. The far field of the alternating dipole array rolls off more steeply than the far field of the simple arrays shown previously in Figures 2-19 and 2-20 [4].

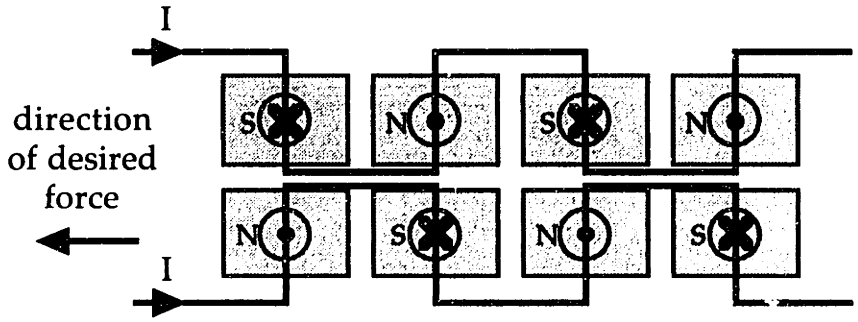


Figure 2-21. Split LSM with conventional stator winding
(only one phase of the stator winding is shown for clarity)

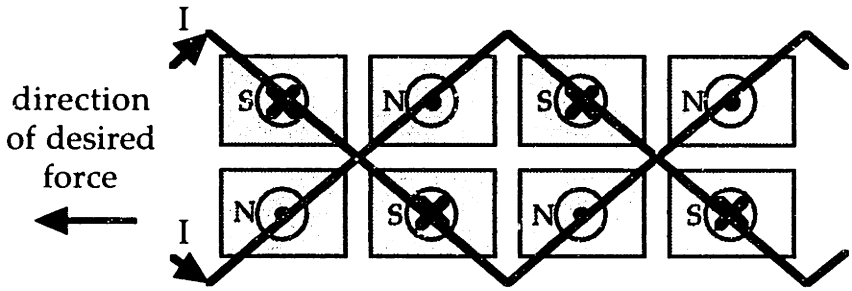


Figure 2-22. Split LSM with helical stator winding
(only one phase of the stator winding is shown for clari)

The conventional stator winding for the split LSM in Figure 2-21 has a very congested mass of end turns in the region between the two rows of magnets. (Figure 2-21 shows only one stator phase, so this mass of end turns is significantly worse than indicated in the figure.) The helical winding shown in Figure 2-22, however, does not have this difficulty. The helically wound stator can be manufactured as a 'single' winding instead of two separate windings. The potential benefit associated with ease of manufacture is the primary reason for the use of the helically wound stator in the scale model vehicle.

Figure 2-23 shows a perspective view of a 24 x 7/24 compacted rectangular litz cable [15]. This cable is manufactured by first twisting 7 strands of #24 AWG magnet wire into a round litz bundle. 24 of these round 7/24 litz bundles are then helically wound around the cylindrical end of a mandrel, which becomes flatter and wider along its length. The helix of litz wires is drawn over the mandrel and through a set of rollers, and is pressed into a rectangular cross section. This compacted rectangular litz cable is used to manufacture the stator winding of the model vehicle system.

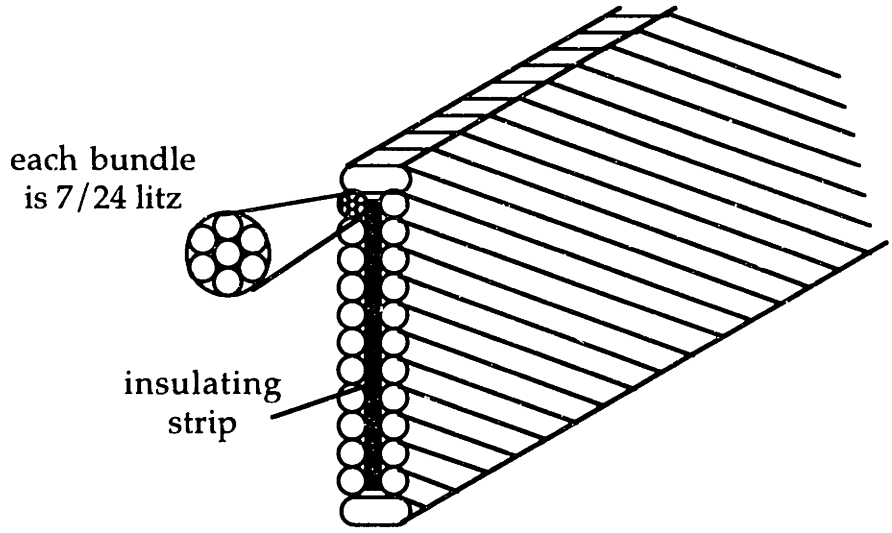


Figure 2-23. Compacted rectangular litz cable

phase connection

The alternating dipole array used for the motor field results in identical flux linkages for pairs of wires within the winding period, or 'lay', of the stator helix. Figure 2-24 shows the arrangement of the field array with respect to one of the stator winding phases; note that the A$\underline{A}$ winding and the a$\underline{a}$ winding link the same field flux. (So will the B$\underline{B}$ and b$\underline{b}$ windings, etc.) The A$\underline{A}$ and a$\underline{a}$ windings are connected to the inverter such that they carry the same propulsion current waveform. The spatial period of the propulsion flux linkage is λ_p.

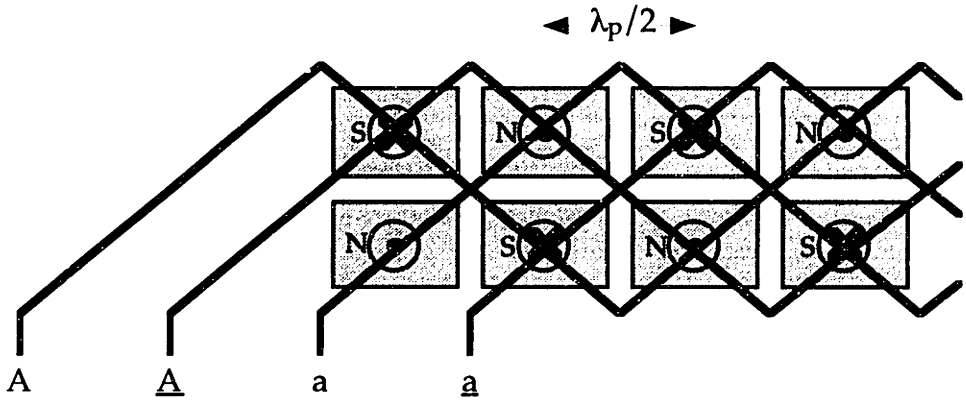


Figure 2-24. Stator winding - propulsion configuration

Figure 2-25 shows the arrangement of the position sensing transducer with respect to the A$\underline{A}$ and a$\underline{a}$ windings. The position transducer is arranged as a pair of poles of alternating polarity, rather than the alternating dipoles used for the field array. The spatial period of the position transducer flux linkage is λ_s, which is twice the spatial period of the propulsion flux linkage. The flux linked to each of the windings by the position transducer is of opposite polarity, so the A$\underline{A}$ and a$\underline{a}$ windings will conduct position sensor signal currents of opposite polarity.

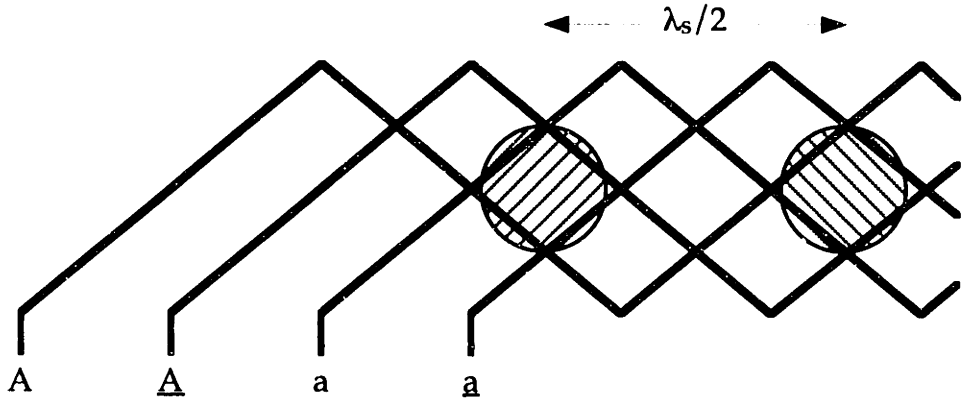


Figure 2-25. Stator winding - position sensor configuration

Using different pole pitches for the propulsion and position sensing signals suggests the winding connection shown in Figure 2-26. The A$\underline{A}$ and a$\underline{a}$ windings are connected in parallel. The propulsion current splits and half flows through the A$\underline{A}$ winding, the other half through the a$\underline{a}$ winding; both windings conduct a propulsion current of the same magnitude and polarity. (*i.e.* the propulsion current appears as a common mode signal.) The position sensor current flows in a circulating path; both windings conduct a position sensor current of equal magnitude and opposite polarity. (*i.e.* the position sensor current appears as a differential mode signal.) The current transformer used to detect the position sensor current flowing in the stator phase is connected with the $\underline{A}$ and $\underline{a}$ leads passing through the core in opposite directions. Thus the magnetomotive forces (mmf's) due to propulsion current components cancel in the transformer while the mmf's due to the position sensor currents reinforce one another. This configuration discriminates the position sensor currents and the propulsion currents based on orientation in the stator winding. (A similar discrimination results for a configuration where the propulsion system is based on single poles with a double pitch length and the position sensor is based on dipoles with a single pitch length. However, the phase uncertainty due to reconstruction of the carrier must be accounted for.) The propulsion currents are rejected by approximately a factor of 200 relative to the position sensor currents using this technique. Chapter 4 discusses the sensing and processing of the position sensor signals in detail.

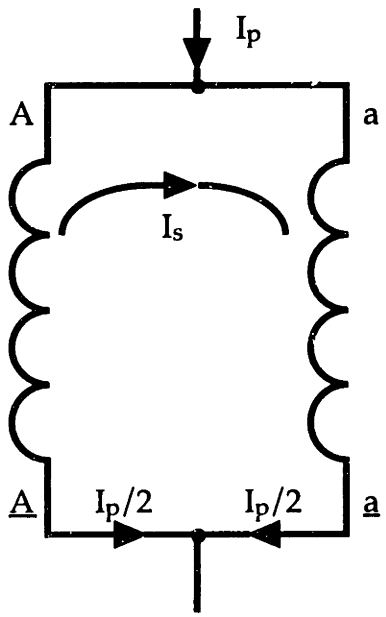


Figure 2-26. Stator winding connection

The motor phases may be connected in either a delta or a wye configuration and driven with a six-switch inverter to produce the propulsion currents. Figure 2-27 shows a three phase winding set connected in a wye configuration. The demonstration system utilizes a six-phase stator winding. The phases are organized as two 3-phase sets, the 'even' or A, B, and C phases, and the 'odd' or X, Y, and Z phases. Each three phase set is connected in a wye configuration and driven with a six-switch inverter, and the six-switch inverters are stacked in series. The configuration of the inverter is covered in Chapter 4.

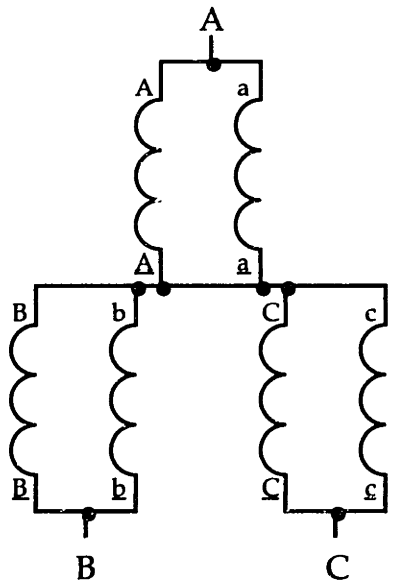


Figure 2-27. LSM phases connected in a wye configuration

stator construction

The guideway used in the model vehicle system is built in 2.1 meter long segments to allow disassembly and relocation. The stator windings, which are bonded to the sides of the guideway, must therefore be carefully arranged to result in smooth joints (both electrically and mechanically). A full size transportation system would also require a segmented guideway to simplify manufacture and repair. The full size system uses a box beam shaped guideway, and the electrical connection of stator segments could be made in the space inside the box beam. The model system does not require the mechanical integrity of the full scale system, so a simple fiberglass web is used for the guideway structure. The web is solid and only 0.1 inches thick, so the stator connections cannot be made in the space inside the guideway structure. Instead, the stator helix is unfolded at each end of a winding

segment and each stator wire is crimped into a connector housing. The mechanical arrangement of a stator joint is depicted in Figure 2-28.

This joint configuration is satisfactory from a mechanical perspective, but leads to a significant disadvantage in the electrical properties of the stator winding. For most of the stator length the periodic winding pattern results in similar self and mutual inductances for each motor phase (*i.e.* the inductance matrix is 'balanced'). In the unfolded portion of the stator winding at the joint, however, some of the phases are no longer surrounded by their neighboring phases. This effect leads to an imbalance in the inductance matrix of the stator winding. The position sensor signals detected in the stator winding are distorted as a consequence of the inductance matrix imbalance. Methods for normalizing the position sensor data to null out the errors resulting from this effect are discussed in Chapter 3.

The appropriate stator winding connections are made by plugging a wiring harness onto the stator connectors. There are three types of wiring harnesses used in the model system. The simplest harness is for a 'straight' joint (*i.e.* the stator winding merely continues across the joint). An 'end' harness is used at a zone boundary joint, and a 'inverter' harness is used at a joint where the inverter connects to the stator windings. The details of each of these wiring harnesses are outlined in appendix A.

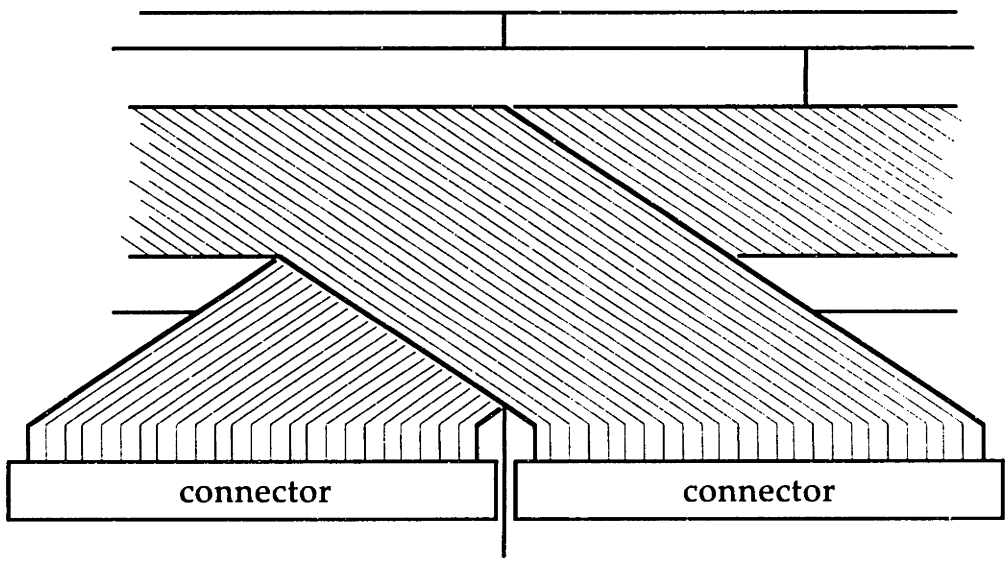


Figure 2-28. Stator winding joint

motor control

Synchronous operation of the LSM is maintained by controlling the power inverter to obtain a stator current pattern that coincides with the measured vehicle position. As the vehicle position changes, the stator

current is adjusted to maintain a stationary or synchronous relationship between the field array on the vehicle and the current pattern in the guideway stator winding. This method is referred to as 'field-oriented control' [16], and it allows the motor to be operated at a torque angle of 90°. When operated at this torque angle the motor produces the maximum possible thrust per Ampere of stator current. Open-loop motor drives, which are common in constant speed motor systems, must operate at a less efficient torque angle to ensure stable synchronous operation. In addition, the motor must have a set of damper windings on the rotor to stabilize the hunting transient associated with perturbations about the nominal torque angle. The hunting transient of a synchronous motor without damper windings can be unstable [17]. Damper windings will not work properly in a motor that utilizes superconducting coils to develop the excitation field since the superconducting coils act as a constant flux source, thus a closed-loop design is desirable.

To achieve optimum efficiency it is desirable to drive the multiphase windings of the guideway with current waveforms that have a shape similar to the back EMF induced in the windings by motion of the vehicle's excitation field. This waveshape is often sinusoidal, but the geometry of the guideway windings and vehicle excitation magnets could be designed to provide other shapes. Research papers concerning the design and analysis of motor drive techniques are readily available [18,19]. Further development of this research topic is not a goal of this thesis, but short discussions of two drive techniques are included here for reference.

PWM Control

A pulse width modulation scheme is an effective and popular means of controlling the shape of the winding currents in small to medium sized motors [20]. This technique is shown diagrammatically in Figure 2-29. The inverter switches the phase voltage rapidly between two states. The inductance of the phase filters out the high frequency components of the applied waveform, and the phase current responds to the average value of the applied voltage. The average value of the phase voltage is determined by the percentage of time spent in each of the switching states.

The PWM technique requires inverter switching at frequencies much higher than the frequency of the fundamental component of the phase currents. The switching devices required to perform this task at megawatt power levels (required in the full-scale system) are quite expensive.

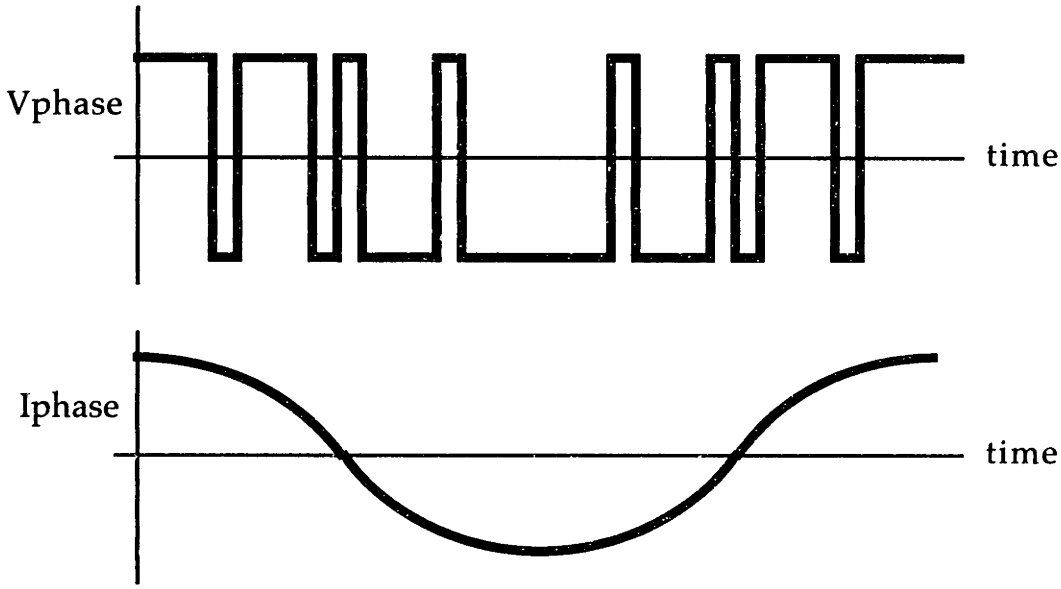


Figure 2-29. PWM waveshape control

6-Step Commutation

A more crude but simpler method of waveshape control is obtained by dividing a cycle of motor operation into a number of commutation steps, and driving the motor phases with a unique combination of switch states during each step. A 6-step commutation sequence (for a 3-phase motor) is shown in Figure 2-30. Note that each phase voltage is made up of transitions at the fundamental frequency of the motor. A 6-phase motor, such as the one used in the demonstration system, has 12 commutation steps.

The magnetic field of the vehicle moves smoothly down the track (*i.e.* its phase angle is a continuous function), but the magnetic field associated with the winding currents moves down the track in discrete steps (its phase angle is a discontinuous function). The resulting phase error between drive and vehicle position is shown in Figure 2-31. The magnitude of the motor's propulsive and normal forces vary with phase error as shown in Figure 2-32. The resulting ripple components of propulsive and normal forces are illustrated in Figure 2-33. For a 12-step sequence, the peak phase error is reduced to 15° and the ripple components of the propulsion and normal components are reduced. The ripple component in both the propulsion and normal forces increases if a position sensing error shifts the commutation pattern in phase. To minimize this effect, the accuracy of the position sensing system should be significantly better than the width of a commutation step. The position sensing accuracy desired with a 12-step commutation sequence is thus on the order of a few electrical degrees of the motor. This accuracy corresponds to a position sensing system error of 1° since the position sensor is based on a pole pitch twice as long as the motor pole pitch.

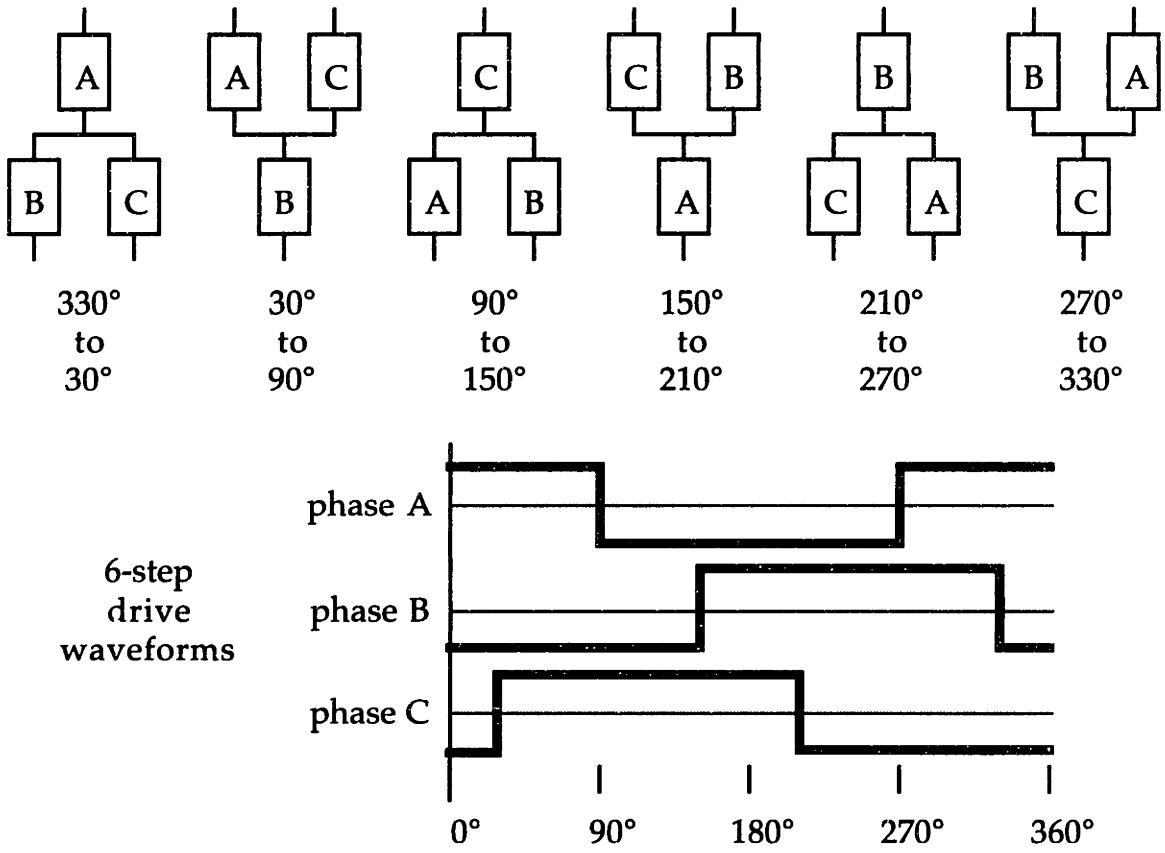


Figure 2-30. 6-step commutation

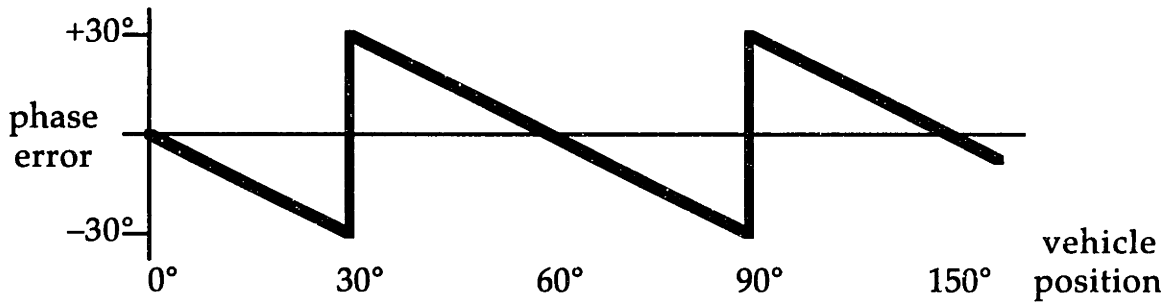


Figure 2-31. Phase error for 6-step commutation

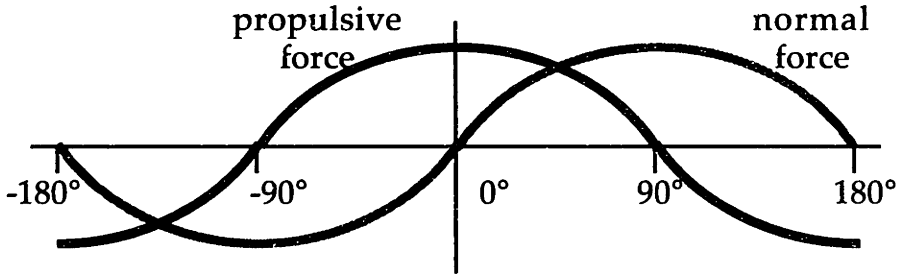


Figure 2-32. Propulsive and normal forces vs. phase error

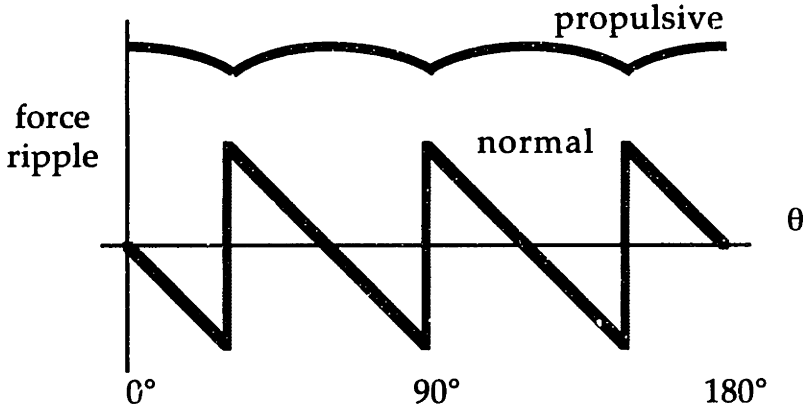
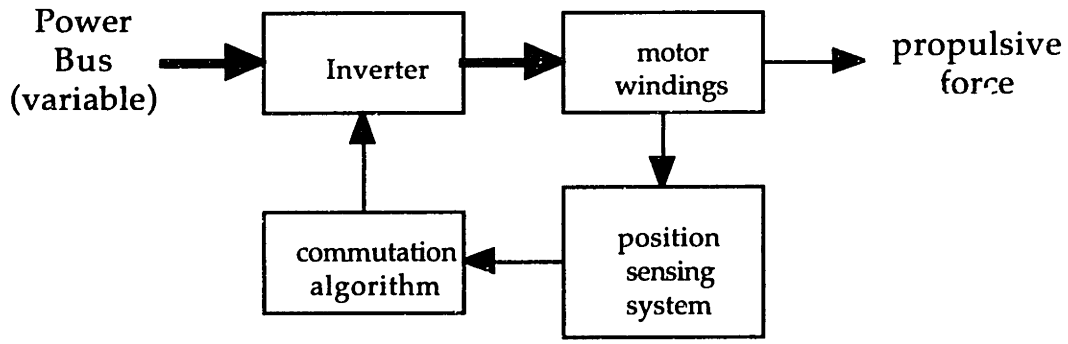
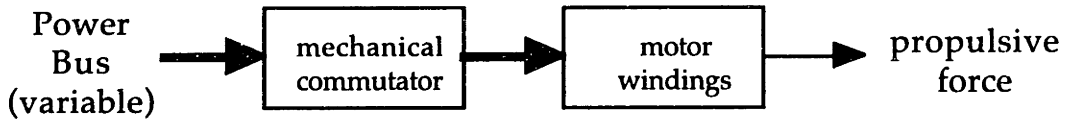


Figure 2-33. Propulsive and normal force ripple vs. phase angle

The combination of the position sensing system, the commutation algorithm, and the power inverter provide a function similar to (but more flexible than) the mechanical commutator found in a dc motor. This relationship is pictured in Figure 2-34. The power bus is varied to control the force produced by the motor. The dynamics of the transfer function relating motor output thrust to the input power bus magnitude are essentially linear, which simplifies the task of controlling the vehicle position or velocity. Power from the dc link is regulated to form the variable power bus by using a chopper.



AC motor with electronic position sensing/commutation system



DC motor with mechanical commutation system

Figure 2-34. Comparison to DC motor

3. Position Sensing

overview

The zone controller located at the wayside requires measurements of the vehicle position and velocity to determine the appropriate currents with which to drive the stator. The vehicle position and velocity must be known to control the spacing of vehicles, to hand-off control at zone boundaries, to commutate the LSM, and so forth. This chapter describes the method used to implement a wayside-based sensing system to accurately measure the vehicle position. The hardware and software algorithms used to implement this sensing technique are described in this chapter. The analog processing board used to detect and demodulate signals and the microcontroller board used to host the observer software are discussed further in Chapter 4.

electrical position vs. mechanical position

For the MagLev linear motor, the electrical (or relative) position of the vehicle is expressed as an angle (in radians or degrees). This periodic position measure is used to control the commutation of phase voltages or currents, and an accuracy of about 1° is necessary to effectively synchronize the stator currents to the vehicle motion. This accuracy corresponds to a displacement of about 1 cm for a full size system, or about 0.2 mm for the model system. The mechanical, or absolute, position of the vehicle is expressed as a distance (in meters or kilometers). This position is used to determine the position or velocity profile of the vehicle and the 'hand-off' of the vehicle from one zone to another. An accuracy of about 1 meter is sufficient to control vehicle spacing, an accuracy of 0.1 meters may be required to position the vehicle for loading or unloading. Figure 3-1 illustrates the relationship between electrical and mechanical position (the figure is drawn for a constant vehicle speed).

Accurate sensing of the electrical position is easier to implement than accurate sensing of the mechanical position. Since the spatial period of the stator windings is a known constant the change in mechanical position of the vehicle may be deduced by counting the number of elapsed periods of electrical position. The absolute mechanical position cannot be deduced via measurement of the electrical position. Additional position information from a mechanical position reference is required to procure an absolute position measurement. Mechanical position references are obtained by using Hall-effect sensors placed at regular intervals along the guideway. The Hall-effect sensor produces a pulse transition when a magnet pole from the

vehicle excitation array passes over the sensor. The vehicle is at a known (*i.e.* reference) mechanical position when the transition occurs. The mechanical position counter is compared to the reference position when the vehicle passes by a mechanical position reference sensor. Any discrepancy is corrected and logged as an error. Once the counter is corrected, the electrical position sensor tracks the changing mechanical position of the vehicle.

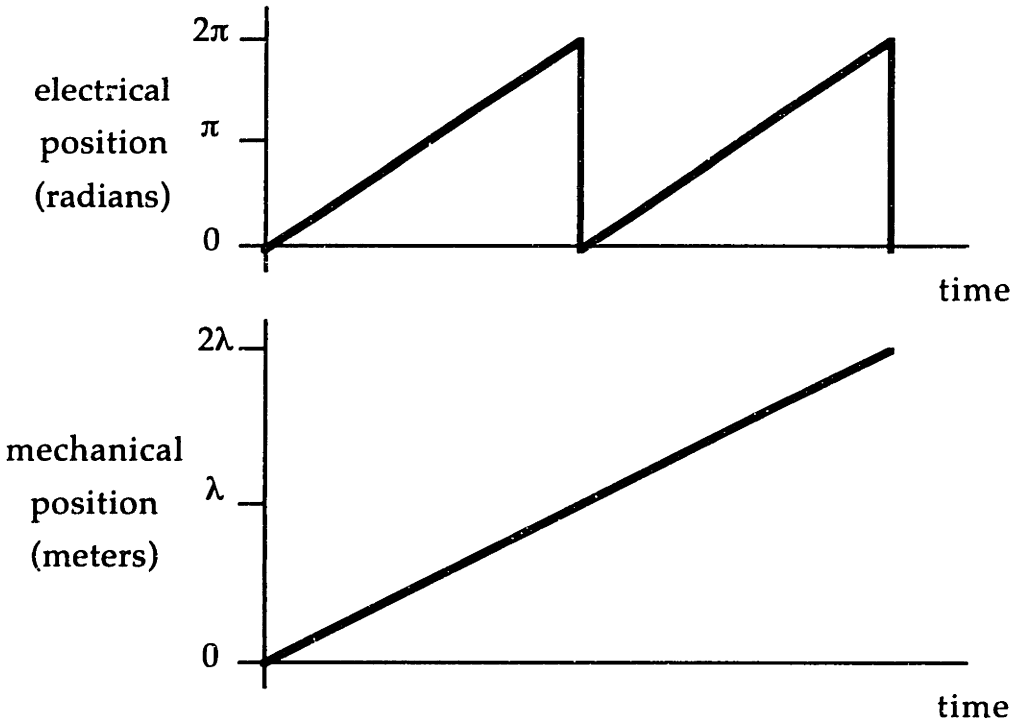


Figure 3-1. Comparison of electrical and mechanical position

position sensor concept

A vehicle-based position sensor with a radio communication link to the wayside controller can be used to provide the necessary position measurements to the zone controller [9]. The vehicle-based sensor must detect the relative position of the stator current pattern. Since the relative position is stationary when the LSM is driven in a synchronous manner, the sensor uses Hall-effect devices. It is difficult to accurately measure the relative position of the small magnetic field associated with the stator current in the presence of the large magnetic field associated with the vehicle field array. The measurement difficulty is further compounded by the heave and sway components of vehicle motion, which can couple field variations into the sensor that appear similar to those resulting from position variations.

An alternative electrical position sensing concept is pictured in Figure 3-2. A transducer mounted on the vehicle induces signals into the stator winding, and the amount of signal induced into each of the stator

phases varies as a function of the vehicle position. Circuitry located at the wayside detects the amount of signal induced in each of the stator phases and uses this information to compute the vehicle position. The stator winding is used both to conduct the propulsion currents that provide thrust to the vehicle and to conduct the position-dependent signals that provide a means of sensing the electrical position of the vehicle with wayside circuitry.

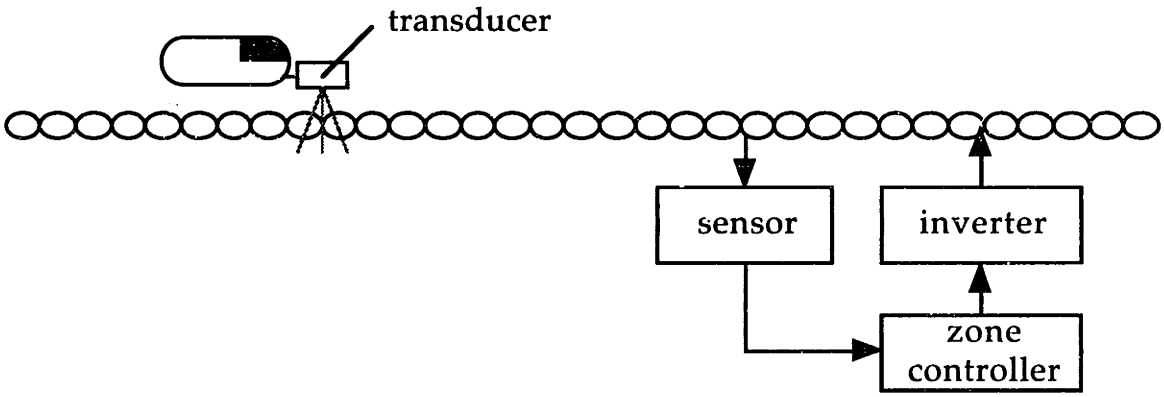


Figure 3-2. Position sensing concept

vehicle transducer

The transducer, which is an inductor that links flux with the stator winding, and its driver are located on board the vehicle. Figure 3-3 shows the vehicle transducer and its flux linkage to the stator winding. The inductor is driven with a current $I_L = I_o Sin(\omega_c t)$, which links a flux $\Phi_L = \Phi_o Sin(\omega_c t)$ with the phases of the stator winding located in the guideway. To distinguish the position signals from the motor drive waveforms, the carrier frequency ω_c is much higher than the electrical drive frequency. The electrical drive frequency at the maximum vehicle speed is approximately 80 Hz. A carrier frequency of 25 kHz is chosen since it is much larger than the drive frequency.

The mutual coupling between the inductor and each of the stator phases, and thus the amount of flux linked by each of the stator phases, varies as a function $\mathcal{F}(\theta)$ of vehicle position. The function $\mathcal{F}(\theta)$ is periodic, and its shape depends on many design details (*e.g.* inductor pole shape and pitch, winding shape). $\mathcal{F}(\theta)$ can be determined analytically through the application of Maxwell's equations, or $\mathcal{F}(\theta)$ can be determined experimentally through simple measurements. The fundamental component of the Fourier series representing $\mathcal{F}(\theta)$ usually dominates; the discussion of the observer error later in this chapter shows that the presence of harmonics degrades the accuracy of the position sensing system.

The mutual coupling function $\mathcal{F}$ has the same amplitude and waveshape for all of the motor phases, but it is offset in position for each phase since the stator phases are spatially offset from one another. The stator winding is divided into two 3-phase sets; the A, B, and C phases are offset by 120° relative to one another as are the X, Y, and Z phases. The propulsion current for phase A is offset 90° from phase X to minimize the thrust ripple from the motor. (*i.e.* The phases are uniformly spaced for propulsion purposes.) The spacing between the phase A and phase X position sensor signals is only 45° since the position sensor utilizes a pole pitch twice as long as the propulsion field array. (Note that the phase letters shown in the stator winding depicted in Figure 3-3 relate to the position sensor signals. Propulsion currents from phase A of the inverter flow into the A and a windings in Figure 3-3, but propulsion currents from phase B of the inverter flow into the C and c windings in Figure 3-3. Similarly, inverter phase C -> B and b; inverter phase X -> X and x; inverter phase Y -> Z and z; and inverter phase Z -> Y and y.)

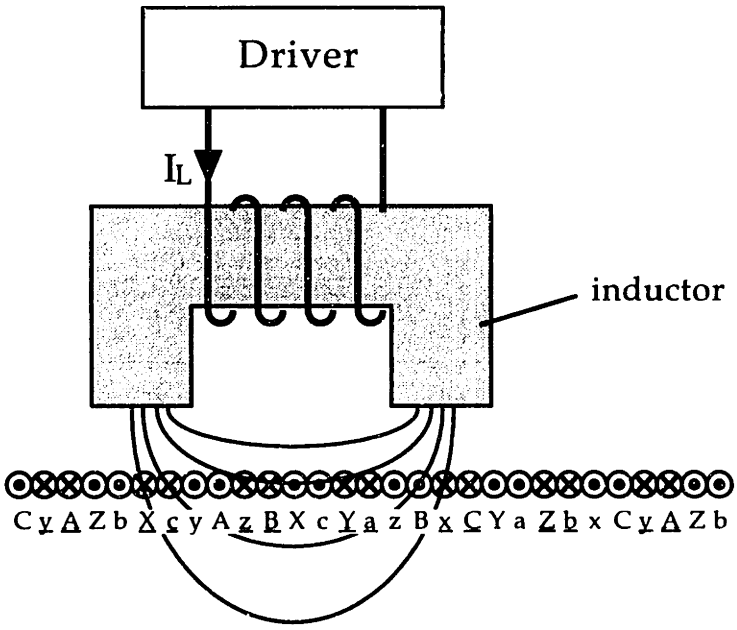


Figure 3-3. Vehicle transducer coupling with stator

Figure 3-3 is drawn with the vehicle in the position that links the maximum possible amount of transducer flux with stator phase A. We will define this position as having a value $\theta = 0°$. The $\mathcal{F}'(\theta) \bullet \mathrm{Sin}\omega_c t$ terms are neglected since $\omega_c >> \omega_{motor}$, so the voltage induced in each of the stator phases by the time-varying flux Φ_L is:

$$
\begin{aligned}
V_{phaseA} &= V_A \cdot \mathrm{Cos}\omega_c t = V_o \cdot \mathcal{F}(\theta) \cdot \mathrm{Cos}\omega_c t \\
V_{phaseB} &= V_B \cdot \mathrm{Cos}\omega_c t = V_o \cdot \mathcal{F}(\theta - 2\pi/3) \cdot \mathrm{Cos}\omega_c t \\
V_{phaseC} &= V_C \cdot \mathrm{Cos}\omega_c t = V_o \cdot \mathcal{F}(\theta + 2\pi/3) \cdot \mathrm{Cos}\omega_c t \\
V_{phaseX} &= V_X \cdot \mathrm{Cos}\omega_c t = V_o \cdot \mathcal{F}(\theta - \pi/4) \cdot \mathrm{Cos}\omega_c t \\
V_{phaseY} &= V_Y \cdot \mathrm{Cos}\omega_c t = V_o \cdot \mathcal{F}(\theta - 11\pi/12) \cdot \mathrm{Cos}\omega_c t \\
V_{phaseZ} &= V_Z \cdot \mathrm{Cos}\omega_c t = V_o \cdot \mathcal{F}(\theta + 5\pi/12) \cdot \mathrm{Cos}\omega_c t
\end{aligned}
$$

Each of the phase voltages represents a signal at the carrier frequency ω_c amplitude modulated by the function $\mathcal{F}(\theta)$ (or $\mathcal{F}$ of θ shifted by a constant). The induced voltages are suppressed carrier amplitude modulated waveforms, and the frequency of the carrier is chosen to be much higher than the frequency of position variation. Figure 3-4 shows a plot of V_{phaseA} for a the vehicle accelerating from a stationary position at $\theta = 0$. For clarity, the modulating frequency ω_c has been reduced to a value of 10 Hz instead of 25 kHz. The function $\mathcal{F}(\theta)$ is assumed to have only a fundamental component, *i.e.* $\mathcal{F}(\theta) = \mathrm{Cos}(\theta)$. Figure 3-5 shows the modulating signal V_A for the same vehicle run. Note that V_A is the 'envelope' of the modulated waveform V_{phaseA}. The remaining motor phases show similar signals, but the envelope of the signal in each phase is offset in position.

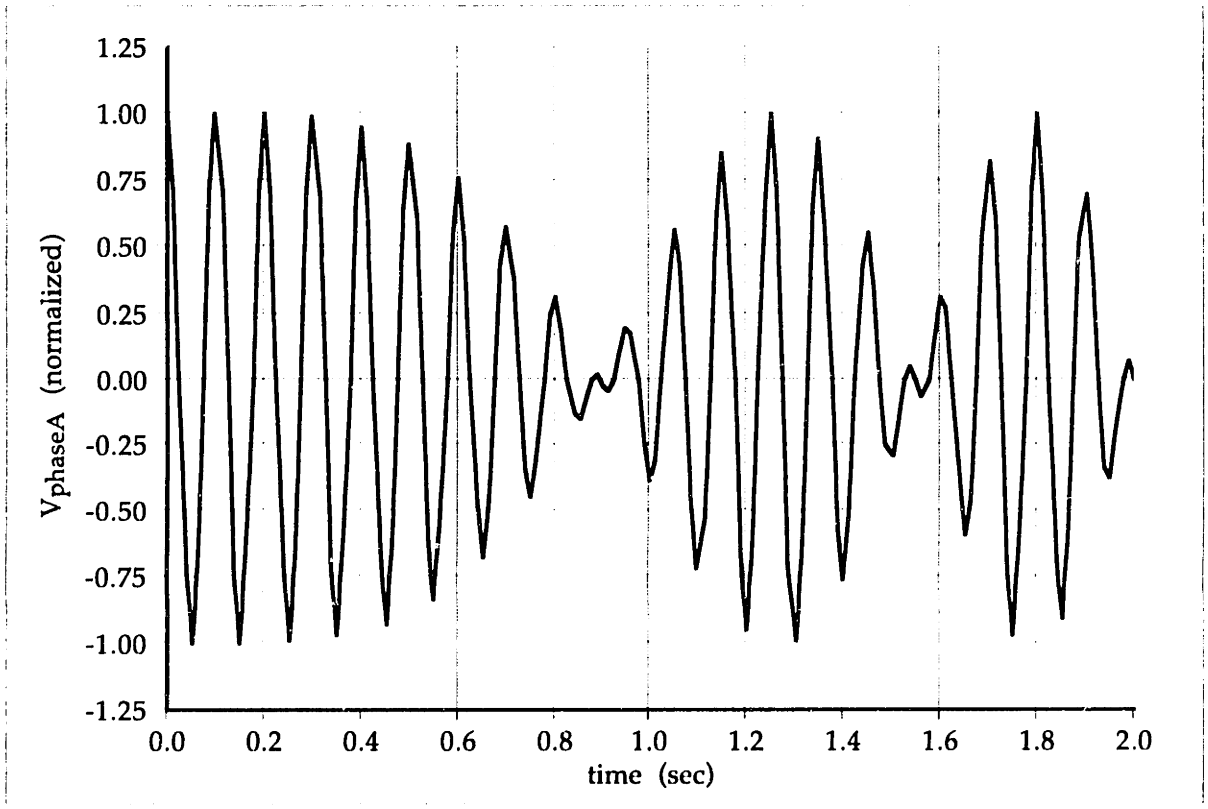


Figure 3-4. Modulated carrier for accelerating vehicle

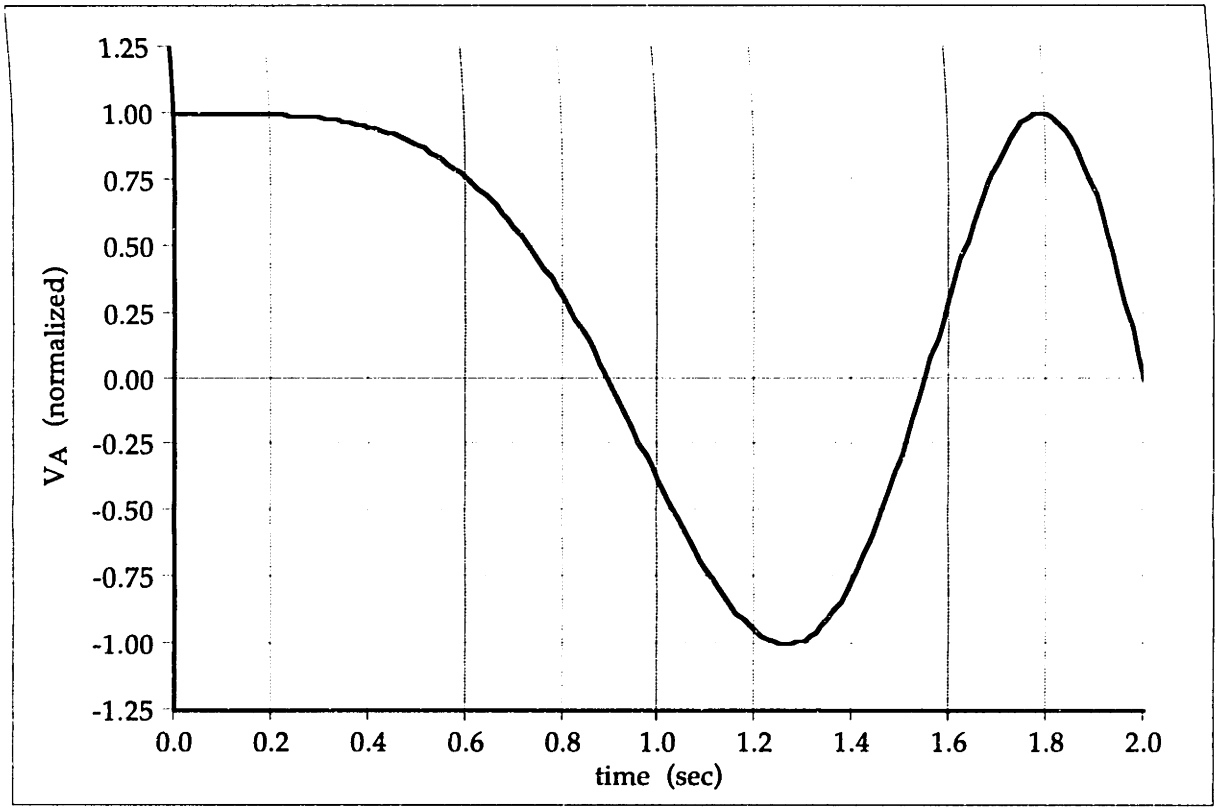


Figure 3-5. Modulating signal for accelerating vehicle

induced currents

The currents induced in the stator windings by the vehicle transducer are related to the induced voltages through the stator impedance:

$$\mathbf{e}_S = (s \bullet \mathbf{L}_S + \mathbf{R}_S) \bullet \mathbf{i}_S$$

where $\mathbf{e}_S$ is the vector of induced voltages, $\mathbf{i}_S$ is the vector of induced currents, $\mathbf{L}_S$ is the inductance matrix, and $\mathbf{R}_S$ is the resistance matrix. ($\mathbf{L}_S$ and $\mathbf{R}_S$ are the stator inductance and resistance matrices for position sensing signals, the stator inductance and resistance matrices for propulsion signals are different.) The induced stator currents are detected at the wayside using current transformers and processed by the analog processor board.

back EMF sensing

The voltage induced in the stator winding by the vehicle field array, known as the 'back EMF' of the motor, can be used to determine the vehicle position. This form of position sensing has been successfully employed in the control of rotary machines [21,22]. The concept is briefly presented here to demonstrate how it might be incorporated into a MagLev system. Since this technique is well represented in the literature, however, it is not a primary

focus of this thesis and was not included in the implementation of the scale model control system.

The flux from the permanent magnets in the vehicle links with the stator windings, thus movement of the vehicle induces a voltage in the stator windings. This induced voltage is often referred to as a 'back EMF' or a 'speed voltage'. The back EMF is periodic and has a waveshape, $\mathcal{G}(\theta)$, that is similar to the function $\mathcal{F}(\theta)$ associated with the position sensing inductor. The magnitude of the back EMF is proportional to the vehicle speed. Both the constant of proportionality associated with the back EMF magnitude, K_t, and the waveshape, $\mathcal{G}(\theta)$, can be determined analytically using Maxwell's equations and the motor geometry, or they can be determined experimentally. The back EMF induced in each phase is of the form:

$$
\begin{aligned}
E_{phaseA} &= \mathcal{G}(\theta) \bullet K_t \bullet v \\
E_{phaseB} &= \mathcal{G}(\theta - 2\pi/3) \bullet K_t \bullet v \\
E_{phaseC} &= \mathcal{G}(\theta + 2\pi/3) \bullet K_t \bullet v \\
E_{phaseX} &= \mathcal{G}(\theta - \pi/2) \bullet K_t \bullet v \\
E_{phaseY} &= \mathcal{G}(\theta - 7\pi/6) \bullet K_t \bullet v \\
E_{phaseZ} &= \mathcal{G}(\theta + \pi/6) \bullet K_t \bullet v
\end{aligned}
$$

Each induced EMF is the product of the mutual coupling term $\mathcal{G}(\theta)$ (or $\mathcal{G}$ of θ offset by a constant) and an amplitude proportional to the vehicle velocity. The EMFs are grouped in two 3-phase sets; the A, B, and C waveforms are offset by 120° from one another, as are the X, Y, and Z waveforms. The phase A and phase X waveforms are offset by 90° from one another. There is no sinusoidal carrier term in the induced back EMF waveforms as there was in the position sensor signals. The field magnets that result in the back EMFs are constant flux sources, not sinusoidally varying sources.

The magnitude of the back EMFs is proportional to the speed of the vehicle; thus at slow vehicle speeds the back EMFs do not provide data adequate for position sensing. The induced back EMF cannot be measured directly due to the voltage drops associated with the stator current flowing through the resistance and inductance of the stator windings. However, by measuring both the stator voltage and stator current, the induced back EMF can be determined:

$$\mathbf{e_p} = \mathbf{v_p} - (s \bullet \mathbf{L_p} + \mathbf{R_p}) \bullet \mathbf{i_p}$$

where $\mathbf{e_p}$ is the vector of induced EMFs, $\mathbf{v_p}$ is the vector of phase voltages, $\mathbf{i_p}$ is

the vector of phase currents, $\mathbf{L_p}$ is the inductance matrix, and $\mathbf{R_p}$ is the resistance matrix. ($\mathbf{L_p}$ and $\mathbf{R_p}$ are the stator inductance and resistance matrices for propulsion signals. The stator inductance and resistance matrices for position sensing signals are different.) The value of vehicle position which best fits the back EMF data can be determined in exactly the same manner used in the inductor-based position sensing system, by locking an observer onto the incoming back EMF data. If both position sensing systems are used, the same observer states would be compared to both sets of incoming data, and the observer innovation would be a weighted sum of the error from each data set.

observer

An identity observer [23] is used to provide estimates of the vehicle position and velocity based on measurements of the position sensor signals induced into the stator winding by the vehicle transducer. The observer is a dynamic model of the actual system; a second order observer is used to provide an estimate of both the vehicle position and velocity. The position estimate from the observer is compared to the position sensor data, and any resulting error is used to improve the observer estimate. This process forces the observer states to track the states of the actual system. (As discussed previously, the absolute position of the vehicle cannot be resolved solely from the electrical position data presented to the observer. Once the observer's position is reset by a mechanical position reference transition the observer will provide the correct absolute position of the vehicle.) In addition to providing velocity estimation from position measurements, the observer filters the incoming position measurements to provide a smoother position estimate. The tracking dynamics and filtering dynamics of the observer are related. The observer is implemented digitally in the code written for the microcontroller board.

Two approaches to implementing an observer were used in the model vehicle system. The first approach used a nonlinear computation algorithm to compute the vehicle position based on the position sensor signal measurements, followed by a linear observer to filter the computed position data and provide an estimate of velocity. The second approach used a nonlinear observer; the nonlinear computations are included inside the observer loop. Both approaches are discussed in this chapter, and the latter option was chosen for use in the model system. Most of the discussion in this chapter is based on a three phase system, but is easily extended to systems with a larger number of motor phases (such as the six-phase motor used in the model system).

nonlinear computation / linear observer

A block diagram for the observer implementation using a nonlinear computation algorithm followed by a linear observer is shown in Figure 3-6. This method offers the advantage of simple modelling and specification of the observer dynamics. The design and analysis of a linear observer is presented in [23], so the focus of the following material will be on the nonlinear computation algorithm. Note that the vehicle transducer, the coupling of flux to the stator winding, and the detection and demodulation hardware have been abstracted in Figure 3-6. The thrust applied to the vehicle by the LSM is not provided as an input to the observer implemented in the demonstration system. This omission introduces a known modelling error which is discussed in a later section.

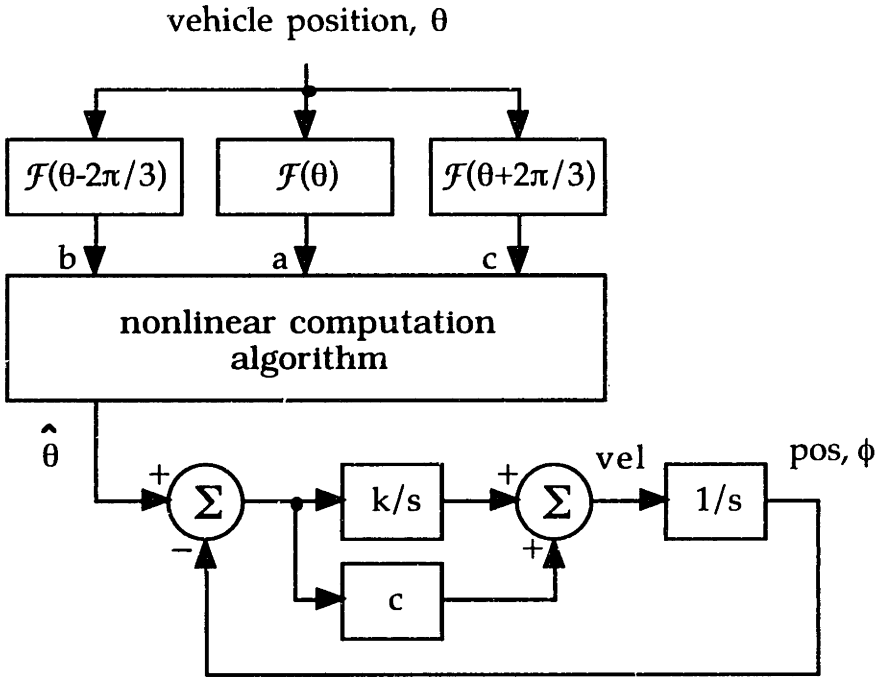


Figure 3-6. Nonlinear computation / linear observer

The nonlinear computation algorithm must compute the vehicle position based on the sensor measurements a, b, and c. The functions $\mathcal{F}(\theta)$, $\mathcal{F}(\theta-2\pi/3)$, and $\mathcal{F}(\theta+2\pi/3)$ relating vehicle position to a, b, and c are known. The data measurement a can be operated on by the inverse of $\mathcal{F}(\theta)$ to produce two candidate position values, as shown in Figure 3-7. The function $\mathcal{F}(\theta)$ is periodic and therefore the inverse of $\mathcal{F}(\theta)$ must be multiple valued over the possible range of the a data. In the model system $\mathcal{F}(\theta)$ is approximately equal to Cos(θ). Similar computations on the data b and c lead to three pair of candidate position values, as shown in Figure 3-8.

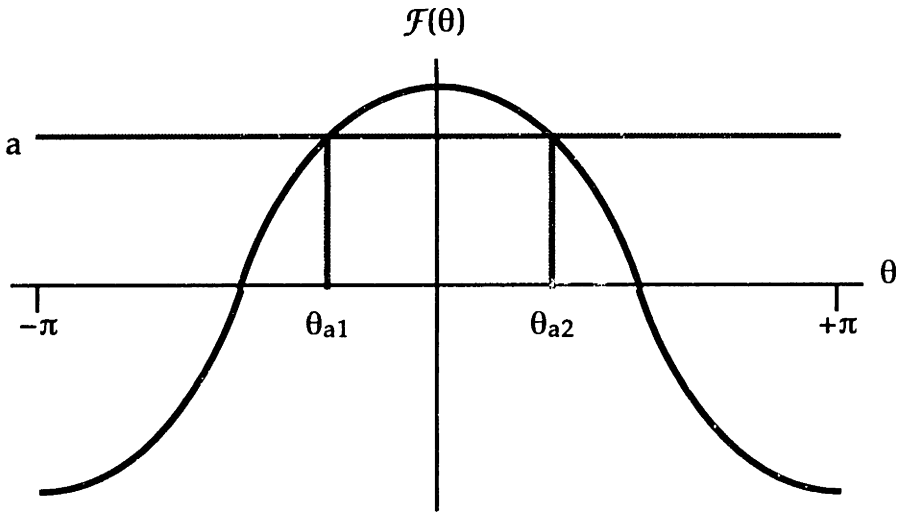


Figure 3-7. Computing the function $\mathcal{F}^{-1}(a)$

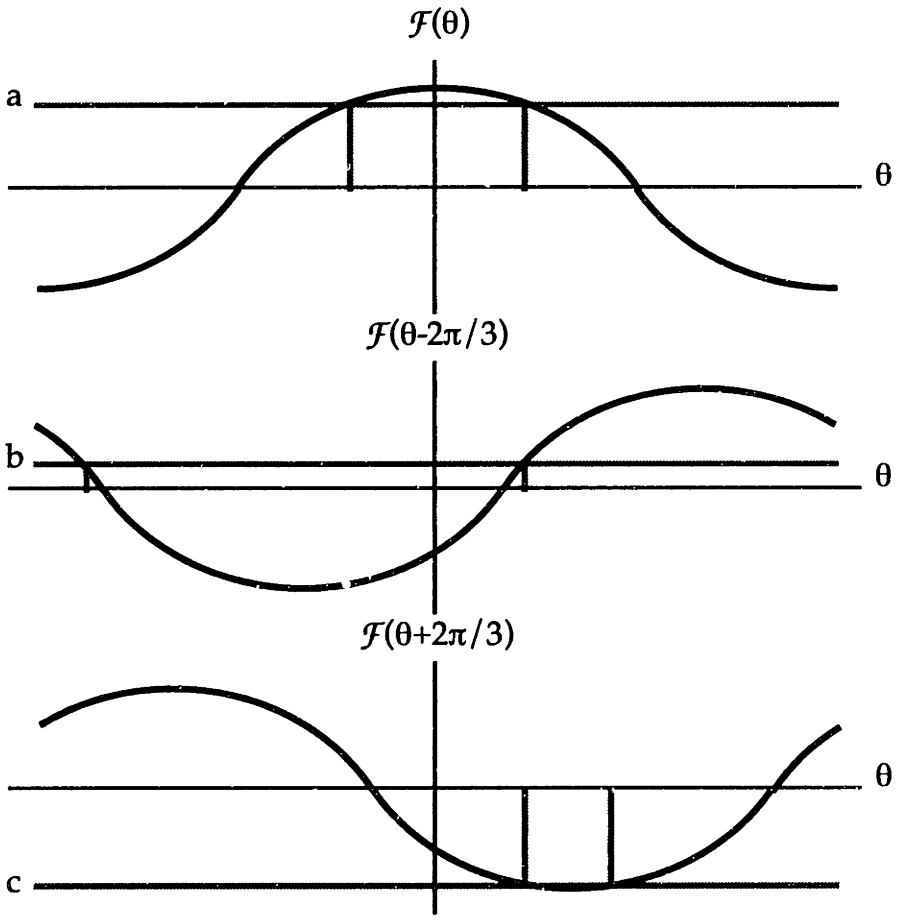


Figure 3-8. $\mathcal{F}^{-1}$ computation for all three phases

There are eight possible combinations using one candidate position from each of the three motor phases that result from the three pair of candidate positions. The combination with the smallest variance is chosen as the 'best', and the average of this combination is the 'best estimate'.

This algorithm was successfully implemented in a three phase version of the model vehicle system. The function $\mathcal{F}^{-1}$ was stored in a look-up table to reduce computation time, and data points out of the valid range (due to measurement noise) were clipped before look-up. The performance of the observer was adequate, but this implementation does have a few shortcomings.

The number of combinations resulting from the $\mathcal{F}^{-1}$ computation increases dramatically as the number of motor phases is increased; the number of combinations is 2^n where n is the number of motor phases. Computing the average and variance of each combination requires $3n$ additions and n multiplications, so a six-phase system would require 1200 additions and 400 multiplications. Moving to a 12-phase system would require 150,000 additions and 50,000 multiplications. (This configuration is quite plausible in a full-size system.) For a system of more than three phases the microcontroller used to implement the observer code cannot perform the required computations in the time allotted. It would be plausible to split the data into 3-phase groups before applying the nonlinear computation algorithm. The results of the computation performed on each group of data would then be averaged to produce an aggregate 'best estimate'. This technique substantially reduces the computational complexity of the nonlinear computation algorithm.

A second problem with the nonlinear computation / linear observer implementation is the lack of a solid model for the effects of noise on the input data. The system has many nonlinearities that are difficult to model, including the limiting that precedes table look-up and the discontinuities resulting from a switch of the combination chosen as 'best'. Without a model to sufficiently describe the effect of input noise, it is difficult to determine how to optimize the algorithm to result in the best rejection of noise. The effect on the noise rejection performance of splitting the data into 3-phase groups is not obvious.

The final difficulty with the nonlinear computation / linear observer method is the need to normalize the incoming data to unity magnitude before computing the $\mathcal{F}^{-1}$ computation. This normalization slightly increases the computational complexity of the nonlinear computation algorithm.

nonlinear observer

The nonlinear observer implementation shown as a block diagram in Figure 3-9 incorporates the nonlinearities associated with the function $\mathcal{F}(\theta)$ inside the observer loop. The observer position estimate ϕ is used to compute the expected value of the position sensor signal in each motor phase. This

computation is implemented by using a look-up table for the function $\mathcal{F}(\phi)$. $\mathcal{F}(\phi)$ is single-valued and no range clipping is required, so this operation is simpler than the $\mathcal{F}^{-1}$ look-up used with the nonlinear computation / linear observer. The measured and expected position sensor values are compared and the resulting error from each phase is multiplied by a weighting function $w(\phi)$, which is also computed via a look-up table. (Criteria used to choose the weighting function will be addressed in the following section.) The weighted errors are combined to obtain the observer innovation, which drives the observer states along a trajectory to reduce the error between the estimated and actual vehicle position. Once again a second order observer is used to obtain an estimate of the vehicle velocity and a filtered estimate of the vehicle position.

The benefits of the nonlinear observer topology are a reduction in the computational complexity of the microcontroller implementation of the observer and a method for modelling the effects of noise or irregularities in the position sensor data. The dynamics associated with the nonlinear observer are nonlinear, so the lock properties of the observer must be verified.

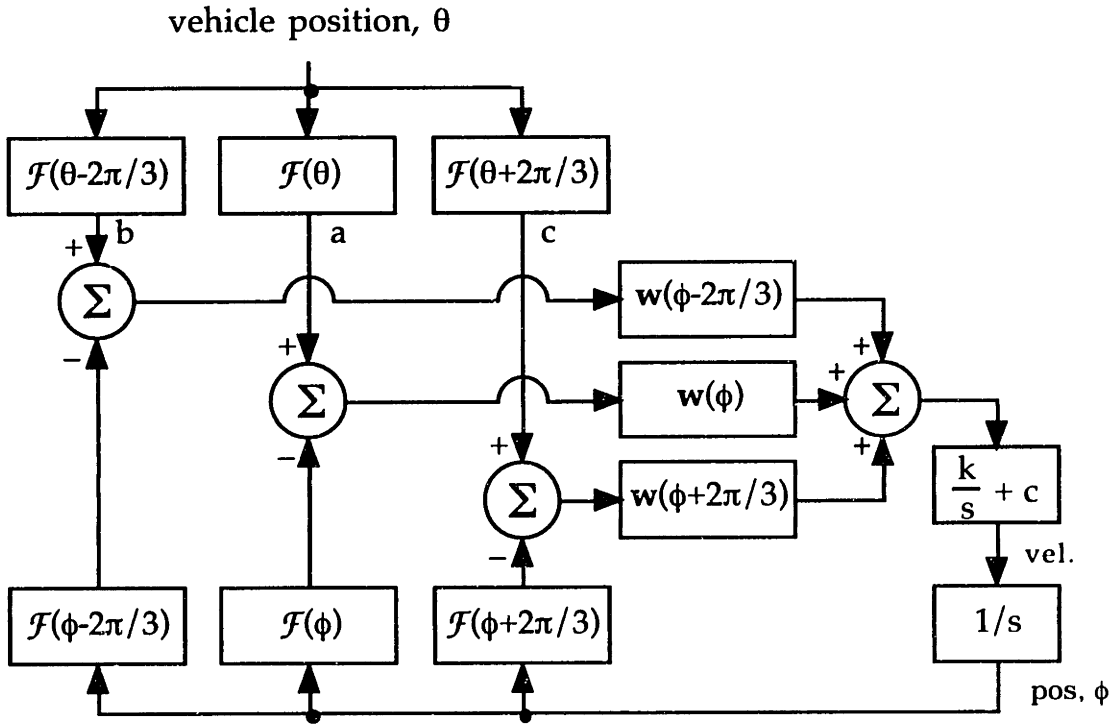


Figure 3-9. Nonlinear observer

weighting function

The weighting function can be selected to provide the optimal rejection of uncorrelated noise on the input position sensor data. The observer can be modelled as a Fisher type system; an analysis of the linearized observer with additive uncorrelated Gaussian noise on each of the inputs results in the weighting function choice of $\mathbf{w}(\phi)=d/d\phi\ \{\mathcal{F}(\phi)\}$ [24]. Since the function $\mathcal{F}$ is a cosine, $\mathbf{w}(\phi) = -\mathrm{Sin}(\phi)$. (The function $\mathcal{F}$ may also contain odd harmonics; the system is designed such that the harmonics are as small as possible. The effect of the harmonics is considered in a later section.) With these choices for $\mathcal{F}$ and $\mathbf{w}$, the observer topology in Figure 3-9 can be simplified by noting that:

$$\mathrm{Cos}(\phi)\bullet\mathrm{Sin}(\phi) + \mathrm{Cos}(\phi-2\pi/3)\bullet\mathrm{Sin}(\phi-2\pi/3) + \mathrm{Cos}(\phi+2\pi/3)\bullet\mathrm{Sin}(\phi+2\pi/3) = 0$$

for any value of ϕ. The comparison and the $\mathcal{F}(\phi)$ look-up operations can thus be eliminated resulting in the simplified observer topology shown in Figure 3–10. The phase detector used in this observer is implemented with a bank of multipliers, and is a three phase version of a phase detector circuit commonly found in phase locked loop designs [25]. The three phase bank of multipliers has some interesting and beneficial properties when used as a phase detector. As shown in a later section, the double frequency term associated with a single multiplier phase detector is eliminated. This effect removes the requirement for the maximum loop bandwidth to be less than the frequency of the incoming data. This feature is essential since the input data is static when the vehicle is stationary.

Computation of the observer innovation in the simplified nonlinear topology requires n multiplications and n additions, where n is the number of motor phases. The computation complexity increases only linearly as the number of motor phases is increased, and the time required to perform the computation for the six-phase system is insignificant. Thus the nonlinear observer implementation offers a substantial improvement in terms of computational complexity over the nonlinear computation / linear observer implementation.

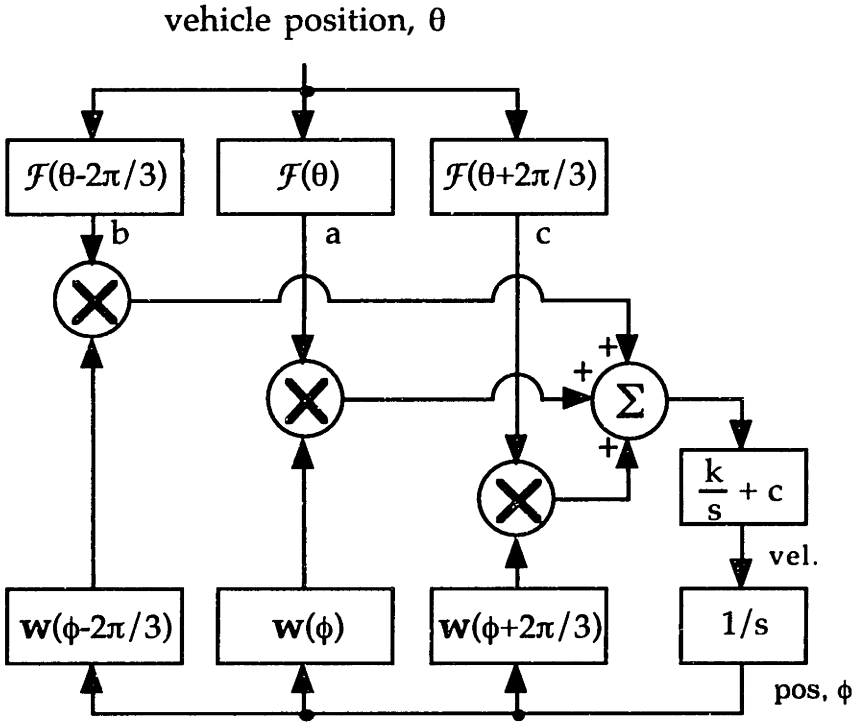


Figure 3-10. Simplified nonlinear observer

nonlinear observer dynamics

The nonlinear observer topology may be further simplified for the purpose of investigating the loop dynamics by noting that:

$$\cos(\theta)\cdot\sin(\phi) + \cos(\theta-2\pi/3)\cdot\sin(\phi-2\pi/3) + \cos(\theta+2\pi/3)\cdot\sin(\phi+2\pi/3) = 1.5\sin(\theta-\phi).$$

The simplified block diagram is shown in Figure 3-11, and the error dynamics of this loop are governed by the state equation:

$$\ddot{x} + 1.5\,c\,\dot{x}\,\mathrm{Cos}(x) + 1.5\,k\,\mathrm{Sin}(x) = 0$$

This differential equation also describes the dynamics of a phase locked loop using a multiplier phase detector and a P+I loop filter. This system is guaranteed to acquire lock [25]. For small error values $\mathrm{Cos}(x) \approx 1$ and $\mathrm{Sin}(x) \approx x$, thus the linearized dynamics (in the neighborhood of zero error) of the nonlinear observer are of the same form as the linear observer dynamics.

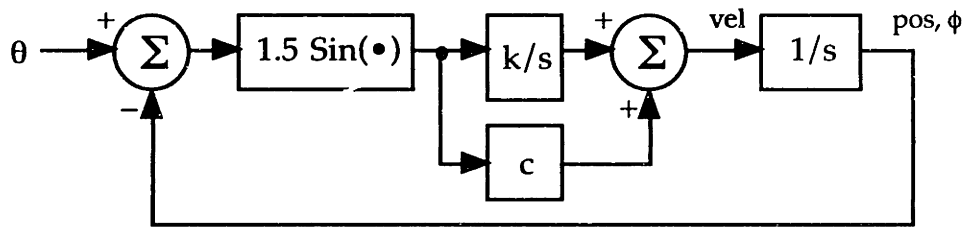


Figure 3-11. Nonlinear observer dynamic model

The observer loop gain constants k and c specify the dynamic response of the observer. There are several tradeoffs involved in selecting appropriate values for k and c. It is desirable for the observer to have a wide bandwidth to quickly eliminate any position error, but a narrow bandwidth helps to reject noise. The digitally implemented observer is a sampled data system, and the sampling delay will significantly degrade the stability of the observer dynamics if the observer bandwidth is set too high. The sampling interval used to update the observer in the demonstration system is 312.5 μsec, thus the observer bandwidth should be set well below 3200 rad/sec. The closed loop bandwidth of the velocity and position control loops used in the demonstration system are about 10 rad/sec. The effect of unmodelled system behavior on the closed loop dynamics is minimized by keeping the observer dynamics much faster than 10 rad/sec. As a compromise between these limits the linearized observer poles are set at 70 rad/sec and 180 rad/sec by choosing $k = 2084$ rad/sec^2 and $c = 83.3$ rad/sec. The response of the nonlinear observer to a position and a velocity error is shown in Figures 3-12 and 3-13. Note that the 'net error' plotted refers to the observer innovation (*i.e.* the output of the phase detector).

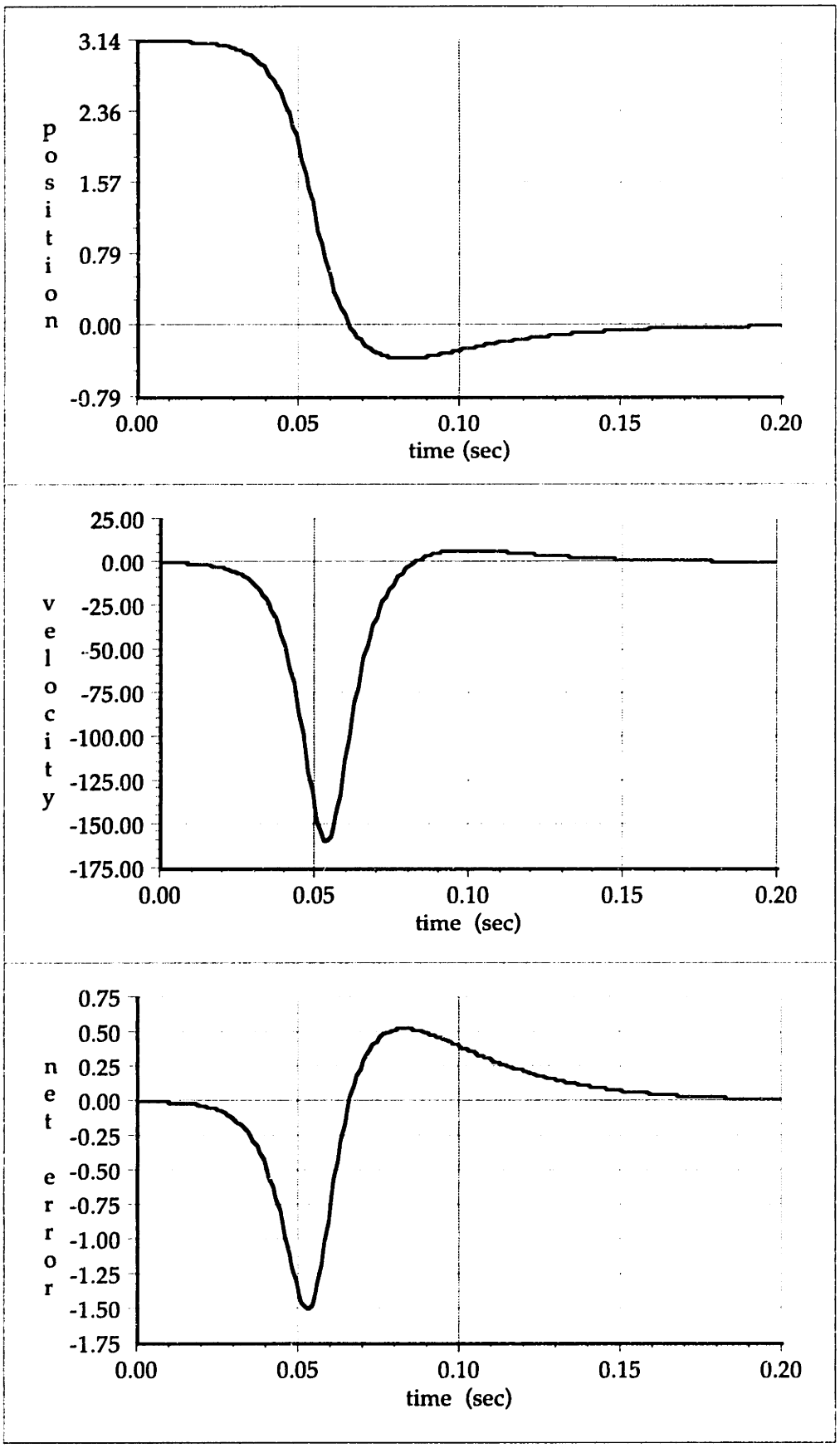


Figure 3-12. Observer response to step position error

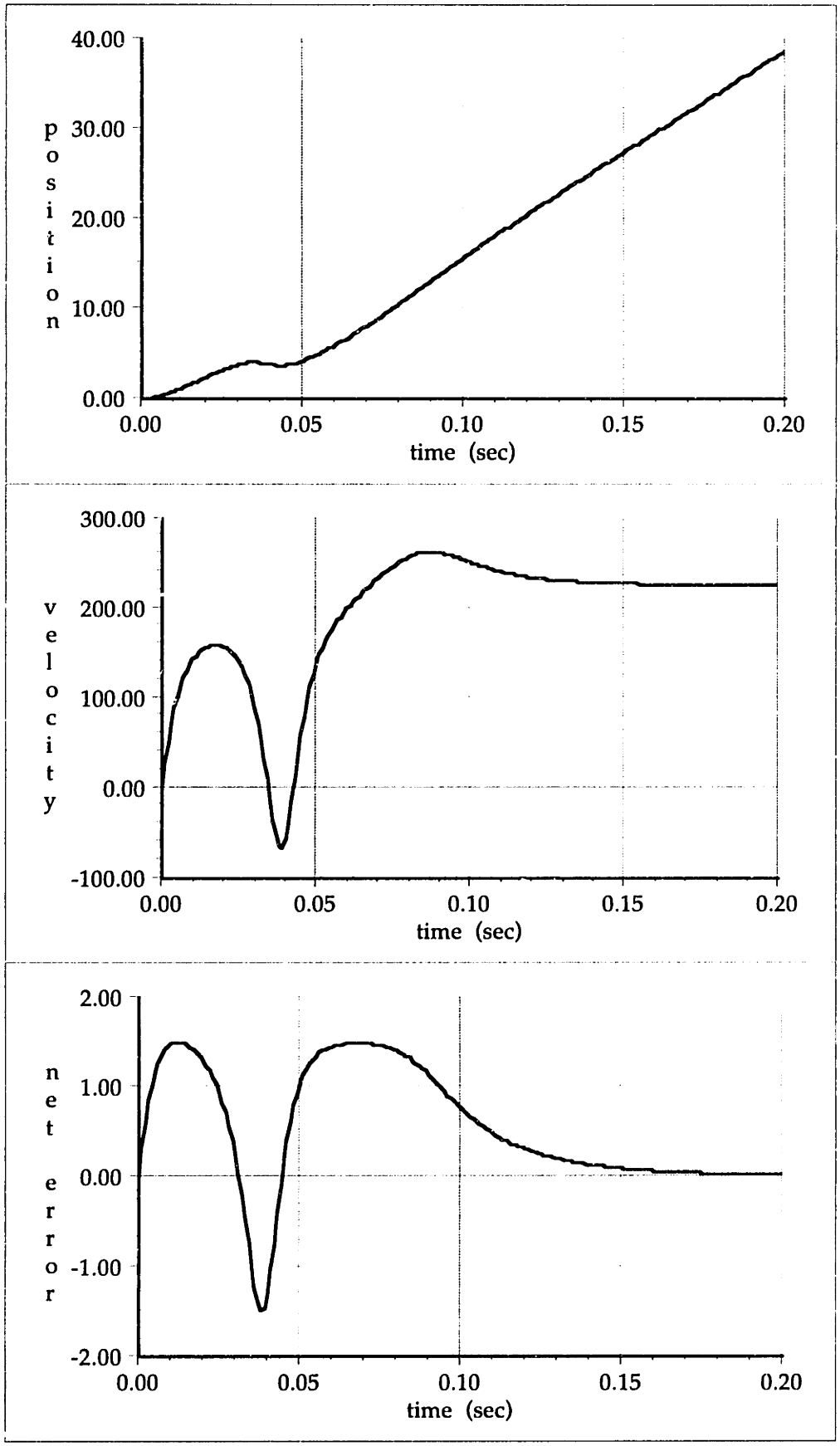


Figure 3-13. Observer response to step velocity error

The initial response to the step error of 3.14 radians is sluggish due to the nonlinearity of the phase detector. The system is near an unstable equilibrium point since Sin(3.14) ≈ 0. As the position error decreases towards $\pi/2$ the net error actually increases. For position error values smaller than $\pi/6$ the system response is approximately like that of a linear system. The response to the step velocity error of 225 rad/sec (this velocity error value corresponds to a full speed error) shows the observer slipping a position cycle before locking-in to track the vehicle.

The loop gain of the observer changes as the amplitude of the position sensor signals is varied. The previous calculations assume a position sensor signal amplitude of 10.24 Volts (this is the maximum value allowed before saturating the A/D converters). The observer dynamics become slower and less stable as the amplitude of the position sensor signals is reduced. This effect could be eliminated by normalizing the incoming position sensor signals to an amplitude of 10.24 Volts before applying the signals to the phase detector. This form of normalization was not included in the demonstration system. The position sensor signals in the demonstration must be larger than about 3.5 Volts to ensure adequate stability of the observer dynamics. The measured response of the observer to position and velocity errors is depicted in the zone switching transients shown in Chapter 2, Figures 2-4 and 2-5. The amplitude of the position sensor signals was reduced to the minimum value of 3.5 Volts for these plots.

The 3-phase (or 6-phase) multiplier used for the phase detector rejects the 'double frequency' term usually associated with the use of a multiplier phase detector in a single phase PLL system. In a single phase system the double frequency term must be attenuated by the loop filter to prevent harmonic generation in the voltage controlled oscillator of the PLL. This constraint limits the minimum input frequency at which the PLL can operate. (The constraint may also be viewed as a maximum limit on the PLL bandwidth.) The constraint between loop bandwidth and input frequency is not a concern since the multiple phase system eliminates the double frequency term, and the multiple phase observer can track even stationary inputs. A wide loop bandwidth can be used to improve the lock acquisition performance of the observer.

motor force input

The observer is intended to model the behavior of the actual vehicle and to track the states of the actual vehicle by forcing the error detected between the estimated and measured position sensor signals to zero. A motor force is applied to the actual vehicle to overcome drag or to provide

acceleration. The observer model implemented in the demonstration system does not include this feature, thus a known modelling error exists in the observer. The acceleration term is not included because its effect is small. The observer error is proportional to the acceleration applied to the vehicle, and this acceleration is limited to a maximum of 0.25 g's (2.5 m/sec^2). Since the observer bandwidth is much faster than the vehicle dynamics, this level of acceleration corresponds to a position error of about 0.8 mm or 3.8°. This error level is indeed large compared to the capability of the phase detector, but since the maximum acceleration is not often present in normal operation the acceleration error is tolerated. In a full size system the 0.8 mm error would correspond to a position error of only 0.077°, which is just below the accuracy limit of the phase detector.

The error term is easily corrected by computing the vehicle acceleration and applying this value to the observer. The vehicle acceleration is calculated by multiplying the motor current by the thrust constant to obtain the applied electrical thrust, subtracting the drag force, and dividing by the vehicle mass. A more refined computation would also include subtracting an acceleration component due to the guideway grade. The drag force is a nonlinear function of velocity, and is probably easiest to compute through a look-up table. Similarly, the grade varies as a function of position and this information could be obtained through a look-up table.

signal processing errors

The phase detector expects to receive a sinusoidal, balanced three phase set of position sensor waveforms, but error sources in the signal processing path alter the waveforms from this ideal behavior. Barring harmonics from the current discussion, the actual input signals to the observer will be of the form:

$$a = A \bullet \mathrm{Cos}(\theta+\psi_a)+\delta_a$$
$$b = B \bullet \mathrm{Cos}(\theta+\psi_b-2\pi/3)+\delta_b$$
$$c = C \bullet \mathrm{Cos}(\theta+\psi_c+2\pi/3)+\delta_c$$

where the amplitudes A, B, and C are nominally equal to one, the phase offsets ψ_a, ψ_b, and ψ_c are nominally equal to zero, and the dc offsets δ_a, δ_b, and δ_c are nominally equal to zero. Errors in the amplitudes and dc offsets arising from component value tolerance and op-amp offsets on the analog signal processor board are discussed in Chapter 4. (The amplitude errors associated with the analog processor board are about 1 to 2% and the dc offset errors are about 10 mV.) Variations in the self and mutual inductance of each phase of the stator winding result in amplitude and phase offset errors. The inductance variations are largely due to the method used to form the joints of

the stator segments. The inductance matrix of the six-phase stator winding used in the demonstration system was measured at a frequency of 25 kHz:

$$L_s = \begin{bmatrix} 66.9 & -13.8 & -14.2 & 22.3 & -44.0 & 6.8 \\ -13.8 & 72.9 & -14.0 & 6.8 & 28.6 & -50.5 \\ -14.2 & -14.0 & 66.8 & -45.0 & 6.6 & 21.9 \\ 22.3 & 6.8 & -45.0 & 66.8 & -13.7 & -14.0 \\ -44.0 & 28.6 & 6.6 & -13.7 & 73.0 & -19.9 \\ 6.8 & -50.5 & 21.9 & -14.0 & -19.9 & 72.7 \end{bmatrix} \mu H$$

The resistance the six stator phases at 25 kHz is 1.8Ω. The envelopes of the voltages induced have the same amplitude, but the variation of the inductance matrix results in induced current envelope amplitudes of different magnitudes as well as offsets in the relative phase of the envelopes. An experiment to measure the effect of the unbalanced inductance matrix on the magnitude and phase of the induced current envelopes was conducted by placing the vehicle and guideway in a milling machine. The milling machine allows the relative position of the vehicle and the guideway to be precisely adjusted, and measurements of the amount of current induced into each stator phase are logged. The current envelopes resulting from measured data are plotted in Figure 3-14. The envelope amplitude, amplitude error, and phase offset of each data channel are tabulated in table 3-1. The envelope amplitude variation is up to 12%, which is much larger than the errors associated with the analog processor board. The phase offset errors are about 2° to 6°.

Table 3-1. Measured amplitude variation and phase offset

Channel #	amplitude	amp. error	phase offset
a	1871.6	12.1%	-2.23°
b	1541.6	-7.7%	3.22°
c	1726.0	3.3%	5.53°
x	1787.4	7.0%	-3.84°
y	1560.1	-6.6%	3.57°
z	1534.1	8.1%	-6.25°

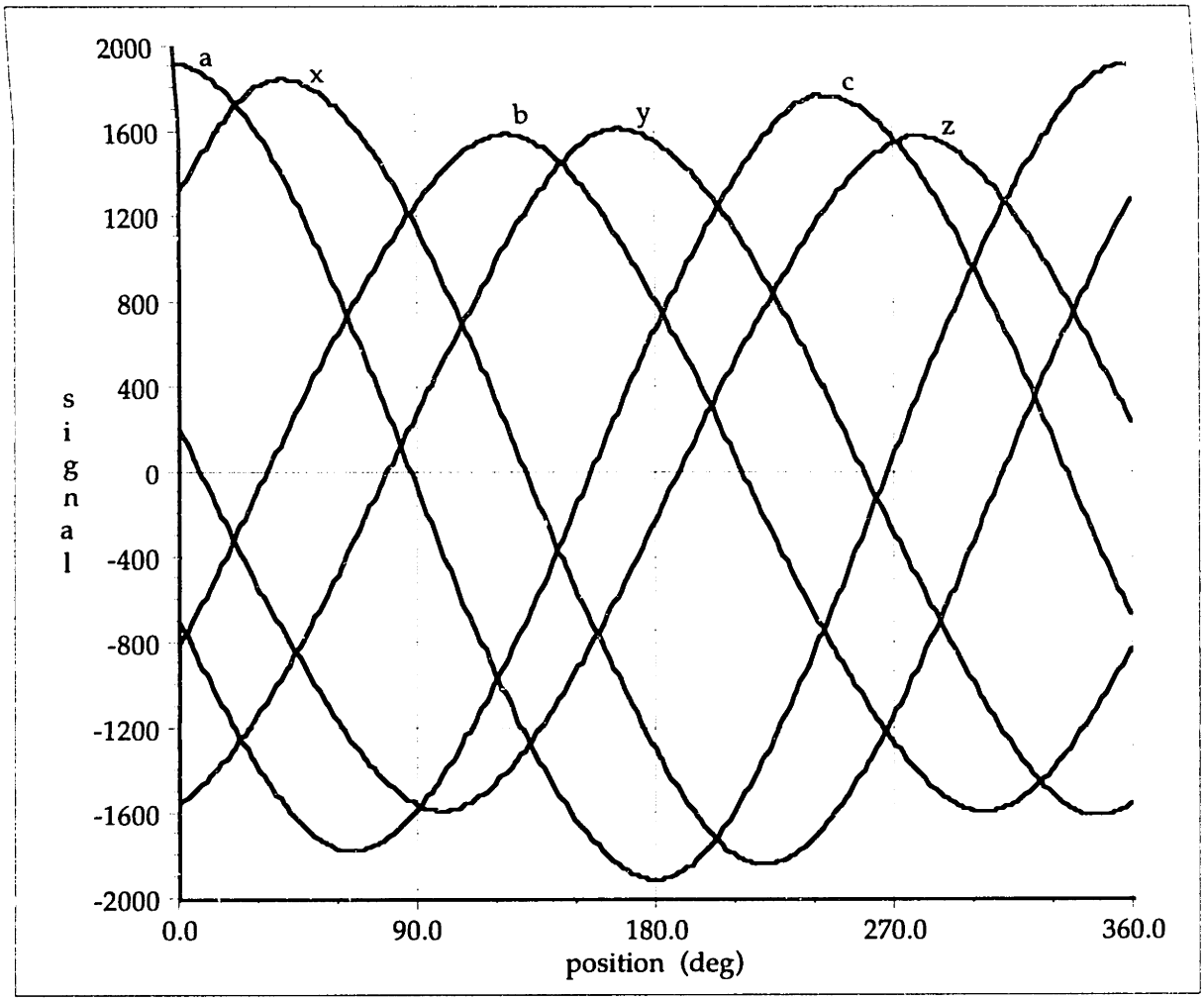


Figure 3-14. Stator current envelope amplitudes vs. position

error effects

The effects of amplitude, dc offset, and phase offset errors on the phase detector are developed in this section. The phase detector is nonlinear, thus the error sources will interact with one another creating higher order cross terms (*i.e.* superposition does not apply to a nonlinear system). However, the errors are small compared to the 'nominal' signals, and the nonlinearity of the phase detector is quite benign for small errors since $\sin \varepsilon \approx \varepsilon$ for small ε. The application of superposition to the phase detector is therefore based on a reasonable engineering approximation, and the effect of each error will be analyzed individually. The phase detector errors are examined around the operating point of equal vehicle and observer position.

dc offset

The presence of dc offsets on the inputs to the phase detector results in a phase detector error of:

$$\varepsilon_{dc\ offset} = -\delta_a \bullet \mathrm{Sin}(\theta) - \delta_b \bullet \mathrm{Sin}(\theta - 2\pi/3) - \delta_c \bullet \mathrm{Sin}(\theta + 2\pi/3).$$

The error term is sinusoidal in θ. Note that the average dc offset cancels. This error term is easily corrected by adding the appropriate dc correction to each input before applying the inputs to the phase detector. In practice, the dc offsets are very small (on the order of 1 or 2 LSBs of the A/D converter) and their effect on position accuracy is not severe. The dc offsets could be measured on line with a background computation and used to provide an adaptive correction of dc offset, but a simpler off line measurement and calibration routine is used in the model system.

amplitude variation

Phase detector input signals of different envelope amplitudes results in a phase detector error of:

$$\varepsilon_{amp\ var} = -0.5\ (\ A \bullet \mathrm{Sin}(2\theta) + B \bullet \mathrm{Sin}(2\theta - 4\pi/3) + C \bullet \mathrm{Sin}(2\theta + 4\pi/3)\ ).$$

The error term is sinusoidal in 2θ, and is zero when A=B=C. This error term is easily corrected by multiplying each input by the appropriate constant before applying the inputs to the phase detector. This error source, which stems largely from the inductance matrix of the stator winding, has a significant impact on the accuracy of the phase detector. An adaptive, on line calibration routine to measure and correct amplitude variations is quite feasible, but a simpler off line calibration routine was used in the model system.

phase offset

The phase offset terms are the most unusual of the signal processing errors. They are a result of unequal or unbalanced self and mutual inductance terms in the stator inductance matrix. The presence of phase error terms in the input signals to the phase detector result in a phase detector error of:

$$\varepsilon_{phase\ offset} = -0.5\ (\ \mathrm{Sin}(2\theta + \psi_a) + \mathrm{Sin}(2\theta - 4\pi/3 + \psi_b) + \mathrm{Sin}(2\theta + 4\pi/3 + \psi_c)\ )$$
$$-0.5\ (\ \mathrm{Sin}(\psi_a) + \mathrm{Sin}(\psi_b) + \mathrm{Sin}(\psi_c)\ ).$$

The error contains two terms; one is sinusoidal in 2θ and the other is stationary. These error terms are not as simple to correct. Phase shifting the inputs before applying them to the phase detector is an easy concept but it is difficult to implement and is computationally intensive. The approach used in the model system is to phase shift the weighting functions by an amount that cancels the error terms. This approach is simple to implement; the appropriate phase offset is added to the index used in the table look-up. The correcting offset added to each of the weighting functions is:

$$\psi_{wa} = (-\psi_a + 2\psi_b + 2\psi_c)/3$$
$$\psi_{wb} = (2\psi_a - \psi_b + 2\psi_c)/3$$
$$\psi_{wc} = (2\psi_a + 2\psi_b - \psi_c)/3$$

The phase detector error measured as a function of vehicle position in the milling machine experiment is plotted in Figure 3-15. The 'raw error' plot results from the direct application of the input data to the phase detector, and the 'normalized error' plot shows the improvement due to normalization of the input data to correct for amplitude variations, dc offsets, and phase offsets. The rms magnitude of the raw error is approximately 2°. This error slightly exceeds the accuracy desired to effectively synchronize the LSM stator current pattern to the vehicle movement. The rms magnitude of the normalized error is 0.1°, which corresponds to a displacement of approximately 25 μm (0.001"). The magnitude of the position sensor error is therefore improved by a factor of 20 with the application of the normalization routines outlined in the preceding section. The raw error is dominated by terms sinusoidal in 2θ, thus the amplitude variation and phase offset terms are the most detrimental to the phase detector performance. The normalized error is dominated by a term sinusoidal in 6θ. The lack of lower frequency error terms in the normalized error plot is a good indication of the validity of the error models outlined in the preceding section. The residual error term sinusoidal in 6θ is the result of 5th and 7th harmonics present on the input data. The effects of input data harmonics on the phase detector performance are detailed in the following section. It will be shown that the next most prominent error source is indeed due to the presence of a 5th harmonic component in the incoming data, thus the measured performance of the position sensing system is in excellent agreement with the predicted performance.

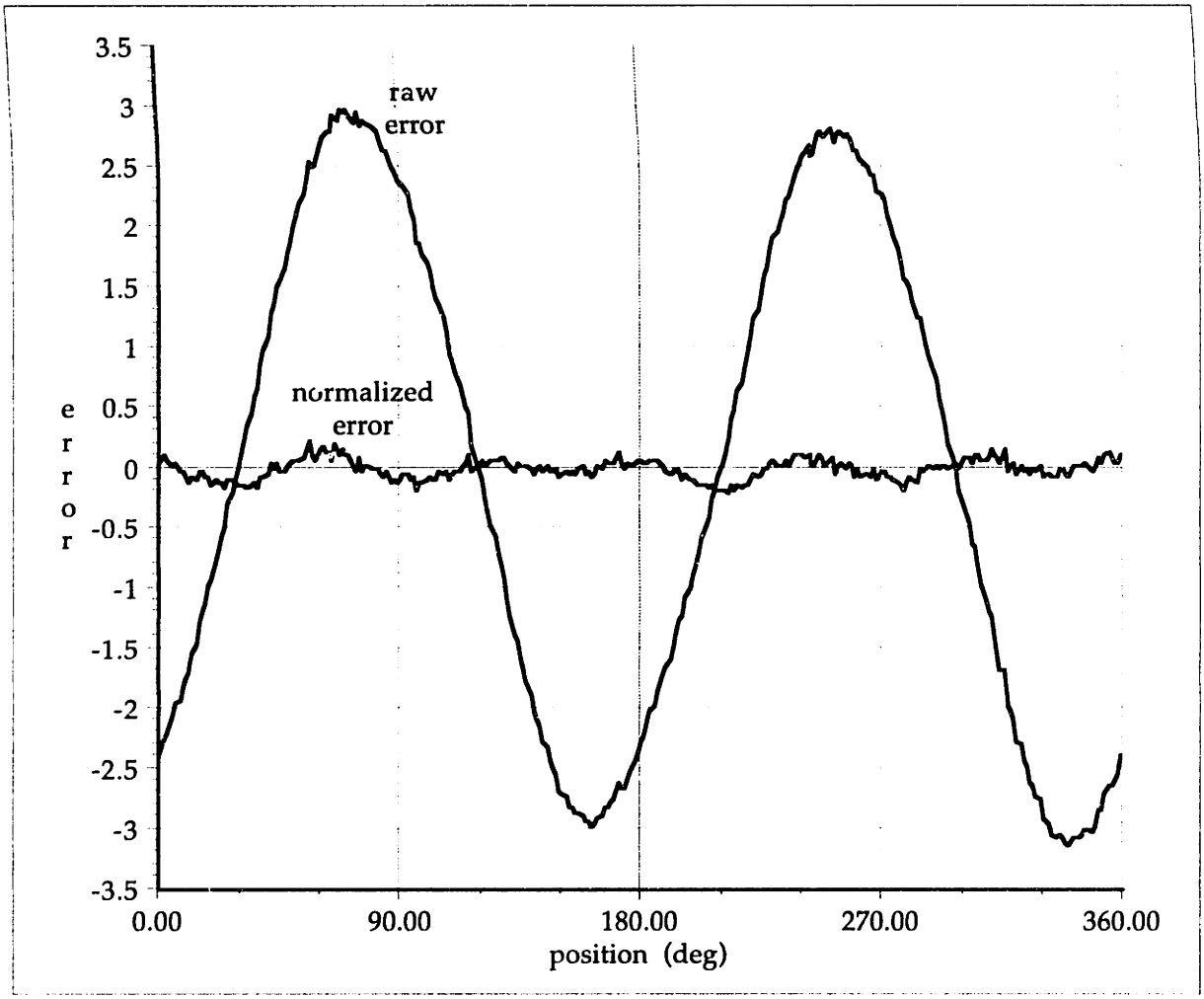


Figure 3-15. Phase detector error vs. position

harmonics

The position sensor signals induced in the stator winding have envelopes described by the periodic function $\mathcal{F}$. In an ideal system the function $\mathcal{F}$ contains only a fundamental component, thus the input signals to the phase detector contain no harmonic terms. The waveshape of the function $\mathcal{F}$ is determined by the shape of the vehicle transducer poles, the shape of the stator turns and the proximity of the transducer to the stator. In any actual implementation there will be some harmonic content in the function $\mathcal{F}$, and the input signals to the phase detector will be of the form:

$$a = \sum_n a_n \cdot \text{Cos } n(\theta)$$

$$b = \sum_n a_n \cdot \text{Cos } n(\theta - 2\pi/3)$$

$$c = \sum_n a_n \cdot \text{Cos } n(\theta + 2\pi/3)$$

where n is the harmonic number and a_n are the amplitudes of each harmonic component. (Note that the engineering approximation allowing the concept of superposition to be applied to this nonlinear system for the purpose of studying the effect of small perturbations is once again invoked. All other error sources are assumed to be absent in the discussion of harmonic effects.) Because of symmetry, even harmonics will not be introduced in the stator signals so n = 1, 3, 5, ... For the vehicle position θ and observer position ϕ, the output of the phase detector is:

$$\sum_n a_n \cdot \cos n(\theta) \cdot \sin(\phi)$$
$$+\sum_n a_n \cdot \cos n(\theta - 2\pi/3) \cdot \sin(\phi - 2\pi/3)$$
$$+\sum_n a_n \cdot \cos n(\theta + 2\pi/3) \cdot \sin(\phi + 2\pi/3)$$

Near the equilibrium operating point $\theta = \phi$, the error introduced by the harmonic content is:

$$\varepsilon = 0.5 \cdot \sum_n a_n \left[\sin((1+n)\theta) + \sin((1+n)(\theta - 2\pi/3)) + \sin((1+n)(\theta + 2\pi/3)) \right.$$
$$\left. + \sin((1-n)\theta) + \sin((1-n)(\theta - 2\pi/3)) + \sin((1-n)(\theta + 2\pi/3)) \right]$$

The first three terms in the summation cancel for any n such that n+1 is not a multiple of three, and the last three terms cancel for any n such that n-1 is not a multiple of three. The terms that remain are:

$$\varepsilon = 0.5 \cdot \sum_n a_n \left[\sin((1 \pm n)\theta) + \sin((1 \pm n)(\theta - 2\pi/3)) + \sin((1 \pm n)(\theta + 2\pi/3)) \right]$$

where n = 5, 7, 11, 13, ... The '+' sign applies for n = 5, 11, 17, ... and the '-' sign for n = 7, 13, 19, ... The remaining error is therefore a summation of terms sinusoidal in 6θ, 12θ, 18θ, ... The normalized error plotted in Figure 3-15 clearly shows the term sinusoidal in 6θ resulting from 5th and 7th harmonics on the phase detector inputs. The amplitudes of the harmonics measured in the milling machine experiment are tabulated in table 3-2. The 5th harmonic represents the dominant harmonic error source since the presence of a third harmonic has no effect on the phase detector error.

Table 3-2. Amplitude of harmonics at phase detector inputs

harmonic #:	1	3	5	7	9
amplitude:	1	0.0243	0.0065	0.0009	0.0010

4. Wayside Electronics Design

analog processor board

The analog processor board receives, filters, amplifies, and demodulates the position-dependent signals injected into the stator winding by the inductive transducer mounted on the vehicle. A block diagram of the analog processor board is shown in Figure 4-1. The board has six identical signal processing channels, one for each stator winding phase. The output of each channel provides position sensor data to an A/D converter located on the microcontroller board. A phase-locked loop is used to reconstruct the carrier associated with the position-dependent stator signals. This reconstructed carrier is used to generate all of the timing signals required to control the demodulator and 3-mode integrator circuitry, thus accommodating synchronous detection of the position signals. The operation and construction of each block of circuitry shown in Figure 4-1 are described in the following sections.

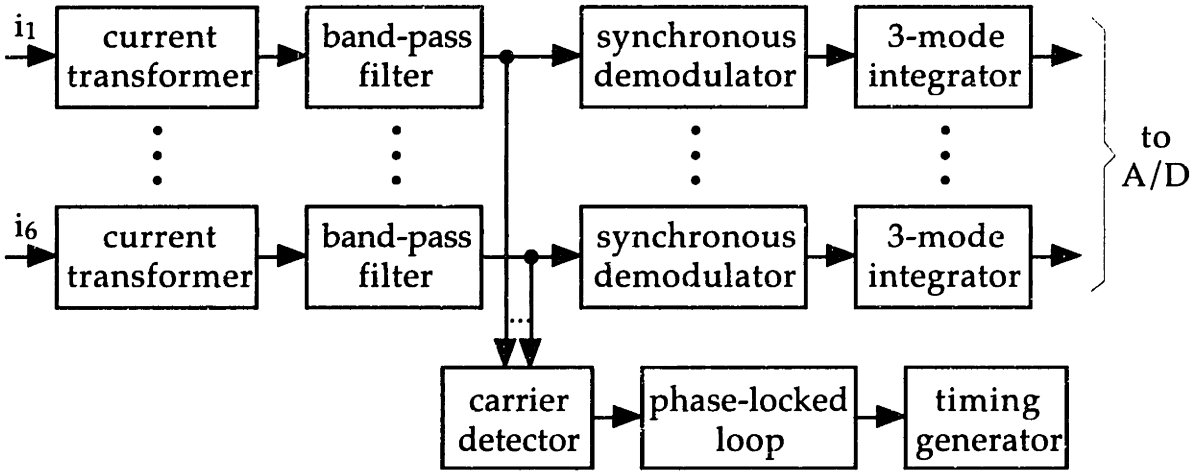


Figure 4-1. Block diagram of analog processor board

current transformer

The current transformers respond to the position-dependent currents injected into the stator winding by the inductive transducer mounted on board the vehicle, while rejecting the propulsion currents driven through the stator winding by the inverter. Both the orientation and the frequency content of the stator currents are used to distinguish the desired position-dependent currents from the propulsion currents. The current transformer also provides electrical isolation between the power electronics and the

processing circuitry, thus protecting the low level processing circuitry in the event of a fault in the power electronics or motor.

Figure 4-2 shows the schematic diagram of a current transformer (with the band-pass filter circuitry). The secondary of the current transformer consists of 88 turns of #30 AWG magnet wire wound on a Phillips 266T125-3E2A ferrite toroid; the two 1 turn primaries are formed by passing the connections from the inverter to the stator through the toroid. The primary windings are oriented such that propulsion currents flowing from the inverter, through the transformer, and to the stator result in opposing magnetomotive forces in the toroid; while the circulating position sensor currents flowing from the stator, through each primary, and back to the stator result in aiding magnetomotive forces in the toroid. This orientation allows the current transformer to reject propulsion currents and pass position sensor currents. The effectiveness of this technique is demonstrated in Figure 4-3. The top trace of this scope photo shows the current through one of the current transformer primaries, which is dominated by the propulsion current. The propulsion current through each leg of the winding has a peak magnitude of about 4 Amps, thus the peak propulsion current is about

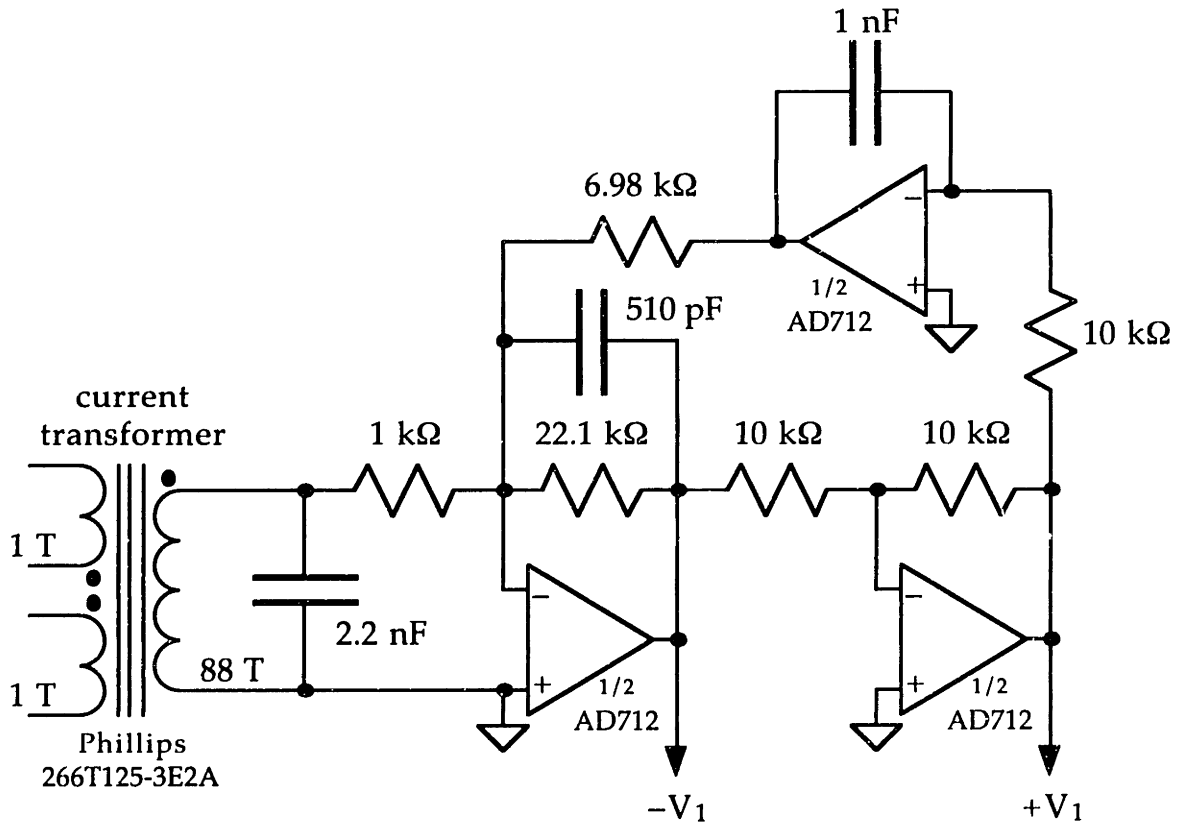


Figure 4-2. Current transformer and band-pass filter circuitry

8 Amps. The center trace shows the difference of the primary currents (*i.e.* the mmf driving the current transformer). The peak magnitude of the propulsion current component has been reduced to approximately 40 mAmps. Thus the propulsion current is reduced by a factor of about 200, while the magnitude of the position sensor current is unchanged. The bottom trace shows the output of the current transformer. The bandpass filter characteristics of the current transformer have effectively rejected the low frequency propulsion current signal. The period of the envelope of the position sensor signal is twice the period of the propulsion current due to the double pole pitch used for the vehicle transducer. (Note that the sampling operation of the digital oscilloscope has affected the appearance of the 25.6 kHz position sensor waveform. The magnitude of the envelope of the position sensor signal is displayed correctly if aliasing effects are ignored.)

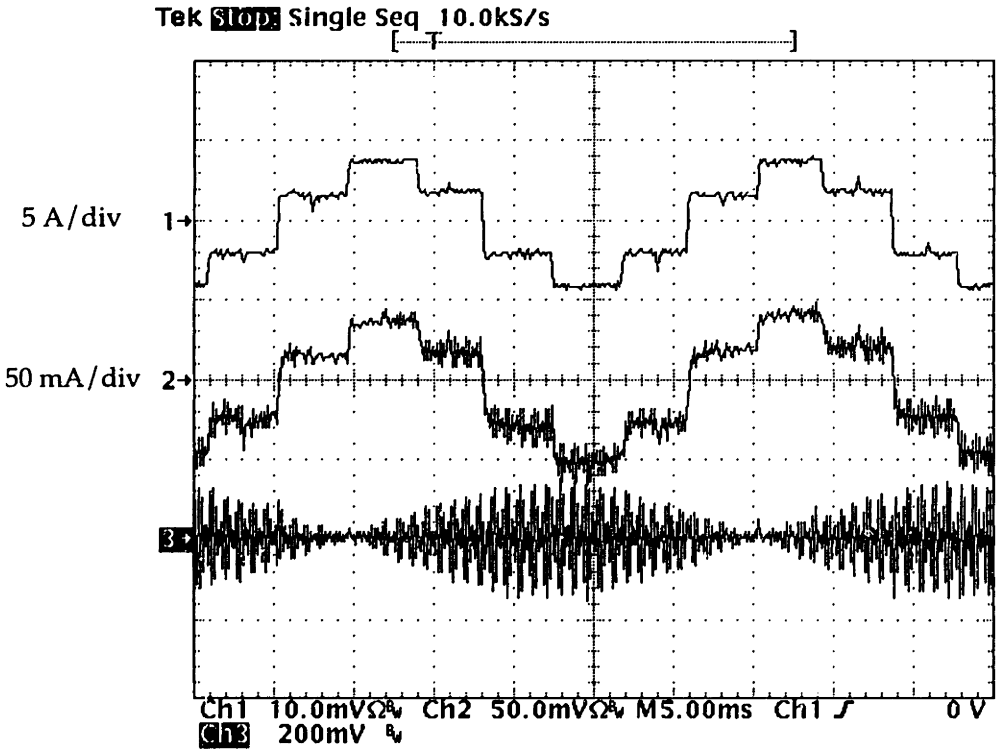


Figure 4-3. Cancellation of propulsion currents

The magnetizing inductance of the transformer viewed from the secondary is approximately 17.5 mH. A 2.2 nF capacitor is placed in parallel with the secondary winding of the current transformer to result in a pair of band-pass poles centered at the 25.6 kHz carrier frequency of the position-dependent signals in the stator. The 1 kΩ input impedance of the band-pass filter strongly damps the current transformer poles; the Q of the current transformer poles is approximately 1/8. This low Q value permits variations

in the magnetizing inductance of the current transformer without significantly altering the phase response of the band-pass filter poles associated with the current transformer. The filter poles allow signals near the expected carrier frequency of the position sensor signals to pass, while other signals and noise are rejected. The bottom trace of the scope photo in Figure 4-3 shows the filtered output of the current transformer. Note that the propulsion current has been effectively eliminated.

band-pass filter

The band-pass filters amplify and filter the position sensor signals received by the current transformers, and provide a differential pair of outputs to the synchronous demodulator. A schematic diagram of a band-pass filter is shown with the current transformer circuitry in Figure 4-2. Three AD712 precision op-amps are used to implement each filter. One op-amp is configured as a unity gain inverter to provide a differential pair of outputs from the filter. The other two op-amps are configured to provide the appropriate gain and filter dynamics. The natural frequency of the filter poles is 25.6 kHz, and the Q of the filter is 2. (The 510 pF capacitor is a bit smaller than the 554 pF value that results from calculations based on ideal op-amps, to compensate for the pole shift caused by the AD712 dynamics.) The gain of

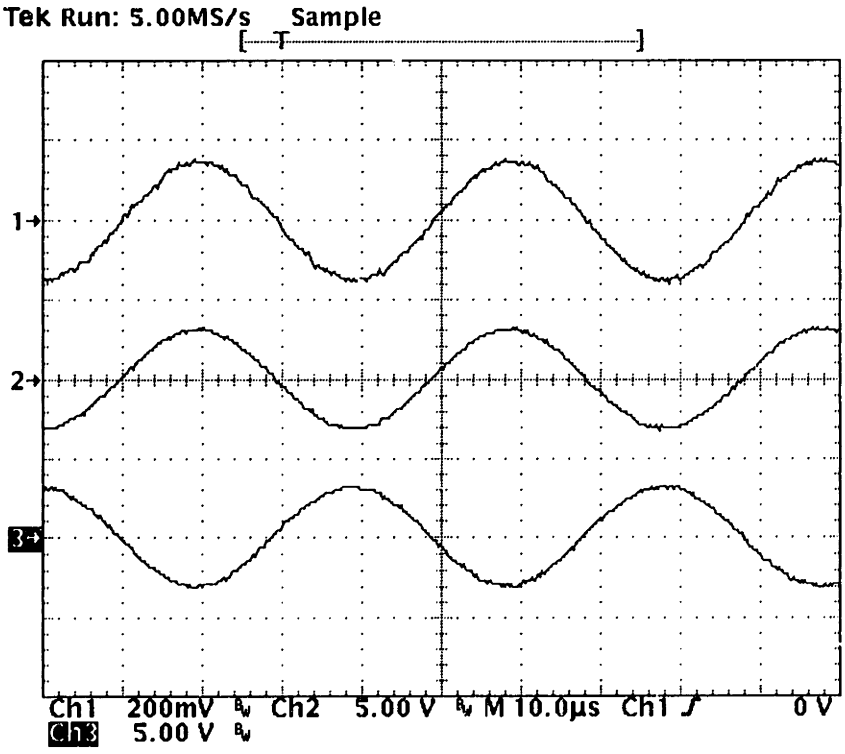


Figure 4-4. Band-pass filter signals

the filter at its center frequency is 22.1. The top trace of the scope photo in Figure 4-4 is the input to the bandpass filter from the current transformer. The center trace shows the positive output and the bottom trace shows the negative output of the bandpass filter.

synchronous demodulator

The synchronous demodulators extract the modulating signal (envelope) from the position sensor signals. The timing signals used to drive the demodulator are synchronized to the carrier; the phase-locked loop and timing generator circuitry are described in a later section. A schematic diagram of a synchronous demodulator is shown with the 3-mode integrator circuitry in Figure 4-5. A pair of AD7510 analog switches are used to implement the synchronous demodulator function. When the angle of the sinusoidal carrier is in the range $0 \leq \text{angle} < \pi$, the negative differential output of the band-pass filter is connected to the 3-mode integrator. (–S is on, +S is off.) The incoming position signal is multiplied by –1 during this interval. When the angle of the carrier is in the range $\pi \leq \text{angle} < 2\pi$, the positive differential output of the band-pass filter is connected to the 3-mode integrator; the incoming position sensor signal is multiplied by +1 during this interval. Thus a position signal in phase with the carrier results in a positive accumulation on the integrator, and a position signal out of phase with the carrier results in a negative accumulation on the integrator. (Note that the integrator is inverting.) The rate of accumulation on the integrator is proportional to the magnitude of the position signal. The top trace of the scope photo in Figure 4-6 shows the output of the synchronous demodulator for a position sensor signal in phase with the carrier.

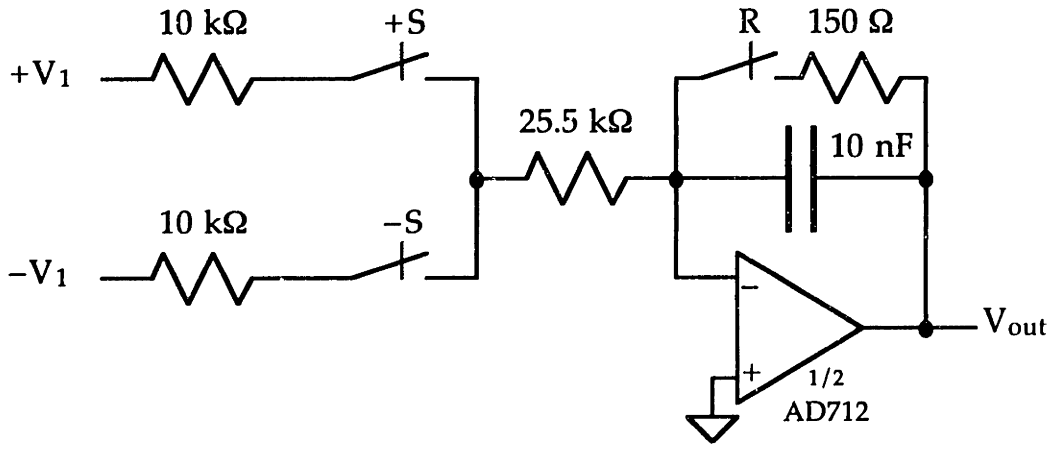


Figure 4-5. Synchronous demodulator and 3-mode integrator circuitry

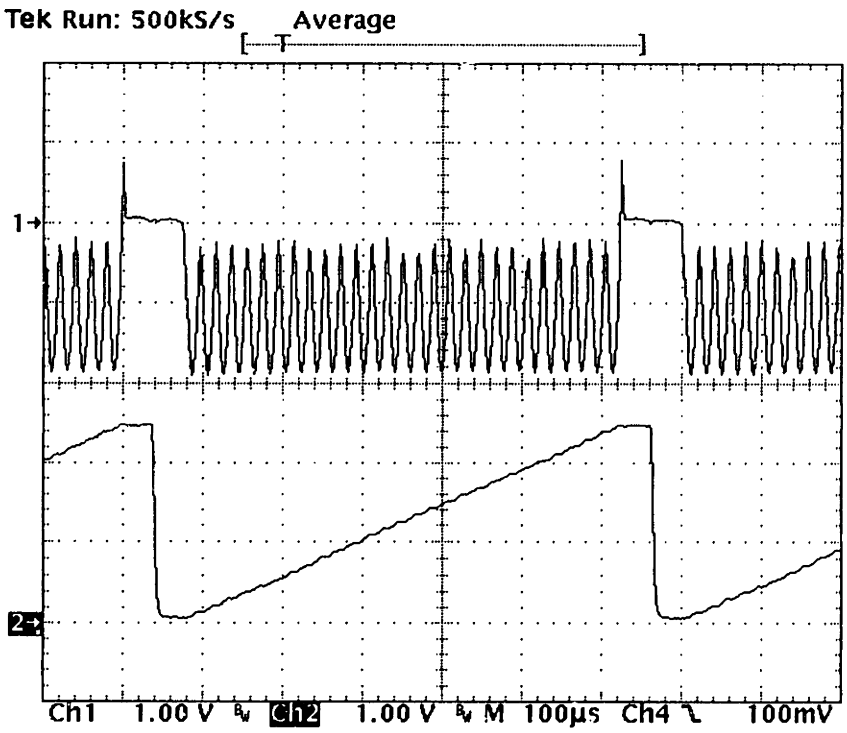


Figure 4-6. Synchronous demodulator and 3-mode integrator signals

3-mode integrator

The 3-mode integrators accumulate position signals over a number of carrier cycles, hold the signal during A/D conversion, and then reset before starting another accumulation. The timing signals used to drive the 3-mode integrator are synchronized to the carrier; the phase-locked loop and timing generator circuitry are described in a later section. Figure 4-5 shows the schematic diagram of a 3-mode integrator (with the synchronous demodulator circuitry). The AD712 precision op-amp is configured as an inverting integrator, with the gain constant set to provide a 10 Volt output signal with an integrator input of -10 Volts (peak), averaged over a time period of 14 carrier cycles, or 547 µsec. A switch and current limiting resistor are placed across the integrating capacitor to allow the integrator output to be reset to zero. The reset time constant is chosen to provide adequate accuracy with a reset interval as short as 18 µsec. (The nominal reset interval is one carrier cycle, or 39.1 µsec.) When all three switches are off the integrator is in the hold mode, which is used to keep the input to the A/D converter constant during the conversion process. The bottom trace of the scope photo in Figure 4-6 shows the output signal from the 3-mode integrator for a position sensor signal in phase with the carrier.

carrier detector

The carrier detector combines the signals from each of the 6 band-pass filters and extracts a square wave at twice the carrier frequency. Figure 4-8 shows a schematic diagram of the carrier detector circuitry. The input diodes pass the absolute value of the signal from each band-pass filter to a summing amplifier. An absolute value or squaring operation is necessary before summing the signals since they are of different phases. The straight sum of the signals is zero because they are arranged as two balanced 3-phase sets. A nonlinearity is required to reconstruct a carrier from the suppressed carrier position sensor signals [25]. An absolute value function is used since it is easier to implement than a squaring function. The absolute value operation doubles the frequency of the fundamental ac component of the combined signal from the summing amplifier.

The comparator produces a square wave with a frequency equal to the fundamental ac component of the combined signal, which is twice the carrier frequency (nominally 2 x 25.6 kHz or 51.2 kHz). The integrator feeding back to the input of the comparator forces the square wave to have a duty cycle of 50%, which is necessary since the phase-locked loop utilizes an XOR phase detector. The top trace of the scope photo in Figure 4-7 shows the sum of the absolute value of the input signals, the bottom trace shows the output of the carrier detector circuitry.

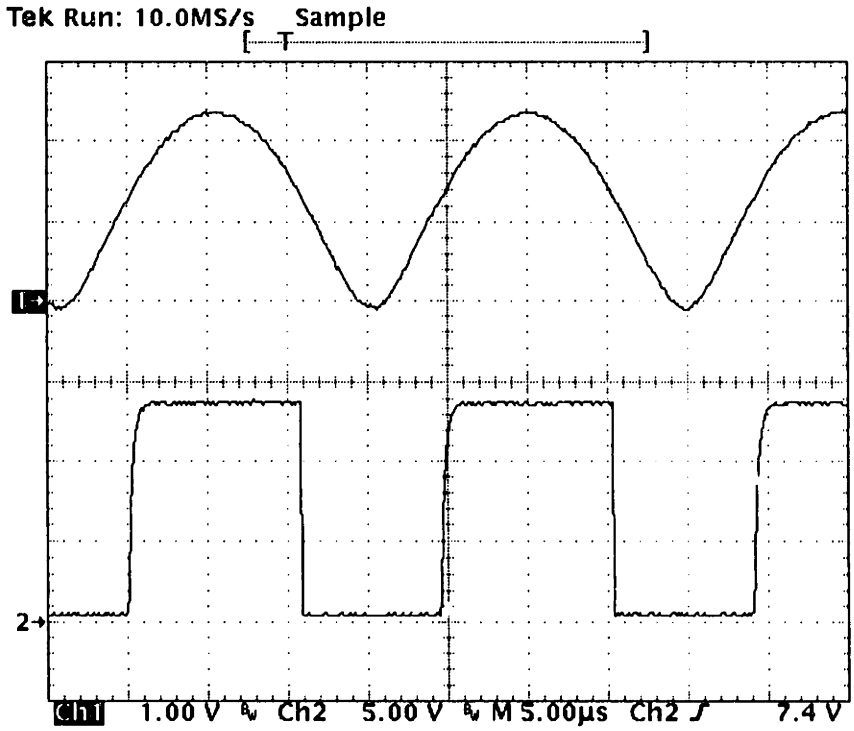


Figure 4-7. Carrier detector signals

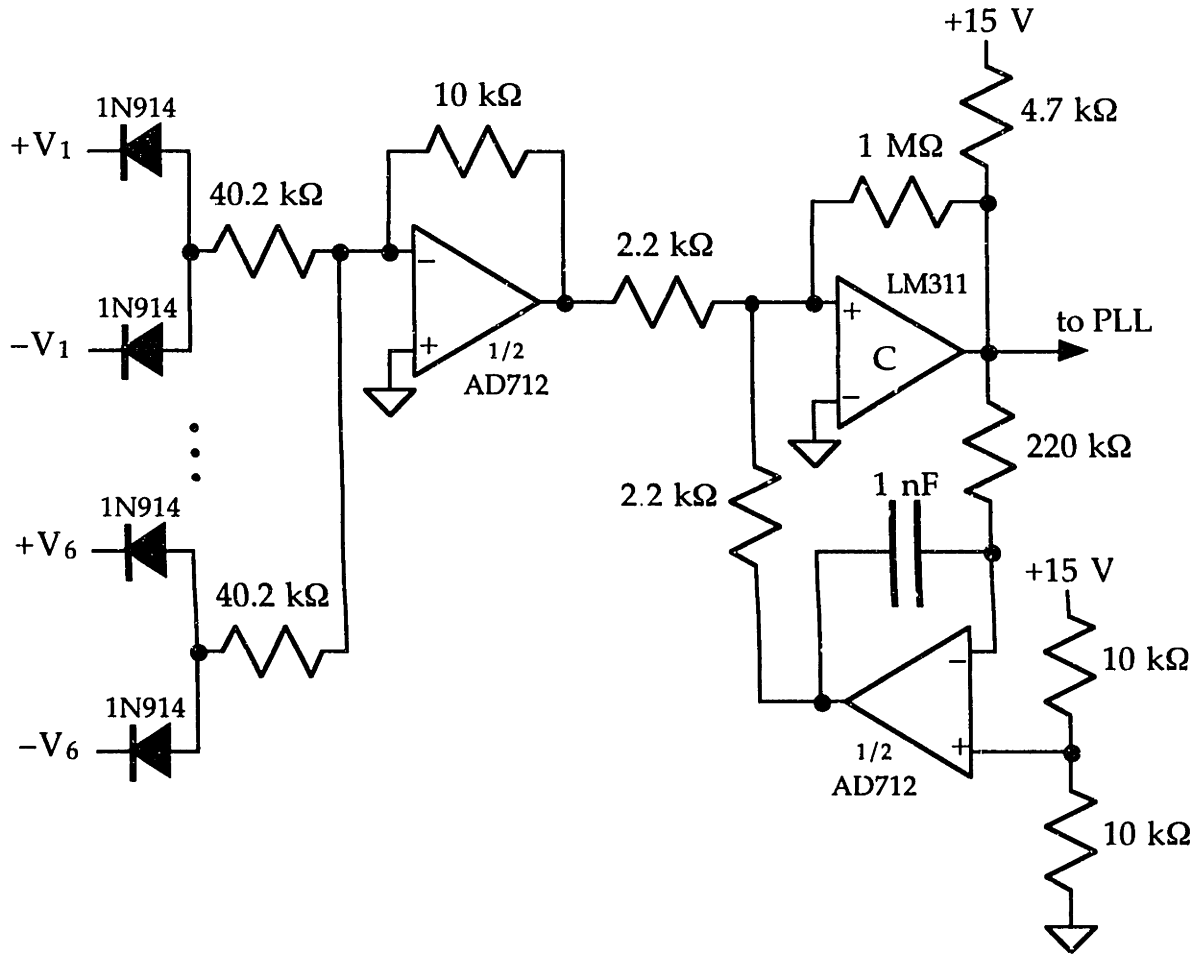


Figure 4-8. Carrier detector circuitry

phase-locked loop

The phase-locked loop (PLL) circuitry locks an oscillator to the square wave from the carrier detector, thus filtering the effects of noise (phase jitter) or glitches (spurious edges) from the detected carrier. The filtering action of the PLL is essential since the incoming position sensor signals contain noise and a significant amount of glitch energy. The PLL is an implementation of an identity observer and provides filtering of the detected carrier signal without introducing any phase shift. Large glitches are induced in the stator when the power inverter commutates a motor phase. The PLL tracks the fundamental component of the detected carrier signal and rejects momentary glitches. A block diagram of the PLL circuitry is shown in Figure 4-9.

Note that there is a phase ambiguity inherent in reconstructing the carrier from suppressed carrier signals [25]. The phase of the reconstructed carrier may be either at 0° or 180° relative to the original carrier with equal probability. As a result of this ambiguity the electrical position measured by the observer may be correct or it may be off by 180°. However, the position

sensor cycle has a period twice as long as the motor (propulsion) period. Thus the ambiguity of the motor position is either 0 or 1 period. An ambiguity of an integer number of motor periods has no effect on motor operation (and is corrected when a mechanical position reference transition is encountered).

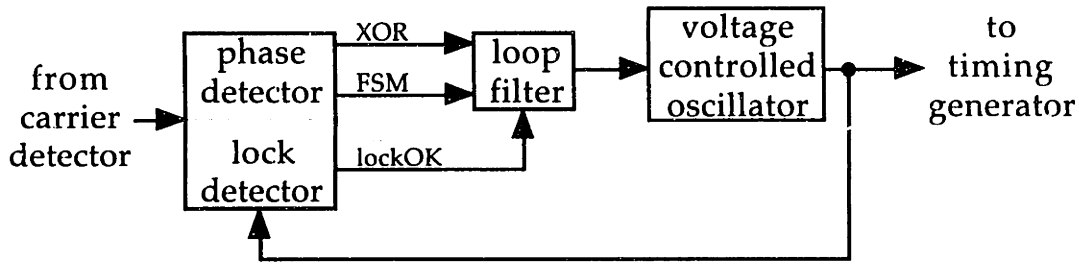


Figure 4-9. Block diagram of phase-locked loop circuitry

The phase-locked loop utilizes two phase detectors; a finite state machine (FSM) phase detector is used when the loop is in a lock acquisition mode, and an exclusive-or (XOR) phase detector is used when the loop has obtained a lock. Both phase detectors are included in the 4046 phase-locked loop IC. The FSM phase detector responds to both phase and frequency differences, which allows it to perform well in an unlocked loop condition. However, the FSM phase detector is edge-sensitive and therefore not effective at rejecting spurious edges in the detected carrier signal. The XOR phase detector is level-sensitive and thus rejects spurious edges well, but it cannot detect a frequency difference. The best features of both phase detectors are combined by using a lock detector to allow the loop filter to select either the FSM or the XOR phase detector output.

The voltage controlled oscillator (VCO) function is implemented in the 4046 IC. Resistor and capacitor values are chosen to give the VCO an operating range that extends 5 kHz above and below a center frequency of 51.2 kHz.

The lock detector function is implemented with a D flip-flop and a filter/comparator, and is shown in schematic form in Figure 4-10. The flip-flop, which physically resides in the PAL used in the timing generator circuitry, is clocked by the VCO output and has the detected carrier as its D input. When the loop is locked, the flip-flop output is low with a probability of nearly 100%. A glitch in the detected carrier signal at the same instant as a rising edge on the VCO output can cause the flip-flop output to go into the high state for one VCO cycle. This type of glitch event occurs infrequently, but can be induced by commutation of the power inverter. When the loop is unlocked, the flip-flop output can be in either state with

equal probability. The flip-flop output toggles between the two states, with an average value of 1/2. The filter/comparator circuit rejects occasional glitches, selects the XOR phase detector if the flip-flop output is low most of the time (a locked loop condition), and selects the FSM phase detector if the flip-flop output is low only half of the time (an unlocked loop condition).

Figure 4-11 shows the transient associated with the PLL acquiring and locking to the detected carrier. The top trace of the scope photo is the VCO input and the bottom trace is the lock detector output. For most of the acquisition transient the PLL is operating with the FSM phase detector. The FSM phase detector responds to the large frequency difference between the incoming carrier and the VCO output by driving the loop filter with a large error signal. The VCO frequency converges rapidly toward the input frequency. Cycle slips of the PLL due to the frequency mismatch between the input signal and the VCO output can be seen as ripple on the VCO input signal. When the VCO output frequency nears the input frequency the FSM phase detector causes the loop to lock the phase of the VCO output to the input carrier. Once phase lock is obtained the lock detector switches to the XOR phase detector and the PLL tracks the phase of the input signal. A delay of only 4 msec is needed to acquire lock, and an additional 2 msec is required for the phase of the VCO output to settle into quadrature with the phase of the detected carrier.

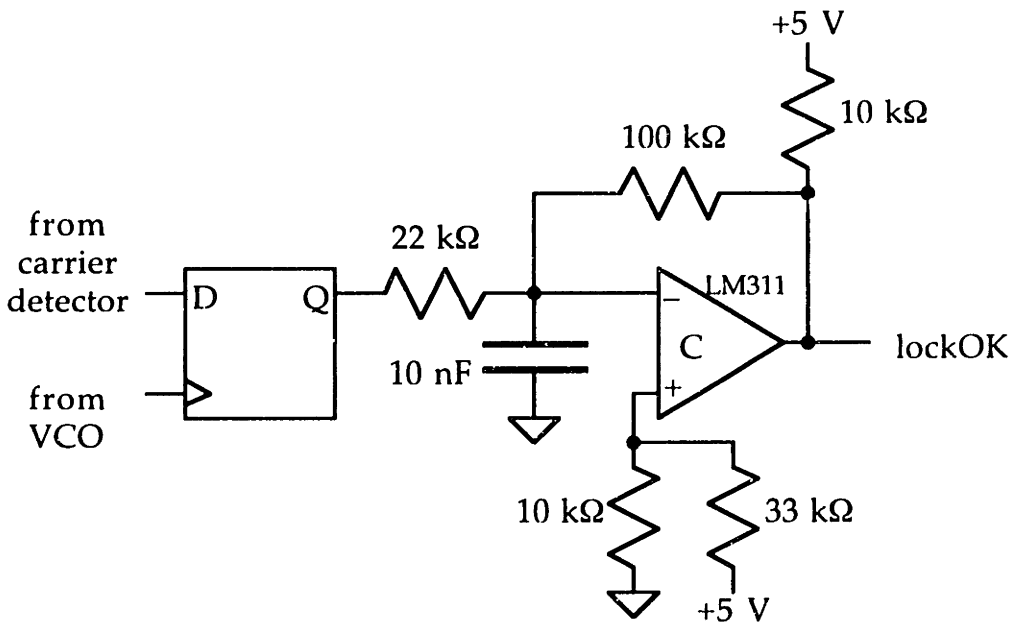


Figure 4-10. PLL lock detector circuitry

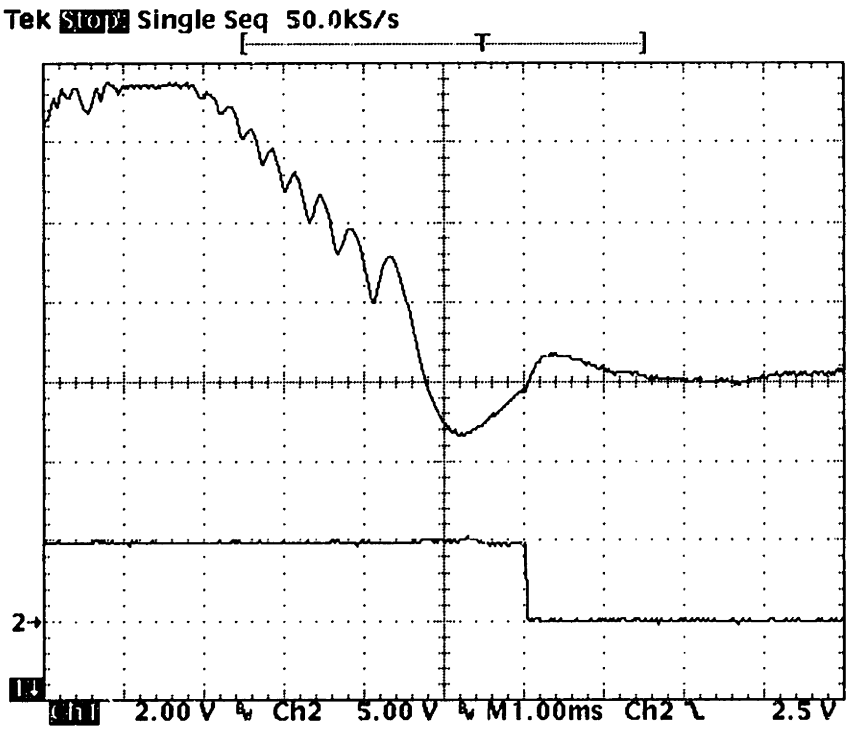


Figure 4-11. PLL acquisition transient

The loop filter circuitry used to stabilize the dynamics of the phase-locked loop is shown schematically in Figure 4-12. An AD712 precision op-amp is configured as an inverting integrator with a lead network. When the loop is locked, the XOR phase detector provides an input signal to the loop filter. The integration allows the VCO output to track the detected carrier with zero steady-state phase error (the VCO output leads the detected carrier by 90°). The gain constant of the integrator is chosen to force loop crossover at approximately 2000 rad/sec; the lead network provides a positive phase shift near this frequency to ensure an adequate phase margin value of about 47°. When the loop is not locked, the FSM phase detector signal is inverted and drives the loop filter input. In this configuration, the loop crossover frequency is approximately 4000 rad/sec, with a phase margin of about 33°. A pair of high frequency poles are placed beyond the loop crossover frequency to help filter the high frequency harmonics from the phase detector. The inverting amplifier is required with the FSM phase detector to keep the overall loop gain negative. The potentiometer can be adjusted to trim the steady-state phase difference between the VCO output and the detected carrier.

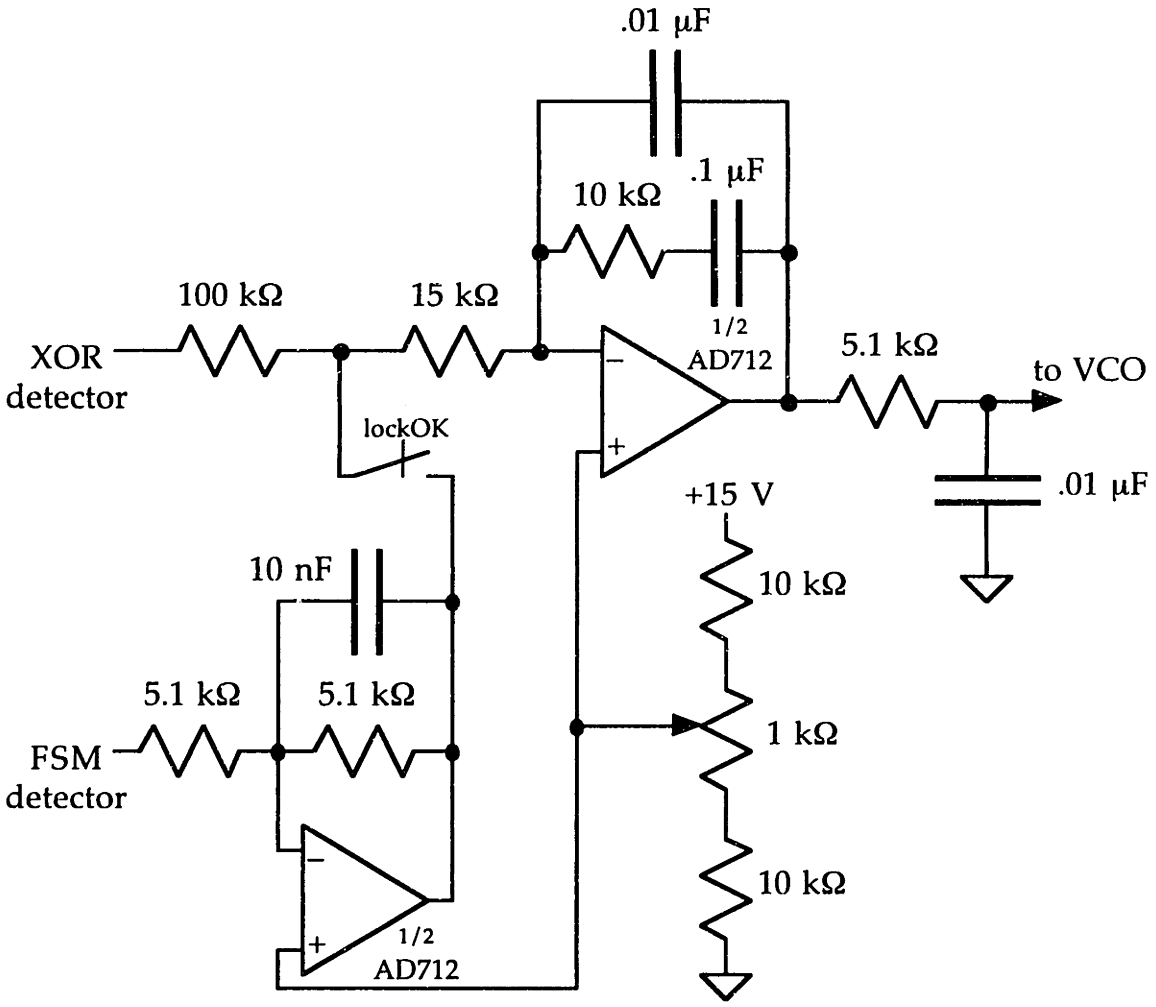


Figure 4-12. PLL loop filter circuitry

timing generator

All timing signals used on the analog processor board are derived from the VCO output of the PLL, which is synchronized to the carrier in the incoming position signals. A 32-state finite state machine is used to generate the timing signals used to control the switches in the synchronous demodulators and the 3-mode integrators and to signal the microcontroller that a data set is ready for A/D conversion. The timing generator circuitry consists of a 4-bit counter and a programmable array logic (PAL) IC. The 4-bit counter and one PAL register are used to count through the required 32 states. The remaining PAL logic is used to compute the appropriate logic level for each of the timing signals in each of the 32 states. (Actually, one PAL register is utilized to implement the D flip-flop used in the lock detector circuitry.) A state diagram for the timing generator is shown in Figure 4-13. Each timing state is one-half of a carrier cycle in duration (nominally 19.53 μsec). Each

processing channel operates on a 16 carrier-cycle interval (nominally 625 μsec); one cycle for integrator reset, 14 cycles for data accumulation, and one cycle to hold the data for A/D conversion. The timing signals for the 'even' motor phases (phases a, b, and c) and the 'odd' motor phases (phases x, y, and z) are interleaved; the microcontroller receives data from a three-phase set every 8 carrier cycles (nominally 312 μsec).

A falling edge on Q_E signals the microcontroller that data from the three even motor phases is ready; a rising edge indicates that data is ready on the odd motor phases. A logic high on R_{abc} puts the 3-mode integrators for the even phases in the reset mode, R_{xyz} resets the 3-mode integrators for the odd phases. $+S_{abc}$ and $-S_{abc}$ control the synchronous demodulator switches for the even phases, $+S_{xyz}$ and $-S_{xyz}$ control the synchronous demodulator switches for the odd phases. A 3-mode integrator is in the hold mode if the R, +S, and –S switches are all off.

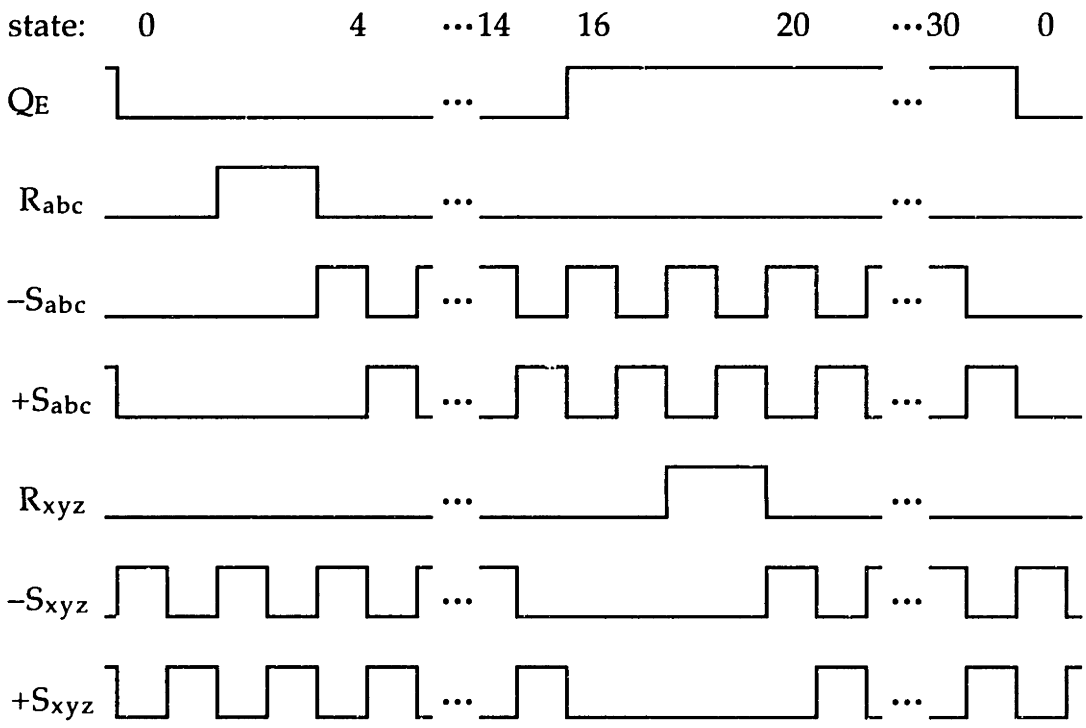


Figure 4-13. State diagram for timing generator

error sources

Since the components used to build the analog processor board are not ideal, errors are introduced into the processed signals. These errors may be a result of component value variation due to manufacturing tolerances or of non-ideal component behavior such as op-amp offset voltage or bias current. A variety of error sources and their effect on the processed signals are outlined in the following section. It is most important to note that all error sources can be combined as an equivalent offset and an equivalent gain error for each processing channel.

Phase errors are introduced in the current transformers and bandpass filter stages, primarily due to component value tolerance. The capacitors used to set the center frequency of the filter dynamics have a tolerance of 5%, and geometry and permeability variations in the ferrite cores of the current transformers result in magnetizing inductance value tolerances of 20%. The phase error introduced by the current transformer could be as high as 3.8°, and the phase error introduced by the bandpass filter could be as high as 5.6°. A phase error will reduce the effective gain of the synchronous demodulator by a factor of $\text{Cos}(\text{ø}_{\text{error}})$, thus the gain variations resulting from the phase errors are quite small; approximately 0.25% and 0.5%.

DC offsets introduced by two of the op-amps in the bandpass filter are rejected by the synchronous demodulator. However, the dc offset introduced by the third BPF op-amp (the op-amp used to implement the inverting gain of one stage) pass through the synchronous demodulator. Similarly, the dc offset introduced by the op-amp used to implement the three-mode integrator is passed through to the output. The offset voltage of the AD712 op-amps used for these stages is quite small, approximately 0.5 mV.

The charge injection resulting from opening the reset switch in the three-mode integrator results in a dc offset. The AD7510 switch used to reset the integrator can have a charge injection of as much as 30 pC, which would result in an offset voltage on the 10 nF integrator capacitor of 3 mV. The synchronous demodulator switches also inject charge when opened and closed. Their effect is minimized since one switch closes as the other opens, however, these switches are cycled many times during a signal acquisition. The actual dc offset introduced by charge injection from the synchronous demodulator switches is difficult to analyze.

Component value tolerances of the resistors and capacitors used throughout the analog processing circuitry directly introduce gain variations. The resistors used in critical areas have a 1% component value tolerance, the capacitors used in critical areas have a 5% component value tolerance.

The gain variations and dc offsets of the six processing channels on one of the analog processor boards were measured and are tabulated in table 4-1. Note that the A/D pre-filter circuitry and the A/D converters on the microcontroller board will introduce additional gain and dc offset errors. The 1% component value tolerance resistors used in the A/D pre-filter circuitry will have little effect on the overall gain variation, but will introduce dc offsets that dominate the overall offset error.

Table 4-1. DC offset and gain variation of analog processor board

channel #	gain	gain error	dc offset
a	22.82	1.90%	4.38 mV
b	22.04	-1.60%	3.13 mV
c	22.50	0.45%	-2.50 mV
x	22.82	1.90%	1.25 mV
y	21.88	-2.30%	9.38 mV
z	22.34	-0.26%	10.0 mV

microcontroller board

The microcontroller board contains the Motorola HC16 processor that is used to implement the zone controller functions. The microcontroller board scales and digitizes the position sensor signals received from the analog processor board and the chopper current measurement received from the power electronics module. A nonlinear observer tracks the position signals, thus providing estimates of the vehicle position and velocity. These estimates are compared with the desired vehicle profile, and the propulsion controller computes the motor current necessary to propel the vehicle along the desired trajectory. The chopper current measurement, the computed motor current, and the estimated vehicle position are compared and combined in the power electronics controller to compute the appropriate states for all of the switches in the power electronics module. A block diagram of these control functions, which are implemented digitally in the HC16 microcontroller, is shown in Figure 4-14.

The microcontroller board also contains a dual universal asynchronous receiver and transmitter (DUART) and a pair of optocouplers, which are used to provide a serial communication link to the preceding and to the succeeding zone controllers. These communication links allow the zone controllers to coordinate the hand-off of control as a vehicle crosses a zone boundary.

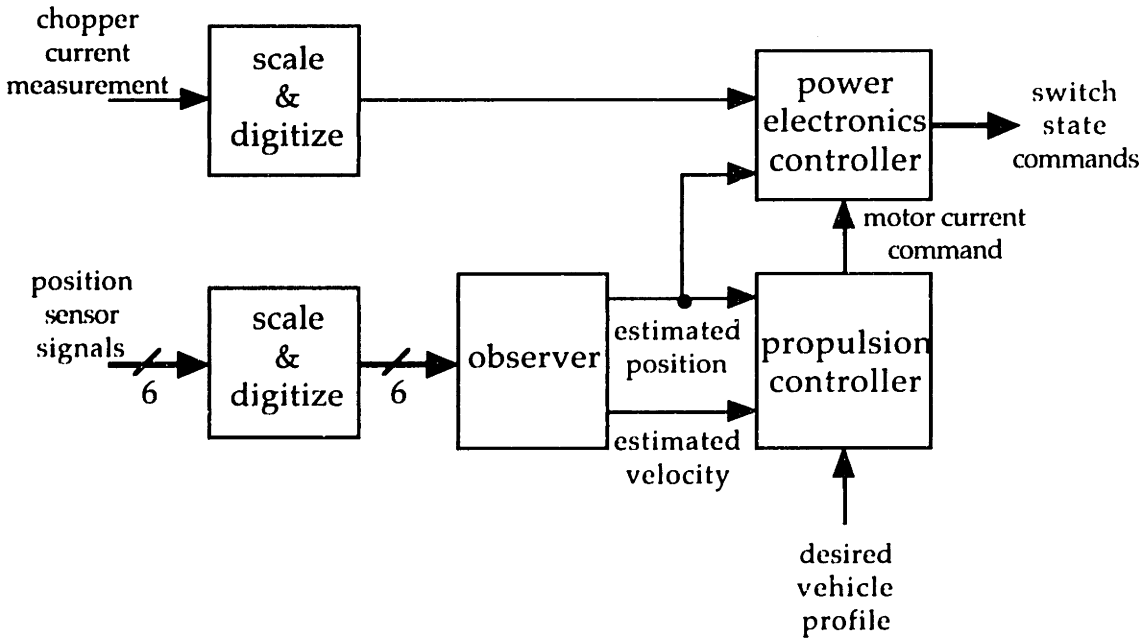


Figure 4-14. Block diagram of zone controller

HC16 EVB

The microcontroller board is constructed by adding circuitry to the HC16EVB, an evaluation board produced by Motorola to provide a simple path for developing a system based on the HC16 microcontroller IC [26]. In addition to the HC16 microcontroller, the HC16EVB provides a 32Kx16 block of RAM, a 32Kx16 block of EPROM (This block can also be populated with a 32Kx16 RAM, used as a 'pseudo-ROM'. The pseudo-ROM can be easily reprogrammed by downloading code from a computer over the host interface.), connectors for interfacing to additional circuitry or a logic analyzer, a prototyping area for adding additional circuitry on the evaluation board, and a host computer interface to allow system testing and debugging. The circuitry described in the following sections is added to the prototyping area of the HC16EVB.

A/D reference

A 5.12 Volt reference is provided to the A/D converter by the three-terminal regulator circuit shown in Figure 4-15. The LM309 is a 5 Volt reference, and the 1kΩ resistor and 50 Ω potentiometer are used to trim the reference output to 5.12 Volts.

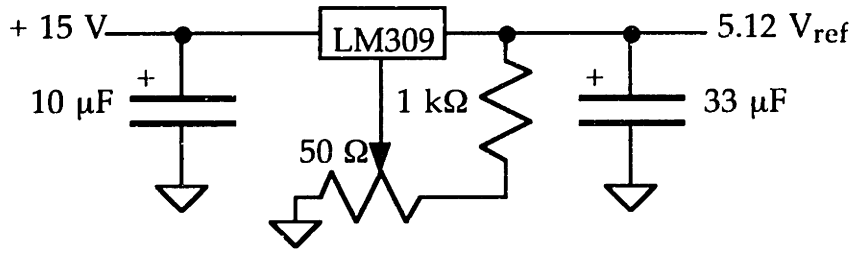


Figure 4-15. 5.12 Volt reference circuitry

A/D pre-filter

The RC network shown schematically in Figure 4-16 scales and shifts the demodulated position sensor signals from the analog processor board into the appropriate range for the A/D converter in the HC16 microcontroller. The network also provides a low-pass filter pole to help reject high frequency noise components. The position sensor signals from the analog processor board cover a ±10.24 Volt range; the resistor network maps this into the 0 to 5.12 Volt range required by the A/D converter. The capacitor value chosen results in a pole with a time constant of 1 μsec. A similar pre-filter network is used to scale and shift the signal from the chopper current sensor, with resistor values a factor of 10 larger to result in a low-pass filter pole with a time constant of 10 μsec.

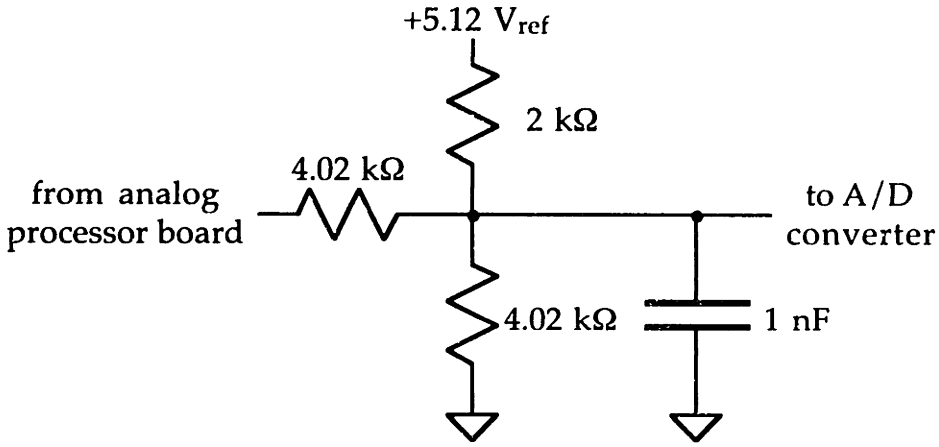


Figure 4-16. A/D converter pre-filter network

QSPI filter

The queued serial peripheral interface (QSPI) bus outputs from the HC16 are passed through RC filters before leaving the microcontroller board. The purpose of these filters is to slow down the logic transition edges to help reduce the amount of signal induced in other system wiring. The QSPI filter is shown in Figure 4-17.

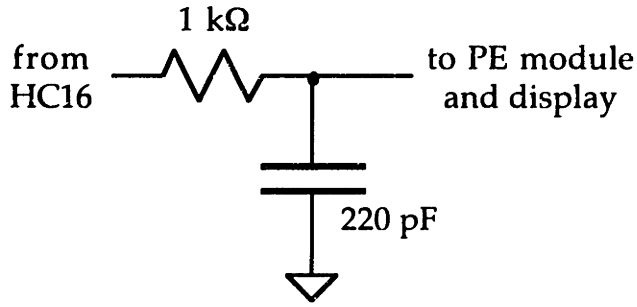


Figure 4-17. QSPI output filter network

DUART

The Motorola 68681 dual universal asynchronous receiver/transmitter (DUART) is placed in the memory map of the HC16 microcontroller to provide an additional two serial I/O ports to the zone controller. These additional serial ports are used to allow each zone controller to communicate with its nearest neighbors in order to coordinate the hand-off of control as a vehicle crosses a zone boundary. The DUART registers are mapped to page F of the HC16 memory map. Thus from a programming standpoint the DUART appears very similar to any of the internal peripherals of the HC16. The optocouplers allow the DUART signals to pass through an electrical

isolation boundary between zone controllers. A 3.58 MHz ceramic resonator is used to set the baud rate of the DUART channels to a value of approximately 19.2 kBaud. A schematic diagram of the DUART circuitry is shown in Figure 4-18.

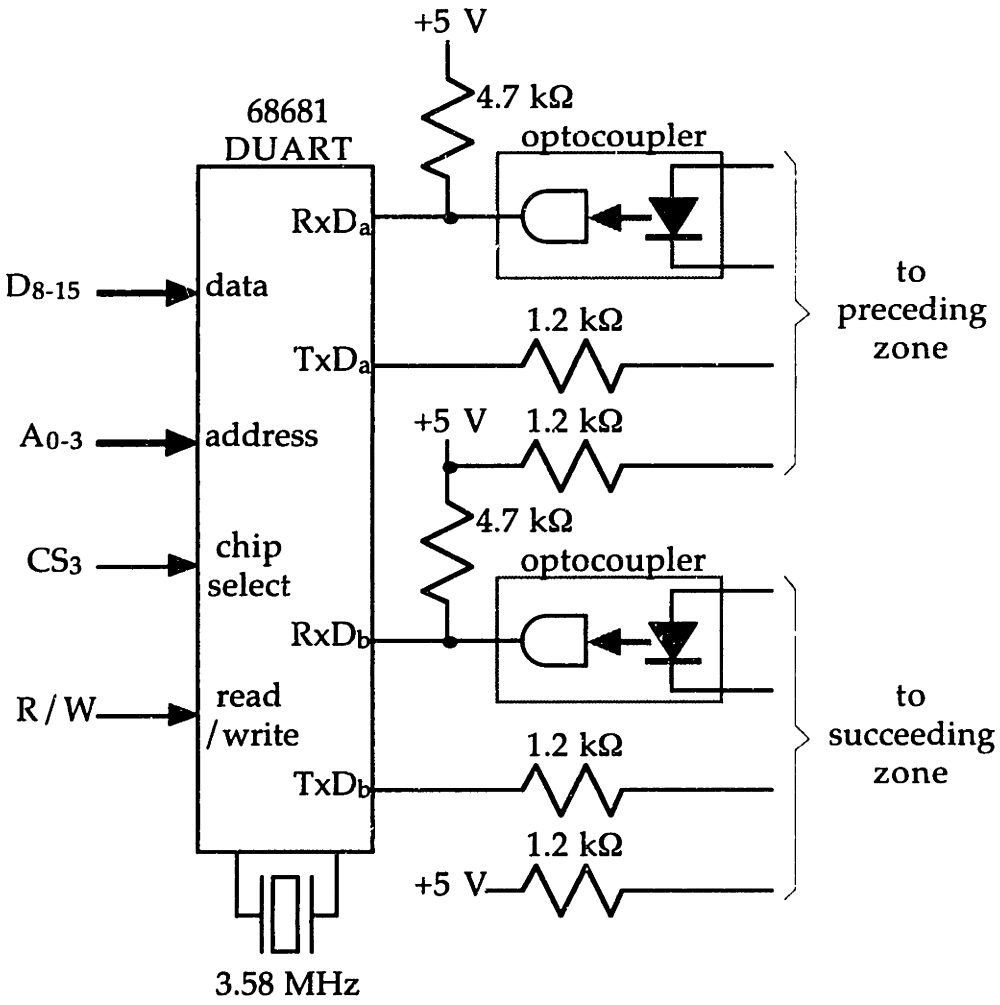


Figure 4-18. DUART circuitry

Hall effect switch

A Hall effect switch is embedded in the guideway at the center of each zone. This switch provides an absolute position reference to the zone controller, and is used to correct any cycle-slip events in the observer. (The observer tracks a periodic, or relative, electrical position measurement rather than an absolute position measurement.) The Hall effect switch produces a transition each time a vehicle magnet pole causes a reversal of magnetic flux by passing the switch, and the microcontroller keeps a record of the transition event. The occurrence of the first rising edge transition from the Hall switch

for a vehicle travelling through the zone in a forward direction implies that the vehicle is at a position of zero. (If the vehicle is travelling through the zone in reverse, the last falling edge transition implies a position of zero. For this circumstance, it is helpful for the controller to know how many magnet poles long the vehicle is.) When this reference edge occurs, the observer position estimate is checked. If the observer position is not equal to zero, it is reset and a note of the error is recorded. The Hall effect switch is shown in Figure 4-19, along with the waveform generated as a 2-bogey long vehicle passes the switch. When travelling in a forward direction the time axis increases from left to right, and the reference edge of the vehicle is the first rising edge from the switch. For a vehicle travelling in a reverse direction the time axis increases from right to left, and the fourth falling edge transition from the switch is the reference edge of the vehicle.

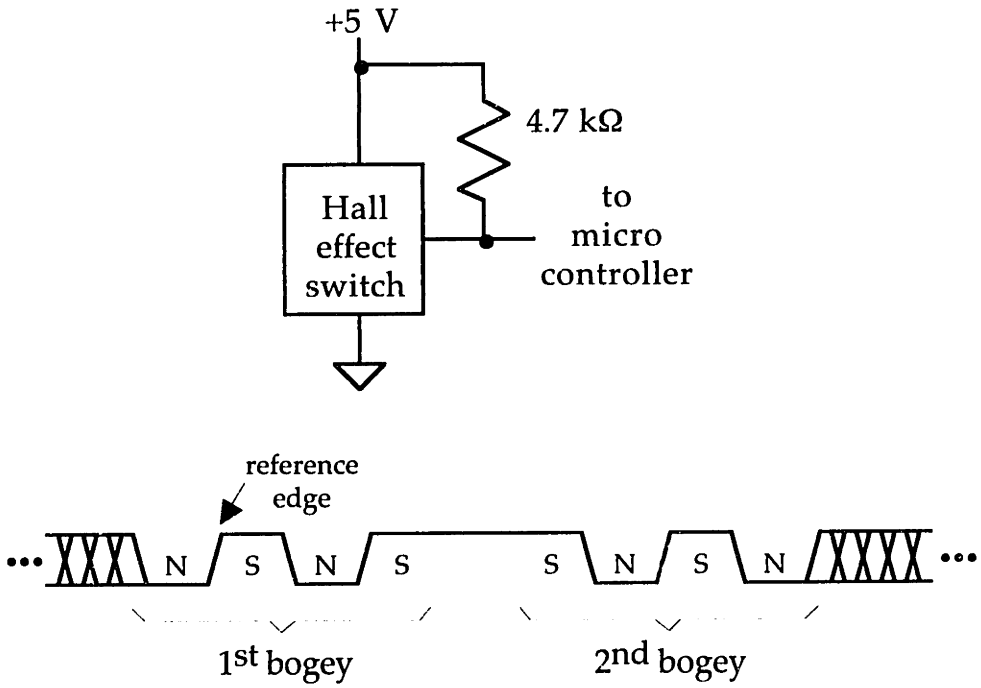


Figure 4-19. Hall effect switch and the output it produces

micro-code

A set of assembly language software routines govern the operation of the microcontroller board. Complete program listings are available in appendix B; only a brief description of the microcontroller code is included in this section. Initialization routines (to specify operation of the HC16 peripherals, *e.g.* clock speed, baud rates, A/D resolution) are not discussed in this section. Refer to [27] for details on the HC16 architecture. In all routines

vehicle velocity is stored as a 16-bit word with a range of ±256 (position sensor) radians/sec or ±3.06 meters/sec, and position is stored as a 32-bit long word with a range of ±16384 (position sensor) cycles, or ±1.23 kilometers. (These demonstration system ranges correspond to full scale values of ±153 m/sec and ±61.5 km.)

The code is written in a modular format. Each program calls on a number of subroutines to perform specific tasks. This approach streamlines generation of new programs, since much of the code required to perform basic tasks will be available in existing subroutines. In addition, making a modification or fixing a bug in a subroutine improves all programs using that subroutine. However, frequent jumps to and returns from subroutines tend to slow program execution. Since the code uses only approximately 30% of the execution time available, the modular code approach is advantageous. Each program includes unique 'Var.ASM' and 'Display.SUB' files. The 'Var.ASM' file specifies variable names, storage locations, and initial values. The 'Display.SUB' subroutine sends the data appropriate for the particular program to the display board for monitoring or debugging purposes.

The core of the code written for the microcontroller board is the implementation of the observer. This code digitizes the signals received from the analog processor board and updates estimates of the vehicle position and velocity to track the incoming data. The theoretical operation of the observer is addressed in Chapter 3. The program 'Obs2.ASM' is an implementation of a second order observer.

Data from the analog processor board is presented to the microcontroller board at discrete intervals, approximately every 312 μsec. When a new data set is ready, the QE signal from the analog processor board changes state, triggering an interrupt in the HC16. The interrupt handler routine, 'IC1Int', performs all operations required for the observer implementation. Upon completion of the required computations, the microprocessor simply waits (doing nothing) for the next interrupt event.

The interrupt handler routine first determines (by the logic state of QE) whether the incoming data corresponds to the even or odd motor phases, and then starts a set of A/D conversions on the appropriate channels. Next the observer states are updated using a forward Euler integration routine (the subroutine 'Update') based on the observer error calculated during the previous interrupt. When the A/D conversions on the incoming data have been completed, the observer error is calculated in the subroutine 'ErrCalc' by normalizing the incoming data (to reject gain, dc offset, and phase offset errors) and then performing the phase detector computations outlined in Chapter 3. The nonlinear functions $\mathcal{F}(\phi)$, $\mathcal{F}(\phi-2\pi/3)$, and $\mathcal{F}(\phi+2\pi/3)$ are

computed using an indexed look-up table. The observer states are transferred to the display board by the subroutine 'Display', and the interrupt handler routine is finished.

The program 'ToAndFro.ASM' builds on the observer by adding a commutation algorithm to ensure that the stator current is driven in quadrature to the vehicle field excitation. The subroutine 'DrivePE' is called in the IC1 interrupt handler, this subroutine uses a look-up table indexed by the observer's estimate of vehicle position to determine the appropriate state for the inverter switches. (This commutation algorithm results in motor operation analogous to a 'brushless dc' motor.) This program was written to operate on a short, straight segment of guideway, so the vehicle must travel in a back and forth motion. This behavior is sustained by the main program loop (outside the interrupt handler). When the vehicle travels to a position near the right end of the guideway segment the inverter commutation pattern, and therefore the motor thrust, is reversed. When the vehicle position nears the left end of the guideway thrust is switched back to the forward direction, thus causing the vehicle to oscillate to and fro.

The program 'ILoop.ASM' adds a chopper control algorithm to the 'ToAndFro' code. The chopper control algorithm allows the stator current magnitude, and therefore the motor thrust magnitude, to be controlled. The subroutine 'DrivePE' is modified to provide this feature. The chopper control algorithm allows four quadrant operation of the motor. The stator current reference is reversed by the main loop when the vehicle nears the ends of the straight guideway segment, causing the vehicle to travel back and forth with an approximately constant acceleration.

The program 'VLoop.ASM' adds a velocity control loop to the 'ILoop' code. The velocity control loop adjusts the stator current magnitude, and thus the propulsive force on the vehicle, to cause the vehicle to follow the reference velocity profile. Velocity control is implemented by the call to the 'VLoop' subroutine; the algorithm used is a simple proportional control law to compute a reference stator current based on the velocity error with an additional filter pole to smooth the resulting value of stator current. The velocity reference is reversed by the main loop when the vehicle nears the ends of the straight guideway segment, causing the vehicle to travel back and forth with an approximately constant velocity profile.

An additional feature makes its debut in the 'VLoop' code; transitions of the Hall-effect sensor embedded in the guideway are recorded by the 'IC2' subroutine. Each time the vehicle passes this sensor, which is placed at a known position on the guideway, the 'IC2' subroutine checks to make sure that the observer's position estimate is correct. If incorrect, the error is logged and the position estimate is corrected.

A significant amount of hardware and code are added for the program 'HandOff.ASM'. This program forces the vehicle to travel back and forth with a constant velocity just as the simpler 'VLoop' program does, however, the 'HandOff' program allows the vehicle to cross a zone boundary. As the vehicle crosses the zone boundary the neighboring zone controllers communicate with one another over a serial interface, thus facilitating a coordinated hand-off of vehicle control. The messages sent over the serial interface are several bytes in length, so a set of message queues are established to organize the transfer of data. Four queues are required in each zone controller; a transmit and a receive queue for communicating with both the preceding and the succeeding zone. The 'QueChk' subroutine, which is called from the main program loop, manages the activity of each queue. A zone controller sends messages to the adjacent zone when the vehicle approaches the zone boundary. The 'HOChk' subroutine, which is called from the main program loop, checks the observer's position estimate to see if the vehicle is near a zone boundary. If it is, a message is placed in the appropriate transmit queue. The 'SigChk' subroutine, which is called from inside the IC1 interrupt handler routine, implements the remaining new feature in the 'HandOff' code. 'SigChk' checks the quality of the data from the analog processor board to confirm the presence of a vehicle within the zone. If no vehicle is detected the zone controller is placed in a 'slave' mode, and simply waits for a message corresponding to an approaching vehicle. Upon receiving a message, the slave mode controller synchronizes with the master mode controller (*i.e.* the controller that sent the message), thus allowing the vehicle to cross the zone boundary with no discontinuity in propulsion. If the 'SigChk' subroutine detects a vehicle within the zone, the controller in placed in a 'master' mode.

The 'Cruise.ASM' program is actually slightly simpler than the 'HandOff' code. 'Cruise' propels the vehicle around the oval guideway at a constant velocity rather than back and forth on a straight guideway. This change is implemented by deleting the reference velocity reversal statements from the main program loop.

With the 'PLoop.ASM' program, the vehicle is once again travelling back and forth on a straight guideway segment with no zone boundaries (*i.e.* this program builds on 'VLoop', not 'HandOff' or 'Cruise'). A position control loop is added, causing the vehicle to move to a reference position. This mode of operation may be useful in a terminal, where it is desirable to have the vehicle stop at a specific location to load or unload passengers or freight. The position loop is implemented by the call to the subroutine 'PLoop'. A simple proportional control law is used to compute a reference velocity based on the position error.

The 'CubicP.ASM' program builds on 'PLoop' by defining the reference position profile as a function of time, rather than using a static reference position. This feature allows the controller to force the vehicle to follow a reference position trajectory rather than the reference velocity trajectory followed with the 'VLoop' code. Following a position trajectory provides integral control of the vehicle velocity, which ensures precise control of inter-vehicle spacing. For computational ease the reference position profile is made up of piecewise cubic segments. Each segment is one second in duration and is characterized by four coefficients. The coefficients for each segment are stored in a table. The subroutine 'PSetCalc' computes the reference position as a function of the system clock's time index.

The remaining three programs, 'Drag.ASM', 'DataLog.ASM', and 'Fourier.ASM', are used for test purposes. 'Drag' uses the current control loop from 'ILoop' to drive the vehicle back and forth with a constant current stator drive, and thus a constant motor force. The subroutine 'LogData' computes the acceleration from a series of velocity measurements and logs the result in an array indexed by velocity (*i.e.* the array data represents acceleration as a function of velocity). The array data is averaged over many vehicle runs to help filter out uncorrelated disturbances and the extent of the vehicle's oscillation is randomly modulated so the vehicle velocity is not correlated with position along the guideway.

The 'DataLog' program is used to measure the magnitude of the signals induced in each stator phase as a function of position. The vehicle and guideway are attached to a milling machine during this experiment to allow accurate positioning of the vehicle. This experiment was first conducted without the current transformers to measure the open circuit voltage induced in each stator phase by the vehicle transducer. The experiment was then repeated with the current transformers in place to measure the induced currents. In both experiment runs many data samples are averaged at each position to filter uncorrelated noise. The current transformer measurement data clearly shows the effects of the unbalanced stator inductance matrix on the position sensor signals.

The 'Fourier' program is used to measure the dc and fundamental sine and cosine components of the position sensor data induced in each motor phase. The data is logged while the vehicle travels back and forth at a constant velocity. The Fourier coefficients are calculated using the 'FourCalc' subroutine. The data from this routine is reduced off-line to compute the normalizing coefficients required to reject position sensor errors due to amplitude variations, dc offsets, and phase offsets in the position sensor data.

power electronics module

To produce the desired propulsive force, the LSM stator windings must be driven with a set of currents of variable magnitude and frequency. The magnitude of the stator currents establishes the magnitude of the propulsive force applied to the vehicle and the phase of the stator currents is chosen to keep the stator current pattern synchronized to the vehicle. The microcontroller determines the required state for each switch in the power electronics circuitry to provide the appropriate stator current magnitude and phase, and sends the switch state commands to the power electronics module over a synchronous serial interface. A block diagram of the power electronics circuitry is shown in Figure 4-20.

Research literature concerning the design and analysis of power electronics circuitry suitable for driving a large scale propulsion motor is readily available [28], and some papers specifically address issues concerning use in a MagLev system [29]. Further development of this research topic is not a goal of this thesis.

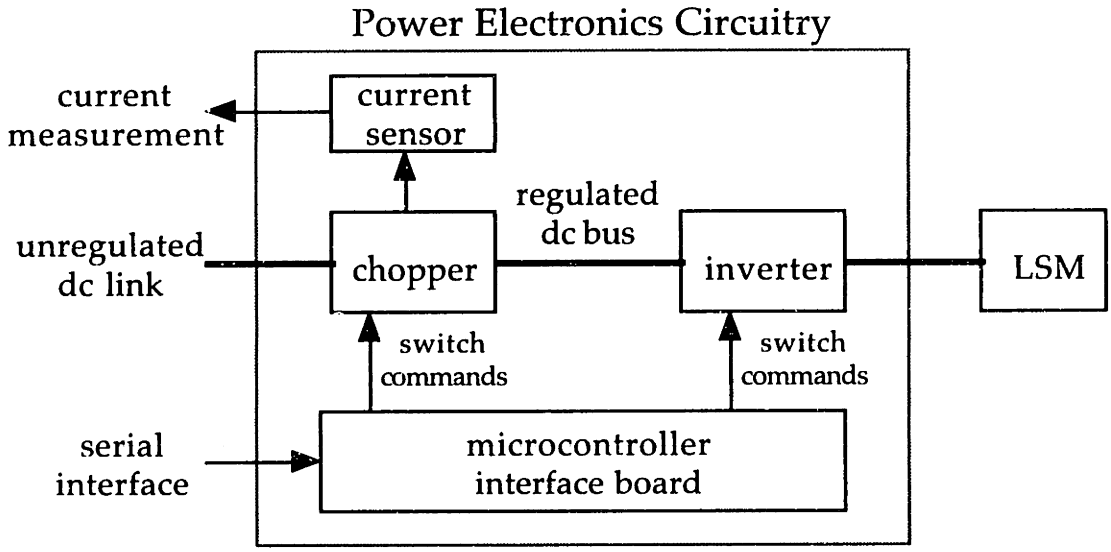


Figure 4-20. Block diagram of power electronics circuitry

Power from a 12 Volt lead-acid battery is distributed to the power electronics module in each of the four zones. The power electronics module chops this 'unregulated' dc link, thus providing a regulated dc voltage bus to the inverter. The magnitude of the regulated bus voltage determines the magnitude of the stator currents. The inverter transforms the regulated dc power into a set of variable frequency waveforms for the stator windings. The inverter regulates the phase of these variable frequency waveforms to match the commanded value from the controller.

interface board

The interface board receives switch state commands from the microcontroller over a synchronous serial interface (the queued serial peripheral interface, or QSPI). Figure 4-21 shows the QSPI receiver; the commands are shifted into a HC164 8-bit shift register, loaded into a pair of HC379 quad D flip-flops, and then drive the switch control logic of the chopper and inverter. The interface board also includes circuitry to supply power (a 30 Volt, 250 kHz square wave) to the switch drivers on each switch module board. A Hall-effect current sensor located on the microcontroller interface board measures the current flowing through the chopper inductor. This current measurement is digitized by the microcontroller and used to determine the chopper switch state commands.

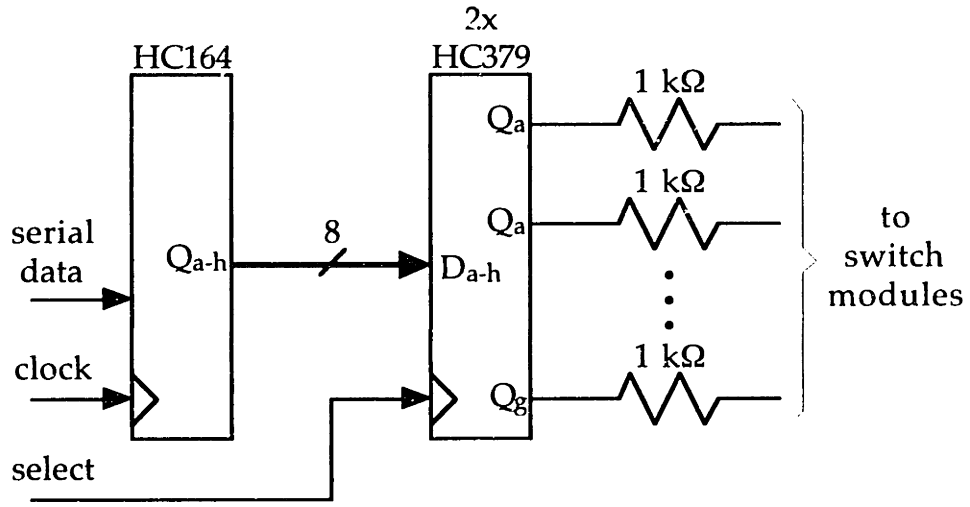


Figure 4-21. QSPI receiver

chopper

The chopper provides an ideally lossless means of transferring power between the unregulated dc link and the regulated dc bus [30]. Figure 4-22 is a simplified diagram of the power handling circuitry for a voltage source chopper. MOSFETs are used to implement the ideal switches shown in Figure 4-22. The microcontroller monitors the current level in the chopper circuitry and determines the proper state for each of the chopper switches. The chopper is capable of two-quadrant operation. When power flow is from the unregulated dc link to the inverter, the chopper is operating as a buck converter. This condition is the normal operating mode for the system; energy from the dc link propels the vehicle. (The LSM is operating as a motor.) When power flow is from the inverter to the dc link the chopper is

operating as a boost converter. This condition is the regenerative braking mode for the system; kinetic energy from the vehicle is absorbed by the dc link and made available to propel vehicles in other guideway zones. (The LSM is operating as a generator.)

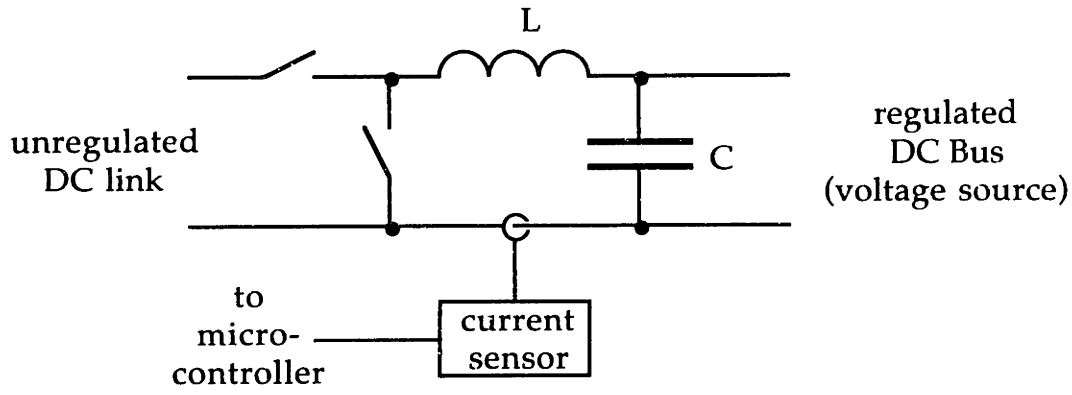


Figure 4-22. Chopper circuitry

The switching devices used to implement a chopper at power levels in the MWatt range (the power level necessary to propel a high speed vehicle) cannot support a switching frequency beyond a few kHz. The MOSFETs used as switch elements in the demonstration system are capable of switching frequencies in the MHz regime. The switching frequencies allowed for operation in the demonstration system are artificially limited to more closely model the behavior of a full size system. Switching events are only allowed at discrete times spaced at 312.5 μsec intervals, so the maximum possible switching frequency in the demonstration system is limited to 1.6 kHz.

The inductor and capacitor values used in the chopper are selected to provide adequate filtering of the high frequency ripple introduced by the chopper's switching action. The inductor used in the demonstration system has an inductance of 1.6 mH and can support currents up to 12 A. The chopper capacitance is physically distributed. A 470 μF capacitor is placed on each switch module in the inverter, thus the net capacitance is 0.7 mF.

inverter

The inverter provides an ideally lossless means of transferring power between the regulated dc bus and the LSM. There are many inverter topologies capable of performing this task. The simplest and most common inverter topology is the 6-switch bridge shown in Figure 4-23. This figure depicts a three phase motor connected in a wye configuration, but the concept can be easily extended to a six-phase motor such as the LSM used to propel the vehicle, and can be used with wye or delta connected motors.

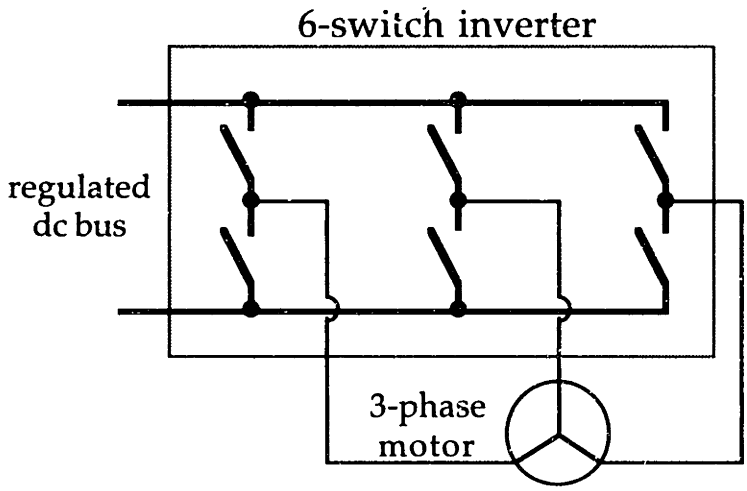


Figure 4-23. Simplified inverter topology

Figure 4-24 shows a six-phase inverter topology. The six motor phases are divided into two groups of three phases, the 'even' phases and the 'odd' phases. Each set of phases is driven by a three phase inverter and the inverters are stacked in series.

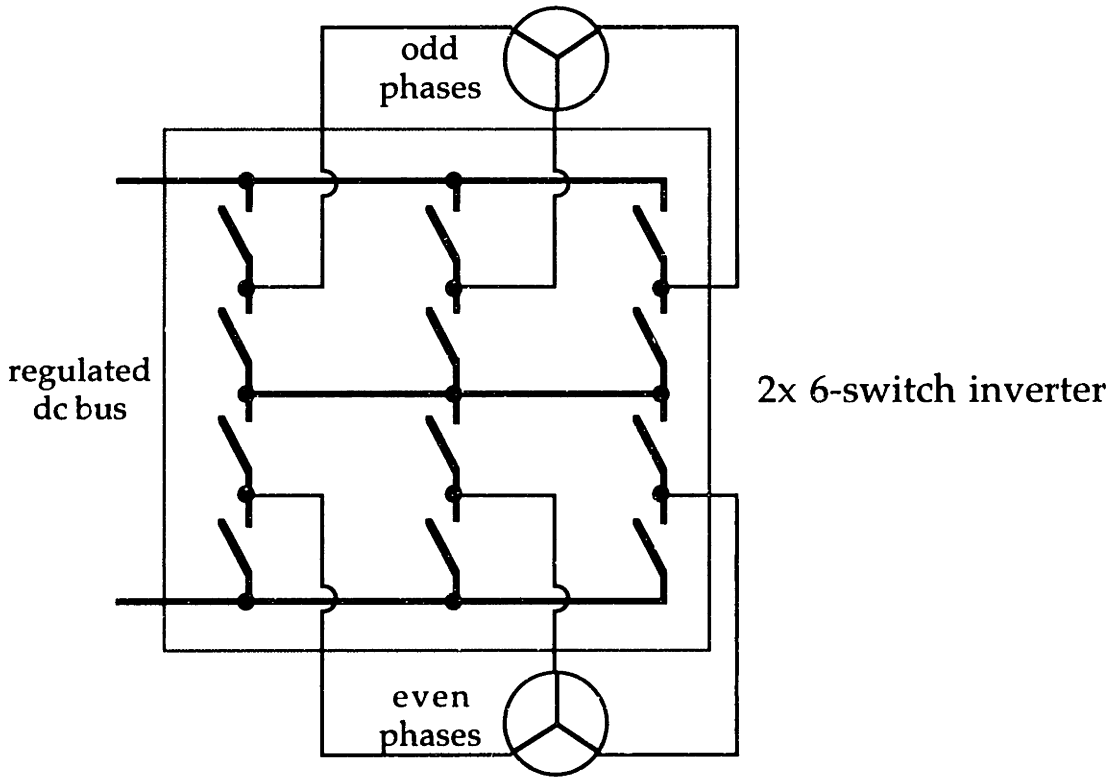


Figure 4-24. Six-phase inverter

switch module

The topologies for the chopper and inverter are based on a two-switch module, shown as a block diagram in Figure 4-25. The control inputs to the switch module are optically coupled to electrically isolate the power electronics switches from the control circuitry. The power supplies for the switch driver circuitry are isolated by utilizing a transformer to couple power into the switch module. Isolation of the switch driver circuitry and power devices provides a great deal of flexibility in the configuration of switch modules to build a variety of chopper and inverter topologies, as well as providing protection for the control circuitry in the event of a power electronics failure.

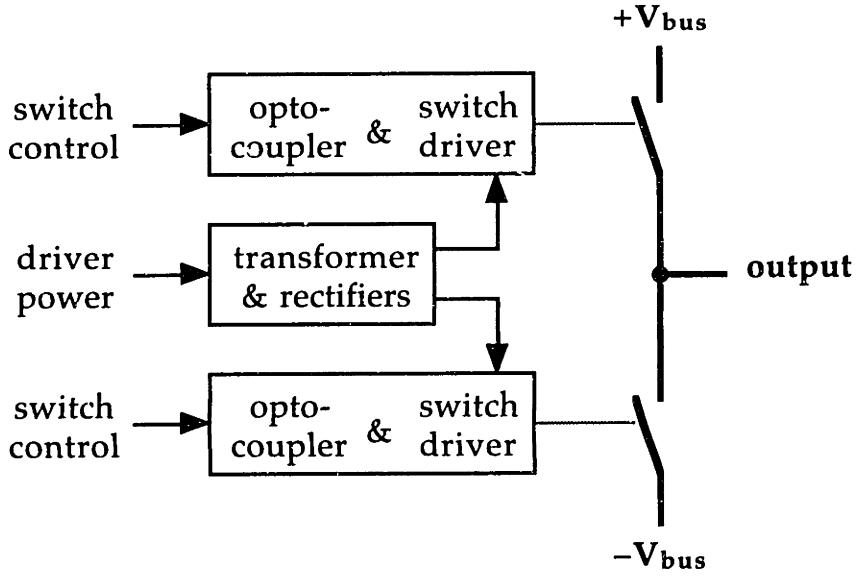


Figure 4-25. Block diagram of switch module

A schematic diagram of the optocoupler and switch driver circuitry is shown in Figure 4-26. There are two logic inputs to each switch driver channel, one input commands the switch state and the second input acts as an interlock to prevent turning on both the top and the bottom switch. The switch state command input passes through a diode-resistor-capacitor network to result in a longer delay for a switch turn-on transition. This 'shoot-through' delay prevents the possibility of both switches conducting during a switch transition. The interlock input acts as a switch enable; the switch cannot be turned on unless this input is driven on. The comparator fed with the interlock signal also performs an under-voltage lockout function. If the power supply for the switch driver circuitry is below 10 Volts, the switch is disabled. The switch command signal and the interlock signal are logically 'anded' by connecting the comparator outputs together. A CMOS buffer is used to drive the gate of the switch.

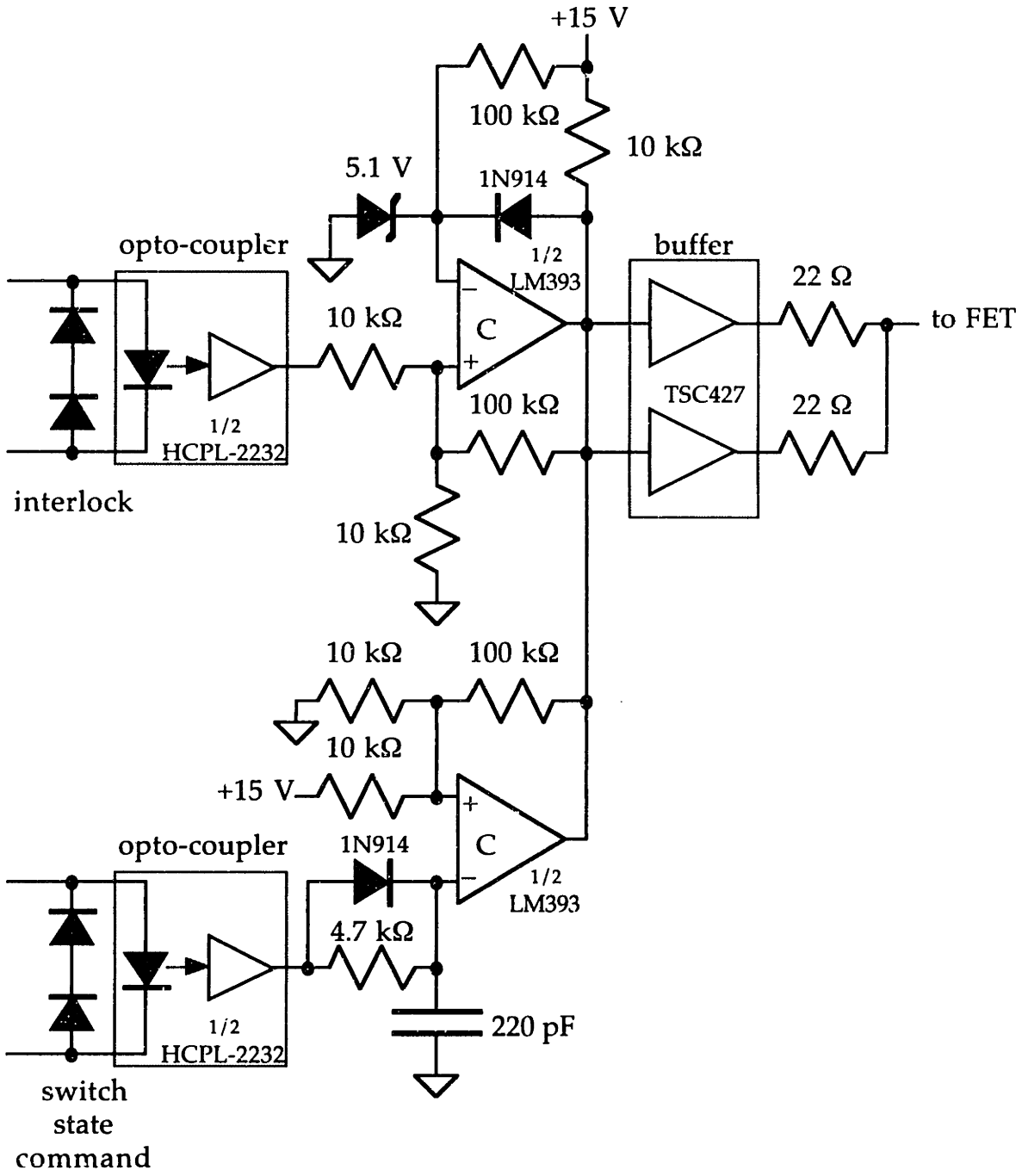


Figure 4-26. Optocoupler and switch driver schematic

The top and bottom switches are implemented as shown in Figure 4-27. Each switch is protected with a snubber to prevent damage from the energy stored in the stray inductance associated with the circuit board traces connecting the power devices. The diode in parallel with the gate of each switch device protects the CMOS buffer from possible latch-up. The 470 μF capacitor provides local decoupling of the regulated bus voltage and implements a portion of the LC filter for the chopper circuitry.

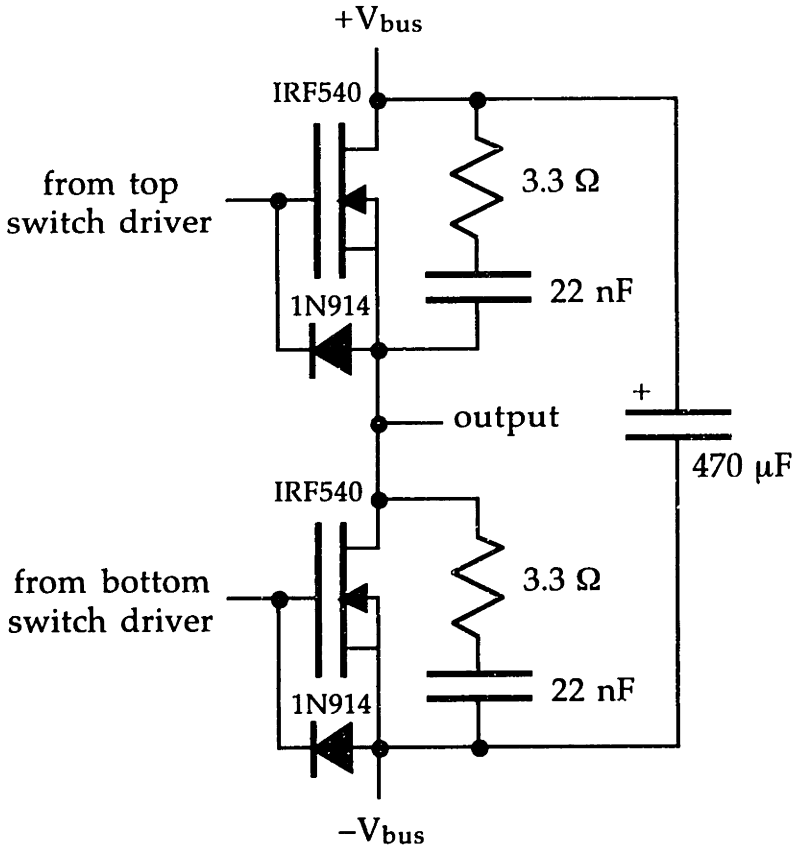


Figure 4-27. Switch configuration

Power for the optocoupler and switch driver logic is provided by a small transformer, as shown in Figure 4-28. The transformer has two secondaries to supply an isolated power source for each switch driver channel.

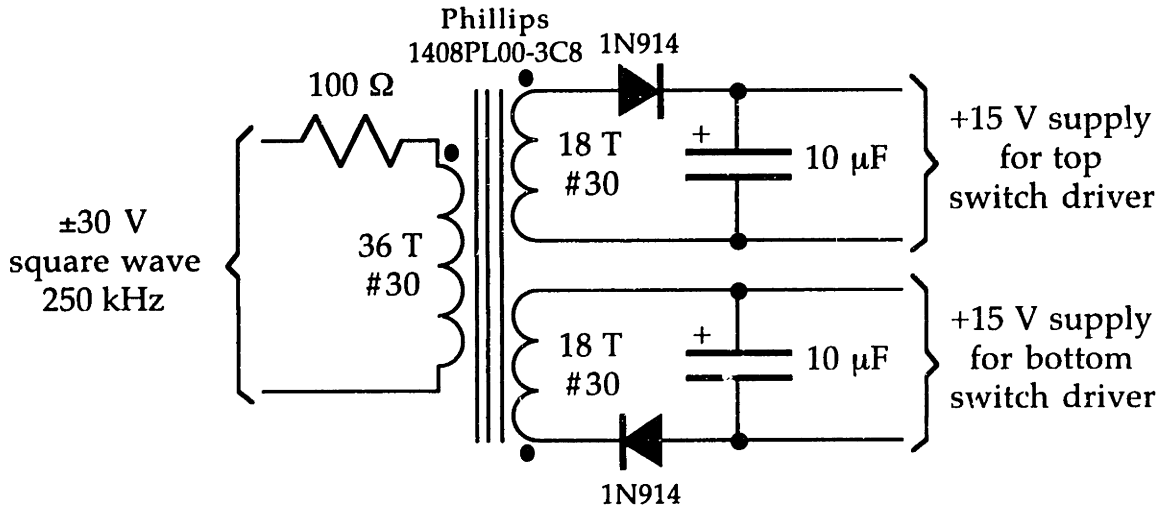


Figure 4-28. Switch module power transformer

The switch module can be tailored to system needs by changing the power devices. For topologies which require a switch that can conduct bipolar currents in the on state and can withstand unipolar voltages in the off state, a MOSFET is used to implement the switch. The switch module has a 40 Volt, 10 Ampere rating when populated with IRFZ40 devices, a 90 Volt, 7 Ampere rating when populated with IRF540 devices, and a 190 Volt, 5 Ampere rating when populated with IRF640 devices. For topologies that require a switch that can conduct a unipolar current in the on state and can withstand a bipolar voltage in the off state, a diode in series with a MOSFET is used to implement the switch. When configured in this manner, the forward drop of the diode results in an increase in the power dissipation of the switch while in the conducting state. To keep the operating temperature of the switch devices within the device limits, the current rating of a switch module configured for bipolar voltage and unipolar current must be reduced by approximately a factor of two. All current ratings assume the use of a heatsink with convective cooling, higher current ratings are possible if a forced air cooling system is utilized.

The chopper and inverter topologies described previously require switches that conduct bipolar currents in the on state and support unipolar voltages in the off state. IRF540 MOSFETs are used to implement the switches.

display board

The display board is used primarily for debugging purposes; it provides a means of viewing the status of the system. Information from the microcontroller board is transmitted to the display board via the queued serial peripheral interface, or QSPI, bus. (This is the same interface bus used to transmit switch state commands from the microcontroller board to the power electronics module.) The display board provides outputs in the form of hexadecimal LED digits, discrete LEDs, and DACs. The board contains five hex digit displays of four digits each, one bank of 16 discrete LEDs, and one bank of two 8-bit DACs. A block diagram of the display board is shown in Figure 4-29. Each data transmission from the microcontroller board is a 24-bit serial message. (Actually, the HC16 performs an 8-bit transfer immediately followed by a 16-bit transfer. The select line is held active continuously through both transfers, so the display board sees a single 24-bit transfer.) The incoming data is shifted into a 24-bit register. The first 8 bits received contain address information to select one of the hex digits, the LED bank, or the DAC bank as the transfer destination. (Since there are seven possible choices, only 3 of the address bits are actually used.) The last 16-bits received contain the data being transferred. After all 24 bits are shifted into the register the microcontroller board deactivates the select line, and the address decoder strobes the data into the appropriate destination.

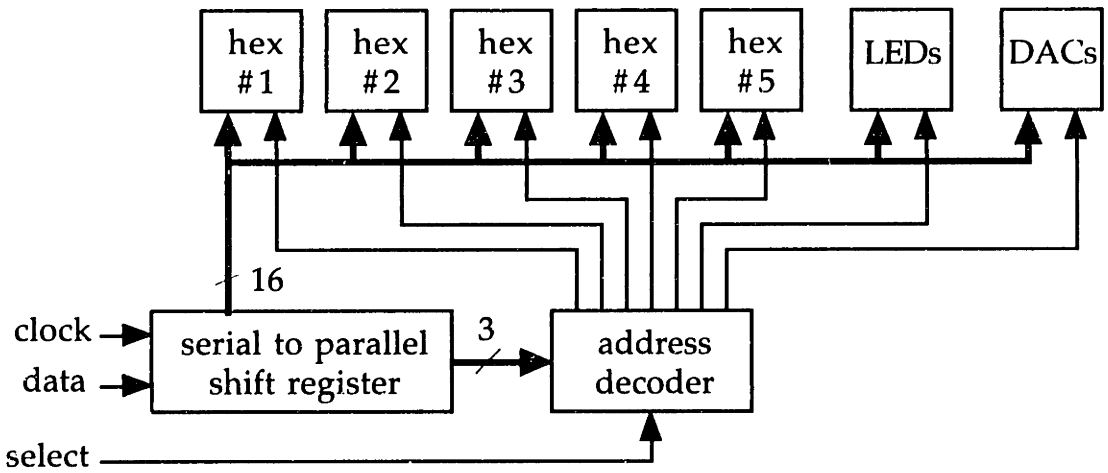


Figure 4-29. Block diagram of display board

5. Summary

The use of linear synchronous motor (LSM) technology rather than a conventional rotary electric motor for the propulsion system in a surface transportation system can lead to a number of performance advantages. The primary benefit associated with the use of LSM propulsion is a substantial increase in the available tractive force. The performance of a conventional drive using rotary electric motors to deliver traction via a steel wheel / rail interface is limited by the available friction. The LSM produces thrust through the interaction of excitation field magnets on the vehicle and stator winding currents in the guideway. The thrust available from the LSM is restricted only by infrastructure cost, stator heating, or safety issues. The LSM can produce enough thrust to accelerate a vehicle at 2.5 to 3.5 m/sec^2, while the acceleration limit of a conventional drive is nearly an order of magnitude smaller.

An increase in motor thrust capability can enhance the performance of a transportation system in a variety of ways. Safety is improved due to the decrease in the time and distance required to decelerate and stop the vehicle. Rapid acceleration and deceleration capability allows a reduction of the minimum headway requirement for safe operation. A low speed (50 km/h / 31 mph) system using an LSM might operate with a headway of a fraction of a second, and a high speed (500 km/h / 311 mph) system might operate with a headway of a fraction of a minute. A smaller headway leads to an increase in the capacity of the system. On a cost per mile basis the LSM drive is more expensive than a conventional drive, but the capacity improvement of the LSM system helps to keep the LSM system cost competitive with the conventional system on a cost per passenger mile basis.

The grade climbing capability of the system is directly related to the maximum thrust limit of the propulsion motor. A LSM drive significantly improves the performance of a vehicle operating over uneven terrain. The bulk of the LSM is not carried by the vehicle, and this interesting feature of the LSM can be exploited to reduce the system cost. The stator winding is mounted on the guideway and the power electronics are located at the wayside. The propulsion capability of the LSM can be tailored to meet local requirements by using a stator winding and power electronics package of the appropriate rating. More substantial components can be placed in areas of the guideway where maximum acceleration capability is required (*e.g.* a steep grade or ramps into and out of a station). Less expensive components can be used in areas of the guideway where acceleration is not critical.

The LSM does not require the transfer of propulsion power from the wayside to the vehicle. This feature is particularly important in a high speed transportation system where several MWatts of power must be transferred to a vehicle moving 500 km/h (311 mph). The maintenance costs and reliability problems associated with this type of power transfer are prohibitive. The elimination of the power handling hardware in the vehicle results in a lighter vehicle weight.

A linear induction motor (LIM) drive delivers a thrust capability similar to a LSM. However, the active vehicle LIM (the only LIM structure that has been successfully implemented in a full-size system) requires the transfer of propulsion power to the vehicle. In addition, the efficiency of a LIM is typically lower than the efficiency of a LSM using either permanent magnets or persistent-mode superconducting coils to generate the excitation field. The reduction in efficiency is more severe in a system with a substantial motor gap.

Some of the early magnetically levitated (MagLev) vehicle systems used a LIM for propulsion, but nearly all new designs employ LSM technology. Rotary electric motors used in conventional drive systems have evolved from dc machines to induction machines to synchronous machines. This trend is the result of improvements in devices available both for the power electronics and for the control hardware. Synchronous motors have been shown to be more efficient and less bulky than dc or induction machines in high speed rail transportation applications, but they require the use of more sophisticated drive electronics and control techniques.

Virtually all transportation systems using wheels for suspension employ a conventional rotary electric motor propulsion drive. This choice is appropriate for most applications. The cost associated with a LSM drive may not be warranted in an application with modest propulsion requirements. However, the LSM drive may be cost effective in a system with a requirement for high capacity, high speed, or steep grade climbing.

A number of control techniques designed to enhance the performance of a LSM drive were developed during the course of this project. Both theoretical and practical aspects of these techniques have been explored in this thesis. A laboratory scale system has been constructed to aid in the development, testing, and demonstration of several core control concepts necessary to support the operation of a basic transportation system.

The demonstration system is approximately 1/50 scale relative to a full size high speed MagLev system. The guideway is arranged as an oval with two 180° banked curves 4 meters in radius connected by two 4 meter long straight sections. Thus the total length of guideway is 33.6 meters and it

occupies an area 8 meters wide and 12 meters long. The stator windings for the LSM are attached to both sides of the guideway with a vertical orientation. The stators utilize a helical winding technique and are wound with six motor phases. This thesis project is focused on the propulsion system rather than a MagLev suspension, so a simple set of wheels is used for suspension and guidance of the vehicle. The field excitation array for the LSM is composed of neodymium iron boron (NdFeB) rare earth permanent magnets. The vehicle consists of a number of bogies connected with articulated joints. Each bogey contains a port and a starboard magnet array to link excitation flux with the stator winding on both sides of the guideway. The design of the vehicle and guideway are outlined in Chapter 2.

The guideway is divided into four zones to support the operation of multiple vehicles. Each zone contains a full set of wayside electronics including an analog processor board, a microcontroller board, a six-phase power inverter, and a chopper. A dc power distribution cable and a serial communication cable link each zone. The demonstration system is the first implementation of a multiple zone system using an LSM for propulsion. The inter-zone communication link is necessary to facilitate a coordinated hand-off of control as a vehicle crosses a zone boundary. The communication protocols implemented in the demonstration allow the vehicle to cross a zone boundary with no discontinuity in applied propulsive thrust. An overall description of the zone controller functions is included in Chapter 2, and the design and operation of the wayside electronics hardware are described in Chapter 4. Experimental data showing the hand-off of control as a vehicle crosses a zone boundary is presented in Chapter 2.

The wayside electronics requires measurements of the vehicle position and velocity to control the applied propulsive thrust. Absolute or mechanical position measurements are needed to control the spacing between vehicles or the speed profile of a vehicle as it nears a station. Relative or electrical position measurements are needed to maintain synchronous operation of the LSM. The accuracy required of the electrical position measurements to adequately maintain smooth and efficient motor operation is quite stringent.

An inexpensive position sensing system capable of an accuracy of 0.1 electrical degrees was developed and implemented in the demonstration system. This sensing system employs a transducer mounted on the vehicle to induce position-dependent signals into the stator winding. The amplitude of the signal induced in each phase of the motor varies as the position of the vehicle changes. A signal processing board located at the wayside detects the amount of signal present in each of the motor phases. This detected signal information is used to compute the vehicle position.

An observer is used to compute the vehicle position and velocity. The observer's estimate of the vehicle position is compared to the signals detected in the stator winding. Any error resulting from this comparison is used to drive the observer's estimate along a trajectory to more closely match the detected signals. This process forces the observer's state to track the state of the actual vehicle, thus providing measurements of vehicle position and velocity. The observer performs the calculations required to deduce the position and velocity of the vehicle in a very computationally efficient manner. A description of the position sensor hardware and the observer algorithm are provided in Chapter 3. Two observer implementations are discussed in Chapter 3. The nonlinear computation / linear observer approach is shown to have a few undesirable attributes, and an alternative approach based on a nonlinear observer is developed. The performance of the nonlinear observer is impressive, and this topology is incorporated into the demonstration system. A discussion of the theoretical and experimental performance of the position sensing system is included in Chapter 3.

A two-tiered control architecture is employed in the demonstration system. The detailed, high-speed, data-intensive computations required to maintain synchronous operation of the LSM are performed by zone controllers distributed along the guideway. Each zone controller concentrates its processing capability on a single vehicle. Higher level control functions such as scheduling, routing, and control of vehicle headway are performed by a central controller. The central controller views each vehicle in a more abstract sense than the zone controller. Less frequent and less precise measurements of vehicle position and velocity are adequate for the central controller to perform its tasks. The two-tiered control architecture simplifies the complexity of each control element and minimizes the need for a large data transfer bandwidth. The total cost of the control system is reduced by using inexpensive control hardware linked by inexpensive serial communication lines. In addition, the control architecture can be easily upgraded as the system expands.

A helical winding shape is used in the stator of the LSM in the demonstration system. This winding shape has been employed in a few rotary machines. A helical winding offers an improvement in the electrical properties of a slotless machine. The demonstration system constructed in this thesis project is the first application of the helical winding concept to a linear machine. The use of a helical winding technique dramatically simplifies the manufacture of the stator winding for the linear motor. Simplification of the manufacturing process of the stator is a compelling benefit since a transportation system requires a lot of stator winding. Helical and conventional stator windings are compared in Chapter 2.

The position sensor transducer and the field excitation array on the vehicle use different pole pitches. The pole pitch of the position sensor transducer is twice as long as the pole pitch of the field array. This configuration enables the wayside electronics to discriminate position sensor signals from propulsion signals using orientation as well as spectral information. Improved discrimination is beneficial since the propulsion signals are many orders of magnitude larger than the position sensor signals. This technique is discussed in Chapter 2, and experimental measurements showing the effectiveness of discrimination based on orientation are presented in Chapter 4. The rejection of unwanted information from the position sensor data is further enhanced by the synchronous detection scheme used in the analog processor board. The hardware implementation, circuit operation, and experimental measurements of the synchronous detector are presented in Chapter 4.

This thesis project has been quite rewarding from a technical perspective. Several important control concepts necessary for operation of a basic transportation system propelled by a LSM have been developed and successfully demonstrated. The hardware required to support the techniques outlined is quite modest due to an innovative implementation of algorithms and the use of the two-tiered control architecture. The application of the helical winding method to the stator resulted in a less expensive system with better performance, but the method used to join guideway segments created position sensor inaccuracies due to an imbalance in the inductance matrix of the stator. Signal processing techniques to correct the position sensor errors were developed. However, a symmetrical connection pattern physically located inside the box beam of a full-size system could eliminate this difficulty at its source.

The demonstration hardware constructed to support this project will be useful as a platform for the development of more advanced control techniques. There are several areas that require further effort. An adaptive tuning algorithm to compute the normalization coefficients required to correct position sensor errors should be implemented. Data from the sensing of the motor back EMF should be incorporated into the position sensor. A study of fault-tolerant control structures and cost-effective methods of achieving redundancy in a transportation system is an important area of research. Another interesting project might explore methods for using the accurate position sensing data available in the system to detect misalignment or wear in the guideway structure prior to serious damage or an extended loss of system availability. Finally, further study of the losses induced in the superconducting field magnets by stepped motor commutation is needed.

Appendix A

stator wiring harnesses

This appendix contains the connection lists for the wiring harnesses used to provide the appropriate stator winding connections at guideway joints. Three harness types are used, a 'straight' harness for a joint within a zone, an 'end' harness for a zone boundary joint, and an 'inverter' harness for connecting the power electronics to the stator winding. Each guideway joint requires a harness to provide the proper connections for the four 24-pin stator connectors. (Each 24-pin connector is actually made up of two 12-pin Molex .093" standard connectors.) Figure A-1 shows a top view of a guideway joint. The stator windings are crimped into 'plug' pins, thus the connectors are designated P1 through P4. The connections for each harness type are specified in tables A-1 through A-3. The harnesses use 'jack' pins, and their connectors are designated J1 through J4.

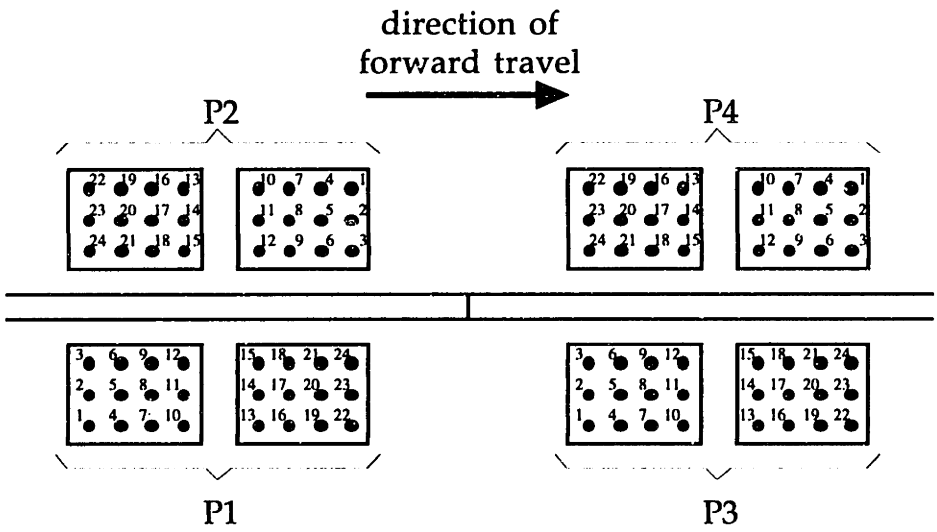


Figure A-1. Top view of connectors at a guideway joint

Table A-1. Connection list for 'straight' harness

J1.1-J3.1	J1.2-J3.2	J1.3-J3.3	J1.4-J3.4	J1.5-J3.5	J1.6-J3.6
J1.7-J3.7	J1.8-J3.8	J1.9-J3.9	J1.10-J3.10	J1.11-J3.11	J1.12-J3.12
J1.13-J3.13	J1.14-J3.14	J1.15-J3.15	J1.16-J3.16	J1.17-J3.17	J1.18-J3.18
J1.19-J3.19	J1.20-J3.20	J1.21-J3.21	J1.22-J3.22	J1.23-J3.23	J1.24-J3.24
J2.1-J4.1	J2.2-J4.2	J2.3-J4.3	J2.4-J4.4	J2.5-J4.5	J2.6-J4.6
J2.7-J4.7	J2.8-J4.8	J2.9-J4.9	J2.10-J4.10	J2.11-J4.11	J2.12-J4.12
J2.13-J4.13	J2.14-J4.14	J2.15-J4.15	J2.16-J4.16	J2.17-J4.17	J2.18-J4.18
J2.19-J4.19	J2.20-J4.20	J2.21-J4.21	J2.22-J4.22	J2.23-J4.23	J2.24-J4.24

Table A-2. Connection list for 'end' harness

J1.1-J1.7	J1.2-J1-20	J1.3-J1.21	J1.4-J1.10	J1.5-J1.11	J1.6-J1.24
J1.8-J1.14	J1.9-J1.15	J1.12-J1.18	J1.13-J1.19	J1.16-J1.22	J1.17-J1.23
J2.1-J2.19	J2.2-J2.8	J2.3-J2.9	J2.4-J2.22	J2.5-J2.23	J2.6-J2.12
J2.7-J2.13	J2.10-J2.16	J2.11-J2.17	J2.14-J2.20	J2.15-J2.21	J2.18-J2.24
J3.1-J3.7	J3.2-J1-20	J3.3-J3.21	J3.4-J3.10	J3.5-J3.11	J3.6-J3.24
J3.8-J3.14	J3.9-J3.15	J3.12-J3.18	J3.13-J3.19	J3.16-J3.22	J3.17-J3.23
J4.1-J4.19	J4.2-J4.8	J4.3-J4.9	J4.4-J4.22	J4.5-J4.23	J4.6-J4.12
J4.7-J4.13	J4.10-J4.16	J4.11-J4.17	J4.14-J4.20	J4.15-J4.21	J4.18-J4.24

Table A-3. Connection list for 'inverter' harness

J1.2-J2.5	J1.3-J2.4	J1.6-J2.1	J1.7-J2.24	J1.10-J2.21	J1.11-J2.20
J1.14-J2.17	J1.15-J2.16	J1.18-J2.13	J1.19-J2.12	J1.22-J2.9	J1.23-J2.8
J2.2-J4.2	J2.3-J4.3	J2.6-J4.6	J2.7-J4.7	J2.10-J4.10	J2.11-J4.11
J2.14-J4.14	J2.15-J4.15	J2.18-J4.18	J2.19-J4.19	J2.22-J4.22	J2.23-J4.23
J3.2-J4.5	J3.3-J4.4	J3.6-J4.1	J3.7-J4.24	J3.10-J4.21	J3.11-J4.20
J3.14-J4.17	J3.15-J4.16	J3.18-J4.13	J3.19-J4.12	J3.22-J4.9	J3.23-J4.8

J1.1-J1.13-Inverter 'A' J1.5-J1.17-Inverter 'B' J1.9-J1.21-Inverter 'C'
J1.4-J1.16-Inverter 'X' J1.8-J1.20-Inverter 'Y' J1.12-J1.24-Inverter 'Z'

J3.1-J3.13-J3.5-J3.17-J3.9-J3.21 (center of wye for ABC phases)*
J3.4-J3.16-J3.8-J3.20-J3.12-J3.24 (center of wye for XYZ phases)*

*Note: These connection pairs must thread through the appropriate current transformer before connecting to the center of wye points:

J3.1-J3.13 : phase A J3.5-J3.17 : phase B J3.9-J3.21 : phase C
J3.4-J3.16 : phase X J3.8-J3.20 : phase Y J3.12-J3.24 : phase Z

Appendix B

microcode

A brief description of the code written to implement the zone controller functions on a HC16 microcontroller is included in the discussion of the microcontroller board in Chapter 4. This appendix contains a full, commented listing of the following code segments:

Inits:
Equates.ASM, Vec.ASM, Sys.ASM, RAM.ASM, ADC.ASM, QSM.ASM, GPT.ASM, DUART.ASM

Subroutines:
QueChk.SUB, HOChk.SUB, Update.SUB, Limit.SUB, VLoop.SUB, DrivePE.SUB, ErrCalc.SUB, SigChk.SUB, IC2.SUB, PLoop.SUB, PSetCalc.SUB, Fourier.SUB, LogData.SUB

Programs:
Obs2.ASM, ToAndFro.ASM, ILoop.ASM, VLoop.ASM, HandOff.ASM, Cruise.ASM, PLoop.ASM, CubicP.ASM, Drag.ASM, DataLog.ASM, Fourier.ASM

Look-up Tables:
An abbreviated listing of the commutation table and the weighting function table in the file Table.ASM is included.

inits

```
*
*       Title : Vec.ASM
*       Description : This file is included to set up the reset
*                     vector ($00000 - $00006), and the GPT module
*                     IC1 interrupt vector ($0082,$0083)
*
*******************************************************************************

InitVec:      ORG      $0000          ;put the following reset vector information
                                      ;at address $00000 of the memory map
              DC.W     $00F0          ;zk=0, sk=F, pk=0
              DC.W     $0200          ;pc=200 -- initial program counter
              DC.W     $F3FE          ;sp=F3FE -- initial stack pointer
              DC.W     $0000          ;iz=0 -- direct page pointer

               ORG     $0082
              DC.W     IC1Int         ;jump vector for IC1 interrupt handler

*
*       Title : Sys.ASM
*       Description : Initialize & configure system including
*                     the Software Watchdog and System Clock.
*******************************************************************************

INITSYS:                             ;give initial values for extension registers
                                     ;and initialize system clock and COP

        LDAB    #$0F
        TBEK                         ; point EK to bank F for register access
        TBYK                         ; point YK to bank F
        LDAB    #$00
        TBXK                         ; point XK to bank 0
        LDAB    #$01
        TBZK                         ; point ZK to bank 1

;        LDD     #$00CF               ; make IMB available to CPU
;        STD     SIMMCR               ; (for some reason this statement crashes the
debugger)

        LDAA    #$7F                 ; w=0, x=1, y=111111
        STAA    SYNCR                ; set system clock to 16.78 Mhz

        CLR     SYPCR                ; turn COP (software watchdog) off,
                                     ; since COP is on after reset


        LDD     #$0003               ; at reset, the CSBOOT block size is 512k.
        STD     CSBARBT              ; this line sets the block size to 64k since
                                     ; that is what physically comes with the EVB16
        LDD     #$0103
        STD     CSBAR0               ; set up 64K RAM in page 1 of address space
        STD     CSBAR1
        STD     CSBAR2
        LDD     #$FFF8
        STD     CSBAR3               ; set up 2K block for external DUART
                                     ; (DUART uses first 16 bytes of this block)
```

```
        LDD     #$7830          ; no wait states for the pseudo-rom
        STD     CSORBT
        LDD     #$5030
        STD     CSOR0           ; CS0 -> upper byte, write only
        LDD     #$3030
        STD     CSOR1           ; CS1 -> lower byte, write only
        LDD     #$7830
        STD     CSOR2           ; CS2 -> both bytes, read and write
        LDD     #$38F0
        STD     CSOR3           ; CS3 -> 3 wait states, read and write

        LDD     #$16FF          ; CSBOOT chip select for 16-bit port
        STD     CSPAR0          ; EVB board also requires CS0,CS1 & CS2
        LDD     #$0055          ; to be configured as chip selects
        STD     CSPAR1          ; CS3 is chip select for DUART (8-bit port)
                                ; ADDR23/CS10 is set foe ECLK output

*
*       Title : RAM.ASM
*       Description : Initialize the HC16's 1K internal SRAM
*                     (put SRAM in memory map at $FFF000 to FFF3FF)
*                     and set the stack inside it.
*******************************************************************************

INITRAM:                        ;initialize internal SRAM and stack
        LDD     #$00FF
        STD     RAMBAH          ; store high ram array, bank F
        LDD     #$F000
        STD     RAMBAL          ; store low ram array
        CLR     RAMMCR          ; enable ram

        LDAB    #$0F
        TBSK                    ; set SK to bank F for system stack
        LDS     #$F3FE          ; put SP at top of 1k internal SRAM

        LDX     #$F000          ; initialize RAM contents...
        LDY     #$F000
RAMLoop:LDD     0,X
        STD     0,Y
        AIX     #2
        AIY     #2
        CPX     #$F3FE
        BNE     RAMLoop

*
*       Title : ADC.ASM
*       Description :  Initializes the A/D module for 10-bit conversions
*                      with a 9 uSecond conversion time.
*
*******************************************************************************

InitADC:
        LDD   #$0000
        STD   ADCMCR      ; enable A/D module
        LDD   #$0083
        STD   ADCTL0      ; configure A/D for 10-bit, 9 uSec conversions
                          ; write to ADCTL1 to start a conversion
```

```
                        ; $0020 for `odd' phases
                        ; $0024 for `even' phases

*
*       Title : QSM.ASM
*       Description : Initializes the Queued Serial Module for operation
*                     with the display board and the Power Electronics module.
*
*****************************************************************************

InitQSM:
        LDAA    #$34
        STAA    QPDR            ; PCS2, PCS1, & SCK inactive high, PCS0 inactive low
        LDAA    #$3A
        STAA    QPAR            ; assign PCS0, PCS1, PCS2, & MOSI pins to QSP module
        LDAA    #$FE
        STAA    QDDR            ; configure QSPI pins as outputs
        LDD     #$8308
        STD     SPCR0           ; master, CPOL=CPHA=1, 1.05 MHz, 16-bit transfer
        LDD     #$0000
        STD     SPCR2           ; newqp=0, endqp=0; 1 byte sent
                                ; (1 control byte for Power Electronics modele)
                                ; for transfer to display SPCR2 -> $0E01
                                ; (address and data for 6 displays)
                                ; (address and data for DAC)
;fill Command RAM:
        LDAA    #$04
        STAA    CR0             ; cont=0, bitse=0, PCS2,1,0 = 100, no delay
                                ; (8-bit transfer to PE module)

oddCR:  LDAA    #$82
        STAA    CR1             ; cont=1, bitse=0, PCS2,1,0 = 010, no delay
        STAA    CR3             ; (8 bits address transferred to display board)
        STAA    CR5             ; (continues into 'even' transfer)
        STAA    CR7
        STAA    CR9
        STAA    CRB
        STAA    CRD

evenCR: LDAA    #$62
        STAA    CR2             ; cont=0, bitse=1, PCS2,1,0 = 010, DT set
        STAA    CR4             ; (16 bits data transferred to display board)
        STAA    CR6             ; (no continuation)
        STAA    CR8
        STAA    CRA
        STAA    CRC
        STAA    CRE
;
;fill Transmit RAM (addresses only):
oddTR:  LDAA    #$00
        STAA    TR1+1           ; 8-bit address for first display...
        INCA
        STAA    TR3+1           ; second display...
        INCA
        STAA    TR5+1           ; third display...
        INCA
        STAA    TR7+1           ; fourth display...
        INCA
```

```
        STAA    TR9+1           ; fifth display...
        INCA
        STAA    TRB+1           ; sixth display...
        INCA
        STAA    TRD+1           ; and DACs 1 and 2.
;
;  'even'TR  -> 16-bit data

PFModule  EQU     TR0+1     ; -> PE data (8-bits)
Display1  EQU     TR2       ; -> display #1 data (16-bits)  A-Data
Display2  EQU     TR4       ; -> display #2 data (16-bits)  B-Data
Display3  EQU     TR6       ; -> display #3 data (16-bits)  C-Data
Display4  EQU     TR8       ; -> display #4 data (16-bits)  Position
Display5  EQU     TRA       ; -> display #5 data (16-bits)  Velocity
Display6  EQU     TRC       ; -> display #6 data (16-bits)  Switch states
DACData1  EQU     TRE       ; -> DAC1 data (8-bits)
DACData2  EQU     TRE+1     ; -> DAC2 data (8-bits)

*
*       Title : GPT.ASM
*       Description :  configure IC1 to record rising edge event time
*                      and interrupt processing
*
*******************************************************************************

InitGPT:
         LDD    #$0081
         STD    GPTMCR          ;IARB=1 for GPT module
         LDD    #$0240
         STD    GPTICR          ;Int. request level =2, Int. vector base =40h
                                ;(IC1 vector is 41h, or address 82 & 83h)
         LDAA   #$01
         STAA   TMSK1           ;IC1 generates interrupt
         LDAA   #$06
         STAA   TMSK2           ;prescaler is /256 (or 15.26 us per count)
         LDAA   #$0F
         STAA   TCTL2           ;IC1 & IC2 capture on both edges

*
*       Title : DUART.ASM
*       Description :  configure (external) Serial ports for 19.2 kBaud,
*                      8-bits, no parity, 1 stop-bit, no interrupts.
*
*******************************************************************************

InitUART:
        LDAB    #$E0
        STAB    URTACR              ; timer mode, baud set 2
        LDD     #$0003
        STAA    URTCTU              ; set timer for divide by 6 (19.2 kBaud)
        STAB    URTCTL
        LDD     #$1307
        STAA    URTMRA              ; no parity, 8-bits, 1 stop-bit
        STAB    URTMRA
        STAA    URTMRB
        STAB    URTMRB
        LDAB    #$DD
        STAB    URTSRA              ; use timer to set baud rate
```

```
STAB    URTSRB
LDAB    #$00
STAB    URTPCR              ; port pins are general purpose outputs
STAB    URTICR              ; mask interrupts
LDAB    #$05
STAB    URTCRA              ; enable TxA and RxA
STAB    URTCRB              ; enable TxB and RxB
```

subroutines

```
QueChk:                          ; check serial port queues
        BRCLR   URTSRA,#$01,BRxChk ; skip if no byte received by UART A
        LDY     #ARxQue          ; load IY with base of receive queue
        LDE     ARxPtr           ; load E with queue pointer
        LDD     URTDRA           ; read data byte
        STD     E,Y              ; store in receive queue
        INCW    ARxPtr           ; increment queue pointer
        CPE     #9               ; see if received message is complete
        BNE     BRxChk           ; skip if not complete
        CLRW    ARxPtr           ; reset pointer to beginning of queue
        BRSET   Flags,#$40,BRxChk ; ignore message if SigOK
        LDD     0,Y              ; load (relative) time index from queue
        ADDD    TCNT             ; convert to an absolute time index
        STD     PrevTime         ; store as previous observer update time
        LDD     8,Y              ; load chopper current from queue
        STD     ISet             ; store as chopper current set-point
        LDD     6,Y              ; load velocity from queue
        STD     IntErr           ; store as observer velocity
        LDD     4,Y              ; load (relative) position from queue
        LDE     2,Y
        ADDE    #BorderA         ; convert to this zone reference
        STED    Position         ; store as observer position

BRxChk: BRCLR   URTSRB,#$01,ATxChk ; skip if no byte received by UART B
        LDY     #BRxQue          ; load IY with base of receive queue
        LDE     BRxPtr           ; load E with queue pointer
        LDD     URTDRB           ; read data byte
        STD     E,Y              ; store in receive queue
        INCW    BRxPtr           ; increment queue pointer
        CPE     #9               ; see if received message is complete
        BNE     ATxChk           ; skip if not complete
        CLRW    BRxPtr           ; reset pointer to beginning of queue
        BRSET   Flags,#$40,ATxChk ; ignore message if SigOK
        LDD     0,Y              ; load (relative) time index from queue
        ADDD    TCNT             ; convert to an absolute time index
        STD     PrevTime         ; store as previous observer update time
        LDD     8,Y              ; load chopper current from queue
        STD     ISet             ; store as chopper current set-point
        LDD     6,Y              ; load velocity from queue
        STD     IntErr           ; store as observer velocity
        LDD     4,Y              ; load (relative) position from queue
        LDE     2,Y
        ADDE    #BorderB         ; convert to this zone reference
        STED    Position         ; store as observer position

ATxChk: LDE     ATxPtr           ; load Tx queue Pointer for UART A
        CPE     #9
        BEQ     BTxChk           ; skip if queue is empty
        BRCLR   URTSRA,#$04,BTxChk ; skip if Tx buffer is full
        LDY     #ATxQue          ; load IY with base of queue
        ADDE    #1               ; increment queue pointer
        STE     ATxPtr
        LDAA    E,Y              ; get next byte from queue
        STAA    URTDRA           ; put byte in Tx buffer
```

```
BTxChk: LDE     BTxPtr            ; load Tx queue Pointer for UART B
        CPE     #9
        BEQ     QCDone            ; skip if queue is empty
        BRCLR   URTSRB,#$04,QCDone ; skip if Tx buffer is full
        LDY     #BTxQue           ; load IY with base of queue
        ADDE    #1                ; increment queue pointer
        STE     BTxPtr
        LDAA    E,Y               ; get next byte from queue
        STAA    URTDRB            ; put byte in Tx buffer

QCDone: RTS

HOChk:                            ; check to see if vehicle is in hand-off region
        BCLR    Flags,#$0E        ; clear Hand-Off Region flags
        LDED    Position          ; first check 'A' border region...
        SUBE    #BorderA
        SUBE    #HandOff
        BPL     NotHORA           ; not in hand-off region A, skip
        ADDE    #HandOff
        ADDE    #HandOff
        BMI     NotHORA           ; not in hand-off region A, skip
        BSET    Flags,#$0C        ; set flags for Hand-Off Region A
        BRCLR   Flags,#$40,NotHORA ; don't send message if in slave mode
        LDD     MesgTrgA          ; see if time to send new message...
        CPD     TCNT              ; compare trigger to timer
        BPL     HOChkB            ; no trigger, check B hand-off region
        ADDD    #TrigPer          ; set next trigger time
        STD     MesgTrgA
        LDY     #ATxQue           ; load message into queue...
        CLRW    ATxPtr            ; set pointer to beginning of queue
        LDD     Velocity          ; put velocity in queue
        STD     6,Y
        LDE     #MesgDly          ; load message length (delay)
        FMULS                     ; predict vehicle movement during delay
        ADCE    #0
        JSR     TEDWSX
        ADDD    PosL              ; add predicted movement to position
        ADCE    PosH
        SUBE    #BorderA          ; put position (relative to border) in queue
        STE     2,Y
        STD     4,Y
        LDD     ISet              ; put chopper current in queue
        STD     8,Y
        LDD     TIC1              ; get (absolute) time index
        SUBD    TCNT              ; convert to a relative time index
        STAA    URTDRA            ; start message transmission
        STD     0,Y               ; put time index (relative to message) in queue
        BRA     HOChkB            ; check B hand-off region

NotHORA:LDD     TCNT              ; if not in hand-off region, reset trigger
        ADDD    #TrigPer
        STD     MesgTrgA

HOChkB: LDED    Position
        SUBE    #BorderB
        SUBE    #HandOff
        BPL     NotHORB           ; not in hand-off region B, skip
        ADDE    #HandOff
```

```
        ADDE    #HandOff
        BMI     NotHORB          ; not in hand-off region B, skip
        BSET    Flags,#$0A       ; set flags for Hand-Off Region B
        BRCLR   Flags,#$40,NotHORB ; don't send message if in slave mode
        LDD     MesgTrgB         ; see if time to send new message...
        CPD     TCNT             ; compare trigger to timer
        BPL     HOCDone          ; no trigger
        ADDD    #TrigPer         ; set next trigger time
        STD     MesgTrgB
        LDY     #BTxQue          ; load message into queue...
        CLRW    BTxPtr           ; set pointer to beginning of queue
        LDD     Velocity         ; put velocity in queue
        STD     6,Y
        LDE     #MesgDly         ; load message length (delay)
        FMULS                    ; predict vehicle movement during delay
        ADCE    #0
        JSR     TEDWSX
        ADDD    PosL             ; add predicted movement to position
        ADCE    PosH
        SUBE    #BorderB         ; put position (relative to border) in queue
        STE     2,Y
        STD     4,Y
        LDD     ISet             ; put chopper current in queue
        STD     8,Y
        LDD     TIC1             ; get (absolute) time index
        SUBD    TCNT             ; convert to a relative time index
        STAA    URTDRB           ; start message transmission
        STD     0,Y              ; put time index (relative to message) in queue
        RTS                      ; return from subroutine

NotHORB:LDD     TCNT             ; if not in hand-off region, reset trigger
        ADDD    #TrigPer
        STD     MesgTrgB

HOCDone:RTS                      ; return from subroutine

Update:                          ; use forward Euler integration routine to update
                                 ;values for Position, Velocity, and IntErr
        LDD     TIC1             ; calculate time since last update...
        TDE
        SUBD    PrevTime
        STE     PrevTime
        LDE     #128
        EMUL
        STD     DeltaT
        LDE     #K1              ; load loop gain constant
        FMULS                    ; compute (K1 x DeltaT)
        ADCE    #0
        LDD     ObsErr           ; load observer error
        FMULS                    ; compute (K1 x DeltaT x ObsErr)
        ADCE    #$0              ; round MSWord of product
        ADDE    IntErr           ; add integrated error to old IntErr value
        JSR     LimitE           ; (correct for over/under flow if necessary)
        STE     IntErr           ; to obtain the updated IntErr value
        LDD     ObsErr           ; load observer error value
        LDE     #K2              ; load loop gain constant
        FMULS                    ; compute proportional error term (K2 x ObsErr)
        ADCE    IntErr           ; round MSWord of product, add in IntErr term
```

```
        JSR     LimitE          ; correct for over/under flow if necessary
        STE     Velocity        ; store velocity value
        LDD     #K3             ; load loop gain constant
        FMULS                   ; compute (K3 x Velocity)
        ADCE    #0              ; round MSWord of product
        LDD     DeltaT
        FMULS                   ; compute (K3 x DeltaT x Velocity)
        ADCE    #0
        JSR     TEDWSX          ; transfer E to D with sign extend
        ADDD    PosL            ; add integrated Velocity to the old Position
        ADCE    PosH            ;value to obtain the updated Position value
        STED    Position        ; store new Position value
        LSRE                    ; shift down to +- pi range
        RORD                    ;(1 position sensor cycle)
        STD     LUTPos          ; store Look-Up Table Position
        RTS                     ; return from subroutine


LimitE:                         ; correct over/under flow of ADDE instruction
        BVC     Eok             ; if no over/under flow, jump to return
        BPL     EUnder          ; V=1 and N=0 => underflow
        LDE     #$7FFF          ; V=1 and N=1 => overflow, set E at maximum
        RTS                     ; return from subroutine
EUnder: LDE     #$8000          ; set E at minimum
Eok:    RTS                     ; return from subroutine

LimitD:                         ; correct over/under flow of ADDD instruction
        BVC     Dok             ; if no over/under flow, jump to return
        BPL     DUnder          ; V=1 and N=0 => underflow
        LDD     #$7FFF          ; V=1 and N=1 => overflow, set D at maximum
        RTS                     ; return from subroutine
DUnder: LDD     #$8000          ; set D at minimum
Dok:    RTS                     ; return from subroutine

LimitED:                        ; correct over/under flow of ADDD/E instructions
        BVC     EDok            ; if no over/under flow, jump to return
        BPL     EDUnder         ; V=1 and N=0 => underflow
        LDE     #$7FFF          ; V=1 and N=1 => overflow, set ED at maximum...
        LDD     #$FFFF
        RTS                     ; return from subroutine
EDUnder:LDE     #$8000          ; set ED at minimum...
        LDD     #$0000
EDok:   RTS                     ; return from subroutine

TEDWSX:                         ; transfer E to D with sign extend
        TED                     ; move E contents to D
        BMI     NegSgn          ; see if LSWord is negative
        CLRE                    ; if LSWord is not negative, extend plus
        RTS                     ; return from subroutine
NegSgn: LDE     #$FFFF          ; if LSWord is negative, extend minus
        RTS                     ; return from subroutine

VLoop:                          ; velocity control loop calculations...
                                ; ISet => (VSet - Velocity) x Kv / (.01s + 1)
        BRSET   Flags,#$40,ICalc ; do calculations for ISet if SigOK
        CLRD
        BRCLR   Flags,#$08,IFilter ; Sig notOK and not HORegion, current -> 0
        LDD     ISet
```

```
        BRA     IFilter          ; Sig notOK and in HORegion, skip calculations
ICalc:  LDD     VSet             ; get velocity set-point value
        SUBD    Velocity         ; calculate velocity error
        JSR     LimitD           ; check for over/underflow of D register
        LDE     #Kv              ; load velocity control loop gain constant
        FMULS                    ; multiply error by gain constant
        ADCE    #0               ; round MSWord of result
        TED                      ; put result in accumulator D
        CPD     #IMax            ; see if result is > IMax
        BLE     IMinChk          ; if not >, check minimum limit
        LDD     #IMax            ; if >, set at limit
IMinChk:CPD     #IMin            ; see if result is < IMin
        BGE     IFilter          ; if not <, no correction needed
        LDD     #IMin            ; if <, set at limit
IFilter:LDE     #K5              ; filter to smooth ISet...
        FMULS
        ADCE    ISet
        LDD     #K6
        FMULS
        ADCE    #0
        STE     ISet             ; store ISet value
VLDone: RTS                      ; return from subroutine

DrivePE:                         ; set switch states for the inverter and chopper
        LDD     Velocity         ; compute predicted position value for next
        LDE     #K4              ;commutation interval
        FMULS
        ADCE    PosL
        LSRE                     ; shift out unused bits...
        LSRE                     ; (only 10 bits used in table look-up)
        LSRE
        LSRE
        LSRE
        LSRE
        ADCE    #$0              ; round predicted position value
        ANDE    #$03FF           ; mask unused bits
        LDX     #TablePE         ; load look-up table base address
        LDAA    E,X              ; get switch states from look-up table
        LDE     ISet             ; get ISet value
        ASLE                     ; shift for comparison with A/D data...
        ASLE
        ASLE
        ASLE
        BRCLR   Flags,#$80,Chop  ; skip if ForceDir is forward
        EORA    #$77             ; invert switch states for reverse force
        NEGE                     ; negate ISet if reverse force
Chop:   SUBE    LJSRR3           ; see if IMeas >= ISet
        BLE     ChopLow          ; if IMeas >= ISet then chop low (MSB = 0)
        ORAA    #$80             ; if IMeas < ISet then chop high (MSB = 1)
        CLR     CLCnt            ; reset ChopLow counter
        BRA     QUE
ChopLow:INC     CLCnt            ; increment ChopLow counter
        LDAB    CLCnt
        SUBB    #MaxCnt          ; see if ChopLow count exceeds limit
        BMI     QUE              ; if not, then continue
        LDAB    Flags            ; if count exceeds limit, flip direction
        EORB    #$80
        STAB    Flags
```

```
        CLR     CLCnt           ; and reset counter
QUE:    STAA    PEModule        ; put switch states in queue for transfer
        LDD     #$0000          ; configure QSPI to transfer word 0
        STD     SPCR2           ; (control byte for power electronics module)
        LDD     #$8101
        STD     SPCR1           ; start transfer to power electronics module
        RTS

ErrCalc:                        ; calculate observer error...
        LDAB    #0
        TBXK                    ; X index points to bank 0
        LDAB    #$F
        TBYK                    ; Y index points to bank F
        LDE     LJSRR0          ; load phase A/X data
        ADDE    4,Y             ; add DC offset normalization
        LDD     0,Y             ; load normalization gain for phase A/X
                                ;(IY was set to select ABC or XYZ values)
        FMULS                   ; normalize A/X data
        ADCE    #0              ; round MSWord of product
        STE     Data0           ; save normalized data for SigChk subroutine
        LDD     LUTPos          ; load Look-Up Table Position
        ADDD    2,Y             ; add phase shift for phase A/X
        JSR     LookUp          ; get multiplier from Look-Up Table
        FMULS                   ; calculate phase A/X error component
        ADCE    #0              ; round MSWord of result
        STE     ObsErr          ; store result (A/X error component)

        LDE     LJSRR1          ; load phase B/Y data
        ADDE    10,Y            ; normalize B/Y offset
        LDD     6,Y             ; load normalization gain for phase B/Y
        FMULS                   ; normalize B/Y data
        ADCE    #0              ; round MSWord of product
        STE     Data1           ; save normalized data for SigChk subroutine
        LDD     LUTPos          ; load Look-Up Table Position
        ADDD    8,Y             ; add phase shift for phase B/Y
        JSR     LookUp          ; get multiplier from Look-Up Table
        FMULS                   ; calculate phase B/Y error component
        ADCE    ObsErr          ; round MSWord and add to A/X error component
        STE     ObsErr          ; store result (A/X + B/Y error components)

        LDE     LJSRR2          ; load phase C/Z data
        ADDE    16,Y            ; normalize C/Z offset
        LDD     12,Y            ; load normalization gain for phase C/Z
        FMULS                   ; normalize C/Z data
        ADCE    #0              ; round MSWord of product
        STE     Data2           ; save normalized data for SigChk subroutine
        LDD     LUTPos          ; load Look-Up Table Position
        ADDD    14,Y            ; add phase shift for phase C/Z
        JSR     LookUp          ; get multiplier from Look-Up Table
        FMULS                   ; calculate phase C/Z error component
        ADCE    ObsErr          ; round MSWord ,add to A/X+B/Y error components
        STE     ObsErr          ; store result (total error component)
        RTS                     ; return from subroutine

LookUp: ADDD    #2              ; round position value
        LSRD                    ; shift out unused bit (14-bit LUT address)
        ANDD    #$7FFE          ; mask unused bits
        ADDD    #FxTable        ; add look-up table base address
```

```
        XGDX                    ; put look-up table position in IX
        LDD    0,X              ; load multiplier value from Look-Up Table
        RTS                     ; return from subroutine

SigChk:                         ; compute signal magnitude and balance...
MagChk: LDE    Data0            ; SigMag = (Data0^2 + Data1^2 + Data2 ^2)/(.01s+1)
        TED                     ; (filtered, pole at 100 rad/sec)
        FMULS
        ADCE   #0
        LDD    #K5
        FMULS
        ADCE   SigMag
        STE    SigMag
        LDE    Data1
        TED
        FMULS
        ADCE   #0
        LDD    #K5
        FMULS
        ADCE   SigMag
        STE    SigMag
        LDE    Data2
        TED
        FMULS
        ADCE   #0
        LDD    #K5
        FMULS
        ADCE   SigMag
        LDD    #K6
        FMULS
        ADCE   #0
        STE    SigMag

        CPE    #$2000
        BMI    MagBad
        BSET   Flags,#$20       ; set MagOK flag
        BRA    BalChk
MagBad: BCLR   Flags,#$60       ; clear SigOK and Mag OK flags
        CLRW   ObsErr

BalChk: LDE    Data0            ; SigBal = Data0 + Data1 + Data2
        ADDE   Data1            ; (filtered, pole at 100 rad/sec)
        ADDE   Data2
        LDD    #K5
        FMULS
        ADCE   SigBal
        LDD    #K6
        FMULS
        ADCE   #0
        STE    SigBal
        BPL    SBPlus
        NEGE
SBPlus: CPE    #$1000
        BMI    BalOK
        BCLR   Flags,#$50       ; clear SigOK and BalOK flags
        CLRW   ObsErr
        RTS                     ; return from subroutine
BalOK:  BSET   Flags,#$10       ; set BalOK flag
```

```
        BRCLR Flags,#$20,SCDone ; skip if MagOK not set
        BSET  Flags,#$40      ; set SigOK flag
SCDone: RTS

IC2:    LDAA   TFLG1            ; read GPT status flags
        ANDA   #$FD
        STAA   TFLG1            ; reset IC2 flag
        BRSET  GPTPDR,#$02,REdge ; see if rising or falling edge

FEdge:  TSTW   Velocity         ; see if Velocity is positive or negative
        BMI    IC2Done          ; do nothing if Velocity < 0
        INC    EdgeCnt          ; increment EdgeCounter if Velocity > 0
        LDAA   EdgeCnt
        CMPA   #1               ; compare EdgeCount to Forward Reference Count
        BEQ    PosChk           ; if at reference edge, check Position value
        BRA    IC2Done          ; do nothing if not at reference edge
REdge:  TSTW   Velocity         ; see if Velocity is positive or negative
        BPL    IC2Done          ; do nothing if Velocity > 0
        DEC    EdgeCnt          ; decrement EdgeCounter if Velocity < 0
        LDAA   EdgeCnt
        BNE    IC2Done          ; do nothing if not at Reverse Reference Edge

PosChk: INCW   Passes           ; increment counter for vehicle passes
        LDD    PosH             ; get MSWord (global portion) of Position
        BRCLR  PosL,#$80,ZChk   ; skip if PosL < 0.5
        ADDD   #1               ; round PosH up if PosL >= 0.5
ZChk:   BEQ    IC2Done          ; if global position = 0, then skip correction
        INC    AdjCnt           ; increment position Adjustment Counter
        NEGD
        STD    GPosErr          ; store Global Position Error
        ADDD   NetChng          ; keep track of net adjustment required
        STD    NetChng
        LDD    PosH
        ADDD   GPosErr          ; correct global position
        STD    PosH
        BRCLR  GPosErr,#$01,IC2Done ; skip if GPosErr is an even number
        NEGW   AGain            ; if PosHErr is an odd number of half-cycles,
        NEGW   BGain            ;then all normalization gains for the observer
        NEGW   CGain            ;error detector must be negated...
        NEGW   XGain
        NEGW   YGain
        NEGW   ZGain
IC2Done:RTS                     ; return from subroutine

PLoop:                          ; position control loop calculations...
                                ; VSet => (PSet - Position) x Kp
        LDED   PSet             ; get set position value
        SUBD   PosL             ; subtract to get position error...
        SBCE   PosH
        STED   PErr             ; save position error value
        LDE    #Kp              ; get position loop gain constant
        EMUL                    ; multiply (PErrL x Kp)
        ADCE   #0               ; round result
        STE    VSet             ; store (intermediate) result
        LDD    PErrH            ; get MSWord of Position Error value
        LDE    #Kp              ; get position loop gain constant
        EMULS                   ; multiply (PErrH x Kp)
        ADDD   VSet             ; add in (PErrL x Kp)...
```

```
        STD     VSet            ; store VSet value
        ADCE    #0              ; carry into MSWord
LimitVSet:
        BMI     VMinChk         ; if VSet is negative, check VMin limit
        SUBD    #VMax           ; if VSet is positive, check VMax limit...
        SBCE    #0
        BMI     Vok             ; if VSet < VMax then it is ok
        LDD     #VMax           ; if VSet > VMax then set at maximum limit
        STD     VSet            ; store corrected VSet value
        BRA     Vok
VMinChk:SUBD    #VMin           ; check to see if VSet exceeds VMin limit...
        SBCE    #$FFFF
        BPL     Vok             ; if VSet > VMin then jump to Vok
        LDD     #VMin           ; if VSet < VMin then set at minimum limit
        STD     VSet            ; store corrected VSet value
Vok:    RTS                     ; return from subroutine


PSetCalc:                       ; calculate PSet value...
        BRCLR   TFLG2,#$80,SecOK ; skip if TOF not set
        BRSET   TIC1,#$80,SecOK ; skip if TOF occurred after IC1 capture
        INCW    SecCnt          ; increment Seconds Counter
        BCLR    TFLG2,#$80      ; reset TOF flag
        LDD     SecCnt          ; get Seconds Count
        ANDD    #$0007          ; modulo 8 pointer for loop
        ASLD                    ; multiply by 8...
        ASLD
        ASLD
        ADDD    #CoefBase       ; add Coefficient Base address
        STD     CoefPntr        ; store updated Coefficient Pointer

SecOK:  PSHM    Y               ; save IY contents
        LDY     CoefPntr        ; load IY with pointer to coef table entries
        LDE     0,Y             ; get A coef
        LDD     TIC1            ; get timer value
        LSRD
        FMULS                   ; multiply: At
        ADCE    2,Y             ; round and add B coef: At + B
        LDD     TIC1
        LSRD
        FMULS
        ADCE    4,Y             ; (At + B)t + C
        LDD     TIC1
        LSRD
        FMULS
        ADCE    6,Y             ; ((At + B)t + C)t + D
        LDD     #$0080          ; shift result to get PSet
        FMULS
        STED    PSet            ; store PSet value
        PULM    Y               ; restore IY contents
        RTS

LogData:                        ; average and store acceleration vs velocity
        LDD     Passes
        LBEQ    LDone           ; skip if counter = 0
        CPD     #$0400
        LBGT    LDone           ; skip if counter > 1024 runs
```

```
        LDE     V4              ; keep list of 5 recent velocity values...
        STE     V5
        LDE     V3
        STE     V4
        LDE     V2
        STE     V3
        LDE     V1
        STE     V2
        LDD     #$7C00          ; use IIR filter to compute V1
        FMULS
        ADCE    #0
        STE     V1
        LDE     Velocity
        LDD     #$0400
        FMULS
        ADCE    V1
        STE     V1

        LDD     V3
        STAA    AVavg+1         ; store average velocity
        LDE     #$0200
        FMULS
        ADDE    #$0201          ; round and convert to offset binary
        ANDE    #$03FE          ; mask unused bits
        BRCLR   ISet,#$80,LoadIZ ; keep +I and -I data in separate arrays
        ADDE    #$0400
LoadIZ: XGEZ                    ; store average velocity in index register

        LDE     0,Z             ; get value from data array
        LDD     #$7F80          ; multiply by (255/256)
        FMULS
        ADCE    #0
        LDD     V1              ; calculate acceleration...
        SUBD    V5              ; Accel = (V1 + 2V2 - 2V4 - V5)
        ADDD    V2
        SUBD    V4
        ADDD    V2
        SUBD    V4
        STAB    AVavg           ; store average acceleration value
        ADE                     ; add Accel to data array value
        STE     0,Z             ; and store back in data array

LDone:  RTS

FourCalc:
        LDD     PosH
        CPD     #$0010
        LBPL    FDone
        CPD     #$FFF0
        LBMI    FDone
        LDD     Passes
        LBLE    FDone
        CPD     #1000
        LBGT    FDone

        LDE     LJSRR0
        LDD     Velocity
        BPL     Vok1
```

```
        NEGD
Vok1:   FMULS
        ADCE    #0
        STE     Temp
        JSR     TEDWSX
        ADDD    76,Y
        ADCE    40,Y
        STD     76,Y
        STE     40,Y
        LDE     Temp
        LDD     LUTPos
        ADDD    2,Y
        JSR     LookUp
        FMULS
        ADCE    #0
        JSR     TEDWSX
        ADDD    72,Y
        ADCE    36,Y
        STD     72,Y
        STE     36,Y
        LDE     Temp
        LDD     LUTPos
        ADDD    2,Y
        ADDD    #$C000
        JSR     LookUp
        FMULS
        ADCE    #0
        JSR     TEDWSX
        ADDD    74,Y
        ADCE    38,Y
        STD     74,Y
        STE     38,Y

        LDE     LJSRR1
        LDD     Velocity
        BPL     Vok2
        NEGD
Vok2:   FMULS
        ADCE    #0
        STE     Temp
        JSR     TEDWSX
        ADDD    82,Y
        ADCE    46,Y
        STD     82,Y
        STE     46,Y
        LDE     Temp
        LDD     LUTPos
        ADDD    8,Y
        JSR     LookUp
        FMULS
        ADCE    #0
        JSR     TEDWSX
        ADDD    78,Y
        ADCE    42,Y
        STD     78,Y
        STE     42,Y
        LDE     Temp
        LDD     LUTPos
```

```
        ADDD    8,Y
        ADDD    #$C000
        JSR     LookUp
        FMULS
        ADCE    #0
        JSR     TEDWSX
        ADDD    80,Y
        ADCE    44,Y
        STD     80,Y
        STE     44,Y

        LDE     LJSRR2
        LDD     Velocity
        BPL     Vok3
        NEGD
Vok3:   FMULS
        ADCE    #0
        STE     Temp
        JSR     TEDWSX
        ADDD    88,Y
        ADCE    52,Y
        STD     88,Y
        STE     52,Y
        LDE     Temp
        LDD     LUTPos
        ADDD    14,Y
        JSR     LookUp
        FMULS
        ADCE    #0
        JSR     TEDWSX
        ADDD    84,Y
        ADCE    48,Y
        STD     84,Y
        STE     48,Y
        LDE     Temp
        LDD     LUTPos
        ADDD    14,Y
        ADDD    #$C000
        JSR     LookUp
        FMULS
        ADCE    #0
        JSR     TEDWSX
        ADDD    86,Y
        ADCE    50,Y
        STD     86,Y
        STE     50,Y

FDone:  RTS
```

```
*
*       MIT
*       Lab. for Electromagnetic and Electronic Systems
*       Cambridge, MA
*
*       File Name : Obs2.ASM
*
*       Description : This program digitizes signals from each of the stator
*                     windings and locks a 2nd order observer to track the
*                     position and velocity of the vehicle.
*
*       History : 11/09/93  Created.
*
*       Note : This program is written for the M68HC16Z1EVB.
*******************************************************************************

        nolist                      ;turn listing off for equates table
        INCLUDE '..\EQUATES.ASM'    ;table of EQUates for register addresses
        list                        ;turn listing back on after equates
        INCLUDE '..\Init\Vec.ASM'   ;initialize reset and interrupt vectors
        INCLUDE 'Var.ASM'

        ORG     $0200           ;start program after interrupt vectors and look-up
tables

*****   Initialization Routines *****
        INCLUDE '..\INIT\SYS.ASM'   ;initially set EK=F, XK=0, YK=0, ZK=0
                                    ;set sys clock at 16.78 MHz, disable COP
        INCLUDE '..\INIT\RAM.ASM'   ;initialize and turn on SRAM
                                    ;set stack (SK=F, SP=F3FE)
        INCLUDE '..\INIT\ADC.ASM'   ;configure A/D for 10-bit, 9 uSec conversion
        INCLUDE '..\INIT\QSM.ASM'   ;configure QSPI to interface with display
        INCLUDE '..\INIT\GPT.ASM'   ;configure IC1 interrupt

*****   Start of user program area  *****

        LDD     #$8000
        TDP                         ; allow interrupts

Main:                               ; Main program loop...
        BRA     Main                ; main loop does nothing yet

*****   Interrupt handler routines  *****

IC1Int:                             ; IC1 interrupt handler routine
        PSHM    D,E,X,Y,Z,K         ; save register contents

        BRSET   GPTPDR,#$01,OddSet ; see if even or odd data set
        LDD     #$0020
        STD     ADCTL1              ; start A/D conversion on even set
        LDY     #AGain              ; load IY for gain normalization base
        BRA     ResetIC1
OddSet: LDD     #$0024
        STD     ADCTL1              ; start A/D conversion on odd set
        LDY     #XGain              ; load IY for gain normalization base
ResetIC1:
```

```
        LDD     TFLG1           ; read GPT interrupt status flags
        ANDD    #$FEFF
        STD     TFLG1           ; reset IC1 interrupt flag

        JSR     Update          ; update observer values

WaitAD: BRCLR   ADSTAT,#$80,WaitAD ; wait for A/D conversion to finish

        JSR     ErrCalc         ; calculate observer error
        JSR     Display

IC1Done:PULM    D,E,X,Y,Z,K     ; restore register contents
        RTI                     ; return from IC1 interrupt handler

*****   Subroutines   *****
        INCLUDE '..\Subs\Update.SUB'
        INCLUDE '..\Subs\Limit.SUB'
        INCLUDE '..\Subs\ErrCalc.SUB'
        INCLUDE 'Display.SUB'


*
*       MIT
*       Lab. for Electromagnetic and Electronic Systems
*       Cambridge, MA
*
*       File Name : ToAndFro.ASM
*
*       Description : This program digitizes signals from each of the stator
*                     windings and locks a 2nd order observer to track the
*                     position and velocity of the vehicle. The vehicle
*                     oscillates back and forth about the start-up position.
*
*       History : 5/19/94  Created.
*
*       Note : This program is written for the M68HC16Z1EVB.
*******************************************************************************

        nolist                      ;turn listing off for equates table
        INCLUDE '..\EQUATES.ASM'    ;table of EQUates for register addresses
        list                        ;turn listing back on after equates
        INCLUDE '..\Init\Vec.ASM'   ;initialize reset and interrupt vectors
        INCLUDE 'Var.ASM'           ;assign constants and variable storage

        ORG     $0200           ;start program after interrupt vectors and look-up
tables

*****   Initialization Routines *****
        INCLUDE '..\INIT\SYS.ASM'   ;initially set EK=F, XK=0, YK=0, ZK=0
                                    ;set sys clock at 16.78 MHz, disable COP
        INCLUDE '..\INIT\RAM.ASM'   ;initialize and turn on SRAM
                                    ;set stack (SK=F, SP=F3FE)
        INCLUDE '..\INIT\ADC.ASM'   ;configure A/D for 10-bit, 9 uSec conversion
        INCLUDE '..\INIT\QSM.ASM'   ;configure QSPI to interface with
                                    ;Power Electronics Module and display
        INCLUDE '..\INIT\GPT.ASM'   ;configure IC1 interrupt

*****   Start of user program area  *****
```

```
        LDD     #$8000
        TDP                         ; allow interrupts

Main:                               ; Main program loop...
FWD:    BRSET   ForceDir,#$80,REV ; skip if force is in reverse direction
        LDD     PosH                ; get PosH value
        SUBD    #$0006              ; see if position > 6.0000
        BMI     REV                 ; if not > 6.0000, then do nothing
        BSET    ForceDir,#$80       ; if > 6.0000, then set ForceDir to reverse
REV:    BRCLR   ForceDir,#$80,FWD ; skip if force is in forward direction
        LDD     PosH                ; get PosH value
        ADDD    #$0006              ; see if position < -6.0000
        BPL     FWD                 ; if not < -6.0000, then do nothing
        BCLR    ForceDir,#$80       ; if < -6.0000, then set ForceDir to forward
        BRA     Main

*****   Interrupt handler routines  *****

IC1Int:                             ; IC1 interrupt handler routine
        PSHM    D,E,X,Y,Z,K         ; save register contents

        BRSET   GPTPDR,#$01,OddSet ; see if even or odd data set
        LDD     #$0020
        STD     ADCTL1              ; start A/D conversion on even data set
        LDY     #AGain              ; load IY for gain normalization base
        BRA     ResetIC1
OddSet: LDD     #$0024
        STD     ADCTL1              ; start A/D conversion on odd data set
        LDY     #XGain              ; load IY for gain normalization base
ResetIC1:
        LDD     TFLG1               ; read GPT interrupt status flags
        ANDD    #$FEFF
        STD     TFLG1               ; reset IC1 interrupt flag

        JSR     Update              ; update observer values

WaitAD: BRCLR   ADSTAT,#$80,WaitAD ; wait for A/D conversion to finish

        JSR     DrivePE             ; set switch states for the inverter and chopper
        JSR     ErrCalc             ; calculate observer error
        JSR     Display

IC1Done:PULM    D,E,X,Y,Z,K         ; restore register contents
        RTI                         ; return from IC1 interrupt handler


*****   Subroutines   *****
        INCLUDE '..\Subs\Update.SUB'
        INCLUDE '..\Subs\Limit.SUB'
        INCLUDE '..\Subs\ErrCalc.SUB'
        INCLUDE 'DrivePE.SUB'    ; note: this DrivePE version has no chopper controller
        INCLUDE 'Display.SUB'
```

```
*
*       MIT
*       Lab. for Electromagnetic and Electronic Systems
*       Cambridge, MA
*
*       File Name : ILoop.ASM
*
*       Description : This program digitizes signals from each of the stator
*                     windings and locks a 2nd order observer to track the
*                     position and velocity of the vehicle. The vehicle
*                     oscillates back and forth about the start-up position,
*                     controlling the motor current (force) in each direction.
*
*       History : 8/11/94  Created.
*
*       Note : This program is written for the M68HC16Z1EVB.
*******************************************************************************

        nolist                      ;turn listing off for equates table
        INCLUDE '..\EQUATES.ASM'    ;table of EQUates for register addresses
        list                        ;turn listing back on after equates
        INCLUDE '..\Init\Vec.ASM'   ;initialize reset and interrupt vectors
        INCLUDE 'Var.ASM'           ;assign constants and variable storage

        ORG     $0200           ;start program after interrupt vectors and look-up
tables

*****   Initialization Routines *****
        INCLUDE '..\INIT\SYS.ASM'   ;initially set EK=F, XK=0, YK=F, ZK=1
                                    ;set sys clock at 16.78 MHz, disable COP
        INCLUDE '..\INIT\RAM.ASM'   ;initialize and turn on SRAM
                                    ;set stack (SK=F, SP=F3FE)
        INCLUDE '..\INIT\ADC.ASM'   ;configure A/D for 10-bit, 9 uSec conversion
        INCLUDE '..\INIT\QSM.ASM'   ;configure QSPI to interface with
                                    ;Power Electronics Module and display
        INCLUDE '..\INIT\GPT.ASM'   ;configure IC1 interrupt

*****   Start of user program area  *****

        LDD     #$8000
        TDP                         ; allow interrupts

Main:                               ; Main program loop...
FWD·    LDD     PosH                ; get PosH value
        SUBD    #$000A              ; see if position > 10.0000
        BMI     REV                 ; if not > 10.0000, then do nothing
        LDD     RefCrnt             ; if > 10.0000, then ISet => -RefCrnt
        NEGD
        STD     ISet
REV:    LDD     PosH                ; get PosH value
        ADDD    #$000B              ; see if position < -10.0000
        BPL     FWD                 ; if not < -10.0000, then do nothing
        LDD     RefCrnt             ; if < -10.0000, then ISet => +RefCrnt
        STD     ISet
        BRA     Main

*****   Interrupt handler routines  *****
```

```
IC1Int:                         ; IC1 interrupt handler routine
        PSHM    D,E,X,Y,Z,K     ; save register contents

        BRSET   GPTPDR,#$01,OddSet ; see if even or odd data set
        LDD     #$0020
        STD     ADCTL1          ; start A/D conversion on even data set
        LDY     #AGain          ; load IY for gain normalization base
        BRA     ResetIC1
OddSet: LDD     #$0024
        STD     ADCTL1          ; start A/D conversion on odd data set
        LDY     #XGain          ; load IY for gain normalization base
ResetIC1:
        LDD     TFLG1           ; read GPT interrupt status flags
        ANDD    #$FEFF
        STD     TFLG1           ; reset IC1 interrupt flag

        JSR     Update          ; update observer values

WaitAD: BRCLR   ADSTAT,#$80,WaitAD ; wait for A/D conversion to finish

        JSR     DrivePE         ; set switch states for the inverter and chopper
        JSR     ErrCalc         ; calculate observer error

        BRCLR   TFLG1,#$02,Disp ; skip IC2 handler if flag not set
        JSR     IC2             ; process Hall sensor edge

Disp:   JSR     Display

IC1Done:PULM    D,E,X,Y,Z,K     ; restore register contents
        RTI                     ; return from IC1 interrupt handler

*****   Subroutines   *****
        INCLUDE '..\Subs\Update.SUB'
        INCLUDE '..\Subs\Limit.SUB'
        INCLUDE '..\Subs\ErrCalc.SUB'
        INCLUDE '..\Subs\DrivePE.SUB'; note: this DrivePE version has chopper control
        INCLUDE '..\Subs\IC2.SUB'
        INCLUDE 'Display.SUB'


*
*       MIT
*       Lab. for Electromagnetic and Electronic Systems
*       Cambridge, MA
*
*       File Name : VLoop.ASM
*
*       Description : This program digitizes signals from each of the stator
*                     windings and locks a 2nd order observer to track the
*                     position and velocity of the vehicle. The vehicle
*                     oscillates back and forth about the start-up position,
*                     controlling the velocity in each direction.
*
*       History : 6/7/94  Created.
*
*       Note : This program is written for the M68HC16Z1EVB.
*******************************************************************************
```

```
        nolist                      ;turn listing off for equates table
        INCLUDE '..\EQUATES.ASM'    ;table of EQUates for register addresses
        list                        ;turn listing back on after equates
        INCLUDE '..\Init\Vec.ASM'   ;initialize reset and interrupt vectors
        INCLUDE 'Var.ASM'           ;assign constants and storage for variables

        ORG     $0200           ;start program after interrupt vectors

*****   Initialization Routines *****
        INCLUDE '..\INIT\SYS.ASM'   ;initially set EK=F, XK=0, YK=F, ZK=1
                                    ;set sys clock at 16.78 MHz, disable COP
        INCLUDE '..\INIT\RAM.ASM'   ;initialize and turn on SRAM
                                    ;set stack (SK=F, SP=F3FE)
        INCLUDE '..\INIT\ADC.ASM'   ;configure A/D for 10-bit, 9 uSec conversion
        INCLUDE '..\INIT\QSM.ASM'   ;configure QSPI to interface with
                                    ;Power Electronics Module and display
        INCLUDE '..\INIT\GPT.ASM'   ;configure IC1 interrupt

*****   Start of user program area  *****

        LDD     #$8000
        TDP                         ; allow interrupts

Main:                               ; Main program loop...
FWD:    LDD     PosH                ; get PosH value
        SUBD    #$001A              ; see if position > 26.0000
        BMI     REV                 ; if not > 26.0000, then do nothing
        LDD     Speed               ; if > 26.0000, then VSet => -Speed
        NEGD
        STD     VSet
REV:    LDD     PosH                ; get PosH value
        ADDD    #$001B              ; see if position < -26.0000
        BPL     FWD                 ; if not < -26.0000, then do nothing
        LDD     Speed               ; if < -26.0000, then VSet => +Speed
        STD     VSet
        BRA     Main

*****   Interrupt handler routines  *****

IC1Int:                             ; IC1 interrupt handler routine
        PSHM    D,E,X,Y,Z,K         ; save register contents

        BRSET   GPTPDR,#$01,OddSet ; see if even or odd data set
        LDD     #$0020
        STD     ADCTL1              ; start A/D conversion on even data set
        LDY     #AGain              ; load IY for ABC normalization base
        BRA     ResetIC1
OddSet: LDD     #$0024
        STD     ADCTL1              ; start A/D conversion on odd data set
        LDY     #XGain              ; load IY for XYZ normalization base
ResetIC1:
        LDAA    TFLG1               ; read GPT interrupt status flags
        ANDA    #$FE
        STAA    TFLG1               ; reset IC1 interrupt flag

        JSR     Update              ; update observer values
        JSR     VLoop               ; do velocity control loop calculations
```

```
WaitAD: BRCLR  ADSTAT,#$80,WaitAD ; wait for A/D conversion to finish

        JSR    DrivePE          ; set switch states for the inverter and chopper
        JSR    ErrCalc          ; calculate observer error

        BRCLR  TFLG1,#$02,Disp  ; skip IC2 handler if flag not set
        JSR    IC2              ; process Hall sensor edge

Disp:   JSR    Display          ; load queue for transfer to display board

IC1Done:PULM   D,E,X,Y,Z,K      ; restore register contents
        RTI                     ; return from IC1 interrupt handler


*****   Subroutines   *****
        INCLUDE '..\Subs\Update.SUB'
        INCLUDE '..\Subs\Limit.SUB'
        INCLUDE '..\Subs\VLoop.SUB'
        INCLUDE '..\Subs\ErrCalc.SUB'
        INCLUDE '..\Subs\DrivePE.SUB'
        INCLUDE '..\Subs\IC2.SUB
        INCLUDE 'Display.SUB'


*
*       MIT
*       Lab. for Electromagnetic and Electronic Systems
*       Cambridge, MA
*
*       File Name : HandOff.ASM
*
*       Description : This program uses the velocity control loop from VLoop
*                     to force the vehicle to travel back and forth over a
*                     pair of track segments. As the vehicle approaches the
*                     zone boundary (joint), a series of messages is sent
*                     over a serial data link to allow a slave controller to
*                     synchronize its stator drive to the master.
*
*       History : 9/26/94  Created.
*
*       Note : This program is written for the M68HC16Z1EVB.
*******************************************************************************

        nolist                      ;turn listing off for equates table
        INCLUDE '..\EQUATES.ASM'    ;table of EQUates for register addresses
        list                        ;turn listing back on after equates
        INCLUDE '..\Init\Vec.ASM'   ;initialize reset and interrupt vectors
        INCLUDE 'Var.ASM'           ;assign constants and storage for variables

        ORG     $0200             ;start program after interrupt vectors

*****   Initialization Routines *****
        INCLUDE '..\INIT\SYS.ASM'   ;initially set EK=F, XK=0, YK=F, ZK=1
                                    ;set sys clock at 16.78 MHz, disable COP
        INCLUDE '..\INIT\RAM.ASM'   ;initialize and turn on SRAM
                                    ;set stack (SK=F, SP=F3FE)
        INCLUDE '..\INIT\ADC.ASM'   ;configure A/D for 10-bit, 9 uSec conversion
        INCLUDE '..\INIT\QSM.ASM'   ;configure QSPI to interface with
```

```
                                ;Power Electronics Module and display
        INCLUDE '..\INIT\GPT.ASM'  ;configure IC1 interrupt
        INCLUDE '..\INIT\DUART.ASM';configure external serial ports

*****   Start of user program area  *****

        LDD     #$8000
        TDP                     ; allow interrupts

Main:                           ; Main program loop...
QChk:   JSR     QueChk          ; check serial port queues
        JSR     HOChk           ; check for vehicle in a hand-off region
FWD:    LDD     PosH            ; get PosH value
        SUBD    #$0008          ; see if position > 8.0000
        BMI     REV             ; if not > 8.0000, then do nothing
        LDD     Speed           ; if > 8.0000, then VSet => -Speed
        NEGD
        STD     VSet
REV:    LDD     PosH            ; get PosH value
        ADDD    #$0009          ; see if position < -8.0000
        BPL     Main            ; if not < -8.0000, then do nothing
        LDD     Speed           ; if < -8.0000, then VSet => +Speed
        STD     VSet
        BRA     Main

*****   Interrupt handler routines  *****

IC1Int:                         ; IC1 interrupt handler routine
        PSHM    D,E,X,Y,Z,K     ; save register contents

        BRSET   GPTPDR,#$01,OddSet ; see if even or odd data set
        LDD     #$0020
        STD     ADCTL1          ; start A/D conversion on even data set
        LDY     #AGain          ; load IY for gain normalization base
        BRA     ResetIC1
OddSet: LDD     #$0024
        STD     ADCTL1          ; start A/D conversion on odd data set
        LDY     #XGain          ; load IY for gain normalization base
ResetIC1:
        LDAA    TFLG1           ; read GPT interrupt status flags
        ANDA    #$FE
        STAA    TFLG1           ; reset IC1 interrupt flag

        JSR     Update          ; update observer values
        JSR     VLoop           ; do velocity control loop calculations

WaitAD: BRCLR   ADSTAT,#$80,WaitAD ; wait for A/D conversion to finish

        JSR     DrivePE         ; set switch states in the inverter and chopper
        JSR     ErrCalc         ; calculate observer error
        JSR     SigChk          ; check integrity of incoming position data

        BRCLR   TFLG1,#$02,Disp ; skip IC2 handler if flag not set
        JSR     IC2             ; process Hall sensor edge

Disp:   JSR     Display         ; load queue and transfer to display board

IC1Done:PULM    D,E,X,Y,Z,K     ; restore register contents
```

```
        RTI                     ; return from IC1 interrupt handler

*****   Subroutines   *****

        INCLUDE  '..\Subs\QueChk.SUB'
        INCLUDE  '..\Subs\HOChk.SUB'
        INCLUDE  '..\Subs\Update.SUB'
        INCLUDE  '..\Subs\Limit.SUB'
        INCLUDE  '..\Subs\VLoop.SUB'
        INCLUDE  '..\Subs\DrivePE.SUB'
        INCLUDE  '..\Subs\ErrCalc.SUB'
        INCLUDE  '..\Subs\SigChk.SUB'
        INCLUDE  '..\Subs\IC2.SUB'
        INCLUDE  'Display.SUB'


*
*       MIT
*       Lab. for Electromagnetic and Electronic Systems
*       Cambridge, MA
*
*       File Name : Cruise.ASM
*
*       Description : This program uses the velocity control loop from VLoop
*                     to force the vehicle to cruise around the oval guideway
*                     at a constant velocity. As the vehicle approaches a
*                     zone boundary (joint), a series of messages is sent
*                     over a serial data link to allow a slave controller to
*                     synchronize its stator drive to the master.
*
*       History : 11/9/94  Created.
*
*       Note : This program is written for the M68HC16Z1EVB.
*******************************************************************************

        nolist                  ;turn listing off for equates table
        INCLUDE '..\EQUATES.ASM'  ;table of EQUates for register addresses
        list                    ;turn listing back on after equates
        INCLUDE '..\Init\Vec.ASM'  ;initialize reset and interrupt vectors
        INCLUDE 'Var.ASM'         ;assign constants and storage for variables

        ORG    $0200       ;start program after interrupt vectors

*****   Initialization Routines *****
        INCLUDE '..\INIT\SYS.ASM'  ;initially set EK=F, XK=0, YK=F, ZK=1
                                   ;set sys clock at 16.78 MHz, disable COP
        INCLUDE '..\INIT\RAM.ASM'  ;initialize and turn on SRAM
                                   ;set stack (SK=F, SP=F3FE)
        INCLUDE '..\INIT\ADC.ASM'  ;configure A/D for 10-bit, 9 uSec conversion
        INCLUDE '..\INIT\QSM.ASM'  ;configure QSPI to interface with
                                   ;Power Electronics Module and display
        INCLUDE '..\INIT\GPT.ASM'  ;configure IC1 interrupt
        INCLUDE '..\INIT\DUART.ASM';configure external serial ports

*****   Start of user program area  *****

        LDD    #$8000
        TDP                ; allow interrupts
```

```
Main:                       ; Main program loop...
        JSR     QueChk      ; check serial port queues
        JSR     HOChk       ; check for vehicle in a hand-off region
        BRA     Main

*****   Interrupt handler routines  *****

IC1Int:                     ; IC1 interrupt handler routine
        PSHM    D,E,X,Y,Z,K ; save register contents

        BRSET   GPTPDR,#$01,OddSet ; see if even or odd data set
        LDD     #$0020
        STD     ADCTL1      ; start A/D conversion on even data set
        LDY     #AGain      ; load IY for gain normalization base
        BRA     ResetIC1
OddSet: LDD     #$0024
        STD     ADCTL1      ; start A/D conversion on odd data set
        LDY     #XGain      ; load IY for gain normalization base
ResetIC1:
        LDAA    TFLG1       ; read GPT interrupt status flags
        ANDA    #$FE
        STAA    TFLG1       ; reset IC1 interrupt flag

        JSR     Update      ; update observer values
        JSR     VLoop       ; do velocity control loop calculations

WaitAD: BRCLR   ADSTAT,#$80,WaitAD ; wait for A/D conversion to finish

        JSR     DrivePE     ; set switch states in the inverter and chopper
        JSR     ErrCalc     ; calculate observer error
        JSR     SigChk      ; check integrity of incoming position data

        BRCLR   TFLG1,#$02,Disp ; skip IC2 handler if flag not set
        JSR     IC2         ; process Hall sensor edge

Disp:   JSR     Display     ; load queue and transfer to display board

IC1Done:PULM    D,E,X,Y,Z,K ; restore register contents
        RTI                 ; return from IC1 interrupt handler

*****   Subroutines   *****

        INCLUDE  '..\Subs\QueChk.SUB'
        INCLUDE  '..\Subs\HOChk.SUB'
        INCLUDE  '..\Subs\Update.SUB'
        INCLUDE  '..\Subs\Limit.SUB'
        INCLUDE  '..\Subs\VLoop.SUB'
        INCLUDE  '..\Subs\DrivePE.SUB'
        INCLUDE  '..\Subs\ErrCalc.SUB'
        INCLUDE  '..\Subs\SigChk.SUB'
        INCLUDE  '..\Subs\IC2.SUB'
        INCLUDE  'Display.SUB'


*
*       MIT
*       Lab. for Electromagnetic and Electronic Systems
```

```
*       Cambridge, MA
*
*       File Name : PLoop.ASM
*
*       Description : This program digitizes signals from each of the stator
*                     windings and locks a 2nd order observer to track the
*                     position and velocity of the vehicle. The vehicle
*                     follows a predefined position profile.
*
*       History : 6/28/94  Created.
*
*       Note : This program is written for the M68HC16Z1EVB.
*******************************************************************************

        nolist                      ;turn listing off for equates table
        INCLUDE '..\EQUATES.ASM'    ;table of EQUates for register addresses
        list                        ;turn listing back on after equates
        INCLUDE '..\Init\Vec.ASM'   ;initialize reset and interrupt vectors
        INCLUDE 'Var.ASM'           ;assign constants and variable storage

        ORG     $0200           ;start program after interrupt vectors and look-up
tables

*****   Initialization Routines *****
        INCLUDE '..\INIT\SYS.ASM'   ;initially set EK=F, XK=0, YK=F, ZK=1
                                    ;set sys clock at 16.78 MHz, disable COP
        INCLUDE '..\INIT\RAM.ASM'   ;initialize and turn on SRAM
                                    ;set stack (SK=F, SP=F3FE)
        INCLUDE '..\INIT\ADC.ASM'   ;configure A/D for 10-bit, 9 uSec conversion
        INCLUDE '..\INIT\QSM.ASM'   ;configure QSPI to interface with
                                    ;Power Electronics Module and display
        INCLUDE '..\INIT\GPT.ASM'   ;configure IC1 interrupt

*****   Start of user program area  *****

        LDD     #$8000
        TDP                         ; allow interrupts

Main:                               ; Main program loop...
        BRA     Main

*****   Interrupt handler routines  *****

IC1Int:                             ; IC1 interrupt handler routine
        PSHM    D,E,X,Y,Z,K         ; save register contents

        BRSET   GPTPDR,#$01,OddSet ; see if even or odd data set
        LDD     #$0020
        STD     ADCTL1              ; start A/D conversion on even data set
        LDY     #AGain              ; load IY for gain normalization base
        BRA     ResetIC1
OddSet: LDD     #$0024
        STD     ADCTL1              ; start A/D conversion on odd data set
        LDY     #XGain              ; load IY for gain normalization base
ResetIC1:
        LDAA    TFLG1               ; read GPT interrupt status flags
        ANDA    #$FE
        STAA    TFLG1               ; reset IC1 interrupt flag
```

```
        JSR     Update          ; update observer values
        JSR     PLoop           ; do position control loop calculations
        JSR     VLoop           ; do velocity control loop calculations

WaitAD: BRCLR   ADSTAT,#$80,WaitAD ; wait for A/D conversion to finish

        JSR     DrivePE         ; set switch states for the inverter and chopper
        JSR     ErrCalc         ; calculate observer error
        JSR     Display         ; load queue for transfer to display board

IC1Done:PULM    D,E,X,Y,Z,K     ; restore register contents
        RTI                     ; return from IC1 interrupt handler


*****   Subroutines   *****
        INCLUDE '..\Subs\Update.SUB'
        INCLUDE '..\Subs\Limit.SUB'
        INCLUDE '..\Subs\PLoop.SUB'
        INCLUDE '..\Subs\VLoop.SUB'
        INCLUDE '..\Subs\ErrCalc.SUB'
        INCLUDE '..\Subs\DrivePE.SUB'
        INCLUDE 'Display.SUB'


*
*       MIT
*       Lab. for Electromagnetic and Electronic Systems
*       Cambridge, MA
*
*       File Name : CubicP.ASM
*
*       Description : This program uses the position control loop from PLoop
*                     to force the vehicle to follow a position profile
*                     made up of piecewise cubic segments.  Each segment
*                     is one second in length, and is characterized by 4
*                     coefficients (e.g. PSet = At^3 + Bt^2 + Ct + D).  The
*                     coefficients are stored in a table.
*
*       History : 9/12/94  Created.
*
*       Note : This program is written for the M68HC16Z1EVB.
*******************************************************************************

        nolist                      ;turn listing off for equates table
        INCLUDE '..\EQUATES.ASM'    ;table of EQUates for register addresses
        list                        ;turn listing back on after equates
        INCLUDE '..\Init\Vec.ASM'   ;initialize reset and interrupt vectors
        INCLUDE 'Var.ASM'           ;assign constants and storage for variables

        ORG     $0200           ;start program after interrupt vectors

*****   Initialization Routines *****
        INCLUDE '..\INIT\SYS.ASM'   ;initially set EK=F, XK=0, YK=F, ZK=1
                                    ;set sys clock at 16.78 MHz, disable COP
        INCLUDE '..\INIT\RAM.ASM'   ;initialize and turn on SRAM
                                    ;set stack (SK=F, SP=F3FE)
        INCLUDE '..\INIT\ADC.ASM'   ;configure A/D for 10-bit, 9 uSec conversion
```

```
        INCLUDE '..\INIT\QSM.ASM'  ;configure QSPI to interface with
                                   ;Power Electronics Module and display
        INCLUDE '..\INIT\GPT.ASM'  ;configure IC1 interrupt

*****   Start of user program area  *****

        LDD    #$8000
        TDP                        ; allow interrupts

Main:                              ; Main program loop...
        BRA    Main

*****   Interrupt handler routines  *****

IC1Int:                            ; IC1 interrupt handler routine
        PSHM   D,E,X,Y,Z,K         ; save register contents

        BRSET  GPTPDR,#$01,OddSet ; see if even or odd data set
        LDD    #$0020
        STD    ADCTL1              ; start A/D conversion on even data set
        LDY    #AGain              ; load IY for gain normalization base
        BRA    ResetIC1
OddSet: LDD    #$0024
        STD    ADCTL1              ; start A/D conversion on odd data set
        LDY    #XGain              ; load IY for gain normalization base
ResetIC1:
        LDAA   TFLG1               ; read GPT interrupt status flags
        ANDA   #$FE
        STAA   TFLG1               ; reset IC1 interrupt flag

        JSR    Update              ; update observer values
        JSR    PSetCalc            ; calculate PSet value
        JSR    PLoop               ; do position control loop calculations
        JSR    VLoop               ; do velocity control loop calculations

WaitAD: BRCLR  ADSTAT,#$80,WaitAD ; wait for A/D conversion to finish

        JSR    DrivePE             ; set switch states for the inverter and chopper
        JSR    ErrCalc             ; calculate observer error

        BRCLR  TFLG1,#$02,Disp    ; skip IC2 handler ir flag not set
        JSR    IC2

Disp:   JSR    Display             ; load queue for transfer to display board

IC1Done:PULM   D,E,X,Y,Z,K         ; restore register contents
        RTI                        ; return from IC1 interrupt handler

*****   Subroutines   *****
        INCLUDE  '..\Subs\Update.SUB'
        INCLUDE  '..\Subs\Limit.SUB'
        INCLUDE  '..\Subs\PSetCalc.SUB'
        INCLUDE  '..\Subs\PLoop.SUB'
        INCLUDE  '..\Subs\VLoop.SUB'
        INCLUDE  '..\Subs\ErrCalc.SUB'
        INCLUDE  '..\Subs\DrivePE.SUB'
        INCLUDE  'Display.SUB'
        INCLUDE  '..\Subs\IC2.SUB'
```

```
*
*       MIT
*       Lab. for Electromagnetic and Electronic Systems
*       Cambridge, MA
*
*       File Name : Drag.ASM
*
*       Description : This program uses ILoop to drive the vehicle back and
*                     forth with a constant applied force, and collects data
*                     to provide a plot of acceleration (due to drag) vs.
*                     velocity.
*
*       History : 9/6/94  Created.
*
*       Note : This program is written for the M68HC16Z1EVB.
*******************************************************************************

        nolist                      ;turn listing off for equates table
        INCLUDE '..\EQUATES.ASM'    ;table of EQUates for register addresses
        list                        ;turn listing back on after equates
        INCLUDE '..\Init\Vec.ASM'   ;initialize reset and interrupt vectors
        INCLUDE 'Var.ASM'

        ORG     $0200           ;start program after interrupt vectors and look-up
tables

*****   Initialization Routines *****
        INCLUDE '..\INIT\SYS.ASM'   ;initially set EK=F, XK=0, YK=F, ZK=1
                                    ;set sys clock at 16.78 MHz, disable COP
        INCLUDE '..\INIT\RAM.ASM'   ;initialize and turn on SRAM
                                    ;set stack (SK=F, SP=F3FE)
        INCLUDE '..\INIT\ADC.ASM'   ;configure A/D for 10-bit, 9 uSec conversion
        INCLUDE '..\INIT\QSM.ASM'   ;configure QSPI to interface with
                                    ;Power Electronics Module and display
        INCLUDE '..\INIT\GPT.ASM'   ;configure IC1 interrupt

*****   Start of user program area  *****

        CLRE                    ; initialize external RAM data to 0's
        LDZ     #$FFFE
RAM0:   STE     0,Z
        AIZ     #-2
        BNE     RAM0
        STE     0,Z

        LDD     #$8000
        TDP                     ; allow interrupts

Main:                           ; Main program loop...
FWD:    LDE     Random          ; see if Position > 10 + 8*Random
        LDD     #8
        FMULS
        ADDE    #10
        SUBD    PosL
        SBCE    PosH
        BPL     REV             ; if not > 10.0000, then do nothing
        LDD     RefCrnt         ; if > 10.0000, then ISet => -RefCrnt
        NEGD
```

```
        STD     ISet
REV:    LDE     Random          ; see if Position < -10 - 8*Random
        LDD     #8
        FMULS
        ADDE    #10
        ADDD    PosL
        ADCE    PosH
        BPL     FWD             ; if not < -10.0000, then do nothing
        LDD     RefCrnt         ; if < -10.0000, then ISet => +RefCrnt
        STD     ISet
        BRA     Main

*****   Interrupt handler routines  *****

IC1Int:                         ; IC1 interrupt handler routine
        PSHM    D,E,X,Y,Z,K     ; save register contents

        BRSET   GPTPDR,#$01,OddSet ; see if even or odd data set
        LDD     #$0020
        STD     ADCTL1          ; start A/D conversion on even data set
        LDY     #AGain          ; load IY for gain normalization base
        BRA     ResetIC1
OddSet: LDD     #$0024
        STD     ADCTL1          ; start A/D conversion on odd data set
        LDY     #XGain          ; load IY for gain normalization base
ResetIC1:
        LDD     TFLG1           ; read GPT interrupt status flags
        ANDD    #$FEFF
        STD     TFLG1           ; reset IC1 interrupt flag

        JSR     Update          ; update observer values

WaitAD: BRCLR   ADSTAT,#$80,WaitAD ; wait for A/D conversion to finish

        JSR     DrivePE         ; set switch states for the inverter and chopper
        JSR     ErrCalc         ; calculate observer error
        JSR     LogData

        BRCLR   TFLG1,#$02,Disp ; skip IC2 handler if flag not set
        JSR     IC2             ; process Hall sensor edge

Disp:   JSR     Display

IC1Done:PULM    D,E,X,Y,Z,K     ; restore register contents
        RTI                     ; return from IC1 interrupt handler


*****   Subroutines   *****
        INCLUDE '..\Subs\Update.SUB'
        INCLUDE '..\Subs\Limit.SUB'
        INCLUDE '..\Subs\ErrCalc.SUB'
        INCLUDE '..\Subs\DrivePE.SUB'
        INCLUDE '..\Subs\LogData.SUB'
        INCLUDE '..\Subs\IC2.SUB'
        INCLUDE 'Display.SUB'
```

```
*
*       MIT
*       Lab. for Electromagnetic and Electronic Systems
*       Cambridge, MA
*
*       File Name : DataLog.ASM
*
*       Description : This program digitizes signals from each of the stator
*                     windings and accumulates (averages) these signals over
*                     4096 measurements.  The averaged data is placed in a
*                     block of RAM for up-loading to the computer.
*
*       History : 7/15/94  Created.
*
*       Note : This program is written for the M68HC16Z1EVB.
*******************************************************************************

        nolist                      ;turn listing off for equates table
        INCLUDE '..\EQUATES.ASM'    ;table of EQUates for register addresses
        list                        ;turn listing back on after equates
        INCLUDE '..\Init\Vec.ASM'   ;initialize reset and interrupt vectors

*****   Variables   *****

Position    EQU     $F000       ; Position of vehicle (in inches) -BCD format-
PosH        EQU     $F000       ; MSByte of position -BCD format-
PosL        EQU     $F001       ; LSByte of position -BCD format-
Counter     EQU     $F002       ; number of data points averaged
AValue      EQU     $F008       ; averaged phase A data (long-word)
BValue      EQU     $F00C       ; averaged phase B data
CValue      EQU     $F010       ; averaged phase C data
XValue      EQU     $F014       ; averaged phase X data
YValue      EQU     $F018       ; averaged phase Y data
ZValue      EQU     $F01C       ; averaged phase Z data

*****   Constants   *****

InitCnt     EQU     $2000       ; 2x number of data sets to average

        ORG     $0200           ;start program after interrupt vectors and look-up
tables

*****   Initialization Routines *****
        INCLUDE 'INITSYS.ASM'       ;initially set EK=F, XK=0, YK=F, ZK=1
                                    ;set sys clock at 16.78 MHz, disable COP
        INCLUDE '..\INIT\RAM.ASM'   ;initialize and turn on SRAM
                                    ;set stack (SK=F, SP=F3FE)
        INCLUDE '..\INIT\ADC.ASM'   ;configure A/D for 10-bit, 9 uSec conversion
        INCLUDE '..\INIT\GPT.ASM'   ;configure IC1 interrupt

*****   Start of user program area  *****

        LDD     #$0000
        STD     Position        ; set initial value for Position
        LDZ     #$0000          ; set initial RAM pointer value
        BGND                    ; wait for initial vehicle positioning

Init:   LDD     #InitCnt        ; initialize counter
```

```
        STD     Counter
        CLRE                        ; reset data accumulators...
        CLRD
        STED    AValue
        STED    BValue
        STED    CValue
        STED    XValue
        STED    YValue
        STED    ZValue

        LDD     #$8000
        TDP                         ; allow interrupts

Acquire:TSTW    Counter             ; see if counter = 0
        BNE     Acquire             ; keep acquiring data until counter = 0
        LDD     #$80E0              ; turn off interrupts
        TDP
Log:    LDD     AValue              ; round and log accumulator values in RAM...
        TSTW    AValue+2            ; set N bit if LSWord of AValue > 0.5
        BPL     StoreA              ; skip if N = 0
        ADDD    #1                  ; round MSWord
StoreA: STD     0,Z                 ; store in accumulator
        LDD     BValue
        TSTW    BValue+2
        BPL     StoreB
        ADDD    #1
StoreB: STD     2,Z
        LDD     CValue
        TSTW    CValue+2
        BPL     StoreC
        ADDD    #1
StoreC: STD     4,Z
        LDD     XValue
        TSTW    XValue+2
        BPL     StoreX
        ADDD    #1
StoreX: STD     6,Z
        LDD     YValue
        TSTW    YValue+2
        BPL     StoreY
        ADDD    #1
StoreY: STD     8,Z
        LDD     ZValue
        TSTW    ZValue+2
        BPL     StoreZ
        ADDD    #1
StoreZ: STD     10,Z

        AIZ     #12                 ; increment RAM pointer
        LDAA    PosL                ; increment position -BCD format- ...
        LDAB    PosH
        ADDA    #10                 ; (increment size is 10 mils)
        DAA
        ADCB    #0
        STAA    PosL
        TBA
        DAA
        STAA    PosH
```

```
        BGND                    ; wait for vehicle to be re-positioned
        JMP     Init

*****   Interrupt handler routines  *****

IC1Int:                         ; IC1 interrupt handler routine
        PSHM    D,E             ; save register contents

        BRSET   GPTPDR,#$01,OddSet ; see if even or odd data set
        LDD     #$0020
        STD     ADCTL1          ; start A/D conversion on even data set
        DECW    Counter         ; decrement counter
WaitE:  BRCLR   ADSTAT,#$80,WaitE ; wait for A/D conversion to finish
        LDED    AValue          ; add A/D results to accumulators...
        ADDD    LJURR0
        ADCE    #0
        SUBD    #$8000
        SBCE    #0
        STED    AValue
        LDED    BValue
        ADDD    LJURR1
        ADCE    #0
        SUBD    #$8000
        SBCE    #0
        STED    BValue
        LDED    CValue
        ADDD    LJURR2
        ADCE    #0
        SUBD    #$8000
        SBCE    #0
        STED    CValue
        BRA     ResetIC1

OddSet: LDD     #$0024
        STD     ADCTL1          ; start A/D conversion on odd data set
        DECW    Counter         ; decrement counter
WaitO:  BRCLR   ADSTAT,#$80,WaitO ; wait for A/D conversion to finish
        LDED    XValue          ; add A/D results to accumulators...
        ADDD    LJURR0
        ADCE    #0
        SUBD    #$8000
        SBCE    #0
        STED    XValue
        LDED    YValue
        ADDD    LJURR1
        ADCE    #0
        SUBD    #$8000
        SBCE    #0
        STED    YValue
        LDED    ZValue
        ADDD    LJURR2
        ADCE    #0
        SUBD    #$8000
        SBCE    #0
        STED    ZValue

ResetIC1:
```

```
        LDD     TFLG1           ; read GPT interrupt status flags
        ANDD    #$FEFF
        STD     TFLG1           ; reset IC1 interrupt flag

        PULM    D,E             ; restore D and E register contents
        RTI                     ; return from IC1 interrupt handler


*
*       MIT
*       Lab. for Electromagnetic and Electronic Systems
*       Cambridge, MA
*
*       File Name : Fourier.ASM
*
*       Description : This program uses the VLoop code to propel the vehicle
*                     back and forth about the start-up position, while
*                     integrating the incoming data to determine the Fourier
*                     coefficients of the DC and fundamental (Sin and Cos)
*                     components of each phase's position sensor data.
*
*       History : 9/2/94  Created.
*
*       Note : This program is written for the M68HC16Z1EVB.
*******************************************************************************

        nolist                      ;turn listing off for equates table
        INCLUDE '..\EQUATES.ASM'    ;table of EQUates for register addresses
        list                        ;turn listing back on after equates
        INCLUDE '..\Init\Vec.ASM'   ;initialize reset and interrupt vectors
        INCLUDE 'Var.ASM'           ;assign constants and variable storage

        ORG     $0200           ;start program after interrupt vectors

*****   Initialization Routines *****
        INCLUDE '..\INIT\SYS.ASM'   ;initially set EK=F, XK=0, YK=F, ZK=1
                                    ;set sys clock at 16.78 MHz, disable COP
        INCLUDE '..\INIT\RAM.ASM'   ;initialize and turn on SRAM
                                    ;set stack (SK=F, SP=F3FE)
        INCLUDE '..\INIT\ADC.ASM'   ;configure A/D for 10-bit, 9 uSec conversion
        INCLUDE '..\INIT\QSM.ASM'   ;configure QSPI to interface with
                                    ;Power Electronics Module and display
        INCLUDE '..\INIT\GPT.ASM'   ;configure IC1 interrupt

*****   Start of user program area  *****

        LDD     #$8000
        TDP                     ; allow interrupts

Main:                           ; Main program loop...
FWD:    LDD     PosH            ; get PosH value
        SUBD    #$001A          ; see if position > 26.0000
        BMI     REV             ; if not > 26.0000, then do nothing
        LDD     Speed           ; if > 26.0000, then VSet => -Speed
        NEGD
        STD     VSet
REV:    LDD     PosH            ; get PosH value
        ADDD    #$001B          ; see if position < -26.0000
```

```
        BPL     FWD             ; if not < -26.0000, then do nothing
        LDD     Speed           ; if < -26.0000, then VSet => +Speed
        STD     VSet
        BRA     Main

*****   Interrupt handler routines  *****

IC1Int:                         ; IC1 interrupt handler routine
        PSHM    D,E,X,Y,Z,K     ; save register contents

        BRSET   GPTPDR,#$01,OddSet ; see if even or odd data set
        LDD     #$0020
        STD     ADCTL1          ; start A/D conversion on even data set
        LDY     #AGain          ; load IY for ABC normalization base
        BRA     ResetIC1
OddSet: LDD     #$0024
        STD     ADCTL1          ; start A/D conversion on odd data set
        LDY     #XGain          ; load IY for XYZ normalizaticn base
ResetIC1:
        LDAA    TFLG1           ; read GPT interrupt status flags
        ANDA    #$FE
        STAA    TFLG1           ; reset IC1 interrupt flag

        JSR     Update          ; update observer values
        JSR     VLoop           ; do velocity control loop calculations

WaitAD: BRCLR   ADSTAT,#$80,WaitAD ; wait for A/D conversion to finish

        JSR     DrivePE         ; set switch states for the inverter and chopper
        JSR     ErrCalc         ; calculate observer error
        JSR     FourCalc

        BRCLR   TFLG1,#$02,Disp ; skip IC2 handler if flag not set
        JSR     IC2             ; process Hall sensor edge

Disp:   JSR     Display         ; load queue for transfer to display board

IC1Done:PULM    D,E,X,Y,Z,K     ; restore register contents
        RTI                     ; return from IC1 interrupt handler


*****   Subroutines   *****
        INCLUDE '..\Subs\Update.SUB'
        INCLUDE '..\Subs\Limit.SUB'
        INCLUDE '..\Subs\VLoop.SUB'
        INCLUDE '..\Subs\ErrCalc.SUB'
        INCLUDE '..\Subs\DrivePE.SUB'
        INCLUDE '..\Subs\Fourier.SUB'
        INCLUDE 'Display.SUB'
        INCLUDE '..\Subs\IC2.SUB'
```

tables

```
*
*       Title : Table.ASM
*       Description : This file contains look-up tables for the power electronics
*                     controller (inverter switch states to commutate the motor)
*                     and for the observer error calculation (weighting function)
*
*************************************************************************

TablePE:      ORG       $6C00         ;PEController look-up table base
                                      ;table has 1024 1-byte entries
                                      ;for 12 switch state combinations

              DCB.B     85,45         ;85 entries, switch state = 0100 0101
              DCB.B     85,65         ;85 entries, switch state = 0110 0101
              DCB.B     86,64         ;86 entries, switch state = 0110 0100
              DCB.B     85,24         ;85 entries, switch state = 0010 0100
              DCB.B     85,26         ;85 entries, switch state = 0010 0110
              DCB.B     86,36         ;86 entries, switch state = 0011 0110
              DCB.B     85,32         ;85 entries, switch state = 0011 0010
              DCB.B     85,12         ;85 entries, switch state = 0001 0010
              DCB.B     86,13         ;86 entries, switch state = 0001 0011
              DCB.B     85,53         ;85 entries, switch state = 0101 0011
              DCB.B     85,51         ;85 entries, switch state = 0101 0001
              DCB.B     86,41         ;86 entries, switch state = 0100 0001

TableFx:      ORG       $7000         ;F(x) look-up table base
                                      ;table has 16384 2-byte entries
                                      ;F(x) = -0.625*Sin(x*2pi/16384)

              DC.W      $0000         ;1st entry = -0.625*Sin(0)
              DC.W      $FFF8         ;2nd entry = -0.625*Sin(2pi/16384)
              DC.W      $FFF0         ;3rd entry = -0.625*Sin(4pi/16384)
              ...                     ;( many entries not shown)
              DC.W      $0008         ;last entry = -0.625*Sin(32766pi/16384)
```

References

[1] J. Vranich, "Supertrains: Solutions to America's Transportation Gridlock", St Martin's Press, 1991.

[2] H. Weh, A. Steingröver, and H. Hupe, "MagLev Transportation with Controlled Permanent Magnets and Linear Synchronous Motors", 13th International Conf. on MagLev Systems and Linear Drives, 1993.

[3] "Compendium of Executive Summaries from the Maglev System Concept Definition Final Reports", DOT/FRA/NMI-93/02.

[4] R.D. Thornton, D. Perreault, and T. Clark, "Linear Synchronous Motors for MagLev", DOT/FRA/NMI-92/13, Jan. 1993.

[5] R.S. Phelan, "High Performance Maglev Guideway Design", PhD Thesis, Jan. 1993. MIT, Dept. of Civil and Environmental Engineering, Cambridge, MA.

[6] J.K. Roberge, "Operational Amplifiers: Theory and Practice", John Wiley and Sons, 1975.

[7] K. Peters, "Design of a Guideway for a Model MAGLEV Vehicle", SB Thesis, Feb. 1993, MIT, Dept. of M.E., Cambridge, MA.

[8] J.E. Anderson, "Transit Systems Theory", Lexington Books, 1978

[9] W.S. Brown, "A 1/25 Scale Magneplane", PhD Thesis, Oct. 1975, MIT Dept. of Elec. Eng., Cambridge, MA.

[10] S. Kuntz, P.E. Burke, and G.R. Slemon, "Active Damping of MagLev Vehicles Using Superconducting Linear Synchronous Motors", Electric Machines and Electromechanics, vol. 2, 1978.

[11] S. Lingaya, and C.P. Parsch, "Characteristics of the Force Components of an Air Cored Linear Synchronous Motor with Superconducting Excitation Magnets", Electric Machines and Electromechanics, vol. 4, 1979.

[12] R.D. Thornton, "Linear Synchronous Motor Design", 13th International Conf. on MagLev Systems and Linear Drives, 1993.

[13] J.L. Kirtley Jr., "Air-Core Armature Shape: A Comparison of Helical and Straight-With-End-Turns Windings", Proc. of SM100, Zurich, Switzerland, Aug 27-29, 1991.

[14] T.S. Perry, "Today's View of Magnetic Fields", IEEE Spectrum, Dec. 1994.

[15] New England Electric Wire Corporation catalog, Lisbon, N.H., tel. (603) 838-6624.

[16] W. Leonhard, "Control of Electrical Drives", Springer-Verlag, 1985.

[17] G.C. Verghese, J.H. Lang, and L.F. Casey, "Analysis of Instability in Electrical Machines", IEEE Trans. on Industry Applications, vol. IA-22 #5, 1986.

[18] H.S. Patel, and R.G. Hoft, "Generalized Techniques of Harmonic Elimination and Voltage Control in Thyristor Inverters: Part I - Harmonic Elimination", IEEE Trans. on Industry Applications, vol. IA–9 #3, 1973.

[19] H.S. Patel, and R.G. Hoft, "Generalized Techniques of Harmonic Elimination and Voltage Control in Thyristor Inverters: Part II - Voltage Control Techniques", IEEE Trans. on Industry Applications, vol. IA–10 #5, 1974.

[20] B.D. Bedford, and R.G. Hoft, "Principles of Inverter Circuits", John Wiley & Sons, 1964.

[21] A. Lumsdaine, and J. Lang, "State Observers for Variable Reluctance Motors", IEEE Trans. on Industrial Electronics, vol. 37 #2, 1990.

[22] L.A. Jones, and J. Lang, "A State Observer for the Permanent-Magnet Synchronous Motor", IEEE Trans. on Industrial Electronics, vol. 36 #3, 1989.

[23] D.G. Luenberger, "Introduction to Dynamic Systems", John Wiley & Sons, 1979.

[24] F.C. Schweppe, "Uncertain Dynamic Systems",Prentice-Hall, 1973.

[25] F.M. Gardner, "Phaselock Techniques", John Wiley & Sons, 1979.

[26] Motorola, "M68HC16Z1EVB User's Manual", Imperial Litho, 1992

[27] Motorola, "M68HC16 Reference Manual", Prentice-Hall, 1989.

[28] B.K. Bose, "Adjustable Speed AC Drives - A Technology Status Review", Proc. of the IEEE, vol. 70 #2, 1982.

[29] D.J. Perreault and R.D. Thornton, "Power Electronics for Linear Synchronous Motor Propulsion Systems", 13th International Conf. on MagLev Systems and Linear Drives, 1993.

[30] J. Kassakian, M. Schlecht, and G. Verghese, "Principles of Power Electronics", Addison Wesley, 1990.