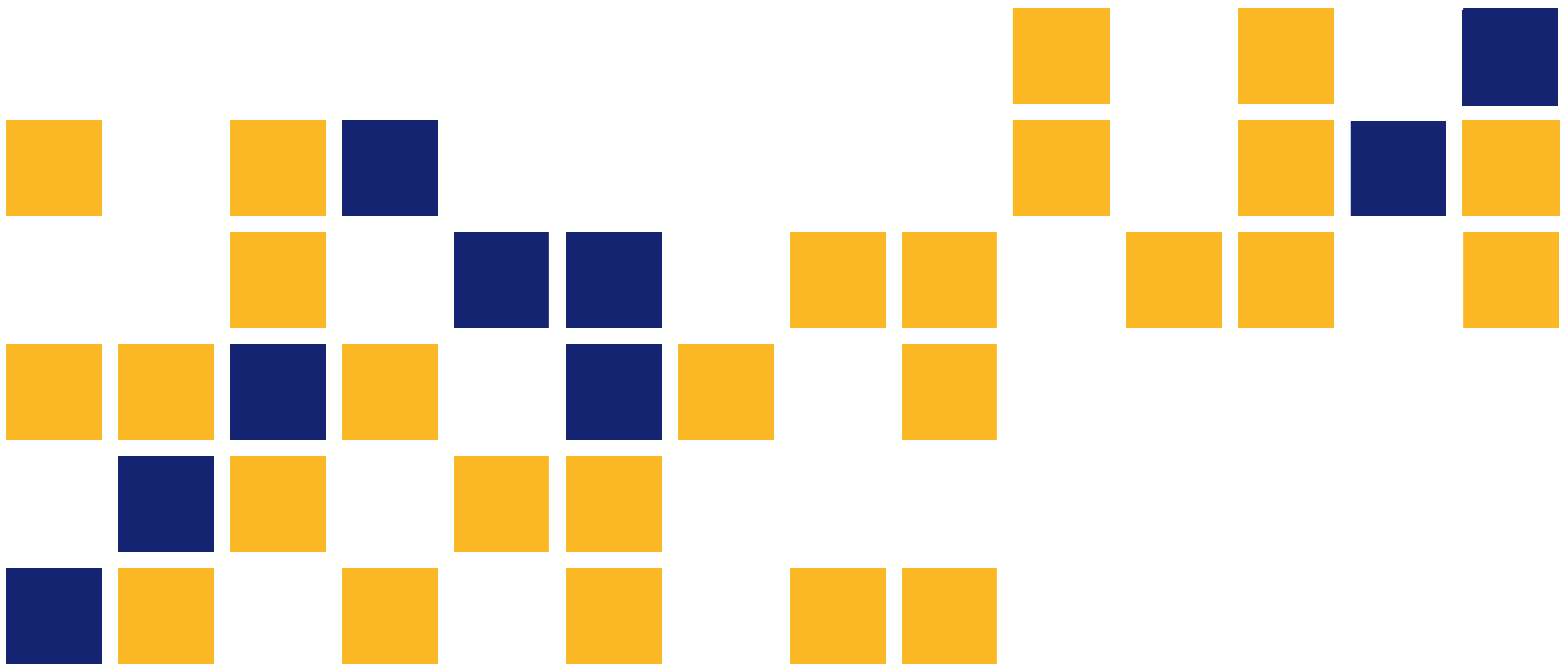


# Design of a 3-D Magnetic Mapping System to Locate Reinforcing Steel in Concrete Pavements

Nathanael W. Holle

*Kansas State University Transportation Center*





<b>1 Report No.</b> KS-17-06	<b>2 Government Accession No.</b>	<b>3 Recipient Catalog No.</b>	
<b>4 Title and Subtitle</b> Design of a 3-D Magnetic Mapping System to Locate Reinforcing Steel in Concrete Pavements		<b>5 Report Date</b> December 2017	
		<b>6 Performing Organization Code</b>	
<b>7 Author(s)</b> Nathanael W. Holle		<b>7 Performing Organization Report No.</b>	
<b>9 Performing Organization Name and Address</b> Kansas State University Transportation Center Department of Electrical and Computer Engineering 3108 Engineering Hall Manhattan, Kansas 66506		<b>10 Work Unit No. (TRAIS)</b>	
		<b>11 Contract or Grant No.</b>	
<b>12 Sponsoring Agency Name and Address</b> Kansas Department of Transportation Bureau of Research 2300 SW Van Buren Topeka, Kansas 66611-1195		<b>13 Type of Report and Period Covered</b> Final Report	
		<b>14 Sponsoring Agency Code</b>	
<b>15 Supplementary Notes</b> For more information write to address in block 9.			
<p>This report outlines the design, fabrication, and testing of a 3-D magnetic mapping system used to locate reinforcing steel in concrete pavements developed at Kansas State University (KSU) in 2006. The magnetic sensing functionality is based on the principles of magnetic tomography which use time-varying magnetic fields to induce magnetic returns from nearby ferrous objects. The purpose of this device is to provide a process for inspecting the depth and orientation of embedded steel bars. The device provides real-time feedback and detailed reports that can be archived and geospatially referenced.</p> <p>The mapping device extends the work previously done with versions that incorporated single sensors. Multi-sensor capability was added to enable determination of spatial orientation with a single data pass over a pavement joint. Additional reporting features such as GPS and in-field calibration techniques were used to streamline the data collection and report generation process.</p> <p>An embedded microprocessor communication interface between the peripheral sensing devices and the data collection computer was designed to offload some of the data compilation and manipulation from the laptop. This new interface alleviated speed issues encountered with the user interface programs running too slowly and allowed greater extensibility for adding more sensors or changing the platform architecture in the future.</p> <p>Verification and field testing was performed on all functional components of the system and the results from these tests are presented. The functionality of this device makes it attractive for commercial use by both construction companies and Departments of Transportation (DOTs) for inspection and archiving purposes. At the time of writing this report, the mapping device was at the stage of being prototyped and hardened for possible production.</p>			
<b>17 Key Words</b> Reinforcing Steel, Magnetic Tomography, Steel Placement		<b>18 Distribution Statement</b> No restrictions. This document is available to the public through the National Technical Information Service <a href="http://www.ntis.gov">www.ntis.gov</a> .	
<b>19 Security Classification (of this report)</b> Unclassified	<b>20 Security Classification (of this page)</b> Unclassified	<b>21 No. of pages</b> 167	<b>22 Price</b>

Form DOT F 1700.7 (8-72)

This page intentionally left blank.

# **Design of a 3-D Magnetic Mapping System to Locate Reinforcing Steel in Concrete Pavements**

Final Report

Prepared by

Nathanael W. Holle

Kansas State University Transportation Center

A Report on Research Sponsored by

THE KANSAS DEPARTMENT OF TRANSPORTATION  
TOPEKA, KANSAS

and

KANSAS STATE UNIVERSITY TRANSPORTATION CENTER  
MANHATTAN, KANSAS

December 2017

© Copyright 2017, **Kansas Department of Transportation**

## **NOTICE**

The authors and the state of Kansas do not endorse products or manufacturers. Trade and manufacturers names appear herein solely because they are considered essential to the object of this report.

This information is available in alternative accessible formats. To obtain an alternative format, contact the Office of Public Affairs, Kansas Department of Transportation, 700 SW Harrison, 2<sup>nd</sup> Floor – West Wing, Topeka, Kansas 66603-3745 or phone (785) 296-3585 (Voice) (TDD).

## **DISCLAIMER**

The contents of this report reflect the views of the authors who are responsible for the facts and accuracy of the data presented herein. The contents do not necessarily reflect the views or the policies of the state of Kansas. This report does not constitute a standard, specification or regulation.

## **Abstract**

This report outlines the design, fabrication, and testing of a 3-D magnetic mapping system used to locate reinforcing steel in concrete pavements developed at Kansas State University (KSU) in 2006. The magnetic sensing functionality is based on the principles of magnetic tomography which use time-varying magnetic fields to induce magnetic returns from nearby ferrous objects. The purpose of this device is to provide a process for inspecting the depth and orientation of embedded steel bars. The device provides real-time feedback and detailed reports that can be archived and geospatially referenced.

The mapping device extends the work previously done with versions that incorporated single sensors. Multi-sensor capability was added to enable determination of spatial orientation with a single data pass over a pavement joint. Additional reporting features such as GPS and in-field calibration techniques were used to streamline the data collection and report generation process.

An embedded microprocessor communication interface between the peripheral sensing devices and the data collection computer was designed to offload some of the data compilation and manipulation from the laptop. This new interface alleviated speed issues encountered with the user interface programs running too slowly and allowed greater extensibility for adding more sensors or changing the platform architecture in the future.

Verification and field testing was performed on all functional components of the system and the results from these tests are presented. The functionality of this device makes it attractive for commercial use by both construction companies and Departments of Transportation (DOTs) for inspection and archiving purposes. At the time of writing this report, the mapping device was at the stage of being prototyped and hardened for possible production.

## **Acknowledgements**

I want to express my gratitude toward Stanley E. Young for his insight and guidance throughout the project, especially at critical decision junctures. I also want to thank Professor James E. DeVault for his assistance, practical instruction, and freedom to pursue my own design avenues during this and other projects. I would like to thank the Kansas Department of Transportation, Koss Construction Co., and the American Concrete Pavement Association for their financial contributions that funded this research project.



# Table of Contents

Abstract .....	v
Acknowledgements .....	vi
Table of Contents .....	vii
List of Tables .....	x
List of Figures .....	xi
Chapter 1: Introduction .....	xiv
Kansas Common Concrete Construction Practices .....	1
Covermeter Applications and Operation .....	4
MIT-Scan .....	5
Features .....	6
Drawbacks .....	7
Chapter 2: Application Requirements .....	9
Sensing Requirements .....	9
Environmental Constraints .....	10
Ruggedness .....	10
User Interface Requirements .....	10
Usability .....	11
Reporting Requirements .....	11
Real-Time Plotting .....	11
Spatial Interpretation and Reports .....	12
Geospatially Linked Data .....	13
Chapter 3: Solution Architecture .....	15
Concept of Operation .....	15
Components .....	15
Kolelectric Covermeters .....	16
Optical Encoders .....	16
GPS .....	16
Microprocessor .....	16
Interface Architecture .....	17

Original Architecture vs. Microprocessor.....	17
Advantages of Incorporating a Microprocessor.....	18
Chapter 4: Hardware and Electronics Detailed Design .....	20
Mechanical Design .....	20
Non-Metallic Construction .....	20
Sensor Spacing.....	21
Electronics Enclosures and Connectors .....	23
Hardware Integration .....	24
Overview .....	24
Synchronized Covermeters .....	25
MPC555 .....	28
Power Distribution .....	30
Chapter 5: Software Architecture and Design .....	31
MPC555 .....	31
Concept of Operation .....	31
Detailed Design and Flow.....	31
Ruggedized Laptop .....	33
Concept of Operation .....	33
User Interface.....	35
Reporting Software .....	42
Chapter 6: Testing and Field Trials .....	48
Calibration .....	48
Performance Verifications .....	49
Sample Runs .....	52
Result Interpretation .....	66
Trace Plots .....	66
Histograms .....	71
Repeatability .....	73
Chapter 7: Future Work .....	80
Chapter 8: Conclusions .....	82
References .....	84

Appendix A: Kansas Concrete Construction Specifications .....	85
Appendix B: Proposed Kansas Concrete Construction Specifications .....	86
Appendix C: Embedded Source Files .....	89
Appendix D: Synchronized Covermeter Product Specification .....	128
Appendix E: Raw Data Files.....	138

## List of Tables

Table 4.1: Power Requirements by Component .....	30
Table 6.1: Depth Verification Test .....	51

## List of Figures

Figure 1.1:	Standard Reinforcement Steel Layout.....	2
Figure 1.2:	Transverse Joint Detail.....	3
Figure 1.3:	Longitudinal Joint Detail.....	3
Figure 1.4:	Magnetic Pulse Induction.....	5
Figure 1.5:	MIT-Scan Device .....	6
Figure 1.6:	MIT-Scan Typical Operation .....	7
Figure 1.7:	Movement of Two-Lane MIT-Scan Track.....	8
Figure 2.1:	Real-Time Plotting .....	12
Figure 2.2:	Google Earth Map with Linked Data Runs .....	14
Figure 3.1:	Architectural Block Diagram .....	15
Figure 3.2:	Initial Communication Interface Architecture .....	17
Figure 3.3:	Final Communication Interface Architecture.....	18
Figure 4.1:	Current Version of the Magnetic Sensing Cart .....	20
Figure 4.2:	Effects of Puck Spacing on Infinity Calibration Readings .....	22
Figure 4.3:	Polycarbonate Sensor Bar .....	23
Figure 4.4:	Polycarbonate Electronics Enclosure .....	23
Figure 4.5:	Electronics Block Diagram .....	25
Figure 4.6:	Synchronized Covermeters .....	26
Figure 4.7:	Covermeter Puck and Connectors .....	27
Figure 4.8:	X-Ray Image of Covermeter Puck .....	27
Figure 5.1:	MPC555 Main Control Loop .....	34
Figure 5.2:	Ruggedized Laptop Responsibilities.....	35
Figure 5.3:	Scooter Program .....	36
Figure 5.4:	Calibration Program .....	37
Figure 5.5:	Distance Calibration Program .....	38
Figure 5.6:	Zero Calibration Cart Orientation .....	39
Figure 5.7:	Data Acquisition Flowchart.....	41
Figure 5.8:	Report Generation Program .....	42
Figure 5.9:	Infinity Reading Stability .....	44

Figure 5.10: Calibration Curve Fitting .....	45
Figure 6.1: Calibration Jig .....	49
Figure 6.2: Verification Test Track .....	50
Figure 6.3: Condensed Plot of I-70 Testing Along Center Joint .....	52
Figure 6.4: Expanded Plot of I-70 Testing Along Center Joint (1/3) .....	53
Figure 6.5: Expanded Plot of I-70 Testing Along Center Joint (2/3) .....	54
Figure 6.6: Expanded Plot of I-70 Testing Along Center Joint (3/3) .....	55
Figure 6.7: Histogram of Left Sensor Bar Placement from I-70 Testing .....	56
Figure 6.8: Histogram of Right Sensor Bar Placement from I-70 Testing .....	56
Figure 6.9: Condensed Ideal Steel Placement Report.....	57
Figure 6.10: Expanded Ideal Steel Placement Report (1/3).....	58
Figure 6.11: Expanded Ideal Steel Placement Report (2/3).....	59
Figure 6.12: Expanded Ideal Steel Placement Report (3/3).....	60
Figure 6.13: Histogram of Ideal Steel Placement, Left Sensor.....	61
Figure 6.14: Histogram of Ideal Steel Placement, Right Sensor .....	61
Figure 6.15: Condensed Poor Steel Placement Report .....	62
Figure 6.16: Expanded Poor Steel Placement Report (1/2) .....	63
Figure 6.17: Expanded Poor Steel Placement Report (2/2) .....	64
Figure 6.18: Histogram of Poor Steel Placement, Left Sensor .....	65
Figure 6.19: Histogram of Poor Steel Placement, Right Sensor.....	65
Figure 6.20: Ideal Bar Placement Example Plot.....	67
Figure 6.21: Bar Rotated About an Axis Parallel to Longitudinal Joint Plot .....	68
Figure 6.22: Bar Rotated About an Axis Perpendicular to Concrete Surface Plot .....	69
Figure 6.23: Dowel Basket Interaction Along a Center Joint.....	70
Figure 6.24: Dowel Basket Interaction Along a Shoulder Joint .....	71
Figure 6.25: Example Histogram from Left Sensor .....	72
Figure 6.26: Example Histogram from Right Sensor .....	72
Figure 6.27: Repeatability Example of Condensed Plot of I-70 Center Joint .....	74
Figure 6.28: Repeatability Example of Expanded Plot of I-70 Center Joint (1/3) .....	75
Figure 6.29: Repeatability Example of Expanded Plot of I-70 Center Joint (2/3) .....	76
Figure 6.30: Repeatability Example of Expanded Plot of I-70 Center Joint (3/3) .....	77

Figure 6.31: Repeatability Example of Left Sensor Histogram.....	78
Figure 6.32: Repeatability Example of Right Sensor Histogram .....	78
Figure A.1: KDOT Standard Concrete Construction Specification 2006 .....	85
Figure B.1: KDOT Conceptual Dowel Bar Specification .....	87
Figure B.2: KDOT Conceptual Tie Bar Specification .....	88

This page intentionally left blank.



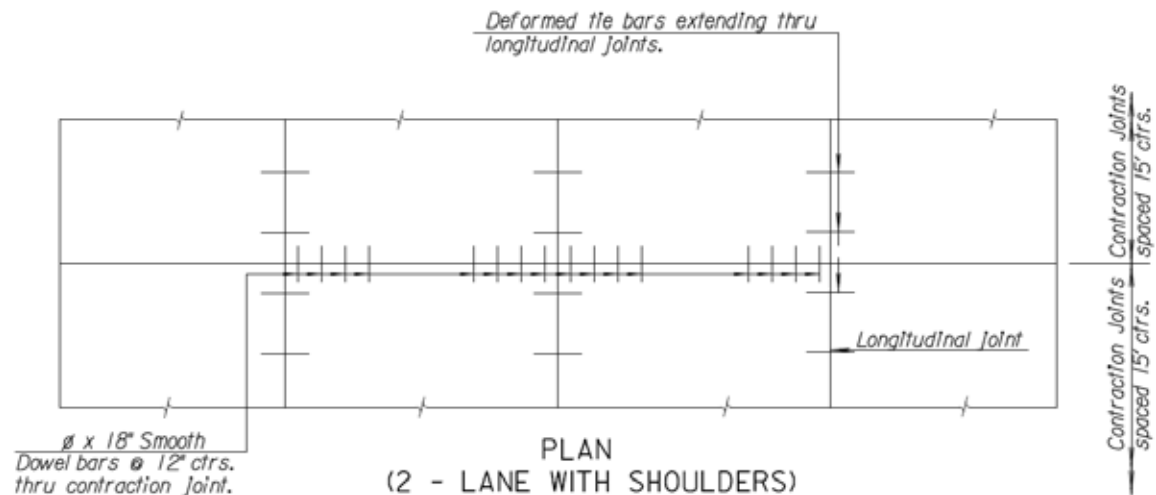
# **Chapter 1: Introduction**

This report outlines the design, fabrication, and testing of a 3-D magnetic mapping system used to locate reinforcing steel in concrete pavements developed at Kansas State University (KSU) in 2006. There is currently only one commercial product designed for locating steel reinforcement in concrete pavements and it is highly specialized for measuring the depth and orientation of dowel bars. These bars are used along transverse joints to tie slabs of concrete together for load transfer and thermal expansion and contraction. The motivation for this research is to develop a lower-cost device that can measure the location of both dowel bars and various sizes of steel rebar, typically referred to as tie bars, which are used for longitudinal joint control. The developed device incorporates multiple sensing units to enable taking multiple streams of data simultaneously. Steel reinforcement depth and orientation can be determined from these traces. The device incorporates GPS data into the data stream to provide permanent geo-referencing that can be easily transformed to common project coordinates such as project stationing.

The following introductory sections further describe the nature of the problem, the currently available commercial product capable of mapping steel reinforcement location, and the reasons why this product does not satisfy the needs of the project sponsors. A detailed description of the developed device follows which discusses the design of the electronics, the software, and the mechanical cart. The testing, verification, and interpretation procedures are then discussed to prove the functionality of the working system.

## **Kansas Common Concrete Construction Practices**

Steel in the form of dowel bars and tie bars is embedded in concrete pavements in Kansas according to the current specification, which can be found in Figure A.1 in Appendix A [1]. Two other proposed standards, found in Appendix B, have been written [2][3]. Their implementation is based on the availability of a suitable inspection device similar to the one discussed in this report. Figure 1.1 shows a top-down view of the standard layout of the reinforcement steel.



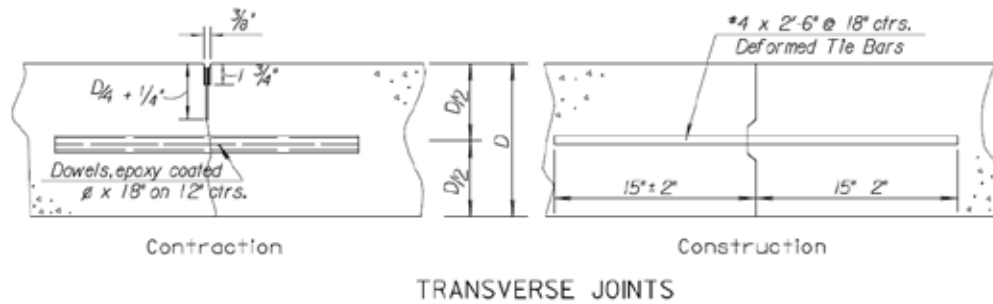
Reproduced from [1]

**Figure 1.1: Standard Reinforcement Steel Layout**

Dowel bars reinforce the transverse joints. They help provide load transfer between adjoining slabs of concrete and allow for thermal expansion and contraction of the concrete. Dowels are typically placed on 12 inch centers, are 18 inches long, and are coated in epoxy [1].

There are two procedures for placing the dowel bars. The first utilizes wire baskets that have loops that hold the bars in place while concrete is placed around them. This process was the most frequently encountered during data collection. The problem with this method is that the force exerted on the baskets during concrete placement can possibly move or deform the baskets, thus misaligning the dowel bars. Dowel bars along a transverse joint must be uniformly aligned if the joint is going to perform to specifications. The second system uses a mechanical inserter that plunges the bar into the wet concrete during construction. This procedure streamlines the construction process by eliminating the need to place dowel baskets before concrete placement. Mechanical problems with the inserter and dowel bar movement during consolidation in which the concrete paste is subjected to vibration are the primary sources of steel placement error in this process.

Figure 1.2 shows the steel placement in a cross-section of a standard concrete slab.



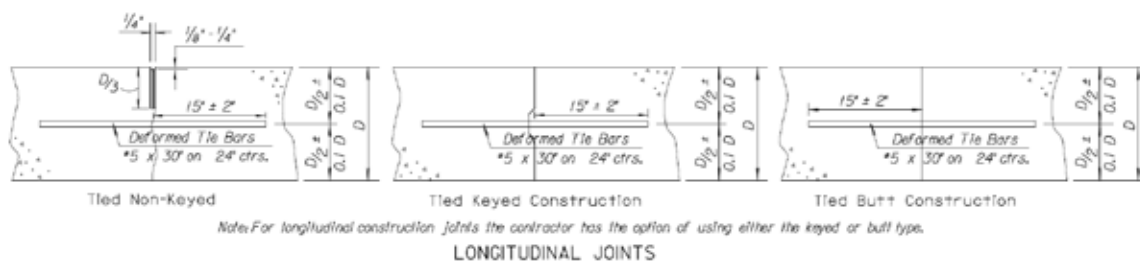
Reproduced from [1]

**Figure 1.2: Transverse Joint Detail**

The depth of the bar is defined by  $D/2$  which is half of the thickness of the slab [1]. The typical slab thickness observed during data collection was 12 inches, which dictates that 6 inches was the target bar depth. Partial-depth saw cuts over the dowel bars control the slab cracking and provide a center line for taking data over the dowel joints.

Longitudinal joints are typically reinforced with tie bars which are placed as shown in Figure 1.1. Tie bars provide a small amount of load transfer between adjacent slabs but are mainly used to lock the individual slabs in relative proximity to one another. Tie bars were commonly placed using a mechanical inserter on the tested construction projects. This process is similar to the dowel insertion process except the bars are oriented perpendicular to the dowel bars. Ties bars are also placed manually along open joint faces such as the keyed joints shown in the right diagram of Figure 1.2 and in the middle diagram of Figure 1.3.

Figure 1.3 shows some cross-section views of the steel placement in longitudinal joints.



Reproduced from [1]

**Figure 1.3: Longitudinal Joint Detail**

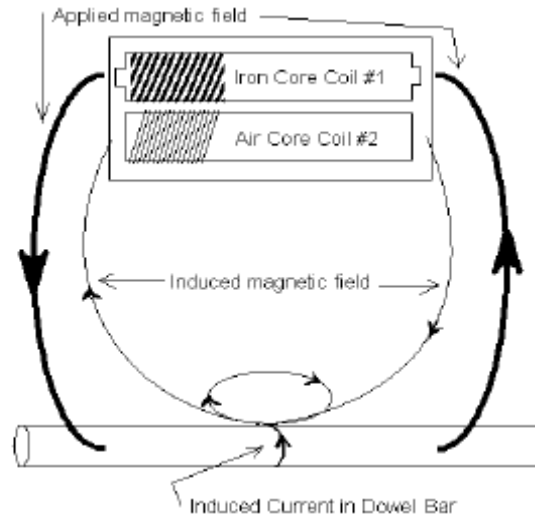
Thirty-inch #5 tie bars were most frequently encountered and were placed on 24 inch centers. They were inserted to a nominal mid-depth of 6 inches in 12-inch concrete slabs.

Placement errors of both dowel bars and tie bars typically result from a mechanical malfunction of the steel inserter. These mechanical failures result in steel being grossly misaligned and placed in a non-uniform fashion. Being able to detect these mechanical failures and other less common sources of error in real-time is the main focus of this research project. Other sources that contribute to misplacement or misalignment include the viscosity and composition of the concrete mix, the consolidation process and operation of the vibrator, and small inconsistencies in the inserter. These factors typically add small, random error to the placement process but can, on occasion, cause significant error.

## **Covermeter Applications and Operation**

Covermeters are devices that use the magnetic pulse induction methods to measure the amount of cover material (typically concrete) between a ferrous object and a sensor head. These devices have applications in all manners of construction that utilize steel, such as buildings, bridges, and roadways [4]. Currently, the Kansas Department of Transportation (KDOT) inspects pavement construction projects using handheld covermeters. An operator kneels on the roadway, manually locates the steel by waving the sensor head over the area of interest, takes a depth reading, and records it on a depth log.

Figure 1.4 shows an illustration of two-coil magnetic pulse induction.



Reproduced from [5]

**Figure 1.4: Magnetic Pulse Induction**

A large, time-varying magnetic field, sometimes called the send signal, is generated by a wire coil, called the send coil, wound around a metallic core [5]. Resulting eddy currents generate an opposite, but smaller magnetic field that is sensed by a second coil, called the receive coil. The received signal is then amplified and processed to yield a number that is proportional to the magnetic return. This measurement technique ignores the static magnetization of the bar and only measures the effects of the time-varying send signal. In this process, the diameter and composition of the bar affect the size of the magnetic return. Bars with larger diameters, such as dowel bars, will generate larger returns than a smaller diameter tie steel bar at the same distance.

## MIT-Scan

A commercially available device for mapping steel placement is the MIT-Scan. It was developed by a German company called Magnetic Imaging Tools and designed specifically measuring dowel bar location and orientation [6]. Figure 1.5 shows a view of the MIT-Scan device.



Reproduced from [7]

**Figure 1.5: MIT-Scan Device**

The green enclosure houses the magnetic sensing elements, support circuitry, and lead-acid battery for powering the electronics. A small handheld computer takes the data generated by the electronics and parses it into depth and orientation information for each bar. Data is stored on a compact flash card that can then be taken to a laptop or personal computer for plotting and additional data manipulation.

### *Features*

MIT-Scan is capable of taking five traces of data per run. This allows for a complete representation of the steel depth and orientation to be generated from a single run of the device over a joint. The handheld computer that collects and analyzes the data generates a print out that details each bar's location, depth, and orientation in terms of shifts and rotations from the expected values. The data is stored on a solid-state memory card which can then be taken to a computer for further analysis. The MIT-Scan software for the computer is capable of generating color intensity plots for each joint. These plots are highly effective at conveying the steel placement beneath the pavement. MIT-Scan is capable of measuring both dowel bars and tie steel, but the measuring process is specialized for measuring the dowel bars along the transverse joints which is discussed further in the next section.

## *Drawbacks*

Several drawbacks of the MIT-Scan device became apparent after field use of the system. The first problem was the weight of the device which was approximately 45 pounds. The size of the enclosure also made the device unwieldy when picking it up and moving it to the next transverse joint. The device had to be set down with the wheels lined up on the track rails before each run. Figure 1.6 shows the track and the typical operation of the MIT-Scan device.



**Figure 1.6: MIT-Scan Typical Operation**

The typical operating process dictated that the operator be bent over while holding and aligning the device with the track which was not suited to large numbers of runs due to the effort involved. Runs were time limited to 60 seconds and distance limited to the length of the track or the distance between perpendicular joints, whichever was shorter. Dowel bars and baskets at the transverse joints cause abnormalities in the data that the software cannot account for and causes the program to generate erroneous results which required manual editing. The track system allowed for relatively easy centering of the instrument over the joint, but it had to be moved and

realigned before every run, which limited the data collection speed. Figure 1.7 shows the process of moving the track configured for measuring the steel placement of two lanes.



**Figure 1.7: Movement of Two-Lane MIT-Scan Track**

Streamlined operation of the device configured for two-lane data collection required three people, two to move the track and one to carry the sensing device. Single person operation is possible but is less efficient than with two or more people. The MIT-Scan is customized for transverse joints. Technically, it can be used to scan longitudinal joints to determine placement accuracy of tie-steel, but the need to constantly reposition the track makes it impractical for all but very small spot samples.



## **Chapter 2: Application Requirements**

Initial instrument design was based on a set of requirements outlined by the needs of the research sponsors. Sensing, ruggedness, usability, and reporting requirements were formed to direct the design of the device and to make it feasible to use in the field. These requirements will be discussed in this chapter.

### **Sensing Requirements**

The purpose of the magnetic mapping device is to detect tie-steel embedded underneath the longitudinal joints and dowel bars under the transverse joints in concrete pavements. Tie steel was the primary target for detection so the device needed to be capable of generating valid depth results for the various reinforcement bar sizes. The prototype device was mainly tested on #5 reinforcement bar since that bar size was the most abundant in active construction projects during product design and testing, covering approximately 2003 to 2006. As a byproduct, steel dowels, which are typically found under the transverse saw cuts in the concrete, are also detectable since they have a larger diameter and generate a larger magnetic response for equivalent depths. The device was required to have a maximum sensing depth of at least 12 inches because a standard slab of concrete is 12 inches thick. This ensured that reinforcement steel that had been pushed to the bottom of the slab could be detected and not incorrectly assumed to be missing. The device also needed to be able to detect steel at the upper surface of the concrete. This constraint may seem unnecessary, but concrete pavements were observed during the project that had steel at the surface of the concrete or slightly protruding from the surface. The ability to map close proximity steel and steel at the extents of the sensing range was necessary to create a device that can accurately map the condition of all embedded steel. The final sensing constraint was that the device be able to take simultaneous measurements. This would provide a means for determination of bar orientation which will be discussed later in further detail.

## **Environmental Constraints**

The magnetic sensing cart needed to be operational in outdoor locations, mainly on construction sites. Outdoor testing of various prototypes throughout the project revealed several environmental issues. The following sections discuss some of the problems encountered during the prototyping of the magnetic cart.

### *Ruggedness*

Because the magnetic mapping device was designed to work outdoors, factors such as wind, dust, debris, and water had to be taken into account. Polycarbonate enclosures were fabricated to house all the electronics. This provided protection from the dust and other debris blown by the wind. The enclosures also provide some protection from rain showers, although they are not water tight. All these measures make the cart safe to use in most conditions where construction would be taking place. The cart was designed to travel over road debris up to half an inch tall. Earlier versions of the cart could not handle road debris, so a broom was used to completely clear the data collection path of unwanted material.

The most exposed piece of electronics is the laptop that rides on top of the polycarbonate chassis. The Panasonic Toughbook laptop provides a fully ruggedized platform to the design of the device and is tested to the military specification MIL-STD-810F [9]. The computer is resistant to water, physical shocks, and dust and other debris associated with concrete construction sites. Having a laptop onboard the device allows the control of the programs and visualization of the results to be minimally difficult for the user and allows for simpler and more flexible software design since many software tools exist for the Windows operating system.

### *User Interface Requirements*

Prototype testing revealed several issues with the usability of the device that had to be addressed. One of the greatest of these issues was being able to see the user interface on the laptop screen. Initial designs incorporated a standard IBM laptop for data collection, control of the system, and report generation. Problems with screen visibility in outdoor conditions and especially in direct sunlight situations necessitated the use of a higher brightness computer screen. This was found in the form of a Panasonic Toughbook with an ultra-bright LCD. The

higher brightness screen and high contrast settings made viewing the user interface possible in direct sunlight situations and improved viewing for all outdoor conditions.

### *Usability*

The physical design of the device incorporates several features that make usage as simple and as quick as possible. The height of the handle is adjustable so that operators can comfortably operate the device at a level that is appropriate for them. The handle can also be disconnected from the rest of the device which makes stowing the device in the back of a minivan possible. The weight of the device is low enough that two individuals can easily load and unload it.

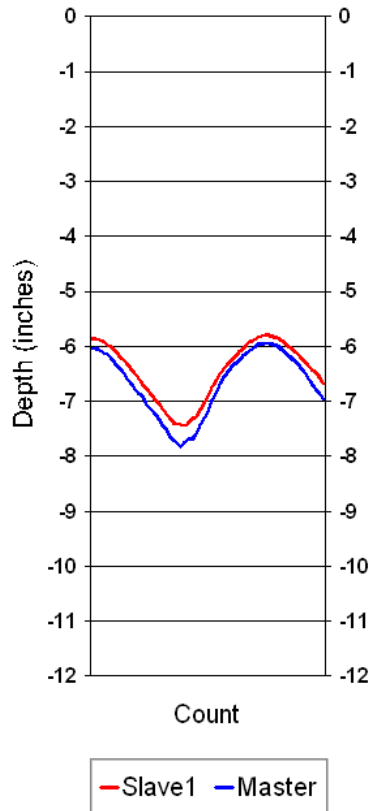
The sensor bar at the front of the current device is detachable. This allows several different sensor configurations to interface with a common electronics and reporting platform. The sensor bar can be tailored to the application. A shorter sensor bar would be used for measuring the depth of the dowel bars placed along the transverse joints in the roadway and a longer bar could be used for the tie steel found along the longitudinal joint in the pavement. This ability to swap sensor bars also allows two or three sensor varieties of the device to be used with one platform. Eventually more sensors may be possible, but the current generation of the electronics limits the number of sensors to three.

## **Reporting Requirements**

Part of the work done with this research project involved interfacing with research sponsors and potential consumers of the device to find out what reporting features would make it appealing for their use. Three main forms of feedback to the user were discussed and determined to be necessary for the product to be viable for field application. They were real-time plotting, reports that reflected 3-D alignment, and geospatially linked data.

### *Real-Time Plotting*

One implemented feature of the reporting software that was well received by the research sponsors was the ability to see depth plots in real-time. The real-time plots resemble those generated by an echocardiogram. Figure 2.1 shows a screenshot of the real-time plotting portion of the user interface software.



**Figure 2.1: Real-Time Plotting**

The peak in the traces corresponds to the depth of the bars underneath the sensor bar at the front of the device. This example plot shows the program picking up two embedded steel bars both at a depth of 6 inches. The first peak is at the left edge of the plot and the second is closer to the right edge. This plotting routine is the main feedback to the user about the status of the steel placement during data collection. The real-time plotting is the fastest way to discern whether reinforcement steel is placed correctly or is grossly misaligned.

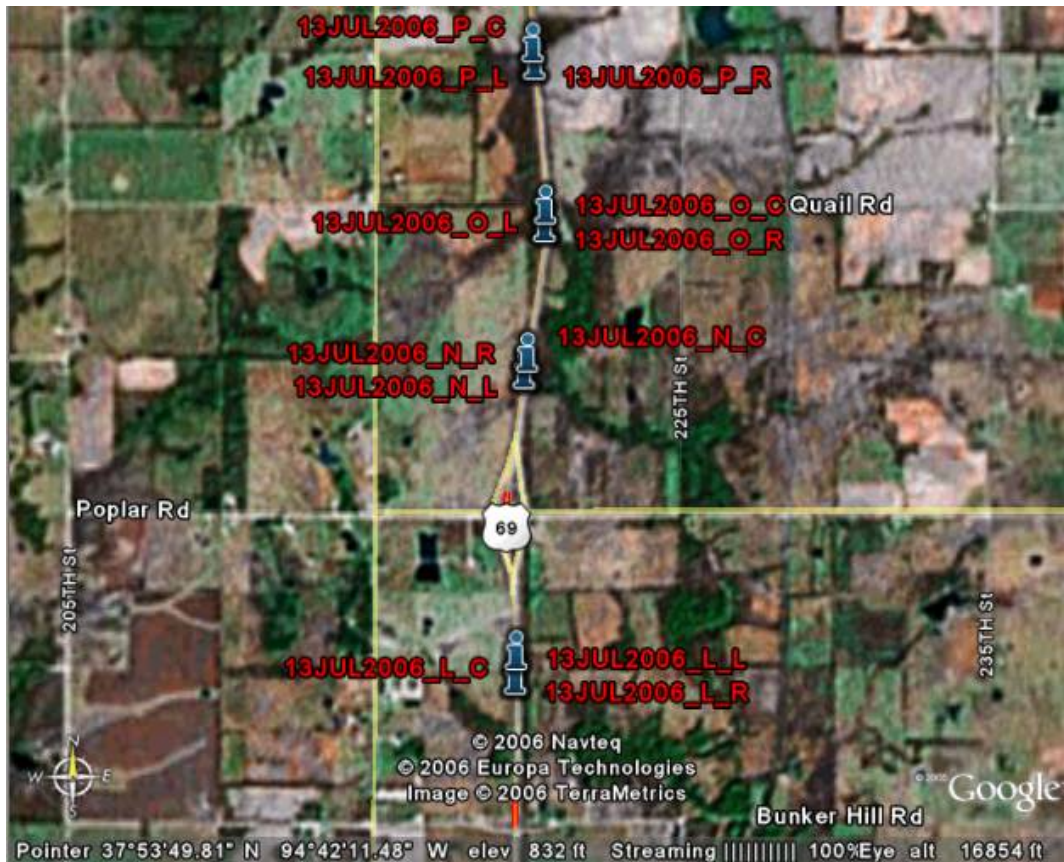
### *Spatial Interpretation and Reports*

The reports generated by the mapping device portray the results of an entire data run, usually a few hundred feet in length. These files give specific information about the placement of the embedded steel. Steel depth is plotted against the distance traveled in order to ascertain the 3-D orientation of the steel bars. A peak finding algorithm marks the most likely longitudinal location of each tie bar. Placement and depth accuracy and consistency can then be measured

using summary statistics. The report files can be archived for future reference or analysis. This provides a simple way for contractors to verify and document that all specifications and tolerances have been met.

### *Geospatially Linked Data*

Geodetic coordinate data obtained from an integrated GPS data logger allows for steel reinforcement placement data to be easily geo-referenced. A user can quickly locate an occurrence of poorly placed steel and travel to that location if further testing or examination is necessary. This provides a simple solution to the problem of how to store and easily access measured data. Traces are generated and plotted on the map by information stored in files that are compliant with KML, a customized XML developed for Google Maps and Google Earth. Clicking on a trace brings up an option to open the report associated with that location. Figure 2.2 shows a screenshot of a map of a segment of U.S. Highway 69 overlaid with traces and labels associated with data collection runs.



**Figure 2.2: Google Earth Map with Linked Data Runs**

Each trace is hyperlinked to an .htm file that is generated during the report generation process. Each linked report holds visual representations of the layout of the steel beneath the surface of the concrete for the marked geographical location. This feature also allows easy data referencing since it can be attached to a specific highway or project. Although the georeferencing of satellite imagery provided by Google Earth is sometimes suspect, particularly for rural areas, the usefulness of the embedded geodetic coordinates is not diminished. Road network layers can be turned on within Google Earth that clearly indicate highway route numbers or street names. Project stationing can only be cross-referenced to plan drawings. Geodetic coordinates provide a time-stable reference and increase ease of data collection. They can easily be cross-referenced to project coordinates as well.

## Chapter 3: Solution Architecture

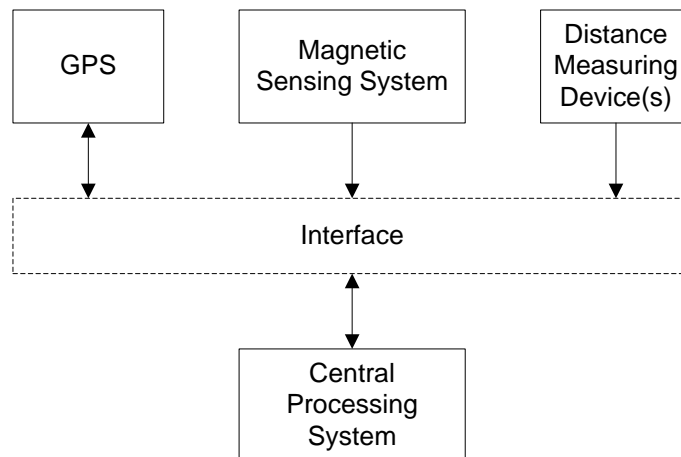
The architecture of the steel mapping device was designed to provide future extensibility and platform changes without altering the device's structure. The following sections cover the hierarchical design of the steel mapping device.

### Concept of Operation

Communication is central to the operation of the mapping device. Half and full-duplex serial communication connections are the primary forms of information exchange between the different functional components. The following sections outline the specifics of the device architecture.

### Components

The device's components can be divided into three categories. They are the peripheral transducers, the interface electronics, and the central processing system. Figure 3.1 shows the architectural block diagram for the steel mapping device.



**Figure 3.1: Architectural Block Diagram**

The peripheral transducers include the GPS, magnetic sensing system, and the distance measuring devices. The central processing system, in the current design, is a laptop that acquires

and saves the data, provides user control of the mapping device, and generates reports. The interface electronics are responsible for the communication between the previous two groups.

### *Kolectric Covermeters*

Customized Kolectric covermeters form the basis for the magnetic sensing system. These proprietary electronic devices work using the principles of magnetic tomography and are capable of generating raw magnetic return values via serial communication to the data acquisition device. The current set of covermeters can produce three streams of data concurrently which enables steel orientation to be determined with only a single pass over a pavement joint. Adding additional sensors to the current configuration is possible but would require consultation with Kolectric to ensure power and signal strength limits are not exceeded.

### *Optical Encoders*

The chosen distance measurement devices are 128 pulse-per-revolution optical encoders mounted on each rear drive wheel. These encoders measure the distance traveled during data collection runs and enable the bar placement spacing to be referenced from the beginning of data collection. These devices generate pulses that are counted by an acquisition device to keep track of the distance traveled.

### *GPS*

Geodetic coordinates were collected using a Garmin OEM GPS receiver. This device provides standard NMEA messages once per second over a variable speed serial channel. Output sentences can be enabled and disabled via prescribed serial command sentences. This receiver can also be programmed to output spatial coordinates at prescribed time intervals [10]. An appropriate active, external GPS antenna was selected and used in conjunction with the receiver.

### *Microprocessor*

The final design of the interface electronics incorporated a microprocessor to handle all communications between the peripheral devices and the central processing system. The Freescale MPC555 was selected based on familiarity, availability of tools, and functionality. The MPC555



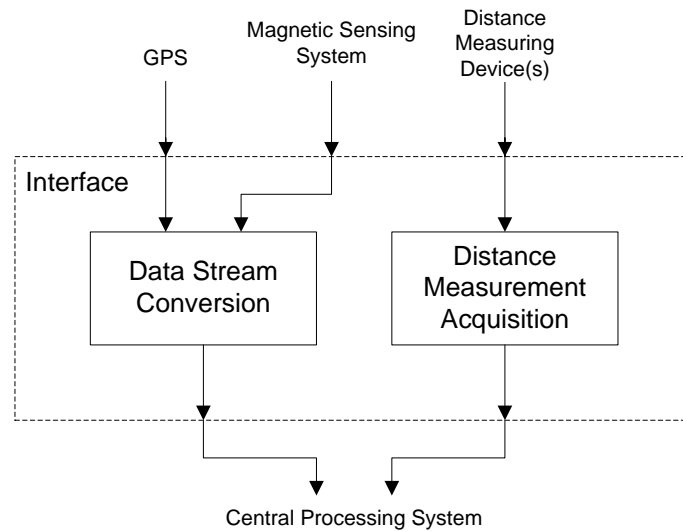
contains sub-processors that contain built-in serial communication functionality. All communication between the central processing system and the peripheral devices, except the encoders, used serial links.

## Interface Architecture

One facet of the project that went through multiple iterations before a final solution was developed involved the problem of communication between the various peripheral devices and the main processing system, the laptop. The following sections describe the design considerations for the communications interface.

### *Original Architecture vs. Microprocessor*

Time constraints and pressure to have a working device dictated that initial configurations of the mapping device be composed of off-the-shelf components. Figure 3.2 shows a block diagram of the initial communication interface.

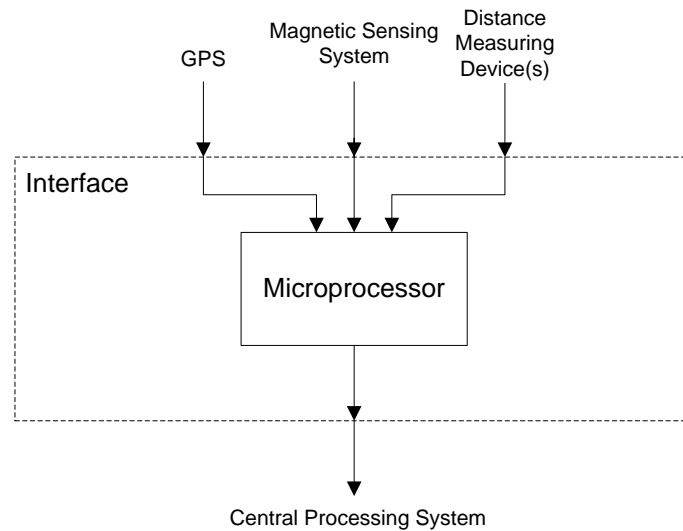


**Figure 3.2: Initial Communication Interface Architecture**

The two functional blocks were commercially available devices used to perform two very specific tasks. The distance measurement acquisition block was capable of reading the distance data from the distance measuring devices, packaging the data into a format that the central

processing system could understand, and communicating the distance data when requested by the central processing system. The data stream conversion block was capable of condensing the communication streams from four peripheral devices into a single communication link. This arrangement pushed the communication abilities of the central processing system and limited the processing time it had to complete its other tasks. This led to an interface redesign.

The redesigned interface incorporated a microprocessor that replaced the commercial devices mentioned previously. The block diagram in Figure 3.3 shows the final architecture of the communication interface.



**Figure 3.3: Final Communication Interface Architecture**

The number of functional components was reduced and the microprocessor was capable of combining the data received from all peripheral devices, pre-processing that data, and formatting it such that it reduced the load on the central processing system's communications. This change allows for the development of more reporting routines and greater instrumentation functionality within the same architecture.

#### *Advantages of Incorporating a Microprocessor*

Initial prototypes of the device utilized a quad RS232 to USB converter that piped data into a single USB line that could be connected to the laptop. The converter appeared as four

additional RS232 connection ports in the computer's device manager. These COM ports were utilized by the Visual Basic programs for retrieving the data from the synchronized covermeters and the GPS unit. This setup dictated that the laptop had to do a lot of data processing and handling of multiple serial channels which, in addition to the other real-time features of the program, caused the execution of the program to become sluggish. A solution for offloading some of the processing from the laptop was necessary for fast execution as well as the potential for adding more sensors in the future.

The final solution was to use a microprocessor to handle all of the serial communication from the covermeters and GPS. The main problems with the original prototype architecture were the multiple serial interfaces with the laptop and the need to install specialized drivers to utilize the commercial acquisition devices. Incorporating a microcontroller into the instrumentation consolidated all the communication into a single, simple serial stream. The MPC555 also has built-in functionality for fast quadrature decoding to handle the optical encoders that measured the distance traveled by the device. With this capability, the microprocessor consolidates the encoder and serial data to a single serial data link that interfaces with the laptop. This system eliminated two peripheral devices and their associated software drivers. The new architecture is more flexible for future scaling of the system. Eventually, this single link could be replaced by wireless transceivers so that the data collection and visualization hardware, the laptop in this case, would not have to be within cabling distance of the peripheral hardware. This would be advantageous if the platform of the current device is modified to something more suitable to being pulled behind a paving machine.

## Chapter 4: Hardware and Electronics Detailed Design

The following sections present the details of the mechanical and electronic design for the steel mapping device.

### Mechanical Design

The design of the mobile platform for the magnetic mapping device went through several iterations. Each new version added different functionality and incorporated lessons learned from previous prototypes. The following sections highlight some of the pertinent steps and final decisions made in the design process.

#### *Non-Metallic Construction*

Preliminary testing of the covermeter sensitivity showed that any steel further than 14 inches from the pucks yielded results that were within the thresholds of the ambient noise of the meters. This formed the basis for the rule of keeping any metal in the cart design further than 18 inches from the pucks at the front of the cart. Figure 4.1 shows the current design of the cart.

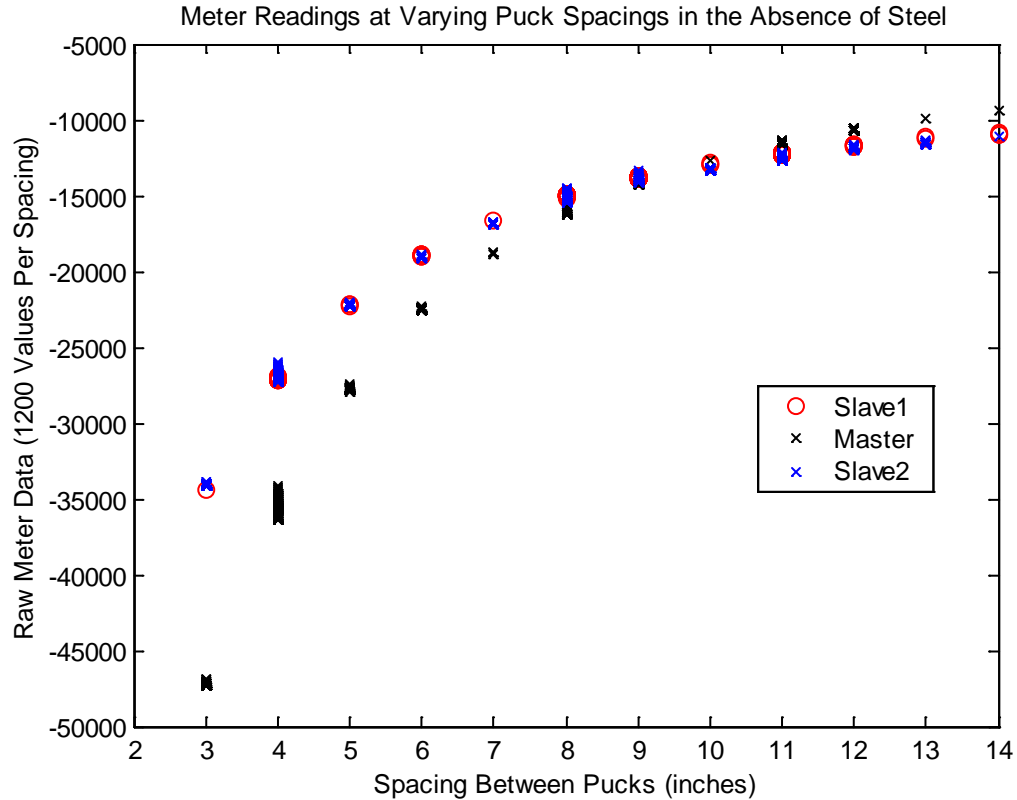


**Figure 4.1: Current Version of the Magnetic Sensing Cart**

The entire cart was composed of non-metallic materials, to ensure that nothing would interfere with the sensor readings. Plywood forms the base of the cart and the support structures for housing the electronics, supporting the handle, and mounting the rear wheel axles are created from formed sheets of polycarbonate. The handle is composed of PVC and large diameter wood dowel rods. The wheels are plastic with a rubber tire and are hooked to the polycarbonate portion of the chassis by plastic axles. The rear wheels also provide a platform for the encoder wheels to ride on to measure distance traveled. Non-magnetic casters, similar to those used on medical carts and chairs around MRI machines, were incorporated to support the front of the cart and to allow the cart to pivot around a center point between the rear axles. A polycarbonate tray that is capable of riding over roadway debris such as small rocks holds the sensor bar that houses the pucks. This cart design allows the front end to be tipped into the air, thus isolating the pucks from any surrounding metal. This motion is used to perform the on-the-fly calibrations in the field that capture the magnetic readings necessary for calculating bar depth.

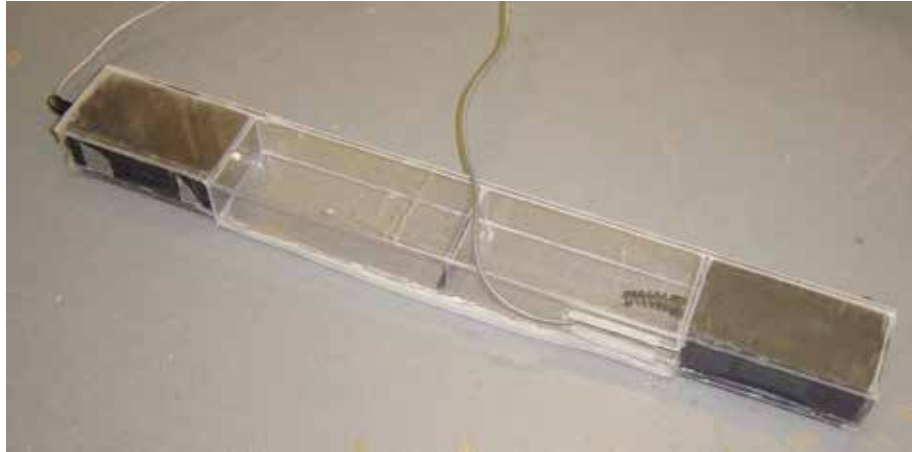
### *Sensor Spacing*

Migrating from a single sensor to multiple sensors introduced calibration issues of sensors interfering with one another. Pulse induction sensing with multiple devices requires synchronization. If the devices are not synchronized, the active pulse from one device will saturate the ‘echo listening’ of another device. Calibrating the device required that the sensor pucks stay in the same position relative to one another so that accurate depth measurements can be calculated from the raw data. Testing was performed to determine the effects of puck spacing on the magnetic readings in the absence of steel, referred to as the ‘infinity readings’ or ‘zero readings.’ A summary plot of the infinity reading as a function of sensor spacing is shown in Figure 4.2.



**Figure 4.2: Effects of Puck Spacing on Infinity Calibration Readings**

The distance between three sensors, commonly referred to as pucks, was varied from 3 inches to 14 inches at 1-inch intervals. Data was taken for 1 minute at each distance. The plot in Figure 4.2 shows that the sensor infinity readings are highly dependent on the puck proximity, especially for puck spacing of less than 9 inches. The infinity reading is analogous to the ambient magnetic background. Varying the proximity of the sensors is analogous to changing the ambient background. It is therefore critical to lock these sensors rigidly in place to prevent relative movement from effecting instrument accuracy. Since infinity readings are so crucial for determining the depth of the steel, a method for locking sensor pucks in fixed positions relative to one another was necessary to make the device's readings accurate and repeatable. A rigid polycarbonate structure was constructed to house two sensor pucks. Figure 4.3 shows a picture of the sensor bar.

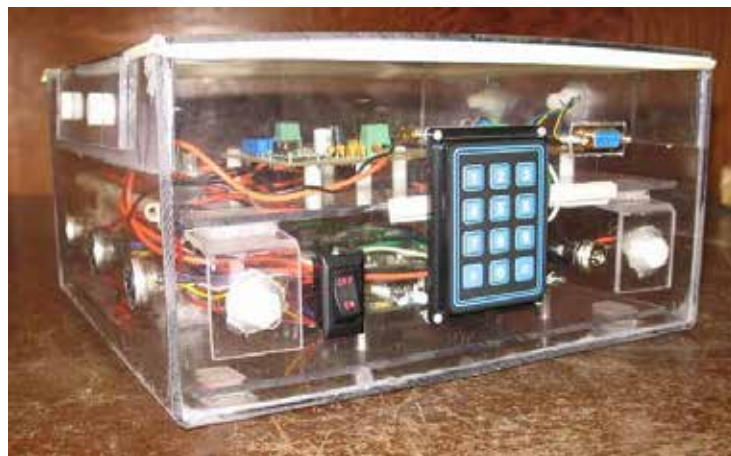


**Figure 4.3: Polycarbonate Sensor Bar**

This structure disallowed any relative movement between the two pucks and provided a simple way for removing the pucks from the cart in order to perform a full calibration in the field.

#### *Electronics Enclosures and Connectors*

The electronics had to be protected from adverse environments such as the ones mentioned previously in the Ruggedness section. The enclosures designed for this purpose were created from polycarbonate and solvent adhesive. The resulting boxes were not air or water tight but were capable of protecting the circuitry from wind-driven dust or the occasional rain shower during data collection. Figure 4.4 shows a picture of the main circuitry box that houses the GPS unit, power regulator, and covermeter electronics.



**Figure 4.4: Polycarbonate Electronics Enclosure**

The top of this box is secured with nylon bolts and the upper rim of the box is covered with a weather stripping material used to weather proof doors and windows. Appropriate connectors, switches, and device controls were mounted on the sides of the enclosure for easy access and device operation.

The connectors were a major design consideration due to the amount of vibration caused by running the cart over a textured concrete surface. Vibration caused normal four or five pin block connectors and DC power plugs to shake loose or flex such that contact could be intermittently lost. During testing, cabling to the encoder wheels was especially susceptible to such problems. The final design incorporated power sockets that have large contact area and locking capability. The connectors have a small keyed protrusion on the female connector that locks with a small cutout in the male connector and also only fit together in one orientation which prevents power from being applied in a reverse polarity to the circuitry.

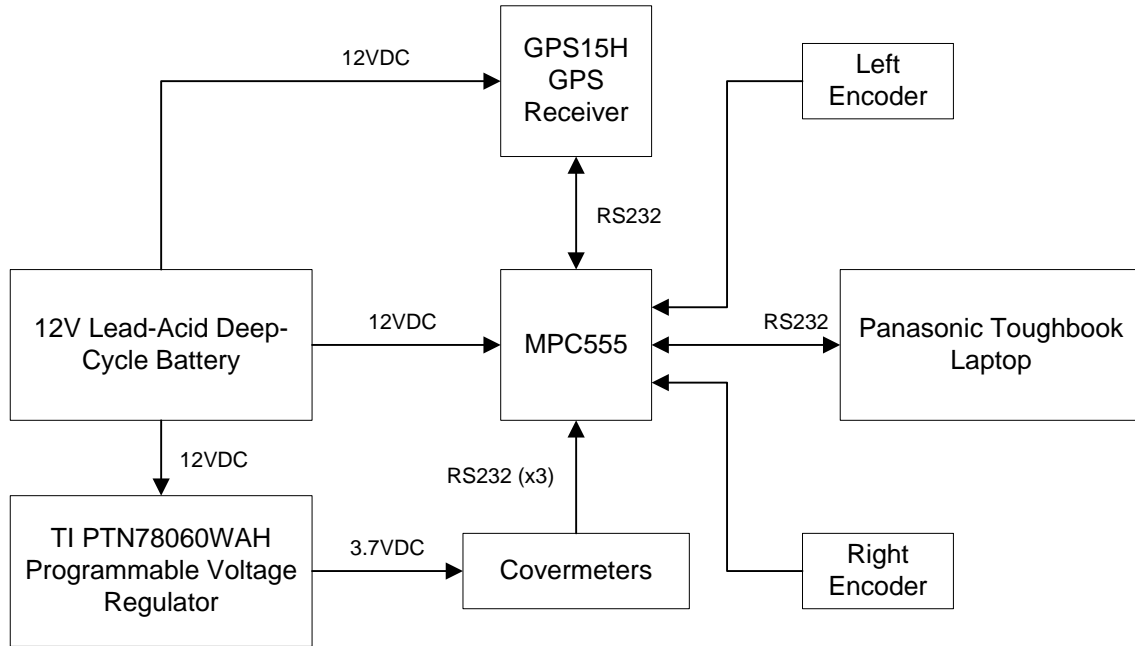
## **Hardware Integration**

Electronic integration was the primary challenge in this project. The preferred architecture dictated that the MPC555 microprocessor be integrated into the system. Peripheral boards were added as needed to adjust voltage levels and interface the main components of the system. The following sections discuss the different pieces of the electronic design.

### *Overview*

The majority of the electronics design involved taking commercial circuitry and interfacing it to both power and a communications channel. Figure 4.5 shows the block diagram for the electronics of the steel mapping device.



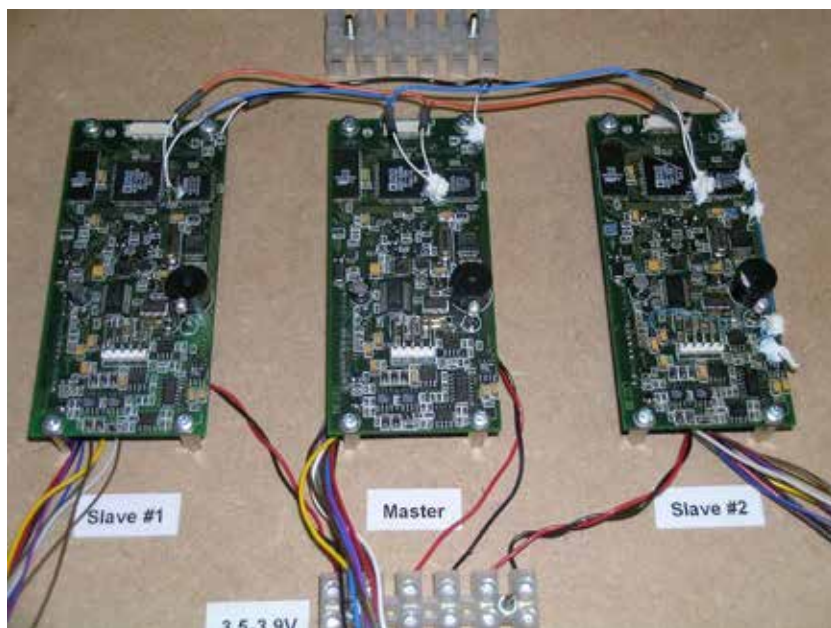


**Figure 4.5: Electronics Block Diagram**

The MPC555 forms the hub of the system and coordinates all communication between the user interface on the laptop and the rest of the peripheral devices. The following sections will describe the details of the various electronic components.

### *Synchronized Covermeters*

The covermeters are proprietary electronics developed and customized by Koletric and Tallix. They are modified to be synchronized and capable of RS232 serial communication as a result of negotiations between the suppliers, KDOT, and Kansas State University. The product specification document for the synchronized meters can be found in Appendix D. Figure 4.6 shows a picture of the first iteration of the synchronized meters.



**Figure 4.6: Synchronized Covermeters**

These synchronized meters have the on-board digital signal processing capability to determine a magnetic return from steel near the sensor heads. The orange, blue, and black wires seen at the top are used for synchronizing the three boards.

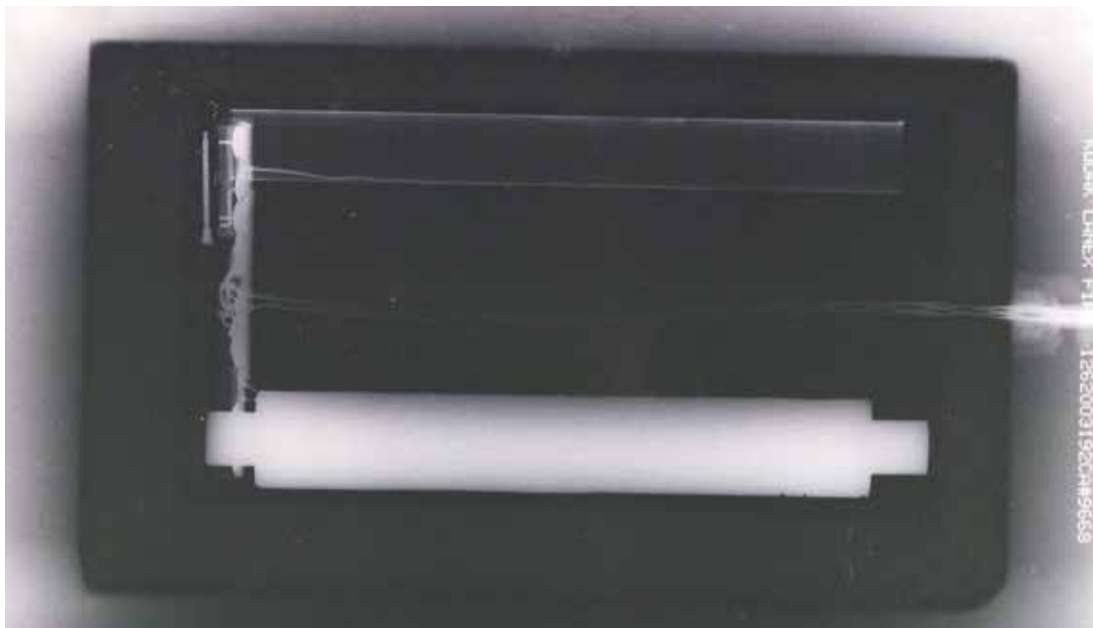
The power up sequence of the three boards is critical to proper operation. Both slave devices must be powered on and initialized before turning on the master device. If this order is not followed, the devices may appear to operate properly, but the electronics cannot be relied upon to yield time-aligned measurements. There are two methods for powering on the devices. The first is referred to as the soft method. Power is connected to the devices at all times during this method. The devices are turned on by shorting control pins four and five together for 1 second. They are turned off by shorting the same pins together for 2 seconds. This operation mimics the way the commercial meters operate with battery power always applied to the circuitry. The second is referred to as the hard method which involves applying power to turn the devices on and disconnecting power to turn them off. Switches can be used in the power loop to connect or break the circuit.

The second piece of each covermeter is a sensor head, sometimes called a puck, which connects to the boards shown in Figure 4.6. Figure 4.7 shows a picture of a puck.



**Figure 4.7: Covermeter Puck and Connectors**

This iteration of the puck has two connectors at the end. One connector hooks to the female connectors on the covermeters and the other is a DB9 connector that can hook to any standard RS232 computer serial port. The illustration in Figure 4.8 shows an X-ray of the puck.



**Figure 4.8: X-Ray Image of Covermeter Puck**

The puck houses a metallic core and two coils of wire that are secured by an epoxy that fills the voids in the puck. The ferromagnetic core can be seen sticking out beyond the coils of wire in the lower coil. This core has a tendency to break loose from the epoxy that holds it in place. If the core is loose, movement and/or vibration can invalidate calibration measurements, also referred to as the infinity readings. The lower core is wound with a larger gauge wire, which was evident by looking at its silhouette in the original X-ray. The upper coil appears to be wound with a finer gauge wire and has an air core [5].

The synchronized covermeters report magnetic readings over the RS232 compliant serial port at a rate of 20 Hz and are formatted in ASCII coded hexadecimal. The standard message has 10 characters. The first eight digits are 0-9 or A-F which form the numerical part of the message. Negative numbers are given in two's complement hexadecimal notation. The last two digits of a message string are carriage return and line feed [11]. The example file `SingleCovermeterExample.txt` in Appendix E shows an illustration of sample output data, positive and negative, from a single covermeter.

### *MPC555*

As previously mentioned, an embedded communication solution was attractive for purposes of eliminating costly third-party devices and enabling greater device scalability in the future.

### Microprocessor Selection

The choice of the MPC555 as the interface microprocessor was based on several factors. The first factor was the two TPU units located in the MPC555 [12]. These sub processors have built-in UART and fast quadrature decode functionality [13][14]. This allows many serial connections to be created through software. This programmability allows serial channels that only require simplex communication to utilize only one pin on the processor, thus not wasting any resources on unnecessary channels. The current device configuration uses two receive channels and two pairs of transmit/receive serial connections. The second decision factor was based on familiarity with the hardware and development environment. This processor is used in

several embedded systems courses at Kansas State University and the development tools are readily available in the Electrical and Computer Engineering Department's labs. The third factor was the number of digital I/O pins. Initial designs included controlling the power on/off sequence of the covermeters with the MPC555. This required six digital I/O pins for the current set of three synchronized meters. More I/O pins would be necessary if additional covermeters are added to the design, so choosing a processor with enough digital I/O to scale later was attractive. The final factor was ruggedness. The MPC555 is a microprocessor designed for the automotive industry. Automotive specifications typically reflect high operating temperatures and vibration. Operation of the sensing cart with its hard rubber wheels on textured concrete surfaces induces considerable vibration. All these factors led to the selection of the MPC555 for this project.

### Communication

All user-defined communication with the MPC555 is asynchronous serial communication and is handled through the TPUA sub processor's UART function. The peripheral devices all communicate with RS232 signal levels and polarity. All signals had to be shifted to 0VD to 5VDC input logic levels before reaching the MPC555 input pins. This was accomplished using three MAX233 level shifting chips. The two input serial channels associated with the covermeters and the transmit/receive pair that handles the communication with the laptop all ran at 19.2kBaud. The GPS communicated with the MPC555 at a rate of 4800Baud. All receive channels were polled by the main control loop as previously discussed.

The MPC555 program also has the ability to control the power on/off sequence of the synchronized covermeters. The sequence and timing are built into the program and can be controlled through defined command characters that can be sent from the laptop to the MPC555. Further design is required to create an interface between the MPC555's digital output and the control pins on the covermeters. This interface would ensure that no logic levels or drive currents exceed the maximum limits of the devices. Furthermore, it would also ensure that no cascading damage would occur to the covermeters or MPC555 in the event of an electronics malfunction.

### *Power Distribution*

All power to the electronics was derived from the 12V deep-cycle lead-acid battery. Table 4.1 shows the power requirements for the major components in the system.

**Table 4.1: Power Requirements by Component**

<b>Component</b>	<b>Voltage</b>	<b>Maximum Current</b>
Covermeter	3.5 VDC – 3.9 VDC	200 mA
GPS15H GPS Receiver	8.0 VDC* – 40.0 VDC*	60 mA
PB-555 Eval. Board and MPC 555	6.0 VDC* – 20.0 VDC*	385 mA
MAX233 Level Shifter	5.0 VDC	20 mA
61K128-050 Optical Encoder	5.0 VDC	30 mA

\* - unregulated

Power to the covermeters was provided by a Texas Instruments PTN78060WAH variable output switching voltage regulator configured to output 3.7VDC. The GPS receiver and PB-0555 evaluation board were both run directly off of the unregulated lead-acid battery. The PB-0555 has an on-board 5.0VDC voltage regulator that is used by the MPC555 contained on the board. This regulator was capable of supplying the current necessary for the three MAX233 level shifters and the two optical encoders at 5.0VDC within device specifications. During full load, the 5.0VDC regulator is supplying about half of its maximum current.

## Chapter 5: Software Architecture and Design

The following sections describe the software architecture for the steel mapping device.

### **MPC555**

The Freescale MPC555 was the chosen microprocessor as was previously discussed. The following sections outline the details of the software associated with this processor.

#### *Concept of Operation*

The MPC555 was added to streamline communications between the peripheral devices and the laptop. This alleviated the problems of dropped data and slow program execution when all processing was done by the Toughbook. The microprocessor can also respond to serial commands. Using this capability, the microprocessor can enable/disable the data stream via defined serial command words received from the laptop. The MPC555 condensed all communication to a single serial channel that required no proprietary device drivers. This architecture simplifies the programming environment and provides an extensible environment for future upgrades.

#### *Detailed Design and Flow*

The following sections detail the software initialization of the MPC555 hardware and the structure of the main control loop.

#### **Initialization**

The initialization routines for the UART and the fast quadrature decode are written in tpuUART.c and tpuFQD.c. All .c and .h source files can be found in Appendix C. The main control function is located in controlMain.c. It first initializes the buffers used to process serial data. It then sets the TPUA input clock to the desired value by calling the function initTPUA(). This function disables TPUA interrupts, sets the clock divider to divide the incoming clock by one, and enables the advanced prescaler which is set to divide the incoming clock by two. These

steps generate a TPUA clock of 10MHz. The main function then initializes the TPUB sub processor to generate the same clock frequency as TPUA.

The main function then initializes the fast quadrature decode functionality in TPUB. The program uses two pairs of channels to utilize the optical encoders mounted on the wheels of the mapping device. Channels six and seven form the pair to decode the right encoder and channels ten and eleven are used to decode the left encoder. Two channels are necessary to detect the phase difference between the pulses sent from the optical encoder. This allows the MPC555 to keep track of direction as well as distance. The initialization of the fast quadrature decode is an involved process that is well documented in the function `initFQD()` found in `tpuFQD.c`.

Serial channels are then initialized using the `initTx(UCHAR channel, USHORT baud)` and `initRx(UCHAR channel, USHORT baud)` helper functions written in `tpuUART.c`. These are general functions that were written to allow for any TPUA channel to be configured as a transmitter or receiver at any baud. The constants used to configure the channels to the correct baud are defined in `defines.h`. These values were calculated based on the 10MHz input clock that was set up in the initializations using the following equation.

$$BAUD\_CONSTANT = \frac{10MHz}{BAUD} = \frac{10^7}{BAUD}$$

The MPC555 communicates with the laptop at 19.2kBaud. Applying the above equation yields a value of 520.833. Rounding this value to the nearest integer produces a value of 521 which is the number used in the program for the peripherals that communicate at this baud.

The final initialization step involves setting up the GPS receiver. The GPS15H transmits seven different NMEA 0183 sentences by default. To lessen the communication load on the MPC555, all sentences but the GPRMC sentence are disabled. This is done by sending the PGRMO command sentence with the appropriate parameters [10]. After this initialization step, the embedded program begins its main control loop which is discussed in further detail in the following section.



## Control Loop

A control loop, rather than interrupts, is the method used to track all data and communications. The receipt of communication bytes from the TPU serial ports, transmission of output GPS strings and covermeter data, and handling of command messages are all handled in the MPC555 main control loop. Figure 5.1 shows a flowchart of the control loop operation.

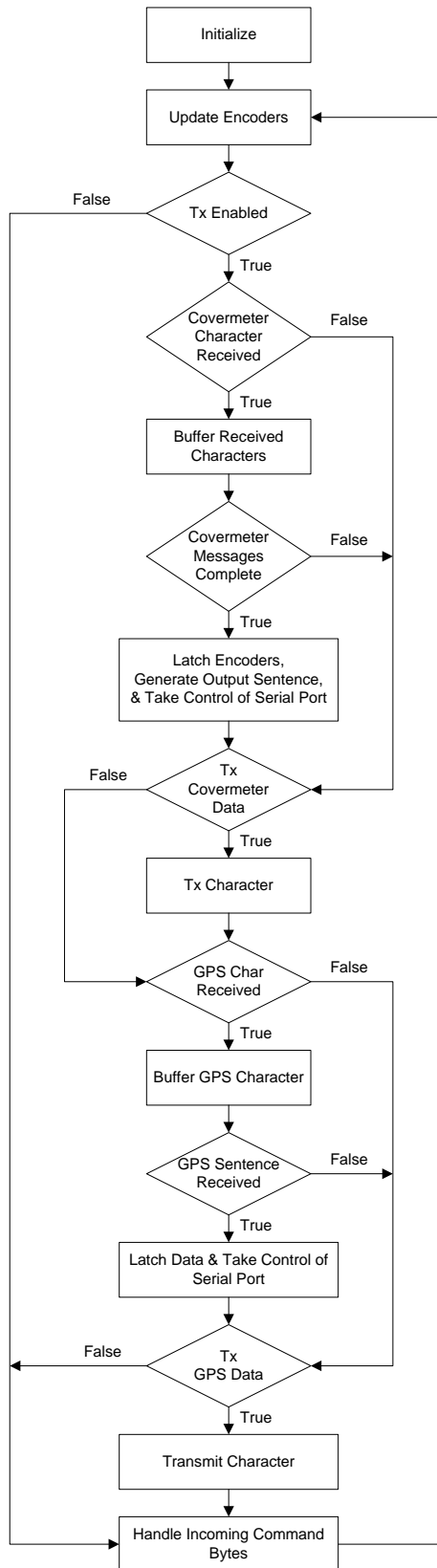
The incoming serial characters from the peripheral devices are all captured by polling the interrupt status register for the appropriate channels. Once a covermeter value has been fully received, the encoder values are latched and the transmission of the output sentence to the laptop begins if the respective communication port is not busy. If the port is busy transmitting the most recent GPS string, the transmission to the laptop will commence as soon as the port is available. Output strings, either GPS or covermeter, are all transmitted one character at a time. This allows the processor to keep polling the input channels while the current output byte is being shifted out at the correct rate to match the baud value. This approach takes advantage of parallel processing since the TPUA sub processor and the main processor are performing calculations at the same time. Without parallel processing the processor would have to wait for a worst case ~3.85msec which is enough time for the processor to miss part of the next incoming sentence. The TPUB sub processor is also performing its fast quadrature decode tasks in parallel so the total system behaves as though it had three independent processors running in parallel. The main control loop for the MPC555 is found in controlMain.c.

## Ruggedized Laptop

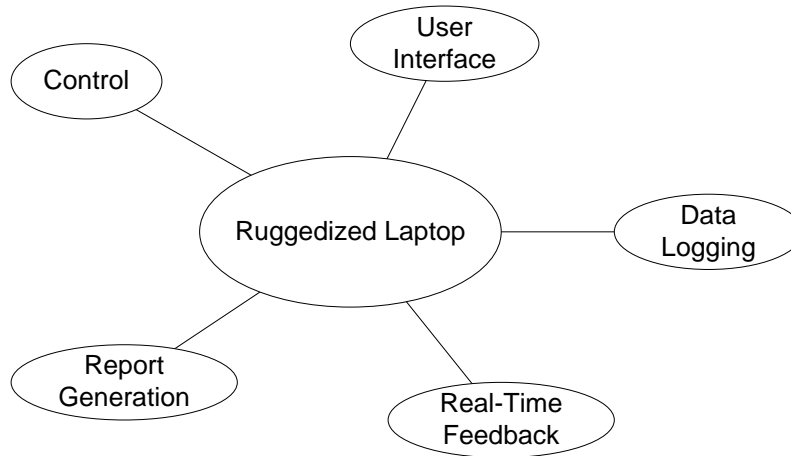
A Microsoft Windows based ruggedized laptop was chosen for developing the application level software. Visual Basic and Matlab formed the basis for data acquisition, user interface, and report generation. The following sections outline the design and typical use of the high-level programs.

### *Concept of Operation*

The concept of the laptop operation revolves around serial communication and providing application control to the user. Figure 5.2 shows the responsibilities and attributes of the laptop.



**Figure 5.1: MPC555 Main Control Loop**



**Figure 5.2: Ruggedized Laptop Responsibilities**

The laptop saves all the data received from the MPC555 including magnetic sensor readings originating from the covermeters, appropriate zero calibration data, GPS readings, and distance calibration values into a defined text format. A customized user interface provides real-time feedback and control of the data collection process to the user. The single serial data communication channel between the microprocessor and the laptop enables the laptop to run the necessary real-time calculations and update the user interface without any noticeable lags in operation. The ruggedized laptop also contains the routines for multi-sensor calibration, distance calibration, and report generation.

### *User Interface*

The main user interface that provides visual feedback and user control is provided by the main scooter program which is written in Visual Basic. Figure 5.3 shows a screenshot of this program.

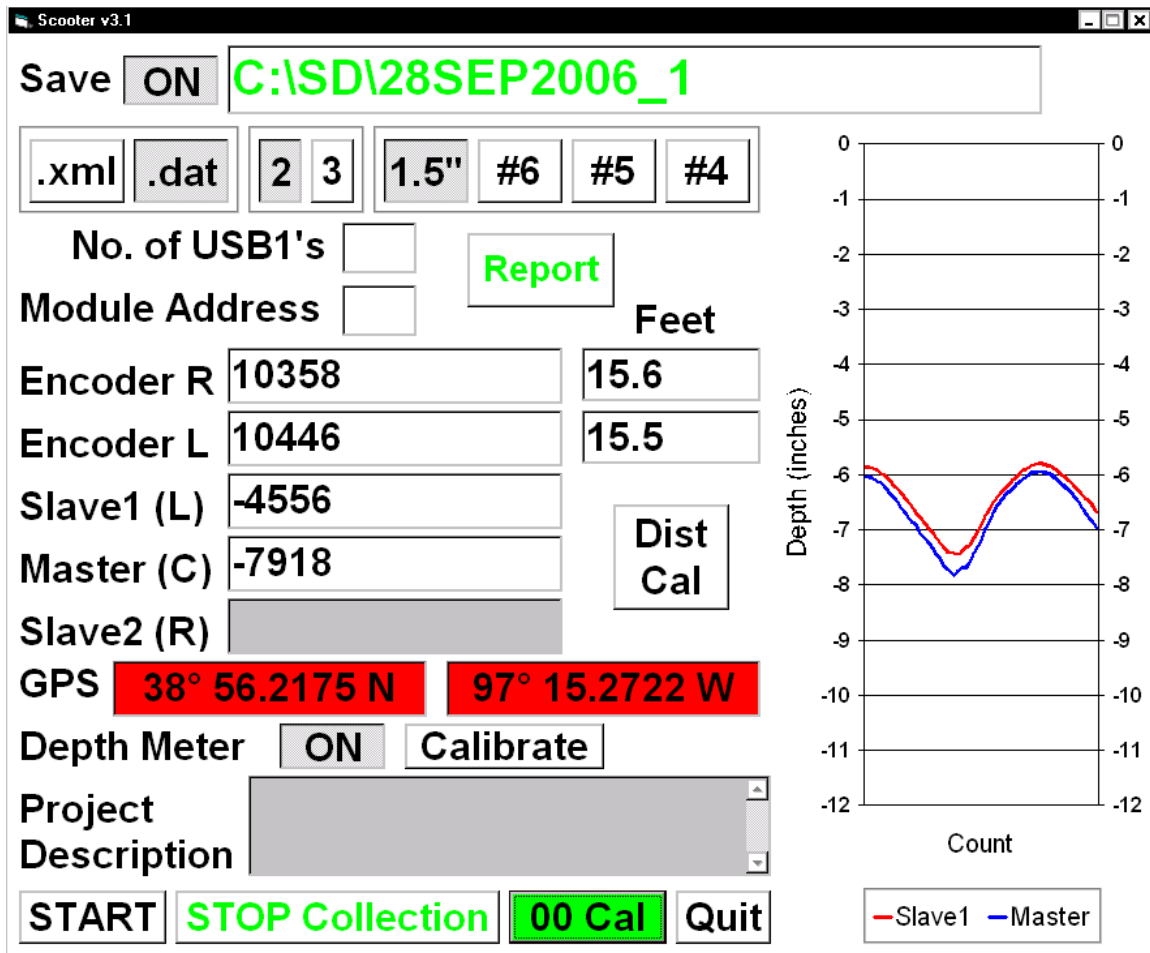


Figure 5.3: Scooter Program

The original software was used for both for operational testing and troubleshooting. As such, it still bears many of the troubleshooting remnants. The captured data is saved in a file with the filename specified in the text box at the top of the window. Naming conventions are left to the operator to adapt to the situation. The default filename is the current date in DDMMYYYY format. The depth meter is turned on by default. It calculates and plots the depth of the steel in real-time on the right side of the window. A default calibration file is used to generate the polynomial coefficients that map the raw data to a depth reading. GPS data is displayed in the adjacent text boxes beside the GPS label in the user interface in degrees/minutes format. When the field is highlighted in red, the GPS data is not valid. Otherwise, the GPS receiver is sending valid fix data.

## Multi-Sensor Calibration Procedure

Multi-Sensor calibration is performed using a calibration program written in Visual Basic. Figure 5.4 shows a screenshot of the interface provided by this program.

The screenshot shows a Windows-style window titled "Proba3D Calibration". The interface includes the following elements:

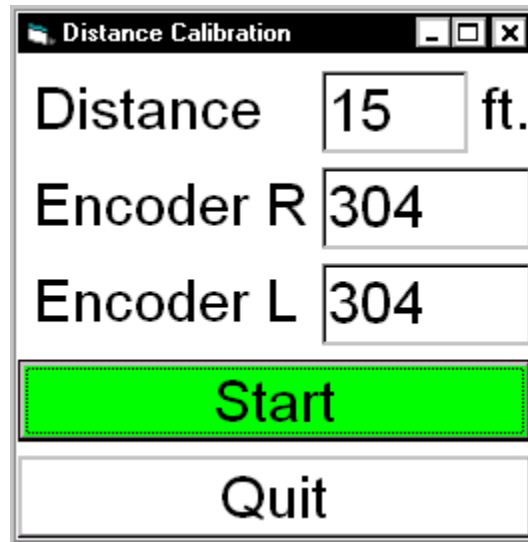
- Output:** A text box containing the path "C:\DATA\CAL\_1".
- File Type Selection:** Two buttons labeled ".xml" and ".cal". The ".cal" button is highlighted.
- Before/After Section:** Two large grey rectangular buttons labeled "Before" and "After".
- Distance Selection:** A row of buttons for distance intervals: "2", "3", "2\"", "4\"", "6\"", "8\"", "10\"", "12\"", and "00". The "8\"" button is highlighted.
- Readings:** Three text boxes for numerical readings. The first two contain "39938" and "39572", while the third is empty.
- Labels:** The labels "Slave1", "Master", and "Slave2" are positioned below the first, second, and third reading boxes, respectively.
- Action Buttons:** Two buttons at the bottom labeled "COLLECT" and "Quit". The "COLLECT" button is highlighted.

**Figure 5.4: Calibration Program**

The output file can be selected by typing a filename and path in the textbox that is labeled 'Output:'. The calibration prompts the user to take 1 second samples of data, 20 data points, at 2-inch intervals starting at 2 inches and ending at 12 inches. Infinity readings are taken before and after the depth measures are recorded. The end result is a .cal file with tab delimited data. See Appendix E for an example calibration file named CAL\_5\_JUL06.cal. The first column in the calibration file is the distance, in inches, of the calibration bar from the sensor bars. The 0 tag found at the beginning of the first and last 20 lines in the calibration file indicate readings taken in the absence of steel sometimes called infinity readings. Calibration files are used by the data acquisition program and the report generation program to generate polynomial coefficients that map the raw data values to depth measurements in inches.

## Distance Calibration Procedure

A distance calibration routine that allows for in-field encoder calibration is built into the Visual Basic user interface. The Dist Cal button shown in Figure 5.3 brings up the window shown in Figure 5.5.

A screenshot of a Visual Basic window titled "Distance Calibration". The window has a standard Windows-style title bar with minimize, maximize, and close buttons. Inside the window, there are three input fields: "Distance" with the value "15" and "ft." to its right, "Encoder R" with the value "304", and "Encoder L" with the value "304". Below these fields are two buttons: a green "Start" button and a white "Quit" button.

**Figure 5.5: Distance Calibration Program**

The distance calibration routine can set new default values for the number of encoder pulses per foot. The Distance field defaults to 15 feet but can be changed to any value greater than zero. The program records the number of encoder pulses as the cart travels the given distance. From this information, the program can calculate the conversion factor between the number of pulses and the distance traveled in feet. This routine is useful on the current version of the cart where the encoder wheels are not linked via gears to the drive wheels or axle. It also allows for the event that different resolution encoders replace the current encoders. Ultimately, this program allows the data acquisition program to adapt to any unforeseen situations causing variations in measuring distance traveled.

## Zero Calibration Procedure

Zero calibration, also called infinity calibration, is a process used to obtain the infinity readings from the covermeters when the pucks are isolated from steel. Figure 5.6 shows the isolation process.



Reproduced from [8]

**Figure 5.6: Zero Calibration Cart Orientation**

The sensor pucks must be at least 18 inches away from any ferrous materials during this process. While in this position, the 00 Cal button, as seen in Figure 5.3, is pressed. Data is collected for 1 second and the new zero reading is found by taking the median of the resulting 20 data points. The zero calibration routine is done so that full multi-sensor calibrations do not have to be performed in the field. Performing multiple zero calibrations during a data run allows the reporting software to account for sensor drift which will be discussed in further detail later.

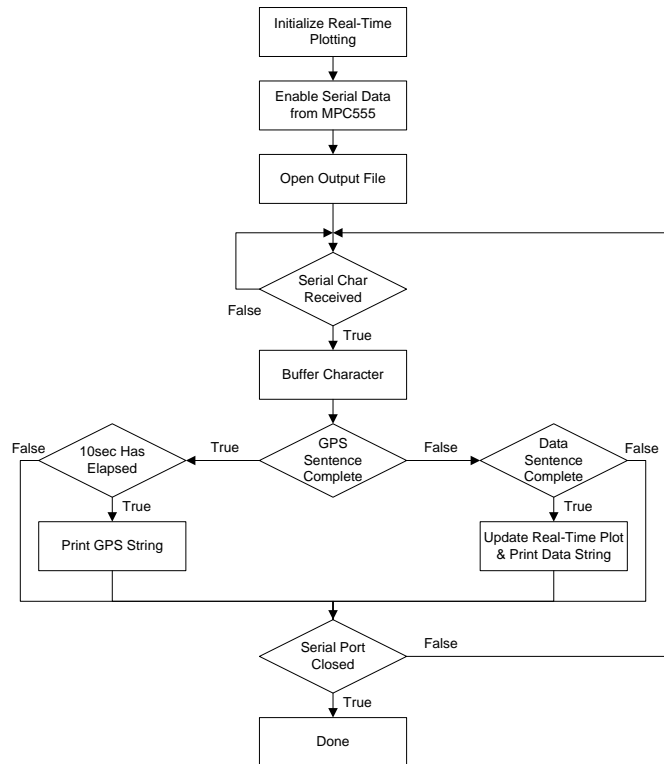
## Data Collection Procedure

Typical operation of the data collection procedure is outlined in the following paragraph. Buttons and text fields that are referred to here can all be found in Figure 5.3. A filename is first chosen and entered into the filename text box. The next step is to use the Dist Cal button to

change the default encoder calibration if desired. A different real-time plotting calibration file can also be selected at this point by hitting the Calibrate button. The 00 Cal button is then pressed to start the data acquisition cycle. Note that the nose of the cart must be at least 18 inches from the concrete for the calibration to be valid. The initial press of the 00 Cal button locks in the values for the distance calibration for the run, gives an initial GPS position for the beginning of the run, and saves 1 second worth of data to the data file to be used as an infinity calibration reading. The most recent infinity calibration is used in the real-time plotting. The median value of the infinity calibration is subtracted from the data before applying the polynomial fit equation to determine steel depth. Once this step is completed, focus will automatically go to the START button. Pressing this button starts the data collection. Values for the raw meter data, encoder pulse count, and translated distance should all start filling the appropriate text fields. The STOP Collection button should be pressed when the run is complete or another on-the-fly calibration needs to be performed. It should automatically have focus after beginning the data collection. An infinity calibration, using the 00 Cal button with the nose of the cart at least 18 inches from the concrete surface, should be performed at the end of every data collection run before generating a report or starting another run. Once the final infinity calibration is completed, a new run can be started by changing the filename in the text box at the top of the window or a report can then be generated by pressing the Report button. The report generation process will be discussed in detail later.

The flow of the data acquisition portion of the main program is shown in the flowchart seen in Figure 5.7.



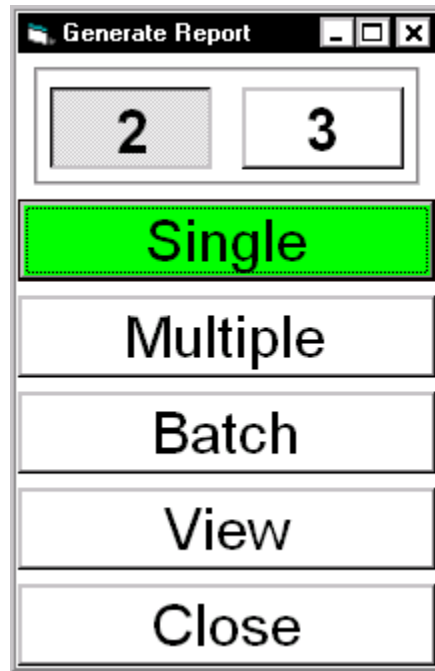


**Figure 5.7: Data Acquisition Flowchart**

The first part of the data acquisition flow is initialization, with the program performing one-time tasks such as initializing the real-time plotting, enabling the data stream from the MPC555, and opening the output file. The program then gets into the main data acquisition loop. The serial port is polled until a character is received. Upon receipt, the character is buffered and a test is done to determine if a complete GPS sentence has been received. If that is true and the appropriate time has elapsed (10 seconds in the current setup), a GPS sentence is formed and written to the output file. If a GPS sentence is not complete, the program checks to see if the incoming string is a completed encoder/covermeter string. If so, the program writes a data line to the output file. The program then checks to see if the STOP Collection button has been pressed. If so, the execution of the control loop is done. Otherwise, the program loops back and waits for another character to be received.

## *Reporting Software*

Report generation is started by clicking the Report button in the data acquisition program which is shown in Figure 5.3. That button press brings up the window shown in Figure 5.8.



**Figure 5.8: Report Generation Program**

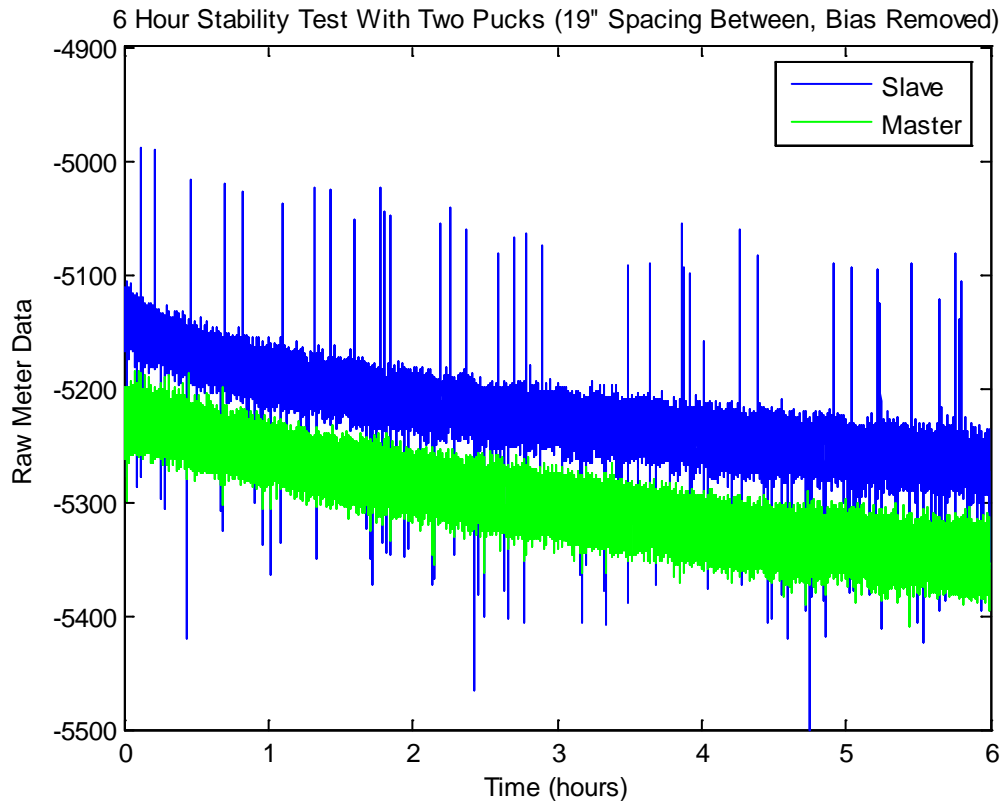
The Single button is pressed to generate a report from a single .dat file from the data acquisition program and one .cal file from the full calibration program. The Multiple button is capable of generating a single report from two data files and one or two calibration files. This function is useful for comparing two runs over the same joint to test repeatability as well as testing the effect of different calibration files on the same input data. The Batch button was added so that multiple reports could be generated in one step using many data files and a single calibration file. This allows for generating all reports from a day of data collection at one time instead of having to generate one report at a time. The View button opens a generated report for viewing in an Internet Explorer window and the Close button terminates this window and returns to the data acquisition program.

## Estimating Bar Position

Bar position estimation is handled by an algorithm written in Matlab and compiled into a COM object that is called by the reporting portion of the Visual Basic program. The raw data from the output .dat file is translated into distances measurements in feet and depth values in inches. The bar finding algorithm then finds the peaks in the trace plots using a windowing approach that assumes a nominal bar spacing of 2 feet. The resulting output is a pair of vectors containing the depth and distance information. These values are used to plot the bar positions on the trace plots and generate the histograms that show the statistical results of the data run. The current version of the bar finding algorithm does not account for the dowel basket effects on the tie bar data runs. This can result in some erroneous bars that show up as outliers in the histograms.

## Plotting

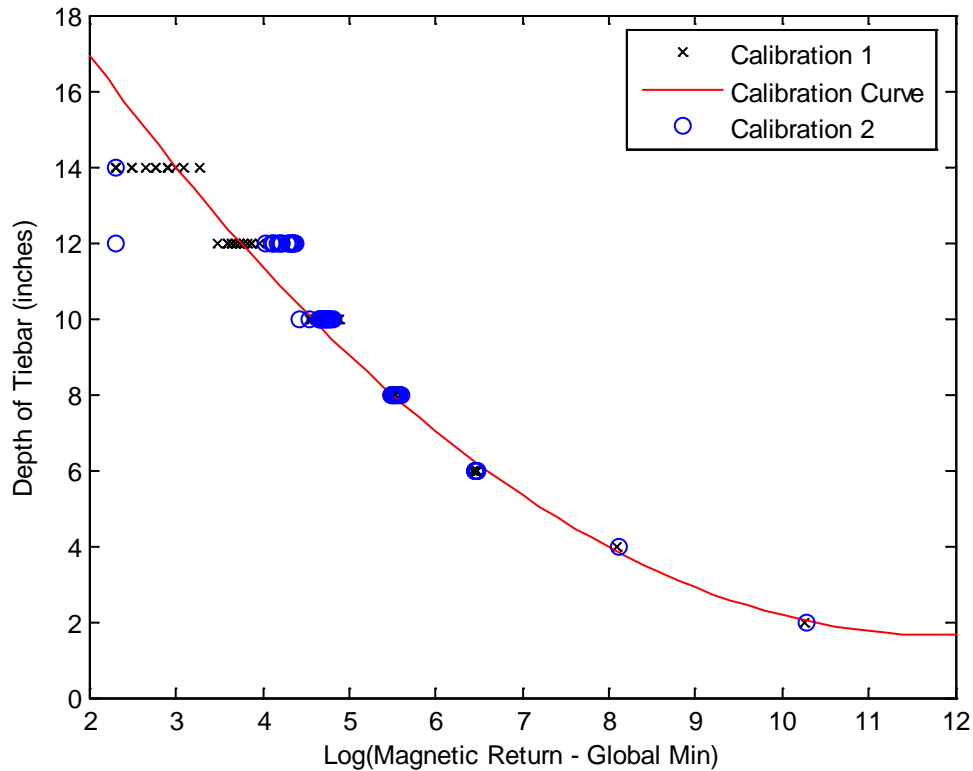
Plotting for the reports is done in a compiled COM object that was generated in Matlab from several .m script files. The global minimum value is found by taking the median of the 40 infinity readings found in the calibration file. This global minimum value represents the DC bias of the meter readings. This value can change at each power on of the meters so an on-the-fly calibration is necessary before each run so that the correct value can be found and used in the plotting of the data that follows. The global minimum value also has a tendency to drift with time. Figure 5.9 shows a 6-hour test of the covermeters without the presence of steel.



**Figure 5.9: Infinity Reading Stability**

The plot shows that the infinity calibration readings vary with time. The test was performed in the lab with no steel within 24 inches of the sensor bar. Controlled experiments were conducted with the sensor pucks in various temperatures, orientations, and degrees of direct sunlight and although there is a repeatable pattern to the drift, no correlation could be found between the experimental factors and the drift of the sensors. The data acquisition program requires that an on-the-fly calibration be performed at the beginning and end of each data run. This provides the reporting software with enough data to do a linear interpolation of the global minimum value during the data collection. This results in a more accurate representation of the steel location in the concrete.

The remainder of the data, the values taken at 2 inch intervals from 2 to 14 inches, is used to calculate coefficients of a second order polynomial that relate signal strength to bar depth using a weighted least-squares approach. Figure 5.10 shows an example of the fitting curve used for this process.



**Figure 5.10: Calibration Curve Fitting**

Weights for the different depth readings were calculated based on the variance of the data at each depth and the estimated accuracy of the jig used for calibration. This weighting allowed more reliable data, the readings found at depths of 2 to 8 inches, to have more of an effect on the shape of the curve. This range (less than 8 inches) corresponds to the expected location of properly placed steel in Kansas concrete pavements.

### Output

The output generated by the data acquisition program is saved to a tab delimited .dat file. The example file 29AUG2006\_S01\_C\_ABBREVIATED.dat which shows the formatting of a typical .dat file can be found in Appendix E. The first line of the example file is as follows:

```
C      -1      -2      664.9  671.9
```

This line is the distance calibration line, marked with a 'C' followed by '-1' and '-2', which is always the first line in a .dat file. The final two numbers are the encoder pulse-to-feet

conversion factors. The first number is used for the right encoder translation and the second is used for the left encoder translation. This allows the data acquisition program to incorporate the distance calibration values gathered from the distance calibration routine instead of using the default values.

The second line of the example file is as follows:

```
G      -1      1      38.928995      0      97.364428      1      290806      153555
```

The 'G' followed by the '-1' and '1' indicate that this line is a GPS string. The fourth value, '38.928995' is the latitude, in degrees, of the GPS fix and the '0' following the latitude is an indication that the fix was in the northern hemisphere. A '1' in this position would indicate position in the southern hemisphere. The fifth value, '97.364428', is the longitude, in degrees, of the GPS fix and the '1' following the longitude indicates that the fix was in the western hemisphere. A '0' in this position would indicate that the fix occurred in the eastern hemisphere. The eighth value, '290806' is the date of the GPS fix in the format DDMMYY. The final value, '153555', is the time of the GPS fix in HHMMSS format. The time information is presented in the 24-hour format. GPS lines occur immediately preceding on-the-fly calibration blocks and every 10 seconds during normal data acquisition.

The third type of line encountered in the example file is as follows:

```
Z      -38958.4418402778  -38958.4418402778  -5268  -8686
```

The 'Z' followed by two identical negative values indicate that this line is an on-the-fly calibration line. Blocks of 20 of these lines are found at the beginning and end of each .dat file and may occur anywhere in the middle of the data collection stream. The negative numbers following the 'Z' are unique numerical values based on the current timestamp. These are used to order the on-the-fly calibration results correctly during the report generation process. The fourth value, '-5268', is the infinity reading for the Slave1 or left sensor. The final value, '-8686' is the infinity reading for the Master or right sensor. These types of lines are used to calculate the infinity readings of each meter so that accurate depth calculations can be performed in the report generation stage.

The final type of line encountered in the example file is as follows:

```
D      89      95      -5070 -8494
```

The 'D' indicates that this is a data line. The second value of the string, '89', is the current encoder count for the right encoder. The third value, '95', is the current encoder count for the left encoder. The fourth value, '-5070', is the current reading from the Slave1 or left covermeter and the final value, '-8494', is the current reading from the Master or right covermeter. These types of lines are generated during normal data collection.

The Matlab plotting routines plot the results in standard figure windows which are then saved as .jpg files in a folder generated at runtime. The folder name and .jpg filenames are all derived from the name of the data file sent to the routine by the report generation program. An .htm file is generated by the Matlab code that links to the proper images and provides an easy web browser interface for viewing the data collection results. These .htm files can also be uploaded to a web server and linked through Google Earth for geospatial locations as previously discussed.

## Chapter 6: Testing and Field Trials

A great deal of time in the design process of the mapping device involved both lab testing and, more importantly, field testing. The experience gained from taking the device to construction projects was integral to the successful functioning of the device. Field testing yielded results and exposed problems that would have been difficult to predict and troubleshoot in the lab. Usability issues such as screen brightness were blatant problems that needed to be fixed. Other issues such as encoder pulse accumulation errors were not so easy to detect in the lab due to limited space and only exhibited themselves after long runs on the road. The iterative design cycle of this device made field testing a natural part of the evolution.

### Calibration

Two factors contribute to the accuracy of the measurements taken by the device. One is the ability to keep the sensor pucks locked so that there is no relative motion as was previously discussed. The other is the ability to take accurate measurements at given increments for the purpose of performing a curve fit that can translate raw meter data into depth readings. A structure called the calibration jig was built for the purpose of generating reliable and repeatable calibration values and is shown in Figure 6.1.

The large slots cut into the top of the jig near the left edge holds the two-sensor version of the polycarbonate sensor bar. The rest of the smaller slots were cut to accurately place a piece of #5 rebar at specified distances away from the pucks in the sensor bar. There are slots cut at 2-inch increments from 2 inches to 12 inches. The jig was fabricated completely out of pine 2"×4"s, 5/16" dowel rods, and wood glue. This platform allowed for accurate and repeatable calibration of the two-sensor device.





**Figure 6.1: Calibration Jig**

Field testing of the device showed that the results from a single calibration file could be used as long as the sensor bar remained intact with the pucks that were used for the initial calibration. Tests showed that the shape of the curve used for fitting the raw data to the depth reading does not change. This implies that the coefficients driving the shape of the curve also do not change. Standard operation of commercial covermeters includes taking an infinity reading periodically to calibrate the unit. This is consistent with what was found with the multiple-sensor device. Therefore, a single default calibration file could be used for each type of bar being measured. The default file in conjunction with on-the-fly calibrations yielded enough information to accurately map the steel placement. This realization negated the need to haul the calibration jig to the data collection site since only on-the-fly calibrations needed to be performed to generate accurate results.

## **Performance Verifications**

Performance verifications were run on the mapping device using several techniques. Laboratory tests were performed on the data acquisition, calibration, and report generation software to ensure that the programs were correctly handling incoming data and producing

reasonable results. Testing was also performed on the hardware in the lab to ensure that all communications were being handled by the MPC555 correctly. Signaling and voltage levels were all checked using oscilloscopes and multimeters before interfacing different sections of the hardware. Stability testing and battery testing was performed to ensure that the system could run for the several hours without encountering any errors or unexpected events.

Controlled tests were performed outdoors away from the sensitivity altering effects of fluorescent lights and other sources of electromagnetic noise present in the lab. These tests took advantage of a test track that was created to make distance verification simple as well as provide a portable system capable of being taken to off-site locations for demonstrations. Figure 6.2 shows the test track.



Reproduced from [8]

**Figure 6.2: Verification Test Track**

The track is composed of four sections of wood I-beams and two sections of plywood. The entire system is isolated from the ground by corrugated cardboard boxes to ensure that no steel embedded in the concrete floor or road surface could interfere with the magnetic sensing. Holes were drilled at places corresponding to depths of 3 inches to 10 inches at 1 inch increments. Holes offset from the center were also drilled so that rebar could be positioned in practically any orientation to test the results of the device.

Tie bar depth measurement tests using the calibration jig were performed to find the measurement accuracy and precision of the magnetic sensing device. Table 6.1 shows the theoretical and measured results from the measurement verification tests.

**Table 6.1: Depth Verification Test**

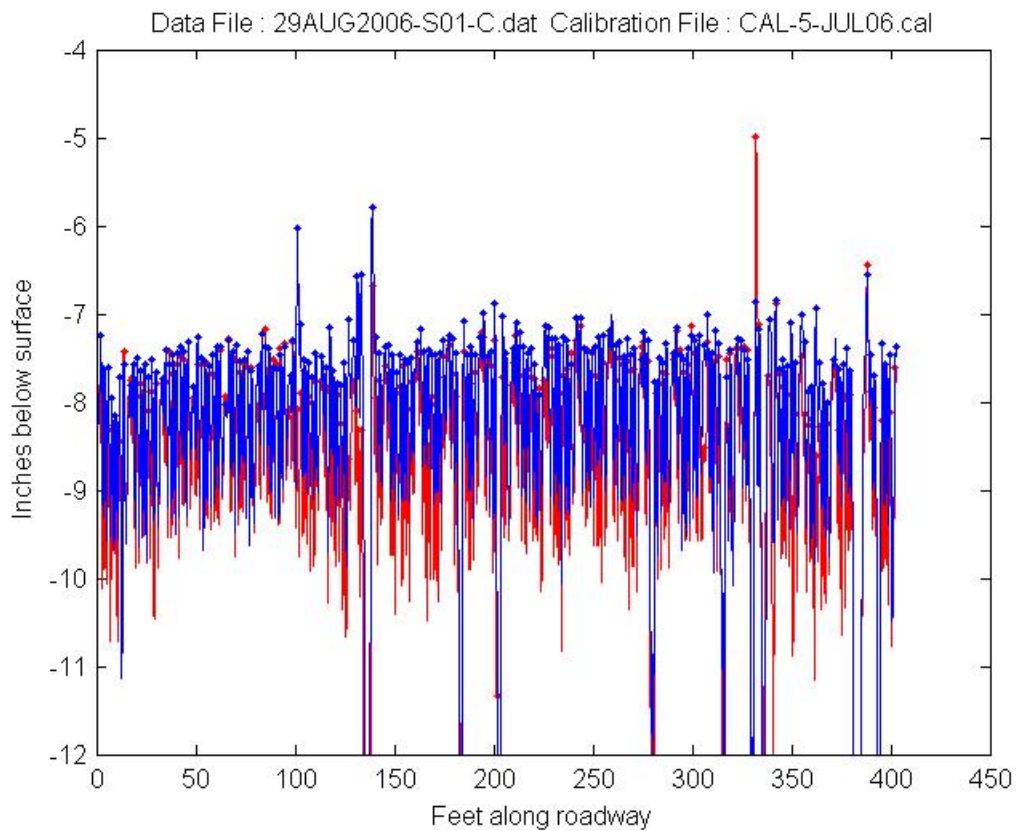
<b>Actual Depth (inches)</b>	<b>Master Measured Depth (inches)</b>	<b>Slave Measured Depth (inches)</b>	<b>Maximum Error Range (<math>\pm</math> inches)</b>
2	2.13	2.13	$\pm 0.00064$
4	4.04	3.73	$+0.011/-0.010$
6	6.04	5.94	$+0.081/-0.086$
8	7.91	8.16	$+0.32/-0.60$
10	9.63	10.62	$+1.00/-2.21$

The measured depth values were found by taking the median of 600 depth samples and the error ranges are calculated from the maximum deviation from the median depths. The results of the test follow what was expected. The measured values match what is known about the steel placement by hand measurements. Measurement susceptibility to noise becomes evident at bar depths of 8 inches and greater due to the decreasing signal to noise ratio. This can be seen from the increasing maximum error range numbers as the depth increases. Real-time and/or post-processing digital filtering may decrease the error ranges and thus increase the precision of the device, but this idea has not been implemented or tested.

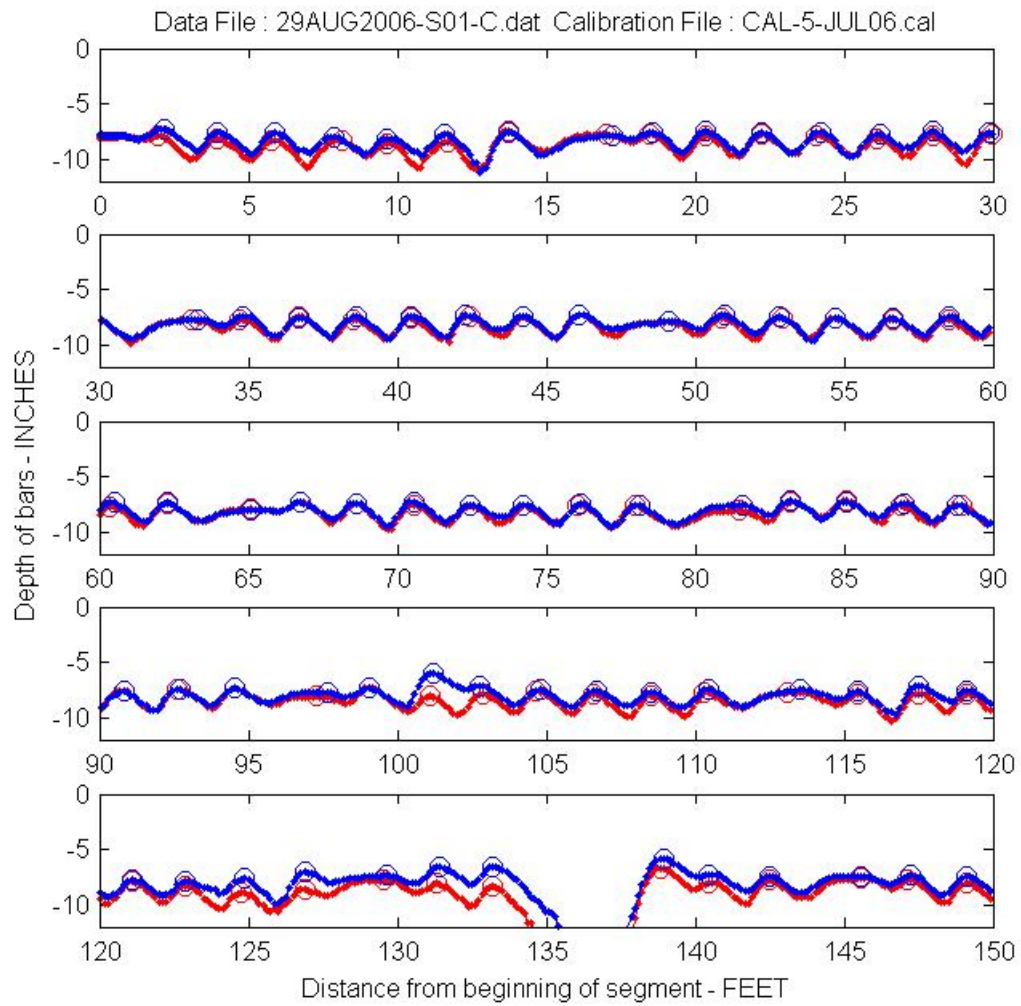
Environmental testing and data collection was performed on construction sites around the State of Kansas. This testing was performed to ensure that the device could operate in the environment for which it was intended. Part of the verification process in the field was taking data measurements over an exposed shoulder joint where the ends of the steel reinforcement could be seen. These runs provided two types of validation. Visual inspection confirmed the depth measurement of the prototype. It also showed that concrete cover did not adversely affect the instrument accuracy, as was expected since concrete does not contain any magnetic or highly conductive material.

## Sample Runs

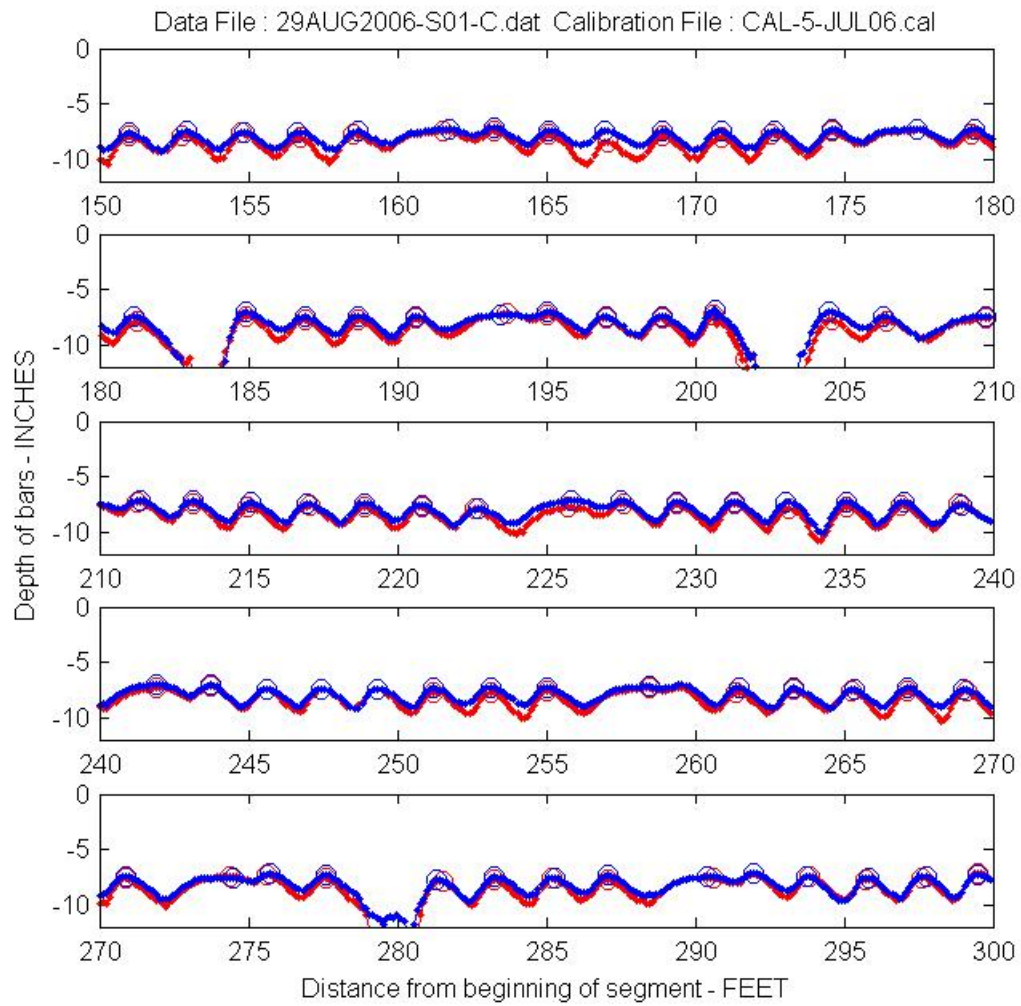
The mapping device was taken to several construction projects during its development and was able to take data that yielded radically different results. The first sample run was taken west of Abilene, KS, on I-70. Figure 6.3 through Figure 6.8 were generated by the reporting software and uploaded with the referencing .html file to the Internet to be viewed through Google Earth.



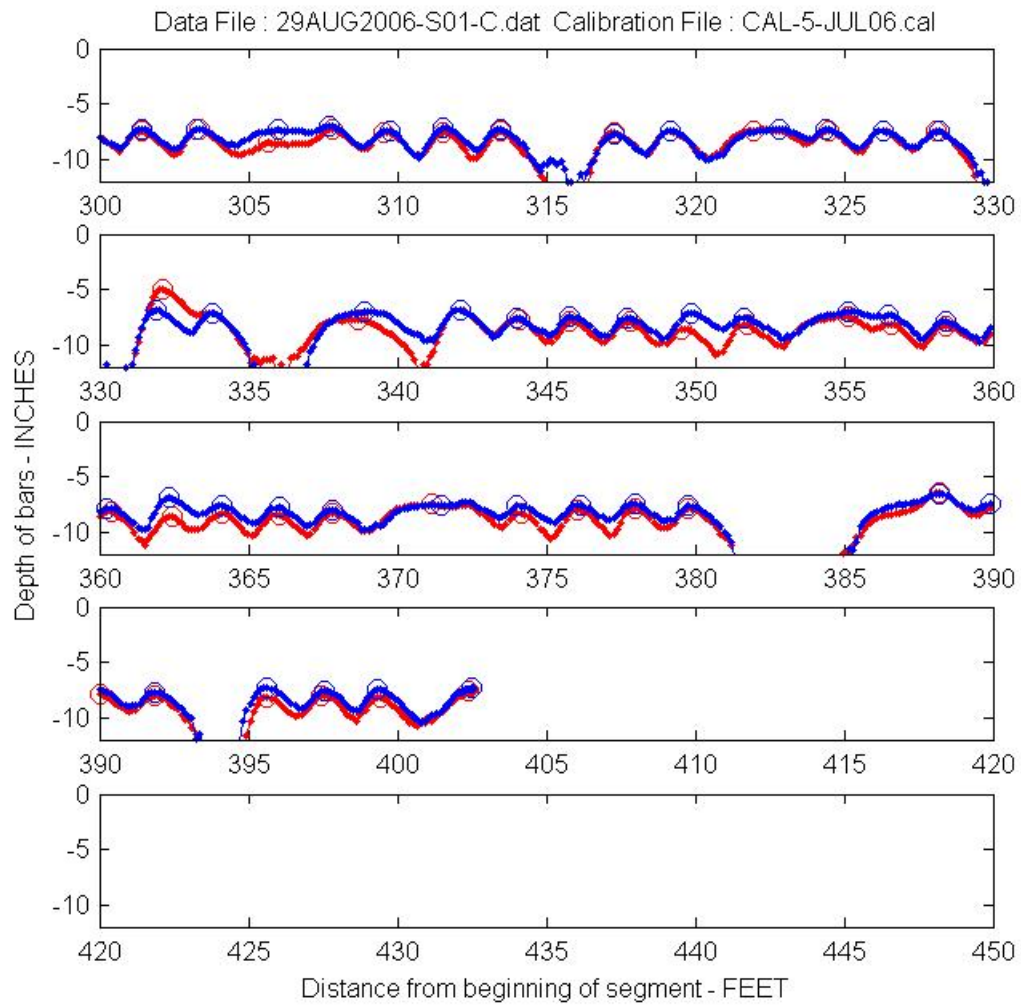
**Figure 6.3: Condensed Plot of I-70 Testing Along Center Joint**



**Figure 6.4: Expanded Plot of I-70 Testing Along Center Joint (1/3)**

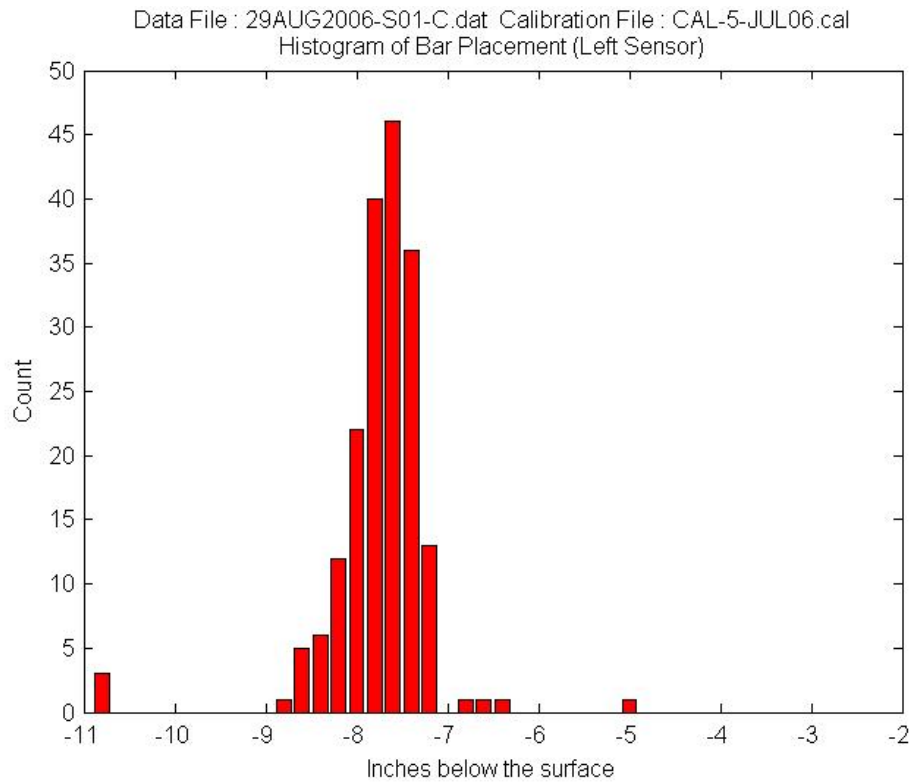


**Figure 6.5: Expanded Plot of I-70 Testing Along Center Joint (2/3)**

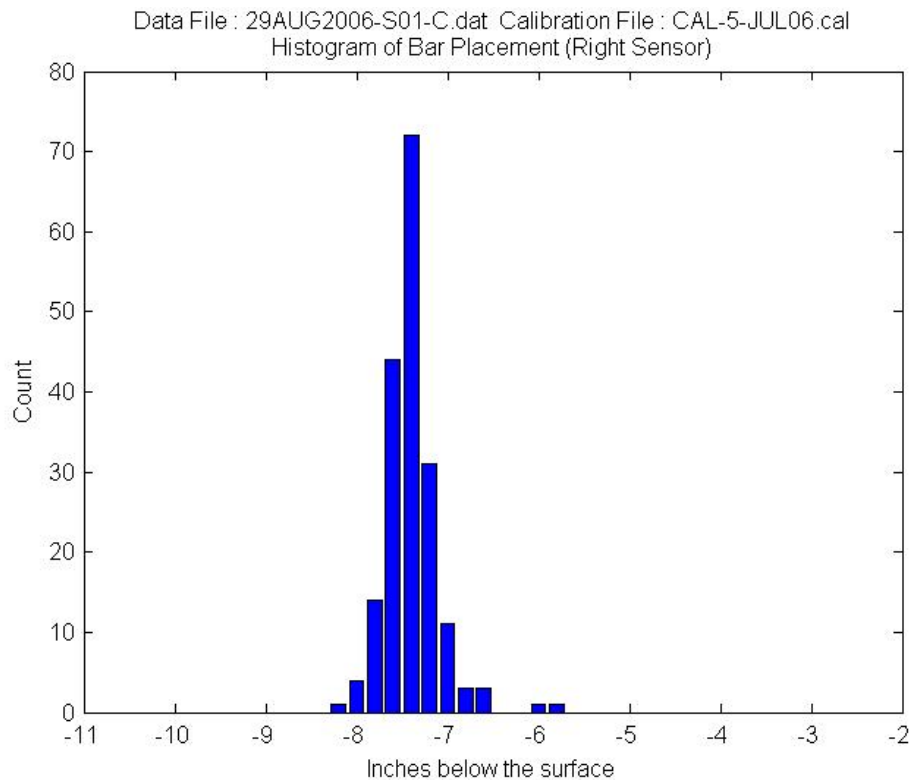


**Figure 6.6: Expanded Plot of I-70 Testing Along Center Joint (3/3)**





**Figure 6.7: Histogram of Left Sensor Bar Placement from I-70 Testing**

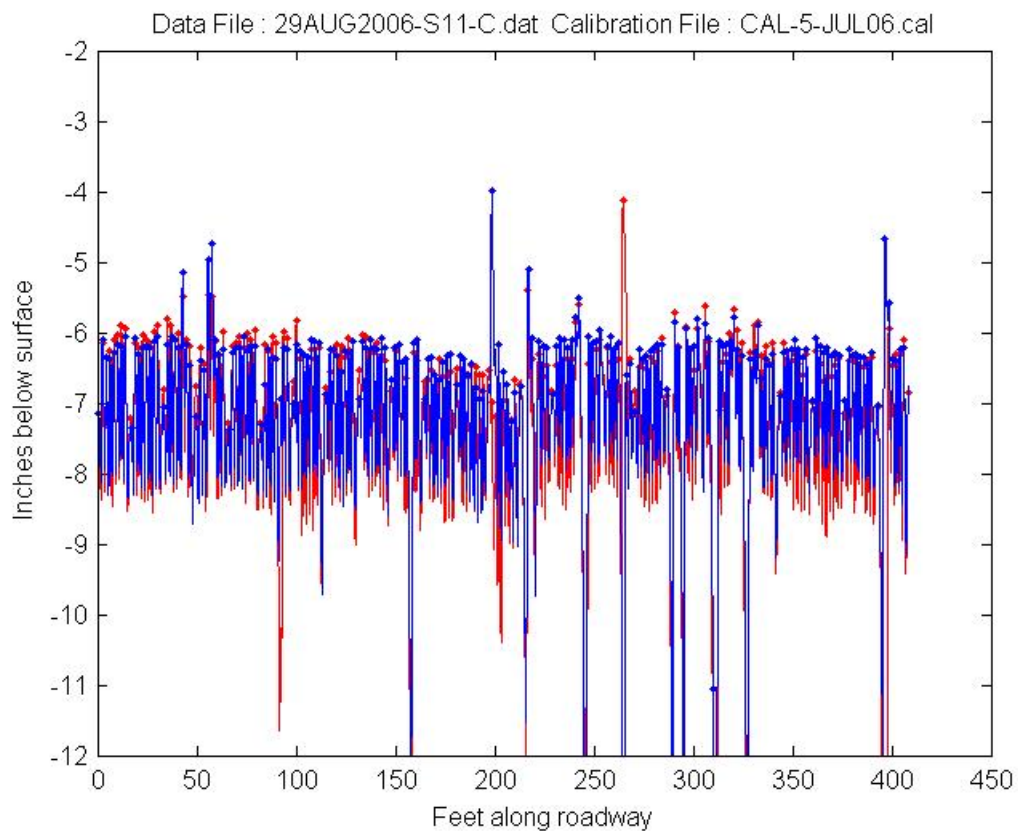


**Figure 6.8: Histogram of Right Sensor Bar Placement from I-70 Testing**

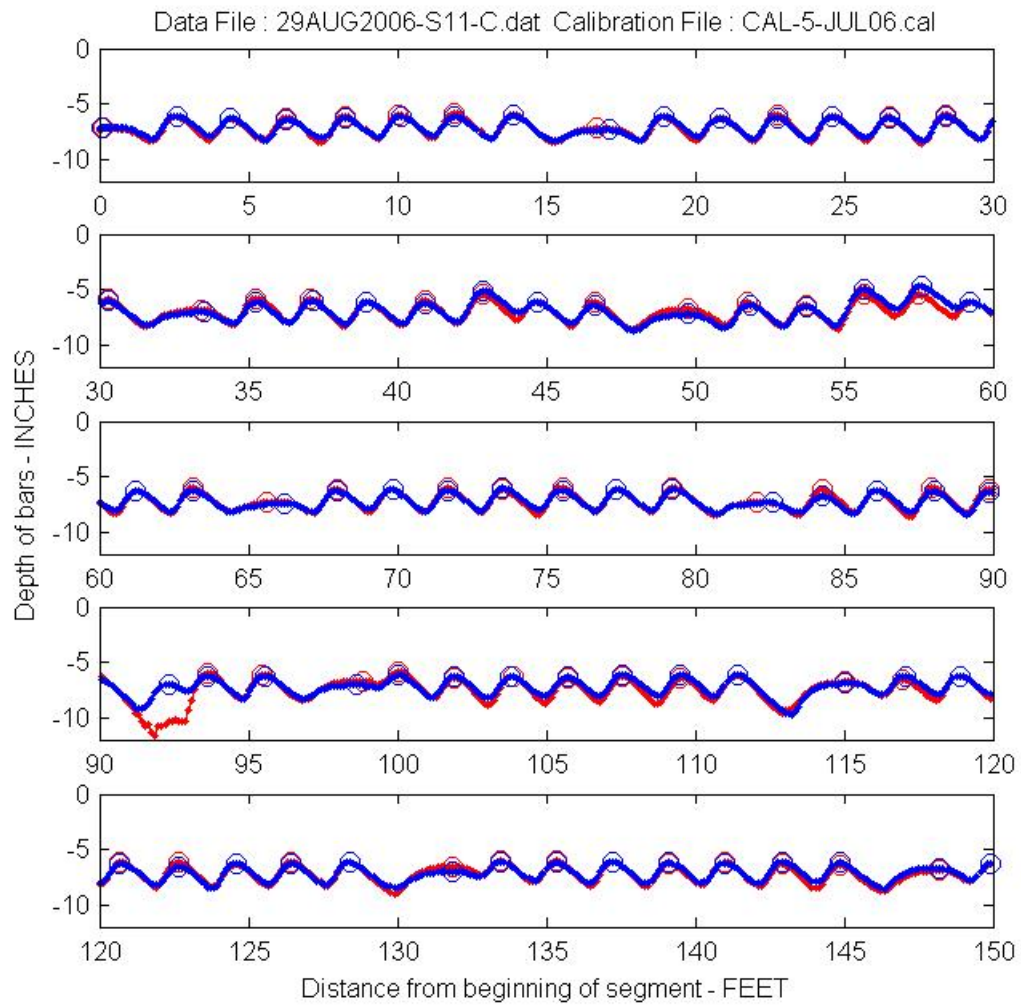


Figure 6.3 through Figure 6.8 show illustrations of good steel placement. The histograms show that the bars are consistently lower than optimal but are consistent. There are some anomalies in the data near the end with instances of missing bars such as around the 383-ft. mark. The data for the last 100 ft. seems to be getting worse, but the overall placement of the steel is acceptable. Details of the interpretation of these plots will be discussed later in further detail.

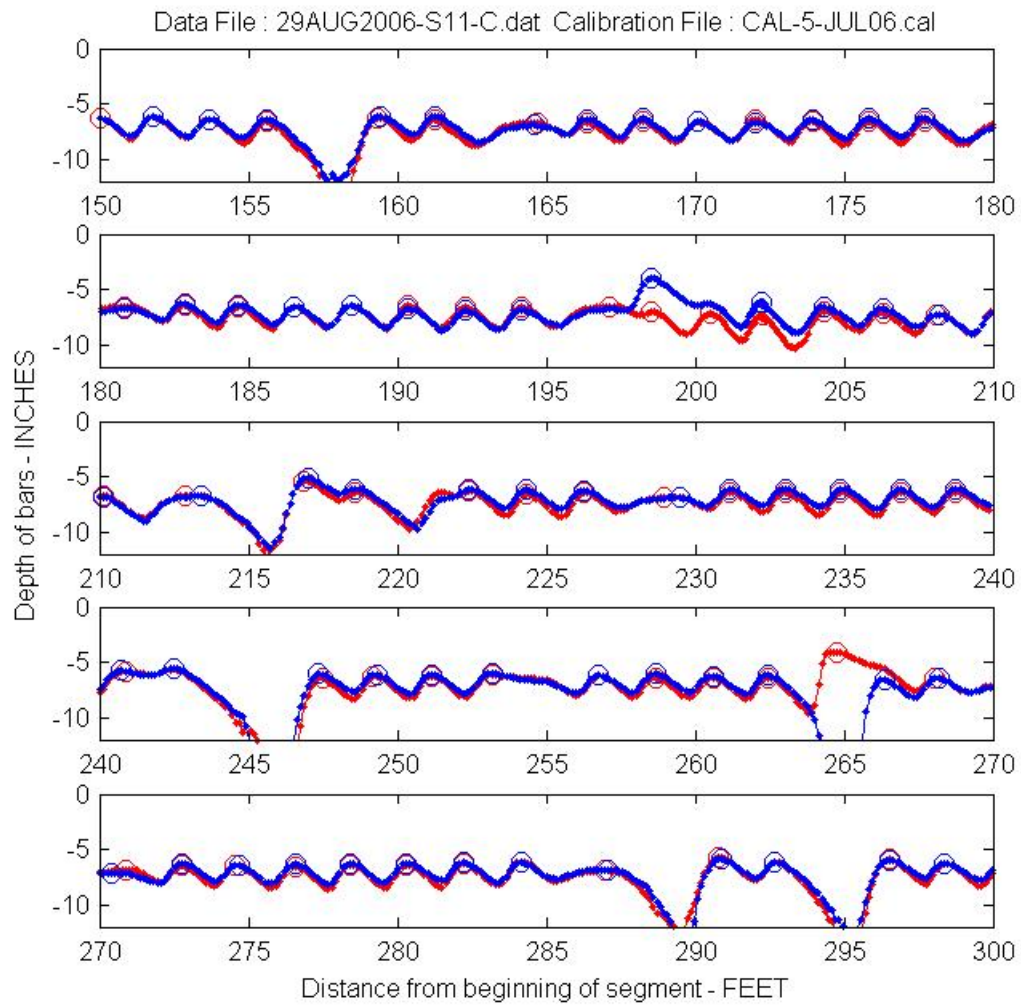
The second sample run was also taken on I-70 west of Abilene. Figure 6.9 through Figure 6.14 show the results of an almost ideal steel placement report.



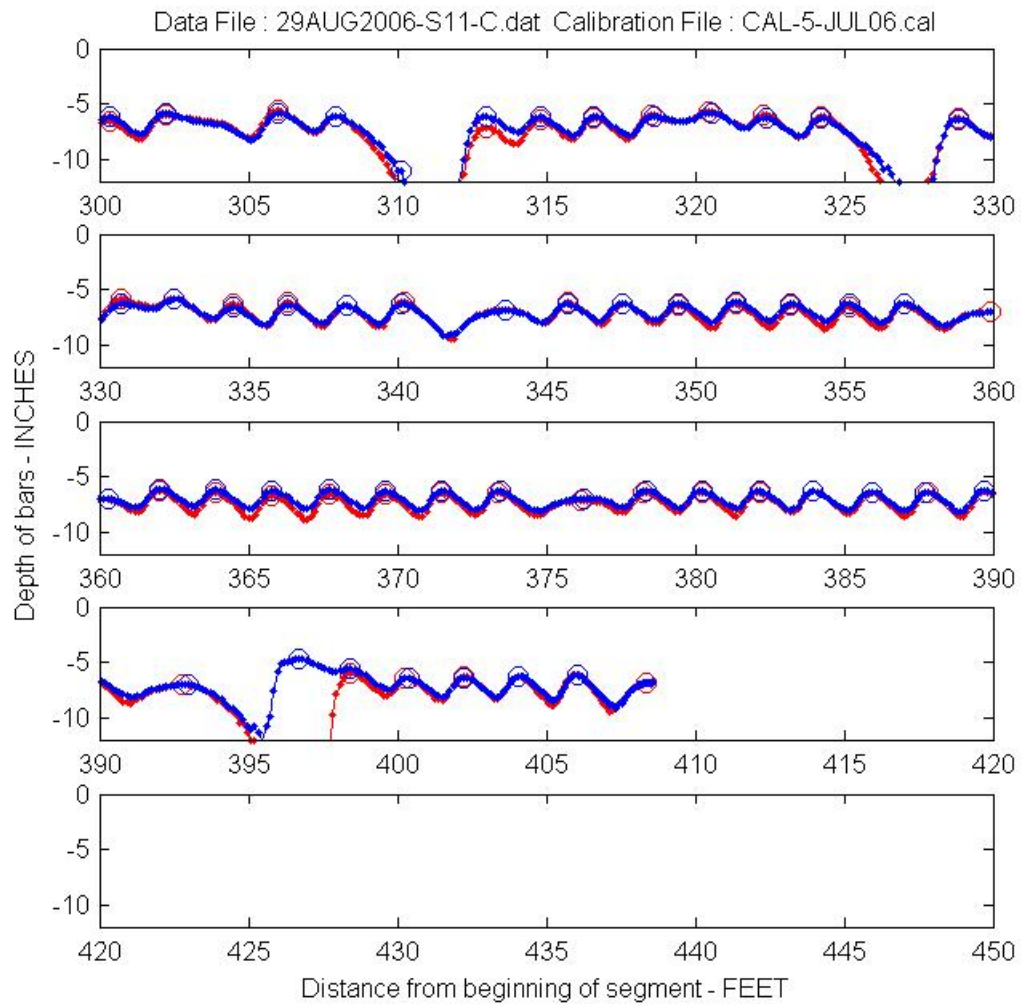
**Figure 6.9: Condensed Ideal Steel Placement Report**



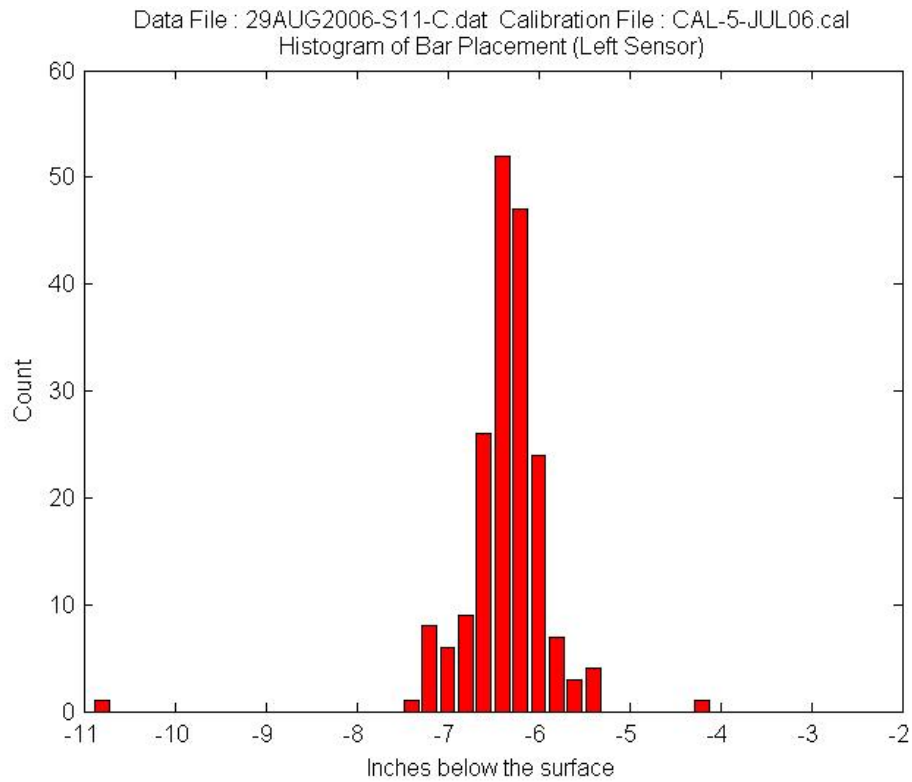
**Figure 6.10: Expanded Ideal Steel Placement Report (1/3)**



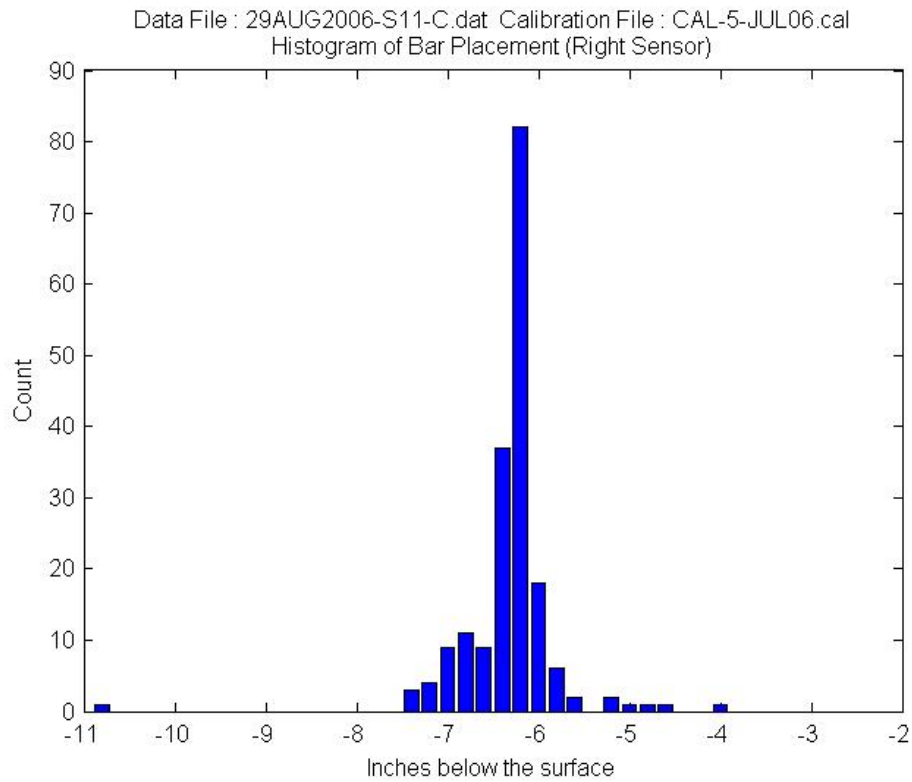
**Figure 6.11: Expanded Ideal Steel Placement Report (2/3)**



**Figure 6.12: Expanded Ideal Steel Placement Report (3/3)**



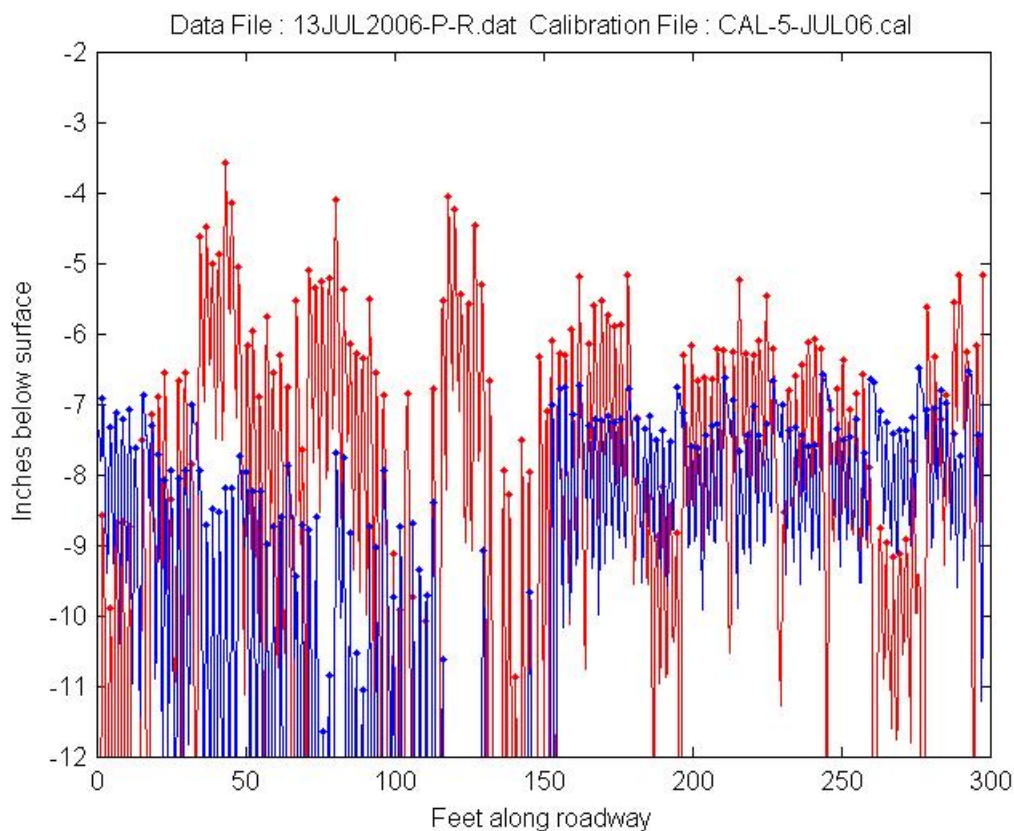
**Figure 6.13: Histogram of Ideal Steel Placement, Left Sensor**



**Figure 6.14: Histogram of Ideal Steel Placement, Right Sensor**

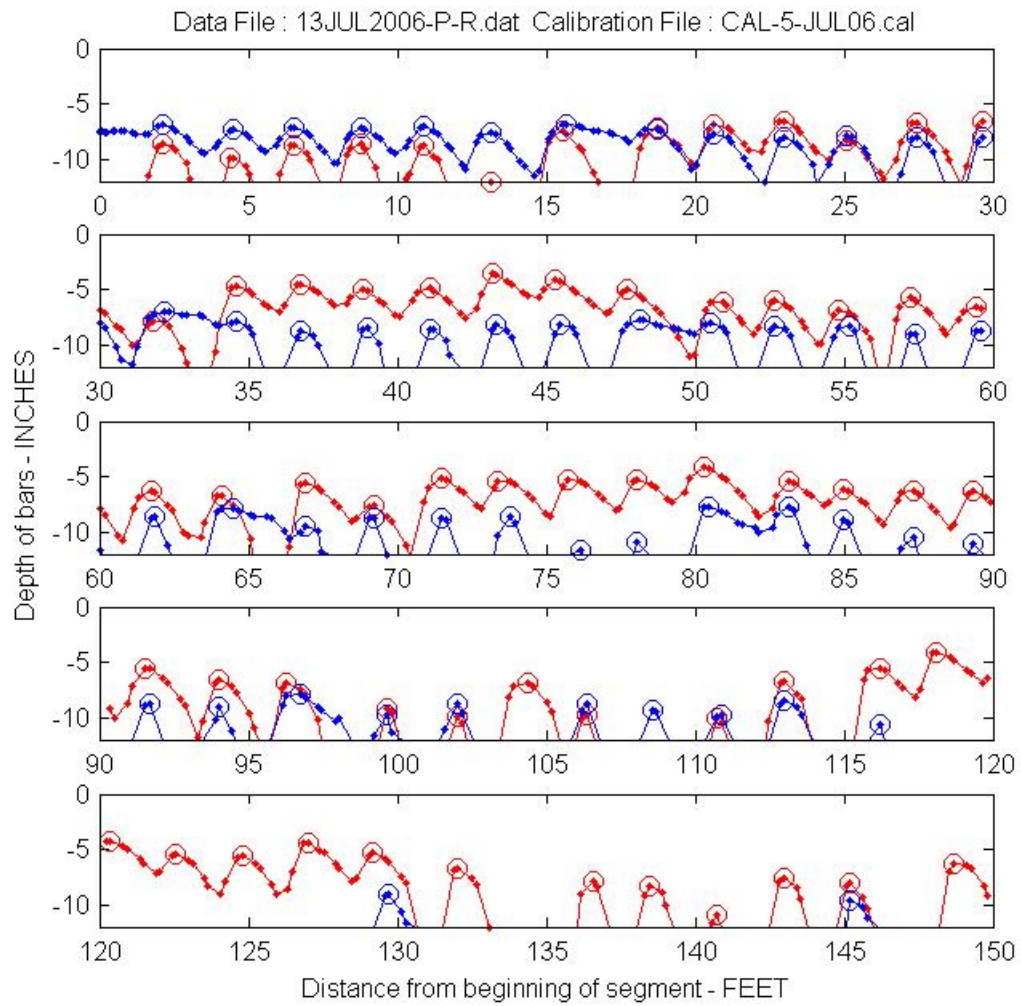
The traces in the trace plots lie right on top of one another. This is a sign that the bars are oriented correctly in the pavement. The histograms in Figure 6.13 and Figure 6.14 show that most of the bars reside just lower than 6 inches beneath the surface of the pavement. This is the ideal location for the reinforcement steel in the 12-inch-thick slabs that were measured [1].

The third sample run which shows the results from poorly placed steel was taken on U.S. Highway 69 north of Fort Scott, KS. Figure 6.15 through Figure 6.19 show the results of this data run.

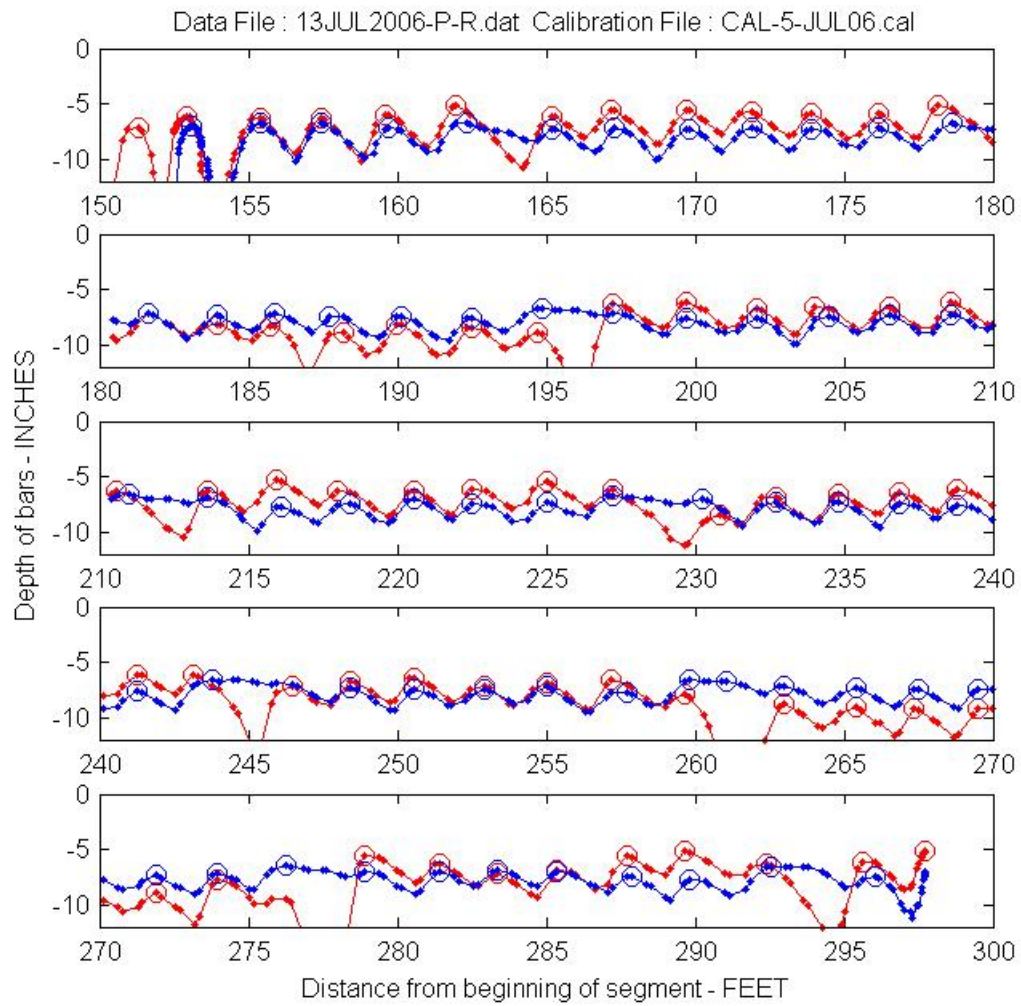


**Figure 6.15: Condensed Poor Steel Placement Report**



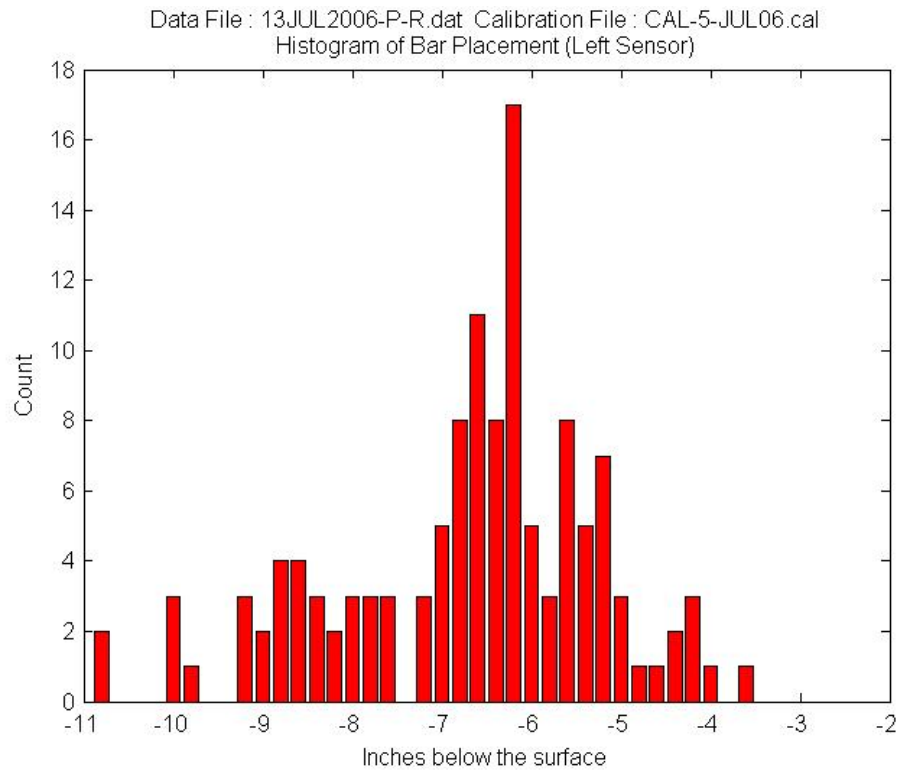


**Figure 6.16: Expanded Poor Steel Placement Report (1/2)**

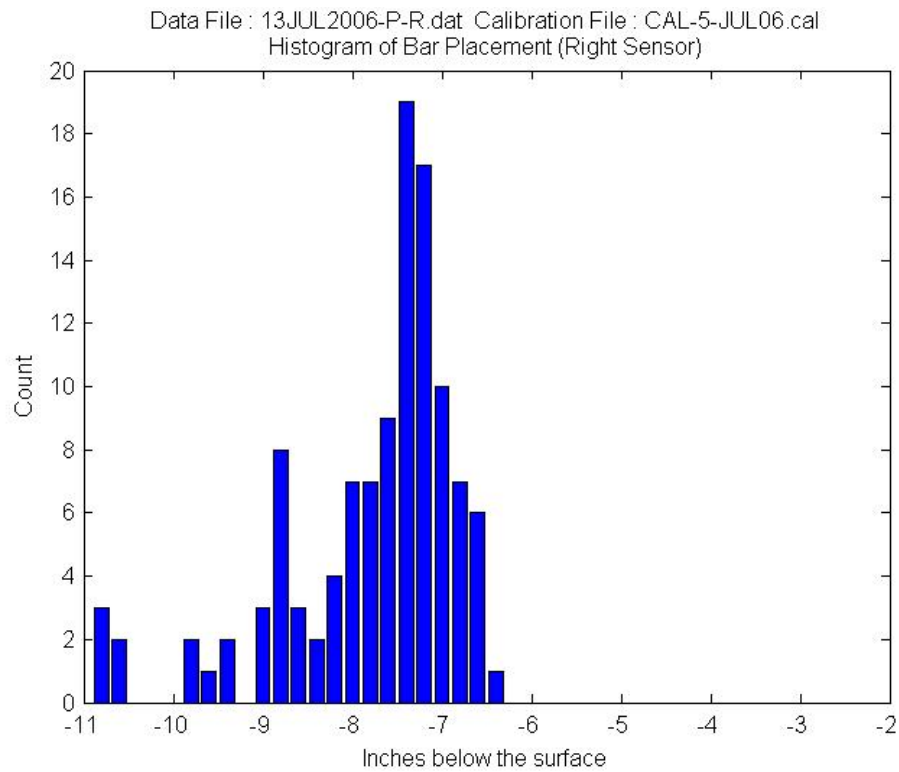


**Figure 6.17: Expanded Poor Steel Placement Report (2/2)**





**Figure 6.18: Histogram of Poor Steel Placement, Left Sensor**



**Figure 6.19: Histogram of Poor Steel Placement, Right Sensor**

The behavior of the measurements in this example is erratic, especially in the first 150 ft. This can be seen in the expanded trace plots. The peaks have drastically different heights which is a good indication that something is wrong with the steel at those locations. The histograms show that there is a large variance in the measured depths. This indicates that the placement is not uniform and is placed poorly.

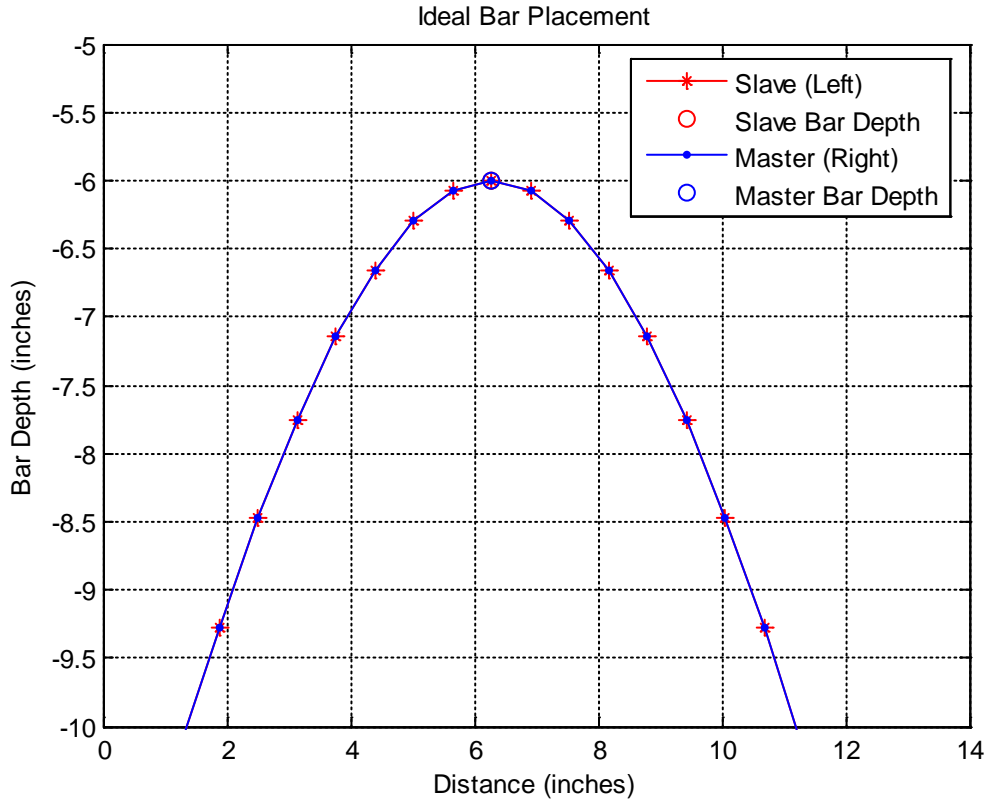
## **Result Interpretation**

Generated reports contain two main types of plots used as a graphical representation of the steel placement. The two types are trace plots and histograms. The following sections discuss the advantages of the two types of plots and the interpretation of the two.

### *Trace Plots*

The trace plots provide illustrations of the steel depth and orientation referenced to the distance measured by the optical encoders. This allows for all steel to be spatially referenced from the starting point and located for future testing if necessary. The key to interpreting the trace plots is the peaks in the waveform. The peaks represent the point where the sensor puck is closest to the steel which is the effective depth of the bar underneath the puck. The magnitude and phase differences between the two traces indicate the orientation of the bar. The plots in the following paragraphs were synthetically generated to give examples of orientation and depth interpretation for tie steel with the current two sensor version of the mapping device. Following the interpretation plots are examples of dowel basket interactions on tie bar measurement.

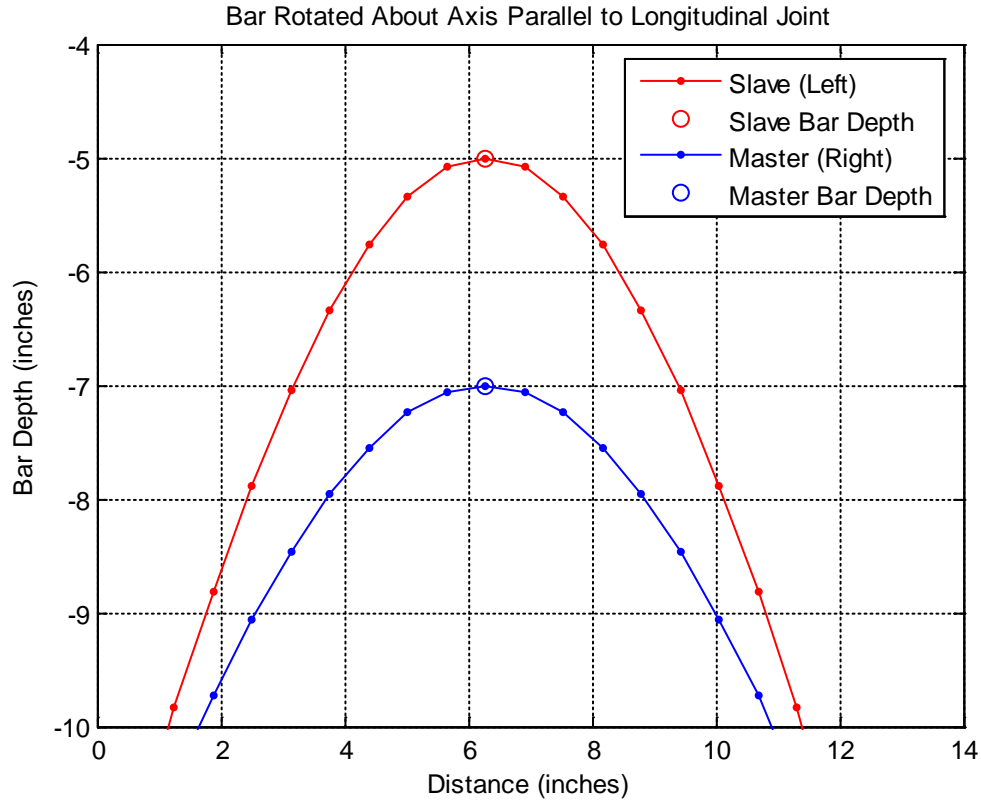
The first case for interpretation is the ideal case, where the bar is perfectly aligned and is located at the optimal depth. Figure 6.20 shows the trace peaks from an ideally placed bar.



**Figure 6.20: Ideal Bar Placement Example Plot**

The two traces perfectly overlap which indicates that the bar is parallel to the concrete surface and perpendicular to the transverse joint above the bars. A plot like this indicates that there is very little or no bar orientation error. The peaks of the two plots lie at a depth of 6 inches which is nominal for 12-inch concrete pavements. Peaks like this can occur at different depth levels. This means that the bar is oriented correctly but is either deeper or shallower than the optimal depth.

The second common case is when the bar is rotated about an axis parallel to the longitudinal joint. In other words, it is tilted up or down with reference to the earth. Figure 6.21 shows an example of this sort of orientation error.

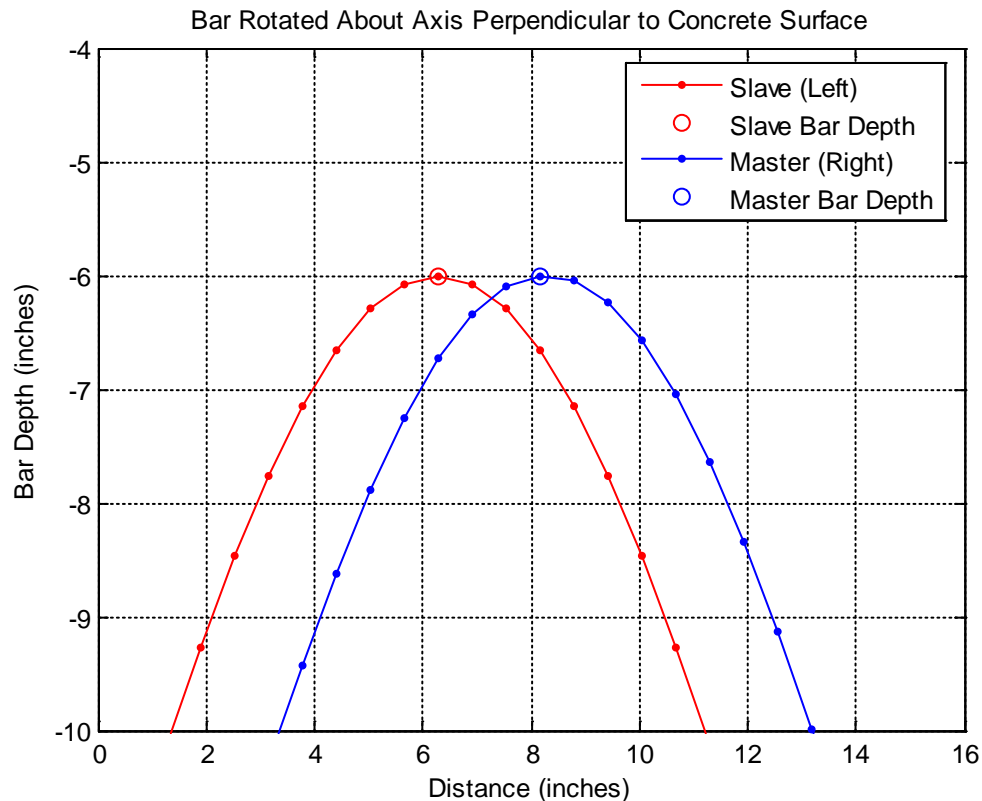


**Figure 6.21: Bar Rotated About an Axis Parallel to Longitudinal Joint Plot**

In this example, the peak from the left sensor resides at 5 inches and the peak of the right sensor is at 7 inches. This sort of return can result from two different problems. The first (and most likely) problem is that the bar is tilted vertically, or one end of the bar is higher than the other. If that were the case in this example, the center of the bar would reside at a depth of 6 inches and the bar would be slanted downward as you move from left to right. The left end of the bar would be at a depth of 5 inches and the right end would be at a depth of 7 inches. The second scenario is that the entire bar sits at a depth of 5 inches but is shifted to the left. This means that the left sensor is directly over the bar. The right sensor is at the end or past the end of the bar and records a smaller return because the measurement is being taken at an angle rather than directly below the puck. There is a possibility that both issues are having an effect on the readings and yielding the above results. The orientation ambiguity stems from the fact that absolute orientation cannot be determined from two traces. The two-sensor version of the mapping device can determine absolute orientation by making subsequent passes over the joint

with a lateral shift relative to the first run. This would result in four or more traces which would be sufficient to find absolute bar orientation.

The third and final common case occurs when the bar is rotated about an axis perpendicular to the surface of the road. In other words, the bar is tilted in the horizontal plane. Figure 6.22 shows an example of this sort of error.



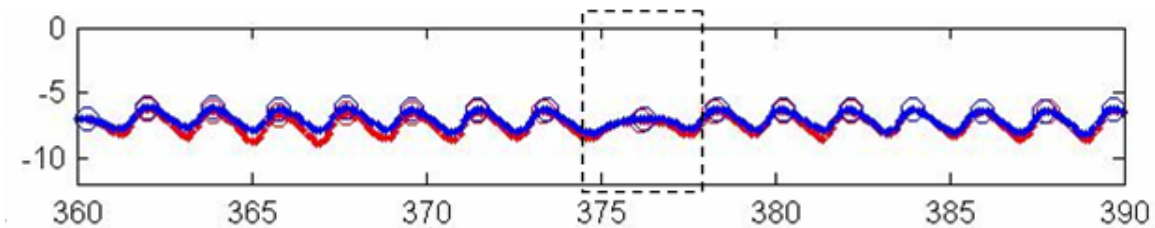
**Figure 6.22: Bar Rotated About an Axis Perpendicular to Concrete Surface Plot**

In this example both peaks are at the same height indicating that the bar is parallel to the surface of the concrete, but the peaks have a phase shift of 2 inches. This sort of return occurs when the left end of the bar is closer to the beginning of the run than the right end. In other words, the left end of the bar is sensed first followed by second bar 2 inches later. Another explanation for this sort of plot is that the bar is deformed. This occurs in the construction process along the exposed shoulder joint. The bars are set in the concrete in a 90-degree angle leaving half of the bar exposed. The bars are then bent out to be straight before pouring the final

shoulder. Sometimes the bars do not get bent perfectly straight, which shows up as a phase shift in the peaks of the plots.

The previously mentioned scenarios form the basis for interpreting the output plots found in the reports. Typical data runs include examples of all of these types of returns and oftentimes have returns that are combinations of these simple cases. Complex cases involving two or more of the above examples can be easily broken down by using the fundamentals mentioned above to decipher the meaning of the returns.

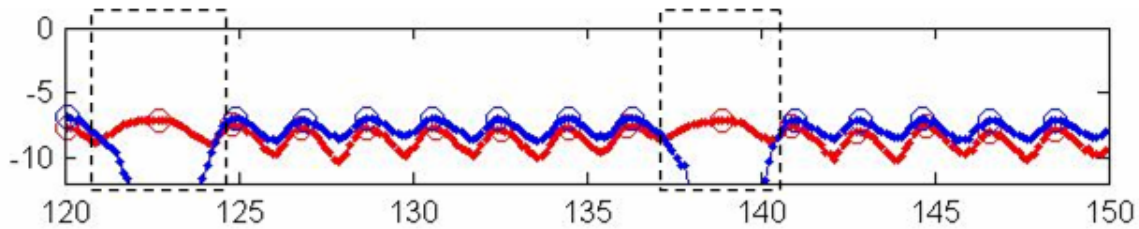
Other artifacts in the trace plots are caused by the interactions of dowel baskets with the magnetic sensing devices. These artifacts have a distinct shape and do not adhere to the previously discussed rules. Figure 6.23 and Figure 6.24, taken from data collection runs on I-70 west of Abilene, KS, show the effects of dowel baskets on the reported trace plots.



**Figure 6.23: Dowel Basket Interaction Along a Center Joint**

The dashed box denotes the return associated with a pair of dowel baskets, one on the right and one on the left. This return characteristic occurs only on the center joint because there are baskets on both sides of the longitudinal joint. The trace shape is typically wider than the characteristic tie bar shape and occurs at regular intervals during the data collection run. In this example, there are 7 tie bars on 24 inch centers in each slab, with the slab beginning and end marked by the dowel basket returns seen at the transverse joints.

The dowel basket interaction on shoulder joints is slightly different than the center joint. Figure 6.24 shows an example of a typical dowel basket artifact found in a data collection run.

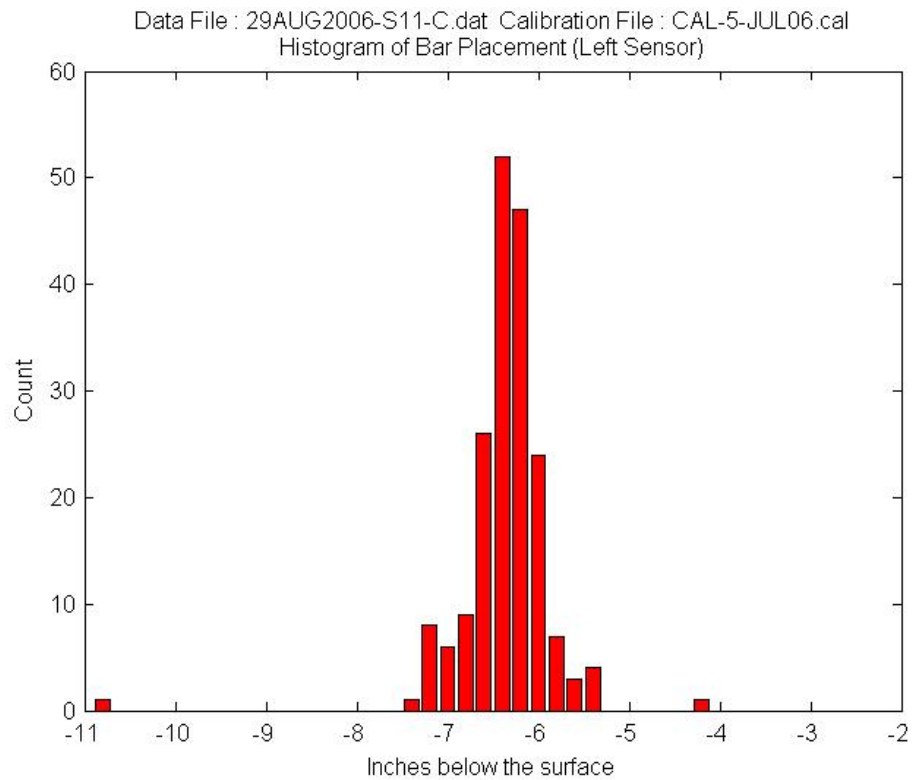


**Figure 6.24: Dowel Basket Interaction Along a Shoulder Joint**

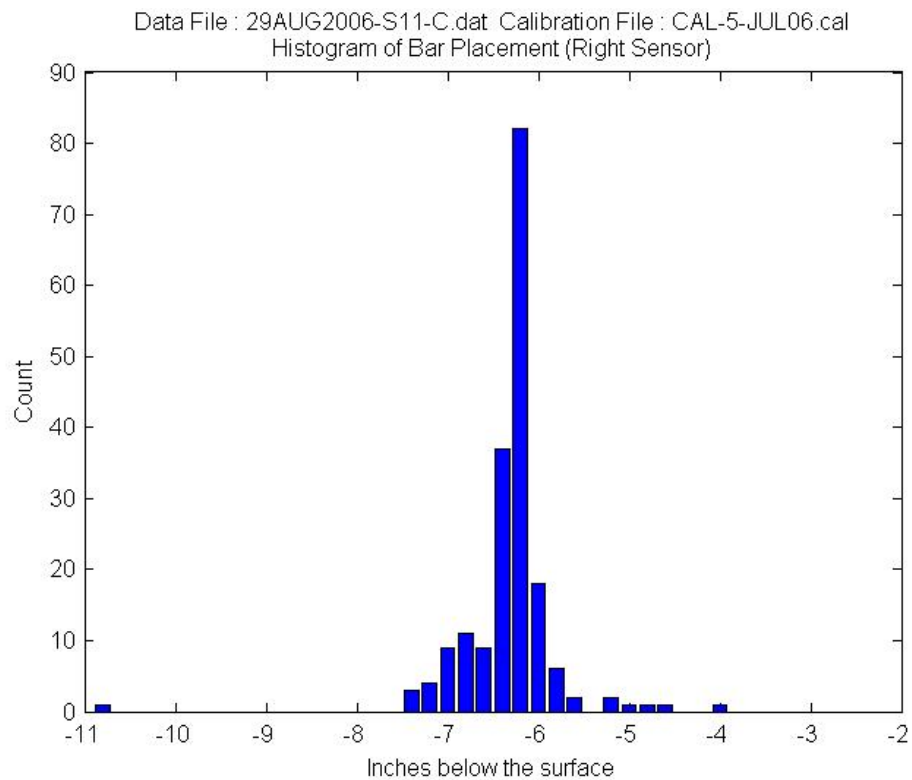
Instead of both traces forming the characteristic dowel basket hump, only one trace shows a return and the other drops off of the plot. This occurs because there are no dowel baskets in the shoulders, so only the sensor nearest to the center of the roadway will pick up a return from a dowel basket. These dowel basket returns are typically marked as though there is a tie bar there. This occurs because the bar finding algorithm is simple and does not take dowel basket interactions into account. It is important to keep this in mind when interpreting histograms, which are covered in the next section.

### *Histograms*

A second type of plots created by the report generation process is histograms for each data collection channel. The histograms provide a visual representation of the statistical properties of the steel placement of the entire data collection run. These plots present the bar placement in a way that makes characterizing the entire data run simple without having to look through pages of trace plots. In effect, the histograms are summaries of the bars found by the reporting software. Figure 6.25 and Figure 6.26 show an example of histograms from a longitudinal joint with well-placed steel.



**Figure 6.25: Example Histogram from Left Sensor**



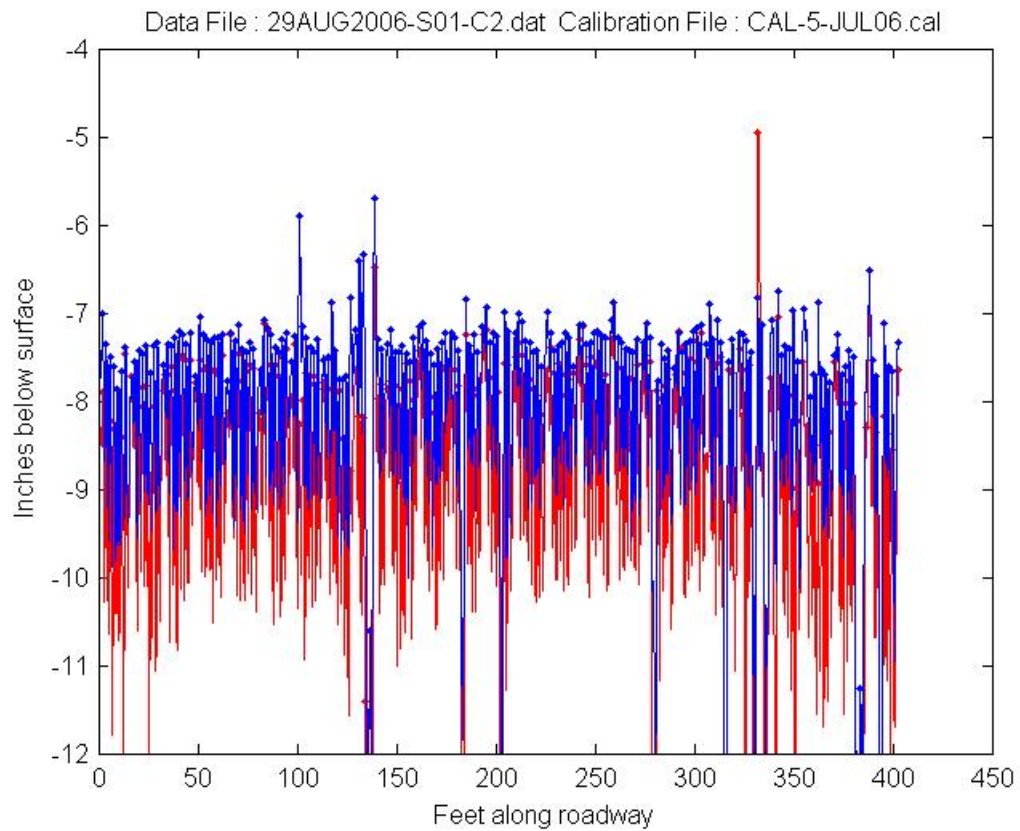
**Figure 6.26: Example Histogram from Right Sensor**



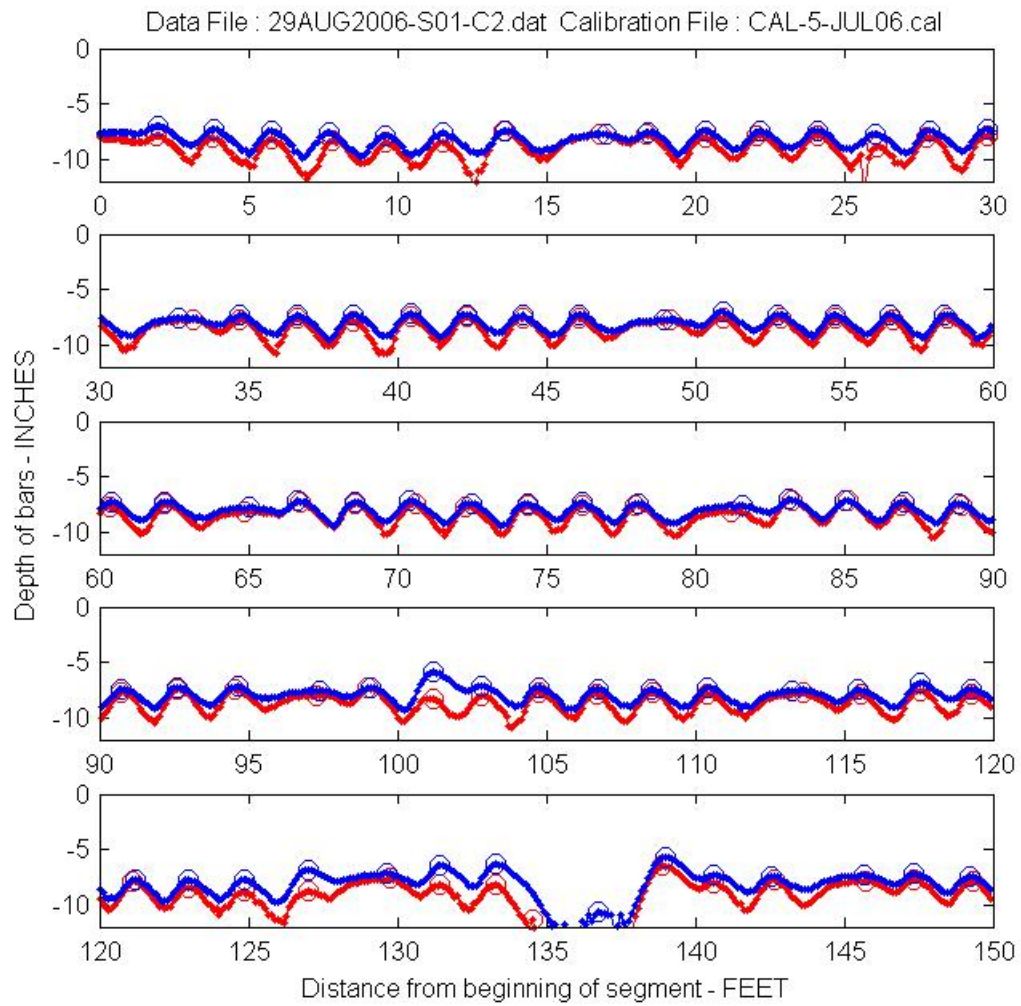
The typical distribution of the histograms is Gaussian with random placement variations caused by minor insertion irregularities, viscosity and composition of the concrete, and bar settling due to the vibration process. The ideal plots would be spikes at 6 inches below the surface which would be a Gaussian distribution with a mean of six and a variance of zero. The histograms also give a sense of steel uniformity. This can indicate whether there has been an irregularity of the steel inserter operation. General trends in the steel placement can also be seen from these plots. If the mean value of the right sensor is lower than that of the left sensor, it can be gathered that most of the bars are either tipped, shifted to the left, or a combination of both. Distributions other than Gaussian have been observed. These cases require more study of the trace plots to determine the specifics of the steel placement. Outliers like those seen in Figure 6.25 and Figure 6.26 at depths of 4 and 11 inches sometimes result from the effects of transverse dowel joints. In the case for these figures, the outliers were caused by dowel joints. Deviations from the Gaussian distribution are usually a result of steel inserter mechanical failure.

## **Repeatability**

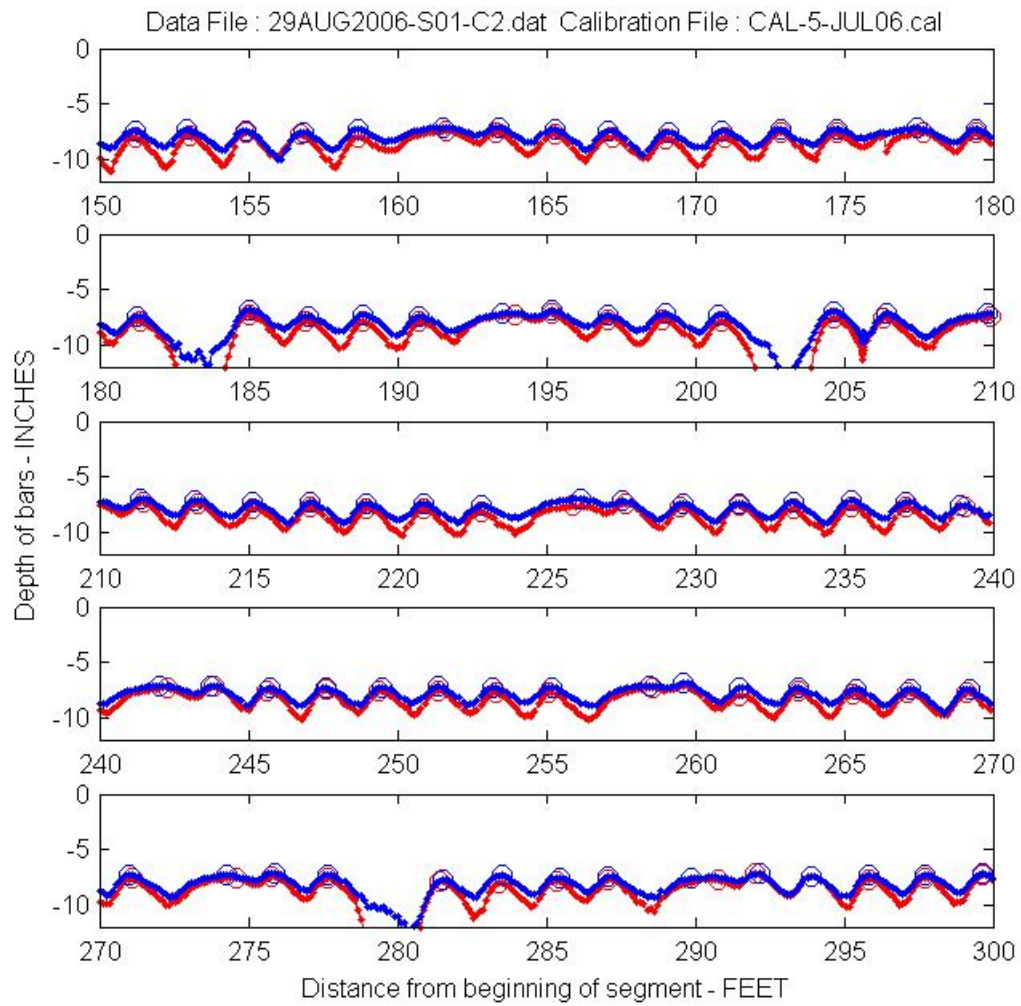
A continuing theme during the design and testing of the mapping device was repeatability of findings. This was particularly true for the project sponsors and potential customers of the device. Occasional tests were run over the same joints to ensure that readings could be replicated to an appropriate degree. Figure 6.27 through Figure 6.32 show the results of a second run that was taken of the first joint discussed in the Sample Runs section.



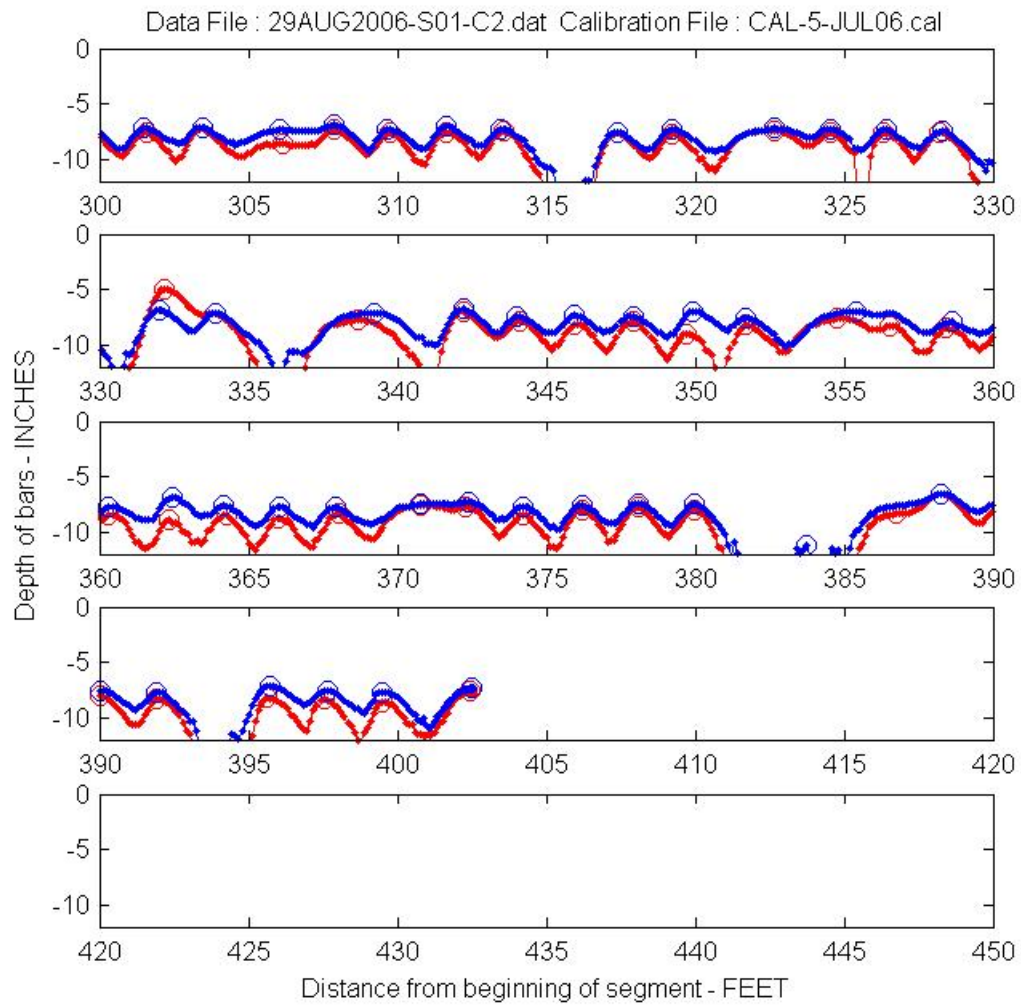
**Figure 6.27: Repeatability Example of Condensed Plot of I-70 Center Joint**



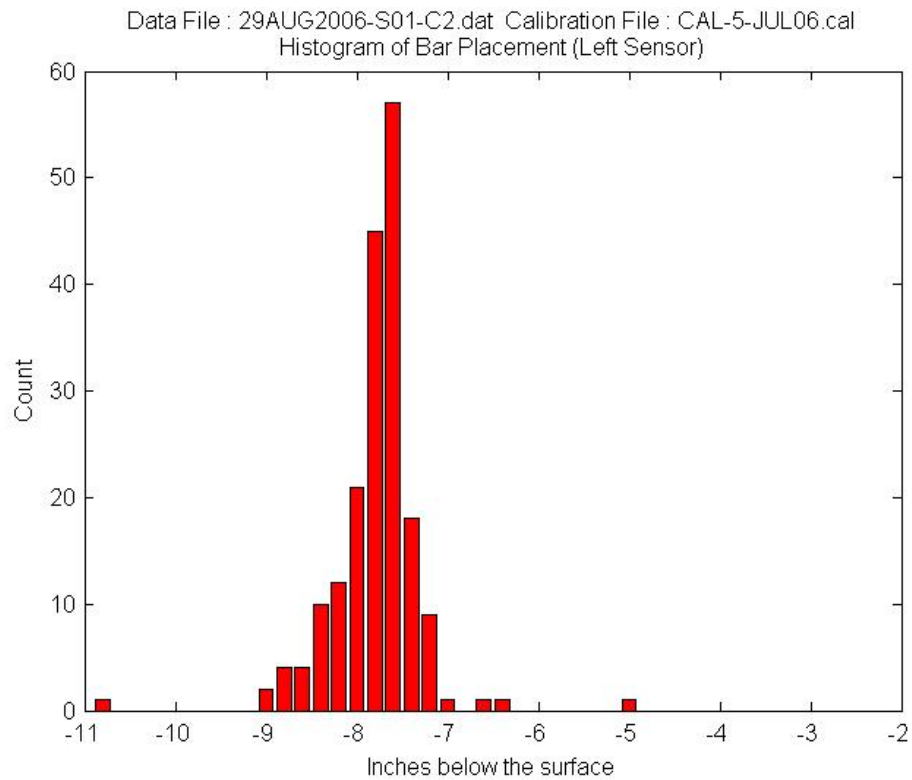
**Figure 6.28: Repeatability Example of Expanded Plot of I-70 Center Joint (1/3)**



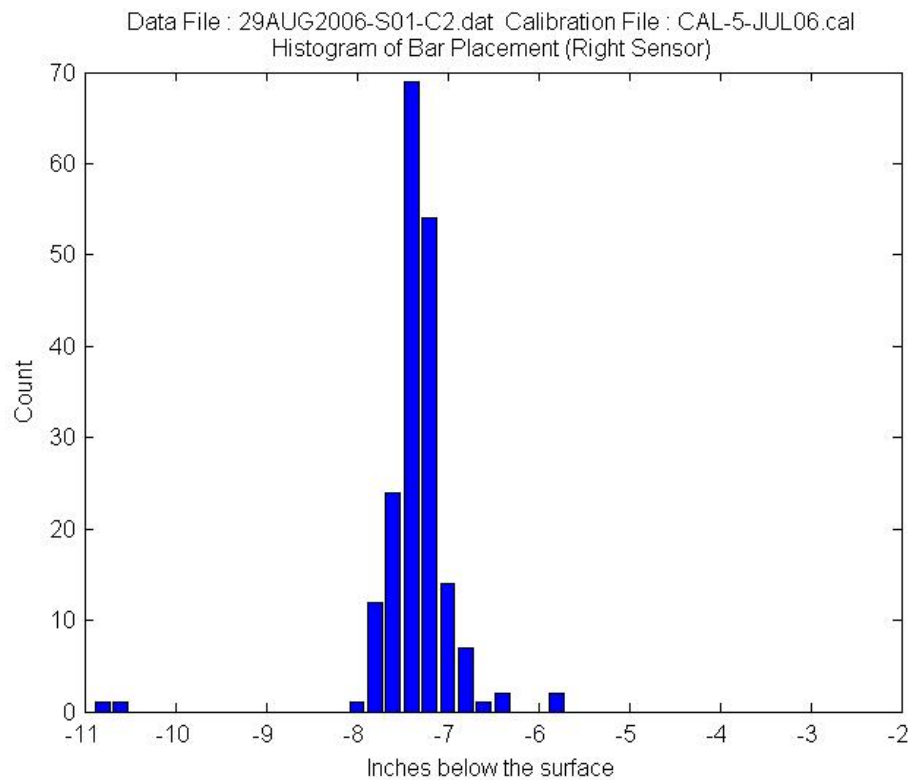
**Figure 6.29: Repeatability Example of Expanded Plot of I-70 Center Joint (2/3)**



**Figure 6.30: Repeatability Example of Expanded Plot of I-70 Center Joint (3/3)**



**Figure 6.31: Repeatability Example of Left Sensor Histogram**



**Figure 6.32: Repeatability Example of Right Sensor Histogram**

The trace plot irregularities can be seen easily in the condensed plot for some of the outlier peaks and large features. The features that stand out, those above a depth of 7 inches, are repeated in both instances of the plot. The expanded plots show similar steel placement between the two runs with minor differences most likely caused by deviation from the center of the joint during the run. The histograms are also good indicators of how well the device performed when compared with the earlier run. Visual inspection shows that both sets of histograms exhibit the same distributions, as should be expected from a run over the same joint. Minor differences in the trace plots and histograms are caused mainly by deviation from the center of the joint during data collection. This pair of compared reports testifies to the precision of the device and the ability for it to generate reliable results.

## Chapter 7: Future Work

This research provides a solid foundation for planned commercialization of the device and for future improvement and modification of the reporting and control software. The current two-sensor version of the device provides plots that give a general idea of the steel reinforcement orientation. Adding additional sensors will allow for a single pass to generate enough data for determining absolute bar orientation and bar deformation. The limited proprietary information available about the customized covermeter electronics prevents modifying the existing hardware to work with five sensors. Wiring the devices together without acquiring further knowledge about the electronics is not prudent based on the potential for overloading output pins from the master board and causing damage to the equipment.

Another extended application is taking the current cart platform and porting it to a system that can be mounted behind a paver. This modification would allow the paver's operator to check the performance of the steel inserter in real-time so that a misplaced bar or equipment malfunction would have a minimal effect on their operation and would virtually eliminate costly remedial actions such as 'stitching.' The data collection would also allow paving companies to keep a record of steel placement as proof of compliance with regulations and stipulations of the paving contract. The device measures and records data at a far greater rate than a person can by hand with a single covermeter and yields a more accurate representation of the steel placement. This extension could also be used in a feedback loop that would control the inserter depth. Data from the device would be automatically transformed into commands to make minor corrections to the mechanical operation of the inserter to yield higher precision steel placement.

The main user interface and data acquisition software was originally authored for a single-sensor version of the device that did not utilize a microprocessor as a communications interface. The reuse of the software led to many artifacts of previous versions that clutter the user interface and cause the program to not run as efficiently as it could. Rewriting the program in a different, preferably faster, language would allow for the user interface to be cleaner and easier to use and would provide more real-time functionality to the software. An embedded approach would be faster and more capable of handling the data real-time. It would also eliminate the need



for a laptop running the Windows operating system, which is not well suited for real-time software. Battery life of the laptop is currently the limiting factor of how long an operator can take data in the field. Additional plots such as a color intensity chart showing the orientation and depth of the bars would be a valuable addition to the software. The bar finding algorithm contained in the reporting software could also be enhanced to filter out the effects of dowel basket interaction in the collection runs.

A substantial part of the cost associated with the device is the customized covermeters provided by Koletric. Additional research on magneto-resistive sensors and other solid-state devices is needed so that the proprietary covermeter electronics and coils can be replaced by much cheaper components. The solid-state magnetic sensors have a fundamental difference in their measurement techniques. Coils sense the change ( $d/dt$ ) of magnetic fields and the magneto-resistive sensors detect static magnetic fields. Preliminary testing showed that bar magnetization caused significant errors when trying to sense steel with the solid-state sensors. Testing also revealed that the sensors may not be capable of being subjected to the large send magnetic field used to generate the magnetic response. This was evident in the magnetic film changing polarity and measurement accuracy degrading over time. Using solid-state sensors is advantageous since they are smaller and can potentially be placed in a higher density grid than wire coils can. The interference caused by one magnetic sensor on others has yet to be determined. This could be the limiting factor in how dense a sensor array could be. An array configuration would generate larger amounts of data for less cost and would yield plots with much higher resolution. This higher resolution would allow for creating a tremendously accurate 3-D model of the orientation and deformation of the reinforcement steel. Having control of the technology involved would also allow for greater flexibility in scaling the device to higher numbers of sensors for different applications.

## Chapter 8: Conclusions

The research discussed in this report outlines the development of a multiple-sensor device for mapping the depth, location, and orientation of reinforcement steel in concrete pavement. Challenges with multiple sensor interference and calibration were encountered during the course of the development. Multi-sensor calibration measurements were taken and used to generate a polynomial that translated the magnitude of a magnetic sensor return into a measure of steel depth. Sensor drift in the covermeters caused errors in the translated bar depth readings and required that an on-the-fly calibration technique be developed and implemented. An embedded communications interface using the MPC555 microprocessor was designed to enhance device scalability and extensibility. This interface also alleviated the computational load on the Toughbook which allowed the data acquisition and real-time feedback portions of the user interface application to run faster.

Initial experimentation showed that relative movement between the sensor pucks resulted in DC shifts in the sensor readings that were detrimental to the accuracy of the calculated depth readings. The solution to this problem involved fabricating a rigid, non-metallic enclosure to lock the pucks in place relative to one another. This structure also allowed for taking the sensor bar out of the tray of the cart to perform reliable calibrations.

Calibration of the device required some effort due to multiple factors. The first was finding a way to reliably take distance measurements. The solution to this problem was creating a wooden calibration jig with slots cut at predetermined depth values. These values were then used to generate a second-order polynomial that could translate between raw sensor data and the depth of the measured bar. The second issue was that of the puck ferromagnetic cores breaking loose from their epoxy containment. This caused the cores to shift during on-the-fly calibrations, thus causing erroneous infinity calibration readings and incorrect depth calculations. The last problem was that of the sensor drift over time. This phenomenon was first observed during multiple hour stability tests in the lab. Experiments were run to test the effects of temperature, direct sunlight, and puck orientation. No conclusions could be gleaned from the results of these tests. The solution to the drift problem was to incorporate the on-the-fly calibration routine into

the data acquisition and reporting process. The user interface software requires that an on-the-fly calibration be done at the beginning and end of every data collection run. This allows the reporting software to linearly interpolate between the two values to generate depth readings that are as accurate as possible.

The embedded MPC555 processor allowed for the elimination of costly third-party electronics. The processor handled the input of the two optical encoders using the fast quadrature decode functionality in the TPUB sub processor. The MPC555 also controlled all serial communication between the peripheral devices and the user interface computer. This allowed all the serial communications from the covermeters and the GPS receiver and the distance values calculated from the encoder pulses to be combined into a single serial channel. This is advantageous from the standpoint of having a remote computer controlling the system in the future. The serial cable can be replaced by a wireless link which would allow the controls to be in the cab of a truck or on top of the paving machine. The MPC555 also offers scalability options if more covermeters are added in the future.

The research resulted with a viable product that could potentially have substantial market value. The device operates as desired and has features that incorporate real-time data visualization, post data collection reporting routines, and procedures for geospatially linking the generated reports. This level of refinement involved detailed exchanges between the developers and the research sponsors about what would make the device attractive for routine use on the construction site. At the time of writing this report, work was being done with the Advanced Manufacturing Institute of Kansas State University for possible marketing and production of the mapping device.

## References

- [1] Kansas Department of Transportation, “RD708 - Concrete Pavement Dowel Jointed Non-Reinforced,” June 13, 2005.
- [2] Kansas Department of Transportation, “RDXXX – Concrete Pavement Dowel Bar Spacing Tolerances Along Transverse Joint,” January 3, 2001.
- [3] Kansas Department of Transportation, “RDXXX – Concrete Pavement Tie Bar Spacing Tolerances Along Conversion Joint,” January 3, 2001.
- [4] ELE International, Inc., “Operating Instructions Micro Covermeter Model: 35-2022 (CT-4950A).” [http://www.eleusa.com/instructions/35-2022\(CT-4950A\).pdf](http://www.eleusa.com/instructions/35-2022(CT-4950A).pdf)
- [5] DeVault, James E., Miller, Ruth D., “A Field Verification Instrument to Assess the Placement Accuracy of Dowel Bars and Bars in PCCP,” K-TRAN: KSU-03-1, October 2005.
- [6] Magnetic Imaging Tools, <http://www.mit-dresden.de/> (in German), June 26, 2001.
- [7] U.S. Department of Transportation Federal Highway Administration, “Use of Magnetic Tomography Technology to Evaluate Dowel Placement,” June 1, 2006.  
<http://www.fhwa.dot.gov/pavement/concrete/mitreport/mits03.cfm>
- [8] Young, Stanley E., “Steel Locator Device: Project Review & Implementation Timeline,” Missouri/Kansas American Concrete Pavement Association, March 1, 2006.
- [9] Panasonic, “Toughbook 29,” August 2006.  
[ftp://ftp.panasonic.com/pub/panasonic/toughbook/specsheets/TB-29\\_ss.pdf](ftp://ftp.panasonic.com/pub/panasonic/toughbook/specsheets/TB-29_ss.pdf)
- [10] Garmin International, Inc., “GPS 15H & 15L Technical Specifications,” February 17, 2006. [http://www.garmin.com/manuals/237\\_TechnicalSpecifications.pdf](http://www.garmin.com/manuals/237_TechnicalSpecifications.pdf)
- [11] Kitchener, Dave, “Product Specification PS0178-001 Issue 0A Synchronised Proba3D Units,” Tallix Embedded System Consultancy, March 29, 2005.
- [12] Freescale Semiconductor, Inc., “MPC555/MPC556 User’s Manual,” March 15, 2006.  
[http://www.freescale.com/files/microcontrollers/doc/user\\_guide/MPC555UM.pdf](http://www.freescale.com/files/microcontrollers/doc/user_guide/MPC555UM.pdf)
- [13] Freescale Semiconductor, Inc., “Fast Quadrature Decode TPU Function (FQD),” 2004.  
[www.freescale.com/files/microcontrollers/doc/app\\_note/TPUPN02.pdf](http://www.freescale.com/files/microcontrollers/doc/app_note/TPUPN02.pdf)
- [14] Freescale Semiconductor, Inc., “Asynchronous Serial Interface TPU Function (UART),” 2004. [www.freescale.com/files/microcontrollers/doc/app\\_note/TPUPN07.pdf](http://www.freescale.com/files/microcontrollers/doc/app_note/TPUPN07.pdf)

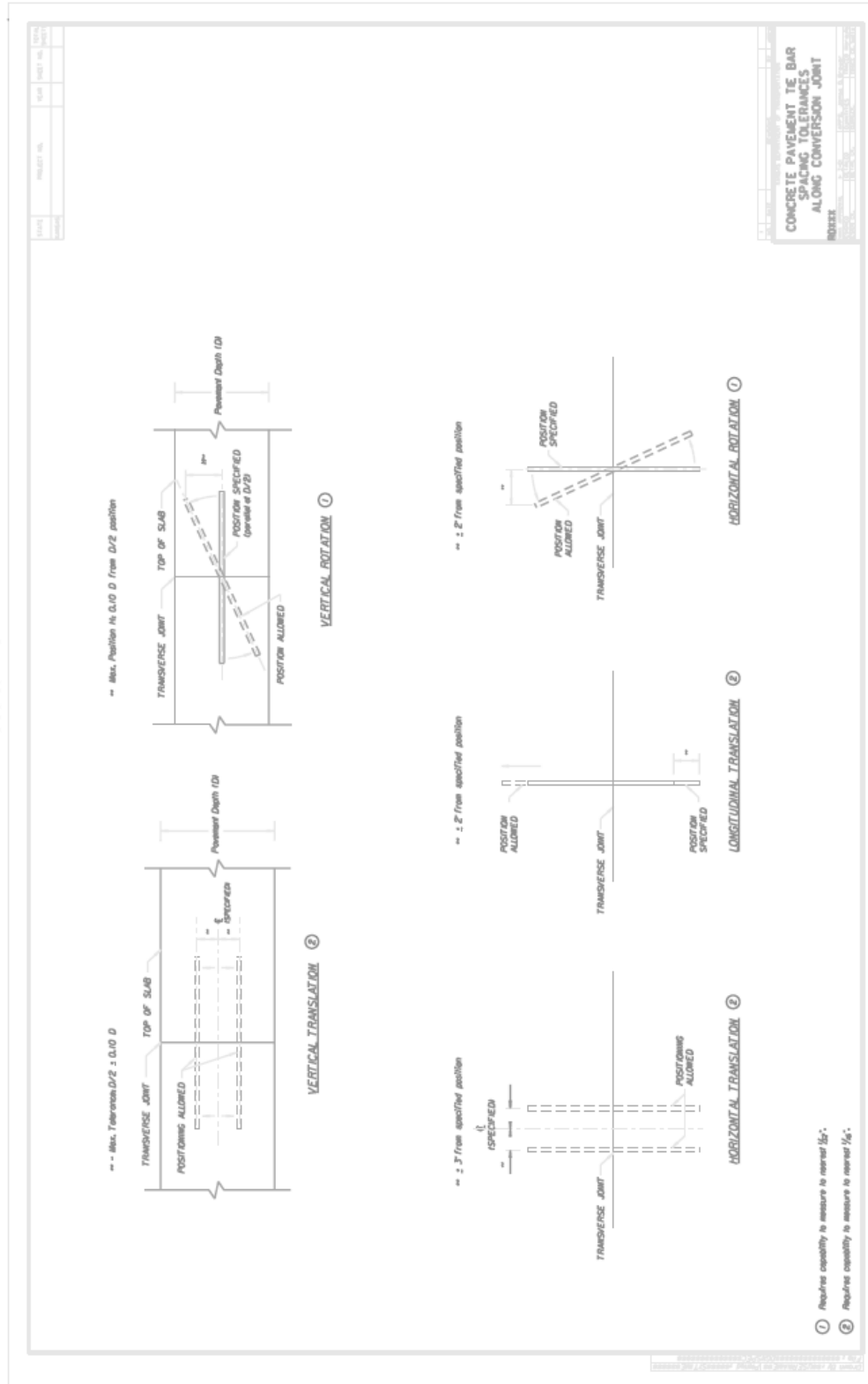


## **Appendix B: Proposed Kansas Concrete Construction Specifications**

See next page.



Scale: NTS



Reproduced from [3]

Figure B.2: KDOT Conceptual Tie Bar Specification



## Appendix C: Embedded Source Files

### controlAux.c

```
//-----  
// File:          controlAux.c  
// Description:   Source file that holds the helper  
//               functions used by the main control  
//               loop to handle digital I/O and  
//               the generation of ASCII coded  
//               hexadecimal values for the  
//               encoder output.  
// Author:        Nathan Holle  
//               Kansas State University  
// Date:          23JUN06  
//-----  
#include "controlAux.h"  
  
// initialize the digital I/O pins for  
// controlling the covermeters  
void initIO() {  
  
    // set pin 5 of the MIOS1 general  
    // purpose I/O to be an output  
    // MASTER_OUT  
    MIOSDDR |= 0x1 << MASTER_OUT;  
  
    // set pin 5 to a '1'  
    MIOSDR |= 0x1 << MASTER_OUT;  
  
    // set pin 7 of the MIOS1 general  
    // purpose I/O to an output  
    // SLAVE1_OUT  
    MIOSDDR |= 0x1 << SLAVE1_OUT;  
  
    // set pin 7 to a '1'  
    MIOSDR |= 0x1 << SLAVE1_OUT;  
  
    // set pin 9 of the MIOS1 general  
    // purpose I/O to be an output  
    // SLAVE2_OUT  
    MIOSDDR |= 0x1 << SLAVE2_OUT;  
  
    // set pin 9 to a '1'  
    MIOSDR |= 0x1 << SLAVE2_OUT;  
  
    // set pin 6 of the MIOS1 general  
    // purpose I/O to be an input  
    // MASTER_IN  
    MIOSDDR &= ~(0x1 << MASTER_IN);  
  
    // set pin 8 of the MIOS1 general  
    // purpose I/O to be an input  
    // SLAVE1_IN  
    MIOSDDR &= ~(0x1 << SLAVE1_IN);  
}
```

```

    // set pin 10 of the MIO1 general
    // purpose I/O to be an input
    // SLAVE2_IN
    MIO1DDR &= ~(0x1 << SLAVE2_IN);
}

// ensure that the covermeters are turned
// off, regardless of their current state
void clearMeters() {

    // bring outputs low
    MIO1DR &= ~(0x1 << MASTER_OUT);
    MIO1DR &= ~(0x1 << SLAVE1_OUT);
    MIO1DR &= ~(0x1 << SLAVE2_OUT);

    // wait for 1 second
    pauseNSeconds(1);

    // force outputs high
    MIO1DR |= 0x1 << MASTER_OUT;
    MIO1DR |= 0x1 << SLAVE1_OUT;
    MIO1DR |= 0x1 << SLAVE2_OUT;

    // wait for 6 seconds
    pauseNSeconds(6);

    // bring outputs low
    MIO1DR &= ~(0x1 << MASTER_OUT);
    MIO1DR &= ~(0x1 << SLAVE1_OUT);
    MIO1DR &= ~(0x1 << SLAVE2_OUT);

    // wait for 2 seconds
    pauseNSeconds(2);

    // force outputs high
    MIO1DR |= 0x1 << MASTER_OUT;
    MIO1DR |= 0x1 << SLAVE1_OUT;
    MIO1DR |= 0x1 << SLAVE2_OUT;
}

// generate a string from an encoder value (24-bit)
// and places it in the string pointed to by refPtr
void genEncoderString(UINT enc, UCHAR *refPtr) {
    UCHAR *encChar;

    encChar = (UCHAR *)&enc;

    // discard the top 8 bits
    encChar += 1;

    // adjust the most significant nibble of enc
    // character is 0-9
    if((((*encChar) >> 4) + 0x30) < 0x3A) {
        *refPtr = ((*encChar) >> 4) + 0x30;
    }
}

```

```

// character is A-F
else {
    *refPtr = ((*encChar) >> 4) + 0x37;
}

// adjust the second most significant nibble of enc
// character is 0-9
if((((*encChar) & 0x0F) + 0x30) < 0x3A) {
    *(refPtr + 1) = ((*encChar) & 0x0F) + 0x30;
}

// character is A-F
else {
    *(refPtr + 1) = ((*encChar) & 0x0F) + 0x37;
}

// adjust the third most significant nibble of enc
// character is 0-9
if((((*(encChar + 1)) >> 4) + 0x30) < 0x3A) {
    *(refPtr + 2) = ((*encChar + 1)) >> 4) + 0x30;
}

// character is A-F
else {
    *(refPtr + 2) = ((*encChar + 1)) >> 4) + 0x37;
}

// adjust the third least significant nibble of enc
// character is 0-9
if((((*(encChar + 1)) & 0x0F) + 0x30) < 0x3A) {
    *(refPtr + 3) = ((*encChar + 1)) & 0x0F) + 0x30;
}

// character is A-F
else {
    *(refPtr + 3) = ((*encChar + 1)) & 0x0F) + 0x37;
}

// adjust the second least significant nibble of enc
// character is 0-9
if((((*(encChar + 2)) >> 4) + 0x30) < 0x3A) {
    *(refPtr + 4) = ((*encChar + 2)) >> 4) + 0x30;
}

// character is A-F
else {
    *(refPtr + 4) = ((*encChar + 2)) >> 4) + 0x37;
}

// adjust the least significant nibble of enc
// character is 0-9
if((((*(encChar + 2)) & 0x0F) + 0x30) < 0x3A) {
    *(refPtr + 5) = ((*encChar + 2)) & 0x0F) + 0x30;
}

```

```

        // character is A-F
        else {
            *(refPtr + 5) = ((*encChar + 2) & 0x0F) + 0x37;
        }
    }
}

```

```

// pause operation for a given amount of time (in seconds)
void pauseNSeconds(UCHAR n) {
    UCHAR i;
    UINT j;

    for(i = 0; i < n; i++) {
        for(j = 0; j < SECOND_PAUSE; j++) {
            // wait
        }
    }
}

```

```

// turn on all meters (with appropriate timing)
void onMeters() {

    // turn on the slave devices first
    onSlaves();

    // wait for slave devices to turn on
    pauseNSeconds(6);

    // turn on the master device
    onMaster();

    // wait for master device to turn on
    // before relenquishing control of
    // the processor
    pauseNSeconds(6);
}

```

```

// turn on the master covermeter
void onMaster() {

    // bring master output low
    MIOSTR &= ~(0x1 << MASTER_OUT);

    // pause for 1 second
    pauseNSeconds(1);

    // force master output high
    MIOSTR |= 0x1 << MASTER_OUT;
}

```

```

// turn on the slave covermeters
void onSlaves() {

```

```

        // bring the slave outputs low
        MIOSDR &= ~(0x1 << SLAVE1_OUT);
        MIOSDR &= ~(0x1 << SLAVE2_OUT);

        // pause for 1 second
        pauseNSeconds(1);

        // force the slave outputs high
        MIOSDR |= 0x1 << SLAVE1_OUT;
        MIOSDR |= 0x1 << SLAVE2_OUT;
    }

// turn on slavel covermeter
void onSlave1() {

    // bring slave1's output low
    MIOSDR &= ~(0x1 << SLAVE1_OUT);

    // pause for 1 second
    pauseNSeconds(1);

    // force slave1's output high
    MIOSDR |= 0x1 << SLAVE1_OUT;
}

// turn on slave2 covermeter
void onSlave2() {

    // bring slave2's output low
    MIOSDR &= ~(0x1 << SLAVE2_OUT);

    // pause for 1 second
    pauseNSeconds(1);

    // force slave2's output high
    MIOSDR |= 0x1 << SLAVE2_OUT;
}

// turn off all meters
void offMeters() {

    // turn off the master device
    offMaster();

    // turn off the slave devices
    offSlaves();
}

// turn off the master covermeter
void offMaster() {

    // bring master output low
    MIOSDR &= ~(0x1 << MASTER_OUT);

```

```

        // pause for 2 seconds
        pauseNSeconds(2);

        // force master output high
        MIOSDR |= 0x1 << MASTER_OUT;
    }

// turn off the slave covermeters
void offSlaves() {

    // bring the slave outputs low
    MIOSDR &= ~(0x1 << SLAVE1_OUT);
    MIOSDR &= ~(0x1 << SLAVE2_OUT);

    // pause for 2 seconds
    pauseNSeconds(2);

    // for the slave outputs high
    MIOSDR |= 0x1 << SLAVE1_OUT;
    MIOSDR |= 0x1 << SLAVE2_OUT;
}

// turn off slavel covermeter
void offSlavel() {

    // bring slavel's output low
    MIOSDR &= ~(0x1 << SLAVE1_OUT);

    // pause for 2 seconds
    pauseNSeconds(2);

    // force slavel's output high
    MIOSDR |= 0x1 << SLAVE1_OUT;
}

// turn off slave2 covermeter
void offSlave2() {

    // bring slave2's output low
    MIOSDR &= ~(0x1 << SLAVE2_OUT);

    // pause for 2 seconds
    pauseNSeconds(2);

    // force slave2's output high
    MIOSDR |= 0x1 << SLAVE2_OUT;
}

```

## controlAux.h

```
//-----
// File:      controlAux.h
// Description: Header file that holds the
//              prototypes for functions used
//              by the main control
//              loop to handle digital I/O and
//              the generation of ASCII coded
//              hexadecimal values for the
//              encoder output.
// Author:     Nathan Holle
//              Kansas State University
// Date:       23JUN06
//-----
#ifndef CONTROLAUX_H
#define CONTROLAUX_H

#include "mpc555.h"
#include "defines.h"

// initialize the digital I/O pins for
// controlling the covermeters
void initIO();

// ensure that the covermeters are turned
// off, regardless of their current state
void clearMeters();

// generate a string from an encoder value (24-bit)
// and places it in the string pointed to by refPtr
void genEncoderString(UINT enc, UCHAR *refPtr);

// pause operation for a given amount of time (in seconds)
void pauseNSeconds(UCHAR n);

// turn on all meters (with appropriate timing)
void onMeters();

// turn on the master covermeter
void onMaster();

// turn on the slave covermeters
void onSlaves();

// turn on slavel covermeter
void onSlavel();

// turn on slave2 covermeter
void onSlave2();

// turn off all meters
void offMeters();

// turn off the master covermeter
```

```
void offMaster();

// turn off the slave covermeters
void offSlaves();

// turn off slave1 covermeter
void offSlave1();

// turn off slave2 covermeter
void offSlave2();

#endif // CONTROLAUX_H
```



## controlMain.c

```
//-----  
// File:          controlMain.c  
// Description:   Source file that holds the  
//               main control loop and controls  
//               module initialization  
// Author:        Nathan Holle  
//               Kansas State University  
// Date:          23JUN06  
//-----  
#include "controlAux.h"  
#include "tpuUART.h"  
#include "tpuFQD.h"  
#include "defines.h"  
  
// the main function housing the main control  
// loop for encoder & serial data capture  
int main() {  
  
    // buffers for receiving the meter data  
    UCHAR receivedMaster[METER_MESSAGE_LENGTH];  
    UCHAR receivedSlave1[METER_MESSAGE_LENGTH];  
  
    // buffers for holding the meter data  
    // to be transmitted  
    UCHAR toBeTransmittedMaster[METER_MESSAGE_LENGTH];  
    UCHAR toBeTransmittedSlave1[METER_MESSAGE_LENGTH];  
  
    // buffer for holding the GPS string  
    // 74 is the max size of GPRMC  
    UCHAR receivedGPS[GPS_COUNT];  
  
    // buffer for holding the GPS string to  
    // be transmitted  
    UCHAR toBeTransmittedGPS[GPS_COUNT];  
  
    // messages for setting up the GPS receiver  
    UCHAR disableMessage[11] = {'$', 'P', 'G', 'R', 'M', 'O',  
                                ', ', '2', CR, LF};  
    UCHAR enableGPRMC[16] = {'$', 'P', 'G', 'R', 'M', 'O',  
                             ', ', 'G', 'P', 'R', 'M', 'C',  
                             ', ', '1', CR, LF};  
  
    // index variables for correctly placing  
    // the received bytes  
    UCHAR iMaster = 0, iSlave1 = 0, iGPS = 0;  
  
    // command received from the PC  
    UCHAR receivedPC = 0;  
  
    // index variable for 'for' loops  
    UCHAR i;  
  
    // status flag of the transmit channel
```

```

    UCHAR txStatus = TX_NOT_BUSY;

    // flag to indicate whether the encoders have
    // been latched after the most recent receipt
    // of meter data
    UCHAR encLatched = FALSE;

    // current position to be transmitted
    UCHAR txPosition;

    // length of the GPS string
    UCHAR GPSCount;

    // buffer for transmitting encoder/covermeter string
    UCHAR toBeTransmittedMeter[METER_COUNT];

    // set up the values in the encoder/covermeter string
    // that do not change
    toBeTransmittedMeter[L_ENCODER_START - 1] = TAB;
    toBeTransmittedMeter[MASTER_START - 1] = TAB;
    toBeTransmittedMeter[SLAVE1_START - 1] = TAB;
    toBeTransmittedMeter[METER_COUNT - 2] = CR;
    toBeTransmittedMeter[METER_COUNT - 1] = LF;

    // reference pointer to the encoder/covermeter string
    UCHAR *meterReference = toBeTransmittedMeter;

    // flags for determining when a message
    // is complete
    UCHAR masterFlag, slave1Flag, GPSFlag;
    masterFlag = FALSE;
    slave1Flag = FALSE;
    GPSFlag = FALSE;

    // flag for determining whether to transmit
    // data to the PC
    UCHAR transmitFlag;

    // initialize transmitFlag to true (transmit by default)
    transmitFlag = TRUE;

    // initialize the digital I/O to control the covermeters
    initIO();

    // initialize the clock and disable interrupts
    // for TPUA
    initTPUA();

    // initialize the clock and disable interrupts
    // for TPUB
    initTPUB();

    // initialize TPUB to handle two pairs
    // of Fast Quadrature Decode (FQD)
    initFQD();

    // initialize the serial link to the PC

```

```

initTx(PC_TX, PC_BAUD);
initRx(PC_RX, PC_BAUD);

// initialize the master receiver
initRx(SLAVE1, METER_BAUD);

// initialize the slave receiver
initRx(MASTER, METER_BAUD);

// initialize the serial link to the GPS
initTx(GPS_TX, GPS_BAUD);
initRx(GPS_RX, GPS_BAUD);

// disable all GPS messages
txString(GPS_TX, disableMessage, 11);

// enable the $GPRMC output message
// (Recommended Minimum Specific GPS/TRANSIT Data (RMC)
txString(GPS_TX, enableGPRMC, 16);

// main control loop
while(1) {

    // update the encoder values (to take care
    // of overflow & underflow problems)
    updateEncoders();

    // if transmit flag is true, then data
    // needs to be gathered and sent
    if(transmitFlag == TRUE) {

        // MASTER data receive code
        // if a new character has been received
        // from the master then put it in the buffer
        if(charReady(MASTER) == 1) {

            // place the character in the buffer
            receivedMaster[iMaster] = rxChar(MASTER);

            // if the end of the message has been reached,
            // set the flag
            if(receivedMaster[iMaster] == LF) {

                // a whole message has been received
                if(iMaster == (METER_MESSAGE_LENGTH-1)) {

                    // set the status flag
                    masterFlag = TRUE;

                    // copy the new value into the
                    // intermediate buffer
                    for(i=0; i < METER_MESSAGE_LENGTH;
                    i++) {
                        toBeTransmittedMaster[i] =
                            receivedMaster[i];

```

```

        }
    }

    // reset the index variable
    iMaster = 0;
}

// not at the end of a message
else {

    // increment the index variable
    iMaster++;
}
}

// SLAVE1 data receive code
// if a new character has been received
// from slavel then put it in the buffer
if(charReady(SLAVE1) == 1) {

    // place the character in the buffer
    receivedSlavel[iSlavel] = rxChar(SLAVE1);

    // if the end of the message has been reached,
    // set the flag
    if(receivedSlavel[iSlavel] == LF) {

        // a whole message has been received
        if(iSlavel == (METER_MESSAGE_LENGTH-1)) {

            // set the status flag
            slavelFlag = TRUE;

            // copy the new value into the
            // intermediate buffer
            for(i=0; i < METER_MESSAGE_LENGTH;
i++) {
                toBeTransmittedSlavel[i] =
                    receivedSlavel[i];
            }

            // reset the index variable
            iSlavel = 0;
        }

        // not at the end of a message
        else {

            // increment the index variable
            iSlavel++;
        }
    }

}

// if all buffers have valid results and the

```

```

// encoder values haven't been latched,
// then latch them
if((masterFlag == TRUE) && (slave1Flag == TRUE) &&
    (encLatched == FALSE)) {

    // latch the current values of the encoders
    latchEncoders();

    // set the encoder latched flag
    encLatched = TRUE;
}

// if all buffers have valid results,
// the output string needs to be created
if((masterFlag == TRUE) && (slave1Flag == TRUE) &&
    (txStatus == TX_NOT_BUSY)) {

    // gain control of the transmit channel
    txStatus = TX_BUSY_METER;

    // place the right encoder value in the
    // output string
    genEncoderString(encoderR, meterReference +
        R_ENCODER_START);

    // place the left encoder value in the
    // output string
    genEncoderString(encoderL, meterReference +
        L_ENCODER_START);

    // place the meter data in the output string
    for(i = 0; i < METER_MESSAGE_LENGTH - 2; i++) {

        // copy the master's characters
        toBeTransmittedMeter[i + MASTER_START] =
            toBeTransmittedMaster[i];

        // copy slave1's characters
        toBeTransmittedMeter[i + SLAVE1_START] =
            toBeTransmittedSlave1[i];

    }

    // clear the character index variable
    txPosition = 0;

    // reset the flags
    masterFlag = FALSE;
    slave1Flag = FALSE;
    encLatched = FALSE;
}

// if a data string (encoders & meters)
// is being transmitted, send out
// a character

```

```

if(txStatus == TX_BUSY_METER) {

    // if channel is ready to send, send a
    // character
    if(CTS(PC_TX) == 1) {

        // send the next character
        txChar(PC_TX,
            toBeTransmittedMeter[txPosition]);

        // increment the index variable
        txPosition++;

        // if the end of the transmission
        // has occurred, relenquish control
        // of the transmission channel
        if(txPosition == METER_COUNT) {

            // set status to not busy
            txStatus = TX_NOT_BUSY;

        }
    }
}

// GPS data receive code
// if a new character has been received
// from the GPS receiver then put it in the buffer
if(charReady(GPS_RX) == 1) {

    // place the character in the buffer
    receivedGPS[iGPS] = rxChar(GPS_RX);

    // if the end of the message has been reached,
    // set the flag
    if(receivedGPS[iGPS] == LF) {

        // if a whole message has been received,
        // send it to the PC
        if(receivedGPS[0] == '$') {

            // transmit the GPS string
            //txString(PC_TX, receivedGPS,
            //          iGPS + 1);

            GPSCount = iGPS + 1;

            // set the GPS flag
            GPSFlag = TRUE;

            // copy the received string into
            // the output buffer
            for(i = 0; i < GPS_COUNT; i++) {
                toBeTransmittedGPS[i] =
                    receivedGPS[i];
            }
        }
    }
}

```

```

        // leave loop early if
        // message doesn't
        // take up the whole buffer
        if(receivedGPS[i] == LF) {
            i = GPS_COUNT;
        }
    }

    // reset the index variable
    iGPS = 0;
}

// not at the end of a message
else {

    // increment the index variable
    iGPS++;
}
}

// if the channel is clear and a GPS
// string has been completed, transmit
// the GPS string
if((GPSFlag == TRUE) && txStatus == TX_NOT_BUSY) {

    // take control of the transmit channel
    txStatus = TX_BUSY_GPS;

    // clear the GPS flag
    GPSFlag = FALSE;

    // clear the character index variable
    txPosition = 0;
}

// if a GPS string is being transmitted,
// send out a character
if(txStatus == TX_BUSY_GPS) {

    // if the channel is clear, send a character
    if(CTS(PC_TX) == 1) {

        // send the next character
        txChar(PC_TX,
            toBeTransmittedGPS[txPosition]);

        // increment the index variable
        txPosition++;

        // if the end of the transmission
        // has occurred, relenquish control
        // of the transmission channel
        if(txPosition == GPSCount) {

```

```

                                // set status to not busy
                                txStatus = TX_NOT_BUSY;
                            }
                        }
                    }
} // end transmitFlag if statement

// if a byte has been received from the
// PC, take the appropriate action
if(charReady(PC_RX) == 1) {

    // put the received byte in the
    // received buffer
    receivedPC = rxChar(PC_RX);

    // take the appropriate action
    switch(receivedPC) {

        // turn on meters (with correct timing)
        case ON_METERS:

            // turn on all meters
            onMeters();
            break;

        // turn on the master
        case ON_MASTER:

            // turn on master
            onMaster();
            break;

        // turn on the slaves
        case ON_SLAVES:

            // turn on slaves
            onSlaves();
            break;

        // turn on slavel
        case ON_SLAVE1:

            // turn on slavel
            onSlavel();
            break;

        // turn on slave2
        case ON_SLAVE2:

            // turn on slave2
            onSlave2();
            break;
    }
}

```



```

// turn off all the meters
case OFF_METERS:

    // turn off all meters
    offMeters();
    break;

// turn off the master
case OFF_MASTER:

    // turn off master
    offMaster();
    break;

// turn off the slaves
case OFF_SLAVES:

    // turn off slaves
    offSlaves();
    break;

// turn off slavel
case OFF_SLAVE1:

    // turn off slavel
    offSlavel();
    break;

// turn off slave2
case OFF_SLAVE2:

    // turn off slave2
    offSlave2();
    break;

// reset the encoders
case RESET_ENCODERS:

    // call the reset encoders routine
    resetEncoders();
    break;

// turn on data transmission
case TRANSMIT_DATA_ON:

    // clear the index variables
    iMaster = 0;
    iSlavel = 0;
    iGPS = 0;

```

```

        // clear the transmit status flag
        txStatus = TX_NOT_BUSY;

        // clear the encoder latched flag
        encLatched = FALSE;

        // clear the receipt flags
        masterFlag = FALSE;
        slave1Flag = FALSE;
        GPSFlag = FALSE;

        // set the transmit data flag
        transmitFlag = TRUE;
        break;

// turn off data transmission
case TRANSMIT_DATA_OFF:

    // clear the transmit data flag
    transmitFlag = FALSE;
    break;

// default case, do nothing
// (command not recognized)
default:
    break;
    }
}

} // end of control loop

return(0);
}

```

## defines.h

```
//-----
// File:         defines.h
// Description:  Header file that defines constants
//              and data types used for the
//              MPC555 communication interface.
// Author:       Nathan Holle
//              Kansas State University
// Date:        23JUN06
//-----
#ifndef DEFINES_H
#define DEFINES_H

// sets the bauds to the appropriate value based
// on whether the debugger is being used
// (comment out the following line for operation without debugger)
// #define DEBUG

#ifdef DEBUG
// 5MHz TPU clock with debugger
#define PC_BAUD    130 // (38.4k with debugger)
#define METER_BAUD 260 // (19.2k with debugger)
#define GPS_BAUD   1042 // (4800 with debugger)
// constant in the for loop that results in
// a one second delay
#define SECOND_PAUSE 455620

#else
// 10MHz TPU clock without debugger
// #define PC_BAUD    260 // (38.4k without debugger)
#define PC_BAUD 521 // (19.2k without debugger)
#define METER_BAUD 521 // (19.2k without debugger)
#define GPS_BAUD 2083 // (4800 without debugger)
// constant in the for loop that results in
// a one second delay
#define SECOND_PAUSE 911240

#endif // DEBUG

// defines the channels used for each device
#define MASTER 8
#define SLAVE1 4
#define PC_TX 1
#define PC_RX 3
#define GPS_TX 7
#define GPS_RX 5

// boolean values
#define TRUE 1
#define FALSE 0

// carriage return character
#define CR 0x0D
```

```

// line feed character
#define LF 0x0A

// tab character
#define TAB 0x09

// define the 8-bit characters
#define UCHAR unsigned char
#define VUCHAR volatile unsigned char

// define the 16-bit characters
#define USHORT unsigned short
#define VUSHORT volatile unsigned short

// define the 32-bit characters
#define UINT unsigned int
#define VUINT volatile unsigned int

// defines for the arbitration of the single
// transmit channel
#define TX_BUSY_METER 'M'
#define TX_BUSY_GPS 'G'
#define TX_NOT_BUSY 'N'

// reference variables for string placement
// (for more streamlined system scaling later)
#define R_ENCODER_START 0
#define L_ENCODER_START 7
#define MASTER_START 14
#define SLAVE1_START 23
#define METER_COUNT 33
#define GPS_COUNT 74

// length of the message sent by the meters
#define METER_MESSAGE_LENGTH 10

// digital I/O pins for turning meters on/off
#define MASTER_OUT 5
#define MASTER_IN 6
#define SLAVE1_OUT 7
#define SLAVE1_IN 8
#define SLAVE2_OUT 9
#define SLAVE2_IN 10

// define shorter names for the MIOS registers
// MIOS data register
#define MIOSDR MIOS1.MPIOSM32DR.R
// MIOS data direction register
#define MIOSDDR MIOS1.MPIOSM32DDR.R

/* command bytes from the PC */

// turn on all meters (with correct timing)
// (ASCII 0)
#define ON_METERS 0x30

```

```

// turn on master (ASCII 1)
#define ON_MASTER 0x31
// turn on both slaves (ASCII 2)
#define ON_SLAVES 0x32
// turn on slave1 (ASCII 3)
#define ON_SLAVE1 0x33
// turn on slave2 (ASCII 4)
#define ON_SLAVE2 0x34
// turn off all meters (ASCII 5)
#define OFF_METERS 0x35
// turn off master (ASCII 6)
#define OFF_MASTER 0x36
// turn off slaves (ASCII 7)
#define OFF_SLAVES 0x37
// turn off slave1 (ASCII 8)
#define OFF_SLAVE1 0x38
// turn off slave2 (ASCII 9)
#define OFF_SLAVE2 0x39

// reset the encoders (ASCII A)
#define RESET_ENCODERS 0x41
// turn on data transmission (ASCII S)
#define TRANSMIT_DATA_ON 0x53
// turn off data transmission (ASCII D)
#define TRANSMIT_DATA_OFF 0x44

/* end - command bytes from PC */

#endif // DEFINES_H

```

## tpuFQD.c

```
//-----
// File:      tpuFQD.c
// Description: Source file that holds the helper
//              functions used to handle the fast
//              quadrature decoding of the
//              two optical encoders.
// Author:     Nathan Holle
//              Kansas State University
// Date:       23JUN06
// Note:       Register initializations in
//              initFQD() were written in
//              collaboration with
//              Justin S. Williams at Kansas
//              State University
//-----
#include "tpuFQD.h"

// initialize the clock of TPUB and disable interrupts
void initTPUB() {

    // TCR1P = 0 (Divide TCR1 by 1)
    // PSCK = X (Fsys / 2 is input to TCR1 prescaler (enhanced))
    //              (10MHz / 2 = 5MHz (with debugger))
    //              (20MHz / 2 = 10MHz (w/o debugger))
    TPU_B.TPUMCR.R = 0x0040;

    // PWOD = 1 (Prescaler write-once disabled)
    // EPSCKE = 1 (Enable enhanced prescaler)
    TPU_B.TPUMCR3.R = 0x0140;

    // disable interrupts
    TPU_B.CIER.R = 0x0000;
}

// initialize TPUB channels (6-7)(R) and (10-11)(L)
// in pairs to perform the Fast Quadrature
// Decode (FQD) function
void initFQD() {
    int k;

    baseR = 0;
    baseL = 0;

    // temporary
    // set up the clock
    TPU_B.TPUMCR.R = 0x4040;
    //PWOD=1, EPSCKE=1, EPSCK=11000(0x18)
    TPU_B.TPUMCR3.R = 0x0158;
    // end temporary

    // no interrupts
    //TPU_B.TICR.R = 0x0000;
```

```

// disable interrupts
TPU_B.CIER.R = 0x0000;

// disable TPUB channels 6 and 7
TPU_B.CPR1.R &= 0x0FFF;

// disable TPUB channels 10 and 11
TPU_B.CPR0.R &= 0xFF0F;

// clear the interrupt status bits
TPU_B.CISR.R = 0x0000;

// set TPUB channels 6 and 7 to Fast
// Quadrature Decode (FQD) function (0x6)
TPU_B.CFSR2.R &= 0x00FF; // clear bits (chan 6 & 7)
TPU_B.CFSR2.R |= 0x6600; // set appropriate bits

// set TPUB channels 10 and 11 to Fast
// Quadrature Decode (FQD) function (0x6)
TPU_B.CFSR1.R &= 0x00FF; // clear bits (chan 10 & 11)
TPU_B.CFSR1.R |= 0x6600; // set appropriate bits

// FQD primary channel (6)
// POSITION_COUNT initialized to RESET_VALUE
// CORR_PINSTATE_ADDR=0x76 (chan 7, offset 6)
// EDGE_TIME_LSB_ADDR=0x61 (chan 6, offset 1)
TPU_B.PARM.R[6][1] = RESET_VALUE;
TPU_B.PARM.R[6][4] = 0x76;
TPU_B.PARM.R[6][5] = 0x61;

// FQD secondary channel (7)
// CORR_PINSTATE_ADDR=0x66 (chan 6, offset 6)
// EDGE_TIME_LSB_ADDR=0x61 (chan 6, offset 1)
TPU_B.PARM.R[7][4] = 0x66;
TPU_B.PARM.R[7][5] = 0x61;

// FQD primary channel (10)
// POSITION_COUNT initialized to RESET_VALUE
// CORR_PINSTATE_ADDR=0xB6 (chan 11, offset 6)
// EDGE_TIME_LSB_ADDR=0xA1 (chan 10, offset 1)
TPU_B.PARM.R[10][1] = RESET_VALUE;
TPU_B.PARM.R[10][4] = 0xB6;
TPU_B.PARM.R[10][5] = 0xA1;

// FQD secondary channel (11)
// CORR_PINSTATE_ADDR=0xA6 (chan 10, offset 6)
// EDGE_TIME_LSB_ADDR=0xA1 (chan 10, offset 1)
TPU_B.PARM.R[11][4] = 0xA6;
TPU_B.PARM.R[11][5] = 0xA1;

// channel 6 set to 00 (primary channel normal mode)
// channel 7 set to 01 (secondary channel normal mode)
TPU_B.HSQR1.R &= 0x0FFF; // clear appropriate bits
TPU_B.HSQR1.R |= 0x4000; // set appropriate bits

// channel 10 set to 00 (primary channel normal mode)

```

```

// channel 11 set to 01 (secondary channel normal mode)
TPU_B.HSQR0.R &= 0xFF0F; // clear appropriate bits
TPU_B.HSQR0.R |= 0x0040; // set appropriate bits

// set channels 6 & 7 to 0b11 (Initialize)
TPU_B.HSRR1.R &= 0x0FFF; // clear HSRR for chan 6 & 7
TPU_B.HSRR1.R |= 0xF000; // initialize

// set channels 10 & 11 to 0b11 (Initialize)
TPU_B.HSRR0.R &= 0xFF0F;
TPU_B.HSRR0.R |= 0x00F0;

// Channel Priority Register
// set all channels 6, 7, 10 and 11 to 0b11 (High Priority)
TPU_B.CPR1.R |= 0xF000;
TPU_B.CPR0.R |= 0x00F0;

for(k = 0; k < 10000; k++) {
    // wait for functions to initialize
}

// end FQD initialization
}

// keep the encoder values up to date (and update
// 'base' to keep track of the overflows and underflows)
void updateEncoders() {
    // temporary variable
    USHORT count;

    // right encoder
    count = TPU_B.PARM.R[RIGHT_ENC][1];

    if(count < THRESHOLD) {
        // decrement the multiplier
        baseR--;

        // reset the counter
        TPU_B.PARM.R[RIGHT_ENC][1] += (RESET_VALUE - THRESHOLD);
    }

    else {
        if(count > (MAX - THRESHOLD)) {
            // increment the multiplier
            baseR++;

            // reset the counter
            TPU_B.PARM.R[RIGHT_ENC][1] -= ((MAX - THRESHOLD) -
                                           RESET_VALUE);
        }
    }

    // left encoder
    count = TPU_B.PARM.R[LEFT_ENC][1];

    if(count < THRESHOLD) {

```



```

        // decrement the multiplier
        baseL--;

        // reset the counter
        TPU_B.PARM.R[LEFT_ENC][1] += (RESET_VALUE - THRESHOLD);
    }

    else {
        if(count > (MAX - THRESHOLD)) {
            // increment the multiplier
            baseL++;

            // reset the counter
            TPU_B.PARM.R[LEFT_ENC][1] -= ((MAX - THRESHOLD) -
                                           RESET_VALUE);
        }
    }
}

// latch the current encoder values to
// be read momentarily (ensures both encoder
// values with be synchronized)
void latchEncoders() {
    // temporary values
    USHORT right, left;

    // grab the current counter values at
    // nearly the same time
    right = TPU_B.PARM.R[RIGHT_ENC][1];
    left = TPU_B.PARM.R[LEFT_ENC][1];

    // calculate the new values
    encoderR = (baseR * ((MAX - THRESHOLD) - RESET_VALUE))
               + (right - RESET_VALUE);

    encoderL = (baseL * ((MAX - THRESHOLD) - RESET_VALUE))
               + (left - RESET_VALUE);
}

// reset the encoder values
void resetEncoders() {

    // reset the overflow/underflow variables
    baseR = 0;
    baseL = 0;

    // set encoder values to 0
    encoderR = 0;
    encoderL = 0;

    // reset the counter registers for
    // each encoder
    TPU_B.PARM.R[RIGHT_ENC][1] = RESET_VALUE;
    TPU_B.PARM.R[LEFT_ENC][1] = RESET_VALUE;
}

```

## tpuFQD.h

```
//-----
// File:      tpuFQD.h
// Description: Header file that defines constants
//              and holds the prototypes for the
//              helper functions used to handle
//              the fast quadrature decoding of
//              the two optical encoders.
// Author:    Nathan Holle
//            Kansas State University
// Date:      23JUN06
//-----
#ifndef TPUFQD_H
#define TPUFQD_H

#include "mpc555.h"
#include "defines.h"

#define RESET_VALUE 32768
#define MAX 65535
#define THRESHOLD 1000
#define RIGHT_ENC 6
#define LEFT_ENC 10

// keeps track of the overflows and
// underflows
USHORT baseR;
USHORT baseL;

// latched values of the encoders (global)
UINT encoderR;
UINT encoderL;

// initialize the clock of TPUB and disable interrupts
void initTPUB();

// initialize TPUB channels (6-7) and (10-11)
// in pairs to perform the Fast Quadrature
// Decode (FQD) function
void initFQD();

// keep the encoder values up to date
void updateEncoders();

// latch the current encoder values to
// be read momentarily
void latchEncoders();

// reset the encoder values
void resetEncoders();

#endif // TPUFQD_H
```

## tpuUART.c

```
//-----
// File:          tpuUART.c
// Description:    Source file that holds the helper
//                functions used by the main control
//                loop to handle serial
//                communication through TPUA
// Author:        Nathan Holle
//                Kansas State University
// Date:          23JUN06
//-----
#include "tpuUART.h"

// disableChannelA disables a specified channel in TPUA
void disableChannelA(UCHAR channel) {

    // check to see if the channel value is valid
    if(channel > 15) {
        // invalid channel, return from function
        return;
    }

    if(channel < 8) {
        // use CPR1
        TPU_A.CPR1.R &= ~(0x3 << (channel * 2));
    }

    else {
        // use CPR0
        TPU_A.CPR0.R &= ~(0x3 << ((channel - 8) * 2));
    }
}

// enable the given channel in TPUA with
// the given priority
void enableChannelA(UCHAR channel, UCHAR priority) {
    // intermediate value;
    unsigned short value;

    // check to see if the channel value is valid
    if(channel > 15) {
        // invalid channel, return from function
        return;
    }

    // check to see if the priority is valid
    if(priority > 3) {
        // invalid priority, return from function
        return;
    }

    if(channel < 8) {
        // use CPR1
```

```

        // clear the bits associated with the channel
        value = TPU_A.CPR1.R & ~(0x3 << (channel * 2));

        // set the appropriate bits
        value |= (priority << (channel * 2));

        // put the new value into the register
        TPU_A.CPR1.R = value;
    }

else {
    // use CPR0

    // clear the bits associated with the channel
    value = TPU_A.CPR1.R & ~(0x3 << ((channel - 8) * 2));

    // set the appropriate bits
    value |= (priority << ((channel - 8) * 2));

    // put the new value into the register
    TPU_A.CPR0.R = value;
}

}

// set the function of a given TPUA channel via the
// Channel Function Select Register (CFSRx)
void selectFunctionA(UCHAR channel, UCHAR function) {
    // intermediate variable
    unsigned short value;

    // check to see if the channel value is valid
    if(channel > 15) {
        // invalid channel, return from function
        return;
    }

    // check to see if the function value is valid
    if(function > 15) {
        // invalid function, return from function
        return;
    }

    if(channel < 4) {
        // use CFSR3

        // clear the nibble associated with the channel
        value = TPU_A.CFSR3.R & ~(0xF << (channel * 4));

        // set the appropriate bits to yield the correct value
        value |= (function << (channel * 4));

        // set the new value in the register
        TPU_A.CFSR3.R = value;
    }

else {
    if(channel < 8) {

```

```

        // use CFSR2

        // clear the nibble associated with the channel
        value = TPU_A.CFSR2.R & ~(0xF<<((channel-4) * 4));

        // set the appropriate bits to yield the
        // correct value
        value |= (function << ((channel - 4) * 4));

        // set the new value in the register
        TPU_A.CFSR2.R = value;
    }

    else {
        if(channel < 12) {
            // use CFSR1

            // clear the nibble associated with the channel
            value = TPU_A.CFSR1.R & ~(0xF<<((channel-8)*4));

            // set the appropriate bits to yield the
            // correct value
            value |= (function << ((channel - 8) * 4));

            // set the new value in the register
            TPU_A.CFSR1.R = value;
        }

        else {
            // use CFSR0

            // clear the nibble associated with the channel
            value = TPU_A.CFSR1.R & ~(0xF<<((channel-12)*4));

            // set the appropriate bits to yield the
            // correct value
            value |= (function << ((channel - 12) * 4));

            // set the new value in the register
            TPU_A.CFSR1.R = value;
        }
    }
}

// set the Host Sequence Register (HSQRx) to
// the desired value
void setHSQ(UCHAR channel, UCHAR valueHSQ) {
    // intermediate variable
    unsigned short value;

    // check to see if the channel value is valid
    if(channel > 15) {
        // invalid channel, return from function
        return;
    }
}

```

```

// check to see if the HSQ value is valid
if(valueHSQ > 3) {
    // invalid HSQ value, return from function
    return;
}

if(channel < 8) {
    // use HSQR1
    value = TPU_A.HSQR1.R & ~(0x3 << (channel * 2));
    value |= (valueHSQ << (channel * 2));
    TPU_A.HSQR1.R = value;
}

else {
    // use HSQR0
    value = TPU_A.HSQR0.R & ~(0x3 << ((channel - 8) * 2));
    value |= (valueHSQ << ((channel - 8) * 2));
    TPU_A.HSQR0.R = value;
}
}

// set the Host Service Request Register (HSRRx) to
// the desired value
void setHSR(UCHAR channel, UCHAR valueHSR) {
    // intermediate variable
    unsigned short value;

    // check to see if the channel value is valid
    if(channel > 15) {
        // invalid channel, return from function
        return;
    }

    // check to see if the HSR value is valid
    if(valueHSR > 3) {
        // invalid HSQ value, return from function
        return;
    }

    if(channel < 8) {
        // use HSRR1
        value = TPU_A.HSRR1.R & ~(0x3 << (channel * 2));
        value |= (valueHSR << (channel * 2));
        TPU_A.HSRR1.R = value;
    }

    else {
        // use HSRR0
        value = TPU_A.HSRR0.R & ~(0x3 << ((channel - 8) * 2));
        value |= (valueHSR << ((channel - 8) * 2));
        TPU_A.HSRR0.R = value;
    }
}

// initialize the clock of TPUA and disable interrupts
void initTPUA() {

```

```

// TCR1P = 0 (Divide TCR1 by 1)
// PSCK = X (Fsys / 2 is input to TCR1 prescaler (enhanced))
//          (10MHz / 2 = 5MHz (with debugger))
//          (20MHz / 2 = 10MHz (w/o debugger))
TPU_A.TPUMCR.R = 0x0040;

// PWOD = 1 (Prescaler write-once disabled)
// EPSCKE = 1 (Enable enhanced prescaler)
TPU_A.TPUMCR3.R = 0x0140;

// disable interrupts
TPU_A.CIER.R = 0x0000;
}

/*
// returns whether the interrupt flag is set for
// the given channel
UCHAR checkInterrupt(UCHAR channel) {

    // check if channel is a valid number
    if(channel > 15) {
        return(0);
    }

    if((TPU_A.CISR.R & (0x1 << channel)) != 0) {
        return(1);
    }
    else {
        return(0);
    }
}
*/

// set up a specified TPUA channel as a UART transmitter
void initTx(UCHAR channel, USHORT baud) {

    // check to see if channel is valid
    if(channel > 15) {
        return;
    }

    // disable channel
    disableChannelA(channel);

    // clear the interrupt status bit
    TPU_A.CISR.R &= ~(0x1 << channel);

    // set up the baud
    TPU_A.PARM.R[channel][1] = baud;

    // set TDRE
    TPU_A.PARM.R[channel][2] = 0x8000;

    // number of data bits = 8
    // DATA_SIZE = 0x8
    TPU_A.PARM.R[channel][3] = 0x0008;

```

```

    // set channel to UART function
    // (UART = 0xB)
    selectFunctionA(channel, 0xB);

    // set parity (no parity)
    setHSQ(channel, 0x0);

    // set channel to transmitter (0x3)
    setHSR(channel, 0x3);

    // enable the channel (high priority, 0x3)
    enableChannelA(channel, 0x3);

    // wait for channel to be initialized
    while(CTS(channel) == 0) {
        // wait
    }
}

// set up a specified TPUA channel as a UART receiver
void initRx(UCHAR channel, USHORT baud) {

    // check to see if channel is valid
    if(channel > 15) {
        return;
    }

    // disable channel
    disableChannelA(channel);

    // clear the interrupt status bit
    TPU_A.CISR.R &= ~(0x1 << channel);

    // set up the baud
    TPU_A.PARM.R[channel][1] = baud;

    // number of data bits = 8
    // DATA_SIZE = 0x8
    TPU_A.PARM.R[channel][3] = 0x0008;

    // set channel to UART function
    // (UART = 0xB)
    selectFunctionA(channel, 0xB);

    // set parity (no parity)
    setHSQ(channel, 0x0);

    // set channel to receiver (0x2)
    setHSR(channel, 0x2);

    // enable the channel (high priority = 0x3)
    enableChannelA(channel, 0x3);
}

// check if channel is clear to send
UCHAR CTS(UCHAR channel) {
    // don't check for channel validity

```



```

    // going for speed

    if((TPU_A.CISR.R & (0x1 << channel)) == 0) {
        // not clear to send
        return(0);
    }

    else {
        // clear to send
        return(1);
    }
}

// transmit a character over the given TPU channel
void txChar(UCHAR channel, UCHAR c) {
    // don't check for channel validity
    // going for speed

    // wait for channel to be ready
    while(CTS(channel) == 0) { }

    // clear the appropriate interrupt flag
    TPU_A.CISR.R &= ~(0x1 << channel);

    // write the character to TRANSMIT_DATA_REG
    TPU_A.PARM.R[channel][2] = ((USHORT)c) & 0x00FF;
}

// transmit a short over the given serial port
// in ASCII hex
void txShort(UCHAR channel, USHORT s) {
    //UCHAR c1, c2, c3, c4;
    UCHAR c[4], i;
    UCHAR *sChar;

    sChar = (UCHAR *)&s;

    // adjust the most significant nibble of s
    // character is 0-9
    if((((*sChar) >> 4) + 0x30) < 0x3A) {
        c[0] = ((*sChar) >> 4) + 0x30;
    }

    // character is A-F
    else {
        c[0] = ((*sChar) >> 4) + 0x37;
    }

    // adjust the second most significant nibble of s
    // character is 0-9
    if((((*sChar) & 0x0F) + 0x30) < 0x3A) {
        c[1] = ((*sChar) & 0x0F) + 0x30;
    }

    // character is A-F

```

```

else {
    c[1] = ((*sChar) & 0x0F) + 0x37;
}

// adjust the second least significant nibble of s
// character is 0-9
if((((*(sChar + 1)) >> 4) + 0x30) < 0x3A) {
    c[2] = ((*sChar + 1)) >> 4) + 0x30;
}

// character is A-F
else {
    c[2] = ((*sChar + 1)) >> 4) + 0x37;
}

// adjust the least significant nibble of s
// character is 0-9
if((((*(sChar + 1)) & 0x0F) + 0x30) < 0x3A) {
    c[3] = ((*sChar + 1)) & 0x0F) + 0x30;
}

// character is A-F
else {
    c[3] = ((*sChar + 1)) & 0x0F) + 0x37;
}

for(i = 0; i < 4; i++) {

    while(CTS(channel) == 0x00) {
        // wait for channel to be ready
        // to transmit
    }

    // write the byte to the serial port
    txChar(channel, c[i]);
}

/*
// write the first byte to the serial port
txChar(channel, c1);

// wait for the first byte to be sent
while(CTS() == 0x00) {
    // wait for channel to be ready
    // to transmit
}

// write the second byte to the serial port
txChar(channel, c2);

// wait for the first byte to be sent
while(CTS() == 0x00) {
    // wait for channel to be ready
    // to transmit
}

```

```

    // write the third byte to the serial port
    txChar(channel, c3);

    // wait for the first byte to be sent
    while(CTS() == 0x00) {
        // wait for channel to be ready
        // to transmit
    }

    // write the fourth byte to the serial port
    txChar(channel, c3);
    */
}

// transmit a string of characters over the give TPU channel
void txString(UCHAR channel, UCHAR *s, UCHAR num) {
    UCHAR i;
    //unsigned int j;

    // don't check for channel validity
    // going for speed

    for(i = 0; i < num; i++) {

        // wait for channel to be ready
        while(CTS(channel) == 0) { }

        //nwhcheck
        //for(j = 0; j < 500; j++) { }

        // send character
        txChar(channel, *(s + i));
    }
}

// transmit an encoder value (24-bit)
void txEncoder(UCHAR channel, UINT enc) {
    UCHAR c[6]; //, i;
    UCHAR *encChar;

    encChar = (UCHAR *)&enc;

    // discard the top 8 bits
    encChar += 1;

    // adjust the most significant nibble of enc
    // character is 0-9
    if((((*encChar) >> 4) + 0x30) < 0x3A) {
        c[0] = ((*encChar) >> 4) + 0x30;
    }

    // character is A-F
    else {
        c[0] = ((*encChar) >> 4) + 0x37;
    }

    // adjust the second most significant nibble of enc

```

```

// character is 0-9
if((((*encChar) & 0x0F) + 0x30) < 0x3A) {
    c[1] = ((*encChar) & 0x0F) + 0x30;
}

// character is A-F
else {
    c[1] = ((*encChar) & 0x0F) + 0x37;
}

// adjust the third most significant nibble of enc
// character is 0-9
if((((*(encChar + 1)) >> 4) + 0x30) < 0x3A) {
    c[2] = ((*encChar + 1)) >> 4) + 0x30;
}

// character is A-F
else {
    c[2] = ((*encChar + 1)) >> 4) + 0x37;
}

// adjust the third least significant nibble of enc
// character is 0-9
if((((*(encChar + 1)) & 0x0F) + 0x30) < 0x3A) {
    c[3] = ((*encChar + 1)) & 0x0F) + 0x30;
}

// character is A-F
else {
    c[3] = ((*encChar + 1)) & 0x0F) + 0x37;
}

// adjust the second least significant nibble of enc
// character is 0-9
if((((*(encChar + 2)) >> 4) + 0x30) < 0x3A) {
    c[4] = ((*encChar + 2)) >> 4) + 0x30;
}

// character is A-F
else {
    c[4] = ((*encChar + 2)) >> 4) + 0x37;
}

// adjust the least significant nibble of enc
// character is 0-9
if((((*(encChar + 2)) & 0x0F) + 0x30) < 0x3A) {
    c[5] = ((*encChar + 2)) & 0x0F) + 0x30;
}

// character is A-F
else {
    c[5] = ((*encChar + 2)) & 0x0F) + 0x37;
}

// transmit the newly created string

```

```

        txString(channel, c, 6);
    }

    // check if a character has been received
    // (non-blocking)
    UCHAR charReady(UCHAR channel) {
        // don't check for channel validity
        // going for speed

        if((TPU_A.CISR.R & (0x1 << channel)) == 0) {
            // no new character
            return(0);
        }

        else {
            // new character
            return(1);
        }
    }

    // receive a character from a given TPUA channel
    // (blocking)
    UCHAR rxChar(UCHAR channel) {
        // don't check for channel validity
        // going for speed

        while(charReady(channel) == 0) {
            // wait for a received character
        }

        // clear the interrupt flag for the appropriate channel
        TPU_A.CISR.R &= ~(0x1 << channel);

        // return the received character
        return((UCHAR)(TPU_A.PARM.R[channel][2] & 0x3FFF));
    }

```

## tpuUART.h

```
//-----
// File:      tpuUART.h
// Description: Header file that holds the
//              prototypes for the helper
//              functions used by the main control
//              loop to handle serial
//              communication through TPUA
// Author:     Nathan Holle
//              Kansas State University
// Date:       23JUN06
//-----
#ifndef TPUUART_H
#define TPUUART_H

#include "mpc555.h"
#include "defines.h"

// disableChannelA disables a specified channel in TPUA
void disableChannelA(UCHAR channel);

// enable the given channel in TPUA with
// the given priority
void enableChannelA(UCHAR channel, UCHAR priority);

// set the function of a given TPUA channel via the
// Channel Function Select Register (CFSRx)
void selectFunctionA(UCHAR channel, UCHAR function);

// set the Host Sequence Register (HSQRx) to
// the desired value
void setHSQ(UCHAR channel, UCHAR valueHSQ);

// set the Host Service Request Register (HSRRx) to
// the desired value
void setHSR(UCHAR channel, UCHAR valueHSR);

// initialize the clock of TPUA and disable interrupts
void initTPUA();

/*
// returns whether the interrupt flag is set
UCHAR checkInterrupt(UCHAR channel);
*/

// set up a specified TPUA channel as a UART transmitter
void initTx(UCHAR channel, USHORT baud);

// set up a specified TPUA channel as a UART receiver
void initRx(UCHAR channel, USHORT baud);

// check if channel is clear to send
UCHAR CTS(UCHAR channel);
```

```

// transmit a character over the given TPUA channel
void txChar(UCHAR channel, UCHAR c);

// transmit a short over the given serial port
// in ASCII hex
void txShort(UCHAR channel, USHORT s);

// transmit a string of characters over the given TPUA channel
void txString(UCHAR channel, UCHAR *s, UCHAR num);

// transmit an encoder value (24-bit)
void txEncoder(UCHAR channel, UINT enc);

// check if a character has been received
// (non-blocking)
UCHAR charReady(UCHAR channel);

// receive a character from a given TPUA channel
// (blocking)
UCHAR rxChar(UCHAR channel);

#endif // TPUART_H

```

## **Appendix D: Synchronized Covermeter Product Specification**

Document reproduced from [11]





**Product Specification**

**PS0178-001 Issue 0A**

**Synchronised Proba3D units**

[illegible]

## TABLE OF CONTENTS

1.	System Description.....	4
1.1	Introduction.....	4
1.2	Nomenclature.....	4
1.3	Abbreviations.....	4
1.4	Architecture .....	4
1.5	Operation .....	5
1.6	Power on and off.....	5
1.7	Optimising signal strength measurements.....	5
2.	Specifications.....	6
2.1	Power Supply Requirement (per unit).....	6
2.2	Interface requirements .....	6
2.3	Data Format .....	6
2.4	Connections .....	6
2.4.1	Serial port connections .....	6
2.4.2	Power connections.....	6
2.4.3	Synchronisation connections.....	6
2.4.4	Keypad connections.....	7
3.	Output Data.....	8
3.1	Output Rate.....	8
3.2	Output Data Format.....	8
3.3	Output Data Description.....	8
3.4	Using a Terminal Emulator .....	9

## TABLE OF FIGURES

Figure 1.1:	Synchronisation overview block diagram.....	4
-------------	---	---

## 1. SYSTEM DESCRIPTION

### 1.1 Introduction

This document details the Proba3D synchronised electronics and how they may be used. There is a single master board and a pair of slave boards. The power up sequence is critical to the synchronisation operation and is detailed below.

### 1.2 Nomenclature

Unit – Master or Slave Proba3D electronics

Master – Proba3D master electronics

Slave – Proba3D slave electronics

### 1.3 Abbreviations

Pcb – Printed circuit board

### 1.4 Architecture

The system comprises three Proba3D sets of electronics which output raw signal strength serial data via modified probes.

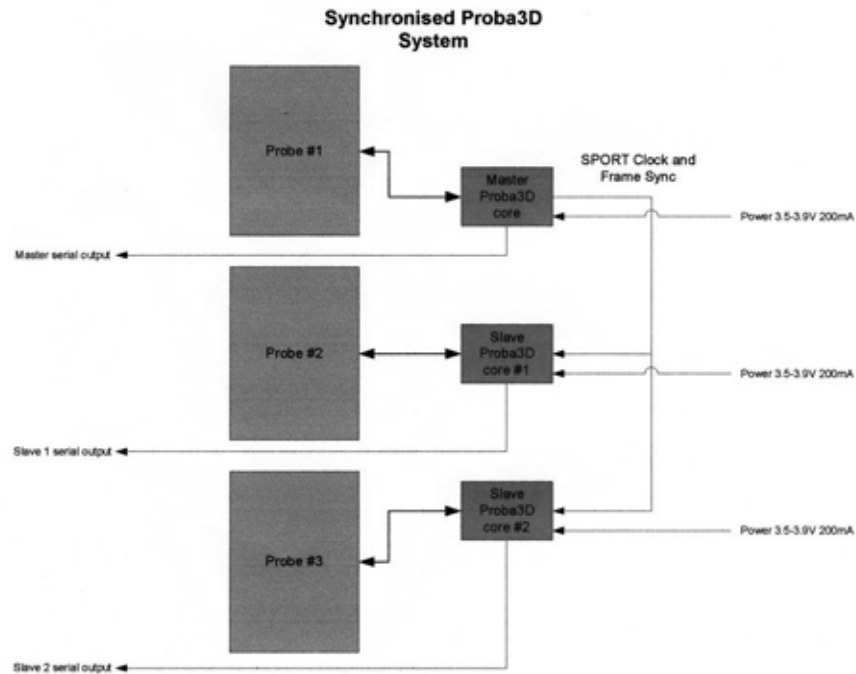


Figure 1.1: Synchronisation overview block diagram

### **1.5 Operation**

In a normal Proba3D the transmit/receive timing and signalling is driven by a pair of signals from the main processor. In order to synchronise Proba3D units it is necessary to use a single pair of these synchronising signals from the master to drive the transmit/receive timing and signalling of itself and the slave units.

To achieve this the master unit runs software which uses its internal signals for the generation of its timing, whereas the slave units have software which mean they expect to have external signals fed in to provide their timing.

The slaves must be powered on before the master, so they are ready to begin operation as soon as the master is operational and begins outputting the synchronising signals. If the slaves are not powered on prior to the master, then although the system will operate in a synchronised manner, there will be an undefined phase difference between the master and slaves.

### **1.6 Power on and off**

In a normal Proba3D unit the internal battery is permanently connected and the power on/off sequence is simply activated by pressing and holding the mode key on the keypad.

This method may still be used by connecting some suitable controls to the keypad connector (connection details appear below) to imitate the keypad mode key.

An alternate but less preferred method for powering on and off is to simply apply and remove the power from each unit. A delay of at least 30 seconds is required between a power off and a subsequent power on to ensure the reliable reset of a unit.

The units have been modified so that the LED, which normally indicates low battery, now indicates that the unit is in active mode when lit, and standby mode when off.

### **1.7 Optimising signal strength measurements**

The raw signal strength measurement is affected by the presence of neighbouring probes. In order to obtain the maximum cover measurement range it is necessary to minimise the affects of neighbouring probes, as a very large signal from a nearby probe will otherwise swamp the small signal from a 'distant' rebar.

This can be achieved by logging measurements with only a single probe connected and noting the raw signal strength with no rebar present. Then by operating the system with all the probes, the effects of the probes on each other can be measured as they are moved.

## 2. SPECIFICATIONS

### 2.1 Power Supply Requirement (per unit)

Parameter	Value	Units	Comments
Supply voltage	3.5-3.9	VDC	
Supply current	<0.2	A	

### 2.2 Interface requirements

Parameter	Value	Units	Comments
External data serial ports	RS232		
Data port speed	19200	Baud	
Reporting rate	20	Hz	

### 2.3 Data Format

The data format is fixed with the following parameters:

- ❖ Eight data bits.
- ❖ No parity.
- ❖ One stop bit

### 2.4 Connections

#### 2.4.1 Serial port connections

The serial port of each unit is accessible by a standard 9-way D-type connector. This connection is suitable for direct connection to a PC serial port.

Signal	Pin
RS232 Tx	2
RS232 Rx	3
RS232 GND	5

#### 2.4.2 Power connections

The power connections to each unit are via flying leads, power supply requirements are detailed above:

Signal	Colour
Supply +	Red
Supply -	Black

#### 2.4.3 Synchronisation connections

The synchronisation connections to each unit are via directly soldered flying leads, the connections should be kept as short as possible:

Signal	Colour
Sync. clock	Orange
Sync. frame	Blue

Ground	Black
--------	-------

#### 2.4.4 Keypad connections

The instrument keypad normally connects to the right angled 5-way connector on the pcb. Each key is operated when pressed by closing one of the four signal lines to ground. Pin 1 is indicated by the square pad on the pcb (the pin nearest to the probe and power pcb connectors).

The mode key is held for 1 second to switch on a unit, and two seconds to switch off a unit.

Pin	Signal
1	Left keypad button
2	Centre keypad button
3	Right keypad button
4	Mode keypad button
5	Ground

### 3. OUTPUT DATA

#### 3.1 Output Rate

This version of the Proba3D covermeter outputs its raw signal data at a rate of 20Hz.

#### 3.2 Output Data Format

This data is in 8 digit Ascii hexadecimal format and delimited by a CRLF, an example of increasing output data appears below:

```
FFFFFAA4
FFFFFBB2
FFFFFCCC
FFFFFE02
FFFFF54
000000C6
00000262
0000041E
00000604
00000814
00000A4C
00000CC0
00000F5E
00001232
0000153C
00001884
```

Two's complement is used to represent negative values as can be seen from the sample of output data above.

#### 3.3 Output Data Description

The output data relates directly to the signal from the Proba3D probe. The signal is digitized to provide a numerical amplitude measurement which is dependent on the proximity and amount of steel reinforcement. Notice that in the presence of zero steel the measurement is non zero, and this 'zero' measurement will vary slightly from one probe to another and slowly with time during operation.

The output data has undergone some digital signal processing, firstly in the form of a low pass filter.

Due to the very large range in probe signal across the measurement range a gain stage in the receive electronics is switched between two gain values dependent on incoming signal strength. This gain 'step' has been accounted for in the output data, so monotonically increasing or decreasing raw signal from the probe will result in output data also being correspondingly monotonic.

**This correction for gain setting is in effect only following an instrument calibration or zeroing. This is done simply by pressing the CAL button (Left**



button >5 seconds after power on) during cover measurement mode with the probe well away from any steel.

### **3.4 Using a Terminal Emulator**

Most simply the Proba3D output may be monitored with the use of a terminal emulator such as Windows Hyperterminal, by setting the appropriate serial communications parameters as detailed above. Such an application allows the user to monitor and log output data to file.

## Appendix E: Raw Data Files

### SingleCovermeterExample.txt

FFFFF4AC  
FFFFF5BC  
FFFFF6F0  
FFFFF846  
FFFFF9CC  
FFFFFB7E  
FFFFFD60  
FFFFFFA2  
00000226  
000004F6  
000007EE  
00000B20  
00000E94  
00001254  
0000163C  
00001A2C  
00001E0C  
000021EE  
0000252C  
000026A0  
000025A4  
0000225E  
00001DB0  
00001876  
0000134E  
00000E7C  
00000A22  
00000646  
000002EE  
0000000A  
FFFFFD92  
FFFFFB78  
FFFFF9BC  
FFFFF83E  
FFFFF702  
FFFFF5FC  
FFFFF512  
FFFFF446  
FFFFF392  
FFFFF2FA  
FFFFF26E  
FFFFF20E  
FFFFF1C8  
FFFFF1A8  
FFFFF19E  
FFFFF1CC  
FFFFF21C  
FFFFF2A8

29AUG2006\_S01\_C\_ABBREVIATED.dat

C	-1	-2	664.9	671.9				
G	-1	1	38.928995	0	97.364428	1	290806	153555
Z	-38958.4418402778	-38958.4418402778	-5268	-8686				
Z	-38958.4418402778	-38958.4418402778	-5264	-8690				
Z	-38958.4418402778	-38958.4418402778	-5264	-8686				
Z	-38958.4418402778	-38958.4418402778	-5262	-8670				
Z	-38958.4418402778	-38958.4418402778	-5268	-8684				
Z	-38958.4418402778	-38958.4418402778	-5274	-8688				
Z	-38958.4418402778	-38958.4418402778	-5278	-8682				
Z	-38958.4418402778	-38958.4418402778	-5274	-8682				
Z	-38958.4418402778	-38958.4418402778	-5264	-8680				
Z	-38958.4418402778	-38958.4418402778	-5264	-8674				
Z	-38958.4418402778	-38958.4418402778	-5264	-8670				
Z	-38958.4418402778	-38958.4418402778	-5260	-8676				
Z	-38958.4418402778	-38958.4418402778	-5262	-8666				
Z	-38958.4418402778	-38958.4418402778	-5264	-8676				
Z	-38958.4418402778	-38958.4418402778	-5262	-8676				
Z	-38958.4418402778	-38958.4418402778	-5254	-8674				
Z	-38958.4418402778	-38958.4418402778	-5254	-8686				
Z	-38958.4418402778	-38958.4418402778	-5254	-8682				
Z	-38958.4418402778	-38958.4418402778	-5258	-8688				
Z	-38958.4418402778	-38958.4418402778	-5264	-8698				
D	0	0	-5062	-8498				
D	0	0	-5068	-8498				
D	0	0	-5068	-8490				
D	0	0	-5068	-8488				
D	0	0	-5060	-8496				
D	0	0	-5060	-8492				
D	0	0	-5062	-8496				
D	0	0	-5072	-8490				
D	0	0	-5076	-8490				
D	0	0	-5078	-8492				
D	0	0	-5080	-8494				
D	0	0	-5068	-8482				
D	0	0	-5070	-8490				
D	0	0	-5058	-8488				
D	0	0	-5058	-8482				
G	-1	1	38.928988	0	97.364432	1	290806	153601
D	1	2	-5048	-8480				
D	3	5	-5048	-8486				
D	8	10	-5046	-8480				
D	15	18	-5052	-8488				
D	25	27	-5046	-8486				
D	36	40	-5054	-8484				
D	51	56	-5060	-8500				
D	69	74	-5064	-8498				
D	89	95	-5070	-8494				
D	113	120	-5074	-8508				
D	140	147	-5066	-8508				
D	170	177	-5062	-8506				
D	203	209	-5070	-8510				
D	237	243	-5070	-8508				
D	271	276	-5072	-8502				
D	304	310	-5072	-8506				
D	339	344	-5064	-8510				
D	373	380	-5062	-8502				
D	407	415	-5062	-8508				

D	443	452	-5054	-8498
D	479	487	-5048	-8492
D	516	523	-5048	-8500
D	554	558	-5044	-8506
D	593	595	-5062	-8508
D	632	635	-5062	-8516
D	673	674	-5064	-8516
D	713	715	-5076	-8526
D	751	756	-5076	-8534
D	792	798	-5078	-8530
D	835	840	-5090	-8530
D	878	885	-5086	-8544
D	922	931	-5088	-8538
D	967	979	-5090	-8518
D	1015	1029	-5084	-8518
D	1062	1081	-5086	-8510
D	1112	1134	-5078	-8494
D	1163	1189	-5070	-8478
D	1215	1242	-5066	-8454
D	1268	1297	-5058	-8416
D	1320	1353	-5044	-8398
D	1375	1411	-5046	-8392
D	1432	1468	-5064	-8392
D	1490	1527	-5080	-8408
D	1550	1588	-5102	-8432
D	1610	1650	-5124	-8464
D	1671	1713	-5134	-8490
D	1734	1775	-5156	-8524
D	1799	1839	-5172	-8542
D	1866	1907	-5184	-8558
D	1933	1974	-5192	-8578
D	2008	2047	-5204	-8588
D	2076	2112	-5198	-8604
D	2147	2183	-5196	-8616
D	2217	2251	-5186	-8610
D	2289	2321	-5174	-8598
D	2361	2392	-5150	-8574
D	2430	2463	-5122	-8532
D	2496	2531	-5098	-8494
D	2566	2602	-5082	-8478
D	2632	2672	-5080	-8464
D	2704	2747	-5086	-8470
D	2773	2817	-5098	-8484
D	2842	2892	-5120	-8498
D	2910	2964	-5134	-8522
D	2981	3037	-5142	-8552
D	3053	3111	-5156	-8568
D	3124	3186	-5174	-8590
D	3195	3260	-5194	-8602
D	3268	3334	-5198	-8614
D	3341	3409	-5202	-8624
D	3411	3483	-5200	-8628
D	3484	3557	-5190	-8618
D	3560	3630	-5178	-8600
D	3632	3702	-5154	-8564
D	3704	3776	-5130	-8530
D	3779	3852	-5108	-8496

D	3856	3929	-5100	-8476
D	3932	4006	-5102	-8462
D	4008	4080	-5104	-8474
D	4086	4156	-5120	-8496
D	4162	4232	-5148	-8528
D	4239	4312	-5156	-8548
D	4315	4388	-5172	-8578
D	4390	4468	-5186	-8600
D	4463	4548	-5204	-8608
D	4540	4628	-5216	-8608
D	4616	4706	-5220	-8620
D	4694	4786	-5218	-8626
D	4768	4862	-5206	-8612
D	4844	4934	-5194	-8596
D	4924	5012	-5170	-8570
D	5006	5090	-5154	-8538
D	5090	5169	-5128	-8514
D	5170	5249	-5114	-8512
D	5250	5330	-5096	-8512
D	5335	5410	-5100	-8526
D	5414	5487	-5094	-8550
D	5496	5564	-5116	-8564
D	5579	5642	-5136	-8582
D	5661	5724	-5142	-8602
D	5738	5806	-5164	-8604
D	5814	5882	-5170	-8612
D	5890	5959	-5180	-8632
D	5966	6035	-5190	-8624
D	6046	6111	-5188	-8616
D	6128	6191	-5170	-8602
D	6206	6270	-5158	-8576
D	6280	6349	-5142	-8546
D	6357	6427	-5122	-8536
D	6432	6503	-5120	-8534
D	6507	6580	-5128	-8538
D	6585	6659	-5136	-8536
D	6662	6741	-5146	-8546
D	6735	6821	-5166	-8566
D	6807	6901	-5178	-8582
D	6880	6979	-5200	-8596
D	6950	7057	-5210	-8612
D	7021	7132	-5216	-8616
D	7096	7207	-5220	-8618
D	7170	7279	-5220	-8628
D	7248	7353	-5200	-8628
D	7324	7424	-5192	-8602
D	7400	7493	-5172	-8590
D	7474	7563	-5162	-8560
D	7547	7634	-5138	-8520
D	7622	7704	-5120	-8500
D	7694	7776	-5112	-8480
D	7767	7849	-5112	-8486
D	7839	7920	-5118	-8512
D	7912	7991	-5130	-8536
D	7986	8065	-5150	-8564
D	8055	8136	-5170	-8586
D	8128	8206	-5196	-8598

D	8201	8278	-5206	-8614
D	8272	8350	-5218	-8632
D	8346	8427	-5222	-8650
D	8418	8500	-5224	-8650
D	8490	8574	-5226	-8664
D	8562	8649	-5218	-8662
D	8636	8726	-5216	-8658
D	8714	8805	-5190	-8642
D	8784	8877	-5150	-8614
D	8857	8950	-5108	-8574
D	8930	9023	-5064	-8534
D	9002	9101	-5016	-8498
D	9074	9177	-4988	-8464
D	9143	9256	-4982	-8458
D	9214	9332	-5000	-8470
D	9280	9406	-5024	-8486
D	9352	9479	-5060	-8516
D	9424	9552	-5082	-8538
D	9496	9628	-5110	-8566
D	9567	9701	-5122	-8588
D	9640	9775	-5142	-8604
D	9714	9850	-5156	-8614
D	9787	9922	-5162	-8630
D	9861	9997	-5168	-8630
D	9935	10071	-5182	-8630
D	10008	10144	-5166	-8622
D	10081	10218	-5162	-8618
D	10154	10292	-5154	-8606
D	10228	10367	-5140	-8604
D	10300	10441	-5124	-8592
D	10372	10512	-5106	-8576
D	10444	10584	-5094	-8568
D	10514	10656	-5088	-8556
D	10588	10730	-5072	-8540
D	10660	10803	-5058	-8536
D	10736	10879	-5064	-8538
D	10806	10952	-5054	-8536
D	10879	11026	-5040	-8528
D	10951	11100	-5044	-8528
D	11022	11175	-5044	-8518
D	11094	11246	-5034	-8516
D	11165	11321	-5036	-8510
D	11241	11395	-5034	-8504
D	11312	11470	-5034	-8504
D	11385	11544	-5044	-8496
D	11459	11619	-5056	-8494
D	11533	11690	-5054	-8498
D	11607	11761	-5076	-8512
D	11683	11829	-5092	-8526
D	11760	11901	-5098	-8536
D	11834	11974	-5114	-8538
D	11907	12050	-5108	-8538
D	11978	12123	-5096	-8534
D	12047	12192	-5084	-8512
D	12119	12264	-5064	-8492
D	12192	12339	-5044	-8478
D	12265	12412	-5030	-8456

D	12336	12485	-5038	-8456				
D	12405	12559	-5048	-8472				
D	12479	12633	-5070	-8488				
D	12550	12706	-5096	-8510				
D	12622	12776	-5126	-8530				
D	12691	12850	-5146	-8554				
D	12763	12924	-5158	-8584				
D	12832	12993	-5168	-8602				
D	12903	13065	-5186	-8614				
D	12974	13137	-5196	-8618				
G	-1	1	38.928988	0	97.364505	1	290806	153611
D	13044	13209	-5196	-8624				
D	13114	13280	-5188	-8610				
D	13183	13352	-5174	-8592				
D	13255	13424	-5144	-8560				
D	13327	13496	-5112	-8516				
D	13396	13566	-5080	-8490				
D	13466	13639	-5062	-8466				
D	13536	13709	-5058	-8446				
D	13604	13780	-5060	-8452				
D	13673	13850	-5076	-8484				
D	13743	13924	-5094	-8496				
D	13814	13994	-5120	-8528				
D	13879	14067	-5130	-8548				
D	13946	14134	-5150	-8564				
D	14011	14202	-5164	-8588				
D	14081	14270	-5184	-8604				
D	14150	14340	-5186	-8602				
D	14222	14410	-5186	-8622				
D	14291	14479	-5194	-8632				
D	14363	14546	-5188	-8620				
D	14434	14613	-5164	-8610				
D	14506	14682	-5136	-8576				
D	14579	14753	-5106	-8536				
D	14652	14824	-5076	-8512				
D	14724	14897	-5050	-8490				
D	14799	14970	-5040	-8468				
D	14872	15048	-5048	-8474				
D	14944	15122	-5066	-8494				
D	15016	15196	-5096	-8520				
D	15087	15271	-5118	-8550				
D	15160	15345	-5130	-8572				
D	15231	15418	-5150	-8598				
D	15307	15495	-5164	-8612				
D	15385	15573	-5170	-8628				
D	15462	15649	-5176	-8628				
D	15537	15726	-5182	-8618				
D	15614	15806	-5180	-8612				
D	15691	15887	-5150	-8598				
D	15769	15968	-5118	-8570				
D	15850	16050	-5090	-8532				
D	15930	16132	-5068	-8490				
D	16012	16216	-5052	-8466				
D	16092	16299	-5058	-8456				
D	16173	16380	-5068	-8472				
D	16254	16464	-5080	-8500				
D	16339	16546	-5108	-8526				

D	16420	16623	-5134	-8556
D	16500	16705	-5148	-8598
D	16584	16787	-5160	-8622
D	16667	16870	-5178	-8632
D	16750	16950	-5186	-8638
D	16836	17030	-5194	-8636
D	16923	17115	-5184	-8624
D	17007	17200	-5160	-8600
D	17088	17286	-5142	-8566
D	17172	17370	-5118	-8534
D	17255	17456	-5088	-8502
D	17339	17543	-5078	-8490
D	17424	17630	-5084	-8480
D	17507	17717	-5088	-8504
D	17590	17800	-5112	-8526
D	17674	17885	-5134	-8548
D	17758	17971	-5154	-8576
D	17846	18058	-5172	-8590
D	17934	18145	-5192	-8602
D	18017	18230	-5188	-8600
D	18102	18315	-5192	-8606
D	18187	18401	-5178	-8588
D	18272	18484	-5148	-8560
D	18356	18571	-5126	-8522
D	18440	18656	-5088	-8478
D	18520	18741	-5054	-8462
D	18599	18826	-5052	-8448
D	18682	18912	-5066	-8462
D	18762	18995	-5090	-8494
D	18843	19076	-5118	-8528
D	18926	19162	-5138	-8548
D	19010	19244	-5160	-8576
D	19094	19328	-5182	-8586
D	19179	19412	-5202	-8606
D	19264	19495	-5212	-8620
D	19349	19579	-5214	-8614
D	19435	19659	-5202	-8600
D	19520	19740	-5182	-8590
D	19605	19825	-5144	-8556
D	19690	19908	-5094	-8512
D	19770	19992	-5054	-8482
D	19851	20077	-5042	-8472
D	19936	20159	-5040	-8478
D	20016	20239	-5058	-8494
D	20098	20321	-5074	-8514
D	20179	20402	-5104	-8552
D	20259	20482	-5130	-8576
D	20339	20566	-5142	-8588
D	20418	20649	-5160	-8606
D	20496	20729	-5172	-8614
D	20575	20810	-5188	-8618
D	20652	20889	-5196	-8624
D	20728	20968	-5188	-8624
D	20804	21051	-5178	-8614
D	20880	21133	-5174	-8612
D	20956	21215	-5150	-8602
D	21035	21293	-5130	-8582



D	21109	21373	-5110	-8574
D	21183	21448	-5094	-8560
D	21261	21529	-5080	-8544
D	21336	21610	-5068	-8532
D	21418	21689	-5062	-8530
D	21502	21766	-5060	-8514
D	21582	21845	-5046	-8508
D	21664	21925	-5048	-8502
D	21745	22004	-5040	-8500
D	21825	22083	-5038	-8494
D	21907	22162	-5038	-8486
D	21992	22242	-5038	-8488
D	22072	22322	-5040	-8488
D	22150	22405	-5044	-8484
D	22228	22487	-5058	-8490
D	22304	22570	-5060	-8492
D	22381	22652	-5072	-8502
D	22460	22732	-5096	-8522
D	22540	22812	-5114	-8542
D	22619	22892	-5126	-8554
D	22694	22971	-5132	-8540
D	22769	23051	-5128	-8550
D	22847	23132	-5112	-8526
D	22922	23210	-5082	-8500
D	23001	23287	-5050	-8468
D	23080	23365	-5036	-8440
D	23155	23443	-5038	-8428
D	23232	23522	-5042	-8444
D	23308	23601	-5060	-8448
D	23387	23679	-5090	-8472
D	23466	23757	-5102	-8504
D	23543	23830	-5126	-8536
D	23622	23910	-5144	-8562
D	23698	23987	-5158	-8594
D	23776	24064	-5178	-8610
D	23857	24145	-5182	-8610
D	23934	24226	-5176	-8614
D	24016	24309	-5158	-8604
D	24095	24390	-5120	-8572
D	24171	24471	-5074	-8544
D	24250	24550	-5032	-8502
D	24328	24633	-5012	-8468
D	24404	24714	-4984	-8458
D	24480	24792	-4996	-8464
D	24559	24871	-5018	-8486
D	24633	24948	-5044	-8510
D	24706	25023	-5080	-8544
D	24779	25096	-5108	-8558
D	24855	25174	-5126	-8578
D	24928	25246	-5146	-8594
D	25004	25321	-5158	-8606
D	25078	25397	-5172	-8620
D	25151	25471	-5182	-8620
D	25225	25548	-5168	-8596
D	25300	25622	-5142	-8572
D	25376	25698	-5116	-8546
D	25449	25774	-5076	-8500

D	25527	25850	-5034	-8474				
D	25607	25928	-5024	-8442				
D	25684	26003	-5028	-8438				
D	25759	26075	-5038	-8460				
D	25834	26147	-5066	-8480				
D	25909	26222	-5088	-8504				
D	25986	26299	-5112	-8536				
D	26063	26374	-5132	-8562				
D	26142	26451	-5150	-8578				
D	26217	26526	-5162	-8594				
D	26295	26600	-5176	-8614				
D	26371	26677	-5170	-8624				
D	26447	26757	-5174	-8628				
D	26524	26834	-5156	-8614				
D	26600	26912	-5122	-8582				
D	26676	26993	-5084	-8532				
D	26753	27074	-5048	-8484				
D	26828	27155	-5012	-8448				
D	26904	27236	-5006	-8438				
D	26981	27315	-5026	-8440				
D	27056	27391	-5048	-8452				
D	27134	27468	-5078	-8478				
D	27212	27548	-5104	-8506				
D	27290	27624	-5130	-8538				
D	27366	27702	-5142	-8568				
D	27442	27779	-5160	-8586				
D	27518	27853	-5170	-8606				
D	27596	27927	-5176	-8612				
D	27676	28002	-5174	-8620				
D	27751	28075	-5192	-8612				
D	27826	28148	-5156	-8578				
D	27902	28225	-5106	-8538				
D	27975	28303	-5052	-8484				
D	28051	28380	-5012	-8444				
D	28131	28457	-4994	-8422				
D	28206	28533	-4992	-8426				
D	28279	28609	-5004	-8444				
D	28357	28689	-5032	-8468				
D	28434	28767	-5058	-8490				
D	28511	28846	-5096	-8520				
G	-1	1	38.928982	0	97.364587	1	290806	153621
D	28588	28922	-5116	-8544				
D	28666	29001	-5136	-8562				
D	28742	29077	-5152	-8574				
D	28817	29153	-5160	-8574				
D	28891	29226	-5162	-8578				
D	28967	29301	-5164	-8582				
D	29041	29377	-5152	-8564				
D	29114	29450	-5130	-8550				
D	29188	29525	-5086	-8510				
D	29261	29599	-5042	-8464				
D	29333	29674	-5004	-8446				
D	29403	29748	-4998	-8432				
D	29475	29824	-5008	-8446				
D	29548	29899	-5024	-8464				
D	29626	29977	-5056	-8496				
D	29698	30052	-5084	-8516				

D	29770	30128	-5102	-8536				
D	29843	30205	-5124	-8562				
D	29918	30284	-5146	-8590				
D	29990	30361	-5156	-8606				
D	30062	30434	-5168	-8618				
D	30139	30509	-5176	-8620				
D	30214	30585	-5166	-8616				
D	30291	30663	-5150	-8598				
D	30363	30739	-5114	-8560				
D	30437	30814	-5060	-8520				
D	30511	30889	-5016	-8470				
D	30583	30960	-4982	-8432				
D	30664	31037	-4962	-8412				
D	30744	31113	-4972	-8420				
D	30819	31188	-4994	-8436				
D	30895	31264	-5026	-8470				
D	30972	31342	-5060	-8492				
D	31050	31422	-5086	-8520				
D	31127	31502	-5116	-8544				
D	31206	31581	-5150	-8566				
D	31284	31660	-5160	-8566				
D	31360	31737	-5160	-8582				
D	31434	31816	-5164	-8580				
D	31506	31892	-5166	-8572				
D	31578	31969	-5160	-8578				
D	31654	32045	-5140	-8568				
D	31729	32121	-5118	-8556				
G	-1	1	38.928973	0	97.365855	1	290806	153624
Z	-38958.4441319444	-38958.4441319444	-5274	-8700				
Z	-38958.4441319444	-38958.4441319444	-5272	-8700				
Z	-38958.4441319444	-38958.4441319444	-5274	-8706				
Z	-38958.4441319444	-38958.4441319444	-5280	-8698				
Z	-38958.4441319444	-38958.4441319444	-5278	-8700				
Z	-38958.4441319444	-38958.4441319444	-5272	-8696				
Z	-38958.4441319444	-38958.4441319444	-5286	-8688				
Z	-38958.4441319444	-38958.4441319444	-5288	-8696				
Z	-38958.4441319444	-38958.4441319444	-5286	-8698				
Z	-38958.4441319444	-38958.4441319444	-5282	-8696				
Z	-38958.4441319444	-38958.4441319444	-5282	-8694				
Z	-38958.4441319444	-38958.4441319444	-5278	-8694				
Z	-38958.4441319444	-38958.4441319444	-5274	-8692				
Z	-38958.4441319444	-38958.4441319444	-5268	-8698				
Z	-38958.4441319444	-38958.4441319444	-5270	-8692				
Z	-38958.4441319444	-38958.4441319444	-5274	-8690				
Z	-38958.4441319444	-38958.4441319444	-5272	-8698				
Z	-38958.4441319444	-38958.4441319444	-5276	-8712				
Z	-38958.4441319444	-38958.4441319444	-5276	-8702				
Z	-38958.4441319444	-38958.4441319444	-5274	-8702				

## CAL\_5\_JUL06.cal

0	-4762	-8800
0	-4762	-8792
0	-4760	-8786
0	-4766	-8792
0	-4764	-8788
0	-4762	-8778
0	-4768	-8780
0	-4756	-8774
0	-4758	-8772
0	-4768	-8774
0	-4768	-8772
0	-4772	-8776
0	-4770	-8780
0	-4774	-8776
0	-4778	-8780
0	-4780	-8782
0	-4770	-8780
0	-4770	-8776
0	-4772	-8782
0	-4776	-8786
2	26654	22588
2	26660	22588
2	26668	22578
2	26668	22586
2	26666	22590
2	26670	22584
2	26668	22578
2	26660	22580
2	26666	22578
2	26660	22574
2	26662	22588
2	26668	22580
2	26664	22582
2	26668	22588
2	26664	22586
2	26652	22572
2	26650	22574
2	26648	22572
2	26644	22574
2	26642	22584
4	-1016	-5002
4	-1018	-5008
4	-1014	-5014
4	-1018	-5006
4	-1014	-5004
4	-1010	-5006
4	-1004	-5002
4	-1000	-5002
4	-1000	-5006
4	-1002	-5002
4	-1002	-5006
4	-1000	-5008
4	-1000	-5006

4	-1002	-5006
4	-1000	-5008
4	-1012	-5000
4	-1010	-5010
4	-1006	-5012
4	-1014	-5004
4	-1004	-5004
6	-4024	-8066
6	-4016	-8074
6	-4022	-8076
6	-4018	-8074
6	-4006	-8078
6	-4018	-8072
6	-4022	-8066
6	-4022	-8070
6	-4016	-8060
6	-4024	-8066
6	-4018	-8056
6	-4016	-8052
6	-4010	-8052
6	-4004	-8056
6	-4004	-8052
6	-4012	-8062
6	-4010	-8068
6	-4022	-8068
6	-4022	-8068
6	-4024	-8062
8	-4578	-8628
8	-4578	-8632
8	-4574	-8626
8	-4580	-8632
8	-4576	-8634
8	-4574	-8622
8	-4580	-8618
8	-4582	-8620
8	-4574	-8614
8	-4576	-8618
8	-4572	-8620
8	-4578	-8612
8	-4586	-8608
8	-4578	-8602
8	-4578	-8606
8	-4588	-8604
8	-4582	-8608
8	-4584	-8606
8	-4588	-8604
8	-4584	-8606
10	-4704	-8732
10	-4706	-8744
10	-4708	-8736
10	-4704	-8738
10	-4706	-8742
10	-4706	-8740
10	-4706	-8740
10	-4712	-8736
10	-4716	-8742
10	-4710	-8742

10	-4716	-8748
10	-4712	-8750
10	-4710	-8762
10	-4708	-8770
10	-4712	-8760
10	-4714	-8746
10	-4724	-8742
10	-4722	-8742
10	-4720	-8752
10	-4718	-8750
12	-4744	-8782
12	-4752	-8786
12	-4746	-8784
12	-4746	-8788
12	-4752	-8788
12	-4754	-8786
12	-4752	-8782
12	-4754	-8780
12	-4748	-8776
12	-4744	-8784
12	-4746	-8786
12	-4744	-8782
12	-4740	-8780
12	-4732	-8780
12	-4736	-8766
12	-4734	-8772
12	-4740	-8772
12	-4738	-8776
12	-4744	-8778
12	-4744	-8784
0	-4758	-8802
0	-4758	-8800
0	-4772	-8808
0	-4770	-8806
0	-4768	-8806
0	-4772	-8806
0	-4766	-8802
0	-4764	-8802
0	-4772	-8798
0	-4778	-8792
0	-4774	-8792
0	-4780	-8790
0	-4774	-8788
0	-4766	-8796
0	-4770	-8794
0	-4770	-8800
0	-4766	-8796
0	-4770	-8790
0	-4770	-8796
0	-4774	-8800

