

California AHMCT Research Center
University of California at Davis
California Department of Transportation

DEVELOPMENT OF A VEHICLE ORIENTATION AND CONTROL SYSTEM FOR AUTOMATED CRACK SEALING MACHINERY

Jonathan N. Wanjiru

AHMCT Research Report
UCD-ARR- 95-01-15-01

Interim Report of Contract
IA65P307 (SHRP H-107A)

January 15, 1995

* This work was supported by the Strategic Highway Research Program and the California Department of Transportation (Caltrans) Advanced Highway Maintenance and Construction Technology Center at UC, Davis

DEVELOPMENT OF
A VEHICLE ORIENTATION AND CONTROL SYSTEM
FOR
AUTOMATED CRACK SEALING MACHINERY

BY

JONATHAN N. WANJIRU

THESIS

Submitted in partial satisfaction of the requirements for a degree of

MASTER OF SCIENCE

in

MECHANICAL ENGINEERING

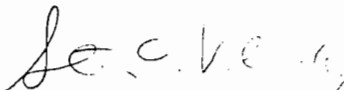
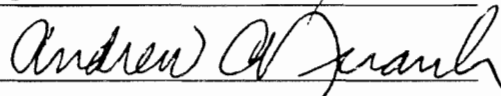
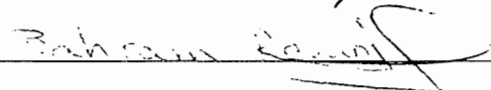
in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA,
DAVIS

Approved:

Committee in Charge

ABSTRACT

The state of California spends over \$100 million annually in highway maintenance and over \$10 million on crack sealing operations specifically. A typical crack sealing crew consists of eight individuals who seal one or two lane miles a day. The cost associated with this is approximately \$1800 per mile of which 66% of the cost are due to labor. To reduce maintenance costs, and to increase efficiency and reliability, work is under way to develop automated road construction and maintenance machines at the Advanced Highway Maintenance and Construction Technology Center at the University of California, Davis. In addition, automated road construction and maintenance machines increase worker health and safety by reducing worker exposure to traffic thus reducing the hazards of accidents and the detrimental health effects associated with breathing exhaust fumes.

One specific highway automation project has involved the design, development and demonstration of an Automated Crack Sealing Machine (ACSM). The goals of this project include minimizing worker exposure to the dangers of traffic, increasing the sealing and maintenance operation speed, improving the quality and consistency of crack seals and ultimately, reducing the cost of highway maintenance. The ACSM identifies cracks, prepares the cracks by removing debris and moisture, routs the cracks and then proceeds to apply sealant on the heated surface.

The purpose of this thesis is to design, develop and demonstrate the Vehicle Orientation and Control system that will track the motion of the vehicle, track the location of cracks identified in the front of the vehicle, and provide the sealing robot with the location of cracks that appear within its work space at the rear of the machine. The Vehicle Orientation and Control system will locate the cracks within a ± 5 centimeters (2 inches) window after the vehicle has traveled approximately 1 meters (35 feet) forward. This thesis includes a detailed literature review, design

description of the Vehicle Orientation and Control system, and test results. Conclusions and Recommendations are made based on performance tests conducted on the integrated ACSM performing all the operations required in the sealing operation. Based on these performance tests, it was concluded that the Vehicle Orientation and Control system meets the design objectives and requirements necessary to track cracks identified in the front of the ACSM until sealed in the rear of the ACSM accurately, consistently and reliably.

EXECUTIVE SUMMARY

The state of California spends over \$100 million annually in highway maintenance. To reduce maintenance costs, and to increase efficiency and reliability, work is under way to develop automated road construction and maintenance machines. In addition, automated road construction and maintenance machines increase worker health and safety by reducing worker exposure to traffic thus reducing the hazards of accidents and health effects associated with breathing exhaust fumes. An example of such a project addressing the automating of highway maintenance is the effort at the University of California, Davis where an Automated Crack Sealing Machine (ACSM) is under development.

Pavement cracking is attributed to road loading, surfacing and sub-grading materials, and environmental factors. Pavement stresses due to daily heating and cooling and vehicle traffic loading, and gradually develops cracks. The cracks gradually get larger and accumulate moisture through to the subgrade. This moisture in the subgrade tends to weaken it and cause it to flow thus further propagating the cracks and resulting in potholes on the surface. Timely repair and sealing of cracks therefore extends pavement life and mitigates expensive pavement repairs.

Research is underway at the Department of Mechanical and Aeronautical Engineering of the University of California, Davis to design, develop and demonstrate an automated crack sealing machine. The goals of this project include minimizing worker exposure to the dangers of traffic, increasing the sealing and maintenance operation speed, improving the quality and consistency of crack seals and ultimately, reducing the cost of highway maintenance. The Davis machine identifies cracks, prepares the cracks by removing debris and moisture, routs the cracks and then proceeds to apply sealant on the heated surface.

The purpose of this report is to design, develop and demonstrate the Vehicle Orientation and Control system that will track the motion of the vehicle, track the location of cracks identified in the front of the vehicle and provide the sealing robot with the location of cracks that appear within its work space in the rear of the machine. The Vehicle Orientation and Control system locates the cracks within a ± 5 centimeters (2 inches) window after the vehicle has traveled approximately 11 meters (thirty-five feet) forward.

A literature review was performed to investigate existing methods of robotics navigation and control. This literature review helped identify conventional methods of tracking vehicles and led to the development of the Vehicle Orientation and Control system. Commercially available position transducers and sensors were selected and used in the development of the prototype crack sealing machine. The Vehicle Orientation and Control system was calibrated and experimentally verified. Tests showed that the system performed reliably and within the requirements of a crack sealing machine under various road conditions and pavements.

Conclusions and recommendations were made based on performance tests conducted on a machine performing all the operations required in the sealing operation. Based on these performance tests, it was concluded that the Vehicle Orientation and Control system meets the design objectives and requirements necessary to track cracks identified in the front of the truck until sealed in the rear of the truck accurately, consistently and reliably. Recommendations for improving the accuracy of the hardware and software and to optimize the performance of the crack sealing machine.

TABLE OF CONTENTS

List of Figures	vii
List of Tables	viii
CHAPTER 1 - INTRODUCTION	1
1.1 - Literature Search	7
1.1.1 - Dead Reckoning Measurement Error Sources	8
1.1.2 - Blanche Vehicle	9
1.1.3 - Sugimoto-Hongo Vehicle	10
1.1.4 - Tsumura Vehicle	12
1.2 - Problem Statement And Objective	14
CHAPTER TWO - VOC CONFIGURATION	15
2.1 - Objectives	15
2.2 - System Requirements	16
2.3 - Hardware Component Descriptions	19
2.3.1 - Fifth Wheels	19
2.3.2 - Encoders	20
2.3.3 - Counter Card	22
2.4 - Software Configuration Description	22
2.5 - Measurement Kinematics Procedure	27
CHAPTER THREE - EXPERIMENTAL VERIFICATION	30
3.1 - Calibration	30
3.2 - Test Setup	31
3.2.1 - Cart Tests	31
3.2.1.1 - Cart Test Objective	32
3.2.1.2 - Cart Test Setup	32
3.2.2 - Truck Tests	35
3.2.2.1 - Truck Test Objective	35
3.2.2.2 - Truck Test Setup	35
3.2.3 - Procedure	38
3.2.3.1 - Mobile Cart Test Procedure	38
3.2.3.2 - Truck Test Procedure	38
3.2.3.3 - Truck Performance Test Procedure	40
3.3 - Data	41
3.3.1 - Mobile Cart Data	41
3.3.2 - Truck Data	42
3.3.3 - Performance Test Data	48
3.4 - Results	49
3.4.1 - Cart Test Results	49
3.4.2 - Truck Test Results	50
3.4.3 - Performance Test Results	54
3.5 - Discussions	56
3.5.1 - Cart Tests	56
3.5.2 - Truck Tests and Performance	57

CHAPTER 4 - CONCLUSION AND RECOMMENDATIONS.....	59
4.1 - Conclusion	59
4.2 - Recommendations.....	59
REFERENCES.....	61
APPENDIX A - VEHICLE ORIENTATION AND CONTROL PROGRAM	63
APPENDIX B - VEHICLE ORIENTATION AND CONTROL HARDWARE TEST PROGRAM.....	87
APPENDIX C - FIFTH WHEEL ASSEMBLY DRAWINGS.....	91
APPENDIX D - MANUFACTURER'S SPECIFICATIONS.....	122

List of Figures

Figure 1.1 - Prototype Automated Crack Sealing Machine	2
Figure 1.2 - A Schematic of the Integration of the VOC with the VSS, RPS, and ICU.	6
Figure 1.3 - Wheel Configuration on the Sugimoto-Hongo Vehicle.....	11
(Sugimoto et. al., 1988).....	11
Figure 1.4 - Arrangement of Sensors on the Sugimoto-Hongo Vehicle.....	11
Figure 2.1 - Placement of Encoders on Crack Sealing Truck.....	17
Figure 2.2 - Process Flow Chart.....	18
Figure 2.3 - Schematic Illustration of Encoder	21
Figure 2.4 - Coordinate Systems of the Various Subsystems.	24
Figure 2.5 - Transformation of Crack Coordinates.	26
Figure 3.1 - Mobile Test Cart Component Schematic.....	33
Figure 3.2 - Cart Flow Chart of VOC Software Operations.....	34
Figure 3.3 - Truck Flow Chart of Software Operations.	37
Figure 3.4 - Plot of Cart Error Measurements vs. Distance Traveled.	50
Figure 3.5 - Truck X Position Error.....	51
Figure 3.6 - Truck Y Position Error.....	51
Figure 3.7 - Truck XY Position Error.....	52
Table 3.12 - Regression Analysis of Truck Test Data.....	53
Figure 3.8 - Truck X Test Position Error without Parameter Estimator.....	54
Figure 3.9 - Truck X Test Position Error with Parameter Estimator.....	54
Figure 3.10 - Truck Y Test Position Error without Parameter Estimator.....	55
Figure 3.11 - Truck Y Test Position Error with Parameter Estimator.....	55
Figure 3.12 - Truck XY Performance Test Position Error.....	56

List of Tables

Table 2.1 - Encoder Requirements.....	21
Table 2.2 - Counter Card Requirements.....	22
Table 3.1 - Cart Tests Hardware Requirements.....	32
Table 3.2 - Truck Tests Hardware Requirements.....	36
Table 3.3 - Mobile Cart Test Data.....	41
Table 3.4 - X Distance Error Data and Analysis.....	42
Table 3.5 - X Distance Error Data and Analysis Continued.....	43
Table 3.6 - Y Distance Error Data and Analysis.....	44
Table 3.7 - Y Distance Error Data and Analysis Continued.....	45
Table 3.8 - θ Heading Error Data and Analysis.....	46
Table 3.9 - θ Heading Error Data and Analysis Continued.....	47
Table 3.10 - X Distance Performance Verification.....	48
Table 3.11 - Y Distance Performance Verification.....	49

CHAPTER 1 - INTRODUCTION

The state of California spends over \$100 million annually in highway maintenance. To reduce maintenance costs, and to increase efficiency and reliability, work is under way to develop automated road construction and maintenance machines (Velinsky, 1993; Ravani, Velinsky and West, 1994; Skibniewski, et. al., 1990). In addition, automated road construction and maintenance machines increase worker health and safety by reducing worker exposure to traffic. An example of such a project addressing the automation of highway maintenance is the effort at the University of California, Davis where an Automated Crack Sealing Machine (ACSM) is under development. The Davis machine identifies cracks, prepares the cracks by removing debris and moisture, routs the cracks and then proceeds to apply sealant on the heated surface.

Pavement cracking is attributed to road loading, surfacing and sub-grading materials and environmental factors such as daily heating and cooling, and moisture (Jing, et. al., 1990; Peshkin, et. al., 1991). Pavement stressed by daily heating and cooling and vehicle traffic loading, gradually develops cracks. The cracks gradually get larger, and accumulate moisture through to the subgrade. This moisture in the subgrade tends to weaken it and cause it to flow thus further propagating the cracks and resulting in potholes on the surface. Timely repair and sealing of cracks therefore extends pavement life and mitigates expensive pavement repairs.

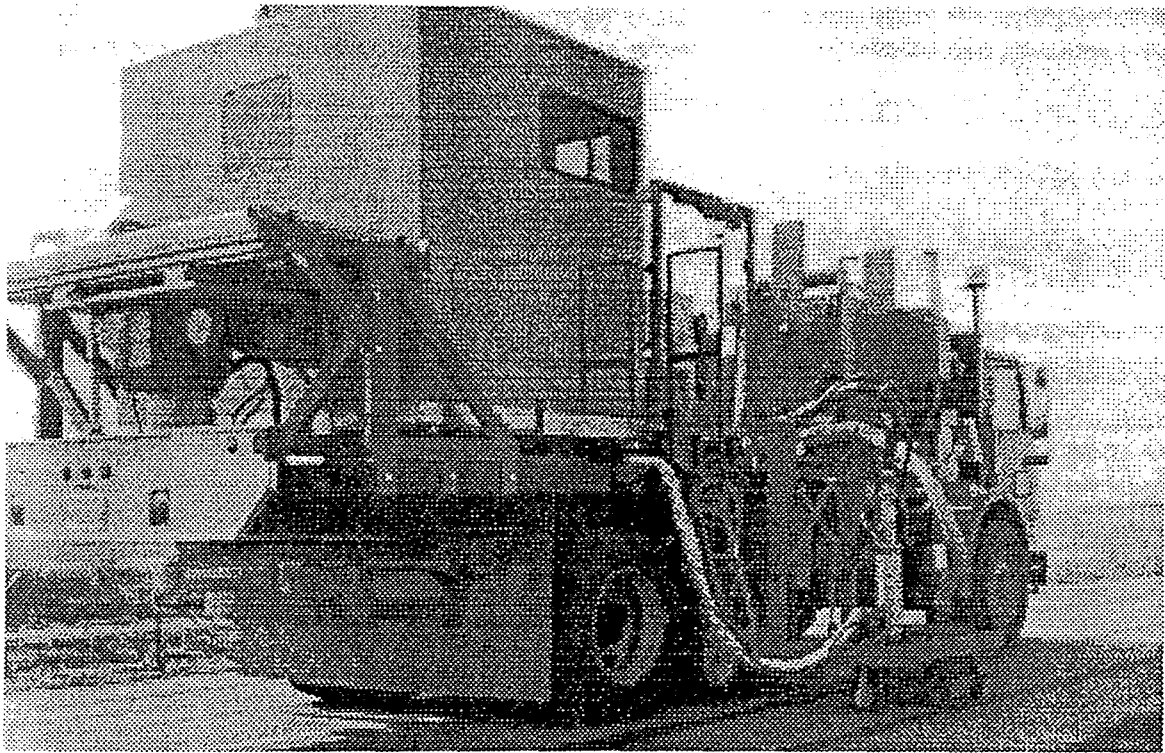


Figure 1.1 - Prototype Automated Crack Sealing Machine

The design objectives of this prototype automated crack sealing machine are to:

- Sense the location and size of cracks which are 3.175 to 25.4 mm wide in an asphalt concrete or Portland cement concrete pavement.
- Prepare cracks for sealing by removing debris and moisture, routing and preheating the pavement surface.
- Apply the sealant.
- Use as many commercially available components as possible.
- Ultimately, perform the crack sealing operation at approximately 3.2 km/h (2 mph).
- Track cracks for at least 11 meters (35 feet) and be within a maximum window of ± 5 centimeters (2 inches) of the crack.

To accomplish these objectives the machine has six operation sub-systems. These sub-systems are:

Vehicle Orientation and Control (VOC)

This sub-system tracks the vehicle's global position and orientation. In addition, the VOC tracks each crack's position relative to each new vehicle position from when each crack is first identified until it is sealed. This enables the Robot Positioning System to position itself over each crack to be sealed. A dead reckoning measurement system is used to monitor the vehicle's position and orientation.

Vision Sensing System (VSS)

This sub-system identifies roadway cracks and tags them with a global locator. It consists of line scan cameras mounted on the front of the truck that scan a line in the roadway 1.60 millimeters (1/16 inches) deep by 3.66 meters (12 feet) wide. Based on these scans, VSS builds a crack picture for the roadway consisting of square pixels for each crack located within a line scan. These pixels are augmented into 5 centimeter (2 inch) crack tiles.

Path Planning (PP)

This sub-system takes VSS data and prepares trajectory paths for the Robot Positioning System. To do this, PP identifies connected cracks and provides an efficient planned path for sealing the connected cracks. The PP program takes into consideration the maximum workspace configuration, plans around any obstacles in the work space and adjusts for both the longitudinal and lateral translations of the vehicle. This subsystem physically resides within the Integration and Control Unit discussed below.

Robot Positioning System (RPS)

This sub-system positions the robot end effector along the crack during sealing. The local sensing sub-system and sealant applicator are mounted upon the robot's end effector assembly. The robot is mounted on the rear of the truck.

Local Sensing System (LSS)

This sub-system works in conjunction with the RPS and further hones in on the crack to be sealed. It uses a laser range finder to compensate for errors between a planned crack path and the actual roadway crack. It is located on the robot end effector assembly and calibrates the crack data obtained from the VOC (via the Path Planning sub-system) to the actual roadway crack.

Applicator and Peripherals System (APS)

This sub-system includes all the equipment necessary to heat and clean debris from cracks, and rout and seal the cracks. The removal of loose material and moisture from the cracks ensures a proper seal that will extend the life of a pavement. The heating assists the bonding of the old pavement to the new sealing material. This operation is necessary to maintain a consistent, high quality seal.

Integration and Control Unit (ICU)

This sub-system coordinates all activities of the sub-systems over the entire sealing operation. ICU initializes all sub-systems, monitors peripherals to ensure proper operation and controls communication between sub-systems. The human interface includes a user friendly control panel with automated operation buttons.

The prototype ACSM is a truck with a line scan camera vision system mounted on the front, and a sealing robot at the rear. Peripheral support such as the sealant tanks, crack heating and cleaning equipment, and the computer system are mounted on the truck bed. In operation, the truck drives down the road at a maximum speed of 3.2 kmph (2 mph) as the VSS detects cracks on the road and tags them with the current vehicle location data obtained from the VOC. The VOC continuously monitors the global position of the vehicle. After the VSS collects a complete buffer of crack data, it signals PP via the ICU to start work on making trajectories for the cracks within the buffer for the RPS. When the RPS is ready to seal, it requests coordinates of cracks within its work space. At this point, the ICU requests the VOC for the current work space coordinates of the robot, and provides these coordinates to PP. PP then prepares trajectories for cracks within the work space and proceeds to send these crack points and trajectories to the VOC. The VOC transforms these points from the VSS coordinate frame to the VOC global coordinate frame to the robot coordinate frame, and the robot positions its end effector to commence sealing along the crack paths provided. The LSS assists the robot in compensating for errors between the planned path and the actual roadway crack.

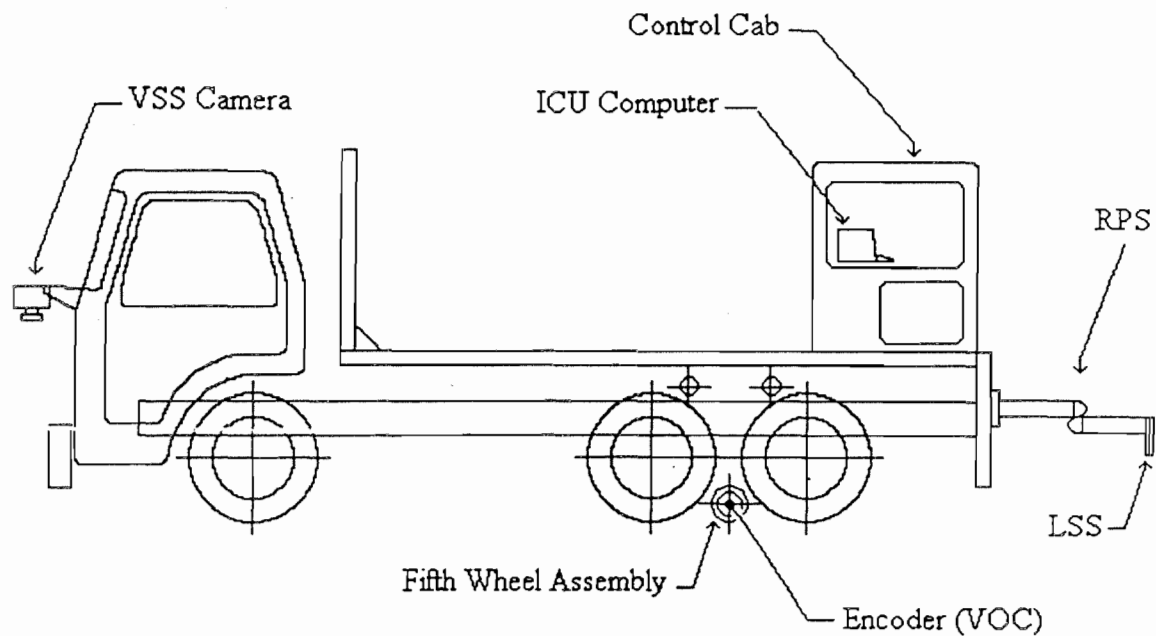


Figure 1.2 - A Schematic of the Integration of the VOC with the VSS, RPS, and ICU.

The benefits of the ACSM include reduced worker exposure to dangerous traffic, increased maintenance and sealing speed (thus lower lane closure times), improved quality and consistency of seals, and reduced highway maintenance costs.

This work is supported by the National Research Council's Strategic Highway Research Program (SHRP) and the California Department of Transportation (CALTRANS). Under this effort an automated highway maintenance machine was developed. The purpose of this thesis was the development and demonstration of the VOC. This sub-system tracks the vehicle's global position and orientation enabling the robot to position itself on cracks to be sealed.

1.1 - Literature Search

A literature search was performed to investigate the tracking and navigation techniques used for automated ground vehicles. The literature review showed that there have been several test vehicles built and in use (Skewis et. al., 1991; Cox et. al, 1990; Sugimoto et. al., 1988; Hongo et. al., 1985; Tsumura et. al., 1985). Most of the vehicles are used in factory-floor type applications to move toxic materials, hospital wares, mail and so on. These vehicles use at least two sensors (such as optical encoders, optical range finders, sonar, embedded metal bars, etc.) due to the inherent errors associated with tracking the position of a vehicle, the need for path planning, navigation and collision avoidance. These vehicles are generally designed to follow memorized paths or command-specified paths, and the main tasks are to actively track the vehicle's position and orientation, and control the error given a specified destination.

The ACSM follows a random path, and no navigation control is integrated into the VOC system at this time. In this sense, the VOC passively tracks the vehicle heading and orientation. The dead reckoning technique is used to measure position and heading. Dead reckoning is a real-time method of measuring a mobile vehicle's position using wheel sensors. This measuring technique is advantageous for this application because no off-the-vehicle support equipment is needed for position or orientation estimation. The major disadvantage is that the measurement error is cumulative and in order to track over long distances (>50 meters) a correction system needs to be incorporated into the tracking system. In this application, no secondary measuring system is used because the distances traveled before re-orienting the vehicle are about 11 meters (35 feet) and the cumulative errors in preliminary tests have been less than 0.5 percent, the maximum acceptable error.

For the dead reckoning measurements optical wheel encoders are attached to the left and right sides of the truck on constant pressure, non-driving fifth wheels. The reason for using fifth wheels is to minimize the error that results from drive wheel slippage and radial changes due to

load changes. The wheels are made of hard neoprene rubber that has minimal radial dimensional changes.

Dead reckoning systems have cumulative errors that increase without bound the further a vehicle travels. For most applications dead reckoning is of limited value in accurately locating the global position of a vehicle. The literature has several articles addressing this problem and cites techniques to reduce the error or to self-adjust the error (Banta, 1988; Hongo et. al., 1985; Tsumura et. al., 1983). The techniques, for example, include the use of the dead reckoning optical encoders and a laser triangulation to reduce the cumulative error (Sugimoto et. al., 1988; Tsumura et. al., 1985). A Kalman filter is used to merge data from the two sensors and have corrected position data (Cox et. al., 1990). Another technique is the use of a constant correction factor calculated experimentally for different road surfaces and conditions (Hongo et. al., 1985). Another technique is the development of a parameter estimation model using a least squares technique (Banta, 1988). This method takes care of dead reckoning errors due to wheel encoder misalignments, calibration and wheel wear.

1.1.1 - Dead Reckoning Measurement Error Sources

The measurement error due to dead reckoning is discussed by several authors including Cox et. al., 1990; Chatila et. al. 1985; Smith et. al., 1986 and Wang, 1988. The key sources of this error have been identified as:

1) Road Way Surface Roughness And Undulations

This error results in overestimating the distance traveled or the heading changed. In the situations where the ACSM will operate, this error may be significant. Dead reckoning is best suited to a two-dimensional traveling space such as factory floors where the undulations are minimal. In a road application, where three-dimensional travel is the case, errors will result because while traveling the same distance horizontally, different vertical distances are traveled by each wheel. In

this case, the dead reckoning system may overestimate the distance traveled and the heading change.

2) Wheel Slippage

This error results in underestimating of the distance traveled or the heading change. The counter card records distance or heading change when the encoders are rotated. When the wheel slips, no distance or heading change is recorded. This error is minimized by having the encoders attached to separate non-driving fifth wheels.

3) Changes In The Radius Of The Measuring Wheel

Under or over estimating the number of encoder pulses per revolution results in additional measurement errors. The number of counts per revolution are linearly related to a specific radius. If the radius decreases due to a difference in the vehicle's tire load, there will be an underestimation of the distance traveled and the heading change. This error is minimized by calibrating the wheels to get an effective wheel radius. This procedure is described in the calibration section. For this project the wheel radius was calculated to be 20.24 centimeters (7.97 inches). To control this error, hard neoprene rubber measuring wheels which have low radial change were used instead of pneumatic tires.

The literature has several examples of automated or semi-automated vehicles that track position and orientation. For this thesis I will summarize three cases: the Blanche vehicle (Cox, et. al., 1990), the Sugimoto-Hongo vehicle (Sugimoto, et. al., 1988) and the Tsumura vehicle (Tsumura, et. al., 1982).

1.1.2 - Blanche Vehicle

This is an autonomous battery-powered vehicle designed to operate in a structured environment such as an office or factory. The control computer is a Motorola 68020 based system operating

on real-time UNIX. The vehicle is a tricycle configuration with a single front steering drive wheel and two passive rear wheels. Encoders are attached to each rear wheel. The vehicle uses an optical range finder and "odometry" (i.e. a dead reckoning encoder system) to get accurate position and orientation data. The vehicle's computer algorithm matches range finder data to a 2-D memory map of its environment to correct and compensate for errors existing in its dead-reckoning position estimate. The estimated current position and range finder data are used to find a world coordinate point and are matched to a map location.

As previously discussed, dead reckoning position estimates drift because of small cumulative errors and must be periodically corrected. With this range finder compensation system the vehicle can move about in an office or factory environment without the need for off-vehicle correction references. The vehicle, however, must have a stored map of its environment. In addition to the stored maps, the Blanche vehicle further minimizes encoder errors that result from wheel radius changes by using a pair of knife-edge, constant pressure (i.e. non-load bearing) wheels that have minimal radial change.

1.1.3 - Sugimoto-Hongo Vehicle

This is another example of an autonomous battery-powered vehicle designed to operate in a structured environment such as an office or factory. The control computer is a Motorola 68000 based system. The vehicle configuration is illustrated below. This vehicle uses three sensors (optical encoders, optical range finders and sonar) to locate itself and adjust for position or orientation measurement errors. The vehicle is designed to operate inside buildings and has two independent rear drive wheels, two front castor wheels and two independent measuring wheels attached co-axially to the drive wheels. The optical encoders are attached to the measuring wheels and provide rotation speed information. Dead reckoning is used to estimate the vehicle's position and orientation. Errors of about 10 centimeters (4 inches) over a distance of 50 meters (164 feet) were reported for road tests on asphalt.

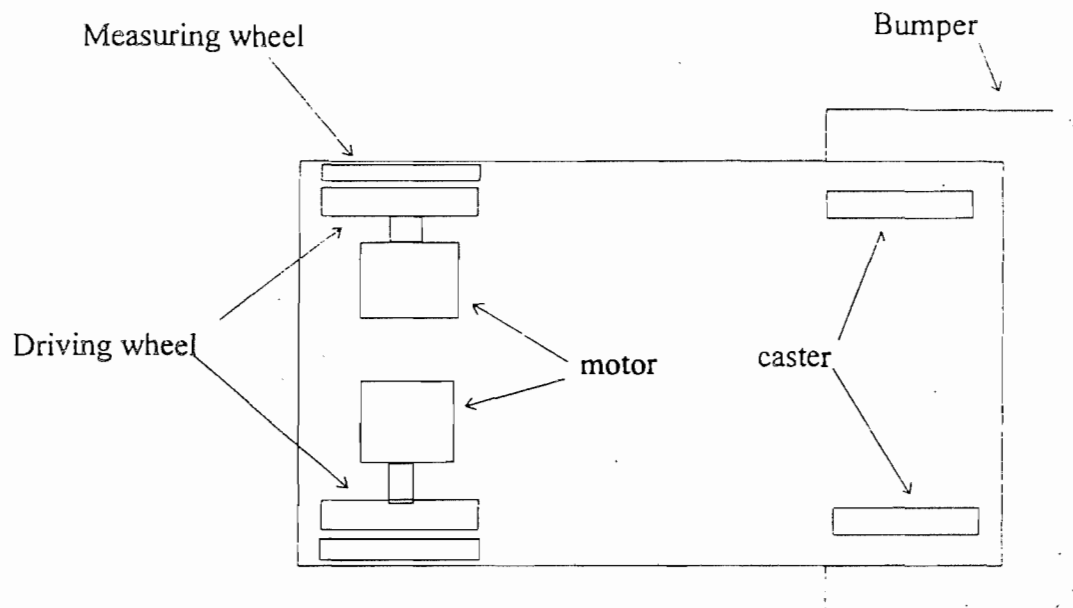


Figure 1.3 - Wheel Configuration on the Sugimoto-Hongo Vehicle.
(Sugimoto et. al., 1988)

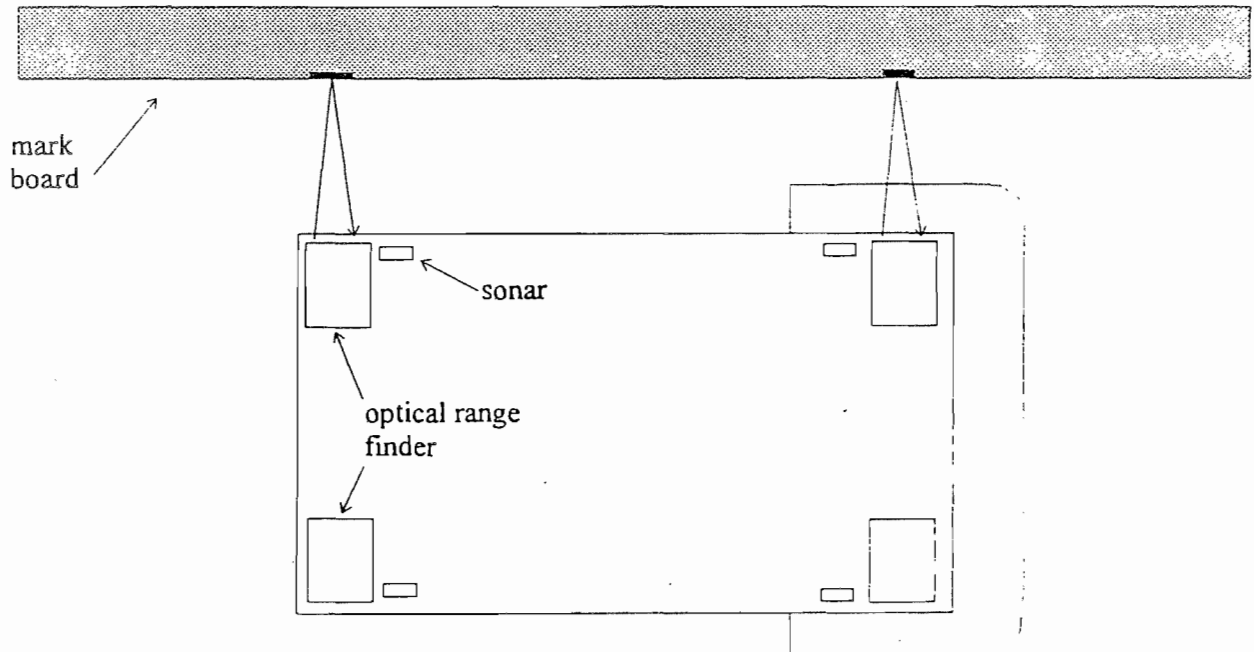


Figure 1.4 - Arrangement of Sensors on the Sugimoto-Hongo Vehicle.
(Sugimoto et. al., 1988)

Optical range finders are attached to both rear and front sides of the vehicle. Mark boards are attached to the wall along the path the vehicle travels. Each optical range finder emits an infrared light beam onto the mark board and, using optical triangulation, determines its position. This determination is used to correct the position errors from the dead reckoning system. The errors from this system are reported to be less than 1 millimeter (0.04 inch) over 5 meters (15 feet).

Sonar is installed next to the optical range finders and is used to provide position information where no mark boards are installed or where the vehicle misses reading a mark board. The sonar emits an ultra sonic pulse and, based on the time interval it takes to detect a pulse reflected off a wall, the distance traveled can be determined. The sonar readings are used to determine the vehicle's position and correct the dead reckoning system. The errors from this system are reported to be ± 5 millimeters (0.20 inches). The authors expect to further develop the sonar component of their measuring system because it is limited to following paths along walls.

1.1.4 - Tsumura Vehicle

This vehicle most closely resembles the working environment and application of the automated crack sealing machine. The authors used a conventional 4-wheel front-drive wagon-type automobile. In this project the authors developed a self-contained position and heading measuring method and tested it by driving on asphalt roads. The control computer was a Z-80 based system. Encoders were attached to the vehicle's pneumatic wheels, and the position and heading were monitored as the vehicle traveled on a random path. The goal was to drive the vehicle on a random path, continuously monitoring its world coordinates and to be able to terminate the trip where the vehicle started. Measurements were made to determine how far the vehicle was at the end of the trip from where it started, i.e. the difference between the calculated and measured position and orientation.

Because the automated crack sealing machine's path is random, and the sealing operation needs to be autonomous, one of our criterion was that the position and orientation measuring be independent of ground support systems. In other words, the measurement system cannot depend on road markers or off-vehicle signals to correct for measurement errors. The Tsumura project most closely meets the application of the ACSM because the measurement system is contained on the vehicle and does not depend on off-the-vehicle devices.

Their experiments showed that the dead reckoning system develops significant errors due to tire radius changes and heading drift. The wheel radius changes because the vehicle weight distribution changes with load, cornering forces, temperature and centrifugal acceleration and this results in the rubber tires deforming. The method used to reduce this error is to experimentally determine an effective tire constant. Using displacement detectors and an effective tire constant, a dynamically corrected tire radius is used to reduce the errors due to radius changes. This method was found effective for distances of up to about 50 meters (164 feet).

Errors due to heading drift were controlled using terrestrial magnetic sensors. The heading drift results from limitations of instantaneous encoder readings and this results in a small cumulative error. Gyroscopes and magnetic sensors have been used to control this error and are effective for distances of up to 25 kilometers (15.5 miles).

Tsumura, et. al. (1985) in a later paper discuss the use of lasers to improve their on-vehicle measuring system. Even with the radius compensation and heading magnetic sensors they found that the position and orientation errors accumulate unbounded and become too large for accurate vehicle location. Therefore, it becomes necessary to have fixed ground references to correct the on-vehicle dead-reckoning location system. A laser compensation system was located on the vehicle which utilized cubes located along the road to provide periodic position and orientation corrections.

1.2 - Problem Statement And Objective

The objective of this thesis is the development and demonstration of a vehicle orientation and tracking system that works in conjunction with a crack sealing robot, and allows the measurement of the vehicle's position and relative position of cracks on a road surface with high accuracy. This is a prototype development and demonstrates the limitations of dead-reckoning and the need to have additional techniques to track the vehicle. The development involves a detailed literature search of each area, development of the VOC software and hardware, the development of a small scale cart platform which was used for testing/verification of the VOC kinematics equations, and finally, the development, implementation and testing of the VOC on the ACSM.

CHAPTER TWO - VOC CONFIGURATION

The purpose of this chapter is to give a general description of the VOC's configuration. This includes a description of the tracking configuration, the measurement sensors configuration, the hardware configuration and the software configuration. Before discussing the different components, a general description of the VOC's objectives and related topics will be presented.

2.1 - Objectives

The objectives of the VOC are as follows: tracking global vehicle position and heading, tracking VSS identified road cracks, providing PP with information on the location of the current robot work space and providing the robot information on the location of cracks within its work space. The VOC system tracks the position of the cracks on the road surface with respect to a fixed point (e.g. the robot base) located on the truck. This system is required so that the position of road cracks identified by the VSS at the front of the ACSM can be tracked continuously as the truck moves forward, and until the cracks enter the work space of the robot arm in the rear of the truck. At that point, the crack position data will be sent to the controller on the robot arm via the ICU. The robot controller then sends signals to position the end effector of the robot arm over the cracks. Once positioned, the end effector routs, cleans, heats, and seals the identified cracks. At this stage of development, the sealing operation is done in a stop-and-go process with the vehicle stopping for each work space block. However, it is our intent that ultimately the entire process of identifying the cracks, tracking their positions, positioning the robot end effector, and performing the crack repair operation will be done in a continuous fashion with the truck moving ahead at a slow forward speed of about 3.2 kmph (2 mph). A schematic illustration of the crack sealing truck and the integrated VOC, VSS, RPS, and ICU is given in Figure 1.2.

The task of tracking points on a crack would be simple if the truck were moving straight ahead or if it were turning a corner of constant radius. However, this is usually not the case. To illustrate this point, it is noted here that after a particular point on a crack is identified by the VSS, the truck must move ahead approximately 11 meters (35 feet) before that same point on the crack appears in the work space of the robot arm. Throughout this distance it is unlikely that the truck will maintain a straight heading or even a turn of constant radius. Instead it is very likely that the truck will be moving with many combinations of left and right turns (of various radii) and straight ahead motion. The heading variations may be small but they will be significant enough to have a large impact on the accurate positioning of the robot end effector.

The ACSM will be used primarily on asphalt and/or concrete roadways which are considered fairly smooth and flat. For purposes of this project, only a two-dimensional orientation and tracking case is considered. While this may introduce errors, these errors are assumed to be small.

2.2 - System Requirements

The VOC system consists of two (2) optical, rotary, incremental encoders, an up/down counter card, and a computer. Each encoder is mounted on a separate "fifth wheel" assembly. The fifth wheel assemblies are mounted on either side of the truck outside of the rear wheels and midway between the two rear axles of the truck so that the center line of the encoders passes through the center of rotation of the truck (see Figure 2.1). The fifth wheel assemblies will be located approximately 2.67 meters (8.75 feet) apart. Each encoder continuously monitors the change in position of each side of the truck. Using counter pulse data obtained from the encoders, the translation of the truck (laterally with respect to the road surface) and rotation of the truck (yaw - about a vertical axis with respect to the road surface) can be used to continuously calculate the vehicle's position and heading change.

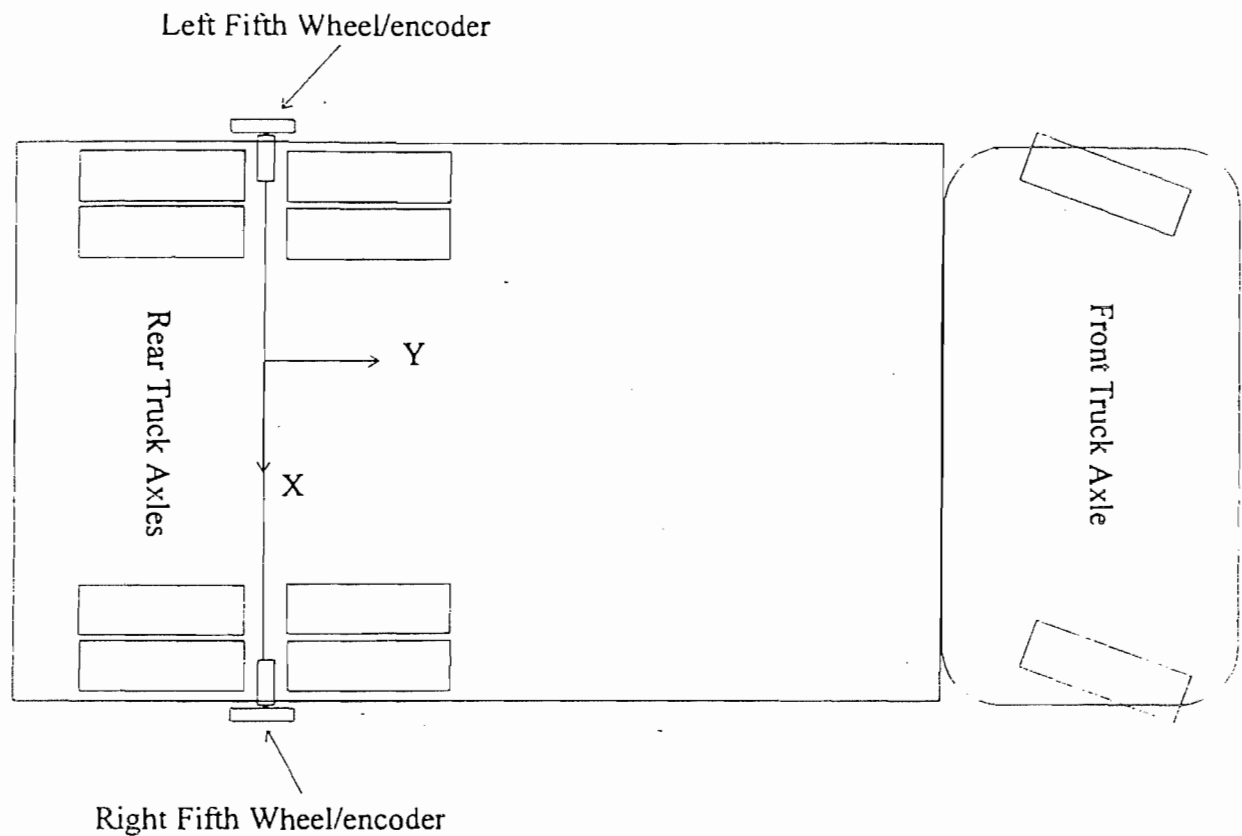


Figure 2.1 - Placement of Encoders on Crack Sealing Truck.

Crack position data is sent to the VOC from the VSS (Count Data Module) via the PP (Path Planning Data Module) and ICU, and truck position data is gathered from the encoders on the fifth wheels. The encoder output is noise filtered before being sent to the VOC. Calculations of truck position and orientation, and crack position (with respect to the truck) are performed by the VOC and the data is made available upon request to the RPS via the ICU (VOC Data Module). The VOC determines the location of robot work space given the global position and orientation of the vehicle. Once the work space is identified, VOC sends PP this information, and receives sealable cracks within the current work space when PP has completed planning crack paths. Based on the initial crack locations, VOC sends RPS the current locations of those cracks in the work space via the VOC data module. Figure 2.2 below is a flow chart of this process.

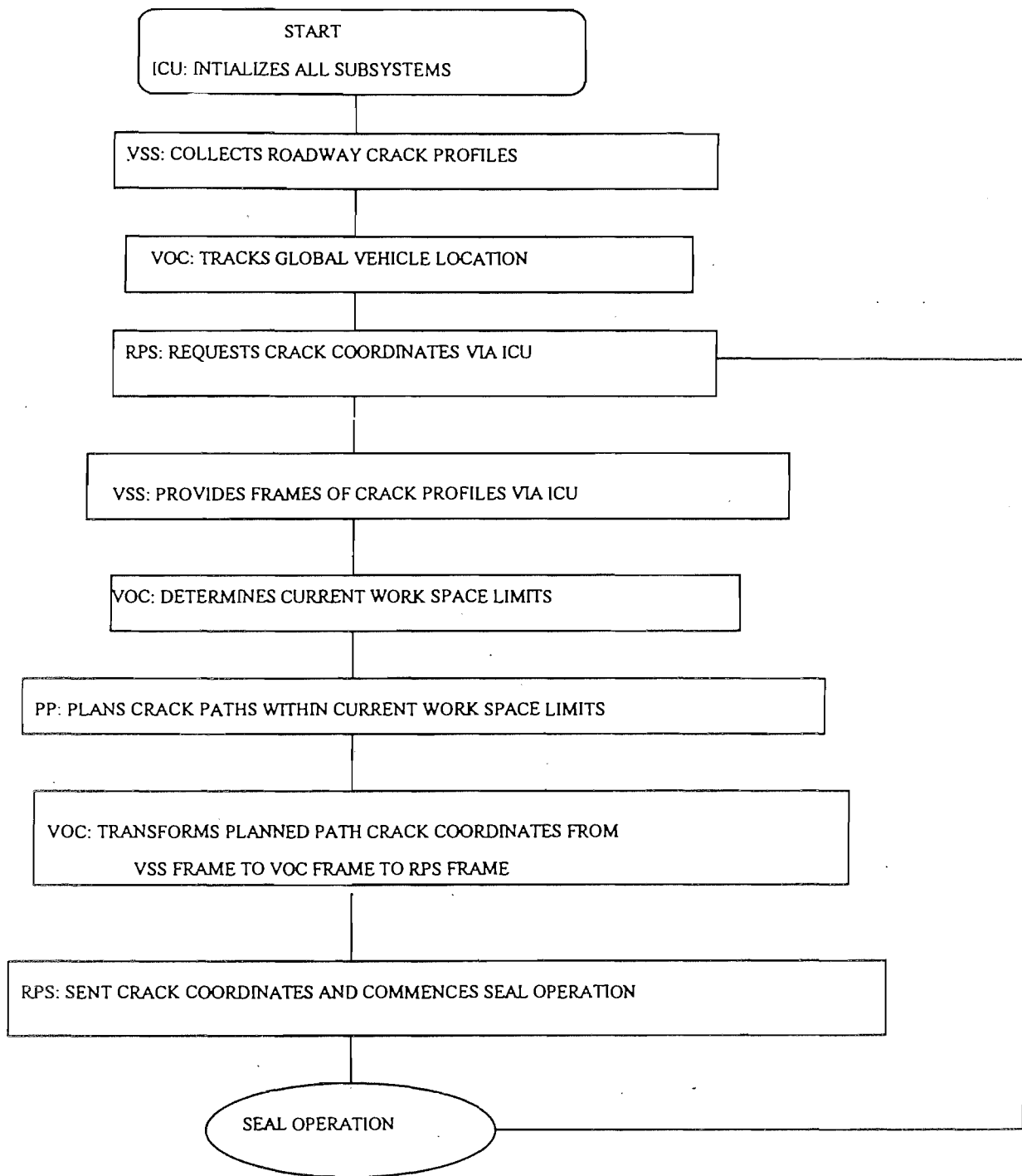


Figure 2.2 - Process Flow Chart.

Specifically, the VOC system performs as follows. The Cartesian reference frame used by the VOC computer program is fixed on the truck at the center of the rear drive wheels (i.e. at the center of the axis connecting the fifth wheels) as shown in Figure 2.1. This reference frame

moves (in position and orientation with respect to the road surface) as the truck moves. The VOC system receives (from the VSS) the position, perpendicular to the direction of travel of the truck (with respect to the truck), of points on cracks as they are identified by the VSS camera at the front of the truck. The VOC system receives a left and right encoder stamp with each crack data indicating the location of the vehicle at the time when the crack was identified by VSS. The VOC monitors the change in global position and orientation of the truck continuously by sampling count data (at a set time frequency) from the right and left encoders simultaneously. When a crack appears within the work space of the RPS and is ready to be sealed, the VOC performs a coordinate transformation of points on the crack with respect to the truck's current position, and transforms the coordinates from the VSS to the VOC frames and then into the RPS frame of reference. This information is sent to the RPS (via the VOC Data Module) so that the RPS end effector can rout, clean, heat, and seal the crack. Once the crack repair operation is completed on that particular portion of crack, that crack segment is no longer tracked by the VOC system. Refer to Figure 3.3 for a flow chart of the operation and integration of the VOC with the VSS, RPS, and ICU.

2.3 - Hardware Component Descriptions

The VOC hardware components consist of the fifth wheels, encoders and the counter card.

2.3.1 - Fifth Wheels

The fifth wheels are constant-pressure, non-driving measuring wheels. The fifth wheel assembly consists of the following components:

- an aluminum hub with a hard neoprene rubber tread that rolls on the road surface and thus turns the encoder,
- an encoder for converting the rotary input to digital, electrical outputs, and
- all linkage, fastening components, and electrical cables necessary for attaching the wheel and encoder to the crack sealing truck and ICU.

Refer to Figures 1.2 and 2.1 for a schematic illustration of the integration and location of the two fifth wheel subassemblies on the ACSM. Complete detailed lists of parts and components necessary to build the 5th wheel subassemblies are provided in Appendix C, drawings SHRP(VOC)M-A100, SHRP(VOC)M-A200, and SHRP(VOC)M-B100.

2.3.2 - Encoders

A simplified optical, rotary encoder consists of three major parts. These include:

- a disk with radial line slits,
- a light source and,
- a light detector.

As the disk rotates, light is transmitted between radial line slits to the light detector which generates a triangular-wave voltage output. This output is fed to a comparator which transforms it to a square-wave voltage. The square wave is then fed to an up/down counter which is triggered by the leading edge of the square wave. The encoders each output A, B and Z pulses as shown on Figure 2.3. The encoder requirements for this project are shown on Table 2.1.

Two incremental pulse encoders are used as the position transducers and provide information on the vehicle's incremental position and heading. To meet the encoder requirements specified below, two encoders were utilized. The encoders have a resolution of 2500 pulses per revolution but have been modified to output 800 pulses per revolution in order to integrate with VSS encoder and timing requirements. This results in a position error of 0.159 centimeters (0.0625 inches) and a heading error of 0.45 degrees per pulse.

Table 2.1 - Encoder Requirements.

ENCODER RESOLUTION (LINES PER REVOLUTION)	2500
MINIMUM DISTANCE RESOLUTION	0.0625" (1.588 mm)
ENCODER OUTPUT	5V
ENCODER SUPPLY VOLTAGE	5Vdc
ENCODER OUTPUT FORMAT	channels A , B and Z
ENCODER CABLES	shield, twisted pair
FIFTH WHEEL RADIUS (nominal)	8" (203 mm)
TEMPERATURE	0-70° C
HUMIDITY	100% RH
VIBRATION	5-2000 Hz
SHOCK	20G

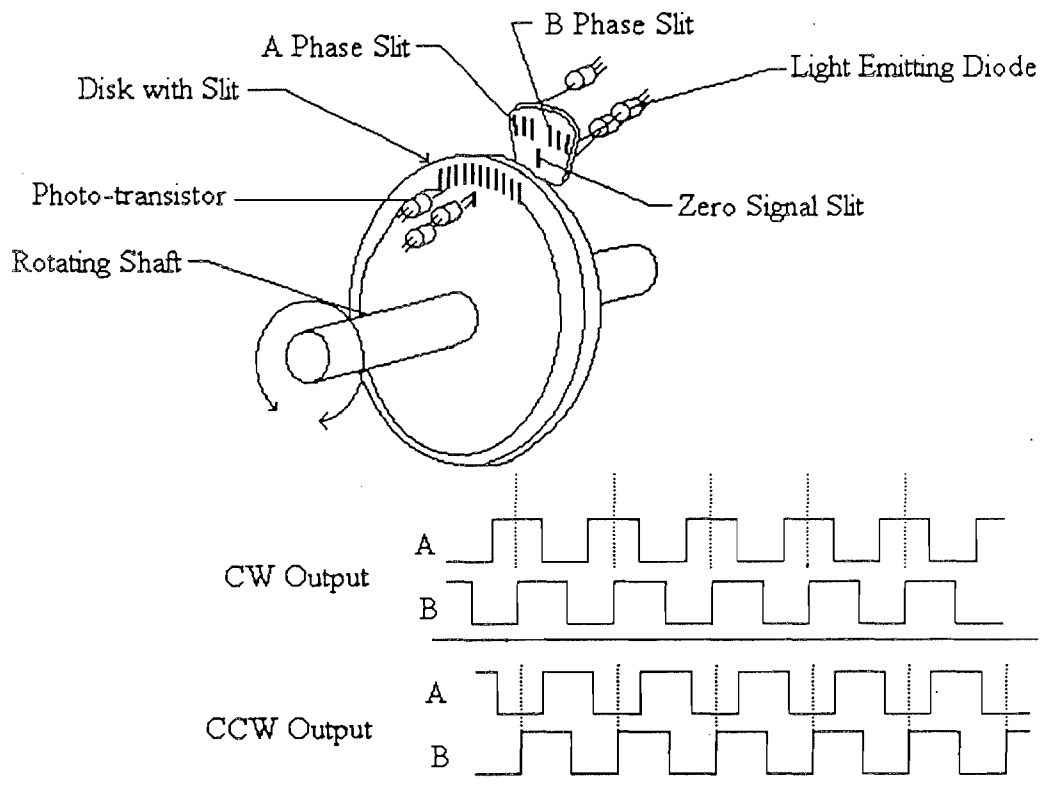


Figure 2.3 - Schematic Illustration of Encoder Light Source, Light Detector and Encoder Disk with Radial Line Slits. (Figure adapted from EME 251 - Mechatronics class notes with permission of Dr. Yamazaki.)

2.3.3 - Counter Card

An up/down counter card is installed on the back plane of the computer and records the outputs of the encoders. Each encoder outputs an A, B and Z signal. The A and B channels of each encoder output the same frequency pulse with a 90 degree phase shift, and enable the counter card to detect if the vehicle is moving forward or backward. The Z signal enables the counter card to track the number of revolutions the wheels have turned. To meet the counter card requirements specified below, a Xycom 230 Intelligent Counter Module was purchased and installed on the VME computer back plane. The Xycom counter card has eight, thirty-two bit counters. The counter card requirements for this project are shown below in Table 2.2.

Table 2.2 - Counter Card Requirements.

MINIMUM NUMBER OF COUNTING CHANNELS	6 up and down channels
BUS COMPLIANCE	VME compatible bus
COMPUTER	Heurikon HK68/VE3
MINIMUM INPUT FREQUENCY	2kHz
POWER REQUIREMENTS	5V
TEMPERATURE	0-65° C
HUMIDITY	95% RH
VIBRATION	2.5 G peak acceleration
SHOCK	30G peak acceleration

2.4 - Software Configuration Description

The VOC system software performs three functions: 1) reads the number of recorded pulses on the counter card, 2) calculates the vehicle's incremental position and heading based on the number of cumulated pulses, and 3) provides the RPS crack locations upon request. The new position and heading are calculated in the program using kinematics schemes described by Tsumura, et. al.(1982) and modified by Wang (1988). These kinematics are described in detail later under Section 2.5. When the RPS requests new crack locations via ICU, the program executes a matrix

transformation routine that provides an updated crack location based on the current vehicle location and the initial crack location provided by the VSS.

The software was written in the C programming language. Original implementation and development was on a IBM PC-compatible running the DOS operating system and using a Metrabyte DAS 5 Counter Card. Stationary tests and cart tests were used to verify the workings of the computer code. The latest implementation is on a Heurikon computer running the OS-9 Professional operating system and using a Xycom 230 Intelligent Counter Card. Cart tests and truck tests were used to verify and debug the final working version of the computer code.

The following descriptions are of the main program, the header file and the key VOC functions. The source code is attached as Appendix A.

VOCRUN.C:

This is the main VOC program which initializes the counter card, starts the links to the data modules, and sets up the process signaling. In its integrated configuration, this program is forked by the ICU, and operates as a subprogram under ICU. Its operations are transparent to the user when the debug mode is off. This program keeps a record of the vehicle's current global position and heading, the current robot work space and the location of cracks within that work space.

The machine is tracked in the global coordinate frame by this program. The origin of this coordinate frame coincides physically with the middle of the left-right encoder lateral axis when this program is first executed by the ICU. This origin location is shown on Figures 2.4 and 2.5.

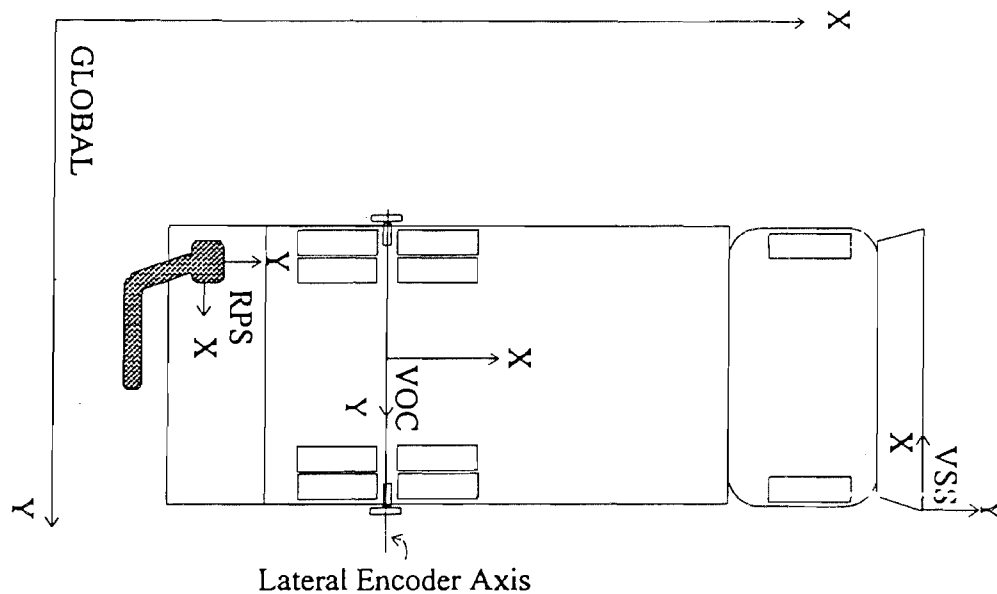


Figure 2.4 - Coordinate Systems of the Various Subsystems.

XVME230M.H

This header file contains all the functions used to perform the operations of tracking the vehicle and the cracks. The functions include `WORLDJW1(LEFT_COUNT, RIGHT_COUNT, X_CUR, Y_CUR, THETA_CUR)`, `WS_CORD()`, `TRANSFORM_PP_DATA()`, `DISPLAY_VOC_MODULE()`, and the Counter Card Interface/Operation functions.

`WORLDJW1(LEFT_COUNT, RIGHT_COUNT, X_CUR, Y_CUR, THETA_CUR):`

This function performs the VOC kinematics and determines the incremental global position and heading. The program updates vehicle location and heading periodically by using the current counter card data to calculate the incremental change and then adding these changes to the global position and heading.

`WS_CORD():`

This function determines what the coordinates of the current work space should be. This information is used by the path planning sub-system to determine which cracks are in the work

space and then to plan a trajectory for those cracks. The function is executed when the path planning sub-system signals VOC with WS_CORD_REQUEST.

Upon receiving this signal, VOC determines the vehicle's current global position and the maximum and minimum rows and columns that would fit the robot's work space. The change in Y position between the present truck position, and the previous truck length frame is used to determine the columns' shift. The X position is similarly used to determine the back of the truck area that would correspond to the maximum and minimum rows. A rectangular work space is thus determined, with maximum row and column, and minimum row and column. PP is signaled with WS_CORD_READY and transmitted the work space data via the WS_DM data module.

TRANSFORM_PP_DATA():

This function performs a matrix transformation on the cracks within the planned path. The transformation is shown below. Using the vehicle's current global position (X_i and Y_i) and the global position (X_{i-1} and Y_{i-1}) the vehicle was in when a crack was first identified by VSS, the function provides the location of the crack within the VOC coordinate frame (P_i) and then transforms it again into the robot's coordinate frame (P_{RPS}) current robot work space. These transformations are shown on Figure 2.5.

This function uses the crack tag data to determine the global position of the crack by transforming the crack tag position data (X_{i-1} and Y_{i-1} - which are really the vehicle's VOC coordinate frame data when the crack was first identified) into the VSS coordinate frame, and subsequently into the global coordinate frame (P_{i-1}).

RPS Y-distance from the robot base to the crack, X_ROBOT is the fixed RPS X-distance from the robot base to the VOC origin on the truck, Y_ROBOT is the fixed RPS Y-distance from the robot base to the VOC origin on the truck, and θ_i is the current global vehicle heading.

DISPLAY_VOC_MODULE():

This function displays the VOC data module containing the TRANSFORM_PP_DATA() output provided to RPS. It is available only in VOCRUN.C's debug mode.

COUNTER CARD FUNCTIONS:

These are the functions needed to operate the Xycom 230 Intelligent Counter Card. The functions include CHKCMD(MODE, CHANNEL, CMDBLK), READ_COUNTER(CHANNEL, CMDBLK), START_COUNTER(CHANNEL, CMDBLK) and RS230(CHANNEL, CMDBLK).

These functions can use any of the eight channels on the Xycom counter card.

RS230(CHANNEL, CMDBLK) is a software reset for the counter card.

2.5 - Measurement Kinematics Procedure

The algorithms for measuring distance and heading are adopted from a measuring system proposed by Tsumura, et. al. (1982) for determining vehicle position and heading using wheel encoders. The procedure consists of the following steps:

1. Compute the distance traveled during one pulse.

$$\frac{2 \times \pi \times \text{Radius}}{\text{Counts_per_Revolution}} \quad (2.3)$$

- D_l and D_r

2. Compute the number of counts recorded during the sampling period.

$$\begin{aligned}N_l &= N_{l2} - N_{l1} \\N_r &= N_{r2} - N_{r1}\end{aligned}\tag{2.4}$$

3. Compute the distance traveled.

$$\begin{aligned}\Delta L_l &= D_l \times N_l \\ \Delta L_r &= D_r \times N_r\end{aligned}\tag{2.5}$$

incremental distance traveled: $\Delta L = \frac{\Delta L_l + \Delta L_r}{2}$

(2.6)

incremental heading change: $\Delta\theta = \frac{\Delta L_l - \Delta L_r}{Wheel_Track}$

(2.7)

4. Compute current heading.

$$\theta_n = \theta_o + \Delta\theta\tag{2.8}$$

5. Compute current position.

$$\begin{aligned}X_n &= X_o + \Delta L * \cos(\theta_o + \Delta\theta/2) \\ Y_n &= Y_o + \Delta L * \sin(\theta_o + \Delta\theta/2)\end{aligned}\tag{2.9}$$

where X_0 and Y_0 are the previous position; θ_0 the previous heading; X_n and Y_n are the current position, and θ_n the current heading.

CHAPTER THREE - EXPERIMENTAL VERIFICATION

In Chapter 2, the Vehicle Orientation and Control system (VOC) hardware and software requirements were established. Initial cart tests were performed which proved that the hardware and software were integrated and functioning properly and that the measurement kinematics were implemented. In Chapter 3, the operation of the dead reckoning system will be experimentally verified and the errors characterized for "typical" operating conditions for the ACSM.

Before the VOC components can be experimentally verified the system must be calibrated. The calibration procedure consists of empirically determining the measuring wheel radius and the vehicle wheel track.

3.1 - Calibration

The purpose of calibration is to determine the effective wheel radius and the wheel track. This calibration procedure is adopted from Tsumura, et. al. (1982). The calibration procedure is as follows:

1. Move the vehicle a straight distance L , and record the cumulative left and right encoder counts.
2. For the distance traveled, determine the distance per encoder pulse:

$$D_l = \frac{L}{N_l} \qquad D_r = \frac{L}{N_r} \qquad (3.1)$$

Steps (1) and (2) were repeated several times and then averaged to get an effective radius which in our case was determined to be 20.00 centimeters (7.97 inches).

3. Move the vehicle along a curved path and record the heading change, $\Delta\Theta$, and the cumulative left (N_l) and right (N_r) encoder counts.
4. Using D_l and D_r values obtained from (2), determine

$$\Delta\Theta = \Theta, \quad \Delta L_r = D_r \times \sum N_r, \quad \Delta L_l = D_l \times \sum N_l \quad (3.2)$$

5. Substituting these values in the following equation, obtain the wheel track, T .

$$T = \frac{(\Delta L_l - \Delta L_r)}{\Delta\Theta} \quad (3.3)$$

Steps (3), (4) and (5) were repeated several times and then averaged to get the wheel track which was determined to be 284.0 centimeters (111.7 inches) for the cart configuration and 266.7 centimeters (105.0 inches) for the truck configuration.

3.2 - Test Setup

Tests were performed on a mobile cart and on the ACSM. The test setup for each is described in the following sections.

3.2.1 - Cart Tests

In order to perform the experimental verification for the cart setup, the hardware and software listed in Table 3.1 was used. The system connections are schematically shown in Figure 3.1. The power supply is 110 AC provided by a standard outlet. The counter card is plugged into a standard VME bus in the back plane of the Heurikon computer. A shielded, six pair twisted cable runs from each encoder to the counter card. The encoders require a 5 VDC power supply which is provided from the computer back plane via the data input cable.

3.2.1.1 - Cart Test Objective

The objective of the mobile cart tests was to verify that the VOC hardware and software correctly track position. Critical to the operation of the crack sealing machine is an accurate estimation of the global location of the truck, the location of cracks on the road surface and the tracking of these cracks as the vehicle moves forward and the cracks come within the robot work space. The purpose of the VOC is to keep track of the vehicle relative to the located cracks, to provide path-planning with the robot's current work space coordinates and to provide the robot with the current location of cracks within its work space.

3.2.1.2 - Cart Test Setup

The hardware and software requirements are specified in Table 3.1 and shown in Figure 3.1. This setup was used to perform the mobile cart VOC tests. Two incremental pulse encoders were used as the position transducers, and, provided information on the cart's incremental position and heading. The mobile cart had the encoders attached co-axially to the back, non-steering wheels and was pulled on a concrete shop floor as the computer recorded the encoder output. The encoders have a resolution of 2500 pulses per revolution. This results in a position error of 0.02 inches and a heading error of 0.14 degrees per pulse. Four of the Xycom 230 card's thirty-two bit counters were used in these tests. The output of A and B channels from each encoder are connected to separate counter channels. The A and B channels of each encoder output the same frequency pulse with a 90 degree phase shift, and could be used to detect if the vehicle is moving forward or reverse.

Table 3.1 - Cart Tests Hardware Requirements.

1.	Heurikon Computer
2.	Xycom 230 Intelligent Counter Card
3.	Mobile Test Cart
4.	Fifth Wheel Assemblies
5.	2 Optical Encoders
6.	VOC program

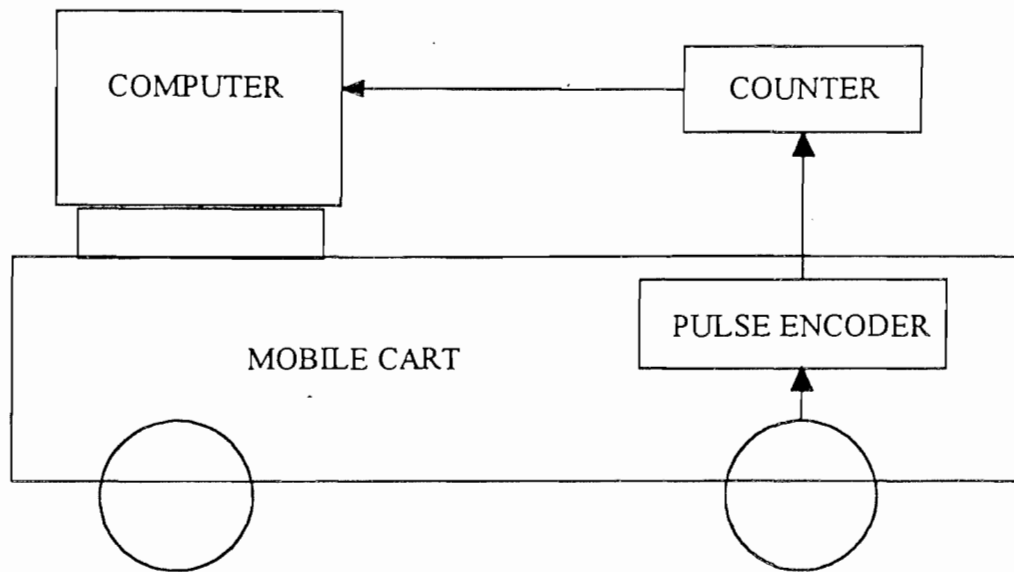


Figure 3.1 - Mobile Test Cart Component Schematic.

The mobile cart VOC software consists of a C program with four main functions that perform the following:

- Reads the encoder output recorded by the counter card;
- Calculates the incremental distance and heading change;
- Accumulates the incremental distances and headings to compute the current global position and heading, and;
- Records the magnitude of the distance traveled.

The program flow chart is shown in Figure 3.2, and illustrates the relationships of the four main functions.

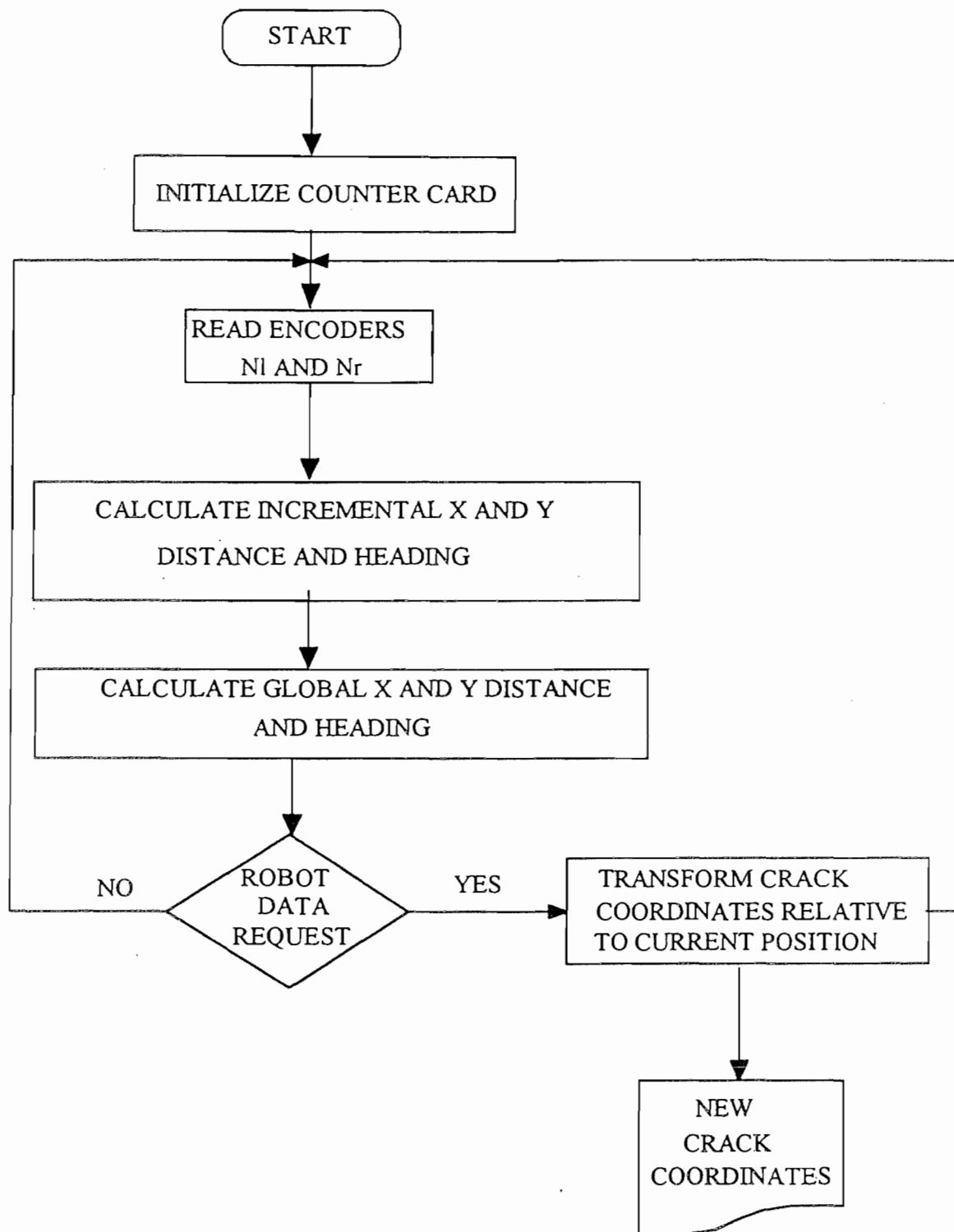


Figure 3.2 - Cart Flow Chart of VOC Software Operations.

3.2.2 - Truck Tests

Field tests on the VOC were performed to verify the operation of the integrated VOC hardware and software on the truck. The overall system configuration is shown on Figure 3.3 and schematically in Figure 1.2. Similar to the cart tests, the counter card is plugged into a standard VME bus in the back plane of the Heurikon computer. A shielded, six pair twisted cable runs from each encoder to the counter card. In these tests, the functions of the VOC are transparent to the user because the VOC is a sub-program under the ICU program. The VOC was operated under its debug mode to capture test results and program output.

3.2.2.1 - Truck Test Objective

The objectives of the truck tests were two-fold: 1) to verify the integration of the VOC software to the ICU program and, 2) to assess the measuring error from the VOC hardware and software and as a result to develop an error correction model. Measuring errors result from road conditions, misalignment of the encoder wheels, wheel slip, encoder resolution and time lag between communicating ICU subsystems. Once the measuring error was determined, a self-adjusting model could be developed to reduce these errors and compensate for systematic sources such as encoder misalignment and time lag.

3.2.2.2 - Truck Test Setup

The hardware and software requirements are specified in Table 3.2. A metal pointer was attached to the middle of the front of the truck in the line scan cameras' track. This provided a point of reference so that a spot pointed to in the front could be identified later when the truck had moved forward. The marked crack could be measured accurately in the before and after truck reference frames, and repeated consistently over different test tracks without having to find the VSS line scan edges.

Table 3.2 - Truck Tests Hardware Requirements.

1.	Heurikon Computer
2.	Xycom 230 Intelligent Counter Card
3.	Truck
4.	Fifth Wheel Assemblies
5.	2 Optical Encoders
6.	VSS hardware
7.	ICU program
8.	VSS program
9.	VOC program

The program flow chart is shown in Figure 3.3, and illustrates the relationships between the ICU, VOC, PP, RPS and VSS subsystem programs in the integrated configuration. Communication between subsystems is facilitated by the ICU.

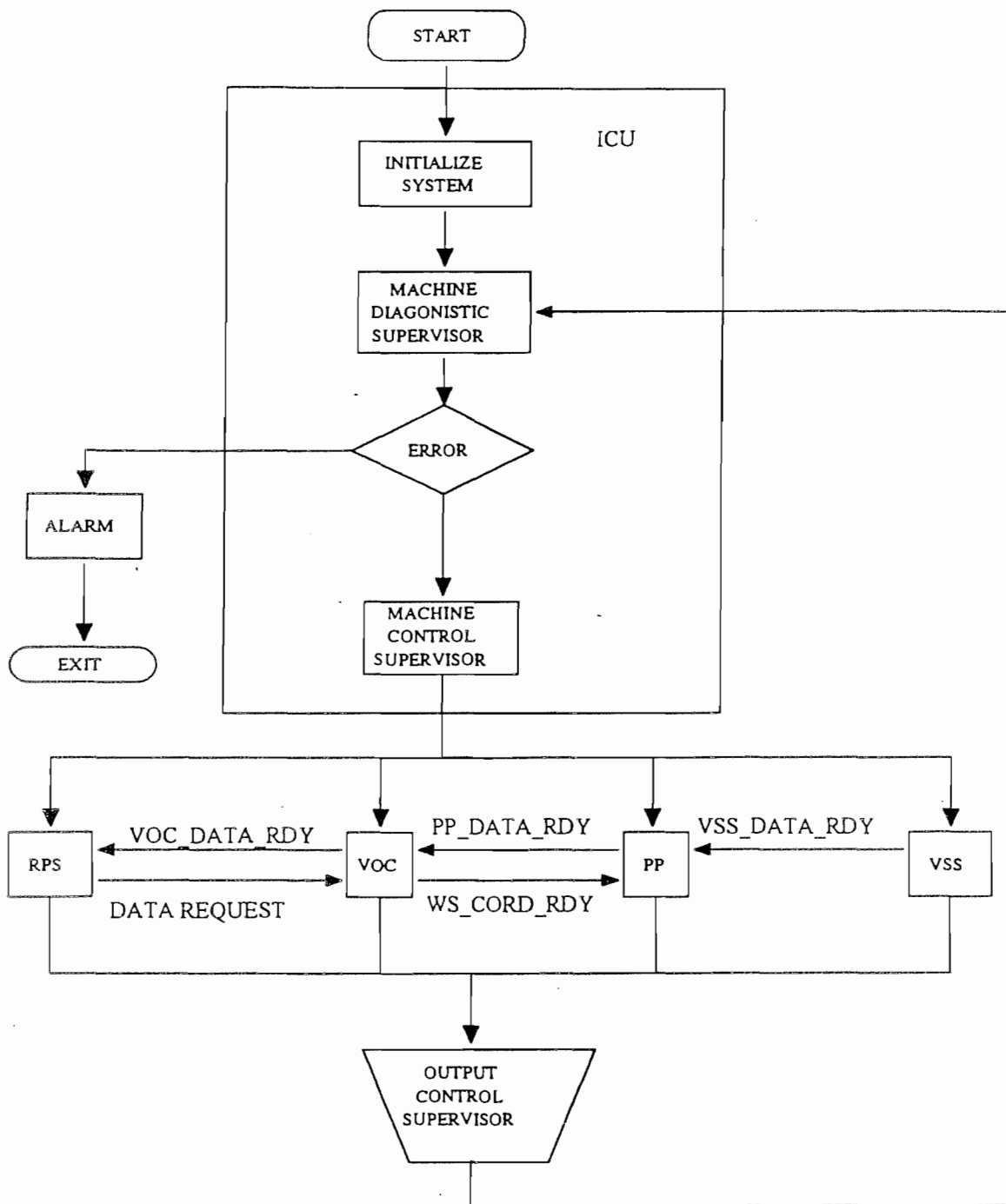


Figure 3.3 - Truck Flow Chart of Software Operations.

3.2.3 - Procedure

3.2.3.1 - Mobile Cart Test Procedure

Several paths were followed and the magnitude of the VOC calculated distances compared to the magnitude of the measured floor distances. The procedure consisted of the following steps:

1. Connect the hardware as shown in Figure 3.1.
2. Mark on the shop floor the middle point of the rear axle to indicate the starting position.
3. Start the measuring system program (VOCRUN.EXE).
4. Pull the cart along a random path for about 2.5 meters (8 feet).
5. Mark on the shop floor the middle point of the rear axle to indicate the end position.
6. Measure and record the floor distance between the start and end positions.
7. Record the VOC measuring system program's calculated distance and compare that to the distance measured on the floor.

3.2.3.2 - Truck Test Procedure

These tests were conducted in different locations, on varying road surfaces and on different days using different drivers. The objective was to test the truck in as realistic road conditions as possible. Various paths and vehicle turns were also tested. The truck test procedure consisted of the following steps:

1. Attach the pointer to the middle of the front of the vehicle, and then measure the perpendicular distance between the pointer location and the right and left encoder axis.
2. Initialize the VSS subsystem and verify that cracks are being detected.
3. Run the VOC test program (VOCTEST.C - shown as Appendix B) and verify that the hardware is connected and functioning properly. One way to do this is to rotate each encoder wheel one revolution, and to verify that there are approximately 800 counts per revolution.

4. Mark the location of the pointer in the front of the vehicle
5. Start the ICU program. This program runs all operations and coordinates the functions of all subsystems. The functioning of the VOC is transparent to the user at this point. The ICU builds user interface screens. When the screen operations are completed, move the screen pointer to the POWER ON button and the GENERAL MACHINE button pressing the mouse each time.
6. Drive the truck forward and stop when the marked point appears within the robot work space. This is approximately a three feet swath around the robot base in the rear of the truck. The path over which the truck drives can be straight or include turns, but the truck must stop when the marked spot is still within the robot work space. The driver needs to pay special attention to the driving speed and road cracks because crack processing takes time, and if there are many cracks and the vehicle speed is too high, cracks will be missed and not be identified by the machine.
7. Mark the new location of the right and left encoders and measure the distance between each encoder wheel and the pointer spot within the robot work space. These measurements are of the sides of a triangle. Based on this information, we can calculate the location of the spot from the right-left encoder axis.
8. The VOCRUN.C in debug mode provides the user the calculated location of the spot from the right-left encoder axis at the end of each run. Record and compare the measured and calculated spot locations.
9. Steps 4-8 were repeated over different road surfaces and conditions, varying turns, on different days and using different drivers. This allows for an experimental design that blocks for these variability factors and allows us to assess the error due to the measuring system software and hardware.

3.2.3.3 - Truck Performance Test Procedure

1. Based on the results of the truck test procedure, a least squares model was developed. This model was then used to re-program the TRANSFORM_PP_DATA() subroutine to develop a self-tuning VOCRUN.C program. This program now adjusts for hardware misalignments and any systematic errors present in the VOC hardware and software configuration.
2. The procedure described under the truck test procedure was then repeated using the self-tuning VOCRUN.C version.

One of the purposes of the truck tests was to estimate the error due to fifth wheel wear, misalignment and calibration degradation. There are several different methods of determining the error estimate used in robotics applications. One method by Chatilia and Laumond uses a scalar error estimator for determining the global position of a mobile robot. They use a multi-sensory approach to determining position. Redundant measurements are made to develop upper and lower error bounds and a gaussian error model. Errors determined to be inconsistent with the error model are handled by a separate control system that has a more global perspective of the robot's locations. The main application of their robot appears to be the maneuvering around obstacles and this approach may have limited value in an ACSM application. Angular errors are not considered in this application, but must be considered in an ACSM application because heading errors are the most significant source of positional error.

Another method is that used by Smith and Cheeseman using a covariance analysis. Their key emphasis is the increased errors resulting from compounding small coordinate transformations into larger displacements and the resulting increased spatial uncertainty. They also use multi-sensors such as acoustic range sensors and a vision system to determine a mobile robot's position. If the errors associated with each sensor are within certain bounds, the merging of different sensor

measurements is done, and mapped back to a world coordinate frame. This methodology may be applicable to a future ACSM where multi-sensors should be considered.

The error analysis method used for this project is a least squares analysis similar to one done by Banta (1988) for an autonomous mobile robot application. The least square estimation gives a biased estimate of the global position and angular heading but is robust and easily implemented on a computer. The computer processing time is a critical factor because the system has many real-time processes occurring concurrently and the encoder reading must be done within a short time frame. The model developed is used to self-tune the measurement data obtained from the dead reckoning system.

3.3 - Data

Data was acquired using the procedures listed in Section 3.2.3. The data collected are shown in this section.

3.3.1 - Mobile Cart Data

The data contained in Table 3.3 consists of ten cart runs. Each run shows the magnitude of the distance traveled, and compares the ground measurements to the VOC measuring system values.

Table 3.3 - Mobile Cart Test Data.

Observation	Measured Distance (inches)	Calculated Distance (inches)	Difference (inches)	Percent Difference
1	60.813	60.670	0.143	0.235
2	55.063	55.112	-0.050	0.090
3	116.438	116.506	-0.068	0.058
4	135.750	135.813	-0.063	0.046
5	87.750	87.548	0.202	0.230
6	89.875	89.882	-0.007	0.008
7	121.375	121.231	0.144	0.118
8	143.875	143.287	0.588	0.409
9	124.125	124.221	-0.096	0.077
10	120.625	120.690	-0.065	0.054

3.3.2 - Truck Data

The data shown below was collected over four days at four different locations and different road surfaces. Each measurement includes X and Y truck displacements and a θ heading change; and are shown as Tables 3.4 through 3.9. The corrected data is shown under the column "Error Correction Model Estimates".

Table 3.4 - X Distance Error Data and Analysis.

Expt. Block	Road Measures	Raw VOC Data	Difference between Road and Raw VOC	Block Error Model Estimates	Difference between Road and Block Model Estimates	Error Correction Model Estimates	Difference between Road and Error Model Estimates
1/6/94	-68.208	-68.295	0.087	-68.264	0.056	-68.137	-0.071
	-131.190	-131.296	0.105	-131.264	0.074	-131.093	-0.097
	-180.672	-180.804	0.132	-180.742	0.071	-180.615	-0.056
	-171.872	-172.006	0.134	-171.951	0.079	-171.867	-0.005
	-93.344	-93.377	0.033	-93.353	0.009	-93.291	-0.053
	-163.005	-162.479	-0.526	-162.448	-0.556	-162.383	-0.622
	-119.242	-118.669	-0.572	-118.636	-0.605	-118.573	-0.669
1/7/94	-108.369	-108.406	0.037	-108.276	-0.094	-108.316	-0.054
	-174.494	-174.817	0.323	-174.574	0.080	-174.649	0.155
	-99.352	-100.194	0.842	-99.999	0.647	-100.052	0.699
	-41.965	-42.554	0.589	-42.506	0.541	-42.518	0.553
	-152.046	-152.551	0.505	-152.575	0.529	-152.584	0.538
	-181.886	-182.416	0.529	-182.305	0.418	-182.334	0.448
	-109.941	-110.014	0.073	-109.970	0.029	-109.986	0.044
	-110.704	-110.543	-0.162	-110.586	-0.118	-110.590	-0.114
	-108.561	-109.343	0.781	-109.235	0.674	-109.267	0.706
	-86.077	-86.318	0.241	-86.138	0.061	-86.181	0.104
	-94.659	-94.938	0.279	-94.938	0.279	-94.686	0.027

Table 3.5 - X Distance Error Data and Analysis Continued.

Expt. Block	Road Measures	Raw VOC Data	Difference between Road and Raw VOC	Block Error Model Estimates	Difference between Road and Block Model Estimates	Error Correction Model Estimates	Difference between Road and Error Model Estimates
1/11/94	-53.914	-53.533	-0.381	-53.653	-0.261	-53.635	-0.279
	-109.635	-109.709	0.075	-109.667	0.033	-109.695	0.060
	-74.856	-74.274	-0.582	-74.254	-0.602	-74.274	-0.582
	-93.269	-93.029	-0.239	-93.018	-0.250	-93.039	-0.230
	-65.670	-65.374	-0.296	-65.288	-0.382	-65.329	-0.341
	-67.598	-67.486	-0.113	-67.408	-0.190	-67.445	-0.154
	-95.041	-95.344	0.302	-95.191	0.149	-95.258	0.216
	-51.987	-52.051	0.064	-51.978	-0.009	-52.011	0.025
	-74.286	-73.692	-0.593	-73.632	-0.654	-73.664	-0.622
	-65.211	-65.554	0.343	-65.485	0.274	-65.517	0.306
	-177.820	-177.849	0.030	-177.665	-0.155	-177.736	-0.083
1/23/94	-115.350	-115.107	-0.243	-114.960	-0.390	-114.928	-0.422
	-108.931	-108.279	-0.652	-108.224	-0.707	-108.207	-0.723
	-106.975	-106.903	-0.072	-106.849	-0.126	-106.833	-0.142
	-113.798	-113.225	-0.573	-113.182	-0.616	-113.167	-0.631
	-116.913	-116.320	-0.593	-116.262	-0.651	-116.245	-0.668
	-111.244	-110.749	-0.495	-110.694	-0.550	-110.678	-0.566
	-112.764	-112.350	-0.414	-112.308	-0.456	-112.293	-0.470
	-118.843	-118.205	-0.638	-118.180	-0.663	-118.168	-0.675
	-131.508	-131.771	0.264	-131.789	0.281	-131.792	0.284
	-151.487	-150.844	-0.643	-150.642	-0.844	-150.610	-0.877

Table 3.6 - Y Distance Error Data and Analysis.

Expt. Block	Road Measures	Raw VOC Data	Difference between Road and Raw VOC	Block Error Model Estimates	Difference between Road and Block Model Estimates	Error Correction Model Estimates	Difference between Road and Error Model Estimates
1/6/94	-4.333	-3.834	-0.499	-4.351	0.018	-6.018	1.760
	-4.764	-4.198	-0.566	-5.083	0.318	-8.406	3.788
	-4.803	-3.115	-1.688	-5.639	0.836	-8.815	4.212
	-3.545	-1.753	-1.792	-4.705	1.160	-7.136	3.779
	-3.246	-1.244	-2.002	-2.457	-0.789	-4.194	1.051
	-3.578	-0.506	-3.072	-3.189	-0.389	-5.599	2.199
	-3.486	-1.211	-2.275	-3.062	-0.424	-4.938	1.583
1/7/94	-5.751	-1.028	-4.723	-5.411	-0.340	-4.542	-1.088
	-9.313	-2.222	-7.091	-9.481	0.168	-7.970	-1.147
	-7.078	-2.747	-4.331	-6.815	-0.263	-6.002	-0.965
	-4.198	-0.476	-3.722	-2.080	-2.119	-1.805	-2.346
	-2.988	3.414	-6.403	-2.062	-0.926	-1.236	-1.586
	-2.635	0.053	-2.689	-6.357	3.722	-5.448	3.010
	-1.985	0.737	-2.722	-3.193	1.208	-2.608	0.743
	-2.139	3.242	-5.381	-0.965	-1.174	-0.230	-1.787
	-4.353	-0.632	-3.721	-4.894	0.541	-4.108	-0.123
	-4.590	-2.945	-1.645	-6.301	1.711	-5.686	1.191
	-4.923	-6.724	1.801	-6.724	1.801	-9.641	1.150

Table 3.7 - Y Distance Error Data and Analysis Continued.

Expt. Block	Road Measures	Raw VOC Data	Difference between Road and Raw VOC	Block Error Model Estimates	Difference between Road and Block Model Estimates	Error Correction Model Estimates	Difference between Road and Error Model Estimates
1/11/94	1.325	4.160	-2.835	2.114	-0.789	2.532	-1.149
	-3.396	1.237	-4.633	-3.274	-0.121	-2.155	-1.121
	-1.816	1.154	-2.970	-2.096	0.280	-1.180	-0.555
	-3.369	1.765	-5.134	-2.268	-1.101	-1.151	-2.116
	-3.446	-0.385	-3.060	-3.378	-0.068	-2.464	-0.910
	-3.507	-0.230	-3.276	-3.238	-0.269	-2.360	-1.072
	-4.304	-1.204	-3.101	-5.528	1.224	-4.227	0.028
	-3.646	-0.418	-3.227	-2.787	-0.859	-2.070	-1.518
	-3.190	0.247	-3.437	-2.990	-0.20	-2.070	-1.039
	-2.450	-0.148	-2.303	-3.016	0.566	-2.207	-0.171
	-3.872	-0.935	-2.937	-8.156	4.283	-6.415	2.737
1/23/94	-6.493	-3.662	-2.831	-6.875	0.382	-7.390	1.025
	-3.660	-0.546	-3.114	-3.499	-0.161	-3.973	0.433
	-3.607	-0.543	-3.064	-3.446	-0.162	-3.912	0.422
	-3.50	-0.040	-3.460	-3.115	-0.385	-3.608	0.233
	-3.131	-0.578	-2.553	-3.710	0.579	-4.213	1.210
	-3.006	-0.486	-2.520	-3.505	0.499	-3.989	1.105
	-3.329	-0.024	-3.306	-3.070	-0.259	-3.559	0.353
	-2.871	0.688	-3.559	-2.519	-0.352	-3.034	0.293
	41.244	41.173	0.071	41.116	0.127	41.107	0.224
	60.474	60.430	0.044	60.931	-0.457	61.011	-0.446

Table 3.8 - θ Heading Error Data and Analysis.

Expt. Block	Road Measures	Raw VOC Data	Difference between Road and Raw VOC	Block Error Model Estimates	Difference between Road and Block Model Estimates	Error Correction Model Estimates	Difference between Road and Error Model Estimates
1/6/94	-0.003	0.004	-0.007	-0.003	0.0	-0.028	0.025
	0.002	0.006	-0.004	-0.001	0.002	-0.026	0.028
	-0.017	-0.007	-0.009	-0.001	0.005	-0.039	0.022
	-0.024	-0.013	-0.010	-0.030	0.007	-0.044	0.021
	-0.027	-0.005	-0.021	-0.018	-0.008	-0.037	0.010
	-0.031	-0.012	-0.019	-0.028	-0.002	-0.043	0.013
	-0.029	-0.010	-0.019	-0.026	-0.003	-0.042	0.013
1/7/94	-0.029	0.015	-0.044	-0.026	-0.003	-0.018	-0.011
	-0.014	0.026	-0.041	-0.015	0.001	-0.007	-0.008
	-0.027	0.017	-0.044	-0.024	-0.003	-0.016	-0.011
	-0.104	-0.014	-0.089	-0.052	-0.052	-0.046	-0.058
	-0.076	-0.033	-0.042	-0.069	-0.006	-0.064	-0.012
	-0.056	-0.041	-0.015	-0.076	0.020	-0.071	0.015
	-0.060	-0.035	-0.025	-0.071	0.011	-0.066	0.006
	-0.059	-0.011	-0.049	-0.049	-0.011	-0.042	-0.017
	-0.035	-0.001	-0.035	-0.040	0.004	-0.033	-0.003
	-0.021	-0.002	-0.019	-0.040	0.020	-0.033	0.013
	-0.008	-0.027	0.019	-0.027	0.019	-0.057	0.049

Table 3.9 - θ Heading Error Data and Analysis Continued.

Expt. Block	Road Measures	Raw VOC Data	Difference between Road and Raw VOC	Block Error Model Estimates	Difference between Road and Block Model Estimates	Error Correction Model Estimates	Difference between Road and Error Model Estimates
1/11/94	-0.089	-0.036	-0.053	-0.075	-0.014	-0.067	-0.022
	-0.065	-0.023	-0.042	-0.064	-0.001	-0.054	-0.011
	-0.050	-0.010	-0.039	-0.054	0.004	-0.042	-0.008
	-0.067	-0.012	-0.055	-0.056	-0.012	-0.044	-0.024
	-0.047	-0.001	-0.046	-0.047	0.0	-0.033	-0.014
	-0.055	-0.006	-0.048	-0.051	-0.004	-0.038	-0.017
	-0.035	-0.003	-0.033	-0.048	0.013	-0.034	-0.001
	-0.064	-0.002	-0.062	-0.048	-0.016	-0.034	-0.030
	-0.055	-0.009	-0.046	-0.053	-0.002	-0.041	-0.014
	-0.046	-0.010	-0.036	-0.054	0.008	-0.042	-0.004
	-0.042	-0.025	-0.017	-0.066	0.024	-0.056	0.014
1/23/94	-0.010	0.014	-0.024	-0.014	0.004	-0.018	0.008
	-0.033	-0.004	-0.028	-0.031	-0.001	-0.036	0.003
	-0.036	-0.008	-0.029	-0.035	-0.001	-0.039	0.003
	-0.038	-0.008	-0.030	-0.035	-0.003	-0.039	0.001
	-0.036	-0.014	-0.022	-0.041	0.005	-0.046	0.010
	-0.027	-0.005	-0.022	-0.032	0.005	-0.036	0.009
	-0.038	-0.009	-0.029	-0.036	-0.002	-0.040	0.002
	-0.038	-0.008	-0.030	-0.035	-0.003	-0.040	0.002
	-0.777	-0.776	-0.001	-0.777	0.0	-0.777	0.0
	-0.881	-0.884	0.003	-0.881	0.0	-0.880	0.0

3.3.3 - Performance Test Data

The function TRANSFORM_PP_DATA() was modified using the results of the truck tests, and performance tests conducted. The observations correspond to different test runs over different pavement tracks.

Table 3.10 - X Distance Performance Verification.

Observation	Road Measures	Raw VOC Data	Difference between Road and Raw VOC	Correction Model Estimates	Difference between Road and Correction Model Estimates
1	-118.646	-118.406	-0.240	-118.439	-0.206
2	-125.065	-124.604	-0.461	-124.570	-0.495
3	-162.893	-162.303	-0.590	-162.281	-0.611
4	-108.724	-108.395	-0.329	-108.294	-0.429
5	-120.703	-120.698	-0.005	-120.574	-0.129
6	-100.564	-100.462	-0.102	-100.357	-0.206
7	-110.533	-110.139	-0.394	-110.026	-0.507
8	-107.379	-107.122	-0.256	-107.067	-0.311
9	-110.322	-110.781	0.459	-111.159	0.837
10	-116.379	-116.448	0.069	-116.767	0.389

Table 3.11 - Y Distance Performance Verification.

Observation	Road Measures	Raw VOC Data	Difference between Road and Raw VOC	Correction Model Estimates	Difference between Road and Correction Model Estimates
1	-2.414	2.917	-5.332	-0.758	-1.656
2	-2.839	0.871	-3.710	-3.011	0.172
3	-2.384	1.806	-4.190	-3.196	0.813
4	-5.625	-1.363	-4.262	-4.864	-0.761
5	-6.741	-1.853	-4.889	-5.766	-0.976
6	-5.554	-1.650	-3.903	-4.864	-0.690
7	-5.70	-1.704	-3.996	-5.262	-0.438
8	-4.175	-0.034	-4.142	-3.432	-0.743
9	14.294	17.865	-3.571	15.337	-1.043
10	10.123	14.034	-3.911	11.066	-0.943

3.4 - Results

3.4.1 - Cart Test Results

The results of the cart tests are shown in Table 3.3 and in Figure 3.4. The maximum difference allowed under the specification of the machine is 0.667 percent (approximately 2 inches over a 300 inch run) . The maximum difference between the measured distance on the ground and the distance calculated using the VOC algorithms is less than 0.409 percent. These tests show that the VOC software and hardware function within the required specifications. The results also show that in the cart configuration there is a difference in whether the path traveled is relatively straight or curved because of heading estimation errors, but that the results are still within the required specifications. Five runs were along relatively straight paths (2, 3, 4, 6 and 7), and three were along curved paths (1, 5 and 8). The correlation between the measured and calculated positions for paths that are relatively straight (runs 2, 3, 4, 6 and 7) have a deviation of less than 0.10 percent, and these measured positions are within the measurement error allowed.

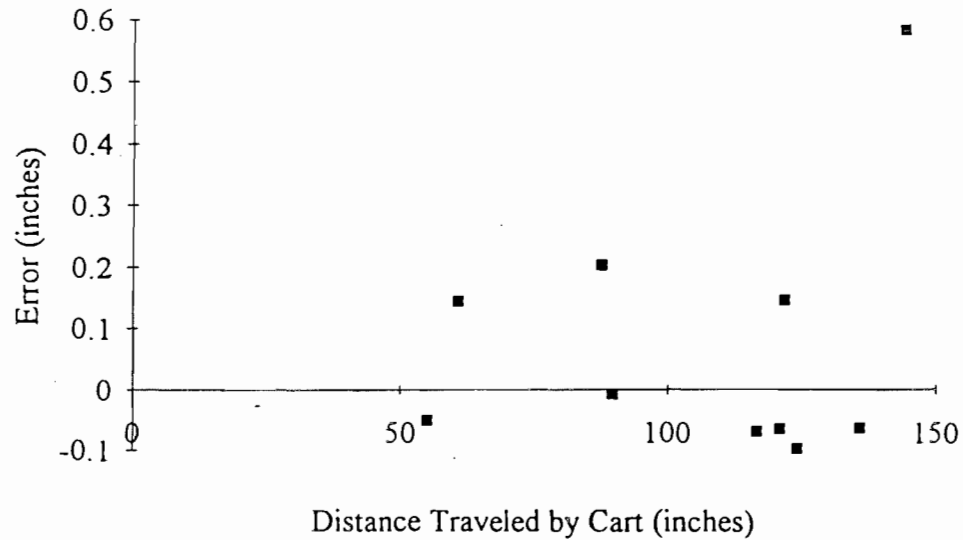


Figure 3.4 - Plot of Cart Error Measurements vs. Distance Traveled.

3.4.2 - Truck Test Results

The truck test results are shown in Tables 3.4 through 3.9 and Figures 3.5 through 3.7. The data collected was used in a least squares algorithm. The results of the least squares algorithm are shown in the last column of these tables and shows a significant improvement. The X direction error is within the ± 5 centimeters (2.0 inches) maximum window required by RPS/LSS to track a crack. However, not all the corrected Y direction crack locations are within the 2 inch specification window. The samples with the high error may be due to ground measurement errors that result in not locating the original crack correctly. The Y direction displacement is much smaller and is very sensitive to heading errors and small triangulation errors. A small angle error becomes a very large error when the measurement is made over a large distance. The block error estimator is an indicator of measurement differences that may be due to this, and it shows that the ones that had a high block difference were also the ones with the highest error.

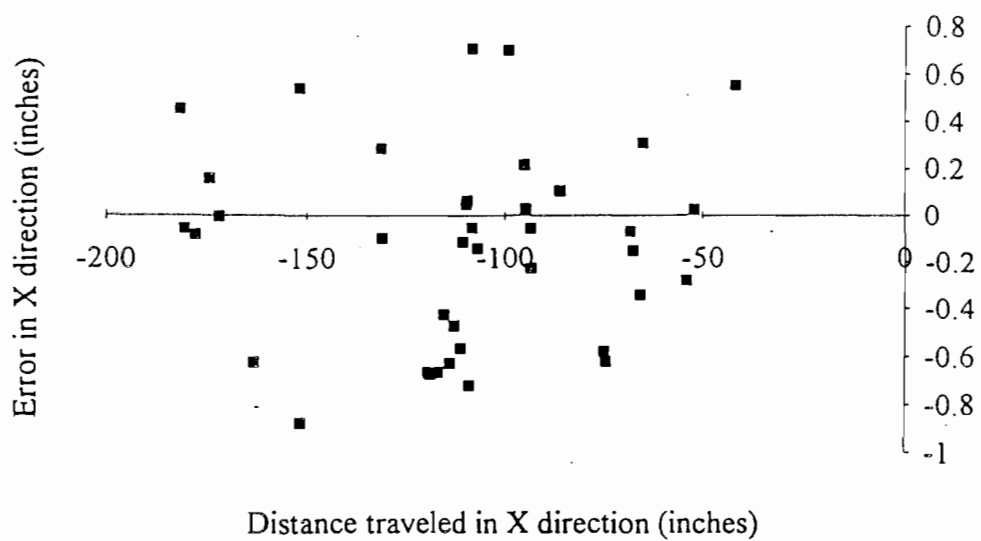


Figure 3.5 - Truck X Position Error.

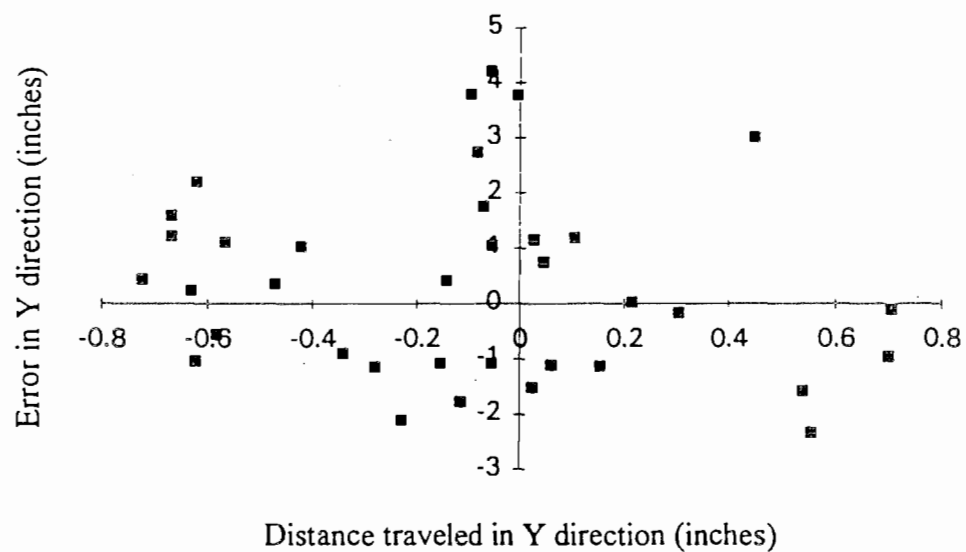


Figure 3.6 - Truck Y Position Error.

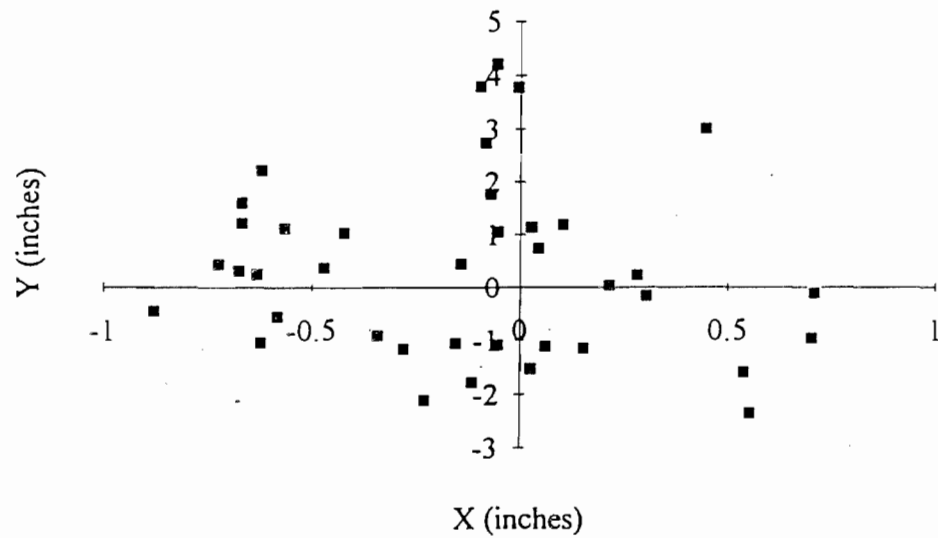


Figure 3.7 - Truck XY Position Error.

A least squares statistical analysis was performed on the truck test data, and the following results obtained.

Regression Statistics

Multiple R	0.994559541
R Square	0.98914868
Adjusted R Square	0.988855401
Standard Error	0.01879427
Observations	39

<i>Analysis of Variance</i>					
	<i>df</i>	<i>Sum of Squares</i>	<i>Mean Square</i>	<i>F</i>	<i>Significance F</i>
Regression	1	1.191328882	1.191328882	3372.723448	5.93717E-38
Residual	37	0.01306931	0.000353225		
Total	38	1.204398192			

Table 3.12 - Regression Analysis of Truck Test Data.

The analysis of variance (Table 3.12) tests the null hypothesis $H_0: \beta_1 = 0$, (where in our case $\beta_1 = 0.959631945$) using the F-distribution. The F-distribution for the null hypothesis shows that variance needs to be very small to be of significance at the 5% level, and thus we reject the null hypothesis and conclude that the independent variable (θ_{measured}) is useful in predicating values of $\theta_{\text{corrected}}$ or in explaining the variation in $\theta_{\text{corrected}}$.

As explained later, only the heading error parameter estimator is used, and the corrected heading is then used to recalculate a new X and Y. The heading correction model developed from the above analysis is:

$$\theta_{\text{corrected}} = \theta_{\text{measured}} \times 0.959631945 - 0.03183131 \quad (3.4)$$

The $\theta_{\text{corrected}}$ was used to determine the X and Y shown in Tables 3.4 to 3.9 as the correction model.

To test the parameter estimation scheme, additional data was collected and this model was used to verify performance. The results of these tests are shown below in Tables 3.10 and 3.11, and also illustrated graphically in Figures 3.9 through 3.13.

3.4.3 - Performance Test Results

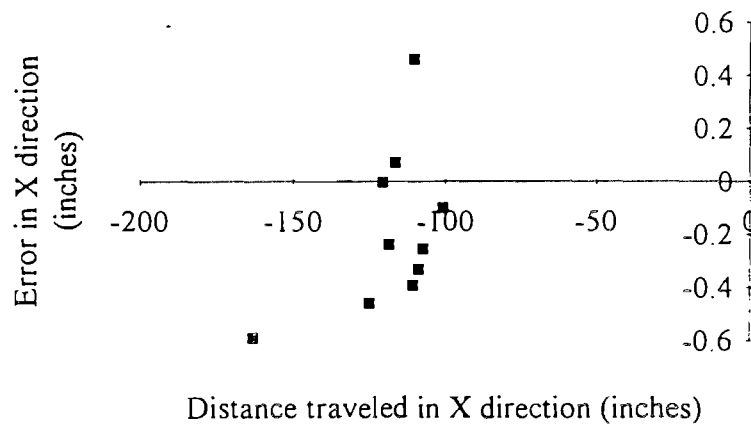


Figure 3.8 - Truck X Performance Test Position Error *without* Parameter Estimator.

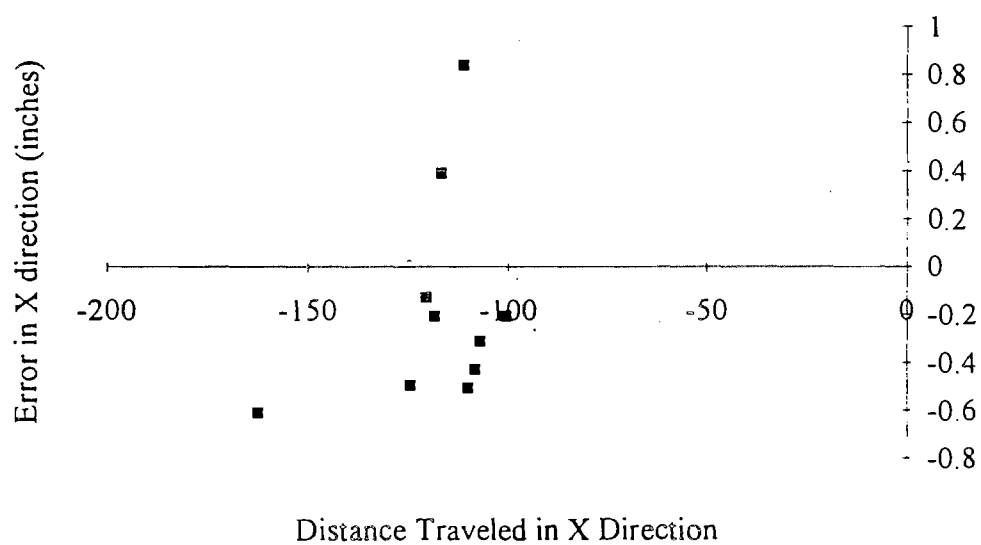


Figure 3.9 - Truck X Performance Test Position Error *with* Parameter Estimator.

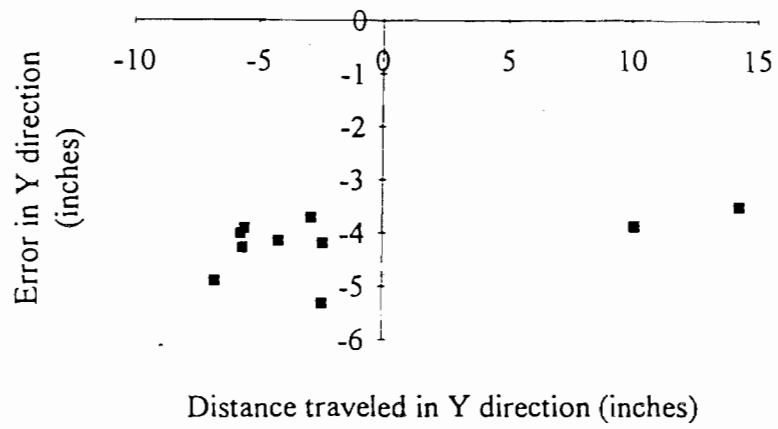


Figure 3.10 - Truck Y Performance Test Position Error *without* Parameter Estimator.

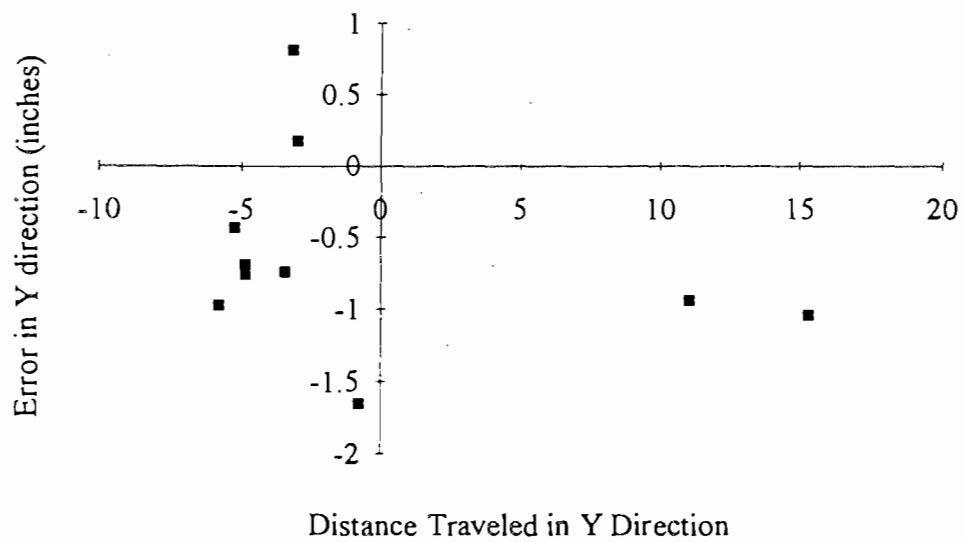


Figure 3.11 - Truck Y Performance Test Position Error *with* Parameter Estimator

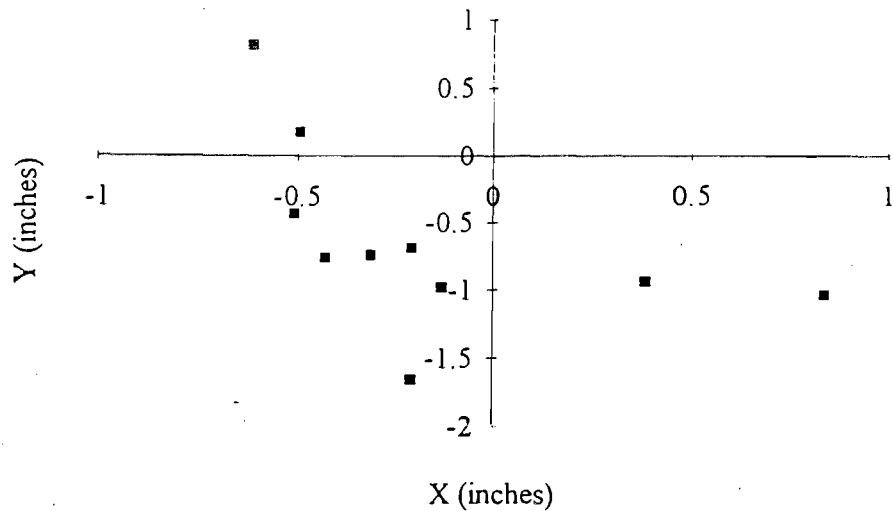


Figure 3.12 - Truck XY Performance Test Position Error.

3.5 - Discussions

3.5.1 - Cart Tests

The objective of the cart tests was to verify the functions of the VOC hardware and software. The results showed that the hardware and software functioned within required specifications and had minimal to insignificant measurement errors. Three types of errors were identified from these tests.

The first type of error is due to electrical-mechanical noise and the result is spurious counts. The spurious counts indicate the encoders have rotated a larger or smaller distance than is the case. Also related to this error is the limitation of discrete time required to read each encoder and record the value. Since this process takes time and cannot be done within the same instant, small variations in the encoder values result between the two encoder readings. This error can be reduced by having the encoders recorded by an interrupt triggered read. In the final

configuration, a low pass filter was implemented and these errors reduced; and as shown in the test results may not be a significant source of inaccuracies.

The second type of error is due to the resolution. Using the current encoders, we can resolve a rotation of up to 0.14 degrees. The true rotation is within this margin. With lower resolution encoders, we resolve larger angles, and thus have a larger error. For purposes of the VOC, this error is not expected to be significant because cracks are tracked for less than forty feet, and the accumulation of the errors is less than the crack resolution.

The third type of error is due to the interaction of the cart wheels and the floor surface. This error includes changes in wheel radius and wheel track, and wheel slip as the cart moves along the floor. In the test setup, this type of error was the most significant and accounts for most of the errors resulting from following a curved path. To illustrate, a wheel radius change of 0.04 inches over a 165 inch run results in a deviation of 1.6 inches. The fifth wheel used on the truck was developed to control this source of error by having a rigid wheel track and hard rubber tires that reduce the wheel radius deviations. The use of the fifth wheel assemblies significantly reduced this type of error.

3.5.2 - Truck Tests and Performance

The objective of the truck tests were to verify the integration of the VOC with the other subsystems and to develop an error correction model. The least squares algorithm was used to develop a parameter estimation scheme for correcting errors due to encoder misalignment, wheel slip, road surface conditions and the interactions of the VOC system and the other subsystems. This procedure should also be used in the future to correct for errors due to wear, misalignment and calibration degradation.

As shown by Banta, the angular error is the most significant source of position error. The data shows that the vehicle tends to drift in the Y direction (i.e. either left or right of the vehicle's forward direction). This drift results in the angular error and is due mainly to misalignments of the encoder wheels. Therefore, for this project, only the angle is determined for errors, modified, and then used to recalculate the X and Y positions.

The error correction model shows a marked improvement in the system's ability to track cracks. This is shown in the results obtained in the performance verification tests and is illustrated graphically in Figures 3.9 through 3.13. All the corrected crack locations are within the ± 5 centimeters (2 inch) specification window. There is no significant improvement in the X direction error. But most importantly, there is a pronounced improvement in the Y direction error, bringing this error to within required specifications.

CHAPTER 4 - CONCLUSION AND RECOMMENDATIONS

4.1 - Conclusion

The purpose of this thesis was to develop and demonstrate a vehicle and crack tracking hardware and software configuration for application in an ACSM. The design objectives of the ACSM required that the Vehicle Orientation and Control sub-system track cracks from the front of the vehicle to the rear within a ± 5 centimeters (2 inch) window. Optical encoders were used as position transducers, and their output was recorded using thirty-two bit counters installed on the back plane of a real-time computer. Initial development and demonstration was done on a mobile cart, and the tests done with this cart configuration showed that the hardware and software were adequate to perform the functions of the VOC. This hardware and software was then transferred to a prototype ACSM truck. The truck configuration required the VOC to function in an integrated environment with other sub-systems required to do the whole crack sealing process.

The truck performance tests demonstrated that the VOC consistently and reliably determines the location of cracks within the ± 5 centimeters (2 inch) specification under varying road conditions and vehicle headings. A least squares method was used to determine a self-tuning correction model, and subsequent tests of the self-tuning parameter estimator produced crack locations consistently and reliably within specifications. For the purposes of the prototype ACSM this configuration is adequate.

4.2 - Recommendations

The dead reckoning system was selected for its low cost, simplicity and because it was an on-board measurement system that does not rely on external support. However, it is a system prone to error, and the following three methods are recommended to increase the accuracy of the position and heading measurement in a future machine:

1. A reference generator that can be used to self-correct the dead reckoning system. This method has been used in the Blanche mobile robot. The external reference guides provide an upper and lower bound for the position estimate given by the dead reckoning system. For the Blanche mobile robot type application, a priori maps and beacons are provided as correction references and this works well in a factory floor application. The method is of limited use in the ACSM because of the difficulty of locating and identifying several known landmarks in a continuous operation. However, under certain conditions where there are road tiles or markers available, these could be used as external beacons to provide error bounds for the dead reckoning system.
2. Almost all mobile robot applications in recent literature use a navigation and position estimation system that uses multi-sensors such as optical range finders, vision systems and laser systems in addition to the dead reckoning - encoder system. Elfes, et. al., for example, describe a system that uses stereo vision to provide navigation and reference position data. The Hongo vehicle illustrated in Figures 1.3 and 1.4 uses encoders to determine position and heading, and corrects its measurement error at regular intervals using optical and sonar methods. This scheme would be most applicable to the ACSM. In all these multi-sensor applications, a Kalman filter is used to integrate the measurements from the different sensors.
3. Reducing the distance between the vision system line scan and the robot work space would significantly reduce dead reckoning errors in the ACSM. The dead reckoning measurement error increases as the distance traveled increases. The longer the distance a specific crack has to be tracked before it enters the robot work space, the higher the error. In a future ACSM prototype, it may be possible to place the vision system in closer proximity to the VOC and robot so as to reduce the distance each crack is tracked before sealing.

REFERENCES

- Banta, L. E. (1988) "A Self Tuning Navigation Algorithm," IEEE International Conference On Robotics and Automation, pp. 1313-1314.
- Chatila, R., and Loumond, J.P. (1985). "Position Referencing and Consistent World Modeling for Mobile Robots," Proceedings IEEE International Conference on Robotics and Automation. St. Louis: IEEE, pp. 138-145.
- Cox, I.J. and Wilfong, G.T. (1990) "Autonomous Robot Vehicles," AT&T.
- Hongo, T., Arakawa, H., Sugimoto, G., Tange, K., and Yamamoto, Y. (1985) "An Automatic Guidance System of a Self-Controlled Vehicle", IEEE International Conference on Industrial Electronics, Control & Instrumentation, pp. 5535-540.
- Ravani, B., Velinsky, S. A. and West, T., "Requirements for Applications of Robotics and Automation in Highway Maintenance Tasks," Proceedings of the ASCE Conference on Robotics for Challenge Environments, pp. 356-364, 1994.
- Ravani, B. and West, T.H. (1991), "Applications of Robotics and Automation in Highway Maintenance Operations", Proceedings of the 2nd ASCE International Conference on Applications of Advanced Technologies in Transportation Engineering, pp. 61-65.
- Skibniewski, M. and Hendrickson, C. (1990), "Automation and Robotics for Road Construction and Maintenance", Journal of Transportation Engineering, Volume 116, Number 2, pp. 261-271
- Smith, R. C. and Cheeseman, P. (1986) "On the Representation and Estimation of Spatial Uncertainty," The International Journal of Robotics Research, Volume 5, Number 1, Spring 1986 pp. 56-68.
- Sugimoto, G., Watanabe, K., Ninomiya, Y., Hongo, T and Arakawa, H. (1988) "Practical Course Follow Performance of an AGV without Fixed Guide ways. Proceedings of the USA-Japan Symposium on Flexible Automation - Crossing Bridges: Advances sin Flexible Automation and Robotics, Minneapolis, MN, pg. 651-655.
- Tsumura, T., Hashimoto, M, and Fujiwara, N. (1982) "A System for Measuring Current Position and/or Heading of Vehicles", Bulletin of JSME, volume 25, number 203:821-826, May 1982.
- Tsumura, T. and Fujiwara, N. (1978), "An Experimental System for Processing Movement Information of Vehicle", 28th IEEE Vehicle Technology Conference Proceedings, 163-168, Denver, Colorado.

Tsumura, T., Hashimoto, M, and Fujiwara, N. (1985), "New Method for Position and Heading Compensation on Ground Vehicle", Proceedings of '85 International Conference on Advanced Robotics, 429-436, September 1985.

Tsumura, T., Hashimoto, M, and Fujiwara, N. (1985), "An Experimental System for Self-Contained Position and Heading Measurement of Ground Vehicle," Proceedings of '83 International Conference on Advanced Robotics, 269-276, September 1983.

Velinsky, S.A., (1993) "Heavy Vehicle System for Automated Pavement Crack Sealing," Heavy Vehicle Systems, Special Series of the International Journal of Vehicle Design, Vol. 1, No. 1, pp. 114-128, 1993.

Wang, C. M. (1988) "Location Estimation and Uncertainty Analysis for Mobile Robots," IEEE International Conference on Robotics and Automation, pp. 1230-1235.

APPENDIX A - VEHICLE ORIENTATION AND CONTROL PROGRAM

```
/*          VOCRUN.C  for H68/VME3E
*
*          BY JONATHAN WANJIRU
*
*          MAY/20/94
*
*/

#include <time.h>
#include <math.h>
#include <strings.h>
#include <signal.h>

#include "/h0/demo/include/common.h"
#include "/h0/demo/source/dmod.c"
#include "xvme230m.h"

char *argv[MAXLEN];

main(argc, argv)
unsigned argc;
char *argv[];
{

    int i, channel=0, mode;
    double xcord1, ycord1, theta1, measured_n1, measured_nr, measured_x;
    double xcord2, ycord2, theta2, wheel_base, mag1, mag2;
    char *execmdp;
    SCMDBLK *cmdblk[MAXCHANNEL];    /* Command block for the
                                     XVME230 data input and output */
    double *x_cur, *y_cur, *theta_cur;

    xcord1=ycord1=theta1=xcord2=ycord2=theta2=0.0;

    if(argc == 3)
    {

        if (( stream=fopen(argv[1],"w")) == NULL)
        {

            printf("\ncould not open >%s<\n", argv[1]);
            exit(0);
        }
    }
}
```

```

    }
}

if(VOC_DEBUG) /*      output file for debugging information      */
{
    if (( stream=fopen("/h0/user/wanjiruj/source/dbg/voc.dbg","w")) == NULL)
    {
        printf("\nvoc:could not open >%s<\n", "voc.dbg");
        exit(0);
    }
}

/*      setup for each encoder channel      */

cmdblk[0]=(SCMDBLK *)(CMDBLK_ADDR0+SHORT_ADDR + MOD_ADDR );
cmdblk[1]=(SCMDBLK *)(CMDBLK_ADDR1+SHORT_ADDR + MOD_ADDR );
cmdblk[2]=(SCMDBLK *)(CMDBLK_ADDR2+SHORT_ADDR + MOD_ADDR );
cmdblk[3]=(SCMDBLK *)(CMDBLK_ADDR3+SHORT_ADDR + MOD_ADDR );
cmdblk[4]=(SCMDBLK *)(CMDBLK_ADDRG+SHORT_ADDR + MOD_ADDR );

/* Perform initialization of data modules and pipes */

if(argc!=2)
{
    printf("voc: usage 'voc icu_pid'\n");
    exit(1);
}

icu_pid = atoi(argv[1]);
printf("voc: ICU pid is %d\n",icu_pid);

printf("Creating Data Modules\n");

printf("%s: Initializing the p.p. data module.\n", argv[0]);
if(!Init_PP_Module())
{
    printf("%s: Error initializing the path planning data module.\n", argv[0]);

    if(VOC_DEBUG)
    {
        fprintf(stream,"%s: Error initializing the path planning data module.\n", argv[0]);
    }
    exit(1);
}

```

```

printf("%s: Attempting to initialize the WS data module.\n", argv[0]);
if(!Init_WS_Module())
{
    printf("%s: Error initializing the WS data module.\n", argv[0]);

    if(VOC_DEBUG)
    {
        fprintf(stream, "%s: Error initializing the WS data module.\n", argv[0]);
    }
    exit(1);
}

printf("%s: Attempting to initialize the PID data module.\n", argv[0]);
if(!Init_PID_Module())
{
    printf("%s: Error initializing the PID data module.\n", argv[0]);
    if(VOC_DEBUG)
    {
        fprintf(stream, "%s: Error initializing the PID data module.\n", argv[0]);
    }
    exit(1);
}

printf("%s: Attempting to initialize the VOC data module.\n", argv[0]);
if(!Init_VOC_Module())
{
    printf("%s: Error initializing the VOC data module.\n", argv[0]);
    if(VOC_DEBUG)
    {
        fprintf(stream, "%s: Error initializing the VOC data module.\n", argv[0]);
    }
    exit(1);
}

printf("%s: Attempting to initialize the COUNT data module.\n", argv[0]);
if(!Init_COUNT_Module())
{
    printf("%s: Error initializing the COUNT data module.\n", argv[0]);
    if(VOC_DEBUG)
    {
        fprintf(stream, "%s: Error initializing the COUNT data module.\n", argv[0]);
    }
    exit(1);
}

```

```

intercept(sighand);          /* signal handler setup */
printf("%s: Installed interrupt handler\n", argv[0]);

channel=4; /* used to reset the XVME230 card */
rs230(channel,cmdblk[channel]);
swait(1);

kill(icu_pid, VOC_INIT_DONE); /* Tell ICU that all initialization is done */

/* initial encoder data input channels */
for(channel=0;channel <MAXCHANNEL-1; channel++)
    start_counter(channel, cmdblk[channel]);
swait(1);
mode=1; /* start is mode=1 */
for(channel=0;channel <MAXCHANNEL-1; channel++)
    chkcmd(mode, channel, cmdblk[channel]);
swait(1);

for(channel=0;channel <MAXCHANNEL-1; channel++)
    read_counter(channel, cmdblk[channel]);
swait(1);
mode=2; /* read is mode=2 */

/* poll channels on read mode */
for(channel=0;channel <MAXCHANNEL-1; channel++)
    chkcmd(mode, channel, cmdblk[channel]);
i=0;
for(channel=0;channel <MAXCHANNEL-1; channel++)
{
    global_data[channel][i]=cmdblk[channel]->operand2;
    printf("global data channel %d: %d\t", channel, global_data[channel][i]);
}

while (TRUE)
{
    for(channel=0;channel <MAXCHANNEL-1; channel++)
        read_counter(channel, cmdblk[channel]);
    swait(1);
    mode=2;
    for(channel=0;channel <MAXCHANNEL-1; channel++)
        chkcmd(mode, channel, cmdblk[channel]);
    i=1;
}

```

```

for(channel=0;channel <MAXCHANNEL-1; channel++)
    global_data[channel][i]=cmdblk[channel]->operand2;

/*      keeps track of the vehicle's global position      */

worldjwl(&(global_data[LEFT_ENCODER][0]),&(global_data[RIGHT_ENCODER][0]),
&global_x_cur, &global_y_cur, &global_theta_cur);

for(channel=0;channel <MAXCHANNEL-1; channel++)
    global_data[channel][0]=global_data[channel][i];

if(FLAG_VOC_END_PROGRAM)
{
    printf("\n%s: Received an exit message\n", "voc");
    Close_Data_Modules();
    if(VOC_DEBUG)
        fclose(stream);
    FLAG_VOC_END_PROGRAM=FALSE;
    exit(sigvalue);
}
else if(FLAG_WS_CORD_REQ)
{
    printf("\n%s: received WS_CORD_REQ signal.\n", "voc");
    ws_cord(cmdblk);
    printf("\n%s: after WS_CORD_REQ signal.\n", "voc");
    FLAG_WS_CORD_REQ=FALSE;
    kill(pp_pid, WS_DATA_READY);
}
else if(FLAG_PP_DATA_READY)
{
    printf("\n%s: Received PP_DATA_READY signal\n", "voc");
    transform_pp_data(cmdblk);
    FLAG_PP_DATA_READY=FALSE;
    kill(icu_pid,VOC_DATA_READY);
}
else if(FLAG_RESET_SYSTEM)
{
    printf("\n%s: Signal received indicating RESET.\n", "voc");
    Close_Data_Modules();
    if(VOC_DEBUG)
        fclose(stream);
    FLAG_RESET_SYSTEM=FALSE;
    kill(icu_pid,3);
}

```

```

        else if(FLAG_UPDATE_PID)
        {
            printf("\n%s: got UPDATE_PID signal\n", "voc" );
            ReadPIDModule();
            FLAG_UPDATE_PID=FALSE;
            printf("\n%s: icu_pid = %d, pp_pid = %d\n", "voc", icu_pid, pp_pid);
        }
    } /*      end of while loop      */

    Close_Data_Modules();

    printf("\nend of program\n");
} /* end of main program */

```

/***/

/***/

XVME230M.H

header file for the xycom 230 counter card programs
this file contain heurcon's old compiler type functions

jan/23/93 jw

/***/

```

#define ADDR_MOD      0x0600
#define SHORT_ADDR    0xC00000
#define MOD_ADDR      0x3800
#define CMDBLK_ADDR   0xF2      /* 242 */

#define RAD_WHEEL      7.97
#define WHEEL_BASE     105

#define COUNTS_PER_REV      800 /* 10000 */
#define VSS_COUNTS_PER_REV  800.0 /* 800 */

#define CMDBLK_ADDR0      242
#define CMDBLK_ADDR1      270
#define CMDBLK_ADDR2      298
#define CMDBLK_ADDR3      326

```

```

#define CMDBLK_ADDRG      354

#define DATA_LENGTH      6
#define MAXLEN            20
#define INTERRUPTS        0x0000
#define BOARD_RESET      0x0010
#define BOARD_DIAGN      0x0011
#define START_32CNT      0x0021
#define READ_32CNT       0x0025
#define MAXCHANNEL        5
#define MAXCYCLES         50
#define VOC_CURRENT_LOC   0xCCF
#define WS_CORD_READY     0xcce
#define VOC_DEBUG         1

#define LEFT_ENCODER      0      /* data channel on which left encoder data is
registered (1) */
#define RIGHT_ENCODER     2      /* data channel on which right encoder data is
registered (3) */
#define ENCODER_VSS_LENGTH 285    /* distance between encoders and front of VSS
camera range                      (inches) */

#define VSS_MAX_COL       56      /* maximum number of columns in VSS line scan
*/
#define TRUCKLENGTH       285     /* distance between vss grd reading and encoder
wheels */
#define FRAMESIZE         30.00
#define WSLENGTH1         88      /* distance between encoder wheels and front of
work space */
#define WSLENGTH2         136     /* distance between encoder wheels and back of
work space */

/*
*****
****
the COUNTER_WS1 is calculated as follows:
COUNTER_WS1=COUNTS_PER_REV*(TRUCKLENGTH+WSLENGTH1)/2*PI*RAD_WH
EEL
and similarly for COUNTER_WS2
TRUCKLENGTH_COUNT =
(COUNTS_PER_REV*(TRUCKLENGTH)/(2*PI*RAD_WHEEL))
*****
****/

```

```
#define TRUCKLENGTH_COUNT 4553 /* counts for distance betwix vss grd reading and
encoder wheels */
#define COUNTER_WS1 6025 /* counts for distance between vss grd reading and front of
work space */
#define COUNTER_WS2 6792 /* counts for distance between vss grd reading and back of
work space */
```

```
typedef struct command_block
```

```
{
    short cmd;
    short rsp;
    short ints;
    short rflgncmd;
    int ncmd;
    short btypbadm;
    short operand1;
    int operand2;
} SCMDBLK;
```

```
/*
*****
```

```
*
*   Global VOC Variables Defination
*
```

```
*****/
```

```
FILE *stream;
char file_name[MAXLEN];
```

```
count_dm *count_dataptr;
PP_Path *pp_dataptr;
VOC_Path *voc_dataptr;
wsdm *ws_dataptr;
id_dm *pid_dataptr;
```

```
int icu_pid, pp_pid;
short sigvalue;
short FLAG_WS_CORD_REQ=FALSE;
short FLAG_UPDATE_PID=FALSE;
short FLAG_PP_DATA_READY=FALSE;
short FLAG_VOC_END_PROGRAM=FALSE;
short FLAG_RESET_SYSTEM=FALSE;
short FLAG_VOC_READ_DONE=FALSE;
```

```
double global_x_cur, global_y_cur, global_theta_cur; /* external global position variables
*/
```



```
int global_data[MAXCHANNEL][3];
```

```

/*****
Function:    INVERSE_WORLD(LEFT_COUNT, RIGHT_COUNT, X_CUR,
Y_CUR, THETA_CUR)

```

This function is given current vehicle location, and calculate what the VSS counter would have been one truck length ago.

```

*****/

```

```

inverse_world(left_count, right_count, x_cur, y_cur, theta_cur)
int *left_count, *right_count;
double *x_cur, *y_cur, *theta_cur;

```

```

{
double Na, Nd;
int Nr, Nl;
double x_new, y_new;
double d_theta, theta_new;

```

```

Nd=*theta_cur*(COUNTS_PER_REV*WHEEL_BASE)/(2*PI*RAD_WHEEL);
Na = *x_cur * COUNTS_PER_REV/( 2*PI*RAD_WHEEL*cos((*theta_cur)));
Nr = ((2*Na) - Nd)/2;
Nl = ((2*Na) + Nd)/2;
/* *left_count=Nl;
*right_count=Nr; */

```

```

*left_count = Nl-TRUCKLENGTH_COUNT;
*right_count= Nr-TRUCKLENGTH_COUNT;

```

```

if(VOC_DEBUG)
{
fprintf(stream,"inverse0: *xur:%f\t*ycur:%f\t*tcu:%f\n",
*x_cur,*y_cur,*theta_cur);
fprintf(stream,"inverse0: *leftcount:%d\t*rightcount:%d\n",
*left_count, *right_count);
}
}

```

```

/*****
FUNCTION: WS_CORD(CMDBLK)

```

This function calculates the boundaries of the current work space. It returns the maximum row and column, and the minimum row and column.

```

*****
*****/.
ws_cord(cmdblk)
SCMDBLK *cmdblk[MAXCHANNEL];
{
double *x_cur, y_cur, *theta_cur;

int data[MAXCHANNEL][3], leftcount1, rightcount1, leftcount2, rightcount2, i, channel=0;
int mode, path_num, item_num, rmin, rmax, index;
char *execmdp;

double xcrack2, ycrack2, theta_cr2;

double pw1x, pw1y, theta_pw1, theta_pw2, pw2x, pw2y;

pw1x=pw1y=theta_pw1=theta_pw2=pw2x=pw2y=0.0;
xcrack2=ycrack2=theta_cr2=0.0;
i=0;

for(channel=0; channel <MAXCHANNEL-1; channel++)
    data[channel][i]=0;

i=1;
for(channel=0; channel <MAXCHANNEL-1; channel++)
    read_counter(channel, cmdblk[channel]);
swait(1);
mode=2;
for(channel=0; channel <MAXCHANNEL-1; channel++)
    chkcmd(mode, channel,
cmdblk[channel]);
for(channel=0;
channel <MAXCHANNEL-1; channel++)
data[channel][i]=cmdblk[channel]->operand2;

worldjw1(&(data[LEFT_ENCODER][0]), &(data[RIGHT_ENCODER][0]), &xcrack2, &ycrack2,
&theta_cr2);

if(VOC_DEBUG)
{
    fprintf(stream, "\nws_cord: vehicle location is
x:%f y:%f theta:%f\n", xcrack2, ycrack2, theta_cr2);
}

```

```

    }

    pw1x= xcrack2 - (WSLENGTH1);
    pw1y= ycrack2;
    theta_pw1= theta_cr2;
    inverse_world(&leftcount1,&rightcount1,&pw1x,&pw1y,&theta_pw1);

    if(VOC_DEBUG)
    {
        fprintf(stream,"\nws_cord: position vector 1 x:%f\ty:%f\ttheta:%f\n",pw1x,pw1y,theta_pw1);
        fprintf(stream,"\nposition vector 1 leftcount1: %d chk: %d\n", leftcount1, data[1][1]-
COUNTER_WS1);
    }

    pw2x= xcrack2 - (WSLENGTH2);
    pw2y= ycrack2;
    theta_pw2 = theta_cr2;
    inverse_world(&leftcount2,&rightcount2,&pw2x,&pw2y,&theta_pw1);

    if(VOC_DEBUG)
    {
        fprintf(stream,"\nws_cord: position vector 2 x:%f\ty:%f\ttheta:%f\n",pw2x,pw2y,theta_pw2);
        fprintf(stream,"\nposition vector 2 leftcount2: %d chk: %d", leftcount2, data[0][1]-
COUNTER_WS2);
    }

    if(VOC_DEBUG)
    {
        fprintf(stream,"\nws_cord: new method leftcount1: %d leftcount2: %d",leftcount1, leftcount2);
        fprintf(stream,"\nws_cord: voc rightcount1: %d rightcount2: %d", rightcount1, rightcount2);
    }

    /*      maximum row      */
    for(index=0;
        leftcount1 > count_dataptr->count_data[index].right_count &&
        count_dataptr->count_data[index].right_count !=0 ;
        ++index);
    rmax=index;

    /*      minimum row      */
    for(index=0;
        leftcount2 > count_dataptr->count_data[index].right_count &&
        count_dataptr->count_data[index].right_count !=0;
        index++);
    rmin=index;

```

```

    if(VOC_DEBUG)
    {
        printf("\nvoc ws rmin :%d\trmax:%d\n", rmin, rmax);
        fprintf(stream, "\nws_cord voc ws rmin :%d\trmax:%d\n", rmin, rmax);
    }

/* the following two lines are for a dynamic RPS frame */
    ws_dataptr->min_row= rmin;
    ws_dataptr->max_row= rmax;
}

/*****
Alternate version of

INVERSE_WORLD(LEFT_COUNT, RIGHT_COUNT, X_CUR, Y_CUR, THETA_CUR)

*****/
inverse_worldjwl(left_count, right_count, x_cur, y_cur, theta_cur)
int *left_count, *right_count;
double *x_cur, *y_cur, *theta_cur;
{
    double Na, Nd;
    int Nr, Nl;
    double x_new, y_new;
    double d_theta, theta_new;

    Nd= *theta_cur*(COUNTS_PER_REV*WHEEL_BASE)/(2*PI*RAD_WHEEL);

    Na=*x_cur *COUNTS_PER_REV/(2*PI*RAD_WHEEL*cos(*theta_cur));

    Nr= Na - (Nd/2);
    Nl= Na + (Nd/2);

    *left_count = Nl-TRUCKLENGTH_COUNT;
    *right_count= Nr-TRUCKLENGTH_COUNT;

    if(VOC_DEBUG)
    {
        fprintf(stream, "\nlc:%d\t", *left_count);
        fprintf(stream, "\nrc:%d\n", *right_count);
        fprintf(stream, "\nl: *xur:%ft*ycur%ft*tcur:%f\n", *x_cur, *y_cur, *theta_cur);
    }
}

```

```
}
```

```
/******
```

```
FUNCTION: TRANSFORM_PP_DATA(CMDBLK)
```

This function transforms cracks from VSS to VOC to RPS coordinte frames. The data is transmittted to RPS in millimeters.

```
*****
```

```
*****/
```

```
transform_pp_data(cmdblk)
```

```
SCMDBLK *cmdblk[MAXCHANNEL];
```

```
{
```

```
double *x_cur, *y_cur, *theta_cur;
```

```
int count_data[MAXCHANNEL][3], delta_data[MAXCHANNEL][3];
```

```
int i, channel=0, mode, path_num, item_num, row,r,column;
```

```
char *execmdp;
```

```
double xi, yi, ximinus1, yiminus1, theta1, xcrack1, ycrack1, theta_cr1;
```

```
double bi, ci;
```

```
channel =RIGHT_ENCODER;
```

```
count_data[channel][0]=0;
```

```
channel =LEFT_ENCODER;
```

```
count_data[channel][0]=0;
```

```
voc_dataptr->num_ws_paths=pp_dataptr->num_ws_paths;
```

```
for(path_num=0; path_num<pp_dataptr->num_ws_paths; path_num++)
```

```
{
```

```
    voc_dataptr->ws_path_size[path_num]=pp_dataptr->ws_path_size[path_num];
```

```
    for(item_num=0; item_num < pp_dataptr->ws_path_size[path_num]; item_num++)
```

```
    {
```

```
        row=pp_dataptr->ws_path_data[path_num][item_num].row;
```

```
        column=pp_dataptr->ws_path_data[path_num][item_num].col;
```

```
        channel =LEFT_ENCODER;
```

```
        count_data[channel][1]=count_dataptr->count_data[row].left_count;
```

```
        channel =RIGHT_ENCODER;
```

```
        count_data[channel][1]=count_dataptr->count_data[row].right_count;
```

```
if(VOC_DEBUG)
```

```
{
```

```
printf("\ndata[LEFT1][0]:%d\tdata[RIGHT1][0]:%d\n",count_data[LEFT_ENCODER][0],count_data[RIGHT_ENCODER][0]);
```

```
printf("\ndata[LEFT2][1]:%d\tdata[RIGHT2][1]:%d\n",count_data[LEFT_ENCODER][1],count_data[RIGHT_ENCODER][1]);
```

```
fprintf(stream,"\ntransform_pp_data: pp_row:%d\n",count_data[LEFT_ENCODER][0]:%d\tcount_data[RIGHT_ENCODER][0]:%d\n",row, count_data[LEFT_ENCODER][0],count_data[RIGHT_ENCODER][0]);
```

```
fprintf(stream,"transform_pp_data:count_data[LEFT_ENCODER][1]:%d\tcount_data[RIGHT_ENCODER][1]:%d\tcdm:%d\n",count_data[LEFT_ENCODER][1],count_data[RIGHT_ENCODER][1],count_dataptr->count_data[row].left_count,count_dataptr->count_data[row].right_count);
}
```

```
ximinus1=yiminus1=theta1=0.0;
```

```
worldjwl(&(count_data[LEFT_ENCODER][0]),&(count_data[RIGHT_ENCODER][0]),&ximinus1,&yiminus1,&theta1);
```

```
ximinus1=ximinus1+ ENCODER_VSS_LENGTH
yiminus1=yiminus1+(VSS_MAX_COL-column*2);
```

```
for(channel=0;channel <MAXCHANNEL-1; channel++)
    read_counter(channel, cmdblk[channel]);
swait(1);
mode=2;
for(channel=0;channel <MAXCHANNEL-1; channel++)
    chkcmd(mode, channel, cmdblk[channel]);
```

```
for(channel=0;channel <MAXCHANNEL-1; channel++)
    global_data[channel][1]=cmdblk[channel]->operand2;
```

```
worldjwl(&(global_data[LEFT_ENCODER][0]),&(global_data[RIGHT_ENCODER][0]),&global_x_cur, &global_y_cur, &global_theta_cur);
```

```
for(channel=0;channel <MAXCHANNEL-1; channel++)
    global_data[channel][0]=global_data[channel][1];
```

```
bi=-global_y_cur*sin(global_theta_cur)-global_x_cur*cos(global_theta_cur);
ci=-global_y_cur*cos(global_theta_cur)+global_x_cur*sin(global_theta_cur);
xi = (cos(global_theta_cur) * ximinus1) + (sin(global_theta_cur) * yiminus1)+bi;
yi = (-sin(global_theta_cur) * ximinus1) + (cos(global_theta_cur) * yiminus1)+ci;
```

```

if(VOC_DEBUG)
{
    fprintf(stream,"\ntransform_pp_data: global_x:%f\tglobal_y:%f\tthetat:%f\n",
        global_x_cur, global_y_cur, global_theta_cur);
    fprintf(stream,"transform_pp_data: ximinus1:%f\tyiminus1:%f\nws_ppcrack theta:%f\n",
        ximinus1,yiminus1,pp_dataptr->ws_path_data[path_num][item_num].theta);
    fprintf(stream,"transform_pp_data: xi:%f\tyi:%f\n\n", xi, yi);
}

    voc_dataptr->voc_path_data[path_num][item_num].xcord=xi;
    voc_dataptr->voc_path_data[path_num][item_num].ycord=yi;
    voc_dataptr->voc_path_data[path_num][item_num].theta=pp_dataptr-
>ws_path_data[path_num][item_num].theta;
}
}

if(VOC_DEBUG)
{
    Display_COUNT_Module();
    Display_PP_Module();
    Display_VOC_Module();
    fclose(stream);
    kill(icu_pid,3);
}
}

/*****
Display crack information from Pathplan data module
*****/
int Display_PP_Module()
{
    int path_num, item_num;

    printf("Display_PP_Module: Number of paths in workspace: %d\n",
        pp_dataptr->num_ws_paths);

    for(path_num=0;path_num<pp_dataptr->num_ws_paths;path_num++)
    {
        printf("Data for path number %d: \n",path_num+1);
        for(item_num=0;item_num < pp_dataptr->ws_path_size[path_num]; item_num++)
        {

            if(VOC_DEBUG)

```

```

{
    fprintf(stream, "\nDisplay_PP_Module(): (%d, %d, %d)\n",
        pp_dataptr->ws_path_data[path_num][item_num].row,
        pp_dataptr->ws_path_data[path_num][item_num].col,
        pp_dataptr->ws_path_data[path_num][item_num].theta);

    printf("(%d, %d, %d)\n",
        pp_dataptr->ws_path_data[path_num][item_num].row,
        pp_dataptr->ws_path_data[path_num][item_num].col,
        pp_dataptr->ws_path_data[path_num][item_num].theta);
    printf("\n\n");
}
}
}
return(TRUE);
}

/*****
Display crack information from VOC data module
*****/
int Display_VOC_Module()
{
    int path_num, item_num;

    printf("Display_VOC_Module: Number of paths in workspace: %d\n",
        voc_dataptr->num_ws_paths);

    for(path_num=0; path_num < voc_dataptr->num_ws_paths; path_num++){
        printf("Data for path number %d: \n", path_num+1);
        for(item_num=0; item_num < voc_dataptr->ws_path_size[path_num]; item_num++)
        {
            if(VOC_DEBUG)
            {
                fprintf(stream, "\nDisplay_VOC_Module(): (%f %f %f)\n",
                    voc_dataptr->voc_path_data[path_num][item_num].xcord,
                    voc_dataptr->voc_path_data[path_num][item_num].ycord,
                    voc_dataptr->voc_path_data[path_num][item_num].theta);

                printf("voc: (%f, %f, %f)\n",
                    voc_dataptr->voc_path_data[path_num][item_num].xcord,
                    voc_dataptr->voc_path_data[path_num][item_num].ycord,
                    voc_dataptr->voc_path_data[path_num][item_num].theta);
            }
        }
    }
}

```



```

    }
}
return(TRUE);
}

/*****
Display crack information from COUNT data module
*****/
int Display_COUNT_Module()
{
    int index;

    if(VOC_DEBUG)
    {
        for(index=0; count_dataptr->count_data[index].left_count !=0 ; index++)
        {
            fprintf(stream, "\nDisplay_COUNT_Module(): %d: (%d\t %d)\n",
                    index, count_dataptr->count_data[index].left_count ,
                    count_dataptr->count_data[index].right_count );
        }

        printf("\n\n");
    }

    return(TRUE);
}

/*****
FUNCTION: chkcmd(mode, channel, cmdblk)

This function polls the XYCOM 230 card to verify that the process executed has been completed
*****/
chkcmd(mode, channel, cmdblk)
short mode, channel;
SCMDBLK *cmdblk;
{
    int notdone;

    notdone=((unsigned)(cmdblk->rflgncmd) >> 8);
    while ( notdone )
    {

```

```

        printf(".");
        notdone=((unsigned)(cmdblk->rflgncmd) >> 8);
    }

    if((short)cmdblk->rsp != 0)
    {
        printf("\nmode %d channel %d error code: >%X<\n",mode, channel,
(short)cmdblk->rsp);
        rs230(channel,cmdblk);
        exit(0);
    }
}

```

```

/*****
FUNCTION: CMDBLKP(CHANNEL, CMDBLK)

```

Function sets up command block on XYCOM 230 card. This is the block that is used to execute commands on the card and receive data.

```

*****/

```

```

cmdblkp(channel, cmdblk)
short channel;
SCMDBLK *cmdblk;
{
    short *seg1, *seg2, *seg0;
    int segment2;

    seg0=(short *)((channel * DATA_LENGTH) + 0x92 + SHORT_ADDR + MOD_ADDR );
    seg1=(short *)((channel * DATA_LENGTH) + 0x94 + SHORT_ADDR + MOD_ADDR );
    seg2=(short *)((channel * DATA_LENGTH) + 0x96 + SHORT_ADDR + MOD_ADDR );

    segment2=(int)cmdblk-SHORT_ADDR;

    *seg0=0x002D;
    *seg1=0x00C0;
    *seg2=(short)segment2;
}

```

```

/*****/
read_counter(channel, cmdblk)
short channel;
SCMDBLK *cmdblk;
{
    char *execmdp;

```

```

    cmdblk->cmd=READ_32CNT;
    cmdblk->rsp = 0xFFFF;
    cmdblk->ints = INTERRUPTS;
    cmdblk->rflgncmd = 0xFFFF;
    cmdblk->ncmd = 0xFFFFFFFF;
    cmdblk->btypbadm = ADDR_MOD;
    cmdblk->operand1 = 0x0000;
    cmdblk->operand2 = 0x00000000;

    cmdblkp(channel,cmdblk);

    execmdp=(char *) (0x82 + channel + SHORT_ADDR + MOD_ADDR);
    *execmdp=0x01;

}

/*****
start_counter(channel, cmdblk)
short channel;
SCMDBLK *cmdblk;
{
    char *execmdp;

    cmdblk->cmd=START_32CNT;
    cmdblk->rsp = 0xFFFF;
    cmdblk->ints = INTERRUPTS;
    cmdblk->rflgncmd = 0xFFFF;
    cmdblk->ncmd = 0xFFFFFFFF;
    cmdblk->btypbadm = ADDR_MOD;
    cmdblk->operand1 = 0x0000;
    cmdblk->operand2 = 0x00000000;

    cmdblkp(channel,cmdblk);

    execmdp=(char *) (0x82 + channel + SHORT_ADDR + MOD_ADDR);
    *execmdp=0x01;

}

*****/
FUNCTION: RS230(CHANNEL, CMDBLK)

This function is the software reset command for the XYCOM 230 card
*****/

```

```

rs230(channel, cmdblk)
short channel;
SCMDBLK *cmdblk;
{
    char *execmdp;

    cmdblk->cmd=BOARD_RESET;
    cmdblk->rsp = 0xFFFF;
    cmdblk->ints = INTERRUPTS;
    cmdblk->rflgncmd = 0xFFFF;
    cmdblk->ncmd = 0xFFFFFFFF;
    cmdblk->btypbadm = ADDR_MOD;
    cmdblk->operand1 = 0x0000;
    cmdblk->operand2 = 0x00000000;

    cmdblkp(channel,cmdblk);

    execmdp=(char *)(0x82 + channel + SHORT_ADDR + MOD_ADDR);
    *execmdp=0x01;

    swait(1);

    printf("done resetting board\n");
}

/*****/
swait(wait_period)
int wait_period;
{
    int cont=1;
    int start_time, finish_time;
    time_t start, finish;

    start_time=time(&start);
    while(cont)
    {
        finish_time=time(&finish);
        if((finish_time-start_time) >= wait_period)
            cont=0;
    }
}

/*****/
FUNCTION: WORLDJWI(LEFT_COUNT, RIGHT_COUNT, X_CUR, Y_CUR,
THETA_CUR)

```

This function does the VOC kinematics based on a procedure by Tsumura and Wang.

```

/*****
worldjwl(left_count, right_count, x_cur, y_cur, theta_cur)
int *left_count, *right_count;
double *x_cur, *y_cur, *theta_cur;
{
    double Na, Nd, adj;
    int Nr, Nl;
    double x_new, y_new;
    double d_theta, theta_new;

    Nl = *(left_count+1) - *left_count;
    Nr = *(right_count+1) - *right_count;

    Na = (Nl + Nr)/2.0;
    Nd = Nl-Nr;
    d_theta = 2*PI*RAD_WHEEL*Nd/(COUNTS_PER_REV*WHEEL_BASE);

    if(d_theta == 0.0)
        adj=1;
    else
        adj=sin(d_theta/2)/(d_theta/2);

    x_new = adj*2*PI*RAD_WHEEL*Na*cos((*theta_cur) +
d_theta/2)/COUNTS_PER_REV;
    y_new = adj*2*PI*RAD_WHEEL*Na*sin((*theta_cur) +
d_theta/2)/COUNTS_PER_REV;

    *x_cur = *x_cur + x_new;
    *y_cur = *y_cur + y_new;
    *theta_cur = *theta_cur + d_theta;
}

/*****/
int Init_PID_Module()
{
    int modsize;

    modsize = sizeof(id_dm);
    if((pid_dataptr = (id_dm *)dmod_link(PID_MOD_NAME))==(id_dm *)ERROR)
        if((pid_dataptr = (id_dm *)dmod_create(PID_MOD_NAME,modsize))==
        (id_dm *)ERROR){
            printf("Init_PID_Module: Failed to link to module\n");

```

```

        return(FALSE);
    }
    return(TRUE);
}
/*****
int Init_WS_Module()
{
    int modsize;

    modsize = sizeof(wsdm);
    if((ws_dataptr = (wsdm *)dmod_link(WS_MOD_NAME))==(wsdm *)ERROR)
        if((ws_dataptr = (wsdm *)dmod_create(WS_MOD_NAME,modsize))==
            (wsdm *)ERROR){
            printf("Init_WS_Module: Failed to link to module\n");
            return(FALSE);
        }
    return(TRUE);
}
*****/
int Init_VOC_Module()
{
    int modsize;

    modsize = sizeof(VOC_Path);
    if((voc_dataptr = (VOC_Path *)dmod_link(VOC_MOD_NAME))==(VOC_Path *)ERROR)
        if((voc_dataptr = (VOC_Path *)dmod_create(VOC_MOD_NAME,modsize))==(VOC_Path
*)ERROR){
            printf("Init_VOC_Module: Failed to link to module\n");
            return(FALSE);
        }
    return(TRUE);
}
*****/
int Init_COUNT_Module()
{
    int modsize;

    modsize = sizeof(count_dm);
    if((count_dataptr = (count_dm *)dmod_link(COUNT_MOD_NAME))==(count_dm *)ERROR)
        if((count_dataptr = (count_dm
*)dmod_create(COUNT_MOD_NAME,modsize))==(count_dm *)ERROR){
            printf("Init_COUNT_Module: Failed to link to module\n");
            return(FALSE);
        }
    return(TRUE);
}

```

```

}
/*****
int Init_PP_Module()
{

    unsigned modsize;

    modsize = sizeof(PP_Path);

    if((pp_dataptr=(PP_Path *)dmod_link(PP_MOD_NAME))==(PP_Path *)ERROR)
        if((pp_dataptr=(PP_Path *)dmod_create(PP_MOD_NAME,modsize))==
            (PP_Path *)ERROR) {
            printf("Init_PP_Module: Could not link or create module\n");
            return(FALSE);
        }
    return(TRUE);
}
*****/

void WriteWSModule(min_i, max_i)
int *min_i, *max_i;
{
    ws_dataptr->min_row=*min_i;
    ws_dataptr->max_row=*max_i;
}
/*****
void ReadPIDModule()
{
    icu_pid = pid_dataptr->icu_pid;
    pp_pid = pid_dataptr->pp_pid;
}
*****/

void Close_Data_Modules()
{
    if(!dmod_unlink(COUNT_MOD_NAME))
        printf("Close_Data_Modules: Couldn't unlink from the COUNT Module\n");
    if(!dmod_unlink(PP_MOD_NAME))
        printf("Close_Data_Modules: Couldn't unlink from the PP Module\n");
    if(!dmod_unlink(VOC_MOD_NAME))
        printf("Close_Data_Modules: Couldn't unlink from the VOC Module\n");
    if(!dmod_unlink(WS_MOD_NAME))
        printf("Close_Data_Modules: Couldn't unlink from the WS Module\n");
    if(!dmod_unlink(PID_MOD_NAME))
        printf("Close_Data_Modules: Couldn't unlink from the PID Module\n");
}

```

```
/**/
```

```
sighand(sigvalue)
short sigvalue;
{
    switch(sigvalue)
    {
        case VOC_END_PROGRAM:
            FLAG_VOC_END_PROGRAM=TRUE;
            break;
        case VOC_CURRENT_LOC:
            FLAG_VOC_CURRENT_LOC=TRUE;
            break;
        case WS_CORD_REQ:
            FLAG_WS_CORD_REQ=TRUE;
            break;
        case PP_DATA_READY:
            FLAG_PP_DATA_READY=TRUE;
            break;
        case RESET_SYSTEM:
            FLAG_RESET_SYSTEM=TRUE;
            break;
        case 3:
            FLAG_RESET_SYSTEM=TRUE;
            break;
        case UPDATE_PID:
            FLAG_UPDATE_PID=TRUE;
            break;
        default:
            printf("\nvoc: >%s< unknown signal: %x\n",sigvalue);
    }
}
```

```
/**/
```


APPENDIX B - VEHICLE ORIENTATION AND CONTROL HARDWARE TEST PROGRAM

```
/*          VOCTEST.C for H68/VME3E
*
*          TESTS VOC-VSS HARDWARE TO ENSURE IT IS FUNCTIONING
*
*          BY JONATHAN WANJIRU
*          MAY/20/94
*/

#include <time.h>
#include <math.h>
#include <strings.h>
#include <signal.h>

#include "/h0/demo/include/common.h"
#include "/h0/demo/source/dmod.c"
#include "xvme230m.h"

#define TESTLENGTH          1000 /* maximum testing cycles this program will perform
    */

char *argv[MAXLEN];

main(argc, argv)
unsigned argc;
char *argv[];
{

    int i, jj, channel=0, mode;
    long data[MAXCHANNEL][MAXLEN];
    double xcord1, ycord1, theta1, measured_n1, measured_nr, measured_x;
    double xcord2, ycord2, theta2, wheel_base, mag1, mag2;
    char *execmdp;
    SCMDBLK *cmdblk[MAXCHANNEL];
    double *x_cur, *y_cur, *theta_cur;
    double bi, ci, ai, xi, yi, ximinus1, yiminus1;

    xcord1=ycord1=theta1=xcord2=ycord2=theta2=0.0;
    ximinus1=yiminus1=theta1=0.0;

    if(VOC_DEBUG)          /* output file for debugging information */
```

```

{
    if (( stream=fopen("voc01114.dat","a+") == NULL)
    {
        printf("\nvoc:could not open >%s<\n", "voc01114.dat");
        exit(0);
    }
}

cmdblk[0]=(SCMDBLK *)(CMDBLK_ADDR0+SHORT_ADDR + MOD_ADDR );
cmdblk[1]=(SCMDBLK *)(CMDBLK_ADDR1+SHORT_ADDR + MOD_ADDR );
cmdblk[2]=(SCMDBLK *)(CMDBLK_ADDR2+SHORT_ADDR + MOD_ADDR );
cmdblk[3]=(SCMDBLK *)(CMDBLK_ADDR3+SHORT_ADDR + MOD_ADDR );
cmdblk[4]=(SCMDBLK *)(CMDBLK_ADDR4+SHORT_ADDR + MOD_ADDR );

channel=4; /*      software reset counter card before starting the test run      */
rs230(channel,cmdblk[channel]);
printf("wait for a little bit \n");
swait(3);

printf("\nstarting start\n");
for(channel=0;channel <MAXCHANNEL-1; channel++)
    start_counter(channel, cmdblk[channel]);
swait(1);
mode=1;
for(channel=0;channel <MAXCHANNEL-1; channel++)
    chkcmd(mode, channel, cmdblk[channel]);

printf("\nstarting reading\n");
channel =RIGHT_ENCODER;
read_counter(channel, cmdblk[channel]);
channel =LEFT_ENCODER;
read_counter(channel, cmdblk[channel]);

/*      poll card for completion of reading      */
mode=2;
channel =RIGHT_ENCODER;
chkcmd(mode, channel, cmdblk[channel]);
channel =LEFT_ENCODER;
chkcmd(mode, channel, cmdblk[channel]);

i=0;
channel =RIGHT_ENCODER;
data[channel][0]= cmdblk[channel]->operand2;

```

```

printf("chan %d: %d ", channel, data[channel][i]);
channel =LEFT_ENCODER;
data[channel][0]= (cmdblk[channel]->operand2);
printf("chan %d: %d ", channel, data[channel][i]);

for(jj=1;jj<TESTLENGTH;jj++)
{
channel =RIGHT_ENCODER;
read_counter(channel, cmdblk[channel]);
channel =LEFT_ENCODER;
read_counter(channel, cmdblk[channel]);

mode=2;
channel =RIGHT_ENCODER;
chkcmd(mode, channel, cmdblk[channel]);
channel =LEFT_ENCODER;
chkcmd(mode, channel, cmdblk[channel]);

printf("\nline:%d >\t",jj);

channel =RIGHT_ENCODER;
data[channel][1]=cmdblk[channel]->operand2;
channel =LEFT_ENCODER;
data[channel][1]=(cmdblk[channel]->operand2);

printf("le: %d: \tre: %d ", data[LEFT_ENCODER][0],data[RIGHT_ENCODER][0]),

worldjwl(&(data[LEFT_ENCODER][0]),&(data[RIGHT_ENCODER][0]),&xcord2,&ycord2,&
theta2);

channel =RIGHT_ENCODER;
data[channel][0]=data[channel][1];
channel =LEFT_ENCODER;
data[channel][0]=data[channel][1];
} /* end of for loop*/

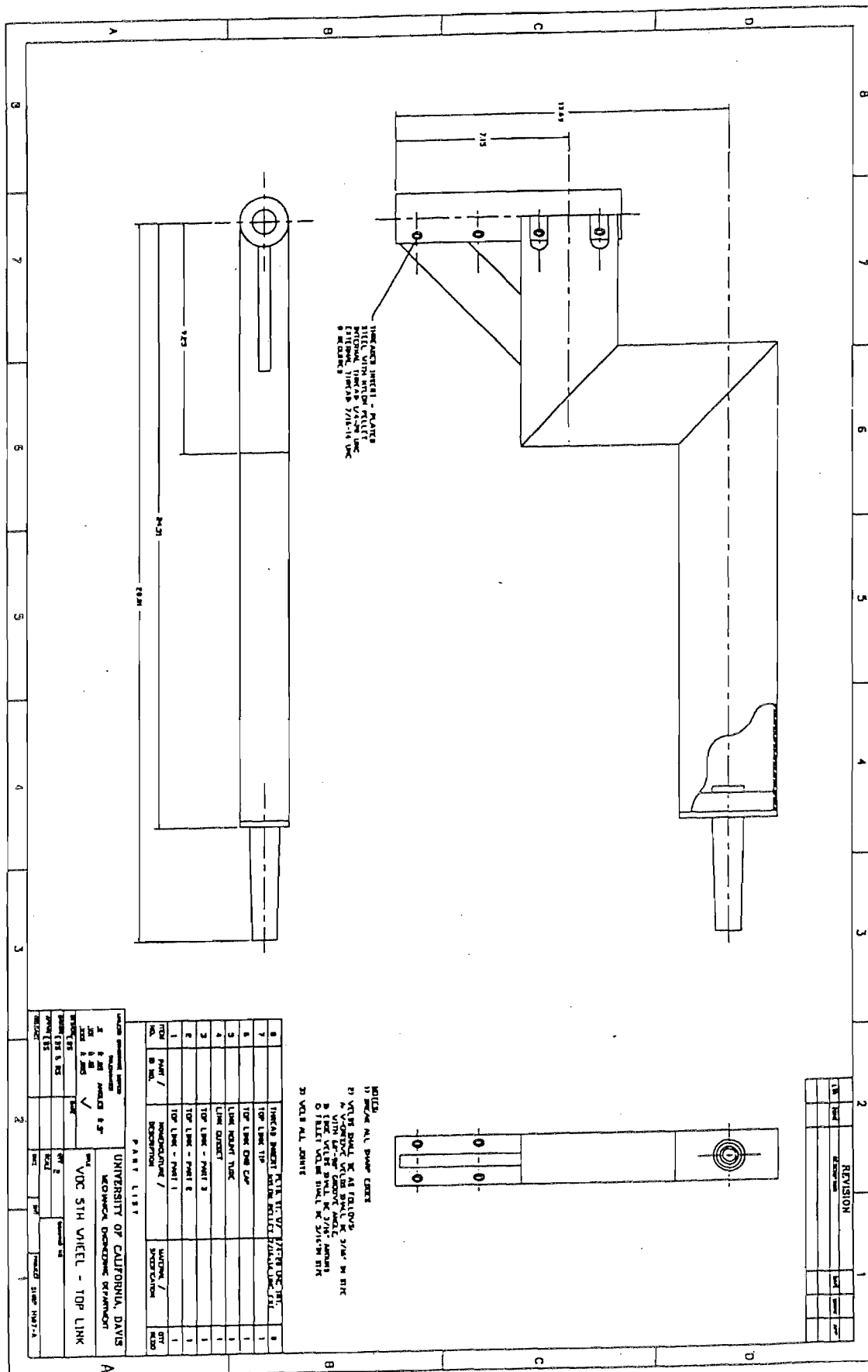
printf("\n\n%s: x:%fty:%fttheta:%f", argv[0],xcord2,ycord2,theta2);

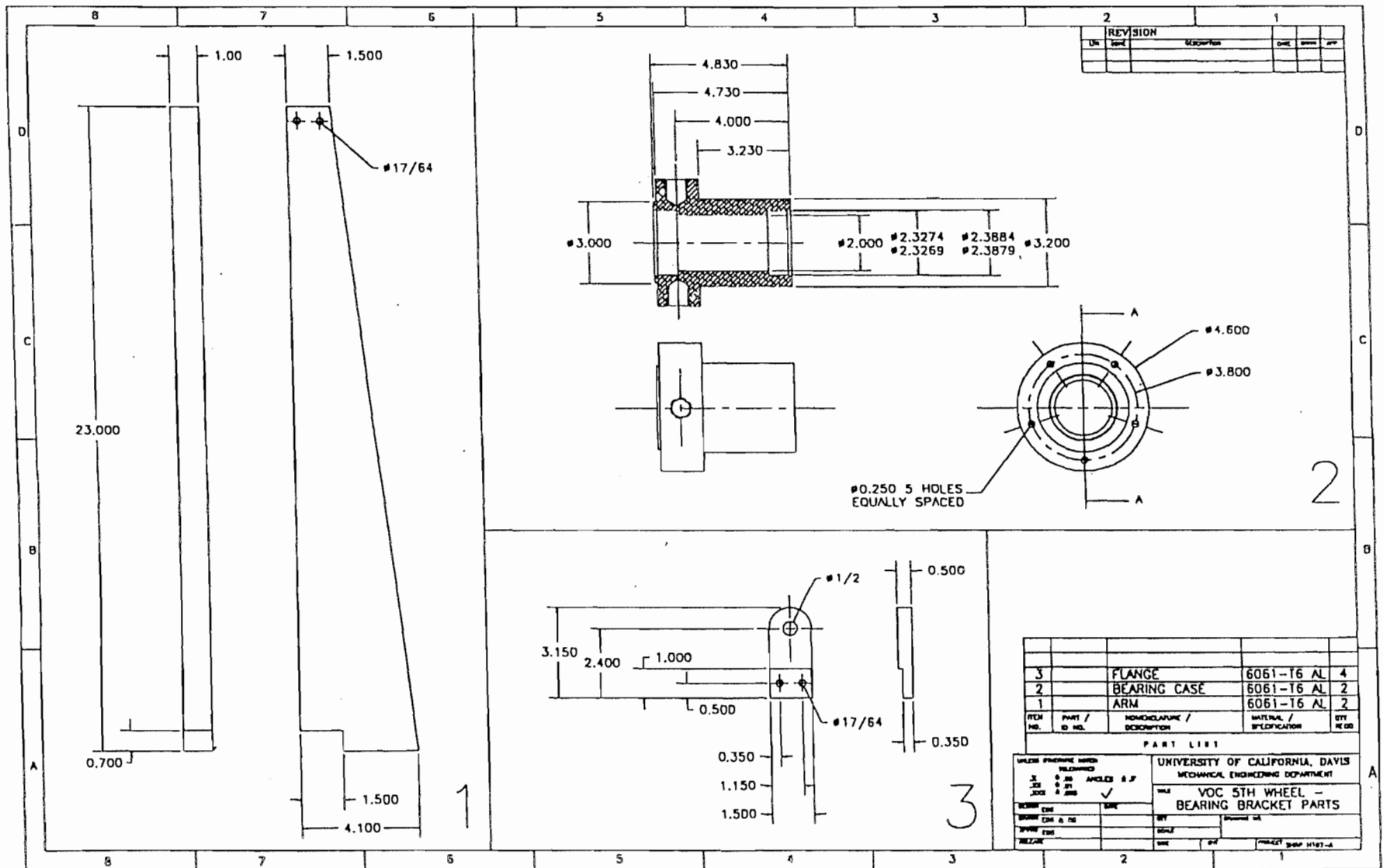
fprintf(stream, "\nle: %d: \tre: %d ", data[LEFT_ENCODER][0],data[RIGHT_ENCODER][0]);
fprintf(stream, "\n%s: x:%fty:%fttheta:%f", argv[0],xcord2,ycord2,theta2);
printf("\nend of program\n");
}

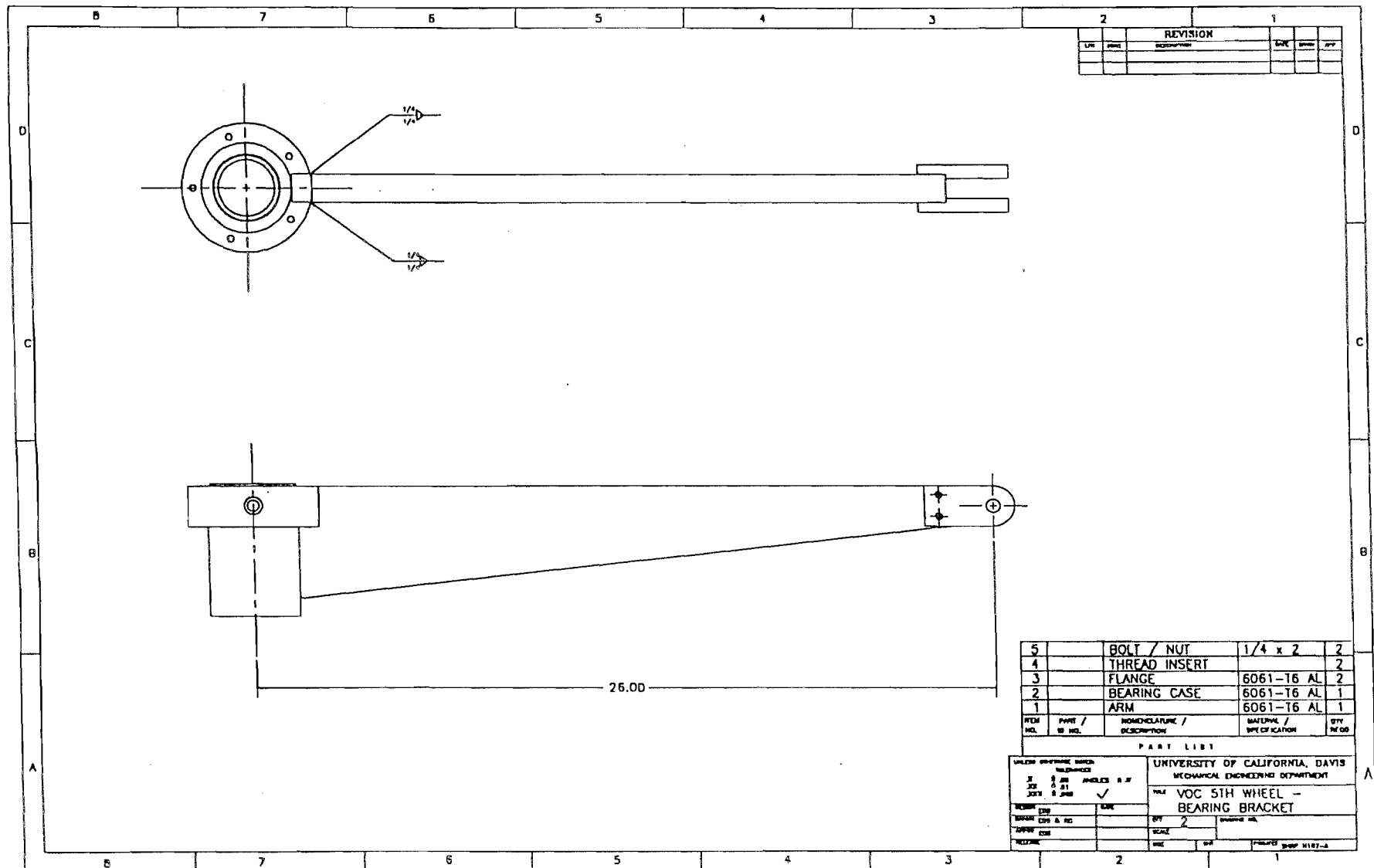
```

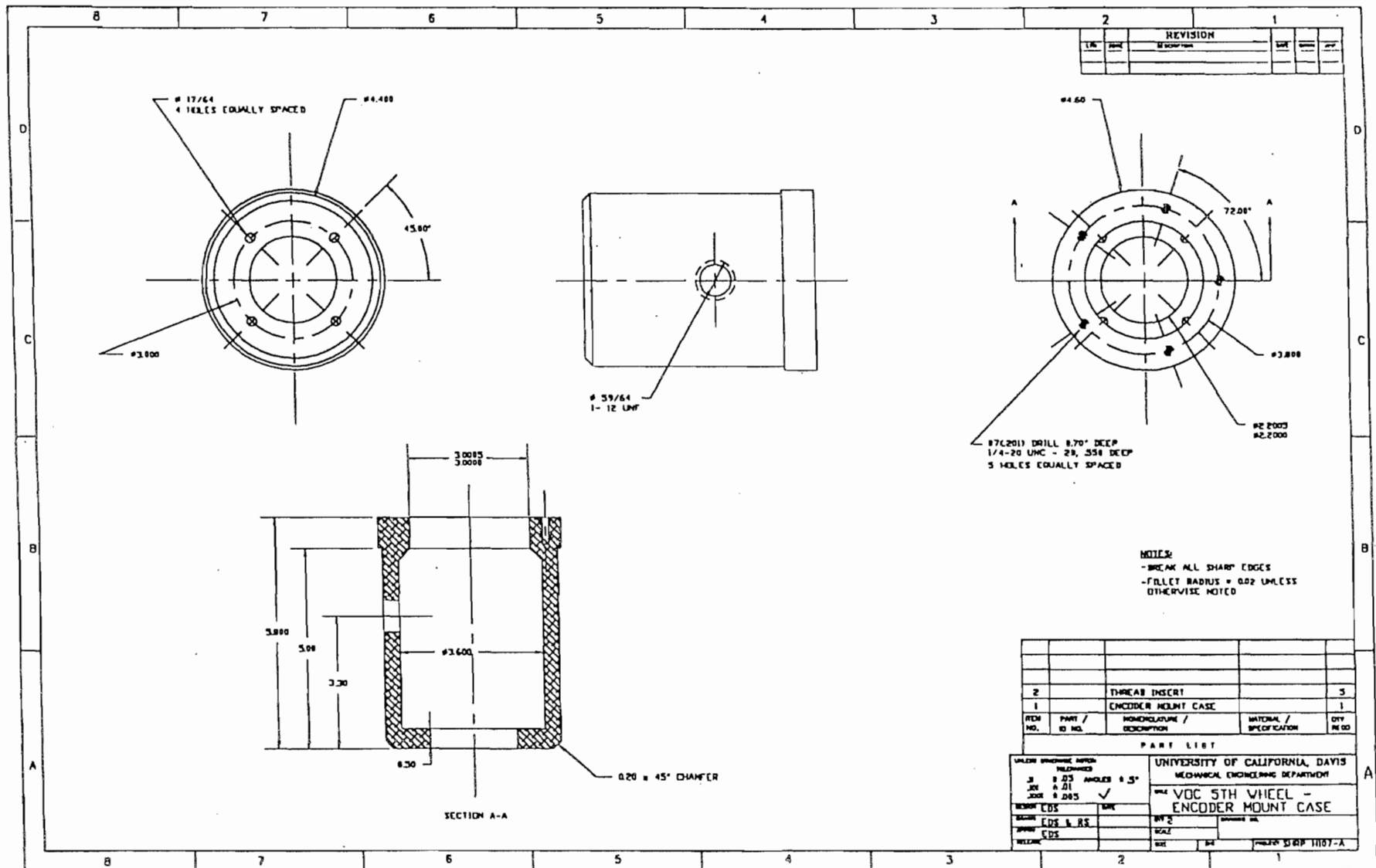
/*****

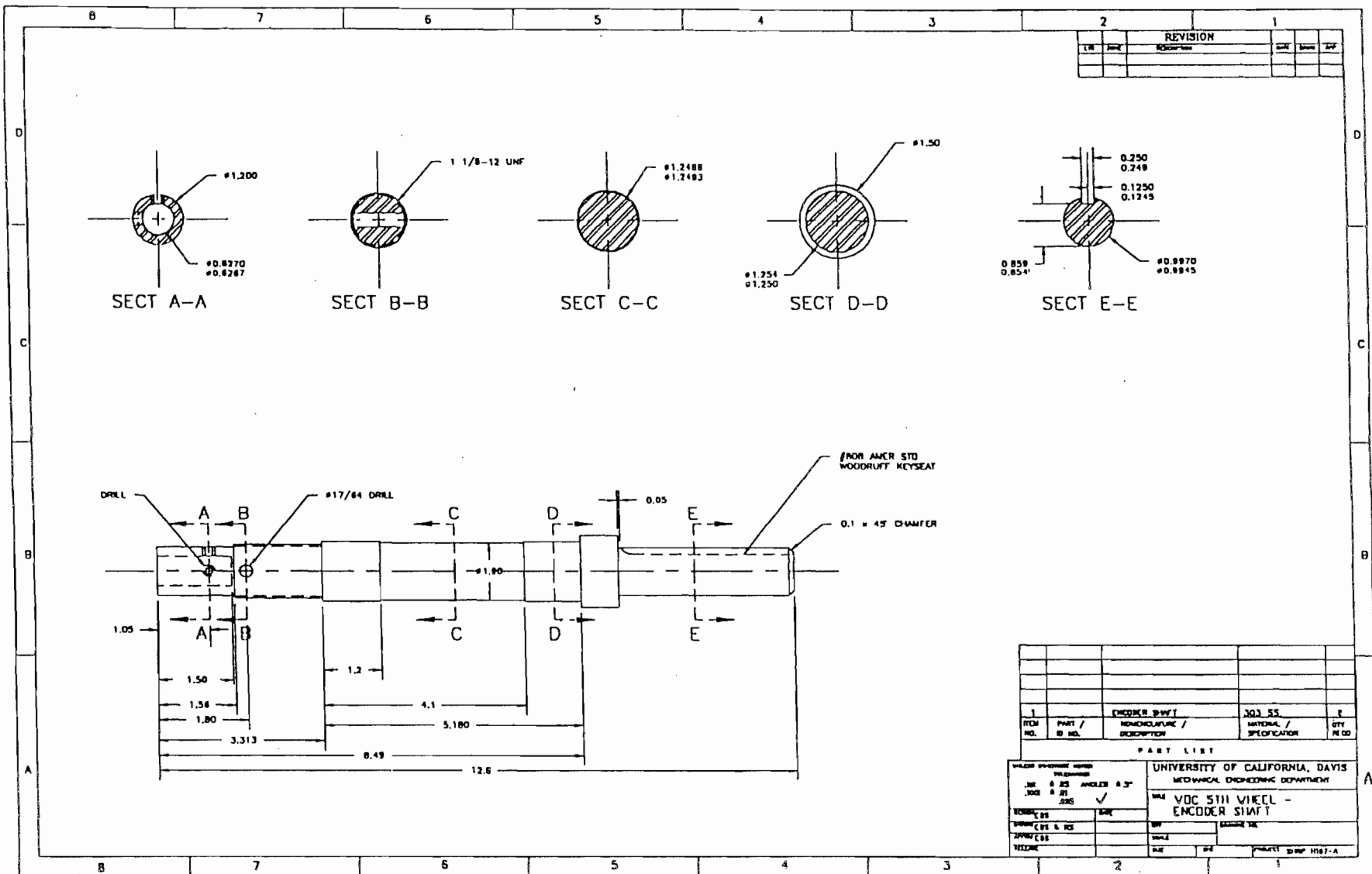
APPENDIX C - FIFTH WHEEL ASSEMBLY DRAWINGS

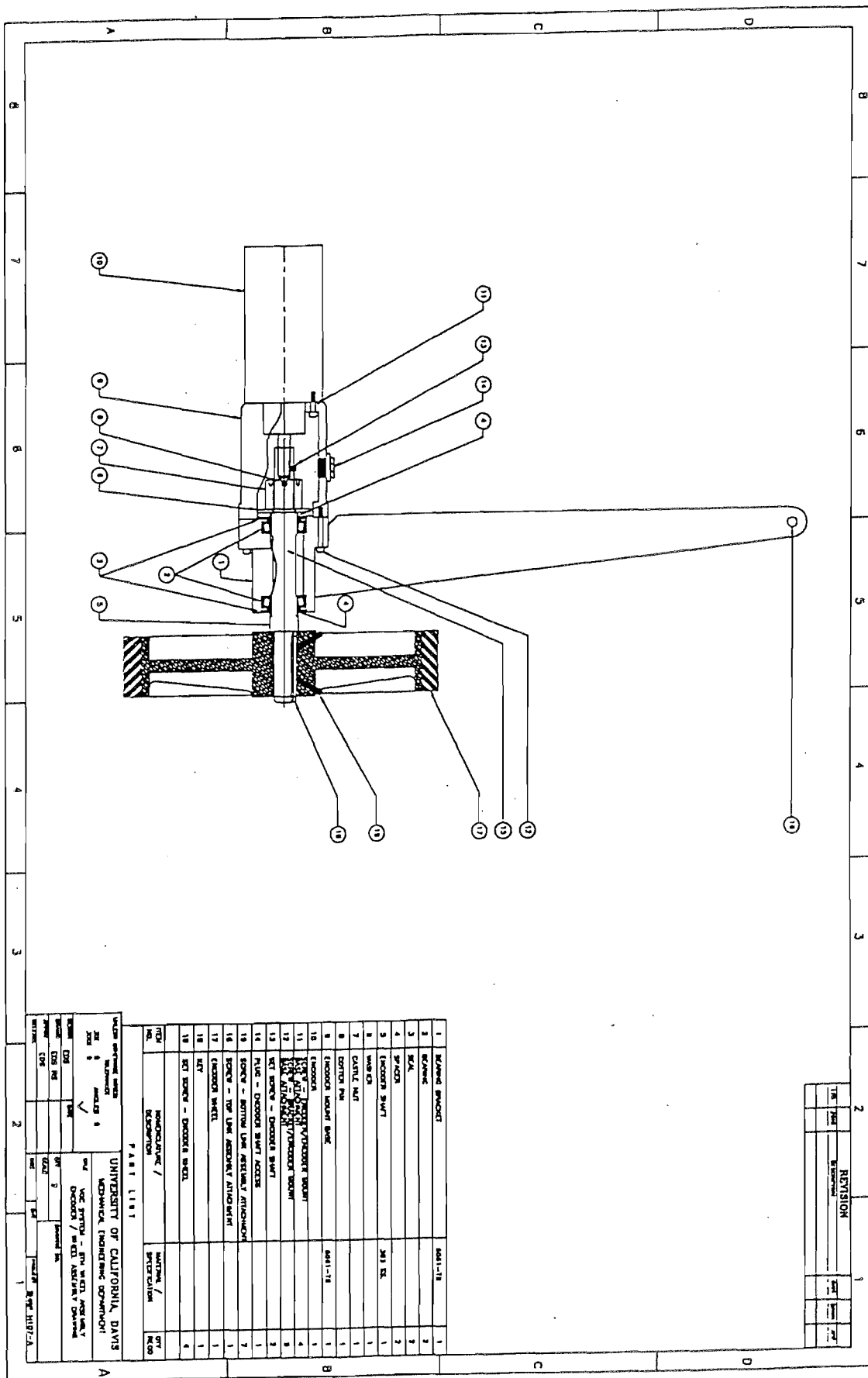








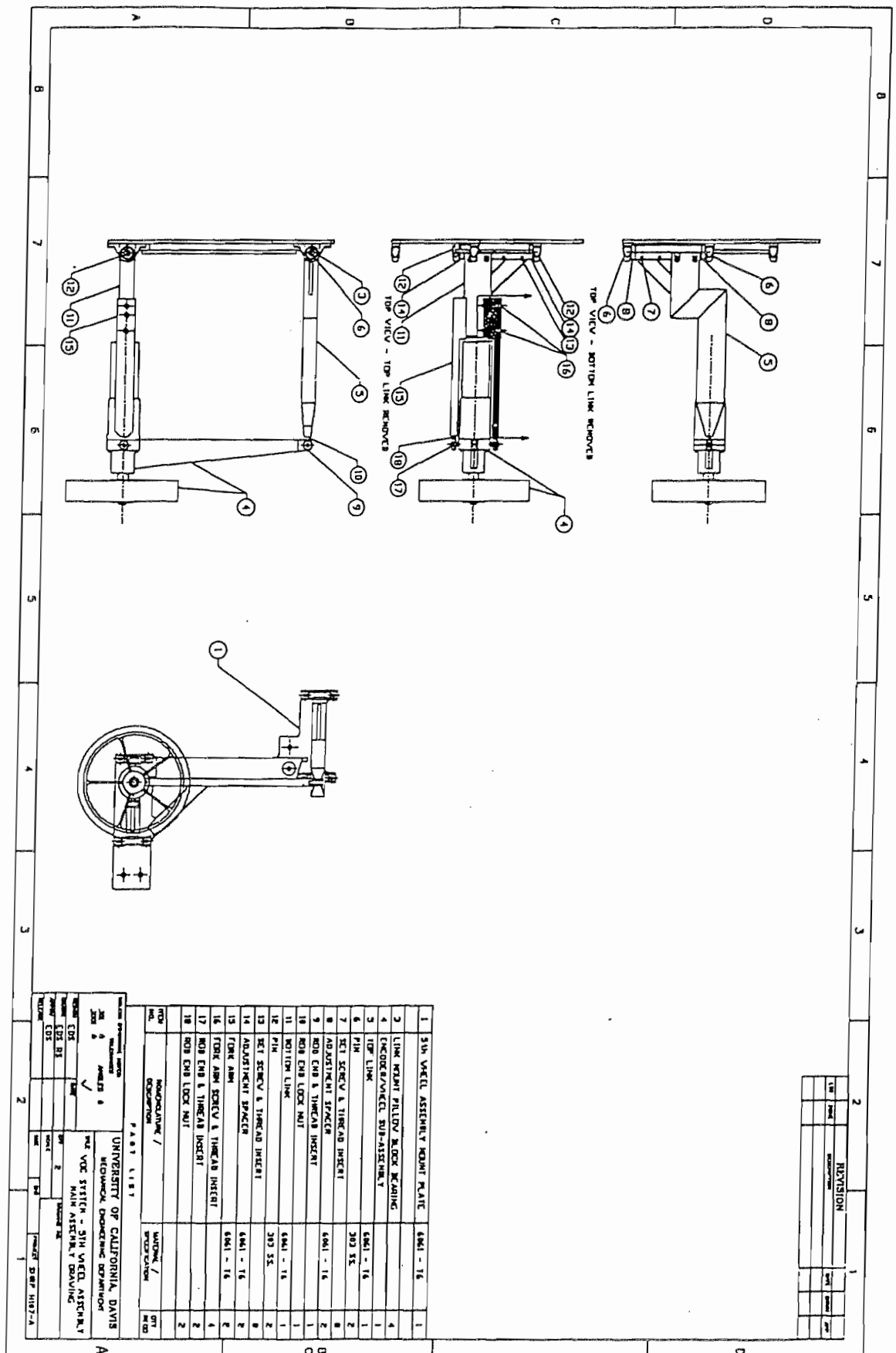


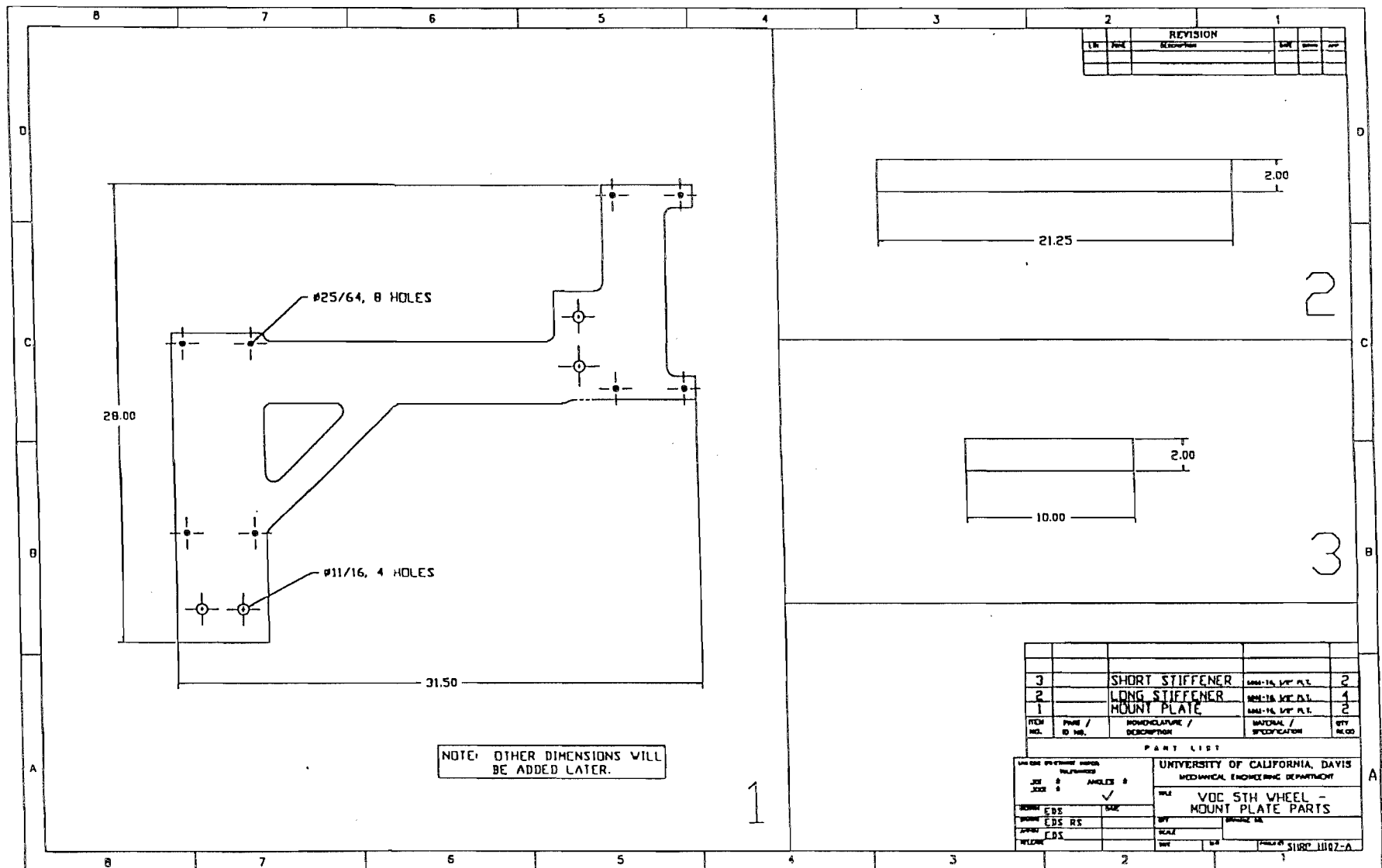


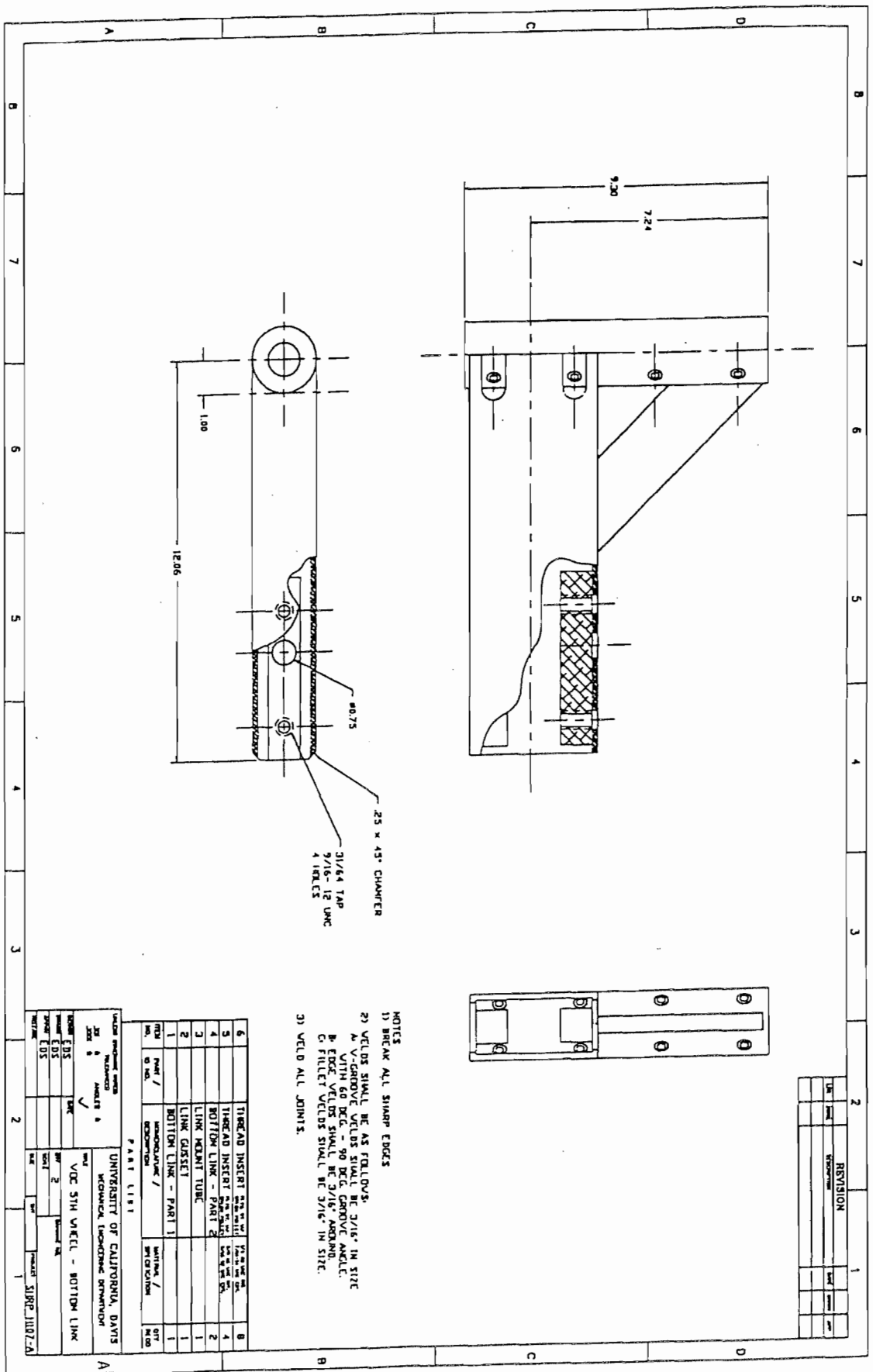
REVISION		1	2
REV	DESCRIPTION	DATE	BY

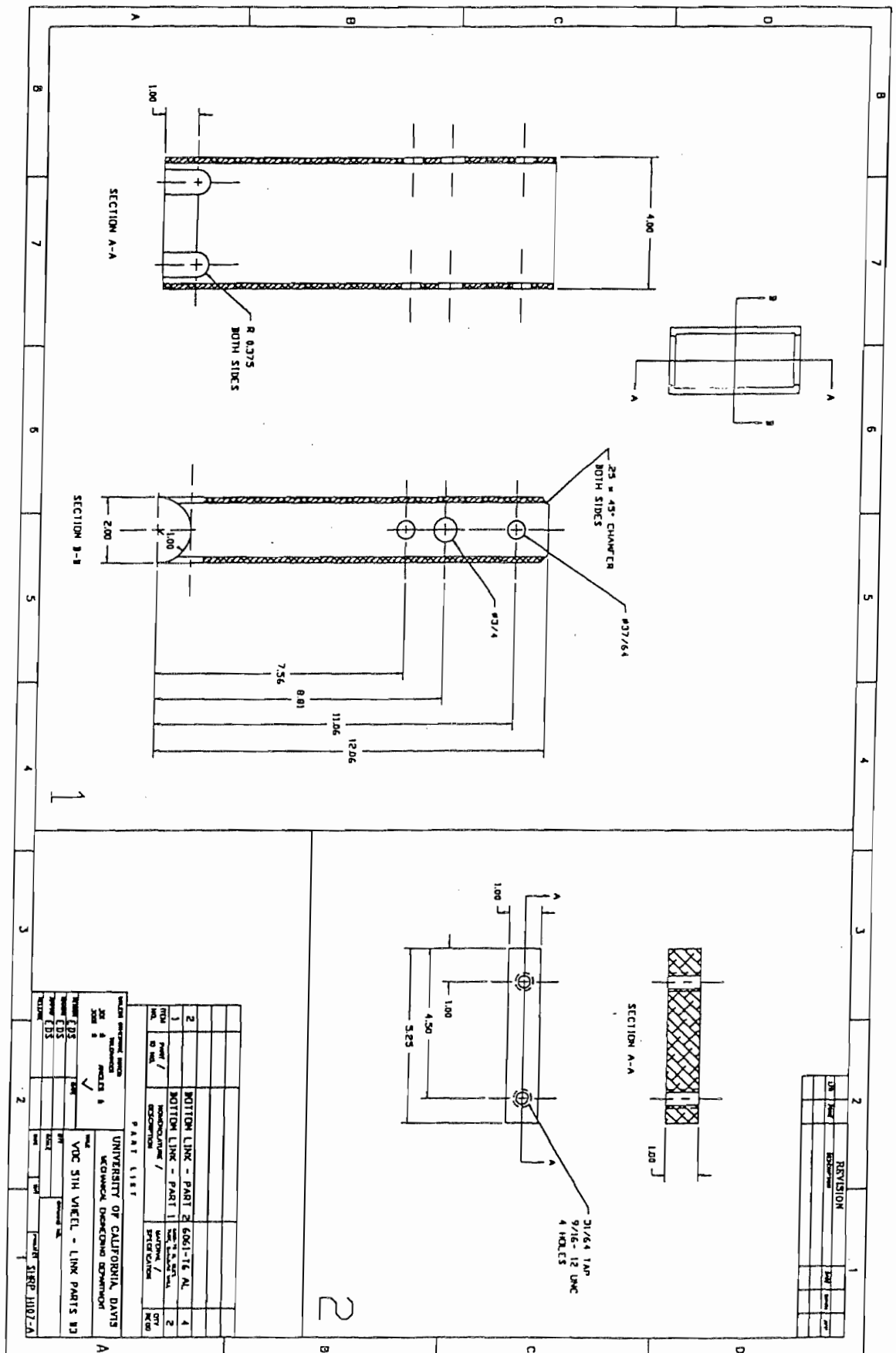
1	BEARING BRACKET	4441-18	1
2	ROTOR		2
3	BEEL		2
4	SPACER	241 EX	1
5	ENDOSC SHAFT		1
6	WASER		1
7	CASTLE NUT		1
8	ROTOR PIN		1
9	ENDOSC HOIST BOLT	4441-18	1
10	ENDOSC		1
11	TOP V - 1/2" DIA. V-LOCKER BUSH		4
12	TOP V - 1/2" DIA. V-LOCKER BUSH		4
13	SET SCREW - ENDOSC SHAFT		2
14	PLUG - ENDOSC SHAFT ACCESS		1
15	SCREW - BOTTOM LINE ASSEMBLY ATTACHMENT		7
16	SCREW - TOP LINE ASSEMBLY ATTACHMENT		1
17	ENDOSC WHEEL		1
18	SET SCREW - ENDOSC WHEEL		1
19	SET SCREW - ENDOSC WHEEL		4
NOTES	NONSTANDARD / MATERIAL / SPECIFICATION		
NO.	REV		

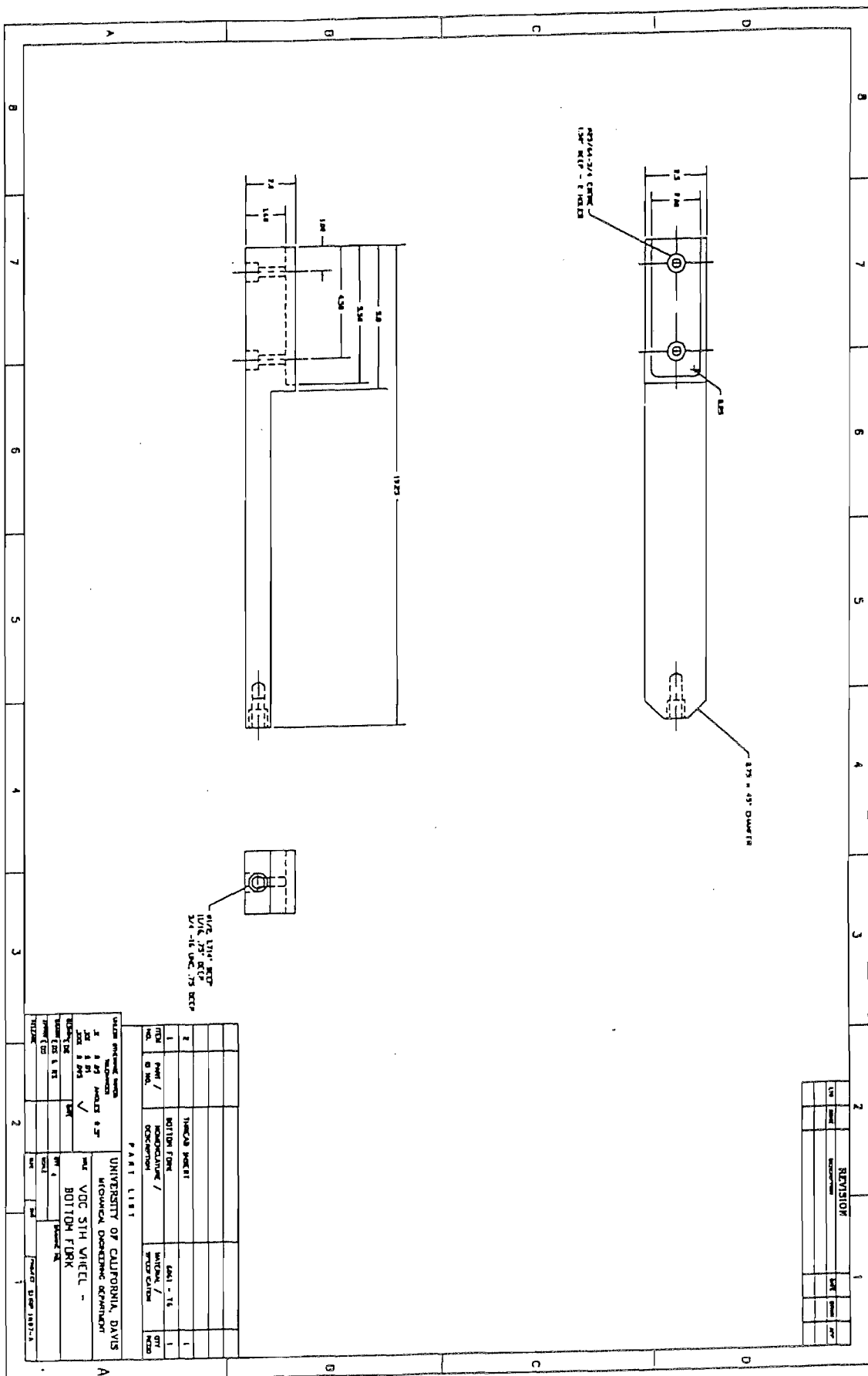
CHECKED BY:
 DESIGNED BY:
 DATE:
 UNIVERSITY OF CALIFORNIA, DAVIS
 MECHANICAL ENGINEERING DEPARTMENT
 400 PETERSON - 5TH FLOOR, AGRICULTURAL
 ENGINEERING / MECHANICAL ENGINEERING
 BUILDING
 DAVIS, CA 95616
 TEL: (916) 752-1111
 FAX: (916) 752-1111

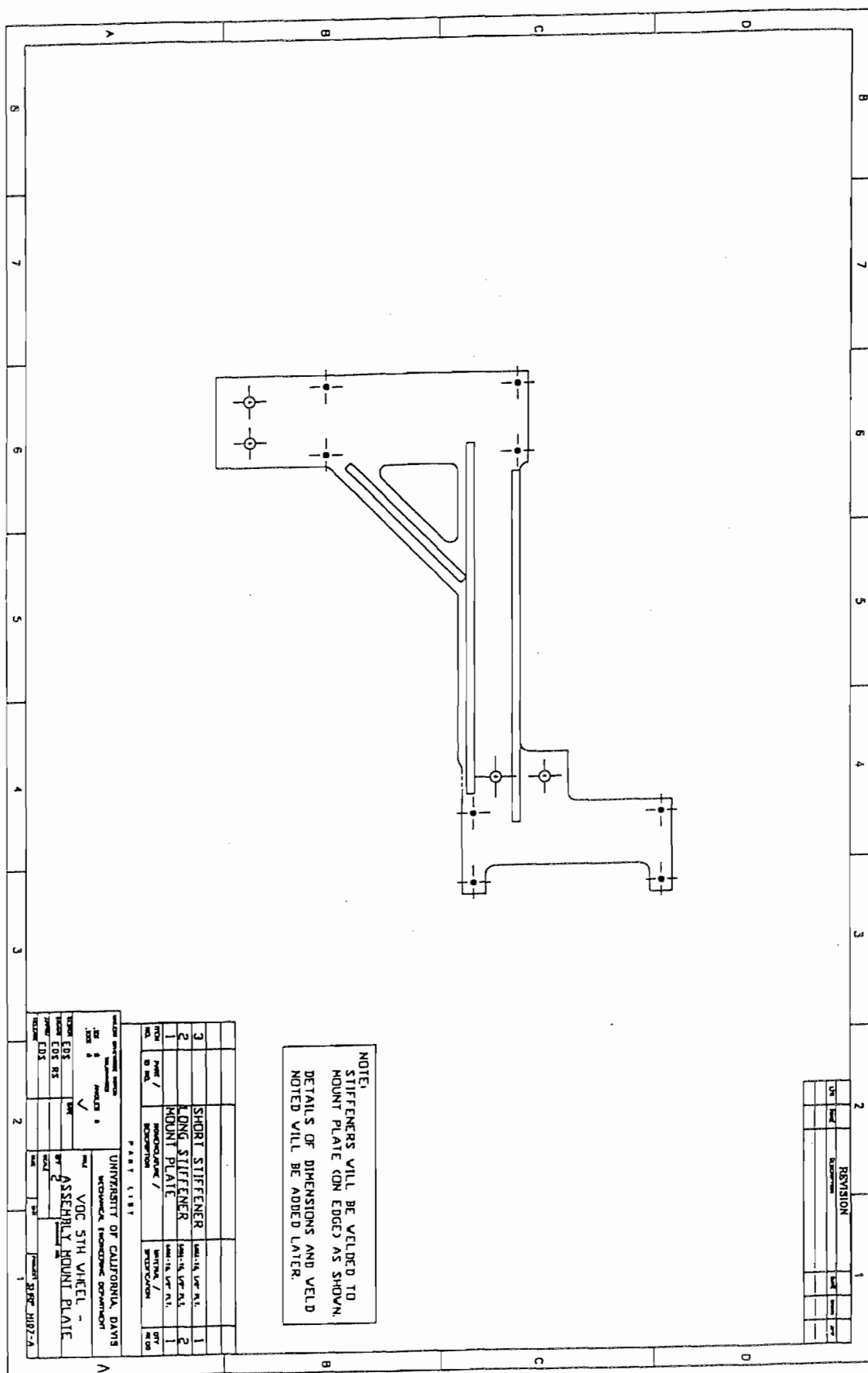




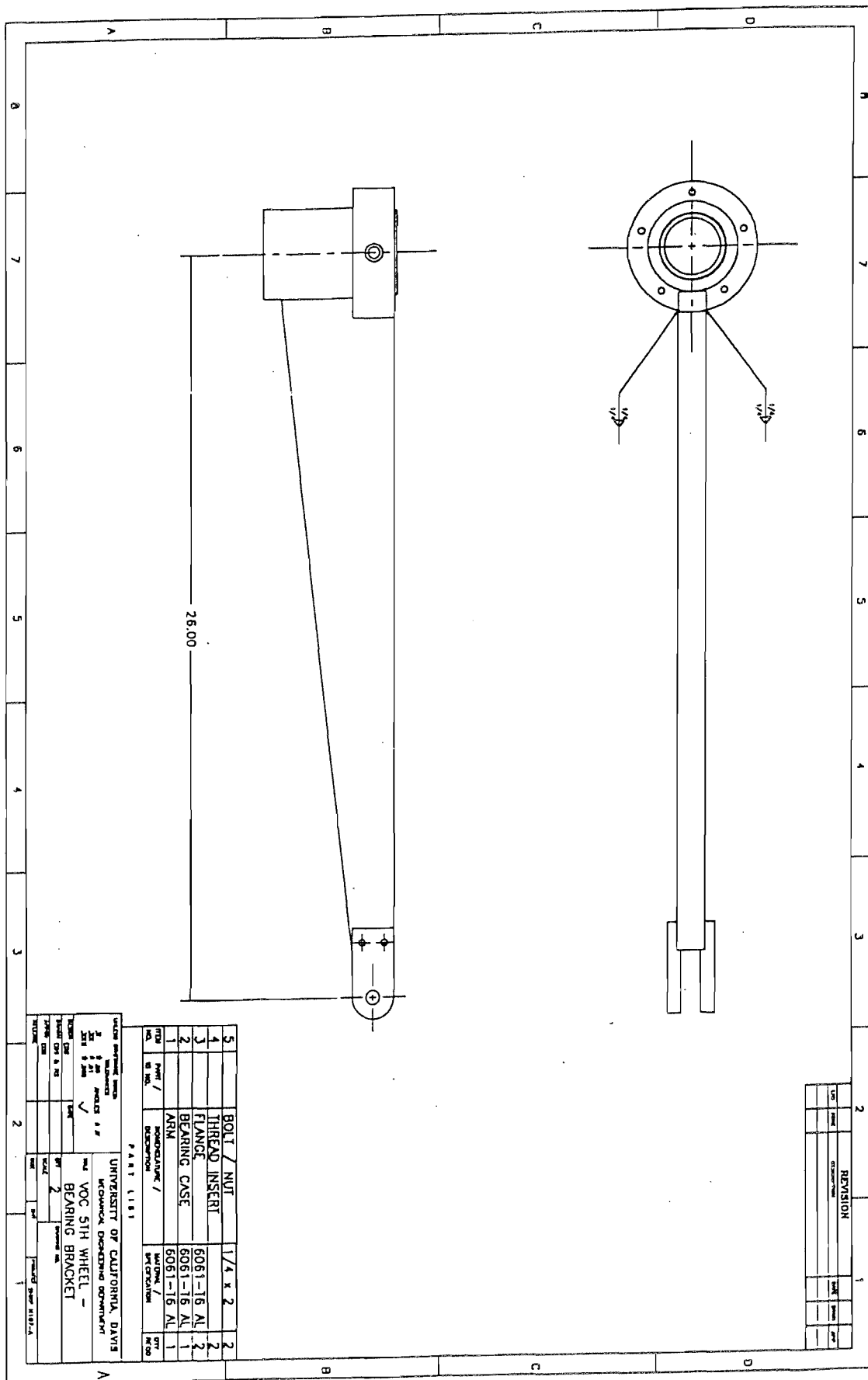


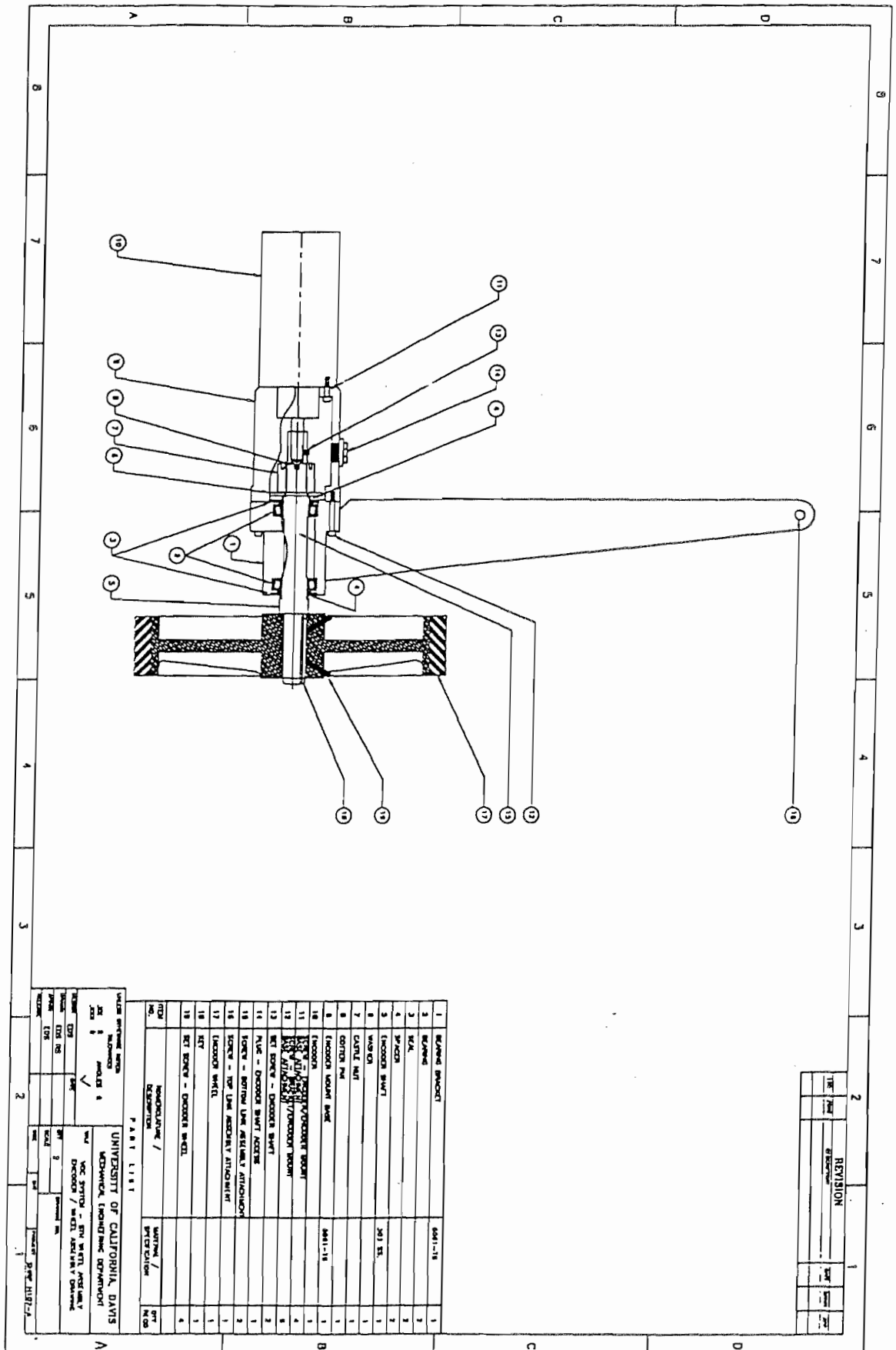


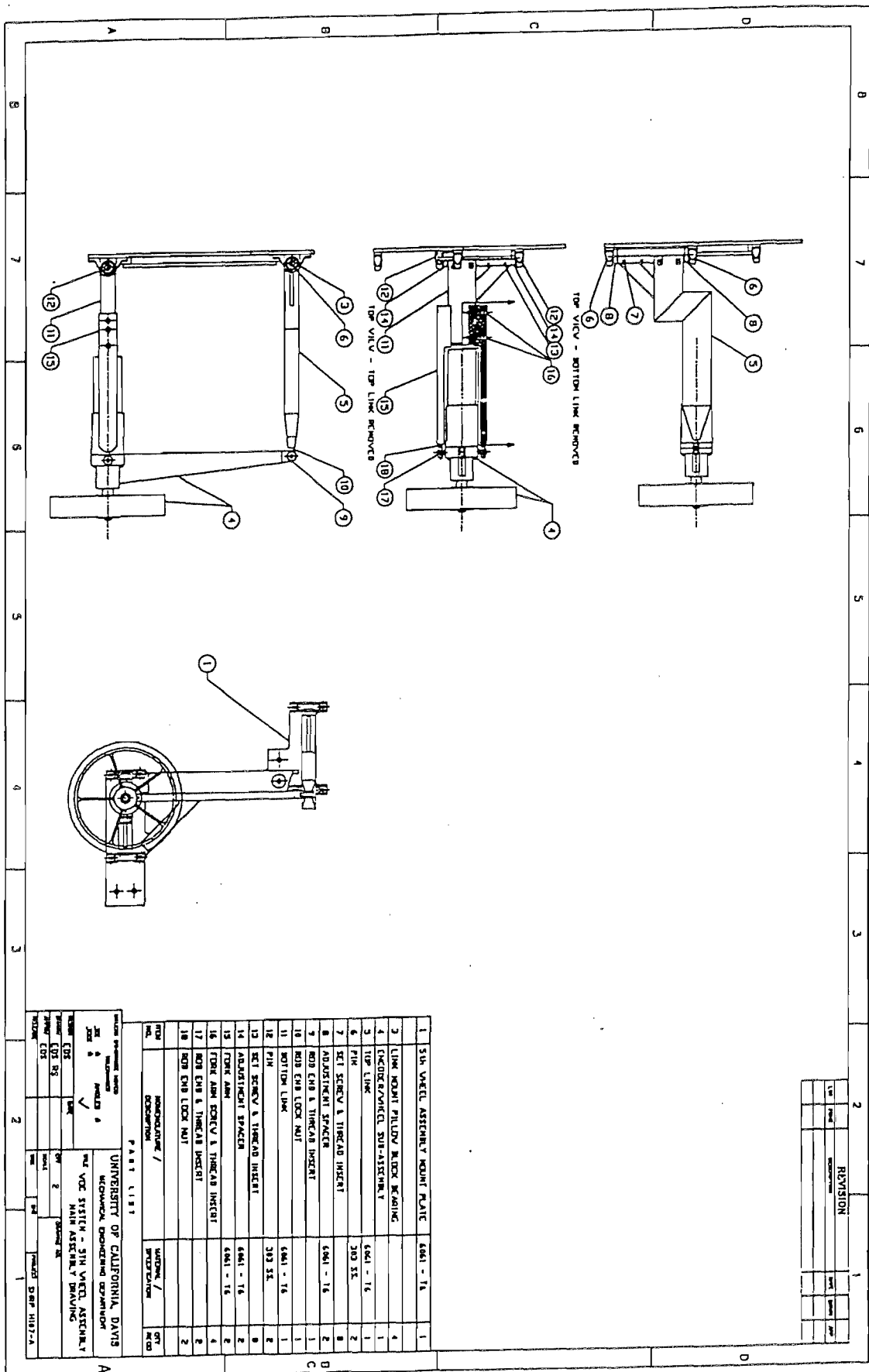


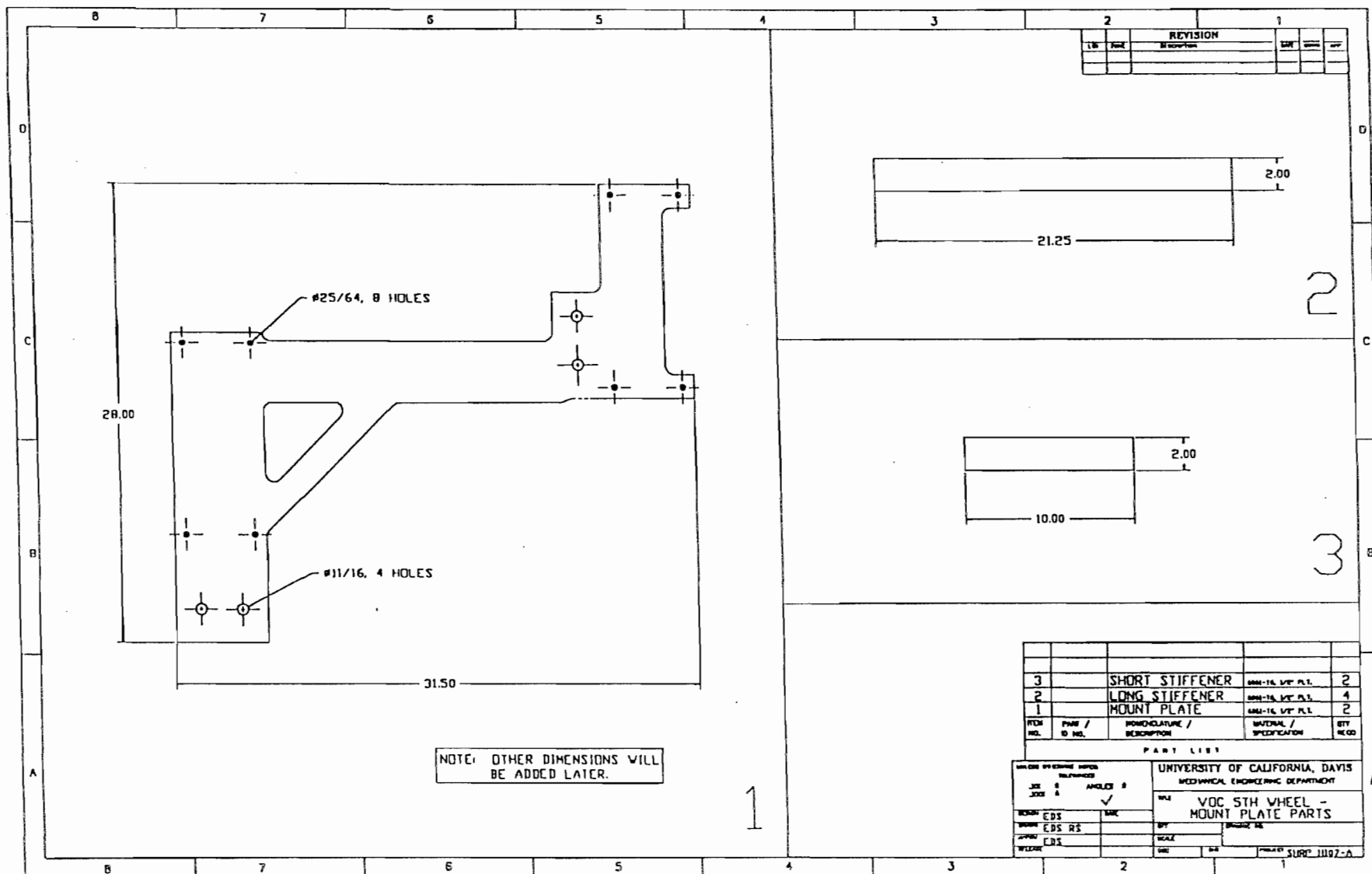


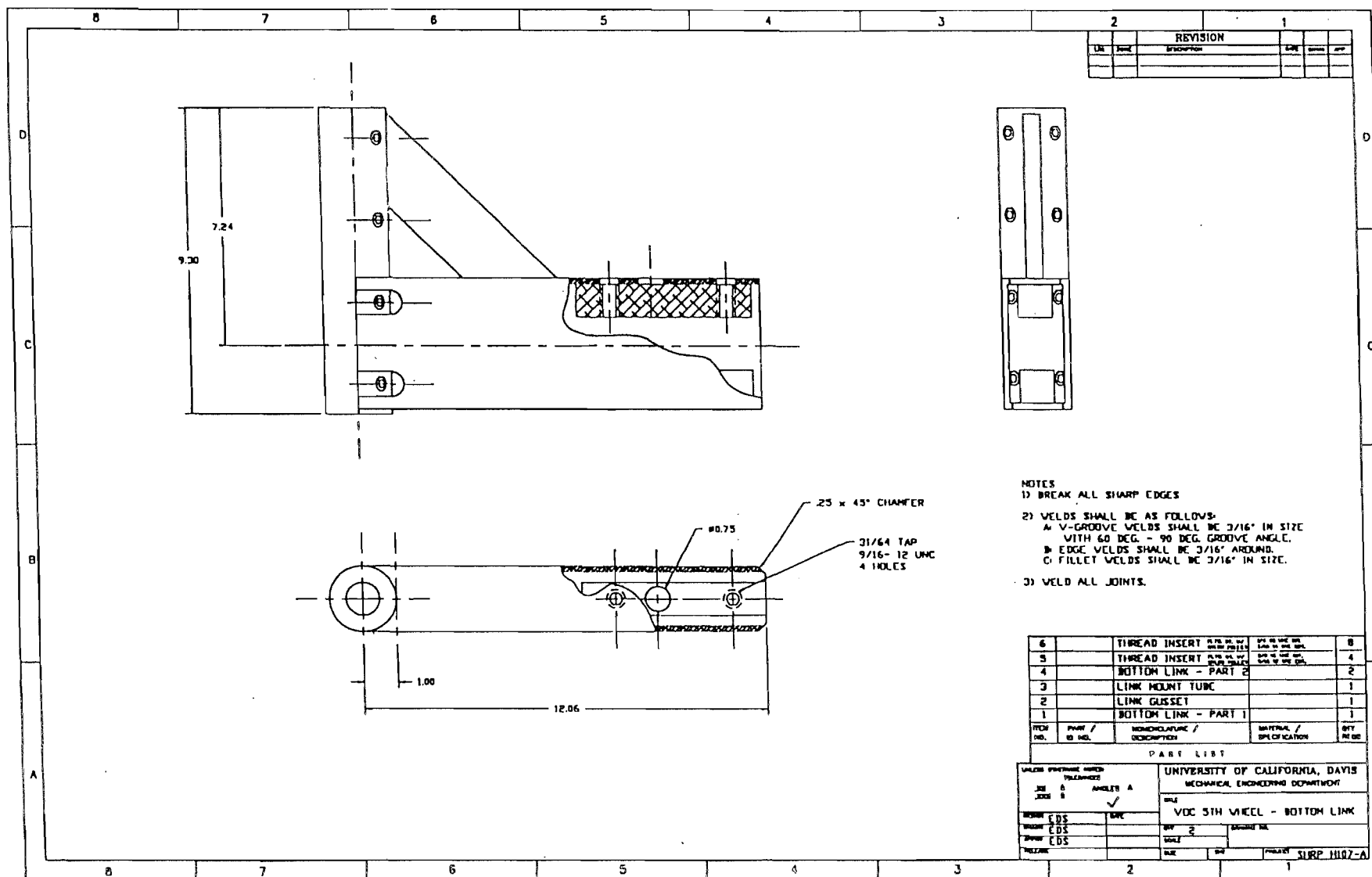


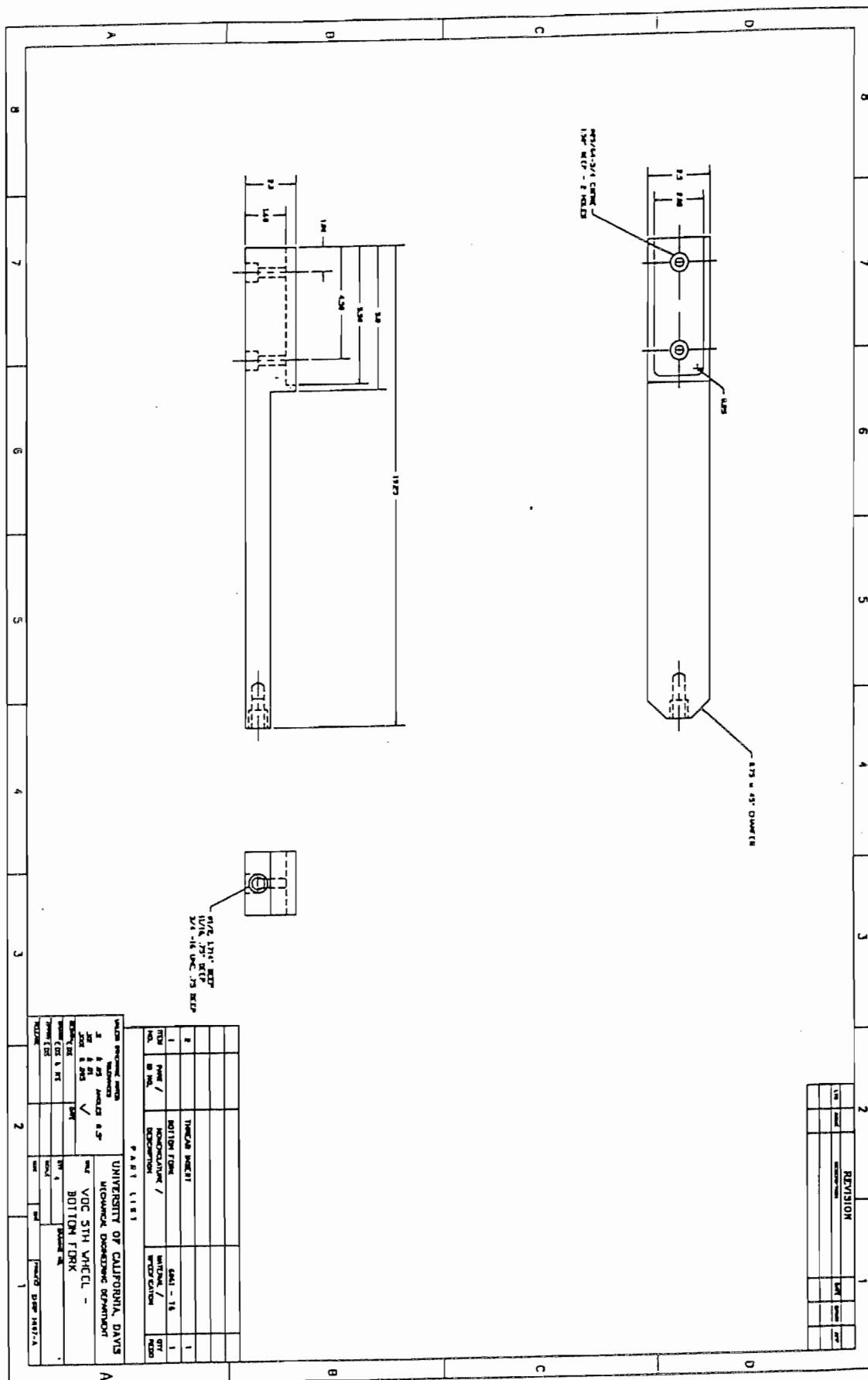




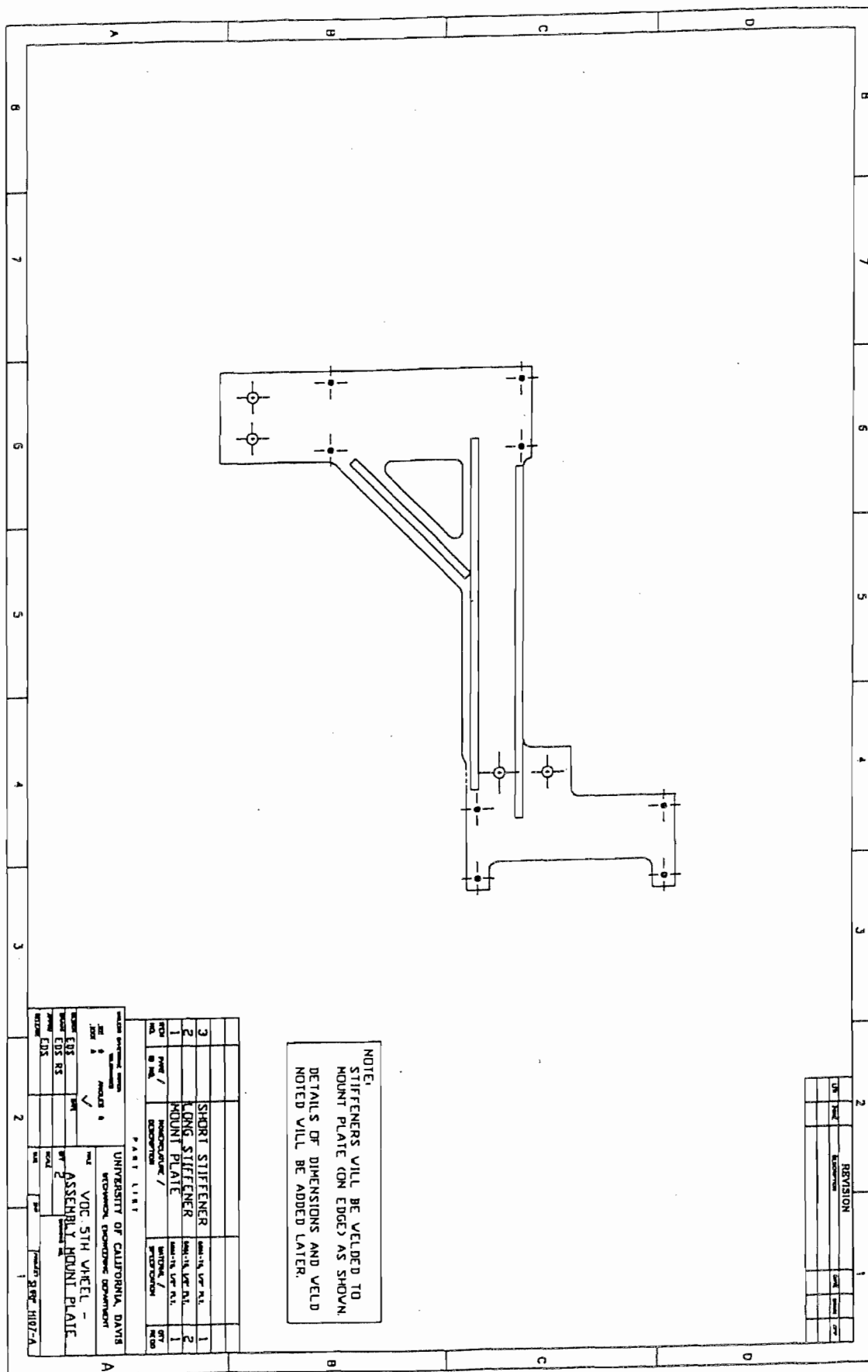












APPENDIX D - MANUFACTURER'S SPECIFICATIONS

SPECIFICATIONS

MECHANICAL

Input Shaft Torque at 25°C: 10 in-oz maximum
 Bearing Life/Shaft Loading: (see figure 1)
 Shaft Material:
 standard: chrome steel case hardened Rc 59 to 65,
 030/055 deep
 optional: stainless steel shaft available for use in
 highly corrosive environments
 Housing Material:
 standard: cast aluminum with chemical film (iridite) finish
 optional: cast aluminum with Mil-A-8625 type III hard
 anodize with dichromate sealed finish
 Bearing Seals: bearings are neoprene sealed against water
 and dirt
 Shaft Seal: standard on all units
 Internal Coupling Windup: The angular error at the encoder
 is a function of acceleration. Error = 4×10^{-3}
 degrees/radian/sec² maximum
 Weight: approximately 9 lbs.

OUTPUT TERMINATIONS

Either conduit or MS3102R connector terminations are available
 Conduit Termination: The housing is threaded with 1/4-14 NPSF
 (Dryseal) straight pipe threads to accept a conduit fitting.
 Wire Termination for Conduit Units: A compression-type
 wire termination strip which accommodates AWG 14 to
 AWG 22 wire is mounted inside the housing. Functions are
 identified by a marker strip.
 MS3102R Connector Termination (optional): Connector is
 mounted on end cap (see table 1 for pin designations).
 Side conduit hole will be plugged when this option is used.
 Not available on UL versions.

ELECTRICAL

The Model H40 can be provided with any IED encoder
 output

ENVIRONMENTAL

Temperature: operating, 0° to 70°C standard;
 optional extended temperatures (see note 8)
 storage, -25° to 90°C
 Shock: 200 G's at 11 msec
 Vibration: 5 to 2000 Hz at 20 G's
 Humidity: 100% RH

Design Standards: The enclosure has been designed
 to meet the following standards:
 Indoor Non-Hazardous Locations: NEMA Enclosure 13
 Outdoor Non-Hazardous Locations: NEMA Enclosure 6
 Hazardous Locations: The optional Underwriters
 Laboratories listed version is for use in hazardous
 locations: NEMA Enclosure 7.
 Class 1, Group D, Division 1, NEC Class 2 circuits only

NOTES

- Non-hazardous models and multiple model locations may
 be used as indicated.
- Complementary pin outs are available with any output IC's.
 Then voltage normally be called out in pin numbering and
 drivers (check pin outs on the next page to determine proper
 connector when using MS connector option and
 available on 8-B).
- Supply voltages greater than 5V can be accommodated by:
 A. Using an internal 75 to 24VDC voltage regulator
 B. Using CMO's at 12VDC or 15VDC (model output use note 8C)
 C. Output voltages greater than 5VDC can be supported by:
 A. Using external output - customer built up output to
 required voltage up to 10VDC max
 B. Output may be pulled up to input voltage on voltage
 regulator 12VDC input = 12VDC output
 C. CMO's model IC's may be used for 12VDC or 15VDC input

and output CAUTION: CMO's may have the trend
 have tendency to drift due to temperature variations on some jobs
 3. Output IC's
 standard
 TABLE 1: IC's used connector output voltage, 30 mA max
 external current, the "T" node pins an internal 4.0 ohm output
 resistor
 TABLE: Without the "T" this output can be used to switch
 customer output to be used returned to voltages as high as
 10VDC. At the max
 When the "T" is used need for voltage regulator state units with
 output higher than 5VDC use note 8B. The resistor is added
 to drop the 10 mA source current, i.e. At 12VDC the pulsed
 resistor value is 1.2 ohms
 CAUTION:
 BEWARE: IC's are driver usually dissipate for some units over
 10% of output frequency is greater than 10 kHz. Must be
 terminated properly to avoid heating

- Other less common used output connector options:
 * 40-pin low driver TTL: MS3102R CMO's use driver
 * 40-pin low driver CMO's: MS3102R CMO's use driver
 * 40-pin low driver CMO's: MS3102R CMO's use driver
- Location "S" at the end of a model number is used for
 communication with the factory to derive a variety of multi-
 hazardous features such as extended temperature range, a
 communication and the 10 mention add a pin
- Frequency response up to 250 kHz may be available upon
 consultation with the factory
- Extended temperature ranges are available to 40 and -40°C
 some may cost a little
 * 40°C requires the standard pin and
- Voltage Sensitive Pin Resistor in
 CMO's may be ordered with factory
 For "M18" use MS3102R 16-15
 For "M18" use MS3102R 16-15
 For "M20" use MS3102R 20-15

OUTPUT TERMINATIONS

Incremental				Interpolation				Direc. Sensing			Term. Strip		Absolute	
Pin	ABZ	ABC	ABZC	AB	ABZ	ABC	ABZC	PZ	PC	PZC	WCR	PULSES	MS Conn (EM201)	Term. IEC or SCI
A	A	A	A	A	A	A	A	CW	CW	CW	A	CW	A	G7 2 ¹ G0
B	B	B	B	B	B	B	B	CCW	CCW	CCW	B	CCW	B	G4 2 ¹ G1
C	C	C	C	C	C	C	C	Z	Z	Z	C	Z	C	G5 2 ¹ G2
D	-	-	-	-	-	-	-	-	-	-	D	-	D	G4 2 ¹ G3
E	-	-	-	-	-	-	-	-	-	-	E	-	E	G3 2 ¹ G4
F	-	-	-	-	-	-	-	-	-	-	F	-	F	G2 2 ¹ G5
G	-	-	-	-	-	-	-	-	-	-	G	-	G	G1 2 ¹ G6
H	-	-	-	-	-	-	-	-	-	-	H	-	H	G0 2 ¹ G7
I	-	-	-	-	-	-	-	-	-	-	J	-	J	Spore
J	-	-	-	-	-	-	-	-	-	-	K	-	K	Lamp Sup.
											L	-	L	Interrogate
											M	-	M	Spore
											N	-	N	Latch (Opt.)
											P	-	P	Lamp Ret.
											R	-	R	Ground
											S	-	S	+VDC
											T	-	T	Case Gnd.

Wire Color	ABZ	ABC	ABZC	AB	ABZ	ABC	ABZC	PZ	PC	PZC	WCR	PULSES
YEL	A	A	A	A	A	A	A	CW	CW	CW	A	CW
BLUE	B	B	B	B	B	B	B	CCW	CCW	CCW	B	CCW
ORH	Z	Z	Z	Z	Z	Z	Z	-	-	-	C	Z
W-Yel	-	-	-	-	-	-	-	-	-	-	D	-
W-Blu	-	-	-	-	-	-	-	-	-	-	E	-
W-Orh	-	-	-	-	-	-	-	-	-	-	F	-
RED	-	-	-	-	-	-	-	-	-	-	G	-
BLK	-	-	-	-	-	-	-	-	-	-	H	-
GRH	-	-	-	-	-	-	-	-	-	-	J	-
VIOLET	-	-	-	-	-	-	-	-	-	-	K	-
GRAY	-	-	-	-	-	-	-	-	-	-	L	-

DISC RESOLUTIONS AVAILABLE

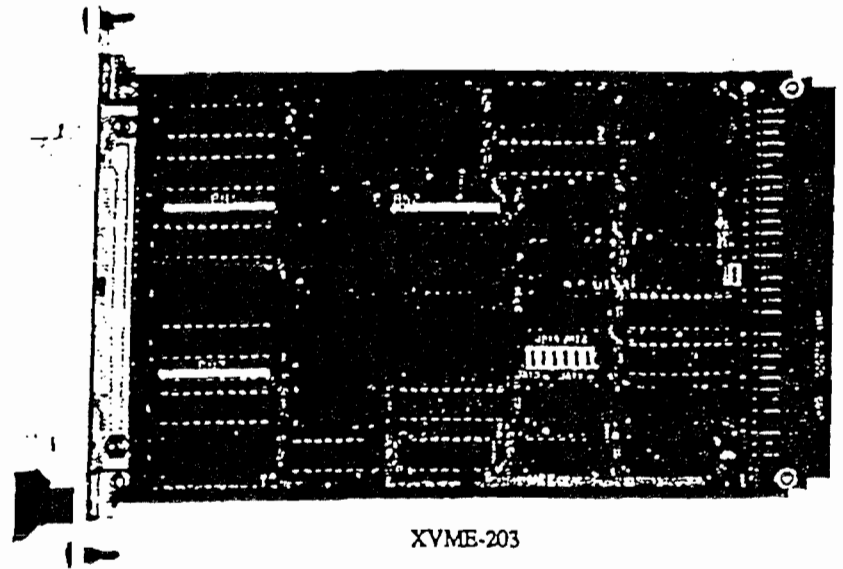
CYCLES PER TURN

1, 2, 3, 5, 8, 7, 8, 10, 13, 15, 16, 20, 24, 25, 30, 32, 33, 34, 36, 40, 45, 48, 50, 51, 56*, 60, 64, 72, 75, 80, 86, 100, 102, 120, 122, 125, 127, 128, 132, 144, 148, 150, 158, 160, 175, 180, 187, 192, 200, 204*, 217, 220, 240, 250, 254, 256, 256*, 274, 283, 288, 292, 300, 312, 320, 325, 360, 364, 372, 375, 377, 381, 384, 393, 400, 430, 432, 450, 462, 480, 490, 500
 502, 508, 510, 512, 522, 530, 567*, 576, 598, 600, 604, 625, 628, 636, 638, 640, 660, 672, 678, 680, 687, 690, 700, 720, 725, 735, 740, 744, 748, 750, 762, 768, 785, 800, 812, 825, 850, 864, 878, 888, 900, 912, 914, 938, 942, 955, 960, 1000, 1018, 1024, 1030, 1036, 1054, 1056, 1074, 1076, 1080, 1088, 1100, 1125, 1136, 1200, 1237, 1250, 1257, 1270, 1280, 1300
 1314, 1333, 1360, 1400, 1414, 1427, 1440, 1484, 1500, 1570, 1600, 1666, 1718, 1745, 1800, 1847*, 1855, 1875, 1894, 1920, 1968, 1979, 1995, 2000, 2048, 2094, 2100, 2160
 2199, 2200, 2250, 2356, 2400, 2485, 2500, 2514, 2519, 2540



XVME-203/293

Counter Module with Quadrature



XVME-203

Features

- High density digital I/O
- Single Eurocard size
- Bidirectional operation
- Buffered inputs and outputs
- Handshaking
- VMEbus interrupts
- Two 16-bit timers

Applications

- Translators for stepper motors
- Flow meters (turbine type)
- Voltage frequency converters
- Voltage-to-pulse width converters
- Duty cycle control of heaters
- Quadrature encoders

Overview

The XVME-203 is a single-high, VMEbus compatible board, using two AM9513A timer/counter devices to provide a total of ten 16-bit counting channels. The timer/counter devices are fully programmable and capable of counting at a rate of up to 5 MHz. In conjunction with the timer/counter devices, the XVME-203 employs an AM9519A Universal Interrupt Controller to provide a complete interrupt structure via eight interrupt channels. This interrupt structure allows for the selection of either a fixed interrupt vector for all eight channels, or a separate vector for each individual channel.

The XVME-203 can also be used for quadrature detection. Advanced encoding circuitry allows the module to be used with as many as four quadrature transducers simultaneously.

Eight output channels (ten on the XVME-293) are available to provide a variety of utility functions including: frequency output, one-shot output, frequency division, and pulse generation.

Hardware Specifications

Channels

Input	10 sources (SRC)
Output	8 on XVME-203 10 on XVME-293
Interrupt	8

Timer/counter device

(2) AM9513A

Interrupt control device

(2) AM9519A

Maximum counting rate

5 MHz

Input buffers

Low-level input current	-200 μ A
High-level input current	+20 μ A

Output buffers

Low-level output current	+24 mA
High-level output current	-15 mA

Power

+5 VDC, 1.85 A typ., 2.05 A max.

Environmental Specifications

Temperature

Operating	0° to 65° C (32° to 140° F)
Non-operating	-40° to 85° C (-40° to 158° F)

Humidity

5 to 95% RH, non-condensing

Altitude

Operating	Sea level to 10,000 ft.
Non-operating	Sea level to 50,000 ft.

Vibration

Operating	.015 in p-p displacement 2.5 g peak (max) acceleration
Non-operating	.030 in p-p displacement 5.0 g peak (max) acceleration

Shock

Operating	30 g peak acceleration, 11 msec duration
Non-operating	50 g peak acceleration, 11 msec duration

VMEbus Compliance

- Complies with VMEbus Specification IEEE 1014
- A16:D08(0) DTB Slave
- Interrupter - I(1)-I(7)(DYN), ROAK
- Interrupter Vector D08(O)(DYN)
- Form Factor:
 - XVME-203: SINGLE
165.1 mm x 100.01 mm (6.5 in x 3.94 in)
 - XVME-293: DOUBLE
233.35 mm x 160 mm (9.2 in x 6.3 in)

Warranty Information

The XVME-203 and 293 each carry a two-year warranty.

Ordering Information

XVME-203:	Single-High VMEbus Counter Module
XVME-293:	6U Version of the XVME-203 with the I/O Routed to the VMEbus P2 Connector
XVME-941:	6U Front Panel Kit for the XVME-203

XYCOM INC
750 North Maple Road, Saline, Michigan 48176
Phone: (313) 429-4971 TWX 810-223-8153
Call toll-free outside of Michigan: 1-800-AT-XYCOM

XYCOM ASIA
Rm. 903, Kowloon Center, 29-39 Ashley Rd.
T.S.T. Kowloon, Hong Kong
Phone: 852 3 311 7855 Telex: 780 33652 ALSITIX



XYCOM EUROPE LTD
6 Scirocco Close
Northampton NN3 1AP England
Phone: +44 604 790767 TWX: 9312102198 XIE G

XYCOM CANADA
461 N. Service Rd., W. Unit B5
Oakville, Ontario L6M 2V5 Canada
Phone: (416) 825-0281 FAX (416) 825-0282