To the Graduate Council

I am submitting herewith a thesis written by Seth Wrenne Rye entitled "Variable Application Rate Control System for Field-Scale Research " I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Biosystems Engineering

John B Wilkerson, Major Professor

We have read this thesis
and recommend its acceptance

Accepted for the Council

Interim Vice Provost

Variable Application Rate Control System for Field Scale Research

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Seth Wrenne Rye

August 2000

ACKNOWLEDGEMENTS

I thank my major professor, Dr. John B Wilkerson, for his guidance, support, and patience. I also thank Dr William E Hart for his constant help, friendship, and confidence in me. I appreciate practical guidance from Dr Paul Denton during this project I extend a special thanks to Henry Moody whose hard work and solid engineering skills made this project's computer hardware a reality. I also appreciate Henry's friendship throughout my career at the university I'm grateful for the excellent workmanship, timely assistance, and technical expertise of the Agricultural Engineering Shop Paul Elliot, Craig Wagoner, and Walker Garner each contributed greatly to the fabrication of equipment used in this project The Agricultural Engineering Department as a whole deserves credit for providing a friendly, professional work environment

I thank the following people for participating in field testing of the system developed in this project Leroy Leonard, Curt McDaniel, Will Pinson, Tim Morrow, and Newman Webb I appreciate the help of Lee Ellis and the Knoxville Experiment Station for providing a field testing location. I thank the many others who helped me with this project including Dr Ruixi Sui, Dr Raj Raman, David Russell, and Jeff Sowders I also appreciate the great friendship and support of Pam Korczynski during completion of my degree

Most importantly, I thank my family for their tremendous support, role modeling, and confidence in me throughout my education.

ABSTRACT

The need for rapid implementation of large, multi-variable field experiments, hindered by limited application-map design software, and inapplicable commercial variable-rate control systems led to the development of the Tennessee Rate Application Control System (TRACS), a variable-rate application (VRA) system devoted specifically for agricultural research TRACS was developed through the design and integration of two separate modules (sub-systems). One module, known as the management system, was created to aid in the design of plot-scale experiments Spatial field data, including boundary coordinates and crop row orientation, was required by this off-site computer based sub-system. The other TRACS module, the field system, was organized for direct implementation of plot-scale experiments created with the management system. This rugged field computer system was built to communicate in real-time to a variable-rate controller. The field system was programmed to execute the VRA treatment plans composed by researchers using the TRACS management system. An overall performance evaluation suggested that TRACS could be used for rapid, labor-reduced design, implementation, of field research experiments.

TABLE OF CONTENTS

Chapter 1: Introduction and Background 1
Introduction 1
Review of Literature 2
Desired Application Map Design 3
Data Exchange 4
Field Control Systems 4
Objectives 8
TRACS Design Overview 8

Chapter 2: Design of a Computer-Based Management System 12
Design Requirements 12
Program Structure 13
Obtaining Test Area Coordinates 13
Translating Coordinates 17
Plot Layout 19
Example of Resulting Software 20
Menu 20
Rate Legend and Experiment Size Display 20
Border Width Adjustment and Color-Coded Field Layout 22

Chapter 3: Design of a Field Control System 23
Design Requirements 23
Hardware Design and Implementation 25
PCMCIA Socket 28
Keypad 28
LCD Display 28
Variable-Rate Controller 29
GPS Receiver 29
LED Light-Bar Display 30
Software Design 31
Program Structure 34
Obtaining GPS Data 35
Converting Data 35
Rotating Data 37
Calibrating Data 37
Buffering Data 40
Calculating Navigation Information 40
Calculating Vector Velocities 40
Feed-Forward Error Rejection 44
Finding the Closest Plot 46
Making Rate Control Decisions and Issuing Set Points 46
Calculating ETA 48
Running Count-Down Timer in Parallel 49
Checking for Previously Covered Areas 49

Updating Displays .. .. .49
Updating LED Light-Bar . ... .51
Updating LCD Screen 51
Storing Performance Data.... 52
Example of Resulting Software..... 52

Chapter 4: Performance Evaluation.... .... .55
Positional Accuracy 55
Equipment..... 56
DPE..... ... .56
Establishing a Rate-Change Line 58
Testing 58
Results. 58
User Survey.. 63
Overview..... 63
Results... 64

Chapter 5: Summary..... 65
Summary .... .65
Conclusions..... 65
Recommendations for Further Research and Testing .... .66
Bibliography..... 67

Appendices..... 70
Appendix A TRACS File Structures .. .. .71
Appendix B. Management System Software Code. .... .75
Appendix C. Coordinate Collector Software Code 85
Appendix D. Field System Software Code ... .93

Vita ... .118

LIST OF FIGURES

Figure 1-1 Basic Variable Rate Application (VRA) Equipment 5

Figure 1-2. TRACS Overall Design Requirements 9

Figure 1-3. TRACS Design Overview.... 10

Figure 2-1. TRACS Management System Design Requirements .... .14

Figure 2-2. Graphical Illustration of System Algorithm 15

Figure 2-3. Management System GUI.. 21

Figure 3-1. TRACS Field System Design Requirements 24

Figure 3-2. TRACS Field System Hardware Schematic 26

Figure 3-3. TRACS Field System Hardware .... .27

Figure 3-4 LED Light-Bar Schematic 32

Figure 3-5. Field System Software Functions 33

Figure 3-6 Procedure to Read in GPS Data 36

Figure 3-7 TRACS Field System Calibration Curve and Procedure .... .39

Figure 3-8. Graphical Illustration of Data Buffering and Data Buffering Routine. .41

Figure 3-9. Calculation and Use of Vector Velocities for Feed-Forward Control. . .43

Figure 3-10 Feed-Forward Error Rejection 45

Figure 3-11 Routine for Finding the Closest Plot Center . ... .47

Figure 3-12. Parallel Count-Down Timer.. 50

Figure 3-13. Field System GUI 53

Figure 4-1. TRACS Field Test Apparatus .... .57

Figure 4-2 Illustration of Dynamic Positional Error (DPE).. 59

Figure 4-3 TRACS Absolute Dynamic Positional Error vs. Speed 60

Figure 4-4. TRACS Average Dynamic Positional Error vs. Speed 62

CHAPTER 1

INTRODUCTION AND BACKGROUND

INTRODUCTION

In attempts to develop prescription equations that predict the amounts of field amendments needed by small sections of large fields, agricultural researchers require massive amounts of spatial data. Many large-scale experiments with multiple variables must be conducted to produce this type of data. If enough data can be obtained to form a meaningful relationship among the variables, a prescription or recommendation can be determined For example, a nitrogen prescription might be a function of several variables:

$$N = f(Y, O_M, P_P, S_T)$$

where

N = nitrogen application rate (lb/ac)
O_M = soil organic matter (%)
Y = yield goal (lb/ac)
P_P = plant population (ft)
S_T = soil type

Development of such a prescription requires extensive analysis of the variables involved This type of analysis would likely result from several large experiments including many experimental units (field plots). Implementation of large numbers of plots may not be feasible using current field research techniques and technology.

Two areas of intense labor exist when dealing with experimental plots· plot layout and treatment application Traditional plot-layout involves using survey equipment and a crew of at least two people Items such as optical squares, crossheads, 3-4-5 triangles, sextants, prismatic

compasses and survey chains are all involved in the process (Dyke, 1988). Marking flags used to identify boundaries can be a nuisance at harvest time. Plot layout techniques are much too time consuming for many large-scale experiments. If boundary flags, conventional surveying equipment, and the manpower needed to physically layout plots could be reduced, experiments at this magnitude would become more feasible.

Rapid implementation of experiments requires intensive labor not only while laying out and making plot boundaries, but while administering treatment applications. The small plot equipment needed for very precise chemical applications is simply inadequate for large experiment deigns Slow chemical application methods could cause problems in research if the timing of treatment is critical. This illustrates the need for rapid implementation of treatments.

A time and labor saving solution to many modern day problems is the computer. While computer-controlled applicators have the potential to minimize the labor involved in plot-layout and treatment applications, the current state of the art equipment does not provide this flexibility. Although commercial systems are currently used to provide variable-rate capabilities to production farmers, no hardware and software system is specifically set up to aid in research plot layout nor to deliver variable rates to research plots An explanation and review of literature on the variable rate application process is necessary to understand why the commercial software is not optimal for research purposes

REVIEW OF LITERATURE

A technique known as variable rate application (VRA) has the potential to assist in agricultural research. This technique allows for application of field amendments to be changed or varied throughout a field As illustrated in the earlier example, many amendments may be required for the prescription development process Therefore, variable rate application of these

amendments is needed Variable rate application is commercially practiced with many amendments including lime (Reichenberger, 1997a), herbicide (Holmberg, 1998), seeds (Reichenberger, 1997b), granular fertilizers (Fulton, 1999), liquid fertilizers (Yang et al., 1998), animal manures and irrigation (Searcy, 1995) Pesticides such as nematicides have also been applied using variable rate technology (Wrather et al., 1998). As discussed in the example, the development of prescription equations requires large multi-variable studies. The wide array of VRA types is promising for implementation of experiments involving multiple variables.

In order to apply different treatments to different areas in a field, a method of making control decisions must be established According to Sudduth (1999), commercial systems come in two basic forms· Map-based systems and sensor-based systems. Map-based systems require the user to collect information about the field, process it off-site, and then bring the processed data back to the field for use in application equipment. The offsite process is often referred to as the development of a "desired application map" because the system user can specify where particular rates are to be assigned spatially (Sudduth, 1999) Sensor based systems rely on no prior information about the area but rather gain information from real time sensors that form the bases for the control decisions A system to be used for field plot research would obviously be map-based to allow the user full control of plot placement and the assignment of randomized treatments.

Desired Application Map Design

In the map-based approach, designing the desired application map is the first step toward variable rate control. Map making can be done with a GIS (geographic information system) or some type of more dedicated mapping software (such as from the manufacturer of a VRA system) Many systems are capable of dividing fields into management zones Each management zone is assigned a homogenous treatment Commercial software displays these zones in the form

of maps. These maps illustrate areas that are managed differently from one another As Sudduth explains, using this software for any purpose other than the one intended by the manufacturer may be "exceedingly difficult" (Sudduth, 1999). Since commercially-available software is designed only with production agriculture in mind, difficulties occur when attempting to layout plots within large fields. In commercial systems, according to Clark and McGuckin (1996), GIS software must be designed to produce maps giving coordinates of field boundaries and locations of application areas. The desired application map must then be transformed into a form usable by an on-board field computer in an application vehicle (sprayer, spreader truck, etc.). This exchange of data is a point of concern for usability of VRA systems.

Data Exchange

Data exchange refers to transferring the application map from the map design software and to a field computer. This exchange can be difficult. Sudduth describes this limitation of current systems as "rudimentary at best." The same problems hold true when trying to create a desired application rate map for a research scale system. As troubling of an issue as this may be, a solution is being pursued One organization interested in organizing standards for the data transfer process is the Ag Electronics Association (AEA). This organization intends to automate both import and export processes, but may not have effects on map-designing software for some time. As noted by the AEA, another development effort known as OpenGIS (http://www.opengis.org), also has the potential to offer a solution for the data exchange dilemma (AEA, 1998)

Field Control Systems

After map development and data export, the data must be imported and used by a field control system This control system serves as an interpreter for the information contained in the desired application map. Figure 1-1 (B) represents a generic field control system As illustrated,

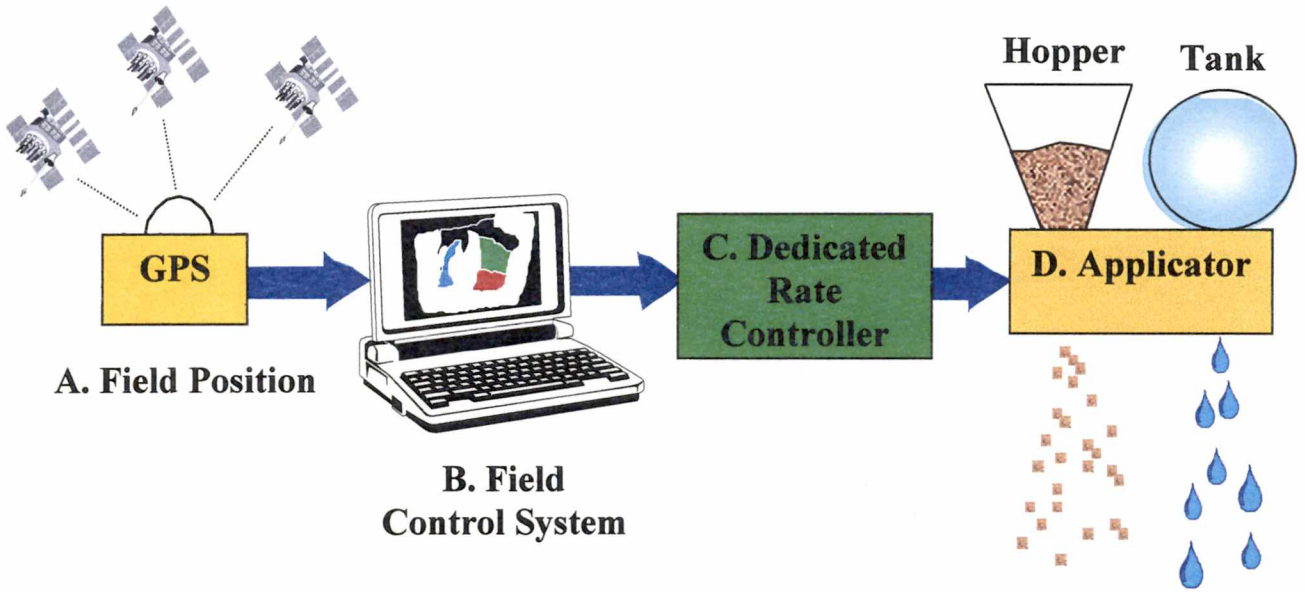


Figure 1-1. Basic Variable-Rate Application (VRA) Equipment

the field control system interprets the digital desired application map, relates the map to a GPS-generated field position (A), and communicates that information in real-time to a rate controller (C) in the form of control set-points. A set-point is a value that the rate controller tries to achieve by adjusting an actuator device that can throttle output of some liquid or granular material (D). The responsibility of the field control system is to issue these set-points. Dedicated rate controllers do the actual adjustment of the crop input.

Several field control systems have been recently developed. Large companies such as Ag-chem (Ag-chem Equipment Company, 1998), Rockwell, and Willmar (Bielke et al., 1997) have taken part in the development of commercial systems. Several universities including The University of Georgia (Clark and McGuckin, 1996) and Colorado State University (Rockwell and Ayers, 1994) have done background development. Despite being designed for production agriculture usage, these systems are based on the same technology as the proposed system

A goal of VRA systems is positional and rate accuracy. Many systems work fairly well on fields where particular application rates are associated with large areas in a field. Although high degrees of positional accuracy may not be necessary in production farm fields, the Rockwell-Willmar spraying system erred by as much as 18 ft from the desired area of application (Bielke et al., 1997) A positional error this massive is unacceptable for most research applications. The accuracy of the Ag-chem system was sharply decreased at lower speeds and application rates (Yang et al., 1998). A research control system must be accurate at multiple ground speeds and a wide dynamic range of application rates.

Software techniques have been employed in some systems to minimize positional errors associated with rate control hardware limitations. One such problem, known as "lag time" refers to the difference in time between a commanded rate change and when the rate actually reaches the desired set-point One such technique is known as look-ahead or feed-forward control Sudduth (1997) describes this method as the ability "to 'look-ahead' at the application rates

needed at future locations in the field to minimize errors due to lag times." Schueller and Wang (1994) found that this type of control is the most accurate when compared to the other general control theories (open-loop and closed-loop). Feed forward control is employed on several commercial systems such as SECTOR (Bielke et al., 1997) which claims the ability to predict lag time.

Although, efforts to offset lag times are made on several commercial units, another issue plagues the commercial systems' GPS update frequency. The Global Positioning System (GPS) is used as a location sensor for most commercial field control systems. Thus, GPS is the means by which field control systems determine their location within fields. Although, some GPS units send out positional information very rapidly (20 Hz), the most common update rate is 1 Hz, meaning that these units output positional data once every second For most static operations such as surveying, this is not a problem. However, in dynamic VRA systems it becomes important. A speed of 12 mph in a chemical applicator means that almost 18 ft will be covered between 1 Hz GPS updates Most commercial software has no method of extrapolating data between GPS updates and therefore could miss desired rate change targets by as much as 18 ft (at 12 mph) (Bielke et al., 1997). For a 60-ft long research plot, that would translate into a coverage error of 30% Compensating for slow GPS update rates is another area in which feed-forward could be applied to allow VRA systems to be used in research plots

Limited application-map design software, insufficient data exchange standards, and inaccurate commercial field control systems illustrate the need for a system devoted specifically for agricultural research. The Tennessee Rate Application Control System (TRACS) was developed in response to this need.

OBJECTIVES

This project includes three main objectives in the development of the TRACS

1. Develop a computer-based management system for the design of plot-scale experiments based on field parameters including field boundary and row orientation.

2. Design a field control system for direct implementation of plot-scale experiments created with the computer-based management system specified in objective 1 The system should communicate in real-time to a variable-rate controller (liquid flow rate, plant population, etc.)

3. Evaluate the performance of both systems with special detail given to dynamic positional error (DPE).

A chapter is devoted to the accomplishment of each of these objectives.

TRACS DESIGN OVERVIEW

TRACS has been constructed for the purpose of designing and executing field plot experiments involving multiple treatments. TRACS is broken up into two separate sub-systems: a management system and a field system (Figure 1-2). The management system (off-site), which can be installed on a personal computer, is used to design the layout of an experiment. The management system aids the user in forming a desired application rate map (Figure 1-3 (A)) by collecting and analyzing boundary coordinates and row orientation of the field in which the experiment will be conducted and by accepting user inputs involving the number and types of treatments needed All maps have rates randomly assigned to the research plots to form a completely randomized design (CRD) The field system (on-site), which takes the form of a mobile control console, receives the desired application map from the management system and aids the operator of the application device (sprayer, fertilizer spreader, etc.) in correctly applying treatments to the assigned locations (B). The field system stores performance data that can later

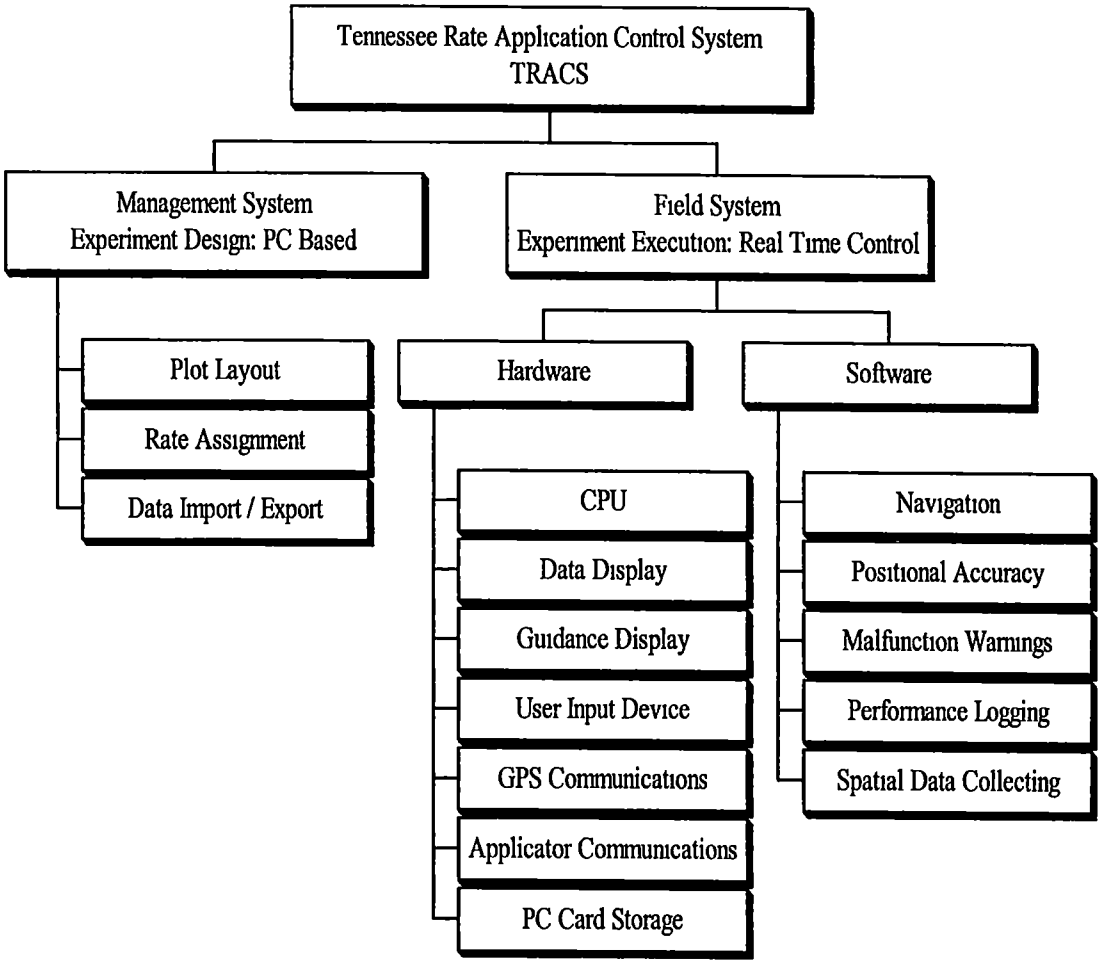


Figure 1-2 TRACS Overall Design Requirements

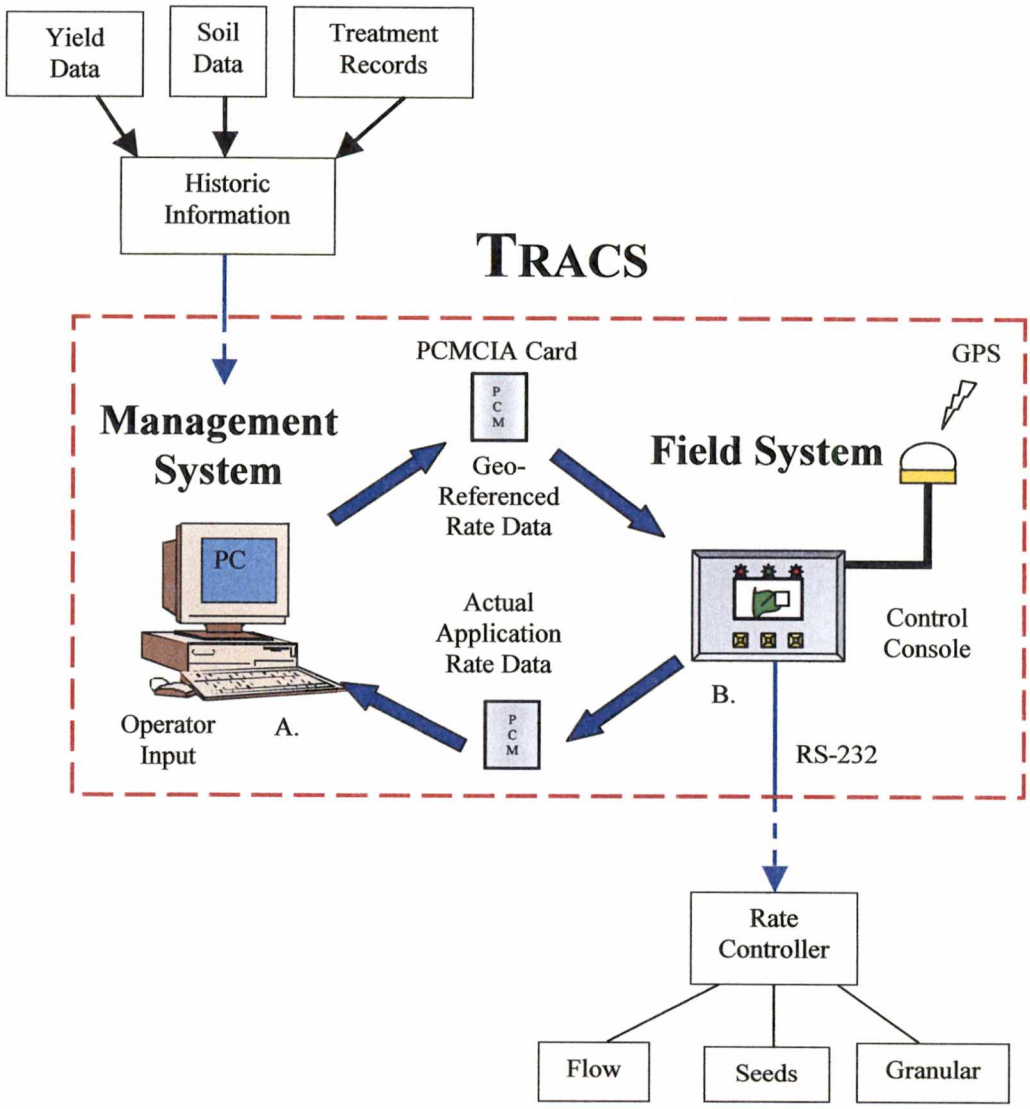


Figure 1-3. TRACS Design Overview

be reviewed in a GIS system.

CHAPTER 2

DESIGN OF A COMPUTER BASED MANAGEMENT SYSTEM

DESIGN REQUIREMENTS

The first objective of this project was to develop a computer-based management system for the design of field plot experiments based on two parameters. field boundary coordinates and row orientation. This sub-system of TRACS, referred to as the management system, bears strong distinction from the other sub-system (field system). While the field system is a computer system that conducts the actual application of treatments to an array of research plots, referred to as a test area, the management system is a computer-based planning tool used to compose CRD experiments. The management system provides an off-site method of designing field plot experiments using field parameter data collected on site. Each field plot experiment takes the form of a desired application rate map in the management system. This map shows the desired application rate (treatment) for each plot in the test area. The management system allows the user to tailor this map to the exact specifications of the field experiment being designed. The desired application rate map is translated into a format directly compatible with the field system. The field system uses the map as a guide to apply the correct treatments in the correct plots. The intended users of this sub-system are researchers interested in evaluating variable levels of crop inputs where a large number of experimental units (plots) are required. Specific users might include but are not limited to scientists in agricultural companies, universities, or other research organizations For clarity in this thesis, "the user" refers to the type of researcher mentioned above.

The design requirements of the management system are illustrated in figure 2-1. This figure also depicts the separate nature of the sub-system within TRACS. The management system was designed to accomplish three main tasks shown in Figure 2-1. The management system must first be capable of laying out research plots within an experimental area. Secondly, the system must assign randomized treatment rates to the research plots Finally, a means of data transfer to and from field system must be established.

PROGRAM STRUCTURE

To meet the specified design requirements, the following algorithm was developed for the management system

1. Obtain test area boundary coordinates and row orientation data.
2. Translate boundary coordinates obtained in step 1 from world coordinates to a local coordinate system
3 Delineate plots inside of field boundary and assign a randomized treatment rate to each.

Each step in the algorithm is defined in detail in the following sections.

Obtaining Test Area Coordinates

In order to aid the user in designing an experiment in a specified area (herein referred to as the test area), the management system must know the size, shape, and orientation of the test area. This information can be determined from two sets of world coordinates (latitude, longitude). The first set of coordinates represents the field boundary (Figure 2-2, A). These coordinates can be obtained by traveling the perimeter of the test area with a GPS receiver and logging the latitude and longitude of points along the path These coordinates allow the management system to determine the maximum size of the test area. By knowing the maximum size, the system can determine how many plots fit in the area The coordinates also define the

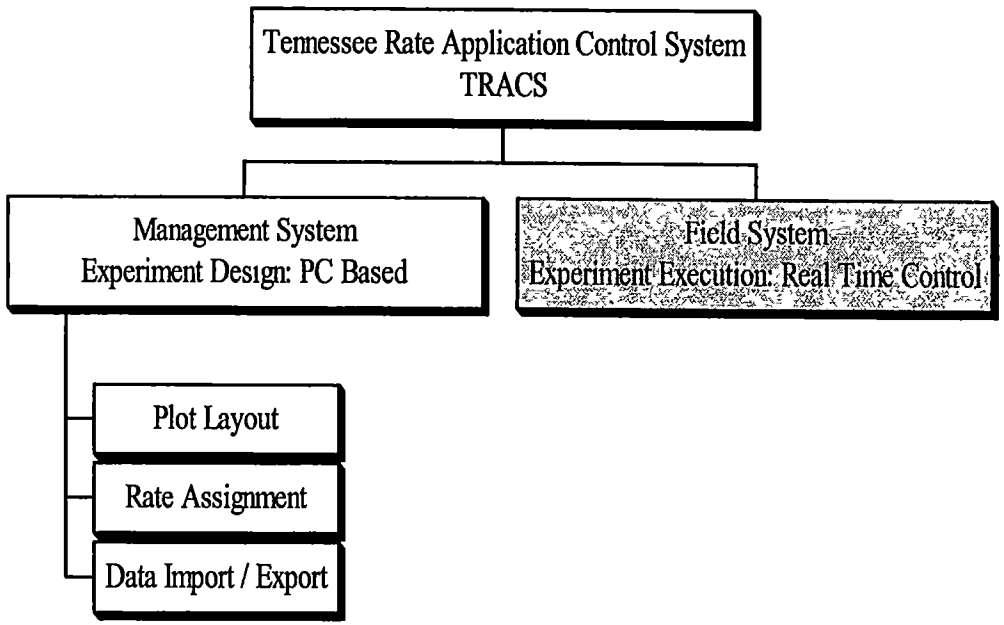


Figure 2-1 TRACS Management System Design Requirements

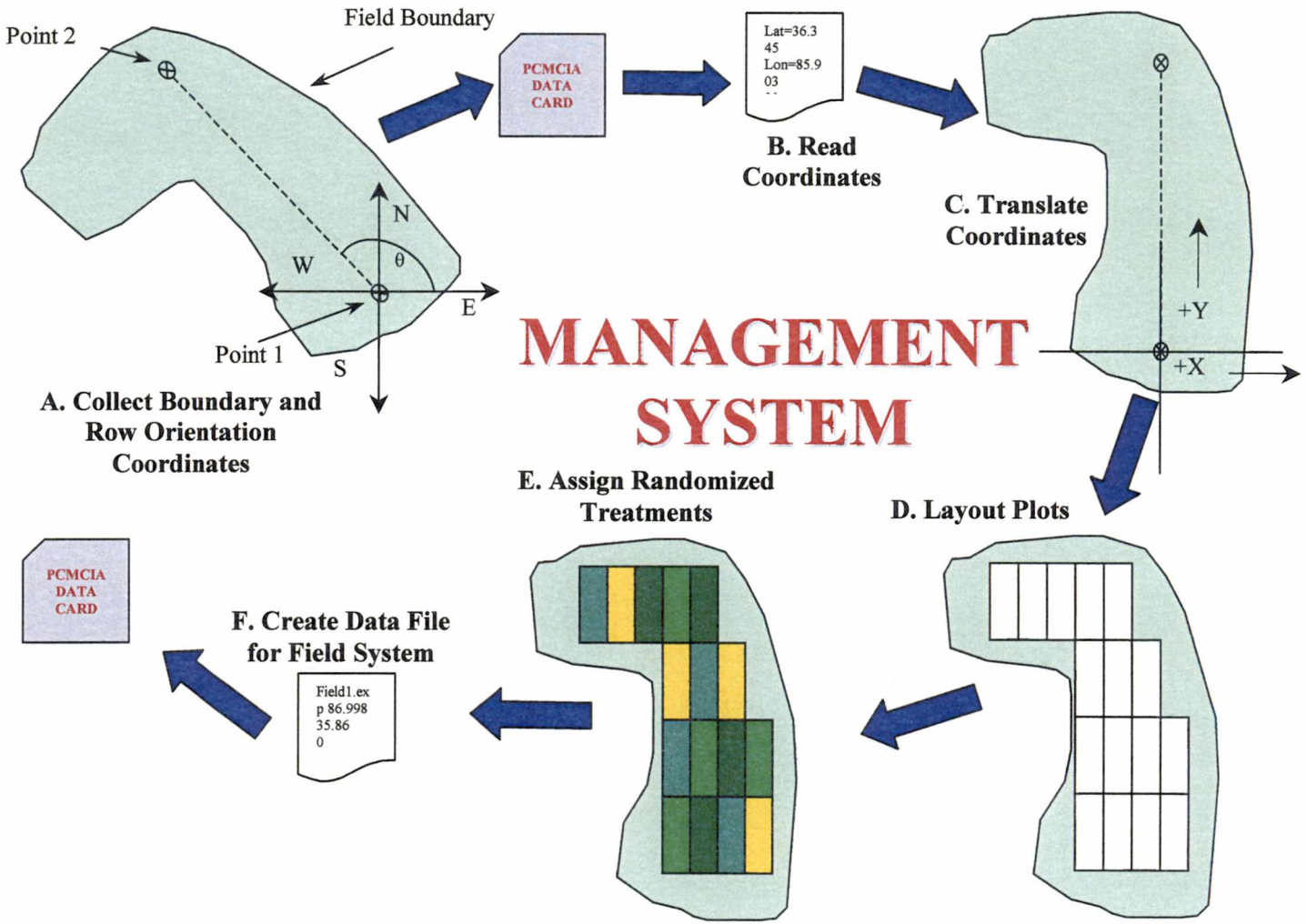


Figure 2-2. Graphical Illustration of Management System Algorithm

shape of the area, which permits TRACS to exclude plots that lie on the outside of the shape boundary. The second set of coordinates represents the orientation of the crop rows within the test area. Knowing which direction the rows run is necessary for proper plot layout If row orientation was not known, plots could still be laid out in a random orientation, but this would obviously cause problems when the plots were treated The application device might have to cross the rows perpendicularly, which would destroy many plants; thus, row orientation is an absolutely necessary. The row orientation data set consists of the world coordinates of two points (Point 1 and Point 2 (A)). A GPS receiver can again be used to collect the coordinates, but this time, average values are taken at each point to ensure that an accurate orientation are determined.

The management system requires that all data be in world coordinates (decimal degrees) stored in tab delimited ASCII files A data collection routine was designed specifically for this purpose. It was installed in the field system control console (but also works on a laptop computer). To use the routine, a GPS receiver is connected to the control console as described in chapter 3 The routine is highlighted in the following algorithm.

1. Collect field boundary points
2. Collect point 1
3 Collect point 2
4 Display data
5. Create spatial data file

In the first step, the software begins logging GPS points in world coordinates and prompts the user to drive along the field boundary The user can watch as the path is displayed on the field system display device and can command the system to stop collecting boundary data when the path is complete. Next, the user is prompted to locate the 2 reference points and collect GPS data on the points for a pre-defined period of time. The routine then averages the GPS data collected on both points separately and graphically displays the 2 reference points along with the field boundary. This allows the user to determine if the data collected is suitable for a proposed experiment. Finally, a spatial data file containing the boundary and reference point coordinates is

created by the system. This file is stored on removable media and can be transported to a personal computer on which the management system design software is installed.

Translating Coordinates

The second step of the management system algorithm is to translate the coordinates gathered with the test area coordinate collector into a local coordinate system (C). This translation not only involves conversion from world to Cartesian coordinates, but also requires rotation of the points to allow for optimum viewing angle by TRACS users Rotation also simplifies calculations made by the software This conversion from world coordinates (longitude, latitude) to a meaningful local coordinate system (X, Y) is performed by the management system by first calculating the row orientation angle (with respect to the east axis)

To simplify management system calculations and to provide row orientation data to the field system, a rotation angle (θ) must be calculated. This angle is calculated from point 1 and point 2 Point 1 is set as the intersection point of the axes A line is traced from point 1 to point 2. The angular difference (measured in a counter-clockwise rotation) between the traced line and the east axis is θ. Thus, if point 2 fell directly along the east axis, θ would be 0° or if point 2 fell directly along the south (S) axis, θ would be 270°. Figure 2-2 (A) shows θ with respect to the east (E) axis. Making the θ calculation requires that point 1 becomes the origin (0,0) and point 2 be converted from world coordinates to spherical Cartesian coordinates by Equations 2-1 and 2-2 (Van Sickle, 1996).

$$X_2 = 365157.480315 * \cos(lat_1) * (lon_2 - lon_1) \quad \text{Equation (2 - 1)}$$

$$Y_2 = 363188.976378 * (lat_2 - lat_1) \quad \text{Equation (2 - 2)}$$

where

X_2 = point 2 spherical Cartesian X coordinate (ft)

Y_2 = point 2 spherical Cartesian Y coordinate (ft)

lat_1 = point 1 latitude (decimal degrees)

lon_1 = point 1 longitude (decimal degrees)

lat_2 = point 2 latitude (decimal degrees)

lon_2 = point 2 longitude (decimal degrees)

After conversion, the sine and cosine of θ are calculated (Equations 2-3 and 2-4). These are the values that are actually used in all further calculations involving θ, because they allow the computer to resolve angles in all four quadrants of the unit circle.

$$\cos\theta = \frac{X_2}{\sqrt{X_2^2 + Y_2^2}} \quad \text{(Equation 2 - 4)}$$

$$\sin\theta = \frac{Y_2}{\sqrt{X_2^2 + Y_2^2}} \quad \text{(Equation 2 - 5)}$$

After calculating the rotation angle, the management system rotates the test area so that the row direction defined by points 1 and 2 is parallel to the Y-axis and perpendicular to the X-axis (C) The points defining the field boundary are read into the management system from the spatial data file and are immediately converted to spherical Cartesian coordinates according to Equation 2-1 (use field boundary points in place of point 2 for field boundary conversion). The points are then rotated according to Equations 2-5 and 2-6.

$$X_{br} = X_{bu} \cos \theta + Y_{bu} \sin \theta \quad \text{(Equation 2 - 5)}$$

$$Y_{br} = Y_{bu} \cos \theta + X_{bu} \sin \theta \quad \text{(Equation 2 - 6)}$$

where

X_{br} = rotated boundary X coordinate (ft)

Y_{br} = rotated boundary Y coordinate (ft)

X_{bu} = original boundary X coordinate (ft)

Y_{bu} = original boundary Y coordinate (ft)

Plot Layout

The next step for the management system is to delineate plots within the field boundary To accomplish this, the maximum distance between boundary points, both in the X and Y directions is found. The user is prompted to specify the size of experimental plots. Then the test field is inscribed inside a grid of plots. The system trims out plots that conflict with the field boundary. Figure 2-2 shows an example plot layout (D). The management system also insures that its proposed plot layout is the proper dimensions for a CRD (Pearce, 1983).

The management system also allows for user input after the test area is laid out By using a PS2 mouse input device, users can highlight plots to remove from the test area or highlight areas in which plots should be removed.

After the test area is configured, the management system assigns user selected treatments at random to the plots Figure 2-2 illustrates a randomized test area (E) Each time a user modification is made, the system re-randomizes the treatment assignment.

When all modifications are made, the system writes a tab delimited data file (F) which includes

1. Rotation angle information (sin θ, cos θ)
2 Reference point in world coordinates (lon_1, lat_1)
3 Plot locations in rotated spherical Cartesian coordinates
4 Randomized rate assignments
5. Boundary points in world coordinates

This data file is later interpreted by the TRACS field system.

EXAMPLE OF RESULTING SOFTWARE

In order to incorporate a simple, logical user interface, an object oriented programming language was chosen to build the management system Software for the TRACS management system was developed using Microsoft® Visual Basic® Version 5. The user can navigate through the management system graphical user interface (GUI) using the mouse mentioned earlier as an input device (Jung et al, 1999). Figure 2-3 shows the main portion of the management system GUI. It is organized into several sections· menu, rate legend, experiment size display, border width adjustment and color-coded field layout.

Menu

The menu aids the user in completing design of the experiment (A). The "SKELTON VIEW" selection allows the user to view all possible locations to insert plots. The "RANDOMIZE / UPDATE" selection implements any changes that the user makes to the experiment, re-randomizes the treatment assignments, and maintains the CRD properties of the experiment. The "Real Scale" selection box changes the aspect ratio of the color-coded field layout (I). The "BACK" selection lets the user view other GUI screens including those which allow the user to enter the plot size and number of treatments for the experiment. The "FINISH" selection commands the management system to write the data file for the field system and then end and exit back to the operating system shell.

Rate Legend and Experiment Size Display

Figure 2-3 shows the legend (B) to decode the colors shown in the field layout (I) Each plot in the plot grid (F) is colored to represent the rate assigned to it The purpose of this feature is to allow the user to visualize the randomization

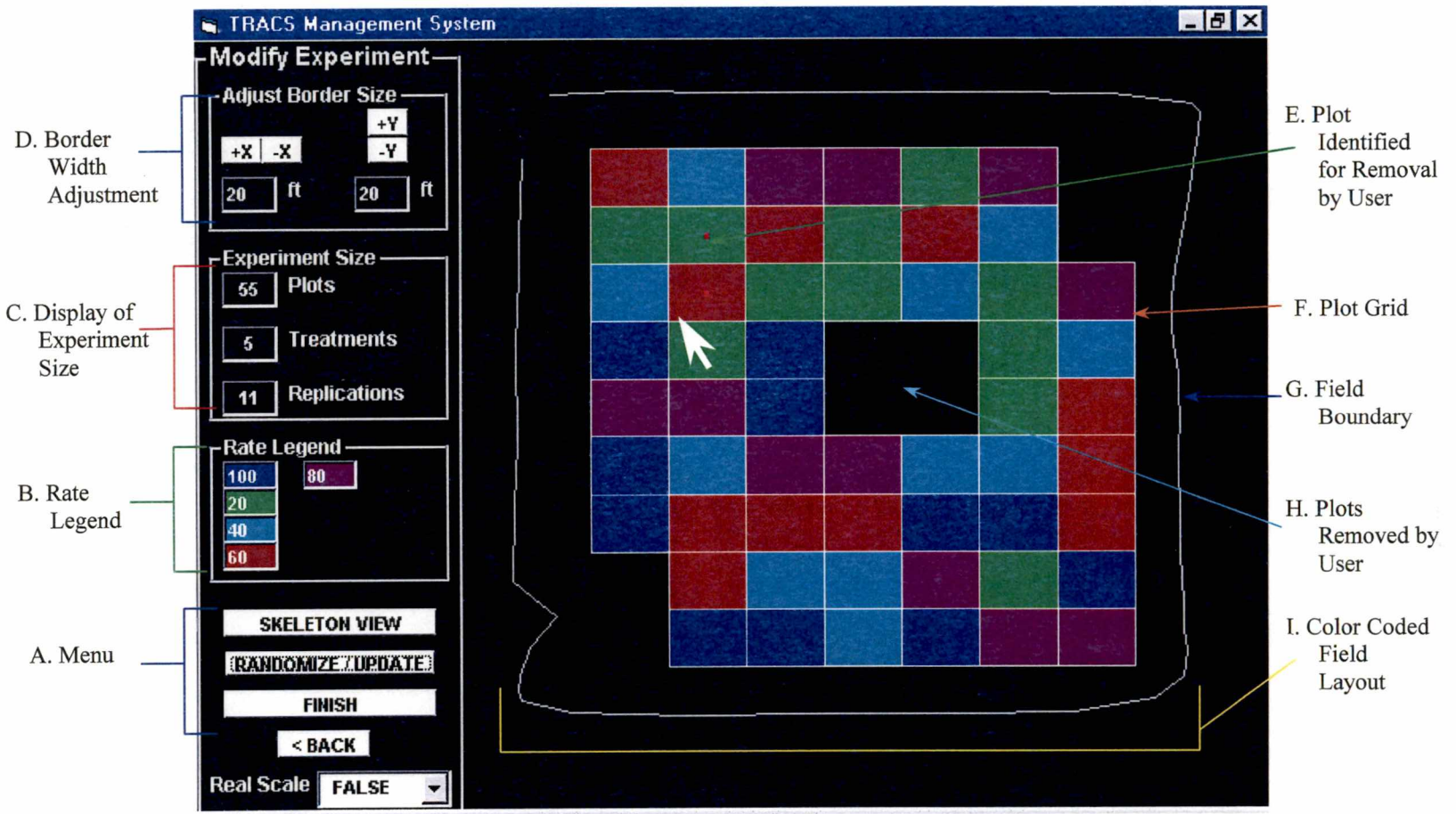


Figure 2-3. Management System GUI

Since the management system insures that the experiment remains a CRD despite user modifications, the management system reports to the user the number of replications applied to the test area (C). It also reminds the user of his previous choice of the number of treatments for the experiment and the total number of plots that can be laid out in the current configuration

Border Width Adjustment and Color-Coded Field Layout

The width of the field border can be adjusted from within the "Adjust Border Size" window (D) This allows the user to add or remove plots around the field boundary to account for applicator turn around.

The system displays the field boundary (G) and plot grid (F) to visually illustrate the experiment and allow easy modifications. The user can manipulate the plot grid, which, is filled with various colors indicating treatments, with the mouse input device. Using the mouse, the user can select plots to remove within the grid (E). After updating the experiment using the "RANDOMIZE / UPDATE" selection, the plots no longer show up in the test area (H).

CHAPTER 3

DESIGN OF A FIELD CONTROL SYSTEM

DESIGN REQUIREMENTS

The second objective of this project was to design a field control system for direct implementation of field plot experiments created with the management system featured in chapter 2. As discussed earlier, this sub-system of TRACS, referred to as the field system, is distinct from the other sub-system (management system) While the management system is a computer-based planning tool used to compose CRD experiments, the field system is a computer system that conducts the actual application of treatments to an array of research plots (test area). The field system provides a method of applying plot treatments according to a desired application map developed with the TRACS management system. This control system serves as an interpreter for the information contained in the desired application map. The field system uses the map as a guide to apply the correct treatments to the correct plots by interpreting the map and making real-time decisions in the form of set-points. The responsibility of the field system is to issue these set-points to some pre-existing variable rate controller (flow controller, planter controller, etc) As mentioned in chapter 1, a set-point is the value that a rate controller tries to achieve by adjusting an actuator device that can throttle the output of some field amendment The intended users of this sub-system are drivers of VRA equipped vehicles. Specific users might include but are not limited to equipment operators for agricultural companies, universities, or other research organizations.

Both hardware and software were developed for the field system Figure 3-1 highlights the design requirements. The hardware consisted of a computer as the communications hub and

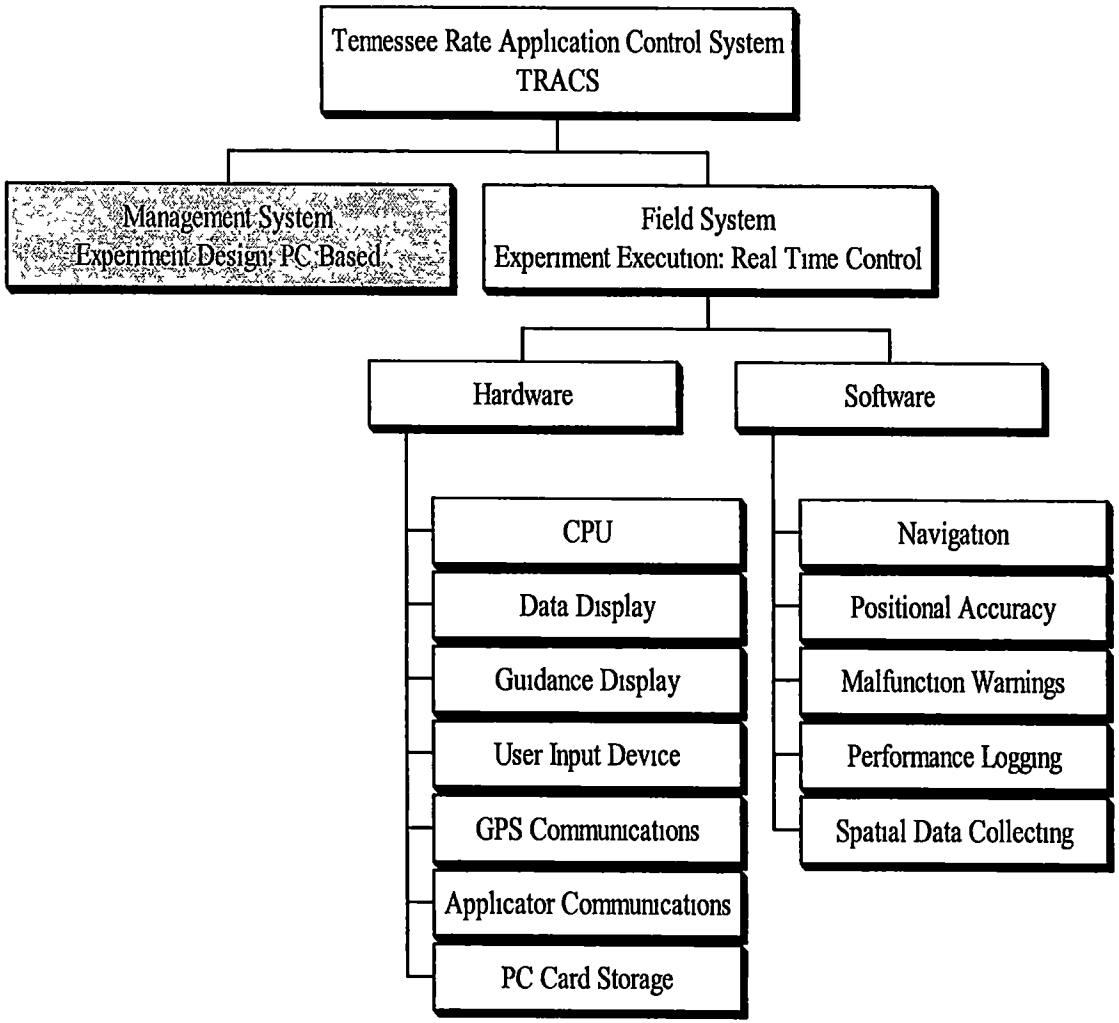


Figure 3-1 TRACS Field System Design Requirements

processing center. A graphical data display was included to prompt the system user for input and assist with navigation A display was developed solely for the purpose of navigation. This device was included to allow the field system operator to easily make steering corrections A simple user input device was designed into the system for operator communications. The system provided a means of communications with a GPS receiver in order to receive spatial data Communications with a variable-rate application controller were made available. Finally, data storage capability via solid-state media was provided.

The field system software design included programming procedures to make the proper calculations for navigation through the test area while maintaining positional accuracy of rate change. Malfunction warnings were built-in to prompt the user if problems detrimental to the integrity of the experiment arise. A routine to log performance data throughout the experiment was employed.

HARDWARE DESIGN AND IMPLEMENTATION

To run the field system software, a field-computer system was developed. Figure 3-2 illustrates the system hardware with its communication interfaces. A single board computer (SBC) was used as the main processing unit (Inside Technology 586 LCD/S) The SBC was equipped with the following

- 133 MHz microprocessor
- 32 MB RAM
- 72 MB EEPROM (M-Systems Disk-On-Chip 2000)
- LCD video controller
- 2 RS-232 serial communication ports
- PC-104 hardware bus
- AT keyboard bus.

Figure 3-3 features the field system components assembled in a rugged aluminum console (control console)

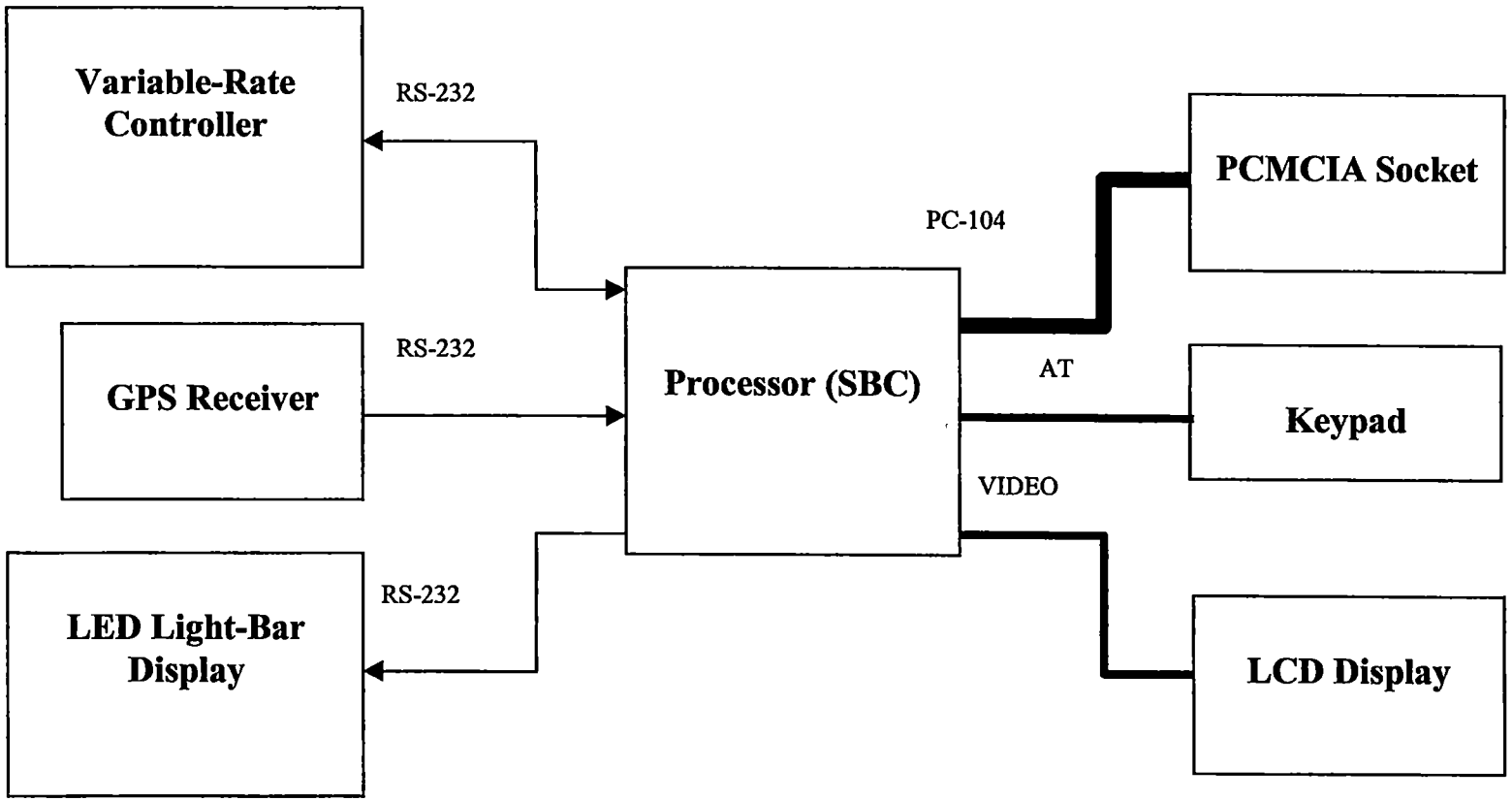


Figure 3-2 TRACS Field System Hardware Schematic

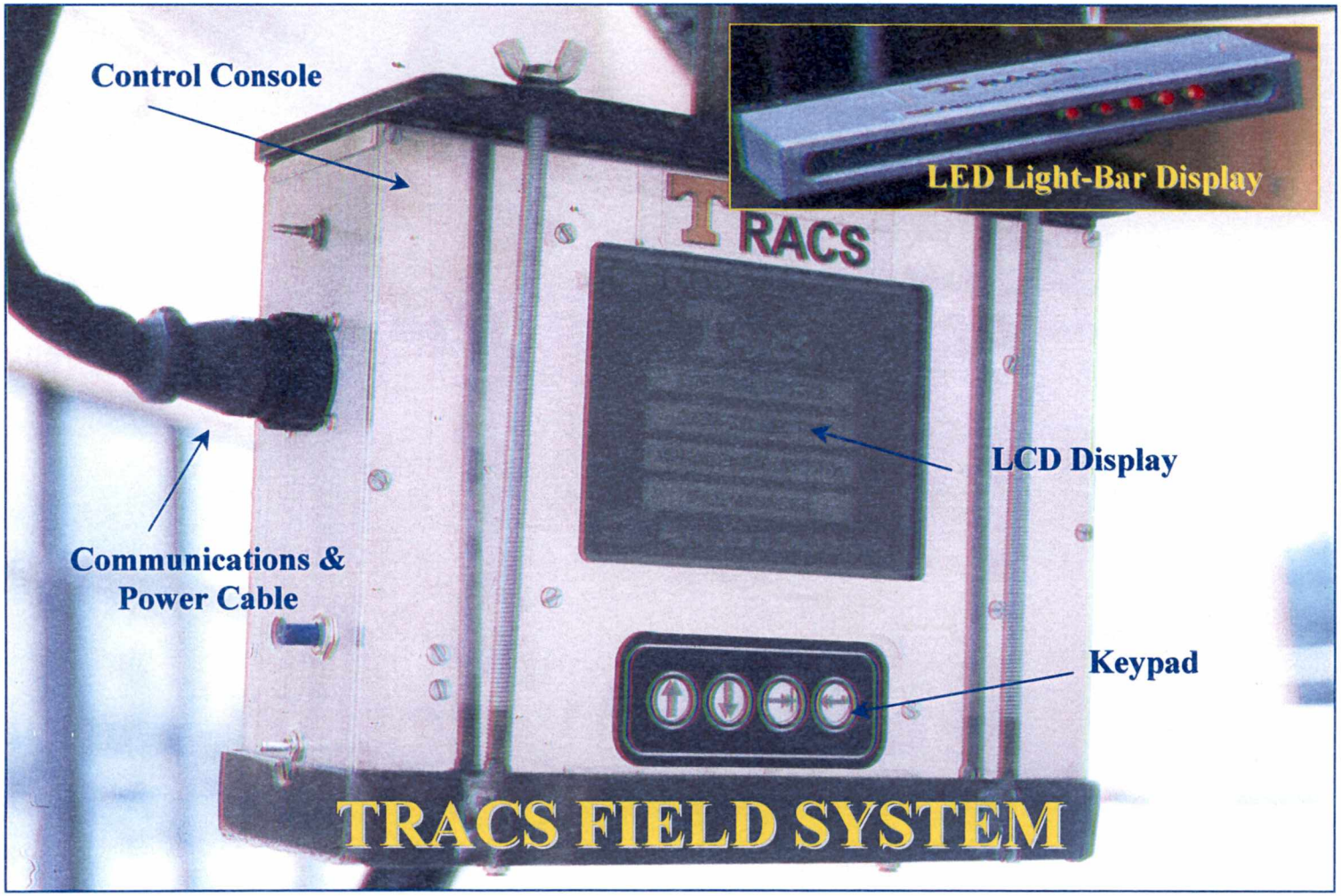


Figure 3-3. TRACS Field System Hardware

PCMCIA Socket

A PCMCIA (Personal Computer Memory Card International Association) card socket (Adtron Devices SDDP Dual Slot) was connected to the SBC via the PC-104 hardware bus PCMCIA cards provided the media of data transfer to and from the TRACS field system. Chosen for their ruggedness, small physical size, and wide range of data storage capabilities, these cards provided a stable alternative to magnetic disks (hard drives, floppy diskettes, etc)

Keypad

User input was provided for with a keypad (C&K Components, Inc) This four key membrane-switch keypad was interfaced with an AT keyboard controller (obtained from an existing personal computer AT keyboard) and connected to the SBC through the AT keyboard buss The keypad produces the following keystrokes· up, down, tab, and enter. The object-oriented field system software was design to respond to these keystrokes. The design insured that the keypad was the only necessary input device needed to operate the field system. Notice the keypad attached to the control console in Figure 3-3

LCD Display

Field layout, navigation information, and user prompts are issued via a data display device. A monochrome, transflective, liquid crystal display (LCD) was used (Sharp LM32K071) The unit was chosen for its readability in sunlight Its transflective properties allowed it to illuminate with either ambient light or a fluorescent backlight source The resolution was 320 x 240 pixels, which is one quarter the resolution of a standard VGA monitors. The LCD was connected to the SBC's video controller by the video bus. Figure 3-3 shows the LCD in an outdoor environment illuminated with ambient light only.

Variable-Rate Controller

The field system can be connected to any dedicated variable-rate controller devices that use an RS-232 serial protocol Bi-directional data is passed between the field system and rate controller. Data sent to the rate controller is in the form of set-points, which were defined in chapter 1 as target rates that the controller tries to achieve The field system is responsible only for issuing these set-points. The means by which the target rate is achieved is solely dependent on the rate controller used. For example, if a liquid fertilizer were being applied, a rate controller, designed specifically to throttle fertilizer flow in a spray boom, would be used If a granular amendment were being applied, TRACS would need to be connected to a rate controller specifically dedicated to adjusting the speed of feeder chains on a spreader truck If variable rate seeding was needed, a rate controller for a seed-hopper would be interfaced with the TRACS field system. Thus, the TRACS field system is not tied to any single type or commercial brand of rate controller As long as the exact serial communications settings and parameters of a particular rate controller are known, TRACS can be used.

GPS Receiver

RS-232 communication was also used to connect a GPS receiver (Trimble Navigation AgGPS 132) to the SBC The GPS receiver provides a constant source of incoming spatial data to the field system. This data allows TRACS to navigate through the test area The particular data string required by TRACS is the GGA string formatted according to NMEA-0183 (National Marine Electronics Association protocol 183) GGA, a standard output string on many commercial GPS receivers, contains time, position, and fix related data. A GGA string is illustrated below:

$GPGGA,123512 00,3556 718180,N,08356.298189,W,2,08,1 2,258 38,M,-32 12,M,7 0,0100*7A

The comma-delimited format allows field system software to separate the string into meaningful

components and extract the following data.

1. Time
2. Latitude
3. Longitude
4. GPS quality indicator
5. Number of satellites in use
6. HDOP (horizontal dilution of precision)

The routine to extract this data is highlighted in the software design section of this chapter. Uses of the GGA data mentioned above is also highlighted in the software section.

LED Light-Bar Display

Successful variable-rate application requires constant decision making on the part of the field system computer Additionally, the operator must be responsible for keeping the application equipment positioned correctly. The field system was designed to make rate change decisions from plot to plot while moving in a straight line down crop rows It is the responsibility of the vehicle driver to keep the application device centered within each row of plots to minimize treatment overlap. However, keeping the vehicle centered is likely to be only one of several activities that require the driver's attention. A large, bright, but non-intrusive display dedicated solely for vehicle alignment was developed The device was modeled after commercial parallel swathing devices commonly referred to as light-bars. Commercial light-bars are sometimes found in large ground-based chemical sprayers as well as in airplane sprayers Typically, light-bars guide vehicle drivers by illuminating a linear array of LEDs (Light Emitting Diodes). LEDs are semiconductor devices that emit colored light when activated Drivers are prompted to adjust vehicle alignment by illumination of certain LEDs The color, number, and/or position of the illuminated LEDs indicate the needed degree of steering correction. These devices are normally placed somewhere in the peripheral vision zone of the driver (Trimble, 1999) The TRACS LED Light-Bar display follows a similar scheme Figure 3-3 contains a photograph of the light-bar illuminated

Figure 3-4 shows the layout of the light bar circuitry. The processing unit is a micro-controller (Parallax, Inc BASIC Stamp I) embedded with a PIC (Programmable Integrated Controller) microprocessor. The micro-controller has 8 digital input / output lines, one of which is programmed to receive RS-232 signals from the SBC Based on the serial command received, the micro-controller addresses one of two EEPROMs (Electronically Erasable Programmable Read Only Memory) The EEPROM in turn addresses the appropriate LEDs from the array.

SOFTWARE DESIGN

The field system software was designed to implement experimental plans set forth by the TRACS management system Implementing these plans successfully requires upholding the integrity of experimental design The field system software was designed with this integrity in mind This TRACS module was intended to act as a transparent process performed between experimental design and experimental data collection. In other words, the VRA methods performed by TRACS should interject minimal experimental error into any given experiment Techniques such as feed-forward control, precision vehicle alignment prompting, malfunction warnings, and performance data logging are incorporated to insure that the field system is not a significant source of experimental error.

The field system software functions are overviewed in Figure 3-5 The first function performed by the software is interpreting the desired application map data file (A) (stored on a PCMCIA card by the management system) This file contains boundary coordinates, row orientation, plot locations and corresponding application set-points Interpreting these parameters initializes field system routines Next, the field boundary, plot grid, and applicator vehicle image are displayed on the LCD To keep the vehicle centered, the LED light-bar provides alignment signals for the driver. These displays (LCD and LED) allow the driver to effectively navigate the

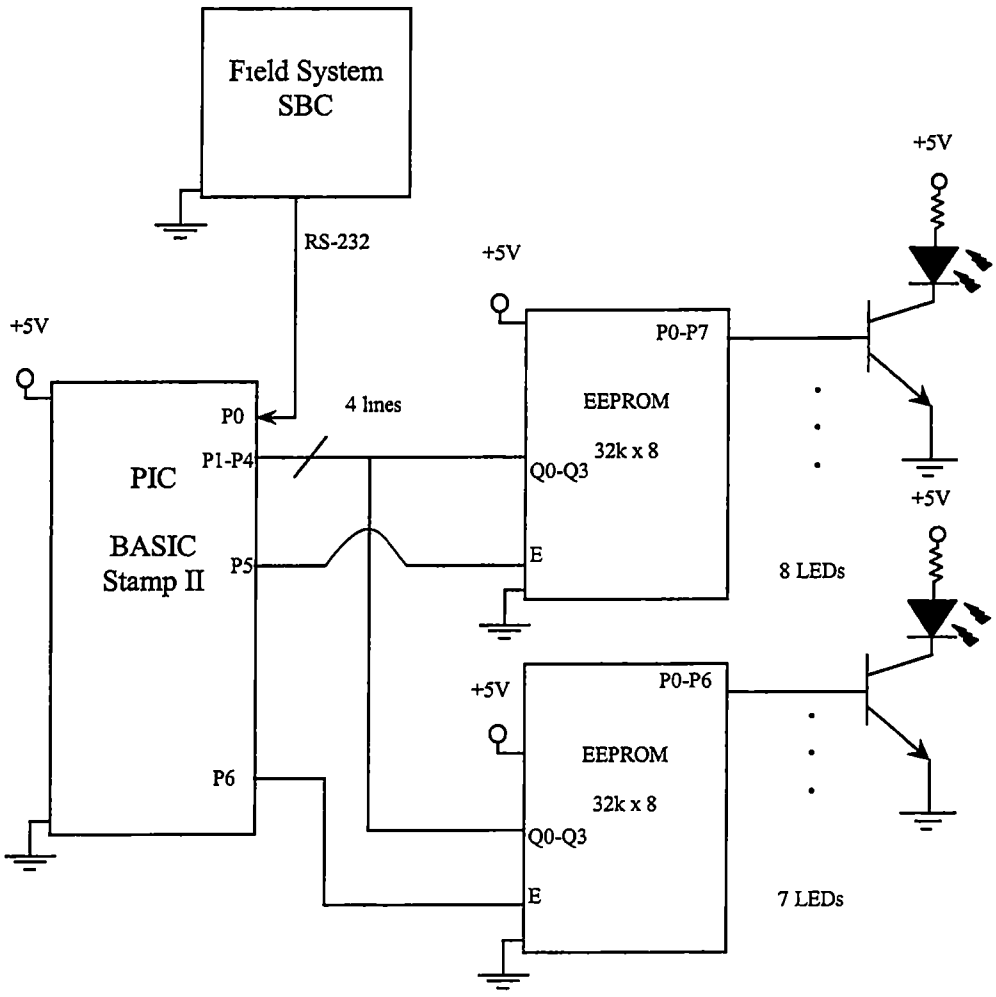


Figure 3-4 LED Light-Bar Circuit Schematic

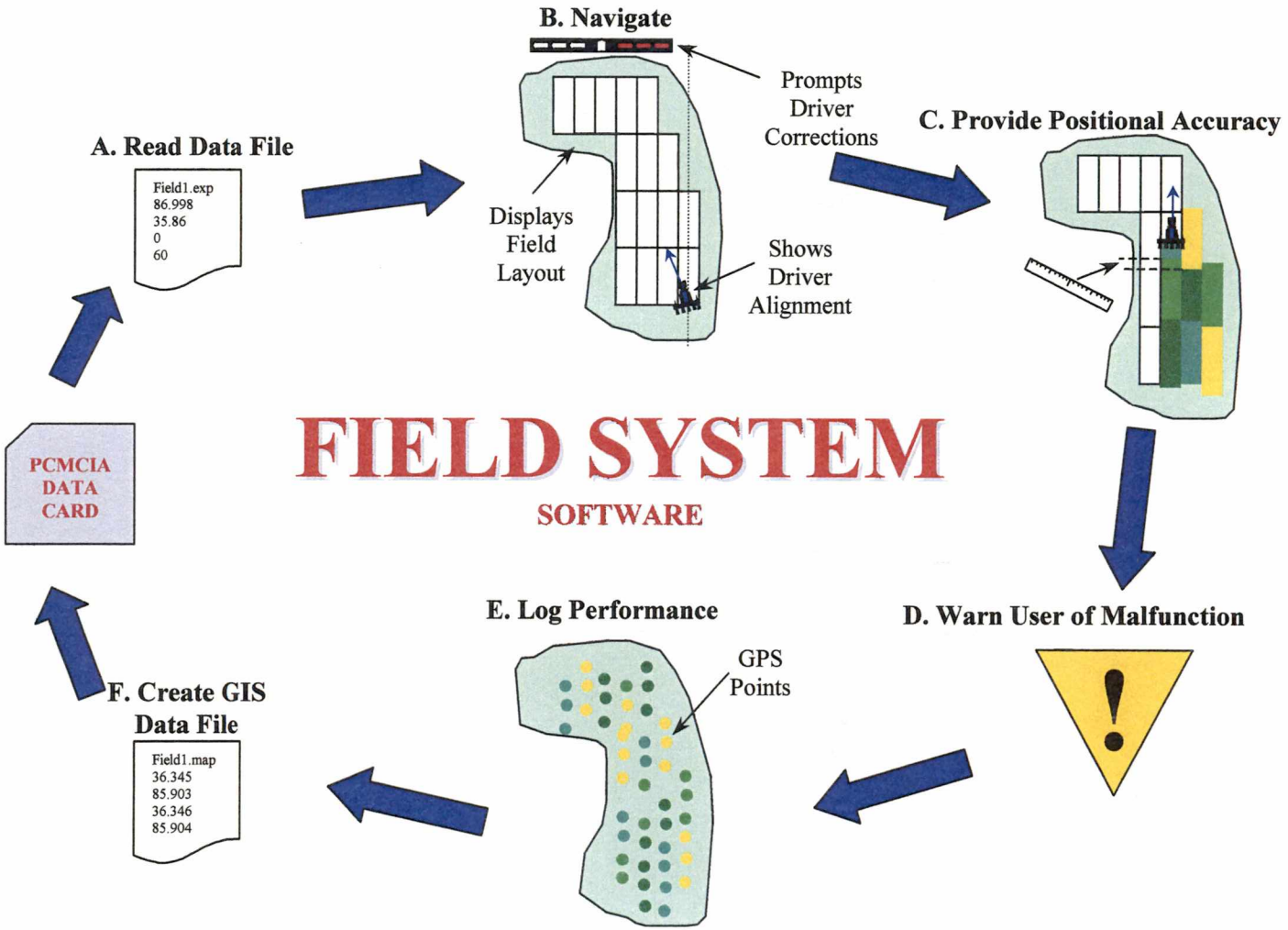


Figure 3-5. Field System Software Functions

applicator through the test area (B). As illustrated in (C), the field system attempts to provide accurate rate changes (positional accuracy) by using feed-forward control techniques. The software promotes experimental integrity by warning the driver of system malfunctions such as poor GPS satellite coverage (D). Performance data consisting of the applicator path coordinates and issued set-points are logged throughout the experiment (E). Finally, The field system stores this data on the same PCMCIA card for possible use in a GIS (geographic information system) database. The following section outlines the software structure in greater detail.

Program Structure

The field system software is structured around one main procedure referred in this thesis as the GPS update loop, which begins by reading in GPS positional data After reading in the data, calculations are performed, control decisions are made, and then the program waits on the next string of GPS data to be sent to the SBC Therefore, the software execution speed is dependent on the GPS update frequency; because of this, the main field system procedure is referred to as the GPS update loop.

Before the GPS update loop can begin, the field system must interpret the application map sent from the management system As discussed in chapter 2, this map is packaged in the form of a tab delimited data file (stored on a PCMCIA card) that includes the following information:

- Rotation angle information ($\sin\theta$, $\cos\theta$)
- Reference point in world coordinates (lon_1, lat_1)
- Plot locations in local coordinates (X,Y)
- Randomized rate assignments
- Boundary points in world coordinates

The field system contains a simple routine to extract this data from its text file format Basically, the routine looks for the data between ASCII tab characters that are used as markers in the file. The field system assigns the data to variables that are stored in the computer memory The use of

these variables is discussed throughout the program structure section.

After interpreting the data from the management system file, the program begins execution of the GPS update loop. The simplified algorithm below overviews this procedure

1. Obtain GPS data
2 Calculate navigation information
3. Make rate control decision and issue set-point
4. Update displays and store data

OBTAINING GPS DATA

The first act performed within the GPS update loop is reading in the GPS data Figure 3-6 shows the basic procedure for reading in GPS data. First, the field system inputs characters from the serial port to which the GPS is attached. It reads one character at a time until it finds the "$" symbol which marks the beginning of a NMEA-0183 output string (Trimble Navigation, 1999). If a "$" is found, the following characters are concatenated together and checked until they form the identifier string for GGA (Grier 1999) The GGA string is the only string used by the field system because it contains the minimum data fields needed for navigation and control including time, world coordinates (latitude and longitude), HDOP (horizontal dilution of precision), number of available GPS satellites, and GPS quality indicator. If GGA is detected, TRACS assembles the aforementioned data fields, character by character using the comma delimiters in the GGA message to indicate ends of data strings After each data string is assembled it is checked for length If any of the strings are too long or too short, all of the data from that GGA string is rejected and TRACS begins sorting the next GGA string in the serial port buffer

Converting Data

After the data strings are extracted from the GGA string, the positional data (world coordinates) is converted directly into spherical Cartesian coordinates. The conversion formulas are shown below in equations 3-1 and 3-2 (Van Sickle, 1996).

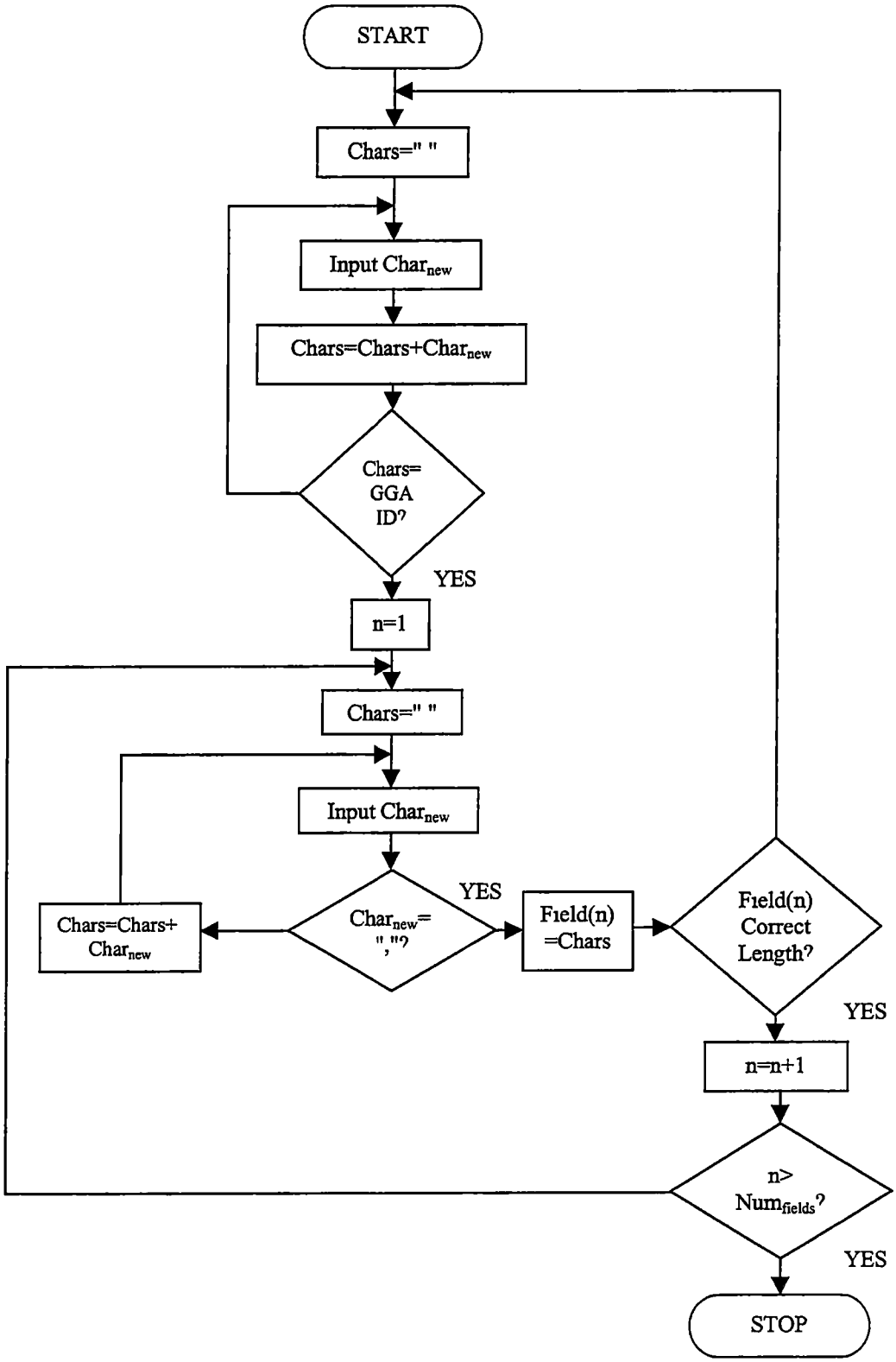


Figure 3-6. Procedure to Read in GPS Data

$$X_u = 365157\,480315 \cos(lat_{ref}) * (lon_{new} - lon_{ref}) \quad \text{(Equation 3 - 1)}$$

$$Y_u = 363188.976378 (lat_{new} - lat_{ref}) \quad \text{(Equation 3 - 2)}$$

where

X_u = spherical Cartesian X coordinate (ft)

Y_u = spherical Cartesian Y coordinate (ft)

lat_{ref} = reference latitude sent from management system (decimal degrees)

lon_{ref} = reference longitude sent from management system (decimal degrees)

lat_{new} = new latitude sent from GPS (decimal degrees)

lon_{new} = new longitude sent from GPS (decimal degrees)

This conversion does not take into account the curvature of the earth's surface but is assumed suitable for localized use over short distances

Rotating Data

After conversion to spherical Cartesian coordinates, the position is rotated to fit the row orientation set up by the management system That is, the y dimension is parallel to the lengthwise direction of the row and the x dimension is perpendicular to the y. Rotating the data is an important step in simplifying all further calculations made by the field system Equations 3-3 and 3-4 are the rotation equations used.

$$X_r = X_u \cos\theta + Y_u \sin\theta \quad \text{(Equation 3 - 3)}$$

$$Y_r = Y_u \cos\theta + X_u \sin\theta \quad \text{(Equation 3 - 4)}$$

where

X_r = rotated X coordinate (ft)

Y_r = rotated Y coordinate (ft)

θ = axis rotation angle (0 - 360°)

Calibrating Data

Next, the positional data is processed with a calibration coefficient and offset as shown in equations 3-5 and 3-6.

$$X_{new} = X_r + V_x m_c + b_c \cos\phi \quad \text{(Equation 3 - 5)}$$

$$Y_{new} = Y_r + V_y m_c + b_c \sin\phi \quad \text{(Equation 3 - 6)}$$

where

X_{new} = calibrated X coordinate (ft)

Y_{new} = calibrated Y coordinate (ft)

ϕ = angle of machine travel (0 - 360°)

V_x = velocity in X direction (ft/s)

V_y = velocity in Y direction (ft/s)

m_c = calibration coefficient (s)

b_c = calibration offset (ft)

The coefficient, m_c, attempts to compensate for the dynamic error associated with the applicator's position This multiplier is designed to take in to account all system delays including computation time and applicator delivery lag time The offset, b_c, compensates for the static component which translates into the distance from the GPS antenna to the spray boom. Figure 3-7 (A) illustrates the TRACS calibration curve Equation 3-7 shows the formula for calculating m_c

$$m_c = \frac{(\bar{e}_c - b_c)}{\bar{s}_c} \quad \text{(Equation 3 - 7)}$$

where

m_c = calibration coefficient (s)

$\bar{e}_c$ = average calibration error (ft)

b_c = calibration offset (distance from GPS antenna to applicator boom (ft))

$\bar{s}_c$ = average speed at time of rate change (ft/s)

To calibrate the field system, it is necessary to perform an off-site accuracy test. Figure 3-7 (B) illustrates the calibration procedure First, the user lays out a straight line between flags 1 and 2 The GPS coordinates of both points are logged by the field system The two points form a theoretical line of rate change Next, the driver navigates the applicator back and forth across the

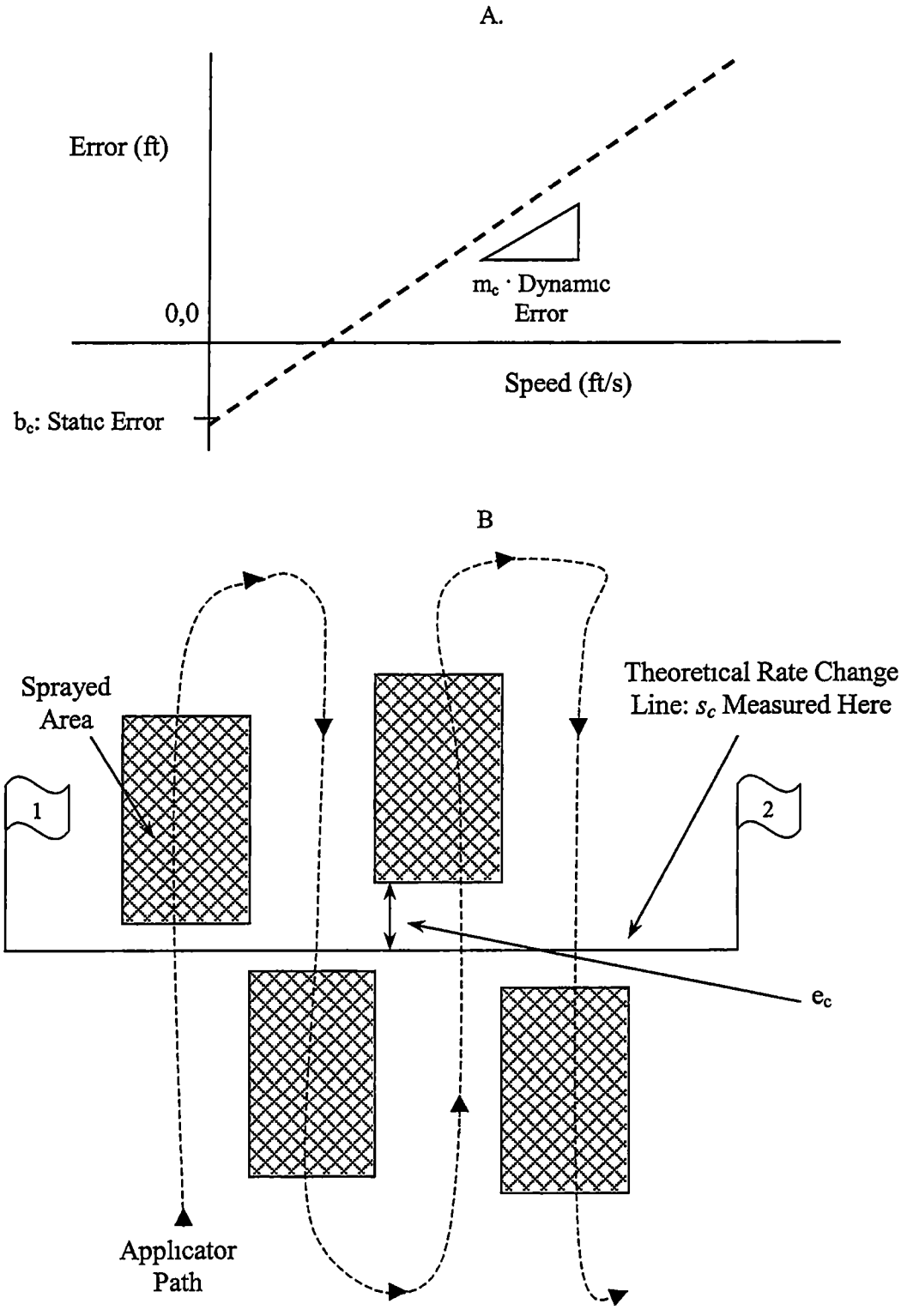


Figure 3-7 TRACS Field System Calibration Curve (A) and Procedure (B)

line several times Each time, the applicator speed (s_c,) and the distance from the line to the actual point that the applicator begins to apply (e_c,) is recorded. The s_c and e_c values are averaged and used in equation 3-7.

Buffering Data

After calibrating the positional data, TRACS stores it along with HDOP in a buffer array. This array serves to keep track of recent data received and processed by TRACS (Figure 3-8 (A)) Its purpose is to provide the field system with enough data to make smooth average vector velocity calculations and establish patterns of GPS signal integrity and reliability The buffer provides a short-term historical performance review to the procedures in the GPS update loop. Figure 3-8 (B) illustrates this procedure The concept of the data buffer array parallels that of a shift register. As it receives new data, it moves all existing data over one place within the buffer. The least recent data is pushed out (erased) from the buffer. The most recent data then fills the first position within the buffer

CALCULATING NAVIGATION INFORMATION

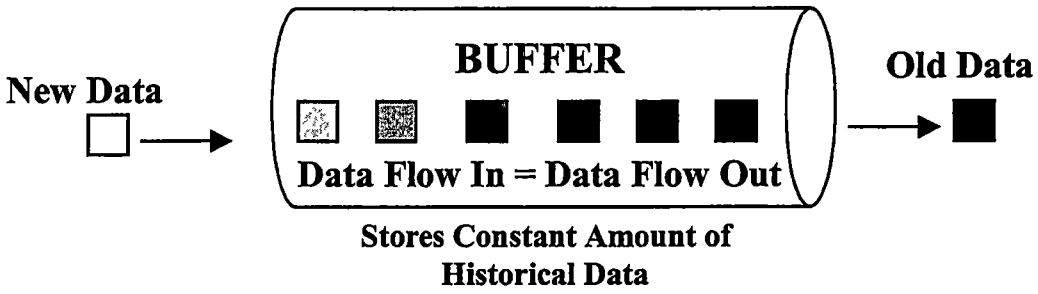
To accurately vary rates in the test area, the field system must know two basic pieces of information: vehicle location with respect to the test area and vehicle velocity In other words, the system needs to know where it is now, what direction it is heading, and how fast it is moving. To determine this information, TRACS executes a series of procedures:

- Calculate vector velocities
- Feed-forward error rejection
- Find closest plot to current location

Calculating Vector Velocities

When the new data is secured in the buffer array, TRACS calculates weighted averages of the vector velocities by equations 3-8 and 3-9.

A.



B.

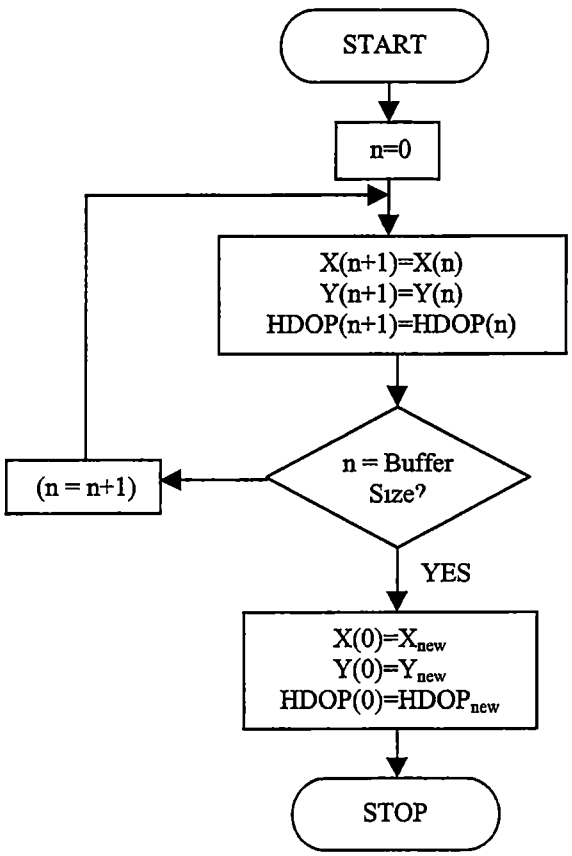


Figure 3-8 Graphical Illustration of Data Buffering (A) and Data Buffering Routine (B)

$$V_x = \left(\sum_{n=0}^{BufferUse} W_n (X_n - X_{n+1}) \right) * f_{GPS} \qquad \text{(Equation 3 - 8)}$$

$$V_y = \left(\sum_{n=0}^{BufferUse} W_n (Y_n - Y_{n+1}) \right) * f_{GPS} \qquad \text{(Equation 3 - 9)}$$

where

W_n = weight of data in buffer position n (%)

X_n = X coordinate in buffer position n (ft)

Y_n = Y coordinate in buffer position n (ft)

f_{GPS} = GPS update rate (Hz)

BufferUse = number of buffer positions used for smoothing

n = buffer position

These formulas use a weighted average procedure to estimate vector velocities with the most recent data is weighted heaviest while the oldest data is weighted the least. Averaging velocities is an important step in reducing the dynamic error associated with GPS By averaging, the effects of slightly erroneous data can be minimized If only instantaneous values are used, an erroneous GPS point might cause drastic spikes in velocity measurements. An unstable source of velocity is detrimental to TRACS feed-forward capabilities. While averaging data reduces velocity spikes, it also dulls the systems ability to detect legitimate velocity changes. To allow TRACS to smoothen spikes, but still detect vehicle acceleration, this weighted averaging technique is used. The weights are determined by the triangular weighting function shown in figure 3-9 (A) The triangle can be enlarged or reduced to take in more or less data, but the sum of the weights must always equal one Equation 3-10 shows the formula for calculating the triangular weights

$$W_n = \frac{(1-n)\left(\left(\sum_{n=0}^{BufferUse} \frac{n}{BufferUse} \right) + 1 \right)}{BufferUse} \qquad \text{(Equation 3 - 10)}$$

This equation is executed upon initialization of the GPS update loop and anytime the

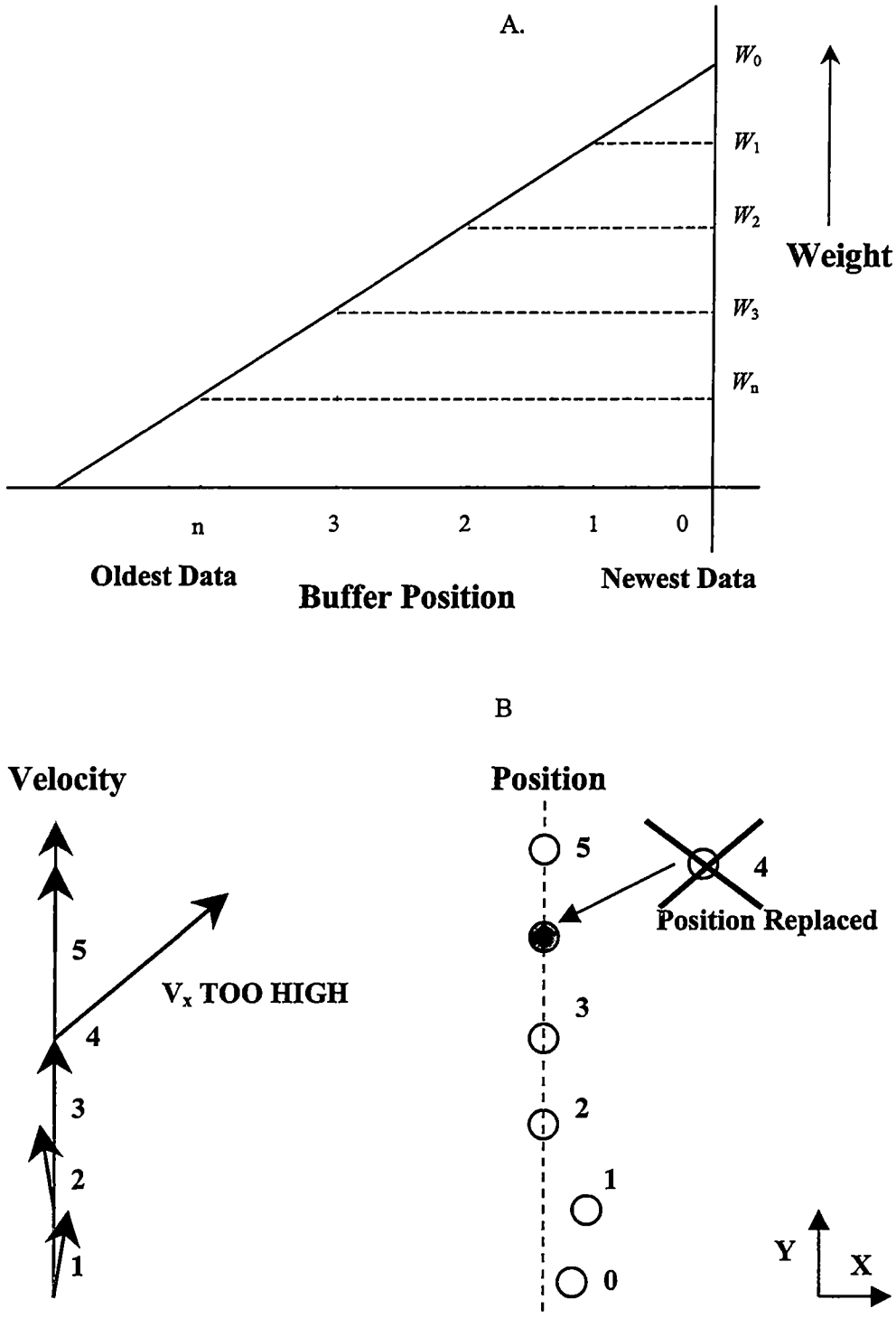


Figure 3-9 Calculation (A) and Use (B) of Vector Velocities for Feed-Forward Control

buffer usage size is changed. The usage size is changeable in the menu system, allowing for adjustable data smoothing while in the middle of an experiment.

Feed-Forward Error Rejection

After the vector velocities have been calculated, TRACS immediately uses them as a source of GPS error rejection referred to in this document as feed-forward error rejection Figure 3-10 shows the flow for this procedure. If the system detects that V_x is unusually high, it will reject the latest GPS position (X_{new}, Y_{new}) and both vector velocities associated with that position, replacing them with projected positions (Figure 3-9 (B)) The x velocity is used as the determining factor because if the applicator is in the test area, then it should be moving fairly straight down one of the rows. Thus, most of the velocity concentrated in the y direction, therefore V_x should be near zero. So, if V_x is above a user defined threshold, the system sets the velocity in the x direction used for further calculations (V_{xused}) to 0. The corresponding y velocity ($V_{yused,}$) is set to the last known good y velocity (V_{ylast}). The x position used for further calculations, (X_{used}) is forced to the last known good x coordinate, (X_{last}) The feed forward step occurs when substituting the y coordinate (Y_{used}) with the projected y coordinate ($Y_{projected}$). As shown in equation 3-11, this value feeds-forward, or extrapolates from Y_{last} by assuming that the current y velocity is similar to V_{ylast}. The desired effect is to project the applicator in a straight line if erroneous GPS data is received. The system also sets a flag each time a bad point is encountered and keeps track of how many consecutive flags are issued If the count is over a certain limit, then a warning is issued to the user If more consecutive flags are issued, then the user is prompted to stop the experiment and the data storage buffers are cleared in attempts to purge the system of the erroneous data .

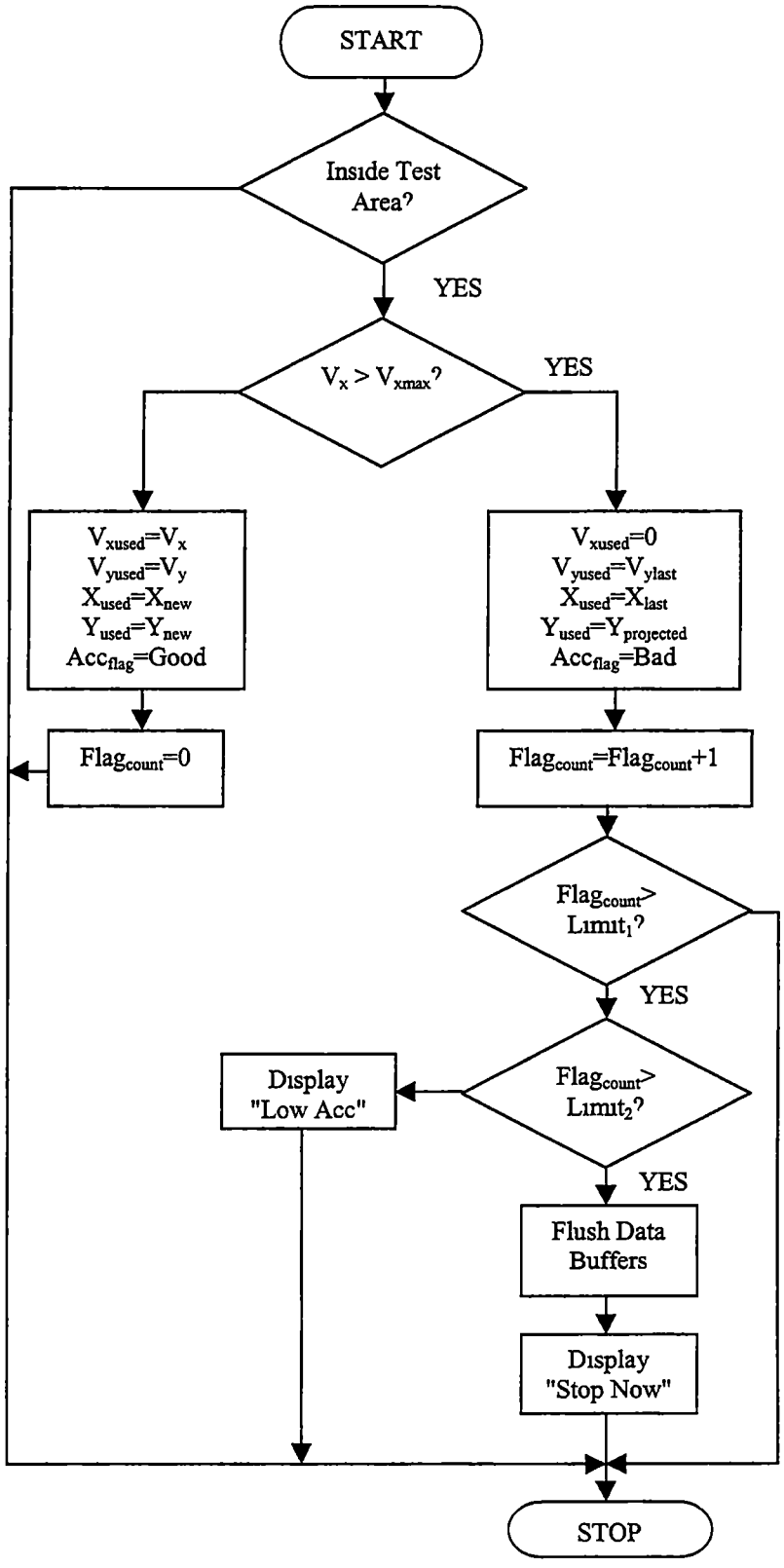


Figure 3-10. Feed-Forward Error Rejection

$$Y_{projected} = Y_{last} + \frac{V_{yused}}{f_{GPS}} \qquad \text{(Equation 3 - 11)}$$

where

$Y_{projected}$ = Y coordinate used if bad GPS data is present (ft)

V_{yused} = velocity in Y direction used for navigation (ft/s)

Y_{last} = last good Y coordinate used (ft)

Finding the Closest Plot

After processing navigational information and rejecting erroneous data, the field system must determine its position in relation to the test area, or in which plot the machine is located Comparing the current position (X_{used}, Y_{used}) with the position of each plot center ($P_x(n)$, $P_y(n)$) determines the closest center point (p_{close},). The closest center point is assumed to be the plot in which the machine is located. Figure 3-11 shows the procedure to determine the closest plot. Basically, the plot center with the shortest distance (P_{dist}) from the current position is picked as p_{close}. Equation 3-12 shows the distance calculation. In Figure 3-11, $P_{distmin}$ is the variable name for the current minimum plot distance and n is the count variable compared with the total number of plots in the experiment (Num_{plots}).

$$P_{dist} = \sqrt{(X_{used} - P_x(n))^2 + (Y_{used} - P_y(n))^2} \qquad \text{(Equation 3 - 12)}$$

where

P_{dist} = distance from current position to plot center (ft)

$P_x(n)$ = X coordinate of n^{th} plot center (ft)

$P_y(n)$ = Y coordinate of n^{th} plot center (ft)

MAKING A RATE CONTROL DECISION AND ISSUING A SET-POINT

Each time the GPS update loop executes, the TRACS field system issues a command to the dedicated rate controller to which it is interfaced This command is a set-point, described

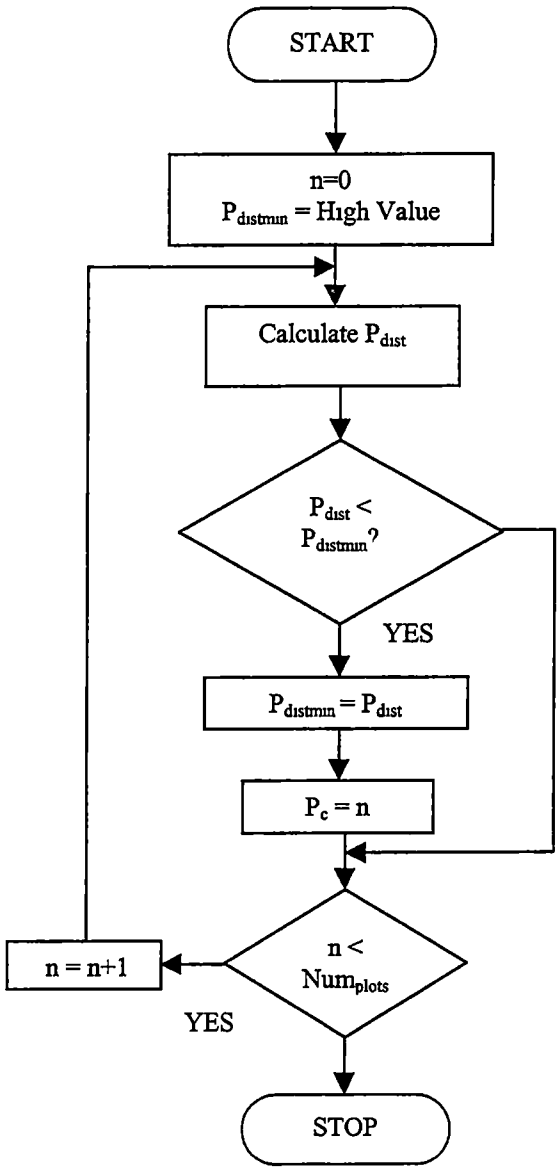


Figure 3-11. Routine for Finding the Closest Plot Center

earlier as the rate that the controller will try apply by whatever means it can Again, it should be emphasized, that the TRACS field system is only responsible for issuing the set-points, the dedicated rate controller (flow controller, seed hopper controller, spreader controller, etc.) must adjust the application vehicle hardware to make the rate change occur. With each GPS update, the field system issues the set-points based on the vehicle location within the test area (as determined by the close-plot finding routine described earlier). However, the system also has the ability to make decisions at a faster rate than the GPS update frequency. This allows TRACS to issue set-points between GPS updates and minimize positional errors associated with low update frequency. This decision making process is accomplished by the following feed-forward procedures

- Calculate ETA
- Run countdown timer in parallel
- Check covered areas and issue set-point

Calculating ETA

Once the closest plot is known, then the time required to reach the next plot (in the vehicle path) can be determined. This time period is referred to in this document as estimated time of arrival (ETA). This interval is calculated each time that a new position is read inform the GPS Equation 3-13 shows this calculation

$$ETA = \frac{\left(P_y(p_{close}) - Y_{used} - D_c * \frac{L_p}{2}\right)}{V_{yused}} \qquad \text{(Equation 3-13)}$$

where

p_{close} = closest plot center to current position

D_c = sign integer indicating direction of V_y (1 or -1)

L_p = plot length (ft)

Running Count-Down Timer in Parallel

A countdown timer that is setup by the GPS update loop executes on 40 ms intervals. Upon execution, the timer checks to see if ETA has occurred. The dotted line in Figure 3-12 denotes that the countdown timer runs in parallel to the GPS loop, thus it can generate an interrupt and perform a task at any point even if the main loop has processor resources occupied. If the timer execution determines that ETA has occurred, TRACS sends the rate (set-point) of the approaching plot to the variable-rate controller. By setting this count-down timer to look for the ETA at small intervals, the effects of a low GPS update rate can be reduced. For example, if the count-down timer were not employed a rate change is only triggered by a GPS update Given perfect GPS data and calibration, if the update rate were 1 Hz (1 update per second) and the sprayer speed was 10 mph, the sprayer misses the rate change line by up to 14 ft. Under the same conditions, but with count-down timer executing at 40 ms intervals, the sprayer hits 0 5 ft of the line

Checking for Previously Covered Areas

If the decision is made by the countdown timer to change rates to that of the approaching plot, the only other stipulation with regards to whether or not to change rates is the plot's status as a covered area If TRACS determines the area has not been covered by the variable-rate applicator, then the plot's set point is issued TRACS determines covered areas by counting the number of GPS updates occurring while the applicator is in the plot. If the number of GPS updates is above an adjustable threshold, then the plot is ruled as covered From that point forward, the covered plot will be issued a rate of zero anytime the applicator passes over it

UPDATING DISPLAYS

As discussed in the hardware design section, the field system provides 2 operator displays (1) an LED light-bar for vehicle alignment and (2) an LCD for field layout and performance information Both displays are updated in the main program procedure (GPS update

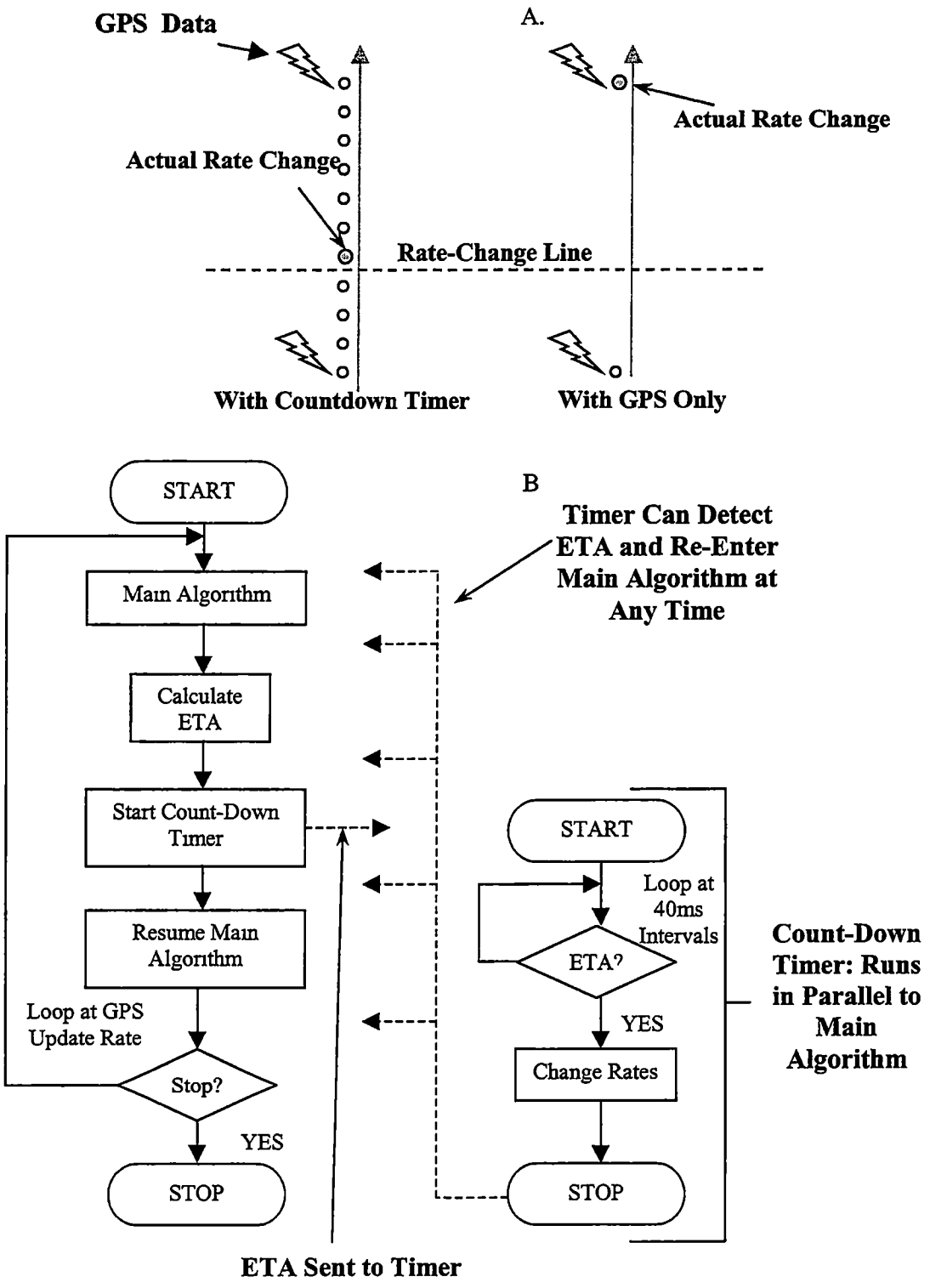


Figure 3-12. Parallel Count-Down Timer

loop)

Updating LED Light-Bar

The projected distance from the row center (X_{drift}) is determined by calculating the x distance from the most recent GPS point to the closest row x coordinate and adding the predicted change due to the x velocity of the applicator. Equation 3-14 makes this calculation possible.

$$X_{drift} = (X_{new} - X_{row}) + V_x * t_{LookAhead} \qquad \text{(Equation 3 - 14)}$$

where

X_{drift} = predicted distance from row center to applicator in X direction (ft)

X_{row} = X coordinate of the nearest row centerline (ft)

$t_{LookAhead}$ = look ahead period for prediction (s)

The look-ahead period can also be adjusted while the system is in use in order to adjust the sensitivity of the light-bar This period is the amount of time that the system extrapolates the historical data in order to estimate X_{drift}. TRACS uses the X_{drift} value to proportion the number of LEDs that are illuminated on the light-bar When the driver steers out of the row center, the light-bar illuminates LEDs on the side of the error For example, if an application vehicle is 4 ft to the left of the row center, 4 LEDs on the left side to the light-bar are activated However, the driver can adjust the sensitivity so that a 4 ft error might result in only 2 LEDs being illuminated.

Updating LCD Screen

The next procedure executed in the main loop is to update the graphic display The moving applicator image on the LCD screen is advanced to represent the movement of the machine. A pointer line represents the angle at which it is pointed Covered plots are crosshatched while all other plots are outlined If the machine turns around, the screen will invert and reposition itself so that the sprayer image appears to be moving toward the top of the screen The row center that the sprayer is closest to is identified, to keep the operator in the proper row.

Three digital gages are also updated First, the rate that TRACS is sending out serially is displayed Next, the machine speed (MPH) is shown. Finally, X_{drift} is printed to the screen Since graphic calculations and execution can be resource intensive, the LCD is updated at a maximum frequency of 1 Hz, no matter what the GPS update rate.

Storing Performance Data

Finally, a storage procedure that writes the performance data to the PCMCIA card is executed Performance data logged on every GPS update includes

1. World coordinates
2. Local coordinates
3. Application rate set-point
4. HDOP
5. Number of available satellites
6 GPS quality indicator
7. TRACS accuracy flag

These values are logged for possible use in a GIS database or to diagnose a potential GPS problem or system accuracy problem in general.

Example of Resulting Software

The software resulting from the field system program structure was designed to have a simple, effective GUI (Wang, 1993). As with the management system, it was written with Microsoft® Visual Basic® Version 5. Figure 3-13 shows the LCD display during the execution of an experiment. An explanation of each image on the screen is given starting with image (A) and working clockwise around the figure. The dotted line shown by (A) represents the center of the plot row in which the vehicle is driving. The curved polygon (B) is the field boundary. The plots within the test area that have already been treated are crosshatched (C) to prompt the driver not to pass through the areas again The applicator is represented by a pointed image (D). The line on the tip of the applicator indicates the direction of travel. The plot grid is shown by (E), notice the

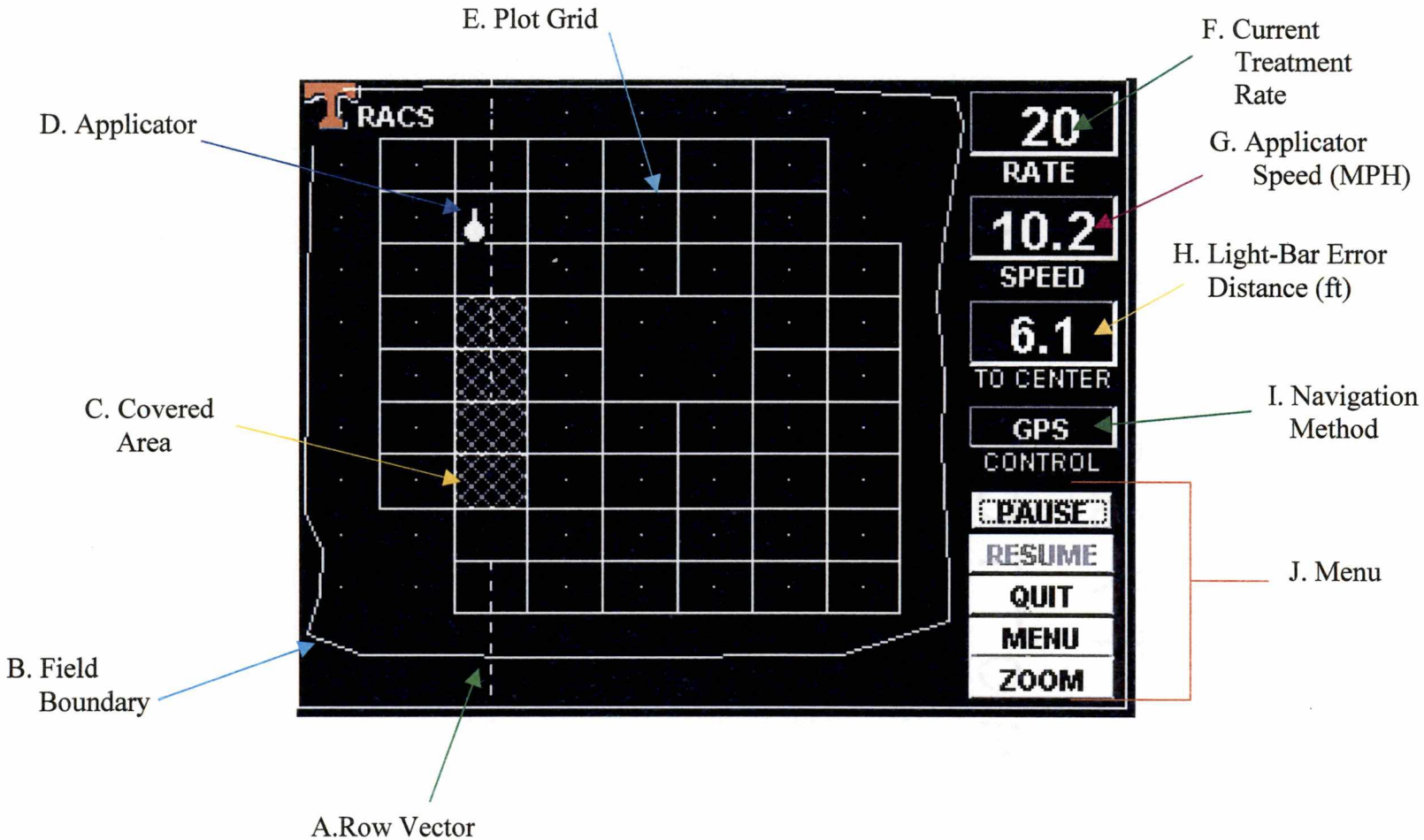


Figure 3-13. Field System GUI (to scale)

area removed near the center of the test area was done during experiment design by the management system (Figure 2-3(H)).

The three digital gages discussed earlier are shown in Figure 3-13 (F), (G), and (H). The current rate being sent to the applicator is listed in (F) The applicator speed, which is the vector sum of V_x and V_y is displayed for the user in (G) X_{drift}, which was defined earlier as the distance from the row center to the applicator, is displayed in (H). The method of navigation is denoted in (I). If the system detects no errors using the feed-forward error rejection method, the display reads "GPS". If, however, an error is detected, then "FF" is displayed to alert the user that feed-forward error rejection is in effect

The menu (J) allows the user to interrupt the GPS update loop to perform alternate operations. The "PAUSE" selection disables the variable rate controller and allows the user to drive around the test area without treating it. The "RESUME" selection toggles this effect Execution of the experiment can be terminated with the "QUIT" selection. The "MENU" option gives the user full access to the field system menu structure which includes screens that allow for changing of settings, thresholds, and sensitivities. Finally, the "ZOOM" feature allows the user to observe smaller areas of the field as well as providing a pan option that allows for the driver to shift the focus of the display

CHAPTER 4

PERFORMANCE EVALUATION

Two types of evaluations were performed on TRACS The first tested the positional accuracy of the system while the other accessed the effectiveness of the software architecture and user interface.

POSITIONAL ACCURACY

Positional accuracy refers to a VRAs ability to change application rates at the desired filed location. While positional accuracy is quantified in length units (ft), it is influenced by several sources of variation including GPS accuracy, system calibration quality, and applicator vehicle acceleration GPS positional accuracy is the first area of concern Typical differential GPS (DGPS) receivers have published static accuracies of about 3 ft RMS This means that if the GPS antenna is not moving, the receiver can predict antenna location within 3 ft of its actual position 50% of the time.

The second variable affecting positional accuracy is the calibration quality of the field control system As discussed in chapter 3, the TRACS field system is calibrated to minimize the effects of computer processing delays, and applicator lag times, and GPS positional delays. Basically, calibration measures the time delays associated with all hardware and software processes involved in a rate change Calibration allows the system to predict both static and dynamic positional error based on the applicator system delays and vehicle velocity, making accurate velocity measurements essential for positional accuracy and further emphasizing the need for minimal vehicle accelerations The third variable (applicator vehicle accelerations)

directly affects the feed forward capability of the field system. For example, if a vehicle speed was 2 mph at one GPS update (1 Hz rate) and between updates the vehicle constantly accelerated to 3 mph, the system assumed that the vehicle was moving at the slower speed throughout the interval between updates This would cause an error of 1 1 ft to be introduced, illustrating the need for constant vehicle velocity while applying treatments to research plots

Equipment

A four-wheeled all-terrain vehicle (ATV) (Kawasaki Mule 2510) was used as the application device for testing (Figure 4-1). The field system control console, LED light-bar, GPS receiver (Trimble AgGPS 132), and a test sprayer were mounted on the vehicle.

Although a constant velocity would have been optimal during testing, the test vehicle had no means to maintain a constant throttle position. It was left to the operator to control velocity with a foot pedal. A hand throttle (common on agricultural tractors) was not available on the vehicle at the time of testing, but other conveniences of the ATV (small size, ease of transport, utility bed, etc.) justified its usage for testing

The test sprayer consisted of a high-pressure carbon dioxide bottle, a two liter plastic bottle filled with dyed water, and a single solenoid valve with an orifice nozzle The field system communicated with this spray controller by the RS-232 protocol discussed earlier. The sprayer was equipped with a PIC microprocessor similar to the one found on the LED light-bar in Figure 3-3. This microprocessor converted the RS-232 ANSI data to a digital (TTL) signal that was connected to a solid state relay, that activated the solenoid Upon activation, the solenoid released a burst of dyed water for 100 ms at a pressure of 25 psi

DPE

TRACS was designed to minimize DPE (dynamic positional error) DPE refers to the distance from a theoretical line to the actual spray-front during execution of an experiment as

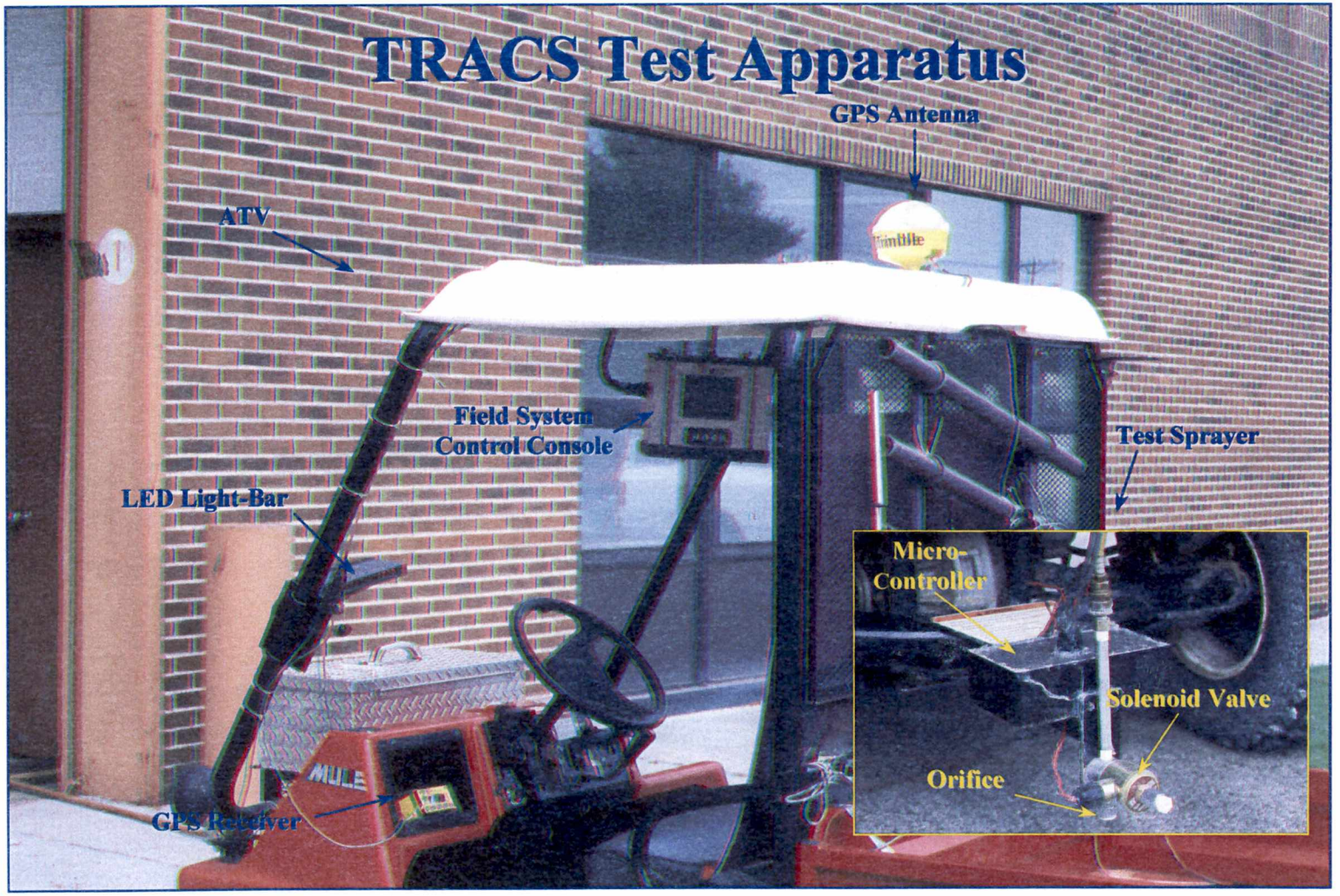


Figure 4-1. TRACS Field Test Apparatus

illustrated in Figure 4-2. The accuracy test attempted to quantify DPE by setting up a theoretical line of rate change similar to the one used in the calibration procedure illustrated in Figure 3-6.

Establishing a Rate-Change Line

Real-time differentially corrected (source satellite subscription) GPS data was collected once a second for 10 minutes on each of the points defining the rate-change line (Figure 4-2). These data points were averaged and used to make the necessary field orientation calculations and entered into the data file for the field system. Before the test, the field system was calibrated by the procedure described in chapter 3.

Testing

After a (theoretical line) was established, the field system control program was started The applicator was then driven across the line 8 times for each of three different speed classes. The low speed class was set at 3 mph, the medium was 5 mph, and the high was 10 mph. Each time the applicator crossed the theoretical line, it would spray a burst of dyed water on the ground This point was marked and the perpendicular distance from it to the line was measured and recorded as DPE. GPS satellite coverage was good (HDOP 1.5 - 3 0) throughout the entire accuracy test.

Results

Figure 4-3 illustrates the results of this test The extension bars represent the standard deviation in each speed class The average absolute DPE in the low speed class was 2.4 ft. The DPE ranged from 1 ft to 4 4 ft in this class with a standard deviation of 1.31 ft The medium speed test showed similar results averaging a 2.7-ft absolute DPE and ranging from 0 2 ft to 4.3 ft with a standard deviation of 1 31 ft A mean separation proved that no significant difference existed between the low and medium speed classes The average absolute DPE in the 10 mph class (4 5 ft) did prove to be significantly higher than the absolute DPE in the other two classes

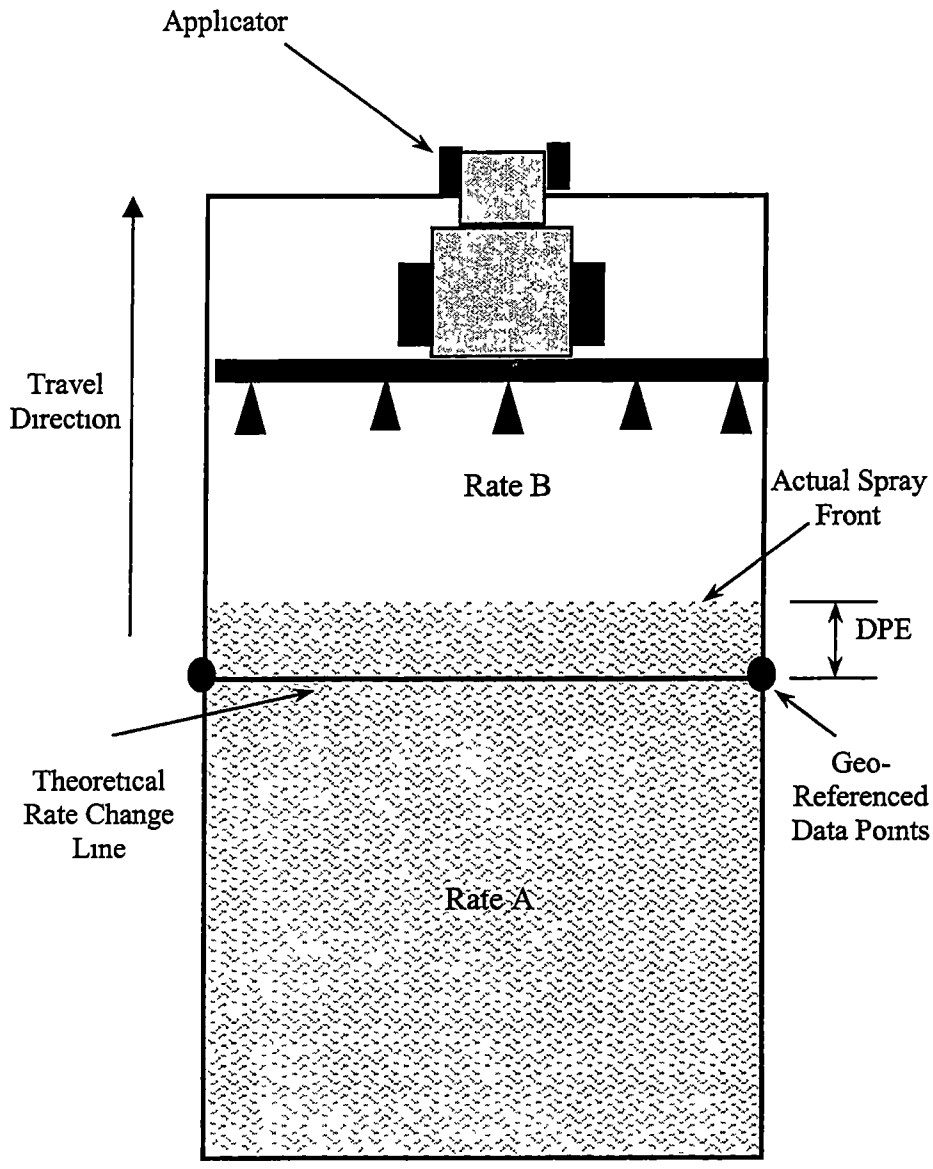


Figure 4-2 Illustration of Dynamic Positional Error (DPE)

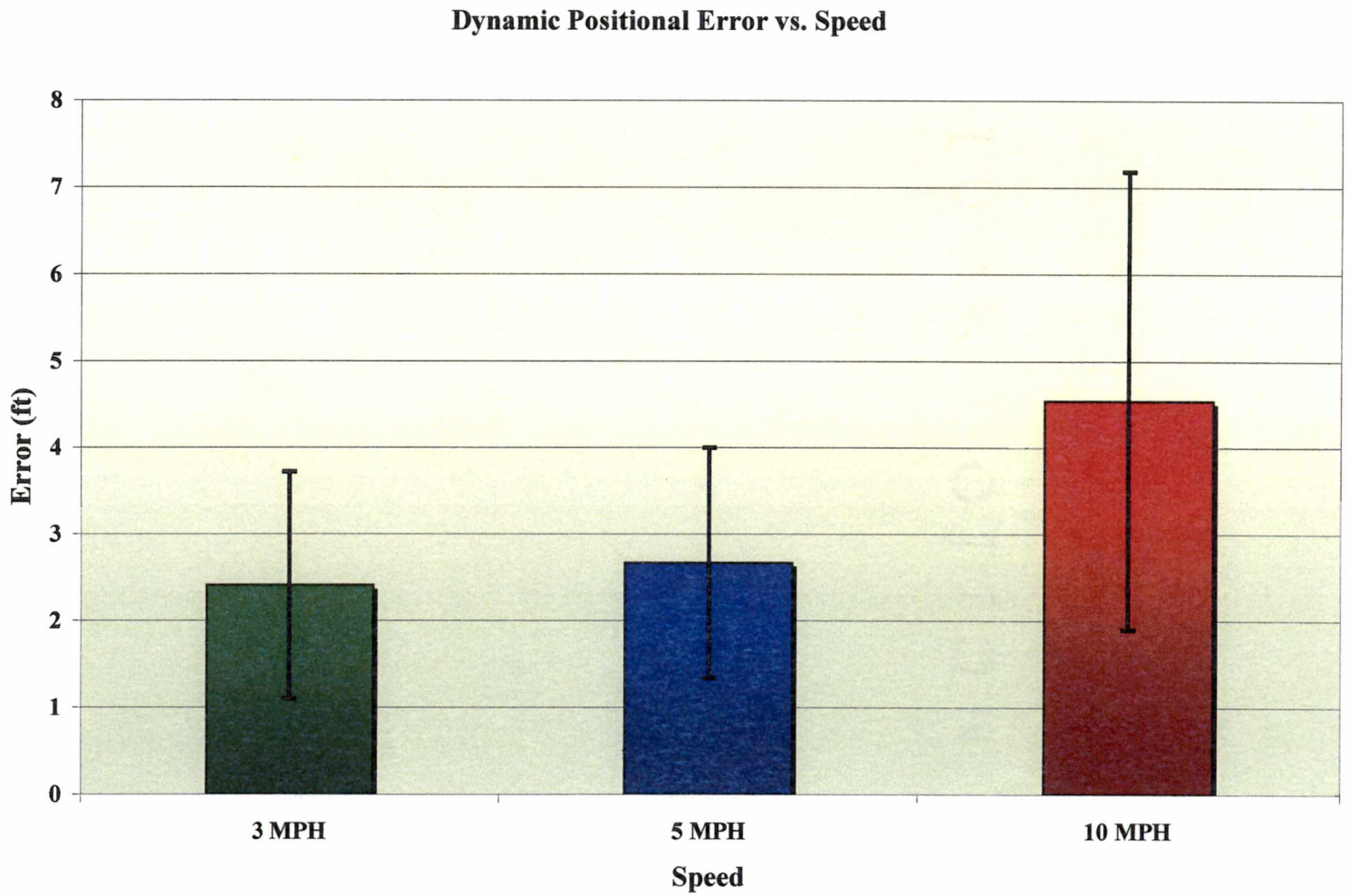


Figure 4-3. TRACS Absolute Dynamic Positional Error vs. Speed

Table 4-1. Positional Error Test Results for TRACS

Speed (MPH)	Mean Absolute DPE (ft)	Standard Deviation (ft)	Significant Differences
3	2.4	1.3	a
5	2.7	1.3	a
10	4.5	2 0	b

It ranged from 1.1 ft to 7 8 ft with a standard deviation of 2.0 ft. Table 4-1 summarizes these results.

The system performed as expected in the low and medium speed classes but erred more in the high speed class. This problem was likely due to a poor calibration. Further examination of the data revealed that most of the errors were leading errors, that is, the system was changing rates too early. This points to a calibration coefficient, m_c that was too high Refinement of m_c would have alleviated this problem Figure 4-4 illustrated the average DPEs (round markers) Notice that all three mean errors are negative, indicating early rate changes. If m_c, was refined from 0.80 s (determined during the calibration routine) to 0.50 s, the mean DPE would have become much closer to zero in all speed classed. Figure 4-4 shows the corrected DPEs (diamond markers) due to the employment of the new m_c

Despite a slightly higher than expected error in the 10 mph tests, the standard deviations in all speed classes fell well within the published static accuracy of the GPS receiver used in testing (3.2 ft). Further suggesting that the early errors were due to an improper calibration coefficient.

Even without the corrected calibration, the results indicate that the feed forward algorithms improved accuracy beyond that of systems not employing feed-forward control For example, consider a VRA using a 1Hz GPS and traveling at 5 mph. If feed forward was not used on a 1 Hz GPS, DPEs ranging up to 8 ft would be expected due to this source of variation alone. However with the TRACS feed-forward methods, absolute DPE was held to an average of 2.7 ft

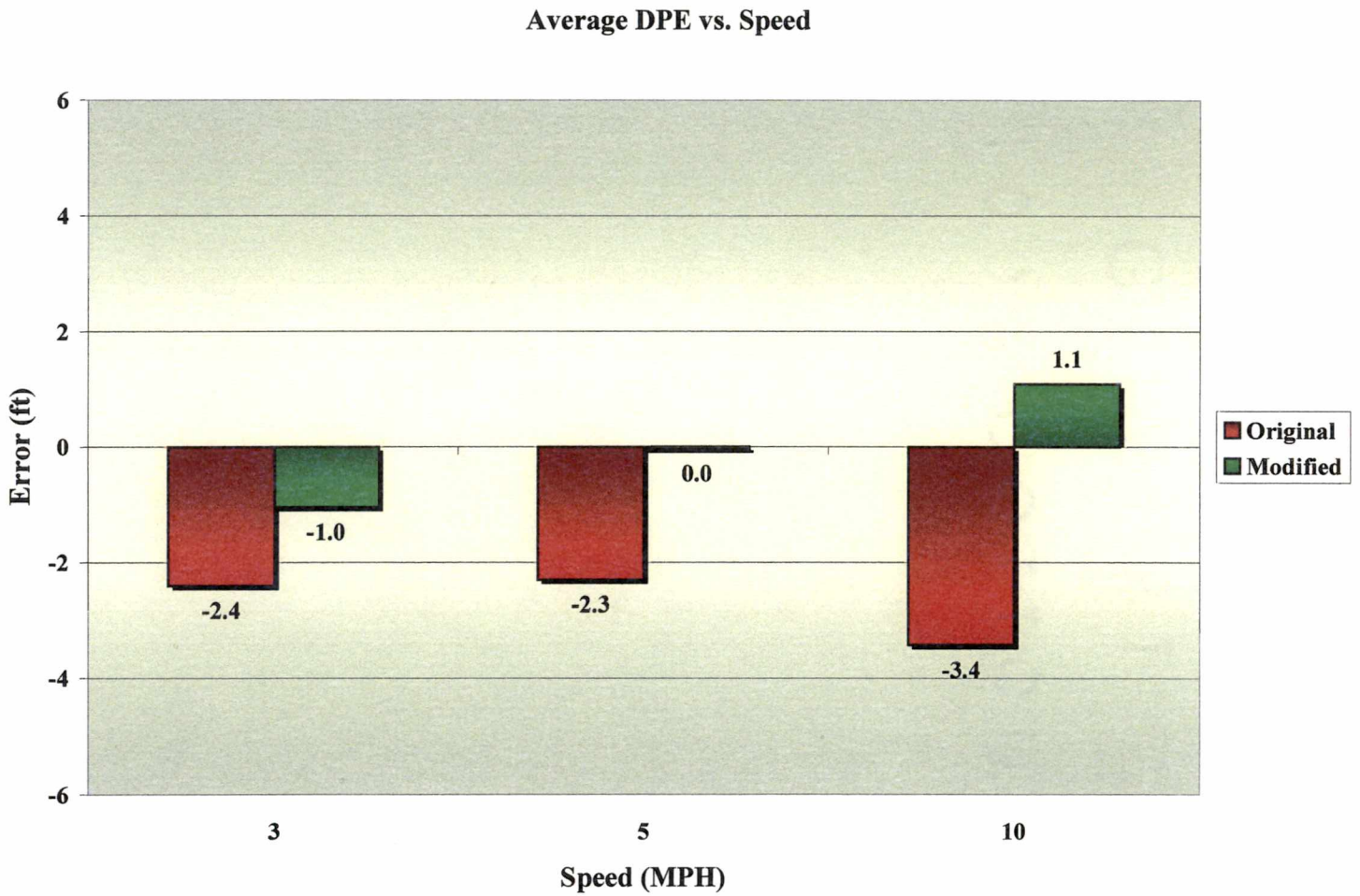


Figure 4-4. TRACS Average Dynamic Positional Error vs. Speed

in the 5 mph class. As noted earlier, refinement of m_c could have driven the error near zero (Figure 4-4) Also, note the trend in the error term was dependent on vehicle velocity, suggesting that further refinements in the coefficient could be obtained

USER SURVEY

Overview

A user survey was performed on a small group of faculty and staff All the subjects were researchers from The University of Tennessee Two professors and four graduate students from The College of Agricultural Sciences and Natural Resources were polled. A test course was setup using survey flags to mark field boundaries in an open field at the Knoxville Experiment Station.

Participants were each given the same set of written instructions prompting them through the entire experiment design and execution process The activity was broken into four phases:

- Field boundary and row orientation coordinate collection (management system)
- Experiment design (management system)
- Treatment application (field system)
- Overall system use.

Two categories for each phase were provided for ranking (1) direction clarity / ease-of-use and (2) user interface Category 1 was intended to assess the prompts and procedures leading the user through the experiment design and execution process Category 2 was used to determine the effectiveness of navigation displays and plot layout graphics

Results

Table 4-2 shows the average rankings on a scale of 0 to 5. The spatial data collection user interface received the highest ranking, averaging a 4.6 out of 5.0. While the treatment application received the lowest average ranking (4.0). Reasons given for the low ranking included problems with glare on the LCD and difficulty following the light-bar Participants also noted problems with direction clarity including the absence of prompts telling the user to "wait" during periods of computer processing.

Table 4-2 TRACS User Survey Results (Ranking. 0-5, 5 Maximum)

Survey Category	Data Collector	Management System	Field System	Overall System
Direction Clarity/Ease of Use	4.1	4 1	4.5	4.2
User Interface	4.6	4 4	4 0	4.3

CHAPTER 5

SUMMARY

SUMMARY

The need for rapid implementation of large, multi-variable field experiments hindered by limited application-map design software, and inapplicable commercial variable-rate control systems led to the development of the Tennessee Rate Application Control System (TRACS), a variable-rate application (VRA) system devoted specifically for agricultural research TRACS was developed through the design and integration of two separate modules (sub-systems). One module, known as the management system was created to aid in the design of plot-scale experiments. Spatial field data, including boundary coordinates and crop row orientation, was required by this off-site computer based sub-system The other TRACS module, the field system, was organized for direct implementation of plot-scale experiments created with the management system. This rugged field computer system was built to communicate in real-time to a variable-rate controller The field system was programmed to conduct the VRA treatment plans composed by researchers using the TRACS management system. An overall performance evaluation suggested that the TRACS could be used for rapid, labor-reduced design, implementation, of field research experiments

CONCLUSIONS

The field system proved to have high positional accuracy at normal research applicator operating speeds, erring by an average of 2.5 ft Precise calibration of the system was found to be critical in minimizing positional error. A small-scale user survey revealed that the system was

easy to use and had an appropriate user interface scoring 85% of the possible points in both categories.

RECOMMENDATIONS FOR FUTURE DEVELOPMENT AND TESTING

Field System

- Improve LCD visibility by using anti-glare covering
- Provide "wait" messages during processing
- Place the LED light-bar with the control console for best guidance combination
- Completely weather-proof control console
- Optimize program for use on palm-top computer
- Use RTK (real time kinematic) GPS
- Test at higher GPS update rates
- Test with constant speed vehicle (tractor)
- Develop a more reliable calibration routine (use RTK)
- Develop Interfaces for various commercial applicators
- Optimize close-plot finding routine
- Conduct evaluations using experiment station staff
- Explore additional zoom/pan viewing methods

Management System

- Refine plot layout technique to work with complex field shapes
- Develop module for direct interface with a popular GIS system
- Allow for block designs (RBD, Latin Squares, Incomplete Block, etc.)
- Develop series of software tools for navigation and data collection
- Use polygon for field boundary to cut/manage plot units

BIBLIOGRAPHY

BIBLIOGRAPHY

AEA. Ag Electronics Association 1998. Transport support layer specification. Version 1.0, Draft 19980713. http://agelectronicsassn org July 10, 2000.

Ag-chem Equipment Company, Inc 1998. Parts and Accessories Catalog Ag-Chem Equipment. pp. 185-199.

Bielke, B., W.W. Rudolph, and C. Brinkley 1997 SECTOR. A fully integrated, site-specific application system for sprayers ASAE Paper No. 973057. Annual International Meeting July 30-August 2 Minneapolis, Minnesota

Clark, R.L. and R L McGuckin. 1996. Variable rate application equipment for precision farming. 1996 Beltwide Cotton Conferences 1: 186-197.

Dyke, G.V. 1988. Comparative Experiments with Field Crops. Charles Griffin and Company Ltd., New York. pp 4-50

Fulton, J.P., S A. Shearer, G Chabra and S.G. Higgins. 1999. Field evaluation of a spinner disc variable-rate fertilizer applicator. ASAE Paper No 991101. Annual International Meeting, Sheraton Centre Toronto, Canada July 18-21

Grier, R 1999. Visual Basic Programmer's Guide to Serial Communications. Second edition Mabry Publishing, Stanwood pp 102-183

Holmberg, M. 1998. Consistent results with variable rates Successful farming online http.//www.agriculture.com/sfonline/sf/1998/may/results/index.html. July 11, 2000.

Jung, D , P Boutquin, J.D. Conley III, L. Eidahl, L Mauer, and J. Purdum. 1999 Visual Basic 6 Superbible Sams Publishing, Indianapolis pp. 347-663

Pearce, S.C 1983. The Agricutural Field Experiment John Wiley and Sons, Chichester. pp. 28-57

Petersen, R G 1994 Agricultural Field Experiments Design and Analysis Marcel Dekker, New York. pp. 3-64

Reichenberger, L. 1997a Liming on the level Successful farming online http //www agriculture.com/sfonline/sf/1997/january/precisn/1 html. July 11, 2000.

Reichenberger, L 1997b Sizing-up variable seeding, Successful farming online. http //www agriculture.com/sfonline/sf/1997/january/precisn/2 html. July 11, 2000.

Rockwell, A.D and P D Ayers 1994 Variable rate sprayer development and evaluation Applied engineering in agriculture 10 (3). 327-333.

Schueler, J.K. and M. Wang. 1994 Spatially-variable fertilizer and pesticide application with GPS and DGPS. Computers and electronics in agriculture. 11 (1) 69-83

Searcy, S.W. 1995. Engineering systems for site specific management. Opportunities and limitations. *In.* Site-specific Management for Agricultural Systems. Proc. 2nd Intl. Conf., ASA, CSSA, SSSA, Madison, WI. pp. 603-612.

Stafford, J.V. and B. Ambler 1994. In-field location using GPS for spatially variable field operations Computers and electronics in agriculture 11 (1): 23-36.

Sudduth, K. A. 1999. Engineering technologies for precision farming. Missouri precesion agriculture center website http'// www.fse.missouri edu/mpac/pubs/vrnemath.htm July 13, 2000.

Trimble Navigation. 1999. AgGPS 124/132 Operation Manual. Trimble Navigation Limited. Appendix D. pp D7-D28.

Van Sickle, J. 1996. GPS for Surveyors Ann Arbor, Chelsea.

Wang, W. 1993. Visual Basic 3 for Dummies. IDG, New York.

Wrather, A., G. Stevens, and T Kirkpatrick 1998. Variable rate nematicide application using global positioning systems for cotton root knot nematode control. Missouri Precision Agriculture Project Report http.//www.fse.missouri.edu/mpac/projects/vrnemath html. July 11, 2000.

Yang, C., G.L. Anderson, J.H Everitt, and J M. Bradford. 1998. Nitrogen and phosphorus management using a variable rate liquid fertilizer applicator. ASAE Paper No 981050. Annual International Meeting, July 17-20. Disney Coronado Springs Convention Center, Orlando, Florida.

APPENDICES

APPENDIX A

TRACS FILE STRUCTURES

Filename: *.PER (*Experiment name, consistent with name picked in data collection routine, ex: "NITROG1.PER")
Generated by Data collection routine
Generated for: Management system and field system
Storage Location. "D.*\" (substitute PCMCIA drive letter for "D" if different)
Purpose: Contains field boundary coordinates
Format· TAB (ASCII character 9) delimited, no header
Example:

Longitude° [TAB]	**Latitude°** [TAB]
83 9381899	*35 9455051*
83 9383096833333	*35 9456499*
83.9384186	*35.9457822*

Filename: *.PTS (*Experiment name, consistent with name picked in data collection routine, ex: "NITROG1 PTS")
Generated by. Data collection routine
Generated for· Management system
Storage Location· "D:*\" (substitute PCMCIA drive letter for "D" if different)
Purpose: Row orientation coordinates
Format· TAB (ASCII character 9) delimited, no header, label on each line
Example:

Label [TAB]	**Longitude°** [TAB]	**Latitude°** [TAB]
Point1	*83.938303076*	*35.94530139*
Point2	*83.939146103*	*35.94629988*

Filename: *.EXP (*Experiment name, consistent with name picked in data collection routine, ex: "NITROG1.EXP")
Generated by. Management system
Generated for Field system
Storage Location: "D:*\" (substitute PCMCIA drive letter for "D" if different)
Purpose· Contains application map information
Format: TAB (ASCII character 9) delimited, header (one line)
Example
Header: **Parent File [TAB] Pt1 Lat° [TAB] Pt1 Long° [TAB] Plot Length [TAB] Plot Width [TAB] Num of Rates [TAB] Cosθ [TAB] Sinθ [TAB] Num of Rows [TAB]**
Rates: **Rate # [TAB] Rate [TAB]** (Rate # 0 is the Border Rate)
Assignments: **Plot # [TAB] X [TAB] Y [TAB] Rate # [TAB]**

NITROG1 pts	*35 945*	*83.938*	*90*	*30*	*2*	*80*	*82*	*- 56*	*8*	
	0	*0*								
	1	*100*								
	2	*20*								
	1	*-58*	*792*	*0*						
	2	*-58*	*702*	*2*						
	3	*-58*	*612*	*1*						

Filename: *.REC (*Experiment name, consistent with name picked in data collection routine, ex. "NITROG1.REC")
Generated by: Field system
Generated for. Field system
Storage Location: "D:*\" on field system computer (substitute PCMCIA drive letter for "D" if different)
Purpose: Indicates which plots have been covered, used for recovery (in case system crashes)
Format: No header
Example.

Plot #
8
9
15

Filename: *.MAP (*Experiment name, consistent with name picked in data collection routine, ex: "NITROG1.MAP")
Generated by: Field system
Generated for. GIS
Storage Location. "D.*\" on field system computer (substitute PCMCIA drive letter for "D" if different)
Purpose: Logs system performance and path Records on each GPS update
Format: TAB (ASCII character 9) delimited, No header
Example:

UTC [TAB] Longitude (DDMM.M) [TAB] Latitude (DDMM.M) [TAB] X [TAB] Y [TAB] Actual Rate [TAB] Set-Point [TAB] Num of Satellites [TAB] HDOP [TAB] GPS Quality [TAB] TRACS Accuracy Flag [TAB]

44373 08356 38 3556 815 2 898 714.92 60 60 7 1 2 OK
44374 08356.37 3556 833 2.724 702 59 60 60 7 1 2 FF

Filename: *.CAL (*Name chosen for particular calibration, ex. "HIGHBOY CAL")
Generated by: Field system
Generated for. Field system
Storage Location: "D:\CALIBR\" on field system computer (substitute PCMCIA drive letter for "D" if different)
Purpose: Records calibration coefficient (m_c) and offset (b_c)
Format: TAB (ASCII character 9) delimited, 1 line
Example:

m_c [TAB] b_c [TAB]
.4384796 -4

Filename: CALIBR.EXP
Generated by: User
Generated for: Field system
Storage Location: "D.\CALIBR\" on field system computer (substitute PCMCIA drive letter for "D" if different)
Purpose Application map file for the field system calibration routine
Format: TAB (ASCII character 9) delimited, same format as all "EXP" files (pt1 lat, lon, and θ info is replaced during calibration routine)
Actual File :

Calibr	*35*	*83*	*100*	*25*	*2*	*8*	*1*	*0*	*4*
	0	*0*							
	1	*1*							
	2	*2*							
	1	*0*	*-50*	*1*					
	2	*0*	*50*	*2*					
	3	*25*	*-50*	*1*					
	4	*25*	*50*	*2*					
	5	*50*	*-50*	*1*					
	6	*50*	*50*	*2*					
	7	*75*	*-50*	*1*					
	8	*75*	*50*	*2*					

APPENDIX B

MANAGEMENT SYSTEM SOFTWARE CODE

Main Code Module

'modCpMatrix.bas

'Variable Declarations

```
Option Explicit   'Makes Declarations Mandatory
Dim nfields As Integer
Dim length1 As Integer
Dim Length2 As Integer
Public strA As String
Public strB As String
Public NumRows As Integer
Public NumColumns As Integer
Public NPlots As Integer
Public NumPlots As Integer
Public PlotWidth As Single
Public PlotLength As Single
Private NoTurn As Integer
Public CpX(1 To 10000)
Public CpY(1 To 10000)
Public XMaxPer As Single
Public YMaxPer As Single
Public DegLat As Double
Public DegLon As Double
Public MinRefLat As Double
Public MinRefLon As Double
Public DegRefLat As Double
Public DegRefLon As Double
Public MinLat As Double
Public MinLon As Double
Public XMeters As Double
Public YMeters As Double
Public XFeet As Double
Public YFeet As Double
Public strLat As String
Public strLon As String
Public DX As Double
Public DY As Double
Public theta As Double
Public thetaf As Double
Public LCount As Double
Dim z As Integer
Dim r As Integer
Dim c As Integer
Dim PlotNum As Integer
Public AvgX(1 To 10000) As Double
Public AvgY(1 To 10000) As Double
Public LatSum As Double
Public LonSum As Double
Public XFeetSum As Double
Public YFeetSum As Double
Public Const ATad As Double = 0 0000000001
Public Const MtoFt As Double = 3 28083989501
Dim pn As Integer
Public xoff As Double
Public px(1 To 10) As Double
Public py(1 To 10) As Double
Public zz As Integer
Public strc As String
Public ww(0 To 50) As Integer
Public qwerty As Integer
Public strreflon As String
Public strreflat As String
Public PointName(1 To 10) As String
Dim i As Integer
Public strD As String
Public gg As Integer
Public LatToFt As Double
Public LonToFt As Double
Const pi = 3 14159265358979
Public LpX(1 To 1000) As Single
Public LpY(1 To 1000) As Single
Public nn As Integer
Public Resultant As Double
Public NumLp As Integer
Public Xo As Single
Public Mult As Integer
Public CosTheta As Double
Public SinTheta As Double
Public CosThetaO As Double
Public SinThetaO As Double
Public n As Integer
Public m As Integer
Public RNums(1 To 10000) As Single
Public RankedIndices(1 To 10000) As Integer
Public NumRates As Integer
Public MaxRNum As Single
Public MaxIndex As Integer
Public AssignedTreatments(1 To 10000) As Integer
Public ReRankedIndices(1 To 10000)
Public Directory As String
Public Drive As String
Public FileName As String
Public Temp As String
Public Dirs(1 To 1000) As String
Public DirCount As Integer
Public DirAlreadyExists As Boolean
Public SecondsPerPoint As Integer
Public File As String
Public PCCardDrive As String
Public Temp1 As String
Public Temp2 As String
Public ExpCount As Integer
Public Exps(1 To 10000) As String
Public DirLength As Integer
Public PerY(1 To 10000) As Double
Public PerX(1 To 10000) As Double
Public PerXr(1 To 10000) As Double
Public PerYr(1 To 10000) As Double
Public PerXrMax As Double
Public PerXrMin As Double
Public PerYrMax As Double
Public PerYrMin As Double
Public LatP As String
Public LonP As String
Public MostOffsetX As Integer
Public MostOffsetY As Integer
Public MostNumColumns As Integer
Public MostNumRows As Integer
Public MostPlotNum As Integer
Public MostCpX(1 To 10000) As Double
Public MostCpY(1 To 10000) As Double
Public LeastPlotNum As Integer
Public LeastCpX(1 To 10000) As Double
Public LeastCpY(1 To 10000) As Double
Public Sweep As Boolean
Public PerRes As Single
Public TurnRoom As Single
Public Diagonal As Single
Public PerCloseX As Single
Public PerCloseY As Single
```

```
Public CloseX As Single
Public CloseY As Single
Public PreviousSweep As Boolean
Public Inborder As Boolean
Public CurveyField As Boolean
Public Surrounded As Boolean
Public Quad1 As Integer
Public Quad2 As Integer
Public Quad3 As Integer
Public Quad4 As Integer
Public TotalQuad As Integer
Public CosPhi As Single
Public SinPhi As Single
Public ResultantPerClose As Single
Public CheckMod As Integer
Public XScale As Single
Public YScale As Single
Public PixOffX As Single
Public PixOffY As Single
Public lx1 As Single
Public ly1 As Single
Public lx2 As Single
Public ly2 As Single
Public PrintRate(0 To 10000) As Integer
Public PrintIndex As Integer
Public TreatmentPlot As Boolean
Public UseMe(0 To 10000) As Boolean
Public TrialX As Single
Public TrialY As Single
Public TrialIndex As Integer
Public CpDist As Single
Public MinCpDist As Single
Public Vectors(0 To 10000) As Single
Public TestFile As Boolean

Public Sub UpdateTable()
  frmManage txtAdjustX = CloseX
  frmManage txtAdjustY = CloseY
  frmManage txtNumPlots = LeastPlotNum
  frmManage txtNumreps = LeastPlotNum / NumRates
  frmManage txtNumtreats = NumRates
End Sub

Public Sub UpdateLegend()
  For n = 1 To 12
    frmManage txtLegend(n - 1) Visible = False
  Next n
  For n = 1 To NumRates
    frmManage txtLegend(n - 1) Visible = True
    frmManage txtLegend(n - 1) BackColor = QBColor(n)
    frmManage txtLegend(n - 1) Text =
        frmManage cboRates(n) Text
  Next n
End Sub

Public Sub Adjustment()
  Call SweepAndSurround
  Call ReNameCps
  Call MainRandom
  Call UpdateTable
  frmManage Cls
  Call DrawCp
  Call DrawOutlines
  Call DrawPerimeter
End Sub

Public Sub DrawOutlines()
  frmManage FillStyle = 0
  frmManage DrawWidth = 1
  For n = 1 To NumPlots
    lx1 = (CpX(n) - PlotWidth / 2 - PerXrMin) * XScale +
        PixOffX
    ly1 = (CpY(n) - PlotLength / 2 - PerYrMin) * YScale +
        PixOffY
    lx2 = (CpX(n) + PlotWidth / 2 - PerXrMin) * XScale +
        PixOffX
    ly2 = (CpY(n) + PlotLength / 2 - PerYrMin) * YScale +
        PixOffY
    frmManage FillColor = QBColor(ReRankedIndices(n))
    frmManage Line (lx1, ly1)-(lx2, ly2), vbWhite, B
  Next n
End Sub

Public Sub DrawPerimeter()
  frmManage DrawStyle = 0
  frmManage ForeColor = vbWhite
  frmManage DrawWidth = 1
  For n = 1 To zz - 1
    lx1 = (PerXr(n) - PerXrMin) * XScale + PixOffX
    ly1 = (PerYr(n) - PerYrMin) * YScale + PixOffY
    lx2 = (PerXr(n + 1) - PerXrMin) * XScale + PixOffX
    ly2 = (PerYr(n + 1) - PerYrMin) * YScale + PixOffY
    frmManage Line (lx1, ly1)-(lx2, ly2)
  Next n
End Sub

Public Sub DrawCp()
  frmManage DrawWidth = 3
  PixOffX = frmManage fraMainMenu Width + 30
  PixOffY = 30
  XScale = (600 - PixOffX) / Abs(PerXrMax - PerXrMin)
  YScale = (400 - PixOffY) / Abs(PerYrMax - PerYrMin)
  If frmManage cboEQUAL = "TRUE" Then XScale =
        YScale
  For n = 1 To NumPlots
    lx1 = (CpX(n) - PerXrMin) * XScale + PixOffX
    ly1 = (CpY(n) - PerYrMin) * YScale + PixOffY
    frmManage PSet (lx1, ly1),
        QBColor(ReRankedIndices(n))
  Next n
End Sub

Public Sub DrawSkeletonCrew()
  frmManage Cls
  For n = 1 To MostPlotNum
    lx1 = (MostCpX(n) - PerXrMin) * XScale + PixOffX
    ly1 = (MostCpY(n) - PerYrMin) * YScale + PixOffY
    frmManage DrawWidth = 3
    If UseMe(n) = True Then
      frmManage PSet (lx1, ly1), vbGreen
    Else
      frmManage PSet (lx1, ly1), vbRed
    End If
    frmManage DrawWidth = 1
    lx1 = (MostCpX(n) - PlotWidth / 2 - PerXrMin) *
        XScale + PixOffX
    ly1 = (MostCpY(n) - PlotLength / 2 - PerYrMin) *
        YScale + PixOffY
    lx2 = (MostCpX(n) + PlotWidth / 2 - PerXrMin) *
        XScale + PixOffX
    ly2 = (MostCpY(n) + PlotLength / 2 - PerYrMin) *
        YScale + PixOffY
    frmManage FillStyle = 1
    frmManage Line (lx1, ly1)-(lx2, ly2), vbWhite, B
```

```
    Next n
    For n = 1 To LeastPlotNum
      lx1 = (LeastCpX(n) - PlotWidth / 2 - PerXrMin) *
            XScale + PixOffX
      ly1 = (LeastCpY(n) - PlotLength / 2 - PerYrMin) *
            YScale + PixOffY
      lx2 = (LeastCpX(n) + PlotWidth / 2 - PerXrMin) *
            XScale + PixOffX
      ly2 = (LeastCpY(n) + PlotLength / 2 - PerYrMin) *
            YScale + PixOffY
      frmManage FillStyle = 7 '5
      frmManage FillColor = vbGreen
      frmManage Line (lx1, ly1)-(lx2, ly2), vbWhite, B
    Next n
End Sub

Public Sub MouseTrap()
    MinCpDist = 100000
    For n = 1 To MostPlotNum
      CpDist = ((TrialX - MostCpX(n)) ^ 2 + (TrialY -
            MostCpY(n)) ^ 2) ^ (0 5)
      If CpDist < MinCpDist Then
        MinCpDist = CpDist
        TrialIndex = n
      End If
    Next n
    If UseMe(TrialIndex) = True Then
      UseMe(TrialIndex) = False
    Else
      UseMe(TrialIndex) = True
    End If
    lx1 = (MostCpX(TrialIndex) - PerXrMin) * XScale +
          PixOffX
    ly1 = (MostCpY(TrialIndex) - PerYrMin) * YScale +
          PixOffY
    frmManage DrawWidth = 3
    If UseMe(TrialIndex) = True Then
      frmManage PSet (lx1, ly1), vbGreen
    Else
      frmManage PSet (lx1, ly1), vbRed
    End If
End Sub

Public Sub FillUseMe()
    For n = 1 To MostPlotNum
      UseMe(n) = True
    Next n
End Sub

Public Sub SearchDirectories()
    Temp = Dir(Drive, vbDirectory)
    DirCount = 0
    Do Until Temp = ""
      DirCount = DirCount + 1
      Dirs(DirCount) = Temp
      Temp = Dir
    Loop
End Sub

Public Sub MakeWay()
    Call SearchDirectories
    DirAlreadyExists = False
    For n = 1 To DirCount
      If Dirs(n) = Directory Then
        DirAlreadyExists = True
      End If
    Next n
End Sub

Public Sub FileInfo()
    FileName = frmManage cboReferenceFile Text
    DirLength = Len(FileName) - 4
    Directory = Mid$(FileName, 1, DirLength)
End Sub

Public Sub MakeFile()
      File = PCCardDrive + " \" + Directory + "\" + Directory
            + " exp"
      Open File For Output As #2
      Call PrintHeader
      Call PrintRates
      If TestFile = True Then
        Call PrintLineAssignments
      Else  Call PrintAssignments
      End If
      Close #2
End Sub

Public Sub FindReferenceFiles()
    On Error GoTo ehandle
    PCCardDrive = frmManage txtPCCardDrive Text
    Temp1 = Dir(PCCardDrive + " \", vbDirectory)
      m = 0
      Do Until Temp1 = ""
        m = m + 1
        Dirs(m) = Temp1
        Temp1 = Dir
      Loop
      ExpCount = 0
      For n = 1 To m
        Temp2 = Dir(PCCardDrive + " \" + Dirs(n) +
              "\* pts")
        If InStr(Temp2, " pts") Then
          ExpCount = ExpCount + 1
          Exps(ExpCount) = Temp2
        End If
      Next n
If n = "-45" Then
ehandle
    frmManage fraSettings Enabled = False
    frmManage fraSettings Visible = True
    frmManage txtPCCardError Visible = True
    frmManage fraReferenceFile Visible = False
    frmManage tmrError Interval = 1500
    frmManage tmrError Enabled = True
End If
End Sub

Public Sub EraseFileNames()
    If ExpCount >= 1 Then
      For n = 1 To ExpCount
        frmManage cboReferenceFile RemoveItem (0)
      Next n
    End If
End Sub

Public Sub ListFileNames()
    frmManage cboReferenceFile Text = Exps(1)
    For n = 1 To ExpCount
      frmManage cboReferenceFile AddItem Exps(n)
    Next n
End Sub

Public Sub Initial()
```

```
    TestFile = False
    CloseX = 20
    CloseY = 20
End Sub

Public Sub PrintHeader()
    Print #2, frmManage cboReferenceFile Text + Chr$(9) +
              Str$(DegRefLat) + Chr$(9) + Str$(DegRefLon) +
              Chr$(9) + frmManage txtPlotLength Text +
              Chr$(9) + frmManage txtPlotWidth Text +
              Chr$(9) + Str$(NumRates) + Chr$(9) +
              Str$(MostPlotNum) + Chr$(9) + Str$(CosTheta)
              + Chr$(9) + Str$(SinTheta) + Chr$(9) +
              Str$(MostNumColumns) + Chr$(9)
End Sub

Public Sub ReadPoints()
    File = PCCardDrive + " \" + Directory + "\" + FileName '
              this is the pts file listed
    Open File For Input As #1
      For zz = 1 To 2
            Line Input #1, strc
            length1 = Len(strc)
            nfields = 0
          For i = 1 To length1
            strD = Mid$(strc, i, 1)
            If strD = Chr$(9) Then
              nfields = nfields + 1
              ww(nfields) = i
            End If
          Next i
          qwerty = ww(0) + 1
          Length2 = ww(1) - ww(0) - 1
          PointName(zz) = Mid$(strc, qwerty, Length2)
          qwerty = ww(1) + 1
          Length2 = ww(2) - ww(1) - 1
          strreflon = Mid$(strc, qwerty, Length2)
          px(zz) = Val(strreflon)
          qwerty = ww(2) + 1
          Length2 = ww(3) - ww(2) - 1
          strreflat = Mid$(strc, qwerty, Length2)
          py(zz) = Val(strreflat)
Next zz
      Close #1
End Sub

Public Sub Convert()
    For gg = 2 To 3
      px(gg) = (px(gg) - DegRefLon) * LonToFt
      py(gg) = (py(gg) - DegRefLat) * LatToFt
    Next gg
    px(1) = 0
    py(1) = 0
End Sub

Public Sub RefPointCalc()
    DegRefLat = py(1)
    DegRefLon = px(1)
    MinRefLon = px(1) * 60
    MinRefLat = py(1) * 60
    Const pi = 3 14159265358979
    LonToFt = Cos(DegRefLat * pi / 180) * 365157 480315
    LatToFt = 363188 976378
End Sub

Public Sub ReNameCps()
    NumPlots = LeastPlotNum
    For n = 1 To LeastPlotNum
      CpX(n) = LeastCpX(n)
      CpY(n) = LeastCpY(n)
    Next n
End Sub

Public Sub ThetaCalc()
    DX = px(2) - px(1)
    DY = py(2) - py(1)
    theta = Abs(Atn(DY / DX))
    thetaf = pi / 2 - theta
    If px(2) > px(1) Then
     If py(3) > py(1) Then
        xoff = PlotWidth * NumColumns
      Else xoff = 0
      End If
    Else
      If py(3) < py(1) Then
        xoff = PlotWidth * NumColumns
      Else xoff = 0
      End If
    End If
End Sub

Public Sub LoadPerimeter()
    File = PCCardDrive + " \" + Directory + "\" + Directory +
           " per"
    zz = 0
    Open File For Input As #1
    Do While Not EOF(1) ' Check for end of file
      Line Input #1, strc
      zz = zz + 1
      length1 = Len(strc)
      nfields = 0
    For i = 1 To length1
      strD = Mid$(strc, i, 1)
      If strD = Chr$(9) Then
        nfields = nfields + 1
        ww(nfields) = i
      End If
    Next i
    qwerty = ww(0) + 1
    Length2 = ww(1) - ww(0) - 1
    LonP = Mid$(strc, qwerty, Length2)
    PerX(zz) = (Val(LonP) - DegRefLon) * LonToFt
    qwerty = ww(1) + 1
    Length2 = ww(2) - ww(1) - 1
    LatP = Mid$(strc, qwerty, Length2)
    PerY(zz) = (Val(LatP) - DegRefLat) * LatToFt
    PerXr(zz) = (PerX(zz) * CosTheta + PerY(zz) * SinTheta)
           + xoff
    PerYr(zz) = (PerY(zz) * CosTheta - PerX(zz) * SinTheta)
Loop
Close #1
End Sub

Public Sub MostNumberPlots()
    MostOffsetX = Fix(PerXrMin + PlotWidth / 2)
    MostOffsetY = Fix(PerYrMin + PlotLength / 2)
    MostNumRows = Fix(Abs((PerYrMax - PerYrMin) /
          PlotLength))
    MostNumColumns = Fix(Abs((PerXrMax - PerXrMin) /
          PlotWidth))
    MostPlotNum = 0
    For c = 1 To MostNumColumns
      Vectors(c) = c * PlotWidth - PlotWidth + MostOffsetX
      For r = 1 To MostNumRows
```

```
        MostPlotNum = MostPlotNum + 1
        MostCpX(MostPlotNum) = Vectors(c)
        MostCpY(MostPlotNum) = (MostNumRows + 1 - r)
            * (PlotLength) - PlotLength + MostOffsetY
      Next r
    Next c
    Call FillUseMe
End Sub

Public Sub SweepPlotsSeparate()
    CloseX = 20
    CloseY = 20
    LeastPlotNum = 1
    Open "c \vbtracs\MostPlotTest txt" For Output As #1
    For n = 1 To MostPlotNum
      Sweep = True
      For m = 1 To zz
        PerCloseX = Abs(MostCpX(n) - PerXr(m))
        PerCloseY = Abs(MostCpY(n) - PerYr(m))
        If PerCloseX < CloseX And PerCloseY < CloseY
            Then
          Sweep = False
        End If
      Next m
      If Sweep = True Then
        LeastCpX(LeastPlotNum) = MostCpX(n)
        LeastCpY(LeastPlotNum) = MostCpY(n)
        Print #1, Str$(LeastCpX(LeastPlotNum)) + "," +
            Str$(LeastCpY(LeastPlotNum))
        LeastPlotNum = LeastPlotNum + 1
      End If
    Next n
Close #1
End Sub

Public Sub SweepAndSurround()
  LeastPlotNum = 0
  For n = 1 To MostPlotNum
      Sweep = True
      Quad1 = 0
      Quad2 = 0
      Quad3 = 0
      Quad4 = 0
      TotalQuad = 0
      Surrounded = False
      For m = 1 To zz
        PerCloseX = -MostCpX(n) + PerXr(m)
        PerCloseY = -MostCpY(n) + PerYr(m)
        If Abs(PerCloseX) < CloseX And Abs(PerCloseY) <
            CloseY Then
          Sweep = False
        End If
        ResultantPerClose = (PerCloseX ^ 2 + PerCloseY ^
            2) ^ 0 5
        CosPhi = PerCloseX / ResultantPerClose
        SinPhi = PerCloseY / ResultantPerClose
        If CosPhi > 0 And SinPhi > 0 Then
          Quad1 = 1
        ElseIf CosPhi < 0 And SinPhi > 0 Then
          Quad2 = 1
        ElseIf CosPhi < 0 And SinPhi < 0 Then
          Quad3 = 1
        ElseIf CosPhi > 0 And SinPhi < 0 Then
          Quad4 = 1
        End If
      Next m
      TotalQuad = Quad1 + Quad2 + Quad3 + Quad4
      If TotalQuad >= 3 Then Surrounded = True
      If Sweep = True And Surrounded = True And UseMe(n)
          = True Then
        LeastPlotNum = LeastPlotNum + 1
        LeastCpX(LeastPlotNum) = MostCpX(n)
        LeastCpY(LeastPlotNum) = MostCpY(n)
      End If
    Next n
    CheckMod = LeastPlotNum Mod NumRates
    If CheckMod <> 0 Then
      LeastPlotNum = LeastPlotNum - CheckMod
    End If
End Sub

Public Sub OffSets()
  PerXrMin = 1000000
  PerYrMin = PerXrMin
  PerXrMax = -1000000
  PerYrMax = PerXrMax
  For n = 1 To zz          ""Must Call this sub after
              FindMaxPer
      If PerXr(n) < PerXrMin Then
        PerXrMin = PerXr(n)
      End If
      If PerYr(n) < PerYrMin Then
        PerYrMin = PerYr(n)
      End If
      If PerXr(n) > PerXrMax Then
        PerXrMax = PerXr(n)
      End If
      If PerYr(n) > PerYrMax Then
        PerYrMax = PerYr(n)
      End If

    Next n
End Sub

Public Sub MaxPlots()
    Call LoadPerimeter
    Call OffSets
    Call MostNumberPlots
    CurveyField = True 'False 'Deactivates unpredictable
            procedure
    If CurveyField = True Then
      Call SweepAndSurround
    Else  Call SweepPlotsSeparate
    End If
End Sub

Public Sub NumberPlots()
    NumRates = frmManage txtNumRates
    PlotWidth = frmManage txtPlotWidth Text
    PlotLength = frmManage txtPlotLength Text
    NumRows = frmManage txtNumRows Text
    NumColumns = frmManage txtNumColumns Text
    NumPlots = NumRows * NumColumns

    PlotNum = 1
    For c = 1 To NumColumns
      For r = 1 To NumRows
        CpX(PlotNum) = c * PlotWidth - PlotWidth  '/ 2 --No
            offset(all should be done in Field System)
        CpY(PlotNum) = (NumRows + 1 - r) * (PlotLength) -
            PlotLength  '/ 2
        PlotNum = PlotNum + 1
      Next r
    Next c
```

```
End Sub

Public Sub Delayy()
For qwerty = 0 To 100000#
Next qwerty
End Sub
Public Sub MainProgram()
  Call FileInfo
  Call ReadPoints
  Call RefPointCalc
  Call Convert
  Call CalcConst
  Call ThetaCalc2pi
  If TestFile = False Then
    Call MaxPlots
    Call ReNameCps
    Call MainRandom
    Call UpdateLegend
  Else  Call MainLineTest
  End If
End Sub

Public Sub MainLineTest()
  Call LpCoords2pi
  Call CpCoords2pi
  SinTheta = SinThetaO
  CosTheta = CosThetaO
  frmManage txtNumRates Enabled = False
  MostPlotNum = NumLp * 2
  MostNumColumns = NumLp * 2
  NumRates = 2
  For n = 1 To 2
    frmManage cboRates(n) Text = n
  Next n
  For n = 1 To MostPlotNum
      Vectors(n) = CpX(n)
  Next n
  Call MakeFile
  frmManage fraDimensions Visible = False
  frmManage fraMainMenu Visible = True
  TestFile = False
End Sub

Public Sub PrintLineAssignments()
  For n = 1 To NumLp * 2
    If n Mod 2 = 0 Then
      PrintRate(n) = 2
    Else  PrintRate(n) = 1
    End If
    Print #2, Str$(n) + Chr$(9) + Str$(CpX(n)) + Chr$(9) +
        Str$(CpY(n)) + Chr$(9) + Str$(PrintRate(n)) +
        Chr$(9)
  Next n
End Sub

Public Sub MainRandom()
  """""Assigning Treatments at Random"""""
  Call GenerateRandomNumbers
  Call RankInDescendingOrder
  Call AssignPlotsToTreatments
  Call ReRank
  '''''''
```

```
Public Sub CpCoords()
  For nn = 1 To NumLp * 2
    CpX(nn) = LpX(nn)
  Next nn
  For nn = 1 To NumLp * 2 Step 2
    CpY(nn) = LpY(nn) + PlotLength / 2
    CpY(nn + 1) = LpY(nn + 1) - PlotLength / 2
  Next nn
End Sub

Public Sub GenerateRandomNumbers()
'*This Module Contains the SubRoutines to Randomize
 'the treatments with in the number of plots created in the
 'Management System
Randomize 'This statement re-seeds the random no  gen from
          the system clock
  For n = 1 To NumPlots
    RNums(n) = Rnd
  Next n
End Sub

Public Sub RankInDescendingOrder()
  For m = 1 To NumPlots
    MaxRNum = 0
    For n = 1 To NumPlots
      If RNums(n) > MaxRNum Then
        MaxRNum = RNums(n)
        MaxIndex = n
      End If
    Next n
    RankedIndices(m) = MaxIndex
    RNums(MaxIndex) = 0
  Next m
End Sub

Public Sub AssignPlotsToTreatments()
  For m = 1 To NumRates
    For n = (m - 1) * (NumPlots / NumRates) + 1 To
          NumPlots / NumRates * m
      AssignedTreatments(n) = m
    Next n
  Next m
End Sub

Public Sub ReRank()
  For n = 1 To NumPlots
    ReRankedIndices(n) =
        AssignedTreatments(RankedIndices(n))
  Next n
End Sub

Public Sub PrintAssignmentsOld2()
  For n = 1 To NumPlots
    Print #2, Str$(n) + Chr$(9) + Str$(CpX(n)) + Chr$(9) +
        Str$(CpY(n)) + Chr$(9) +
        Str$(ReRankedIndices(n)) + Chr$(9)
  Next n
End Sub

Public Sub PrintAssignments()
  For n = 1 To MostPlotNum
    TreatmentPlot = False
    For m = 1 To LeastPlotNum
     If CpY(m) = MostCpY(n) Then
      If CpX(m) = MostCpX(n) Then
        PrintIndex = m
        TreatmentPlot = True
      End If
     End If
    Next m
    If TreatmentPlot = True Then
      PrintRate(n) = ReRankedIndices(PrintIndex)
    Else PrintRate(n) = 0
    End If
    Print #2, Str$(n) + Chr$(9) + Str$(MostCpX(n)) +
        Chr$(9) + Str$(MostCpY(n)) + Chr$(9) +
        Str$(PrintRate(n)) + Chr$(9)
  Next n
End Sub

Public Sub PrintRates()
  For n = 0 To NumRates
    Print #2, Str$(n) + Chr$(9) +
      frmManage cboRates(n) Text + Chr$(9)
  Next n
End Sub

Public Sub PrintVectors()
  For n = 1 To MostNumColumns
    Print #2, "V" + Chr$(9) + Str$(Vectors(n)) + Chr$(9)
  Next n
End Sub

Public Sub RatesVisibility()
  For n = 0 To 12
    frmManage cboRates(n) Visible = False
    frmManage Text1(n) Visible = False
  Next n
  For n = 0 To NumRates
    frmManage cboRates(n) Visible = True
    frmManage Text1(n) Visible = True
  Next n
End Sub
```

Graphical Module

```
'frmManage

Private Sub cmdBack_Click()
  End
End Sub

Private Sub cboRates_LostFocus(Index As Integer)
 For n = 0 To 12
  cboRates(n) Text = CSng(cboRates(n) Text)
  If cboRates(n) Text > 10000 Then
    cboRates(n) Text = 10000
  ElseIf cboRates(n) Text < 0 Then
    cboRates(n) Text = 0
  End If
 Next n
End Sub

Private Sub cmdAddRemove_Click()
  Call DrawSkeletonCrew
  Call DrawPerimeter
End Sub

Private Sub cmdAdjustOK_Click()
  Call Adjustment
End Sub
```

```
Private Sub cmdBrowse_Click()
  frmFiles Visible = True
End Sub

Private Sub cmdDesignExperiment_Click()
  fraMainMenu Visible = False
  fraReferenceFile Visible = True
  Call FindReferenceFiles
  Call ListFileNames
  TestFile = False
End Sub

Private Sub cmdDimensionsBack_Click()
  fraDimensions Visible = False
  fraReferenceFile Visible = True
End Sub

Private Sub cmdDimensionsNext_Click()
  fraDimensions Visible = False
  fraRates Visible = True
  NumRates = txtNumRates Text
  Call RatesVisibility
  Call MainProgram
End Sub

Private Sub cmdExit_Click()
  End
End Sub

Private Sub cmdMakeTestFile_Click()
  cmdDesignExperiment_Click
  TestFile = True
End Sub

Private Sub cmdMinusY_Click()
  CloseY = CloseY - PlotWidth / 4
  If CloseY < 0 Then CloseY = 0
  cmdAdjustOK_Click
End Sub

Private Sub cmdModifyExperimentFinish_Click()
  Call MakeFile
  End
End Sub

Private Sub cmdPlusX_Click()
  CloseX = CloseX + PlotWidth / 4
  cmdAdjustOK_Click
End Sub

Private Sub cmdPlusY_Click()
  CloseY = CloseY + PlotLength / 4
  cmdAdjustOK_Click
End Sub

Private Sub cmdRatesBack_Click()
  fraRates Visible = False
  fraDimensions Visible = True
End Sub

Private Sub cmdRatesNext_Click()
  fraRates Visible = False
  fraModifyExperiment.Visible = True
  frmManage Cls
  Call UpdateTable
  Call UpdateLegend
  Call DrawCp
  Call DrawOutlines
  Call DrawPerimeter
End Sub

Private Sub cmdReferenceFileBack_Click()
  fraReferenceFile Visible = False
  fraMainMenu Visible = True
End Sub

Private Sub cmdReferenceFileNext_Click()
  fraReferenceFile Visible = False
  fraDimensions Visible = True
End Sub

Private Sub cmdSettings_Click()
  fraMainMenu Visible = False
  fraSettings Visible = True
End Sub

Private Sub cmdSettingsBack_Click()
  fraSettings Visible = False
  fraMainMenu Visible = True
End Sub

Private Sub cmdStartAgain_Click()
  fraRates Visible = True
  fraModifyExperiment Visible = False
  Call EraseFileNames
End Sub

Private Sub cmdMinusX_Click()
  CloseX = CloseX - PlotWidth / 4
  If CloseX < 0 Then CloseX = 0
  cmdAdjustOK_Click
End Sub

Private Sub Form_Load()
  Call Initial
End Sub

Private Sub tmrError_Timer()
  tmrError Enabled = False
  frmManage fraSettings Enabled = True
  Call EraseFileNames
End Sub

Private Sub txtNumRates_LostFocus()
  txtNumRates Text = CInt(txtNumRates Text)
  If txtNumRates Text > 12 Then
    txtNumRates Text = 12
  ElseIf txtNumRates Text < 1 Then
    txtNumRates Text = 1
  End If
End Sub

Private Sub form_MouseDown(Button As Integer, Shift
        As Integer, X As Single, Y As Single)
  If X > fraMainMenu Width And
        fraModifyExperiment Visible = True Then
    TrialX = (X - PixOffX) / XScale + PerXrMin
    TrialY = (Y - PixOffY) / YScale + PerYrMin
    Call MouseTrap
  End If
End Sub

Private Sub txtPCCardDrive_click()
```

```
    txtPCCardError Visible = False
End Sub

Private Sub txtPlotWidth_LostFocus()
    txtPlotWidth Text = CInt(txtPlotWidth Text)
    If txtPlotWidth Text > 5280 Then
        txtPlotWidth Text = 5280
    ElseIf txtPlotWidth Text < 1 Then
        txtPlotWidth Text = 1
    End If
End Sub

Private Sub txtplotlength_LostFocus()
    txtPlotLength Text = CInt(txtPlotLength Text)
    If txtPlotLength Text > 5280 Then
        txtPlotLength Text = 5280
    ElseIf txtPlotLength Text < 1 Then
        txtPlotLength Text = 1
    End If
End Sub
```

APPENDIX C

COORDINATE COLLECTOR SOFTWARE CODE

Main Code Module

'Collector.bas

'Variable Declarations
Option Explicit 'Makes Variable Declarations Mandatory

Public Xes(1 To 1000) As Double
Public Yes(1 To 1000) As Double

Public Inp As String
Public PointNumber As Integer
Public Collecting As Boolean
Public File As String
Public Directory As String

Public n As Integer
Public CTime As Single
Public Buf As String
Public gps$
Public CollectingPerimeter As Boolean
Public PCount As Integer
Public LCount As Integer
Public XSum As Double
Public YSum As Double
Public XAvg(1 To 1000) As Double
Public YAvg(1 To 1000) As Double
Public Trash As String
Public xs As String
Public ys As String
Public X As Double
Public Y As Double
Public cc As Integer
Public mm As Double
Public Drive As String
Public FileName As String

Public Temp As String
Public Dirs(1 To 100) As String
Public DirCount As Integer
Public DirAlreadyExists As Boolean
Public SecondsPerPoint As Integer

Public NumPoints As Integer
Public PointNum As Integer

Public f As String

Public Xc As Single
Public Yc As Single
Public XcOld As Single
Public YcOld As Single
Public XScale As Single
Public YScale As Single
Public XMaxPer As Single
Public YMaxPer As Single
Public XMax As Single
Public YMax As Single
Public XMin As Single
Public YMin As Single
Public DegRefLat As Double
Public DegRefLon As Double
Public LatToFt As Double
Public LonToFt As Double
Public XOff As Single
Public YOff As Single

Public lx1 As Single
Public ly1 As Single
Public lx2 As Single
Public ly2 As Single
Public pi As Double
Public PointCount As Integer
Public PerX(0 To 10000) As Single
Public PerY(0 To 10000) As Single
Public PMod As Single

Public Tracking As Boolean
Public PlotX(0 To 10000) As Single
Public PlotY(0 To 10000) As Single

Public PixOffX As Integer
Public PixOffY As Integer
Public CAvgX(1 To 10) As Single
Public CAvgY(1 To 10) As Single
Public CXes(1 To 1000) As Single
Public CYes(1 To 1000) As Single
Public CXsum As Single
Public CYsum As Single
Public LoopCount As Double
Public DoEventsMod As Double

Public Sub ReadNmea()
'ReadNmea - A sub routine to extract the following
' information
' from the NMEA GGA output from a GPS
' Reciever
' 1 Lat
' 2 Long
'To use this sub routine, a form named "frmComm" with an
' MSComm
'(telephone icon)object named MSComm1 included on it
'This sub-routine separates the GGA string into the 2 needed
' strings
'Import only one char at a time
frmPOINT MSComm1 InputLen = 1
Start
'Look at the buffer until there is something in it
DoEventsMod = 0
LoopCount = 0
Do Until frmPOINT MSComm1 InBufferCount > 0
LoopCount = LoopCount + 1
DoEventsMod = LoopCount Mod 10
If DoEventsMod = 0 Then DoEvents
Loop
'Initialize the input string to nothing
Inp$ = ""
'Check the input string until it is a "$" mark
DoEventsMod = 0
LoopCount = 0
Do Until Inp$ = "$"
Inp$ = frmPOINT MSComm1 Input
LoopCount = LoopCount + 1
DoEventsMod = LoopCount Mod 40
If DoEventsMod = 0 Then DoEvents
Loop

'Pull out the GPS identifier using "pull()" function
'created below
gps$ = Pull("")
'If the GPS String is a GGA, then use the "pull()"
'function to pull out the 2 needed strings
'Trash$ is used to skip fields in the comma delimited
'NMEA string

```
    If InStr(gps$, "GGA") Then

        Trash = Pull("")          'Field 1
        ys = Pull("")         'Field 2
        Trash = Pull("")      'Field 3
        xs = Pull("")         'Field 4
        Call CommError

        'Trash = Pull("")       'Field 5
        'Qualitys = Pull("")    'Field 6
        'SVNums = Pull("")      'Field 7
        'HDOPs = Pull("")       'Field 8
    'If the string is not GGA, then start over until
    'a GGA string is found
    Else  GoTo Start
    End If
End Sub

Public Function Pull(inpu$) As String
'This function pulls out strings in between commas
'by adding another character to the string it is checking
'until it finds a comma
'The only argument needed is "Inpu$ "
'Inpu$ is the initial string - it should be set to ""
'in this application
Dim z As String
    DoEventsMod = 0
    LoopCount = 0
    Do Until inpu$ = ","
        z = z + inpu$
        inpu$ = frmPOINT MSComm1 Input
        LoopCount = LoopCount + 1
        DoEventsMod = LoopCount Mod 40
        If DoEventsMod = 0 Then DoEvents
    Loop
Pull = z
End Function

Sub CommError()
    If Len(xs) < 4 Then
        frmPOINT MSComm1 InBufferCount = 0
        Call ReadNmea
    End If
End Sub
Public Sub Convert()
    XcOld = Xc
    YcOld = Yc
    X = Val(Mid$(xs, 1, 3)) + Val(Mid$(xs, 4, ((Len(xs) -
            3)))) / 60
    Xc = LonToFt * (X - DegRefLon) 'Cartesian
    Y = Val(Mid$(ys, 1, 2)) + Val(Mid$(ys, 3, ((Len(ys) -
            2)))) / 60
    Yc = LatToFt * (Y - DegRefLat) 'Cartesian
End Sub

Public Sub Perimeter()
    On Error GoTo EHandle
Start
    SecondsPerPoint =
            CInt(frmPointCollector cboPerimeterTime Text)
    Directory = frmPointCollector txtExperimentName Text
    "Replace the following line's form with one from FS
    Drive = frmMenu cboPCCardDrive Text + " \"
    FileName = Directory
    Call MakeDirectory
    File = Drive + Directory + "\" + FileName + " per"
    Open File For Output As #1
    PCount = 0
    Call BufferFlush
    Call ReferencePoint
    Call InitializePlotVariables
    PointCount = 0
    Call BufferFlush
    Do Until CollectingPerimeter = False
        PCount = PCount + 1
        PMod = PCount Mod SecondsPerPoint
        Call ReadNmea
        Call Convert
        PlotX(PCount) = Xc
        PlotY(PCount) = Yc

        If PMod = 0 Then
            PointCount = PointCount + 1
            PerX(PointCount) = Xc
            PerY(PointCount) = Yc
            Print #1, CStr(X) + Chr$(9) + CStr(Y) + Chr$(9)
            Call OffSets
        End If
        If PointCount >= 2 Then
            Call PlotPerimeter
            Call PlotTracking
        End If
Loop
If n = "a" Then
EHandle
        frmMenu fraSettings Enabled = False
        frmMenu Visible = True
        frmMenu fraMenu Visible = False
        frmMenu fraSettings Visible = True
        frmMenu txtSettingsError Visible = True
        frmPointCollector fraStep3 Visible = False
        frmPointCollector cmdStartPerimeter Enabled = True
        frmPointCollector cmdStep2Next Enabled = False
        frmPOINT tmrError Interval = 1500
        frmPOINT tmrError Enabled = True
End If
End Sub

Public Sub PlotTracking()
    lx1 = (Xc + XOff) * XScale + PixOffX
    ly1 = (Yc + YOff) * YScale + PixOffY
    frmPointCollector ForeColor = vbWhite
    frmPointCollector DrawWidth = 5
    frmPointCollector PSet (lx1, ly1)
End Sub

Public Sub PlotCoverUp()
    frmPointCollector ForeColor = vbBlack
    frmPointCollector PSet (lx1, ly1)
End Sub

Public Sub PlotAvgPoint1()
    lx1 = (CAvgX(1) + XOff) * XScale + PixOffX
    ly1 = (CAvgY(1) + YOff) * YScale + PixOffY
    frmPointCollector ForeColor = vbWhite
    frmPointCollector DrawWidth = 5
    frmPointCollector PSet (lx1, ly1)
End Sub

Public Sub PlotAvgPoint2()
    lx1 = (CAvgX(2) + XOff) * XScale + PixOffX
    ly1 = (CAvgY(2) + YOff) * YScale + PixOffY
    frmPointCollector ForeColor = vbWhite
    frmPointCollector DrawWidth = 5
```

```
    frmPointCollector PSet (lx1, ly1)
End Sub

Public Sub GETPTS()
  Tracking = False
  Call BufferFlush
  PCount = 0
  For n = 1 To NumPoints
     Call ReadNmea
     Call Convert
     Xes(n) = X
     Yes(n) = Y
     CXes(n) = Xc
     CYes(n) = Yc
  Next n
  Tracking = True
End Sub

Public Sub BufferFlush()
  frmPOINT MSComm1 InBufferCount = 0
End Sub

Public Sub Track()
      Do While Tracking = True
     Call PlotTracking
     Call ReadNmea
     Call Convert
     Call PlotCoverUp
  Loop
End Sub

Public Sub AVGPTS()
  XSum = 0
  YSum = 0
  CXsum = 0
  CYsum = 0
  For n = 1 To NumPoints
     XSum = XSum + Xes(n)
     YSum = YSum + Yes(n)
     CXsum = CXsum + CXes(n)
     CYsum = CYsum + CYes(n)
  Next n
  XAvg(PointNum) = XSum / NumPoints
  YAvg(PointNum) = YSum / NumPoints
  CAvgX(PointNum) = CXsum / NumPoints
  CAvgY(PointNum) = CYsum / NumPoints
End Sub

Public Sub SearchDirectories()
  On Error Resume Next
  Temp = Dir(Drive, vbDirectory)
  DirCount = 0
  Do Until Temp = ""
     DirCount = DirCount + 1
     Dirs(DirCount) = Temp
     Temp = Dir
  Loop
End Sub

Public Sub MakeDirectory()
  Call SearchDirectories
  DirAlreadyExists = False
  For n = 1 To DirCount
     If Dirs(n) = Directory Then
        DirAlreadyExists = True
     End If
  Next n
  If DirAlreadyExists = False Then
     MkDir (Drive + Directory)
  End If
End Sub

Public Sub MakeFile()
  Directory = frmPointCollector txtExperimentName Text
  Drive = frmMenu cboPCCardDrive Text + " \"
  FileName = Directory
  Call MakeDirectory
  File = Drive + Directory + "\" + FileName + " pts"
  Open File For Output As #2
  For cc = 1 To 2
     Print #2, "Point" + CStr(cc) + Chr$(9) +
           CStr(XAvg(cc)) + Chr$(9) + CStr(YAvg(cc)) +
           Chr$(9)
  Next cc
  Close #2
End Sub

Public Sub ExperimentNameOK()
  f = ""
  For n = 0 To 6
     Temp$ =
           CStr(frmPointCollector cboExperimentName(n) T
           ext)
     f = f + Temp$
  Next n
  frmPointCollector txtExperimentName Text = f
End Sub

Public Sub OffSets()
     If Xc < XMin Then
        XMin = Xc
     End If
     If Yc < YMin Then
        YMin = Yc
     End If
     If Xc > XMax Then
        XMax = Xc
     End If
     If Yc > YMax Then
        YMax = Yc
     End If
End Sub

Public Sub InitializePlotVariables()
  XMin = 1000000
  YMin = XMin
  XMax = -1000000
  YMax = XMax
  pi = Atn(1) * 4
  LonToFt = Cos(DegRefLat * pi / 180) * 365157 480315
  LatToFt = 363188 976378
  PixOffX = 10
  PixOffY = 90
End Sub

Public Sub PlotPerimeter()
  frmPointCollector DrawWidth = 1
  If PMod = 0 Then Call AutoScale
  frmPointCollector Cls
  frmPointCollector ForeColor = vbWhite
  For n = 2 To PCount
     lx1 = (PlotX(n - 1) + XOff) * XScale + PixOffX
     ly1 = (PlotY(n - 1) + YOff) * YScale + PixOffY
     lx2 = (PlotX(n) + XOff) * XScale + PixOffX
```

```
    ly2 = (PlotY(n) + YOff) * YScale + PixOffY
    frmPointCollector Line (lx1, ly1)-(lx2, ly2)
  Next n
End Sub

Public Sub PlotStoredPerimeter()
  frmPointCollector DrawWidth = 1
  frmPointCollector ForeColor = vbWhite
  For n = 1 To PointCount
    lx1 = (PerX(n - 1) + XOff) * XScale + PixOffX
    ly1 = (PerY(n - 1) + YOff) * YScale + PixOffY
    lx2 = (PerX(n) + XOff) * XScale + PixOffX
    ly2 = (PerY(n) + YOff) * YScale + PixOffY
    frmPointCollector Line (lx1, ly1)-(lx2, ly2)
  Next n
End Sub

Public Sub AutoScale()
  XMaxPer = Abs(XMax - XMin)
  YMaxPer = Abs(YMax - YMin)
  XScale = 300 / XMaxPer '300 Pixels
  YScale = 140 / YMaxPer '150 Pixels
  XOff = -XMin
  YOff = -YMin
End Sub

Public Sub ReferencePoint()
  Call ReadNmea
  DegRefLon = Val(Mid$(xs, 1, 3)) + Val(Mid$(xs, 4,
        ((Len(xs) - 3)))) / 60
  DegRefLat = Val(Mid$(ys, 1, 2)) + Val(Mid$(ys, 3,
        ((Len(ys) - 2)))) / 60
End Sub

Public Sub ReDraw()
  If XMaxPer > 0 Then
    If YMaxPer > 0 Then
      If CollectingPerimeter = False Then
        Call PlotPerimeter
      End If
    End If
  End If
End Sub
```

Graphical Module

'frmMenu

```
Private Sub cmdQuit_Click()
  End
End Sub

Private Sub cmdSettings_Click()
  txtSettingsError Visible = False
  fraSettings Visible = True
  fraMenu Visible = False
End Sub

Private Sub cmdSettingsBack_Click()
  fraMenu Visible = True
  fraSettings Visible = False
  frmPOINT MSComm1 Settings =
        frmMenu cboGPSBaud Text + "n,8,1"
End Sub

Private Sub cmdSTART_Click()
  frmPointCollector Visible = True
  frmMenu Visible = False
  frmPointCollector fraExperimentName Visible = True
End Sub
```

'frmPoint

```
Private Sub tmrError_Timer()
 tmrError Enabled = False
 frmMenu fraSettings Enabled = True
 Exit Sub
End Sub
```

'frmPointCollector

```
Private Sub cmdBack_Click()
  fraMenu Visible = False
  frmMenu Visible = True
    frmMenu fraMenu Visible = True
End Sub

Private Sub cmdBackToMainMenuEC_Click()
  fraExistingCalibration Visible = False
  fraMenu Visible = True
  MuleAnt = txtAntennaToBoomDistance Text
  """Add in Sub to update VARS from text Boxes
End Sub

Private Sub cmdCalibrationResultsBackToMain_Click()
  fraCalibrationResults Visible = False
  fraMenu Visible = True
End Sub

Private Sub cmdExistingCalibration_Click()
  fraMenu Visible = False
  fraExistingCalibration Visible = True
End Sub

Private Sub cmdFinish_Click()
  fraStep15 Visible = False
  fraCalibrationResults Visible = True
End Sub

Private Sub cmdNewCalibration_Click()
  fraMenu Visible = False
  fraStep1 Visible = True
End Sub

Private Sub cmdStep10Back_Click()
  fraStep10 Visible = False
  fraStep9 Visible = True
End Sub

Private Sub cmdStep10Next_Click()
  fraStep10 Visible = False
  fraStep11 Visible = True
End Sub

Private Sub cmdStep11Back_Click()
  fraStep11 Visible = False
  fraStep10 Visible = True
End Sub
```

```
Private Sub cmdStep11Next_Click()
   fraStep11 Visible = False
   fraStep12 Visible = True
End Sub

Private Sub cmdStep12Back_Click()
   fraStep12 Visible = False
   fraStep11 Visible = True
End Sub

Private Sub cmdStep12Next_Click()
   fraStep12 Visible = False
   fraStep13 Visible = True
End Sub

Private Sub cmdstep13Back_Click()
   fraStep13 Visible = False
   fraStep12 Visible = True
End Sub

Private Sub cmdstep13Next_Click()
   fraStep13 Visible = False
   fraStep14 Visible = True
End Sub

Private Sub cmdStep14Back_Click()
   fraStep14 Visible = False
   fraStep13 Visible = True
End Sub

Private Sub cmdSTEP14Next_Click()
   fraStep14 Visible = False
   fraStep15 Visible = True
End Sub

Private Sub cmdStep15Back_Click()
   fraStep15 Visible = False
   fraStep14 Visible = True
End Sub

Private Sub cboExperimentName_Click(Index As
          Integer)
     cmdExperimentNameOK.Enabled = True
End Sub

Private Sub cmdDataCollectingBack_Click()
   fraDataCollecting Visible = False
   fraExperimentName Visible = True
End Sub

Private Sub cmdDataCollectingNext_Click()
   fraDataCollecting Visible = False
   fraStep1 Visible = True
   NumPoints = CInt(frmPointCollector cboNumPoints)
   frmPOINT MSComm1 Settings =
          frmMenu cboGPSBaud Text + "n,8,1"
End Sub

Private Sub cmdExperimentNameBack_Click()
   frmMenu Visible = True
End Sub

Private Sub cmdExperimentNameNext_Click()
   fraExperimentName Visible = False
 If frmPOINT MSComm1 PortOpen = False Then
  frmPOINT MSComm1 CommPort =
          frmMenu cboGPSPort Text
  End If
  frmPOINT MSComm1 Settings =
          frmMenu cboGPSBaud Text + ",n,8,1"
  fraDataCollecting Visible = True
   If frmPOINT MSComm1 PortOpen = False Then
  frmPOINT MSComm1 PortOpen = True
  End If
End Sub

Private Sub cmdExperimentNameOK_Click()
   Call ExperimentNameOK
   cmdExperimentNameOK Enabled = False
End Sub

Private Sub cmdPoint1_Click()
   PointNum = 1
   cmdPoint1 Enabled = False
   cmdStep6Next Enabled = False
   cmdStep6Back Enabled = False
   Call GETPTS
   Call AVGPTS
   cmdPoint1 Enabled = True
   cmdStep6Next Enabled = True
   cmdStep6Back Enabled = True
   frmPointCollector Cls
   Call PlotStoredPerimeter
   Call PlotAvgPoint1
   cmdStep6Next_Click
End Sub

Private Sub cmdPoint2_Click()
   PointNum = 2
   cmdPoint2 Enabled = False
   cmdStep8Next Enabled = False
   cmdStep8Back Enabled = False
   Call GETPTS
   Call AVGPTS
   cmdPoint2 Enabled = True
   cmdStep8Next Enabled = True
   cmdStep8Back Enabled = True
   frmPointCollector Cls
   cmdStep8Next_Click
   Call PlotStoredPerimeter
   Call PlotAvgPoint2
   Call PlotAvgPoint1
   cmdStep9Finish Enabled = True
End Sub

Private Sub cmdStartPerimeter_Click()
   CollectingPerimeter = True
   cmdStartPerimeter Enabled = False
   cmdStopPerimeter Enabled = True
   cmdStep2Back Enabled = False
   cmdStep4Next Enabled = False
   cmdStep2Next Enabled = True
   cmdStep2Next_Click
   Call Perimeter
End Sub

Private Sub cmdStep1Back_Click()
   fraExperimentName Visible = True
   fraStep1 Visible = False
End Sub

Private Sub cmdStep1Next_Click()
   fraStep2 Visible = True
   fraStep1 Visible = False
```

```
End Sub

Private Sub cmdStep2Back_Click()
   fraStep1 Visible = True
   fraStep2 Visible = False
End Sub

Private Sub cmdStep2Next_Click()
   fraStep3 Visible = True
   fraStep2 Visible = False
   Call ReDraw
End Sub

Private Sub cmdStep3Back_Click()
   fraStep2 Visible = True
   fraStep3 Visible = False
   Call ReDraw
End Sub

Private Sub cmdStep3Next_Click()
   fraStep4 Visible = True
   fraStep3 Visible = False
   Call ReDraw
End Sub

Private Sub cmdStep4Back_Click()
  fraStep3 Visible = True
  fraStep4 Visible = False
  Call ReDraw
End Sub

Private Sub cmdStep4Next_Click()
   fraStep5 Visible = True
   fraStep4 Visible = False
   Tracking = True
   Call BufferFlush
   Call Track
End Sub

Private Sub cmdStep5Back_Click()
   fraStep4 Visible = True
   fraStep5 Visible = False
   Call ReDraw
End Sub

Private Sub cmdStep5Next_Click()
   fraStep6 Visible = True
   fraStep5 Visible = False
End Sub

Private Sub cmdStep6Back_Click()
   fraStep5 Visible = True
   fraStep6 Visible = False
End Sub

Private Sub cmdStep6Next_Click()
   fraStep7 Visible = True
   fraStep6 Visible = False
   Call Track
End Sub

Private Sub cmdStep7Back_Click()
   fraStep6 Visible = True
   fraStep7 Visible = False
End Sub

Private Sub cmdStep7Next_Click()
   fraStep8 Visible = True
   fraStep7 Visible = False
End Sub

Private Sub cmdStep8Back_Click()
   fraStep7 Visible = True
   fraStep8 Visible = False
End Sub

Private Sub cmdStep8Next_Click()
   fraStep9 Visible = True
   fraStep8 Visible = False
  frmPointCollector Cls
  Call PlotAvgPoint1
  Call PlotStoredPerimeter
  Call PlotAvgPoint2
  Call MakeFile
End Sub

Private Sub cmdStep9Back_Click()
   fraStep8 Visible = True
   fraStep9 Visible = False
End Sub

Private Sub cmdStep9Next_Click()
  fraStep10 Visible = True
  fraStep9 Visible = False
   Tracking = False
End Sub

Private Sub cmdStep9Finish_Click()
   End
End Sub

Private Sub cmdStopPerimeter_Click()
   CollectingPerimeter = False
   Close #1
   cmdStopPerimeter Enabled = False
   cmdStartPerimeter Enabled = True
   cmdStep2Back Enabled = True
   cmdStep4Next Enabled = True
   cmdStep4Next_Click
End Sub

Private Sub Form_Load()
fraStep1 Top = 0
fraStep1 Left = 0
fraStep2 Top = 0
fraStep2 Left = 0
fraStep3 Top = 0
fraStep3 Left = 0
fraStep4 Top = 0
fraStep4 Left = 0
fraStep5 Top = 0
fraStep5 Left = 0
fraStep6 Top = 0
fraStep6 Left = 0
fraStep7 Top = 0
fraStep7 Left = 0
fraStep8 Top = 0
fraStep8 Left = 0
fraStep9 Top = 0
fraStep9 Left = 0
fraExperimentName Top = 0
fraExperimentName Left = 0
fraDataCollecting Top = 0
fraDataCollecting Left = 0
```

```
End Sub

Private Sub tmrPoints_Timer()
   Collecting = False
End Sub
```

APPENDIX D

FIELD SYSTEM SOFTWARE CODE

Main Code Module

'SprayChoice.bas

'Variable Declarations
```
Option Explicit
Public n As Integer
Public CpDist As Double
Public CpX(0 To 10000) As Single
Public CpY(0 To 10000) As Single
Public MinCpDist As Double
Public T As Integer
Public Diagonal As Single
Public BuffUse As Integer
Public fGPS As Integer
Public dt As Double
Public RateSprayed As Integer
Public XrBuf(0 To 10) As Double
Public YrBuf(0 To 10) As Double
Public XBuf(0 To 10) As Double
Public YBuf(0 To 10) As Double
Public NumRows As Integer
Public NumColumns As Integer
Public BorderRate As Integer
Public dx As Double
Public Rate(0 To 10000) As Integer
Public ind As Integer
Public dydt As Double
Public BuffPos As Integer
Public Multiplier(0 To 10) As Double
Public SumBuff As Double
Public NumPlots As Integer
Public PlotWidth As Integer
Public PlotLength As Integer
Public YMaxPer As Double
Public XMaxPer As Double
Public XRes As Single
Public YRes As Single
Public YScale As Single
Public XScale As Single
Dim LETC As Integer
Dim NIBC As Integer
Public LongEnoughInTarget(0 To 10000) As Boolean
Public OutofTarget(0 To 10000) As Boolean
Public NotInBorder(0 To 10000) As Boolean
Public SpentTargetBool(0 To 10000) As Boolean
Public SpentTarget(0 To 10000) As Integer
Public TopPlot(0 To 1000)  As Boolean
Public BottomPlot(0 To 1000) As Boolean
Public LeftPlot(0 To 1000)  As Boolean
Public RightPlot(0 To 1000) As Boolean
Public DirChange As Integer
Public FirstQuadX As Double
Public FirstQuadY As Double
Public FQ As Integer
Public PlotNumber(0 To 10000) As Integer
Public PlotCount As Integer
Public XOffSet As Single
Public YOffset As Single
Dim ww(0 To 100) As Integer
Public DegRefLon As Double
Public DegRefLat As Double
Public Theta As Double
Dim Trash As String
Dim tt As String
Dim xs As String
Dim SVNums As String
Dim Qualitys As String
Dim ys As String
Dim HDOPs As String
Dim X As Double
Public Time As Double
Dim Y As Double
Public xr As Double
Public yr As Double
Public Quality As Integer
Public SVNum As Integer
Public LonToFt As Double
Public LatToFt As Double
Public CosTheta As Double
Public SinTheta As Double
Const pi = 3 14159265358979
Public HDOPBuf(0 To 9) As Single
Public XUsed As Double
Public YUsed As Double
Public dydtUsed As Double
Public dxdtUsed As Double
Public FeedForwardCount As Integer
Public RSBuf(0 To 9) As Single
Public TBuf(0 To 50) As Integer
Public MinCpBuf(0 To 50) As Double
Public InBorder(0 To 50) As Boolean
Public TotalXoff As Double
Public TotalYoff As Double
Public xoff As Double
Dim ETA As Single
Dim Drift As Double
Public MinTimeInPlot As Integer
Dim lx1 As Integer
Dim ly1 As Integer
Dim lx2 As Integer
Dim ly2 As Integer
Public strDrive As String
Public strFile As String
Public strExperiment As String
Dim StrC As String
Dim Length1  As Integer
Dim NFields As Integer
Dim i As Integer
Dim StrD As String
Dim p1 As Integer
Dim Length2  As Integer
Dim ExpName As String
Public BAFC As Integer
Public AccFlagBuf(0 To 9) As String
Public dxdt As Double
Public Speed As Double
Public dxdtBuf(0 To 9) As Double
Public dydtBuf(0 To 9) As Double
Public SpeedBuf(0 To 9) As Single
Public MaxSpeed As Single
Public PauseRate As Integer
Public Pause As Boolean
Public MuleAnt As Double      'Distance Between Antenna
        and Sprayer Nozzle
Public MuleAntX As Double
Public MuleAntY As Double
Public TCount As Integer
Public PixOffX As Integer
Public PixOffY As Integer
Public strRefLat As String
```

```
Public strRefLon As String
Public LHDOP As String
Public LSatNum As Integer
Public nnn As Integer
Public mp As String
Public mpu As String
Public gps As String
Public HDOP As Byte
Public dy As Double
Public LED As Single
Public qwerty As Integer
Public ee As Integer
Public zz As Integer
Public px(1 To 1000) As Single
Public py(1 To 1000) As Single
Public NoRotBx(1 To 1000) As Single
Public NoRotBy(1 To 1000) As Single
Public latp As Double
Public lonp As Double
Public FoScaleX As Single
Public FoScaleY As Single
Public aa As Integer
Public FoffX As Single
Public FoffY As Single
Public MaxD As Double
Public FoCount As Long
Public fozoom As Integer
Public Aang As Double
Public zin As Boolean
Public FoPixOffX As Integer
Public FoPixOffY As Integer
Public Mult As Integer
Public pxr(0 To 1000) As Single
Public pyr(0 To 1000) As Single
'Drawing Declarations
Public SinAngle As Double
Public CosAngle As Double
Public BothScale As Single
Public ex(1 To 10) As Single
Public ey(1 To 10) As Single
Public tpx(1 To 10) As Single
Public tpy(1 To 10) As Single
Public OutlineWidth As Single
Public Resultant As Double
Public Jumps As Integer
Dim PerDiffX As Single
Dim PerDiffY As Single
Dim m As Integer
Public SprayerComPort As Integer
Public GPSComPort As Integer
Public ApplicatorModel As Integer
Public ScreenResX As Integer
Public ScreenResY As Integer
Public MaxGPSError As Single
Public AccWarnTrigger As Integer
Public StopExpTrigger As Integer
Public HDOPTrigger As Integer
Public LightBarSensitivity As Integer
Public DataSmoothingSensitivity As Integer
Public GPSBaudRate As String
Public GPSStopBits As String
Public GPSDataBits As String
Public GPSPort As String
Public GPSUpdateRate As Integer
Public LBarBaudRate As String
Public LBarStopBits As String
Public LBarDataBits As String
Public LBarPort As String
Public ApplrBaudRate As String
Public ApplrStopBits As String
Public ApplrDataBits As String
Public ApplrPort As String
Public Horizontal As Integer
Public Vertical As Integer
Public PCCardDrive As String
Public OriginalRateList(0 To 12) As Integer
Public RateList(0 To 12) As Integer
Public AccFlag As String
Public BadHDOPCount As Integer
Public NumRates As Integer
Public File$
Public Temp1 As String
Public Temp2 As String
Public ExpCount As Integer
Public Exps(1 To 1000) As String
Public Dirs(1 To 1000) As String
Public Directory As String
Public FileName As String
Public DirLength As Integer
Public OutputFile As String
Public RateNumber(0 To 12)
Public GPSUpdateCount As Single
Public OldDirChange As Integer
Public pxrMin As Single
Public pxrMax As Single
Public pyrMin As Single
Public pyrMax As Single
Public pxMin As Single
Public pxMax As Single
Public pyMin As Single
Public pyMax As Single
Public ResMaxPer As Single
Public XUsed1 As Double
Public YUsed1 As Double
Public CosAngle1 As Double
Public SinAngle1 As Double
Public PrintRates As Boolean
Public NumVectors As Integer
Public VpX(0 To 1000) As Single
Public V As Integer
Public FF As Boolean
Public dxSum As Single
Public dySum As Single
Public dxAvg(1 To 4) As Single
Public dyAvg(1 To 4) As Single
Public CalSpeed(1 To 100) As Single
Public PCount As Integer
Public Xes(1 To 500) As Single
Public Yes(1 To 500) As Single
Public Error1 As Double
Public Error2 As Double
Public Velocity1 As Double
Public Velocity2 As Double
Public CalCoeff As Double
Public AcrossLine As Boolean
Public XAvg(1 To 2) As Single
Public YAvg(1 To 2) As Single
Public XSum As Single
Public YSum As Single
Public NumPoints As Integer
Public PointNum As Integer
Public dxc As Single
Public dyc As Single
Public Calibrating As Boolean
```

```
Public SinThetaCal As Double
Public CosThetaCal As Double
Public DegRefLatCal As Double
Public DegRefLonCal As Double
Public sc As Integer
Public f As String
Public FFControl As Boolean
Public ETATimer As Boolean
Public SentRate As Single
Public RecoveryFile As String
Public STBOld As Boolean
Public CalCount As Integer
Public PlotInQuestion As Integer
Public LoopCount As Double
Public DoEventsMod As Double
Public ScreenFlip As Byte
Public dydtUsed1
Public DrawPeriod As Single
Public DrawMod As Single
Public Sprayer As Byte
Public DotX1 As Single
Public DotY1 As Single
Public STDrawPlot As Integer
Public DrawChange As Integer
Public OldDrawChange As Integer
Public lbx As Single
Public lmx As Single
Public lby As Single
Public lmy As Single
Public Nlbx As Single
Public Nlmx As Single
Public Nlby As Single
Public Nlmy As Single
Public plo2 As Single
Public pwo2 As Single
Public OldVx1 As Integer
Public OldVy1 As Integer
Public OldVx2 As Integer
Public OldVy2 As Integer
Public Vx1 As Integer
Public Vy1 As Integer
Public Vx2 As Integer
Public Vy2 As Integer
Public dcpx(0 To 10000) As Integer
Public dcpy(0 To 10000) As Integer
Public dOutx1(0 To 10000) As Integer
Public dOuty1(0 To 10000) As Integer
Public dOutx2(0 To 10000) As Integer
Public dOuty2(0 To 10000) As Integer
Public Ndcpx(0 To 10000) As Integer
Public Ndcpy(0 To 10000) As Integer
Public NdOutx1(0 To 10000) As Integer
Public NdOuty1(0 To 10000) As Integer
Public NdOutx2(0 To 10000) As Integer
Public NdOuty2(0 To 10000) As Integer
Public dpx1(0 To 1000) As Integer
Public dpy1(0 To 1000) As Integer
Public dpx2(0 To 1000) As Integer
Public dpy2(0 To 1000) As Integer
Public ndpx1(0 To 1000) As Integer
Public ndpy1(0 To 1000) As Integer
Public ndpx2(0 To 1000) As Integer
Public ndpy2(0 To 1000) As Integer
Public DrawVector As Integer
Public OldDrawVector As Integer
Public FQM As Integer
Public DrawChangeM
Public FirstQuadXM As Integer
Public FirstQuadYM As Integer
Public DrawCount As Long
Public mm As Integer
Public rr As Integer
Public ss As Integer
Public yy As Integer
Public Neighbors(1 To 10, 1 To 9) As Integer 'Not Used
Public Increment As Single
Public AbsDrift As Double
Public Step(0 To 7) As Single
Public LookPeriod As Single
Public Method As Byte
Public MinSpeed As Single
Public LineThickness As Byte
Public SprayerSize As Byte
Public dxdtUsed1 As Double

Public Sub MainProgram()
  Call OpenUp
   DrawCount = 0
  If Calibrating = True Then
    strExperiment = PCCardDrive + " \Calibr\Calibr exp"
  Else  strExperiment = PCCardDrive + " \" + Directory +
        "\" + FileName
  End If
  Call ReadMSFile
  Call Initial
  'Call NineNeighbors
  If Calibrating = True Then
    Call LoadCalPerimeter
    Call CalibrationSetup
    FFControl = False
  Else  Call LoadPerimeter
  End If
  Call BufferFill
  'The Following 3 Sub Routines Scale the screen based on
        Perimeter DIA
  Call FindMaxPer
  Call OffSets
  Call ScaleXY
  Call ScaleNoRot
  If frmMenu cboRecovery Text = "RECOVERY" Then
    Call ReadRecoveryFile
  End If
  If Calibrating = False Then
    OutputFile = PCCardDrive + " \" + Directory + "\" +
        Directory + " map"
    RecoveryFile = PCCardDrive + " \" + Directory + "\" +
        Directory + " rec"
   If frmMenu cboRecovery Text = "RECOVERY" Then
      Open OutputFile For Append As #2
      Open RecoveryFile For Append As #4
    Else  Open OutputFile For Output As #2
      Open RecoveryFile For Output As #4
    End If
  End If
  frmComm MSComm2 Settings = GPSBaudRate + "," +
        "n" + "," + GPSDataBits + "," + GPSStopBits
  Do Until Pause = True
    Call RateChoice4 'Main dog
  Loop
End Sub

Public Sub RateChoice4() "This is the GPS Update Loop
  Call DecifferAll
  FoCount = FoCount + 1
```

```
    DrawMod = FoCount Mod (fGPS * DrawPeriod)
    Call Navigation
    dydtUsed1 = dydtUsed
    XUsed1 = XUsed
    YUsed1 = YUsed
    If FFControl = True Then
      Call FeedForward
    Else  Call PositionOK
    End If
    If FoCount <= 2 Then
      Call DirChanger "this must be called to initialize
            offsests
      DrawChange = DirChange
    End If
    Call Alarm
    'If FoCount = 1 Or AccFlag = "FF" Then
      Call TFinder
    'Else  Call NeighborTFinder
    'End If
    Call TBuffer
    If frmSettings cboUsedTargetTracking Text = "ON" Then
      If PauseRate = 1 Then
        Call TInfo
      End If
    End If
    Call SitStill 'Sub SitStill Contains DirChanger
    If ETATimer = True Then
      Call ETACalc 'This is the timer sub
    Else  Call NoTimer
    End If
    If DrawMod = 0 Then
      DrawCount = DrawCount + 1
      Call UpdateTable
      OldDrawVector = DrawVector
      If CpX(TBuf(0)) <> CpX(TBuf(1)) Then DrawVector =
            T
      If DrawCount = 1 Then
        Call DrawingCps
        Call DrawAll
      ElseIf zin = False Then
        Call FarOff
      ElseIf zin = True Then
        Call DrawChoice
      Else  Call FarOff
      End If
    End If
    Call LightBarLookAhead
    If Calibrating = False Then Call Store
End Sub

Public Sub LEDRed1()
  If DirChange * Drift < 0 Then
    LED = 14
  Else  LED = 6
  End If
End Sub

Public Sub LEDRed2()
  If DirChange * Drift < 0 Then
    LED = 13
  Else  LED = 5
  End If
End Sub

Public Sub LEDRed3()
  If DirChange * Drift < 0 Then
    LED = 12
  Else  LED = 4
  End If
End Sub

Public Sub LEDRed4()
  If DirChange * Drift < 0 Then
    LED = 11
  Else  LED = 3
  End If
End Sub

Public Sub LEDRed5()
  If DirChange * Drift < 0 Then
    LED = 10
  Else  LED = 2
  End If
End Sub

Public Sub LEDRed6()
  If DirChange * Drift < 0 Then
    LED = 9
  Else  LED = 1
  End If
End Sub

Public Sub LEDRed7()
  If DirChange * Drift < 0 Then
    LED = 8
  Else  LED = 0
  End If
End Sub

Public Sub LEDGreen()
  LED = 7
End Sub

Public Sub LightBarLookAhead()
  Call LightBarDrift
  Call LEDs
  frmComm MSComm1 Output = Str$(LED) + Chr$(13)
End Sub

Public Sub LEDs()
  If AbsDrift <= Step(0) Then
    Call LEDGreen
  ElseIf AbsDrift <= Step(1) Then
    Call LEDRed1
  ElseIf AbsDrift <= Step(2) Then
    Call LEDRed2
  ElseIf AbsDrift <= Step(3) Then
    Call LEDRed3
  ElseIf AbsDrift <= Step(4) Then
    Call LEDRed4
  ElseIf AbsDrift <= Step(5) Then
    Call LEDRed5
  ElseIf AbsDrift <= Step(6) Then
    Call LEDRed6
  Else  Call LEDRed7
  End If
End Sub

Public Sub StepSize()
  For n = 0 To 7
    Step(n) = (n + 1) * Increment
  Next n
End Sub
```

```
Public Sub LightBarDrift()
  If Method = 2 Then
    Drift = XUsed - CpX(T) + (dxdtUsed) * LookPeriod
  Else  Drift = XUsed - CpX(T) + LookPeriod * (dxdtUsed
        + ((dxdtUsed - dxdtUsed1) / fGPS * LookPeriod))
  End If
  AbsDrift = Abs(Drift)
End Sub

Public Sub VelocityRejector()
  If Speed < MinSpeed Then
    Speed = 0
    dydt = 0
    dxdt = 0
    CosAngle = CosAngle1
    SinAngle = SinAngle1
  ElseIf dxdt < MinSpeed Then
    dxdt = 0
    Speed = Abs(dydt)
  End If
End Sub

Public Sub FeedForward()
  dxdtUsed1 = dxdtUsed
  FF = False
  If FeedForwardCount < StopExpTrigger Then
    If InBorder(0) = False Then
     If SentRate <> 0 Then
       If Rate(T) <> 0 Then
      If Abs(dxdtBuf(0)) > MaxGPSError Then
        FeedForwardCount = FeedForwardCount + 1
        dxdtUsed = 0
        XUsed = XUsed1
        FF = True
          dydtUsed = dydtUsed1
        YUsed = YUsed1 + dydtUsed / fGPS
        AccFlag = "FF"
      Else  Call PositionOK
      End If
     Else  Call PositionOK
     End If
     Else  Call PositionOK
     End If
    Else  Call PositionOK
    End If
  Else  Call PositionOK
        Call BufferFlush
        Call BufferFill
  End If
End Sub

Public Sub PositionOK()
  dydtUsed = dydtBuf(0)
  dxdtUsed = dxdtBuf(0)
  XUsed = XrBuf(0)
  YUsed = YrBuf(0)
  FeedForwardCount = 0
  AccFlag = "OK"
End Sub

Public Sub FindExperimentFiles()
  On Error GoTo ehandle
  Call EraseFileNames
  m = 0
  Temp1 = Dir(PCCardDrive + " \", vbDirectory)
    Do Until Temp1 = ""
      m = m + 1
        Dirs(m) = Temp1
      Temp1 = Dir
    Loop
    ExpCount = 0
    For n = 1 To m
      Temp2 = Dir(PCCardDrive + " \" + Dirs(n) +
        "\* exp")
      If InStr(Temp2, " exp") Then
        ExpCount = ExpCount + 1
        Exps(ExpCount) = Temp2
      End If
    Next n
'If n = "it won't ever equal this" Then
If n = "-45" Then
ehandle
  frmSettings Visible = True
  frmSettings txtPCCardError Visible = True
  frmSettings fraHardWareSpecifications Visible = True
  frmSettings fraHardWareSpecifications Enabled = False
  frmMenu Visible = False
  frmMenu fraCalibrationWarning Visible = False
  frmComm tmrError Interval = 1500
  frmComm tmrError Enabled = True
End If
End Sub

Public Sub ListFileNames()
  frmMenu cboExperimentFiles Text = Exps(1)
  For n = 1 To ExpCount
    frmMenu cboExperimentFiles AddItem Exps(n)
  Next n
End Sub

Public Sub ExperimentNext()
'"This sub starts an experiment
  FileName = frmMenu cboExperimentFiles Text
  DirLength = Len(FileName) - 4
  Directory = Mid$(FileName, 1, DirLength)
  frmGrid Visible = True
  frmSettings cboPCCardDrive Enabled = False
  Call MainProgram
End Sub

Public Sub AutoZoom()
  XScale = XRes / XMaxPer
  YScale = YRes / YMaxPer
  If zin = True Then
    PixOffX = 3
    PixOffY = 3
    Call DrawingCps
    Call DrawAll
  Else
    FoPixOffX = 3
    FoPixOffY = 3
    FoScaleX = XRes / ResMaxPer
    FoScaleY = YRes / ResMaxPer
    frmGrid Cls
    Call FarOff
  End If
End Sub

Public Sub FarOff()
  frmGrid ForeColor = vbWhite
  frmGrid DrawWidth = 10
  lx1 = (0 + FoffX) * FoScaleX + FoPixOffX
  ly1 = (0 + FoffY) * FoScaleY + FoPixOffY
  frmGrid PSet (lx1, ly1), vbWhite
```

```
    frmGrid DrawWidth = 1
    frmGrid DrawStyle = 0
    For aa = 1 To zz - 1
      lx1 = (NoRotBx(aa) + FoffX) * FoScaleX + FoPixOffX
      ly1 = (NoRotBy(aa) + FoffY) * FoScaleY + FoPixOffY
      lx2 = (NoRotBx(aa + 1) + FoffX) * FoScaleX +
            FoPixOffX
      ly2 = (NoRotBy(aa + 1) + FoffY) * FoScaleY +
            FoPixOffY
      frmGrid Line (lx1, ly1)-(lx2, ly2)
    Next aa
    lx1 = (XBuf(1) + FoffX) * FoScaleX + FoPixOffX
    ly1 = (YBuf(1) + FoffY) * FoScaleY + FoPixOffY
    lx2 = (XBuf(0) + FoffX) * FoScaleX + FoPixOffX
    ly2 = (YBuf(0) + FoffY) * FoScaleY + FoPixOffY
    frmGrid Line (lx1, ly1)-(lx2, ly2)
End Sub

Public Sub FarOffSet()
  MaxD = (XMaxPer ^ 2 + YMaxPer ^ 2) ^ 0 5
  If XBuf(0) > 0 Then
    FoffX = fozoom + MaxD
  Else  FoffX = -fozoom - MaxD
  End If
  If YBuf(0) > 0 Then
    FoffY = fozoom + MaxD
  Else  FoffY = -fozoom - MaxD
  End If
End Sub

Public Sub FarOffScales()
  FoScaleX = XRes / XMaxPer 'FoffX
  FoScaleY = YRes / YMaxPer 'FoffY
End Sub

Public Sub ScreenResolution()
  XRes = Horizontal - frmGrid fraRatesAndMenu Width - 5
  YRes = Vertical - frmGrid fraZoom Height - 5
  Call ScaleXY
End Sub

Public Sub ReadRateAssignments()
  For PlotCount = 1 To NumPlots
  Line Input #1, StrC
  Length1 = Len(StrC)
   NFields = 0
   For i = 1 To Length1
     StrD = Mid$(StrC, i, 1)
     If StrD = Chr$(9) Then
       NFields = NFields + 1
       ww(NFields) = i
     End If
   Next i
   p1 = ww(0) + 1
   Length2 = ww(1) - ww(0) - 1
   PlotNumber(PlotCount) = Val(Mid$(StrC, p1, Length2))
   p1 = ww(1) + 1
   Length2 = ww(2) - ww(1) - 1
   CpX(PlotCount) = Val(Mid$(StrC, p1, Length2))
   p1 = ww(2) + 1
   Length2 = ww(3) - ww(2) - 1
   CpY(PlotCount) = Val(Mid$(StrC, p1, Length2))
   p1 = ww(3) + 1
   Length2 = ww(4) - ww(3) - 1
   Rate(PlotCount) = Val(Mid$(StrC, p1, Length2))
  Next PlotCount
End Sub

Public Sub EraseFileNames()
  If ExpCount >= 1 Then
    For n = 1 To ExpCount
      frmMenu cboExperimentFiles RemoveItem (0)
    Next n
  End If
  ExpCount = 0
End Sub

Public Sub ReadHeader()
 Line Input #1, StrC
    Length1 = Len(StrC)
    NFields = 0
  For i = 1 To Length1
    StrD = Mid$(StrC, i, 1)
    If StrD = Chr$(9) Then
      NFields = NFields + 1
      ww(NFields) = i
    End If
  Next i
  p1 = ww(0) + 1
  Length2 = ww(1) - ww(0) - 1
  ExpName = Mid$(StrC, p1, Length2)
  p1 = ww(1) + 1
  Length2 = ww(2) - ww(1) - 1
  strRefLat = Mid$(StrC, p1, Length2)
  DegRefLat = Val(strRefLat)
  p1 = ww(2) + 1
  Length2 = ww(3) - ww(2) - 1
  strRefLon = Mid$(StrC, p1, Length2)
  DegRefLon = Val(strRefLon)
  p1 = ww(3) + 1
  Length2 = ww(4) - ww(3) - 1
  PlotLength = Val(Mid$(StrC, p1, Length2))
  p1 = ww(4) + 1
  Length2 = ww(5) - ww(4) - 1
  PlotWidth = Val(Mid$(StrC, p1, Length2))
  p1 = ww(5) + 1
  Length2 = ww(6) - ww(5) - 1
  NumRates = Val(Mid$(StrC, p1, Length2))
  p1 = ww(6) + 1
  Length2 = ww(7) - ww(6) - 1
  NumPlots = Val(Mid$(StrC, p1, Length2))
  p1 = ww(7) + 1
  Length2 = ww(8) - ww(7) - 1
  CosTheta = Val(Mid$(StrC, p1, Length2))
  p1 = ww(8) + 1
  Length2 = ww(9) - ww(8) - 1
  SinTheta = Val(Mid$(StrC, p1, Length2))
  p1 = ww(9) + 1
  Length2 = ww(10) - ww(9) - 1
  NumVectors = Val(Mid$(StrC, p1, Length2))
End Sub

Public Sub ReadMSFile()
  On Error GoTo ehandle
  Open strExperiment For Input As #1
  Call ReadHeader
  Call ReadOriginalRates
  Call ReadRateAssignments
  Close #1
  If n = "a" Then
ehandle
End If
End Sub
```

```
Public Sub ReadOriginalRates()
For n = 0 To NumRates
   Line Input #1, StrC
   Length1 = Len(StrC)
   NFields = 0
   For i = 1 To Length1
      StrD = Mid$(StrC, i, 1)
      If StrD = Chr$(9) Then
         NFields = NFields + 1
         ww(NFields) = i
      End If
   Next i
   p1 = ww(0) + 1
   Length2 = ww(1) - ww(0) - 1
   RateNumber(n) = Val(Mid$(StrC, p1, Length2))
   p1 = ww(1) + 1
   Length2 = ww(2) - ww(1) - 1
   OriginalRateList(n) = Val(Mid$(StrC, p1, Length2))
   RateList(n) = OriginalRateList(n)
  Next n
 Call RatesDefaults
End Sub

Public Sub LoadPerimeter()
   FileName = Directory + " per"
   strExperiment = PCCardDrive + " \" + Directory + "\" +
         FileName
   zz = 0
   Open strExperiment For Input As #1
   Do While Not EOF(1) ' Check for end of file
      Line Input #1, StrC
      zz = zz + 1
      Length1 = Len(StrC)
      NFields = 0
   For i = 1 To Length1
      StrD = Mid$(StrC, i, 1)
      If StrD = Chr$(9) Then
         NFields = NFields + 1
         ww(NFields) = i
      End If
   Next i
   qwerty = ww(0) + 1
   Length2 = ww(1) - ww(0) - 1
   lonp = Mid$(StrC, qwerty, Length2)
   px(zz) = (Val(lonp) - DegRefLon) * LonToFt
   NoRotBx(zz) = px(zz)
   qwerty = ww(1) + 1
   Length2 = ww(2) - ww(1) - 1
   latp = Mid$(StrC, qwerty, Length2)
   py(zz) = (Val(latp) - DegRefLat) * LatToFt
   NoRotBy(zz) = py(zz)
   pxr(zz) = (px(zz) * CosTheta + py(zz) * SinTheta) + xoff
   pyr(zz) = (py(zz) * CosTheta - px(zz) * SinTheta) +
         MuleAnt
Loop
Close #1
End Sub

Public Sub FindMaxPer()
   XMaxPer = 0
   YMaxPer = 0
   For m = 1 To zz
      For n = 1 To zz
         PerDiffX = Abs(pxr(m) - pxr(n))
         PerDiffY = Abs(pyr(m) - pyr(n))
         If PerDiffX > XMaxPer Then
            XMaxPer = PerDiffX
         End If
         If PerDiffY > YMaxPer Then
            YMaxPer = PerDiffY
         End If
      Next n
   Next m
   ResMaxPer = (XMaxPer ^ 2 + YMaxPer ^ 2) ^ 0 5
End Sub

Public Sub OffSets()
   pxrMin = 1000000
   pyrMin = pxrMin
   pxrMax = -1000000
   pyrMax = pxrMax
   pxMin = 1000000
   pyMin = pxMin
   pxMax = -1000000
   pyMax = pxMax
   For n = 1 To zz          ""Must Call this sub after
         FindMaxPer
      If pxr(n) < pxrMin Then
         pxrMin = pxr(n)
      End If
      If pyr(n) < pyrMin Then
         pyrMin = pyr(n)
      End If
      If pxr(n) > pxrMax Then
         pxrMax = pxr(n)
      End If
      If pyr(n) > pyrMax Then
         pyrMax = pyr(n)
      End If
         If px(n) < pxMin Then
         pxMin = px(n)
      End If
      If py(n) < pyMin Then
         pyMin = py(n)
      End If
      If pxr(n) > pxrMax Then
         pxMax = px(n)
      End If
      If pyr(n) > pyrMax Then
         pyMax = py(n)
      End If
   Next n
   pxrMin = Abs(pxrMin)
   pxrMax = Abs(pxrMax)
   pyrMin = Abs(pyrMin)
   pyrMax = Abs(pyrMax)
   pxMin = Abs(pxMin)  'Perimeter Coords without
         Rotation
   pxMax = Abs(pxMax)
   pyMin = Abs(pyMin)
   pyMax = Abs(pyMax)
   FoffX = pxMin
   FoffY = pyMin
End Sub

Public Sub Multipliers()
   SumBuff = 0
   For BuffPos = 0 To (BuffUse - 1)
      SumBuff = SumBuff + BuffPos / BuffUse
   Next BuffPos
      SumBuff = SumBuff + 1
   For BuffPos = 0 To (BuffUse - 1)
      Multiplier(BuffPos) = (1 - BuffPos / BuffUse) /
            SumBuff
```

```
    Next BuffPos
End Sub

Public Sub DrawingCps()
    FQM = 1
    DrawChangeM = -1
    FirstQuadXM = pxrMax
    FirstQuadYM = pyrMax
    Call DrawCoefficients
    FQM = 1
    DrawChangeM = 1
    FirstQuadXM = pxrMin
    FirstQuadYM = pyrMin
    Call DrawCoefficients
  For n = 1 To NumPlots
    dcpx(n) = CpX(n) * lmx + lbx
    dcpy(n) = CpY(n) * lmy + lby
    dOutx1(n) = (CpX(n) - pwo2) * lmx + lbx
    dOuty1(n) = (CpY(n) - plo2) * lmy + lby
    dOutx2(n) = (CpX(n) + pwo2) * lmx + lbx
    dOuty2(n) = (CpY(n) + plo2) * lmy + lby

    Ndcpx(n) = CpX(n) * Nlmx + Nlbx
    Ndcpy(n) = CpY(n) * Nlmy + Nlby
    NdOutx1(n) = (CpX(n) - pwo2) * Nlmx + Nlbx
    NdOuty1(n) = (CpY(n) - plo2) * Nlmy + Nlby
    NdOutx2(n) = (CpX(n) + pwo2) * Nlmx + Nlbx
    NdOuty2(n) = (CpY(n) + plo2) * Nlmy + Nlby
  Next n
  For mm = 1 To zz - 1
    dpx1(mm) = pxr(mm) * lmx + lbx
    dpy1(mm) = pyr(mm) * lmy + lby
    dpx2(mm) = pxr(mm + 1) * lmx + lbx
    dpy2(mm) = pyr(mm + 1) * lmy + lby

    ndpx1(mm) = pxr(mm) * Nlmx + Nlbx
    ndpy1(mm) = pyr(mm) * Nlmy + Nlby
    ndpx2(mm) = pxr(mm + 1) * Nlmx + Nlbx
    ndpy2(mm) = pyr(mm + 1) * Nlmy + Nlby
  Next mm
End Sub

Public Sub DrawOutline()
  For n = 1 To NumPlots
   If Rate(n) <> 0 Then 'Don't draw border skeleton
    frmGrid DrawStyle = 0
    frmGrid DrawWidth = LineThickness
    frmGrid ForeColor = vbWhite
    If SpentTarget(n) = 0 Then
        frmGrid FillColor = QBColor(8) 'light gray
        frmGrid FillStyle = 7
    Else  frmGrid FillStyle = 1
    End If
    If DrawChange = -1 Then
      frmGrid Line (dOutx1(n), dOuty1(n))-(dOutx2(n),
          dOuty2(n)), vbWhite, B
    Else  frmGrid Line (NdOutx1(n), NdOuty1(n))-
          (NdOutx2(n), NdOuty2(n)), vbWhite, B
    End If
  End If
  DrawCp
  Next n
End Sub

Public Sub DrawCp()
  If LineThickness >= 2 Then
    frmGrid DrawWidth = 2
  Else  frmGrid DrawWidth = LineThickness '2
  End If
    If DrawChange = -1 Then
      lx1 = dcpx(n)
      ly1 = dcpy(n)
    Else  lx1 = Ndcpx(n)
        ly1 = Ndcpy(n)
    End If
    If PrintRates = True Then
      frmGrid CurrentX = lx1
      frmGrid CurrentY = ly1
      frmGrid ForeColor = vbWhite
     frmGrid Print Str$(RateList(Rate(n)))
    Else
        If SpentTarget(n) = 1 Then
          frmGrid PSet (lx1, ly1), vbWhite
        Else  frmGrid PSet (lx1, ly1), vbBlack
        End If
    End If
End Sub

Public Sub DrawCenterLine()
  frmGrid DrawStyle = 2
  frmGrid DrawWidth = LineThickness '1 'CenterLineWidth
   OldVx1 = Vx1
   OldVy1 = Vy1
   OldVx2 = Vx2
   OldVy2 = Vy2
   frmGrid ForeColor = vbBlack
   frmGrid Line (OldVx1, OldVy1)-(OldVx2, OldVy2)
   If DrawChange = -1 Then
     Vx1 = dcpx(DrawVector) 'CpX(T) * lmx + lbx
   Else  Vx1 = Ndcpx(DrawVector)
   End If
   Vy1 = 0
   Vx2 = Vx1
   Vy2 = 240
   frmGrid ForeColor = vbWhite
   frmGrid Line (Vx1, Vy1)-(Vx2, Vy2)
End Sub

Public Sub DrawSpentTargetEachTime()
   frmGrid DrawStyle = 0
   If SpentTarget(STDrawPlot) = 0 Then
      If Rate(STDrawPlot) <> 0 Then
       If DrawChange = -1 Then
          lx1 = dOutx1(STDrawPlot)
          ly1 = dOuty1(STDrawPlot)
          lx2 = dOutx2(STDrawPlot)
          ly2 = dOuty2(STDrawPlot)
       Else  lx1 = NdOutx1(STDrawPlot)
          ly1 = NdOuty1(STDrawPlot)
          lx2 = NdOutx2(STDrawPlot)
          ly2 = NdOuty2(STDrawPlot)
       End If
       frmGrid DrawWidth = LineThickness '2
           'OutlineWidth
       frmGrid FillColor = QBColor(8) 'light gray 'vbWhite
       frmGrid FillStyle = 7
       frmGrid Line (lx1, ly1)-(lx2, ly2), vbWhite, B
     End If
     If LineThickness >= 2 Then
       frmGrid DrawWidth = 2 'CpWidth
     Else  frmGrid DrawWidth = 1
     End If
     lx1 = CpX(STDrawPlot) * lmx + lbx
     ly1 = CpY(STDrawPlot) * lmy + lby
```

```
    frmGrid PSet (lx1, ly1), vbBlack 'vbBlack
    End If
End Sub

Public Sub DrawSprayer()
    frmGrid ForeColor = vbWhite

    frmGrid DrawWidth = SprayerSize
    ex(1) = (CosAngle * 7) * DrawChange
    ey(1) = (SinAngle * 7) * DrawChange
    If DrawChange = -1 Then
      DotX1 = XUsed * lmx + lbx
      DotY1 = YUsed * lmy + lby
    Else
      DotX1 = XUsed * Nlmx + Nlbx
      DotY1 = YUsed * Nlmy + Nlby
    End If
    On Error Resume Next
    frmGrid PSet (DotX1, DotY1), vbWhite

    lx2 = DotX1 + ex(1)
    ly2 = DotY1 + ey(1)
    frmGrid ForeColor = vbWhite
    frmGrid DrawWidth = 2 '/ 2
    frmGrid Line (DotX1, DotY1)-(lx2, ly2)
End Sub

Public Sub DrawCoverUp()
    On Error Resume Next
    frmGrid ForeColor = vbBlack
    frmGrid DrawWidth = SprayerSize
    frmGrid PSet (DotX1, DotY1), vbBlack
    lx2 = DotX1 + ex(1)
    ly2 = DotY1 + ey(1)
    frmGrid ForeColor = vbBlack
    frmGrid DrawWidth = 2 '/ 2
    frmGrid Line (DotX1, DotY1)-(lx2, ly2)
End Sub

Public Sub DrawAll()
  frmGrid Cls
   Call DrawPerimeter
   Call DrawCenterLine
   Call DrawOutline
   Call DrawSprayer
End Sub

Public Sub DrawEachTime()
   Call DrawCoverUp
   Call DrawSpentTargetEachTime
   If DrawVector <> OldDrawVector Then
     Call DrawCenterLine
   End If
   Call DrawSprayer
End Sub
Public Sub DrawChoice()
   If OldDrawChange <> DrawChange Then
     Call DrawAll
   Else
     Call DrawEachTime
   End If
End Sub

Public Sub DrawPerimeter()
If Calibrating = False Then
   frmGrid DrawStyle = 0
   frmGrid ForeColor = vbWhite
   frmGrid DrawWidth = LineThickness
   If DrawChange = -1 Then
     For rr = 1 To zz - 2 Step 2
       lx1 = dpx1(rr)
       ly1 = dpy1(rr)
       lx2 = dpx2(rr + 1)
       ly2 = dpy2(rr + 1)
     frmGrid Line (lx1, ly1)-(lx2, ly2)
     Next rr
   Else  For ee = 1 To zz - 2 Step 2
       lx1 = ndpx1(ee)
       ly1 = ndpy1(ee)
       lx2 = ndpx2(ee + 1)
       ly2 = ndpy2(ee + 1)
   frmGrid Line (lx1, ly1)-(lx2, ly2)
   Next ee
   End If
End If
End Sub

Public Sub DrawCoefficients()
   If DrawChangeM = -1 Then
     lmx = XScale * DrawChangeM
     lbx = XScale * (DrawChangeM * XOffSet + FQM *
           FirstQuadXM) + PixOffX
     lmy = YScale * DrawChangeM
     lby = YScale * (DrawChangeM * YOffset + FQM *
           FirstQuadYM) + PixOffY
   Else
     Nlmx = XScale * DrawChangeM
     Nlbx = XScale * (DrawChangeM * XOffSet + FQM *
           FirstQuadXM) + PixOffX
     Nlmy = YScale * DrawChangeM
     Nlby = YScale * (DrawChangeM * YOffset + FQM *
           FirstQuadYM) + PixOffY
   End If
End Sub

Public Sub ScaleXY()
   If XMaxPer <= 0 Then XMaxPer = 1000
   If YMaxPer <= 0 Then YMaxPer = 1000
   YScale = YRes / YMaxPer 'YRes is Resolution of Y
           Dimension YMaxPer is the maximum diff in
           perimeter points in Y Dir
   XScale = XRes / XMaxPer 'XRes is Resolution of x
           Dimension YMaxPer is the maximum diff in
           perimeter points in Y Dir
End Sub

Public Sub ScaleNoRot()
   FoScaleX = XRes / (ResMaxPer + 0 0000001) '1250  '25
           mi
   FoScaleY = YRes / (ResMaxPer + 0 0000001) ' 1250
End Sub

Public Sub Initial()
   'Initializes settings variables to what is in text boxes in
           Settings
    PrintRates = False
   fozoom = 1250 / 2   '1/4 Mile
   FoCount = 0
   PauseRate = 1
   Pause = False
   MinCpDist = 100000
   Diagonal = (PlotLength ^ 2 + PlotWidth ^ 2) ^ 0 5
   pwo2 = PlotWidth / 2
   plo2 = PlotLength / 2
```

```
    T = 1
    'Initializes OffSets
    PixOffX = 3
    PixOffY = 3
    FoPixOffX = 3
    FoPixOffY = 3
    XOffSet = 0 '100 '(ft)
    YOffset = 0 '100 '(ft)
    dydt = 0
    FirstQuadX = 0
    FirstQuadY = 0
    zin = True
    'Fills LongEnoughInTarget with zeroes (This will change
            to ones)
    'Fills OutofTarget with zeros (This will change to ones) to
            be used in NAND
    'Fills SpentTarget with ones (This is multiplied by the
            ratesprayed)
    For n = 1 To NumPlots
      LongEnoughInTarget(n) = False
      OutofTarget(n) = False
      NotInBorder(n) = False
      SpentTargetBool(n) = False
      SpentTarget(n) = 1
      TopPlot(n) = False
      BottomPlot(n) = False
      LeftPlot(n) = False
      RightPlot(n) = False
    Next n
    Call TopAndBottomPlots
    Call CalcConst
End Sub

Public Sub TopAndBottomPlots()
    On Error GoTo ehandle
    NumRows = NumPlots / NumVectors
    For n = 1 To NumVectors
      TopPlot(n * NumRows) = True
      BottomPlot(n * NumRows - NumRows + 1) = True
    Next n
    For n = 1 To NumRows
      LeftPlot(n) = True
      RightPlot((n + NumVectors * (NumRows - 1)) - 1) =
            True
    Next n
If n = "-45" Then
ehandle
    frmCalibrate fraStep1 Visible = True
    Exit Sub

End If
End Sub

Public Sub Delayy()
    For qwerty = 1 To 150000
    Next qwerty
End Sub

Public Sub Store()
  Print #2, Str$(Time) + Chr$(9) + xs + Chr$(9) + ys +
            Chr$(9), Str$(XUsed) + Chr$(9) + Str$(YUsed) +
            Chr$(9) + Str$(RateSprayed) + Chr$(9) +
            Str$(SentRate) + Chr$(9) + Str$(SVNum) +
            Chr$(9) + Str$(HDOP) + Chr$(9) + Str$(Quality)
            + Chr$(9) + AccFlag
End Sub

Public Sub OpenUp()
    If frmComm MSComm1 PortOpen = False Then
      frmComm MSComm1 PortOpen = True
    End If
    If frmComm MSComm2 PortOpen = False Then
      frmComm MSComm2 PortOpen = True '**IMPT
            before being compiled for FS, port1 must be split
            for LightBar----> change baud to 2400 and switch
            back
    End If
End Sub

Public Sub CloseOut()
    Close #2
    Close #4
    If frmComm MSComm1 PortOpen = True Then
      frmComm MSComm1 PortOpen = False
    End If
    If frmComm MSComm2 PortOpen = True Then
      frmComm MSComm2 PortOpen = False
    End If
End Sub

Public Sub ReStart()
    If frmComm MSComm1 PortOpen = False Then
      frmComm MSComm1 PortOpen = True
    End If
    If frmComm MSComm2 PortOpen = False Then
      frmComm MSComm2 PortOpen = True
    End If
    DrawCount = 0
    Pause = False
    Call BufferFlush
    Call BufferFill
    Do Until Pause = True
      Call RateChoice4
    Loop
End Sub

Public Sub HDOPAlarm()
    BadHDOPCount = 0
    For n = 0 To HDOPTrigger
      If HDOPBuf(n) >= 6 Then
        BadHDOPCount = BadHDOPCount + 1
      End If
    Next n
    If BadHDOPCount >= HDOPTrigger Then
        frmGrid txtWarnAcc Visible = True
        frmGrid txtWarnAcc Text = "Poor Precision Here
            (High HDOP)"
        Beep
    End If
End Sub

Public Sub StopExperimentAlarm()
    If FeedForwardCount >= StopExpTrigger Then
      frmGrid txtWarnAcc Visible = True
      frmGrid txtWarnAcc Text = "Press Pause and Wait for
            Better GPS Signals"
      For n = 1 To 5
        Beep
      Next n
    End If
End Sub

Public Sub WarnAccAlarm()
    If FeedForwardCount >= AccWarnTrigger Then
```

```
    frmGrid txtWarnAcc Visible = True
    frmGrid txtWarnAcc Text = "Poor Accuracy Here"
  End If
End Sub

Public Sub UpdateTable()
  If Speed >= 1# Then
    frmGrid lblSpeedT Caption = Format(Abs(Speed) *
        0 6818, "###0 0") 'DirChange
  Else frmGrid lblSpeedT Caption = "0 0"
  End If
  frmGrid lblTarget = Str$(SentRate)
  If AccFlag = "OK" Then
    frmGrid lblFF = "GPS"
  Else frmGrid lblFF = AccFlag
  End If
  If AbsDrift < 100 Then
    frmGrid lblDrift Caption = Format(AbsDrift, "#0 0")
  Else frmGrid lblDrift Caption = "FAR"
  End If
End Sub

Public Sub ReadNmea()
'ReadNmea - A subroutine to extract the following
        information
' from the NMEA GGA output from a GPS Reciever
'  1 Time - GPS time converted to seconds
'  2 xr - Latitude converted and rotated to x coordinates
'  3 yr - Latitude converted and rotated to y coordinates
'  4 Quality - Quality of GPS Signal (1 or Null) [1 is good]
'  5 SVNum - Number of Satellites in Use
'  6 HDOP - Horizonal Dillution of Precision
'To use this module, a form named "frmComm" with an
        MSComm
'(telephone icon)object named MSComm1 included on it
'This sub-routine separates the GGA string into the 6 needed
        strings
  'Import only one char at a time
  'frmComm MSComm2 Settings = Str$(GPSBaudRate) +
        "," + "n" + Str$(GPSDataBits) + "," +
        Str$(GPSStopBits) ""4800,n,8,1"
  frmComm MSComm2 InputLen = 1
Start
  'Look at the buffer until there is something in it.
  frmComm GPSCommTimer Enabled = True
  DoEventsMod = 0
  LoopCount = 0
  Do Until frmComm MSComm2 InBufferCount > 0
    LoopCount = LoopCount + 1
    DoEventsMod = LoopCount Mod 100
    If DoEventsMod = 0 Then DoEvents
  Loop
  frmComm GPSCommTimer Enabled = False
  'frmGrid txtWarnAcc Visible = False
  'Intialize the input string to nothing
  inp$ = ""
  'Check the input string until it is a "$" mark
  frmComm GPSCommTimer Enabled = True
  DoEventsMod = 0
  LoopCount = 0
  Do Until inp$ = "$"
    inp$ = frmComm MSComm2 Input
    LoopCount = LoopCount + 1
    DoEventsMod = LoopCount Mod 100
    If DoEventsMod = 0 Then DoEvents
  Loop
  frmComm GPSCommTimer Enabled = False
  'Pull out the GPS indentifier using "pull()" function
  'created below
  gps$ = Pull("")
  'If the GPS String is a GGA, then use the "pull()"
  'function to pull out the 6 needed strings
  'Trash$ is used to skip fields in the comma delimited
  'NMEA string
  If InStr(gps$, "GGA") Then
    tt = Pull("")        'Field 1
    ys = Pull("")        'Field 2
    Trash = Pull("")     'Field 3
    xs = Pull("")        'Field 4
    Trash = Pull("")     'Field 5
    Qualitys = Pull("")  'Field 6
    SVNums = Pull("")    'Field 7
    HDOPs = Pull("")     'Field 8
    Trash = Pull("")
    Call CommError
    Trash = ""
  'If the string is not GGA, then start over until
  'a GGA string is found
  Else GoTo Start
  End If
frmGrid txtWarnAcc Visible = False
End Sub

Public Function Pull(inpu$) As String
'This function pulls out strings in between commas
'by adding another character to the string it is checking
'until it finds a comma
'The only argument needed is "Inpu$ "
'Inpu$ is the initial string - it should be set to ""
'in this application
Dim z As String
  frmComm GPSCommTimer Enabled = True
  LoopCount = 0
  DoEventsMod = 0
  Do Until inpu$ = ","
    z = z + inpu$
    inpu$ = frmComm MSComm2 Input
    LoopCount = LoopCount + 1
    DoEventsMod = LoopCount Mod 100
    If DoEventsMod = 0 Then DoEvents
  Loop
  frmComm GPSCommTimer Enabled = False
Pull = z
End Function

Sub CommError()
  If Len(xs) < 4 Or Len(xs) > 15 Then
    Call BufferFlush
    Call DeciferAll
  ElseIf Len(ys) < 4 Or Len(ys) > 15 Then
    Call BufferFlush
    Call DeciferAll
  ElseIf Len(Qualitys) <> 1 Then
    Call BufferFlush
    Call DeciferAll
  ElseIf Len(SVNums) > 2 Then
    Call BufferFlush
    Call DeciferAll
  ElseIf Len(HDOPs) > 5 Or Len(HDOPs) <= 1 Then
    Call BufferFlush
    Call DeciferAll
  End If
End Sub
```

```
Sub Convert()
   Time = Val(Mid$(tt, 1, 2)) * 3600 + Val(Mid$(tt, 3, 2)) *
          60 + Val(Mid$(tt, 5, 2))

   X = Val(Mid$(xs, 1, 3)) + Val(Mid$(xs, 4, ((Len(xs) -
          3)))) / 60
   X = LonToFt * (X - DegRefLon)
   Y = Val(Mid$(ys, 1, 2)) + Val(Mid$(ys, 3, ((Len(ys) -
          2)))) / 60
   Y = LatToFt * (Y - DegRefLat)

   """DATA
   xr = (X * CosTheta + Y * SinTheta)
   yr = (Y * CosTheta - X * SinTheta)
   Call EmployCoeffs
   Quality = Val(Qualitys)
   SVNum = Val(SVNums)
   If Err Then Call CommError
    HDOP = Val(HDOPs)
End Sub

Sub CalcConst()
   LonToFt = Cos(DegRefLat * pi / 180) * 365157 480315
          'Move this to ReadPCMCIA
   LatToFt = 363188 976378 'MOVE this to ReadPcmcia

   MinTimeInPlot = Fix(PlotLength / MaxSpeed / 1 47 *
          fGPS)
       'MinTimeInPlot =aprrox number of gps updates
         'In a plot
End Sub

Public Sub DeciferAll()
   'This is the main routine in this ReadNmea Process
  'It should be the only one called by the MAIN TRACS
  'program to provide ONE instaneous VALUE for each of
  'the 6 fields
   Call CalcConst
   Call ReadNmea
   Call Convert
   Call Buffer
End Sub

Public Sub Buffer()
   For ind = 9 To 1 Step -1
      XBuf(ind) = XBuf(ind - 1)
      YBuf(ind) = YBuf(ind - 1)
      XrBuf(ind) = XrBuf(ind - 1)
      YrBuf(ind) = YrBuf(ind - 1)
      HDOPBuf(ind) = HDOPBuf(ind - 1)
   Next ind
      XBuf(0) = X
      YBuf(0) = Y
      XrBuf(0) = xr
      YrBuf(0) = yr
      HDOPBuf(0) = HDOP
End Sub

Public Sub RateSprayedBuffer()
   For ind = 9 To 1 Step -1
      RSBuf(ind) = RSBuf(ind - 1)
   Next ind
      RSBuf(0) = RateSprayed
End Sub

Public Sub SurveySprayerFlickIt()
   frmComm MSComm2 Output = Str$(RSBuf(0)) +
          Chr$(13)
End Sub

Public Sub FlickIt()
 If frmComm MSComm2 PortOpen = True Then
  If RSBuf(0) <> RSBuf(1) Then
     If Calibrating = True Then
        If (RSBuf(0) = 1 And RSBuf(1) = 2) Or (RSBuf(0) =
           2 And RSBuf(1) = 1) Then
          Call AVGVelPTS
          frmComm MSComm2 Output = Str$(1000) +
           Chr$(13)
        End If
     Else  frmComm MSComm2 Output = Str$(1000) +
           Chr$(13) 'mComm MSComm1 Output = "1000" +
           Chr$(13)
     End If
  End If
 End If
End Sub

Public Sub FlickItLine()
  frmComm MSComm2 OutBufferCount = 0
  If RSBuf(0) <> RSBuf(1) Then
     If RSBuf(0) <> RateList(0) Then 'BorderRate
        If RSBuf(1) <> RateList(0) Then 'BorderRate
           frmComm MSComm2 Output = "100" +
          Chr$(13)
        End If
     End If
  End If
End Sub

Public Sub CountDown()
TCount = TCount + 1
   If TopPlot(T) = True Then
      If dydtUsed > 0 Then
         RateSprayed = RateList(Rate(T + DirChange)) *
            SpentTarget(T + DirChange)
      Else  RateSprayed = RateList(0) 'RateList(0)
            'BorderRate
      End If
   Else
      If BottomPlot(T) = True Then
         If dydtUsed < 0 Then
            RateSprayed = RateList(Rate(T + DirChange)) *
            SpentTarget(T + DirChange)
         Else  RateSprayed = RateList(0)
         End If
      Else  RateSprayed = RateList(Rate(T + DirChange)) *
            SpentTarget(T + DirChange)
      End If
   End If
SentRate = PauseRate * RateSprayed
'The following lines of code adapt the Field System for the
          line test
Call RateSprayedBuffer
If Sprayer = 1 Then
   Call FlickIt
ElseIf Sprayer = 2 Then
   Call SurveySprayerFlickIt
End If
End Sub

Public Sub TFinder() 'Finds Closest Target and Distance to
          that Target
                'Please Refer to Sub TBuffer for Storage
```

```
            info
    MinCpDist = 100000
    For n = 1 To NumPlots
      CpDist = ((XrBuf(0) - CpX(n)) ^ 2 + (YrBuf(0) -
            CpY(n)) ^ 2) ^ (0 5)
      If CpDist < MinCpDist Then
        MinCpDist = CpDist
        T = n
      End If
    Next n
End Sub

Public Sub BufferFill()
    For nnn = 0 To BuffUse
      Call DeciferAll
      Call Navigation
      AccFlagBuf(nnn) = "OK"
    Next nnn
End Sub
Public Sub CommTime()
    frmComm GPSCommTimer Enabled = False
    frmGrid txtWarnAcc Visible = True
    frmGrid txtWarnAcc Text = "Check GPS Connection"
End Sub

Public Sub BufferFlush()
    frmComm MSComm2 InBufferCount = 0
End Sub

Public Sub Alarm()
    Call HDOPAlarm
    Call WarnAccAlarm
    Call StopExperimentAlarm
End Sub

Public Sub DirChanger()
    OldDirChange = DirChange
    OldDrawChange = DrawChange
    If dydtUsed < 0 Then
        DirChange = 1
        FirstQuadX = pxrMin
        FirstQuadY = pyrMin
        FQ = 1
      Else  DirChange = -1
        FirstQuadX = pxrMax
        FirstQuadY = pyrMax
        FQ = 1
    End If
If FoCount = 1 Then
    DrawChange = DirChange
    OldDrawChange = OldDirChange
End If
If DrawMod = 0 Then
    DrawChange = DirChange
End If
End Sub
Public Sub SitStill()
    Jumps = 0
    For n = 0 To ScreenFlip '4
      If (dydtUsed > 0 And dydtBuf(n + 1) < 0) Or (dydtUsed
            < 0 And dydtBuf(n + 1) > 0) Then
        Exit Sub
      End If
      If AccFlagBuf(n) = "FF" Then 'Jumps = Jumps + 1
        Exit Sub
      End If
    Next n
    If Speed > MinSpeed Then
      Call DirChanger
    End If
End Sub

Public Sub ETACalc()
    "IMPORTANT the below line uses FF'd velocity
    dt = (CpY(T) - YUsed - DirChange * PlotLength / 2) /
            (dydtUsed + 0 000000001)
    ETA = Abs(dt) * 1000 '- TimeDelay
    If ETA > 30000 Then
      ETA = 30000
    End If
    If dydtUsed < 0 Then
      If dt < 0 Then
        Call WithTimer
      Else  Call NoTimer
      End If
    Else
      If dt > 0 Then
        Call WithTimer
      Else  Call NoTimer
      End If
    End If
End Sub

Public Sub WithTimer()
        frmComm tmrSpray Enabled = False
        frmComm tmrSpray Interval = ETA
        frmComm tmrSpray Enabled = True
End Sub

Public Sub NoTimer()
    frmComm.tmrSpray Enabled = False
    If InBorder(0) = True Then
      RateSprayed = RateList(0) 'BorderRate
    Else  RateSprayed = RateList(Rate(T)) * SpentTarget(T)
      End If
SentRate = PauseRate * RateSprayed 'RateList(RateSprayed)
Call RateSprayedBuffer
If Sprayer = 1 Then
    Call FlickIt 'Line
ElseIf Sprayer = 2 Then
    Call SurveySprayerFlickIt
End If
End Sub

Public Sub TInfo()
On Error GoTo ehandle
      LETC = 0
      NIBC = 0
      PlotInQuestion = TBuf(MinTimeInPlot)
      If DrawMod = 0 Then STDrawPlot = PlotInQuestion
      If TBuf(MinTimeInPlot) <> TBuf(MinTimeInPlot - 1)
            Then
        OutofTarget(PlotInQuestion) = True
      End If
      For n = 1 To MinTimeInPlot
        If TBuf(n) = TBuf(0) Then
          LETC = LETC + 1
        End If
        If InBorder(n) = False Then
          NIBC = NIBC + 1
        End If
      Next n
      If LETC = MinTimeInPlot Then
        LongEnoughInTarget(T) = True
```

```
      End If
      If NIBC = MinTimeInPlot Then
        NotInBorder(T) = True
      End If
        STBOld = SpentTargetBool(PlotInQuestion)
        SpentTargetBool(PlotInQuestion) =
            OutofTarget(PlotInQuestion) And
            LongEnoughInTarget(PlotInQuestion) And
            NotInBorder(PlotInQuestion)
        SpentTarget(PlotInQuestion) = CInt(Not
            (SpentTargetBool(PlotInQuestion))) * -1
        If SpentTargetBool(PlotInQuestion) = True Then
          If STBOld = False Then
            StoreSpentTarget
          End If
        End If
If n = "-45" Then
ehandle
  Exit Sub
End If
End Sub

Public Sub TBuffer()
  For ind = MinTimeInPlot * 2 To 1 Step -1 'This Buffer is
          twice as big as is
  '       the estimated time required in each plot
    TBuf(ind) = TBuf(ind - 1)
    MinCpBuf(ind) = MinCpBuf(ind - 1)
    InBorder(ind) = InBorder(ind - 1)
  Next ind
    TBuf(0) = T
    MinCpBuf(0) = MinCpDist
    If MinCpBuf(0) > Diagonal Then
      InBorder(0) = True
    Else  InBorder(0) = False
    End If
End Sub

Public Sub Navigation()
  'Call Multipliers "Moved to LimitsAndSensitivitiesUpdate
                   "Because of redundant Calculations
  dy = 0
  dx = 0
  Drift = 0

  For BuffPos = 0 To BuffUse
    dx = (XrBuf(BuffPos) - XrBuf(BuffPos + 1)) *
         Multiplier(BuffPos) + dx 'From within buffer
    dy = (YrBuf(BuffPos) - YrBuf(BuffPos + 1)) *
         Multiplier(BuffPos) + dy 'From within buffer
    Resultant = (dx ^ 2 + dy ^ 2) ^ (0 5) + 0 0000000001
  Next BuffPos
  dydt = dy * fGPS 'in ft per second
  dxdt = dx * fGPS
  Speed = Resultant * fGPS
  Call VelocityRejector  'Dumps slow speed to Zero
 CosAngle1 = CosAngle
 SinAngle1 = SinAngle
  CosAngle = dx / Resultant
  SinAngle = dy / Resultant
  Call VelocityRejector  'Dumps slow speed to Zero
  Call NavigationBuffer
End Sub

Public Sub NavigationBuffer()
  For ind = 9 To 1 Step -1
    dxdtBuf(ind) = dxdtBuf(ind - 1)
    dydtBuf(ind) = dydtBuf(ind - 1)
    SpeedBuf(ind) = SpeedBuf(ind - 1)
    AccFlagBuf(ind) = AccFlagBuf(ind - 1)
  Next ind
    dxdtBuf(0) = dxdt
    dydtBuf(0) = dydt
    SpeedBuf(0) = Speed
    AccFlagBuf(0) = AccFlag
End Sub

Public Sub LimitsAndSensitivitiesUpdate()
  MaxGPSError =
        CSng(frmSettings cboMaxGPSError Text)
  AccWarnTrigger =
        CInt(frmSettings cboAccWarnTrigger Text)
  StopExpTrigger =
        CInt(frmSettings cboStopExpTrigger Text)
  HDOPTrigger = CInt(frmSettings cboHDOPTrigger Text)
  DataSmoothingSensitivity =
        CInt(frmSettings cboDataSmoothing Text)
  BuffUse = DataSmoothingSensitivity
  MaxSpeed = CInt(frmSettings cboMaxSpeed Text)
  ScreenFlip = CByte(frmSettings cboScreenFlip Text)
  DrawPeriod = CSng(frmSettings cboDrawUpdate Text)
  Increment = CSng(frmSettings cboLBARincr Text)
  MinSpeed = CSng(frmSettings cboMinSpeed Text)
  LookPeriod = CSng(frmSettings cboLookPeriod Text)
  LineThickness =
        CByte(frmSettings cboLineThickness Text)
  SprayerSize = CByte(frmSettings cboSprayerSize Text)
  If frmSettings cboMethod Text = "VELOCITY" Then
    Method = 1
  Else  Method = 2
  End If
  Call StepSize
  Call Multipliers
  If frmSettings cboFeedForwardControl Text = "ON" Then
    FFControl = True
  Else  FFControl = False
  End If
  If frmSettings cboETATimer Text = "ON" Then
    ETATimer = True
  Else  ETATimer = False
  End If
End Sub

Public Sub LimitsAndSensitivitiesDefaults()
  frmSettings cboMaxGPSError Text = 3
  frmSettings cboAccWarnTrigger Text = 3
  frmSettings cboStopExpTrigger Text = 10
  frmSettings cboHDOPTrigger Text = 5
  frmSettings cboDataSmoothing Text = 3
  frmSettings cboFeedForwardControl Text = "ON"
  frmSettings cboETATimer Text = "ON"
  frmSettings cboMaxSpeed Text = "15"
  frmSettings cboScreenFlip Text = 4
  frmSettings cboDrawUpdate Text = 1
  frmSettings cboLBARincr Text = 1
  frmSettings cboLookPeriod Text = 0 5
  frmSettings cboMethod Text = "VELOCITY"
  frmSettings cboLineThickness Text = 1
  frmSettings cboSprayerSize Text = 8
  frmSettings cboMinSpeed Text = 0 7
  frmSettings cboUsedTargetTracking Text = "ON"
End Sub

Public Sub LimitsAndSensitivitiesBack()
```

```
    frmSettings fraLimitsAndSensitivities Visible = False
    frmSettings fraMenu Visible = True
End Sub

Public Sub CommunicationsUpdate()
    GPSBaudRate =
            Str$(CSng(frmSettings cboGPSBaud Text) *
            1000)
    GPSStopBits =
            Str$(CInt(frmSettings cboGPSStopBits Text))
    GPSDataBits =
            Str$(CInt(frmSettings cboGPSdataBits Text))
    GPSPort = Str$(CInt(frmSettings cboGPSPort Text))
    GPSUpdateRate =
            CInt(frmSettings cboGPSUpdateRate Text)

    LBarBaudRate =
            Str$(CSng(frmSettings cboLBarBaud Text) *
            1000)
    LBarStopBits =
            Str$(CInt(frmSettings cboLBarStopBits Text))
    LBarDataBits =
            Str$(CInt(frmSettings cboLBarDataBits Text))
    LBarPort = Str$(CInt(frmSettings cboLBarPort Text))
    ApplrBaudRate =
            Str$(CSng(frmSettings cboApplrBaud Text) *
            1000)
    ApplrStopBits =
            Str$(CInt(frmSettings cboApplrStopBits Text))
    ApplrDataBits =
            Str$(CInt(frmSettings cboApplrDataBits Text))
    ApplrPort = Str$(CInt(frmSettings cboApplrPort Text))
    'Call this sub routine after Communication Update
    frmComm MSComm2 Settings = ApplrBaudRate + "," +
            "n" + "," + Str$(ApplrDataBits) + "," +
            Str$(ApplrStopBits)
    If frmComm MSComm2 CommPort <> ApplrPort Then
      frmComm MSComm2 CommPort = ApplrPort
    End If
    frmComm MSComm2 Settings = GPSBaudRate + "," +
            "n" + "," + Str$(GPSDataBits) + "," +
            Str$(GPSStopBits)
    If frmComm MSComm2 CommPort <> GPSPort Then
      frmComm MSComm2 CommPort = GPSPort
    End If
    fGPS = GPSUpdateRate
End Sub

Public Sub CommunicationsDefaults()
    frmSettings cboGPSBaud Text = 9 6 '4 8
    frmSettings cboGPSStopBits Text = 1
    frmSettings cboGPSdataBits Text = 8
    frmSettings cboGPSPort Text = 2
    frmSettings cboGPSUpdateRate Text = 1
    frmSettings cboLBarBaud Text = 2 4
    frmSettings cboLBarStopBits Text = 1
    frmSettings cboLBarDataBits Text = 8
    frmSettings cboLBarPort Text = 2
    frmSettings cboApplrBaud Text = 9 6
    frmSettings cboApplrStopBits Text = 1
    frmSettings cboApplrDataBits Text = 8
    frmSettings cboApplrPort Text = 1
End Sub

Public Sub CommunicationsBack()
    frmSettings fraComm Visible = False
    frmSettings fraMenu Visible = True
End Sub

Public Sub HardwareSpecificationsDefaults()
    frmSettings cboHorizontal Text = "320"
    frmSettings cboVertical Text = "240"
    frmSettings cboPCCardDrive Text = "D"
    frmSettings cboSprayer Text = "TEST SPRAYER"
End Sub

Public Sub HardwareSpecificationsUpdate()
    Horizontal = CInt(frmSettings cboHorizontal Text)
    Vertical = CInt(frmSettings cboVertical Text)
    PCCardDrive = frmSettings cboPCCardDrive Text
    If InStr(CStr(frmSettings cboSprayer Text), "TEST
            SPRAYER") Then
      Sprayer = 1
    ElseIf InStr(CStr(frmSettings cboSprayer Text),
            "SURVEY SPRAYER") Then
      Sprayer = 2
    End If
    Call ScreenResolution
End Sub

Public Sub HardwareSpecificationsBack()
    frmSettings fraHardWareSpecifications Visible = False
    frmSettings fraMenu Visible = True
End Sub

Public Sub RatesDefaults()
    For n = 0 To 12
      frmSettings cboRate(n) = OriginalRateList(n)
    Next n
End Sub

Public Sub RatesUpdate()
    For n = 0 To 12
      RateList(n) = frmSettings cboRate(n) Text
    Next n
End Sub

Public Sub RatesBack()
    frmSettings fraRates Visible = False
    frmSettings fraMenu Visible = True
End Sub

Public Sub ApplicationRates()
    frmSettings fraMenu Visible = False
    frmSettings fraRates Visible = True
End Sub

Public Sub Communications()
    frmSettings fraMenu Visible = False
    frmSettings fraComm Visible = True
End Sub

Public Sub HardWareSpecs()
    frmSettings fraMenu Visible = False
    frmSettings fraHardWareSpecifications Visible = True
End Sub

Public Sub LimitsAndSensitivities()
    frmSettings fraMenu Visible = False
    frmSettings fraLimitsAndSensitivities Visible = True
End Sub

Public Sub Back()
    frmSettings Visible = False
```

```
    frmMenu Visible = True
    frmMenu fraMenu Visible = True
End Sub

Public Sub DeviceTestBack()
    frmTest Visible = False
    frmMenu Visible = True
    frmMenu fraMenu Visible = True
End Sub

Public Sub LightBarTest()
    frmComm MSComm1 Settings = LBarBaudRate + "," +
            "n" + "," + LBarDataBits + "," + LBarStopBits
            """2400,n,8,1"
    If frmComm MSComm1 PortOpen = False Then
        frmComm MSComm1 PortOpen = True
    End If
    frmComm MSComm1 OutBufferCount = 0 '
    For n = 0 To 14
        LED = n
        frmComm MSComm1 Output = Str$(LED) + Chr$(13)
        For m = 1 To 10000
            For ss = 1 To 50
            Next ss
        Next m
    Next n
ss = 0
frmComm MSComm1 Output = "07" + Chr$(13)
End Sub

Public Sub GPSTest()
    If frmComm MSComm2 PortOpen = False Then
        frmComm MSComm2 PortOpen = True
    End If
    Call DeciferAll
    frmTest txtGPSTest Text = "LON=" + xs + " LAT=" + ys
End Sub

Public Sub ApplicatorTest()
    If frmComm MSComm2 PortOpen = False Then
        frmComm MSComm2 PortOpen = True
    End If
frmComm MSComm2 Output = Str$(1000) + Chr$(13)
End Sub

Public Sub GETCalRefPTS()
    NumPoints = 10
    For n = 1 To NumPoints
        Call ReadNmea
        Xes(n) = Val(Mid$(xs, 1, 3)) + Val(Mid$(xs, 4,
                ((Len(xs) - 3)))) / 60
        Yes(n) = Val(Mid$(ys, 1, 2)) + Val(Mid$(ys, 3,
                ((Len(ys) - 2)))) / 60
    Next n
    PointNum = 1
    Call AVGPTS
    DegRefLon = XAvg(1)
    DegRefLat = YAvg(1)
    DegRefLatCal = DegRefLat
    DegRefLonCal = DegRefLon
    XAvg(1) = 0
    YAvg(1) = 0
End Sub

Public Sub GETPTS()
    For n = 1 To NumPoints
        Call ReadNmea
        Call Convert
        Xes(n) = X
        Yes(n) = Y
    Next n
End Sub

Public Sub AVGPTS()
    XSum = 0
    YSum = 0
    For n = 1 To NumPoints 'mm
        XSum = XSum + Xes(n)
        YSum = YSum + Yes(n)
    Next n
    XAvg(PointNum) = XSum / NumPoints ' (numpoints - 1)
            * fGPS
    YAvg(PointNum) = YSum / NumPoints ' (numpoints - 1)
            * fGPS
End Sub

Public Sub CalibrationTheta()
    XAvg(1) = 0
    YAvg(1) = 0
    dxc = XAvg(2) - XAvg(1)
    dyc = YAvg(2) - YAvg(1)
    Resultant = (dxc ^ 2 + dyc ^ 2) ^ 0 5
    CosThetaCal = dxc / Resultant
    SinThetaCal = dyc / Resultant
End Sub

Public Sub GETVelPTS()
        PCount = PCount + 1
        Call ReadNmea
        Call Convert
        Xes(PCount) = xr
        Yes(PCount) = yr
End Sub

Public Sub Calibrate()
    Velocity1 = 0
    For n = 1 To sc
        Velocity1 = Velocity1 + CalSpeed(n)
    Next n
    Velocity1 = Velocity1 / sc
    Velocity2 = 0
    Error1 = 0
    For n = 1 To 4
        Error1 = Error1 + CSng(frmCalibrate cboError(n) Text)
    Next n
    Error1 = Error1 / 4
    Error2 = -
            CSng(frmCalibrate cboAntToBoomDistance Text
            )
    CalCoeff = (Error2 - Error1) / (Velocity2 - Velocity1)
    MuleAnt = Error2
    frmCalibrate txtCalCoeff Text = CalCoeff
    frmCalibrate cboCalCoeff Text = CalCoeff
    frmCalibrate cboAntennaToBoomDist2 Text = MuleAnt
End Sub

Public Sub AVGVelPTS()
    CalSpeed(1) = 0
    sc = sc + 1
    For n = 1 To 9
        CalSpeed(sc) = SpeedBuf(n) + CalSpeed(sc)
    Next n
        CalSpeed(sc) = CalSpeed(sc) / 9
End Sub
```

```
Public Sub CalibrationSetup()
   SinTheta = SinThetaCal
   CosTheta = CosThetaCal
   DegRefLat = DegRefLatCal
   DegRefLon = DegRefLonCal
   CalCoeff = 0
   MuleAnt = 0
End Sub

Public Sub LoadCalPerimeter()
   strExperiment = PCCardDrive + " \Calibr\Calibr per"
   On Error GoTo ehandle
   zz = 0
   Open strExperiment For Input As #1
   Do While Not EOF(1) ' Check for end of file
      Line Input #1, StrC
      zz = zz + 1
      Length1 = Len(StrC)
      NFields = 0
   For i = 1 To Length1
      StrD = Mid$(StrC, i, 1)
      If StrD = Chr$(9) Then
         NFields = NFields + 1
         ww(NFields) = i
      End If
   Next i

   qwerty = ww(0) + 1
   Length2 = ww(1) - ww(0) - 1
   lonp = Mid$(StrC, qwerty, Length2)
   px(zz) = Val(lonp)
   NoRotBx(zz) = px(zz)

   qwerty = ww(1) + 1
   Length2 = ww(2) - ww(1) - 1
   latp = Mid$(StrC, qwerty, Length2)
   py(zz) = Val(latp)
   NoRotBy(zz) = py(zz)

   pxr(zz) = (px(zz) * CosTheta + py(zz) * SinTheta) + xoff
   pyr(zz) = (py(zz) * CosTheta - px(zz) * SinTheta) '+
         MuleAnt
Loop
Close #1
If n = "-45" Then
ehandle
   frmCalibrate Visible = False
   frmSettings Visible = True
   frmSettings fraHardWareSpecifications Visible = True
   Exit Sub
End If
End Sub

Public Sub Point1()
   Call BufferFlush
   frmCalibrate cmdPoint1 Enabled = False
   frmCalibrate cmdStep3Back Enabled = False
   frmCalibrate cmdStep3Next Enabled = False
   Call CommunicationsUpdate
   If frmComm MSComm2 PortOpen = False Then
      frmComm MSComm2 PortOpen = True
   End If
   Call GETCalRefPTS
   Call CalcConst
   frmCalibrate cmdPoint1 Enabled = True
   frmCalibrate cmdStep3Back Enabled = True
   frmCalibrate cmdStep3Next Enabled = True
   frmCalibrate cmdPoint2 Enabled = True
End Sub

Public Sub Point2()
   Call BufferFlush
   frmCalibrate cmdPoint2 Enabled = False
   frmCalibrate cmdStep5Back Enabled = False
   frmCalibrate cmdStep5Next Enabled = False
   NumPoints = 10
   Call GETPTS
   PointNum = 2
   Call AVGPTS
   Call CalibrationTheta
   frmCalibrate cmdPoint2 Enabled = True
   frmCalibrate cmdStep5Back Enabled = True
   frmCalibrate cmdStep5Next Enabled = True
   frmCalibrate cmdStep6Next Enabled = True
End Sub

Public Sub ExperimentNameOK()
   f = ""
   For n = 0 To 6
      Temp1 =
            CStr(frmCalibrate cboExperimentName(n) Text)
      f = f + Temp1
   Next n
   frmCalibrate txtExperimentName Text = f
End Sub

Public Sub StoreCalibration()
   Open PCCardDrive + " \Calibr\" +
         frmCalibrate txtExperimentName Text + " cal"
         For Output As #3
   Print #3, Str$(CalCoeff) + Chr$(9) + Str$(MuleAnt) +
         Chr$(9)
   Close #3
End Sub

Public Sub ReadCalibration()
   Open PCCardDrive + " \Calibr\" +
         frmCalibrate cboCalibrationFiles Text For Input
         As #3
   Line Input #3, StrC
   Length1 = Len(StrC)
   NFields = 0
   For i = 1 To Length1
      StrD = Mid$(StrC, i, 1)
      If StrD = Chr$(9) Then
         NFields = NFields + 1
         ww(NFields) = i
      End If
   Next i
   p1 = ww(0) + 1
   Length2 = ww(1) - ww(0) - 1
   CalCoeff = Val(Mid$(StrC, p1, Length2))
   p1 = ww(1) + 1
   Length2 = ww(2) - ww(1) - 1
   MuleAnt = Val(Mid$(StrC, p1, Length2))
Close #3
   frmCalibrate cboAntennaToBoomDist2 Text = MuleAnt
   frmCalibrate cboCalCoeff Text = CalCoeff
End Sub

Public Sub FindCalibrationFiles()
   Call EraseCalibrationNames
   CalCount = 0
```

```
    Temp1 = Dir(PCCardDrive + " \Calibr\* cal")
      Do Until Temp1 = ""
        CalCount = CalCount + 1

          Dirs(CalCount) = Temp1
        Temp1 = Dir
      Loop
End Sub

Public Sub ListCalibrationFiles()
 frmCalibrate cboCalibrationFiles Text = Dirs(1)
  For n = 1 To CalCount
    frmCalibrate cboCalibrationFiles AddItem Dirs(n)
  Next n
End Sub

Public Sub EraseCalibrationNames()
  If CalCount >= 1 Then
    For n = 1 To CalCount
     frmCalibrate cboCalibrationFiles RemoveItem (0)
    Next n
  End If
End Sub

Public Sub EmployCoeffs()
  MuleAntX = MuleAnt * CosAngle
  MuleAntY = MuleAnt * SinAngle
  xr = xr + MuleAntX + dxdtUsed * CalCoeff
  yr = yr + MuleAntY + dydtUsed * CalCoeff
End Sub

Public Sub CalibrationUpdate()
  CalCoeff = CSng(frmCalibrate cboCalCoeff Text)
  MuleAnt =
          CSng(frmCalibrate cboAntennaToBoomDist2 Te
          xt)
End Sub

Public Sub ReadRecoveryFile()
RecoveryFile = PCCardDrive + " \" + Directory + "\" +
          Directory + " rec"
On Error GoTo SkipRec
Open RecoveryFile For Input As #1
n = 0
Do While Not EOF(1)
n = n + 1
Line Input #1, StrC
  Length1 = Len(StrC)
  NFields = 0
  For i = 1 To Length1
    StrD = Mid$(StrC, i, 1)
    If StrD = Chr$(9) Then
      NFields = NFields + 1
      ww(NFields) = i
    End If
  Next i
  p1 = ww(0) + 1
  Length2 = ww(1) - ww(0) - 1
  Trash = Val(Mid$(StrC, p1, Length2))
   LongEnoughInTarget(Trash) = True
   OutofTarget(Trash) = True
   NotInBorder(Trash) = True
   SpentTarget(Trash) = 0
   SpentTargetBool(Trash) = True
Loop
Close #1
SkipRec
End Sub

Public Sub StoreSpentTarget()
If Calibrating = False Then
  Print #4, Str$(PlotInQuestion) + Chr$(9)
End If
End Sub
```

Graphical Module

'frmCalibrate

```
Private Sub cboAntennaToBoomDist2_Change()
  cmdFinishMDE Enabled = True
End Sub

Private Sub cboCalCoeff_GotFocus()
  cmdFinishMDE Enabled = True
End Sub

Private Sub cboCalibrationFiles_Click()
  cmdFinishEC Enabled = True
End Sub

Private Sub cboCalibrationFiles_GotFocus()
  cmdFinishEC Enabled = True
End Sub

Private Sub cmdBack_Click()
  fraMenu Visible = False
  frmMenu Visible = True
  frmMenu fraMenu Visible = True
End Sub

Private Sub cmdBackToMainMenuEC_Click()
  fraExistingCalibration Visible = False
  fraMenu Visible = True
End Sub

Private Sub cmdCalibrationResultsBackToMain_Click()
  fraCalibrationResults Visible = False
  fraMenu Visible = True
End Sub

Private Sub cmdExistingCalibration_Click()
  Call FindCalibrationFiles
  Call ListCalibrationFiles
  fraMenu Visible = False
  fraExistingCalibration Visible = True
End Sub

Private Sub cmdExistingCalibrationBack_Click()
  fraExistingCalibration Visible = False
  fraMenu Visible = True
End Sub

Private Sub cmdExperimentNameOK_Click()
  Call ExperimentNameOK
  cmdExperimentNameOK Enabled = False
End Sub

Private Sub cboExperimentName_Click(Index As
Integer)
  cmdExperimentNameOK Enabled = True
```

```
End Sub

Private Sub cmdFinish_Click()
  Call StoreCalibration
  Call CalibrationUpdate
  fraStep11 Visible = False
  fraMenu Visible = True
End Sub

Private Sub cmdFinishEC_Click()
  cmdFinishEC Enabled = False
  Call ReadCalibration
End Sub

Private Sub cmdFinishMDE_Click()
  cmdFinishMDE Enabled = False
  Call CalibrationUpdate
End Sub

Private Sub cmdManualDataEntry_Click()
  fraMenu Visible = False
  fraManualDataEntry Visible = True
End Sub

Private Sub cmdManualDataEntryBack_Click()
  fraManualDataEntry Visible = False
  fraMenu Visible = True
End Sub

Private Sub cmdNewCalibration_Click()
  fraMenu Visible = False
  fraStep1 Visible = True
End Sub

Private Sub cmdPoint1_Click()
  Call Point1
End Sub

Private Sub cmdPoint2_Click()
  Call Point2
End Sub

Private Sub cmdStep10Back_Click()
  fraStep10 Visible = False
  fraStep9 Visible = True
End Sub

Private Sub cmdStep10Next_Click()
  fraStep10 Visible = False
  fraStep11 Visible = True
End Sub

Private Sub cmdStep11Back_Click()
  fraStep11 Visible = False
  fraStep10 Visible = True
End Sub

Private Sub cmdStep11Next_Click()
  fraStep11 Visible = False
  fraStep12 Visible = True
End Sub

Private Sub cmdStep12Back_Click()
  fraStep12 Visible = False
  fraStep11 Visible = True
End Sub

Private Sub cmdStep12Next_Click()
  fraStep12 Visible = False
  fraStep13 Visible = True
End Sub

Private Sub cmdstep13Back_Click()
  fraStep13 Visible = False
  fraStep12 Visible = True
End Sub

Private Sub cmdstep13Next_Click()
  fraStep13 Visible = False
  fraStep14 Visible = True
End Sub

Private Sub cmdStep14Back_Click()
  fraStep14 Visible = False
  fraStep13 Visible = True
End Sub

Private Sub cmdSTEP14Next_Click()
  fraStep14 Visible = False
  fraStep15 Visible = True
End Sub

Private Sub cmdStep15Back_Click()
  fraStep15 Visible = False
  fraStep14 Visible = True
End Sub

Private Sub cmdStep1Back_Click()
  fraStep1 Visible = False
  fraMenu Visible = True
End Sub

Private Sub cmdStep1Next_Click()
  fraStep1 Visible = False
  fraStep2 Visible = True
End Sub

Private Sub cmdStep2Back_Click()
  fraStep2 Visible = False
  fraStep1 Visible = True
End Sub

Private Sub cmdStep2Next_Click()
  fraStep2 Visible = False
  fraStep3 Visible = True
End Sub

Private Sub cmdStep3Back_Click()
  fraStep3 Visible = False
  fraStep2 Visible = True
End Sub

Private Sub cmdStep3Next_Click()
  fraStep3 Visible = False
  fraStep4 Visible = True
End Sub

Private Sub cmdStep4Back_Click()
  fraStep4 Visible = False
  fraStep3 Visible = True
End Sub

Private Sub cmdStep4Next_Click()
  fraStep4 Visible = False
```

```
    fraStep5 Visible = True
End Sub

Private Sub cmdStep5Back_Click()
    fraStep5 Visible = False
    fraStep4 Visible = True
End Sub

Private Sub cmdStep5Next_Click()
    fraStep5 Visible = False
    fraStep6 Visible = True
End Sub

Private Sub cmdStep6Back_Click()
    fraStep6 Visible = False
    fraStep5 Visible = True
End Sub

Private Sub cmdStep6Next_Click()
    Call BufferFlush
    fraStep6 Visible = False
    frmCalibrate Visible = False
    frmGrid Visible = True
    frmGrid cmdZOOM Enabled = False
    frmGrid cmdMenu Enabled = False
    frmGrid cmdPause Enabled = False
    frmGrid cmdQuit Enabled = False
    frmGrid cmdResume Enabled = False
    Pause = False
    Calibrating = True
    sc = 0
    frmGrid cmdDone Visible = True
    Call BufferFlush
    Call MainProgram
    frmGrid cmdZOOM Enabled = True
    frmGrid cmdMenu Enabled = True
    frmGrid cmdPause Enabled = True
    frmGrid cmdQuit Enabled = True
    frmGrid cmdResume Enabled = True
End Sub

Private Sub cmdStep7Back_Click()
    fraStep7 Visible = False
    fraStep6 Visible = True
End Sub

Private Sub cmdStep7Next_Click()
    fraStep7 Visible = False
    fraStep8 Visible = True
End Sub

Private Sub cmdStep8Back_Click()
    fraStep8 Visible = False
    fraStep7 Visible = True
End Sub

Private Sub cmdStep8Next_Click()
    fraStep8 Visible = False
    fraStep9 Visible = True
End Sub

Private Sub cmdStep9Back_Click()
    fraStep9 Visible = False
    fraStep8 Visible = True
End Sub

Private Sub cmdStep9Next_Click()
    fraStep9 Visible = False
    fraStep10 Visible = True
    cmdStep10Next Enabled = True
    If sc > 0 Then
        Call Calibrate
    Else  txtCalCoeff Text = "Calibration failed, try again"
          cmdStep10Next Enabled = False
    End If
End Sub

Private Sub Command13_Click()
    AcrossLine = True
End Sub


'frmComm

Public Sub GPSCommTimer_Timer()
    Call CommTime
End Sub

Private Sub tmrbadfile_Timer()
    tmrbadfile Enabled = False
Call MainProgram
End Sub

Private Sub tmrCloseOut_Timer()
    Call CloseOut
    tmrCloseOut Enabled = False
    frmMenu fraMenu Visible = True
    frmMenu cmdStart Enabled = True
    frmMenu fraQuit Visible = False
    frmMenu fraCalibrationWarning Visible = False
    frmMenu fraQuit.Enabled = True ""
End Sub

Private Sub tmrDeviceTest_Timer()
    tmrDeviceTest Enabled = False
    If frmComm MSComm1 PortOpen = True Then
        frmComm MSComm1 PortOpen = False
    End If
    If frmComm MSComm2 PortOpen = True Then
        frmComm MSComm2 PortOpen = False
    End If
    Call DeviceTestBack
    frmMenu Enabled = True
    frmMenu fraMenu Enabled = True
End Sub

Private Sub tmrError_Timer()
    tmrError Enabled = False
    frmSettings fraHardWareSpecifications Enabled = True
    Exit Sub
End Sub

Private Sub tmrPause_Timer()
    'This timer allows for while loops to recognize terminating
    'variable  Keeps user from executing anything
  'Call CloseOut
  tmrPause Enabled = False
  If frmCalibrate fraStep7 Enabled = False Then
      Calibrating = False
      frmCalibrate fraStep7 Enabled = True
  ElseIf frmTest fraDeviceTest Enabled = False Then
      frmTest fraDeviceTest Enabled = True
  Else  frmGrid fraZoom Enabled = True
      frmGrid fraRatesAndMenu Enabled = True
```

```
    frmGrid cmdPause Enabled = False
    frmGrid cmdResume Enabled = True
 End If
End Sub

Public Sub tmrSpray_Timer()
  Call CountDown
End Sub


'FRMGRID

Private Sub cmdAuto_Click()
  Call AutoZoom
End Sub

Private Sub cmdD_Click()
  frmGrid Cls
  If zin = True Then
    PixOffY = PixOffY + 5 'PixOffY * 1 25
    Call DrawingCps
    Call DrawAll
  Else   FoPixOffY = FoPixOffY + 5 '* 1 25
  End If
End Sub

Private Sub cmdDone_Click()
  Pause = True
  'Calibrating = False ""Step moved to a timer routine to
          allow for while loops to terminate
    'in response to Pause variable being set to true
  frmCalibrate fraStep7 Enabled = False
  frmComm tmrPause Interval = 1500
  frmComm tmrPause Enabled = True
  cmdDone Visible = False
  frmGrid Visible = False
  frmCalibrate Visible = True
  frmCalibrate fraStep7 Visible = True
  frmCalibrate fraStep7 Visible = True
End Sub

Private Sub cmdMenu_Click()
  frmMenu Visible = True
  frmMenu fraMenu Visible = True
  frmMenu fraStart Visible = False
  frmMenu fraQuit Visible = False
  frmGrid Visible = False
  frmSettings Visible = False
  frmMenu cmdBACKtoExp Enabled = True
End Sub

Private Sub cmdOriginal_Click()
  If zin = True Then
    Call ScaleXY
  Else  Call ScaleNoRot
  End If
End Sub

Private Sub cmdOUT_Click()
frmGrid Cls
If cmdOUT Caption = "OUT" Then
    cmdOUT Caption = "IN"
    zin = False
  Else  cmdOUT Caption = "OUT"
    zin = True
    cmdAuto_Click
  End If
End Sub

Private Sub cmdPause_Click()
  Pause = True
  PauseRate = 0
  fraZoom Enabled = False
  fraRatesAndMenu Enabled = True
  frmComm tmrPause Interval = 1500
  frmComm tmrPause Enabled = True
End Sub

Private Sub cmdQuit_Click()
  frmMenu Visible = True
  frmMenu fraQuit.Visible = True
  frmGrid Visible = False
  frmMenu fraMenu Visible = False
  frmMenu fraStart Visible = False
  frmMenu fraCalibrationWarning Visible = False
End Sub

Private Sub cmdR_Click()
  frmGrid Cls
  If zin = True Then
    PixOffX = PixOffX + 5 'PixOffX * 1 25
    Call DrawingCps
    Call DrawAll
  Else  FoPixOffX = FoPixOffX + 5 '* 1 25
  End If
End Sub

Private Sub cmdRates_Click()
  If cmdRates Caption = "RATE" Then
    PrintRates = True
    cmdRates Caption = "DOTS"
  Else  PrintRates = False
    cmdRates Caption = "RATE"
  End If
  frmGrid Cls
  If zin = True Then
    Call DrawingCps
    Call DrawAll
  End If
End Sub

Private Sub cmdResume_Click()
  PauseRate = 1
  cmdResume Enabled = False
  cmdPause Enabled = True
  Call ReStart
End Sub

Private Sub cmdU_Click()
  frmGrid Cls
  If zin = True Then
    PixOffY = PixOffY - 5
    Call DrawingCps
    Call DrawAll
  Else   FoPixOffY = FoPixOffY - 5 '/ 1 25
  End If
End Sub

Private Sub cmdZOOM_Click()
  If cmdZOOM Caption = "ZOOM" Then
    cmdZOOM Caption = "HIDE"
    fraZoom Visible = True
  Else  cmdZOOM Caption = "ZOOM"
    fraZoom Visible = False
```

```
    End If
End Sub

Private Sub cmdZOOMin_Click()
    frmGrid Cls
    If zin = False Then
        FoScaleX = FoScaleX * 1 1
        FoScaleY = FoScaleY * 1 1
    Else  XScale = XScale * 1 1
          YScale = YScale * 1 1
          Call DrawingCps
          Call DrawAll
    End If
End Sub

Private Sub cmdZOOMout_Click()
    frmGrid Cls
    If zin = True Then
        XScale = XScale / 1 1
        YScale = YScale / 1 1
        Call DrawingCps
        Call DrawAll
    Else    FoScaleX = FoScaleX / 1 1
            FoScaleY = FoScaleY / 1 1
    End If
End Sub

Private Sub cmdLeft_Click()
    frmGrid Cls
    If zin = True Then
        PixOffX = PixOffX - 5 'PixOffX / 1 25
        Call DrawingCps
        Call DrawAll
    Else    FoPixOffX = FoPixOffX - 5 '/ 1 25
    End If
End Sub


'frmMenu

Private Sub cmdBack1_Click()
    frmMenu fraCalibrationWarning Visible = True
    frmMenu fraQuit Visible = False
    frmMenu fraStart Visible = False
End Sub

Private Sub cmdBACKtoExp_Click()
    frmMenu Visible = False
    frmGrid Visible = True
    frmMenu cmdBACKtoExp Enabled = False
    Call DrawAll
End Sub

Private Sub cmdCalibrate_Click()
    frmMenu Visible = False
    frmCalibrate Visible = True
    frmCalibrate fraMenu Visible = True
    Call LimitsAndSensitivitiesUpdate
    Call HardwareSpecificationsUpdate
    Call CommunicationsUpdate
End Sub

Public Sub cmdCalibrationWarningBack_Click()
    fraCalibrationWarning Visible = False
    fraMenu Visible = True
End Sub

Private Sub cmdCalibrationWarningNext_Click()
    fraCalibrationWarning Visible = False
    fraStart.Visible = True
End Sub

Private Sub cmdDeviceTest_Click()
    frmGrid cmdPause Enabled = False
    frmGrid cmdResume Enabled = True
    Call CommunicationsUpdate
    Call LimitsAndSensitivitiesUpdate
    frmMenu Visible = False
    frmTest Visible = True
    frmTest fraDeviceTest Visible = True
    frmTest fraDeviceTest Enabled = False
    Pause = True
    frmComm tmrPause Interval = 1500
    frmComm tmrPause Enabled = True
End Sub

Private Sub cmdExitBack_Click()
    fraQuit.Visible = False
    frmMenu Visible = True
        frmMenu fraMenu Visible = True
        frmMenu cmdBACKtoExp Enabled = True
End Sub

Private Sub cmdExperimentsNext_Click()
    frmGrid cmdPause Enabled = True
    frmGrid cmdResume Enabled = False
    frmCalibrate cmdNewCalibration Enabled = False
    cmdStart Enabled = False
    Call ExperimentNext
End Sub

Private Sub cmdQuit_Click()
    fraMenu Visible = False
    fraQuit Visible = True
End Sub

Public Sub cmdRunAnother_Click()
    fraQuit Enabled = False
    Pause = True
    frmComm tmrCloseOut Interval = 1500
    frmComm tmrCloseOut Enabled = True
    frmCalibrate cmdNewCalibration Enabled = True
            "disabled in frmmenu cmdExperimentNext
    frmSettings cboPCCardDrive Enabled = True
End Sub

Private Sub cmdSettings_Click()
    frmMenu Visible = False
    frmSettings Visible = True
        frmSettings fraMenu Visible = True
End Sub

Private Sub cmdSTART_Click()
    fraMenu Visible = False
    fraCalibrationWarning Visible = True
    Call LimitsAndSensitivitiesUpdate
    Call HardwareSpecificationsUpdate
    Call CommunicationsUpdate
    Call CalibrationUpdate
    Call FindExperimentFiles
    Call ListFileNames
End Sub

Private Sub filExpFiles_Click()
```

```
    frmMenu Visible = False
    frmSettings Visible = False
    frmGrid Visible = True
    Call MainProgram
End Sub

Private Sub Form_Load()
    frmGrid Visible = False
    frmSettings Visible = False
    frmMenu Visible = True
    frmMenu fraMenu Visible = True
    frmMenu fraStart Visible = False
    frmMenu fraQuit.Visible = False
End Sub


'frmSettings

Private Sub cboAccWarnTrigger_Change()
    cmdLimitsAndSensitivitiesUpdate Enabled = True
End Sub

Private Sub cboApplrBaud_Change()
    cmdCommUpdate Enabled = True
End Sub

Private Sub cboApplrDataBits_Change()
    cmdCommUpdate Enabled = True
End Sub

Private Sub cboApplrPort_Change()
    cmdCommUpdate Enabled = True
End Sub

Private Sub cboApplrStopBits_Change()
    cmdCommUpdate Enabled = True
End Sub

Private Sub cboDataSmoothing_Change()
    cmdLimitsAndSensitivitiesUpdate Enabled = True
End Sub

Private Sub cboDrawUpdate_Click()
    cmdLimitsAndSensitivities2Update Enabled = True
End Sub

Private Sub cboETATimer_Click()
    cmdLimitsAndSensitivities2Update Enabled = True
End Sub

Private Sub cboFeedForwardControl_Click()
    cmdLimitsAndSensitivities2Update Enabled = True
End Sub

Private Sub cboGPSBaud_Click()
    cmdCommUpdate Enabled = True
End Sub

Private Sub cboGPSdataBits_Click()
    cmdCommUpdate Enabled = True
End Sub

Private Sub cboGPSPort_Click()
    cmdCommUpdate Enabled = True
End Sub

Private Sub cboGPSStopBits_Click()
    cmdCommUpdate Enabled = True
End Sub

Private Sub cboGPSUpdateRate_Click()
    cmdCommUpdate Enabled = True
End Sub

Private Sub cboHDOPTrigger_Click()
    cmdLimitsAndSensitivitiesUpdate Enabled = True
End Sub

Private Sub cboHorizontal_Click()
    cmdHardWareSpecificationsUpdate Enabled = True
End Sub

Private Sub cboLBarBaud_Click()
    cmdCommUpdate Enabled = True
End Sub

Private Sub cboLBarDataBits_Click()
    cmdCommUpdate Enabled = True
End Sub

Private Sub cboLBARincr_Click()
    cmdLimitsAndSensitivities2Update Enabled = True
End Sub

Private Sub cboLBarPort_Click()
    cmdCommUpdate Enabled = True
End Sub

Private Sub cboLBarStopBits_Click()
    cmdCommUpdate Enabled = True
End Sub

Private Sub cboLightBar_Click()
    cmdLimitsAndSensitivitiesUpdate Enabled = True
End Sub

Private Sub cboLineThickness_Click()
    cmdLimitsAndSensitivities2Update Enabled = True
End Sub

Private Sub cboLookPeriod_Click()
    cmdLimitsAndSensitivities2Update Enabled = True
End Sub

Private Sub cboMaxGPSError_Click()
    cmdLimitsAndSensitivitiesUpdate2 Enabled = True
End Sub

Private Sub cboMaxSpeed_Click()
    cmdLimitsAndSensitivitiesUpdate Enabled = True
End Sub

Private Sub cboMethod_Click()
    cmdLimitsAndSensitivities2Update Enabled = True
End Sub

Private Sub cboMinSpeed_Change()
    cmdHardWareSpecificationsUpdate Enabled = True
End Sub

Private Sub cboPCCardDrive_Click()
    cmdHardWareSpecificationsUpdate Enabled = True
End Sub
```

```
Private Sub cboRate_Click(Index As Integer)
  cmdRateUpdate Enabled = True
End Sub

Private Sub cboScreenFlip_Click()
  cmdLimitsAndSensitivities2Update Enabled = True
End Sub

Private Sub cboSprayer_Click()
  cmdHardWareSpecificationsUpdate Enabled = True
End Sub

Private Sub cboSprayerSize_Click()
  cmdLimitsAndSensitivities2Update Enabled = True
End Sub

Private Sub cboStopExpTrigger_Click()
  cmdLimitsAndSensitivitiesUpdate Enabled = True
End Sub

Private Sub cboUsedTargetTracking_Click()
  cmdLimitsAndSensitivitiesUpdate Enabled = True
End Sub

Private Sub cboVertical_Click()
  cmdHardWareSpecificationsUpdate Enabled = True
End Sub

Private Sub cmdApplicationRates_Click()
  Call ApplicationRates
End Sub

Private Sub cmdBack_Click()
   Call Back
End Sub

Private Sub cmdCommBack_Click()
  Call CommunicationsBack
End Sub

Private Sub cmdCommDefaults_Click()
   cmdCommUpdate Enabled = True
  Call CommunicationsDefaults
End Sub

Private Sub cmdCommunications_Click()
  Call Communications
End Sub

Private Sub cmdCommUpdate_Click()
  cmdCommUpdate Enabled = False
  Call CommunicationsUpdate
End Sub

Private Sub cmdDisplaySpecificationDefaults_Click()
  cmdHardWareSpecificationsUpdate Enabled = True
  Call HardwareSpecificationsDefaults
End Sub

Private Sub cmdDisplaySpecificationsBack_Click()
  Call HardwareSpecificationsBack
End Sub

Private Sub cmdHardWareSpecificationsUpdate_Click()
  cmdHardWareSpecificationsUpdate Enabled = False
  Call HardwareSpecificationsUpdate
  txtPCCardError Visible = False
End Sub

Private Sub cmdHardWareSpecs_Click()
  Call HardWareSpecs
End Sub

Private Sub cmdLimitsAndSensitivities_Click()
  Call LimitsAndSensitivities
End Sub

Private Sub cmdLimitsAndSensitivities2Defaults_Click()
  Call cmdLimitsAndSensitivitiesDefaults_Click
      cmdLimitsAndSensitivities2Update Enabled = True
End Sub

Private Sub cmdLimitsAndSensitivities2update_Click()
  Call cmdLimitsAndSensitivitiesUpdate_Click
  cmdLimitsAndSensitivities2Update Enabled = False
End Sub

Private Sub cmdLimitsAndSensitivitiesBack_Click()
  fraLimitsAndSensitivities Visible = False
  fraMenu Visible = True
End Sub

Private Sub cmdLimitsAndSensitivitiesBack2_Click
          (Index As Integer)
  fraLimitsAndSensitivities2 Visible = False
  fraLimitsAndSensitivities Visible = True
End Sub

Private Sub cmdLimitsAndSensitivitiesDefaults_Click()
  cmdLimitsAndSensitivitiesUpdate Enabled = True
  Call LimitsAndSensitivitiesDefaults
End Sub

Private Sub cmdLimitsAndSensitivitiesMore_Click()
  fraLimitsAndSensitivities Visible = False
  fraLimitsAndSensitivities2 Visible = True
End Sub

Private Sub cmdLimitsAndSensitivitiesUpdate_Click()
  cmdLimitsAndSensitivitiesUpdate Enabled = False
  Call LimitsAndSensitivitiesUpdate
End Sub

Private Sub cmdRateBack_Click()
  Call RatesBack
End Sub

Private Sub cmdRateDefaults_Click()
  cmdRateUpdate Enabled = True
  Call RatesDefaults
End Sub

Private Sub cmdRateUpdate_Click()
  cmdRateUpdate Enabled = False
  Call RatesUpdate
End Sub
```

VITA

Seth Wrenne Rye was raised in Houston County, Tennessee. He graduated from Houston County High School in 1994. He received a Bachelor of Science (1998) and a Master of Science in Agricultural Engineering (2000) from the University of Tennessee.