A Novel Position-Sensing, Control, and Communication System for Automated Transportation

by

Brian Michael Perreault

B.S., Boston University (1991)
S.M., Massachusetts Institute of Technology (1993)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May, 1997

Author _____
Department of Electrical Engineering and Computer Science
May 30th, 1997

Certified by _____
Professor Richard D. Thornton
Thesis Supervisor

Accepted by _____
Professor Arthur C. Smith
Chairman, Committee on Graduate Students
Department of Electrical Engineering and Computer Science

A Novel Position-Sensing, Control, and Communication System for Automated Transportation

by

Brian Michael Perreault

Submitted to the Department of
Electrical Engineering and Computer Science
on May 30th, 1997, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Electrical Engineering

Abstract

A necessary component of any automated transportation system is an accurate and reliable position sensing mechanism. While several techniques exist to implement such a component, few provide the capability for inter-vehicle position sensing; this capability is desirable for very short headway systems such as Personal Rapid Transit (PRT) and baggage-handling systems, whereby vehicles are better able to regulate inter-vehicle spacing and protect against vehicle collisions. Communication between a vehicle and the wayside is also a desirable trait to have in a transportation system in order to transmit such information as destination, contents, component failures, new routes, and passenger emergencies.

A mechanism has been developed and implemented to track the relative distance between vehicles by means of the detection of relative phase between two points on a multi-phase winding. Such a winding may be a linear motor propulsion winding, utilized for this alternate purpose, or an inexpensive winding designed primarily for position-sensing and communication purposes. The objective is achieved through the use of a transmitter to inject a signal in the winding and a sensor to detect that signal. With an initial value of distance between the two points, the relative position of the two points may be accurately tracked.

Similar methods have been developed, with the addition of a reference signal generated by a wayside controller, to enable each sensor to track its own absolute position. A related technique has also been implemented to allow a sensor to track a moving, virtual reference point on the winding; such a function is useful for transportation systems which utilize point-following strategies. Digital communication has also been achieved over the same signal path through the multi-phase winding, with no additional hardware at a rate of 1500 bits per second. Alternate communications techniques have been investigated to increase the communication rate with an associated increase in complexity. The algorithms to perform these functions have been implemented with a digital signal processor and demonstrated over the winding of an existing linear synchronous motor.

Thesis Supervisor: Richard D. Thornton
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

First and foremost, I would like to thank my thesis advisor, Professor Richard Thornton. His support and advice through my graduate years have been invaluable. He is level-minded and sharp and knows how to enjoy life. He sets a great example to follow.

I would also like to acknowledge the contributions of my thesis readers. Professor Jeff Lang gave me some great ideas for measurements. Professor Steve Leeb passed down to me some great practical knowledge. I would like to thank them both for their tremendously quick effort in reviewing my thesis so I could graduate in June.

I would like to acknowledge both my brother, David J. Perreault, and my colleague Tracy Clark. They were great people to bounce ideas off of and get feedback from. My brother, in particular, was especially helpful in debugging hardware, and was always happy to lend a third hand when I needed it.

I would like to acknowledge George Anagnostopoulos, of the U.S. Department of Transportation. He was a terrific source of knowledge in transportation systems, and was a great source of references.

I would also like to thank the rest of the faculty, staff, and students at the Laboratory for Electromagnetic and Electronic Systems (L.E.E.S.) for their great discussions and for their friendship. I will truly miss the people and the friendly environment in the lab. It was a great place to be!

I would like to thank the National Science Foundation for their fellowship support during my Master's work at MIT and early Doctoral work. I would also like to thank the U.S. Department of Transportation for their partial support of my final year at M.I.T. under the Eisenhower Fellowship. These fellowships have helped free me to follow the path of my choice, and unburden me from financial worries. I would like to acknowledge Professor Thornton, Professor Al Drake, George Anagnostopoulos, and the Director of the L.E.E.S., Professor John Kassakian, for their help in acquiring the Eisenhower Fellowship and for their overall friendliness.

I would like to thank my parents for their steadfast love and support. They have always encouraged me to reach for my fullest potential, and opened up my horizons. I would especially like to thank my father, who has inspired me all my life.

Finally, I would like to thank my wife Maggie for her love, patience, and support. She has kept me organized, and encouraged me throughout development of this thesis. My life would not be complete without her.

For my Family,

and especially my wife Maggie,

whom I love dearly.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As the world enters the twenty-first century, new concepts and capabilities in transportation systems promise a new level of service and convenience. Magnetic levitation (maglev) transportation systems promise fast, convenient transit between cities. Personal Rapid Transit (PRT) [27], as described later in this chapter, promises higher levels of convenience and shorter transit times than even an automobile is able to deliver within a city. Automated highway systems promise more capacity and less congestion on existing highways. In order to implement these new systems, some basic position-sensing and communication technologies are necessary. It is the goal of this thesis to provide several components of these technologies in an inexpensive, integrated system.

A mechanism has been developed and implemented to track the relative distance between vehicles by means of the detection of relative phase between two points on a multi-phase winding. Such a winding may be a linear motor propulsion winding, utilized for this alternate purpose, or an inexpensive winding designed primarily for position-sensing and communication purposes. The objective is achieved through the use of a transmitter to inject a signal in the winding and a sensor to detect that signal. With an initial value of distance between the two points, the relative position of the two points may be accurately tracked.

Similar methods have been developed, with the addition of a reference signal generated by a wayside controller, to enable each sensor to track its own absolute position. A related technique has also been implemented to allow a sensor to track a moving, virtual reference point on the winding; such a function is useful for transportation systems which utilize point-following strategies. Digital communication has also been achieved over the same signal path through the multi-phase winding, with no additional hardware at a rate of 1500 bits per second. Alternate communications techniques have been investigated to increase the communication rate with an associated increase in complexity. The algorithms to perform these functions have been implemented with a digital signal processor and demonstrated over the winding of an existing linear synchronous motor.

1.1 Background and Motivation

Advances in transportation include automated systems such as Personal Rapid Transit (PRT) [27] and automated baggage handling systems [29]. These systems depend on accurate position information and reliable control systems for safe operation. Improved performance in existing, conventional systems may also be achieved through more accurate position information, and new conventional systems will benefit from the inexpensive, integrated techniques developed in this thesis.

1.1.1 Personal Rapid Transit

A woman at a bus stop near Boston was recently quoted as saying, "Mass transportation is convenient if you're not in a hurry." This statement summarizes many of the shortcomings of conventional mass transit systems. Most existing systems have relatively long waiting times, and even longer transit times. An advanced concept known as Personal Rapid Transit was developed to rectify these shortcomings. PRT enables short waiting times and drastically reduced transit times.

The concept of PRT may be described as an automated taxi system. The system is comprised of a guideway with an exclusive right of way, and small vehicles with a capacity of three or four passengers. The small, light vehicles enable an unobtrusive, inexpensive, elevated guideway to be erected above ground; underground systems require considerably more expense.

The guideways are inexpensive as compared with conventional light rail systems since PRT employs small, light vehicles. Due to the fact, the guideways may be laid out in a large grid as shown in Figure 1.1. The dots on the grid indicate interchanges between the lines. Assuming that there is a station every half mile along each line, a passenger, on average, needs to walk less than a quarter mile to arrive at a station, a significantly shorter distance than for a conventional system.

The vehicles are designed to hold up to three or four people, so that each person or group may travel in their own vehicle, enhancing the safety of travel. With such small vehicles, thousands may be available in a given network. In theory, when a passenger arrives at a station, there is often an empty vehicle waiting to be used. If not, an empty vehicle may be quickly routed to the station. A possible goal of such a system is to have a waiting time of less than one minute for most passengers at most times during the day.

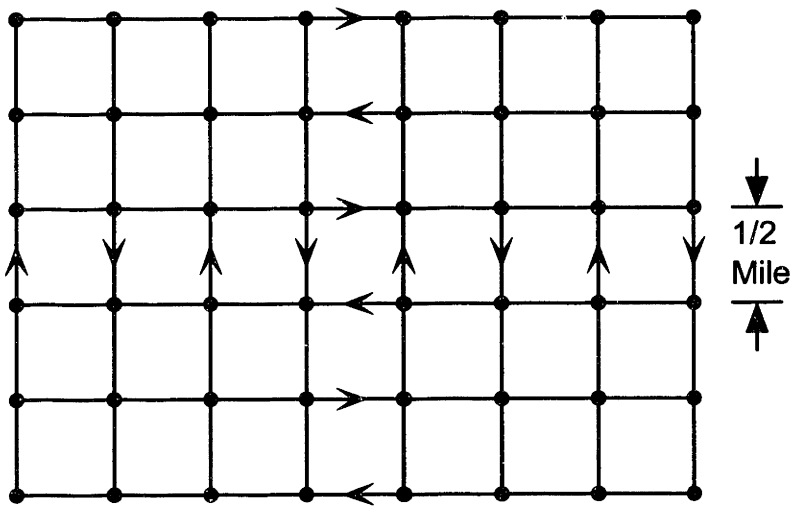

Figure 1.1 PRT Guideway Network

The stations in a PRT system are designed to be off the main guideway. Thus, a vehicle must take an "exit" to get to a station, and an "on-ramp" to return to the main guideway. Thus, like a highway, the vehicles during their course of travel do not have to stop at every station along the way. In conventional systems, stop times at stations often occupy the largest portion of overall travel time. Thus, a PRT system has much shorter travel times than conventional mass transit systems. The vehicles in a PRT system are also designed to change lines for the passenger, thus eliminating any delay that would normally occur in changing lines or modes and

thus vehicles in a conventional system. Finally, a PRT system is entirely automated, eliminating the substantial cost of utilizing a driver for every vehicle.

Clearly, PRT has many advantages over a conventional mass transit system. One of the limitations of the system is in its small vehicle size. Conventional mass transit systems utilize vehicles or entrainments that can carry hundreds of people, and thus have a fairly high line capacity. This limitation is partially offset by the less expensive nature of the PRT guideways. In order to make the capacities of the systems even more equitable, PRT systems are designed to run with very short headways - on the order of one or one half second [22]. While such a practice compensates for the small vehicle size, it places a much larger responsibility on the operational and safety control systems to provide safe, efficient travel.

In order to fully automate such systems, an accurate position-sensing mechanism is needed for these operational and safety purposes. With headways on the order of a second, it is imperative to have accurate, up to date position information to prevent vehicle collisions. Although position-sensing is not a new concept, as such systems have been in use for over a century in the form of track circuits for long headway systems, the required level of accuracy is much higher.

Many solutions have been utilized or proposed for automated and semi-automated systems as shown in Table 1, each with characteristic trade-offs, advantages, and disadvantages. In the case of PRT, where headways between vehicles are less than two seconds, accurate position sensing is absolutely necessary for safe operation. In such a system, it is desirable, and some conservative transportation professionals say absolutely necessary, for each vehicle to directly sense the relative position of other vehicles on the guideway for vehicle protection purposes. It is this basic requirement that drove the development of the advancements in this thesis.

METHOD	ADVANTAGES	DISADVANTAGES	RESOLUTION	SYSTEMS	HEADWAY	COMMENTS
Track Circuit	Simple Fail Safe Operation	Very low resolution Fixed-Block System Steel wheels on rail	> 100 m	BART, AIRTRANS, Railroad, etc.	150 s 18 sec	DC, AC, Audio Freq. Century Old Technology
Induction Loops	May integrate Low speed communication. Moving-block operation possible.	Low Resolution	25-50 m 0.5-2 m	VAL, SELTRAC, Proposed Japanese CVS	≥ 1 min.	Loop Counting must be performed
GPS	System already in place Maintenance by US Government	Low Resolution Unavailable Underground & certain geography Dependent on external systems	> 20 m			May be used with beacons for better accuracy
Odometer	Accurate	Error accumulates Wheel slip problems	< 1 m	ATCS Proposed M-Bahn		May be used with beacons for better accuracy
Vehicle Detection Sensors or R/F ID tags	Simple, Inexpensive Short ID Codes Possible	Only detect vehicles in at certain points along a track	(spacing)	MBTA Green Line, VAL DIA Baggage Handler	> 2 min. ~ 1 min.	
Spread Spectrum Digital Radio	Integrated communication and Position Sensing	Expensive	5 m	Proposed to upgrade BART	80 sec.	Transceivers must be positioned every 1/4 mile
Leaky Waveguide	Position relative to other vehicles may be determined accurately	Expensive Limited Range RF Electronics required Affected by Weather	< 1 m	Proposed Chicago RTA PRT	3 sec.	Uses radar technology Also used for FM radio communication in BART
Beacons/Markers	Simple, Inexpensive	Vehicles only detect position at markers	(spacing)	TGV, ITS	3 min < 2 sec	Ultrasonic, Optical, Magnetic
Short Range Radar	Position relative to neighboring vehicles may be determined Inexpensive	Only useful for nearest vehicle No integrated communication Does not work around curves	< 1m	ITS Highway systems	< 2 sec	
Discrete Circuit Transmission Line	Relative Position Sensing	Gap distance extremely critical Expensive to build Accuracy varies with distance	1m	CABINTAXI prototype		Signal decays exponentially with distance from source
Linear Motor Helical Winding with Position Sensing	Extremely Accurate Integrated with propulsion system	Only 1 vehicle per stationary section	< 0.1° (< 1 cm)	PRT MagLev Retrofits	< 2 sec	May be from GRT to Mag-Lev

Table 1.1 Position sensing systems used in or proposed for transportation

1.1.2 Conventional Systems

Advances in existing systems may also be derived from better system features. Many mass transit systems are currently operating at maximum designed capacity during peak usage hours. One such example is the Bay Area Rapid Transit (BART) system in San Francisco. With the existing track circuit-based system, the minimum headway is approximately two and one half minutes. With more accurate knowledge than is currently available from such a system, the minimum headway of the system may be cut by a factor of two or more [28], and still operate according to a safe headway criteria [8, 9]. Thus, line capacity could be doubled or tripled with only a very minute increase in guideway infrastructure! With the high cost of putting new lines in an underground system, such an alternative is very economically attractive. Advanced, accurate position-sensing features can thus lead to more efficient use of existing infrastructure in an economically feasible manner.

1.1.3 Automated Highway Systems

There has been much activity recently in the field of automated highway systems and Intelligent Transportation Systems (ITS) for highways (previously known as Intelligent Vehicle Highway Systems (IVHS)) [30]. With current automobile headways on existing, non-automated highways as low as one second, the concept of automating a highway with even shorter headways produces many of the same requirements on a control system as does PRT. It is extremely important, as any driver knows, to know the positions of surrounding vehicles in order to operate a vehicle in a safe manner. These same demands are placed on the control system for the vehicle, if operation is to be automated. Again, there is the requirement to know, on board the vehicle, the relative positions and movements of surrounding vehicles. Thus, some sort of mechanism must be in place to track these positions and movements on board the vehicle.

1.1.4 Relevant Technologies

While such a relative position-sensing mechanism may be implemented indirectly through an absolute position sensing mechanism and a communication system, it is

desirable to use an independent, inter-vehicle position sensing system on board the vehicles for safety and reliability reasons. Only three of the systems listed in Table 1 are capable of direct, relative position-sensing.

The leaky waveguide implementation of a relative position-sensing mechanism has been proposed for use in the Chicago Regional Transit Authority (RTA) Personal Rapid Transit (PRT) system as developed by Raytheon Corporation. This technique is very similar to RADAR within a waveguide, where a radio frequency signal is introduced in the waveguide and distance to the preceding vehicle is measured by the time delay of the echo. The waveguide is an extruded structure with a 'U' shaped cross-section. Radio frequency signals are introduced into the waveguide through the opening of the 'U'. The waveguide is designed in such a manner that the radio frequency signals are constrained to travel, for the most part, within the waveguide. Since the waveguide is open on one side to allow these signals to be introduced, the signal strength decays with distance as some of the signal 'leaks' out of this opening. Such a system is fairly expensive to implement, in terms of both the waveguide and the radio frequency transmitter and receiver.

In order to acquire the range necessary for even the short headways of PRT systems, a fairly expensive waveguide is necessary. Range is fairly limited, and is affected by weather conditions, limiting the usefulness of such a technique since the system must be designed for worst case scenarios. Although distance is fairly accurately measured, this technique has been implemented only to measure the distance to the preceding vehicle.

Another system capable of relative position sensing is the discrete circuit transmission line. This system was implemented in the CABINTAXI prototype [4]. The transmission line consists of three parallel conductor cables bridged by discrete resistors at 10 cm intervals in a ladder-type network. In this technique, each vehicle injects a 100 kHz sinusoidal signal into the transmission line through an antenna. The signal decays exponentially in the transmission line according to distance from the source. A following vehicle can detect the signal and extract approximate distance from the signal magnitude.

This system suffers to a small degree from component drift, and to a larger degree from variations in gap distance and track irregularities. With a slightly larger gap between antenna and transmission line, the magnitude of the signal in the line drops. Thus, a following vehicle would detect a lower level signal and determine that it is farther away from the preceding vehicle than it actually is! The accuracy of the system also degrades with increasing distances between vehicles, since the signal decays exponentially. The transmission line is fairly expensive to build due to its discrete nature and the inclusion of ferrite slugs in the guideway every 2.5 cm to improve performance. Again, this system only allows distance to the preceding vehicle to be detected.

The last system capable of relative position sensing is that of low cost vehicle RADAR [31]. Such a system utilizes a low power radar transceiver to detect the position of other nearby vehicles or radar reflectors on those vehicles. In order to hold costs in check in some proposals, no high power RF amplifiers are used and thus only low power signals are utilized; range is very limited in this system. This system is also limited to line of sight measurements, and thus range is very limited on guideway curves. This system had been proposed for use in warning systems on automobiles as well as Group Rapid Transit (GRT) systems. Since this type of radar system is not yet in mass production, the cost is still fairly prohibitive.

Each of the three preceding systems could be successfully implemented in a PRT or other automated system, despite their disadvantages. Thus, any new system should have advantages over current systems in terms of cost, accuracy, or features. The techniques described in this thesis have advantages in all of these categories.

Several new transportation systems have been proposed or implemented utilizing linear motors, due to inherent advantages of the linear motor for certain applications. Such advantages include lower vehicle weight and fewer moving parts. Additionally, higher propulsion forces may be implemented where needed, such as on inclines or acceleration stretches. The system developed in this thesis would be especially appropriate for these systems due to possible integration with the linear motor winding. It should be noted, however, that such a position-sensing system may be implemented without a linear motor - a useful approach for upgrading older systems.

Some of the techniques in this thesis are based on and devised to be utilized in accordance with previous developments at MIT [1]. The initial research investigated a highly accurate method to measure the position of a transmitter on a linear motor winding. The transmitter injects a sinusoidal signal into the winding which is detected in each of the winding phases at the wayside. Utilizing mostly analog and some digital signal processing techniques, the phase of the transmitter may be determined and tracked. A magnetic sensor in the guideway indicates when a vehicle passes a specific location, and exact location may then be tracked.

The techniques described in this thesis extend the capabilities of this system utilizing discrete-time techniques. These features enabled include inter-vehicle position sensing, autonomous position-sensing onboard the vehicle (the vehicle is able to track its own position), communication, and virtual marker tracking. Together with the previous work performed at MIT, these advances form a set of core capabilities useful and necessary for the implementation of an automated transportation system.

1.2 Thesis Objectives and Contributions

This thesis sought to accomplish two major objectives, and several minor objectives were accomplished in the process. The approach taken to accomplish these objectives was based upon work performed at MIT which enabled the tracking of the position of a vehicle at a wayside location [1]. This objective in this thesis is to extend the capabilities of this system with new features in an integrated design. The first primary objective was to develop a vehicle-based position-sensing mechanism capable of detecting and tracking the relative positions of other vehicles. This objective was accomplished through the use of a winding as a communication medium, and utilization of related sensors and transmitters. The technique was implemented on a demonstration system and shown to have an absolute accuracy of better than 0.5 mm.

The demonstration system is based on the linear motor winding built for the position-sensing system developed in [1]. This winding had been previously utilized to demonstrate the detection and tracking of a vehicle on the winding from wayside receivers. A vehicle transmitter and six wayside sensors were utilized to perform this

task. These components were utilized along with receivers on board the vehicles and transmitters on the wayside to implement the techniques described in this thesis. A helical winding used as the stator for the linear synchronous motor was utilized for this purpose. The signal processing for this new technique was implemented, in its entirety, in discrete-time on a digital signal processor.

A related technique was also developed to enable the vehicle to determine its own position relative to the wayside, and was again implemented on the demonstration system. With proper adjustment, this technique is shown to obtain a resolving accuracy of approximately ten microns. While this level of accuracy was not originally a major objective of this thesis, the accuracy of this system opens its use up to a much broader range of applications than transportation systems alone.

Another related technique is used to enable the tracking, by a vehicle, of a moving, virtual point along the guideway. The position of the virtual point is specified by a wayside system. This feature is very useful for transportation systems which utilize a point-following concept for vehicle control. Such a feature will also be shown to be very useful with a certain type of linear motor, as discussed in Chapter 4.

The second primary objective of this thesis was to prove that digital communication could be accomplished over the same channels utilized for position sensing. Indeed, two separate approaches to communication in the system are developed. One of the two approaches was implemented in the demonstration system at rate of 1500 bits per second, and shown to be robust to vehicle movement. The demonstration system implemented both position-sensing and communication features simultaneously, illustrating a typical use of the features. Methods for increasing the communication rate of this feature are also developed.

Finally, an approach is developed to enable the position-sensing and communication techniques on an inexpensive, single phase winding. Communication and position-sensing techniques have been implemented simultaneously on the demonstration system, showing the level of integration possible with the approach in this thesis. The sensor and transmitter components of the demonstration system were built from standard ferrite cores. An inexpensive digital signal processor (DSP) was utilized

for the processing system. The winding was created from standard litz wire by machine. Thus, the basic components of this system are readily available and inexpensive as compared to the systems described in Section 1.1.4; a system based upon the techniques of this thesis can be implemented in a very economical manner. The cost, integration, and performance of the techniques developed in this thesis create an attractive approach to position-sensing and communication in a transportation setting.

1.3 Thesis Organization

The necessity for an inexpensive, integrated system to provide advanced position-sensing, communication and control features is clear. The architecture of a system developed in this thesis to provide such features is investigated in Chapter 2. This architecture is based upon a winding which provides a suitable signal path between the wayside and vehicles and between vehicles. The structure of the winding is designed such that position information may be extracted from the effect of this winding on a signal between two points in the system. The transmitters and sensors necessary to implement such features are described in detail in this chapter.

Chapter 3 illustrates the necessary algorithms for determining position information from received signals. The effect of the signal path upon signals between vehicles is investigated. With this knowledge, a technique is developed to determine and track the distance between vehicles, a very useful feature for short headway systems. A similar technique is created to allow vehicles to track their own position relative to a guideway with extremely high accuracy. Finally, to enable a feature known as "point-following" in the transportation literature [2, 5], a mechanism has been developed to allow a vehicle to track a virtual, moving point on the guideway, as specified by a wayside system. This technique is also very useful for implementing systems utilizing a doubly-excited linear motor (DELM). The accuracy of these techniques is measured and improvements through small changes in the algorithms are described. The noise immunity of these techniques is also investigated in this chapter.

A digital communication channel between vehicles and the wayside is a necessary and useful feature for a transportation system. Chapter 4 develops techniques to

implement a digital channel in this system with no additional hardware beyond that used for position-sensing. The position-varying nature of the available signal paths is investigated, and two approaches to communication over these paths are developed. The first method utilizes a scheme to create a position-independent path from two or more position-dependent paths, by means of a synchronous rectification approach which depends on knowledge of vehicle position. The second approach was developed in order to create a system which is not dependent on knowledge of position and can thus act independently of the position-sensing system. An implementation on a demonstration system is illustrated in this chapter based on frequency shift keying (FSK) modulation at a rate of 1500 bits per second. Extensions to other modulation techniques are also described.

Chapter 5 describes the implementation of the techniques developed in Chapters 3 and 4 using alternative architectures. These alternate architectures include windings with a varying number of phases, and the option of utilizing fewer sensors and more transmitters. Finally, in this chapter, an implementation is described where a simplified winding may be used, containing only a single phase. Such an approach is useful for relatively long headway transit systems, where it may be much more cost effective to use a simplified, less expensive winding at the expense of more complex signal processing on board the vehicle.

Chapter 6 addresses several implementation issues including the accommodation of several vehicles on a single winding. This chapter also tackles the issue of acquiring initial position estimates as well as regular updates of position for verification purposes. Several approaches to acquiring these updates are described. The application of the features described in this thesis to different possible transportation scenarios concludes this chapter.

The features described in this thesis and their benefits are summarized in Chapter 7. The suitability of these features to various applications in transportation is shown and contrasted with other approaches. Finally, conclusions about the usefulness of the techniques developed in this thesis for advanced transportation systems will be made.

Chapter 2

Architecture

2.1 Overview

Several useful position-sensing, control, and communications mechanisms may be implemented utilizing an architecture with a multi-phase winding and appropriate sensors and transmitters, as illustrated in Figure 2.1 and Figure 2.2. Position-sensing mechanisms include detection of position on a winding and detection of position relative to a transmitter on the winding. Control mechanisms include the implementation of moving virtual markers which vehicles may detect. With appropriate signal processing, communication may also be achieved over the winding.

The winding is central to the architecture and provides the signal paths used to implement the desired functions. This winding would typically be laid in the middle of a guideway underneath a vehicle, although in the implementation demonstrated in this thesis, the vehicle wraps around the guideway and winding. This winding may be an independent winding or a winding in a linear or rotary motor whose primary purpose is propulsion. The transmitters on the wayside and on the vehicle are utilized to induce signals in the winding. The sensors are used to detect these signals introduced into the winding.

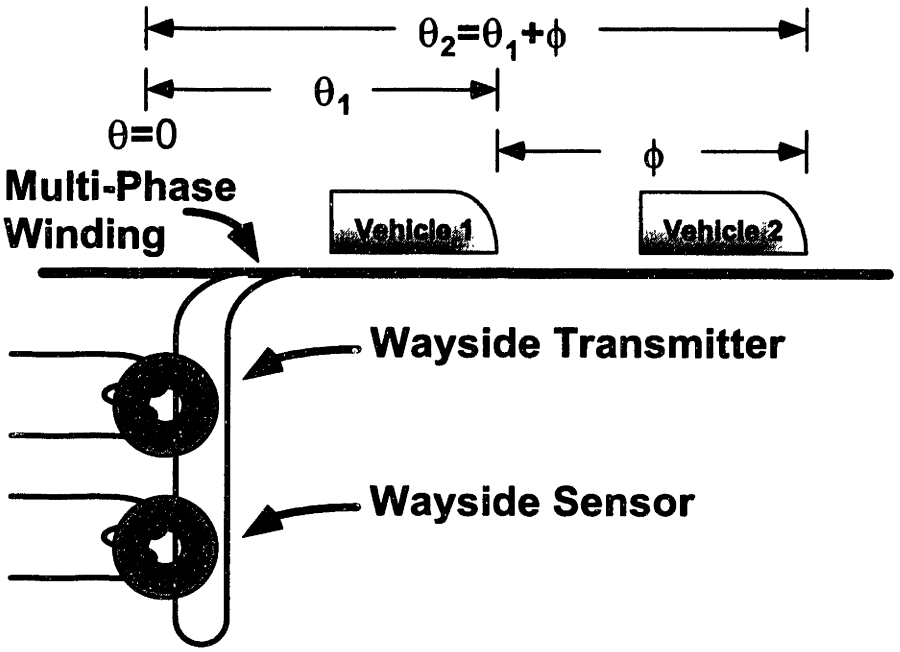

Figure 2.1 Overview of Position-Sensing and Communication System

2.2 Winding

A wide variety of winding types may be utilized for the signal path in this architecture. The basic structural requirement is a periodic pattern of the phases within the winding. The techniques described in this thesis may be implemented on such a winding with two or more regularly spaced phases (this requirement will be dropped in Chapter 5) connected in long loops. It should be noted that it is necessary to tailor the shape of the vehicle transmitters and sensors to that of the winding in order to obtain a compatible system.

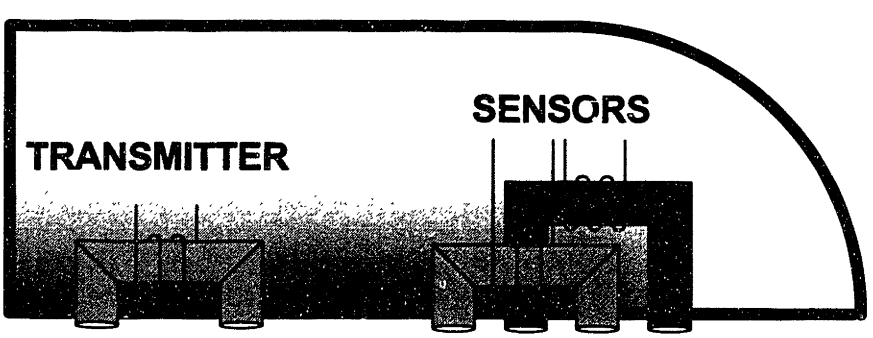

Figure 2.2 Vehicle Sensors and Transmitters

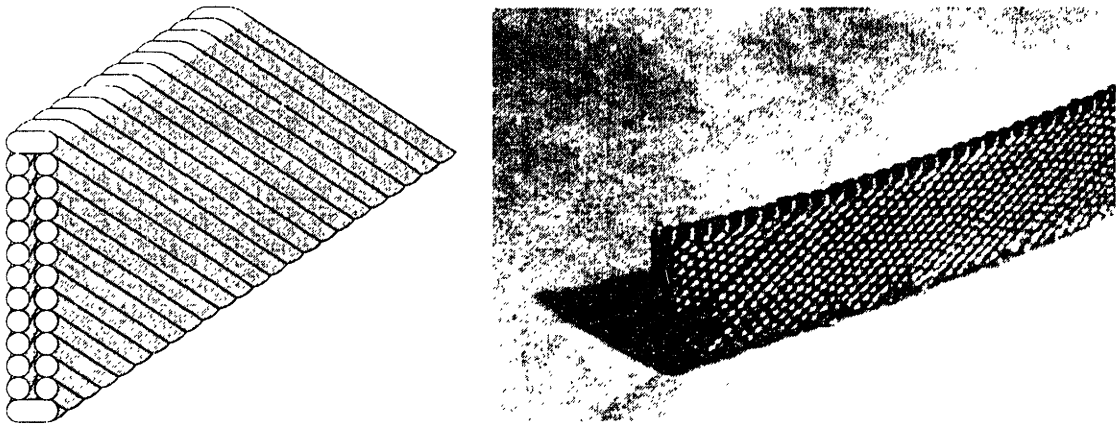

Figure 2.3 Structure of a 24 wire helical winding and winding sample

The winding utilized for the implementation of the system in this thesis is a twelve-phase helical winding. An illustration of the winding pattern of a helical winding is shown in Figure 2.3 - the wires are wound helically around a cylindrical mandrel and flattened to form the winding. Note that although this thesis utilizes a twelve-phase helical winding, a more conventional square winding would work as well. The structure of the winding is illustrated through a side view shown in Figure 2.4. In this figure, two wire bundles are shaded to illustrate how they traverse the winding. Each bundle, in this case, traverses the length of the winding twice, forming a set of slanted loops. It is through these loops that the vehicle transmitters and sensors couple their signals into and out of each of the phases of the winding.

2.3 Vehicle Transmitter

The basic design of the vehicle transmitter was performed and described in 1. Many of the results are summarized, for convenience, in this section. Signals may be introduced into the winding through inductive means, as shown in Figure 2.4, through the use of a ferrite 'C' core. The 'C' core is wound with 96 turns of 8/30 litz wire, and a sinusoidal current $I_{in} = I_0 \sin(\omega_c t)$ is driven through this winding. This current links a flux $\phi_L = \phi_0 \sin(\omega_c t)$ through the helical winding. This time-varying flux induces a voltage $V_w = V_0(\theta)\cos(\omega_c t)$ in the phases of the winding, where $V_0(\theta)$ takes into account

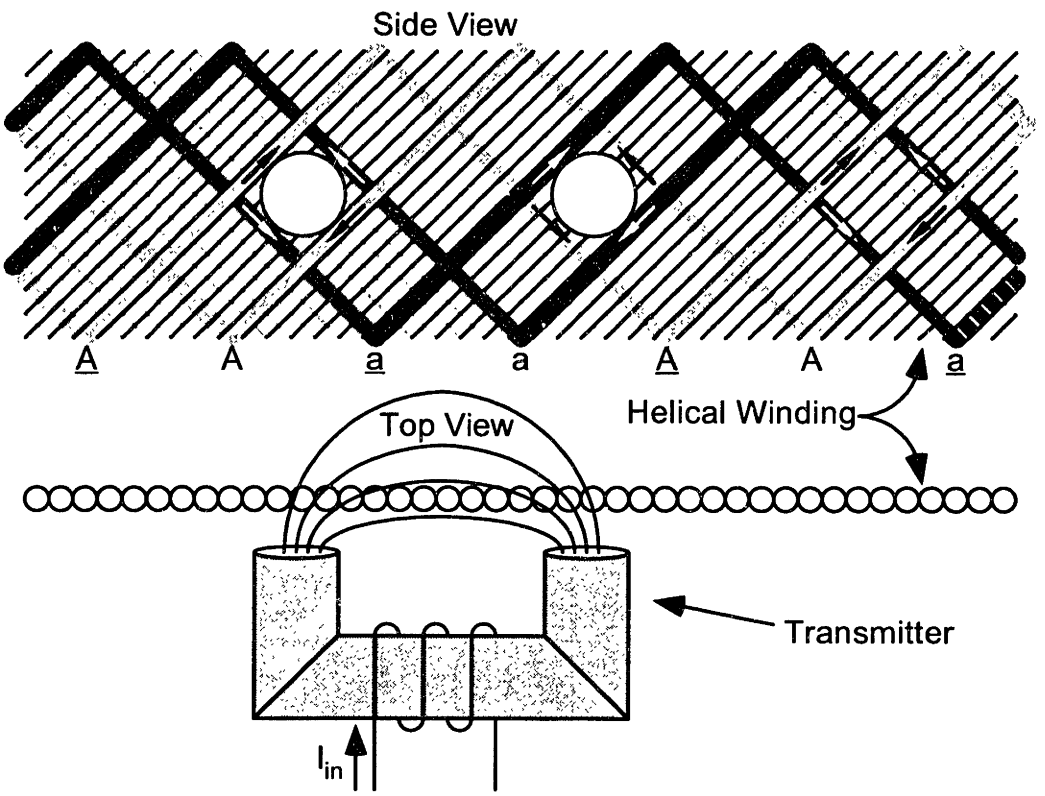

Figure 2.4 Signal transmitter, side view and overhead view

the mutual coupling between the inductor and each winding phase, creating a current signal in the winding.

This mutual coupling between the transmitter and each of the phases of the winding depends on the relative position of the winding and the transmitter. The coupling is periodic in transmitter position due to the periodicity of the winding. This mutual coupling function, $\mathcal{F}(\theta)$, depends on the shape of the pole faces of the transmitter, topology of the winding, and the distance between the transmitter and the winding. By shaping the transmitter correctly, the fundamental component of the Fourier Series dominates the periodic function.

Harmonic #	1	3	5	7	9
Relative Amplitude	1	0.0323	0.0083	0.0004	0.0008

Table 2.1 Relative harmonic amplitudes of coupling function $\mathcal{F}(\theta)$

Figure 2.4 shows the transmitter at a phase of $\theta = 0°$, and thus the coupling between the transmitter and phase *a* is $\mathcal{F}(\theta) \approx \cos(\theta)$. The actual coupling function $\mathcal{F}(\theta)$ also contains odd harmonics due the structure of the winding and shape of the transmitter. Experimental measurements of these harmonics are shown in Table 2.1. Coupling to the other 11 phases shown is identical with a shift of phase in the mutual coupling function. The structure of the helical winding thus induces the following voltages in six of the phases:

$$\begin{aligned}
V_{phase_a} &= V_0 \cdot \mathcal{F}(\theta) \cdot \cos(\omega_c t) \\
V_{phase_b} &= V_0 \cdot \mathcal{F}(\theta - 2\pi/3) \cdot \cos(\omega_c t) \\
V_{phase_c} &= V_0 \cdot \mathcal{F}(\theta + 2\pi/3) \cdot \cos(\omega_c t) \\
V_{phase_x} &= V_0 \cdot \mathcal{F}(\theta - \pi/4) \cdot \cos(\omega_c t) \\
V_{phase_y} &= V_0 \cdot \mathcal{F}(\theta - 11\pi/12) \cdot \cos(\omega_c t) \\
V_{phase_z} &= V_0 \cdot \mathcal{F}(\theta + 5\pi/12) \cdot \cos(\omega_c t)
\end{aligned} \tag{2.1}$$

The voltages in the other six phases, *A, B, C, X, Y* and *Z*, are the negatives of the corresponding voltages listed above, as can be deduced from Figure 2.4. These twelve phases will be referred to as mini-phases, as they are connected in such a way as to form six propulsion phases and six position-sensing phases. The corresponding phases (e.g. *a* and *A* phases) are connected in parallel for propulsion purposes, in this particular winding, as shown in Figure 2.5. As illustrated, the propulsion currents flow in parallel through the paired mini-phases in a common-mode manner. Approximately half of the propulsion current flows through each of the two mini-phases since the impedances of the mini-phases are approximately equal.

Notice that since the induced voltage in the *A* mini-phase is the negative of the voltage induced in the *a* mini-phase, the voltage around the loop is double the induced voltage in the *a* phase. Thus, the position sensing currents flow around the loop of the connected phases while the propulsion currents flow through in parallel. It is now clear to see that there are six position-sensing phases as well, each composed of two paired mini-phases in a loop. Another way to describe the system is that the propulsion currents

are common-mode while the position sensing currents are differential-mode, and this fact is used to help reject propulsion signals at the sensors. Thus, there are twenty-four wire bundles, twelve mini-phases, six common-mode propulsion phases, and six differential-mode position-sensing phases. The phases described in the rest of this document shall refer to the six differential-mode position-sensing phases, unless otherwise specified.

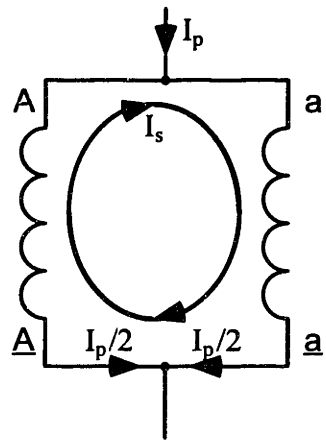

Figure 2.5 Connection of linear motor phases.
Used with permission from [1]

Note that there is also a voltage induced in the phases due to movement of the transmitter relative to the winding. By choosing a carrier frequency ω_c to be high enough, this secondary voltage becomes negligible. The set of signals represented in Equation (2.1) show a six phase set of signals modulated by the carrier frequency of the transmitter excitation. The current signals induced in the winding may be determined directly from the induced voltages and winding impedance matrix. Note that this matrix differs from the propulsion impedance matrix due to the structure illustrated in Figure 2.5. This impedance matrix consists of two components, the winding resistance matrix and inductance matrix. These matrices, for the six phase system illustrated in Figure 2.4, are ideally of the form:

$$\mathbf{R} = \begin{bmatrix} r_w & 0 & 0 & 0 & 0 & 0 \\ 0 & r_w & 0 & 0 & 0 & 0 \\ 0 & 0 & r_w & 0 & 0 & 0 \\ 0 & 0 & 0 & r_w & 0 & 0 \\ 0 & 0 & 0 & 0 & r_w & 0 \\ 0 & 0 & 0 & 0 & 0 & r_w \end{bmatrix} \quad \mathbf{L} = \begin{bmatrix} d & e & e & f & h & g \\ e & d & e & g & f & h \\ e & e & d & h & g & f \\ f & g & h & d & e & e \\ h & f & g & e & d & e \\ g & h & f & e & e & d \end{bmatrix} \tag{2.2}$$

The values of these matrices were empirical measured for an 8.4 meter section of the winding [1]. The resistance of each phase of the winding was consistently measured at approximately 1.8 Ω. The empirical measurements of the inductance matrix values contained more variation from the ideal case, but basically fits the form of Equation (2.2):

$$L_s = \begin{bmatrix} 66.9 & -13.8 & -14.2 & 22.3 & -44.0 & 6.8 \\ -13.8 & 72.9 & -14.0 & 6.8 & 28.6 & -50.5 \\ -14.2 & -14.0 & 66.8 & -45.0 & 6.6 & 21.9 \\ 22.3 & 6.8 & -45.0 & 66.8 & -13.7 & -14.0 \\ -44.0 & 28.6 & 6.6 & -13.7 & 73.0 & -19.9 \\ 6.8 & -50.5 & 21.9 & -14.0 & -19.9 & 72.7 \end{bmatrix} \mu\text{H} \tag{2.3}$$

The impedance matrix at a particular frequency is thus:

$$\mathbf{Z} = \mathbf{R} + j\omega\mathbf{L} \tag{2.4}$$

We may thus calculate the current signals, in phasor form at frequency ω_c, in the stator winding as follows using the voltage around the loop formed by the paired mini-phases (as described previously) and the impedance matrix.

$$\mathbf{v} = 2V_0 \begin{bmatrix} \mathcal{F}(\theta) \\ \mathcal{F}(\theta - 2\pi/3) \\ \mathcal{F}(\theta + 2\pi/3) \\ \mathcal{F}(\theta - \pi/4) \\ \mathcal{F}(\theta - 11\pi/12) \\ \mathcal{F}(\theta + 5\pi/12) \end{bmatrix} \tag{2.5}$$

$$\mathbf{i} = \mathbf{Z}^{-1}\mathbf{v} \tag{2.6}$$

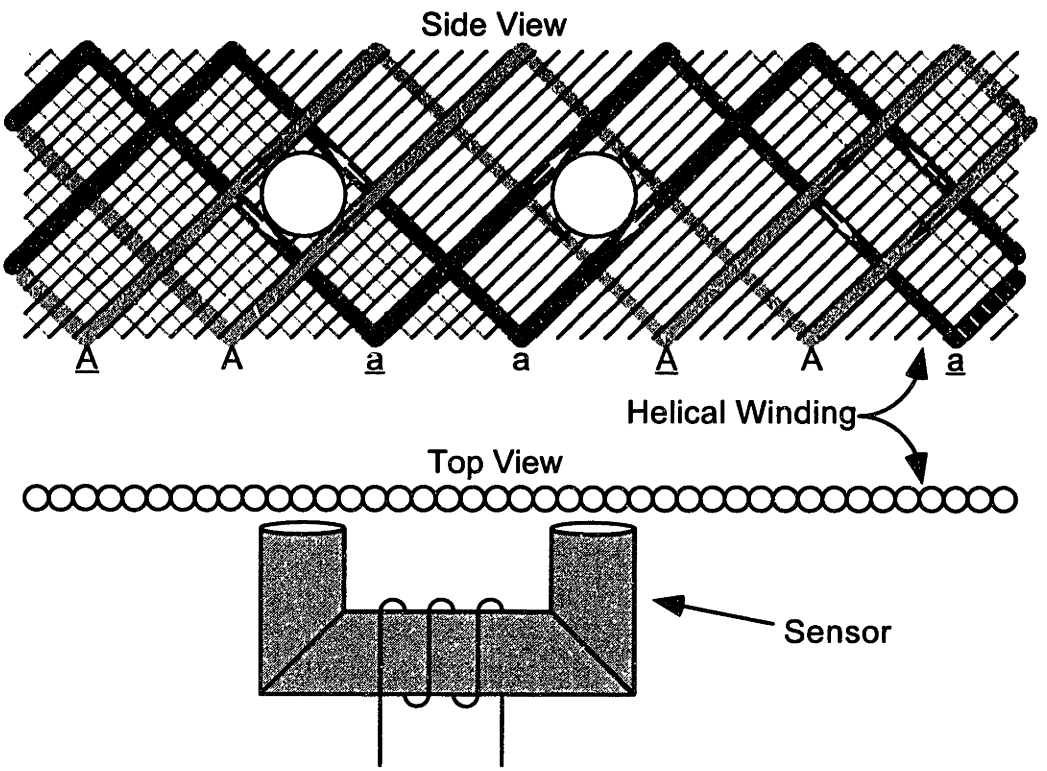

Figure 2.6 Vehicle Sensor

2.4 Vehicle Sensors

The signal sensors utilized in this system are identical in form to the transmitters, as shown in Figure 2.6. The sensors were designed in this manner so that the mutual couplings between the sensor and the helical winding phases are identical to those of the transmitter. The vehicle sensors are wound with 400 turns of 30 gauge wire to acquire a strong signal. This winding is connected to relatively high impedance inputs, and thus little current flows through the sensor output. The voltage on the winding of the sensor is measured and utilized for sensing the position of the transmitters and communication signals.

Figure 2.7 displays two vehicles on a single winding. It is possible to calculate the signal received by the sensor when considering a sensor on board Vehicle 2 and a transmitter on Vehicle 1. By making the assumption that a vector of currents, **i**, exists in the winding at a particular frequency ω_c, this measured voltage may be calculated using the mutual inductance between the winding and the sensor:

$$\mathbf{m} = \begin{bmatrix} \mathcal{F}(\theta_1 + \phi) \\ \mathcal{F}(\theta_1 + \phi - 2\pi/3) \\ \mathcal{F}(\theta_1 + \phi + 2\pi/3) \\ \mathcal{F}(\theta_1 + \phi - \pi/4) \\ \mathcal{F}(\theta_1 + \phi - 11\pi/12) \\ \mathcal{F}(\theta_1 + \phi + 5\pi/12) \end{bmatrix} \tag{2.7}$$

and the following equation (in phasor form):

$$V_{sensor} = \frac{d}{dt}(\mathbf{m}^T\mathbf{i}) \tag{2.8}$$

If it is further assumed that the signals in the winding are generated from a transmitter on vehicle 1, then this equation is of the form:

$$V_{sensor} = \frac{d}{dt}(\mathbf{m}^T\mathbf{i}) = \frac{d}{dt}(\mathbf{m}^T\mathbf{Z}^{-1}\mathbf{v}) \tag{2.9}$$

2.5 Wayside Transmitter

Since there is access to the phases of the winding on the wayside, signals may be inductively coupled into each of these phases through the use of ferrite cores. A special structure for these transmitters is necessary, in the demonstration system, to couple a signal into the differential-mode loops utilized by the vehicle transmitters and sensors. A toroidal, ferrite core is utilized with primary and secondary windings for this task, as shown in Figure 2.8. The primary winding of this transmitter is wound with 88 turns of

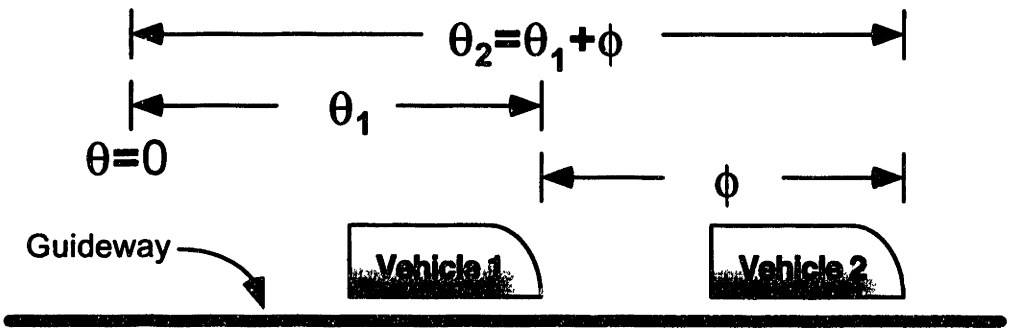

Figure 2.7 Vehicle 1 induces a signal in the winding. Vehicle 2 senses the signal.

30 gauge magnet wire, and the mini-phase secondary windings each form one turn around the core. The two mini-phases in this figure are wound in opposite directions through the core such that the transmitter signal is induced to flow around the loop formed by the mini-phases. This structure could be interpreted equivalently as two turns of this secondary winding loop. Thus, a signal is introduced into the winding in the differential-mode manner that the vehicle sensors and transmitters utilize.

If a sinusoidal current $I_{trans} = I_1 \sin(\omega_c t)$ is driven through the transmitter winding, a flux $\phi_t = \phi_1 \sin(\omega_c t)$ is linked through appropriate position-sensing phase of the helical winding. This time-varying flux induces a voltage $V_t = V_1 \cos(\omega_c t)$ in the appropriate phase of the winding, as shown in Figure 2.8.

2.6 Wayside Sensors

The wayside sensors are identical in form to the wayside transmitters. The sensors thus detect the differential-mode signals utilized for the features of this system. Additionally, since the propulsion currents of the two mini-phases flow in opposite directions through the core, there is a common-mode rejection of the propulsion currents at the output of the sensor. While the actual rejection is not complete, the level of the propulsion current signals are reduced by a factor of approximately 200 in the demonstration system.

Assuming a current $I_2 \sin(\omega_c t)$ is flowing through the appropriate phase of the winding, a flux of $\phi_S = \phi_2 \sin(\omega_c t)$ is linked by the sensor secondary winding. This flux generates a voltage $V_S = V_2 \cos(\omega_c t)$ at the output of the sensor. The wayside sensors are

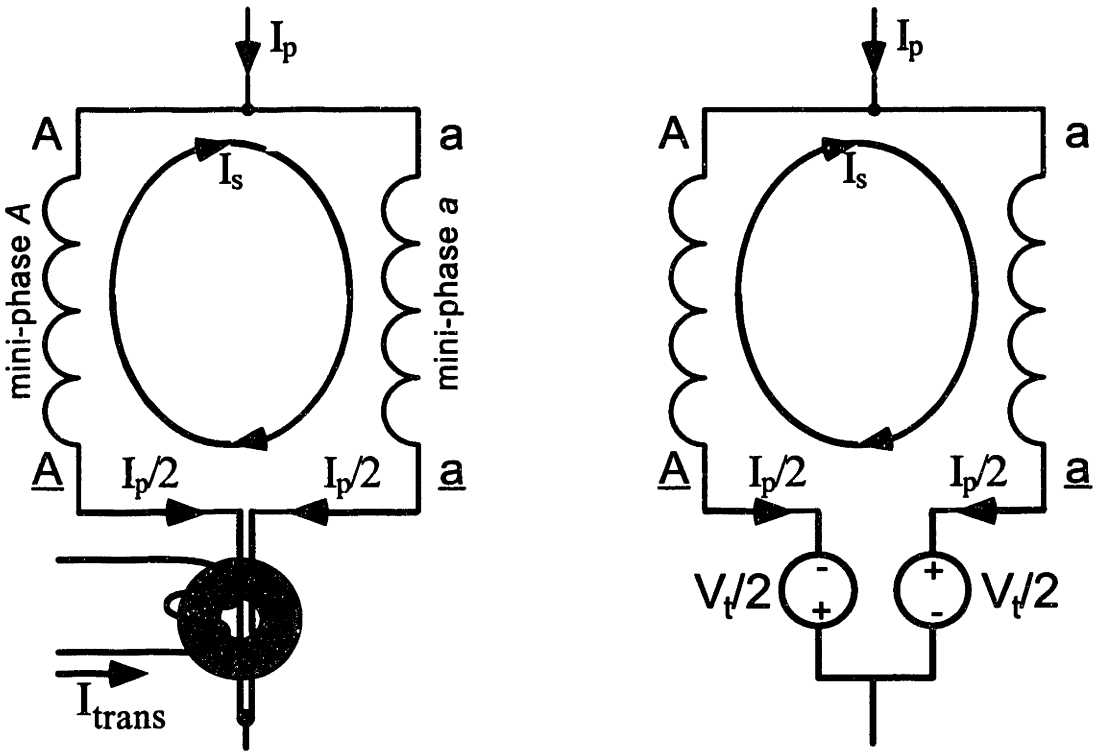

Figure 2.8 Wayside transmitter and equivalent voltage source

connected to relatively high impedance inputs, and thus very little current flows through the secondary winding of the sensors.

Chapter 3

Position-Sensing Techniques

With the architecture of the previous chapter, it is possible to implement several types of position-sensing features. The first position-sensing algorithm to be developed will enable the tracking, on board a vehicle, of the relative position of other vehicles. This technique requires both a transmitter and sensor set on board the vehicle. A related technique to enable autonomous position-sensing by a vehicle of its own precise position is enabled through the use of a single transmitter on the wayside and a set of sensors on board the vehicle.

This system, through similar techniques, is able to track the position of a vehicle from the wayside through the use of a vehicle transmitter and wayside sensors. Such an implementation will not be described here, as a related technique has been developed previously to accomplish the same objective [1]. Finally, a means will be shown whereby a vehicle is able, through its on board sensor set, to track a moving virtual point on the guideway, as specified by a wayside system through wayside transmitters.

3.1 Inter-Vehicle Position-Sensing

Inter-vehicle position-sensing is an extremely useful feature for short headway, automated transportation systems. The ability, on board a vehicle, to sense the distance to other vehicles enables safe operation of such a system. Two such vehicles are illustrated in Figure 3.1; Vehicle 1 induces a signal in the winding through a transmitter and Vehicle

2 detects the signal utilizing the described vehicle sensor. The vehicles are separated in this illustration by a distance ϕ.

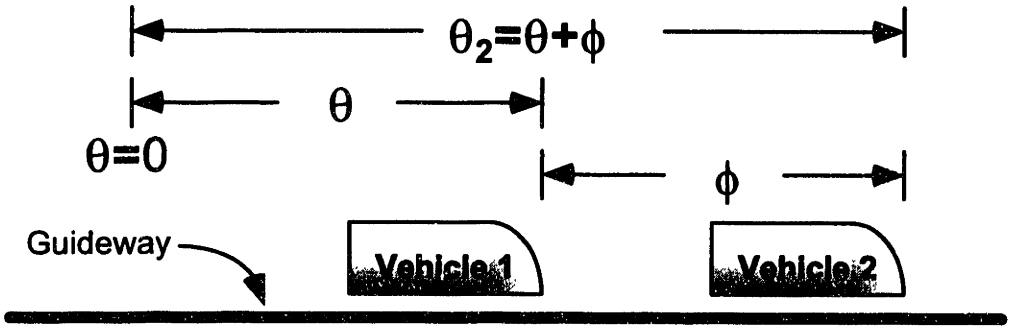

Figure 3.1 Vehicle 1 induces a signal in the winding. Vehicle 2 senses the signal.

The signal received by the sensor on board vehicle 2 is described mathematically by Equation (2.8). This equation does not directly give an intuition into the mechanics of the system. To understand the system more completely, the underlying process may be examined. The distance at which the winding pattern repeats is defined to be 2π radians, or 360 degrees. The case where the two vehicles are at the same position relative to the phases of the winding, $\phi = \pm 2\pi N$ radians (N integer), will be examined first. The transmitter and sensor in this case are assumed to have identical mutual couplings with each of the phases. Thus, the transmitter and sensor in this configuration have the maximum coupling level, relative to other possible positions on the winding.

The case where the two vehicles are $\phi = \pi \pm 2\pi N$ radians apart (N integer) may be examined next. The relative position of the a phase and the transmitter is now the same as the relative position the A phase and the sensor. Since the transmitter induces opposite signals in the a and A phases, the mutual coupling of the transmitter is the negative of the mutual coupling of the sensor with each of the phases. Thus, the sensor and transmitter are now in the configuration where they are maximally coupled in the negative sense. The total coupling between the sensor and transmitter must cross zero, and this crossover occurs when the two are 90 degrees out of phase. Thus, the coupling between the sensor and transmitter is a periodic function of the distance between the two.

This point, in more detail, may be shown with a simplified mathematical model. First, assume that there is no mutual coupling between the phases of the helical winding (i.e., **L** is a diagonal matrix). Also, it is possible to look at just three of the six winding phases, as the other three will contribute in an identical manner. Finally, for simplicity, it is assumed that $\mathcal{F}(\theta)=\cos(\theta)$. Thus, the harmonics are ignored, which are small by design in any case. With these simplifications, it is then straightforward to calculate the voltage induced at the sensor on Vehicle 2 from the signals induced in each of the three phases. The total voltage is the sum of the signals in each phases multiplied by the coupling of each of those phases to the sensor.

$$V_{vehicle2} = V_1 \cos(\omega_c t + \beta) \cdot \begin{bmatrix} \cos(\theta)\cos(\theta+\phi) \\ +\cos(\theta-2\pi/3)\cos(\theta+\phi-2\pi/3) \\ +\cos(\theta+2\pi/3)\cos(\theta+\phi+2\pi/3) \end{bmatrix} \tag{3.1}$$

It is possible to simplify this expression using the following trigonometric identities:

$$\begin{aligned} &\cos(\theta+\phi) = \cos(\theta)\cos(\phi) - \sin(\theta)\sin(\phi) \\ &\cos^2(\theta) = \frac{1}{2} + \frac{1}{2}\cos(2\theta) \\ &\sin(\theta)\cos\theta = \frac{1}{2}\sin(2\theta) \\ &\sin(\theta) + \sin(\theta-2\pi/3) + \sin(\theta+2\pi/3) = 0 \\ &\cos(\theta) + \cos(\theta-2\pi/3) + \cos(\theta+2\pi/3) = 0 \end{aligned} \tag{3.2}$$

and algebraic manipulation:

$$= V_1 \cos(\omega_c t + \beta) \cdot \begin{bmatrix} \cos^2(\theta)\cos(\phi) - \cos(\theta)\sin(\theta)\sin(\phi) \\ +\cos^2(\theta-2\pi/3)\cos(\phi) - \cos(\theta-2\pi/3)\sin(\theta-2\pi/3)\sin(\phi) \\ +\cos^2(\theta+2\pi/3)\cos(\phi) - \cos(\theta+2\pi/3)\sin(\theta+2\pi/3)\sin(\phi) \end{bmatrix}$$

$$
\begin{aligned}
&= V_1 \cos(\omega_c t + \beta) \cdot \begin{bmatrix} \cos(\phi)\left[\cos^2(\theta) + \cos^2(\theta - 2\pi/3) + \cos^2(\theta + 2\pi/3)\right] \\ -\sin(\phi)\begin{bmatrix} \cos(\theta)\sin(\theta) + \cos(\theta - 2\pi/3)\sin(\theta - 2\pi/3) \\ + \cos(\theta + 2\pi/3)\sin(\theta + 2\pi/3) \end{bmatrix} \end{bmatrix} \\
&= V_1 \cos(\omega_c t + \beta) \cdot \begin{bmatrix} \frac{1}{2}\cos(\phi)\left[3 + \cos(2\theta) + \cos(2\theta - 4\pi/3) + \cos(2\theta + 4\pi/3)\right] \\ -\frac{1}{2}\sin(\phi)\left[\sin(2\theta) + \sin(2\theta - 4\pi/3) + \sin(2\theta + 4\pi/3)\right] \end{bmatrix} \\
&= V_1 \cos(\omega_c t + \beta) \cdot \left[\frac{3}{2}\cos(\phi)\right]
\end{aligned}
\tag{3.3}
$$

Thus, the analysis illustrates that the inductive coupling between the sensor and the transmitter is proportional to the cosine of the difference in position. It would now be appropriate to consider the effect on this result when the off-diagonal terms of the impedance matrix are considered. It is interesting to note that the inverse of the impedance matrix has a form identical to that of the impedance matrix. The inverse of the impedance matrix thus has the form:

$$
Z^{-1} = \begin{bmatrix} d' & e' & e' & f' & h' & g' \\ e' & d' & e' & g' & f' & h' \\ e' & e' & d' & h' & g' & f' \\ f' & g' & h' & d' & e' & e' \\ h' & f' & g' & e' & d' & e' \\ g' & h' & f' & e' & e' & d' \end{bmatrix} \tag{3.4}
$$

Even when the off-diagonal terms of the impedance matrix are considered, the result is identical with the exception of a larger magnitude and slight change in phase of the carrier since the added components of the impedance matrix have no associated resistance. Thus, the signal at the sensor, when accounting for the full impedance matrix, is of the form:

$$
\boxed{V_{sensor} = V_2 \cos(\omega_c t + \psi) \cdot \left[\frac{3}{2}\cos(\phi)\right]} \tag{3.5}
$$

A similar result in the two phase case may be derived much more simply with the application of a single trigonometric identity. In either case, this is a powerful result, and it is possible to use this result to detect the difference in position between the two vehicles. This simplified theoretical result holds very closely with the more accurate analysis described previously utilizing the harmonics of the coupling function, as will be illustrated in Section 4.

It is important to note that this signal is not sufficient by itself to uniquely keep track of relative position, as illustrated in Figure 3.2. If the electrical position is represented here by a point on the circle, the coupling level may be interpreted as a measurement of the x coordinate of that point. Without additional information, the point on the circle may not be uniquely tracked as it moves. Consider the case where the transmitter and sensor are an integral number of cycles apart. The sensor and transmitter in this configuration are in the most highly coupled state. If the distance between the two decreases, the coupling, and thus the signal level, drops. If the distance between the two increases, the coupling also drops. It is not possible without additional information to know whether the vehicles are moving closer together or farther apart. It is extremely important to distinguish between these two cases. Therefore, it is very important to have the additional information necessary to keep unique track of the difference in position.

If the case is considered where the sensor and transmitter are an integral number of cycles apart plus or minus a quarter cycle, no signal is received at the sensor. Thus, it is not possible to track the carrier when in this configuration. Without knowledge of the carrier phase, the resulting sign of the coupling is lost when the a suitable signal level is later reestablished. Thus, there must also be some means of tracking the carrier when the sensor and transmitter are in this configuration.

One means to accomplish this task may be deduced from Figure 3.2. A measurement of the sine of the position would give the additional information necessary to uniquely track a point on the circle as it moves. The solution is to add a second sensor similar to the first, except 90 degrees out of phase in position, as illustrated by Figure 3.4.

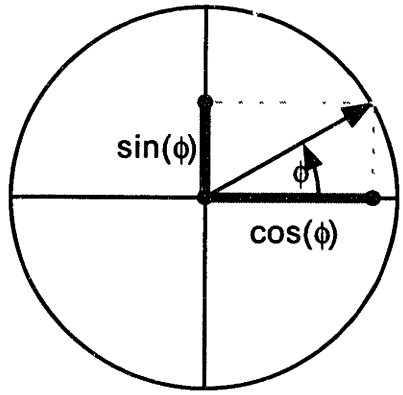

Figure 3.2 Information about electrical position from cosine

Thus,

$$V_{sensor2} = V_2 \cos(\omega_c t + \psi) \cdot \left[\frac{3}{2}\sin(\phi)\right] \tag{3.6}$$

With the addition of this signal, it is possible to keep track of the difference in position. In fact, the two signals together determine the difference in phase uniquely. One coupling represents the x coordinate while the other represents the y coordinate in Figure 3.2. If the phase of the carrier is unknown, however, the position is only known to ±180 degrees, but may still be tracked uniquely and corrected for. This uncertainty is due to the fact that the sign of the carrier is unknown, and thus the sign of the envelope of the signals is also unknown. A simulation of the signals received at these sensors is shown in Figure 3.3 for the case where one vehicle is accelerating away from the other. For illustration purposes, the 10 kHz carrier has been replaced with a 40 Hz carrier.

Thus, with an arbitrary starting value, both signal envelopes will have either the correct sign, or both envelopes will have their sign inverted, resulting is an uncertainty of 180 degrees. Since the position sensing cycle is twice as long as a propulsion cycle in this example, the uncertainty is in whole number increments of the propulsion cycle. Thus, this uncertainty does not affect the usefulness of this system for propulsion purposes. Furthermore, if an initial estimate of position is obtained, the phase of the carrier may be determined and accounted for in the necessary calculations.

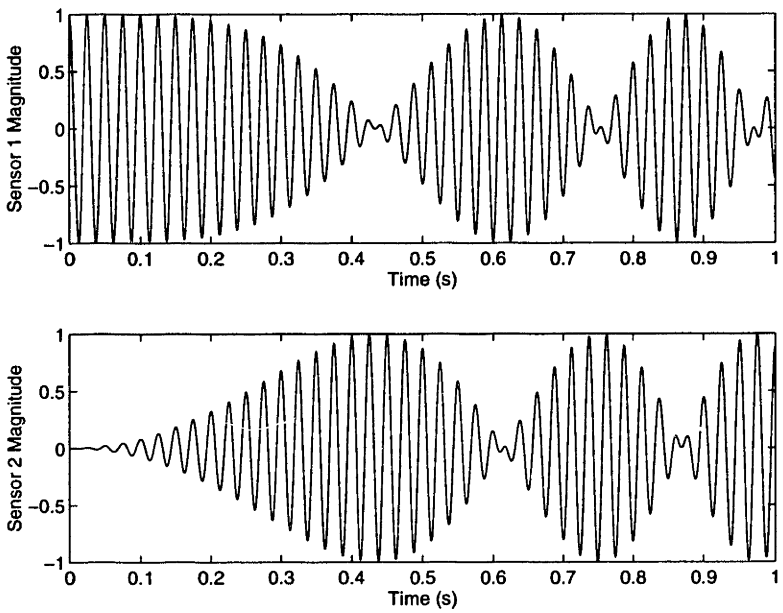

Figure 3.3 Sensor signals for accelerating vehicle

With sufficient information available to uniquely track position, the actual algorithms must now be developed. Since the necessary information in the signal is in the coupling levels of the two sensors, the first step is to extract these parameters from the sensor signals. The sensor signals may be viewed as the coupling levels modulated up to the carrier frequency. If it is possible to track the carrier frequency, it becomes possible to demodulate the signals. The system utilized to demodulate the coupling levels from the

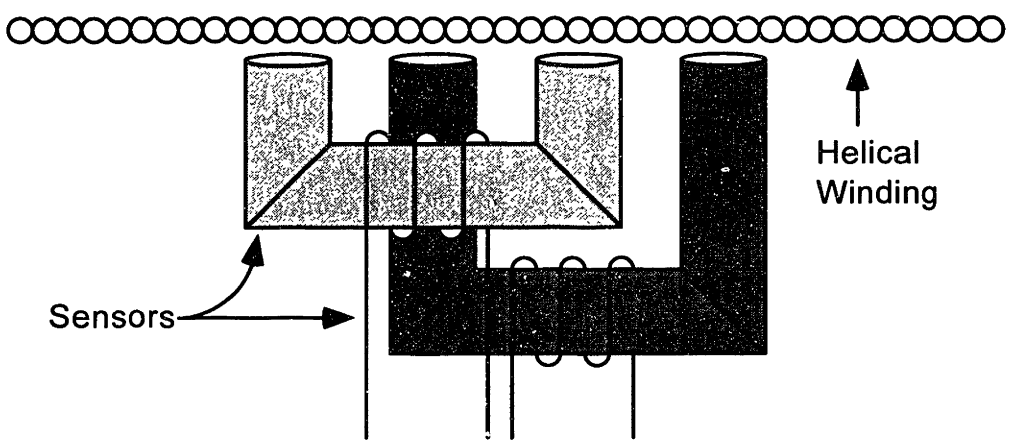

Figure 3.4 Two sensors are used to uniquely determine distance

sensor signals is shown in Figure 3.5.

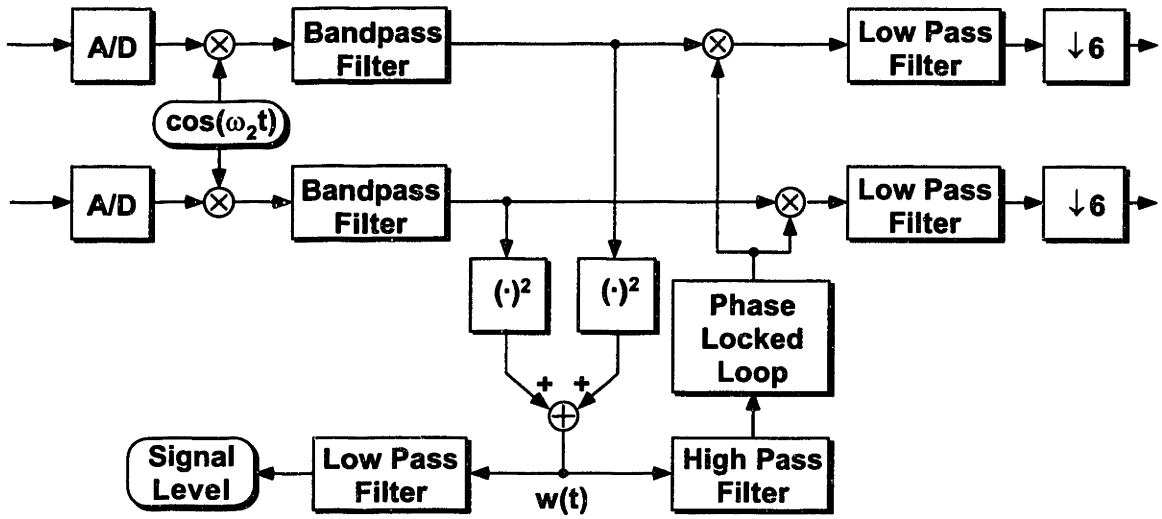

Figure 3.5 Sensor Demodulation Algorithm

Since all the processing in this system is performed in discrete-time, the first step in the process of demodulation is to sample the signals with an analog to digital converter and necessary anti-aliasing measures. The most appropriate way to track a carrier signal is with a phase locked loop (PLL). Phase locked loops have excellent noise rejection characteristics, and have a trade-off characteristic between lock speed and noise rejection. With a discrete-time implementation, both characteristics may be achieved by switching over from a fast lock loop to a high noise immunity loop once the carrier is locked. It is also possible at attain these characteristics by switching the types of phase detectors as well.

An excellent rule of thumb for implementing a PLL [26] in the discrete-time domain is to sample at a rate of at least ten times the carrier frequency. For the 10 kHz carrier used in the demonstration system, a 100 kHz sampling rate would be necessary. Steps were taken in the development of the system to reduce this demand on the sampling system and also on the computational load of the digital signal processor (DSP). Inexpensive 16-bit audio codecs are readily available which sample at rates up to 48 kHz. Such a stereo codec, the Crystal Semiconductor CS4215, was used in the demonstration

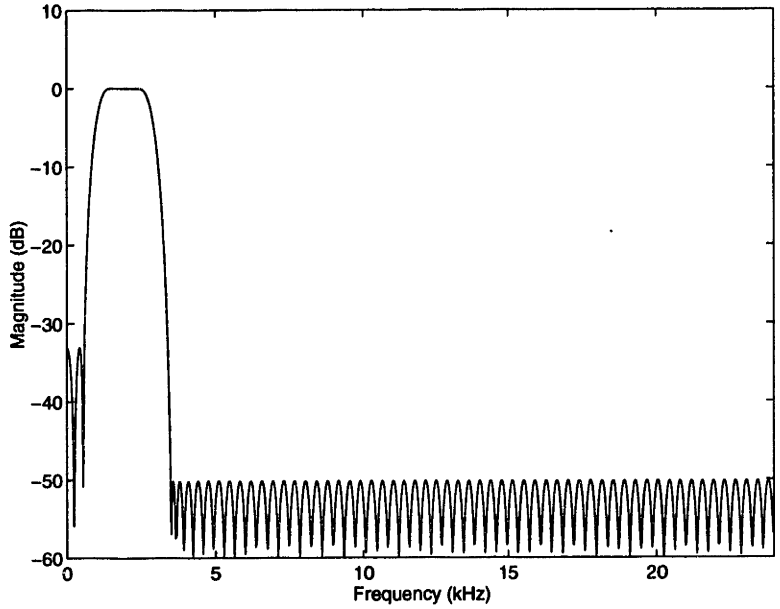

Figure 3.6 Transfer function magnitude of a 128-tap bandpass FIR filter

system at a sample rate of 48 kHz. Sampling the 10 kHz carrier signal at this rate does not lead to any aliasing, and thus the use of such codecs is entirely appropriate.

One solution to accommodate the rule of thumb would be to upsample and filter the discrete-time signal before forwarding to the PLL. This solution is computationally intensive, and thus undesirable. The solution taken in this thesis is to modulate the original signal down closer to baseband before processing. The bandwidth of the original signal is only up to about 160 Hz, and so the signal may be modulated down to a value above 80 Hz without worry of aliasing.

The 10 kHz signal in the demonstration system is modulated down to 2 kHz before processing in this implementation. Since a frequency of double the carrier is tracked with a PLL, the rule of thumb suggests a minimum sampling rate of 40 kHz, which has been satisfied. Without the modulation down, the required sampling rate, or update rate for the PLL, would have been 200 kHz. Thus, with the approach used in the demonstration system, the computational load on the system has been reduced by approximately a factor of four. A linear, finite impulse response (FIR), minimax optimal filter is applied to the signal after the modulation to remove the high frequency component introduced by the modulation and to attenuate any other signals in the winding received by the sensor. The transfer function magnitude of this 128 tap filter is

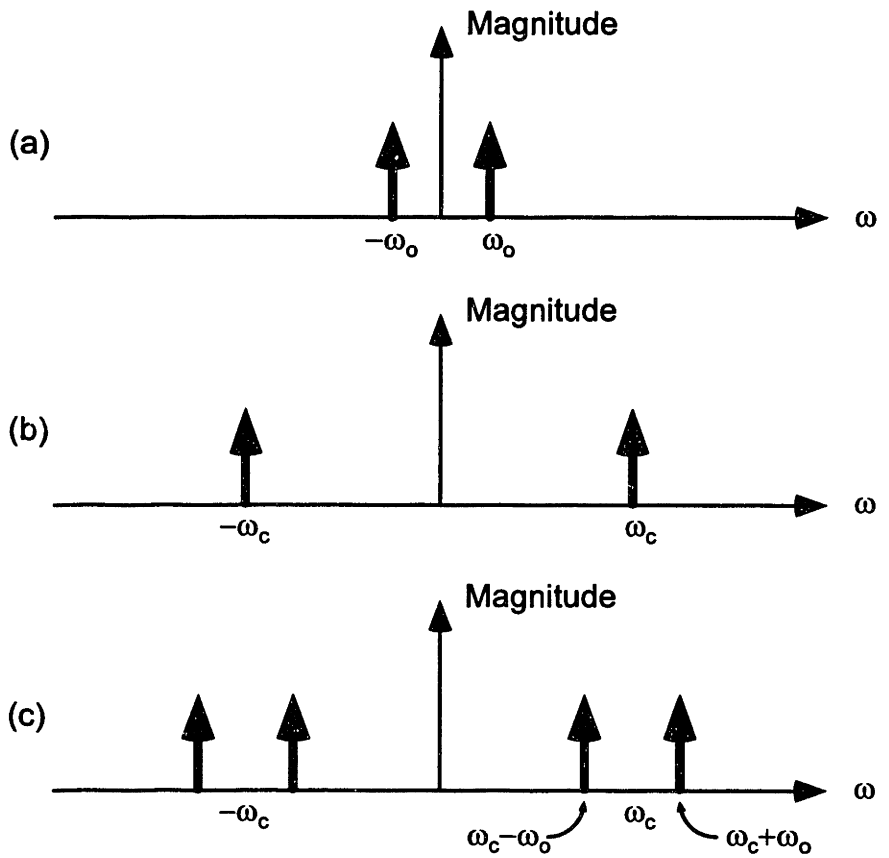

Figure 3.7 Coupling signal, modulation signal, and received sensor signal

shown in Figure 3.6. This filter helps the system reject out of band noise, and may easily be tailored to narrow its selectivity. A linear phase, FIR filter has been utilized in this system for two reasons. The filter is simple to implement directly on a digital signal processor, and the phase of the signal is not distorted.

In order for the phase locked loop to operate properly, it should constantly be fed the carrier signal it is to track. One manner in which to feed the PLL a constant amplitude carrier signal is developed in this thesis. This step is necessary since the two sensor signals carry either a positive or negative version of the carrier depending on the relative sign of the coupling. Also, the original sensor signals contain only a suppressed carrier signal, as illustrated in Figure 3.7. The frequency domain signal representation of the coupling of Sensor 1 at constant relative velocity is shown in subplot (a). The frequency domain representation of the modulating carrier frequency is shown in subplot (b). The modulated coupling signal received at sensor 1 is shown in subplot (c). Note that there is

no signal directly at the carrier frequency. It should be noted that for illustration purposes, the coupling signal frequency ω_b shown in Figure 3.7 is not drawn to scale as compared with the modulation frequency ω_c. The modulation frequency is approximately two orders of magnitude larger than the coupling signal frequency.

It is thus necessary to recover the carrier signal from the sensor signals in order to track the carrier and demodulate the coupling signals. A constant amplitude, double frequency carrier signal may be obtained by squaring and adding the two bandpass signals, and high pass filtering. A modified Butterworth filter is used for this filtering purpose. A signal level detector may be created by low pass filtering. The derivation in the time domain is as follows:

$$\begin{aligned} w(t) &= \left(V_2 \cos(\omega_c t + \psi) \cdot \left[\frac{3}{2}\cos(\phi)\right]\right)^2 + \left(V_2 \cos(\omega_c t + \psi) \cdot \left[\frac{3}{2}\sin(\phi)\right]\right)^2 \\ &= {V_2}^2 \cos^2(\omega_c t + \psi) \cdot \frac{9}{4}\left[\cos^2(\phi) + \sin^2(\phi)\right] \\ &= A\left(1 + \cos(2\omega_c t + 2\psi)\right) \end{aligned} \tag{3.7}$$

This double frequency (4 kHz) signal is fed into a discrete-time PLL with a multiplying phase detector. The PLL is designed for a lock range of 250 Hz, and an approximate lock time of 4 ms. The output frequency of the PLL is limited to be within 375 Hz of the 4 kHz design frequency. This PLL has been demonstrated to lock quickly, even in the presence of significant levels of noise.

Since the objective of the PLL is to track phase, it is simple to allow the software-implemented PLL to track the phase from 0 to 4π instead of just 0 to 2π. With this knowledge it is simple to output a frequency from the PLL at half of the PLL frequency. The carrier signal, with a sign uncertainty, has thus been reconstructed, and is utilized to demodulate the bandpass signals back to the baseband as illustrated in Figure 3.5. The filter utilized to remove the double frequency artifact from the demodulation signal at 4 kHz is an optimal, 32-tap FIR filter, the magnitude of which is shown in Figure 3.8. The coupling signals are now available for use, and are decimated by a factor of six to further lower the computational load on the DSP with no loss in performance.

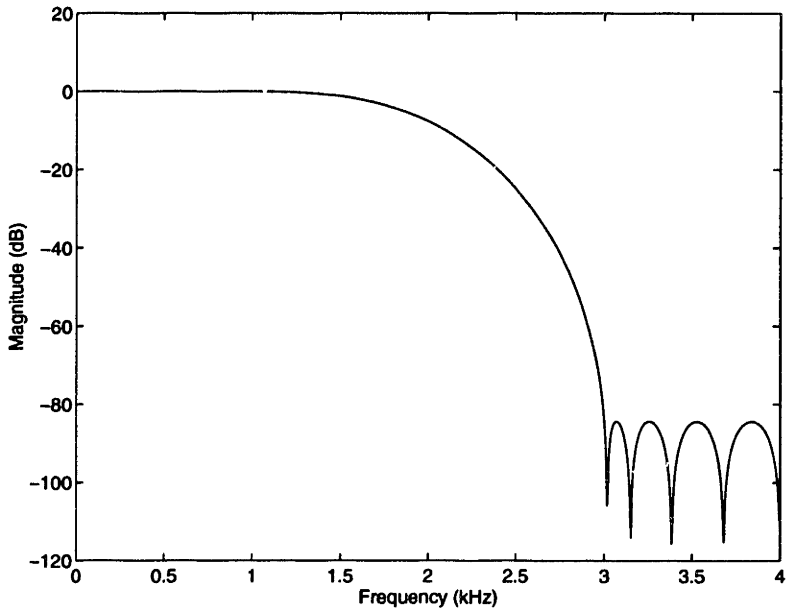

Figure 3.8 Low pass, 32-tap FIR filter to remove double frequency component

One may invent many methods for determining position from the two measurements, as discussed in Chapter 6. As can be seen from Figure 3.2, the two signal magnitudes uniquely identify the relative phase between vehicles, as well as the signal amplitude. One simplistic method to combine the two signals is to divide sensor 2 magnitude by sensor 1 magnitude, resulting in the tangent of the phase. A simple inverse lookup table may be utilized to determine immediate relative phase.

Another choice with several advantages is to use an observer to track position [23]. An observer may track the position and velocity while helping to reject noise and spurious inputs. One reliable method for tracking is to implement a non-linear observer, as illustrated in Figure 3.9 and investigated in [1]. The observer is non-linear because the innovation at the input of the observer is proportional to $\sin(\phi - \hat{\phi})$ rather than $(\phi - \hat{\phi})$. One may view this observer as a two input phase-locked loop, utilizing the quadrature coupling envelopes of the signals from the sensors. The weighting function for the feedback paths of this system is determined using the coupling function of the sensors [1, 24]:

$$W(\phi) = \frac{d}{d\phi}\mathcal{F}(\phi) \cong -\sin(\phi) \tag{3.8}$$

The error in the observer is defined as $\tilde{\phi} = (\phi - \hat{\phi})$. The error dynamics of this system are governed by the nonlinear differential equation, which may be derived from the figure:

$$\frac{d^2\tilde{\phi}}{dt^2} + c\frac{d\tilde{\phi}}{dt}\cos(\tilde{\phi}) + k\sin(\tilde{\phi}) = 0 \tag{3.9}$$

For small errors, the dynamics are linear, and are governed by:

$$\frac{d^2\tilde{\phi}}{dt^2} + c\frac{d\tilde{\phi}}{dt} + k\tilde{\phi} = 0 \tag{3.9}$$

Through proper choice of c and k, we may achieve a tracking system with excellent properties. The poles of the linearized error dynamics have been located at 70 and 180 rad/s, the same locations as originally used in [1]. This choice of pole locations leads to values of k=3126 rad/s^2 and c=125.0 rad/s. These poles were chosen, as discussed in section 3.5, as a compromise between noise immunity, and the effect of unmodeled system dynamics and locking speed.

This system has many benefits including a guarantee to acquire lock during normal operation and linear dynamics for small errors. In addition, in the two input system the sum of positions components $(\phi + \hat{\phi})$ cancel at the innovation to the phase

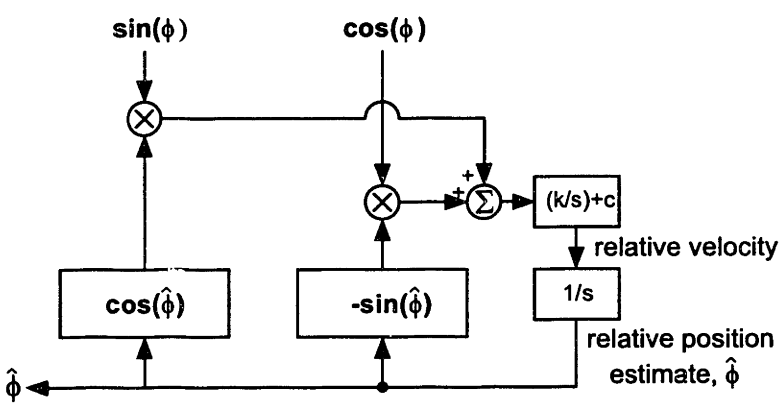

Figure 3.9 Nonlinear observer for relative position-sensing

locked loop, negating the need to filter out such components as is necessary in single input phase locked loops. A feature which is not possible with a single input PLL implementation is enabled through the use of a two input PLL. This feature also allows the observer to track an unchanging relative position. This observer is a two input version of the observer utilized in [1], yet retains all of the advantages of the original three input implementation. One of the best benefits of this non-linear observer is that its weighting function has been designed to minimize the effect from additive uncorrelated Gaussian noise [1].

One limitation of this system is that only the relative phase is known; the actual position is only known to $\pm 2\pi N$ where N is the length of a cycle. However, if the actual distance is identified once, the exact distance may be tracked uniquely. Means of acquiring estimates of actual distance are detailed in Chapter 6.

3.2 Autonomous Position-Sensing

The algorithm developed in the previous section may be utilized for more tasks than just inter-vehicle position sensing. The algorithms may also be utilized for autonomous position-sensing, where the vehicles are able to detect their own position relative to the guideway. This function is very much like that of an odometer, with some additional benefits. The approach taken in this thesis does not suffer from accumulation errors, and is not dependent on wheel traction for its measurements. In fact, one use of such a feature could be for traction control!

Consider the possibility that with access to all phases of the helical winding at the wayside, one may inject desired signals into each phase of the winding. It is then possible to inject "marker" signals into the winding identical to those produced by a stopped vehicle, as illustrated in Figure 3.10. The system on board the vehicle does not distinguish between an actual stopped vehicle and the stationary marker signals, and is thus able to track its distance from the virtual marker. The signals necessary to implement such a virtual marker are identical to those produced by a vehicle transmitter:

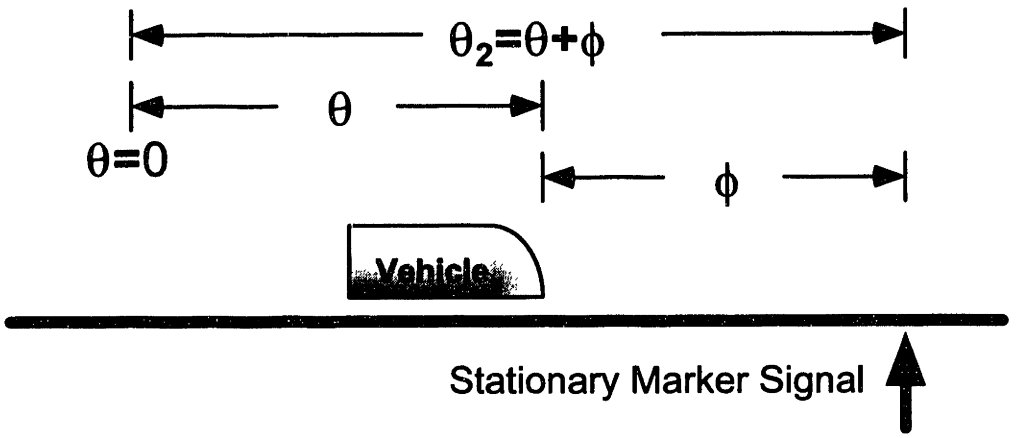

Figure 3.10 Autonomous Position Detection

$$\begin{aligned} V_{phaseA} &= V_0 \cdot \mathcal{F}(\theta_0) \cdot \cos(w_c t) \\ V_{phaseB} &= V_0 \cdot \mathcal{F}(\theta_0 - 2\pi/3) \cdot \cos(w_c t) \\ V_{phaseC} &= V_0 \cdot \mathcal{F}(\theta_0 + 2\pi/3) \cdot \cos(w_c t) \end{aligned} \tag{3.10}$$

A convenient choice of position would be θ=0 resulting in the following signals:

$$\begin{aligned} V_{phaseA} &= 1 \cdot V_0 \cdot \cos(w_c t) \\ V_{phaseB} &= -\frac{1}{2} \cdot V_0 \cdot \cos(w_c t) \\ V_{phaseC} &= -\frac{1}{2} \cdot V_0 \cdot \cos(w_c t) \end{aligned} \tag{3.11}$$

The number of necessary wayside transmitters may be reduced by one by choosing an appropriate position for the marker. For instance, if $\theta_o=\pi/2$, the first of the three signals is zero. A further reduction in the number of necessary transmitters may be achieved by recognizing that common-mode signals on the three phases are rejected at the output of the vehicle sensor. This fact may be derived by examining the output of the vehicle sensor to a common-mode sinusoidal signal in all three phases and utilizing the last of the trigonometric identities listed in Equation (3.2):

$$V_{sensor1} = V_2 \cos(\omega_c t + \psi)\left[\cos(\theta) + \cos(\theta - 2\pi/3) + \cos(\theta + 2\pi/3)\right] \\ = 0 \tag{3.12}$$

Thus, a common-mode signal may be added to all three phases without any change in the signal detected at the sensor. One may thus choose to subtract the signal in phase C of Equation (3.11) from all three phases. Thus, the signals in the B and C phases are identically zero, and no transmitters are required to drive these phases. The signal in phase A is now:

$$V_{phaseA} = \frac{3}{2} \cdot V_0 \cdot \cos(\omega_c t) \tag{3.13}$$

This result makes perfect sense from another point of view. Sensor 1 links the signal in phase A with the cosine of position, and Sensor 2 links the signal in phase A with the sine of position. These signals are, therefore, identical in form to those of Equations (3.5) and (3.6) in the relative position-sensing technique.

A real vehicle may detect its own position relative to the "virtual marker" (ghost vehicle) and thus position relative to a stationary point on the wayside. Again, only electrical position is measured with this technique, and additional information is necessary to track exact position. This objective may be accomplished though several methods, including use of magnetic or optical markers on the wayside which may be easily detected by the vehicle. Various methods are to achieve this goal are discussed in Chapter 6.

3.3 Virtual Marker Tracking

Extending the previous concept one step further enables a point tracking feature. The wayside is not only capable of injecting a stationary marker as in the previous section; it is also capable of injecting a moving marker signal into the winding. The signals in this case are identical to the signals produced by a moving vehicle. If the vehicle controls its own propulsion, it may be directed to follow such a tracking signal, as illustrated in Figure 3.11. It is thus possible to communicate to the actual vehicle exactly where it should be on the guideway at all times by instructing it to follow the marker.

This concept is known as "point-following" in the transportation field [2,5], and is the basis of several transportation control systems.

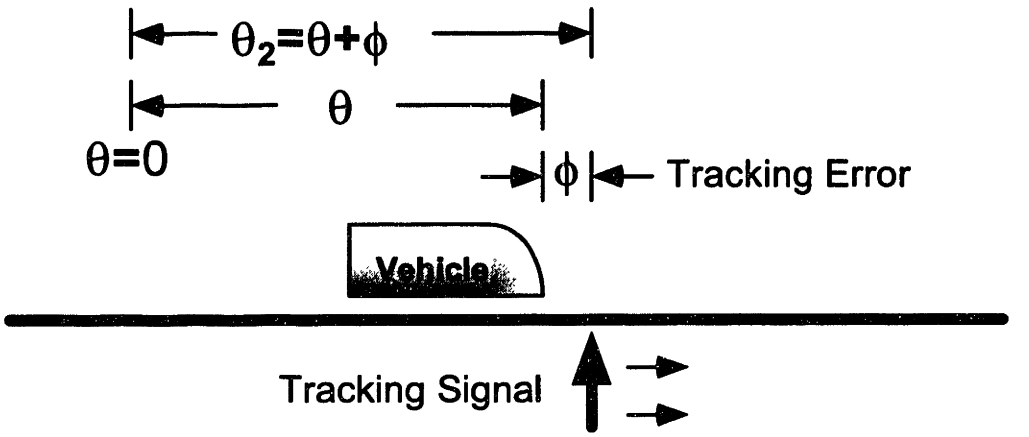

Figure 3.11 Vehicle follows a tracking signal

This feature also has another significant use if a Doubly Excited Linear Motor (DELM) is utilized for the propulsion of a transportation system. The DELM has been proposed for use in PRT systems. In this instance, several vehicles are propelled by the same stator winding. The stator produces a field moving at a constant velocity. The vehicles adjust their own field to lock into synchronism with the moving field while the vehicle advances or recesses relative to the moving field. In order to operate most efficiently, the vehicles need to know the relative phase of the field to their rotors. By placing moving virtual marker signals into the winding that move in synchronism with the stator field, the vehicles are able to track the relative position of the field.

The signals necessary to create a moving marker at desired position $\theta_d(t)$ for a vehicle to track are as follows:

$$\begin{aligned} V_{phaseA} &= V_0 \cdot \mathcal{F}(\theta_d(t)) \cdot \cos(w_c t) \\ V_{phaseB} &= V_0 \cdot \mathcal{F}(\theta_d(t) - 2\pi/3) \cdot \cos(w_c t) \\ V_{phaseC} &= V_0 \cdot \mathcal{F}(\theta_d(t) + 2\pi/3) \cdot \cos(w_c t) \end{aligned} \tag{3.12}$$

Again, the number of wayside transmitters may be reduced by recognizing the common-mode rejection property of the sensors. In this case, signals in two phases are necessary to implement this feature with the following signals:

$$\begin{aligned} V_{phaseA} &= V_0 \cdot \left[F(\theta_d(t)) - F(\theta_d(t) + 2\pi/3)\right] \cdot \cos(\omega_c t) \\ V_{phaseB} &= V_0 \cdot \left[F(\theta_d(t) - 2\pi/3) - F(\theta_d(t) + 2\pi/3)\right] \cdot \cos(\omega_c t) \end{aligned} \tag{3.13}$$

3.4 Accuracy and Measurements

The accuracy of the features of this chapter has been carefully measured in the demonstration system. The accuracy was measured over one period, or 75 mm, of the winding. The causes of errors in the system have been tracked down and accommodated through design changes as necessary. With appropriate adjustments, the absolute accuracy of the autonomous position-sensing system may be brought down to the 10 micron level, and in the relative position-sensing system, absolute accuracy is better than approximately 0.5 mm. It should be noted that these measurements are relative to the winding, and the absolute accuracy of the system depends upon how accurately the winding may be laid in the guideway.

The accuracy of the autonomous position-sensing system will be examined first, since there is only one degree of freedom in the system, the position of the sensors. Some basic strategies applicable to all the position-sensing features are developed in this section. The accuracy of the relative position-sensing technique is then in examined in both degrees of freedom, the position of the transmitter and the position of the sensors.

3.4.1 Autonomous Position-Sensing Accuracy

The accuracy of the autonomous position-sensing technique was measured by attaching the sensor vehicle to a milling machine, so the vehicle could be moved in very precise increments. The measurements from the position-sensing system were recorded with resolution down to the micron level as the milling machine moved the vehicle in

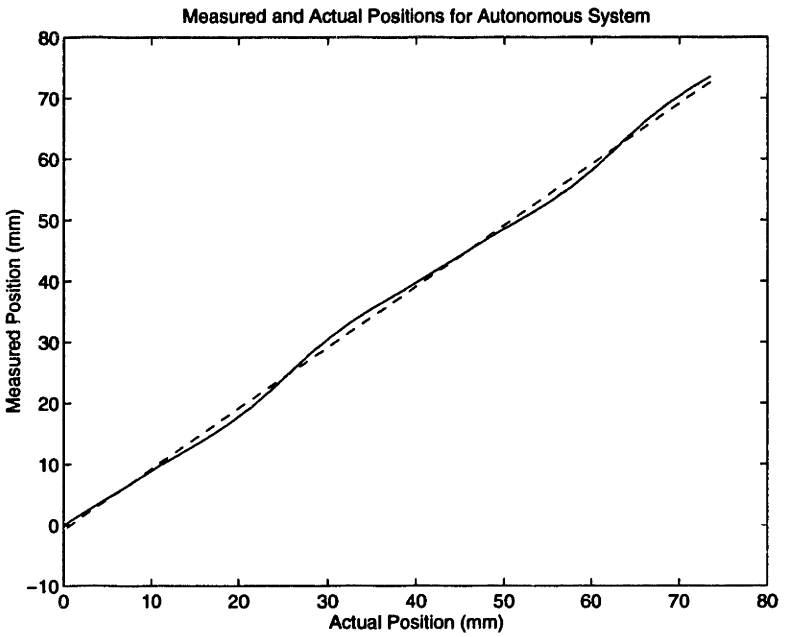

Figure 3.12 Autonomous position-sensing measurement accuracy

increments of 10 mils (approximately 254 microns). The milling machine had measurement marks down to a 1 mil (25.4 micron) resolution.

The first step taken before any measurements were made was to correct for a 3% difference in the amplitude of the signals from the two sensors. This amplitude variation is mostly due to the difference in gap sizes for the two sensors. With no other

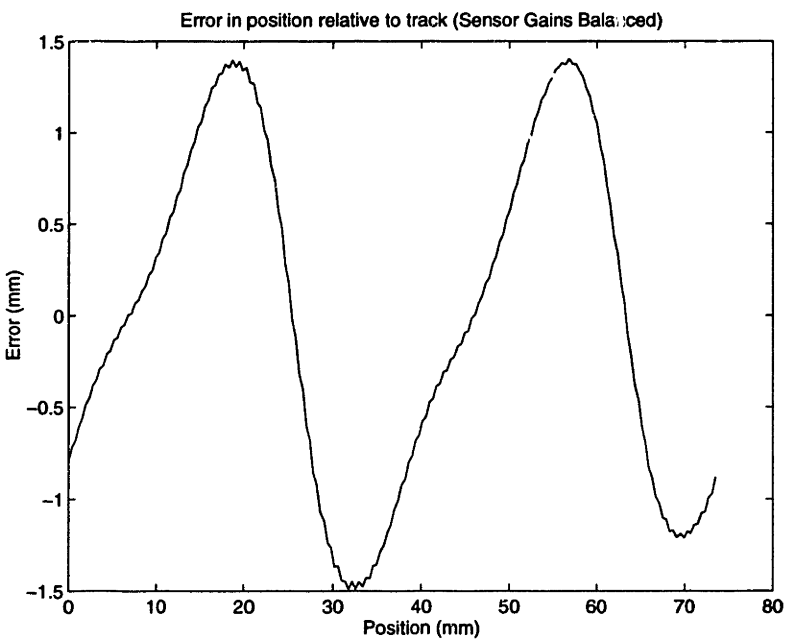

Figure 3.13 Autonomous position-sensing measurement error

modifications to the original algorithm, the measured position versus actual position has been recorded and is illustrated in Figure 3.12. The dashed line shows an ideal measurement while the solid line shows the recorded measurements. One key characteristic to note from the plot is the monotonicity of the measurement. The actual error is shown in Figure 3.13.

Clearly the majority of the error is a second harmonic with respect to the cycle length of the winding. A thorough analysis of the non-linear observer phase detector leads to a possible cause as the error in the phase offset of the two sensors. The error from the phase detector may be analyzed if the received coupling envelopes of the two sensors are assumed to not be exactly a quarter cycle apart. Since the errors are small relative to the length of a cycle the error may be analyzed for a linearized version of the observer. The error is analyzed for an operating point of equal vehicle and observer position.

$$\begin{aligned} Error &= -\cos(\phi)\sin(\phi) + \sin(\phi+\varepsilon)\cos(\phi) \\ &= -\cos(\phi)\sin(\phi) + \left[\sin(\phi)\cos(\varepsilon) + \sin(\varepsilon)\cos(\phi)\right]\cos(\phi) \\ &= \cos(\phi)\sin(\phi)\left[-1+\cos(\varepsilon)\right] + \cos^2(\phi)\sin(\varepsilon) \\ &= \frac{1}{2}\sin(2\phi)\left[-1+\cos(\varepsilon)\right] + \left[\frac{1}{2}+\frac{1}{2}\cos(2\phi)\right]\sin(\varepsilon) \\ &= \frac{1}{2}\left[\sin(2\phi)\cos(\varepsilon) + \cos(2\phi)\sin(\varepsilon)\right] - \frac{1}{2}\sin(2\phi) + \frac{1}{2}\sin(\varepsilon) \\ &= \frac{1}{2}\sin(2\phi+\varepsilon) - \frac{1}{2}\sin(2\phi) + \frac{1}{2}\sin(\varepsilon) \end{aligned} \tag{3.14}$$

It is apparent from Equation (3.14) that the error will be a constant, depending on the phase error, plus an error sinusoidal in 2ϕ. To correct for the error, a lookup table could be utilized to correct for the error, but an easier approach is to correct for the phase error in the weighting function of the non-linear observer. An appropriately enhanced version of the non-linear observer is shown in Figure 3.14. The resulting innovation error at the input of the observer has now been eradicated at the operating point. It should be noted that this correction is useful for all three of the position sensing features, and is utilized in the next section to reduce the relative position-sensing error.

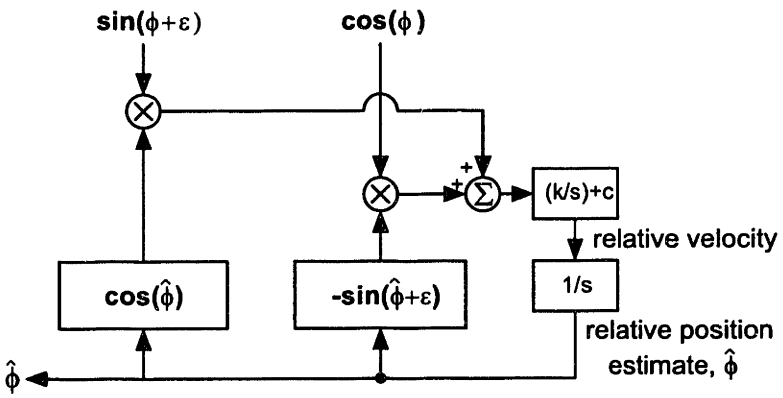

Figure 3.14 Observer corrected for input phase error

In order to correct for such a phase error, the coupling functions of the sensors were measured and the results are displayed in Figure 3.15. It is from this data that the harmonic levels of Table 2.1 are generated. The measured coupling functions are actually about 78 degrees apart instead of the desired 90 degrees apart in phase. There are two causes of this phase error. The first cause is a vehicle machining accuracy of about 0.75 mm in the placement of the sensors and is responsible for only about 3.6 degrees of error. The second source of error is the coupling of the two sensors due to their proximity. This source of error may be eliminated by separating the two sensors by an additional half cycle.

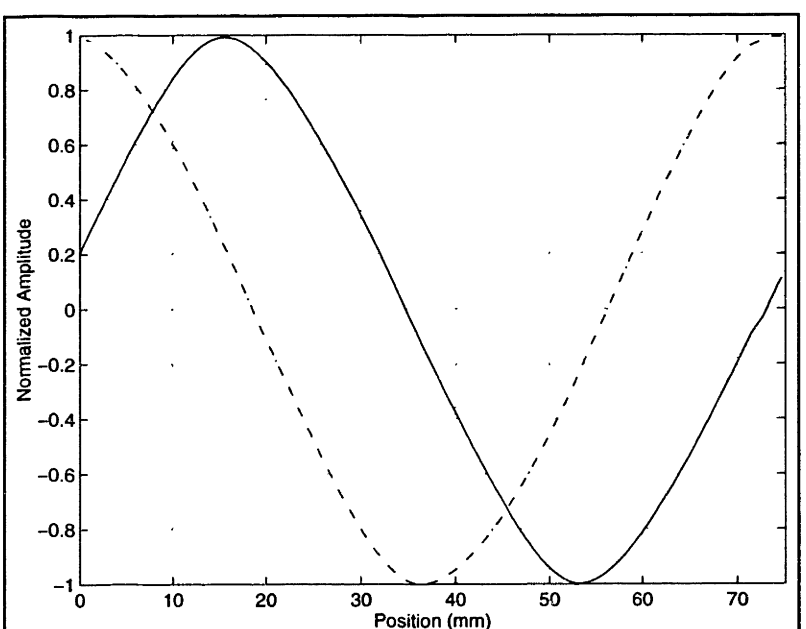

Figure 3.15 Measured sensor coupling functions

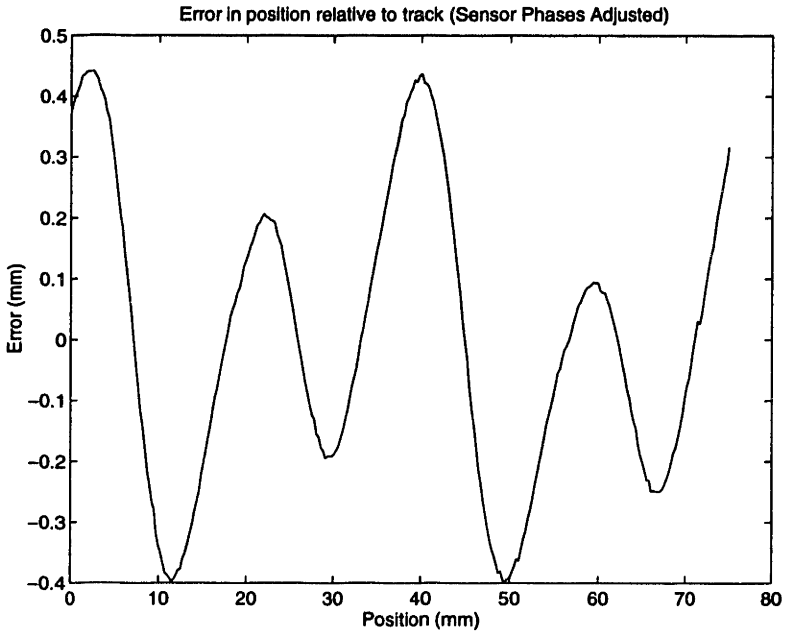

Figure 3.16 Error after phase compensation

Once the phase error in the non-linear observer is corrected, the error of the system was again measured and plotted in Figure 3.16. The correction has improved the error by approximately a factor of 3.5. The primary component of the remaining error is the fourth harmonic relative to the length of a cycle. A careful analysis of the observer near the operating point will show a fourth harmonic error due to the harmonics of the coupling function $\mathcal{F}(\theta_0)$.

It is important to note that this error is virtually identical for every cycle of the winding. Since the measured position is a monotonic function of the actual position, this error may be compensated for with a linearizing lookup table. A table with 256 entries and a linear interpolation function may be used to reduce this source of error. When such a linearization is performed, the resulting error is shown in Figure 3.17. A lookup table was created with 256 entries based upon the five point average of the measured data. This approach can be improved, as an overall 4^{th} harmonic may still be noted in the data. This small harmonic, however, is now on the order of other errors in the measurement.

The error, for the most part, is now below 10 microns. Since the resolution of the milling machine used to perform measurements of the actual position has a resolution of only 25.4 microns, this error is on the order of the measurement error. Thus, a more accurate measurement device such as a laser interferometer must be used to calculate the

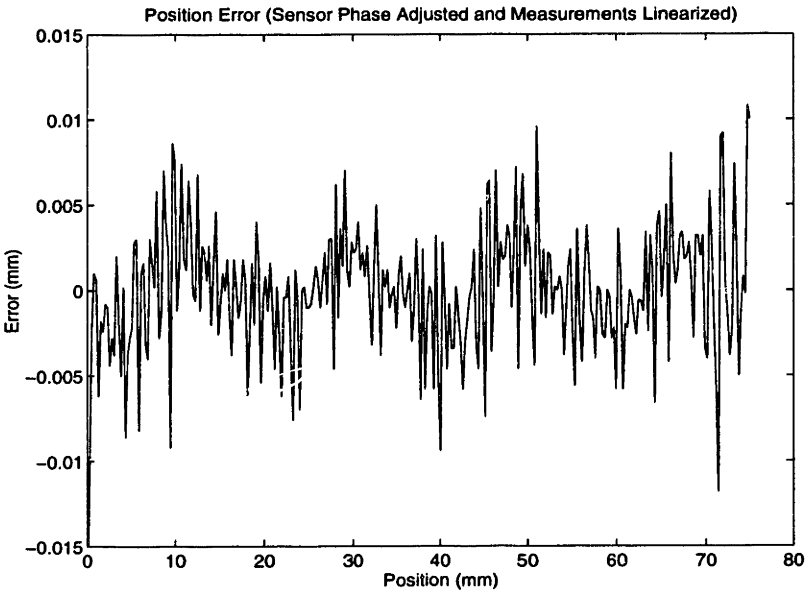

Figure 3.17 Autonomous system error after linearization

actual error of the system. This high accuracy opens up a whole new group of possible applications for this technology, including use in machine tools and optical wafer steppers.

3.4.2 Relative Position-Sensing Accuracy

In order to measure the relative position-sensing accuracy, it is necessary to

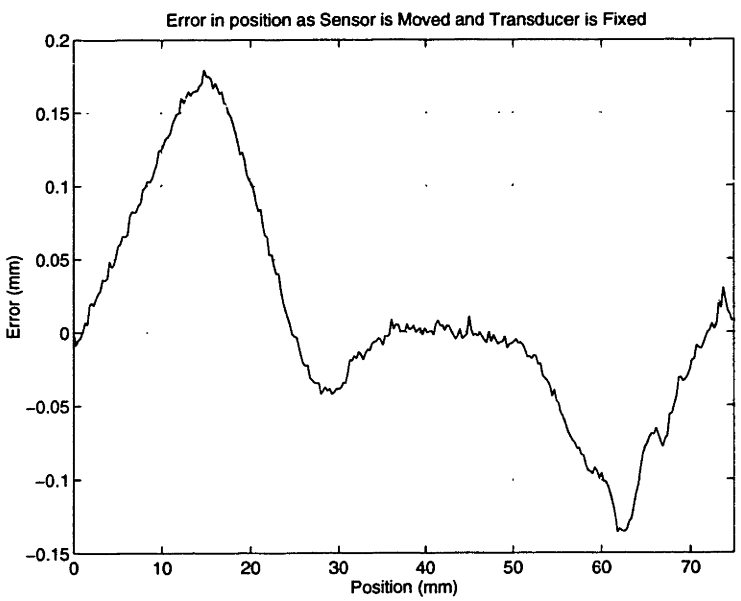

Figure 3.18 Plot of relative position-sensing error as sensors are moved

examine both degrees of freedom - the position of the transmitter and the position of the sensors. The errors along each degree of freedom are first examined independently and then examined simultaneously. Measurements of the position error were first recorded for the case where the transmitter is held in one location and the position of the sensors is varied. This error is plotted in Figure 3.18.

It is interesting to note that the error in the relative position-sensing case is lower than the error in the autonomous case before linearization. This reduced error level is due to the fact that the sensor and transmitter in the relative case link with the winding in exactly the same manner. The coupling function between the vehicle sensor and the vehicle transmitter has lower harmonics than the original coupling function between a sensor and a phase of the winding. The biggest harmonic is approximately five times lower in the relative case, but there are now even harmonics as well as odd harmonics. A plot of the harmonics is shown in Figure 3.19 as calculated from the impedance matrix [1] and the coupling function $\mathcal{F}(\theta_0)$, as a function of transmitter position.

The position error is plotted in Figure 3.20 for the case where the sensor is held fixed while the position of the transmitter is varied. The error in this case is more than double the previous case, and the error is much more sinusoidal. The cause of this error may be determined from a careful examination of the inductance matrix, shown in Equation (2.3). The self-inductances of the phases in the winding vary by as much as 10%. Thus, a stronger signal is generated in some of the phases than originally intended,

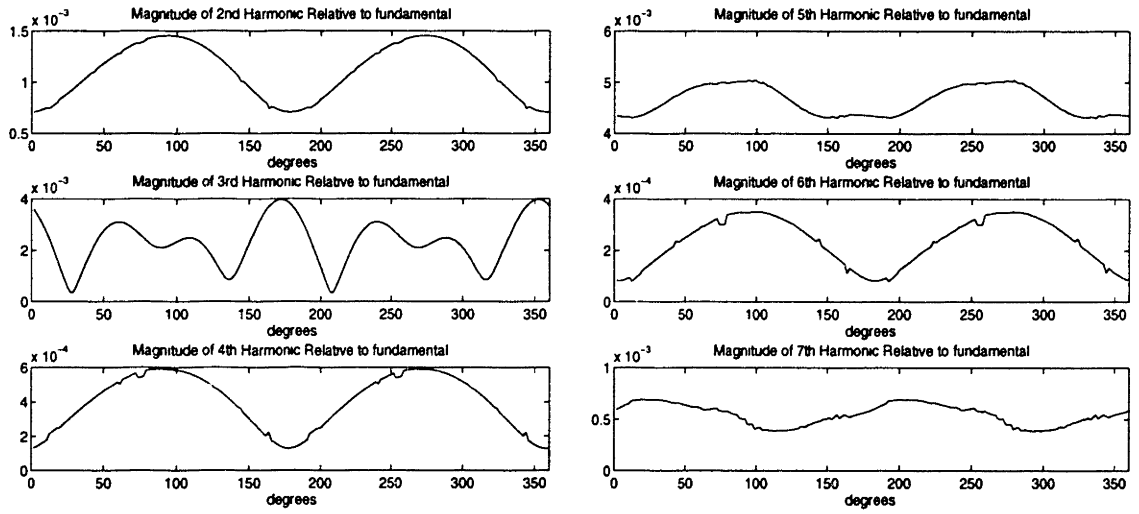

Figure 3.19 Harmonics of coupling function between sensor and transmitter vs. transmitter position

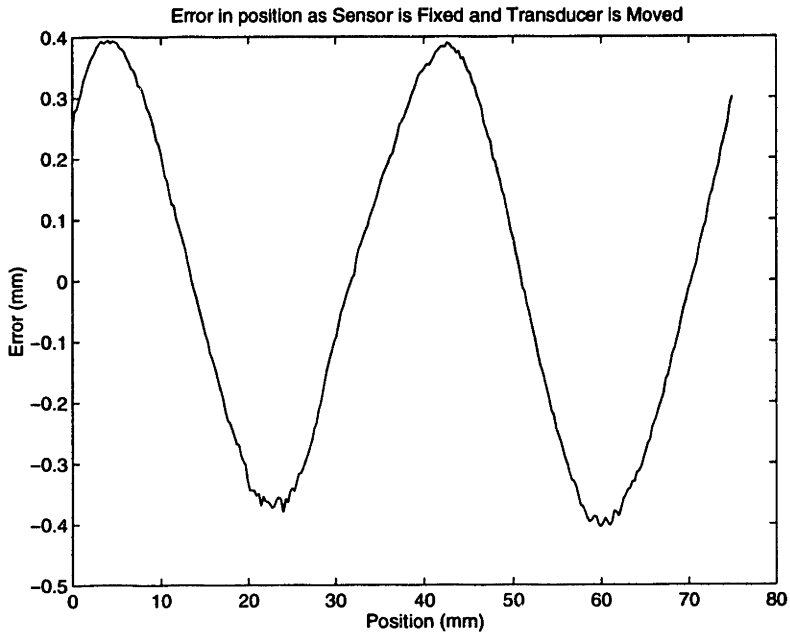

Figure 3.20 Position error as sensor is fixed and transmitter is moved

leading to a variation in signal strength in the winding as the transmitter is moved.

This variation may be calculated directly from the inductance and resistance matrices and the coupling function $\mathcal{F}(\theta_0)$. The predicted accuracy from these measurements is shown below in Figure 3.21. Note that the position does not start at the same point as for the measured error. At the correct sensor position, this predicted error

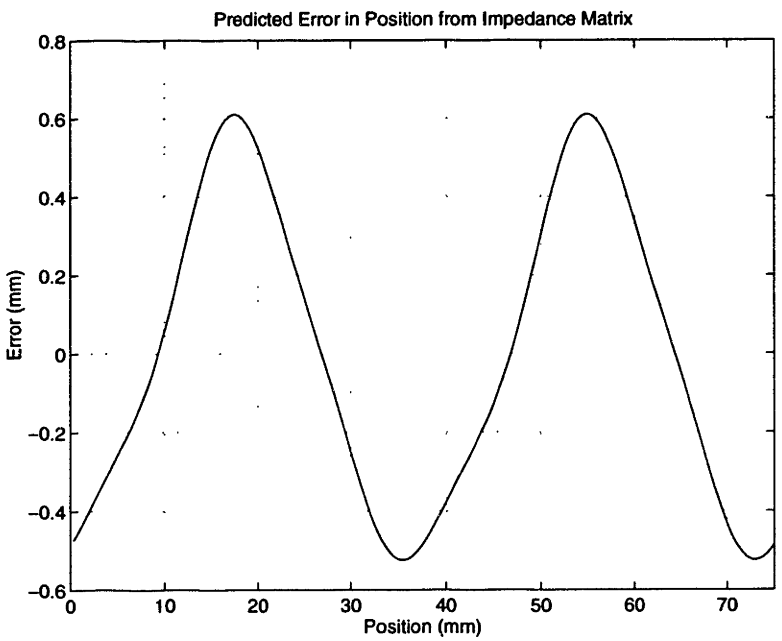

Figure 3.21 Predicted position error as transmitter is moved

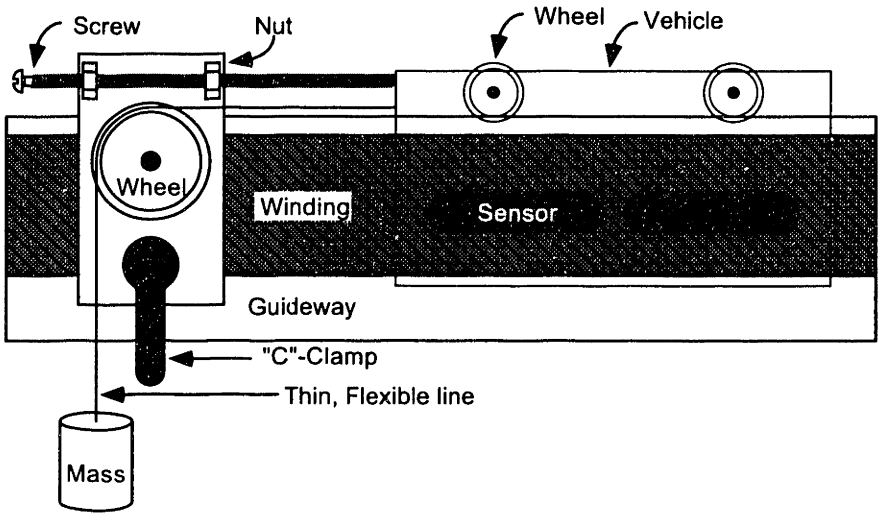

Figure 3.22 Precise screw-based measurement device

agrees quite closely (within 20%) of the actual error.

Finally, the error was measured in a grid across both degrees of freedom. In order to measure both distances accurately, a precise movement device was created as shown in Figure 3.22. The device was clamped to the winding, and the vehicle position varied accurately by turning the screw. Two nuts were used to help lower backlash effects and to hold the screw in a perfectly horizontal position. The vehicle was held to the screw with constant tension on a thin, flexible line generated by a mass. This tension was high enough to overcome any static friction in the wheels, and thus prevent jittery motion of the vehicle.

The plot of error in both dimensions is shown in Figure 3.23. The larger error due to the motion of the transmitter is clear in this figure. It is again possible, through linearization, to improve the error. To do so, it is imperative to know the position values of both axes. This may be accomplished by utilizing the autonomous position-sensing function in conjunction with the relative position sensing function. With a particular value of the sensor position, the relative position, and a two dimensional table, it is possible to reduce the error further.

It should be clearly noted that the accuracy of both features is more than an order of magnitude more accurate than is necessary for the applications described, even before linearization. This high accuracy opens up possibilities of utilizing these techniques for

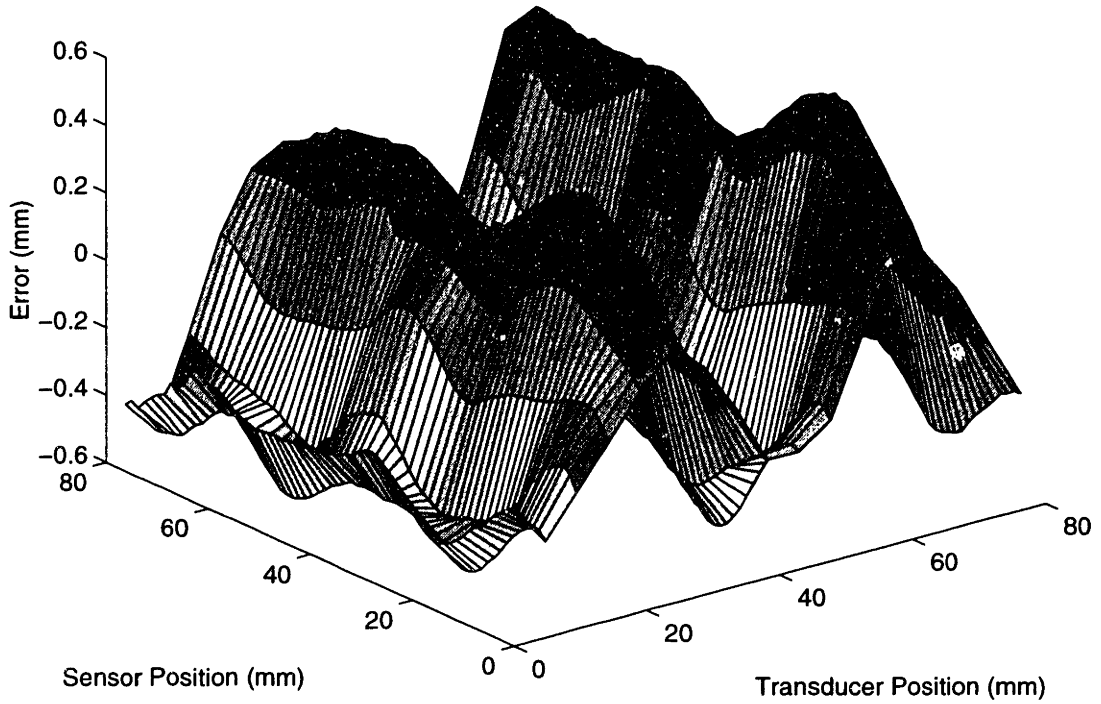

Figure 3.23 Plot of relative error vs. sensor and transmitter positions

applications other than transportation. These applications include machine tools and optical wafer steppers.

3.5 Noise Immunity

The techniques developed in this thesis are designed to possess a fair amount of noise immunity. The bandpass input filter had better than 50 dB of rejection outside the pass band. The bandwidth of this filter could easily be reduced to a value closer to the 160 Hz bandwidth of the modulated signal, for additional noise immunity, with no increase in filter complexity. The rejection of the filter could also be doubled to 100 dB simply by doubling the order of the filter. Thus, the noise immunity of these techniques may easily be tailored to the expected operating environment.

The PLL utilized to lock to the carrier signal also has some noise-immunity features. PLLs are widely known for their ability to lock on to a signal, even in extremely

low signal-to-noise ratios. By narrowing the lock in range, the PLL will lock more slowly but will be more resistant to spurious inputs. Widening the range allows fast lock, but lower immunity to noise. As specified previously, the best of both cases may be achieved through a flexible discrete-time implementation.

As specified earlier, one of the primary reasons for utilizing an observer in the system was for its noise rejection characteristics. The particular non-linear observer utilized was designed upon the assumption of uncorrelated Gaussian additive noise on the input. One of the features of an observer is that there is a tradeoff between fast tracking and noise immunity. By making the dynamics faster, the system locks on faster, but is less immune to noise on the input. By designing the slower dynamics, the system takes longer to lock, but is much more resistant to noise on the input. Thus again, the observer may be tailored to the expected noise in the system. With the flexibility of a DSP implementation, it is possible to use a fast observer to acquire lock and a slow observer once this lock is acquired, and gain the best features of both.

One of the sources of noise in a linear motor implementation is the switching of the motor drive. The implementation utilized in this thesis, as specified in Chapter 2, possessed the feature of common-mode rejection of propulsion currents in the winding. The rejection ratio of the sensors was on the order of 200 to 1. Some of the propulsion signal does, however, register in the sensors, and care should be taken to make the system robust to such noise. In the system designed in [1], the position-sensing feature operated satisfactorily and reliably in the presence of the propulsion signal source. A low Q analog filter was used to reject what little signal was introduced.

The types of noise created by the propulsion signals should be examined in order to reveal the best manner in which to make the system robust to these sources. A plot of the power spectral density (PSD) of an electronically commutated propulsion source used in [1] is illustrated in Figure 3.24 for the case of maximum vehicle velocity. The top plot shows the spectrum of the voltage signal, and the bottom plot shows the spectrum of the current in the winding. This signal contains many harmonics of the square wave commutation signal, the fundamental of which is at the 80 cycles per second switching frequency. As can clearly be noted from the diagram, separating the position-sensing

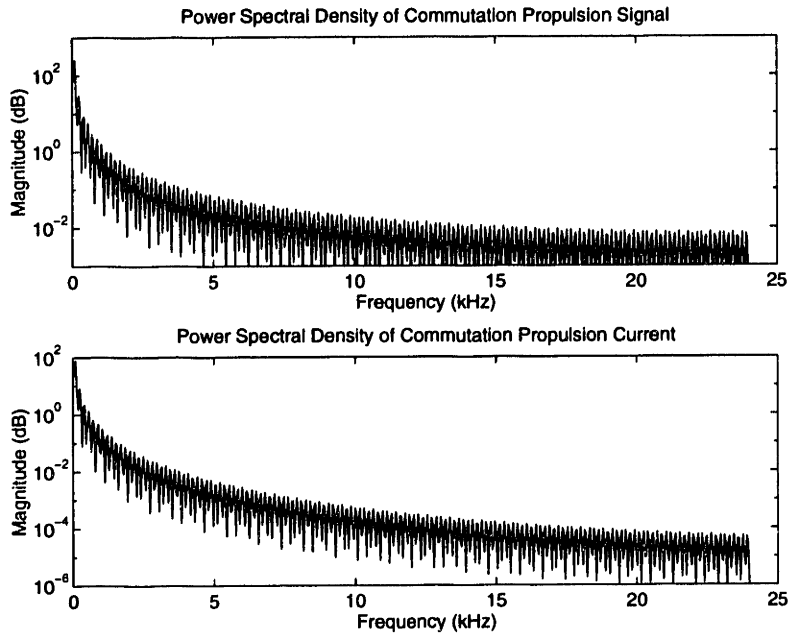

Figure 3.24 Power spectral densities of commutated motor voltage and current

signals in frequency from the main harmonic of the drive can aid a great deal in the noise rejection of the system. The propulsion signal is about five orders lower in magnitude at 10 kHz than at its drive frequency of 80 Hz. Thus, a position sensing system implemented with filtering can greatly reduce the effect of any propulsion signal by utilizing the correct frequency bands.

A second common type of motor drive is the pulse width modulated (PWM) type. By varying the duty ratio of a high frequency square wave, more sinusoidal currents may be achieved in the motor winding due to its filtering effect. A plot of the PSD of the PWM signals is shown in Figure 3.25 for a 50 kHz cycle rate and maximum vehicle velocity. This switching frequency is reasonable for a drive used in low to middle power level systems. It is clear from this plot that there are large troughs between the 50 kHz harmonics where there is a low signal level from the PWM drive. By locating the position-sensing signals at frequencies in these troughs and away from the harmonics, many possible problems may be avoided. Again, filtering is key to separating the position-sensing signals from the propulsion signals.

While at least moderately sensitive to propulsion signals on the same winding, the design of the system should be fairly robust to external interference and noise. From a

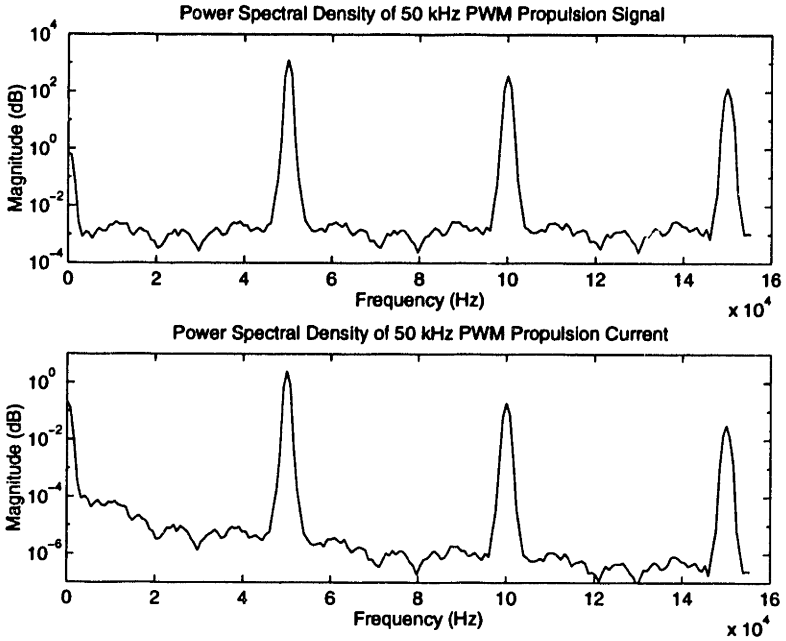

Figure 3.25 Power spectral densities of PWM voltage and current

distance of ten or more widths of the winding, the winding appears very much like a set of twisted pairs which inherently shield the system from external fields. In order for a signal source to couple well with the winding, it is necessary to locate the source in close proximity to the winding. With proper design of the guideway, most noise sources will be located far enough away from the winding so as not to couple well and disturb the system. One final manner in which noise in a system may be accommodated is by means of changing the signal levels used in the transmitters. The signal levels may be tailored to the noise level in any particular system to achieve an acceptable signal-to-noise ratio.

In order to confirm the noise immunity of the techniques in this thesis, the demonstration system was tested in a low signal-to-noise environment. The position-sensing signal levels were reduced until the ambient noise in the winding became significant. Additional white noise was added to the system in order to further disturb the input. The nature of the added signal was uncorrelated, Gaussian white noise, bandlimited to 24 kHz. This noise was added at a signal level comparable to the position-sensing signal levels in the system. Thus, the overall signal-to-noise ratio of the test environment was on the order of 0 dB or lower.

The added noise is shown in the top plot of Figure 3.26. The bottom two plots depict typical signals received at the sensors. As noted, this noise level is very significant relative to the position sensing signals. A sizable portion of the noise in these plots is due to the noise in the windings generated by a nearby switching power supply. In this harsh environment, the position sensing system had no trouble locking on to the signals, and operated reliably in the presence of significant noise. With the proper design of the system, where noise immunity is built into its constituents, and with proper choice of spectrum location for the position-sensing signals, a system may be created which is very robust to noise.

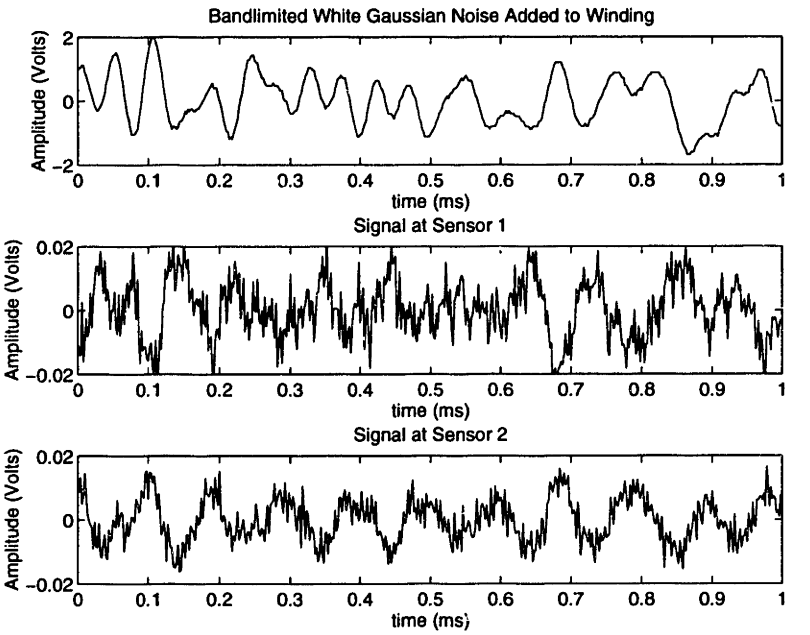

Figure 3.26 Band-limited white noise, and very noisy sensor signals

Chapter 4

Communication

4.1 Introduction

Communication is a very useful feature in a transportation control system, and is used to report vehicle status, destinations, speed, and other required information. Typically, such a system is implemented through a radio link. It will be demonstrated that the architecture utilized for the features described in this thesis requires no additional hardware for the addition of communication link. Communication between vehicles or between wayside and vehicles is possible using the same signal path utilized for position sensing signals.

Two communication methods have been developed to transfer data across the helical winding. The first method utilizes a synchronous receiver approach. The term synchronous, in this instance, refers to synchronization of the receiver with vehicle position. This system may be utilized with a wide variety of modulation techniques able to transmit over a high-pass channel.

The second method utilizes an asynchronous receiver, which has the advantage that it is not dependent on knowledge of vehicle position for operation. This method was implemented on the demonstration system at a rate of 1500 bits per second, and utilizes a form of frequency shift keying (FSK) modulation. In this approach, separate frequencies are used to transmit each bit or symbol (if more than one bit per symbol). The techniques

used to implement the asynchronous receiver are equally applicable to other forms of modulation such as quadrature amplitude modulation (QAM).

In the following sections, the transmission from wayside to vehicle will be addressed. The techniques used may be applied to vehicle to vehicle communication without any modification. The techniques may be modified slightly for vehicle to wayside communication due to the fact that there are three sensors at the wayside in the demonstration system instead of two. These changes are very minor, and the basic techniques do not change.

4.2 Position-Varying Channels

It is necessary to understand the transmission of a communication signal between a wayside transmitter and vehicle sensors in order to implement a communication feature in this system. Suppose it is desired to transmit a modulated signal *s(t)* from the wayside to the receiver on the vehicle. A current signal proportional to *s(t)* is driven through the transmitter of one of the phases of the winding. The signals received in the sensors on board the vehicle are:

$$\begin{aligned} s_1(t) &= \hat{s}(t)\cos(\theta) \\ s_2(t) &= \hat{s}(t)\sin(\theta) \end{aligned} \tag{4.1}$$

As with the position-sensing signals, the amplitude of signals coupled at the two sensors depends upon the position of the vehicle, θ. The original signal *s(t)* is filtered by the characteristics of the channel to form the component $\hat{s}(t)$. The synchronous receiver attempts to correct for the dependency on position, and a channel equalizer may be utilized to compensate for the secondary filtering effect [25].

4.3 Synchronous Receiver

One possible compensation for position dependency would be to send to the receiver the largest amplitude signal from either sensor. Care must be taken not to change the sign of the signal, so one small adjustment must be made. Consider four

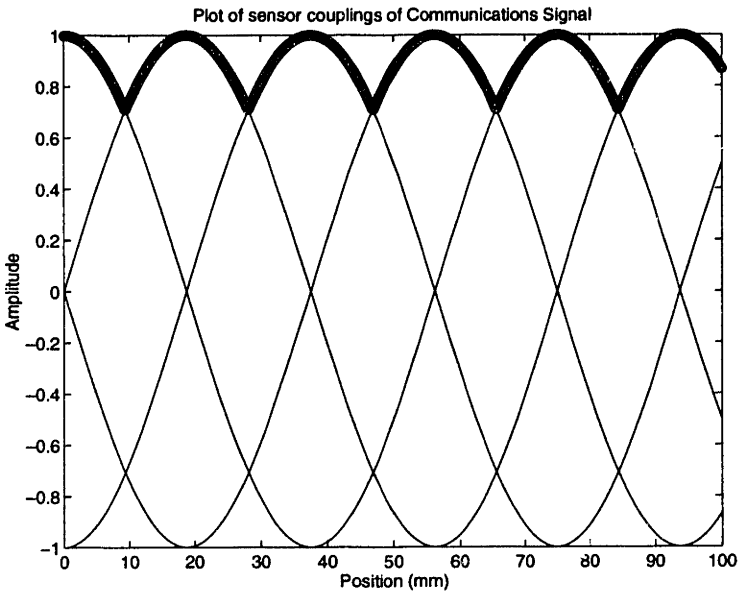

Figure 4.1 Synchronous rectification of communication signals

signals - the two at the sensors and the negatives of these signals. It is possible to forward to the receiver the signal with the largest positive amplitude, and thus not inadvertently change the sign of the signal when a transition from one of the four signals to the next takes place. This technique is exactly the function of a synchronous rectifier, the behavior of which depends upon the coupling envelopes of the signals. This rectifying effect is shown in Figure 4.1, along with the coupling envelopes of the four signals.

Thus, as position changes, the choice of signal sent to the receiver would change every quarter cycle of the winding. In order to know which signal to forward to the receiver at any point in time, it is necessary to know the position of the vehicle. The autonomous position-sensing feature described in Chapter 3 could be used for exactly this purpose. For the communication between vehicles, the relative position-sensing feature would fulfill this requirement. This technique leaves a position-dependent ripple on the signal sent to the receiver. While certain receivers may be able to compensate for this distortion, the ripple may be easily removed with a position-dependent gain which repeats every quarter cycle. The necessary position-dependent gain is illustrated in Figure 4.2.

The synchronous receiver has effectively linearized the channel, removing any position dependence from the communication signal. The communication now has a linear, time-invariant channel over which to communicate, and any modulation and

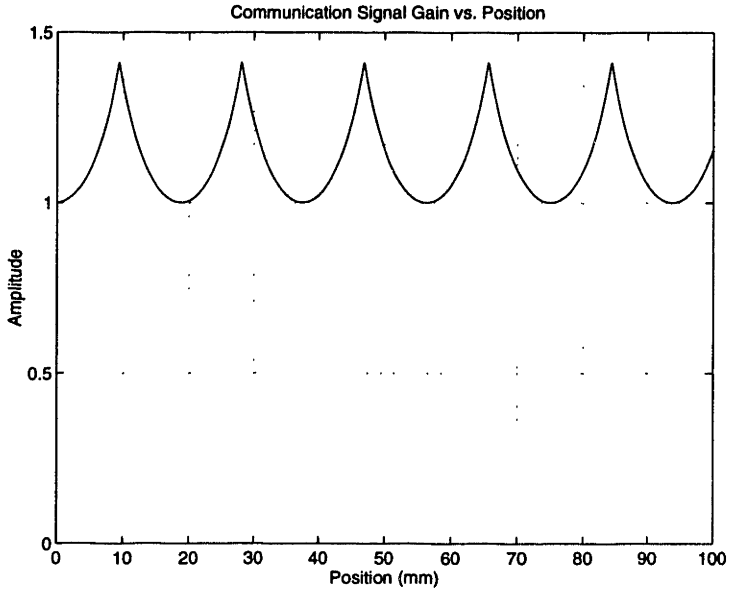

Figure 4.2 Position-dependent gain

communication technique which can endure the characteristics of this channel (high-pass, etc.) may be utilized. Note that the only modification necessary to support vehicle to wayside communication in this system is the use of a six-pulse rectifier instead of a four-pulse rectifier, used in conjunction with the three phase sets of sensors.

4.4 Asynchronous Receiver

It is desirable, in many cases, to possess a communications system which is not dependent upon knowing the position of the vehicle. For instance, if the communication system is used to assign carrier frequencies for the position-sensing features, it is necessary that the communication system not be dependent of the position-sensing system. Also, in the case where a component has failed and the position sensing feature is not in operation, there is significant merit in still possessing communication capability.

A second system was designed to operate while fulfilling this requirement and is appropriately called an asynchronous receiver, since it does not require any synchronization with the position of the vehicle. This system does, however, require some knowledge of the specific modulation techniques utilized. This method was implemented in the demonstration system using a frequency shift keying (FSK) approach,

where a separate frequency is utilized to transmit each symbol. The pulse shapes in an FSK pulse are shown below in equation (4.1)

$$p_i(t) = \begin{Bmatrix} A\sin(\omega_i t) & 0 \le t < T \\ 0 & otherwise \end{Bmatrix} \tag{4.1}$$

The analog signal transmitted is thus

$$s(t) = \sum_{m=-\infty}^{\infty} A p_{d[m]}(t - mT) \tag{4.2}$$

where d[m] is the digital signal to be transmitted.

In this technique, a separate receiver is used for each of the two channels, as shown in Figure 4.3. Each receiver was designed to output a separate detection level for each possible symbol. In this demonstration technique, only two symbols were used to represent either a zero bit or a one bit. A bandpass filter is utilized in each receiver to isolate the communication signals from position-sensing and propulsion signals and noise in the winding. The filter is implemented with a 128 tap, optimal, linear-phase filter. The transfer function shown in Figure 4.4. Note that this filter has about 30 dB of rejection in the stop band. This rejection could be increased above 60 dB through the use

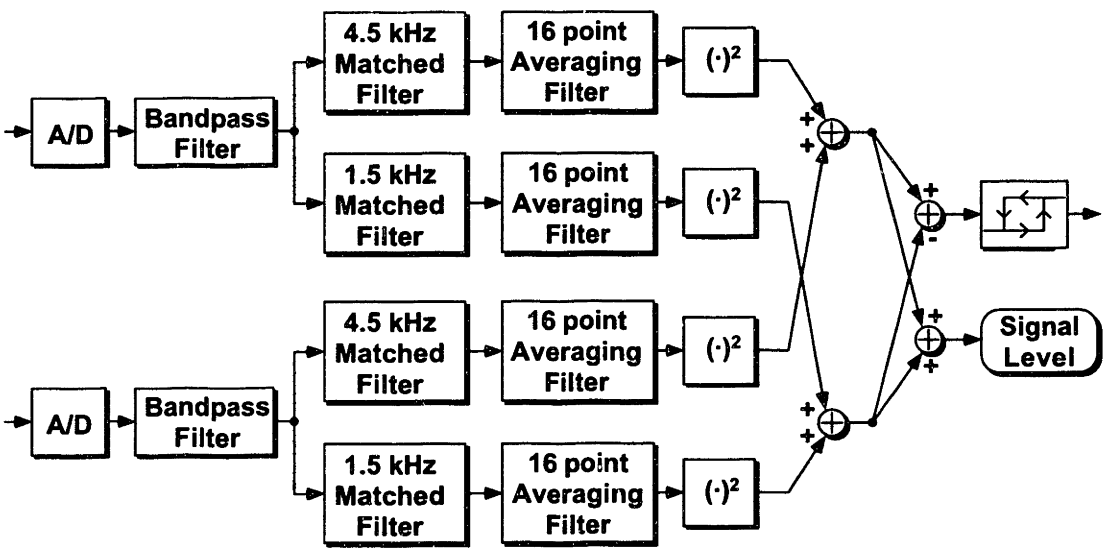

Figure 4.3 Asynchronous FSK receiver

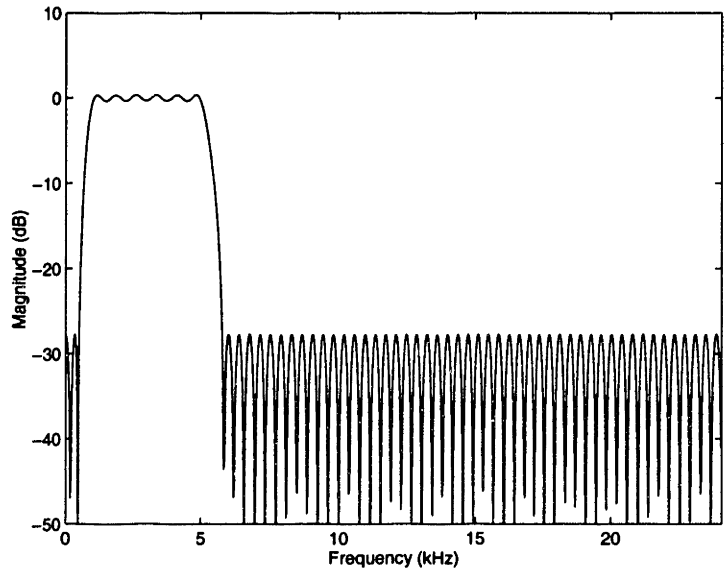

Figure 4.4 Communication filter magnitude

of a 256 tap filter.

A filter matched to the transmitted symbol, followed by a 16 point averaging filter, was used to detect the level of each symbol in the communication signal for each phase. Since the level of the signal from sensor 1 is proportional to the cosine of position, the detected levels of the zero and one symbols at the output of the first receiver will also be proportional to $\cos(\theta)$. The level of the signal from sensor 2 is proportional to the sine of position, so in a likewise manner the detected levels of the zero and one symbols at the output of the second receiver will both be proportional to $\sin(\theta)$.

In order to remove these dependencies on position, the detected zero symbol levels are squared and added and in a likewise manner the detected one symbol levels are squared and added. Due to the trigonometric identity $\sin^2(\theta)+\cos^2(\theta)=1$, these two signals are now proportional, respectively, to the square of the detected one symbol level and the square of the detected zero symbol level. While it is possible to take the square root of these new signals, it is computationally intensive and unnecessary in a practical implementation. By subtracting the squared zero symbol level from the squared one symbol level, a new signal is acquired which is negative when a zero is transmitted and positive when a one is transmitted. In order to add an extra level of noise immunity of the

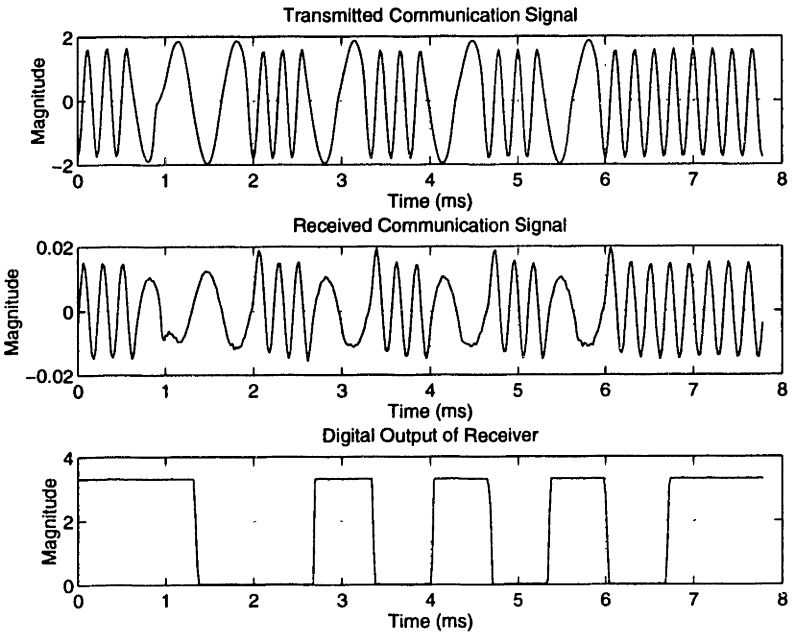

Figure 4.5 Transmitted and Received Communication Signals

communication system, hysteresis was added to the decision slicer (bit level detector). By adding the two signals, a separate signal level detector may be created.

For the implementation of this technique in the demonstration system, one cycle of a 1.5 kHz signal was utilized for a zero symbol pulse and three cycles of a 4.5 kHz signal were utilized for a one symbol pulse. The signal level of the zero symbol was boosted to account for the attenuation of the winding, but no other equalization was performed on the channel. An RS232-C protocol was utilized for the transmission of information over the channel so that timing recovery was simplified. The bit level of the slicer was output on a digital port of the DSP and fed in to the RS232-C receive pin on the same processor. A rate of 150 bytes per second was thus achieved over the communication link. This implementation produced a robust communication system over the winding, regardless of the movement of the vehicle.

Figure 4.5 illustrates the FSK signal at the input of the transmitter, the received signal at one of the two sensors, and the output of the receiver for a transmitted ASCII 'U'. Note that the output of the receiver is delayed from the input due to filter delays. Figure 4.6 illustrates the overall levels of the detected one symbol and detected zero symbol. This figure also shows the input and output of the slicer, and the effect of the hysteresis is clearly evident.

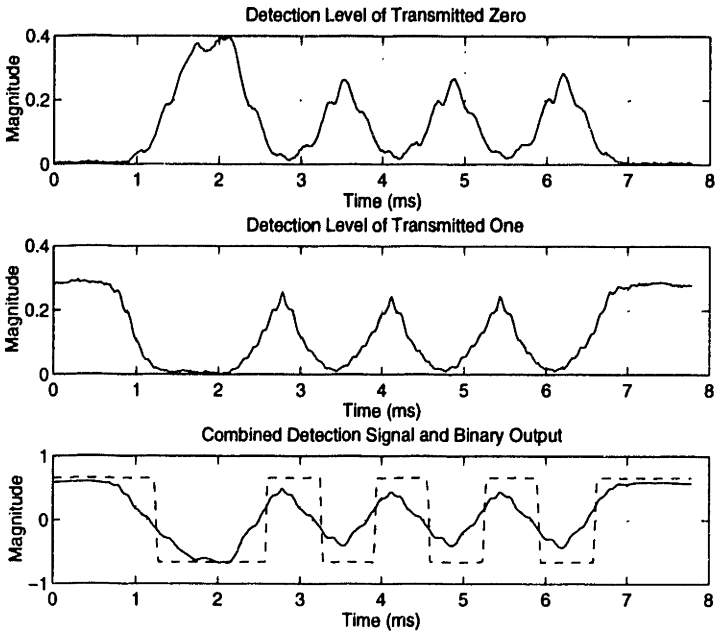

Figure 4.6 Detected zero and one levels and slicer input and output

4.4.1 Alternative Modulation Techniques

Similar techniques may be utilized with other modulation methods to acquire higher bit rates. For instance, these techniques could be utilized with minimum shift keying (MSK) with no changes to double the bit rate. This approach may be used with almost any communication scheme which derives signal levels as outputs. For instance, these techniques can also be extended for use in a quadrature amplitude modulation (QAM) system. Since the output level is squared with this approach, only one quadrant of the QAM constellation would be usable since the sign of the output levels are lost. One further change necessary would be the use of a non-linear decision slicer, since the output signal levels are squared. Although the square root of the levels could be taken, the numerical routine would be computationally intensive. It is simpler to take the original slicer levels, and square them for use in a non-linear quantized slicer.

A related approach to communication would square the 2 sensor inputs directly after the bandpass filter and before the receiver, as contrasted with squaring them after the receivers in the previous method. Using an FSK transmission, the resulting signal is still

an FSK signal with a DC offset and doubled frequencies as shown in Equation (4.3). Thus, a standard FSK receiver may be utilized to receive the signal. A similar method may be used with a QAM receiver, with the use of a pulse shape that is a squared version of the original.

$$\begin{aligned} r_i(t) &= A^2 \cos^2(\omega_i t + \varphi)\left[\cos^2(\theta) + \sin^2(\theta)\right] \\ &= A^2\left[\frac{1}{2} + \frac{1}{2}\cos(2\omega_i t + 2\varphi)\right] \end{aligned} \tag{4.3}$$

Chapter 5

Alternative Architectures

The techniques developed in this thesis may be utilized on a wide variety of winding types. Several compatible winding structures will be described with their related characteristics. One variation on the architecture utilized for the system is the use of two vehicle transmitters and one vehicle sensor instead of one vehicle transmitter and two sensors. While not very practical, this approach gives some insight on how to create a system with only a single winding phase, which is also discussed in this chapter.

5.1 Alternative Windings

The winding utilized for the implementation of the demonstration system in this thesis is a helical winding with 24 wire bundles, 12 mini-phases, and 6 position sensing phases. The utilization of this winding was a practical matter, and such an approach would likely only be used in the case where the winding has a primary purpose as the winding of a linear motor. In the case where the sole purpose of the winding is for a position-sensing and communication system, a simpler winding structure would be more practical. A reduction in the amount of copper used in a winding reduces its cost, and the number of connections to be made to the winding increases the cost. Thus, it is desirable to implement a winding which has as few phases possible and still meets the requirements of the position-sensing and communication techniques.

The primary manner in which the winding may be simplified is in the reduction of the number of phases. Only one phase is required for the autonomous position-sensing and wayside to vehicle communication, but more are required for relative position sensing, virtual marker tracking, and vehicle to vehicle and vehicle to wayside communication. The minimum number of phases required to implement these techniques, with no changes in the algorithms, is two. The two phases in this case should be spaced a quarter cycle apart, in quadrature.

More than one phase is necessary since the amount of signal linked into a single phase by a vehicle transmitter crosses zero at regularly spaced intervals along the winding. Two phases are enough to carry all the information necessary to implement the techniques in this thesis. The derivation of the vehicle transmitter to vehicle sensor coupling is simple to prove in the two phase case, and is shown in Equation (5.1). In the general case, there is a scaling of this solution by N/2 where N is the number of phases. In summary, the number of phases may be reduced by a factor of three.

$$\begin{aligned} V_{vehicle2} &= V_1 \cos(\omega_c t + \beta) \cdot \begin{bmatrix} \cos(\theta)\cos(\theta+\phi) \\ +\cos(\theta-\pi/2)\cos(\theta+\phi-\pi/2) \end{bmatrix} \\ &= V_1 \cos(\omega_c t + \beta) \cdot \left[\cos(\theta)\cos(\theta+\phi) + \sin(\theta)\sin(\theta+\phi)\right] \\ &= V_1 \cos(\omega_c t + \beta) \cdot \cos(\theta+\phi-\theta) \\ &= V_1 \cos(\omega_c t + \beta) \cdot \cos(\phi) \end{aligned} \tag{5.1}$$

The second manner in which the winding may be simplified is by utilizing only one mini-phase per position-sensing phase. In the demonstration system, two mini-phases were connected in a loop to form a position sensing phase and propulsion phase. With no propulsion system, there is no need for the common-mode/differential mode winding structure. Each individual mini-phase may be connected in a loop, and all of the other characteristics of the winding are maintained. Thus, the number of mini-phases in the winding may be cut by another factor of two.

The third manner in which the winding may be simplified is through the reduction of the number of wire bundles per mini-phase. Only one bundle is truly necessary with an

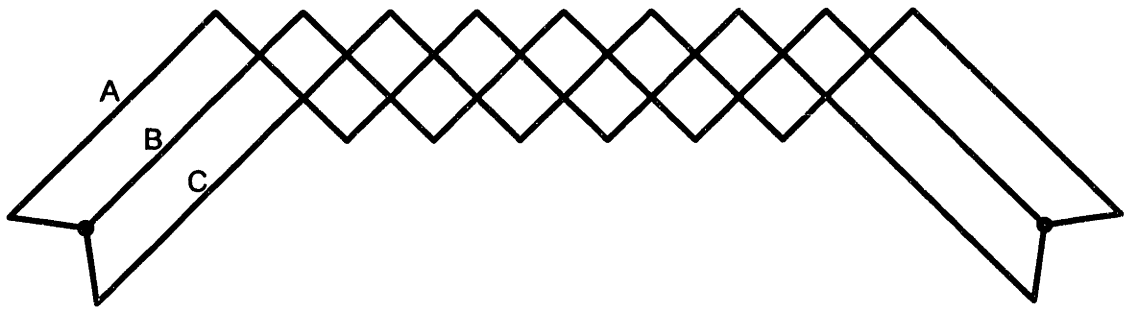

Figure 5.1 3-phase implementation with only 3 wire bundles

appropriate winding structure. A return for the position sensing currents is still necessary, but a common ground may be used for all of the phases. The one limitation for this approach is that the ground return must have a much lower impedance than a phase of the winding.

A related approach which has a lot of merit is a winding structure composed of only three single-bundle phases, as illustrated by Figure 5.1. The phases at either end of the winding are connected together in a wye configuration. Consider the signals introduced into the winding by a vehicle transmitter. The three signals form a three phase set, and thus the currents at the end of the bundles add to zero (assuming the three impedances are equal); no ground return is required!

Likewise, consider that a signal driven by a wayside transmitter in the *A* phase bundle returns equally through the other two bundles. Since the bundles form a three phase set and the returning currents flow in the opposite direction in the alternate two bundles, the signal received by a sensor is of the correct form. This fact is shown in Equation (5.2). Thus, a winding with only 3 wire bundles is sufficient for the techniques outlined in this thesis, and is perhaps the best approach. This particular configuration even competes well with the single phase approach discussed later in this chapter, which uses 2 wire bundles and significantly more complexity in the signal demodulation scheme.

$$\begin{aligned} V_{vehicle2} &= V_1 \cos(\omega_c t + \beta) \cdot \left[\cos(\theta) - \frac{1}{2}\cos(\theta - 2\pi/3) - \frac{1}{2}\cos(\theta + 2\pi/3) \right] \\ &= V_1 \cos(\omega_c t + \beta) \cdot \left[\frac{3}{2}\cos(\theta) \right] \end{aligned} \tag{5.2}$$

5.2 Dual Transmitters, Single Sensor

Another possible variation in the architecture is to change the configuration of sensors and transmitters. Clearly, for instance, it is possible to expand the number of vehicle sensors, equally space them, and achieve the same results with very minor changes to the algorithms. One particularly interesting case is the dual of the approach taken in this thesis -- to utilize two vehicle transmitters spaced a quarter cycle apart and only a single vehicle sensor.

This approach requires the use of two separate frequencies for the two transmitters. Otherwise, the signals generated in the winding are similar in form to the signals driven by a single transmitter. It is also necessary to use harmonically related frequencies for the two sensors. It is not possible to track the two carriers individually, as the signals received at the sensor will each go through zero crossings at regular vehicle spacings. One successful approach would be to use harmonically related carriers of the following form:

$$\begin{aligned} \cos(\omega_{c1}) &= \cos((N-1)\omega_c) \\ \cos(\omega_{c2}) &= \cos(N\omega_c) \end{aligned} \tag{5.3}$$

A phase locked loop (PLL) may be created to track the fundamental frequency ω_c from the two inputs. Each of the inputs may be squared, high pass filtered to remove the DC component, and fed into a separate multiplying phase detector. With the digital implementation of the phase locked loop, it is possible for the PLL to output two frequencies. Each output frequency in this case would be double the carrier frequency in Equation (5.3). These outputs would be fed back to the two phase detectors. The two phase detector outputs would be summed to feed the innovation to the PLL filter. Due to the signal couplings, one phase detector output signal would be proportional to the cosine of position squared and the other would be proportional to the sine of the position squared. Thus, there would be a constant input to the PLL to keep it locked. The PLL would also have dual demodulation outputs to demodulate the sensor signals.

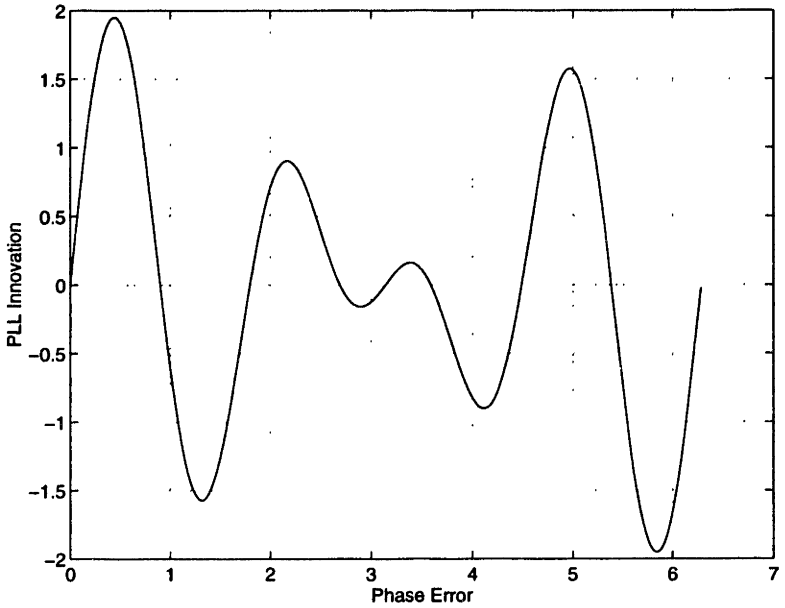

Figure 5.2 PLL innovation vs. fundamental phase error

There is a slight problem with this implementation which must be overcome. The PLL with the input from the two phase detectors has N stable equilibrium points, as shown in Figure 5.2, with N=4. Thus, the system is not guaranteed to lock at the correct fundamental phase. One solution is to utilize a two step locking scheme. Initially, only one of the two phase detectors is utilized to lock the PLL onto the strongest of the two signals. The system is locked onto one of N possible, equally spaced choices for the fundamental phase.

When the other signal is strong enough to acquire an accurate reading, a choice among the N possible phases is made. In order to make the correct choice, the system utilizes one cycle of the signal not used for the initial lock. This signal is then correlated with the N possible outputs of the PLL. The correct choice will have the largest positive correlation, as the N correlations will have the amplitudes of N equally spaced points of one cycle of a cosine function. With the proper phase locked, both phase detectors are put into operation.

Although this system would work, the locking mechanism is unnecessarily complex, considering the alternative described throughout most of this thesis. This solution does, however, pave the way for the implementation of the techniques in this thesis on a single phase winding implementation.

5.3 Single Phase Implementation

While the original approach in this thesis is attractive for short headway systems, the economics change when the system is to be used for long headway transportation systems. For long headway systems, where the vehicles are separated by long stretches of empty guideway, it may make sense to implement a system with a simpler, less expensive winding at the cost of implementing more complex algorithms on board each vehicle. While it is clear that a system with a single phase cannot be implemented with a single transmitter, a system with two vehicle sensors and two vehicle transmitters may be implemented successfully. Again, consider the case where two harmonically related carriers specified in Equation (5.3) drive the two transmitters. Four signals are received at the sensors -- two frequencies at each sensor. These four signals may be separated through filtering, and are of the form:

$$\begin{aligned}
V_{t1s1} &= V_2 \cos(\omega_{c1} t + \psi_1) \cdot \left[\frac{1}{2}\cos(\theta)\cos(\theta+\phi)\right] \\
V_{t1s2} &= V_2 \cos(\omega_{c1} t + \psi_1) \cdot \left[\frac{1}{2}\cos(\theta)\sin(\theta+\phi)\right] \\
V_{t2s1} &= V_2 \cos(\omega_{c2} t + \psi_2) \cdot \left[\frac{1}{2}\sin(\theta)\cos(\theta+\phi)\right] \\
V_{t2s2} &= V_2 \cos(\omega_{c2} t + \psi_2) \cdot \left[\frac{1}{2}\sin(\theta)\sin(\theta+\phi)\right]
\end{aligned} \tag{5.4}$$

Squaring and adding the first two signals and squaring and adding the second two signals leads to the same form of inputs as is used in the dual phase detectors of the previous section. Thus, the demodulation scheme of the previous section may be used to recover the two carriers. These carriers may be used to demodulate the four coupling envelopes of Equation (5.4). By linearly combining the first and fourth coupling envelopes, and combining the second and third coupling envelopes, signals of the same form used in the original non-linear observer may be created.

$$C_{1+4} = \cos(\theta)\cos(\theta+\phi) + \sin(\theta)\sin(\theta+\phi) = \cos(\phi)$$
$$C_{2-3} = \cos(\theta)\sin(\theta+\phi) - \sin(\theta)\cos(\theta+\phi) = \sin(\phi) \tag{5.5}$$

With the proper signal processing, the demodulator of the previous section, and the original non-linear observer, the relative position sensing feature is enabled on a single phase winding.

Chapter 6

Implementation Issues

A variety of implementation issues related to the techniques in this thesis should be addressed. Details of the actual design of the demonstration system will be divulged. Also, the actual hardware design of the sensors in the demonstration system will be described, as it is not simple to orient them as shown schematically in Chapter 3. One of the limitations of the position-sensing techniques developed in this thesis is that only the phase within a cycle is determined directly. In order to track actual distances, initial estimates of positions must be acquired. The acquisition of these estimates is the topic of Section 2. A short description of how the techniques in this thesis may be adapted to two very different control strategies is conferred in Section 3. Finally, a description of the practical limitations of the system is given in Section 4.

6.1 System Implementation

The techniques described in this thesis were implemented on a Motorola DSP56303 digital signal processor (DSP). A Motorola evaluation board was utilized for code development. The board contains a Crystal Semiconductor CS4215 audio codec which was utilized for the sampling of the signals in the system. The stereo codec was operated at the maximum rate of 48000 samples per second. An image of the processing board and associated liquid crystal display is shown in Figure 6.1.

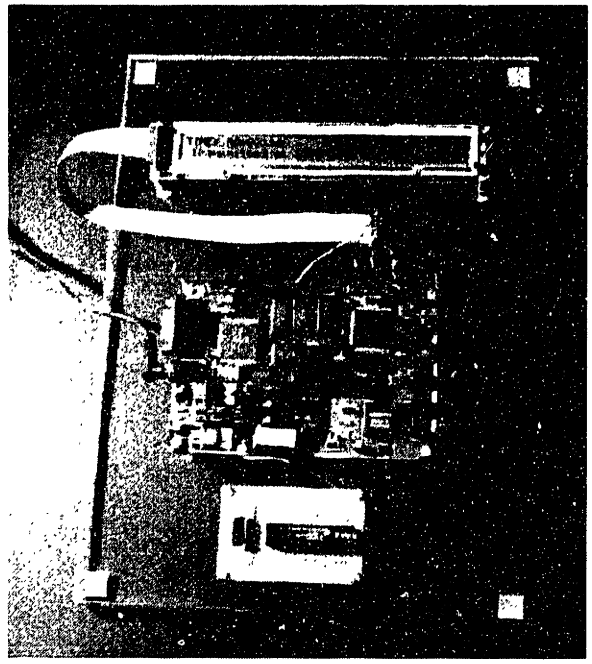

Figure 6.1 DSP board and display

Although most of the techniques in this thesis could have been implemented in analog hardware, a digital implementation has many advantages. With a digital implementation, the designer does not have to worry about such mundane problems as component drift, component value accuracy, and offset voltages. Also, with many fewer components, the system is more reliable.

An implementation on a DSP is also inherently more flexible. With a single piece of generic hardware, a wide variety of algorithms may be implemented. The hardware of the system does not need to be modified, in most cases, to accommodate changes in algorithms. With sufficient processing power, multiple features may also be implemented with a single DSP. Up to three of the features described in this thesis could be implemented on the specified development board simultaneously. Faster DSPs exist if more features are warranted, and the speed of DSPs is increasing exponentially with time. Thus, time and integration are significant benefits of digital implementations.

This thesis utilized two DSP boards for various functions. One board was utilized at the wayside to drive the wayside transmitters with a communication signal and an autonomous position-sensing signal, as specified by the respective programs **comm8.asm**

and **stopped.asm** listed in Appendix A. A function generator was used to excite the vehicle transmitter.

The second board was utilized to decode the signals received on the vehicle. The sensor vehicle is connected to the DSP board through a stereo coaxial cable. The program **pos23.asm** listed in the Appendix decoded the communication signal and the relative or autonomous position-sensing signals, and displayed both on the LCD display along with the running time, as shown in Figure 6.1. Program **measure3.asm** displayed the position of the vehicle down to the micron. When a button on the board was pushed, a value was transmitted through the DSP serial port to be recorded automatically. Program strngth3.asm displayed and recorded the signal strengths of the two sensors.

6.1.1 Spacing of Sensors

For the sake of clarity, Figure 3.4 in Chapter 3 showed two sensors a quarter of a cycle apart. Implementing the sensors as shown in this figure is not possible due to the fact that the sensors would have to physically pass through each other. An equivalent implementation used in the demonstration system is shown in Figure 6.3. The second sensor in this figure has been shifted by an additional half a cycle. Thus, the sensor will pick up the negative of the signal it picked up previously. By winding the sensor in the opposite direction, the same signal is received as is calculated in Chapter 3. An image of the actual vehicle used is displayed in Figure 6.2.

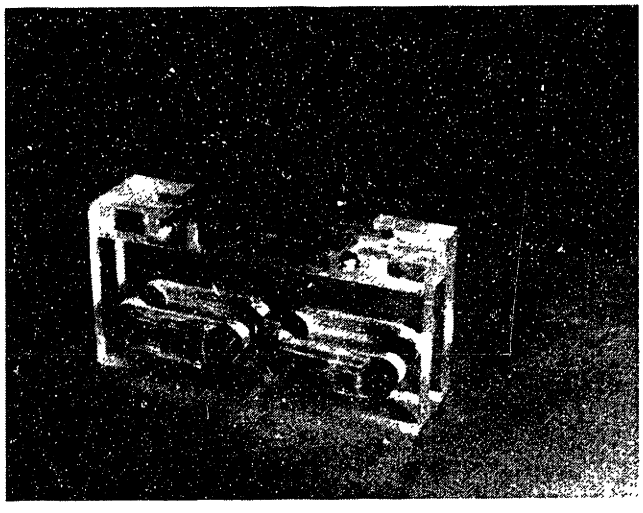

Figure 6.2 Image of sensor vehicle

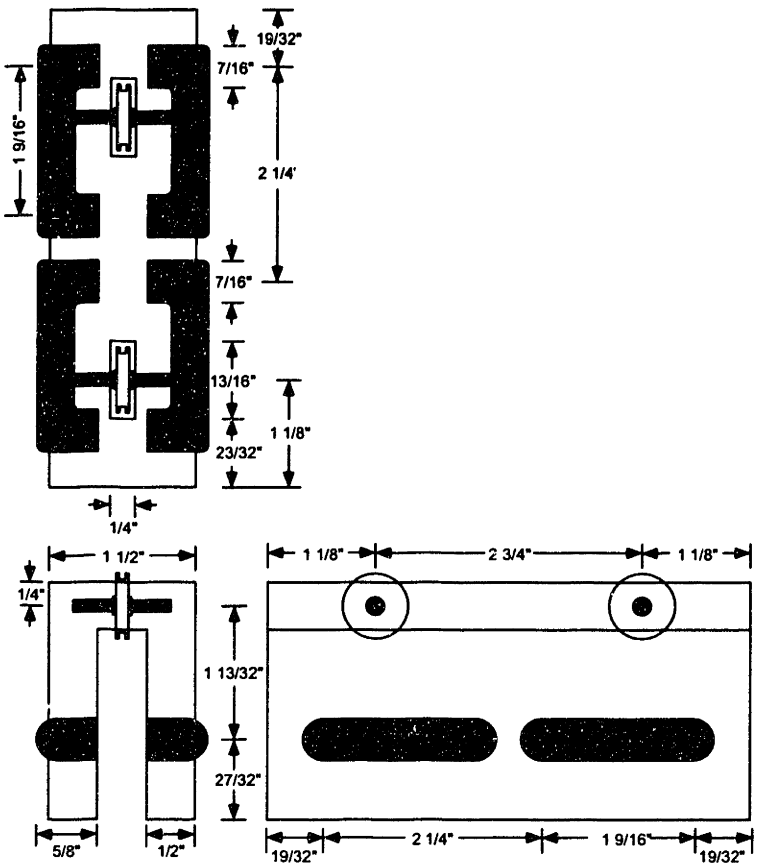

Figure 6.3 Illustration of sensor vehicle schematic

6.2 Multiple Vehicles Per Winding

Multiple vehicles may operate on a single winding, and multiple functions may be utilized simultaneously through the use of separate carrier frequencies for each vehicle or function (frequency division multiplexing). Each frequency may be independently filtered and used to implement any of the features described in Chapter 3. A separate frequency band may also be set aside for time-multiplexed communication.

One of the most efficient manners in which to use the features of the system would be to track only two vehicles, the one directly behind and the one directly in front. Thus, only two vehicle frequencies must be tracked at any one time, lending to a

computationally efficient implementation. A third frequency may be utilized by the system to implement autonomous position-sensing for all of the vehicles on a winding.

6.2.1 Acquiring Initial Position Estimates.

As previously mentioned, the position sensing features of the system are only able to detect the position within a cycle directly. In order to track actual position, the techniques require an initial estimate of vehicle positions. Many techniques can be utilized for this purpose, several of which will be described. These techniques may be utilized for both initial estimates, as well as a means of checking against errors.

One of the best means of acquiring position estimates is through the use of wayside markers. Markers which vehicles are able to detect may be placed at regular intervals along the wayside. These positions of these markers would update the vehicles on their current locations. A wide variety of marker types may be used including magnetic, inductive, optical, and ultrasonic. One possible option, to update other vehicles on the same winding, requires a vehicle to place a chirp signal in the winding when it passes such a marker. These chirps could be easily detected with a matched filter, and thus other vehicles would be notified when a vehicle passed the marker.

If a communication system is utilized in a transportation system, each vehicle could regularly transmit its own position to update other vehicles on the winding. This implementation is a very straightforward approach to solving the described problem.

Another possible approach is to utilize two duplicate windings, spaced side by side on guideway, as shown in Figure 6.4. Each vehicle could have duplicate sensors for each of the two windings. With such a setup, each vehicle may track when other vehicles enter or leave the two windings that it is situated over. Thus, relative positions may be updated at these occurrences. Thus, any vehicle within a half a winding length may be absolutely tracked. One of the main disadvantages of this system is the duplicated hardware.

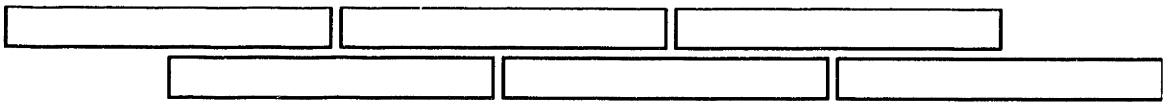

Figure 6.4 Staggered windings laid side by side

In order to avoid the complete duplication, another strategy is proposed. In the new scheme, two sets of overlapping windings are utilized, as illustrated from a side view in Figure 6.5. In this new scheme, three basic signal levels exist. For two vehicles in the same two blocks, the signal is a full strength. For two vehicles with one winding block in common, the signal is at half strength. When two vehicles do not have any blocks in common, no signal is received. By detecting when existing signals change levels, knowledge of the position of a vehicle, crossing a specific boundary, is updated. This system has all of the benefits of the first system, but requires only half of the hardware.

There are now a wide variety of means to acquiring position estimates, any of which may be used successfully to implement a transportation control system.

Figure 6.5 Side view of overlapping windings

6.3 Control Scenarios

Many different scenarios may be implemented using the concepts described in this proposal. To demonstrate the flexibility of the techniques developed in this thesis, two possible control strategies will be described. The two possible control strategies are asynchronous control and synchronous control and may be implemented using the functions previously described.

For the asynchronous scenario, vehicles sense their own position with the autonomous position sensing feature, and position relative to other vehicles. With this knowledge, each vehicle directs its own operation according to its destination and contents. A wayside system may then detect all vehicle positions and handle protection functions.

In the synchronous scenario, the wayside system also detects all vehicle positions, yet directs all vehicle movements utilizing the vehicle marker tracking feature described

in Chapter 3. The vehicles would then also sense their own position and relative positions and handle their own safety and protection functions. The techniques in this thesis are thus flexible enough to handle a wide variety of approaches to transportation control systems.

6.4 System Limitations

The techniques developed in this thesis for position-sensing and communication have a set of limitations that must be recognized when applied to a particular transportation system. These limitations include filter delay, scaling issues, limited propagation speed, and eddy current losses. Although these issues are addressed here, the solutions of these limitations are to be performed in future work.

One limitation of the position-sensing techniques is due to the delays created by the filters. The two filters used in the demodulation of the sensor signals have a total inherent delay of approximately two milliseconds. The position estimates obtained by the observer are thus approximately two milliseconds old. At a typical PRT speed of 15 m/s, this results in a constant error of approximately 3 cm. The time delay in the PRT case is not significant relative to a one second headway, but this delay may be more important to other applications. One manner in which to improve this performance is through the use of a predictor or predictive estimator [32] with the available position and velocity measurements.

Another, smaller limitation is the scaling of the inaccuracies of the system with winding size. The inaccuracies should scale linearly with size, unless the causes of the errors can be eliminated or reduced. One of the causes of inaccuracy in the relative position-sensing technique is the imbalance in the self-inductances of each of the phases. The main cause of this imbalance has been determined to be the termination of the winding [1]. Since the winding in a PRT would be significantly longer than the length of the winding in the demonstration system, the contribution of the termination to the overall inductance should reduced. As previously stated, the accuracy of the position-sensing techniques is more than an order of magnitude greater than is necessary for the purposes

of transportation control systems, so the scaling of inaccuracies is a relatively minor issue.

Of greater import to the scaling of the system is the scaling of parasitic components such as inter-phase capacitances. It is not known at this time what effect such parasitic components will have on the system as it is scaled up. One issue which is clear in the scaling limits of the system is the limit on winding length by the propagation speed of the signals in the system. The inductance of the phases of the winding and the inter-phase capacitances contribute to a propagation speed which is some fraction of the speed of light. This propagation speed limits the maximum allowable length of the winding. Assuming a speed of light upper limit, the wavelength of a 10 kHz carrier frequency is 30 km. Thus, the maximum length of a section of winding should be some small fraction of this length. Trade-offs may be made, however, between carrier frequency and maximum length. This limitation applies to communication signals as well, limiting the possible bandwidth of the winding medium.

When the position-sensing and communication techniques are implemented with the stator winding of a linear motor, an addition limitation exists if back iron is utilized. Eddy currents in the back iron rise, approximately, as the square of the frequency. Significant attenuation of the position-sensing and communication signals will occur if high frequencies are used. This effect limits the number of possible frequencies used on a particular winding and limits the possible communication bandwidth which may be achieved over the winding.

Chapter 7

Summary and Conclusions

7.1 Features and Benefits

Several features applicable to automated and semi-automated transportation systems have been developed in this thesis. Each of these features can benefit a wide variety of transportation systems, and in some cases applications outside of transportation. The implementation of these features may also be performed fairly inexpensively, leading to an economically attractive system.

The autonomous position-sensing feature, developed in Chapter 3, allows vehicles to track their own position. This feature can replace conventional odometers, with some added benefits. This feature does not suffer from error accumulation as odometers do, and is also not dependent on wheel traction for accurate readings. Thus, this system may be utilized for traction control purposes - a feature that has become increasingly important for locomotives. With traction control, vehicles are able to increase the maximum acceleration and deceleration of the vehicle safely, and help to prevent wheel wear due to slippage.

The techniques used to implement this feature have been demonstrated to have an accuracy down to the ten micron level. This accuracy is orders of magnitude better than the accuracy required for transportation. These techniques may thus be utilized for other types of applications where a higher degree of required accuracy exists. These applications include machine tools and optical wafer steppers, among others.

The relative position-sensing system developed in Chapter 3 is extremely useful for short headway systems, since it is important for the safety systems to have accurate, up to date information on the positions of nearby vehicles. This feature is also useful in longer headway systems to prevent vehicle collisions in the event that two vehicles get close enough together. This feature may also be used for push recovery systems, where a live vehicle is used to move a dead vehicle to a maintenance yard. The absolute accuracy of this feature in the demonstration system was better than a half of a millimeter, again at least an order of magnitude better than is required for transportation purposes. The limiting factor in the accuracy of this feature in the demonstration system was due to the imbalance of the self-inductances of the phases of the winding.

The virtual marker tracking feature, described in Chapter 3, is very useful in transportation systems which utilize a point-following concept. In this concept, a vehicle is given a moving reference point to track and follow at all times. The feature developed in this thesis can directly implement this concept in a control system. This feature is also extremely useful for the implementation of a transportation system which utilizes a doubly excited linear motor. The feature is used, in this case, to allow the vehicle to track the stator field in order to efficiently implement propulsion.

The communication methods in this thesis, described in Chapter 4, are especially useful when implemented in conjunction with the other position-sensing features, since they require no additional hardware. Such an implementation can replace the fairly expensive digital radio links which are typically used. The implementation developed for this thesis demonstrated a very robust communication system, even in the presence of movement and noise.

The techniques in this thesis may be easily adapted for use on the wayside to detect vehicle positions on the guideway. A single digital signal processor (DSP) may be utilized to track several vehicles on a winding, an improvement over the original system developed in [1]. In fact, all of the features described in this thesis may be implemented on a single DSP. This integration thus leads to cost advantages over analog implementations as well as higher reliability due to the lower parts count. The flexibility of the implementation in discrete time is a tremendous asset as well, as one generic set of

hardware may be utilized to implement a wide range of features. This flexibility enables the choice of which particular set of features to enable at any time, and allows upgrade of features without changing the physical hardware.

Finally, the approach taken in this thesis leads to a very inexpensive implementation as compared with other conventional approaches. The cost of the sensors and transmitters is low because commodity parts may be used. The performance of digital signal processor architectures is increasing exponentially as time progresses while the cost of existing processors is dropping. Thus, over time, the cost of implementing a system will drop while more features are enabled.

An approach to creating a low cost winding was also taken in this thesis. The production of the helical winding utilized for the demonstration system was performed in an automated fashion, and the ultimate cost of a mass produced winding is fairly close to the cost of the wire utilized. In order to reduce cost further, implementations on sparse windings have been investigated, including implementation on a single phase winding. Thus, the accomplishments of this thesis have enabled a series of very useful features on an inexpensive set of hardware.

7.2 Applications

There are many possible applications for which these concepts may be well utilized. They include Person Rapid Transit (PRT), Elevators in Skyscrapers, Baggage Handling Systems, Intelligent Transportation Systems (ITS-IVHS), Roller Coasters, and Maglev applications. Such a system may even be used, without the linear motor propulsion, to retrofit existing transportation systems and improve performance.

Perhaps the most closely matched application for the techniques developed in this thesis is that of PRT. In one possible PRT scenario, the vehicles are driven by a doubly excited linear motor. Each long stator section is driven at a particular frequency, with a particular equivalent speed. Each vehicle controls its own rotor field, and controls its own speed by varying this field. One possible manner in which each vehicle may sense the position of the stator field is to create a virtual marker at the wayside which moves in synchronism with the stator field. Vehicles are thus able to control their speed, detect

other vehicles on the winding, determine their own absolute position, and communicate with the wayside. By utilizing two windings overlapping longitudinally, vehicles are able to track any vehicles close enough to affect their own operation.

Some of the features in this thesis are also applicable to Maglev applications. A Maglev vehicle is typically driven by a linear motor, so the winding necessary to implement the features of this thesis already exists. The relative position-sensing feature is not very useful for high speed Maglev applications since the headways between the vehicles are on the order of kilometers. However, the autonomous position-sensing, wayside position-sensing, and communication would be very useful features for implementing a Maglev system.

One possible application of these methods to ITS may be implemented by encasing two large, sparse windings in the road bed below the top layer of asphalt. These two windings would be overlapped longitudinally so that two closely spaced vehicles may always sense each other's signals. Vehicles could be assigned a carrier frequency as they enter on onramp to the ITS highway, and changed, if necessary, through communication for a lane change or other reason. With such a system in place, vehicles could sense the position of preceding and following vehicles, and take any actions necessary in case of an emergency.

These windings may be utilized for lateral guidance as well as longitudinal with additional wide sensors in which the guideway. Signals in these sensors would drop off approximately linearly as a vehicle moves out of its lane, and thus may be utilized for lateral guidance feedback.

Several scenarios for different levels of implementation may be developed. At the wayside, in one scenario, vehicles would be tracked, and commands given over the winding for vehicles to follow in order to optimize the system performance. Vehicles could request actions such as lane changes or speed profiles to be confirmed or denied by an area controller. In another scenario, each vehicle could be given a virtual marker to track and follow. The area controller would be in complete charge of the system, and would direct all vehicles from onramp to offramp. In the third and most conservative scenario, vehicles would be driven by their owners, and the system used for warnings and

emergencies only. The system could also give suggested routes to minimize travel time to the driver.

The set of techniques developed in this thesis may also be utilized without the integration of a linear motor, much as in the ITS application. With a winding laid upon an existing guideway, older systems may be retrofitted with a better vehicle tracking system. Tighter headways may then be achieved on existing lines by migrating from a fixed block control system to a moving block system, increasing the utilization of existing infrastructure. Traction control systems may also be implemented by means of the autonomous position-sensing technique. Additionally, by knowing the exact position of vehicles, adjustments in speed profiles may be made to keep the vehicles on a tighter schedule. Again, by utilizing overlapping windings, vehicle collisions may be prevented, as each vehicle will be able to detect and track nearby vehicles.

The core set of features described in this thesis are very useful for implementing a wide variety of transportation control systems. They provide both the basis for a new automated system and the possible improvement in the performance and infrastructure utilization in existing systems.

7.3 Conclusions

Two basic advancements have been developed in the course of this thesis. It has been demonstrated that it is possible to implement three accurate position-sensing features on an inexpensive, winding-based architecture. It has been shown that communication over the same signal path in the winding is also possible without the need for any additional hardware. These techniques have been successfully implemented on a demonstration system simultaneously. Finally, it is possible to implement these position-sensing and communication techniques on a simplified, inexpensive winding.

The techniques implemented in this thesis have been shown to be equivalent of superior to existing alternatives in terms of cost, features, and accuracy for the application of Personal Rapid Transit. The integration of the features, in the approach taken in this thesis, is possible on a single discrete-time system. The accomplishments of this thesis have enabled a core set of very useful features on an inexpensive set of hardware, and

have advanced the state-of-the-art in transportation control systems in terms of accuracy, integration, and cost.

7.4 Recommendations for Future Work

Several of the limitations discussed in Chapter 6 should be addressed in future work. Most importantly, the scaling effects on the parasitic components in the winding should be studied to understand their contribution to the scaling limits. A more detailed analysis on the limited propagation speed should also be performed to appreciate its effect on the maximum winding length. Finally, it would be useful to create a predictor to compensate for the delay of the filters used in the position-sensing techniques. While the delays do not significantly affect the application of transportation control systems, they may be important in other applications.

Appendix A

A.1 stopped.asm

```
;*****************************************************************************
;stopped.asm - will place a signal in the winding similar to a stopped vehicle
;              to be utilized with the autonomous position-sensing feature
;*****************************************************************************
      nolist
      include 'ioequ.asm'
      include 'intequ.asm'
      include 'ada_equ.asm'
      include 'vectors3.asm'
      list

;*****************************************************************************

;---Buffer for talking to the CS4215

       org    x:0
RX_BUFF_BASE  equ     *
RX_data_1_2   ds      1       ;data time slot 1/2 for RX ISR
RX_data_3_4   ds      1       ;data time slot 3/4 for RX ISR
RX_data_5_6   ds      1       ;data time slot 5/6 for RX ISR
RX_data_7_8   ds      1       ;data time slot 7/8 for RX ISR

TX_BUFF_BASE  equ     *
TX_data_1_2   ds      1       ;data time slot 1/2 for TX ISR
TX_data_3_4   ds      1       ;data time slot 3/4 for TX ISR
TX_data_5_6   ds      1       ;data time slot 5/6 for TX ISR
TX_data_7_8   ds      1       ;data time slot 7/8 for TX ISR

RX_PTR          ds      1  ; Pointer for rx buffer
TX_PTR          ds      1  ; Pointer for tx buffer

sixcnt  dc  6
sinarg  dc  sincoef
      org    x:$020
      include 'asm\sintble2.asm'
;sincoef dc  0.5,-0.5,-0.9999999,-0.5,0.5,0.9999999 ;last one is supposed to be
1

TONE_OUTPUT     EQU     HEADPHONE_EN+LINEOUT_EN
TONE_INPUT      EQU     MIC_IN_SELECT+(15*MONITOR_ATTN)
CTRL_WD_12    equu    NO_PREAMP+HI_PASS_FILT+SAMP_RATE_48+STEREO+DATA_16    ;CLB=0
CTRL_WD_34    equu    IMMED_3STATE+XTAL1_SELECT+BITS_64+CODEC_MASTER
CTRL_WD_56    equu    $000000
CTRL_WD_78    equu    $000000

HPCR            equ     $FFFFC4
HDDR            equ     $FFFFC8
HDR             equ     $FFFFC9

       org     y:$100
;      include       'asm\sintable.asm' ; location of sine table in memory
                              ; also location of cos table with
                              ; shift and wraparound


       org     p:$100
```

```
START
main
        movep   #$040003,x:M_PCTL     ; set PLL for MPY of 4X
        movep   #$012421,x:M_BCR    ; set up one ext. wait state for all AAR
areas
        ori     #3,mr             ;mask interrupts
        movec   #0,sp             ;clear hardware stack pointer
        move    #0,omr            ;operating mode 0
        move    #$40,r6           ; initialise stack pointer
        move    #-1,m6            ; linear addressing
       include 'dispini2.asm'
        jsr     ada_init          ; initialize codec
        movep   #$A40033,x:M_PCTL     ; set PLL for MPY of 52X and DIV by 11
                                      ; for actual speed of 80.053527 MHz
        move    #$80FF,m6         ; force it to stay in 256 entry table


loop_1

        jset    #2,x:M_SSISR0,*     ;wait for frame sync to pass
        jclr    #2,x:M_SSISR0,*     ;wait for frame sync

        move    x:RX_BUFF_BASE,a          ;receive left
        move    x:RX_BUFF_BASE+1,b        ;receive right
        jsr     process_stereo
        move    a,x:TX_BUFF_BASE          ;transmit left
        move    b,x:TX_BUFF_BASE+1        ;transmit right

        move    #TONE_OUTPUT,y0 ;set up control words
        move    y0,x:TX_BUFF_BASE+2
        move    #TONE_INPUT,y0
        move    y0,x:TX_BUFF_BASE+3

        jmp     loop_1

process_stereo
       move   x:sinarg,r2
       move   #23,m2
       nop
         move x:(r2)+,y1               ;look up cos value
       move   #$FFFFFF,m2
       move   r2,x:sinarg           ; update pointer into sin table
       move   y1,a
       move   a,b
         nop
         nop
         nop
         rts

        include 'ada_init.asm'
echo
        end
```

A.2 adainit.asm

```
;*************************************************************************
; adainit.asm - initialization routine for the CS4215
;
;*************************************************************************


;*************************************************************************
;
;    portc usage:
;  bit8: SSI TX (from DSP to Codec)
;  bit7:
;  bit6:
;  bit5:
;  bit4: codec reset (from DSP to Codec)
;  bit3:
;      bit2: data/control bar
;          0=control
;          1=data
;
;*************************************************************************
;*****        initialize the CS4215 codec                *****
;*************************************************************************
;
;
;  PROGRAM OUTLINE:
;
;1 program fsync and sclk == output
;2 write pc0 = 0 (control mode)
;3 send 64 bit frame x times, with dcb bit = 0, keep doing until read back as 0
;4 send 64 bit frame x times, with dcb bit = 1, keep doing until read back as 1
;5 re-program fsync and sclk == input
;6 write pc0 = 1 (data mode)
;7 receive/send data (echo slots 1,2,3,4; slots 5,6,7,8 == constants)
;
;*************************************************************************
;
;    initialize ssi -- fsync and sclk ==> outputs
;
      org    p:
ada_init
      movep  #$0000,x:M_PCRC        ; turn off ESSI0 port (for now)
      movep  #$103807,x:M_CRA0      ; 40MHz/16 = 2.5MHz SCLK, WL=16 bits, 4W/F
      movep  #$ff313C,x:M_CRB0      ; RIE,TIE,RLIE,TLIE,RE,TE,sc2/sck outputs
      movep  #$0003,x:M_PRRC        ; setup pd0 and pd1 as gpio output
      movep  #$0,x:M_PDRC           ; send out a 0 on DC~ and RST_CODEC~

        ;----reset delay for codec ----
      do     #1000,_delay_loop
      rep    #2000                  ; 100 us delay (assuming 40MHz VCO)
      nop
_delay_loop

      bset   #0,x:M_PDRC            ; sends out a 1 on pd0 (rst_codec=1)
      movep  #$000C,x:M_IPRP        ; set interrupt priority level for ESSI0 to 3
      andi   #$FC,mr                ; enable interrupts

;*************************************************************************
; The following data sets up the CS4215 control mode data:
```

```
;   (CTS = Control Time Slot, U/LN = upper/lower Nibble)
;
;     +------ CTS1-UN:  0    0    1    MLB     0 0 0 0
;     |+----- CTS1-LN: OLB   CLB   X    X     0 0 0 0
;     ||+---- CTS2-UN: HPF   X    DFR2 DFR1    0 0 1 0
;     |||+--- CTS2-LN: DFR0  ST   DF1  DF0     1 1 0 0
; x0 = $002Cxx
;
;     +------ CTS3-UN: ITS   MCK2  MCK1  MCK0    1 0 0 0
;     |+----- CTS3-LN: BSEL1 BSEL0 XCLK  XEN     1 0 0 0
;     ||+---- CTS4-UN: TEST  TEST  TEST  TEST    (TEST MUST BE 0)
;     |||+--- CTS4-LN: TEST  TEST  ENL   DAD     0 0 0 0
; x0 = $8800xx
;*************************************************************************

;--- set up buffer with control mode data
          move  #CTRL_WD_12,x0
          move        x0,x:TX_BUFF_BASE
          move  #CTRL_WD_34,x0
          move        x0,x:TX_BUFF_BASE+1
          move  #CTRL_WD_56,x0
          move          x0,x:TX_BUFF_BASE+2
     move  #CTRL_WD_78,x0
     move        x0,x:TX_BUFF_BASE+3

     movep  #$003C,x:M_PCRC       ;turn on ESSI0 except for sc0 and sc2

;
; CLB == 0
;
     jclr   #3,x:M_SSISR0,*       ; wait until rx frame bit==1
     jset   #3,x:M_SSISR0,*       ; wait until rx frame bit==0
     jclr   #3,x:M_SSISR0,*       ; wait until rx frame bit==1
     jset   #18,x:RX_BUFF_BASE,*  ; loop until CLB set

;
; CLB == 1
;
     bset   #18,x:TX_BUFF_BASE    ;set CLB
     do     #4,_init_loopB
     jclr   #2,x:M_SSISR0,*       ; wait until tx frame bit==1
     jset   #2,x:M_SSISR0,*       ; wait until tx frame bit==0
_init_loopB
     movep  #$0000,x:M_PCRC       ; disable ESSI0

;*************************************************************************
;  now CLB should be 1 -- re-program fsync and sclk direction (i/p) -- also,
; circular buffer pointers for echoing data r0=current, r1=old data to send
; 1 frame later
;
     movep  #$103807,x:M_CRA0     ; 40MHz/16 = 2.5MHz SCLK, WL=16 bits, 4W/F
     movep  #$FF310C,x:M_CRB0      ; sckd and fsync (sc02) as inputs
     movep  #$0003,x:M_PDRC       ; D/C~ pin = 1 ==> data mode
     movep  #$003C,x:M_PCRC       ; turn on ESSI0 except for sc0 and sc2
     rts

;*************************************************************************
;          SSI0_ISR.ASM   Ver.2.0
;          Example program to handle interrupts through
;     the 56303 SSI0 to move audio through the CS4215
;
```

```
;          Copyright (c) MOTOROLA 1995, 1996
;                         Semiconductor Products Sector
;                         Digital Signal Processing Division
;
;     upon entry:
;          R6 must be the stack pointer
;     corrupts:
;          R6
;
;     History:
;          14 June 1996: RLR/LJD - ver 1.0
;*******************************************************************************


;----the actual interrupt service routines (ISRs)  follow:

;************************ SSI TRANSMIT ISR *********************************
ssi_txe_isr
     bclr   #4,x:M_SSISR0        ; Read SSISR to clear exception flag
                                 ; explicitly clears underrun flag
ssi_tx_isr
     move   r0,x:(r6)+           ; Save r0 to the stack.
     move   m0,x:(r6)+           ; Save m0 to the stack.
     move   #3,m0                ; Modulus 4 buffer.
     move   x:TX_PTR,r0          ; Load the pointer to the tx buffer.
     nop
     movep  x:(r0)+,x:M_TX00     ; SSI transfer data register.
     move   r0,x:TX_PTR          ; Update tx buffer pointer.
     move   x:-(r6),m0           ; Restore m0.
     move   x:-(r6),r0           ; Restore r0.
     rti

;********************** SSI TRANSMIT LAST SLOT ISR **************************
ssi_txls_isr
     move   r0,x:(r6)+           ; Save r0 to the stack.
     move   #TX_BUFF_BASE,r0     ; Reset pointer.
     move   r0,x:TX_PTR          ; Reset tx buffer pointer just in
                                 ; case it was corrupted.
     move   x:-(r6),r0           ; Restore r0.
     rti

;************************** SSI receive ISR *********************************
ssi_rxe_isr
     bclr   #5,x:M_SSISR0        ; Read SSISR to clear exception flag
                                 ; explicitly clears overrun flag
ssi_rx_isr
     move   r0,x:(r6)+           ; Save r0 to the stack.
     move   m0,x:(r6)+           ; Save m0 to the stack.
     move   #3,m0                ; Modulo 4 buffer.
     move   x:RX_PTR,r0          ; Load the pointer to the rx buffer.
     nop
     movep  x:M_RX0,x:(r0)+      ; Read out received data to buffer.
     move   r0,x:RX_PTR          ; Update rx buffer pointer.
     move   x:-(r6),m0           ; Restore m0.
     move   x:-(r6),r0           ; Restore r0.
     rti

;********************** SSI receive last slot ISR ***************************
ssi_rxls_isr
     move   r0,x:(r6)+           ; Save r0 to the stack.
     move   #RX_BUFF_BASE,r0     ; Reset rx buffer pointer just in
```

```
                    ; case it was corrupted.
move    r0,x:RX_PTR         ; Update rx buffer pointer.
move        x:-(r6),r0          ; Restore r0.
rti
```

A.3 comm8.asm

```
;*****************************************************************************
;comm8.asm - sends a communication signal into the winding using FSK modulation
;          - the bit rate is 1500 bits per second. This program sends ascii
;          - characters from a string 3 times per second at a channel
;          - utilization rate of 2%
;*****************************************************************************
      nolist
      include 'ioequ.asm'
      include 'intequ.asm'
      include 'ada_equ.asm'
      include 'vectors3.asm'
      list

;*******************************************************************************

;---Buffer for talking to the CS4215

      org    x:0
RX_BUFF_BASE equ      *
RX_data_1_2  ds     1      ;data time slot 1/2 for RX ISR
RX_data_3_4  ds     1      ;data time slot 3/4 for RX ISR
RX_data_5_6  ds     1      ;data time slot 5/6 for RX ISR
RX_data_7_8  ds     1      ;data time slot 7/8 for RX ISR

TX_BUFF_BASE equ    *
TX_data_1_2  ds     1      ;data time slot 1/2 for TX ISR
TX_data_3_4  ds     1      ;data time slot 3/4 for TX ISR
TX_data_5_6  ds     1      ;data time slot 5/6 for TX ISR
TX_data_7_8  ds     1      ;data time slot 7/8 for TX ISR

RX_PTR          ds      1 ; Pointer for rx buffer
TX_PTR          ds      1 ; Pointer for tx buffer

sixcnt  dc  6
sinarg  dc  logic1
bitlevel dc 1
savex  dc  0
savey  dc  0
bitnum dc  andmasks
charnum       dc  charbuff
char   dc  $1FF
sendnum       dc  sendbuff
      org    x:$040
sincoef
       include 'asm\commsin6.asm'
;sincoef dc  0.5,-0.5,-0.9999999,-0.5,0.5,0.9999999 ;last one is supposed to be
1

      org    x:$0200
andmasks dc   $0100,$080,$040,$020,$010,$008,$004,$002,$001
      org    x:$0300
charbuff dc   $0AA,$1FF,$1FF,$1FF,$1FF,$1FF,$1FF,$1FF
       dc     $1FF,$1FF,$1FF,$1FF,$1FF,$1FF,$1FF,$1FF
       dc     $1FF,$1FF,$1FF,$1FF,$1FF,$1FF,$1FF,$1FF
       dc     $1FF,$1FF,$1FF,$1FF,$1FF,$1FF,$1FF,$1FF
       dc     $1FF,$1FF,$1FF,$1FF,$1FF,$1FF,$1FF,$1FF
       dc     $1FF,$1FF,$1FF,$1FF,$1FF,$1FF,$1FF,$1FF
       dc     $1FF,$1FF,$1FF,$1FF,$1FF,$1FF,$1FF,$1FF
       dc     $1FF,$1FF,$1FF,$1FF,$1FF,$1FF,$1FF,$1FF
      org    x:$0400
sendbuff
       ;dc    $0A0,$0A1,$0A2,$0A3,$0A4,$0A5,$0A6,$0A7
       ;dc    $0A8,$0A9,$0AA,$0AB,$0AC,$0AD,$0AE,$0AF
       dc     ' W e l c o m e   t o   t h i s '
       dc     ' R e l a t i v e   P o s i t i o n '
       dc     ' S e n s i n g   D e m o n s t r a t i o n
!      '
       dc      ' T h i s   M e s s a g e   i s '
```

```
        dc    '  b  e  i  n  g     t  r  a  n  s  m  i  t  t  e  d   '
        dc    '  a  t     1  5  0  0     b  i  t  s     p  e  r   '
        dc    '  s  e  c  o  n  d     a  t     2  %   '
        dc    '  u  t  i  l  i  z  a  t  i  o  n     (  o  n  l  y   '
        dc    '  e  v  e  r  y     5  0  t  h     s  l  o  t     i  s   '
        dc    '  f  i  l  l  e  d  )  .   '
        dc    '                                                        '
TONE_OUTPUT     EQU     HEADPHONE_EN+LINEOUT_EN
TONE_INPUT      EQU     MIC_IN_SELECT+(15*MONITOR_ATTN)
CTRL_WD_12    equ    NO_PREAMP+HI_PASS_FILT+SAMP_RATE_48+STEREO+DATA_16    ;CLB=0
CTRL_WD_34    equ    IMMED_3STATE+XTAL1_SELECT+BITS_64+CODEC_MASTER
CTRL_WD_56    equ    $000000
CTRL_WD_78    equ    $000000

HPCR            equ     $FFFFC4
HDDR            equ     $FFFFC8
HDR             equ     $FFFFC9

        org     y:$100
;      include       'asm\sintable.asm' ; location of sine table in memory
                              ; also location of cos table with
                              ; shift and wraparound


        org     p:$100
START
main
        movep   #$040003,x:M_PCTL     ; set PLL for MPY of 4X
        movep   #$012421,x:M_BCR   ; set up one ext. wait state for all AAR
areas
        ori     #3,mr             ;mask interrupts
        movec   #0,sp             ;clear hardware stack pointer
        move    #0,omr            ;operating mode 0
        move    #$80,r6           ; initialise stack pointer
        move    #-1,m6            ; linear addressing
       include 'dispini2.asm'
        jsr     ada_init          ; initialize codec
        movep   #$A40033,x:M_PCTL     ; set PLL for MPY of 52X and DIV by 11
                                      ; for actual speed of 80.053527 MHz
        move    #$80FF,m6         ; force it to stay in 256 entry table


loop_1

        jset    #2,x:M_SSISR0,*     ;wait for frame sync to pass
        jclr    #2,x:M_SSISR0,*     ;wait for frame sync

        move    x:RX_BUFF_BASE,a          ;receive left
        move    x:RX_BUFF_BASE+1,b        ;receive right
        jsr     process_stereo
       move   x:savex,a
       move   x:savey,b
        move    a,x:TX_BUFF_BASE          ;transmit left
        move    b,x:TX_BUFF_BASE+1        ;transmit right

        move    #TONE_OUTPUT,y0 ;set up control words
        move    y0,x:TX_BUFF_BASE+2
        move    #TONE_INPUT,y0
        move    y0,x:TX_BUFF_BASE+3

        jmp     loop_1

process_stereo

       move   x:bitlevel,b
       cmp    #0,b
       jeq    sendzero
       jmp    sendone

sendzero
       move   x:sinarg,r2
        move    #31,m2
```

```
        nop
          move  x:(r2)+,y1              ;look up cos value
        move    #$FFFFFF,m2
        move    y1,x:savey            ; write the same thing to both
        move    r2,x:sinarg           ; update pointer into sin table
        move    r2,b
        move    #>logic0,x0
        cmp     x0,b
        jeq     nextbit
        jmp     donerts

sendone
        move    x:sinarg,r2
          move     #31,m2
        nop
          move  x:(r2)+,y1              ;look up cos value
        move    #$FFFFFF,m2
        move    y1,x:savey            ; write the same thing to both
        move    r2,x:sinarg           ; update pointer into sin table
        move    r2,b
        move    #>logic1,x0
        cmp     x0,b
        jeq     nextbit
        jmp     donerts

nextbit
        move    x:char,a      ; get character to send
        move    x:bitnum,r1     ; get mask for proper bit
        move    #8,m1         ; assume we send 9 bits
        move    x:(r1)+,x0      ; get mask for particular bit
        move    r1,x:bitnum     ; set position of new mask
        and     x0,a          ; mask out all but proper bit
        jeq     setadd0
setadd1
        move    #1,n1
        move    n1,x:bitlevel
        move    #>logic1,x0
        move    x0,x:sinarg
        jmp donenext
setadd0
        move    #0,n1
        move    n1,x:bitlevel
        move    #>logic0,x0
        move    x0,x:sinarg
donenext
        move    r1,b
        move    #andmasks,x0
        cmp     x0,b
        jeq     newchar
        jmp donerts

newchar
        move    x:charnum,r4
        move    #50,m4        ; assume up to 50 characters in buffer
        move    x:(r4)+,x0      ; load new character into x0
        move    x0,x:char     ; put it in the character buffer
        move    r4,x:charnum  ; update the pointer to the buffer
        move    #>charbuff,x1
        move    r4,b
        cmp     x1,b
        jeq     newsend
        jmp     donerts

newsend                             ; put in a new char to send since we have
                              ; gone throught the entire buffer.
        move    x:sendnum,r4
        move    #>180,m4            ; assume 180 chars in buffer
        move    x:(r4)+,x0      ; load the new character into x0
        move    x0,x:charbuff   ; store the new char in the output buffer
        move    r4,x:sendnum  ; update the pointer

donerts
```

```
        nop
          nop
          nop
          rts

          include 'ada_init.asm'
echo
          end
```

A.4 pos23.asm

```
;*****************************************************************************
;POSITION.ASM - Reads in 2 quadrature signals modulated to 10 kHz
;               Demodulates the signals with help from a PLL
;               Uses 2 input PLL (non-linear observer) to track position
;                 difference between vehicles
;               May be used with either autonomous or relative position-
;                 sensing schemes
;               Converts binary values to decimal positions and displays on
;                 LCD screen
;             - Takes in sensor signals and filters for communication signal
;               Demodulates and decodes communication signal
;               Displays received message on LCD screen
;             - Tracks running time and displays on LCD screen
;*****************************************************************************
; Note: R5 is designated as the stack pointer for routines.
;*****************************************************************************
      nolist
      include 'ioequ.asm'
      include 'intequ.asm'
      include 'ada_equ.asm'
      include 'vectors4.asm'
      list
       opt cc
;*****************************************************************************
; 4/6/96 added in 2 input PLL for position detection
; 4/6/96 added in display of position
;---Buffer for talking to the CS4215

       org    x:0
RX_BUFF_BASE  equ     *
RX_data_1_2   ds      1       ;data time slot 1/2 for RX ISR
RX_data_3_4   ds      1       ;data time slot 3/4 for RX ISR
RX_data_5_6   ds      1       ;data time slot 5/6 for RX ISR
RX_data_7_8   ds      1       ;data time slot 7/8 for RX ISR

TX_BUFF_BASE  equ     *
TX_data_1_2   ds      1       ;data time slot 1/2 for TX ISR
TX_data_3_4   ds      1       ;data time slot 3/4 for TX ISR
TX_data_5_6   ds      1       ;data time slot 5/6 for TX ISR
TX_data_7_8   ds      1       ;data time slot 7/8 for TX ISR

RX_PTR          ds      1  ; Pointer for rx buffer
TX_PTR          ds      1  ; Pointer for tx buffer
tempr5        ds      1
dispptr               dc      DISPLIST
strength      dc      0
      org     x:$010
sigxy   bsc 10,$0          ; circular buffer for butterworth LPF
                           ;filter zero coefficients for butterworth LPF
                           ;organized as: b0 b1 b2 b3 b4 -a1 -a2 -a3 -a4
s3o     dc  0              ; states for high freq PLL
s1o     dc  0
s2o     dc  0
sixcnt  dc  6
sinarg  dc  sincoef
      org     x:$020
sincoef dc  0.5,-0.5,-0.9999999,-0.5,0.5,0.9999999 ;last one is supposed to be 1
st3o    dc  0              ; states for low freq PLL
st1o    dc  0
st2o    dc  0
sig1strt dc input1 ;start circular buffer at 0
sig2strt dc lpsig1 ;start circular buffer at lpsig1
sig3strt dc lpsig3 ;start circular buffer at 0
sig4strt dc lpsig5 ;start circular buffer at lpsig1
sig5strt dc lpsig7
sig6strt dc lpsig9
sig7strt dc lpsig11
```

```
count   dc  6
bpfiltx      dc  0
bpfilty dc  0
savex  dc  $400000
savey   dc  0
newx     dc  0
newy    dc  0
comfiltx dc 0
comfilty dc 0
logic1  dc  0
logic0  dc  0
level   dc  0
clkcnt  dc  8000

clock
seconds          dc      0
tenseconds       dc      0
minutes          dc      0
tenminutes       dc      0
hours            dc      0
tenhours         dc      0
nothing          dc      0
       org    x:$0050
compclk          dc      10,6,10,6,10,10,0
       org    x:$0060
digits0          dc
lett00,lett10,lett20,lett30,lett40,lett50,lett60,lett70,lett80,lett90
       org    x:$0070
digits1          dc
lett01,lett11,lett21,lett31,lett41,lett51,lett61,lett71,lett81,lett91

        org     x:$0080
input1  bsc 128,$0      ; circular buffer for input 1 filter
input2  bsc 128,$0      ; circular buffer for input 2 filter

lpsig1  bsc 16,$0       ; circular buffer for demodulated input 1
lpsig2  bsc 16,$0       ; circular buffer for demodulated input 2

         org      x:$0200
lpsig3  bsc 128,$0      ; circular buffer for input 1 filter
lpsig4  bsc 128,$0      ; circular buffer for input 2 filter

lpsig5  bsc 32,$0       ; circular buffer for matched 1.5 Khz filter for input 1
lpsig6  bsc 32,$0       ; circular buffer for matched 1.5 Khz filter for input 2
lpsig7  bsc 32,$0   ; circular buffer for matched 4.5 Khz filter for input 1
lpsig8  bsc 32,$0   ; circular buffer for matched 4.5 Khz filter for input 2


lpsig9   bsc 16,$0  ; circular buffer for averaging filter for 1.5 khz input 1
lpsig10  bsc 16,$0  ; circular buffer for averaging filter for 1.5 khz input 2
lpsig11  bsc 16,$0  ; circular buffer for averaging filter for 4.5 khz input 1
lpsig12  bsc 16,$0  ; circular buffer for averaging filter for 4.5 khz input 2


       org    y:0

DISPLEN          equ 63 ; should be ((*-DISPLIST)-1) but isn't

        include 'asm\butter.asm'    ; include butterworth filter coeff's
                                    ;filter zero coefficients for butterworth LPF
                                    ;organized as: b0 b1 b2 b3 b4 -a1 -a2 -a3 -a4
c1      dc 0.0021418412               ; coefficients for fast PLL
c2      dc 0.5235987756
w2max   dc 0.0490873852
w2min   dc -0.0490873852
deltap  dc 0.52359877560
c3      dc (1/(4*3.14159265359))
s2max   dc 0.5
s2min   dc -0.5
scalesin dc 512

;Coefficients for two input PLL (Non-linear Observer)
```

```
k       dc  .0007815           ; coefficients for fast PLL
c       dc  .2499
deltap2 dc  0
c4      dc (1/(4*3.14159265359))
st2max   dc 0.25
st2min   dc -0.25
;st2max   dc 0.5
;st2min   dc -0.5
bigpos  dc 0
scalesin2 dc 512
;scalesin2 dc 10
                ; THESE numbers are for the display routine
       org    y:$0040
decode  dc      0.0010001,500,0.0100002,50,0.1000002,5 ; These constants will be
used to
                    ; convert binary to decimal. It is VERY important to note
that every
                    ; other one of these number is a small delta above 10^(-k).
This
                    ; prevents errors at borders of 1000's, 100's and 10's. (if
it were a
                    ; little less, we would be low by 1 digit and have problems.
digits  dc      $88,$EB,$4C,$49,$2B,$19,$18,$CB,$08,$09

        org     y:$0080 ; next thing must start on a 128 byte boundary;

        org     y:$100
       include      'asm\sintable.asm' ; location of sine table in memory
                            ; also location of cos table with
                            ; shift and wraparound

        org     y:$0200         ;definitely in internal memory
        include 'asm\bandpass.asm' ; include bandpass filter coeff's
        include 'asm\lowpass.asm'  ; include lowpass filter coefficients
         org     y:$0300
        include 'asm\commflt2.asm' ; include bandpass filter coeff's
        include 'asm\sinfilt1.asm'  ; include lowpass filter coefficients


      org    y:$0400
buffer1 bsc     $0100,$0
       org    y:$0500
DISPLIST
                dc      lcdcommand,lcdpos800,lcdpos801,lcddata
                dc      lettt0,lettt1,letti0,letti1,lettm0,lettm1,lette0,lette1
                dc
lettcol0,lettcol1,lett0,lett1,lett00,lett01,lett00,lett01
                dc
lettcol0,lettcol1,lett00,lett01,lett00,lett01,lettcol0,lettcol1
                dc      lett00,lett01,lett00
endtime         dc      lett01
             dc     lcdcommand,lcdpos940,lcdpos941,lcddata
                dc      lettp0,lettp1,letto0,letto1,letts0,letts1
                dc      letti0,letti1,lettt0,lettt1,letti0,letti1
                dc      letto0,letto1,lettn0,lettn1,lettcol0,lettcol1
strtpos         dc      lett0,lett1
             dc     lett00,lett01,lett00,lett01,lett00,lett01
endpos       dc     lett00,lett01,lett0,lett1,(lettm0+32),(lettm1+32)
             dc     (lettm0+32),(lettm1+32)
             dc     lcdcommand,lcdposc00,lcdposc01,lcddata
strtchar     dc     lettr0,lettr1
nextchar     dc     lette0,lette1,lettl0,lettl1
             dc     letta0,letta1,lettt0,lettt1,letti0,letti1
             dc     lettv0,lettv1,lette0,lette1,lett0,lett1
             dc     lettp0,lettp1,letto0,letto1,letts0,letts1
             dc     letti0,letti1,lettt0,lettt1,letti0,letti1
             dc     letto0,letto1,lettn0,lettn1,lett0,lett1
             dc     letts0,letts1,lette0,lette1,lettn0,lettn1
             dc     letts0,letts1,letti0,letti1,lettn0,lettn1
             dc     lettg0,lettg1,lett0,lett1,lettd0,lettd1,lette0,lette1
             dc     lettm0,lettm1,letto0,letto1,lettex0,lettex1
charpos      dc     lett0,lett1
```

```
            bsc 128,lcdcommand

TONE_OUTPUT     EQU     HEADPHONE_EN+LINEOUT_EN
TONE_INPUT      EQU
MIC_IN_SELECT+(15*MONITOR_ATTN)+(15*LEFT_GAIN)+(15*RIGHT_GAIN)
CTRL_WD_12    equ    NO_PREAMP+HI_PASS_FILT+SAMP_RATE_48+STEREO+DATA_16    ;CLB=0
CTRL_WD_34    equ    IMMED_3STATE+XTAL1_SELECT+BITS_64+CODEC_MASTER
CTRL_WD_56    equ    $000000
CTRL_WD_78    equ    $000000

;HPCR             equ     $FFFFC4
;HDDR             equ     $FFFFC8
;HDR              equ     $FFFFC9

        include 'dispequ.asm'


        org     p:$100
START
main
         movep   #$A40033,x:M_PCTL     ; set PLL for MPY of 52X and DIV by 11
                                       ; for actual speed of 80.053527 MHz
        movep   #$012421,x:M_BCR   ; set up one ext. wait state for all AAR
areas
         movep   #6,x:M_PRRE
        ori     #3,mr             ;mask interrupts
        movec   #0,sp             ;clear hardware stack pointer
        move    #0,omr            ;operating mode 0
        move    #$400,r5           ; initialise stack pointer
        move    #-1,m5            ; linear addressing
         movep   #$000108,x:M_IPRP


       include 'dispinit.asm'
         jsr     ada_init         ; initialize codec
         jsr     time_init        ; initialize timer
          jsr      sci_init
       ; initialization of necessary registers
         move    #sigxy,r3
          move #128,N1              ; to jump from left input or lowpass
         ; initialize registers for filtering
         move    #127,m1          ; use 128 sample circular buffers
         move    m1,m4            ; use 128 sample circular buffers
         move    #9,m3
         move    #$80FF,m6        ; force it to stay in 256 entry table
         move    #5,N3
         move    #buffer1,r0
         move    #$00FF,m0


loop_1

        jset    #2,x:M_SSISR0,*     ;wait for frame sync to pass
        jclr    #2,x:M_SSISR0,*     ;wait for frame sync

        move    x:RX_BUFF_BASE,a         ;receive left
        move    x:RX_BUFF_BASE+1,b       ;receive right
         move    a1,x:newx
         move    b1,x:newy

       asl    #3,a,a               ; scale up by 8!
       asl    #3,b,b
       move   a,x0
       move   b,y0
         jsr     process_stereo
       move   x:savex,a
       move   x:savey,b
         move    a,x:TX_BUFF_BASE         ;transmit left
         move    b,x:TX_BUFF_BASE+1       ;transmit right

         move    #TONE_OUTPUT,y0 ;set up control words
         move    y0,x:TX_BUFF_BASE+2
```

```
        move    #TONE_INPUT,y0
        move    y0,x:TX_BUFF_BASE+3

        jmp     loop_1

process_stereo
        ; demodulate new values of channel 1 and channel 2
        ;  from 10 kHz to 2 kHz
       bset   #1,x:M_PDRE   ; signals start of processing

       move   x:sinarg,r2
       move   #5,m2
       nop
        move x:(r2)+,y1              ;look up cos value
       move   #$FFFFFF,m2
       move   r2,x:sinarg          ; update pointer into sin table
        mpy     x0,y1,b           ;demodulate channels
        mpy     y0,y1,a b1,x0    ;modulate channel 1 and 2 from 10 to 2 kHz
        move            a1,y0


        ;Bandpass Filter Channel 1
        ;r1 must point to signal 1 buffer
        ;r4 must point to the filter coefficients
        ;m1 and m4 must be 127
        move    x:sig1strt,r1           ; load pointer to signal buffer
        move    #>bandpass,r4            ; load pointer to filter coeff's
        move #127,m1
        move m1,m4
       move #128,N1

        ; I guess the new value of signal 1 is in x0 and the new value
        ;   of signal 2 is in y0
        nop                         ;necessary for move function
        clr     a       x0,x:(r1)+      y:(r4)+,y1
        do #127,endflt1
        mac     x0,y1,a x:(r1)+,x0      y:(r4)+,y1
endflt1
        macr     x0,y1,a    (r1)+N1 ; do not decrement r1 as we use it for next
filter
                            ;    which is located 128 samples further in
memory
        ;Bandpass Filter Channel 2
       move   y0,x0
        clr     b       x0,x:(r1)+      y:(r4)+,y1
        do #127,endflt2
        mac     y1,x0,b x:(r1)+,x0      y:(r4)+,y1
endflt2
        macr    y1,x0,b (r1)- ;fix r1 in the circular buffer for next time
       move   (r1)-N1
       nop
        move    r1,x:sig1strt           ; save pointer in buffer

       asl    a                    ; scale up to matlab levels
       asl    b                    ; scale up to matlab levels
        ;Combine two filtered channels by squaring and adding
        move    a1,x0
        mpy     x0,x0,a b1,y0
        macr     y0,y0,a
                            ; x0 and y0 now contain filtered signals
;        asl     a                   ; scale up signal by 2
       move   x0,x:bpfiltx
       move   y0,x:bpfilty

;      bset   #1,x:M_PDRE   ; signals end of bp filtering

            ; This section writes out intermediate value to output
;      move   x0,x:savex


        ;Filter to detect signal (DC component) use AC signal for PLL
        ;coeff b0 b1 b2 b3 b4 scaleb -a1 -a2 -a3 -a4
```

```
        ;storage locations are for x0 x1 x2 x3 x4 y1 y2 y3 y4
        ;m3 must be 8=9-1, N3 must be 5
        ; new value starts out in acc. a
        ; The b coefficients are scaled up by 16384 for numerical accuracy
        ; The a coefficients are scaled down by 8 since they are greater than 1
        ; The scaleb scales the input portion down by 16384*8 to match the
        ;    output feedback portion
        ; The total is then scaled up by 8 to give the correct amplitude
        ;  (this is done with 3 asl's)
        move    a1,x0
        tfr     a,b   #butterb,r7
        clr     a       x0,x:(r3)+      y:(r7)+,y0
        do #5,endfltb
        mac     x0,y0,a x:(r3)+,x0      y:(r7)+,y0
endfltb
        move            a1,x1
;        mpy     x1,y0,a x:(r3)+,x0      y:(r7)+,y0  ;???
                ; prev inst replaced with next two
       asr    #17,a,a
       move          x:(r3)+,x0   y:(r7)+,y0
        do #3,endflta
        mac     x0,y0,a x:(r3)+,x0      y:(r7)+,y0
endflta
       macr   x0,y0,a
        asl     #3,a,a
        move    (r3)-N3
        move    a1,x:(r3)-N3
        sub     a,b     (r3)-           ; B contains AC part of signal
                                        ; A contains DC signal

       move a1,x:strength
;      bset   #1,x:M_PDRE   ; signals end of bp filtering


        ;Use PLL to lock on to 4 KHz signal in B
        ;   Data is organized in X: as: s3o s1o s2o
        ;   Dats is organized in Y: as: c1 c2 w2max w2min deltap c3
        ;                               s2max s2min scalesin sinloc
        move    #s3o,r2
        move    #c1,r6
        move    #$06,x1                  ; x1=.0469
        cmp     x1,a    X:(r2)+,x0      Y:(r6)+,y0  ; is DC<0.0469?
                       ;x0=s3o and y0=c1
        jmi     update2                  ; jump if it is


update1                 ; This update if signal strength is strong
        mpy     x0,y0,a X:(r2),x1       Y:(r6)+,y1
                        ; x1=s1o and y1=c2
        add     x1,a    ; a=s1o+c1*s3o=s1
        move            a1,X:(r2)+      ; store new s1
        mpy     x0,y1,a                 Y:(r6)+,x0 ; load x0=w2max
        add     x1,a    X:(r2),y1
        move            Y:(r6)+,y0
                        ; a=s1o+c2*s3o and y0=w2min and y1=s2o

        cmp     x0,a    ; is a>w2max?
        tpl     x0,a    ; if so, a=w2max
        cmp     y0,a                    Y:(r6)+,x0
                        ; is a<w2min and x0=deltap
        tmi     y0,a    ; if so, a=w2min
        add     x0,a                    Y:(r6)+,x0 ; x0=c3=1/(4*pi)
        move            a1,x1
        mpy     x0,x1,a ; scale for correct s2 level
        add     y1,a                    Y:(r6)+,x0 ; x0=s2max=0.5
                        ; a=s2=s2o+scale*(deltap+s1o+c2*s3o)
                        ;     =s2o+scale*(deltap+delta)
        cmp     x0,a                    ;
        jmi     chk2sml
        sub     x0,a    (r6)+
        sub     x0,a                    ; subtract 1 if >0.5
```

```
        jmp     donechk1
chk2sml
        move                Y:(r6)+,x1    ; x1=s2min=-0.5
        cmp     x1,a
        jpl     donechk1
        add     x0,a                      ; add 1 if < -0.5
        add     x0,a
donechk1
        move            a1,x0                       ;move s2 to x0 and
        move                    y:(r6)+,y0          ;load y0=scalesin
        mpyr    y1,y0,a x0,x:(r2)-                  ;store s2
        move            a1,n6                       ; old s2o value! as we want
        move            #>sinloc,r6
        move            (r2)-
        move                    y:(r6+n6),x1        ;x1=sin(s2o)
        asr   a                 b1,y1               ;y1=new input=W(kk) and a=a/2
       move         a1,n6                   ; index to find sin(s2o/2)
        mpy     x1,y1,a                             ;a=s3=sin(s2o)*W(new input)
        move            a1,x:(r2)+                  ;save s3=a
        jmp     updtdone                            ;x0 contains s2


update2                 ; This update if signal strength is weak
        move            #$7F,y0         ;y0=127/128
        clr     a       X:(r2),x1       Y:(r6)+,y1
                        ; x1=s1o and y1=c2
        mpy     x1,y0,a ; a=0.99*s1o=s1
        move            a1,X:(r2)+      ; store new s1
        clr     a                       Y:(r6)+,x0 ; load x0=w2max
        add     x1,a    X:(r2),y1
        move            Y:(r6)+,y0
                        ; a=s1o and y0=w2min and y1=s2o

        cmp     x0,a    ; is a>w2max?
        tpl     x0,a    ; if so, a=w2max
        cmp     y0,a                    Y:(r6)+,x0
                        ; is a<w2min and x0=deltap
        tmi     y0,a    ; if so, a=w2min
        add     x0,a                    Y:(r6)+,x0 ; x0=c3=1/(4*pi)
        move            a1,x1
        mpy     x0,x1,a ; scale for correct s2 level
        add     y1,a                    Y:(r6)+,x0 ; x0=s2max=0.5
                        ; a=s2=s2o+scale*(deltap+s10)
                        ;     =s2o+scale*(deltap+delta)
        cmp     x0,a                    ;
        jmi     chk2sml2
        sub     x0,a    (r6)+
        sub     x0,a                    ; subtract 1 if >0.5
        jmp     donechk2
chk2sml2
        move                            Y:(r6)+,x1 ; x1=s2min=-0.5
        cmp     x1,a
        jpl     donechk2
        add     x0,a                    ; add 1 if < -0.5
        add     x0,a
donechk2
        move            a1,x0                       ;move s2 to x0 and
        move                    y:(r6)+,y0          ;load y0=scalesin
        mpyr    y1,y0,a x0,x:(r2)-                  ;store s2
        move            a1,n6                       ; old s2o value! as we want
        move            #>sinloc,r6
        move            (r2)-
        move            y:(r6+n6),x1                ;x1=sin(s2o)
        asr   a                 b1,y1               ;y1=new input=W(kk) and a=a/2
       move   a1,n6
        mpy     x1,y1,a                             ;a=s3=sin(s2o)*W(new input)
        move    a1,x:(r2)+                          ;save s3=a
                                                    ;x0 contains s2
updtdone
```

```
;*************************************************************************
; COMMUNICATION Routine - Uses Matched Filters to decode FSK signals at   *
;   1.5 KHz and 4.5 KHz. Should run for every sample                      *
;*************************************************************************

         move    x:newx,x0
         move    x:newy,y0
        ;Bandpass Filter Channel 1
        ;r1 must point to signal 1 buffer
        ;r4 must point to the filter coefficients
        ;m1 and m4 must be 127
        move    x:sig3strt,r1          ; load pointer to signal buffer
        move    #>commfilt,r4           ; load pointer to filter coeff's
        move #127,m1
        move m1,m4
       move #128,N1

        ; I guess the new value of signal 1 is in x0 and the new value
        ;   of signal 2 is in y0
        nop                         ;necessary for move function
        clr     a       x0,x:(r1)+      y:(r4)+,y1
        do #127,endflt11
        mac     x0,y1,a x:(r1)+,x0      y:(r4)+,y1
endflt11
        macr     x0,y1,a    (r1)+N1 ; do not decrement r1 as we use it for next
filter
                              ;    which is located 128 samples further in
memory
        ;Bandpass Filter Channel 2
       move   y0,x0
        clr     b       x0,x:(r1)+      y:(r4)+,y1
        do #127,endflt12
        mac     y1,x0,b x:(r1)+,x0      y:(r4)+,y1
endflt12
        macr    y1,x0,b (r1)- ;fix r1 in the circular buffer for next time
       move   (r1)-N1
       nop
        move    r1,x:sig3strt          ; save pointer in buffer

       asl    a                    ; scale up to matlab levels
       asl    b                    ; scale up to matlab levels

        move    a1,x0
       move   b1,y0         ; x0 and y0 now contain filtered signals
;        move    x0,x:savey
       move   x0,x:comfiltx
       move   y0,x:comfilty
       move x0,x:savex
       move y0,x:savey

;*************************************************************************
; Find 1.5 KHz signal level now                                          *
;*************************************************************************

        ;Filter to detect 1.5 KHz signal in channel 1 and channel 2.
       ;  This is a matched filter. Results will be averaged over a
       ;  period of 16 samples, and then squared and added.

        ;r1 must point to signal 2 buffer
        ;r4 must point to the filter coefficients
        ;m1 and m4 must be 31
        move #31,m1
        move m1,m4
       move #32,N1
        move    x:sig4strt,r1          ; load pointer to signal buffer
        move    #>sinfilt0,r4           ; load pointer to filter coeff's
        ; I guess the new value of signal 1 is in x0 and the new value
        ;   of signal 2 is in y0
        nop                         ;necessary for move function
        clr     a       x0,x:(r1)+      y:(r4)+,y1
        do #31,endflt13
        mac     x0,y1,a x:(r1)+,x0      y:(r4)+,y1
```

```
endflt13
        macr    x0,y1,a    (r1)+N1       ; do not decrement r1 as we use it for
next filter

        ;Low pass Filter Channel 2
       move   y0,x0
        clr     b       x0,x:(r1)+      y:(r4)+,y1
        do #31,endflt14
        mac     y1,x0,b x:(r1)+,x0      y:(r4)+,y1
endflt14
        macr    y1,x0,b (r1)- ;fix r1 in the circular buffer for next time
       move   (r1)-N1
        move    r1,x:sig4strt           ; save pointer in buffer
                        ; a now contains channel x magnitude
                        ; b now contains channel y magnitude
       asr    #2,a,a
       asr    #2,b,b
       abs    a
       abs    b
        move    a1,x0
       move   b1,y0         ; x0 and y0 now contain filtered signals


        ;Filter to average detected 1.5 KHz signal in channel 1 and channel 2.
       ;  over a period of 16 samples, and then squared and added to
       ; form a 1.5 Khz signal level.

        ;r1 must point to signal 2 buffer
        ;r4 must point to the filter coefficients
        ;m1 and m4 must be 31
        move #15,m1
        move m1,m4
       move #16,N1
        move    x:sig6strt,r1           ; load pointer to signal buffer
        move    #>average1,r4           ; load pointer to filter coeff's
        ; I guess the new value of signal 1 is in x0 and the new value
        ;   of signal 2 is in y0
        nop                     ;necessary for move function
        clr     a       x0,x:(r1)+      y:(r4)+,y1
        do #15,endflt5
        mac     x0,y1,a x:(r1)+,x0      y:(r4)+,y1
endflt5
        macr    x0,y1,a    (r1)+N1      ; do not decrement r1 as we use it for
next filter

        ;Low pass Filter Channel 2
       move   y0,x0
        clr     b       x0,x:(r1)+      y:(r4)+,y1
        do #15,endflt6
        mac     y1,x0,b x:(r1)+,x0      y:(r4)+,y1
endflt6
        macr    y1,x0,b (r1)- ;fix r1 in the circular buffer for next time
       move   (r1)-N1
        move    r1,x:sig6strt           ; save pointer in buffer
                        ; a now contains channel x magnitude
                        ; b now contains channel y magnitude
        move    a1,x0

       mpyr   x0,x0,a      b1,y0       ; x0 and y0 now contain filtered signals
       macr   y0,y0,a              ; a now contains the 1.5 KHz signal level
       asl    #1,a,a       ; a=2*aorig
       move    #$70,x0             ;01110000=0.875=x0
       move   a1,y0        ;
       mpyr   x0,y0,a              ;a=0.875*a=1.75*aorig
       move   a1,x:logic0


;*************************************************************************
; Find 4.5 KHz signal level now                                          *
;*************************************************************************
```

```
        move   x:comfiltx,x0 ; load bandpass filtered signals again
        move   x:comfilty,y0


         ;Filter to detect 4.5 KHz signal in channel 1 and channel 2.
        ;  This is a matched filter. Results will be averaged over a
        ;  period of 16 samples, and then squared and added.

         ;r1 must point to signal 2 buffer
         ;r4 must point to the filter coefficients
         ;m1 and m4 must be 31
         move #31,m1
         move m1,m4
        move #32,N1
         move    x:sig5strt,r1          ; load pointer to signal buffer
         move    #>sinfilt1,r4           ; load pointer to filter coeff's
         ; I guess the new value of signal 1 is in x0 and the new value
         ;   of signal 2 is in y0
         nop                          ;necessary for move function
         clr     a       x0,x:(r1)+      y:(r4)+,y1
         do #31,endflt7
         mac     x0,y1,a x:(r1)+,x0      y:(r4)+,y1
endflt7
         macr    x0,y1,a    (r1)+N1      ; do not decrement r1 as we use it for
next filter

         ;Low pass Filter Channel 2
        move   y0,x0
         clr     b       x0,x:(r1)+      y:(r4)+,y1
         do #31,endflt8
         mac     y1,x0,b x:(r1)+,x0      y:(r4)+,y1
endflt8
         macr    y1,x0,b (r1)- ;fix r1 in the circular buffer for next time
        move   (r1)-N1
         move    r1,x:sig5strt          ; save pointer in buffer
                         ; a now contains channel x magnitude
                         ; b now contains channel y magnitude
        abs    a
        abs    b
        asr    #2,a,a
        asr    #2,b,b
         move    a1,x0
        move   b1,y0         ; x0 and y0 now contain filtered signals



         ;Filter to average detected 4.5 KHz signal in channel 1 and channel 2.
        ;  over a period of 16 samples, and then squared and added to
        ; form a 4.5 Khz signal level.

         ;r1 must point to signal 2 buffer
         ;r4 must point to the filter coefficients
         ;m1 and m4 must be 31
         move #15,m1
         move m1,m4
        move #16,N1
         move    x:sig7strt,r1          ; load pointer to signal buffer
         move    #>average1,r4           ; load pointer to filter coeff's
         ; I guess the new value of signal 1 is in x0 and the new value
         ;   of signal 2 is in y0
         nop                          ;necessary for move function
         clr     a       x0,x:(r1)+      y:(r4)+,y1
         do #15,endflt9
         mac     x0,y1,a x:(r1)+,x0      y:(r4)+,y1
endflt9
         macr    x0,y1,a    (r1)+N1      ; do not decrement r1 as we use it for
next filter

         ;Low pass Filter Channel 2
        move   y0,x0
         clr     b       x0,x:(r1)+      y:(r4)+,y1
         do #15,endflt10
```

```
        mac     y1,x0,b x:(r1)+,x0      y:(r4)+,y1
endflt10
        macr    y1,x0,b (r1)- ;fix r1 in the circular buffer for next time
       move   (r1)-N1
        move    r1,x:sig7strt           ; save pointer in buffer
                        ; a now contains channel x magnitude
                        ; b now contains channel y magnitude
        move    a1,x0
       mpyr   x0,x0,a       b1,y0         ; x0 and y0 now contain filtered signals
       macr   y0,y0,a               ; a now contains the 4.5 KHz signal level
;      move   a1,x:savey
       move   a1,x:logic1
       move   x:logic0,b
       sub    b,a             ; final result!
       asl    #1,a,a
;      move   a1,x:savex
       move   x:level,b
       cmp    #0,b
       jeq    still0
still1
       move   #$FB,x0              ;x0=-0.0391
       cmp    x0,a
       jpl    bitsame
       bclr   #2,x:M_PDRE   ; signals a 0 in communication
       move   #0,r1
       move   r1,x:level
       jmp    bitsame
still0
       move   #$05,x0              ;x0=0.0391
       cmp    x0,a
       jmi    bitsame
       bset   #2,x:M_PDRE   ; signals a 1 in communication
       move   #1,r1
       move   r1,x:level

bitsame


;*************************************************************************
; The following routines are only executed every 6th sample
;      Count=1:  Update 2 input PLL & Display Position
;      Count=2:  Update Real-Time Clock
;     Count=3:  Poll Serial Port and Display Comm. Message
;*************************************************************************

       clr    a      y:(r6+n6),x0          ;x0=sin(s2o/2)
        move    x:count,a1                      ;x1=count (down from six)
        sub   #1,a                              ;count=count-1
         move a1,x:count
        jne     rtc

        ; If it is the 6th time around, then process
        ;    This acts like a downsampling
        ;Use PLL to create 2 KHz demodulation signal and
        ;  demodulate channel 1 and channel 2 filtered signals
        move    #$6,a0
        move    a0,x:count                      ; reset sixcnt to 6
        move    x:bpfiltx,x1                    ; load channel 1 new val
        mpy     x0,x1,a                         ; demodulate ch 1
       move   x:bpfilty,y1              ; load ch2 val
        mpy     x0,y1,b         a1,x0           ; demodulate ch 2
        move                    b1,y0
                ;We now have X3(kk) in x0 and Y3(kk) in y0. They are
                ; now demodulated completely down from 10 KHz


       ;Filter demodulated signals for channel 1 and channel 2 to
        ;  remove  double frequency (4 KHz) component

        ;r1 must point to signal 2 buffer
        ;r4 must point to the filter coefficients
```

```
        ;m1 and m4 must be 15
        move #15,m1
        move m1,m4
       move #16,N1
        move    x:sig2strt,r1          ; load pointer to signal buffer
        move    #>lowpass,r4           ; load pointer to filter coeff's
        ; I guess the new value of signal 1 is in x0 and the new value
        ;   of signal 2 is in y0
        nop                         ;necessary for move function
        clr     a       x0,x:(r1)+      y:(r4)+,y1
        do #15,endflt3
        mac     x0,y1,a x:(r1)+,x0      y:(r4)+,y1
endflt3
        macr    x0,y1,a    (r1)+N1      ; do not decrement r1 as we use it for
next filter

        ;Low pass Filter Channel 2
       move   y0,x0
        clr     b       x0,x:(r1)+      y:(r4)+,y1
        do #15,endflt4
        mac     y1,x0,b x:(r1)+,x0      y:(r4)+,y1
endflt4
        macr    y1,x0,b (r1)- ;fix r1 in the circular buffer for next time
       move   (r1)-N1
        move    r1,x:sig2strt          ; save pointer in buffer
                        ; a now contains channel 1 (X4(kk))
                        ; b now contains channel 2 (Y4(kk))
        move    a1,b0   ; b1 now contains channel 2
                        ; b0 now contains channel 1

       move   b0,x0
       move   b1,y0
;      move   y0,x:savey
;      move   x0,x:savex



        ;Use channel 1 and channel 2 demodulated filtered signals and
        ;  non-linear observer (PLL) to track position
        ;   Data is organized in X: as: st3o st1o st2o
        ;   Dats is organized in Y: as: k c deltap2 c4=1/(4*pi)
        ;                           st2max st2min bigpos scalesin2
       clr    a
       move   x:strength,a1
        move    #st3o,r2
        move    #k,r6
        move    #$06,x1                  ; x1=.0469
        cmp     x1,a    X:(r2)+,x0      Y:(r6)+,y0  ; is DC<0.0469?
                       ;x0=st3o and y0=k
        jmi     update4                  ; jump if it is

update3                 ; This update if signal strength is strong
        mpy     x0,y0,a X:(r2),x1       Y:(r6)+,y1
                        ; x1=s1o and y1=c
        add     x1,a    ; a=st1o+k*st3o=s1
        move            a1,X:(r2)+      ; store new st1
        mpy     x0,y1,a                 X:(r2),y1 ; load y1=st2o
        add     x1,a    Y:(r6)+,x0      ;a=st1o+c*st3o
                                ; x0=deltap2
        add     x0,a                    Y:(r6)+,x0 ; x0=c4=1/(4*pi)
        move            a1,x1
        mpy     x0,x1,a ; scale for correct s2 level
        add     y1,a                    Y:(r6)+,x0 ; x0=st2max=0.25
                        ; a=st2=st2o+scale*(deltap2+st1o+c*st3o)
                        ;       =st2o+scale*(deltap2+delta2)
        cmp     x0,a                    ;
        jmi     chk2sml3
        sub     x0,a    (r6)+
        sub     x0,a                    ; subtract .5 if >0.25
        move    a1,x0                   ; move st2 to x0
        clr     a                       ; bigpos=bigpos+1
        move    y:(r6),a0
```

```
        inc     a
        move    a0,y:(r6)
       move   x0,a
        jmp     donechk3
chk2sml3
        move                                Y:(r6)+,x1 ; x1=st2min=-0.25
        cmp     x1,a
        jpl     donechk3
        add     x0,a                        ; add .5 if < -0.25
        add     x0,a
        move    a1,x0                       ; move st2 to x0
        clr     a                           ; bigpos=bigpos-1
        move    y:(r6),a0
        dec     a
        move    a0,y:(r6)
       move   x0,a
donechk3
       move    a1,x0
       move   x0,y:(r0)+
       move                (r6)+           ; point from bigpos to scalesin2
        move                    y:(r6)+,y0      ;load y0=scalesin2
        mpyr    y1,y0,a x0,x:(r2)-              ;store st2
        move            a1,n6                   ; old st2o value! as we want
        move            #>sinloc,r6             ; r6 points to sin table
        move            (r2)-
        move                    y:(r6+n6),x1    ;x1=sin(st2o)
        move            #>cosloc,r6
        move                    b1,y0           ;y0=new input=Y4(kk)
        move                    b0,x0           ;x0=new input=X4(kk)
        move                    y:(r6+n6),y1    ;y1=cos(st2o)
        mpy     x0,x1,a                         ;a=sin(st2o)*X4(kk)
        macr    y0,y1,a                         ; +cos(st2o)*Y4(kk)
        move            a1,x:(r2)+              ;save s3=a
        jmp     updtdone3                       ;


update4                 ; This update if signal strength is weak
         jmp      updtdone3  ; skip this part to see if we can eliminate
unwanted change
        clr     a       X:(r2),x1       Y:(r6)+,y1
                        ; x1=st1o and y1=c
        move    x1,a    ; a=st1o=st1 : don't update st1!
        move            a1,X:(r2)+      ; store new st1
        move            X:(r2),y1       ; y1=st2o
        move            Y:(r6)+,x0      ; x0=deltap2
        add     x0,a                    Y:(r6)+,x0 ;a=st1+deltap2
                                                   ;x0=c4=1/(4*pi)
        move            a1,x1
        mpy     x0,x1,a ; scale for correct st2 level
        add     y1,a                    Y:(r6)+,x0 ; x0=st2max=0.25
                        ; a=st2=st2o+scale*(deltap2+st1o)
                        ;       =st2o+scale*(deltap2+delta2)
        cmp     x0,a                    ;
        jmi     chk2sml4
        sub     x0,a    (r6)+
        sub     x0,a                    ; subtract 0.5 if >0.25
        move    a1,x0                   ; move st2 to x0
        clr     a                       ; bigpos=bigpos+1
        move    y:(r6),a0
        inc     a
        move    a0,y:(r6)
       move   x0,a
        jmp     donechk4
chk2sml4
        move                            Y:(r6)+,x1 ; x1=st2min=-0.25
        cmp     x1,a
        jpl     donechk4
        add     x0,a                    ; add 0.5 if < -0.25
        add     x0,a
        move    a1,x0                   ; move st2 to x0
        clr     a                       ; bigpos=bigpos-1
        move    y:(r6),a0
```

```
        dec     a
        move    a0,y:(r6)
       move   x0,a
donechk4
       move   a1,x0
       move   (r6)+
        move                    y:(r6)+,y0      ;load y0=scalesin2
        mpyr    y1,y0,a x0,x:(r2)-              ;store s2
        move            a1,n6                   ; old s2o value! as we want
        move            #>sinloc,r6
        move            (r2)-
        move                    y:(r6+n6),x1    ;x1=sin(st2o)
        move            #>cosloc,r6
        move                    b1,y0           ;y0=new input=Y4(kk)
        move                    b0,x0           ;x0=new input=X4(kk)
        move                    y:(r6+n6),y1    ;y1=cos(st2o)
        mpy     x0,x1,a                         ;a=sin(st2o)*X4(kk)
        macr    y0,y1,a                         ; +cos(st2o)*Y4(kk)
        move    a1,x:(r2)+                      ;save s3=a
                                                ;x0 contains s2
updtdone3
disprtn
       move   x:st2o,y0          ; st2o is position (mod 75 mm)
;      move   y0,x:savey
;      move   x0,x:savex
       move   #150,r1                  ; convert fraction to mm (.5=75mm)
       move   r1,x1
       mpy    x1,y0,b
       move   y:bigpos,y0
       move   #75,r1
       move   r1,x1
       mpy    y0,x1,a
       asl    #23,a,a
       add    a,b
;      neg    b                  ; make right positive position
       move   b1,x1

;*************************************************************************
; Display Routine                                                        *
;     This routine takes a number in x1 between 0 and 8400 and displays it  *
; on the 7 Segment, 4 digit LED Display.                                 *
;*************************************************************************
        clr     b                ;assume number is in x1
        move    #strtpos,r2              ;initialize indexes
        move    #decode,r4
        move    #digits1,r7
       move   #digits0,r1
;       move    #>-001234,x1
        move    x1,b                    ;assume 4 digit integer is in x1
       asr    #24,b,b
       cmp    #0,b
       jmi    negsign
       move   #lett0,y0
       move   y0,y:(r2)+
       move   #lett1,y0
       move   y0,y:(r2)+
       jmp    donesign
negsign
       move   #lettmin0,y0
       move   y0,y:(r2)+
       move   #lettmin1,y0
       move   y0,y:(r2)+
       neg    b
       move   b0,x1
donesign
        move    y:(r4)+,x0              ;load in first constant to convert
        mpy     x0,x1,a
        move    a,n1                    ;get ready to look up digit
       move   a,n7
        move    y:(r4)+,x0
        move    x:(r1+n1),y0            ;look up digit
        move    y0,y:(r2)+              ;display digit
```

```
        move    x:(r7+n7),y0            ;look up digit
        move    y0,y:(r2)+              ;display digit
        move    a,y1
        mac     -x0,y1,b
        move    y:(r4)+,x0
        move    b0,x1
        mpy     x0,x1,a
        move    a,n1                    ;get ready to look up digit
        move    a,n7                    ;get ready to look up digit
        move    y:(r4)+,x0
        move    x:(r1+n1),y0            ;look up digit
        move    y0,y:(r2)+              ;display digit
        move    x:(r7+n7),y0            ;look up digit
        move    y0,y:(r2)+              ;display digit
        move    a,y1
        mac     -x0,y1,b
        move    y:(r4)+,x0
        move    b0,x1
        mpy     x0,x1,a
        move    a,n1                    ;get ready to look up digit
        move    a,n7                    ;get ready to look up digit
        move    y:(r4)+,x0
        move    x:(r1+n1),y0            ;look up digit
        move    y0,y:(r2)+              ;display digit
        move    x:(r7+n7),y0            ;look up digit
        move    y0,y:(r2)+              ;display digit
        move    a,y1
        mac     -x0,y1,b
        move    b0,n1                    ;get ready to look up digit
        move    b0,n7                   ;get ready to look up digit
        move    y:(r4)+,x0
        move    x:(r1+n1),y0            ;look up digit
        move    y0,y:(r2)+              ;display digit
        move    x:(r7+n7),y0            ;look up digit
        move    y0,y:(r2)+              ;display digit
      jmp   donerts

;*************************************************************************
; This is a real-time clock routine.                                     *
; It keeps track of seconds, minutes, and hours. We can count up to 99 *
;    hours 59 minutes and 59 seconds befor we display something bad.    *
;*************************************************************************
;REAL-TIME CLOCK routine                                                 *
;*************************************************************************
rtc
        sub   #1,a                              ;count=count-1
        jne     checkcom                      ; if it wasn't 2, go to next option
                                               ; if it was 2, run RTC routine
         move       x:clkcnt,a
         sub  #1,a                           ; update the countdown
         move a1,x:clkcnt                    ; store the new value
         jne  donerts                       ; have we counted down 8000 (to 1 sec)
         move #8000,x0                       ; if so, re-initialize variable
         move x0,x:clkcnt                    ; and store it


         move r3,x:(r5)+
         move m3,x:(r5)+

        move        #compclk,r2             ; numbers to compare w/ each digit
for overflow
        move        #digits1,r4             ; since we are placing digits in
        move        #digits0,r1             ; reverse order, we use 2nd table
1st
        move        #endtime,r6             ; end of time section of display
mem
         move #$FFFFFF,m2
         move #$FFFFFF,m3
         move #seconds,r3

        clr     a
```

```
        move    x:(r3),a1                   ; SINGLE SECONDS
        add     #1,a
        move    x:(r2)+,x0          ; load a with single seconds
                                            ; load x with max+1 single seconds
        cmp     x0,a                        ; have we exceeded 9 seconds?
        jeq     ssok
        move    a1,n4
        move    a1,x:(r3)+
        move    n4,n1
        nop
        move    x:(r4+n4),x0
        move    x0,y:(r6)-
        move    x:(r1+n1),x0
        move    x0,y:(r6)-
        jmp     tim2isrdone

ssok
        clr     a
        nop
        move    a1,n4
        move    a1,x:(r3)+
        move    n4,n1
        nop
        move    x:(r4+n4),x0
        move    x0,y:(r6)-
        move    x:(r1+n1),x0
        move    x0,y:(r6)-


        move    x:(r3),a1                   ; TENS OF SECONDS
        add     #1,a
        move    x:(r2)+,x0          ; load a with single seconds
                                            ; load x with max+1 single seconds
        cmp     x0,a                        ; have we exceeded 9 seconds?
        jeq     tsok
        move    a1,n4
        move    a1,x:(r3)+
        move    n4,n1
        nop
        move    x:(r4+n4),x0
        move    x0,y:(r6)-
        move    x:(r1+n1),x0
        move    x0,y:(r6)-
        jmp     tim2isrdone
tsok
        clr     a
        nop
        move    a1,n4
        move    a1,x:(r3)+
        move    n4,n1
        nop
        move    x:(r4+n4),x0
        move    x0,y:(r6)-
        move    x:(r1+n1),x0
        move    x0,y:(r6)-


        move    x:(r3),a1                   ; SINGLE MINUTES
        add     #1,a
        move    x:(r2)+,x0          ; load a with single seconds
                                            ; load x with max+1 single seconds
        move    (r6)-
        cmp     x0,a        (r6)-           ; have we exceeded 9 seconds?
        jeq     smok
        move    a1,n4
        move    a1,x:(r3)+
        move    n4,n1
        nop
        move    x:(r4+n4),x0
        move    x0,y:(r6)-
```

```
        move    x:(r1+n1),x0
        move    x0,y:(r6)-
        jmp     tim2isrdone
smok
        clr     a
        nop
        move    a1,n4
        move    a1,x:(r3)+
        move    n4,n1
        nop
        move    x:(r4+n4),x0
        move    x0,y:(r6)-
        move    x:(r1+n1),x0
        move    x0,y:(r6)-


        move    x:(r3),a1                       ; TENS OF MINUTES
        add     #1,a
        move    x:(r2)+,x0          ; load a with single seconds
                                            ; load x with max+1 single seconds
        cmp     x0,a                        ; have we exceeded 9 seconds?
        jeq     tmok
        move    a1,n4
        move    a1,x:(r3)+
        move    n4,n1
        nop
        move    x:(r4+n4),x0
        move    x0,y:(r6)-
        move    x:(r1+n1),x0
        move    x0,y:(r6)-
        jmp     tim2isrdone
tmok
        clr     a
        nop
        move    a1,n4
        move    a1,x:(r3)+
        move    n4,n1
        nop
        move    x:(r4+n4),x0
        move    x0,y:(r6)-
        move    x:(r1+n1),x0
        move    x0,y:(r6)-


        move    x:(r3),a1                       ; SINGLE HOURS
        add     #1,a
        move    x:(r2)+,x0          ; load a with single seconds
                                            ; load x with max+1 single seconds
        move    (r6)-
        cmp     x0,a        (r6)-           ; have we exceeded 9 seconds?
        jeq     shok
        move    a1,n4
        move    a1,x:(r3)+
        move    n4,n1
        nop
        move    x:(r4+n4),x0
        move    x0,y:(r6)-
        move    x:(r1+n1),x0
        move    x0,y:(r6)-
        jmp     tim2isrdone
shok
        clr     a
        nop
        move    a1,n4
        move    a1,x:(r3)+
        move    n4,n1
        nop
        move    x:(r4+n4),x0
        move    x0,y:(r6)-
        move    x:(r1+n1),x0
        move    x0,y:(r6)-
```

```
        move    x:(r3),a1                       ; TENS OF HOURS
        add     #1,a
        move    x:(r2)+,x0                      ; load a with single seconds
        cmp     x0,a
        jeq     thok
        move    a1,n4
        move    a1,x:(r3)+
        move    n4,n1
        nop
        move    x:(r4+n4),x0
        move    x0,y:(r6)-
        move    x:(r1+n1),x0
        move    x0,y:(r6)-
        jmp     tim2isrdone
thok
        clr     a
        nop
        move    a1,n4
        move    a1,x:(r3)+
        move    n4,n1
        nop
        move    x:(r4+n4),x0
        move    x0,y:(r6)-
        move    x:(r1+n1),x0
        move    x0,y:(r6)-

tim2isrdone
       move   x:-(r5),m3
       move   x:-(r5),r3
       jmp    donerts


checkcom
       jclr   #2,x:M_SSR,donerts ; if no receive, continue
       move   #>nextchar,r4      ; place characters at end of display buffer
       move   #$FFFFFF,m4
       move   x:M_SRXL,a
       move   #strtchar,r1
       move   m4,m1
       move   y:(r4)+,x0    ; shift characters on display left one
       do     #78,endshift
       move   x0,y:(r1)+
       move   y:(r4)+,x0
endshift
       or     #>$000500,a
       move   a1,y:(r1)+
       and    #>$0001FF,a
       move   a1,y:(r1)


donerts

;      bclr   #0,x:M_PDRE   ; signals end of processing
       bclr   #1,x:M_PDRE

        rts

      include 'sciinit1.asm'
      include 'adainit8.asm'
       include 'timinit7.asm'
echo
        end
```

A.5 sciinit1.asm

```
;*****************************************************************************
;       SCIINIT1.ASM    Ver.2.0
;       Example program to initialize the SCI port to receive data
;        from the communications routine and turn it into characters
;
;       Copyright (c) Brian Perreault
;
;       History:
;               3/27/97 Created  ver.1.0
;*****************************************************************************
; R5,M5, and N5 must be reserved for these ISR's!
;*****************************************************************************

;*****************************************************************************
;
;      port usage:
;   RXD : used to receive data from comm. routine
;   SCLK : used to send data to RXD fro comm. routine
;*****************************************************************************


;*****************************************************************************
;
;
;  PROGRAM OUTLINE:
;
;1 Set up timer 0 to toggle twice per second at 16.9344 MHz*4 clock freq
;
;*****************************************************************************
;
;
        org     p:
sci_init
       movep  #1,x:M_PCRE   ; enable RXD for SCI input
         movep    #$00010A,x:M_SCR          ; start counter at 0
       movep    #$000340,x:M_SCCR    ; divide by 833 for 1500 bps
         rts
```

A.6 adainit8.asm

```
       page    132,60
;*******************************************************************************
;      adainit8.asm
;      initialization routine for the CS4215
;
;*******************************************************************************
; R0,M0, and N0 must be reserved for these ISR's!
; R7 and M7 are reserved for display output
;*******************************************************************************

;*******************************************************************************
;
;      portc usage:
;   bit8: SSI TX (from DSP to Codec)
;   bit7:
;   bit6:
;   bit5:
;   bit4: codec reset (from DSP to Codec)
;   bit3:
;        bit2: data/control bar
;             0=control
;             1=data
;
;*******************************************************************************
;*****          initialize the CS4215 codec                              *****
;*******************************************************************************
;
;
;  PROGRAM OUTLINE:
;
;1 program fsync and sclk == output
;2 write pc0 = 0 (control mode)
;3 send 64 bit frame x times, with dcb bit = 0, keep doing until read back as 0
;4 send 64 bit frame x times, with dcb bit = 1, keep doing until read back as 1
;5 re-program fsync and sclk == input
;6 write pc0 = 1 (data mode)
;7 receive/send data (echo slots 1,2,3,4; slots 5,6,7,8 == constants)
;
;*******************************************************************************
;
;      initialize ssi -- fsync and sclk ==> outputs
;
        org     p:
ada_init
        movep   #$0000,x:M_PCRC          ; turn off ESSI0 port (for now)
        movep   #$103807,x:M_CRA0        ; 40MHz/16 = 2.5MHz SCLK, WL=16 bits,
4W/F
        movep   #$ff313C,x:M_CRB0        ; RIE,TIE,RLIE,TLIE,RE,TE,sc2/sck
outputs
        movep   #$0003,x:M_PRRC          ; setup pd0 and pd1 as gpio output
        movep   #$0,x:M_PDRC             ; send out a 0 on DC~ and RST_CODEC~

           ;----reset delay for codec ----
        do      #1000,_delay_loop
        rep     #2000                    ; 100 us delay (assuming 40MHz VCO)
        nop
_delay_loop

        bset    #0,x:M_PDRC              ; sends out a 1 on pd0 (rst_codec=1)
        movep   #$0108,x:M_IPRP          ; set interrupt priority level for ESSI0
to 3
                                         ; also set interrupt level for timer to
be equal
                                         ;   to that of ESSIO. I think the actual
IPL is 2.
        andi    #$FC,mr                  ; enable interrupts

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

```
; The following data sets up the CS4215 control mode data:
;    (CTS = Control Time Slot, U/LN = upper/lower Nibble)
;
;       +------ CTS1-UN:   0      0      1      MLB      0 0 0 0
;       |+----- CTS1-LN:  OLB    CLB     X       X       0 0 0 0
;       ||+---- CTS2-UN:  HPF     X     DFR2   DFR1      0 0 1 0
;       |||+--- CTS2-LN:  DFR0   ST      DF1    DF0      1 1 0 0
; x0 = $002Cxx
;
;       +------ CTS3-UN:  ITS    MCK2    MCK1   MCK0     1 0 0 0
;       |+----- CTS3-LN:  BSEL1  BSEL0   XCLK   XEN      1 0 0 0
;       ||+---- CTS4-UN:  TEST   TEST    TEST   TEST     (TEST MUST BE 0)
;       |||+--- CTS4-LN:  TEST   TEST    ENL    DAD      0 0 0 0
; x0 = $8800xx
;*****************************************************************************

;--- set up buffer with control mode data
        move    #CTRL_WD_12,x0
        move            x0,x:TX_BUFF_BASE
        move    #CTRL_WD_34,x0
        move            x0,x:TX_BUFF_BASE+1
        move    #CTRL_WD_56,x0
        move           x0,x:TX_BUFF_BASE+2
         move    #CTRL_WD_78,x0
         move            x0,x:TX_BUFF_BASE+3

        movep   #$003C,x:M_PCRC         ;turn on ESSI0 except for sc0 and sc2

;
; CLB == 0
;
        jclr    #3,x:M_SSISR0,*         ; wait until rx frame bit==1
        jset    #3,x:M_SSISR0,*         ; wait until rx frame bit==0
        jclr    #3,x:M_SSISR0,*         ; wait until rx frame bit==1
        jset    #18,x:RX_BUFF_BASE,*    ; loop until CLB set

;
; CLB == 1
;
        bset    #18,x:TX_BUFF_BASE      ;set CLB
        do      #4,_init_loopB
        jclr    #2,x:M_SSISR0,*         ; wait until tx frame bit==1
        jset    #2,x:M_SSISR0,*         ; wait until tx frame bit==0
_init_loopB
        movep   #$0000,x:M_PCRC         ; disable ESSI0

;*****************************************************************************
;    now CLB should be 1 -- re-program fsync and sclk direction (i/p) -- also,
;  circular buffer pointers for echoing data r0=current, r1=old data to send
;  1 frame later
;
        movep   #$103807,x:M_CRA0       ; 40MHz/16 = 2.5MHz SCLK, WL=16 bits,
4W/F
        movep   #$FF310C,x:M_CRB0       ; sckd and fsync (sc02) as inputs
        movep   #$0003,x:M_PDRC         ; D/C~ pin = 1 ==> data mode
        movep   #$003C,x:M_PCRC         ; turn on ESSI0 except for sc0 and sc2
        rts

;*****************************************************************************
;       SSI0_ISR.ASM    Ver.2.0
;       Example program to handle interrupts through
;        the 56303 SSI0 to move audio through the CS4215
;
;       Copyright (c) MOTOROLA 1995, 1996
;                     Semiconductor Products Sector
;                     Digital Signal Processing Division
;
;        upon entry:
;                R5 must be display counter only!
;        corrupts:
;                R0,
;
```

```
;       History:
;               14 June 1996: RLR/LJD - ver 1.0
;*******************************************************************************


;----the actual interrupt service routines (ISRs)  follow:

;************************ SSI TRANSMIT ISR *********************************
ssi_txe_isr
        bclr    #4,x:M_SSISR0           ; Read SSISR to clear exception flag
                                        ; explicitly clears underrun flag
ssi_tx_isr
        move    r0,x:(r5)+              ; Save r0 to the stack.
        move    m0,x:(r5)+              ; Save m0 to the stack.
        move    #3,m0                   ; Modulus 4 buffer.
        move    x:TX_PTR,r0             ; Load the pointer to the tx buffer.
        nop
        movep   x:(r0)+,x:M_TX00        ; SSI transfer data register.
        move    r0,x:TX_PTR             ; Update tx buffer pointer.
        move    x:-(r5),m0              ; Restore m0.
        move    x:-(r5),r0              ; Restore r0.
        rti

;********************** SSI TRANSMIT LAST SLOT ISR **************************
ssi_txls_isr
        move    r0,x:(r5)+              ; Save r0 to the stack.
        move    #TX_BUFF_BASE,r0        ; Reset pointer.
        move    r0,x:TX_PTR             ; Reset tx buffer pointer just in
                                        ; case it was corrupted.
        move    x:-(r5),r0              ; Restore r0.
        rti

;************************** SSI receive ISR *********************************
ssi_rxe_isr
        bclr    #5,x:M_SSISR0           ; Read SSISR to clear exception flag
                                        ; explicitly clears overrun flag
ssi_rx_isr
        move    r0,x:(r5)+              ; Save r0 to the stack.
        move    m0,x:(r5)+              ; Save m0 to the stack.
        move    #3,m0                   ; Modulo 4 buffer.
        move    x:RX_PTR,r0             ; Load the pointer to the rx buffer.
        nop
        movep   x:M_RX0,x:(r0)+         ; Read out received data to buffer.
        move    r0,x:RX_PTR             ; Update rx buffer pointer.
        move    x:-(r5),m0              ; Restore m0.
        move    x:-(r5),r0              ; Restore r0.
        rti

;********************** SSI receive last slot ISR ***************************
ssi_rxls_isr
        move    r0,x:(r5)+              ; Save r0 to the stack.
      move   r2,x:(r5)+          ;NEW
      move   m2,x:(r5)+
        move    #255,m2                 ; 256 bytes of display memory!
        move    #RX_BUFF_BASE,r0        ; Reset rx buffer pointer just in
                                        ; case it was corrupted.
      move   x:dispptr,r2        ;NEW
        move    r0,x:RX_PTR             ; Update rx buffer pointer.
        move    y:(r2)+,r0        ; load new command for display
      move   r2,x:dispptr        ;NEW
      move   x:-(r5),m2
      move   x:-(r5),r2          ;NEW
        move    r0,x:HDR                ; write 1/2 instruction to display
        move    x:-(r5),r0              ; Restore r0.
        rti
```

8.1 A.7 timinit7.asm

```
       page    132,60
;*****************************************************************************
;       TIMINIT1.ASM    Ver.2.0
;       Example program to initialize the triple timer module
;
;       Copyright (c) Brian Perreault
;
;       History:
;               3/27/97 Created  ver.1.0
;*****************************************************************************
; R5,M5, and N5 must be reserved for these ISR's!
;*****************************************************************************

;*****************************************************************************
;
;      port usage:
;   TIO0 : connected to LED in series with resistor.
;*****************************************************************************


;*****************************************************************************
;
;
;  PROGRAM OUTLINE:
;
;1 Set up timer 0 to toggle twice per second at 16.9344 MHz*4 clock freq
;
;*****************************************************************************
;
;
        org     p:
time_init
                                        ; TIMER 0
        movep   #0,x:M_TLR0             ; start counter at 0
        movep   #16000000,x:M_TCPR0     ; toggle about twice per second
        movep   #$000A21,x:M_TCSR0      ; put timer in toggle mode with
                                      ; TIO pin as output

      ; Real time clock function moved to main routine  TIMER 2
;       movep   #>35610,x:M_TPLR          ; prescale divides to 1/1124 second
intervals
;       movep   #>01,x:M_TLR2             ; start counter at 0
;       movep   #>01124,x:M_TCPR2         ; toggles at almost exactly 1 sec
;       movep   #$008A25,x:M_TCSR2        ; put timer in toggle mode with
;       andi    #$FC,mr                 ; enable interrupts
        rts

;*****************************************************************************
; This is a real-time clock routine.
; It keeps track of seconds, minutes, and hours. We can count up to 99
;    hours 59 minutes and 59 seconds befor we display something bad.
```

A.8 measure4.asm

```
;*******************************************************************************
;measure4.ASM - Reads in 2 quadrature signals modulated to 10 kHz
;               Demodulates the signals with help from a PLL
;               Uses 2 input PLL (non-linear observer) to track position
;                 difference between vehicles
;               Uses interpolated sin tables for better accuracy
;               Implements adjustments to non-linear observer to improve
;                 accuracy
;               Converts position to decimal, and displays position to
;                 micron resolution
;               Sends position value in ASCII out serial port when button is
;                 pushed.  Utilized timeout to prevent double push
;*******************************************************************************
; Note: R5 is designated as the stack pointer for routines.
;*******************************************************************************
      nolist
      include 'ioequ.asm'
      include 'intequ.asm'
      include 'ada_equ.asm'
      include 'vectors5.asm'
      list
       opt cc
;*******************************************************************************
; 4/6/97 added in 2 input PLL for position detection
; 4/6/97 added in display of position
; 5/2/97 added compensation for different sensor amplitudes!
; 5/10/97 added compensation (in sin table) for phase offset!
;---Buffer for talking to the CS4215

       org    x:0
RX_BUFF_BASE  equ     *
RX_data_1_2   ds      1       ;data time slot 1/2 for RX ISR
RX_data_3_4   ds      1       ;data time slot 3/4 for RX ISR
RX_data_5_6   ds      1       ;data time slot 5/6 for RX ISR
RX_data_7_8   ds      1       ;data time slot 7/8 for RX ISR

TX_BUFF_BASE  equ     *
TX_data_1_2   ds      1       ;data time slot 1/2 for TX ISR
TX_data_3_4   ds      1       ;data time slot 3/4 for TX ISR
TX_data_5_6   ds      1       ;data time slot 5/6 for TX ISR
TX_data_7_8   ds      1       ;data time slot 7/8 for TX ISR

RX_PTR          ds      1  ; Pointer for rx buffer
TX_PTR          ds      1  ; Pointer for tx buffer
tempr5        ds      1
dispptr               dc      DISPLIST
strength      dc      0
      org     x:$010
sigxy   bsc 10,$0          ; circular buffer for butterworth LPF
                           ;filter zero coefficients for butterworth LPF
                           ;organized as: b0 b1 b2 b3 b4 -a1 -a2 -a3 -a4
s3o     dc  0              ; states for high freq PLL
s1o     dc  0
s2o     dc  0
sixcnt  dc  6
sinarg  dc  sincoef
      org     x:$020
sincoef dc  0.5,-0.5,-0.9999999,-0.5,0.5,0.9999999 ;last one is supposed to be 1
st3o    dc  0              ; states for low freq PLL
st1o    dc  0
st2o    dc  0
sig1strt dc input1 ;start circular buffer at 0
sig2strt dc lpsig1 ;start circular buffer at lpsig1
sig3strt dc input3
count   dc  6
bpfiltx       dc  0
bpfilty dc  0
temp33  dc 0
```

```
temp44  dc 0
temp55  dc 0
clock
seconds          dc       0
tenseconds       dc       0
minutes          dc       0
tenminutes       dc       0
hours            dc       0
tenhours         dc       0
nothing          dc       0
irqaset dc  0        ; 0 if not pushed yet and 1 if it needs handling
irqdset dc  0        ; 0 if not pushed yet and 1 if it needs handling
xmitstrt dc 0        ; 0 if transmitting has not yet started
savex  dc  $400000
savey   dc  0
wait1sec      dc 0
       org    x:$0050
compclk          dc       10,6,10,6,10,10,6,0
savepos           dc 0

       org    x:$0060
digits0          dc
lett00,lett10,lett20,lett30,lett40,lett50,lett60,lett70,lett80,lett90
       org    x:$0070
digits1          dc
lett01,lett11,lett21,lett31,lett41,lett51,lett61,lett71,lett81,lett91

        org     x:$0080
input1  bsc 128,$0       ; circular buffer for input 1 filter
input2  bsc 128,$0       ; circular buffer for input 2 filter

lpsig1  bsc 16,$0        ; circular buffer for demodulated input 1
lpsig2  bsc 16,$0        ; circular buffer for demodulated input 2
        org      x:$0200
asciiout dc 72,69,76,76,79,13,7,7,7,32,32
asciiend dc  32
        org      x:$0210
digits2       dc     48,49,50,51,52,53,54,55,56,57
asciicnt      dc     0
asciineg      equ    45
asciispc      equ    32
asciibeep    equ     7
asciilf       equ    10
asciicr       equ    13
asciiper      equ    46
        org      x:$0400
input3  bsc 512,$0       ; cirular buffer for output averaging


       org    y:0

DISPLEN          equ 63 ; should be ((*-DISPLIST)-1) but isn't

        include 'asm\butter.asm'    ; include butterworth filter coeff's
                                  ;filter zero coefficients for butterworth LPF
                                  ;organized as: b0 b1 b2 b3 b4 -a1 -a2 -a3 -a4
c1      dc 0.0021418412             ; coefficients for fast PLL
c2      dc 0.5235987756
w2max   dc 0.0490873852
w2min   dc -0.0490873852
deltap  dc 0.52359877560
c3      dc (1/(4*3.14159265359))
s2max   dc 0.5
s2min   dc -0.5
scalesin dc 512

;Coefficients for two input PLL (Non-linear Observer)
k       dc  .0007815            ; coefficients for fast PLL
c       dc  .2499
deltap2 dc  0
c4      dc (1/(4*3.14159265359))
st2max   dc 0.25
```

```
st2min   dc -0.25
;st2max   dc 0.5
;st2min   dc -0.5
bigpos  dc 0
scalesin2 dc 512
;scalesin2 dc 10
                ; THESE numbers are for the display routine
       org    y:$0020
decode  dc      0.0010001,500,0.0100002,50,0.1000002,5 ; These constants will be
used to
                    ; convert binary to decimal. It is VERY important to note
that every
                    ; other one of these number is a small delta above 10^(-k).
This
                    ; prevents errors at borders of 1000's, 100's and 10's. (if
it were a
                    ; little less, we would be low by 1 digit and have problems.
digits  dc      $88,$EB,$4C,$49,$2B,$19,$18,$CB,$08,$09

        org     y:$0080 ; next thing must start on a 128 byte boundary;

        org     y:$100
       include      'asm\sintable.asm' ; location of sine table in memory
                                ; also location of cos table with
                                ; shift and wraparound

        org     y:$0200         ;definitely in internal memory
        include 'asm\bandpass.asm' ; include bandpass filter coeff's
        include 'asm\lowpass.asm'  ; include lowpass filter coefficients
       org    y:$0300
DISPLIST
                dc      lcdcommand,lcdpos800,lcdpos801,lcddata
                dc      lettt0,lettt1,letti0,letti1,lettm0,lettm1,lette0,lette1
                dc
lettcol0,lettcol1,lett0,lett1,lett00,lett01,lett00,lett01
                dc
lettcol0,lettcol1,lett00,lett01,lett00,lett01,lettcol0,lettcol1
                dc      lett00,lett01,lett00
endtime         dc      lett01
             dc     lcdcommand,lcdpos940,lcdpos941,lcddata
                dc      lettp0,lettp1,letto0,letto1,letts0,letts1
                dc      letti0,letti1,lettt0,lettt1,letti0,letti1
                dc      letto0,letto1,lettn0,lettn1,lettcol0,lettcol1
strtpos         dc      lett0,lett1
             dc     lett00,lett01,lett00,lett01,lett00,lett01
endpos          dc
lett00,lett01,lett00,lett01,lett0,lett1,lett0,lett1,(lettm0+32),(lettm1+32)
             dc     (lettm0+32),(lettm1+32)
             dc     lcdcommand,lcdposc00,lcdposc01,lcddata
;            dc     lettr0,lettr1,lette0,lette1,lettl0,lettl1
;            dc     letta0,letta1,lettt0,lettt1,letti0,letti1
;            dc     lettv0,lettv1,lette0,lette1,lett0,lett1
             dc     lettp0,lettp1,letto0,letto1,letts0,letts1
             dc     letti0,letti1,lettt0,lettt1,letti0,letti1
             dc     letto0,letto1,lettn0,lettn1,lett0,lett1
             dc     letts0,letts1,lette0,lette1,lettn0,lettn1
             dc     letts0,letts1,letti0,letti1,lettn0,lettn1
             dc     lettg0,lettg1,lett0,lett1,lettd0,lettd1,lette0,lette1
             dc     lettm0,lettm1,letto0,letto1,lettex0,lettex1
crappos
             bsc 134,lcdcommand
endcrap
         org      y:$0400
         include 'asm\ave2.asm' ; include averaging filter coefficients
          org       y:$0600
          include 'asm\coserr.asm' ; include sin table with offset error!


TONE_OUTPUT      EQU      HEADPHONE_EN+LINEOUT_EN
TONE_INPUT       EQU
MIC_IN_SELECT+(15*MONITOR_ATTN)+(15*LEFT_GAIN)+(15*RIGHT_GAIN)
CTRL_WD_12    equ    NO_PREAMP+HI_PASS_FILT+SAMP_RATE_48+STEREO+DATA_16    ;CLB=0
```

```
CTRL_WD_34    equ    IMMED_3STATE+XTAL1_SELECT+BITS_64+CODEC_MASTER
CTRL_WD_56    equ    $000000
CTRL_WD_78    equ    $000000

;HPCR            equ     $FFFFC4
;HDDR            equ     $FFFFC8
;HDR             equ     $FFFFC9

        include 'dispequ.asm'


        org     p:$100
START
main
;        movep   #$040003,x:M_PCTL     ; set PLL for MPY of 4X
        movep   #$A40033,x:M_PCTL     ; set PLL for MPY of 52X and DIV by 11
                                      ; for actual speed of 80.053527 MHz
        movep   #$012421,x:M_BCR    ; set up one ext. wait state for all AAR
areas
;       movep  #3,x:M_PRRE
        ori     #3,mr             ;mask interrupts
        movec   #0,sp             ;clear hardware stack pointer
        move    #0,omr            ;operating mode 0
        move    #$300,r5           ; initialise stack pointer
        move    #-1,m5            ; linear addressing
       movep   #$000108,x:M_IPRP
;        move    #>31,m5           ; display buffer is 32 bytes long!
;        move    #>DISPLIST,r5     ; we will always write to this list buffer,
                                   ; and the receive ISR will take care of
                                   ; displaying the buffer.


       include 'dispinit.asm'
;      jmp    disprtn
        jsr     ada_init          ; initialize codec
;        jsr     time_init         ; initialize timer
       jsr    sci_init            ; initialize sci for output

       ; initialization of necessary registers
        move    #sigxy,r3
       move   #128,N1              ; to jump from left input or lowpass
;      move   #16,N5        ;    buffer to the right one
        ; initialize registers for filtering
        move    #127,m1           ; use 128 sample circular buffers
        move    m1,m4             ; use 128 sample circular buffers
        move    #9,m3
        move    #$80FF,m6         ; force it to stay in 256 entry table
        move    #5,N3
;        move    #buffer1,r0
;        move    #$00FF,m0

        movep #$000FFF,x:M_IPRC    ; enable IRQ interrupts, and set to level 2

loop_1

        jset    #2,x:M_SSISR0,*    ;wait for frame sync to pass
        jclr    #2,x:M_SSISR0,*    ;wait for frame sync

        move    x:RX_BUFF_BASE,a         ;receive left
        move    a,x0
        move    #>0.9672,y0
        mpy     x0,y0,a                      ; scale measurement so signal levels
are equal
        move    x:RX_BUFF_BASE+1,b       ;receive right
       asl    #3,a,a               ; scale up by 8!
       asl    #3,b,b
       move   a,x0
       move   b,y0
        jsr     process_stereo
       move   x:savex,a
       move   x:savey,b
        move    a,x:TX_BUFF_BASE         ;transmit left
```

```
        move    b,x:TX_BUFF_BASE+1      ;transmit right

        move    #TONE_OUTPUT,y0 ;set up control words
        move    y0,x:TX_BUFF_BASE+2
        move    #TONE_INPUT,y0
        move    y0,x:TX_BUFF_BASE+3

        jmp     loop_1

process_stereo
        ; demodulate new values of channel 1 and channel 2
        ;  from 10 kHz to 2 kHz
;       bset    #0,x:M_PDRE   ; signals start of processing

       move   x:sinarg,r2
       move   #5,m2
       nop
        move x:(r2)+,y1              ;look up cos value
       move   #$FFFFFF,m2
       move   r2,x:sinarg          ; update pointer into sin table
        mpy     x0,y1,b          ;demodulate channels
        mpy     y0,y1,a b1,x0    ;modulate channel 1 and 2 from 10 to 2 kHz
        move             a1,y0


        ;Bandpass Filter Channel 1
        ;r1 must point to signal 1 buffer
        ;r4 must point to the filter coefficients
        ;m1 and m4 must be 127
        move    x:sig1strt,r1           ; load pointer to signal buffer
        move    #>bandpass,r4            ; load pointer to filter coeff's
        move #127,m1
        move m1,m4
       move #128,N1

        ; I guess the new value of signal 1 is in x0 and the new value
        ;   of signal 2 is in y0
        nop                         ;necessary for move function
        clr     a       x0,x:(r1)+      y:(r4)+,y1
        do #127,endflt1
        mac     x0,y1,a x:(r1)+,x0      y:(r4)+,y1
endflt1
        macr     x0,y1,a    (r1)+N1 ; do not decrement r1 as we use it for next
filter
                                ;    which is located 128 samples further in
memory
        ;Bandpass Filter Channel 2
       move   y0,x0
        clr     b       x0,x:(r1)+      y:(r4)+,y1
        do #127,endflt2
        mac     y1,x0,b x:(r1)+,x0      y:(r4)+,y1
endflt2
        macr    y1,x0,b (r1)- ;fix r1 in the circular buffer for next time
       move   (r1)-N1
       nop
        move    r1,x:sig1strt           ; save pointer in buffer

       asl    a                    ; scale up to matlab levels
       asl    b                    ; scale up to matlab levels
        ;Combine two filtered channels by squaring and adding
        move    a1,x0
        mpy     x0,x0,a b1,y0
        macr     y0,y0,a
                            ; x0 and y0 now contain filtered signals
;        asl     a                   ; scale up signal by 2
       move   x0,x:bpfiltx
       move   y0,x:bpfilty

;      bset   #1,x:M_PDRE   ; signals end of bp filtering

              ; This section writes out intermediate value to output
;      move   x0,x:savex
```

```
        ;Filter to detect signal (DC component) use AC signal for PLL
        ;coeff b0 b1 b2 b3 b4 scaleb -a1 -a2 -a3 -a4
        ;storage locations are for x0 x1 x2 x3 x4 y1 y2 y3 y4
        ;m3 must be 8=9-1, N3 must be 5
        ; new value starts out in acc. a
        ; The b coefficients are scaled up by 16384 for numerical accuracy
        ; The a coefficients are scaled down by 8 since they are greater than 1
        ; The scaleb scales the input portion down by 16384*8 to match the
        ;    output feedback portion
        ; The total is then scaled up by 8 to give the correct amplitude
        ;  (this is done with 3 asl's)
        move    a1,x0
        tfr     a,b   #butterb,r7
        clr     a       x0,x:(r3)+      y:(r7)+,y0
        do #5,endfltb
        mac     x0,y0,a x:(r3)+,x0      y:(r7)+,y0
endfltb
        move            a1,x1
;        mpy     x1,y0,a x:(r3)+,x0      y:(r7)+,y0  ;???
                ; prev inst replaced with next two
       asr    #17,a,a
       move         x:(r3)+,x0    y:(r7)+,y0
        do #3,endflta
        mac     x0,y0,a x:(r3)+,x0      y:(r7)+,y0
endflta
       macr   x0,y0,a
        asl     #3,a,a
        move    (r3)-N3
        move    a1,x:(r3)-N3
        sub     a,b     (r3)-           ; B contains AC part of signal
                                        ; A contains DC signal

       move a1,x:strength
;      bset   #1,x:M_PDRE   ; signals end of bp filtering


        ;Use PLL to lock on to 4 KHz signal in B
        ;   Data is organized in X: as: s3o s1o s2o
        ;   Dats is organized in Y: as: c1 c2 w2max w2min deltap c3
        ;                               s2max s2min scalesin sinloc
        move    #s3o,r2
        move    #c1,r6
        move    #$06,x1                  ; x1=.0469
        cmp     x1,a    X:(r2)+,x0      Y:(r6)+,y0  ; is DC<0.0469?
                       ;x0=s3o and y0=c1
        jmi     update2                  ; jump if it is


update1                 ; This update if signal strength is strong
        mpy     x0,y0,a X:(r2),x1       Y:(r6)+,y1
                        ; x1=s1o and y1=c2
        add     x1,a    ; a=s1o+c1*s3o=s1
        move            a1,X:(r2)+      ; store new s1
        mpy     x0,y1,a                 Y:(r6)+,x0 ; load x0=w2max
        add     x1,a    X:(r2),y1
        move            Y:(r6)+,y0
                        ; a=s1o+c2*s3o and y0=w2min and y1=s2o

        cmp     x0,a    ; is a>w2max?
        tpl     x0,a    ; if so, a=w2max
        cmp     y0,a                    Y:(r6)+,x0
                        ; is a<w2min and x0=deltap
        tmi     y0,a    ; if so, a=w2min
        add     x0,a                    Y:(r6)+,x0 ; x0=c3=1/(4*pi)
        move            a1,x1
        mpy     x0,x1,a ; scale for correct s2 level
        add     y1,a                    Y:(r6)+,x0 ; x0=s2max=0.5
                        ; a=s2=s2o+scale*(deltap+s1o+c2*s3o)
                        ;      =s2o+scale*(deltap+delta)
```

```
        cmp     x0,a                    ;
        jmi     chk2sml
        sub     x0,a    (r6)+
        sub     x0,a                    ; subtract 1 if >0.5
        jmp     donechk1
chk2sml
        move              Y:(r6)+,x1    ; x1=s2min=-0.5
        cmp     x1,a
        jpl     donechk1
        add     x0,a                    ; add 1 if < -0.5
        add     x0,a
donechk1
        move            a1,x0                   ;move s2 to x0 and
        move                    y:(r6)+,y0      ;load y0=scalesin
        mpyr    y1,y0,a x0,x:(r2)-              ;store s2
        move            a1,n6                   ; old s2o value! as we want
        move            #>sinloc,r6
        move            (r2)-
        move                    y:(r6+n6),x1    ;x1=sin(s2o)
        asr   a                 b1,y1           ;y1=new input=W(kk) and a=a/2
       move         a1,n6                   ; index to find sin(s2o/2)
        mpy     x1,y1,a                         ;a=s3=sin(s2o)*W(new input)
        move            a1,x:(r2)+              ;save s3=a
        jmp     updtdone                        ;x0 contains s2


update2                 ; This update if signal strength is weak
        move            #$7F,y0         ;y0=127/128
        clr     a       X:(r2),x1       Y:(r6)+,y1
                        ; x1=s1o and y1=c2
        mpy     x1,y0,a ; a=0.99*s1o=s1
        move            a1,X:(r2)+      ; store new s1
        clr     a                       Y:(r6)+,x0 ; load x0=w2max
        add     x1,a    X:(r2),y1
        move            Y:(r6)+,y0
                        ; a=s1o and y0=w2min and y1=s2o

        cmp     x0,a    ; is a>w2max?
        tpl     x0,a    ; if so, a=w2max
        cmp     y0,a                    Y:(r6)+,x0
                        ; is a<w2min and x0=deltap
        tmi     y0,a    ; if so, a=w2min
        add     x0,a                    Y:(r6)+,x0 ; x0=c3=1/(4*pi)
        move            a1,x1
        mpy     x0,x1,a ; scale for correct s2 level
        add     y1,a                    Y:(r6)+,x0 ; x0=s2max=0.5
                        ; a=s2=s2c+scale*(deltap+s10)
                        ;     =s2o+scale*(deltap+delta)
        cmp     x0,a                    ;
        jmi     chk2sml2
        sub     x0,a    (r6)+
        sub     x0,a                    ; subtract 1 if >0.5
        jmp     donechk2
chk2sml2
        move                            Y:(r6)+,x1 ; x1=s2min=-0.5
        cmp     x1,a
        jpl     donechk2
        add     x0,a                    ; add 1 if < -0.5
        add     x0,a
donechk2
        move            a1,x0                   ;move s2 to x0 and
        move                    y:(r6)+,y0      ;load y0=scalesin
        mpyr    y1,y0,a x0,x:(r2)-              ;store s2
        move            a1,n6                   ; old s2o value! as we want
        move            #>sinloc,r6
        move            (r2)-
        move            y:(r6+n6),x1            ;x1=sin(s2o)
        asr   a                 b1,y1           ;y1=new input=W(kk) and a=a/2
       move   a1,n6
        mpy     x1,y1,a                         ;a=s3=sin(s2o)*W(new input)
        move    a1,x:(r2)+                      ;save s3=a
                                                ;x0 contains s2
```

```
updtdone

;       bset    #1,x:M_PDRE   ; signals end of PLL


        clr     a       y:(r6+n6),x0            ;x0=sin(s2o/2)
         move     x:count,a1                             ;x1=count (down from six)
         sub    #1,a                                    ;count=count-1
        move    a1,x:count
         jne      donerts

         ; If it is the 6th time around, then process
         ;    This acts like a downsampling
         ;Use PLL to create 2 KHz demodulation signal and
         ;  demodulate channel 1 and channel 2 filtered signals
         move     #$6,a0
         move     a0,x:count                             ; reset sixcnt to 6
         move     x:bpfiltx,x1                           ; load channel 1 new val
         mpy      x0,x1,a                                ; demodulate ch 1
        move    x:bpfilty,y1                    ; load ch2 val
         mpy      x0,y1,b          a1,x0                ; demodulate ch 2
         move                      b1,y0
                  ;We now have X3(kk) in x0 and Y3(kk) in y0. They are
                  ; now demodulated completely down from 10 KHz



        ;Filter demodulated signals for channel 1 and channel 2 to
         ;  remove  double frequency (4 KHz) component

         ;r1 must point to signal 2 buffer
         ;r4 must point to the filter coefficients
         ;m1 and m4 must be 15
         move #15,m1
         move m1,m4
        move #16,N1
         move     x:sig2strt,r1                  ; load pointer to signal buffer
         move     #>lowpass,r4                   ; load pointer to filter coeff's
         ; I guess the new value of signal 1 is in x0 and the new value
         ;   of signal 2 is in y0
         nop                         ;necessary for move function
         clr      a        x0,x:(r1)+       y:(r4)+,y1
         do #15,endflt3
         mac      x0,y1,a x:(r1)+,x0       y:(r4)+,y1
endflt3
         macr     x0,y1,a     (r1)+N1       ; do not decrement r1 as we use it for
next filter

         ;Low pass Filter Channel 2
        move    y0,x0
         clr      b        x0,x:(r1)+       y:(r4)+,y1
         do #15,endflt4
         mac      y1,x0,b x:(r1)+,x0       y:(r4)+,y1
endflt4
         macr     y1,x0,b (r1)- ;fix r1 in the circular buffer for next time
        move    (r1)-N1
         move     r1,x:sig2strt                  ; save pointer in buffer
                          ; a now contains channel 1 (X4(kk))
                          ; b now contains channel 2 (Y4(kk))
         move     a1,b0    ; b1 now contains channel 2
                          ; b0 now contains channel 1

       move    b0,x0
       move    b1,y0
;      move    y0,x:savey
;      move    x0,x:savex


        ;Use channel 1 and channel 2 demodulated filtered signals and
```

```
        ;  non-linear observer (PLL) to track position
        ;   Data is organized in X: as: st3o st1o st2o
        ;   Dats is organized in Y: as: k c deltap2 c4=1/(4*pi)
        ;                              st2max st2min bigpos scalesin2
       clr    a
       move   x:strength,a1
        move     #st3o,r2
        move     #k,r6
        move     #$06,x1                    ; x1=.0469
        cmp      x1,a     X:(r2)+,x0       Y:(r6)+,y0  ; is DC<0.0469?
                        ;x0=st3o and y0=k
        jmi      update4                    ; jump if it is

update3                  ; This update if signal strength is strong
        mpy      x0,y0,a X:(r2),x1         Y:(r6)+,y1
                         ; x1=s1o and y1=c
        add      x1,a    ; a=st1o+k*st3o=s1
        move             a1,X:(r2)+        ; store new st1
        mpy      x0,y1,a                   X:(r2),y1 ; load y1=st2o
        add      x1,a    Y:(r6)+,x0        ;a=st1o+c*st3o
                                  ; x0=deltap2
        add      x0,a                      Y:(r6)+,x0 ; x0=c4=1/(4*pi)
        move             a1,x1
        mpy      x0,x1,a ; scale for correct s2 level
        add      y1,a                      Y:(r6)+,x0 ; x0=st2max=0.25
                         ; a=st2=st2o+scale*(deltap2+st1o+c*st3o)
                         ;       =st2o+scale*(deltap2+delta2)
        cmp      x0,a                      ;
        jmi      chk2sml3
        sub      x0,a    (r6)+
        sub      x0,a                      ; subtract .5 if >0.25
        move     a1,x0                     ; move st2 to x0
        clr      a                         ; bigpos=bigpos+1
        move     y:(r6),a0
        inc      a
        move     a0,y:(r6)
       move   x0,a
        jmp      donechk3
chk2sml3
        move                               Y:(r6)+,x1 ; x1=st2min=-0.25
        cmp      x1,a
        jpl      donechk3
        add      x0,a                      ; add .5 if < -0.25
        add      x0,a
        move     a1,x0                     ; move st2 to x0
        clr      a                         ; bigpos=bigpos-1
        move     y:(r6),a0
        dec      a
        move     a0,y:(r6)
       move   x0,a
donechk3
       move     a1,x0
;      move   x0,y:(r0)+
       move                  (r6)+            ; point from bigpos to scalesin2
        move                      y:(r6)+,y0      ;load y0=scalesin2
        mpy    y1,y0,a x0,x:(r2)-               ;store st2
        move             a1,n6                  ; old st2o value! as we want
       move          a0,r1                ; r1 is fraction between 2 sin values
                                          ;  use it to interpolate!
       move   a0,x:temp55
        move             #>sinloc,r6            ; r6 points to sin table
        move             (r2)-
        move                     y:(r6+n6),x1    ;x1=sin(st2o)
       move x1,x:savex
         move        (r6)+                ; get next value in table, too
         move               y:(r6+n6),x0
         clr  a
         move r1,a0
         asl  #23,a,a                          ; convert to positive fraction
         move a1,y0                       ; y0 is fraction between two sin values
       move a1,x:temp33
         move #>$7FFFFF,a                 ; a=1.00 (just a little less)
```

```
        sub  y0,a                          ; a is now fraction down from 2nd sin
value
        move a1,y1
       move a1,x:temp44
        mpy  x0,y0,a                       ; do the interpolation
        macr x1,y1,a
        move a,r1                          ; x1=interpolated sin(st2o)
       move r1,x:savey


        move            #>coserr,r6              ;point to cosine table with
phase offset!
        move                    y:(r6+n6),x1     ;y1=cos(st2o)
;      move y1,x:savey
        move       (r6)+                   ; get next value in table, too
        move               y:(r6+n6),x0
        mpy  x0,y0,a                       ; do the interpolation
        macr x1,y1,a

       move r1,x1
       move a,y1
        move                    b1,y0            ;y0=new input=Y4(kk)
        move                    b0,x0            ;x0=new input=X4(kk)
        mpy     x0,x1,a                          ;a=sin(st2o)*X4(kk)
        macr    y0,y1,a                          ; +cos(st2o)*Y4(kk)
        move            a1,x:(r2)+               ;save s3=a
        jmp     updtdone3                        ;

update4                 ; This update if signal strength is weak
        clr     a       X:(r2),x1       Y:(r6)+,y1
                        ; x1=st1o and y1=c
        move    x1,a    ; a=st1o=st1 : don't update st1!
        move            a1,X:(r2)+      ; store new st1
        move            X:(r2),y1       ; y1=st2o
        move            Y:(r6)+,x0      ; x0=deltap2
        add     x0,a                    Y:(r6)+,x0 ;a=st1+deltap2
                                                   ;x0=c4=1/(4*pi)
        move            a1,x1
        mpy     x0,x1,a ; scale for correct st2 level
        add     y1,a                    Y:(r6)+,x0 ; x0=st2max=0.25
                        ; a=st2=st2o+scale*(deltap2+st1o)
                        ;       =st2o+scale*(deltap2+delta2)
        cmp     x0,a                    ;
        jmi     chk2sml4
        sub     x0,a    (r6)+
        sub     x0,a                    ; subtract 0.5 if >0.25
        move    a1,x0                   ; move st2 to x0
        clr     a                       ; bigpos=bigpos+1
        move    y:(r6),a0
        inc     a
        move    a0,y:(r6)
       move   x0,a
        jmp     donechk4
chk2sml4
        move                            Y:(r6)+,x1 ; x1=st2min=-0.25
        cmp     x1,a
        jpl     donechk4
        add     x0,a                    ; add 0.5 if < -0.25
        add     x0,a
        move    a1,x0                   ; move st2 to x0
        clr     a                       ; bigpos=bigpos-1
        move    y:(r6),a0
        dec     a
        move    a0,y:(r6)
       move   x0,a
donechk4
       move   a1,x0
       move   (r6)+
        move                    y:(r6)+,y0       ;load y0=scalesin2
        mpy    y1,y0,a x0,x:(r2)-               ;store s2
                                ; a=index into sin table
```

```
        move            a1,n6                   ; old s2o value! as we want
       move          a0,r1                 ; r1 is fraction between 2 sin values
                                           ;  use it to interpolate!
        move            #>sinloc,r6             ; look up sin table
        move            (r2)-
        move                        y:(r6+n6),x1    ;x1=sin(st2o)
        move        (r6)+                   ; get next value in table, too
        move                y:(r6+n6),x0
        clr  a
        move r1,a0
        asl  #23,a,a                            ; convert to positive fraction
        move a1,y0                          ; y0 is fraction between two sin values
        move #>$7FFFFF,a                    ; a=1.00 (just a little less)
        sub  y0,a                           ; a is now fraction down from 2nd sin
value
        move a1,y1
        mpy  x0,y0,a                        ; do the interpolation
        macr x1,y1,a
        move a,r1                           ; x1=interpolated sin(st2o)

        move            #>coserr,r6             ;point to cosine table with
offset error!
        move                        y:(r6+n6),x1    ;x1=cos(st2o)
        move        (r6)+                   ; get next value in table, too
        move                y:(r6+n6),x0
        mpy  x0,y0,a                        ; do the interpolation
        macr x1,y1,a

        move    a,y1
        move    r1,x1
       move                         b1,y0           ;y0=new input=Y4(kk)
       move                         b0,x0           ;x0=new input=X4(kk)
       mpy      x0,x1,a                            ;a=sin(st2o)*X4(kk)
       macr     y0,y1,a                            ; +cos(st2o)*Y4(kk)
       move     a1,x:(r2)+                         ;save s3=a
                                                   ;x0 contains s2
updtdone3
disprtn
       move   x:st2o,y0          ; st2o is position (mod 75 mm)

;      move   y0,x:savey
;      move   x0,x:savex
        m       #>150000,r1                 ; convert fraction to mm (.5=75mm)
       move   r1,x1
       mpy    x1,y0,b
       move   y:bigpos,y0
        move    #>75000,r1
       move   r1,x1
       mpy    y0,x1,a
       asl    #23,a,a
       add    a,b

       asl    #5,b,b
       move   b1,x0
        ;Bandpass Filter Channel 9
        ;r1 must point to signal 1 buffer
        ;r4 must point to the filter coefficients
        ;m1 and m4 must be 511
        move    x:sig3strt,r1          ; load pointer to signal buffer
        move    #>ave2,r4           ; load pointer to filter coeff's
        move #511,m1
        move m1,m4

        ; I guess the new value of signal 1 is in x0 and the new value
        nop                         ;necessary for move function
        clr     a       x0,x:(r1)+      y:(r4)+,y1
        do #511,endflt9
        mac     x0,y1,a x:(r1)+,x0      y:(r4)+,y1
endflt9
        macr     x0,y1,a   (r1)- ; decrement r1 to change start pos
         nop
        move    r1,x:sig3strt           ; save pointer in buffer
```

```
        move    a,b
        asr     #5,b,b

       rnd      b
        move    b1,x1
        move   x1,x:savepos

;*****************************************************************************
; Display Routine                                                            *
;      This routine takes a number in x1 between 0 and 8400 and displays it  *
; on the 7 Segment, 4 digit LED Display.                                     *
;*****************************************************************************
         clr     b                  ;assume number is in x1
         move    #strtpos,r2              ;initialize indexes
         move    #decode,r4
         move    #digits1,r7
        move    #digits0,r1
;         move     #>-001234,x1
         move    x1,b                      ;assume 4 digit integer is in x1
        asr     #24,b,b
        cmp     #0,b
        jmi     negsign
        move    #lett0,y0
        move    y0,y:(r2)+
        move    #lett1,y0
        move    y0,y:(r2)+
        jmp     donesign
negsign
        move    #lettmin0,y0
        move    y0,y:(r2)+
        move    #lettmin1,y0
        move    y0,y:(r2)+
        neg     b
        move    b0,x1
donesign
        move    #>10000,x0
        move    x1,b
        clr     a
cmp10k
        cmp     x0,b
        jmi     done10k
        add     #1,a
        sub     x0,b
        jmp     cmp10k
done10k
         move     a,n1                    ;get ready to look up digit
         move     a,n7                    ;get ready to look up digit
         move     x:(r1+n1),y0            ;look up digit
         move     y0,y:(r2)+              ;display digit
         move     x:(r7+n7),y0            ;look up digit
         move     y0,y:(r2)+              ;display digit
       move    b,n7
       clr    b
       move   n7,b0
       move    b0,x1


         move     y:(r4)+,x0              ;load in first constant to convert
         mpy      x0,x1,a
         move     a,n1                    ;get ready to look up digit
          move     a,n7
         move     y:(r4)+,x0
         move     x:(r1+n1),y0            ;look up digit
         move     y0,y:(r2)+              ;display digit
         move     x:(r7+n7),y0            ;look up digit
         move     y0,y:(r2)+              ;display digit

         move     #lettper0,y0            ; display decimal point
         move     y0,y:(r2)+
         move     #lettper1,y0
         move     y0,y:(r2)+
```

```
        move    a,y1
        mac     -x0,y1,b
        move    y:(r4)+,x0
        move    b0,x1
        mpy     x0,x1,a
        move    a,n1                    ;get ready to look up digit
        move    a,n7                    ;get ready to look up digit
        move    y:(r4)+,x0
        move    x:(r1+n1),y0            ;look up digit
        move    y0,y:(r2)+              ;display digit
        move    x:(r7+n7),y0            ;look up digit
        move    y0,y:(r2)+              ;display digit
        move    a,y1
        mac     -x0,y1,b
        move    y:(r4)+,x0
        move    b0,x1
        mpy     x0,x1,a
        move    a,n1                    ;get ready to look up digit
        move    a,n7                    ;get ready to look up digit
        move    y:(r4)+,x0
        move    x:(r1+n1),y0            ;look up digit
        move    y0,y:(r2)+              ;display digit
        move    x:(r7+n7),y0            ;look up digit
        move    y0,y:(r2)+              ;display digit
        move    a,y1
        mac     -x0,y1,b
        move    b0,n1                    ;get ready to look up digit
        move    b0,n7                   ;get ready to look up digit
        move    y:(r4)+,x0
        move    x:(r1+n1),y0            ;look up digit
        move    y0,y:(r2)+              ;display digit
        move    x:(r7+n7),y0            ;look up digit
        move    y0,y:(r2)+              ;display digit


donerts
       move x:wait1sec,b
       cmp #0,b
       jeq chkirq
       bclr #0,x:irqaset
       sub #1,b
       move b1,x:wait1sec
       jset #0,x:xmitstrt,xmitnext
       jmp donedone

chkirq
;      jmp irqdchk
;*************************************************************************
; These next routines handle output through the serial port to record
;   data
;*************************************************************************
       jclr #0,x:irqaset,donedone
       nop
       jset #0,x:xmitstrt,xmitnext
       nop
       bset   #0,x:xmitstrt

      move     #asciiout,r2             ;initialize indexes
      move     #strtpos,r4
      move     #>$0000FF,x0

       move   y:(r4)+,a
       and    x0,a   (r4)+
       move   a1,x:(r2)+

       move   y:(r4)+,a
       and    x0,a   (r4)+
       move   a1,x:(r2)+

       move   y:(r4)+,a
       and    x0,a   (r4)+
       move   a1,x:(r2)+
```

```
        move   y:(r4)+,a
        and    x0,a   (r4)+
        move   a1,x:(r2)+

        move   y:(r4)+,a
        and    x0,a   (r4)+
        move   a1,x:(r2)+

        move   y:(r4)+,a
        and    x0,a   (r4)+
        move   a1,x:(r2)+

        move   y:(r4)+,a
        and    x0,a   (r4)+
        move   a1,x:(r2)+

          move    #>asciispc,y0
          move    y0,x:(r2)+
         move    #>asciibeep,y0            ; sound beep
         move    y0,x:(r2)+
         move    #>asciicr,y0            ; display carriage return
         move    y0,x:(r2)+
         move    #>asciilf,y0            ; sound beep
         move    y0,x:(r2)+


        move #asciiout,r2
        move x:(r2)+,x0
        move x0,x:M_STXL                          ; transmit first byte
        move r2,x:asciicnt

        move #>48000,x0
        move x0,x:wait1sec
        jmp donedone
xmitnext
        jclr #1,x:M_SSR,donedone
        move x:asciicnt,r2
        move x:(r2)+,x0
        move x0,x:M_STXL
        move r2,x:asciicnt
        move r2,b
        cmp    #>asciiend,b
        jne donedone
xmitdone
        bclr #0,x:xmitstrt
        bclr #0,x:irqaset
donedone
;       bclr   #0,x:M_PDRE   ; signals end of processing
;       bclr   #1,x:M_PDRE
;       jclr #1,x:M_SSR,donedone
;       move   #>65,x0
;       move x0,x:M_STXL

         rts

         include 'adainit8.asm'
        include 'timinit6.asm'
;       include 'switches.asm'
        include 'sciinit2.asm'
echo
         end
```

A.9 strngth3.asm

```
;*****************************************************************************
;      Switch to other channel since strngth2.asm
;
;POSITION.ASM - Reads in 2 quadrature signals modulated to 10 kHz
;               Demodulates the signals with help from a PLL
;               Displays demodulated signal level for both sensors on LCD
;               Sends ASCII measurements out serial port when button is pushed
;*****************************************************************************
; Note: R5 is designated as the stack pointer for routines.
;*****************************************************************************
      nolist
      include 'ioequ.asm'
      include 'intequ.asm'
      include 'ada_equ.asm'
      include 'vectors5.asm'
      list
       opt cc
;*****************************************************************************
; 4/6/96 added in 2 input PLL for position detection
; 4/6/96 added in display of position
;---Buffer for talking to the CS4215

       org    x:0
RX_BUFF_BASE  equ     *
RX_data_1_2   ds      1       ;data time slot 1/2 for RX ISR
RX_data_3_4   ds      1       ;data time slot 3/4 for RX ISR
RX_data_5_6   ds      1       ;data time slot 5/6 for RX ISR
RX_data_7_8   ds      1       ;data time slot 7/8 for RX ISR

TX_BUFF_BASE  equ     *
TX_data_1_2   ds      1       ;data time slot 1/2 for TX ISR
TX_data_3_4   ds      1       ;data time slot 3/4 for TX ISR
TX_data_5_6   ds      1       ;data time slot 5/6 for TX ISR
TX_data_7_8   ds      1       ;data time slot 7/8 for TX ISR

RX_PTR          ds      1  ; Pointer for rx buffer
TX_PTR          ds      1  ; Pointer for tx buffer
tempr5        ds      1
dispptr             dc      DISPLIST
strength      dc      0
      org     x:$010
sigxy   bsc 10,$0         ; circular buffer for butterworth LPF
                         ;filter zero coefficients for butterworth LPF
                         ;organized as: b0 b1 b2 b3 b4 -a1 -a2 -a3 -a4
s3o     dc  0            ; states for high freq PLL
s1o     dc  0
s2o     dc  0
sixcnt  dc  6
sinarg  dc  sincoef
      org     x:$020
sincoef dc  0.5,-0.5,-0.9999999,-0.5,0.5,0.9999999 ;last one is supposed to be 1
st3o    dc  0            ; states for low freq PLL
st1o    dc  0
st2o    dc  0
sig1strt dc input1 ;start circular buffer at 0
sig2strt dc lpsig1 ;start circular buffer at lpsig1
sig3strt dc input3
count   dc  6
bpfiltx       dc  0
bpfilty dc  0
temp33  dc 0
temp44  dc 0
temp55  dc 0
clock
seconds         dc      0
tenseconds      dc      0
minutes         dc      0
tenminutes      dc      0
```

```
hours           dc      0
tenhours        dc      0
nothing         dc      0
irqaset dc  0       ; 0 if not pushed yet and 1 if it needs handling
irqdset dc  0       ; 0 if not pushed yet and 1 if it needs handling
xmitstrt dc 0       ; 0 if transmitting has not yet started
savex  dc  $400000
savey   dc  0
wait1sec     dc 0
      org    x:$0050
compclk         dc      10,6,10,6,10,10,0,0
savepos          dc 0

      org    x:$0060
digits0         dc
lett00,lett10,lett20,lett30,lett40,lett50,lett60,lett70,lett80,lett90
      org    x:$0070
digits1         dc
lett01,lett11,lett21,lett31,lett41,lett51,lett61,lett71,lett81,lett91

       org     x:$0080
input1  bsc 128,$0      ; circular buffer for input 1 filter
;input2  bsc 128,$0      ; circular buffer for input 2 filter

lpsig1  bsc 16,$0       ; circular buffer for demodulated input 1
lpsig2  bsc 16,$0       ; circular buffer for demodulated input 2
        org      x:$0200
asciiout dc 72,69,76,76,79,13,7,7,7,32,32
asciiend dc  32
        org     x:$0210
digits2      dc     48,49,50,51,52,53,54,55,56,57
asciicnt     dc     0
asciineg     equ    45
asciispc     equ    32
asciibeep    equ    7
asciilf      equ    10
asciicr      equ    13
asciiper     equ    46
        org     x:$0400
input3  bsc 512,$0      ; cirular buffer for output averaging


      org    y:0

DISPLEN         equ 63 ; should be ((*-DISPLIST)-1) but isn't

       include 'asm\butter.asm'   ; include butterworth filter coeff's
                                 ;filter zero coefficients for butterworth LPF
                                 ;organized as: b0 b1 b2 b3 b4 -a1 -a2 -a3 -a4
c1      dc 0.0021418412             ; coefficients for fast PLL
c2      dc 0.5235987756
w2max   dc 0.0490873852
w2min   dc -0.0490873852
deltap  dc 0.52359877560
c3      dc (1/(4*3.14159265359))
s2max   dc 0.5
s2min   dc -0.5
scalesin dc 512

;Coefficients for two input PLL (Non-linear Observer)
k       dc  .0007815           ; coefficients for fast PLL
c       dc  .2499
deltap2 dc  0
c4      dc (1/(4*3.14159265359))
st2max   dc 0.25
st2min   dc -0.25
;st2max   dc 0.5
;st2min   dc -0.5
bigpos  dc 0
scalesin2 dc 512
;scalesin2 dc 10
               ; THESE numbers are for the display routine
```

```
      org    y:$0020
decode  dc      0.0010001,500,0.0100002,50,0.1000002,5 ; These constants will be
used to
                ; convert binary to decimal. It is VERY important to note
that every
                ; other one of these number is a small delta above 10^(-k).
This
                ; prevents errors at borders of 1000's, 100's and 10's. (if
it were a
                ; little less, we would be low by 1 digit and have problems.
digits  dc      $88,$EB,$4C,$49,$2B,$19,$18,$CB,$08,$09

        org     y:$0080 ; next thing must start on a 128 byte boundary;

        org     y:$100
;       include     'asm\sintable.asm' ; location of sine table in memory
                            ; also location of cos table with
                            ; shift and wraparound

        org     y:$0200         ;definitely in internal memory
        include 'asm\bp2.asm' ; include bandpass filter coeff's
;        include 'asm\lowpass.asm'  ; include lowpass filter coefficients
      org    y:$0300
DISPLIST
                dc      lcdcommand,lcdpos800,lcdpos801,lcddata
                dc      lettt0,lettt1,letti0,letti1,lettm0,lettm1,lette0,lette1
                dc
lettcol0,lettcol1,lett0,lett1,lett00,lett01,lett00,lett01
                dc
lettcol0,lettcol1,lett00,lett01,lett00,lett01,lettcol0,lettcol1
                dc      lett00,lett01,lett00
endtime         dc      lett01
             dc     lcdcommand,lcdpos940,lcdpos941,lcddata
                dc      letts0,letts1,lettt0,lettt1,lettr0,lettr1
                dc      letta0,letta1,lettn0,lettn1,lettg0,lettg1
                dc      lettt0,lettt1,letth0,letth1,lettcol0,lettcol1
strtpos         dc      lett0,lett1
             dc     lett00,lett01,lett00,lett01,lett00,lett01
endpos          dc
lett00,lett01,lett00,lett01,lett0,lett1,lett0,lett1,(lettm0+32),(lettm1+32)
             dc     (lettm0+32),(lettm1+32)
             dc     lcdcommand,lcdposc00,lcdposc01,lcddata
;            dc     lettr0,lettr1,letta0,letta1,lettl0,lettl1
;            dc     letta0,letta1,lettt0,lettt1,letti0,letti1
;            dc     lettv0,lettv1,letta0,letta1,lett0,lett1
             dc     lettp0,lettp1,letto0,letto1,letts0,letts1
             dc     letti0,letti1,lettt0,lettt1,letti0,letti1
             dc     letto0,letto1,lettn0,lettn1,lett0,lett1
             dc     letts0,letts1,letta0,letta1,lettn0,lettn1
             dc     letts0,letts1,letti0,letti1,lettn0,lettn1
             dc     lettg0,lettg1,lett0,lett1,lettd0,lettd1,letta0,letta1
             dc     lettm0,lettm1,letto0,letto1,lettex0,lettex1
crappos
             bsc 134,lcdcommand
endcrap
         org     y:$0400
         include 'asm\ave2.asm' ; include averaging filter coefficients


TONE_OUTPUT     EQU     HEADPHONE_EN+LINEOUT_EN
TONE_INPUT      EQU
MIC_IN_SELECT+(15*MONITOR_ATTN)+(15*LEFT_GAIN)+(15*RIGHT_GAIN)
CTRL_WD_12    equ    NO_PREAMP+HI_PASS_FILT+SAMP_RATE_48+STEREO+DATA_16    ;CLB=0
CTRL_WD_34    equ    IMMED_3STATE+XTAL1_SELECT+BITS_64+CODEC_MASTER
CTRL_WD_56    equ    $000000
CTRL_WD_78    equ    $000000

;HPCR             equ     $FFFFC4
;HDDR             equ     $FFFFC8
;HDR              equ     $FFFFC9

        include 'dispequ.asm'
```

```
        org     p:$100
START
main
;        movep   #$040003,x:M_PCTL     ; set PLL for MPY of 4X
         movep   #$A40033,x:M_PCTL     ; set PLL for MPY of 52X and DIV by 11
                                       ; for actual speed of 80.053527 MHz
         movep   #$012421,x:M_BCR    ; set up one ext. wait state for all AAR
areas
;       movep  #3,x:M_PRRE
         ori     #3,mr             ;mask interrupts
         movec   #0,sp             ;clear hardware stack pointer
         move    #0,omr            ;operating mode 0
         move    #$300,r5           ; initialise stack pointer
         move    #-1,m5            ; linear addressing
        movep   #$000108,x:M_IPRP
;         move    #>31,m5           ; display buffer is 32 bytes long!
;         move    #>DISPLIST,r5     ; we will always write to this list buffer,
                                   ; and the receive ISR will take care of
                                   ; displaying the buffer.


        include 'dispinit.asm'
;       jmp    disprtn
         jsr     ada_init          ; initialize codec
;         jsr     time_init         ; initialize timer
        jsr    sci_init             ; initialize sci for output

        ; initialization of necessary registers
         move    #sigxy,r3
        move   #128,N1               ; to jump from left input or lowpass
;       move   #16,N5        ;   buffer to the right one
         ; initialize registers for filtering
         move    #127,m1           ; use 128 sample circular buffers
         move    m1,m4             ; use 128 sample circular buffers
         move    #9,m3
         move    #$80FF,m6         ; force it to stay in 256 entry table
         move    #5,N3
;         move    #buffer1,r0
;         move    #$00FF,m0

         movep #$000FFF,x:M_IPRC    ; enable IRQ interrupts, and set to level 2

loop_1

         jset    #2,x:M_SSISR0,*    ;wait for frame sync to pass
         jclr    #2,x:M_SSISR0,*    ;wait for frame sync

         move    x:RX_BUFF_BASE,b         ;receive left
         move    x:RX_BUFF_BASE+1,a       ;receive right
        asl    #3,a,a                ; scale up by 8!
        asl    #3,b,b
        move   a,x0
        move   b,y0
         jsr     process_stereo
        move   x:savex,a
        move   x:savey,b
         move    a,x:TX_BUFF_BASE         ;transmit left
         move    b,x:TX_BUFF_BASE+1       ;transmit right

         move    #TONE_OUTPUT,y0 ;set up control words
         move    y0,x:TX_BUFF_BASE+2
         move    #TONE_INPUT,y0
         move    y0,x:TX_BUFF_BASE+3

         jmp     loop_1

process_stereo

       move a1,x0
        ;Bandpass Filter Channel 1
```

```
        ;r1 must point to signal 1 buffer
        ;r4 must point to the filter coefficients
        ;m1 and m4 must be 127
        move    x:sig1strt,r1          ; load pointer to signal buffer
        move    #>bp2,r4            ; load pointer to filter coeff's
        move #127,m1
        move m1,m4
       move #128,N1

        ; I guess the new value of signal 1 is in x0 and the new value
        ;   of signal 2 is in y0
        nop                         ;necessary for move function
        clr     a       x0,x:(r1)+      y:(r4)+,y1
        do #127,endflt1
        mac     x0,y1,a x:(r1)+,x0      y:(r4)+,y1
endflt1
        macr    y1,x0,a (r1)- ;fix r1 in the circular buffer for next time
       nop
        move    r1,x:sig1strt           ; save pointer in buffer


         abs        a               ; measure the DC value for signal strength
                                    ; by taking the absolute value and low pass
filtering

        ;Filter to detect signal (DC component) use AC signal for signal
strength
        ;coeff b0 b1 b2 b3 b4 scaleb -a1 -a2 -a3 -a4
        ;storage locations are for x0 x1 x2 x3 x4 y1 y2 y3 y4
        ;m3 must be 8=9-1, N3 must be 5
        ; new value starts out in acc. a
        ; The b coefficients are scaled up by 16384 for numerical accuracy
        ; The a coefficients are scaled down by 8 since they are greater than 1
        ; The scaleb scales the input portion down by 16384*8 to match the
        ;    output feedback portion
        ; The total is then scaled up by 8 to give the correct amplitude
        ;  (this is done with 3 asl's)
        move    a1,x0
        tfr     a,b   #butterb,r7
        clr     a       x0,x:(r3)+      y:(r7)+,y0
        do #5,endfltb
        mac     x0,y0,a x:(r3)+,x0      y:(r7)+,y0
endfltb
        move            a1,x1
;        mpy     x1,y0,a x:(r3)+,x0      y:(r7)+,y0  ;???
                ; prev inst replaced with next two
       asr    #17,a,a
       move          x:(r3)+,x0    y:(r7)+,y0
        do #3,endflta
        mac     x0,y0,a x:(r3)+,x0      y:(r7)+,y0
endflta
       macr   x0,y0,a
        asl     #3,a,a
        move    (r3)-N3
        move    a1,x:(r3)-N3
        sub     a,b     (r3)-           ; B contains AC part of signal
                                        ; A contains DC signal

       move a1,x:strength


disprtn
       move   a1,y0         ; st2o is position (mod 75 mm)
       move   a,b
       move   b1,x0
        ;Bandpass Filter Channel 9
        ;r1 must point to signal 1 buffer
        ;r4 must point to the filter coefficients
        ;m1 and m4 must be 511
        move    x:sig3strt,r1          ; load pointer to signal buffer
        move    #>ave2,r4           ; load pointer to filter coeff's
        move #511,m1
```

```
        move m1,m4

        ; I guess the new value of signal 1 is in x0 and the new value
        nop                         ;necessary for move function
        clr     a       x0,x:(r1)+      y:(r4)+,y1
        do #511,endflt9
        mac     x0,y1,a x:(r1)+,x0      y:(r4)+,y1
endflt9
        macr     x0,y1,a    (r1)- ; decrement r1 to change start pos
         nop
        move    r1,x:sig3strt           ; save pointer in buffer
       move   a,b
       asr    #5,b,b
      rnd     b
       move   b1,x1
       move  x1,x:savepos

;*****************************************************************************
; Display Routine                                                            *
;     This routine takes a number in x1 between 0 and 8400 and displays it   *
; on the 7 Segment, 4 digit LED Display.                                     *
;*****************************************************************************
        clr     b                   ;assume number is in x1
        move    #strtpos,r2             ;initialize indexes
        move    #decode,r4
        move    #digits1,r7
       move   #digits0,r1
;        move    #>-001234,x1
        move    x1,b                    ;assume 4 digit integer is in x1
       asr    #24,b,b
       cmp    #0,b
       jmi    negsign
       move   #lett0,y0
       move   y0,y:(r2)+
       move   #lett1,y0
       move   y0,y:(r2)+
       jmp    donesign
negsign
       move   #lettmin0,y0
       move   y0,y:(r2)+
       move   #lettmin1,y0
       move   y0,y:(r2)+
       neg    b
       move   b0,x1
donesign
       move   #>10000,x0
       move   x1,b
       clr    a
cmp10k
       cmp    x0,b
       jmi    done10k
       add    #1,a
       sub    x0,b
       jmp    cmp10k
done10k
        move    a,n1                    ;get ready to look up digit
        move    a,n7                    ;get ready to look up digit
        move    x:(r1+n1),y0            ;look up digit
        move    y0,y:(r2)+              ;display digit
        move    x:(r7+n7),y0            ;look up digit
        move    y0,y:(r2)+              ;display digit
       move   b,n7
       clr    b
       move  n7,b0
       move   b0,x1


        move    y:(r4)+,x0              ;load in first constant to convert
        mpy     x0,x1,a
        move    a,n1                    ;get ready to look up digit
         move    a,n7
        move    y:(r4)+,x0
```

```
        move    x:(r1+n1),y0            ;look up digit
        move    y0,y:(r2)+              ;display digit
        move    x:(r7+n7),y0            ;look up digit
        move    y0,y:(r2)+              ;display digit

;        move    #lettper0,y0            ; display decimal point
;        move    y0,y:(r2)+
;        move    #lettper1,y0
;        move    y0,y:(r2)+

        move    a,y1
        mac     -x0,y1,b
        move    y:(r4)+,x0
        move    b0,x1
        mpy     x0,x1,a
        move    a,n1                    ;get ready to look up digit
        move    a,n7                    ;get ready to look up digit
        move    y:(r4)+,x0
        move    x:(r1+n1),y0            ;look up digit
        move    y0,y:(r2)+              ;display digit
        move    x:(r7+n7),y0            ;look up digit
        move    y0,y:(r2)+              ;display digit
        move    a,y1
        mac     -x0,y1,b
        move    y:(r4)+,x0
        move    b0,x1
        mpy     x0,x1,a
        move    a,n1                    ;get ready to look up digit
        move    a,n7                    ;get ready to look up digit
        move    y:(r4)+,x0
        move    x:(r1+n1),y0            ;look up digit
        move    y0,y:(r2)+              ;display digit
        move    x:(r7+n7),y0            ;look up digit
        move    y0,y:(r2)+              ;display digit
        move    a,y1
        mac     -x0,y1,b
        move    b0,n1                   ;get ready to look up digit
        move    b0,n7                   ;get ready to look up digit
        move    y:(r4)+,x0
        move    x:(r1+n1),y0            ;look up digit
        move    y0,y:(r2)+              ;display digit
        move    x:(r7+n7),y0            ;look up digit
        move    y0,y:(r2)+              ;display digit


donerts
       move x:wait1sec,b
       cmp #0,b
       jeq chkirq
       bclr #0,x:irqaset
       sub #1,b
       move b1,x:wait1sec
       jset #0,x:xmitstrt,xmitnext
       jmp donedone

chkirq
;      jmp irqdchk
;*************************************************************************
; These next routines handle output through the serial port to record
;   data
;*************************************************************************
       jclr #0,x:irqaset,donedone
       nop
       jset #0,x:xmitstrt,xmitnext
       nop
       bset   #0,x:xmitstrt

      move    #asciiout,r2              ;initialize indexes
      move    #strtpos,r4
      move    #>$0000FF,x0

       move   y:(r4)+,a
```

```
        and    x0,a   (r4)+
        move   a1,x:(r2)+

        move   y:(r4)+,a
        and    x0,a   (r4)+
        move   a1,x:(r2)+

        move   y:(r4)+,a
        and    x0,a   (r4)+
        move   a1,x:(r2)+

        move   y:(r4)+,a
        and    x0,a   (r4)+
        move   a1,x:(r2)+

        move   y:(r4)+,a
        and    x0,a   (r4)+
        move   a1,x:(r2)+

        move   y:(r4)+,a
        and    x0,a   (r4)+
        move   a1,x:(r2)+

        move   y:(r4)+,a
        and    x0,a   (r4)+
        move   a1,x:(r2)+

          move    #>asciispc,y0
          move    y0,x:(r2)+
         move    #>asciibeep,y0            ; sound beep
         move    y0,x:(r2)+
         move    #>asciicr,y0            ; display carriage return
         move    y0,x:(r2)+
         move    #>ascii1f,y0            ; sound beep
         move    y0,x:(r2)+


       move #asciiout,r2
       move x:(r2)+,x0
       move x0,x:M_STXL                          ; transmit first byte
       move r2,x:asciicnt

       move #>48000,x0
       move x0,x:wait1sec
       jmp donedone
xmitnext
       jclr #1,x:M_SSR,donedone
       move x:asciicnt,r2
       move x:(r2)+,x0
       move x0,x:M_STXL
       move r2,x:asciicnt
       move r2,b
       cmp    #>asciiend,b
       jne donedone
xmitdone
       bclr #0,x:xmitstrt
       bclr #0,x:irqaset
donedone
;      bclr   #0,x:M_PDRE   ; signals end of processing
;      bclr   #1,x:M_PDRE
;      jclr #1,x:M_SSR,donedone
;      move   #>65,x0
;      move x0,x:M_STXL

        rts

        include 'adainit8.asm'
       include 'timinit6.asm'
;      include 'switches.asm'
       include 'sciinit2.asm'
end
```

REFERENCES

[1] T. M. Clark, *Position Sensing and Control of a Linear Synchronous Motor*, Ph.D. Thesis, Massachusetts Institute of Technology, May, 1995.

[2] F. T. Barwell, *Automation and Control in Transport*, Pergamon Press, New York, 1983.

[3] E. Nishinaga, J.A. Evans, and G.L. Mayhew, Wireless Advanced Automatic Train Control, *IEEE Vehicular Technology Society News*, Vol. 41, No. 2, May 1994.

[4] U.S. Department of Transportation, *Development/Deployment Investigation of CABINTAXI/CABINLIFT SYSTEM*, Report No. UMTA-MA-06-0067-77-02, December, 1977.

[5] G. Uster, M. Heddebaut, G. Couvreur, P. Helle, A. Leostic, Operation and Evolution of the VAL in Lille, *Control, Computers, Communications in Transportation Systems: Proceedings of the 6th IFAC/IFIP/IFORS Conference* (J.-P. Perrin, ed.), Paris, France, 1989, pp. 167-176.

[6] G. W. McLean, Review of recent progress in linear motors, *IEE Proceedings*, Vol. 135, Pt. B, No. 6, November 1988, pp. 380-416.

[7] Walter Friesen, *SELTRAC Modern Train Control Technology*, presented at the Transportation Research Board, Alcatel Canada, January 1988.

[8] A. J. Pue, Implementation Trade-Offs for a Short-Headway Vehicle-Follower Automated Transit System, *IEEE Transactions on Vehicular Technology*, Vol. VT-28, No. 1, February, 1979.

[9] S. J. Sklar, J. P. Bevans, and G. Stein, "Safe-Approach" Vehicle-Follower Control, *IEEE Transactions on Vehicular Technology*, Vol. VT-28, No. 1, February, 1979.

[10] R. E. Fenton, A Headway Safety Policy for Automated Highway Operations, *IEEE Transactions on Vehicular Technology*, Vol. VT-28, No. 1, February, 1979.

[11] H. Strobel, Ed., *Computer Controlled Urban Transportation*, John Wiley & Sons, New York, 1982.

[12] J. P. Guilloux, Train Control on French Railroads, In *Transportation Research Record No 1314: Advanced Train Control Systems 1991*, Transportation Research Board, Washington, D.C., 1991. pp. 1-5.

[13] *Automatic Train Control in Rail Rapid Transit*, Office of Technology Assessment, Washington, D.C., May 1976.

[14] D. A. Poltorak and J. H. Bailey, Railroad Operation Using the Advanced Train Control System, In *Transportation Research Record No 1314: Advanced Train Control Systems 1991*, Transportation Research Board, Washington, D.C., 1991. pp. 96-101.

[15] D. A. Gary, Driven, Attended, and Fully Automated Transit: Qualitative Comparison, In *Transportation Research Record No 1221: Research in Bus and Rail Transit Operations*, Transportation Research Board, Washington, D.C., 1989, pp. .

[16] H. Bachmann, K. Niemitz, The Automatic Train Control and Safety Technology of the M-Bahn Public Transit System, *Control in Transportation Systems (1986): Proceedings of the 5th IFAC/IFIP/IFORS Conference* (J.-P. Perrin, ed.), Vienna, Austria, July, 1986, pp. 273-280.

[17] M. Mulazzani, Reliability and Safety in Electronic Interlocking, *Control in Transportation Systems (1986): Proceedings of the 5th IFAC/IFIP/IFORS Conference* (J.-P. Perrin, ed.), Vienna, Austria, July, 1986, pp. 321-328.

[18] R. D. Pascoe, C. Rossi, S. Savio, G. Sciutto, Comparison between Fixed and Moving Block Signaling Systems Performance by Digital Simulation, In *Computers in Railway Management: Proceedings of COMPRAIL 92 Conference in Washington D.C., USA* (T.K.S. Murthy, J. Allan, R. J. Hill, G. Sciutto, S. Sone, editors), Computational Mechanics Publications, Boston, 1992.

[19] Akio Seki, Junji Kesho, Masaki Katahira, New "COMTRAC" Computer-Aided Traffic Control System for the Tokaido-Sanyo Shinkansen, In *Computer Applications in Railway Operations: Proceedings of COMPRAIL 90 Conference in Rome, Italy* (T.K.S. Murthy, B. Mellitt, S. Lehmann, G. Astengo, editors), Computational Mechanics Publications, Boston, 1990.

[20] K. H. Hümmer, Operation Control and Signaling System for High Speed lines, In *Transportation Research Record No 1314: Advanced Train Control Systems 1991*, Transportation Research Board, Washington, D.C., 1991. pp. 114-121.

[21] X.-S. Wang, B. Ning, Z.-T. Ding, C.-K. Pu, Y.-H. Cheng, T. Tang, An Approach to a Moving Autoblock System in Railway-Computer-Aided Regulation of Traffic, In *Computers in Railway Management: Proceedings of COMPRAIL 92 Conference in Washington D.C., USA* (T.K.S. Murthy, J. Allan, R.J. Hill, G. Sciutto, S. Sone, editors), Computational Mechanics Publications, Boston, 1992.

[22] S. Shladover, *Operation of Automated Guideway Transit Vehicles in Dynamically Reconfigured Trains and Platoons*, U.S. Department of Transportation, 1979.

[23] Franklin, Powell, Emami-Naeini, *Feedback Control of Dynamic Systems*, Addison-Wesley, Reading, MA, 1991.

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

[25] E. A. Lee and D. G. Messerschmitt, *Digital Communication*, Kluwer Academic Publishers, Boston, 1988.

[26] R. E. Best, *Phase-Locked Loops: design, simulation, and applications*, New York, McGraw Hill, 1997.

[27] J. H. Irving, *Fundamentals of Personal Rapid Transit*, Lexington, MA, Lexington Books, 1978.

[28] E. Nishinaga and J. A. Evans, *Wireless Advanced Automatic Train Control*, IEEE Vehiclular Technology Society News, May, 1994.

[29] B. Knill, Flying Blind At Denver International Airport: Reality Checks Into Baggage Handling System, *Material Handling Engineering*, Volume 49, No. 8, August, 1994.

[30] J. G. Bender, *An overview of Systems Studies of Automated Highway Systems*, IEEE Transactions on Vehicular Technology, Vol. 40, No. 1, pp. 82-99, February, 1991.

[31] P. Heide, V. Magoir, R. Schwarte, Coded 24 GHz Doppler Radar Sensors: A New Approach to High-Precision Vehicle Position and Ground-Speed Sensing in Railway and Automobile Applications, *IEEE MIT-S International Microwave Symposium Digest*, Vol. 2, p. 965, 1995.

THESIS PROCESSING SLIP

FIXED FIELD: ill _____ name _____

index _____ biblio _____

▶ COPIES: (Archives) Aero Dewey (Eng) Hum

Lindgren Music Rotch Science

TITLE VARIES ▶☐ _____

_____

_____

NAME VARIES: ▶☐ _____

_____

IMPRINT (COPYRIGHT) _____

▶ COLLATION: 161 p

_____

▶ ADD. DEGREE: _____ ▶ DEPT.: _____

SUPERVISORS: _____

_____

_____ _____

_____ _____

_____ _____

_____ _____

_____ _____

_____ _____

_____ _____

NOTES:

cat'r: date:

page: J172

▶ DEPT: E.E.

▶ YEAR: 1997 ▶ DEGREE: Ph.D.

▶ NAME: PERREAULT, Brian Michael

_____